Министерство образования и науки Российской Федерации

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра информационной безопасности электронно-вычислительных систем (КИБЭВС)

В.Н. Кирнос

Основы программирования на языке C++

Учебно-методическое пособие

Томск – 2012

В.Н. Кирнос

Основы программирования на языке С++: Учебно-методическое пособие. – Томск: ТУСУР, 2012, – 223 с.

В пособии кратко рассматривается порядок разработки программ на языке С++ (среда Microsoft Visual Studio 2008). При этом, как и положено, рассматривается алфавит языка, реализация основных типов алгоритмических структур (линейная, разветвленная, циклическая), работа с массивами и функциями на данном языке. Особое внимание уделено работе со структурами, классами, файлами.

В конце данного пособия, имеется глава с заданиями для выполнении лабораторных работ), а также в ней даны задания и указания для выполнения курсовой работы по данной дисциплине.

СОДЕРЖАНИЕ

ПРЕДИСЛОВИЕ

Дисциплина "Основы программирования" впрямую не входит в список дисциплин из списка ФГОС 3 поколения. Вместе с тем, она впрямую связана с дисциплинами "Информатика", "Языки программирования" и "Технологии и методы программирования". По этой причине здесь частично будут реализовываться компетенции указанных дисциплин:

Таким образом, согласно ФГОС процесс изучения дисциплины " Основы программирования " направлен на формирование следующих *компетенций*:

– способностью к логически-правильному мышлению, обобщению, анализу, критическому осмыслению информации, систематизации, прогнозированию, постановке исследовательских задач и выбору путей их решения на основании принципов научного познания (ОК-9);

– способностью самостоятельно применять методы и средства познания, обучения и самоконтроля для приобретения новых знаний и умений, в том числе в новых областях, непосредственно не связанных со сферой деятельности, развития социальных и профессиональных компетенций, изменения вида своей профессиональной деятельности (ОК-10);

– способностью выявлять естественнонаучную сущность проблем, возникающих в ходе профессиональной деятельности, и применять соответствующий физико-математический аппарат для их формализации, анализа и выработки решения (ПК-1);

– способностью применять математический аппарат, в том числе с использованием вычислительной техники, для решения профессиональных задач (ПК-2);

– способностью использовать языки, системы и инструментальные средства программирования в профессиональной деятельности (ПК-3);

– способностью применять методологию научных исследований в профессиональной деятельности, в том числе в работе над междисциплинарными и инновационными проектами (ПК-5);

– способностью применять современные методы исследования с использованием компьютерных технологий (ПК-10);

В результате изучения дисциплины студент должен:

Знать:

– основные понятия информатики;

- формы и способы представления данных в персональном компьютере;
- состав, назначение функциональных компонентов и программного обеспечения персонального компьютера;
- классификацию современных компьютерных систем;

• типовые структуры и принципы организации компьютерных сетей.

Уметь:

– применять типовые программные средства сервисного назначения (средства восстановления системы после сбоев, очистки и дефрагментации диска и т.п.);

– оценивать сложность алгоритмов и вычислений;

– вычислять теоретико-информационные характеристики источников сообщений и каналов связи;

– решать типовые задачи кодирования и декодирования;

Владеть:

– навыками работы с офисными приложениями (текстовыми процессорами, электронными таблицами, средствами подготовки презентационных материалов и т.п.);

– навыками обеспечения безопасности информации с помощью типовых программных средств (антивирусов, архиваторов, стандартных сетевых средств обмена информацией и т.п.)

– способами оценки сложности работы алгоритмов;

– основами построения математических моделей систем передачи информации;

– навыками применения математического аппарата для решения прикладных теоретико-информационных задач;

– навыками пользования библиотеками прикладных программ для решения прикладных математических задач;

Указанные задачи (хотя и не в полной мере) могут быть решены с помощью данного учебного пособия.

ГЛАВА 1. ОСНОВЫ ПРОГРАММИРОВАНИЯ В СРЕДЕ VISUAL C++ 2008

ВВЕДЕНИЕ

Современные системы программирования на C++ состоят из нескольких составных частей. Это такие части, как сама среда программирования, язык, стандартная библиотека C-функций и различные библиотеки C-классов.

Сразу заметим, что C++ является *объектно-ориентированным* языком. Основное отличие его от прежних, *структурных*, языков (примером таких является Турбо-Паскаль или C) является то, что он (C++) способен оперировать не только с переменными и структурами (функциями и процедурами), но и с целыми объектами. *Объекты* есть комплексы переменных и процедур (функций) по их обработке.

Как правило, чтобы выполнить программу на C++, необходимо пройти через 6 этапов: редактирование, препроцессорную (то есть предварительную) обработку, компиляцию, компоновку, загрузку и выполнение. В данном пособии мы будем рассматривать программирование в среде *Visual C++* 2008, входящей в состав Visual Studio 2008[1].

Первый этап представляет создание и редактирование файла с исходным текстом программы. Он может выполняться с помощью простейшего редактора текстов программ. Программист набирает в этом редакторе свою C++ программу. При необходимости он снова обращается к ней и вносит с помощью этого редактора изменения в исходный текст программы. Далее программа запоминается на диске. Имена файлов C/C++ программ оканчиваются на «c» или «cpp». Однако, пакет программ Visual C++ 2008 имеет встроенный редактор, которым также можно пользоваться.

На втором этапе компилятор начинает препроцессорную обработку текста программы, прежде чем ее компилировать. (Что же делает компилятор? Он переводит программу в машинный код. То есть в результате получаем объектный код программы, но это третий этап.). Следует знать, что в системе C++ программирования перед началом этапа самой трансляции всегда выполняется программа предварительной обработки. Что она делает? Она отыскивает так называемые «директивы трансляции» или «дирек-

---

[1] В настоящее время имеется и версия 2010 данной среды, но мы рекомендуем не увлекаться новейшими версиями. Поскольку приложения, созданные в более поздней версии, не могут быть открыты в более ранней версии, что приводит к проблемам с переносом созданного приложения на другие компьютеры. Заметим, что приложения, созданные в более ранней версии, — в более поздней открываются!

тивы препроцессора», которые указывают, какие нужно выполнить преобразования перед трансляцией исходного текста программы. Обычно это включение других текстовых файлов в файл, который подлежит компиляции. Препроцессорная обработка инициируется компилятором перед тем, как программа будет преобразована в машинный код. Это позволяет забирать нужные программы-функции в текст компилируемой программы до начала процесса компоновки.

Третий этап – это **компиляция**. Как правило, программы на языке C++ содержат ссылки на различные функции, которые определены вне самой программы. Например, в стандартных библиотеках или в личных библиотеках программистов. Объектный код, созданный компилятором, содержит «дыры» на месте этих отсутствующих частей.

Четвертый этап – компоновка. Компоновщик связывает объектный код с кодами отсутствующих функций и создает, таким образом, исполняемый загрузочный модуль (без пропущенных «дыр»).

Пятый этап – загрузка. Перед выполнением программа должна быть размещена в памяти. Это делается с помощью загрузчика, который забирает загрузочный модуль программы с диска и перемещает его в память.

Наконец шестой этап – это **выполнение**. Программа редко заработает с первой попытки. Каждый из названных этапов может заканчиваться ошибкой или неудачей из-за ошибки.

Тогда программист должен вернуться к редактированию исходного текста программы. Он должен внести необходимые изменения в текст программы, предварительно его хорошо проанализировав. Затем снова пройти через все этапы работы с исходным текстом программы до получения работающего без ошибок загрузочного модуля.

---

1. РАЗРАБОТКА ПРОГРАММЫ

Запуск среды Visual Studio 2005 выполняется командой ***Пуск*/*Программы*/*Microsoft Visual Studio* 2008/*Microsoft Visual Studio* 2008**[2]. Далее, если это первый запуск данной интегрированной среды, следует выбрать Visual C++ (напомним, что Visual Studio представляет собой единую среду, содержащую комплекс из Visual C++, Visual Basic и др. программных сред). При этом появится окно вида (рис. 2.1).

---

[2] Для целей обучения проще использовать бесплатно распространяемую версию *Microsoft Visual Studio 2008 Express Edition*. При ее использовании далее описываемые окна будут иметь несколько более простой вид: не будет некоторых дополнительных (не нужных нам здесь) возможностей.

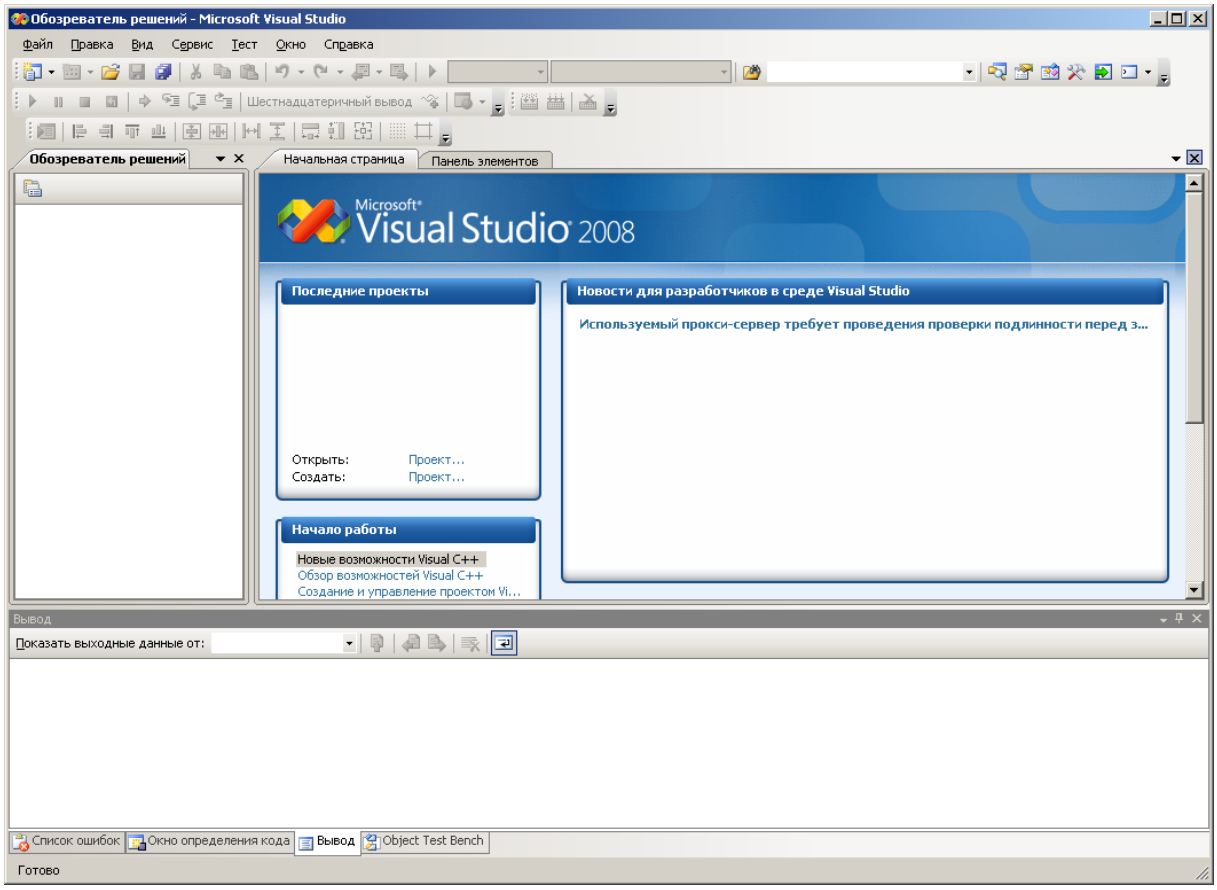


Рис. 2.1. Начальное окно среды Microsoft Visual Studio 2008

Часть окна слева на рис. 2.1 называется **окном обозревателя решений** (в английской версии – Solution Explorer), правое верхнее окно, содержащее начальную страницу (в английской версии – Start Page) – это **окно редактора** (Editor), а окно в нижней части называется **окном вывода** (Output). Окно обозревателя решений позволяет осуществлять навигацию по программным файлам, отображать их содержимое в окне редактора, а также добавлять новые файлы к вашей программе.

Окно редактора – это место, где вы вводите и модифицируете исходный код программы. Окно вывода отображает сообщения, полученные при компиляции и компоновке вашей программы.

Первый шаг при написании программы в среде Visual C++ состоит в создании проекта. Проект – это контейнер для всех составляющих его программ: обычно один или более исходных файлов, содержащих ваш код и ряд вспомогательных файлов. Все файлы проекта обычно сохраняются в отдельной папке проекта.

Итак, для создания проекта в среде Visual C++ нужно дать команду ***Файл/Создать/Проект*** (или нажать комбинацию клавиш **Ctrl-Shift-N**). В левой части возникшего окна (рис. 2.2) отображаются типы проектов, которые можно создавать. *Мы с вами пока будем создавать только консоль-*

ные приложения. Поэтому в левой части выберем **Win32**, а на правой панели выберем **Консольное приложение Win32**.

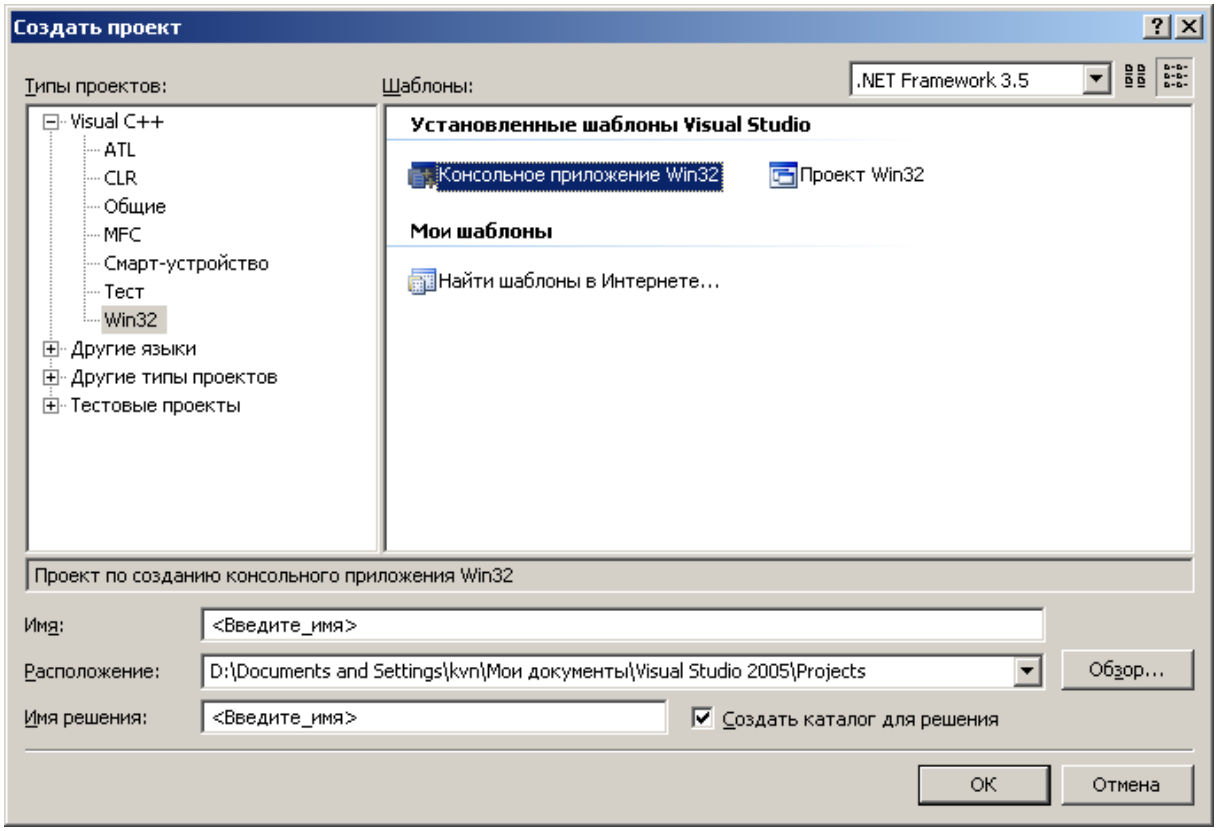


Рис 2.2. Окно создания проекта консольного приложения

Ниже (в строке Имя) ввести имя для проекта (зададим, например, **Pr1-1**) и нажать ОК. Обратите внимание, что при этом должна быть установлена «птичка» **Создать каталог для решения**. В следующем окне нажмем кнопку **Далее** и в очередном окне (см. рис. 2.3) выберем **Пустой проект** и нажмем **Готово**.

В результате будет окно (рис. 2.4), в котором слева (в **обозревателе решений**) появляется список подпапок данного проекта. Нас интересует теперь создание исходного файла. Для этого следует щелкнуть правой клавишей по папке **Файлы исходного кода** и выбрать ***Добавить/Создать элемент...*** (см. рис. 2.4) и в возникшем окне (рис. 2.5) выбрать **Код**, указать **Файл C++ (.cpp)**, ввести имя для исходного файла (в строке **Имя** – мы здесь ввели **pr1-1**) и нажать **Добавить**. В результате появится окно редактора, где и вводим исходный текст кода программы. Затем сохраняем его командой **Файл/Сохранить все.**

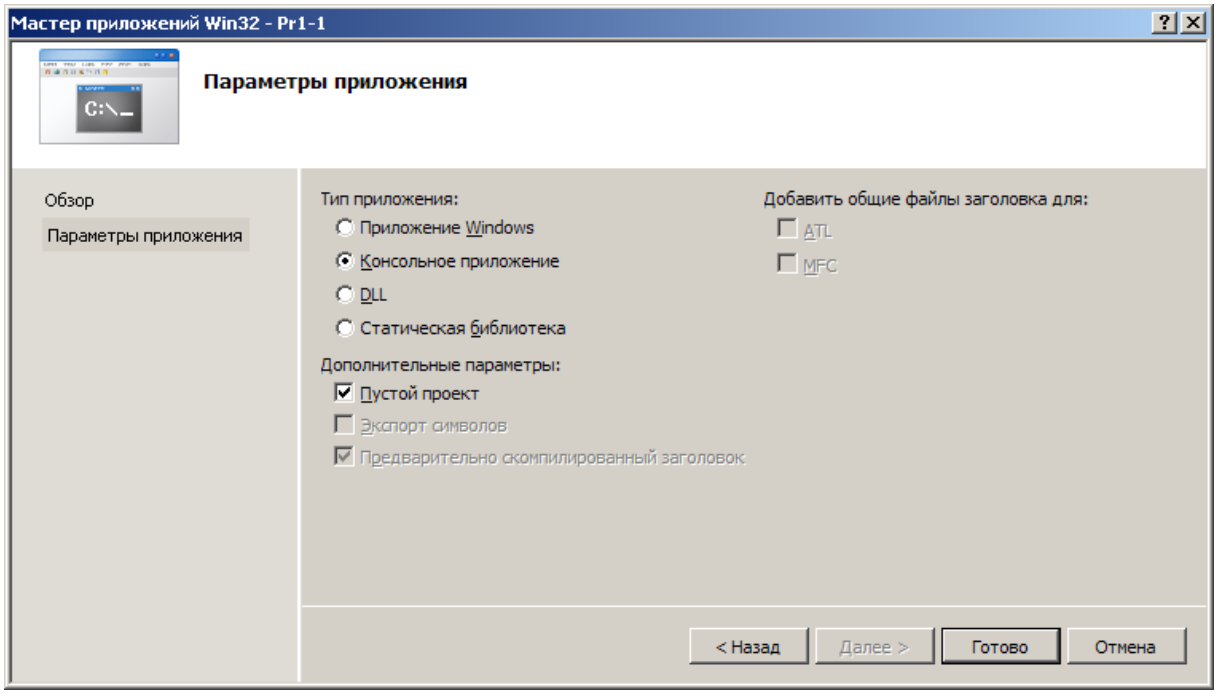


Рис 2.3. Окно свойств приложения с выбором типа проекта

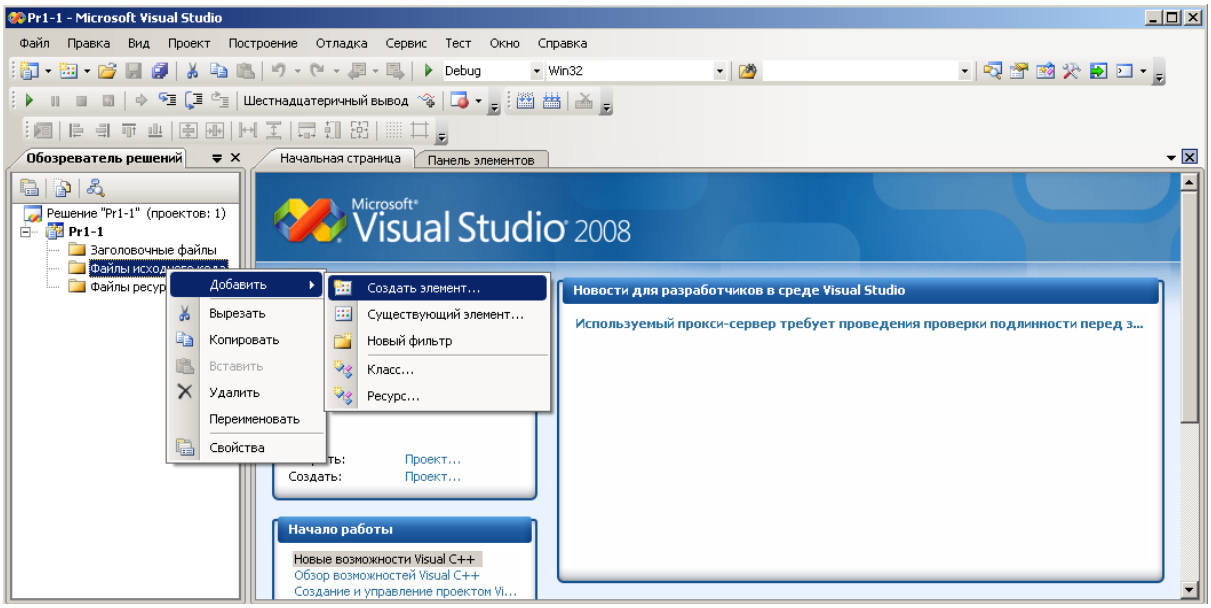


Рис. 2.4. Выбор вида исходного файла для проекта

Замечание. Если у вас еще не отображаются номера строк, дайте команду **Сервис/Параметры**, затем разверните пункт **Текстовый редактор** и подпункт **C/C++**. Выберите у последнего **Общие**, а справа установите «птичку» у **Номера строк** (см. рис. 2.6), нажмите ОК. Нумерация строк удобна потому, например, что транслятор нам будет сообщать, в какой строке (по номеру) ошибка.

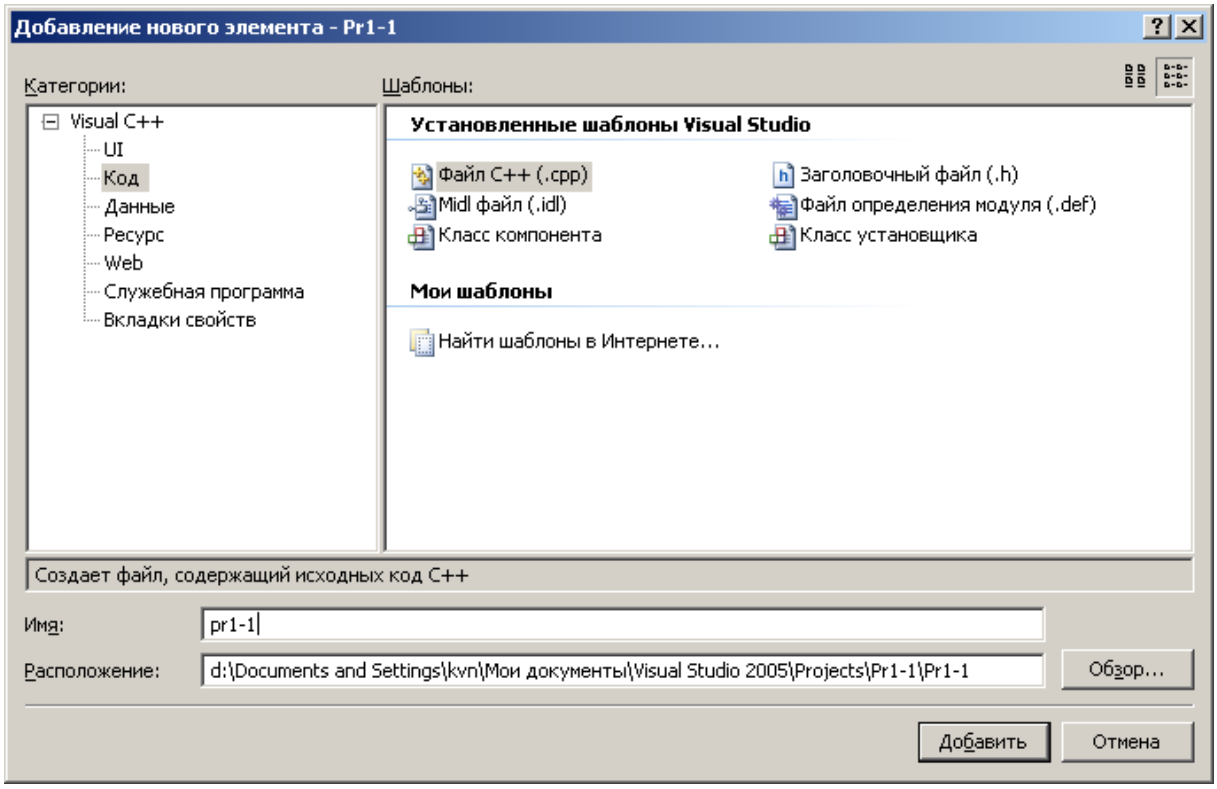


Рис. 2.5. Добавление нового исходного файла к проекту

Попробуем ввести текст простой программы (о содержимом каждой строки поговорим ниже). В результате окно редактора с введенным текстом программы приобретет вид (см. рис. 2.7).

Далее нужно сначала провести компиляцию и компоновку – дать команду ***Построение/Построить решение*** (или клавиша **F7**). Если все в порядке – в окне **Вывод** в нижней его части должно быть «успешно: 1, с ошибками: 0» (см. рис. 2.7). Если же будут ошибки при компиляции, то искать ошибки в соответствии с надписями, которые появятся в нижней части окна – в окошке **Output**. Там, как правило, указывается в какой строке, и какая ошибка обнаружена. После исправления ошибок, снова надо провести компиляцию и компоновку. Если же все пройдет без ошибок, то собственно запускаем на исполнение клавишами **Ctrl-F5** (**Отладка/Запуск без отладки**).

В результате откроется «черное окно», где вводим исходные данные и получаем результат – см. рис. 2.8.

Заметим, что результаты работы программы выводятся на черном фоне, но в данном пособии мы приводим эти изображения в инвертированном цвете, чтобы не перегружать «чернотой».

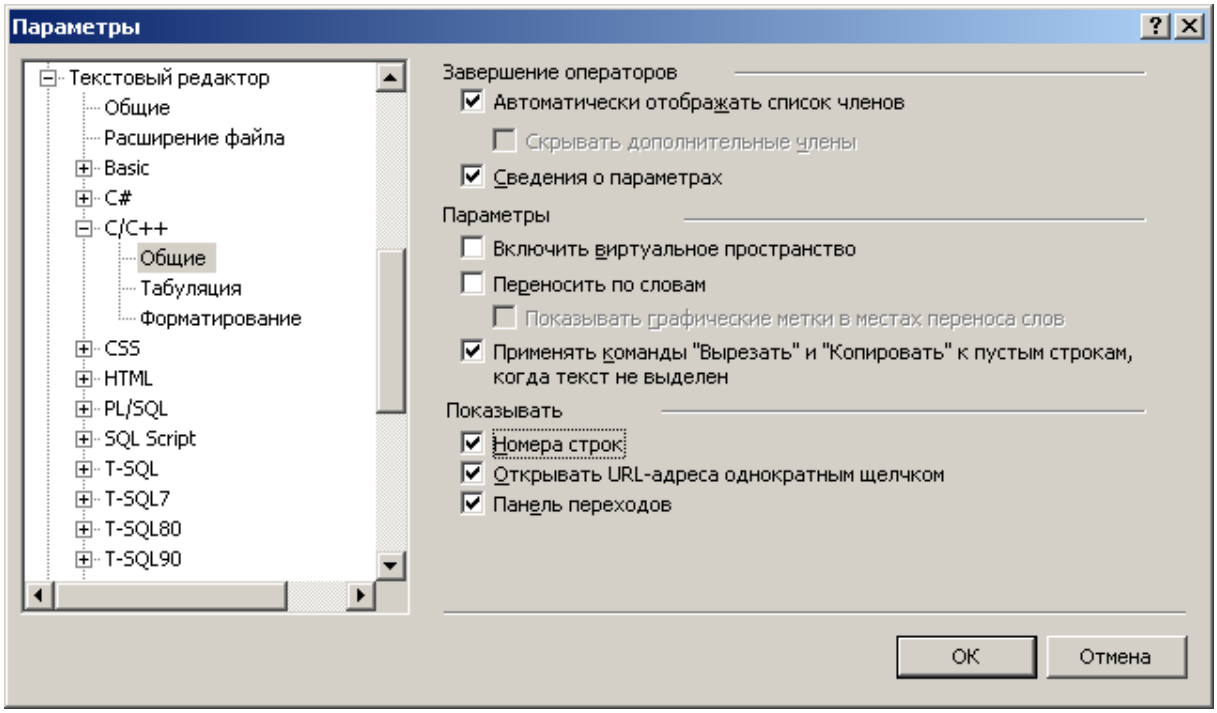


Рис. 2.6. Настройка отображения нумерации строк

Итак, **порядок разработки программы на Visual C++ 2008 (консольное приложение)**:

1) Создать проект: дать команду ***Файл/Создать/Проект*** (или нажать комбинацию клавиш **Ctrl-Shift-N**).

2) В возникшем окне на левой панели выбрать **Win32** и на правой выберем **Консольное приложение Win32 (**для создания консольного приложения), в строке **Имя** введем имя проекта (при этом должна быть установлена «птичка» **Создать каталог для решения** и нажать ОК.

3) В очередном окне появляется список подпапок данного проекта. Для создания исходного файла следует щелкнуть правой клавишей по папке **Файлы исходного кода** и выбрать ***Добавить/Создать элемент...*** (см. рис. 2.4).

4) В возникшем окне (рис. 2.5) выбрать **Код**, указать **Файл C++ (.cpp)**, ввести имя для исходного файла (в строке **Name**) и нажать **Add**.

5) В окне редактора вводим текст программы и сохраняем его командой ***Файл/Сохранить все*** (или клавишами **Ctrl-Shift-S**).

6) Для запуска программы на выполнение, нужно сначала провести компиляцию и компоновку – дать команду ***Построение/Построить решение*** (или клавиша **F7**).

7) Если все в порядке – в окне **Вывод** должно быть «успешно: 1» (см. рис. 2.7), то собственно запускаем на исполнение – клавишами **Ctrl-F5** (***Отладка/Запуск без отладки***).

8) В результате откроется «черное окно», где вводим исходные данные и получаем результат – см**.** рис. 2.8.

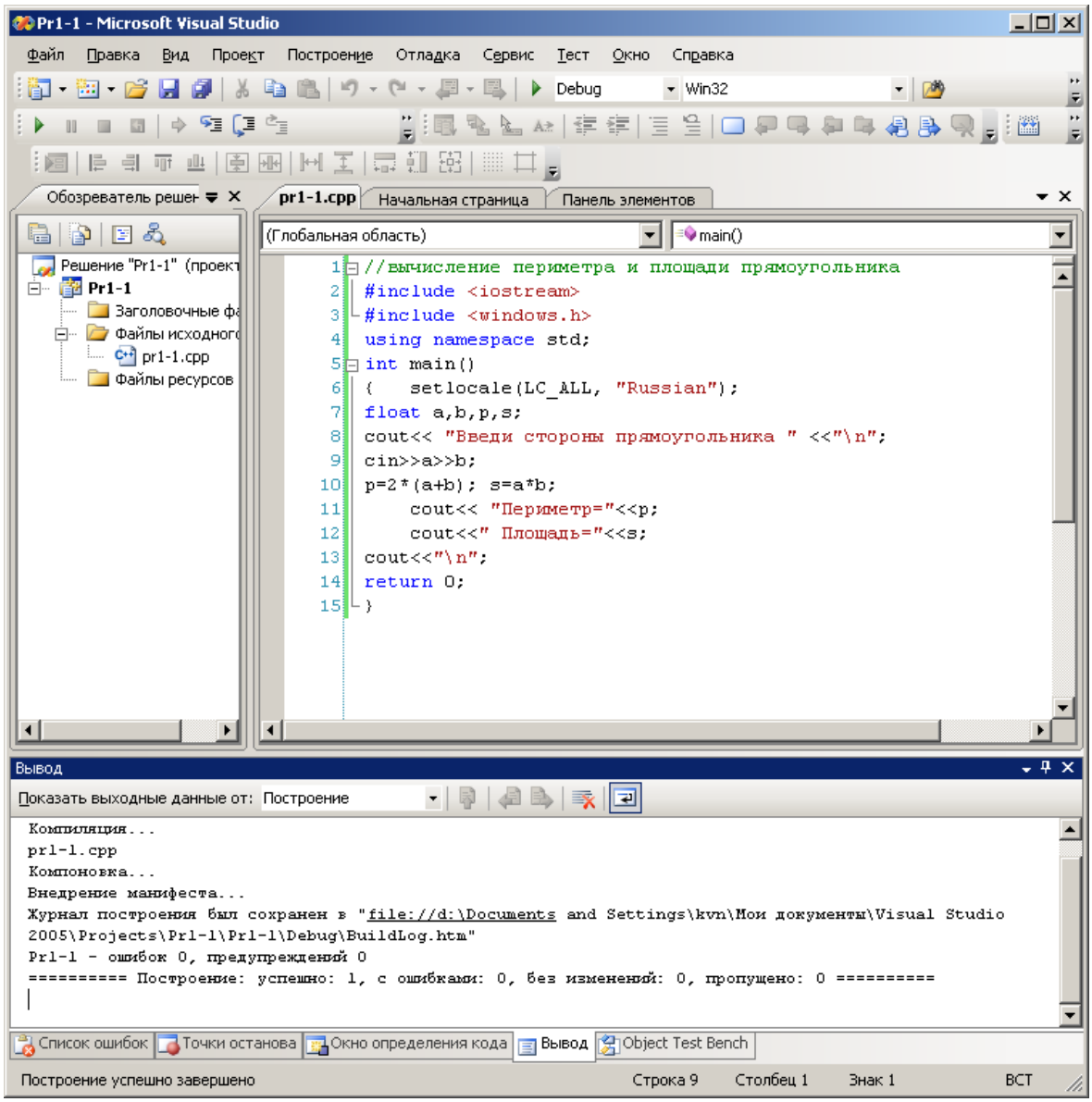


Рис. 2.7. Окно с результатами компиляции и компоновки

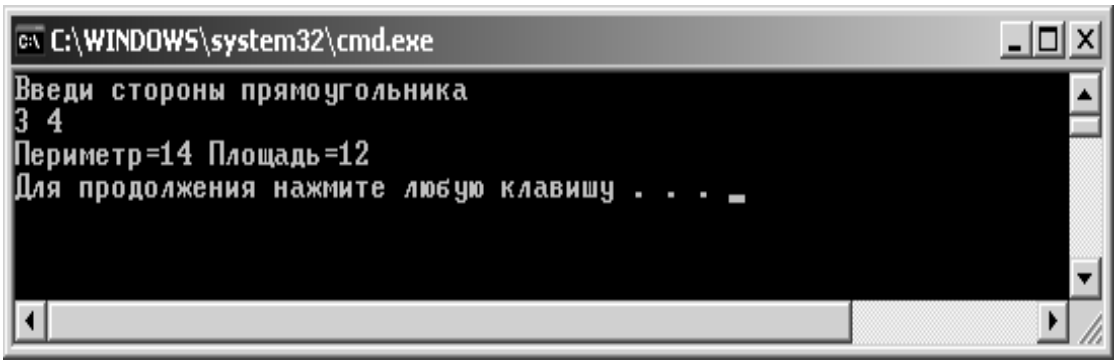


Рис. 2.8. Окно с результатами исполнения программы

Контрольные вопросы

1. Назовите основные элементы окна среды Microsoft Visual Studio.
2. Опишите порядок разработки проекта в Visual C++ с консольным приложением.

2. ПЕРЕМЕННЫЕ И ФУНКЦИИ

2.1. Переменные в C++

Ясно, что всякая программа, так или иначе, обрабатывает данные. Эти данные, как правило, размещаются в переменных. Каждая переменная имеет **имя, тип, размер и значение**. Основные типы с указанием размеров занимаемой памяти и областью значений приведены в табл. 1.

Таблица 1

Основные типы переменных в C++

Тип	Тип (по-русски)	Размер памяти, байт	Интервал допустимых значений
bool	логический	1	**true** или **false**
char	символьный	1	От –128 до 127 (код символа)
int	целый	4	От –2 147 483 648 до 2 147 483 647
long	целый	4	От –2 147 483 648 до 2 147 483 647
short	Короткий целый	2	От –32 768 до 32 767
unsigned char	Беззнаковый символьный	1	От 0 до 255
unsigned int	Беззнаковый целый	4	От 0 до 4 294 967 295
unsigned short	Беззнаковый короткий целый	2	От 0 до 65535
float	Вещественный плавающий	4	От ≈ 3.4e-38 до ≈ 3.4e+38 (по модулю)
double	Вещественный двойной точности	8	От ≈ 1.7e-308 до ≈ 1.7e+308 (по модую)

Объявление типов переменных делается соответствующим служебным словом с последующим перечислением имен переменных:
int i,j,k,l,m – перечисленные переменные будут целого типа и т.п. Причем, объявление типов можно совместить с присваиванием.

Присваивание значения переменным делается с помощью *команды присваивания* (в C++ используется *знак равенства*). Значения для символьных переменных заключаются в одинарные кавычки:

```
int i=33; float a=.156;
char f  =’5’, f2=’$’, ff=’1’
```

Основные арифметические и логические операции, используемые в C++, приведены в табл. 2.

Таблица 2

Основные арифметические и логические операции в С++

Арифметические операции	Символы	Логические операции	Символы
Сложение	+	Равно	= =
Вычитание	–	Не равно	!=
Умножение	*	Больше	>
Деление	/	Меньше	<
Вычисление остатка	%	Больше или равно	>=
Присваивание	=	Меньше или равно	<=
Отрицание	!	Логическое умножение	&&
		Логическое сложение	||

Присваивание значений переменных делается командой присваивания: *j*=5; *i*=*j*. Кстати, в С++ можно такие операции совмещать: *i*=*j*=5;

В С++ имеются помимо привычных и сокращенные формы записи арифметических операций. Примеры:

Операция присваивания:

a+=3 эквивалентно a=a+3;
a– =3 эквивалентно a=a–3;
a*=3 эквивалентно a=a*3;
a/=3 эквивалентно a=a/3;
a%=3 эквивалентно a=a%3.

Операции инкремента:

i++ постфиксная форма i=i+1;
++i префиксная форма i=i+1.

Операции декремента:

i– – постфиксная форма i=i–1;
– – i префиксная форма i=i–1.

Операции инкремента и декремента применяются к целым числам. Различие между постфиксной и префиксной формами иллюстрируется приведенным ниже примером.

Например, пусть имеем переменные целого типа *t* и *c*. Если *c*=5, тогда при записи *t*=++*c* + 6 получим, что *t* равно 12.

Если же применить постфиксную форму (при том же *c*=5): *t*=*c*++ + 6, то получим, что *t*=11, потому что начальное значение *c* используется для вычисления выражения до того, как *c* увеличится на единицу операцией инкремента. Этот оператор эквивалентен следующим двум: *t*=*c*+6; ++*c*.

Эти же правила применимы и к операции декремента – –. Например, если *c*=5, то *t* = – –*c* +6 даст значение 10 для *t*, в то время как *t* = 6 + *c*– – даст значение 11 для переменной *t*.

2.2. Математические функции в С++

Поскольку язык С++ разрабатывался, прежде всего, для решения научно-технических задач, то здесь имеется целый набор математических функций. Основные функции приводятся в табл. 3, более широкий набор дан в прил. 1.

Таблица 3

Основные математические функции в С++

Математическая запись	Запись на С++	Тип функции	Тип аргумента	Примечание
$\sin x$	sin(x)	double	double	Аргумент в радианах
$\cos x$	cos(x)	double	double	Аргумент в радианах
$\mathrm{tg}\,x$	tan(x)	double	double	Аргумент в радианах
$\mathrm{arctg}\,x$	atan(x)	double	double	Угол, тангенс которого есть X
$\ln x$	log(x)	double	double	Натуральный логарифм числа X
e^x	exp(x)	double	double	Функция, обратная натуральному логарифму числа X
$\sqrt{x}$	sqrt(x)	double	double	Квадратный корень числа X>=0
$\lvert x\rvert$	fabs(x)	double	double	Модуль (абс. величина) числа X
x^y	pow(x,y)	double	double	Число X в степени Y

Для примера запишем несколько выражений с использованием математических функций по правилам языка С++.

$\frac{\alpha^2+\beta}{ax^2+bx+c}$ – запишется так: **(**alfa*alfa + beta) / (a*x*x + b*x + c);

$\frac{-b\pm\sqrt{b^2-4ac}}{2a}$ – запишется так**:** (–b + sqrt(b*b – 4*a*c))/(2*a);

Запишем еще следующие выражения:

а) $\sqrt[8]{x^8+8x}$; б) $\frac{xyz-3{,}3|x+\sqrt[4]{y}|}{10^7+\sqrt{tg4!}}$; в) $\frac{\beta+sin^2\pi^4}{cos^2\gamma+|ctg\gamma|}$

а) sqrt(sqrt(sqrt (pow(x,8)+8*x)));
б) (x*y*z – 3.3*fabs(x+pow(y, 1.0/4)))/(1e7+sqrt(tan(2*3*4)));
в)(b+sin(pow(pi,4))*sin(pow(pi,4)))/(cos(g)*cos(g)+fabs(1/tan(g))).

3. ЛИНЕЙНАЯ ПРОГРАММА

Программа на С++ состоит из одной или более функций. Причем, начинается выполнение программы с функции **main()**, которая, по сути, есть главный элемент программы. Каждая функция, вообще говоря, должна возвращать какой-либо результат. Поэтому и функция **main()** должна заканчиваться командой возврата **return 0;,** показывающей, что программа завершена. Причем у функции (как и у переменных) указывается тип. Функция **main()** имеет тип **int**. Собственно алгоритм (исполняемая часть программы) заключается в фигурные скобки {} после выражения **int main()**. Каждая фраза алгоритма заканчивается точкой с запятой **;**.

Обычно перед функцией **main** дают так называемые директивы препроцессору. Такие директивы начинаются со знака # (произносится «хаш»). Например, директива

 # include <iostream>

подключает встроенные в С++ функции, которые обеспечивают ввод/вывод.

Для *вывода на экран* служит команда

cout << (читается «си-аут»).

В ней для вывода текстовых сообщений помещают их в кавычки. Также там можно разместить управляющие последовательности для разделения строк вывода:

“**n**” – дает команду начать вывод с новой строки

‘ ‘ (пробел в одинарных кавычках) – разделяет пробелом выводимые знаки или строки. Иначе – весь вывод сливается в одну строку.

“**\t**” – табуляция;

Для *ввода в программу* (в процессе ее исполнения) служит команда:

cin >> (читается «си-ин»).

В ней указывается переменная, которой и будет присвоено значение, введенное с экрана.

В программе можно размещать комментарии, которым предшествует // (двойная дробная черта).

Важной в начале исходного текста программы является директива объявления пространства имен:

using namespace std;

Эффект от ее применения состоит в том, что вы можете свободно применять, в частности, вышеупомянутые команды ввода-вывода **cout** и **cin**. И процессор при этом будет четко понимать, что это команды, а не переменные.

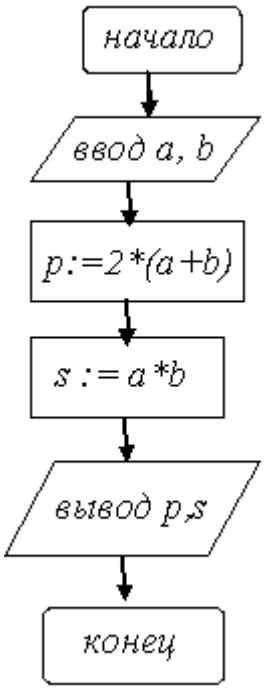
Изученных сведений достаточно для составления простой программы на С++. По аналогии со структурным языком назовем такую программу,

исполняющуюся прямолинейно от начала и до конца – *линейной программой*. Рассмотрим простой пример.

Пример 3-1. Вычислить периметр и площадь прямоугольника, длина которого равна *a*, а ширина равна *b*.

Пример: *a*=3 м, *b*=4 м. Ответ: периметр = 14 м, площадь = 12 м^2.

Приведем сначала блок-схему алгоритма для решения данной задачи.




```
//вычисление периметра и площади прямоугольника
#include <iostream>
using namespace std;
int main()
{ float a,b,p,s;
cout<<"Vvedi storony pryamoug"  <<"\n";
cin>>a>>b; p=2*(a+b); s=a*b;
cout<<"Perimetr="<<p<<" Ploshad'="<<s;
cout<<"\n";
return 0;  }
```

Замечание. Одна из проблем при программировании в Visual C++ 2008 (при создании консольного приложения) в том, что в консольном «черном» окне не отображаются сообщения, написанные по-русски. Это связано с тем, что в Windows (где мы пишем текст программы) и в MS DOS (где отображается ее результат) используется разная кодировка русских букв. Чтобы справится с этой проблемой, достаточно в начале программы, т.е. сразу после **int main()** вставить строку

setlocale(LC_ALL,"Russian");

И далее в программе смело писать русские сообщения.

Как именно это делается, проиллюстрируем на том же примере 3-1. (нужное дополнение выделено **жирным** шрифтом)

```
//вычисление периметра и площади прямоугольника
#include <iostream>
using namespace std;
int main()
{    setlocale(LC_ALL, "Russian");
float a,b,p,s;
cout<< "Введи стороны прямоугольника " <<"\n";
cin>>a>>b; p=2*(a+b); s=a*b;
    cout<< "Периметр="<<p; cout<<" Площадь="<<s;
cout<<"\n";
```

```
return 0;
}
```

Проведем «построение» решения (напомним, что для этого достаточно нажать клавишу **F7**) и, если нет ошибок[3], запускаем на выполнение (**Ctrl-F5**).

После запуска на исполнение получим результат как на рис. 2.9.

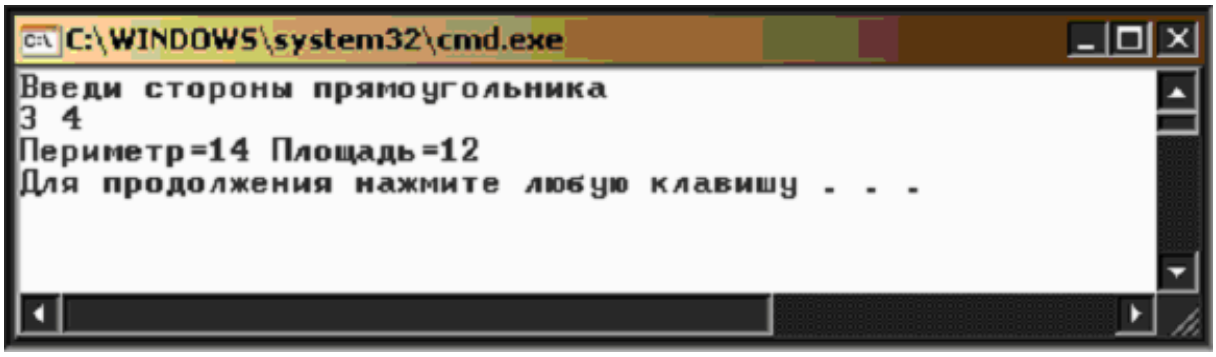


Рис. 2.9. Результат работы примера 3.1 (с выводом сообщений по-русски)

Контрольные вопросы

1. Для чего служат директивы препроцессору в Visual C++?
2. Какую структуру имеет главная функция программы на Visual C++?
3. Как устроены команды ввода и вывода на Visual C++?
4. Как организуется вывод сообщений по-русски в среде Visual C++?

---

4. ПРОГРАММА С ВЕТВЛЕНИЕМ

Опять-таки по аналогии со структурными языками будем называть программу, использующую структуру выбора, *программой с ветвлением*.

Структура выбора на C++ имеет вид:

if (*условие*) *оператор* 1; else *оператор* 2

Обратим внимание, что в отличие, например, от Паскаля в таком операторе отсутствует служебное слово **then** и ***условие*** обязательно заключается в скобки. ***Оператор*1** исполняется в случае истинности ***условия***. ***Оператор*2** – в случае ложности ***условия***.

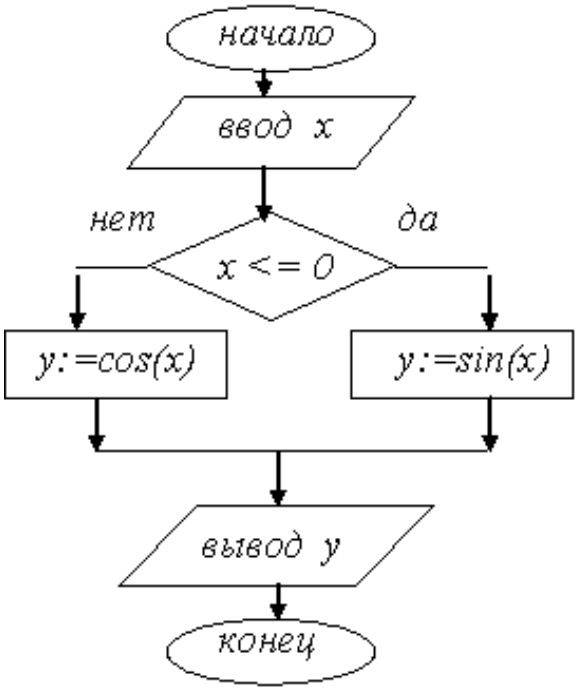
Напомним, что в C++ для использования математических функций необходимо подключать заголовочный файл `math.h`.

---

[3] А вот если есть ошибки, то их, конечно, нужно исправить. Следует в окне **Вывод** дважды щелкнуть курсором мыши по строке сообщения с ошибкой (там будут некоторые пояснения к ошибке) — курсор автоматически перепрыгнет в строку с ошибкой и эту ошибку нужно исправить. После чего повторно провести построение решения.

Пример 4.1. Составить программу для вычисления по заданному x значения функции: $y = \begin{cases} sinx, & x \le 0 \\ cosx, & x > 0 \end{cases}$

Приведем сначала блок-схему алгоритма для этой задачи.




```
#include <iostream>
#include <math.h>
using namespace std;
int main()
{    setlocale(LC_ALL, "Russian");
     double x,y;
     cout<<" Введи значение x "; cin>>x;
     if (x<=0) y=sin(x); else y=cos(x);
     cout<<"\n Функция y="<<y<<"\n";
return 0;}
```

Например, при x=0 получим 0 (sin0 = 0), а при x=3.14159265 (π радиан) будет получено –1 (т.к. cosπ = –1).

В тех случаях, когда требуется выполнить несколько действий (в случае истинности или в случае ложности условия), то применяют фигурные скобки:

if ***условие*** **{*оператор* 1_1; *оператор* 1_2;} else {*оператор* 2_1; *оператор* 2_2;}**

Пример 4.2. Составить программу, которая перераспределит заданные значения *x, y* так, что в *x* окажется большее значение, а в *y* меньшее.

```
//перестановка x,y; большее - вперед
#include <iostream>
#include <math.h>
using namespace std;
int main()
{    setlocale(LC_ALL, "Russian");
     double x,y,z;
     cout<<" Введи значение x y ";     cin>>x>>y;
     if (x<y) {z=x; x=y; y=z;}
     cout<<"\n Итог x="<<x;
     cout<<"\n Итог y="<<y<<"\n";
return 0;}
```

В результате, например, получим (рис. 2.12):

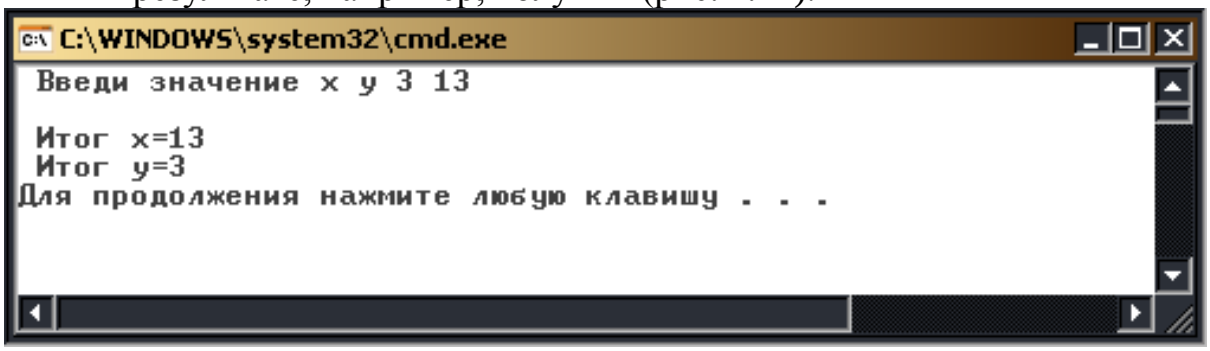


Рис. 2.10. Результат работы примера 4.2

Обратите внимание, что здесь использована сокращенная форма записи условного оператора: **if *условие оператор*.**

Условие может быть и весьма сложным. Здесь допускается *конъюнкция* условий (одновременное выполнение условий) – условия связываются при помощи &&, *дизъюнкция* условий (альтернативное выполнение условий) – обозначается || и *инверсия* условий (обозначается знаком !).

Пример 4.3. Выяснить, принадлежит ли точка с координатами *(x, y)* кольцу с внешним радиусом r_1 и внутренним r_2.

```
// принадлежит ли точка кольцу от r2 до r1
#include <iostream>
using namespace std;
int main()
{ setlocale(LC_ALL, "Russian");
     double x,y,r1,r2;
 cout<<"Введи x,y,r1,r2 "; cin>>x>>y>>r1>>r2;
 if ((x*x+y*y)<r1 && (x*x +y*y)>r2)
```

```
      cout<<"\n Точка принадлежит кольцу \n";
else cout<<"\n Точка не принадлежит кольцу \n";
return 0;}
```

Если зададим: x=1, y=1, $r1$=3, $r2$=1, получим «принадлежит», а если x=2, y=2, $r1$=2, $r2$=1, получим «не принадлежит».

Контрольные вопросы

1. Как организуется ветвление в программах на Visual C++?
2. Что делают в случае, когда для каждой ветви требуется выполнение нескольких действий?
3. Что такое конъюнкция, дизъюнкция и инверсия условий? Как они оформляются на Visual C++?

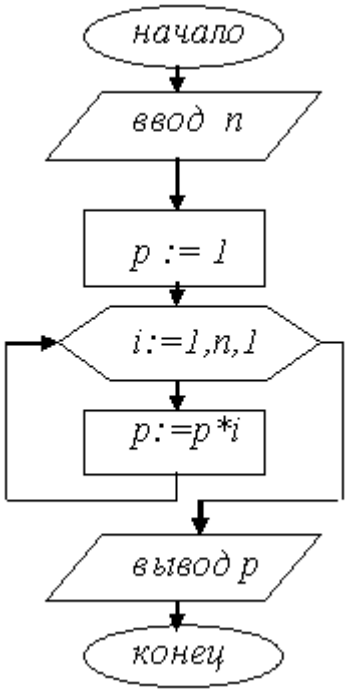
5. ЦИКЛ С ПАРАМЕТРОМ

На языке C++ циклы с параметром организуются с помощью структуры повторения:

for (*парам*=нач.знач; *парам*<кон.знач; *парам*++) *тело цикла*

Отметим, что параметр может изменяться и не на единицу, а на другое (целое) число. Тогда вместо ***парам*++** следует писать, например, ***парам*+=2** (если параметр приращается на 2).

Пример 5-1. Рассчитать для заданного N величину $N!$




```
#include <iostream>
using namespace std;
int main()
```

```
{  setlocale (LC_ALL, "Russian");
     unsigned long i,n, p;
     cout<<" Введи значение N\n";   cin>>n; p=1;
  for (i=1; i<n+1; i++)      p=p*i;
   cout<<"Факториал числа "<<n<<" равен "<<p<<"\n";
return 0;}
```

Заметим, что на этой задаче мы сталкиваемся с проблемой ограниченного представления целых чисел: мы сможем вычислить максимум 12!= 479 001 600. Следующий 13! = 6 227 020 800, т.е. это число уже не помещается в интервал представления для переменных типа `unsigned long int`. Поэтому, чтобы вычислять факториал для достаточно больших чисел, следует описание `p` сменить на `double`, правда, при этом мы будем получать округленное значение факториала (тип `double` дает не более 15 достоверных значащих цифр).

Заметим, что при программировании в среде CLR существует дополнительный целочисленный тип `long long` (см. табл. 4).

В этом случае (при использовании `long long`) удастся точно вычислить до 20!=2 432 902 008 176 640 000.

Таблица 4

Дополнительные целые типы в Visual C++

Тип	Тип (по-русски)	Размер памяти	Интервал допустимых значений
long long	Целый	8 байт	от –9 223 372 036 854 775 808 до 9 223 372 036 854 775 807
unsigned long long	Целый	8 байт	От 0 до 18 446 744 073 709 551 615

Разумеется, внутри цикла можно организовывать и вложенные циклы, причем число вложений, в принципе, не ограничено.

Пример 5-2. Получить таблицу умножения в виде таблицы Пифагора:

```
//таблица умножения
#include <iostream>
using namespace std;
int main()
{int i,j,k ;
for (i=1;i<=9;i++)
  {cout<<"\n";
  for (j=1; j<=9;j++)
     {k=i*j;
     if (k<10)cout<<"  " ; else cout<<" ";
     cout<<k; }; } cout<<"\n";
return 0;}
```

1	2	3	4	5	6	7	8	9
2	4	6	8	10	12	14	16	18
3	6	9	12	15	18	21	24	27
4	8	12	16	20	24	28	32	36
5	10	15	20	25	30	35	40	45
6	12	18	24	30	36	42	48	54
7	14	21	28	35	42	49	56	63
8	16	24	32	40	48	56	64	72
9	18	27	36	45	54	63	72	81

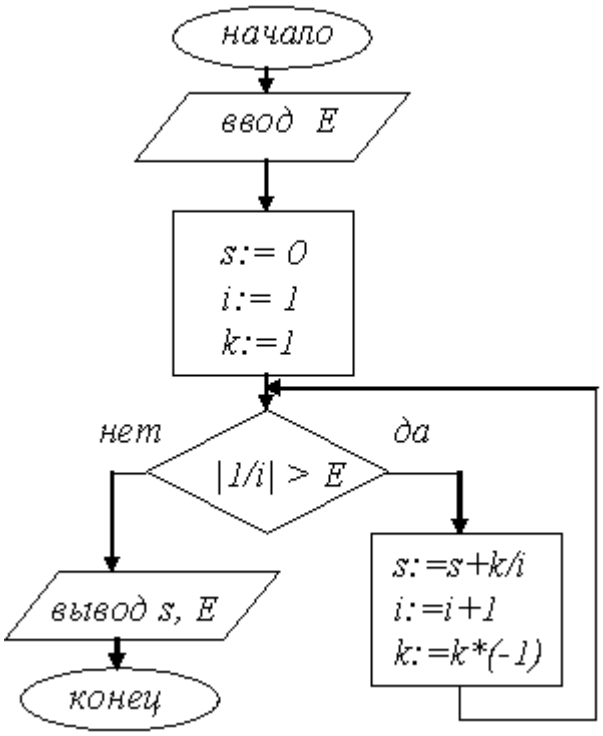
6. ЦИКЛ «ПОКА»

В С++ имеется также структура повторения **While**, которая служит для организации цикла «ПОКА»:

While (*условие*) *тело цикла*

Работает он так: ***тело цикла*** повторяется, пока заданное ***условие*** остается истинным.

Пример 6-1. Рассчитать сумму вида: $1-\frac{1}{2}+\frac{1}{3}-\frac{1}{4}+\cdots$ с заданной точностью *E*. Иначе говоря, проводить суммирование, пока очередное слагаемое является большим *E* (по абсолютной величине).




```
//сумма бесконечного ряда
#include <iostream>
#include <windows.h>
using namespace std;
int main()
{    setlocale(LC_ALL, "Russian");
     int i,k; double s,e;
cout<<"введи точность \n"; cin>>e;
s=0; i=1; k=1;
while(1.0/i>e){s=s+(double)k/i; i++; k=k*(-1);};
cout<<" сумма = "<<s<<" с точностью "<<e<<"\n";
return 0;}
```

В данной программе использован известный нам алгоритм суммирования. Суть его в том, что вводится переменная (здесь это переменная ***s***), задается ей нулевое значение, а затем в нее накапливается сумма. Очередное слагаемое – обратное к очередному числу натурального ряда. Однако четные (по номеру) слагаемые идут со знаком «минус», нечетные – со знаком «плюс». Для учета этого знака введена дополнительная переменная ***k***, которая последовательно принимает значение то +1, то –1. Переменная ***i*** последовательно дает очередное натуральное число (2, 3, 4 и т.д.). Для удобства она описана как целая (`int`).

Расчет по этой программе при E=0.1 дает значение 0.6456349, а точное значение для такой суммы (оказывается у этой бесконечной суммы результат – конечное число!) есть $ln(2) = 0.6931...$

Обратите внимание, что нам здесь потребовалось произвести в программе преобразование типов (**целый** к типу **double** при выполнении суммы), иначе в результате деления целого на целое получался нуль! И в условии, задающем цикл, мы сделали явное преобразование целого к вещественному – записав вместо целой единицы вещественную единицу.

Для изменения порядка работы в цикле в языке C++ применяются два оператора: **break** (для досрочного прерывания цикла) и **continue** – для пропуска следующих за ним операций с переходом в конец цикла, но из цикла выхода не производится.

Пример 6-2. Среди K первых членов последовательности вида: 1, 1+1/2, 1+1/2+1/3,... найти первый, больший заданного числа A. Если же его нет (среди первых К членов), то напечатать об этом сообщение.

```
//поиск члена последовательности, меньшего
//заданного числа a
#include <iostream>
using namespace std;
int main()
{ setlocale(LC_ALL, "Russian");
     int i,k; double a,s;
cout<<"Введи количество \n"; cin>>k;
cout<<"Введи число a \n"; cin>>a;
s=0; i=1;
while (i<=k) { s=s+1.0/i; if (s>a) break ; i++;};
if (i>k) cout<<" такого числа нет \n";
else cout<<" при i ="<<i<<" большее a число "<<s<<"\n";
return 0;}
```

Контрольные вопросы

1. Как организуется цикл с параметром на Visual C++?
2. Как организуется цикл-ПОКА на Visual C++?
3. Для чего служат команды break и continue?

7. ОДНОМЕРНЫЕ МАССИВЫ

7.1. Понятие об одномерном массиве

В программах на С++ можно использовать массивы.

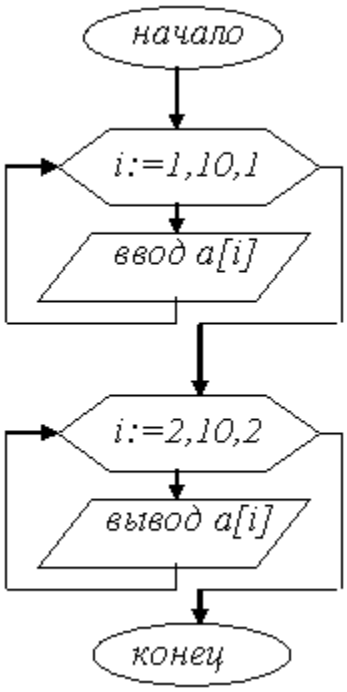
Напомним, что *массив – это упорядоченный набор величин, обозначаемых одним именем. Данные, являющиеся элементами массива, располагаются в памяти компьютера в определенном порядке, который задается индексами (порядковыми номерами элементов массива).*

В С++ массив, как и любая переменная, должен быть объявлен. Делается это с помощью служебного слова указывающего тип, затем указывается имя массива и в квадратных скобках его длина. Заметим, что индексы массива ведут счет с нуля, поэтому запись вида:

```
double b[14]
```

означает, что резервируется память для 14 чисел типа `double` с именем `b` и порядковыми номерами от 0 до 13. Отдельный элемент масссива записывается с указанием имени и индекса в квадратных скобках.

Пример 7-1. Ввести одномерный массив из 10 целых чисел. Вывести четные по порядковому номеру элементы этого массива.




```
//ввод и вывод одномерного массива
#include <iostream>
using namespace std;
int main()
{setlocale (LC_ALL,"Russian");
int i, a[11];
for (i=1; i<=10; i++)
{ cout<<"введи элемент номер "<<i<<" ";
cin>>a[i];}
for (i=2; i<=10; i+=2)
{cout<<"\n число a[ "<<i<<"]="<<a[i]; }
cout<<"\n";
return 0;}
```

При запуске программы получим такой результат (рис. 2.14).

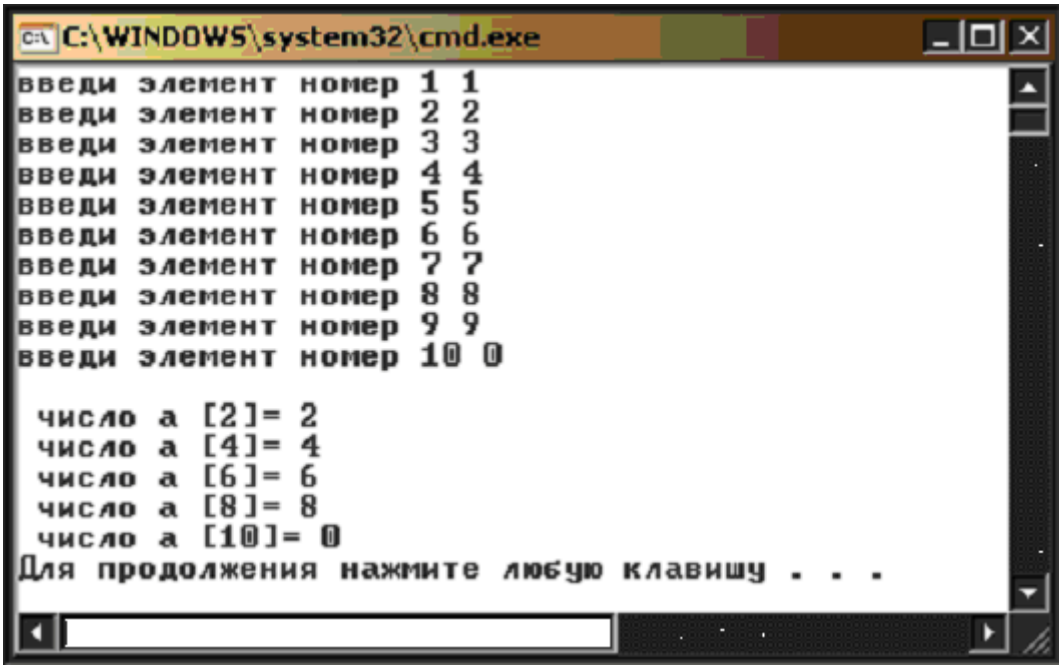


Рис. 2.14. Результат работы примера 7-1

Замечание. При работе с массивами необходимо внимательно следить за тем, чтобы не выходить за их объявленные границы. Компилятор C++ (в отличие, например, от Паскаля) не предупреждает об этой ошибке! Попытка ввести больше элементов, чем описано, приведет к неверным результатам, а попытка вывести – выведет случайный результат, находящийся в памяти.

Попробуйте в предыдущем примере описать массив **a[10]** (напомним, что при этом максимальный элемент **a[9]**, т.к. счет элементам идет с нуля), и вы не получите последнего, десятого, числа (точнее получите какое-то случайное число при выводе).

Еще раз обратим внимание, что работа с целыми переменными (и массивами) в C++ требует осторожности. Как отмечалось в примере 6-2, деление целого на целое дает в результате целое! Если требуется получать «правильный», вещественный, результат следует использовать преобразование – дописать перед нужной операцией (**float**) или (**double**). Если же требуется получать остаток от деления целого на целое, то применяют операцию **%.** Проиллюстрируем использование операции деления и вычисления остатка на простом примере.

Пример 7-2. Ввести четное и нечетное число и вычислить результаты деления их на 2 и остатки от такого деления.

```
#include<iostream>
using namespace std;
int main()
```

```
{int n,k;
setlocale(LC_ALL, "Russian");
cout <<"введи четное и нечетное"; cin>>n>>k;
   cout <<" остаток от деления четного 2="<< n%2;
   cout<<"\n результат деления четного на 2= "<<  n/2;
   cout<<"\n остаток от деления нечетного на 2="<< k%2;
cout <<"\n результат деления нечетного на 2="<<
(float)k/2<<"\n";
   return 0;}
```

В результате получим (рис. 2.15).

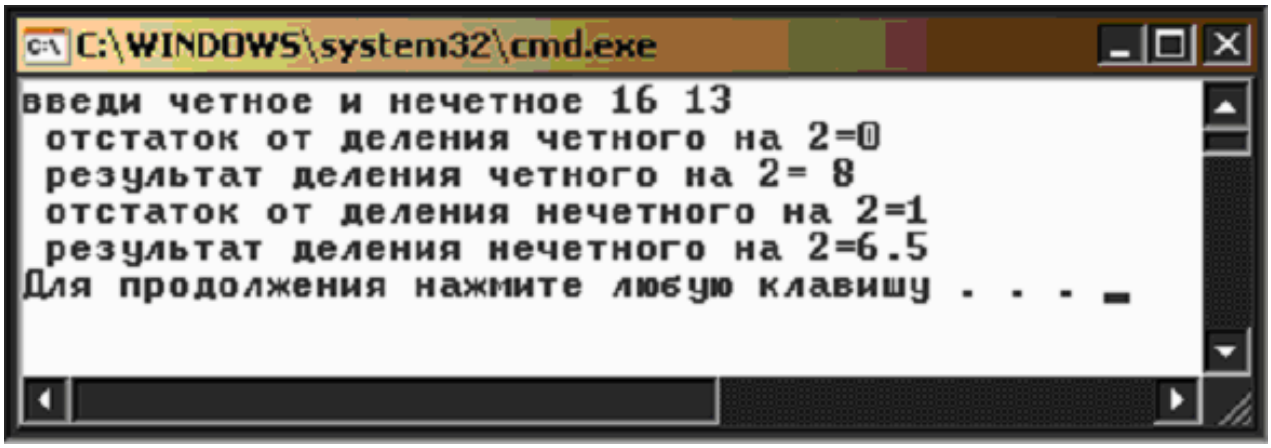


Рис. 2.15. Результат работы примера 7-2

7.2. Сортировка в одномерном массиве

Очевидно, что элементы массива, как правило, располагаются в произвольном порядке. Но во многих случаях может понадобиться расположить их, например, в порядке возрастания. Такая процедура упорядочения называется сортировкой.

Рассмотрим простейший способ сортировки – *метод пузырька*. Суть его в том, что последовательно сравниваются элементы массива и, если очередной элемент меньше предыдущего, то их меняют местами.

Пример 7-3. Ввести одномерный массив и упорядочить его методом пузырька.

Для ясности приведем блок-схему алгоритма такой сортировки (рис. 2.16).

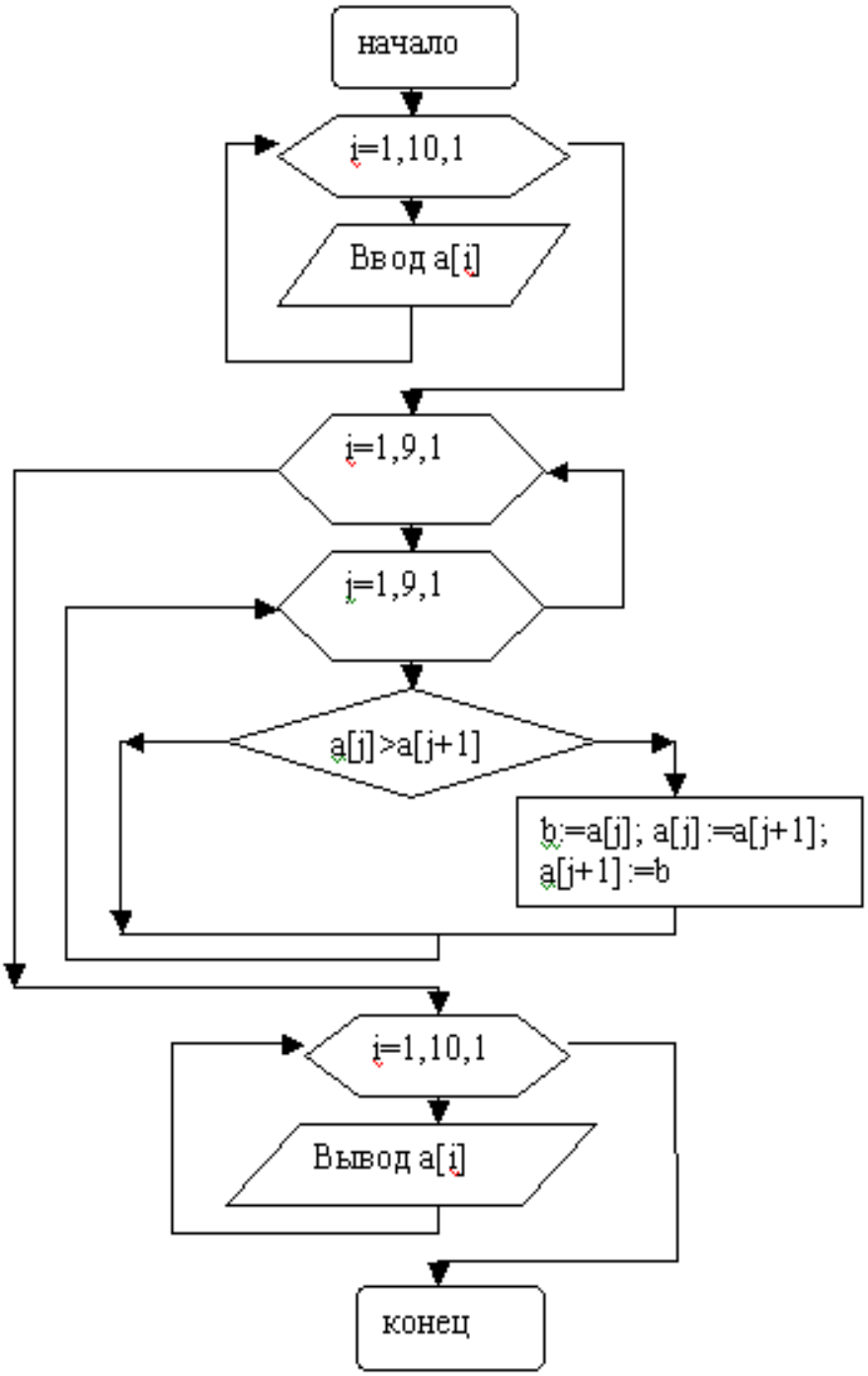


Рис. 2.16. Блок-схема алгоритма метода пузырька

Соответственно текст программы имеет вид:

```
#include <iostream>
using namespace std;
int main()
{  int i,j, a[11],b;
setlocale(LC_ALL, "Russian");
     for (i=1; i<=10; i++)
```

```
  {    cout<<"введи  a["<<i<<"] "; cin>>a[i];}
for (i=1; i<=9; i++)
for (j=1; j<=9; j++)
{ if (a[j]>a[j+1]){b=a[j];a[j]=a[j+1]; a[j+1]=b;}}
for (i=1; i<=10; i++)
{ cout<<"\n a["<<i<<"] "<<a[i];}   cout<<"\n";
return(0);}
```

Результат работы такой программы (см. рис. 2.17).

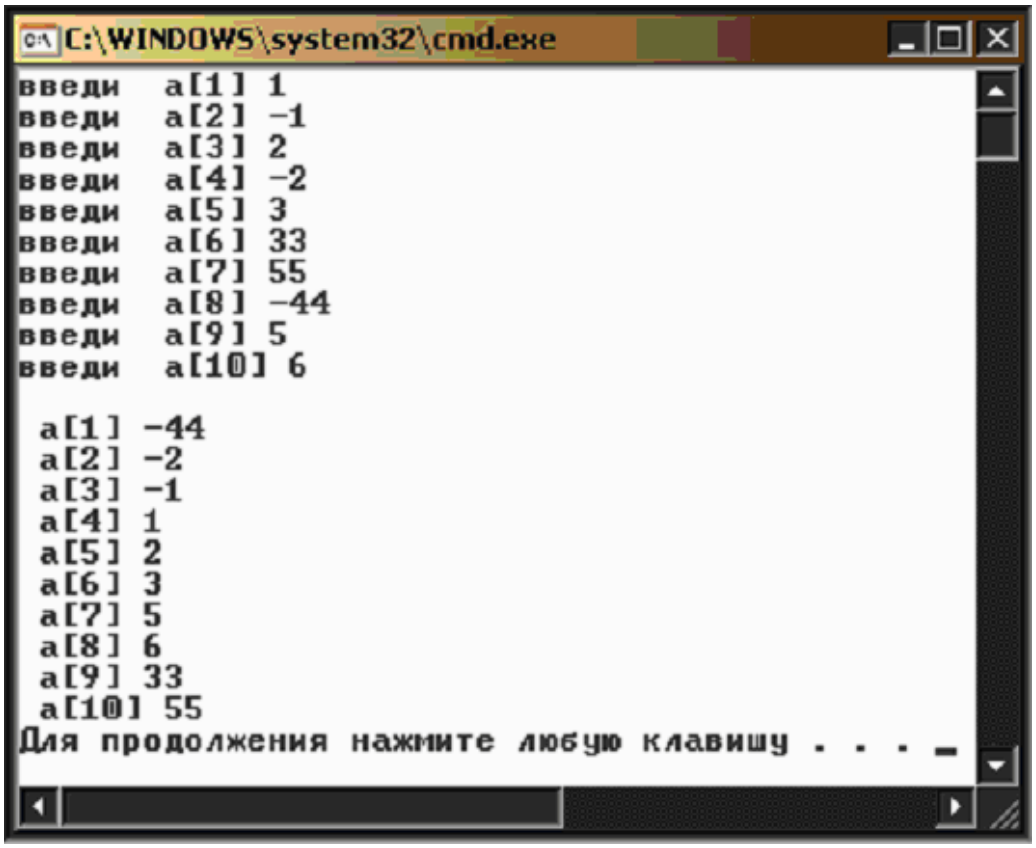


Рис. 2.17. Результат работы программы сортировки массива методом пузырька

Рассмотрим пример программы с комплексной обработкой массива.

Пример 7-4. Ввести одномерный массив, найти в нем минимальный элемент и указать его номер. Затем вычислить сумму элементов от первого до последнего отрицательного элемента (включая их). Вывести массив так, чтобы сначала были все положительные элементы, а потом — все отрицательные.

```
#include <iostream>
using namespace std;
int main()
```

```
{    setlocale (LC_ALL, "Russian");
     int i,j,k1,k2,a[11],b, min,s;
     for (i=1; i<=10; i++)
     { cout<<" Введи a["<<i<<"] "; cin>>a[i]; }
     min=a[1];
     for (i=1; i<=10; i++)
     {    if (a[i]<min) min=a[i]; }
     cout<<"\n минимальный элемент="<<min;
     for (i=1; i<=10; i++)
          if (a[i]<0) k1=i;
     for (i=10; i>=1; i--)
          if (a[i]<0) k2=i;
     s=0;
     for (i=k1; i<=k2; i++)
          s=s+a[i];
     for (i=1; i<=9; i++)
          for (j=1; j<=9; j++)
               if (a[j]<0) { b= a[j]; a[j]=a[j+1]; a[j+1]=b;}
               cout<<"\n преобразованный массив: \n";
     for (i=1; i<=10; i++)
          cout<<a[i]<<" ";
     cout<<"\n";
     return 0;
}
```

Результат работы программы представлен на рис. 2.18

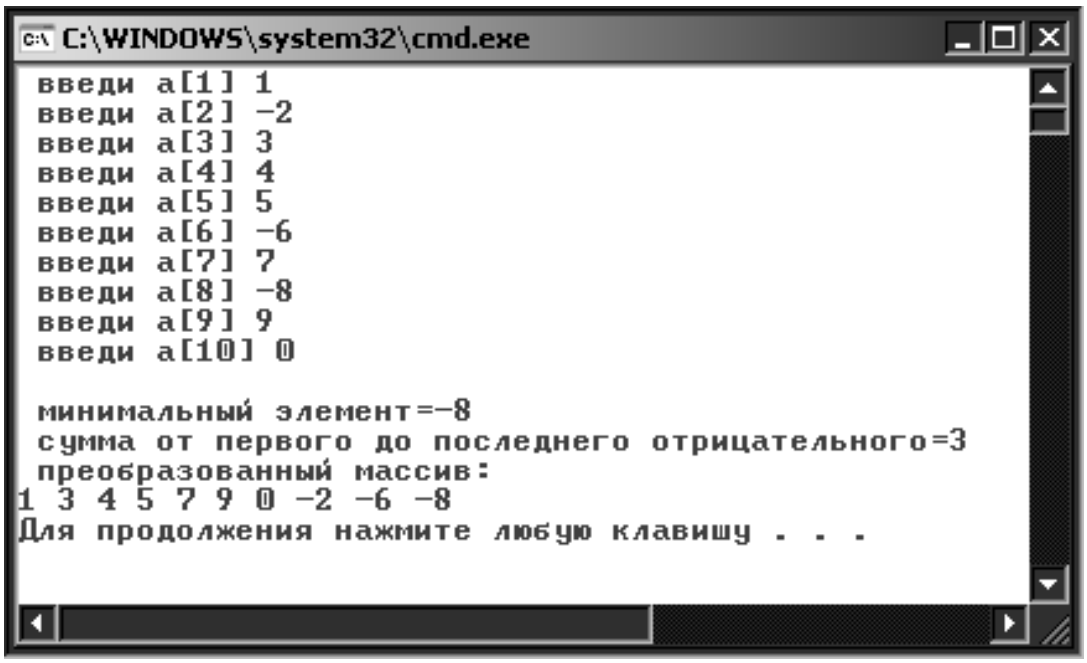


Рис. 2.18. Результат работы программы примера 7-4.

Контрольные вопросы

1. Как описывается одномерный массив на Visual C++? Как идет нумерация индексов массива?
2. Как выполняется деление и получение остатка от деления для целых чисел и целочисленных переменных?
3. Как производят преобразование типов от целого к вещественному и наоборот?
4. Опишите алгоритм сортировки по возрастанию одномерного массива методом пузырька.

8. ДВУМЕРНЫЕ МАССИВЫ

8.1. Понятие о двумерном массиве

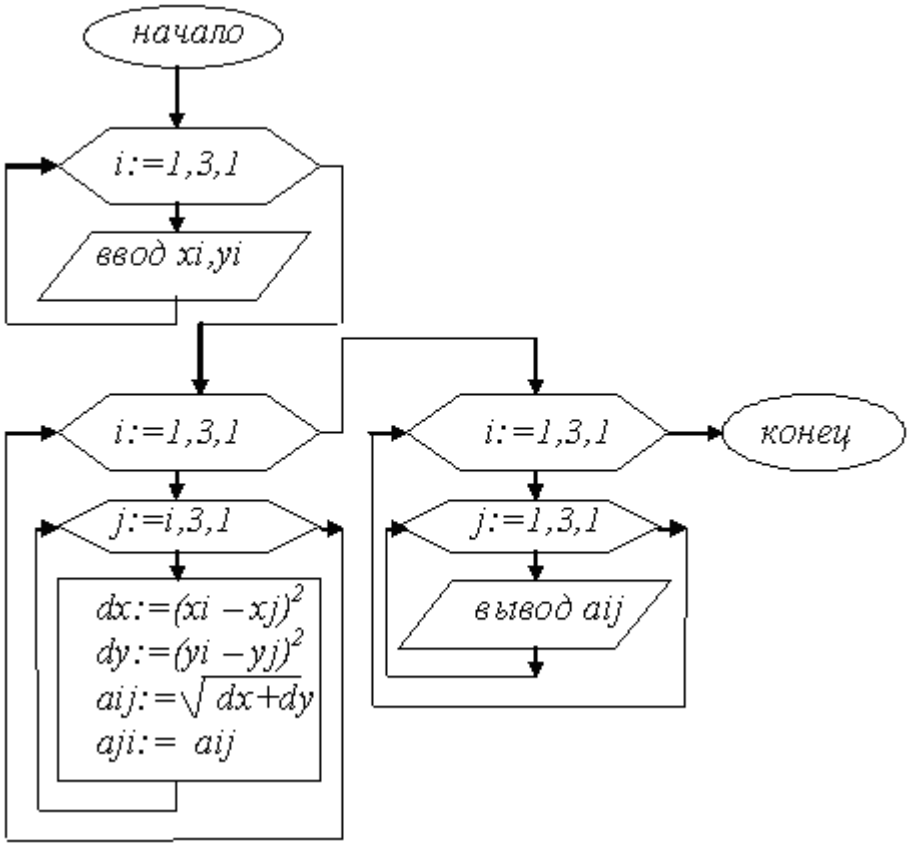
Массивы могут быть и многомерными, например, двумерными. В этом случае размерности записываются в двух квадратных скобках:
`int d[5][4]` – описан целочисленный массив из 5 строк и 4 столбцов.

Напомним, что в математике двумерный массив принято называть матрицей, при этом матрица, у которой число строк равно числу столбцов называется *квадратной*. Если у квадратной матрицы элементы симметрично относительно главной диагонали равны ($a_{21} = a_{12}$ и т.д.), то ее называют *симметричной*. Напомним, что *главная диагональ* матрицы содержит элементы с одинаковыми индексами: a_{11}, a_{22} и т.д. *Побочная диагональ* – вторая диагональ матрицы; *верхний треугольник* – элементы над главной диагональю (включая и элементы на диагонали); *нижний треугольник* – элементы под главной диагональю (включая и элементы на диагонали).

В случае, когда элементы верхнего и нижнего треугольника совпадают, то говорят, что матрица *симметрична*.

Пример 8-1. Построить матрицу расстояний между N ($N = 3$) точками на плоскости, для которых заданы координаты.

Заметим, что эта матрица будет квадратной и симметричной – расстояние от точки с номером i до точки с номером j равно расстоянию от точки j до точки i. Поэтому можно облегчить процесс построения – вычислять только элементы верхнего треугольника, а элементам нижнего треугольника просто присваивать уже вычисленные соответствующие значения. Заметим также, что элементы на главной диагонали будут равны нулю (расстояние от точки до самой себя).




```
//построение двумерного массива
#include <iostream>
#include <cmath>
using namespace std;
int main()
{setlocale(LC_ALL, “Russian”);
 int i,j,x[3+1],y[3+1]; double a[3+1][3+1],dx,dy;
char str[256],str1[256];
for (i=1;i<=3;i++)
{cout<<"введи координату X номер "<<i<<" ";cin>>x[i];
 cout<<"введи координату Y номер "<<i<<" ";
   cin>>y[i];};
for (i=1;i<=3;i++)
{ for (j=1;j<=3;j++)
  { dx=(double)(x[i]-x[j])*(x[i]-x[j]);
    dy=(double)(y[i]-y[j])*(y[i]-y[j]);
    a[i][j]=sqrt(dx+dy); a[j][i]=a[i][j];} };
for (i=1;i<=3;i++)
{cout<<" \n";
  for (j=1;j<=3;j++)
  {cout<<"  a["<<i<<"]["<<j<<"]="<< a[i][j];} }
cout<<"\n"; return 0;}
```

В результате исполнения такой программы, например, получим (рис. 2.19).

Еще раз обращаем внимание, что хотя мы и используем массивы с размерностью на 3 элемента (берем три точки), но описываем массивы на 4 (чтобы не путаться с нулевыми по номеру элементами). Обратите также внимание, что поскольку координаты заданы в виде целых чисел, а на их основе получаем расстояние, которое является вещественным числом, то для преобразования типа от целого к вещественному (точнее – типу двойной точности) используем (`double`). Здесь для вычисления квадратного корня `sqrt()` нужно подключить `cmath` (в отличие от традиционного `math.h`), поскольку при вычислении корня, вообще говоря, возможны комплексные числа.

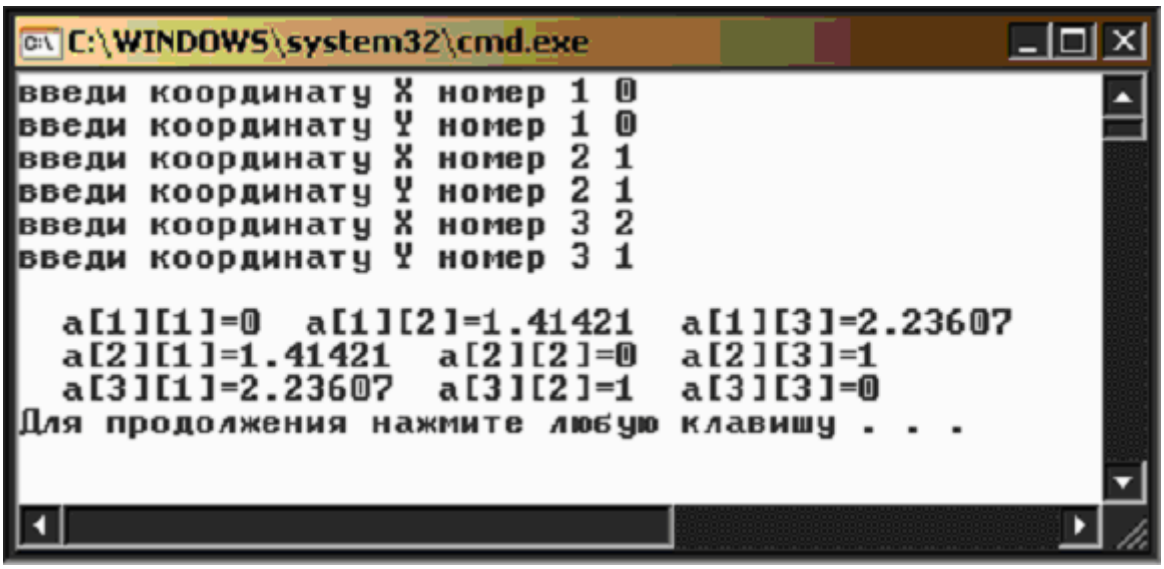


Рис. 2.19. Результат работы примера 8-1

8.2. Датчик случайных чисел

В рассмотренном ниже примере будем использовать датчик случайных чисел. На языке C++ он реализуется функцией

`rand()%RAND_MAX` – дает случайное целое число из интервала от `0` до положительного целого `RAND_MAX-1` (в качестве `RAND_MAX` можно указать сразу конкретное число). Кроме того, для запуска этого датчика случайных чисел (чтобы получать новый набор случайных чисел при каждом новом запуске программы), используют функцию

```
srand(time(0)),
```

т.е. для старта генератора случайных чисел будет использоваться системное время компьютера. Причем это время следует преобразовать в беззнаковое целое (`unsigned int`). Кроме того, требуется подключить заголовочный файл `<stdlib.h>`, содержащий эти функции. Для получения системного времени нужно подключить заголовочный файл `<time.h>`

В примере 8-2 будем впервые использовать так называемые *манипуляторы*. Они позволяют, например, выводить числа по определенному формату – скажем, отводить на вывод числа три позиции (независимо от того, что число может быть и двузначным). Для использования манипуляторов нужно подключить заголовочный файл `<iomanip>`.

Пример 8-2. В двумерном массиве, заданном с помощью датчика случайных чисел, подсчитать количество строк, имеющих элемент, равный первому элементу строки, и указать номера этих строк.

```
# include <iostream>
# include <stdlib.h>
# include <iomanip>
# include <time.h>
using namespace std;
 int main()
{  setlocale(LC_ALL, "Russian");
     const int n=5; bool nal;
   int i,j,k,b, a[n+1][n+1], num[n+1];
// запуск генератора случайных чисел
srand((unsigned int)time(0));
   //формируем массив из случайных чисел (от 0 до 4)
   for (i=1; i<=n; i++)
   for (j=1; j<=n; j++)
   { a[i][j]=rand()%5;}
   // вывод массива
   for (i=1; i<=n; i++)
   { cout<<"\n";
   for (j=1; j<=n; j++)
   { cout<<setw(3)<< a[i][j];} }
// подсчет числа строк (по условию)
   k=0;
   for (i=1; i<=n; i++)
   { b=a[i][1]; nal=false;
     for (j=2; j<=n; j++)
      { if (a[i][j] = = b) nal=true; if nal break;}
      if (nal) {k=k+1; num[k]=i;};
   }
   cout<<" \n количество строк="<<k<<" их номера: ";
   for (i=1; i<=k; i++) cout<<setw(3)<<num[i];
   cout<<"\n";   return(0); }
```

Результат работы этой программы представлен на рис. 2.20.

Обратите внимание, что здесь использована целочисленная константа `const int n=5`. Благодаря этому, мы можем легко изменить размерность массива и количество оборотов цикла (при очередном запуске программы изменим значение этой константы в тексте программы). В программе использована также логическая переменная `nal`, которая примет значение «истина» (true), как только найдем элемент, равный первому в строке. Это позволит сократить число оборотов цикла (не будет продолжаться поиск до конца строки).

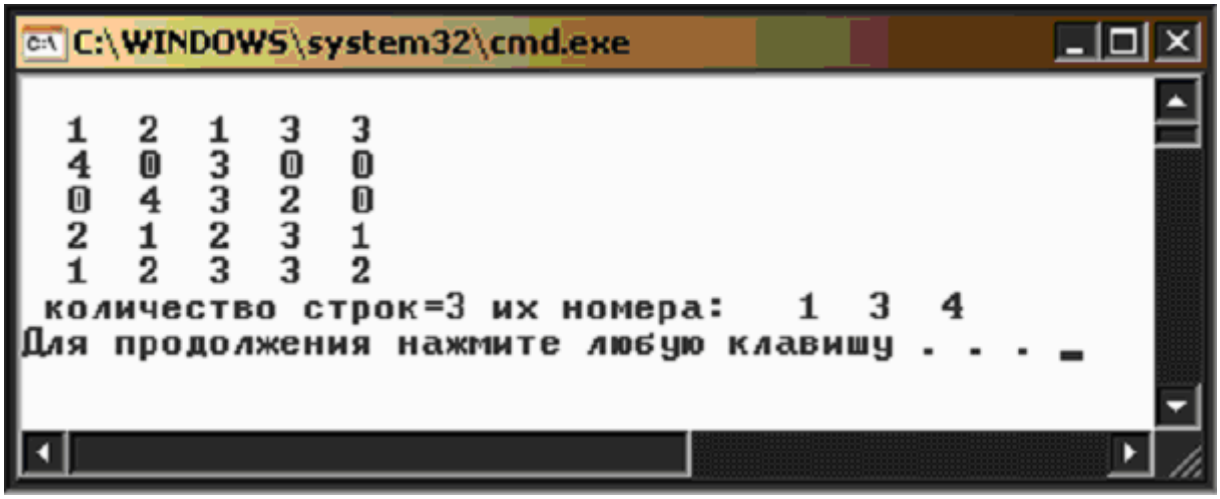


Рис. 2.20. Результат работы программы примера 8.2

Контрольные вопросы

1. Как описывается двумерный массив на Visual C++?
2. Как представляется симметричная матрица в виде двумерного массива?
3. Что такое датчик случайных чисел? Как его используют?
3. Для чего служат манипуляторы в команде вывода?
4. Как описываются и используются константы на Visual C++?.

9. ФУНКЦИИ

9.1. Понятие о пользовательских функциях

В C++ можно создавать пользовательские функции. Собственно говоря, мы с самого начала создавали функции: **main()** – не что иное, как главная функция пользователя.

В библиотеке C++ имеется немало встроенных функций. Они размещены в отдельных заголовочных файлах (тех самых, которые подключаются с помощью **#include**). Заголовочный файл имеет имя и расширение.**h** Мы уже использовали заголовочные файлы:

iostream.h – содержит функции для ввода-вывода;

math.h – содержит математические функции.

Для *создания функции пользователя* (и ее последующего исполнения) необходимо ее описать:

1) задать прототип функции:

***тип имя_функции* (*параметры*);**

это делается там же, где описываются переменные, т.е. до заголовка **main() (**обратим внимание, что в конце записи прототипа точка с запятой ставится обязательно!) ;

2) собственно описать функцию

***тип имя_функции*(*параметры*)**

{ тело функции }

Такое описание делается вне тела другой функции, в том числе и функции **main()** (как правило, за закрывающей фигурной скобкой главной функции).

В конце описания (после фигурной скобки) точка с запятой не ставится. Функция может иметь только одно возвращаемое значение, которое определяется служебным словом **return (*возвращаемое_значение*),** которое размещается в конце функции (перед закрывающей фигурной скобкой).

Внутри функции могут быть описаны свои локальные переменные.

Пример 9-1. Вычислить площадь пятиугольника, у которого известны длины сторон и двух диагоналей (см. рис. 2.21).

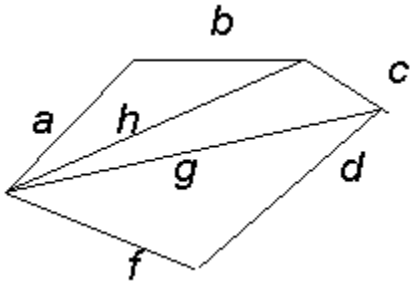
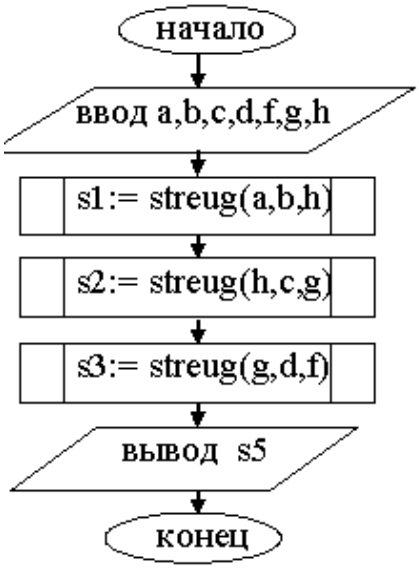
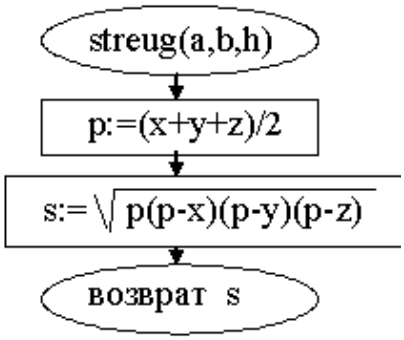


Рис. 2.21. Пятиугольник со сторонами a, b, c, d, f и диагоналями h, g

Очевидно, что площадь такого пятиугольника – сумма площадей треугольников, для каждого из которых известны длины сторон. Для вычисления площади каждого треугольника (условно обозначим его стороны за *x*, *y*, *z*) применим формулу Герона: $S = \sqrt{p(p-x)(p-y)(p-z)}$, где $p = (x+y+z)/2$.






```
// площадь 5-угольника через 3-угольники
#include <iostream>
#include <cmath>
using namespace std;
float streug(float x,float y,float z);
int main()
{    float a,b,c,d,f,g,h,s1,s2,s3,s5;
     cout<<"\n введи a,b,c,d,f,g,h ";
     cin>>a>>b>>c>>d>>f>>g>>h;
     s1=streug(a,b,h); s2=streug(h,c,g);
   s3=streug(g,d,f);     s5=s1+s2+s3;
  cout<<"\n"; cout<<s5<<"\n";
return 0;}
float streug(float x,float y,float z)
//ф-я для вычисления площади 3-ника по его сторонам
   {   float p,s;
       p=(x+y+z)/2;  s=sqrt(p*(p-x)*(p-y)*(p-z));
       return (s); }
```

Если провести расчеты по этой программе с такими данными: a = 1,5; b = 1; c = 2; d = 2,5; f = 1; g = 2,5; h = 2, то получим площадь 5-угольника = 3,90 (примерно).

9.2. Рекурсия

В C++ допустима *рекурсия* – обращение функции к самой себе. Проиллюстрируем использование рекурсии на примере.

Пример 9-2. Рассчитать число сочетаний из N элементов по M:

$$C_N^M = \frac{N!}{M!(N-M)!}$$ для ($N > M$).

По такой формуле можно рассчитать, например, каким числом способов можно рассадить учеников класса (скажем, N=30) по двое (M=2).

Здесь потребуется трижды вычислять факториал для разных чисел. Вычисление факториала и оформим рекурсивной функцией.

```
// число сочетаний из N по M
#include <iostream>
#include <cmath>
using namespace std;
int n,m; double c;
float fact(int n);
int main()
{    setlocale(LC_ALL, "Russian");
     cout<<"\n  введи N и M "; cin>>n>>m;
     c=floor(fact(n)/(fact(m)*fact(n-m)) );
cout<<"\n число сочетаний из "<<n<<" по "<<m<<" = "<<c<<"\n";
   return (0); }
float fact(int n)
{if (n==0) return(1);else return (n*fact(n-1));}
```

Как видно, внутри функции `fact` имеется обращение к ней же, но со значением параметра, меньшим на 1. Поэтому, если, например, задать n=5, то внутри функции произойдет обращение к ней же, но с n = 4, затем с n=3, n = 2, n = 1, n = 0. Для n = 0 функция примет значение 1 и процесс пойдет в обратную сторону, последовательно выполняя умножение на 1, на 2, на 3, 4, 5. В результате и получим значение 5!.

Функцию факториала лучше оформить функцией вещественного типа (хотя факториал – целое число) во избежание переполнения при вычислениях. Однако, поскольку число сочетаний это строго целое число, то для его окончательного вычисления применили функцию округления вещественного до целого `floor` (она выдает наибольшее целое, не превосходящее данное число). Заметим, кстати, что для правильного округления вещественного числа x нужно указывать `floor(x+0.5)`. Если запустить программу и задать n = 9, m = 5, то получим в ответе 126.

Пример 9-3. Провести сортировку одномерного массива последовательным поиском максимума и перестановкой его в конец, причем применить рекурсивную процедуру сортировки.

```
#include <iostream>
using namespace std;
const int n=5; float a[n+1];
int fmax(float a[], int nach, int kon);
```

```
void sort (float a[], int nach, int kon);
int main()
{    setlocale(LC_ALL, "Russian");
     int i;
     for (i=1; i<=n; i++)
   { cout<<"\n введи a["<<i<<"] "; cin>>a[i];}
   sort (a,1,n);
   for (i=1; i<=n; i++)
   {cout<<"  a["<<i<<"]="<<a[i];} cout<<"\n";
   return(0);}
int fmax(float a[], int nach, int kon)
{//поиск максимума в массиве и запоминание номера
 // макс. элемента
     int k, kmax; float max;
     kmax=nach; max=a[nach];
   for (k=nach; k<=kon; k++)
   {if (a[k]>max) {max=a[k]; kmax=k;}}
   return kmax;}
void sort (float a[], int nach, int kon)
{//сортировка в массиве перестановкой максимума
 // в конец рекурсивным образом
     float b; int nmax,kons;      kons=kon;
   if (nach<=kon)
   { nmax=fmax(a,nach, kons);
     b=a[nmax]; a[nmax]=a[kon]; a[kon]=b;
      sort(a, nach, kons-1);}  }
```

Результат работы такой программы на рис. 2.22.

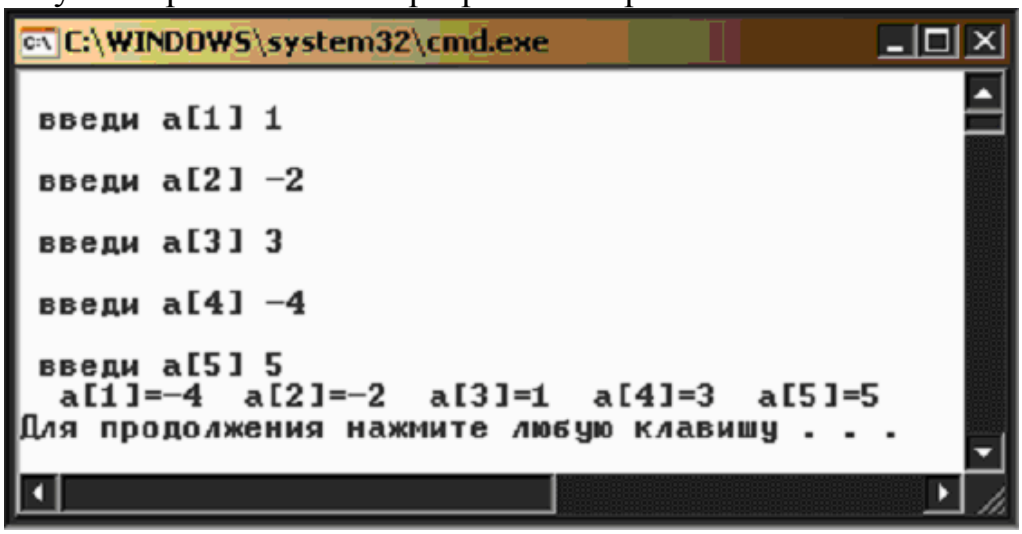


Рис. 2.22. Результат работы программы примера 9-3

9.3. Вызов функции из функции

Рассмотрим пример с использованием *нескольких функций*. Особенностью языка С++ является невозможность для функции определять внутри своего тела другую функцию. То есть не допускаются вложенные определения. Но вполне допускается вызов функции из другой функции.

Пример 9-4. Рассчитать треугольник Паскаля[4] вида:

$$C_0^0$$

$$C_1^0 \quad C_1^1$$

$$C_2^0 \quad C_2^1 \quad C_2^2$$

$$C_3^0 \quad C_3^1 \quad C_3^2 \quad C_3^3$$

и т.д. Здесь используется число сочетаний из n элементов по m, формула для расчета которого приведена выше. Оформим расчет числа сочетаний функцией, а вычисление факториала – внутренней функцией, причем, как и ранее, оформим вычисление факториала рекурсивным образом.

```
#include <cmath>
#include <iomanip>
using namespace std;
int n,m,r,i,j; double c;
float fact(int n);  int soch(int n,int m);
int main()
{ setlocale(LC_ALL, "Russian");
cout<<"\n  введи размер треугольника Паскаля"; cin>>r;
 for (i=0;i<=r;i++)
     {cout<<"\n";
         for (j=0;j<=i;j++)
          {c=soch(i,j);cout<<setw(r*(r-i+j+1))<<c;};
          cout<<"\n";   }
 return (0);}
float fact(int n)
{if (n==0) return(1); else return (n*fact(n-1)); }
```

[4] Треугольник Паскаля образуется по правилу: по краям каждой строки стоят единицы, а каждое из остальных чисел равно сумме двух стоящих над ним чисел предыдущей строки.

```
int soch(int n,int m)
{ int c= floor(fact(n)/(fact(m)*fact(n-m)) );
  return(c);}
```

В данном примере нам потребовался форматированный вывод. Для его организации в C++ вновь применим так называемые *манипуляторы* (для их использования нужно подключить заголовочный файл **iomanip**). Конкретно мы использовали манипулятор `setw(int)` – который задает ширину поля (количество позиций, отведенных для вывода переменной) с выравниванием по правому краю.

Если запустить данную программу и задать размер треугольника равным 3, то получим (рис. 2.23).

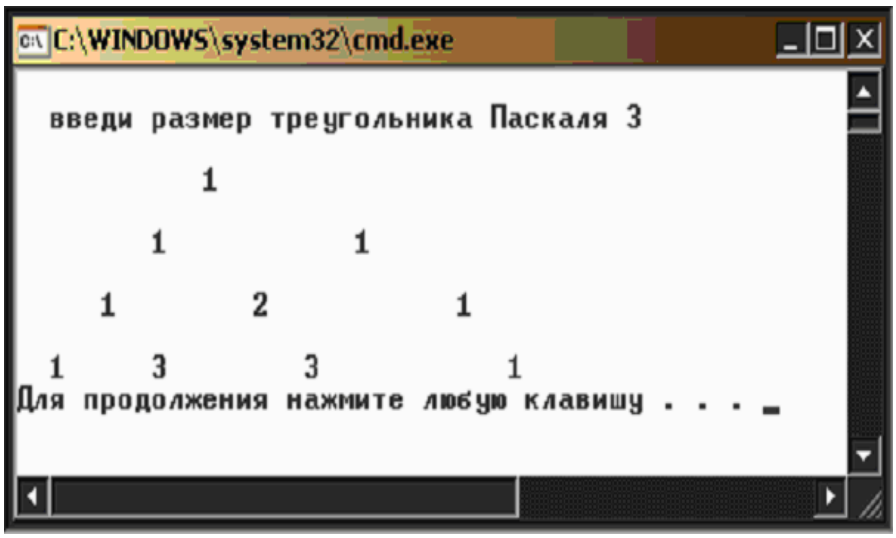


Рис. 2.23. Результат работы программы примера 9-4

9.4. Функция типа void и глобальные переменные

Но как быть, когда нам потребуется вызвать функцию, которая должна получить нам *несколько значений в результате*. В Паскале мы организовывали так называемые *процедуры*. Здесь для этого проще всего использовать так называемые функции типа **void**. Такие функции не содержат оператора **return** и поэтому как бы ничего не передают наружу. Однако в C++ существует понятие *глобальной переменной*, т.е. такой переменной, которая доступна в любой функции данной программы. Если, таким образом, обратится к функции типа **void** просто по имени (без использования присваивания) и внутри нее пересчитать глобальные переменные, то после этого в основной программе значения глобальных переменных изменятся. Напомним, что глобальные переменные описываются до открытия главной функции **main**().

Замечание. Более строго ограничение на *возврат из функции только одного значения* можно обойти с помощью *указателей*, но работа с ними будет рассмотрена нами в разд. 12.2 данной главы.

Пример 9-5. Даны три тройки чисел*: a1, b1, c1; a2, b2, c2; a3, b3, c3.* Найти в каждой из них наибольшее и наименьшее, а затем наибольшее и наименьшее среди наибольших и отдельно – среди наименьших. (Здесь и организуем функцию, в которой получим «на выходе» два значения – максимум и минимум для заданной тройки чисел.)

```
#include <iostream>
#include <iomanip>
using namespace std;
void maxmin(double x,double y,double z);
double mx, mn;// глобальные переменные (из maxmin)
int main()
{ setlocale(LC_ALL, "Russian");
double max1,min1,max2,min2,max3,min3,mxmax,mnmax,
mxmin, mnmin, a1,b1,c1, a2,b2,c2, a3,b3,c3;
cout<<"\n введи a1,b1,c1 "; cin>>a1>>b1>>c1;
cout<<"\n введи a2,b2,c2 "; cin>>a2>>b2>>c2;
cout<<"\n введи a3,b3,c3 "; cin>>a3>>b3>>c3;
maxmin(a1,b1,c1);   max1=mx; min1=mn;
cout<<"\n max1,min1"<<setw(5)<<max1<<setw(5) <<min1;
maxmin(a2,b2,c2);   max2=mx; min2=mn;
cout<<"\n max2,min2"<<setw(5)<<max2<<setw(5) <<min2;
maxmin(a3,b3,c3);   max3=mx; min3=mn;
cout<<"\n max3,min3"<<setw(5)<<max3<<setw(5) <<min3;
maxmin(max1,max2,max3);   mxmax=mx; mnmax=mn;
cout<<"\n maxmax,minmax "<<setw(5)<<mxmax<<setw(5) <<mnmax;
maxmin(min1,min2,min3);   mxmin=mx; mnmin=mn;
cout<<"\n maxmin,minmin "<<setw(5)<<mxmin<<setw(5) <<mnmin;
cout<<"\n";   return (0);}
void maxmin(double x,double y,double z)
     {if ((x>y)&&(x>z)) mx=x; if ((y>x)&&(y>z)) mx=y;
      if ((z>x)&&(z>y)) mx=z;if ((x<y)&&(x<z)) mn=x;
      if ((y<x)&&(y<z)) mn=y;if ((z<x)&&(z<y)) mn=z;
 }
```

В результате, например, получим (рис. 2.24):

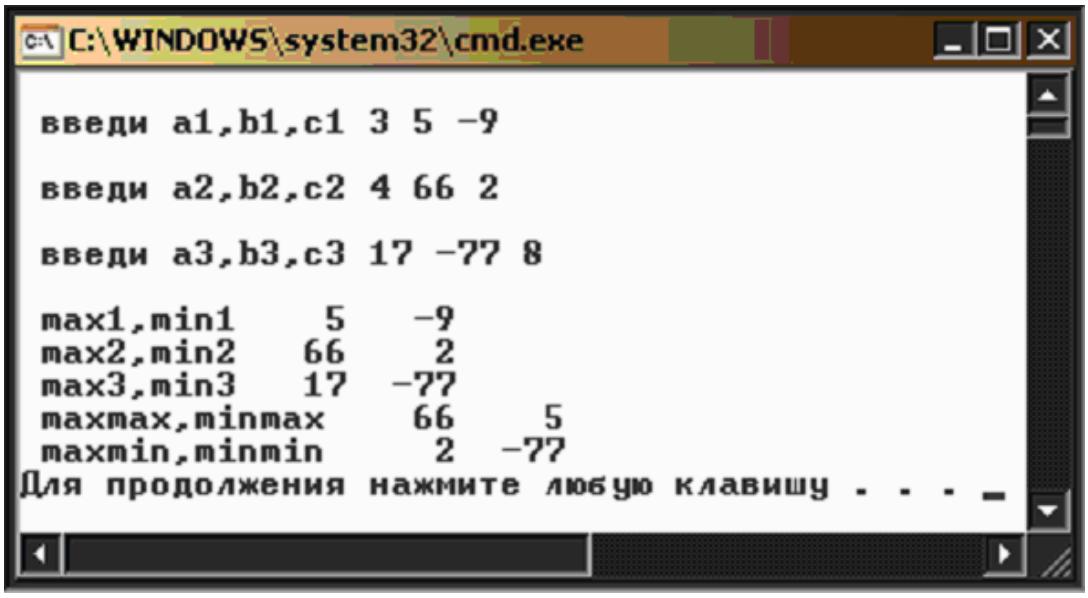


Рис. 2.24. Результат работы примера 9-5

Пример 9-6. Составить программу для решения задачи «Ханойские башни». Напомним, что в ней имеется три стержня, на которых находятся диски последовательно уменьшающегося диаметра. Требуется переложить диски с 1-го стежня (Source) на 3-й (Dest), используя второй стержень (Tmp). Диски можно брать только по одному и класть можно только меньший (по диаметру) на больший. В результате на 3-м стержне диски также должны быть сложены по возрастанию диаметров (как и было на исходном).

Для того чтобы перенести самый большой диск, нужно сначала перенести все диски, кроме последнего, на второй стержень, потом перенести самый большой на третий, после чего останется перенести все остальные диски со второго на третий. Задачу о переносе N–1 диска мы решаем аналогично, только поменяем стержни местами (при первом переносе конечным стержнем будем считать второй, а не третий, при втором переносе начальным вместо первого будет второй). Как видите, задачка разрешима. И правда, ведь задачу о N–1 дисков мы сведем к задаче о N–2 дисков, ту в свою очередь к N–3 дискам, и так вплоть до 1 диска. Этот метод легко программируется с помощью рекурсии.

```
#include <iostream>
using namespace std;
//ханойские башни (рекурсивно)
void move_disks(int n, char Source, char Dest, char Tmp);
int main()
{    setlocale(LC_ALL, "Russian");
     int n; cout<<"? n disks "; cin>>n;
     move_disks(n,'A','C', 'B');
return 0;}
```

```
void move_disks(int n, char Source, char Dest, char Tmp)
// процедура перемещения диска
{if (n==1) cout<<"Disk1 From "<<Source<<" To "<<Dest<<endl;
     else
    { move_disks(n-1, Source, Tmp, Dest);
     cout<<"Disk"<<n<<" From "<<Source<<" To "<<Dest<<endl;
     move_disks(n-1, Tmp, Dest, Source);
     }
}
```

9.5. Передача в функцию имени функции

Как передать в функцию имя функции? Точно так же, как любую другую величину: в списке параметров перед именем параметра указать его тип. Значит, специально нужно будет определить собственный тип для функции. А правильнее задать синоним с помощью ключевого слова **typedef**:

```
typedef double (*Pfun) (double, double)
```

(в данном случае описали функциональный тип `Pfun` для функции от двух параметров).

Пример 9-7. Создадим программу с функцией для вывода таблицы значений произвольной функции в заданных пределах и с указанным шагом. И соответственно нужную функцию будем туда передавать (в функцию вывода).

```
// передача в функцию имени функции
// для построения таблицы переданной функции
#include <iostream>
#include <math.h>
#include <iomanip>
using namespace std;
typedef double(*Pfun)(double);
void print_tabl(Pfun fun, double xn,  double xk, double dx);
double cosh(double x);
int main()
{    setlocale(LC_ALL, "Russian");
     double xn, xk, dx;
     cout<<"введите xn, xk, dx "; cin>>xn>>xk>>dx;
         print_tabl(cosh, xn, xk, dx);
return(0);
}
void print_tabl(Pfun fun, double xn, double xk,  double dx)
{double x;
 cout<<"---------------------------------------\n";
```

```
cout<<"|      X      |           Y            |\n" ;
cout<<"-------------------------------------|\n";
for (x= xn; x<=xk; x+=dx)
cout<<"\n"<<setw(15)<< x <<setw(15)<<fun(x);
cout<<"\n-------------------------------------|\n";
}
double cosh(double x)
{ double  y;
 y=(exp(x)+exp(-x))/2;
    return(y)   ;  }
```

В результате выполнения программы получим, например, такой результат (рис. 2.25) .

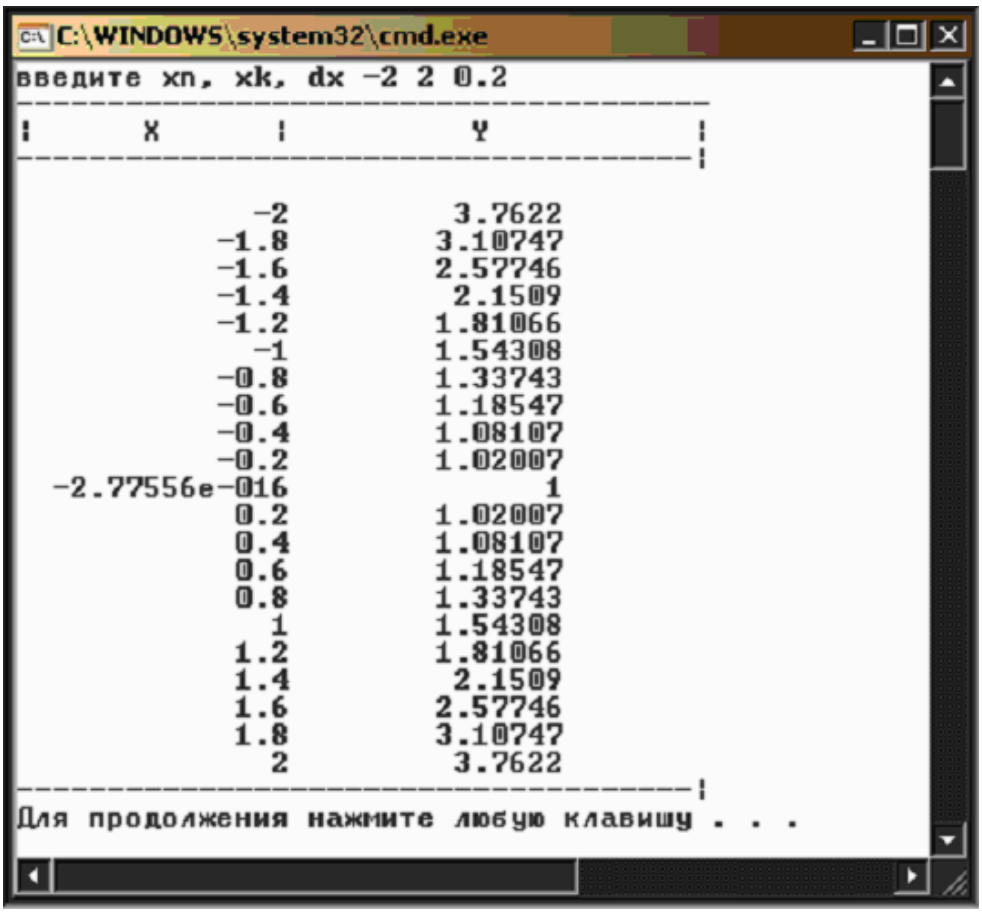


Рис. 2.25. Результат работы примера 9-7.

Контрольные вопросы

1. Как оформляются пользовательские функции на Visual C++?
2. Как вызывается пользовательская функция из главной функции программы? Как происходит возврат результата работы функции?
3. Что такое рекурсия?
4. Как осуществляется вызов функции из функции?
5. Что такое функция типа void? В каких случаях их используют?
6. Как передать в функцию имя другой функции?

10. СОБСТВЕННАЯ БИБЛИОТЕКА ПРОГРАММИСТА

Созданные пользователем функции целесообразно объединить в отдельную собственную библиотеку программиста. Для этого создадим (в своей папке !) на диске специальную папку под названием **SUBPROG**. В ней и будем размещать заголовочные файлы, содержащие функции пользователя.

Всякий заголовочный файл оформляется директивами препроцессору:
#infdef *имя_файла*
#define *имя_файла*
…
#endif

Это сочетание директив предотвращает повторные включения директив-компонентов. ***Имя_файла*** может быть любым, но не следует использовать его еще в каком-либо качестве в своих программах. Таким образом будет создан заголовочный файл ***имя_файла*.h** Такой файл содержит прототипы соответствующих функций, константы и т.п. Кроме того, там могут содержаться директивы на включение в программу тех заголовочных файлов из библиотеки компилятора, использование которых неизбежно в любой задаче (ввод/вывод, математические и т.п.)

В отличие от встроенных в компилятор, собственные списки и функции включаются с указанием пути к файлу, в который они записаны, и в обычных двойных кавычках.

Будем каждую библиотечную функцию записывать в своем отдельном программном файле. Имя файла и имя записанной в него функции могут и различаться, но желательно, чтобы они совпадали. Расширения таких файлов – как и для любой программы C++ –.cpp.

10.1. Перегрузка функций

C++ позволяет определить несколько функций с одним и тем же именем, но с различным набором аргументов – это называется *перегрузкой функций*. Приведем пример, иллюстрирующий, зачем нужна перегрузка.

В библиотеке стандартных функций (см. прил. 1) имеется три функции для определения абсолютной величины числа трех типов:

```
int abs(int)
long labs(long)
double fabs(double)
```

Гораздо удобнее было бы вызывать одну функцию для выполнения всех вариантов.

Для осуществления перегрузки необходимо записать три варианта прототипа новой функции и три варианта описания функции в форме

функции **abs1**. Теперь при вызове функции **abs1** компилятор определит тип переменной и исполнит нужный вариант. Итак, перегруженная функция **abs1** имеет вид:

```
int abs1(int);
double abs1(double);
long abs1(long);

int abs1(int i);
{ return abs(i); }

double abs1(double a);
{ return fabs(a); }

long abs1(long i);
{ return labs(i); }
```

А программа с ее использованием будет выглядеть примерно так:

```
#include <iostream>
#include <math.h>
using namespace std;
int x=-7;double z=-1.56789e2; long y=-33156;
int abs1(int);
double abs1(double);
long abs1(long);
int main()
{ setlocale(LC_ALL, "Russian");
cout<<"\n"<<abs1(x)<<"\n"<<abs1(y)<<"\n"<<abs1(z);
  cout<<"\n";    return (0); }
int abs1(int x)
{ return abs(x); }
double abs1(double z)
{ return fabs(z); }
long abs1(long y)
{ return labs(y); }
```

Когда запустим на исполнение, то получим (рис. 2.26):

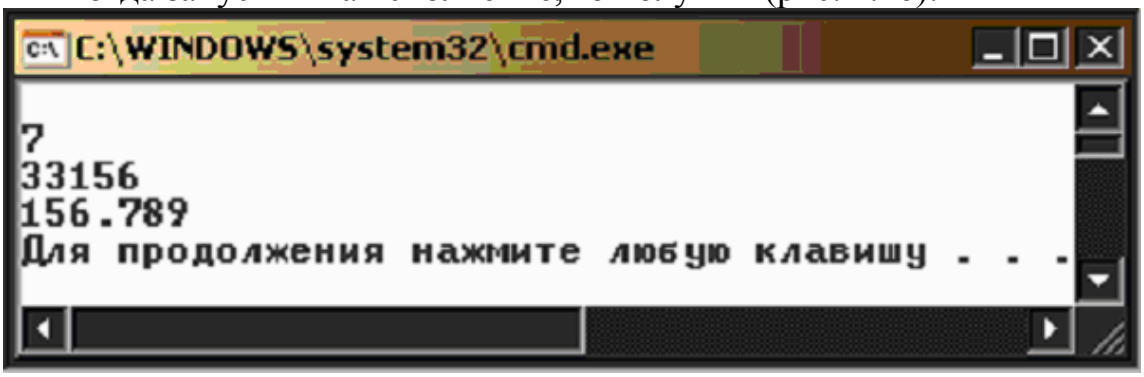


Рис. 2.26 Результат работы программы примера

11. ПЕРЕЧИСЛИМЫЙ ТИП

11.1. Понятие о перечислимом типе

В C++ имеется *перечислимый* тип. Его называют **enum**. При описании такого типа задаются наборы констант (в фигурных скобках). При этом идентификаторам-константам присваиваются значения натурального ряда от нуля. Если значение одного из идентификаторов задается явно, то последующим присваиваются номера по возрастанию через 1. Рассмотрим на примере.

Пример 11-1. Подсчитать число часов рабочей недели, если известно, что суббота и воскресенье – выходные, а в пятницу рабочий день длится 8 ч (в остальные дни 8,25 ч).

```
#include <iostream>
using namespace std;
float t;
enum dni {Monday=1, Tuesday, Wednesday, Thursday,
Friday, Saturday, Sunday} ;
    int main()
    { setlocale(LC_ALL, "Russian");
    dni d;  t=0;
      for (d=Monday; d<=Friday; d=dni(d+1))
        { if (d == Friday) t=t+8; else t=t+8.25; } ;
       cout << " сумма часов раб.недели="<<t<<"\n";
    return(0); }
```

Обратите внимание на необычную форму записи приращения параметра цикла: `d=dni(d+1)`. К сожалению, операция инкремента `d++` для перечислимого типа не работает. Как обойти эту проблему, рассмотрим в следующем разделе («12. Указатели»).

11.2. Множественный выбор

В C++ имеется *структура множественного выбора* **switch** (переключатель). Переключатель начинается с заголовка, определяющего имя метки. Тело переключателя заключено в фигурные скобки. Текст тела переключателя разделен метками **case**. Двоеточие – признак метки. Рассмотрим использование структуры множественного выбора на следующем примере.

Пример 11-2. Дан перечислимый тип muns (*названия месяцев года*), описаны переменные int *d*1, *d*2*;* muns *m*1, *m*2. Извне вводится дата в виде двух чисел: номер дня и номер месяца. Проверить, предшествует ли (в рамках года) дате *d*2, *m*2 дата *d*1, *m*1. Вывести соответствующее сообщение.

```
#include <iostream>
using namespace std;
int s;  float t;  int nm1,nm2;
int main()
{ setlocale(LC_ALL, “Russian”);
enum muns
{jan=1,feb,mar,apr,may,jun,jul,aug,sep,oct,nov,dec};
int d1,d2;
muns m1,m2;
cout<<"введи день и номер месяца первой даты ";
cin>>d1>>nm1;
cout<<" введи день и номер месяца второй даты ";
cin>>d2>>nm2;
   switch (nm1)
   { case 1 :      m1=jan; break;
     case 2 :      m1=feb; break;
     case 3 :      m1=mar; break;
     case 4 :      m1=apr; break;
     case 5 :      m1=may; break;
     case 6 :      m1=jun; break;
     case 7 :      m1=jul; break;
     case 8 :      m1=aug; break;
     case 9 :      m1=sep; break;
     case 10 :     m1=oct; break;
     case 11 :     m1=nov;  break;
     case 12 :     m1=dec;  break; }
    switch (nm2)
   { case 1 :     m2=jan; break;
     case 2 :     m2=feb; break;
     case 3 :     m2=mar;  break;
     case 4 :     m2=apr; break;
     case 5 :     m2=may; break;
     case 6 :     m2=jun; break;
     case 7 :     m2=jul; break;
     case 8 :     m2=aug; break;
     case 9 :     m2=sep;  break;
     case 10 :    m2=oct; break;
     case 11 :    m2=nov;  break;
```

```
     case 12 :    m2=dec; break;};
   if (m1<m2) cout<<" предшествует \n"; else
   if  ((m1==m2)&&(d1<d2))cout<<"  предшествует  \n";
else cout<<"не предшествует \n";
return(0); }
```

Контрольные вопросы

1. Что такое перегрузка функций?
2. Дайте понятие о перечислимом типе.
3. Как организуется структура множественного выбора?

---

12. УКАЗАТЕЛИ

12.1. Понятие об указателях

Указатель – это целочисленная переменная, содержащая адрес (номер ячейки памяти) другой переменной.

Описание:

тип **имя*, т.е перед именем указателя ставится знак *

Например, float a, *aP;

Так как указатель есть адрес, а адреса распределяет сам компьютер, то указатель не может быть инициализирован непосредственно. Операция *адресации*, т.е. присваивание объявленному указателю адреса переменной, производится, например, следующим образом:

```
float a, *aP;
aP=&a;
```

Знак `&` перед именем переменной возвращает ее адрес, или, как говорят, дает *ссылку*.

Для того чтобы получить значение переменной, производится операция разыменования. Знак * перед указателем возвращает значение переменной, на которую указатель ссылается. Например,

```
float a1, a=6.0, *aP;
aP=&a; a1=*aP;
```

Здесь в результате `a1` примет значение 6.0 .

Операции адресации и разыменования взаимно компенсируют друг друга при их применении к указателю последовательно и в любом порядке:

`&*aP;` эквивалентно `*&aP;`

Пример 12-1. Проведем манипулирование данными с помощью указателей: переменной *a* присваивается значение 5, указателю *aP* присваивается ее адрес, затем по адресу указателю присваивается значение 7. В ре-

зультате оказывается, что значение переменной *a* изменилось на 7, хотя непосредственно к ней никто не обращался. И далее изменим непосредственно значение *a* на 9 и при обращении к ней, как непосредственно, так и через указатели увидим, что значение изменилось на 9.

```
//  использование указателей
#include <iostream>
using namespace std;
int a, *aP;
int main()
{    setlocale(LC_ALL, "Russian");
     a=5; cout<<"\n a= "<<a;
     aP=&a;  cout<<"\n *aP= "<<*aP;
     cout<<"\n присваиваем *aP значение 7 ";
   *aP=7;  cout<<"\n *aP= "<<*aP;  cout<<" a= "<<a;
   AnsiToOem(, str);
   cout<<"\n присваиваем a значение 9 ";
   a=9;    cout<<"\n a= "<<a;  cout<<" *aP= "<<*aP;
   cout<<"\n";
   return (0); }
```

В результате получим (рис. 2.27):

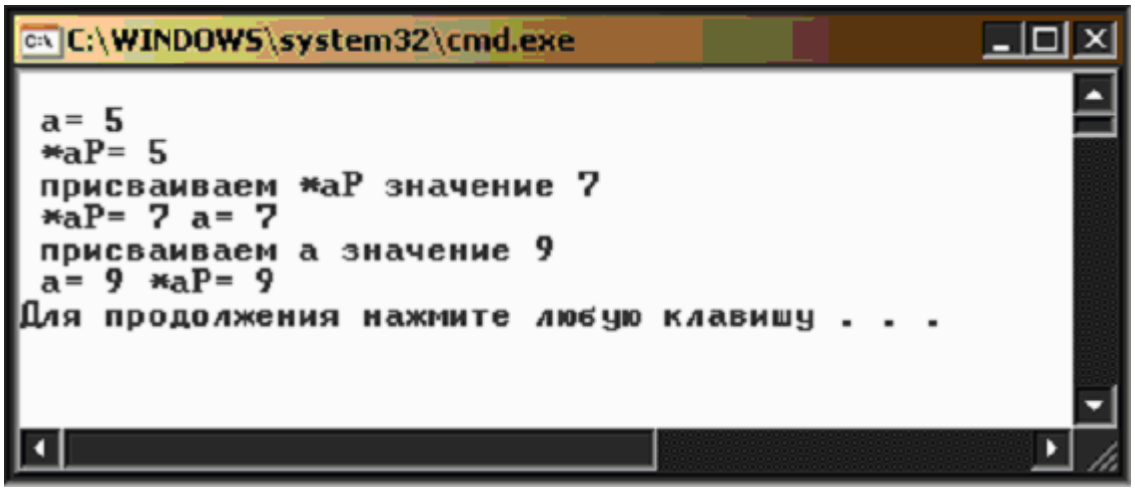


Рис. 2.27. Результат работы программы примера 12-1

12.2. Указатели и функции

Как известно, при обращении к функции параметры, передаваемые функции, являются локальными переменными данной функции. Изменения, внесенные в аргументы во время выполнения функции не влияют на переменные, значения которых передаются функции. (Это одна из причин, почему функция по результатам работы выдает *только одно значение*.)

Проиллюстрируем это на примере.

Пример 12-2. Создать функцию `swap(x,y)`, которая меняет значения переменных *x,y* местами и проверить значения переменных до обращения к функции, внутри функции и после обращения к функции.

```
// передача параметров в функцию (по значению)
#include <iostream>
#include <iomanip>
using namespace std;
void swap(int x, int y);
int main()
{   setlocale(LC_ALL, “Russian”);
int x,y;     x=5, y=10;
cout<<"\n x,y до обращения"<<setw(3)<<x<<setw(3) <<y;
    swap(x,y);
cout<<"\n x,y после обращения"<<setw(3)<<x<<setw(3)
<<y<<"\n";
   return(0);}
void swap(int x, int y)
{   int z;
cout<<"\nx,y внутри функции до преобразования"
<<setw(3)<<x<<setw(3)<<y; z=x; x=y; y=z;
 cout<<"\n x,y внутри функции после преобразования"
<<setw(3)<<x<<setw(3)<<y;    }
```

В результате исполнения программы получим (рис. 2.28):

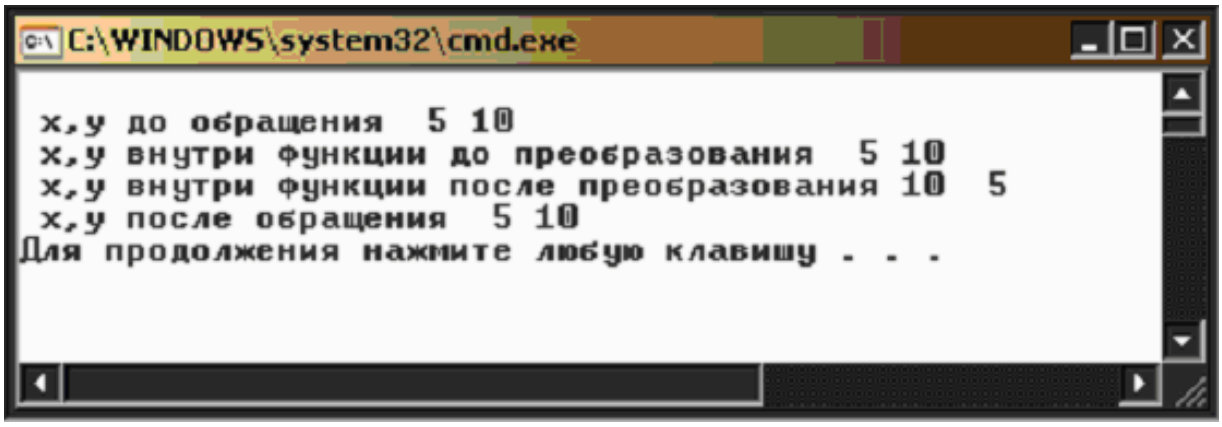


Рис. 2.28. Результат работы программы примера 12-2

Как видим, несмотря на то, что внутри функции значения переменных поменялись, по окончании работы, в основной программе они оказались неизменными.

С данной проблемой можно справиться, используя передачу аргументов по ссылке с помощью указателей. При передаче указателя на самом деле передается лишь адрес объекта, а, следовательно, функция получает возможность манипулировать значением, находящимся по этому адресу.

```
//передача параметров (по ссылке с пом. указателей)
#include <iostream>
#include <iomanip>
using namespace std;
void swap(int *x, int *y);
int main()
{   setlocale(LC_ALL, "Russian");
int x,y;     x=5, y=10;
cout<<"\n x,y до обращения "<<setw(3)<<x<<setw(3)<<y;
    swap(&x,&y);
cout<<"\n x,y после обращ-я "<<setw(3)<<x<<setw(3)
<<y<<"\n";
  return(0);}
void swap(int *px, int *py)
{int z;
cout<<"\n x,y внутри функции до преобраз." <<setw(3)
<<*px<<setw(3)<<*py;
z=*px; *px=*py; *py=z;
cout<<"\nx,y внутри функции после преобр." <<setw(3)
<<*px<<setw(3)<<*py;    }
```

В результате получаем (рис. 2.29):

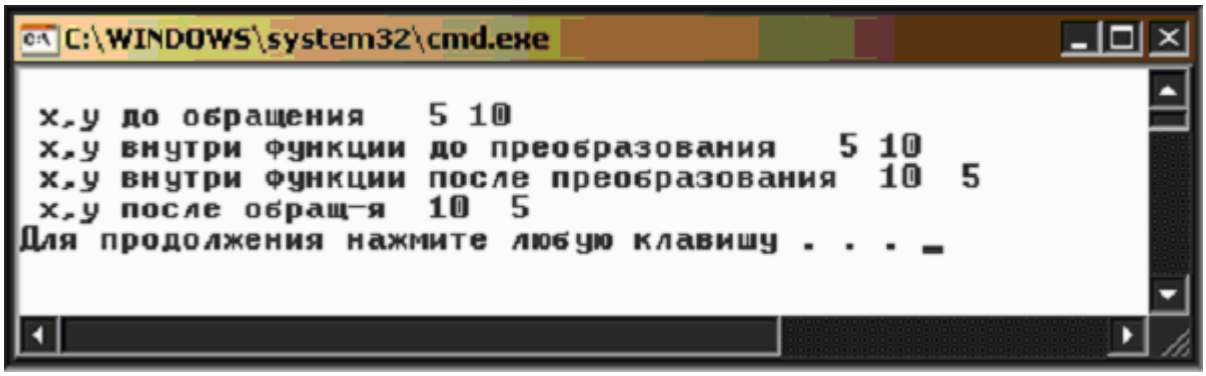


Рис. 2.29. Результат работы примера 12-2 (с указателями)

Но можно в таких случаях обойтись и просто *ссылками*. В этом случае удается скрыть от пользователя функции подробности ее реализации. И к тому же исчезает необходимость многократно возвращать значения указателей.

```
// передача параметров в функцию (по ссылке)
#include <iostream>
#include <iomanip>
using namespace std;
void swap(int &x, int &y);
int main()
{   setlocale(LC_ALL, “Russian”);
int x,y; x=5, y=10;
cout<<"\n x,y до обращения "<<setw(3)<<x<<setw(3)<<y;
      swap(x,y);
cout<<"\n x,y после обращения "<<setw(3)<<x <<setw(3)
<<y<<"\n";
return(0);}
void swap(int &rx, int &ry)
{   int z;
cout<<"\n x,y внутри функции до преобраз."<<setw(3)
<<rx<<setw(3) <<ry;
      z=rx; rx=ry; ry=z;
cout<<"\n x,y внутри функции после преобр." <<setw(3)
<<rx<<setw(3)<<ry;  }
```

В результате имеем (рис.2.30):

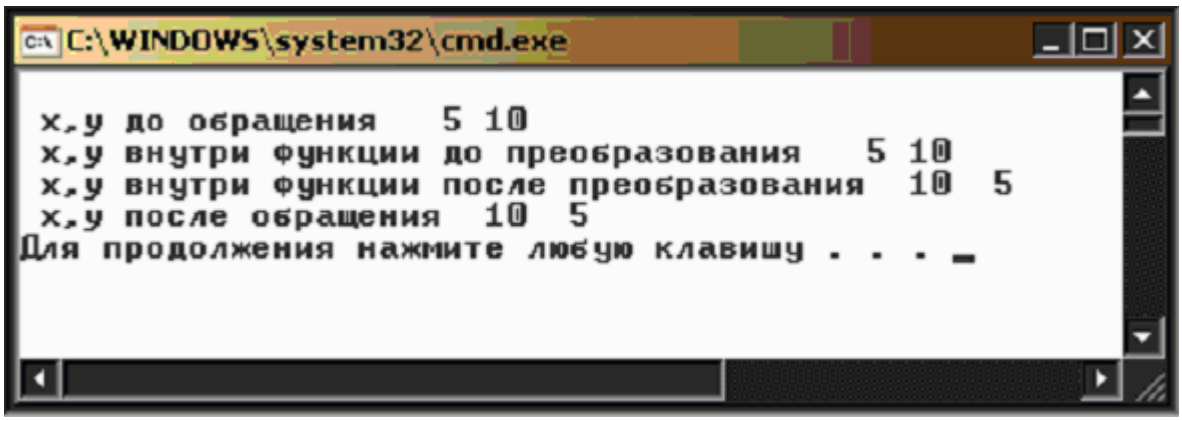


Рис. 2.30. Результат работы программы примера 12-2 (с использованием ссылок)

12.3. Указатели и динамические массивы

Если до начала разработки программы неизвестно, сколько в массиве элементов, в программе следует использовать *динамические массивы*. Память под них выделяется с помощью операции **new** в динамической области памяти во время выполнения программы. Адрес начала массива хранится в переменной, называемой указателем.

Например,

```
int n=10;
int *a = new int[n]
```

Заметим, что обнуление памяти при ее выделении не происходит. Инициализировать динамический массив нельзя. Обратится к элементу динамического массива можно как обычно `a[3]`, так и с использованием указателя `*(a+3)`. Дело в том, что в переменной указателе хранится адрес начала массива. Для получения адреса третьего элемента к этому адресу прибавляется смещение 3. Операция сложения с константой для указателей учитывает размер адресуемых элементов, т.е. на самом деле индекс умножается на длину элемента массива:

```
a + 3*sizeof(int)
```

Затем с помощью операции * (разадресации) выполняется выборка значения из указанной области памяти.

Если динамический массив в какой-то момент работы программы перестает быть нужным, и мы собираемся впоследствии использовать эту память повторно, необходимо освободить ее с помощью операции `delete[]`, например

```
delete [] a;
```

размерность массива при этом не указывается.

Пример 12-3. Рассчитать сумму элементов правее последнего отрицательного.

```
// сумма элементов правее последнего отрицательного
// элемента (с использованием динамич. массива)
#include <iostream>
using namespace std;
int main()
{ setlocale(LC_ALL, "Russian");
 int  n;
    cout<<"введи размерность массива "; cin>>n;
    float *a=new float[n];   int i, notr;
    cout<<"вводите элементы массива ";
    for (i=0; i<n; i++) cin>>a[i];
    notr=-1;
      for (i=0; i<n; i++) if (a[i]<0) notr=i;
      if (notr!=-1)
```

```
     {float sum=0;
      for (i=notr+1;i<n;i++)sum=sum+a[i];
      cout<<"\n  сумма= "<<sum;}
     else cout<<"\n отрицательных нет ";
      cout<<"\n";
return(0); }
```

Пример 12-4. Создать программу быстрой сортировки одномерного массива (с использованием указателей и динамической памяти).

Имеется один алгоритм быстрой сортировки одномерного массива, предложенный *Хоаром*. Для него время сортировки в зависимости от количества элементов имеет вид:

t ~ N*lgN + N

В отличие от стандартного метода «сортировки выбором», у которого время выполнения пропорционально:

t ~ N2 + N*lgN

Суть этого алгоритма в следующем.

Применим к массиву так называемую процедуру разделения относительно среднего элемента. Вообще-то, в качестве «среднего» можно выбрать любой элемент массива, но для наглядности мы будем выбирать действительно, по возможности, средний по своему номеру элемент.

Процедура разделения делит массив на две части. В левую помещаются элементы, меньшие, чем элемент, выбранный в качестве среднего, а в правой – бОльшие. Это достигается путем просмотра массива попеременно с обоих концов, при этом каждый элемент сравнивается с выбранным средним, и элементы, находящиеся в «неподходящей» части, меняются местами. После завершения процедуры разделения, средний элемент оказывается на своем окончательном месте.

Далее процедуру разделения необходимо повторить отдельно для левой и правой части: в каждой части выбирается среднее, относительно которого она делится на две и т.д.

Понятно, что одновременно процедура не может заниматься и левой, и правой частями, поэтому необходимо каким-то образом запомнить запрос на обработку одной из двух частей (например, правой), и заняться оставшейся частью (например, левой).

Так продолжается до тех пор, пока не окажется, что очередная обрабатываемая часть содержит ровно один элемент. Тогда нужно вернуться к последнему из необработанных запросов, применить к нему все ту же процедуру разделения и т.д. В конце концов, массив окажется полностью упорядочен.

Для хранения границ еще не упорядоченных частей массива более всего подходит так называемый **стек.**

При написании программы для быстрой сортировки стек реализуем в виде двух массивов **stackr** и **stackl** и одной переменной **sp**, используемой как указатель на вершину стека (она хранит номер последнего заполненного элемента массива).

Для этого алгоритма количество элементов в стеке не может превышать **n**, поэтому размер массивов задан равным именно этой величине.

При занесении в стек переменная **sp** увеличивается на единицу, а при выборке – уменьшается.

```
#include <iostream.h>
using namespace std;
int main()
{ setlocale(LC_ALL, “Russian”);
  const int n=20;
    float arr[n], middle,temp;
    int *stackl=new int [n], *stackr=new int [n], sp=0;
    int i,j,k, left, right;
    cout<<"Введите элементы массива:";
        for (i=0; i<n; i++) cin>>arr[i];
//сортировка
    sp=1; stackl[1]=0; stackr[1]=n-1; // 1
    while (sp>0)                              // 2
    {   //выборка из стека последнего запроса
    left=stackl[sp];                          // 3
    right=stackr[sp];                         // 4
    sp--;                                     // 5
    while (left<right)                        // 6
    { // разделение arr[left] .. arr[right]
    i=left; j=right;                          // 7
    middle= arr[(left+right)/2];              // 8
    while (i<j)                               // 9
    { while (arr[i]<middle)i++;               // 10
       while (middle<arr[j]) j--;     // 11
     if (i<=j)
        { temp=arr[i]; arr[i]=arr[j]; arr[j]=temp;
          i++; j--; }
         }cout<<"\n";
    if (i<right)                         // 12
    {// запись в стек запроса из правой части
    sp++;
    stackl[sp]=i; stackr[sp]=right; }
```

```
right=j;                                  // 13
// теперь left и right ограничивают левую часть
}    }
//вывод результата
for (i=0;i<n; i++) cout<< arr[i]<<" ";
cout<< endl;
    return(0);  }
```

На каждом шаге сортируется один фрагмент массива. Левая граница фрагмента хранится в переменной **left**, правая – в переменной **right**. Сначала фрагмент устанавливается размером в массив целиком (строка 1). В операторе 8 выбирается средний элемент фрагмента массива.

Для продвижения по массиву слева направо в цикле используется переменная i, справа налево – j (в цикле 11). Их начальные значения устанавливаются в операторе 7. После того, как оба счетчика сойдутся где-то в средней части массива, происходит выход из цикла 9 на оператор 12, в котором в стек заносятся границы правой части фрагмента. В операторе 13 устанавливаются новые границы левой части для сортировки на следующем шаге.

Если сортируемый фрагмент уже настолько мал, что сортировать его не требуется, происходит выход из цикла 6, после чего выполняется выборка из стека границ еще не отсортированного фрагмента (операторы 3, 4). Если стек пуст, происходит выход из главного цикла 2. Массив отсортирован.

Пример 12-5. Создать программу быстрой сортировки одномерного массива с использованием рекурсии.

Вышеописанный алгоритм Хоара можно реализовать с помощью рекурсии, если оформить сам алгоритм в виде отдельной функции. Приведем текст программы с реализацией алгоритма Хоара в виде рекурсивной функции `qsort`.

```
//быстрая сортировка с рекурсией
#include <iostream.h>
using namespace std;
//прототип функции сортировки
void qsort(float* array, int left, int right);
int main()
{ setlocale(LC_ALL, "Russian");
const int n=20;
    float arr[n];
    int i,l,r;   randomize();
    cout<<"Вводите элементы массива ";
```

```
    for (i=0; i<n; i++)   cin>>arr[i];
    l=0; r=n-1; //лев. и прав. границы нач.фрагмента
    qsort(arr,l, r);  cout<<"\n";
    for (i=0; i<n;i++) cout<< arr[i]<<" ";
return(0);
}
 void qsort(float* array, int left, int right)
  { int i=left, j=right;
    float middle=array[(left+right)/2];
    float temp;
    while (i<j)
    { while (array[i]< middle) i++;
      while (middle < array [j]) j--;
      if (i<=j)
      { temp= array[i];
        array[i]=array[j];;
    array[j]= temp;
    i++;
    j--;
      }
    }
    if (left<j) qsort(array, left,j);      // 2
    if (i< right) qsort(array, i, right); // 3
   }
```

12.4. Указатели и перегрузка операций

Если мы вводим данные нового типа (таковым является, например, перечислимый тип), то желательно сделать так, чтобы стандартные операции для них имели тот же внешний вид. Такого рода переобозначение операций (для новых типов данных) называют *перегрузкой операций*. Для уяснения этого вспомните также, как мы ввели перегрузку функций (см. раздел 10.1). Общая форма для перегрузки операций имеет вид:

***тип* operator@ (*параметр*)**

Здесь ***тип*** – нужный тип переменной, **operator** – служебное слово, **@** – символическое обозначение нужной операции.

Указатели могут быть использованы и для *перегрузки операций*. Вспомним, пример 11-1, где мы не смогли использовать операцию инкремента. Достаточно ввести перегрузку для операции ‘++’ для перечислимого типа и проблема исчезнет. Такая перегрузка имеет вид:

```
enum dni{Monday=1, Tuesday, Wednesday, Thursday, Fri-
day, Saturday, Sunday} ;
dni operator++(dni &m)
{ return (m=dni(m+1));}
```

Тогда окончательно текст программы примера 11-1 примет вид:

```
// перечислимый тип - часы рабочей недели
#include <iostream>
using namespace std;
float t;
enum dni{Monday=1, Tuesday, Wednesday, Thursday, Fri-
day, Saturday, Sunday} ;
dni operator++(dni &m)
{ return (m=dni(m+1));}
int main()
{ setlocale(LC_ALL, “Russian”);
    dni d;
    t=0;
    for (d=Monday; d<=Friday; d++)
    { if (d == Friday) t=t+8; else t=t+8.25; } ;
   cout << " сумма часов раб.недели="<<t<<"\n";
return(0); }
```

Контрольные вопросы

1. Дайте понятие об указателях?
2. Как указатели используются при обращении к функциям?
3. Что такое динамические массивы? Как они описываются?
4. Как указатели используют для перегрузки операций?

13. ОБРАБОТКА СИМВОЛЬНЫХ СТРОК

13.1. Символьные переменные

Напомним, что для описания символьных переменных служит тип `char`, соответственно массив символов должен при описании рядом с именем иметь в квадратных скобках указание на размер.

Но для начала рассмотрим работу просто с отдельными символами. Значение символьной переменной может быть задано как с помощью команды присваивания (обратите внимание, что отдельный символ заключается в апострофы), так и с помощью команды ввода `cin`.

Поскольку мы уже освоили работу с функциями и указателями, то для организации вывода русских сообщений можем разработать отдельную функцию символьного ссылочного типа

```
char* Rus(const char* str)
{    AnsiToOem(str,str_rus);
     return str_rus;}
```

У этой функции на входе задается символьная константа `str` (тот самый русский текст), которая внутри преобразуется с помощью той самой функции `AnsiToOem` и преобразованный текст помещается в `str_rus`. Благодаря этому, выводить русский текст в основной программе можно просто вызовом этой функции (см. пример 13-1)

Пример 13-1. Среди последовательно вводимых *N* символов подсчитать число восклицательных знаков.

```
// подсчет числа символов !
#include <iostream>
#include <windows.h>
using namespace std;
int main()
{    setlocale(LC_ALL, "Russian");
     char c,c1; int sum=0,i,n;
     c1='!'; cout<<" \n введи значение N \n";
        cin>>n;
     for (i=1; i<=n; i++)
        {cout<<" \n введи очередной символ "; cin>>c;
          if (c==c1) sum++; }
     cout<<" количество знаков "<<c1;
     cout<<" равно "<<sum<<"\n";
   return (0); }
```

Символы в C++ также имеют коды. Для получения цифрового кода символа достаточно применить операцию перевода в целые: `(int) c`, где `c` – символьная переменная. Для получения по коду самого символа достаточно применить операцию по переводу целых в символьные `(char) k`, где `k` – целое число (код символа).

В рассмотренном ниже примере будем также использовать датчик случайных чисел, который мы уже использовали в разделе 8. Напомним, что на языке C++ он реализуется функцией `rand()%RAND_MAX` – дает случайное целое число из интервала от `0` до положительного целого `RAND_MAX-1` (в качестве `RAND_MAX` можно указать сразу конкретное число). Кроме того, для запуска этого датчика случайных чисел (чтобы получать новый набор случайных чисел при каждом новом запуске программы), используют функцию `srand(time(0))`, т.е. для старта генератора случайных чисел будет использоваться системное время компьютера. Причем это время следует преобразовать в беззнаковое целое (`unsigned`

int). Кроме того, требуется подключить заголовочный файл <stdlib.h>, содержащий эти функции. Для получения системного времени нужно подключить заголовочный файл <time.h>

Наиболее употребительные символы имеют коды в интервале от 33 до 122, поэтому в программе предусмотрено дополнительно ограничение на получение случайных чисел из интервала от 33 до 123.

Здесь мы также впервые использует команду goto *метка*. Метка в C++ задается как и любой идентификатор (название метки начинается с буквы) и не требует описания (в отличие от Паскаля). Признаком метки является двоеточие сразу после нее.

Пример 13-2. Составить программу «клавиатурный тренажер» (случайным образом выводится символ, который следует нажать на клавиатуре).

```
//клавиатурный тренажер(распознание символа по коду)
#include <iostream>
#include <stdlib.h>
#include <time.h>
using namespace std;
int kod,n, r_max, r_min; char c,d;
int main ()
{srand( (unsigned)time(0) );
One:
r_min=33; r_max=123;
kod=(double)rand()/(RAND_MAX+1)*(r_max-r_min)+r_min;
c=(char)kod; cout<<" нажми "<<c<<"\n";
cin>>d;
if ((int)d==kod) cout<<" правильно \n ";
       else cout<<" нет \n";
goto One;
return (0);}
```

Данная программа будет работать непрерывно («зациклили» оператором перехода goto One). Чтобы прервать ее работу достаточно нажать Ctrl-C.

Замечание. Коды символов *кириллицы* задаются в C++ отрицательными числами!

Для демонстрации кодов русских букв рекомендуем набрать следующую программку (см. пример 13-3).

13.2. Символьные строки (как массивы символов)

К сожалению, в C++ отсутствует тип «строка» (хотя для создания строк здесь имеются более мощные средства). Мы сначала будем рассматривать строки как массивы символов.

Пример 13-3. Распечатать коды русских букв (предварительно записав их в виде массива символов).

```
// вывод кодов русских букв
#include <iostream>
#include <iomanip>
#include <windows.h>
using namespace std;
char str_rus[256],salf[65],sal;
int main()
{  setlocale(LC_ALL, "Russian");
int i;   char rusalf[65]=
    "АаБбВвГгДдЕеЁёЖжЗзИиЙйКкЛлМмНнОоПпРрСсТтУуФфХхЦц
ЧчШшЩщЪыЬьЭэЮюЯя";
    cout<< " \n КОДЫ РУССКИХ БУКВ "<<"\n";
    AnsiToOem(rusalf, salf);
    for (i=0; i<=64; i++)
// для вывода  ровными столбиками по 8 применяется if
   {sal=salf[i];
       cout<<setw(5)<<sal<<setw(3)<<(int)rusalf[i];
        if (i%8 == 7)cout<<"\n";}
return (0);}
```

При запуске программки получим (рис.2.31):

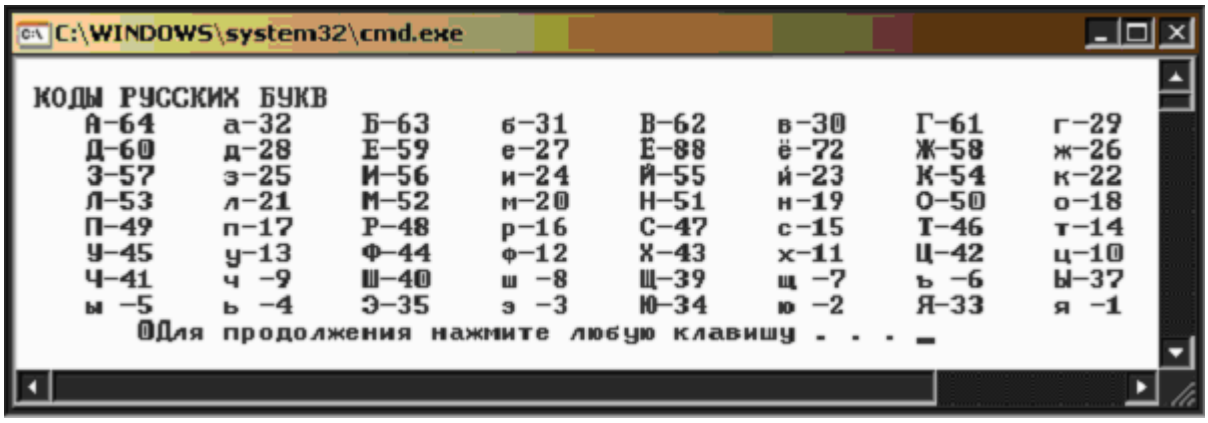


Рис. 2.31. Результат работы программы примера 13-3

В рассмотренном примере строка текста (в виде массива символов) заполнялась внутри программки путем присваивания. Значительно важнее было бы научиться *вводить значение строки извне* с помощью той же команды `cin`. К сожалению, если при вводе в строке будет пробел, то `cin` воспримет его как конец строки и остановит запись в буфер. Кроме того, если пользователь введет больше символов, чем описано, то оператор `cin` отбросит последние символы. Для преодоления этого организуют так называемый *метод* `cin.get()`. В нем указывается строка для ввода и ее максимальная длина. В данном случае указываем 79, хотя строка описана на 80 символов потому, что символы на самом деле нумеруются с 0 и последний символ массива всегда «\0» (null).

Пример 13-4. Введенную строку текста разбить на отдельные слова, т.е. вывести каждое слово в отдельную строку.

```
// ввод строки текста и разбиение на слова
#include <iostream>
#include <string>
using namespace std;
int main()
{setlocale(LC_ALL, "Russian");
 char s[80];int n,i;
 cout <<"\n введи строку текста \n";
 cin.get(s,79);
 n=strlen(s);cout<<"\n текст разбит на слова:\n";
 for (i=0; i<n; i++)
 { if (s[i] ==' ') cout<<"\n"; else cout<<s[i]; }
 cout<<"\n";
return (0);  }
```

Здесь мы использовали специальную функцию `strlen(s)`, которая позволяет получить реальную длину строки `s` (сколько именно символов было введено).

13.3. Обработка массивов строк

В некоторых случаях для ввода строк правильнее использовать метод `cin.getline(s,n)`. Отличия его от метода `cin.get(s,n)` в следующем:

- Метод **getline** считывает из входного потока n-1 символов или менее (если символ перевода строки встретится раньше) и записывает их в строковую переменную S.

• Символ перевода строки (\n встречается, когда нажимает Enter) также считывается (удаляется) из входного потока, но не записывается в строковую переменную, вместо него размещается завершающий 0.

• Если в строке исходных данных более n-1 символов. Следующий ввод будет выполняться из той же строки, начиная с первого несчитанного символа.

• Метод **get** работает аналогично, но оставляет в потоке символ перевода строки. В строковую переменную добавляется завершающий 0.

Для обработки строк существует ряд специальных функций:

• функция **strcpy**(s1,s2) – она копирует в строку S1 содержимое строки S2

• функция **strcmp**(s1, s2) – она сравнивает содержимое строк s1 и s2: если они равны, то функция дает 0, если s1<s2, то отрицательное число и если s1>s2, то положительное

• функция **_itoa**(I,s,r) – преобразует целое I в строку s, пользуясь системой счисления с основанием r

Все эти функции находятся в заголовочном файле **string.h.**

Пример 13-5. Дан список из 10 фамилий. Найти однофамильцев и, если они есть, указать их порядковые номера в списке.

```
// поиск однофамильцев в списке из 10 фамилий
#include <iostream>
#include <string.h>
#include <stdlib.h>
using namespace std;
 const int n=10; int i,j,k;
char fam [n+1] [20], s[20],s2[20],s3[20];
int main()
{setlocale(LC_ALL, "Russian");
for (i=1;i<=n;i++)
 { cout<<"введи фамилию № "<<i<<" ";
   cin.getline(s,19); strcpy(fam[i],s);}  k=0;
 for (i=1;i<n;i++)
 {  strcpy(s,fam[i]);
  for (j=i+1;j<=n;j++)
  {  strcpy(s2,fam[j]);  if (strcmp(s, s2)==0)
{  cout<<s;  k++; _itoa(k,s3,2); strcpy(fam[j],s3);
//затирание уже найденных однофамильцев
      cout<<" "<<i<<" "<<j;cout<<endl;  } }}
return(0);}
```

Результат работы программы на рис. 2.32.

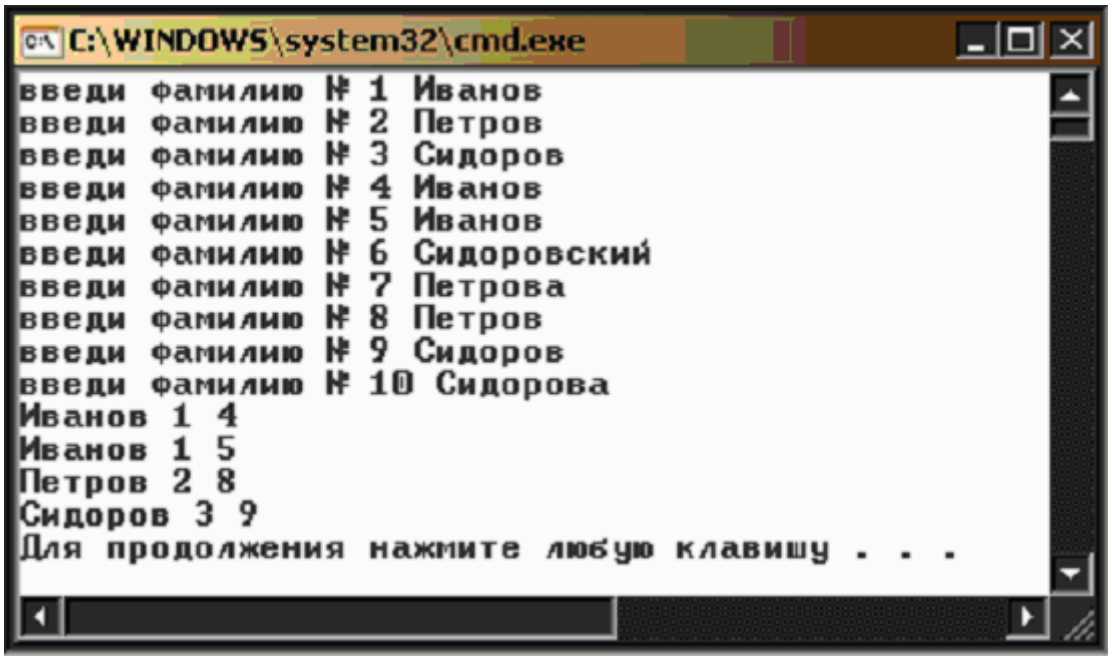


Рис. 2.32. Результат работы программы примера 13-5

Рассмотрим, как реализуется задача сортировки в массиве строк.

Пример 13-6. Дан список из 10 фамилий. Упорядочить их по алфавиту.

```
// упорядочение фамилий в списке из 10 фамилий
#include <iostream>
#include <string.h>
#include <stdlib.h>
#include <windows.h>
using namespace std;
const int n=10; int i,j, k; char fam [n+1] [20],
s[20],s2[20],s3[20];
int main()
{ setlocale(LC_ALL, “Russian”);
for (i=1;i<=n;i++)
{cout<<"введи фамилию №"<<i<<" ";
 cin.getline(s,19);
 strcpy(fam[i],s);
 }  k=0;
 for (i=1;i<n;i++)
 { for (j=1;j<n;j++)
   {  strcpy(s,fam[j]);
    strcpy(s2,fam[j+1]);
    if (strcmp(s, s2)>0)
    { strcpy(s3,fam[j]);
     strcpy(fam[j],fam[j+1]);strcpy(fam[j+1],s3);
    } } }
  cout<<"\n";
```

```
  for (i=1;i<=n;i++)
{cout<<"теперь фамилия №"<<i<<" "<<fam[i]<<endl;}
return(0); }
```

Результат работы на рис. 2.33.

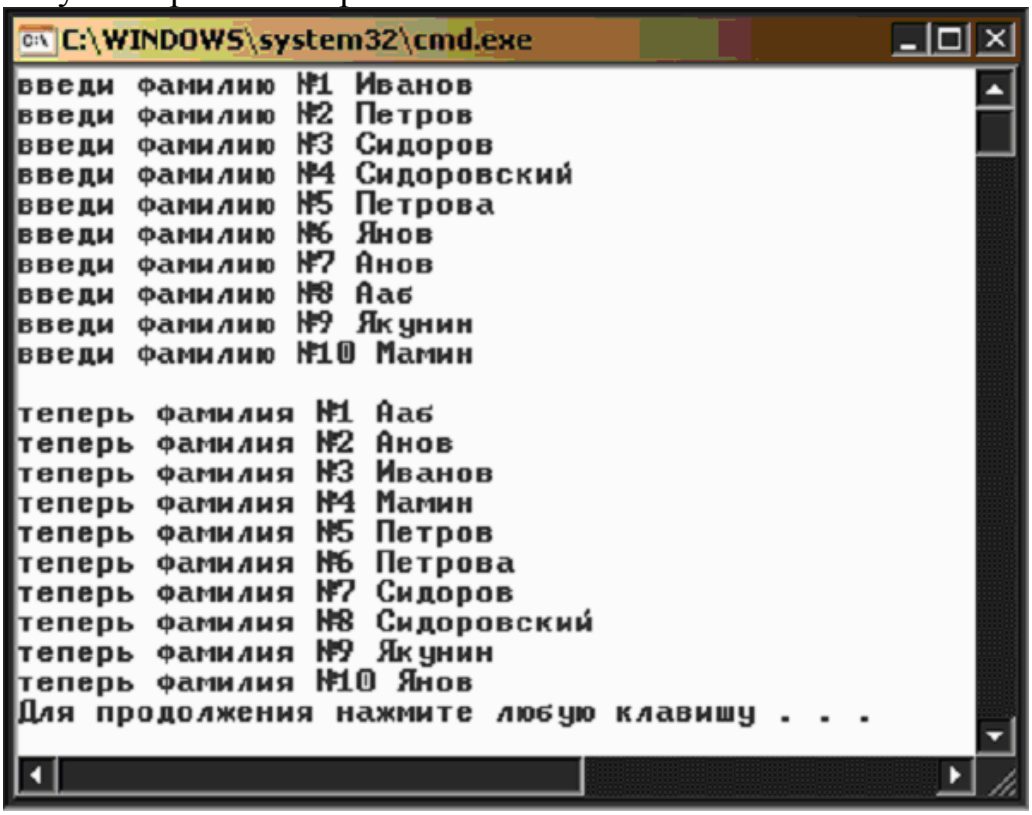


Рис. 2.33. Результат работы программы примера 13-6

Пример 13-7. Дан список из 10 фамилий. Упорядочить их по алфавиту, используя динамические массивы.

```
//алфавитное упорядочение фамилий с использованием
// динамического массива
#include <iostream>
#include <string.h>
#include <windows.h>
using namespace std;
int main()
{ setlocale(LC_ALL, "Russian");
int  i,j,n,m; char s[20],s2[20], s3[20];
    cout<<"введите количество фамилий "; cin>>n;
//объявление переменной типа "указатель на указатель" и
// выделение памяти под массив указателей на число фамилий
    char **fam = new char* [n+1];
    cout<<"введи макс. длину фамилии "; cin>>m;
// ниже - цикл для выделения памяти под каждую фамилию
// на максимально указанную длину фамилии
```

```
    for (i=1;i<=n;i++) fam[i]= new char[m];
//цикл ввода фамилий
    for (i=1; i<=n; i++) { cout<<"введи "<<i;
    cout<<"-ю фамилию "; cin>>fam[i];}
// упорядочение фамилий путем сравнения соседних
// элементов и перестановки их
    for (i=1; i<n; i++)  for (j=1; j<n; j++)
    { if (strcmp(fam[j],fam[j+1])>0)
        {strcpy(s3,fam[j]); strcpy(fam[j],fam[j+1]);
     strcpy(fam[j+1],s3);}    }
for (i=1;i<=n; i++)
cout<<"теперь фам №"<<i<<" "<<fam[i]<<endl;
return(0); }
```

Результат работы на рис. 2.34.

Контрольные вопросы

1. Как описываются символьные переменные и массивы на Visual C++?
2. В чем особенность кодировки букв кириллицы?
3. Как обрабатывают массивы строк на Visual C++? Перечислите встроенные функции, используемые для такой обработки.

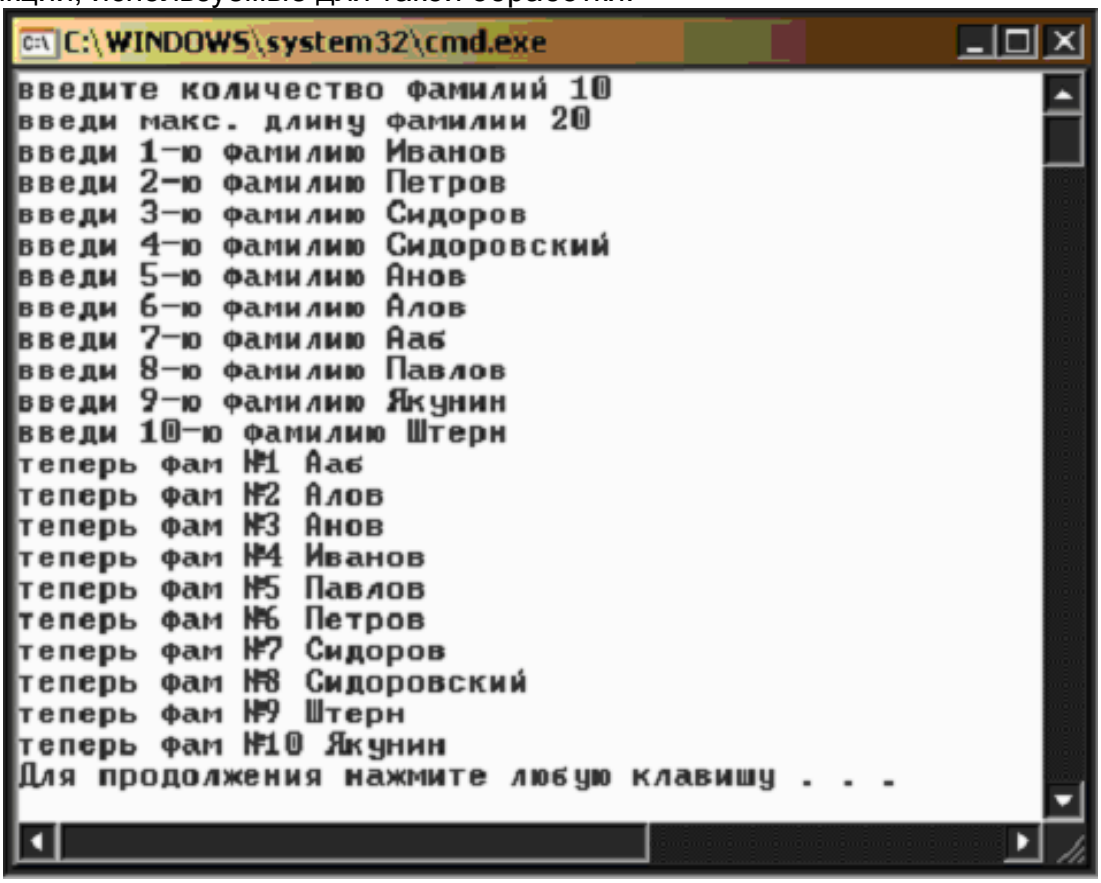


Рис. 2.34. Результат работы программы примера 13-7

14. СТРУКТУРЫ

Одна из главных особенностей языка C++ – это возможность создавать данные новых типов. Причем, данные нового типа могут быть *комбинацией данных различного типа*. Такие данные называют *структурами*.

Описание структуры делается так**:**

Struct ***имя структуры***
{ ***тип и имя элемента структуры*** **1;**
тип и имя элемента структуры **2;**
…
тип и имя элемента структуры **2**};

А затем новый структурный тип, как обычно, можно использовать для объявления новых переменных. Доступ к отдельным элементам структуры осуществляется операцией. (точка) для объекта и операцией –> (стрелка) для указателя на объект.

Например, мы создадим структуру типа **date**:

```
struct  date
{int day;
 enum muns
{jan,feb,mar,apr,may,jun,jul,aug,sep,oct,nov,dec};
 int year; }
```

И затем объявим переменную **den** типа **date**. Тогда элементы такой структурной переменной будут обозначаться: **den.day, den.muns, den.year**.

Пример 14-1. Даны два класса учащихся. Определить, являются ли они параллельными.

```
#include <iostream>
#include <iomanip>
#include <windows.h>
using namespace std;
int main()
{ setlocale(LC_ALL, "Russian");
 struct clas {int god;
 char bukwa;};
 clas x,y;
 cout<<"\n введи номера классов ";
 cin>>x.god>>y.god;
 cout<<"\n введи буквы классов ";
 cin>>x.bukwa>>y.bukwa;
  if ((x.god == y.god) && (x.bukwa != y.bukwa))
    cout <<" ЯВЛЯЕТСЯ ПАРАЛЛЕЛЬНЫМИ ";
       else cout <<" не параллельны " ;
    cout<<"\n"; return 0;}
```

Заметим, что в качестве элемента структуры может быть в свою очередь структура. Кроме того, из структур можно формировать и массивы.

Пример 14-2. Список учащихся состоит из 10 записей. Каждая запись содержит фамилию и дату рождения ученика. Найти тех, кто родился в один и тот же день (год рождения может и отличаться).

```
// ученики, родившиеся в один день
#include <iostream>
#include <windows.h>
using namespace std;
struct date {int day;
        int muns;
        int year;};
struct child {char fam[15];
        date den;    };
int main()
{ setlocale(LC_ALL, “Russian”);
child spis[11]; const int n=10; int i,k;
for (i=1; i<=n; i++)
{ cout<<"\n введи фамилию ученика "<<i<<' ';
cin>> spis[i].fam;
cout<<"введи день,месяц и год рождения ученика"<<i<<" ";
cin>>spis[i].den.day>>spis[i].den.muns>>spis[i].den.year;};
cout<<"\n РОДИЛИСЬ В ОДИН ДЕНЬ \n";
for (i=1; i<=n-1; i++)
for (k=i+1; k<=n; k++)
if  ((spis[i].den.day==spis[k].den.day) &&
     (spis[i].den.muns==spis[k].den.muns))
{cout<<spis[i].den.day<<" "<<spis[i].den.muns<<" ";
 cout<<spis[i].fam<<"  "<< spis[k].fam; cout<<"\n";}
return 0; }
```

При запуске получим примерно такой результат (рис. 2.35).

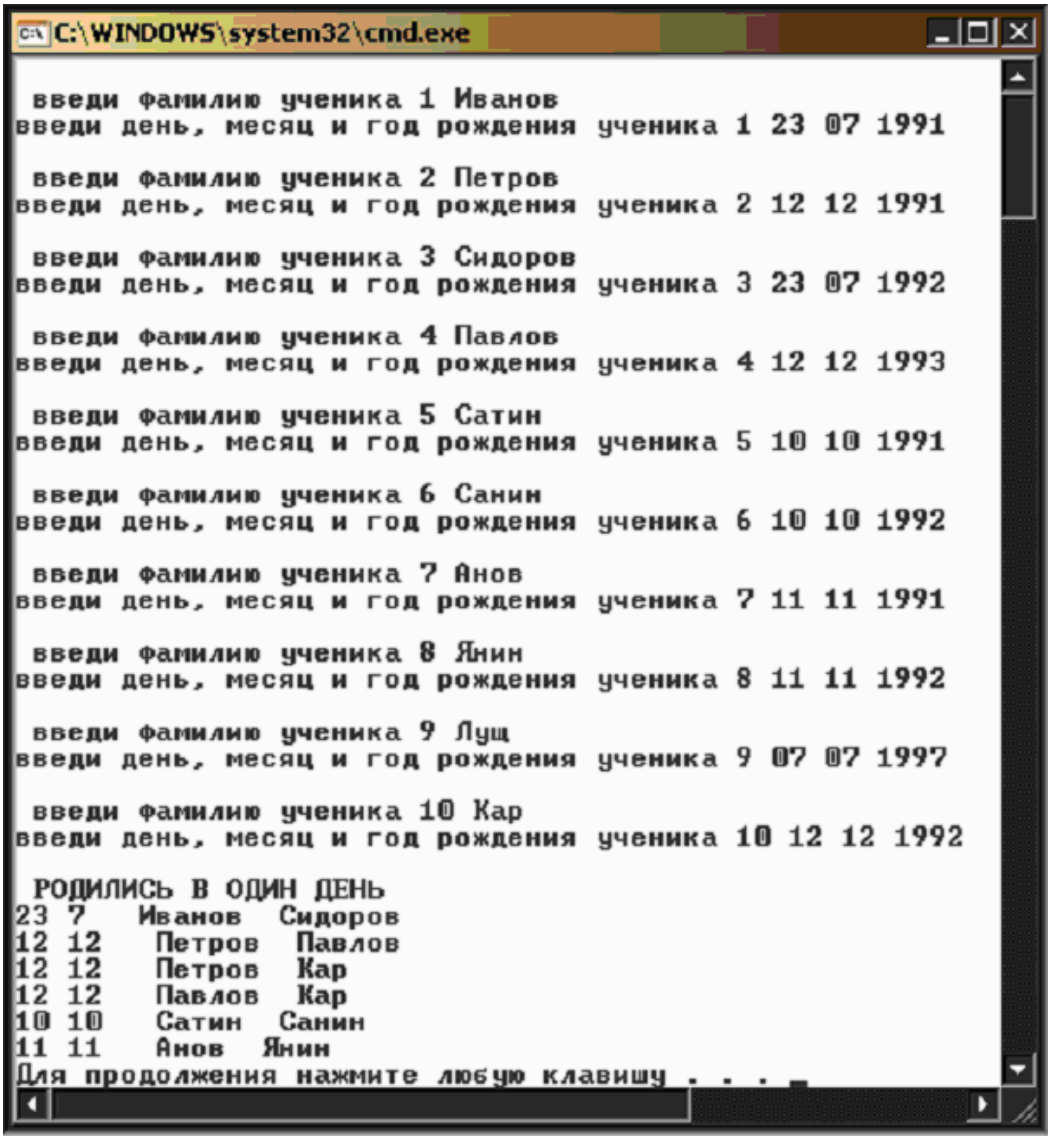


Рис. 2.35. Результат работы программы примера 14.2

В качестве одного из элементов структуры может быть *массив*. Это продемонстрируем на следующем примере. Кроме того, над структурными переменными можно выполнять операции как над едиными объектами (присваивать одной структурной переменной другую, сравнивать их и т.д.), но только в том случае, если структурные переменные однотипны, т.е. описаны на основе одного структурного типа.

Пример 14-3. Список учащихся состоит из 10 записей. Каждая запись содержит фамилию ученика, класс и поля – оценки по семи предметам. Вывести фамилии и класс тех учеников, кто имеет пятерки по всем предметам и сформировать из них отдельный список – список отличников.

```
// поиск отличников в списке
#include <iostream>
#include <windows.h>
```

```
using namespace std;
    struct clas{ int god;
                 char bukwa;};
     struct chil {char fam[15];
                   clas cl;
                   int ocenky[8];};
    chil spis[11], spisotl[11];  int k,s,i,j;
int main()
{setlocale(LC_ALL, "Russian");
 for (i=1; i<=10; i++)
{cout<<Rus("\n введи фамилию ученика, номер и букву его
класса");
 cin>>spis[i].fam >> spis[i].cl.god >>
spis[i].cl.bukwa;
 for (j=1; j<=7; j++)
 {cout<<"введи оценку "<<j<<" ";
  cin>>spis[i].ocenky[j];};  cout<<"i="<<i; };
k=0;
for (i=1; i<=10; i++)
{ s=0;
  for (j=1; j<=7; j++)
    if (spis[i].ocenky[j]==5)  s=s+1;
  if (s==7) { k=k+1; spisotl[k]=spis[i]; } };
cout<<"\n СПИСОК ОТЛИЧНИКОВ ";
for (i=1; i<=k; i++)
{cout<<"\n"<<spisotl[i].fam<<" " <<spisotl[i].cl.god
<<" " <<spisotl[i].cl.bukwa;}
return 0;}
```

15. КЛАССЫ

15.1. Понятие класса

Фундаментальное нововведение языка C++ – понятие *класс*. Если структура объединяет разнородные по типам переменные, то ***класс*** – *объединяет переменные и функции, предназначенные для работы с ними*. Здесь, пожалуй, впервые мы сталкиваемся с элементами объектно-ориентированного программирования. Можно сказать, что класс – это нового типа объект, который имеет *свойства* (переменные-члены класса) и *методы* (функции-члены класса). Определив класс, мы потом можем, как обычно, объявить определенный *экземпляр*, т.е. описать объект на основе заданного класса. Объявление класса производится с помощью служебного

слова **class** с указанием имени и перечислением его членов в фигурных скобках. Например,

```
class DayOfYear
    {public:
      void output();// прототип функции-члена
                int month;
      int day; };
```

Обратите внимание, что при описании указывается только прототип функции, а саму функцию надо затем описать с указанием имени класса и через :: имя самой функции :

```
        void dayOfYear :: output()
        {cout << « месяц « << month;
        cout << « день « << day << endl;  }
```

Впрочем, допускается описание функции-члена тут же, при описании класса.

Обращение в программе к членам класса делается также через точку (подобно тому, как это делалось для элементов структуры).

Отметим, что члены класса могут быть *открытыми* (public) и *закрытыми* (private). При этом закрытые члены могут использоваться только другими элементами класса, а к открытым элементам класса разрешен доступ из программы через объекты соответствующего типа (так же, как к элементам структуры – через точку). При попытке вызвать из программы закрытый элемент класса, компилятор укажет на ошибку. Заметим, что по умолчанию члены класса являются закрытыми (private).

Пример 15-1. Создать программу, которая запрашивает у пользователя сегодняшнее число и дату дня рождения, а затем сообщает: «Поздравляем с днем рождения!», если дата совпадает с днем рождения или просто сообщает «Счастливого дня!», если сегодня не ваш день рождения.

```
#include <iostream>
#include <windows.h>
using namespace std;
class DayOfYear
    {  public:
    void output(); // прототип функции-члена
    int month;
    int day; };
int main()
{  setlocale(LC_ALL, "Russian");
    DayOfYear today, birthday;
   cout <<" Введите текущую дату: \n";
   cout << " ВВЕДИТЕ МЕСЯЦ: ";
   cin >> today.month;
```

```
   cout << " ВВЕДИТЕ ДЕНЬ МЕСЯЦА: ";
   cin >> today.day;
   cout << " ВВЕДИТЕ ВАШ ДЕНЬ РОЖДЕНИЯ: \n";
   cout << " ВВЕДИТЕ МЕСЯЦ: ";
   cin >> birthday.month;
   cout << " ВВЕДИТЕ ДЕНЬ МЕСЯЦА: ";
   cin >> birthday.day;
   cout << "СЕГОДНЯ ";
   today.output();
   cout << "ВАШ ДЕНЬ РОЖДЕНИЯ  ";
   birthday.output();
   if ((today.month = = birthday.month)
    && (today.day == birthday.day) )
      cout << "ПОЗРАВЛЯЕМ С ДНЕМ РОЖДЕНИЯ ! \n";
      else cout << "ЖЕЛАЕМ УДАЧНОГО ДНЯ \n";
   return 0; }
void DayOfYear :: output()
    {   cout << " МЕСЯЦ " << month;
        cout << " ДЕНЬ " << day << endl;  }
```

15.2. Открытые и закрытые члены класса

Рассмотренный пример реализован с открытыми членами класса. Закрытые члены могут понадобиться, когда мы захотим создать более универсальный класс – такой, чтобы при изменении описания элементов класса, изменения в основной программе не вносились. Например, если мы изменим описание элемента **month** с целого **int** на строковый **char(3)** – месяц обозначается тремя буквами – тогда переменная-член **month** исчезнет и нужно менять, по сути, всю программу. После объявления переменной-члена закрытой, не существует другого способа изменить ее значения (или обратиться к нему), кроме использования одной из предназначенных для этого функций-членов.
Реализуем тот же пример, но с использованием закрытых членов.

Пример 15-2. Создать программу, которая запрашивает у пользователя сегодняшнее число, а затем сообщает: «С ДНЕМ РОЖДЕНИЯ, Иоганн Себастьян!», если дата совпадает с днем рождения И.С. Баха или просто сообщает «Желаем удачного дня, Иоганн Себастьян», если сегодня не день рождения И.С. Баха.

```
// пример с закрытыми членами класса
#include <iostream>
#include <windows.h>
```

```
using namespace std;
class DayOfYear
    { public:
      void input();
      void output();
      void set(int new_month, int new_day);
      int get_month(); //возвращает значение месяца
      int get_day(); //возвращает день месяца
      private:
        int month;
        int day;
    };
int main()
{  setlocale(LC_ALL, "Russian");
    DayOfYear today, bach_birthday;
    cout << Rus(" Введите текущую дату: \n");
    today.input();
cout << "СЕГОДНЯ ";
today.output();
bach_birthday.set(3,21);
cout <<"День рождения И.С. Баха ";
bach_birthday.output();
  if (today.get_month() == bach_birthday.get_month()
    && today.get_day() == bach_birthday.get_day() )
cout <<"С ДНЕМ РОЖДЕНИЯ, Иоганн Себастьян ! \n";
else cout<<"Желаем удачного дня,Иоганн Себастьян \n";
return 0;}
   void DayOfYear :: input()
{  cout << " ВВЕДИТЕ МЕСЯЦ: ";   cin >> month;
   cout << " ВВЕДИТЕ ДЕНЬ МЕСЯЦА: ";
   cin >> day;}
   void DayOfYear :: output()
    {   cout << " МЕСЯЦ " << month;
        cout << " ДЕНЬ " << day <<"\n";   }
void DayOfYear :: set(int new_month, int new_day)
{   month=new_month;
    day=new_day;}
int DayOfYear ::get_month()
{   return month; }
int DayOfYear ::get_day()
{   return day; }
```

При запуске программы и вводе месяца и дня рождения И.С. Баха получим (рис. 2.36):

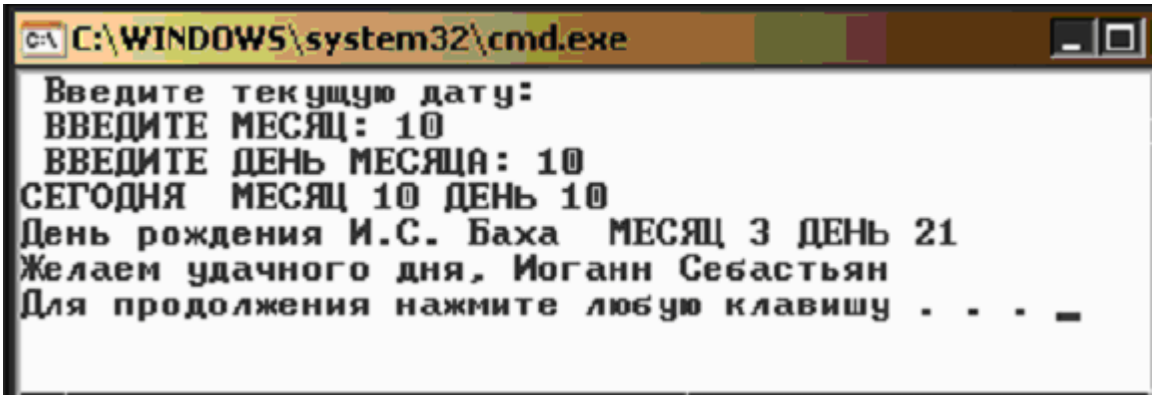


Рис. 2.36. Результат работы программы примера 15.2

Рассмотрим пример с использованием указателей.

Пример 15-3. Создать программу для расчета суммы двух чисел

```
#include <iostream>
#include <windows.h>
using namespace std;
class Sum
{  private:
     int x,y,s;
   public:
      void getx(int x1){x=x1;}
      void gety(int y1){y=y1;}
      void summa();   };
int main()
{ setlocale(LC_ALL, "Russian");
    Sum z,*b=&z;
   int x2,y2;
   cout <<"\n Введи 1-е слагаемое:";
   cin >>x2;
   cout <<"\n Введи 2-е слагаемое:";
   cin >>y2;
   z.getx(x2);
   z.gety(y2);
   b->summa();
   cout<<"\n";
   return 0; }
void Sum :: summa()
{   s=x+y;
    cout<<"\n сумма "<<x;
    cout<<" и " <<y;
    cout<<" равна:"<<s; }
```

15.3. Конструкторы и деструкторы

Объявление класса не производит выделения памяти. Это происходит потом, когда мы, на основе объявленного класса, проводим описание переменной.

Поэтому при объявлении класса нельзя сразу производить инициализацию (присваивание данных) для объекта. Но если в определение класса включить функцию-член специального вида, известную как *конструктор*, то она автоматически вызывается при объявлении объекта данного класса. Конструктор определяется так же, как и любая другая функция-член; существует лишь два отличия:

1) имя конструктора должно совпадать с именем класса;

2) определение конструктора не может иметь возвращаемого значения. Более того, в начале прототипа функции или в ее заголовке не должно присутствовать никакого типа, даже `void`.

Например, для класса `Sum` описание конструктора может выглядеть так:

```
Sum (int x2=0, int y2)
{    x=x2;
     y=y2;  }
```

Далее конструктор `Sum()` можно вызвать двумя способами: `Sum A=Sum(1,2)` и `Sum A(1,2)`. В обоих случаях создается объект `A`, элементы данных `x` и `y` которого получают начальные значения 1 и 2 соответственно. А можно было и так: `Sum A(,2)`, т.к. первый параметр задан сразу при описании конструктора (`x2=0`). В одном классе может быть несколько конструкторов.

Внутри класса могут быть и так называемые *деструкторы*. Деструкторы уничтожают объекты класса и освобождают занимаемую ими память. Деструктор представляет собой метод с именем, совпадающий с именем класса, перед которым стоит символ ~ (тильда). Деструктор не должен иметь ни параметров, ни типа возвращаемого значения. Например, для класса `Sum` деструктор имеет вид: `~Sum () {}`.

Пример 15-4. Создать программу для расчета произведения вида $s=a \bullet b+c \bullet k+a \bullet c$.

```
// применение конструктора и деструктора
#include <iostream>
#include <windows.h>
using namespace std;
class  Pro
```

```
{   private:
        int x,y,z;
    public:
   // прототипы методов
    Pro(int,int); // конструктор
      int putx(); //доступ к x
      int puty(); //доступ к y
      int putz(); //доступ к z
      void proizv(); // произведение
      ~Pro(); // деструктор
};
// описание методов
Pro::Pro(int x1,int y1){x=x1; y=y1;}
int Pro::putx(){return x;}
int Pro::puty(){return y;}
int Pro::putz(){return z;}
void Pro::proizv(){z=x*y;}
Pro::~Pro(){}
int main()
{setlocale(LC_ALL, “Russian”);
 int s,a,b,c,k;
 cout << "\n введите a,b,c,k \n";
 cin>>a>>b>>c>>k;
  Pro D=Pro(a,b); //создание и инициализация объекта D
  Pro E(c,k);     //создание и инициализация объекта E
  Pro F(a,c);     //создание и инициализация объекта F
  D.proizv();     // получение произведения a*b
  E.proizv();     // получение произведения c*k
  F.proizv();     // получение произведения a*c
   cout <<"\n D.a="<<D.putx();
   cout <<"\t D.b="<<D.puty();
 cout <<"\t D.z="<<D.putz();
 s=D.putz()+E.putz()+F.putz();
 cout<<"\n  s="<<s<<"\n";
   F.Pro::~Pro(); // уничтожение объекта F
    E.Pro::~Pro(); // уничтожение объекта E
     D.Pro::~Pro(); // уничтожение объекта D
}
```

При запуске получим примерно следующее (рис. 2.37).

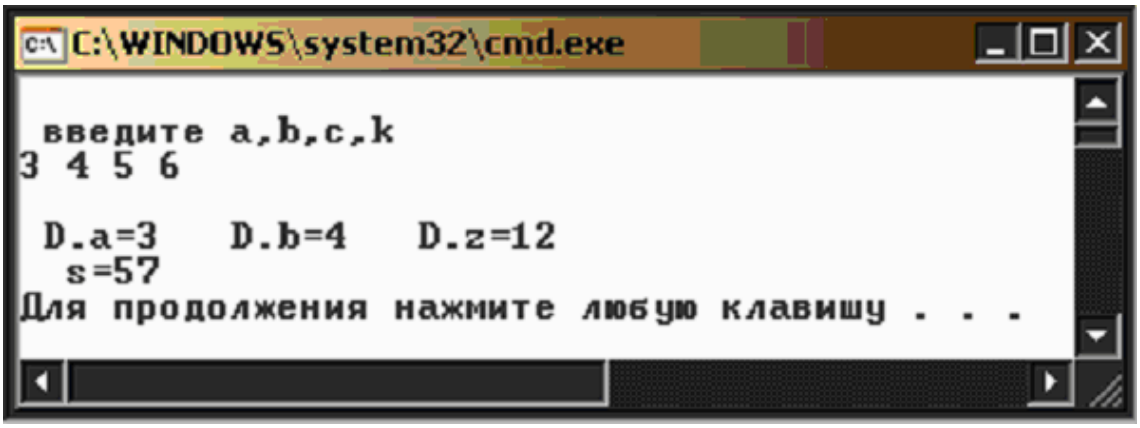


Рис. 2.37. Результат работы программы примера 15.4

Здесь создан класс `Pro`, элементами которого являются сомножители `x,y` и переменная `z` (результат их произведения). Для доступа к ним имеются три метода `putx()`, `puty`(), `putz`(), а для получения произведения `proizv`(). Кроме того, для класса определены конструктор `Pro` и деструктор `~Pro` (в классе описаны их прототипы).

Контрольные вопросы

1. Дайте понятие структуры. Как обращаются к элементам структуры?
2. Дать понятие класса. В чем основное отличие класса от структуры?
3. В чем отличие закрытых и открытых членов класса?
4. Дайте понятие конструктора и деструктора.

16. ФАЙЛЫ

В С++ ***файл***, как и в других языках, представляет собой *поименованную совокупность данных, находящуюся на внешнем устройстве и имеющую определенные атрибуты* (*характеристики*). Правило именования файлов определяются операционной системой. Для организации работы с файлами необходимо усвоить такое понятие как ***поток*** – *абстрактный канал связи, который создается в программе для обмена данными с файлами*. Это понятие введено, чтобы можно было не учитывать детали физической организации канала связи между источником и приемником информации.

Для того, чтобы в С++ начать работу с файлами, находящимися на логических дисках, необходимо «создать» соответствующие потоки. Средства С++ позволяют работать с *текстовыми* и *бинарными* файлами после того, как в программе установлена их взаимосвязь с текстовыми и бинарными потоками соответственно. В данном пособии мы ограничимся рассмотрением работы с текстовыми файлами.

16.1. Работа с текстовыми файлами

Файл, рассматриваемый как последовательность строк символов, называется ***текстовым***. Строки символов в текстовых файлах могут интерпретироваться при вводе как данные определенного вида, представленные в некотором формате. Например, последовательность 125, ограниченная с двух сторон пробелами, может быть преобразована при вводе в вещественное число типа **double** или целое число типа **int.**

Пример 16-1. Создать программу, которая на диске С: создаст файл numbers.txt и запишет в него 5 введенных пользователем целых чисел.

(Просмотрите затем созданный файл в обычном текстовом редакторе и убедитесь, что каждое число находится в отдельной строке).

Для работы с файлами на *запись* нужно сделать следующее:

1) подключить заголовочный файл **fstream**: **#include <fstream>**;

2) создать объект класса **ofstream** для управления потоком вывода: **ofstream fout**;

3) поставить этот объект в соответствие определенному файлу (с помощью метода **open**()): **fout.open("c:\\numbers2.txt ");**

4) использовать этот объект так же, как используется объект **cout** (только вывод теперь будет в файл, а не на экран): **fout<< "запись"** – это запись в файл слова "запись".

5) Для закрытия работы с файлом используют метод **close()** : **fout.close ();**

```
#include <fstream>
#include <iostream>
```

```
#include <stdio.h>
#include <conio.h>
using namespace std;
int main()
{ ofstream fout;   int i; int n;
  fout.open("C:\\numbers2.txt ");
   for (i=1;i<=5;i++)
   {cout<<"ВВЕДИ "<<i;cout<<" ЧИСЛО "; cin>> n;
          fout<<n<<"\n"; }
  fout.close ();
return 0;}
```

При *чтении* из файла действуют аналогично, но только в пункте 2-м создать объект класса **ifstream**: **ifstream fin**; поставить этот объект в соответствие определенному файлу: **fin.open(“c:\\numbers2.txt ”);**
и для чтения из файла использовать этот объект как объект **cin** : для ввода вместо **cin** писать **fin** >>numb

Пример 16-2. Создать программу, которая считает информацию из созданного ранее на диске С: файла numbers.txt, выведет ее на экран и подсчитает среднее арифметическое этих чисел.

```
// чтение из текстового файла
#include <fstream>
#include <stdio.h>
#include <conio.h>
#include <iostream>
using namespace std;
int main()
{ setlocale(LC_ALL, "Russian");
  ifstream fin;  int i,n,sum,sr; sum=0; n=0;
   const int len = 81 ;  int a;
  fin.open("C:\\numbers2.txt ");
   for (i=1; i<=5; i++)
     {fin>>a; cout<<"\n ЧИСЛО"<<i<<" = "<<a;
      sum=sum+(int)a; n++; }
   fin.close(); sr=(float) sum/n;
  cout<<"\n количество чисел в файле = "<<n;
  cout<<"\nсреднее арифметическое их = "<<sr<<"\n";
return 0;}
```

При открытии файлов могут быть заданы режимы обращения к ним (они также задаются при открытии файлов). Мы их пока не записывали, поскольку они нередко открываются по умолчанию (для **ifstream** будет режим *считывания*, для **ofstream** – режим *записи*). Здесь они записываются несколько более громоздко (см. табл.5).

Таблица 5

Режимы работы с файлами (в духе C++)

Значение режима	Описание режима	Позиция ввода-вывода
ios : : in	Считывание, для потока **ifstream** устанавливается по умолчанию	в начало файла (по умолчанию), но возможно позиционирование в любое место файла
ios : : out	Запись поверх имеющейся (старая запись стирается без предупреждения); для потока **ofstream** устанавливается по умолчанию	в начало файла (по умолчанию), но возможно позиционирование в любое место файла
ios : : ate	Запись и считывание	в конец файла (по умолчанию), но возможно позиционирование в любое место файла
ios : : app	Только для записи ; Добавление в конец файла после уже имеющейся записи	в конец файла

16.2. Работа со структурами в файлах

Рассмотрим, как можно записывать/считывать целые структуры в файлы. Правила работы со структурами мы рассмотрели в разд. 14.

Пример 16-3. В текстовом файле хранится база отдела кадров предприятия. На предприятии 10 сотрудников. Каждая строка файла содержит запись об одном сотруднике. Формат записи: фамилия и инициалы (30 поз., фамилия должна начинаться с первой позиции), год рождения (5 поз.), оклад (10 поз.). Написать программу, которая по заданной фамилии выводит на экран сведения о сотруднике, подсчитывая средний оклад всех запрошенных сотрудников.

Очевидно, что при считывании данных из файла их следует в программе разместить в массив (типа структуры). Поскольку далее будем вводить фамилию сотрудника, то для хранения фамилии будем использовать строку символов (на те же 30 поз.). Для подсчета среднего оклада понадобится цикл, в котором будем вычисляться сумма окладов задаваемых сотрудников и их количество. Для простоты условимся, что для выхода из цикла (где обрабатываются сведения о сотрудниках) вместо фамилии будем вводить слово end.

Итак, алгоритм решения задачи:

1. Считать из файла в массив сведения о сотрудниках;

2. Организовать цикл вывода сведений о сотруднике:
- ввести с клавиатуры фамилию;
- выполнить поиск сотрудников в массиве;
- увеличить суммарный оклад и счетчик количества сотрудников;
- вывести сведения о сотруднике или сообщение об их отсутствии.

3. Вывести средний оклад.

Разумеется, следует подготовить текстовый файл `dbase.txt` с нужными данными, т.е. создать текстовый файл с нужным количеством строк, причем каждая строка должна иметь нужную длину и содержимое. Рекомендуется создавать такой файл в DOS-овском текстовом файле (например, в редакторе, встроенном в файловый менеджер FAR). В этом случае не будет проблем с кодировкой русских букв. Но если заполняете файл `dbase.txt` в windows-редакторе Блокнот или Word, то для отображения русских букв надо раскомментировать строки с функциями AnsiToOem и OemToAnsi (см. листинг ниже).

Например, создадим такой файл:

```
Мамин              1989 10000.00
Папин              1979 11000.00
Papin              1969 12000.00
Mamin              1959 13000.00
Маминский          1949 14000.00
Dedov              1939 15000.00
Pradedov           1929 16000.00
Ионов              1919 17000.00
Яковлев            1909 18000.00
Elenin             1899 19000.00
```

В рассмотренной ниже программе в цикле `while (!fin.eof())` происходит построчное считывание из файла в строку `buf` и заполнение очередного элемента массива `dbase` (типа структуры). Здесь `eof()` – метод для проверки конца файла (он дает логическое значение). Для формирования полей структуры используются:

- функция копирования строк `strncpy(p,a,n)` – она копирует `n` символов из строки `a` в строку `s`;
- функция `atoi(str)` – преобразует символьное представление целого числа в целое число;
- функция `atof(str)` – преобразует символьное представление вещественного числа в вещественное число.

Обратите внимание, что завершающий нуль-символ в поле Фамилия заносится «вручную», поскольку функция `strncpy(p,a,n)` делает это только в том случае, если строка-источник короче строки-приемника. В

функцию atoi передается адрес начала подстроки, в которой находится год рождения. В цикл добавлен контрольный вывод на экран считанной строки (в виде сформированных полей структуры). В конец цикла для выхода вставлен if (i>=l_dbase) break;, поскольку в C++ после чтения из файла последнего элемента условие конца файла *не возникает*. Оно возникает при следующем чтении, когда программа пытается считать данные за последним элементом в файле.

Цикл поиска сотрудников по фамилии while (true) организован как бесконечный с принудительным выходом (при вводе строки «end»). В этом цикле используется для сравнения функция поиска подстроки strstr(s1,s2) – она ищет вхождение строки s2 в строку s1 и принимает соответствующее логическое значение (если входит, то true). Функция strcmp(s1,s2) просто проверяет равенство строки s1 строке s2, она также принимает логическое значение (false, true).

```
#include <fstream>
#include <iostream>
#include <string.h>
#include <stdlib.h>
using namespace std;
int main()
{ setlocale(LC_ALL, "Russian");
const int l_name=30, l_year=5, l_pay=10,
l_buf = l_name+l_year+l_pay;
struct Man
{    int birth_year;
     char name[l_name+1];
     float pay; };
const int l_dbase=10;
Man dbase [l_dbase+1];
char buf[l_buf+1];
char sName[l_name+1];
ifstream fin;
fin.open( "c:\\dbase.txt", ios::in );
if (!fin){cout<<" Ошибка открытия файла ";
return 1;}
int i=0;
while (!fin.eof())
{    fin.getline(buf,l_buf);
     strncpy(dbase[i].name, buf, l_name);
     dbase[i].name[l_name]= '\0';
     dbase[i].birth_year=atoi(&buf[l_name]);
     dbase[i].pay=atof(&buf[l_name+l_year]);
```

```
 cout<<dbase[i].name<<dbase[i].pay
   <<"    "<<dbase[i].birth_year<<"\n";
 i++; if (i>=l_dbase) break; }
int n_record =i, n_man=0;
float mean_pay=0;
while (true)
{    cout<<" Введите фамилию или слово end: ";
  cin>>sName;
    if (strcmp(sName, "end")== 0) break;
    bool not_found=true;
    for (i=0; i<n_record; i++)
    {    if (strstr(dbase[i].name, sName))
             if (dbase[i].name[strlen(sName)]==' ')
         {    strcpy(sName, dbase[i].name);
              n_man++; mean_pay+=dbase[i].pay;
              not_found=false;          }
    }
if (not_found) cout<<"такого сотрудника нет"<< endl;
}
if (n_man>0) cout<<Rus" Средний оклад: "
<<mean_pay/n_man <<"\n";
return 0;}
```

Результат работы программы будет иметь, например, такой вид (рис. 2.38).

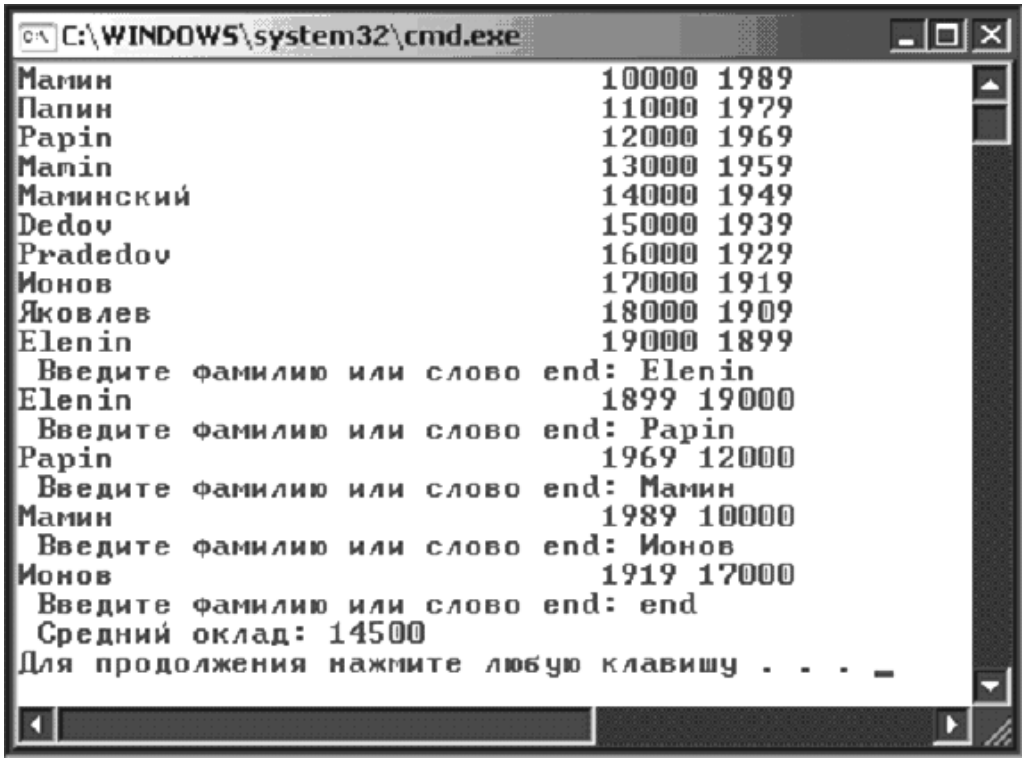


Рис. 2.38. Результат работы программы примера 16-3

Пример 16-4. Считать данные из файла (использовавшегося в примере 16-3), упорядочить их по году рождения и записать в другой файл.

Для сортировки здесь будем использовать простой алгоритм: выбирается наименьший элемент массива и меняется местами с первым элементом, затем рассматриваются элементы, начиная со второго, наименьший из них меняется местами со вторым и т.д.

Элементами массива в данной задаче являются структуры. Для структур одного типа определена операция присваивания, поэтому обмен двух элементов массива структур выглядит точно так же, как для основных типов данных.

```
#include <fstream>
#include <iostream>
#include <string.h>
#include <stdlib.h>
using namespace std;
int main()
{ setlocale(LC_ALL, "Russian");
const int l_name=30, l_year=5, l_pay=10,
    l_buf=l_name+l_year+l_pay;
struct Man
{   int birth_year;
    char name[l_name+1];
    float pay; };
const int l_dbase=10;
Man dbase [l_dbase+1];
char buf[l_buf+1]; char sName[l_name+1];
ifstream fin;
fin.open("c:\\dbase.txt", ios::in );
if (!fin){cout<<" Ошибка открытия файла "; return 1;}
int i=0;
while (!fin.eof())
{   fin.getline(buf,l_buf);
    strncpy(dbase[i].name, buf, l_name);
    dbase[i].name[l_name]= '\0';
    dbase[i].birth_year=atoi(&buf[l_name]);
    dbase[i].pay=atof(&buf[l_name+l_year]);
    cout<<dbase[i].name<<dbase[i].pay
      <<" "<<dbase[i].birth_year<<"\n";
 i++;  if (i>=l_dbase) break; }
int n_record=i;
fin.close();
ofstream fout;fout.open("c:\\dbase2.txt", ios::out);
if (!fout){cout<<" Ошибка открытия файла "; return 1;}
```

```
for (i=0; i<n_record-1; i++)
{// принимаем за наименьший первый из элементов
 int imin=i;
//поиск номера минимального элемента из неупорядоченных
 for (int j=i+1; j<n_record; j++)
 if (dbase[j].birth_year<dbase[imin].birth_year)
imin=j;
 // обмен двух элементов массива структур
 Man a=dbase[i]; dbase[i]=dbase[imin];
dbase[imin]=a;}
//запись в файл отсортированной базы данных
for (i=0; i<n_record; i++)
{    fout<<dbase[i].name<<dbase[i].birth_year
    <<" "<<dbase[i].pay<<endl;}
fout.close();
cout<<"Сортировка базы данных завершена"<< endl;
//вывод на экран отсортированной базы данных
for (i=0; i<n_record; i++)
{    cout<<dbase[i].name<<dbase[i].birth_year<<"
"<<dbase[i].pay<<endl;}
return 0;}
```

Результат работы программы, например, такой (рис. 2.39).

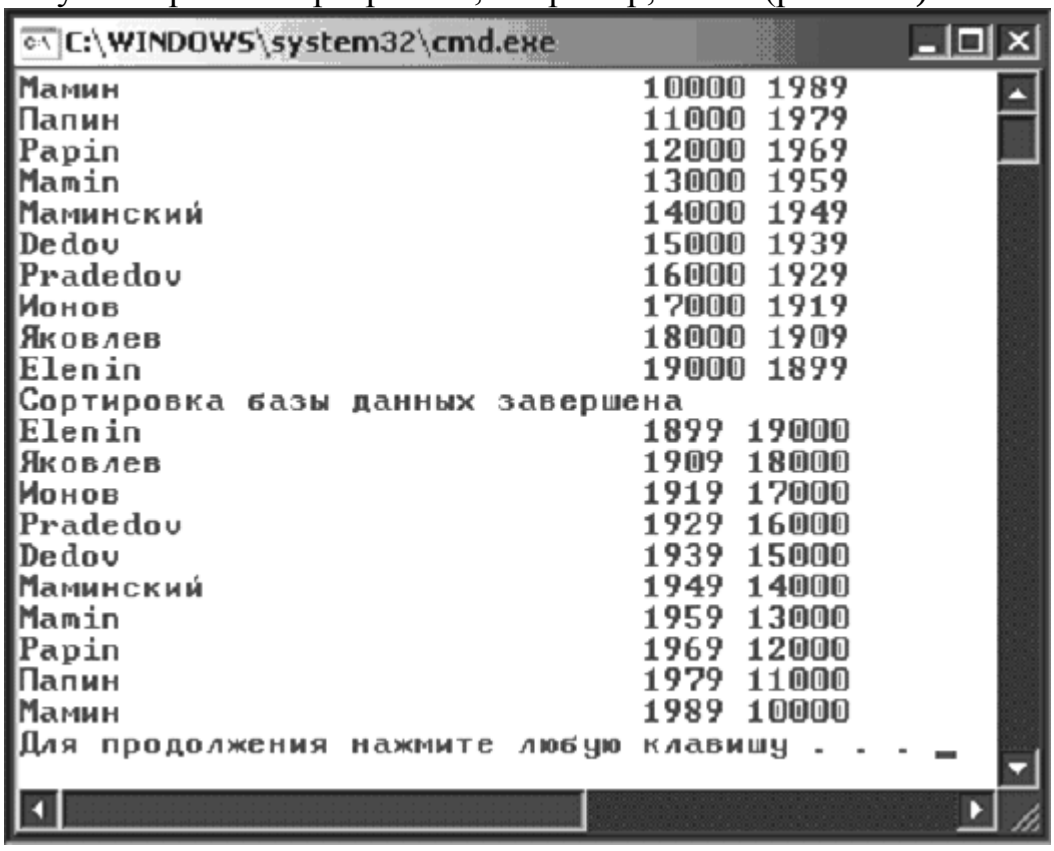


Рис. 2.39. Результат работы программы примера 16-4

16.3. Работа с классами в файлах

Рассмотрим, как работать с *классами* (вместо структур) при чтении (или записи) в файл (из файла). Рассмотрим тот же пример 16.3, но пользуясь технологией работы с классами.

Пример 16-5. В текстовом файле хранится база отдела кадров предприятия. На предприятии 10 сотрудников. Каждая строка файла содержит запись об одном сотруднике. Формат записи: фамилия и инициалы (30 позиций, фамилия должна начинаться с первой позиции), год рождения (5 позиций), оклад (10 позиций). Написать программу, которая по заданной фамилии выводит на экран сведения о сотруднике, подсчитывая средний оклад всех запрошенных сотрудников.

Напомним, что в программе, предложенной для решения примера 16-3 для хранения сведений об одном сотруднике была использована следующая структура `Man`:

```
struct Man
{ char name[l_name + 1];
   int birth_year;
   float pay; };
```

Начнем с того, что преобразуем эту структуру в *класс*, так как мы предполагаем, что наш новый тип будет обладать более сложным поведением, чем просто чтение и запись его полей:

```
class Man
{ char name[l_name + 1];
   int birth_year;
   float pay;};
```

Замечание здесь в том, что все поля класса по умолчанию – закрытые (private). Так что если клиентская функция `main()` объявит объект `Man man`, а потом попытается обратиться к какому-либо его полю, например: `man.pay =value`, то компилятор быстро пресечет это безобразие, отказавшись компилировать программу. Поэтому в состав класса надо добавить методы доступа к его полям. Эти методы должны быть общедоступными, или открытыми (public).

Однако предварительно вглядимся внимательнее в определения полей. В решении примера 16-3 поле `name` объявлено как статический массив длиной `l_name + 1`. Это не очень гибкое решение. Мы хотели бы, чтобы наш класс `Man` можно было использовать в будущем в разных приложениях, например для случая больших длин поля `name`. Решение состоит в использовании *динамического* массива символов с требуемой длиной.

Поэтому заменим поле `char name[l_name + 1]` на поле `char* pName`. Сразу возникает вопрос: кто и где будет выделять память под этот массив? Для этого в состав класса необходимо включить метод, который обеспечил бы выделение памяти под указанный динамический массив при создании объекта (переменной типа `Man`). Напомним, что метод, который автоматически вызывается при создании экземпляра класса, называется *конструктором*. Компилятор безошибочно находит этот метод среди прочих методов класса, поскольку его имя всегда совпадает с именем класса.

Парным конструктору является другой метод, называемый *деструктором*, который автоматически вызывается перед уничтожением объекта. Имя деструктора отличается от имени конструктора только наличием предваряющего символа ~ (тильда). Ясно, что если в конструкторе была выделена динамическая память, то в деструкторе нужно побеспокоиться об ее освобождении. Напомним, что объект, созданный как локальная переменная в некотором блоке { }, уничтожается, когда при выполнении достигнут конец блока. Если же объект создан с помощью операции `new`, например:

```
Man* pMan = new Man;
```

то для его уничтожения применяется операция `delete`, например: `delete pMan`.

Итак, наш класс принимает следующий вид:

```
class Man
{public:
    Man(int lName = 30)
    { pName = new char[lName + 1]; } // конструктор
     ~Man() { delete [] pName; } // деструктор
private:
    char* pName;
    int birth_year;
    float pay; };
```

Обратим ваше внимание на одну *синтаксическую* деталь – объявление класса должно обязательно завершаться точкой с запятой (;). Если вы забудете это сделать, то получите от компилятора длинный список маловразумительных сообщений о чем угодно, но только не об истинной ошибке.

Рассмотрим теперь одну важную *семантическую* деталь: в конструкторе класса параметр `lName` имеет значение по умолчанию (30). Если все параметры конструктора имеют значения по умолчанию или если конструктор вовсе не имеет параметров, он называется конструктором по умолчанию. Зачем понадобилось специальное название для такой разновидности конструктора? Разве это не просто удобство для клиента – передать некоторые значения по умолчанию одному из методов класса? Нет! Конст-

руктор – это особый метод, а конструктор по умолчанию имеет несколько специальных областей применения.

Во-первых, такой конструктор используется, если компилятор встречает определение массива объектов, например: `Man man[25]`. Здесь объявлен массив из 25 объектов типа `Man`, и каждый объект этого массива создан путем вызова конструктора по умолчанию! Поэтому если вы забудете снабдить класс конструктором по умолчанию, то вы не сможете объявлять массивы объектов этого класса.

В данном случае мы описали методы внутри описания самого класса (как встроенные функции), но это можно (а чаще – нужно) делать вне описания класса, как показано ниже:

```
class Man
{public:
    Man(int lName = 30); // конструктор
    ~Man(); // деструктор
private:
    char* pName;
    int birth_year;
    float pay; };
// (реализация класса)
Man::Man(int IName) { pName = new char[lName + 1]; }
Man::~Man() { delete [] pName: }
```

При внешнем определении метода перед его именем указывается имя класса, за которым следует операция доступа к области видимости `::`. Выбор способа определения метода зависит в основном от его размера: короткие методы можно определить как встроенные, что может привести к более эффективному коду.

Продолжим процесс проектирования интерфейса нашего класса. Какие методы нужно добавить в класс? На этом этапе очень полезно задаться следующим вопросом: какие обязанности должны быть возложены на класс `Man`?

Первую обязанность мы уже реализовали: объект класса хранит сведения о сотруднике. Чтобы воспользоваться этими сведениями, клиент должен иметь возможность получить эти сведения, изменить их и вывести на экран. Кроме этого, для поиска сотрудника желательно иметь возможность сравнивать его имя с заданным.

Начнем с методов, обеспечивающих доступ к полям класса. Для считывания значений полей добавим методы `GetName()`, `GetBirthYear()`,`GetPay()`. Очевидно, что аргументы здесь не нужны, а возвращаемое значение совпадает с типом поля.

Для записи значений полей добавим методы SetName(), SetBirthYear(), SetPay().

Чтобы определиться с содержимым этих методов, надо представить себе, как они будут вызываться клиентом. Напомним, что в примере 16-3 фрагмент ввода исходных данных был реализован следующим образом (здесь мы используем идентификатор man вместо идентификатора dbase):

```
int i = 0;
while (fin.getline(buf, l_buf))
{ strncpy(man[1].name, but, l_name);
  man[i].name[l_name] = '\0';
  man[i].birth_year = atoi(&buf[l_name]);
  man[i].pay = atof(&buf[l__name + l_year]);
  i++; }
```

Как видно из этого фрагмента, в теле цикла while на i-й итерации выполняются следующие действия:

- очередная строка читается из файла fin в символьный массив buf;
- из массива buf в поле name структуры man[i] копируются первые l_name символов;
- из буфера buf извлекается очередная порция символов, отличных от пробела, преобразуется к типу int и записывается в поле birth_year структуры man[i];
- извлекается следующая порция символов, отличных от пробела, преобразуется к типу float и записывается в поле pay структуры man[i].

Если чисто механически скопировать такое разделение обязанностей между клиентом (main) и сервером (объект класса Man), то мы получим в теле цикла код вида:

```
man[i].SetName(buf);
man[i].SetBirthYear(atoi(&buf[l_name]));
man[i].SetPay(atof(&buf[l_name + l_year])):
i++;
```

Вряд ли такое решение можно признать удачным. Глаз сопровождающего программиста наверняка «споткнется» на аргументах вида atoi (&buf[l_name]). Ведь подобные выражения фактически являются сущностями типа «как делать», а не сущностями типа «что делать»! Иными словами, крайне нежелательно нагружать клиента избыточной информацией. Второстепенные детали, или детали реализации, должны быть упрятаны внутрь класса. Поэтому в данной задаче более удачной является оформление метода void SetBirthYear(const char* fromBuf). Аналогичные размышления можно повторить и для метода SetPay().

Разобравшись с методами доступа, перейдем теперь к основной части алгоритма функции main(), решающей подзадачу поиска сотрудника в базе по заданной фамилии и вывода сведений о нем. Напомним, что в

примере 16.3 фрагмент поиска сотрудника с заданным именем был реализован с помощью следующего цикла:

```
for (i = 0; i < n_record; ++i)
{    if (strstr(man[i].name. name))          // 1
  if (man[i].name[strlen(name)] = = ' ' )
 { strcpy(name. man[i].name);
cout <<name<<man[i].birth_year<<' '<<man[i].pay<< "\n"; }
}
```

Опять задумаемся над вопросом: какая информация в этом решении избыточна для клиента? Здесь решаются две подзадачи: сравнение имени в очередной прочитанной записи с заданным именем `name` и вывод информации в поток `cout`.

Поручим решение первой подзадачи методу `CompareName(const char* name)`, спрятав в него подробности решения, а решение второй подзадачи – методу `Print ()`.

Проектирование интерфейса класса `Man` завершено. Давайте посмотрим на текст программы, в которой реализован и использован описаный выше класс `Man`.

```
#include <iostream>
#include <cstring>
#include <fstream>
const char filename[] = "c:\\dbase.txt";
using namespace std;
const int l_name = 30; const int l_year = 5;
const int l_pay = 10;
const int l_buf = l_name + l_year + l_pay;
class Man
{public:
    Man(int lName = 30);
    ~Man();
    bool  CompareName(const char*) const;
    int   GetBirthYear() const { return birth_year; }
    float GetPay() const { return pay; }
    char* GetName() const { return pName; }
    void  Print() const;
    void  SetBirthYear(const char*);
    void  SetName(const char*);
    void  SetPay(const char*);
private:
    char* pName;
    int   birth_year;
    float pay; };
```

```
Man::Man(int lName)
{   cout << "Constructor is working" << endl;
    pName = new char[lName + 1];}
Man::~Man()
{   cout << "Destructor is working" << endl;
    delete [] pName;}
void Man::SetName(const char* fromBuf)
{   strncpy(pName, fromBuf, l_name);
    pName[l_name] = 0;}
void Man::SetBirthYear(const char* fromBuf)
{    birth_year = atoi(fromBuf + l_name);}
void Man::SetPay(const char* fromBuf)
{    pay = atof(fromBuf + l_name + l_year);}
bool Man::CompareName(const char* name) const
{if((strstr(pName,name))&&(pName[strlen(name)]==' '))
    return true;    else        return false; }
void Man::Print() const
{ cout << pName << birth_year << ' ' << pay << "\n";}
int main()
{ setlocale(LC_ALL, "Russian");
const int maxn_record = 10; Man man[maxn_record+1];
 char buf [l_buf + 1]; char name[l_name + 1];
 ifstream fin(filename);
 if (!fin)
{cout <<Rus("Нет файла ")<<filename<<endl; return 1;}
  int i = 0;
  while (fin.getline(buf, l_buf))
 { if (i > maxn_record)
   { cout << "Слишком длинный файл"; return 1;}
        man[i].SetName(buf);
        man[i].SetBirthYear(buf);
        man[i].SetPay(buf);
        i++;  }
 int n_record = i, n_man = 0;
 float mean_pay = 0;
 while (true)
{  cout << "Введите фамилию или слово end: ";
   cin >> name;
   if (0 = = strcmp(name, "end")) break;
   bool not_found = true;
   for (i = 0; i < n_record; ++i)
   { if (man[i].CompareName(name))
       {       man[i].Print();
               n_man++; mean_pay += man[i].GetPay();
```

```
                not_found = false;
                break;        }     }
 if (not_found) cout<<"Такого сотрудника нет"<<e }
 if (n_man) cout<< " Средний оклад: " << mean_pay /n_man <<
endl;
    return 0; }
```

Вставим дополнительно отладочную печать в конструкторе и деструкторе. Вывод сообщений типа «Constructor is working», «Destructor is working» очень помогает на начальном этапе освоения классов.

Результат работы такой программы смотри на рис. 2.40:

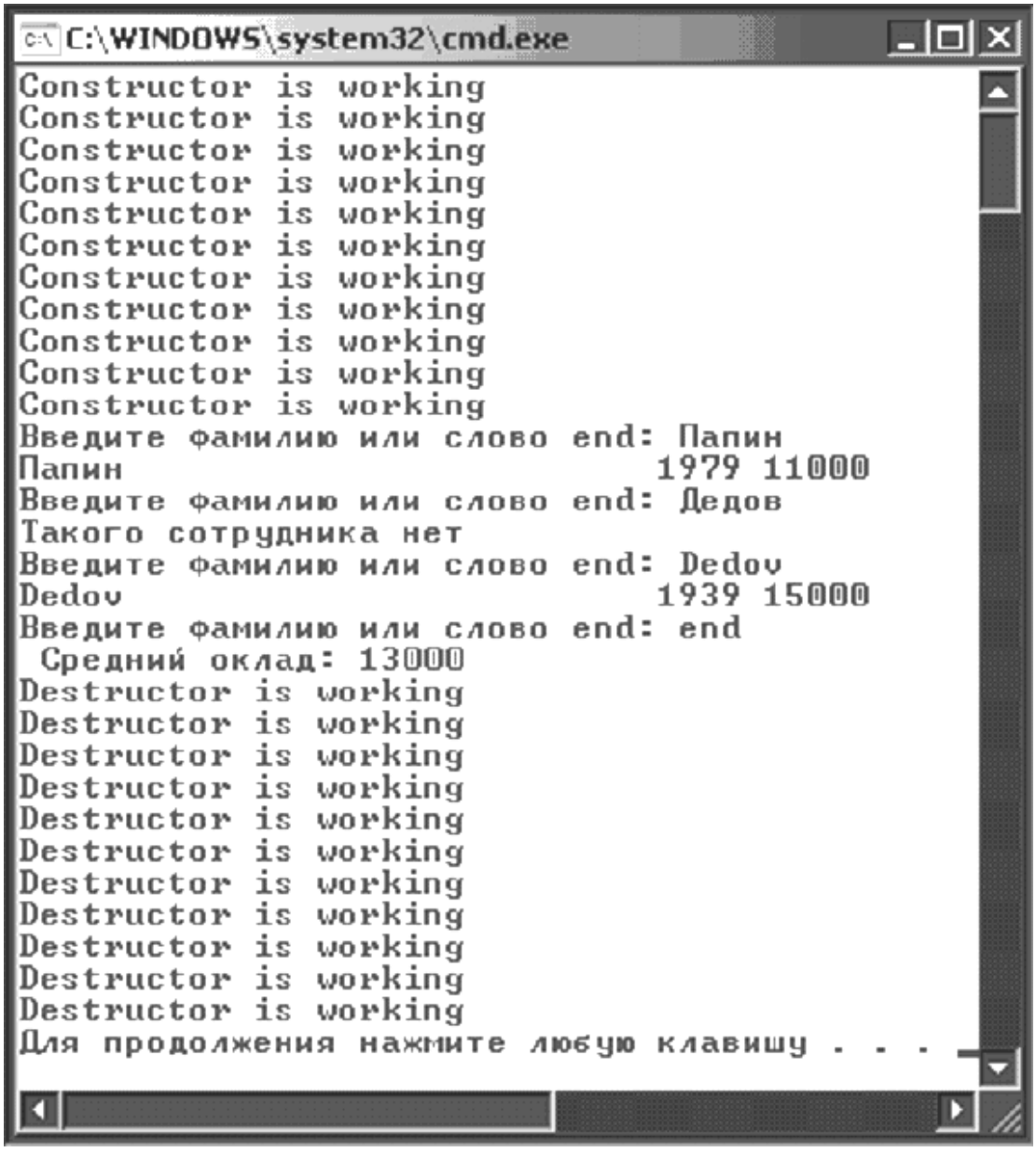


Рис. 2.40 Результат работы программы примера 16.5

Контрольные вопросы

1. Какие методы служат для открытия и записи в текстовый файл?
2. Какие методы служат для чтения из файла?
3. В чем особенность работы со структурами в файлах?
4. Опишите особенности работы с классами в файлах.

ЛИТЕРАТУРА К ГЛАВЕ 2

1. Павловская Т. C/C++. Программирование на языке высокого уровня. Учебник. – СПб. : Питер, 2009. - 464 с.
2. Павловская Т., Щупак Ю. C++. Объектно-ориентированное программирование. Практикум. – СПб.: Питер, 2006 – 264 с.
3. Лафоре Р. Объектно-ориентированное программирование в C++. – СПб.: Питер, 2007. – 928 с.
4. Хортон А. Visual C++ 2005: базовый курс. – М.: ООО «ИД Вильямс», 2007 – 1152 с.

ПРИЛОЖЕНИЯ

Приложение 1. Список математических функций

Приведенный ниже список содержит основные функции с указанием их типа и типов аргументов (без указания имени аргумента). При использовании функции следует добавлять.**h** к имени заголовочного файла.

Прототип	Описание	Заголовочный файл
int abs(int)	Абсолютное значение	stdlib
long labs(long)	Абсолютное значение	stdlib
double fabs(double)	Абсолютное значение	math
double sqrt(double)	Квадратный корень	math complex
double pow(double, double)	Первый аргумент в степени второго	math complex
double exp(double x)	Функция $\mathbf{e^x}$ **(**где x – аргумент)	math complex
double log(double)	Натуральный логарифм (**ln**)	math
double log10(double)	Логарифм по основанию 10 (**lg**)	math
double ceil(double)	Наименьшее целое большее или равное аргументу	math
double floor(double)	Наибольшее целое меньшее или равное аргументу	math
int rand % n	Случайное число от 0 до n-1	stdlib
void srand(unsigned int)	Реинициализирует генератор случайных чисел	stdlib
double acos(double)	арккосинус	math
double asin(double)	арксинус	math
double atan(double)	арктангенс	math
double cos(double)	косинус	math
double cosh(double)	косинус гиперболический	math
double sin(double)	синус	math
double sinh(double)	синус гиперболический	math
double tan(double)	тангенс	math
double tanh(double)	тангенс гиперболический	math

Приложение 2. Список строковых функций (для работы с символьными массивами)

Прототип	Описание	Заголовочный файл
char* strcat(char *Dest, const char *Source)	Добавляет строку Source в строку Dest и возвращает указатель на Dest	string
char* strchr(const char* string, int ch)	Выполняет поиск символа с кодом ch слева направо в строке string; возвращает указатель на первое вхождение символа. Если символ не обнаружен, возвращает NULL (нулевой указатель)	string
char* strstr(const char* string, const char* Search)	Выполняет поиск строки ch слева направо в строке string; возвращает указатель на первое вхождение символа. Если символ не обнаружен, возвращает NULL (нулевой указатель)	string
char* strcpy(char *Dest, const char *Source)	Копирует строку Source в строку Dest и возвращает указатель на Dest	string
char* strncpy(char *Dest, const char *Source, size_t count)	Копирует не более чем count символов из строки Source в строку Dest и возвращает указатель на Dest	string
int strcmp(const char* string1, const char* string2)	Сравнивает строки string1, string2 лексикографически: возвращает –1, если string1< string2; 0, если равны; +1, если string1> string2	string
double atof (const char* str)	Преобразует строку str в вещественное число	stdlib
int atoi (const char* str)	Преобразует строку str в целое число	stdlib

ГЛАВА 2. ПРИЛОЖЕНИЯ WINDOWS FORMS

ВВЕДЕНИЕ

Windows Forms — это набор средств для создания windows-приложений, выполняющихся в среде CLR (Common Language Runtime — общеязыковая исполняющая среда). **Форма** — это окно, служащее основой окна приложения или диалогового окна, в которое можно добавлять другие элементы управления, предназначенные для взаимодействия с пользователем. Пакет Visual C++ 2008 поставляется со стандартным набором более 60 элементов управления, которые можно использовать в формах. Многие стандартные элементы управления, такие как `Button`, представляющие кнопки, предназначенные для обработки щелчков мыши на них, или `TextBox`, которые позволяют вводить текст, реализуют простые функции взаимодействия с пользователем. Другие являются **контейнерами** — то есть могут содержать другие элементы управления. Например, `GroupBox` может содержать другие элементы управления вроде `Button` или `TextBox`, и его функция состоит просто в группировании элементов управления.

Форма и используемые с ней элементы управления представлены классом C++/CLI. Каждый класс обладает набором свойств, которые определяют поведение и внешний вид элемента управления или формы. Определение свойств элементов управления может выполняться:

- интерактивно при построении графического интерфейса пользователя с помощью средств IDE (Integrated Developement Environment — интегрированная среда разработки)
- изменение значений свойств может производиться также во время выполнения с помощью добавляемых в программу функций,
- либо с помощью кода, добавляемого в существующие функции.

При создании проекта приложения создается как окно приложения Windows Forms, построенное на основе класса Form, так и весь код, обеспечивающий отображение этого окна приложения. После создания проекта Windows Forms разработка приложения сводится к выполнению четырех отдельных операций:

– Интерактивное создание графического интерфейса пользователя на вкладке `Form Конструктор` (Конструктор формы), отображаемой в панели `Редактор` (Редактор), путем выбора элементов управления в окне `Панель элементов` и их помещения в форму. На этом этапе можно также создавать дополнительные окна форм.
– Изменение свойств элементов управления и форм в окне `Свойства` в соответствии с потребностями приложения.

– Обработчики событий щелчков для элементов управления можно создавать, дважды щелкая на элементе управления на вкладке `Form Design`. В окне `Properties` элемента управления в качестве его обработчика события можно также определять существующую функцию.
– Для удовлетворения потребностей приложения можно изменять и расширять классы, автоматически создаваемые в результате взаимодействия с вкладкой `Form Design`.

1. РАЗРАБОТКА ПРИЛОЖЕНИЯ.

Как обычно, запускаем среду Visual Studio 2008 командой ***Пуск/Программы/Microsoft Visual Studio 2008/ Microsoft Visual Studio 2008***. Далее, приступаем к созданию проекта. Для создания проекта, как обычно, нужно дать команду ***Файл/Создать/ Проект*** (или нажать комбинацию клавиш **Ctrl-Shift-N**). В левой части возникшего окна (***рис. 1.1***) отображаются типы проектов, которые можно создавать. В данном случае в левой части выберем **CLR**, а на правой панели выберем **Windows Forms Application**.

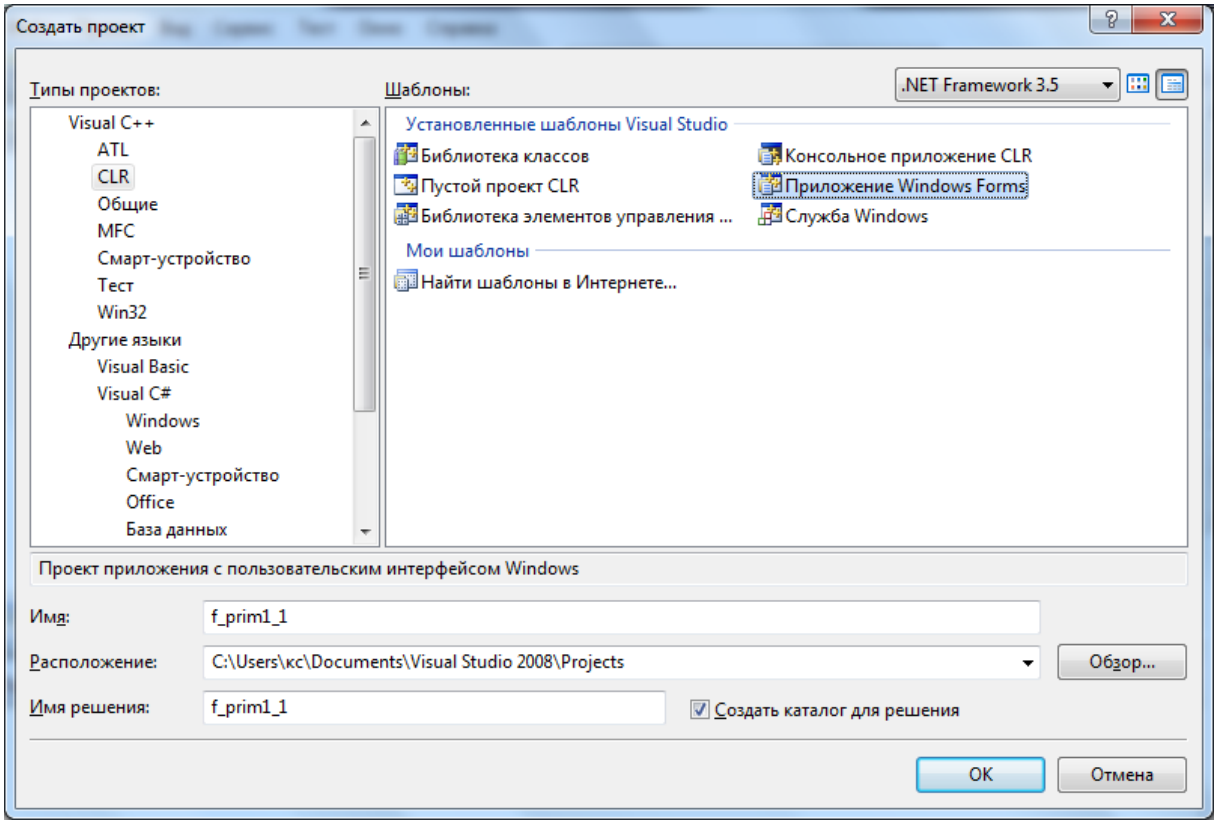


Рис 1.1 Окно создания приложения Windows Forms

Ниже (в строке Name) ввести имя для проекта (зададим, например, **f_prim1_1**) и нажать ОК. Обратите внимание, что при этом должна быть

установлена «птичка» **Создать каталог для решения**. В результате мастер создания приложений сгенерирует код приложения формы Windows и отобразит окно конструктора, содержащее форму, как она будет отображена приложением (см. ***рис. 1.2***)

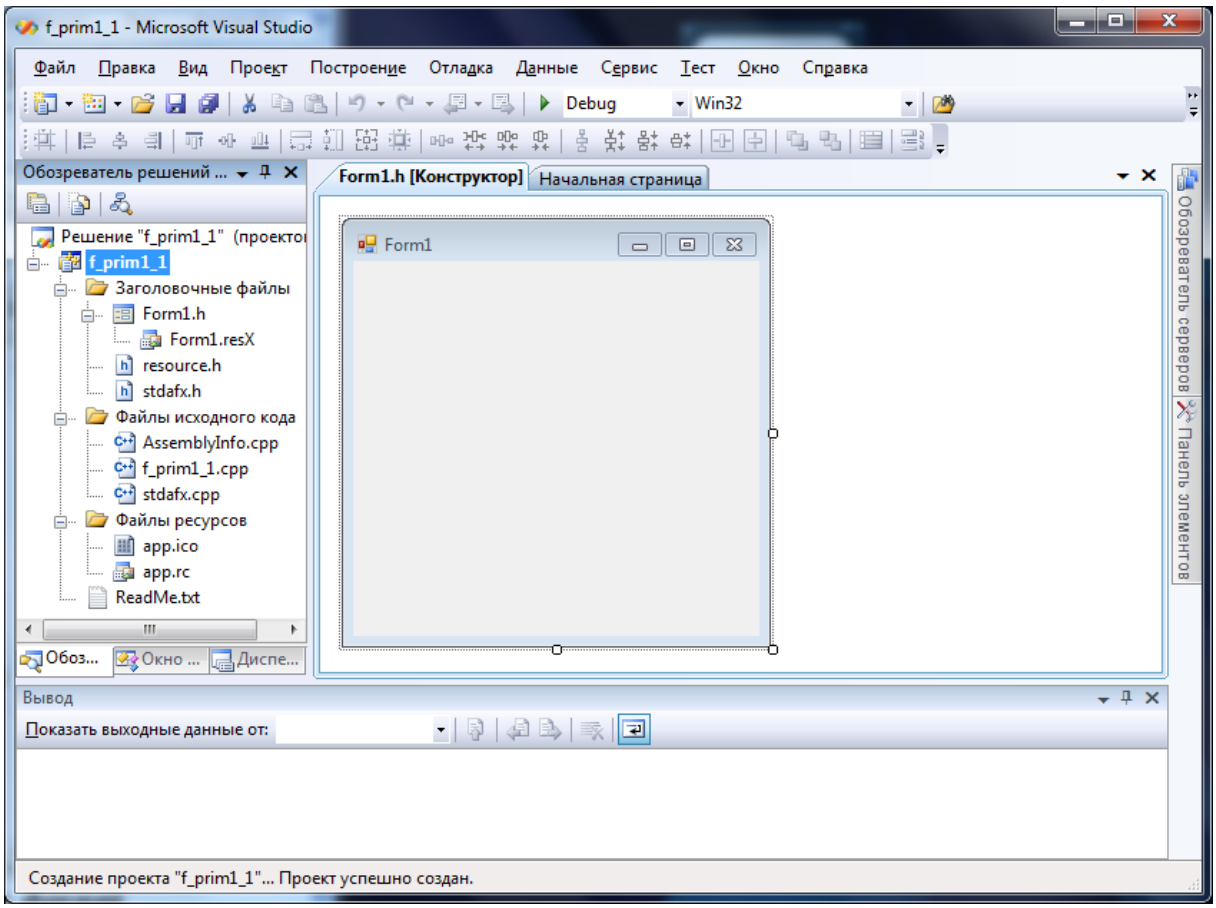


Рис 1.2 Окно конструктора с пустой формой

Оказывается, что даже такое пустое приложение уже имеет код, хотя мы еще не создавали никаких описаний, функций и т.п. Для открытия окна кода достаточно щелкнуть по форме правой клавишей и мыши и в контекстном меню выбрать `Перейти к коду`. При этом появится вкладка `Form1.h`, где и будет располагаться код.

Код определяет класс Form1, представляющий окно приложения. Прежде всего, следует отметить, что он определен в собственном пространстве имен:

```
namespace f_prim1_1 {
    using namespace System;
    using namespace System::ComponentModel;
    using namespace System::Collections;
    using namespace System::Windows::Forms;
    using namespace System::Data;
    using namespace System::Drawing;
```

Компиляция проекта приводит к созданию новой сборки, код которой относится к пространству имен, совпадающему с именем проекта (в данном случае `f_prim1_1)`. Пространство имен позволяет различать типы с одинаковыми именами в различных сборках, поскольку каждое имя типа уточняется именем конкретного пространства имен.

Существуют шесть рекомендаций к использованию пространств имен библиотеки .NET, охватывающих функциональные возможности, которые, скорее всего, потребуются в приложениях (см. *табл. 1*)

Таблица 1. Пространство имен библиотеки .NET

Пространство имен	**Содержимое**
System	Содержит классы, которые определяют типы данных, используемые во всех приложениях CLR. Оно содержит также классы событий и обработчиков событий, исключения и классы, поддерживающие функции общего применения.
System::ComponentModel	Содержит классы, которые поддерживают работу компонентов графического интерфейса пользователя в приложениях CLR
System::Collections	Содержит классы коллекций, предназначенные для всевозможной организации данных, в том числе классы определения списков, очередей и стеков
System::Windows::Forms	Содержит классы, которые поддерживают использование в пиложении средств Windows Forms
System::Data	Содержит классы, которые поддерживают набор компонентов ADO.NET, используемый для доступа и обновления источников данных
System:: Drawing	Содержит классы, которые поддерживают основные графические операции, подобные рисованию на форме или компоненте

Далее в окне кода приводятся строки вида:

```
public ref class Form1 : public Sys-
tem::Windows::Forms::Form
    {
    public:
        Form1(void)
        {
```

```
            InitializeComponent();
            //
            //TODO: добавьте код конструктора
            //
        }

    protected:
        /// <summary>
        /// Освободить все используемые ресурсы.
        /// </summary>
        ~Form1()
        {
            if (components)
            {
                delete components;
            }
        }
```

Класс `Form1` — производный от класса `Form`, который определен в пространстве имен `System::Windows::Forms`. Класс `Form` представляет окно приложения или диалогового окна, а класс `Form1`, который определяет окно для пространства имен `f_prim11`, наследует все члены класса `Form`.

Этому разделу на странице кода предшествует такой комментарий:

```
/// Внимание! При изменении имени этого класса
///необходимо также изменить свойство имени
/// файла ресурсов ("Resource ///File Name")
///для средства компиляции управляемогоресурса,
/// связанного со всеми файлами с расширением
///.resx, от которых зависит данный класс.
///В противном случае, конструкторы не смогут
/// правильно работать с локализованными
/// ресурсами, сопоставленными данной форме.
```

Раздел в конце класса `Form1` содержит определение функции `InitializeComponent()`. Эта функция вызывается конструктором для определения окна приложения и любых компонентов, добавляемых в форму. Согласно приведенным комментариям, этот раздел кода не следует модифицировать с помощью редактора кода, поскольку он обновляется автоматически при интерактивном изменении окна приложения. При использовании средств `Form[Конструктор]` важно соблюдать рекомендации этого комментария и не изменять вручную автоматически сгенерированный код. В противном случае рано или поздно это приведет к возникновению ошибок. Конечно, весь код для приложения `Windows Forms`

можно создать с нуля, но применение возможностей Form[Конструктор] для интерактивного создания графического интерфейса пользователя приложения значительно ускоряет работу и снижает вероятность ошибок. Тем не менее, это не означает, что не нужно знать, как работает этот код.

Вначале код функции InitializeComponent() выглядит следующим образом:

```
void InitializeComponent(void)
{  this->components =
 gcnew System::ComponentModel::Container();
   this->Size = System::Drawing::Size(300,300);
   this->Text = L"Form1";
   this->Padding = System::Windows::Forms::Padding(0);
   this->AutoScaleMode =
 System::Windows::Forms::AutoScaleMode::Font;        }
```

Вначале components класса Form1 унаследован от базового класса, и его задача — отслеживание компонентов, постепенно добавляемых в форму. Первый оператор сохраняет в члене components дескриптор объекта Container, представляющего коллекцию, которая хранит компоненты графического интерфейса пользователя в списке. Каждый новый компонент, добавляемый в форму с помощью средств Form Design, добавляется в этот объект Container.

Остальные операторы функции InitializeComponent() определяют свойства объекта **Form1**. Ни одно из этих свойств не следует изменять непосредственно в коде, но их значения можно выбирать посредством окна Properties (см. ***рис. 1.3***) формы.

Если переключится обратно на вкладку Form1.h[Конструктор] в окне редактора и щелкнуть правой клавишей мыши, выбрать Свойства, то и увидим окно свойств формы (***рис.1.3***):

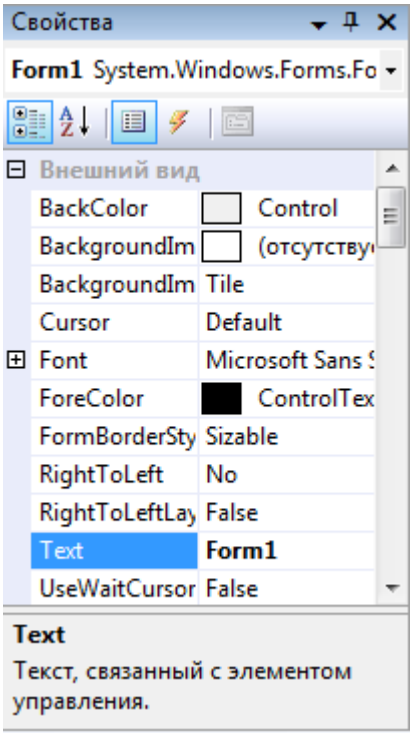


Рис. 1.3 Окно Properties объекта Form1

Чтобы получить представление о доступных возможностях, имеет смысл просмотреть список свойств формы. Щелчок на любом из них приводит к отображению описания в нижней части окна.

Например, если сменить значение свойства **Text** (вместо *Form1* написать, например, «Первый проект» и нажать клавишу Enter), то изменится надпись в заголовке окна формы на «Первый проект».

Свойства, слева от которых отображен символ +, обладают несколькими значениями, и щелчок на этом символе отображает эти значения. Например, если выберем свойство **Size** (щелкнем по +), то появятся свойства **Width** (ширина) и **Height** (высота). Там можно задать другие числовые значения.

Если вернуться к вкладке `Form1.h` в окне Редактора, то увидим, что код функции `InitializeComponent()` изменился, отражая внесенные изменения свойств.

Рассмотрим порядок разработки конкретного приложения.

Пример 1-1. *Создать проект с тремя кнопками* **:**

1) для вывода надписи на форме
2) для смена цвета формы (на красный и обратно)
3) для закрытия формы (см. ***рис. 1-4***)

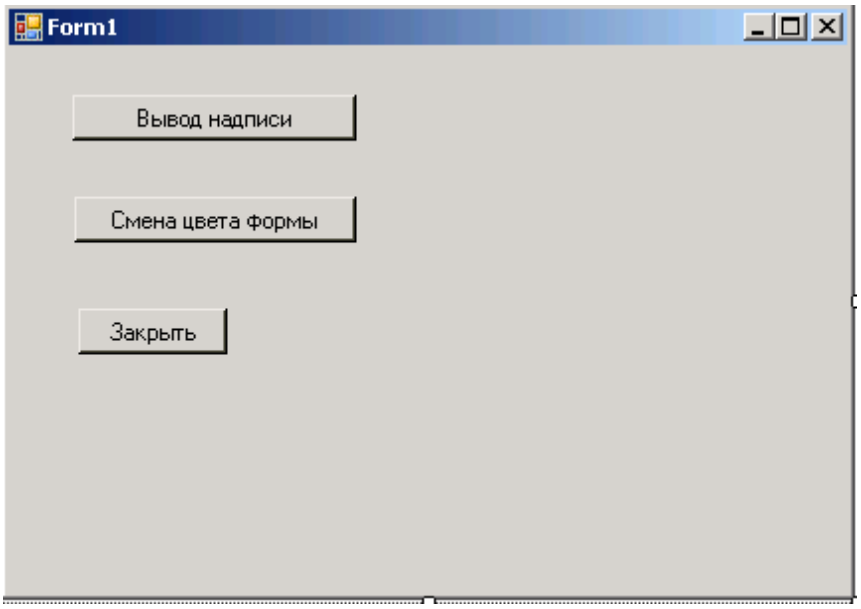


Рис. 1.4 Форма для проекта примера 1-1.

Для размещения кнопок на форме включим окошко `Панель элементов` — см. ***рис. 1.5а*** (командой ***Вид/Панель элементов*** или клавишами **Ctrl+Alt+X**). Выберем там кнопку `Button` (щелкнем по ней левой клавишей мыши) и затем щелкнем (также левой кнопкой) в нужном месте формы. Повторим это еще два раза, т.к. нужно три кнопки. Затем меняем надписи на них. Для этого щелкнем правой клавишей мыши, например, по первой кнопке и выберем пункт Свойства. При этом откроется окошко `Свойства` для данного объекта (***рис 1.5б***) и здесь свойство **Text** сменим на «Вывод надписи». Проделаем аналогичную операцию для двух других кнопок.

Рис. 1.5а. Панель инструментов Toolbox с выбранным Button

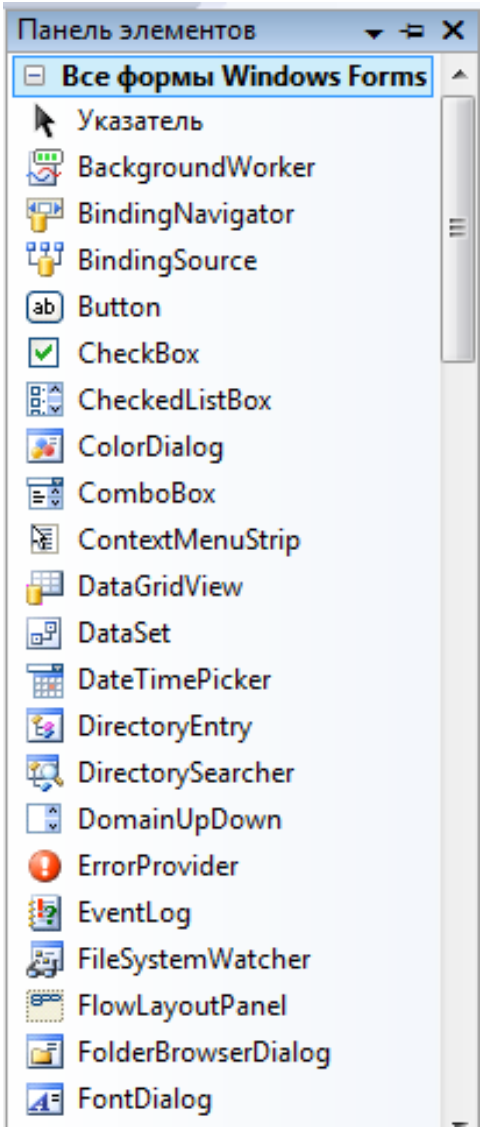
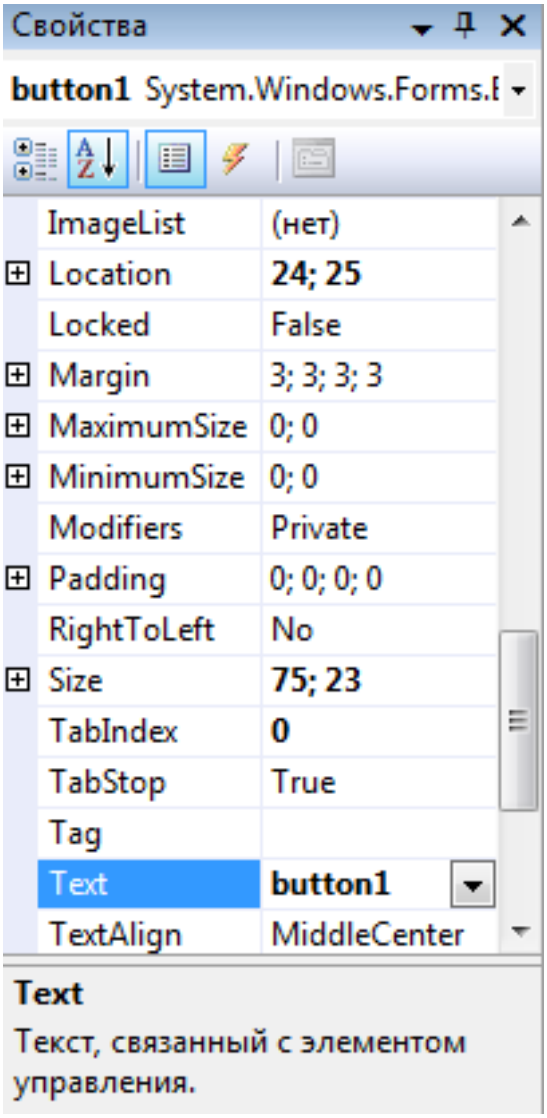


Рис. 1.5б. Панель свойств объектов Properties для кнопки Button



Затем разместим на форме объект `Label` (из `Панели элементов`) — он нужен для вывода надписи на форму. Его разместим справа от кнопки «Вывод надписи». Его свойство **Text** просто очистим.

Обратите внимание еще на одно важное свойство объекта – свойство **Name** (имя). По умолчанию у кнопок свойство **Name** будет **button1**, **button2**, **button3**. У надписи имя будет **label1**. Эти имена можно поменять, но пока в этом нет необходимости.

Теперь для каждой кнопки будем программировать *событие* (event). Для этого выделим первую кнопку и в окошке Свойства перейдем на вкладку *События* – щелкнем по значку “молния” (см. ***рис. 1.5б***) и там, в

строке **Click** (событие при щелчке мышью по данной кнопке), дважды щелкаем мышью – событие будет названо **button1Click**. Откроется окно для ввода кода программы (см. ***рис. 1.6***). Обратите внимание, что заголовок функции, а также открывающая и закрывающая скобки появились автоматически. И здесь вводим всего одну строку:

```
label1->Text="Кнопка работает!";
```

Аналогично создадим событие для кнопки «Смена цвета формы» (будем менять цвет формы на красный **Red** и обратно – на исходный **ButtonFace**):

```
if (this->BackColor == Color::Red )
      this->BackColor= SystemColors::ButtonFace;
          else this->BackColor= Color::Red;
```

И так же создадим событие для кнопки «Закрыть» :

```
this->Close();
```

Результат окна с кодом приведен на ***рис. 1.6***.

Обратите внимание, что здесь активно используются понятия объектно-ориентированного программирования. Указываются классы `Color`, `SystemColors` и используется `this` — указание на текущий активный объект (в данном случае на форму).

Теперь проект следует сохранить (***Файл/Сохранить все*** или щелкнуть по кнопке ). Для запуска программы на выполнение, нужно сначала провести компиляцию и компоновку — дать команду ***Построение/Построить решение*** (или клавиша **F7**). Если все в порядке — в окне **Вывод** должно быть «ошибок 0» (см. ***рис. 1.6***), то собственно запускаем на исполнение — клавишами **Ctrl-F5** (***Отладка/Запуск без отладки***). Если в процессе создания программы не было ошибок, то мы получим действующее приложение. Можно будет пощелкать по кнопкам и убедиться в их действии:

- при щелчке по кнопке «Вывод надписи» появится на форме надпись: Кнопка работает! (в том месте, где мы разместили объект **Label**);
- при щелчке по кнопке «Смена цвета формы» форма поменяет цвет на красный, а при повторном щелчке по этой же кнопке цвет вернется – станет серым;
- при щелчке по кнопке «Закрыть» приложение закрывается, и мы вернемся в среду.

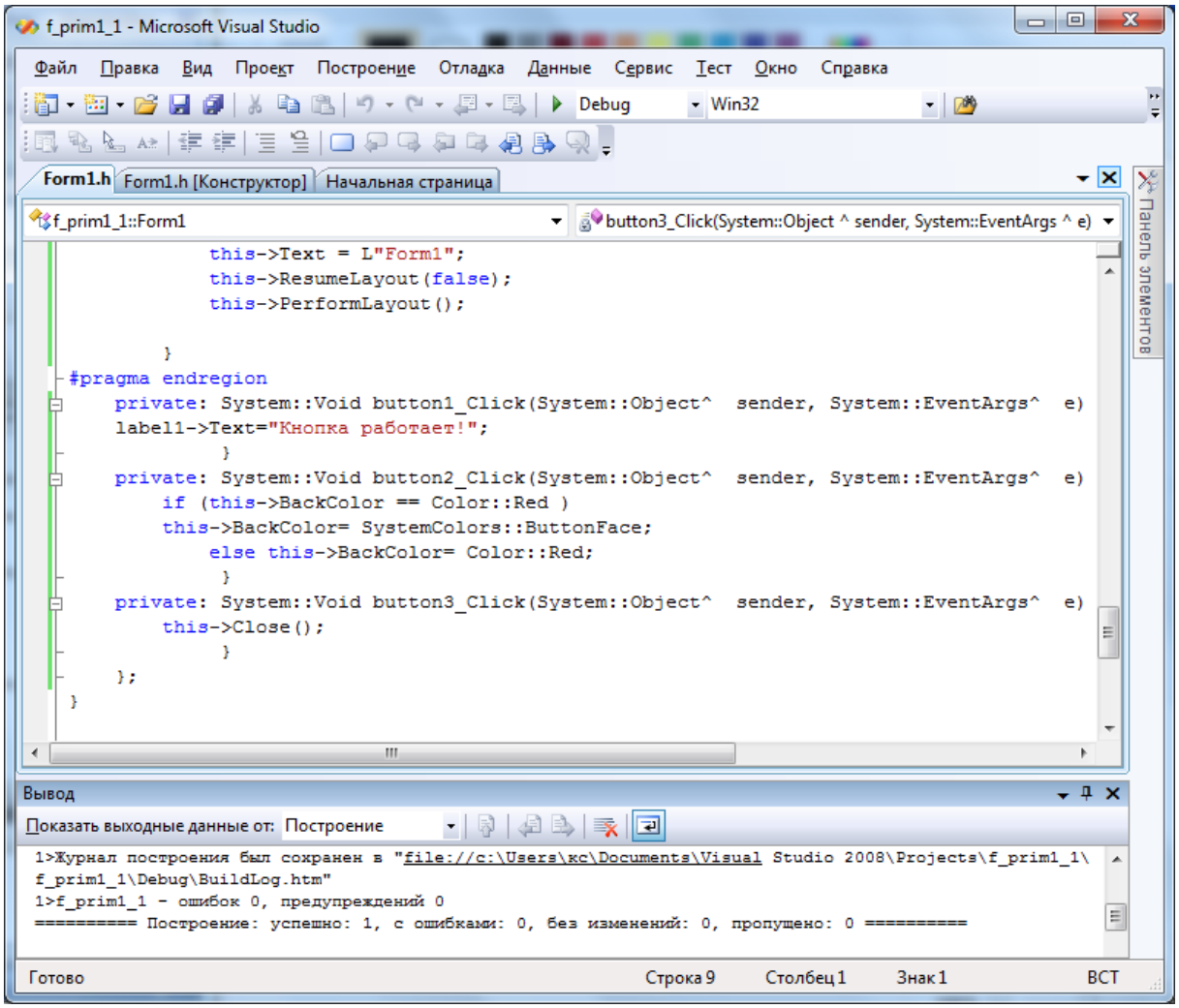


Рис. 1.6 Окно кода программы примера 1-1.

2. ОКНО ВВОДА ТЕКСТА Textbox и РАДИО-КНОПКА RadioButton. БОЛЕЕ СЛОЖНЫЙ ПРОЕКТ

Среди компонентов на *Панели элементов* есть еще две наиболее часто использующиеся – окно ввода текста **TextBox** и радиокнопка-переключатель **RadioButton**. **TextBox** позволяет вводить и редактировать текст и использовать его в процессе работы приложения. **RadioButton** предназначена для переключений в процессе работы приложения, для выбора вариантов работы приложения и т.д. У нее основное свойство – это свойство *Checked* логического типа (значение **false** или **true**).

Пример 2-1. *Составить проект для начисления зар.платы, исходя из задаваемого на форме оклада, процента надбавки и предусмотреть варианты расчета – с премией (с указанием %) или без премии*. (*Не забудем, что в конце необходимо учесть подоходный налог в 13%*) .

Вид приложения см. на ***рис. 2.1***

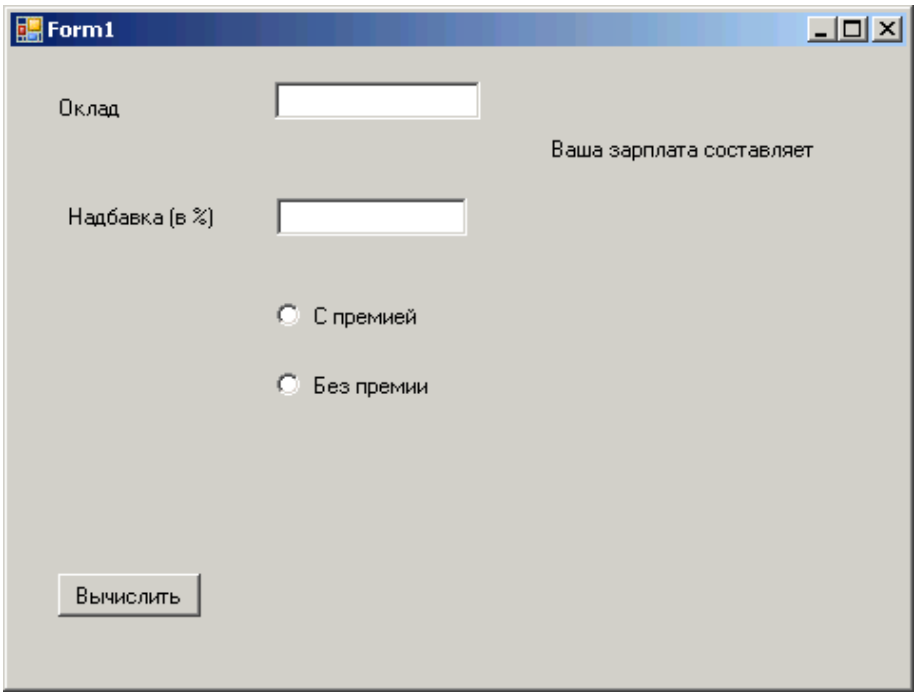


Рис. 2.1 Вид формы приложения для начисления зар.платы

Итак, создадим новый проект (***Файл/Создать/Проект***), выберем **CLR** и **Приложение Windows Forms**. Разместим на нем нужные элементы управления (см. ***рис. 2.2***).

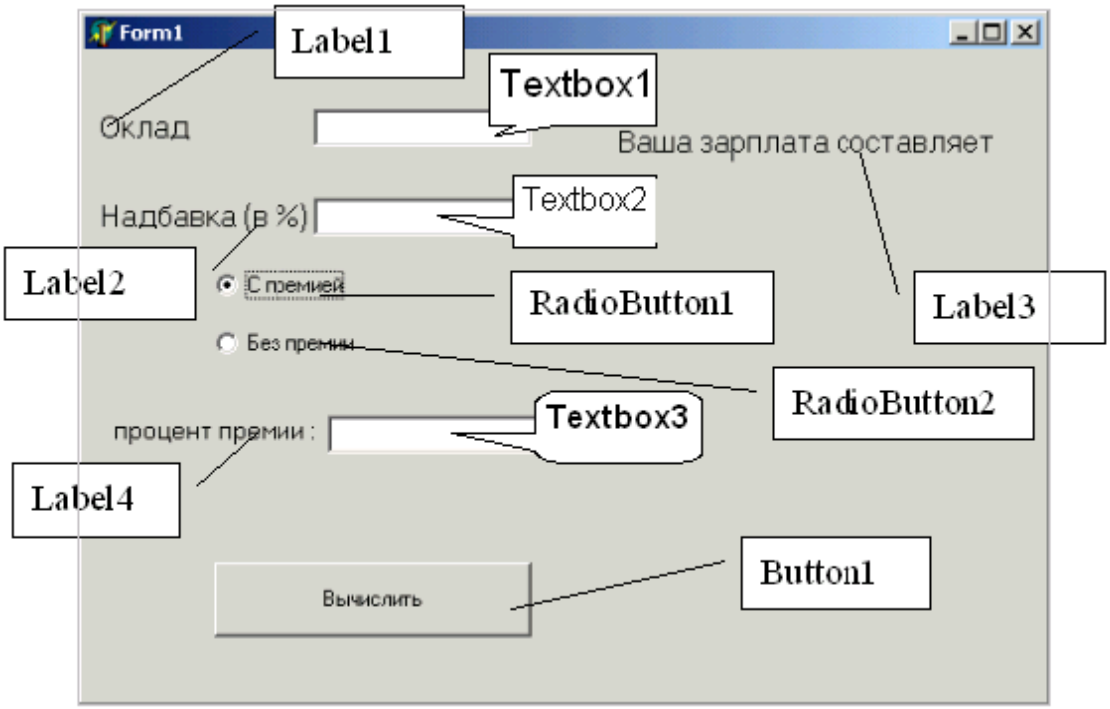


Рис. 2.2 Вид формы приложения для начисления зар.платы при наличии премии

Для объектов **Label** сменим надписи и сделаем их более крупными. Для этого выберем свойство *Font* для этих надписей (см. ***рис. 2.3а***), щелкнем там по многоточию (…), появится очередное окно (см. ***рис. 2.3б***) и там выберем больший размер шрифта (а можно выбрать и тип шрифта).

Окошки **Textbox** просто очистим – выберем у них свойство *Text* и очистим его. Поменяем надписи (*Text*) у кнопок **RadioButton** и **Button**.

Теперь приступим к созданию кода – соответствующих функций обработки событий для того или иного объекта.

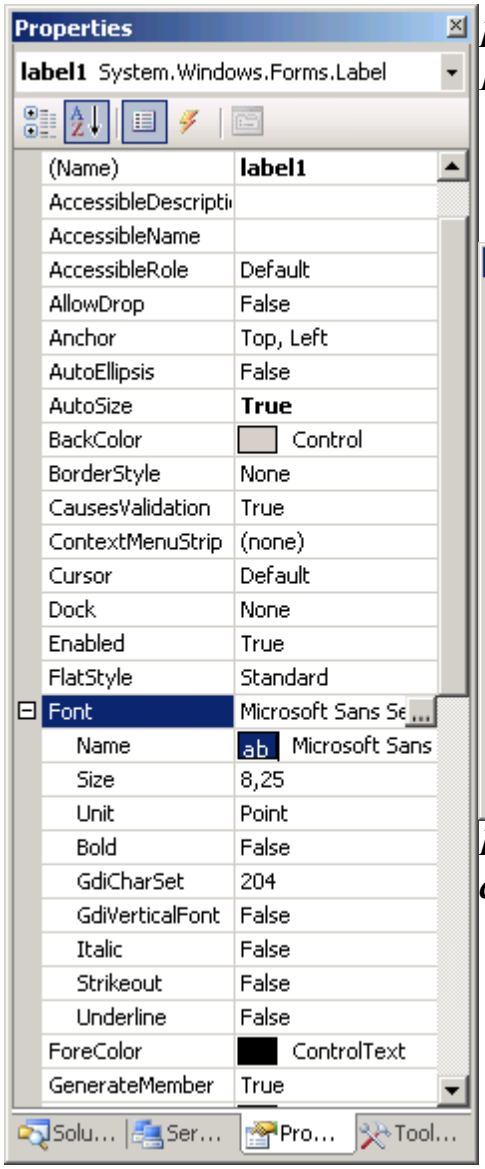


Рис. 2.3а. Свойство Font в окне Properties для объекта Label1

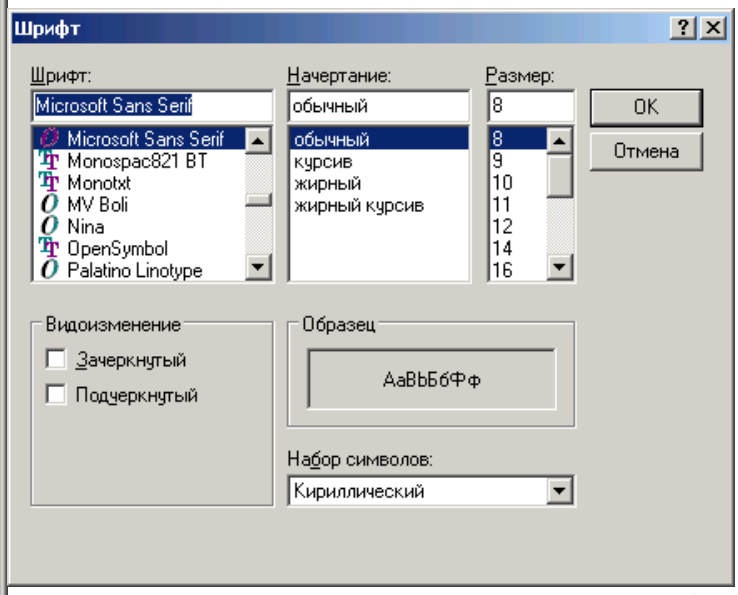


Рис. 2.3б. Выбор типа и размера шрифта (после нажатия … в строке Font)

Сначала позаботимся о том, чтобы при запуске приложения надпись «процент премии» и окошко **Textbox** для ввода этого процента премии *не были видны* (они потом появятся только при выборе радиокнопки «С премией»). Для этого создадим событие *Load* для формы – выберем форму (щелкнем по чистому месту формы) и в **Properties** перейдем на вкладку **Events** и в строке *Load* щелкнем два раза. Откроется окно кода с процедурой Form1_Load. Введем там такие строки:

```
textBox3->Visible=false;
  label4->Visible=false;
```

Здесь мы использовали свойство *Visible*, которое принимает два значения: **false** (объект невидим) и **true** (объект видим).

Помните: событие *Load* срабатывает в момент запуска приложения.

Начнем программировать событие *Click* для кнопки «Вычислить». Очевидно, что сначала нужно будет считать данные из окошек **TextBox**, а

там информация дается в текстовом виде. Поэтому нужно будет текстовую информацию (`Textbox->Text`) преобразовать в число, в данном случае в вещественное число. Делается это следующим образом:

```
float::Parse(textBox1->Text)
```

Обратно – для вывода результата на форму – нужно преобразовать вещественное число в строку с помощью метода `ToString()`.

Итак, выберем на форме кнопку и выберем событие *Click*. Введем туда следующий код программы:

```
private: System::Void button1_Click (System:: Object^
sender, System::EventArgs^  e)
  {
  float okl; // переменная для величины оклада
  float proc; // переменная для процента надбавки
  float zar; // переменная для зарплаты
  float procprem; //переменная для процента премии
  okl=float::Parse(textBox1->Text);
  proc=float::Parse(textBox2->Text);
  zar= okl+okl*proc/100 ;
  if (radioButton1->Checked)
  // если выбрано "с премией"
       { procprem=float::Parse(textBox3->Text);
         zar=zar+zar*procprem/100; }
  zar=zar - 0.13*zar;
  label3->Text=
  label3->Text+"\n"+zar.ToString()+" руб.";          }
```

Чтобы при выборе радиокнопки «С премией» появлялись надпись и окошко для ввода процента премии, нужно для **RadioButton1** создать событие *Click* вида:

```
  if (radioButton1->Checked)
         {textBox3->Visible=true;
                      label4->Visible=true; }
```

Однако если снова выберем «Без премии», то нужно чтобы надпись и окошко для ввода процента премии исчезали. Это получится, если создать событие *Click* для кнопки **RadioButton2** вида:

```
  if (radioButton2->Checked)
         {textBox3->Visible=false;
                      label4->Visible=false; }
```

Замечание. Приложения, содержащие окошко редактирования **TextBox**, следует усовершенствовать таким образом, чтобы предусмотреть «защиту от дурака» – установить ограничение на ввод данных определенного вида. Для этого используют событие *KeyPress* данного объекта **textBox**. Например, если при выполнении проекта в окошко **textBox1** должны вводится только числа, то такое событие имеет вид:

```
private: System::Void textBox1_KeyPress (System:: Object^
sender, System:: Windows::Forms:: KeyPressEventArgs^  e)
   { if (! ((e->KeyChar>='0') && (e->KeyChar<='9')
         || (e->KeyChar==',')) ) e->KeyChar=Char(0); } };
```

Обратите внимание, у функции для этого события имеется параметр `e->KeyChar`, контролируя который мы и задаем ограничение на ввод значений. В данном случае: если нажимаемая клавиша не лежит в интервале от 0 до 9 и не является десятичной запятой, то символ не отображается (Char(0)).

---

3. ДИНАМИЧЕСКИЕ ССЫЛКИ НА ОБЪЕКТЫ

3.1 Понятие о динамических ссылках.

Помимо обычных непосредственных ссылок на объекты допустимы и так называемые динамические ссылки на объекты. Обычная ссылка, например, на свойство *TabIndex* кнопки **Button** имеет вид: `Button->TabIndex`. Аналогичная динамическая ссылка имеет вид `(Button^)sender->TabIndex` . В чем преимущество (и необходимость) динамических ссылок? В том, что можно подразумевать не один объект, а целую группу объектов. В результате можно назначить одну и ту же функцию в качестве события для целой группы объектов. Продемонстрируем на примере.

Пример 3-1. *Организовать форму с семью кнопками, в зависимости от нажатия на которые будет меняться цвет формы (см. рис. 3.1). Причем для реализации события Click для всех кнопок использовать одну и ту же процедуру.*

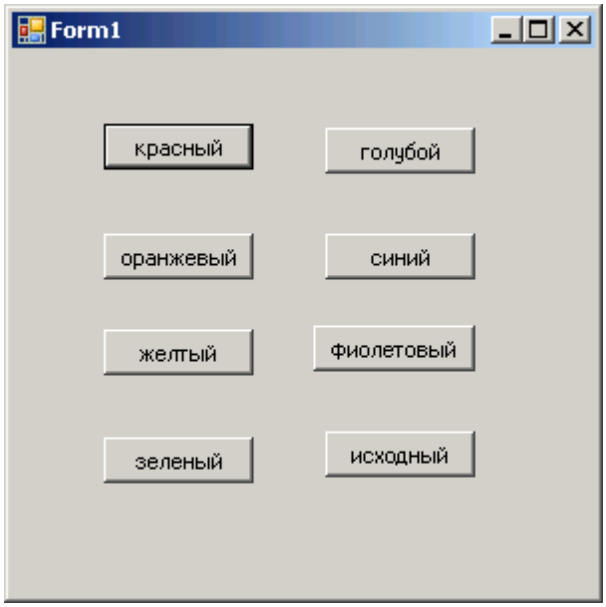


Рис. 3.1.

Итак, здесь потребуется разместить семь кнопок, сменить их названия и, главное, у каждой из них указать свой номер и свойства *TabIndex*: для кнопки «красный» зададим 1, для кнопки «оранжевый» 2 и т.д. Затем создадим событие для кнопки «красный» следующего вида:

```
switch (((Button^ )sender)->TabIndex)
   {     case 0:    this->BackColor=Color::Red; break;
         case 1: this->BackColor=Color::Orange; break;
         case 2: this->BackColor=Color::Yellow; break;
         case 3: this->BackColor=Color::Green; break;
         case 4: this->BackColor=Color::SkyBlue; break;
         case 5: this->BackColor=Color::Blue; break;
         case 6: this->BackColor=Color::Violet; break;
         case 7: this->BackColor=SystemColors::Control; break;
          }
```

Далее в окне *Свойства* перейдем на вкладку *События* и укажем там, в строке *Click* (везде, для каждой кнопки!) ту же процедуру **button1Click**.

Остается запустить приложение (естественно, при этом сохранить проект и программу) и проверить кнопки в действии.

3.2 Программа «Калькулятор»

Пример 3*-2. Создать программу «Калькулятор вида (****рис. 3.2****). Обеспечить выполнение нужных операций с отображением задаваемых цифр и результата на индикаторе.*

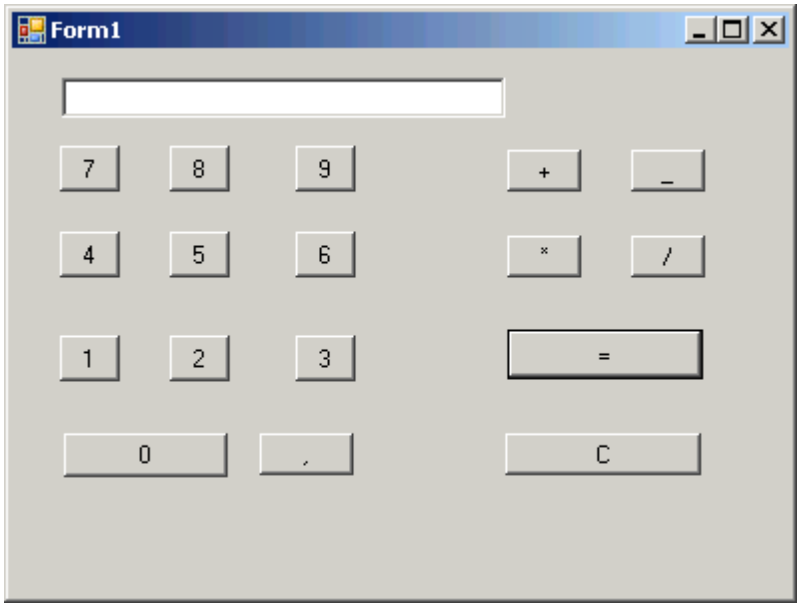


Рис. 3.2

Разместим соответствующие цифровые кнопки. Выполним надписи. Для каждой из них укажем свойство *TabIndex*, равным цифре на кнопке.

В окне кода после строк:

```
public ref class Form1 : public System::Windows:: Forms:: Form
      {     public:
```

опишем глобальные переменные:

```
            int f;   // признак для проверки
      // f=0 – вводится первый операнд операции
      // f=1 - вводится второй
   int oper;  // код операции:
      //0 - "=", 1- "+", 2 - "-", 3 - "*", 4 -"/"
   float accum; // результат операций
```

Создадим для кнопки «1» для события *Click* функцию следующего вида:

```
   private: System::Void button9_Click(System::Object^  sender, System::EventArgs^  e)
      // функция отображения символа цифры
   {   Char Ch;  // символ цифры
    // преобразование цифры в ее символ
```

```
 // т.е. код 0 есть 48, код 1 есть 49 и т.д.
Ch=Char(((Button^)sender)->TabIndex+48);
switch ( ((Button^)sender)->TabIndex)
 {case 0:
 case 1:
 case 2:
 case 3:
 case 4:
 case 5:
 case 6:
 case 7:
 case 8:
 case 9: if (f = =0) {textBox1->Text=Ch.ToString();f=1;}
     else textBox1->Text=textBox1->Text+Ch.ToString(); }
 f=1;  }
```

Обратите внимание, что мы сразу предусмотрели и все другие цифры. Поэтому можем назначить эту же процедуру и всем остальным цифровым кнопкам, т.е. перейдем на вкладку *События* и, последовательно выделяя каждую кнопку, событию *Click* для нее назначим ту же функцию `button1Click`.

Аналогично разместим кнопки для операций и для кнопки очистки **C**. Для кнопок операций также укажем свойство *TabIndex* следующим образом: 0 – для “=”, 1 – “+”, 2 – “-“, 3 – “*”, 4 – “/”.

Создадим процедуру, например, *для кнопки “+”:*

```
private: System::Void button12_Click (System::Object^  sender, System::EventArgs^  e)
  { // функция выполнения операции
     float numb;
     numb=(float::Parse)(textBox1->Text);
     if (f==0) oper=((Button^)sender)->TabIndex;
     else
     {  switch(oper)
        { case 0: accum=numb; break;
          case 1: accum=accum+numb; break;
          case 2: accum=accum-numb; break;
          case 3: accum=accum*numb; break;
          case 4: accum=accum/numb; break;
          }
          oper=((Button^)sender)->TabIndex;
          f=0;
          textBox1->Text=(accum).ToString();
       }
  }
```

И также назначим эту функцию для остальных кнопок с операциями.

Создадим отдельно функцию для *кнопки "C":*

```
private: System::Void button17_Click(System::Object^  sender,
System::EventArgs^  e)
{     textBox1->Text="0"; oper=0; accum=0;   }
```

И создадим для кнопки "," такую функцию:

```
private: System::Void button11_Click(System::Object^  sender,
System::EventArgs^  e)
{    textBox1->Text=textBox1->Text+",";      }
```

Кроме того, создадим функцию, срабатывающую в момент создания формы:

```
private:  System::Void  Form1_Load(System::Object^    sender,
System::EventArgs^  e)
                {    textBox1->Text="";
                  oper=0;           }
```

Работающий калькулятор готов!

4. ИСПОЛЬЗОВАНИЕ ТАЙМЕРА. КОМПОНЕНТ CHECK-BOX

4.1 Таймер

До сих пор рассматривали компоненты, которые непосредственно видны на форме и выполняют видимую роль.

Однако, есть еще и компоненты, которые выполняют свою работу, не будучи видными, для постороннего глаза. Одним из таких компонентов является **Timer (таймер)**. В его задачу входит обрабатывать какое-либо сообщение через определенные интервалы времени. Хотя этот компонент и невидим в процессе работы программы, у него тоже есть свои события и свойства. Вернее, событие одно – **Tick**, которое определяет, что должна делать программа, когда истечет заданный интервал.

Самое важное свойство таймера — **Interval** (Интервал). Оно указывает, когда (через какое время) в следующий раз таймер должен сработать. Промежуток времени задается в тысячных долях секунды — миллисекундах. По умолчанию, свойство **Interval** содержит число 1000, следовательно, таймер будет срабатывать каждую секунду.

Рассмотрим пример.

Пример 4.1 *Создать программу-игру «Прыгающая кнопка»* ***(рис.4.1)****: кнопка будет прыгать по поверхности формы с заданным интервалом и следует попасть по ней (щелкнуть на ней мышью). Предусмотреть возможность ускорить и замедлить движение кнопки.*

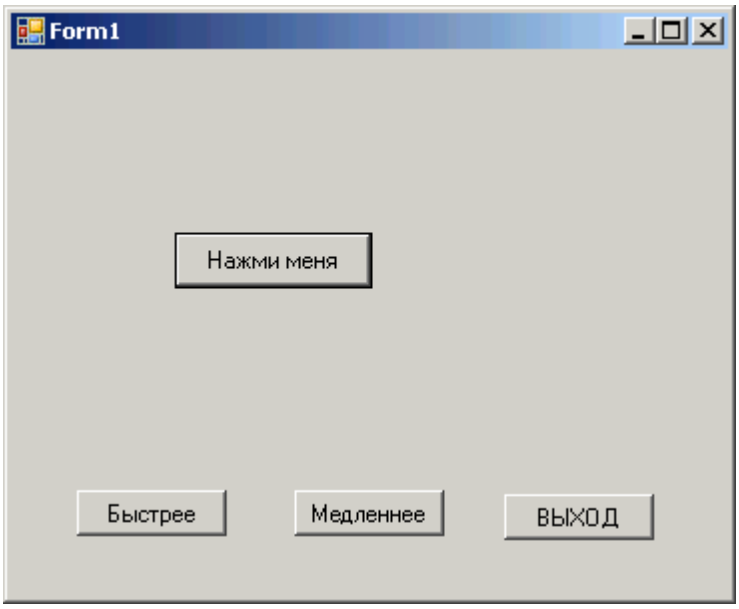


Рис. 4.1

Итак, создадим новый проект и разместим, прежде всего, объект **Timer**. Обратите внимание, что он разместится не на самой форме, а сразу под ней. Поместим также нужное число кнопок и сменим на них надписи. Давайте сделаем так, чтобы через каждые пол-секунды кнопка прыгала куда-нибудь. А задача пользователя – ее поймать. Свойство **Interval** объекта **Timer** сделайте равным 500.

Затем дважды щелкните по значку **Timer**, чтобы открыть обработчик события. Начинаем писать код:

```
private: System::Void timer1_Tick(System::Object^   sender,
System::EventArgs^  e)
                {int x,y;
           // кнопка может прыгать по всей длине формы
        x=rand()%((this->Size.Width)-100);;
        // кнопка может прыгать по всей ширине формы
        y=rand()%(this->Size.Height-100);
           button1->Location=Point(x, y);
                          }
```

Обратите внимание, что положение кнопки на форме (координаты ее левой верхней точки *x*,*y*) задаются с помощью датчика случайных чисел. Дополнительный сдвиг на 100 предусмотрели, чтобы кнопка своими краями не выскакивала за пределы формы.

Когда нужно закончить игру? Когда пользователь щелкнет по кнопке. То есть для кнопки произойдет событие **Click**. Его и будем программи-

ровать. Щелкните два раза по кнопке **button1**, чтобы создать процедуру обработки щелчка. Введите следующий код:

```
private: System::Void button1_Click(System::Object^  sender,
System::EventArgs^  e)
          {   timer1->Enabled=false;
              button1->Text="ÓÐÀ!" ;
          }
```

Осталось доработать кнопку **Выход**, а также кнопку **Медленнее** и кнопку **Быстрее**, при нажатии на которые увеличивается или уменьшается интервал таймера. Функции события **Click** для них имеют вид:

```
private: System::Void button2_Click(System::Object^  sender,
System::EventArgs^  e)
          { //кнопка «Быстрее»
              timer1->Interval=timer1->Interval/2;
          }
private: System::Void button3_Click(System::Object^  sender,
System::EventArgs^  e)
          { //кнопка «Медленнее»
              timer1->Interval=timer1->Interval*2;
          }
private: System::Void button4_Click(System::Object^  sender,
System::EventArgs^  e)
          { //кнопка «Выход»
              this->Close();      }
```

И остается проверить в работе.

Пример 4.2. *Создадим электронные часы, на которых будет отображаться текущее время, дата и день недели (**рис. 4.2**).*

Создадим новый проект и разместим на форме два объекта **Label**. сразу зададим шрифт побольше для этих надписей (например, 24 кегля). Для этого, напомним, нужно выделить этот объект и в окне Инспектора объектов выбрать свойство Font, щелкнуть там по многоточию и в возникшем окне выбрать тип и размер шрифта. Поместим также объект **Timer**.

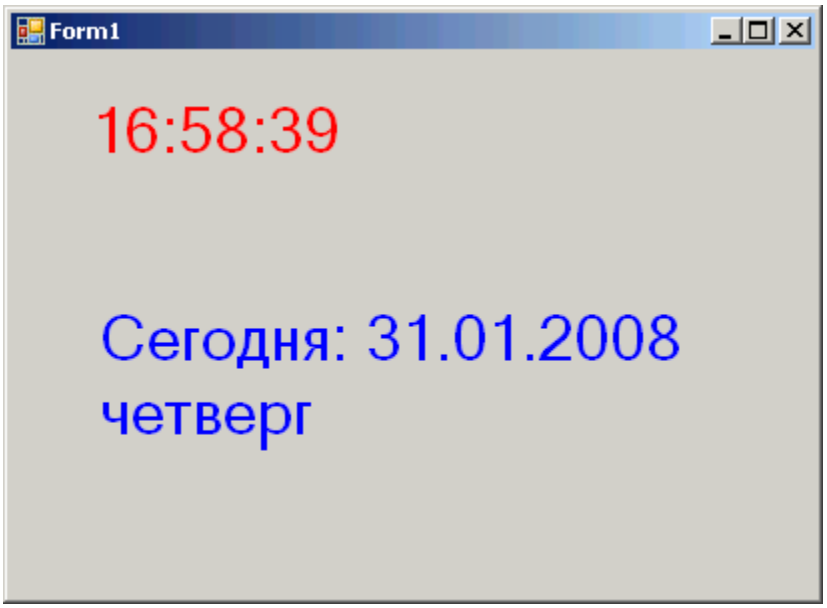


Рис. 4.2

Далее нужно создать события. Одно событие для формы **Form_Load** (выбрать форму и на вкладке *Events* выбрать Load):

```
private:  System::Void  Form1_Load(System::Object^    sender,
System::EventArgs^  e)
{ int dn; String^ named;
timer1->Interval = 1000;//период сигналов таймера 1с
  timer1->Enabled = true; // пуск таймера
  //получить дату
label2->Text="Сегодня: " + DateTime::Now.Today.ToShortDateString() ;
  // вывести день недели
dn= (int)DateTime::Now.DayOfWeek ;
  switch ( dn)
  {case 0: named="воскресенье"; break;
   case 1: named="понедельник"; break;
   case 2: named="вторник";    break;
   case 3: named="среда";     break;
   case 4: named="четверг";   break;
   case 5: named="пятница";  break;
   case 6: named="суббота";  break;
   }
  label2->Text+=" \n"+named;            }
```

Второе событие создадим для **Timer** — событие Tick:

```
private:  System::Void  timer1_Tick(System::Object^    sender,
System::EventArgs^  e)
     { // получить системное время
     label1->Text = DateTime::Now.ToLongTimeString();
     }
```

Обратите внимание, что для получения даты и времени служит класс `DateTime` и его элемент `Now`. Для получения сегодняшней даты применяют метод `Today`. А для перевода даты в строку (причем в сокращенном представлении даты) применяют метод `ToShortDateString()`. Для перевода времени в строку используют метод `ToLongTimeString()` Для получения номера дня недели применяют метод `DayOfWeek`, он дает номер дня недели в «американском» виде — у них счет дней недели идет с воскресенья:
0 — воскресенье, 1 — понедельник и т.д. Далее, используя структуру множественной развилки, получаем название дня недели.

4.2 Компонент CheckBox

Данный компонент (а это просто окошко, в котором ставят «галочку») удобно использовать в том случае, когда требуется ответить: да или нет. У этого компонента есть свойство **Checked**, которое принимает значение **true** (истина) или **false** (ложь) в зависимости от того стоит галочка на этом компоненте (**true**) или нет (**false**).

Пример 4.3. *Создать приложение вида (**рис. 4.3**).В нем кнопка «Нажми меня» служит для закрытия данного окна, но при условии, что стоит «галочка» напротив «Разрешить закрытие программы». Если же поставить галочку напротив «Отключить кнопку», то кнопка вовсе становится недоступной.*

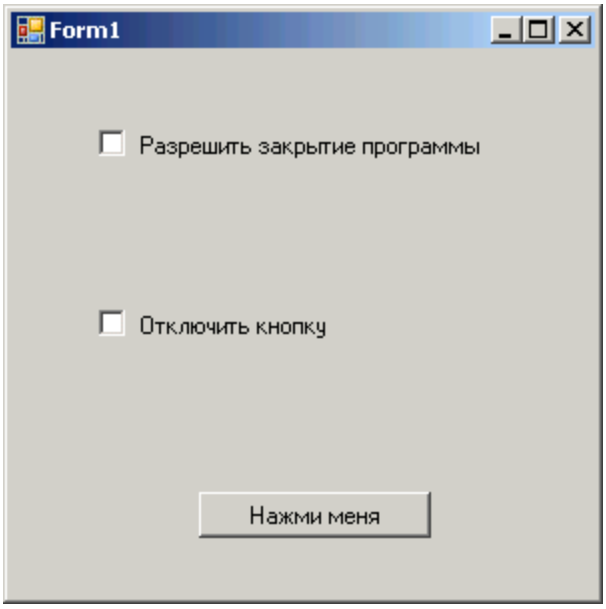


Рис. 4.3

Создадим новое приложение и оформим такие функции:

```
private: System::Void button1_Click(System::Object^  sender,
System::EventArgs^  e)
{// кнопка «Нажми меня»
```

```
    if (checkBox1->Checked) this->Close();      }
private:    System::Void    checkBox2_CheckedChanged    (Sys-
tem::Object^  sender, System::EventArgs^  e)
    { // «Отключить кнопку»
            button1->Enabled=!checkBox2->Checked;   }
```

Пример 4.4. *Создать приложение вида (**рис. 4.4**). Такое приложение позволяет рассчитать затраты для поездки на дачу на автомобиле. Как видим, нужно будет вводить расстояние и цену на бензин, а также потребление бензина (на 100 км пути). Здесь же потребуется использовать компонент **CheckBox**, а значит – если там стоит «галочка» – полученные затраты будут удваиваться.*

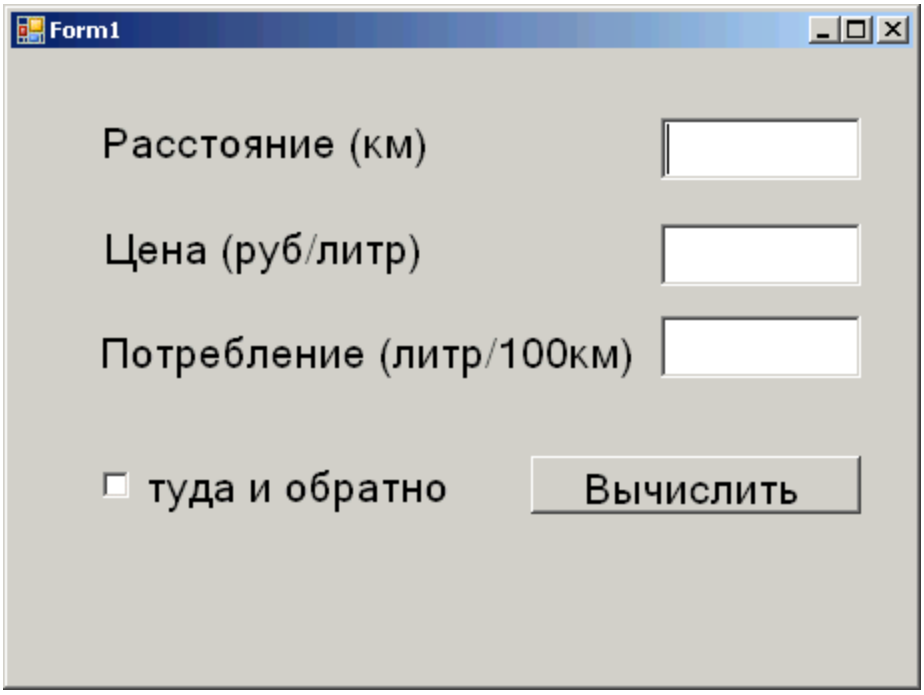


Рис. 4.4

Здесь нужно, по сути, разработать только функцию для щелчка по кнопке «Вычислить» :

```
private: System::Void button1_Click(System::Object^  sender,
System::EventArgs^  e)
    {  // кнопка "Вычислить"
    float r,c,p,stoim;
    r=float::Parse(textBox1->Text)  ;
    c=float::Parse(textBox2->Text);
    p=float::Parse(textBox3->Text);
    stoim=r/100*p*c;
    if (checkBox1->Checked)  stoim=stoim*2;
    label4->Text="Стоимость вашей поездки на дачу \n состав-
ляет:" + (stoim).ToString()+" рублей";
```

```
                    }
```

5. СПИСКИ ВЫБОРА И ПОЛОСЫ ПРОКРУТКИ. ГРАФИЧЕСКИЕ КОМПОНЕНТЫ.

5.1 Список выбора ListBox

Этот компонент находится на вкладке *Standart*. Используется для выбора одного или нескольких параметров из списка. Доступ к строкам списка – через свойство *Items* строкового типа *TString*, при этом номер строки определяется свойством *ItemIndex* (счет идет от нуля, т.е. первый элемент списка имеет нулевой индекс).

Пример 5-1. *Создать приложение со списком цветов. При выборе каждого из них – в поле* ***Edit*** *появляется имя выбранного цвета и меняется цвет формы на выбранный (см.* ***рис. 5.1****).*

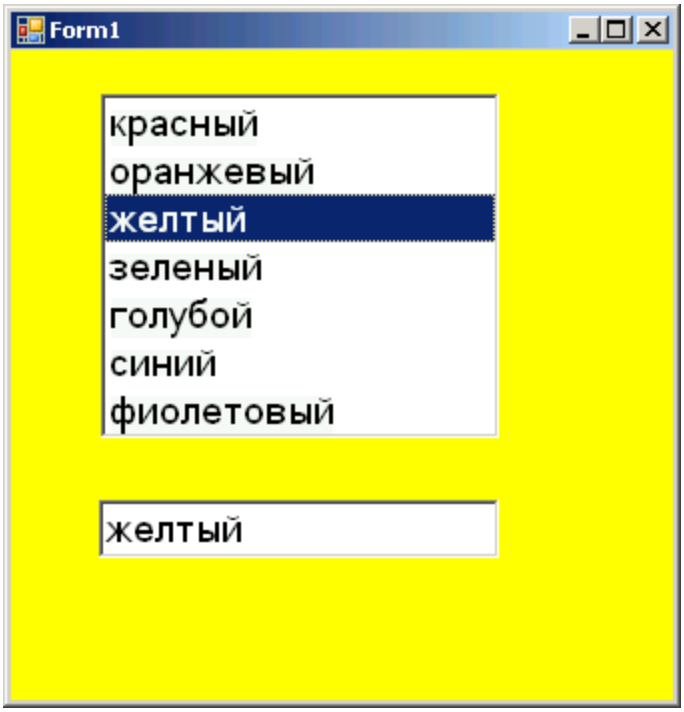


Рис. 5.1

Разместим **ListBox** и **TextBox**. Дважды щелкаем на свойстве *Items* компонента **ListBox** и вводим список цветов: красный, оранжевый…

Создадим событие `SelectedIndexChanged` для него:

```
private:  System::Void  listBox1_SelectedIndexChanged  (Sys-
tem::Object^  sender, System::EventArgs^  e)
{textBox1->Text=listBox1->SelectedItem->ToString() ;
                    //Strings[listBox1->SelectedIndex];
switch ( listBox1->SelectedIndex )
```

```
{case   0: this->BackColor=Color::Red; break;
 case    1: this->BackColor=Color::Orange; break;
 case    2: this->BackColor=Color::Yellow; break;
 case    3: this->BackColor=Color::Green; break;
 case    4: this->BackColor=Color::SkyBlue; break;
 case    5: this->BackColor=Color::Blue; break;
 case    6: this->BackColor=Color::Violet; break;}   }
```

5.2 Полосы прокрутки

Полосы прокрутки **VScrollBar (**вертикальная полоса прокрутки) и **HScrollBar (**горизонтальная полоса прокрутки) используются для прокручивания информации в окне. Значение свойства полосы прокрутки *Value* меняется (по умолчанию) от 0 (Min) до 100 (Max), но может быть изменено пользователем.

Пример 5-2. *Создать приложение с двумя полосами прокрутки с выбором чисел от 1 до 10 и вычислением суммы и произведения выбранных таким образом чисел (см. рис. 5.2).*

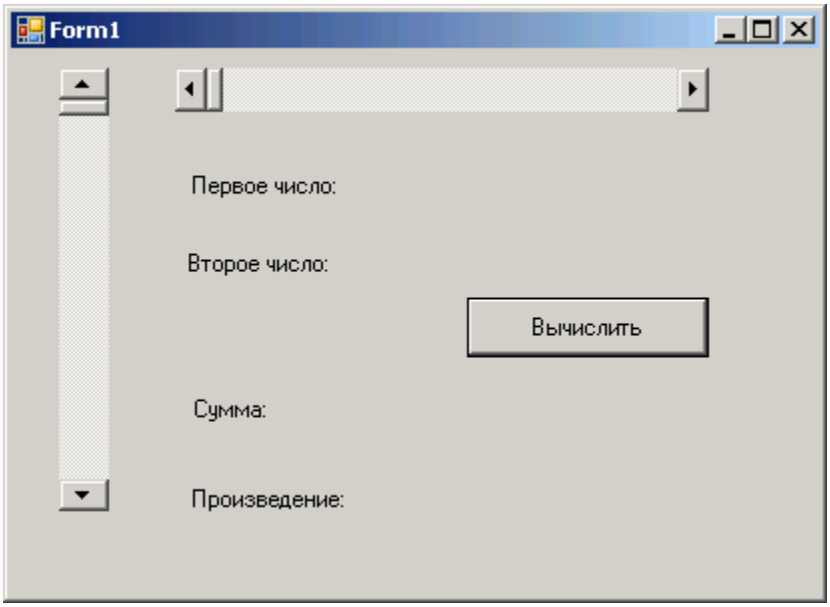


Рис. 5.2

Разместим на форме две полосы: горизонтальную и вертикальную. Зададим у каждой из них *Min* равным 1, а *Max* равным 10. Рекомендуется также изменить свойство *LargeChange* (сделать равным 1). Разместим также четыре надписи **Label**. Создадим для первой полосы событие `Scroll`:

```
private: System::Void hScrollBar1_Scroll(System::Object^  sender, System::Windows::Forms::ScrollEventArgs^  e)
{     label1->Text=
      "Первое число: "+(hScrollBar1->Value).ToString();        }
```

Аналогично создаем функцию события Scroll и для второй полосы прокрутки:

```
private: System::Void vScrollBar1_Scroll(System::Object^  sender, System::Windows::Forms::ScrollEventArgs^  e)
{ label2->Text=
  "Второе число: "+(vScrollBar1->Value).ToString();      }
```

И для кнопки «Вычислить» событие **Click** :

```
private: System::Void button1_Click(System::Object^  sender, System::EventArgs^  e)
{ int a,b,s,p;
  a=hScrollBar1->Value;
  b=vScrollBar1->Value;
  s=a+b;
  p=a*b;
  label3->Text="Сумма = " + s.ToString();
  label4->Text="Произведение = " + p.ToString();          }
```

Остается проверить приложение в работе.

5.3 Графика

Для рисования графических объектов на форме служит событие для формы Paint. При использовании этого события обычно следует указать

- включение графического режима командой вида:
  e = this->CreateGraphics()
- настроить «карандаш» для рисования Pen (его цвет и толщину),
- при рисовании закрашенных фигур указывают тип закраски SolidBrush и ее цвет,
- при выводе текста в это графическое окно настраивают шрифт и т.п.

Для собственно вывода графических объектов служат специальные методы. Например:

- для рисования линии:
  e->DrawLine(myPen, 0, 0, 200, 300);
- для рисования эллипса:
  e->DrawEllipse(myPen,Rectangle(0,0,200, 300));
- для рисования прямоугольника:
  e->DrawRectangle(myPen, Rectangle(0, 0, 200, 300));
- для рисования закрашенного прямоугольника:
  e->FillRectangle(myBrush, Rectangle(0, 0, 200, 300));
- для вывода текста на рисунке:

```
e->DrawString(s,myFont,myBrush, 10, 10);
```

Здесь myPen – заданный вид карандаша, myBrush – заданный вид закраски, Rectangle – заданный прямоугольник (первые два числа – координаты левой верхней точки прямоугольника на форме, вторые — это его длина и высота). При выводе текста s — содержит строку текста, myFont — заданный тип шрифта.

Пример 5-3. *Нарисовать флаг России и сделать снизу надпись (см.* ***рис. 5.3****).*

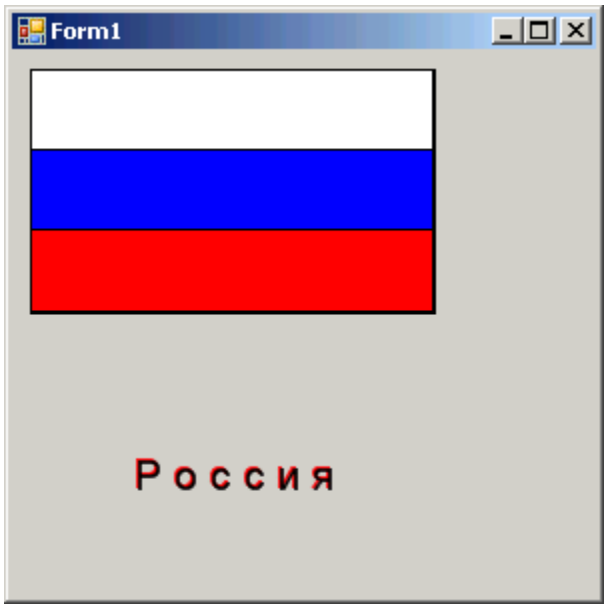


Рис. 5.3

```
// ФЛАГ РОССИИ
     private: System::Void Form1_Paint(System::Object^  sender, Sys-
tem::Windows::Forms::PaintEventArgs^  e)
                  {
Pen^ myPen = gcnew Pen(Color::Black,3);// цвет и толщина пера
SolidBrush^ myBrush =
     gcnew SolidBrush(Color::Black); //цвет закраски
     System::Drawing::Font^ myFont =
gcnew System::Drawing::Font("Arial",16);//тип и размер шрифта
           String^ myString= " Р о с с и я ";
           int x=10, y=10, //начальная точка
           L=200, H=40; //ширина флага и высота полосы
// задание размера прямоугольника (одной полосы флага)
                 Rectangle rect = Rectangle(x,y,L,H);
     //рисуем белую полосу флага
```

```
                e->Graphics->DrawRectangle(myPen,rect);
                myBrush->Color=Color::White;
                e->Graphics->FillRectangle(myBrush, rect);
    //рисуем синюю полосу флага
                rect = Rectangle(x,y+H,L,H);
                e->Graphics->DrawRectangle(myPen,rect);
                myBrush->Color=Color::Blue;
                e->Graphics->FillRectangle(myBrush, rect);
    //рисуем красную полосу флага
                rect = Rectangle(x,y+2*H,L,H);
                e->Graphics->DrawRectangle(myPen,rect);
                myBrush->Color=Color::Red;
                e->Graphics->FillRectangle(myBrush, rect);
    //вывод надписи(сначала красным, потом черным цветом)
e->Graphics->DrawString(myString, myFont, myBrush,50,200);
                myBrush->Color=Color::Black;
e->Graphics->DrawString(myString, myFont, myBrush,51,201);
                 }
```

6. РАБОТА С ТЕКСТОВЫМИ ФАЙЛАМИ.

6.1 Чтение и запись текстового файла

В среде Visual C++ (Windows Forms) имеется возможность обращаться к внешним файлам: отображать информацию из файла на форме, помещать информацию с формы в файл и т.п. Мы рассмотрим работу с текстовыми файлами в простейшем варианте – с помощью специальных компонентов.

Для работы с файлами будем использовать окна диалога. В Windows Forms имеются специальные элементы управления, организующие диалоги именно для открытия и сохранения файлов: `openFileDialog` и `saveFileDialog`.

Пример 6-1. *Создать приложение, которое выводит в поле richTextBox, размещенное на форме, содержимое текстового файла, а также позволяет отредактировать этот текст (в том числе и изменить шрифт) и сохранить в файл. (см.* ***рис. 6.1****).*

При реализации этого проекта попутно освоим работу по созданию *меню* на форме и работу с *окнами диалога*.

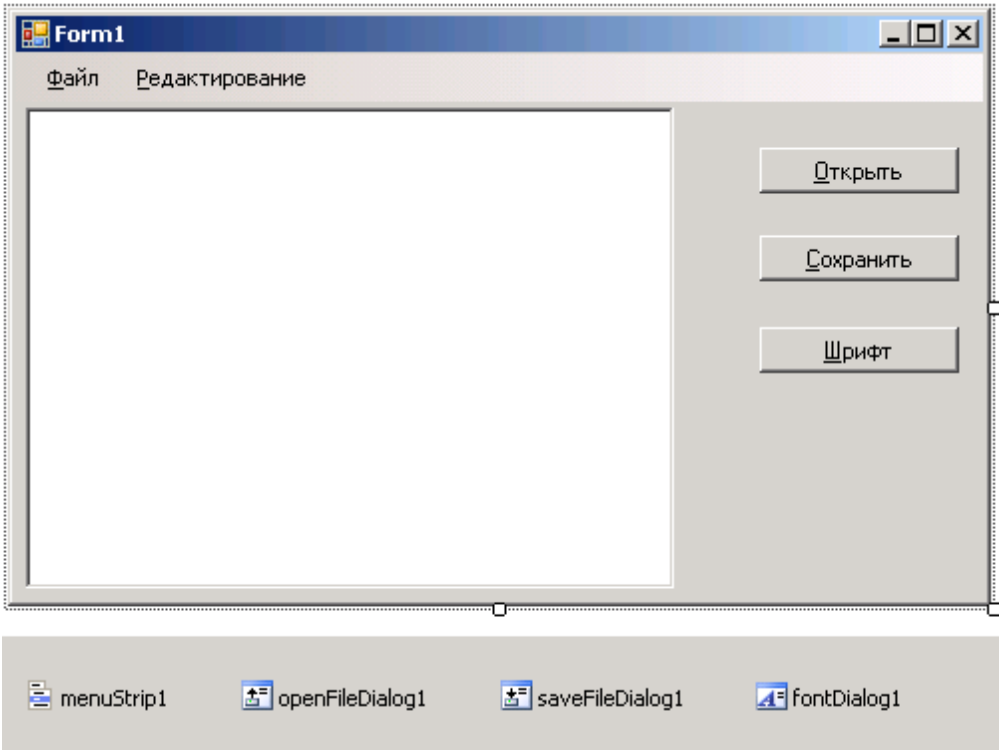


Рис. 6.1

Как видим, нам нужно разместить элемент управления `richTextBox` (куда будет помещаться текст из файла и где его будем редактировать), три кнопки: `Открыть` (файл), `Сохранить` (файл), `Шрифт` (изменить шрифт текста). Разместив эти элементы, сразу их настроим: сменим надписи на кнопках (свойство `Text` у каждой кнопки), поле `richTextBox` растянем на всю оставшуюся часть формы. Отметим, что содержимое этого поля хранится в свойстве `Lines`, которое имеет вид массива:
первая строка в окошке `richTextBox` может быть получена так:
`richTextBox1->Lines[0]`,
вторая так: `richTextBox1->Lines[1]`, т.е. обратите внимание, что нумерация строк идет с нуля.

Кроме того, разместим элементы: `menuStrip` — для организации меню на форме, `openFileDialog`, `saveFileDialog` — для организации диалогов при открытии и сохранении файлов, а также `fontDialog` — для организации диалога при смене шрифта. Мы эти элементы, как обычно, перетаскиваем на форму, но они будут находиться внизу, под самой формой (см. ***рис. 6.1***). Для окон диалога проведем настройку. Выберем `openFileDialog1` на форме и у свойства Filter установим **richTextFile|*.rtf.** Аналогично сделаем и у `saveFileDialog1`. В результате в окне диалога будут высвечиваться только файлы указанного типа. Обратим внимание, что данное окно `richTextBox1` предназначено для работы именно с файлами типа **rtf**.

Далее нужно настроить *меню*: задать его пункты и подпункты. Для этого щелкнем по значку `menuStrip1` и вводим меню вида:

&Файл
    &Открыть
    &Закрыть
&Редактирование
&Шрифт
&Очистка

Значок «&» здесь помещаем для того, чтобы можно было вызывать соответствующий пункт с помощью клавиши на клавиатуре. В данном случае, пункт «Файл» будет вызываться клавишей «Ф», подпункт «Открыть» клавишей «О» и т.д.

Теперь разработаем события для всех этих пунктов. Дважды щелкаем по подпункту “Открыть” и оформляем событие вида:

```
private: System::Void открытьToolStripMenuItem_Click (System::Object^ sender, System::EventArgs^ e)
{openFileDialog1->ShowDialog();
richTextBox1->LoadFile(openFileDialog1->FileName);}
```

Аналогично создадим событие и для пункта «Сохранить» :

```
private: System::Void сохранитьToolStripMenuItem_Click (System::Object^ sender, System::EventArgs^ e)
{saveFileDialog1->ShowDialog();
richTextBox1->SaveFile(saveFileDialog1->FileName); }
```

Для подпункта «Шрифт» событие имеет вид:

```
private: System::Void шрифтToolStripMenuItem_Click (System::Object^ sender, System::EventArgs^ e)
{fontDialog1->ShowDialog();
richTextBox1->Font=fontDialog1->Font; }
```

Для подпункта «Очистить» :

```
private:System::Void очиститьToolStripMenuItem_Click (System::Object^ sender, System::EventArgs^ e)
        {richTextBox1->Clear();      }
```

Заметим, что на форме имеются кнопки с аналогичными действиями. Поэтому кнопке «Открыть» назначим то же событие `открытьToolStripMenuItem_Click`. Напомним, что для этого достаточно выделив эту кнопку, перейти в окошко `Properties` и переключится на вкладку `Events` (щелкнуть по значку «молния»), а там, в строке **Click,** щелкаем и выбираем нужное событие.

Аналогично кнопке «Сохранить» назначим событие `сохранитьToolStripMenuItem_Click` , а кнопке «Шрифт» событие `шрифтToolStripMenuItem_Click`.

Проект готов. Остается проверить в работе. Как обычно, сохраняем его, проводим сборку (Build) и отправляем на выполнение.

После выбора пункта «Файл» и подпункта «Открыть» появится окно диалога вида (***рис. 6.2***).

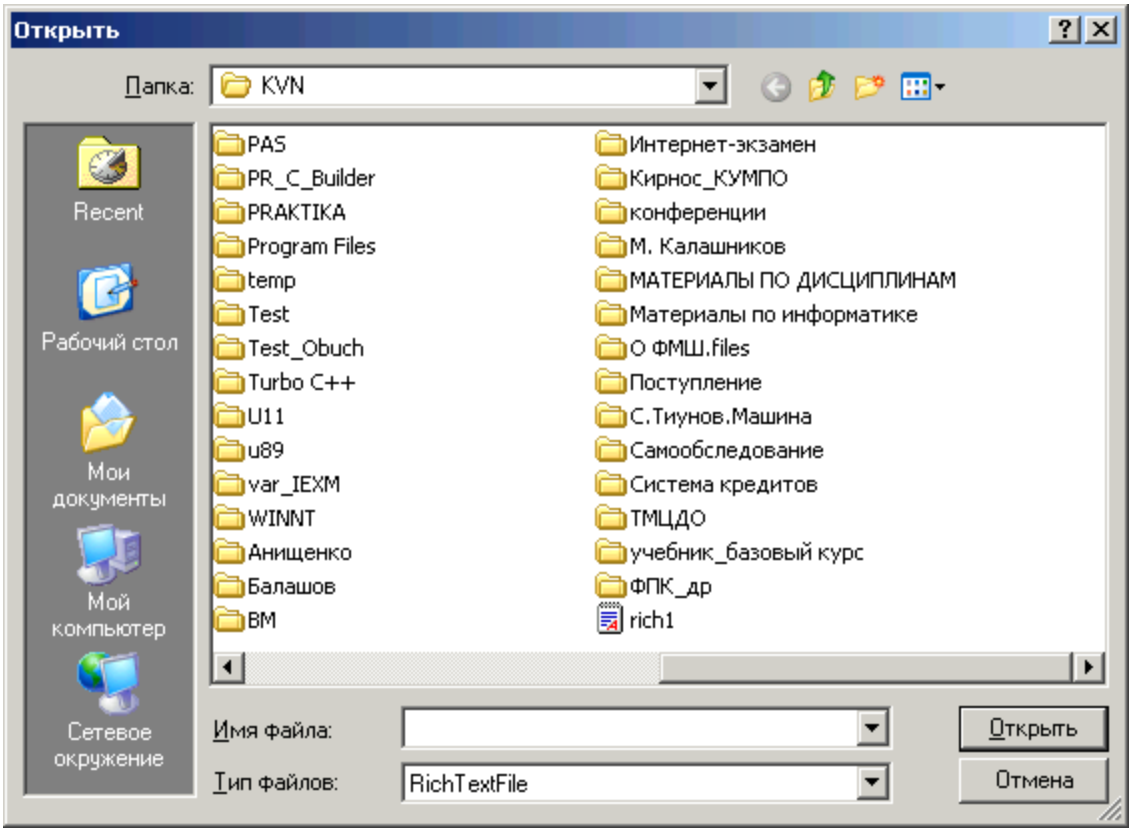


Рис. 6.2

Как видно, отображается единственный файл заданного нами типа — его имя **rich1**. Выбрав его и щелкнув по кнопке **Открыть**, мы и получим его содержимое на форме в окне richTextBox (см. ***рис. 6.3***).

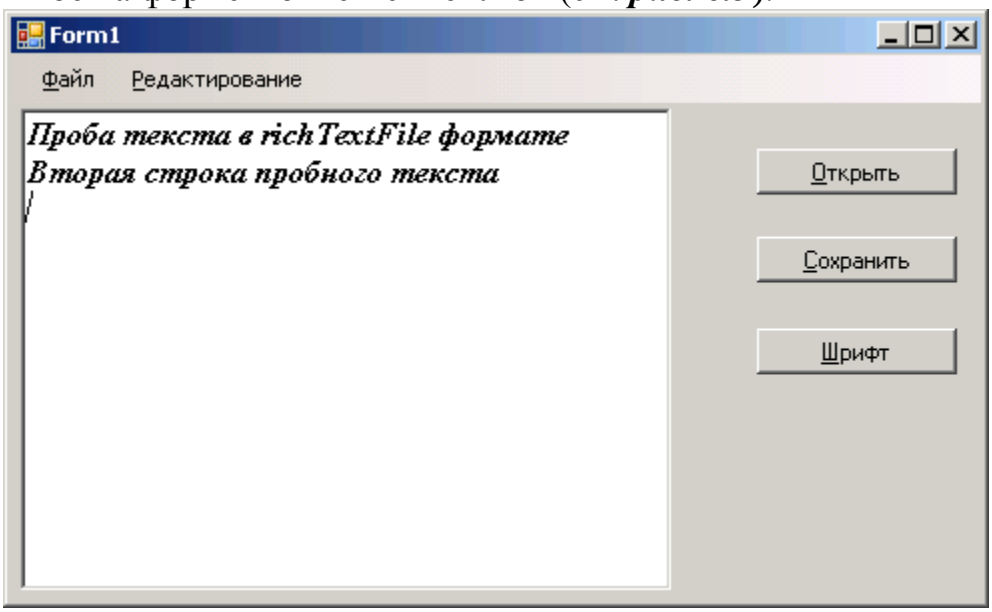


Рис. 6.3

В данном окне можем дописать текст и попробовать изменить шрифт. Щелкнем по пункту «Редактирование» и подпункту «Шрифт» - получим окно вида (***рис. 6.4***)

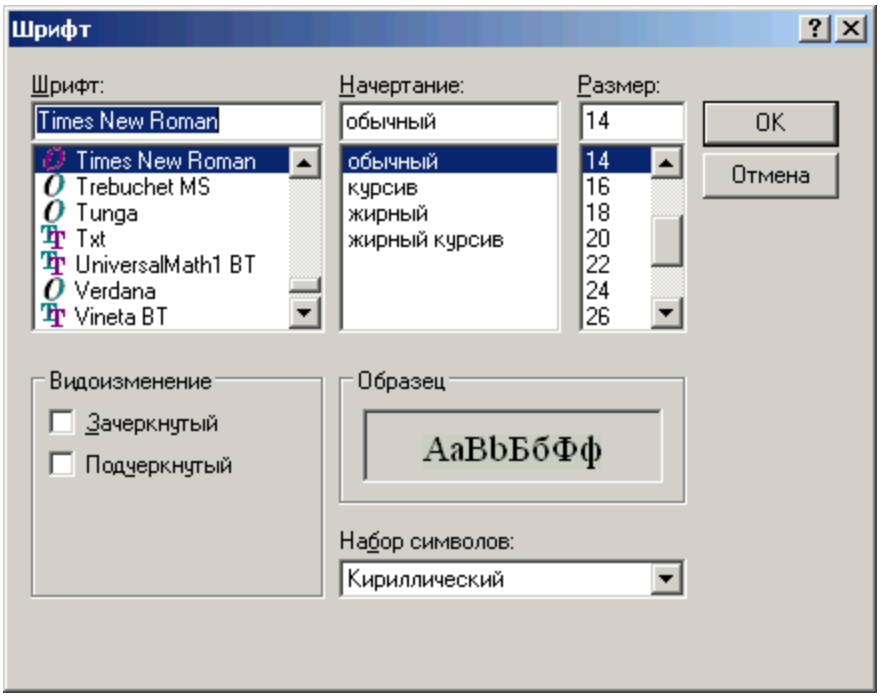


Рис. 6.4

В результате (когда щелкнем ОК в данном окошке), получим на форме результат, например, такого вида (см. ***рис. 6.5)***:

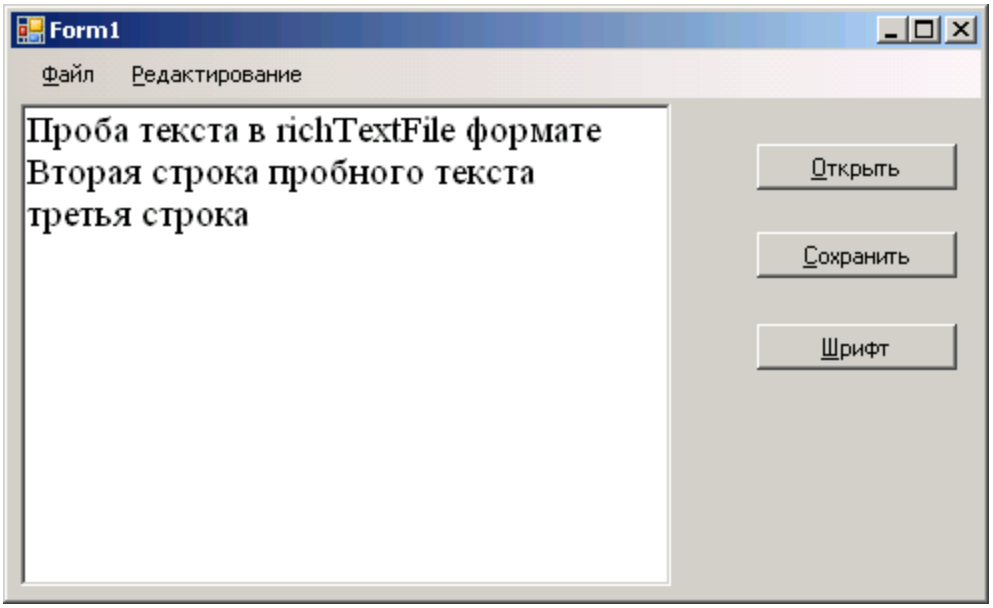


Рис. 6.5

Теперь для сохранения файла достаточно вызвать меню ***Файл/Сохранить***, откроется окно диалога для сохранения (см. ***рис. 6.6***), указать имя для сохраняемого файла (или выбрать прежнее имя) и щелкнуть ОК. Измененный файл записан.

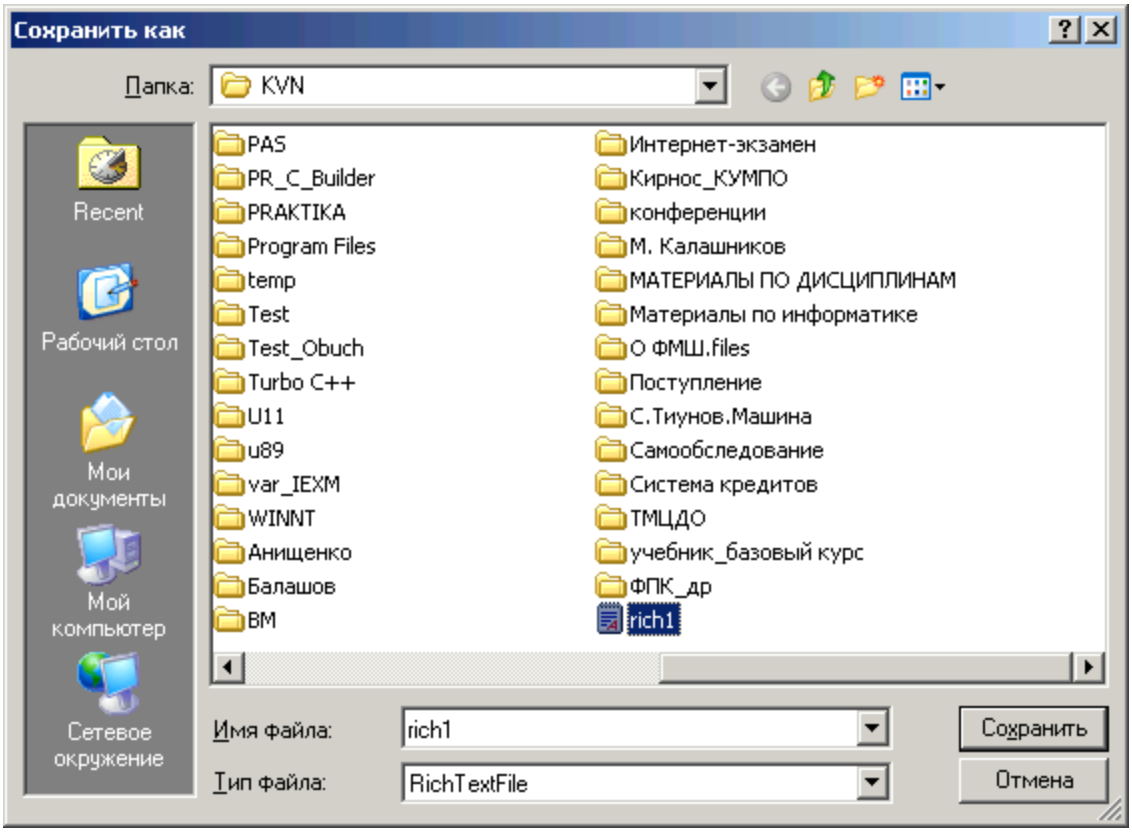


Рис. 6.6

ЛИТЕРАТУРА К ГЛАВЕ 3

1. Павловская Т. C/C++. Программирование на языке высокого уровня. Учебник. – СПб.: Питер, 2001 – 460 с.
2. Павловская Т., Щупак Ю. C++. Объектно-ориентированное программирование. Практикум. – СПб.: Питер, 2006 – 264 с.
3. Лафоре Р. Объектно-ориентированное программирование в C++. – СПб.: Питер, 2007. – 928 с.
4. Хортон А. Visual C++ 2005: базовый курс. – М.: ООО «И.Д. Вильямс», 2007 – 1152 с.

ГЛАВА 3. КОНТРОЛЬ ОБУЧЕНИЯ

Лабораторная работа №1. Линейная и разветвленная программа.

Первая лабораторная работа посвящена созданию блок-схем алгоритма и программ простого типа: линейных и программ с ветвлением.

Для выполнения лабораторной работы №1 следует изучить соответствующий разделы Главы 1.

В ходе выполнения контрольной работы необходимо, определив свой вариант, выполнить задания, перечисленные в таблице.

Тема / **Вариант**	Линейная программа	Программа с ветвлением
1	1-1	2-1
2	1-2	2-2
3	1-3	2-3
4	1-4	2-4
5	1-5	2-5
6	1-6	2-6
7	1-7	2-7
8	1-8	2-8
9	1-9	2-9
10	1-10	2-10
11	1-11	2-11
12	1-12	2-12
13	1-13	2-13
14	1-14	2-14
15	1-15	2-15
16	1-16	2-16
17	1-17	2-17
18	1-18	2-18
19	1-19	2-19
20	1-20	2-20
21	1-21	2-21
22	1-22	2-22
23	1-23	2-23
24	1-24	2-24
25	1-25	2-25

Раздел 1. «Линейные программы»

Для решения задач этого раздела следует предварительно изучить *разделы 1, 2 и 3 Главы 1*.

Составить программу для решения нижеприведенных задач в режиме **Win32 Console Application**, вводя данные в ходе выполнения программы. Для каждой задачи данного раздела дан пример входных данных и соответствующий ответ (для проверки правильности работы программы).

1-1. Найти массу x литров молока, если известно, что плотность молока p кг/м^3.

Пример: x=7 л, p=1030 кг/м^3. *Ответ:* 7,21 кг.

1-2. Объем цилиндра равен V, а площадь основания – S. Какова высота цилиндра H?

Пример: V=10 м^3, S=5 м^2. *Ответ:* 2 м.

1-3. Дана длина ребра куба a. Найти объем куба V и площадь его боковой поверхности S.

Пример: a=5 *Ответ:* V=125, S=100.

1-4. Каков объем кислорода, содержащегося в комнате размером $a \bullet b \bullet c$, если кислород составляет 21% объема воздуха?

Пример: a=3, b=4, c=5. *Ответ:* 12,6.

1-5. Найти площадь равнобочной трапеции с основаниями a и b и углом при большем основании равным x.

Пример: a=6, b=5, x=45^o. *Ответ:* 2,75.

1-6. Найти угол между отрезком прямой, соединяющей начало координат с точкой $A(x,y)$, и осью OX (точка лежит в 1-й четверти).

Пример: x=3, y=4. *Ответ:* 53,13°.

1-7. Определить время падения камня на поверхность земли с высоты h.

Пример: h=10 м. *Ответ:* 1,4278 с.

1-8. Три сопротивления R_1,R_2,R_3 соединены параллельно. Найти сопротивление соединения.

Пример: R_1=10, R_2=15, R_3=20. *Ответ:* 4,62.

1-9. Написать программу вычисления площади параллелограмма. Извне вводятся стороны a, b и угол между ними x.

Пример: a=10, b=15, x=30^o. *Ответ:* 75.

1-10. Написать программу вычисления объема прямоугольного параллелепипеда. Извне вводятся длина a, ширина b и высота c.

Пример: a=10, b=15, c=20. *Ответ:* 3000.

1-11. Написать программу вычисления площади поверхности прямоугольного параллелепипеда. Извне вводятся длина a, ширина b и высота c.

Пример: a=10, b=15, c=20. *Ответ:* 1300.

1-12. Написать программу вычисления объема цилиндра. Извне вводятся радиус основания *R* и высота цилиндра *h*.

Пример: *R*=10, *h*=15. *Ответ:* 4712,39.

1-13. Написать программу вычисления стоимости покупки, состоящей из нескольких тетрадей и карандашей. Извне вводятся цена одной тетради *Ct* и количество тетрадей *Kt*, а также цена карандаша *Ck* и количество карандашей *Kk*.

Пример: *Ct*=1, *Kt*=15, *Ck*=0.2, *Kk*=5 *Ответ:* 16.

1-14. Написать программу вычисления стоимости покупки, состоящей из нескольких тетрадей и такого же количества обложек к ним. Извне вводятся цена одной тетради *Ct*, одной обложки *Cb* и количество тетрадей *Kt*.

Пример: *Ct*=1.2, *Kt*=15, *Cb*=0,2. *Ответ:* 21.

1-15. Написать программу вычисления стоимости некоторого количества (по весу) яблок. Извне вводятся цена одного килограмма яблок *C* и вес яблок *V*.

Пример: *C*=25, *V*=1,5. *Ответ:* 37,5.

1-16. Написать программу вычисления сопротивления электрической цепи, состоящей из двух параллельно соединенных сопротивлений. Извне вводятся величина первого и второго сопротивления.

Пример: R_1=10, R_2=15. *Ответ:* 6.

1-17. Написать программу вычисления сопротивления электрической цепи, состоящей из двух последовательно соединенных сопротивлений. Извне вводятся величина первого и второго сопротивления.

Пример: R_1=10, R_2=15. *Ответ:* 25.

1-18. Написать программу вычисления силы тока в электрической цепи. Извне вводятся напряжение *U* и сопротивление *R*.

Пример: *U*=10, *R*=15. *Ответ:* 0,6667.

1-19. Составить программу, которая поменяет местами значения введенных переменных *x* и *y:*

а) используя дополнительную переменную;

б)* не используя дополнительной переменной.

1-20. Составить программу, которая поменяет местами значения введенных переменных *x*,*y*,*z* так, чтобы в переменной *x* оказалось значение переменной *y*, в *y* – значение переменной *z*, а в *z* – прежнее значение переменной *x*:

а) используя дополнительную переменную;

б) не используя дополнительной переменной.

1-21. Составить программу перевода километров в мили, версты, сажени (см. табл. 3)

1-22. Составить программу перевода метров в версты, сажени, футы, аршины (см. табл. 3)

1-23. Составить программу перевода сантиметров в вершки, футы, дюймы (см. табл. 3)

1-24. Составить программу перевода килограммов в пуды, фунты, лоты, золотники (см. табл. 3)

1-25. Составить программу перевода литров в бочки, ведра, штофы, бутылки, чарки (см. табл. 1)

Таблица 1

Русская система мер

Меры массы (*веса*)	
1 пуд = 40 фунтов	0,016 т = 0,164 ц = 16,3805 кг
1 фунт = 32 лота = 96 золотников	0,410 кг = 409,512 г
1 лот = 3 золотника	12,7973 г
1 золотник	4,26575 г
Меры емкости (*для жидкости*)	
1 бочка = 40 ведер	4, 920 гл (гекталитров)
1 ведро = 10 штофов = 20 бутылок	1,2299 дкл = 12,2994 л
1 штоф = 10 чарок	1,230 л
1 чарка	0,123 л
1 бутылка	0,615 л
Меры длины	
1 миля = 7 верст	7,4676 км
1 верста = 500 саженей	1,0668 км
1 сажень = 3 аршина = 7 футов	2,1336 м
1 аршин = 16 вершков	0,711 м = 71,120 см
1 вершок	4,445 см = 44,45 мм
1 фут = 12 дюймов	0,305 м = 30,48 см
1 дюйм	2,540 см = 25,4 мм

Раздел 2. «Программы с ветвлением»

Для решения задач этого раздела следует предварительно изучить *раздел 4 Главы 1*.

2-1. Даны числа *a, b, c*. Проверить, выполняется ли неравенство $a<b<c$. Вывести об этом сообщение.

2-2. Даны три действительных числа *a, b, c*. Выбрать из них те, которые принадлежат интервалу (1, 3).

2-3. Даны числа *x, y* ($x \neq y$). Меньшее из них заменить полусуммой, а большее – их удвоенным произведением.

2-4. Найти наибольшее для трех заданных чисел *a*, *b*, *c*.

2-5. Выяснить, существует ли треугольник с длинами сторон *x*, *y*, *z* (в треугольнике большая сторона меньше суммы двух других сторон).

2-6. На окружности с центром в точке (x_0,y_0) задана дуга с координатами начальной (x_n, y_n) и конечной (x_k, y_k) точек. Определить номера четвертей окружности, в которых находятся начальная и конечная точки.

2-7. Даны координаты точки (*x*, *y*). Выяснить, принадлежит ли эта точка области, указанной на рис. 1, *а*.

2-8. Даны координаты точки (*x*, *y*). Выяснить, принадлежит ли эта точка области, указанной на рис. 1, *б*.

2-9. Даны координаты точки (*x*,*y*). Выяснить, принадлежит ли эта точка области, указанной на рис. 1, *в*.

2-10. Даны координаты точки (*x*,*y*). Выяснить, принадлежит ли эта точка области, указанной на рис. 1, *г*.

2-11. Даны координаты точки (*x*,*y*). Выяснить, принадлежит ли эта точка области, указанной на рис. 1, *д*.

2-12. Даны координаты точки (*x*, *y*). Выяснить, принадлежит ли эта точка области, указанной на рис. 1, *е*.

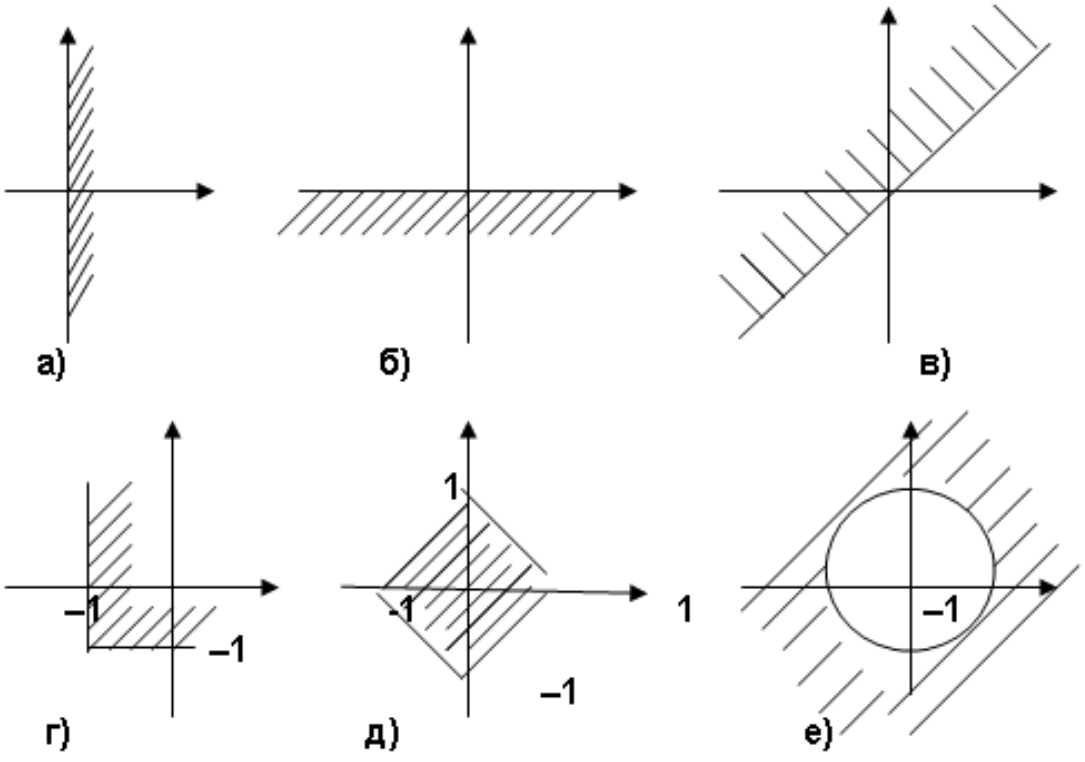


Рис. 1. Область (заштрихованная) на координатной оси для заданий 2-7 – 2-12. Штриховка правее или ниже, или над линией означает всю область соответственно правее или ниже, или над линией.

2-13. Даны действительные положительные числа *a*, *b*, *c*, *x*, *y*. Выяснить, пройдет ли кирпич с ребрами *a*, *b*, *c* в прямоугольное отверстие со сторонами *x* и *y*. Просовывать кирпич разрешается только так, чтобы каж-

дое из его ребер было параллельно или перпендикулярно каждой из сторон отверстия.

2-14. Даны действительные числа x_1, x_2, x_3, y_1, y_2, y_3. Выяснить, является ли треугольник с вершинами (x_1, y_1), (x_2, y_2), (x_3, y_3) прямоугольным?

2-15. Для пятиугольника, заданного координатами своих вершин, найти наибольшую и наименьшую стороны**.**

2-16.. Написать программу вычисления площади кольца. Извне вводятся радиус кольца и радиус отверстия. В программе предусмотреть проверку правильности вводимых данных (радиусы положительны, причем радиус кольца больше радиуса отверстия).

2-17. Написать программу, которая переводит время из минут и секунд в секунды. Извне вводятся минуты и секунды. В программе предусмотреть проверку на правильность введенных данных (только положительные, кроме того, число минут <=60 и число секунд <=60)

2-18. Написать программу вычисления сопротивления электрической цепи, состоящей из двух сопротивлений. Извне вводятся величина первого, второго сопротивления и указывается тип соединения (например, 1 – последовательное, 2 – параллельное соединение).

2-19. Написать программу вычисления стоимости покупки с учетом скидки. Скидка в 10% предоставляется, если сумма покупки более 1000 руб. Извне вводится сумма покупки.

2-20. Написать программу вычисления стоимости разговора по телефону с учетом 20% скидки, предоставляемой по субботам и воскресеньям. Извне вводится длительность разговора (в целых минутах) и день недели цифрой (1 – понедельник, … 7 – воскресенье).

2-21. Написать программу, которая вычисляет оптимальный вес для пользователя, сравнивает его с реальным и выдает рекомендацию о необходимости поправится или похудеть. Оптимальный вес вычисляется по формуле: рост (см) – 100. Извне вводятся рост (в см) и вес (в кг).

2-22. Написать программу, которая запрашивает у пользователя номер месяца и затем выводит соответствующее название времени года. Если вводится недопустимое число (<1 или >12), должно появиться сообщение «ошибка ввода».

2-23. Написать программу, которая запрашивает у пользователя номер дня недели и выводит одно из сообщений: «рабочий день», «суббота» или «воскресенье».

2-24. Написать программу, которая запрашивает у пользователя номер дня недели и выводит название дня недели.

2-25. Написать программу, которая переводит время из часов и минут в минуты. Извне вводятся часы (целое положительное) и минуты (целое положительное и <=59). Программа должна проверять правильность введенных данных.

Лабораторная работа №2. Циклические программы

Вторая лабораторная работа посвящена созданию блок-схем алгоритма и програм, использующих цикл двух видов: цикл со счетчиком и цикл-ПОКА.

Для выполнения лабораторной работы №3 следует изучить соответствующие разделы Главы 1. Задания нужно выполнять на языке **Visual C++** в среде **Microsoft Visual Studio 2008.**

В ходе выполнения контрольной работы необходимо, определив свой вариант, выполнить задания, перечисленные в таблице.

Тема / **Вариант**	Цикл с параметром	Цикл-ПОКА
1	3-1	4-1
2	3-2	4-2
3	3-3	4-3
4	3-4	4-4
5	3-5	4-5
6	3-6	4-6
7	3-7	4-7
8	3-8	4-8
9	3-9	4-9
10	3-10	4-10
11	3-11	4-11
12	3-12	4-12
13	3-13	4-13
14	3-14	4-14
15	3-15	4-15
16	3-16	4-16
17	3-17	4-17
18	3-18	4-18
19	3-19	4-19
20	3-20	4-20
21	3-21	4-21
22	3-22	4-22
23	3-23	4-23
24	3-24	4-24
25	3-25	4-25

Раздел 3. «Цикл с параметром»

Для решения задач этого раздела следует предварительно изучить *раздел 5 Главы 1*.

Составить программу для решения нижеприведенных задач в режиме **Win32 Console Application**, вводя данные в ходе выполнения программы.

В трех последующих задачах условие одинаковое: для заданного натурального числа n рассчитать сумму:

3-1. $\cos x + \cos x^2 + \cos x^3 + ... + \cos x^n$. *($x$ вводится извне)*

3-2. $\sin 1 + \sin(1+0{,}1) + \sin(1+0{,}1^2) + ... + \sin(1+0{,}1^n)$.

3-3. $1! + 2! + 3! + ... + n!$

В трех следующих заданиях условие одинаковое: для заданного натурального числа n рассчитать сумму и сравнить со значением y:

3-4. $1 + 2 + 3 + ... + n$; $y = n(n+1)/2$.

3-5. $1 + 3 + 5 + ... + (2n-1)$; $y = n^2$.

3-6. $2 + 4 + 6 + ... + 2n$ $y = n(n+1)$.

3-7. Для заданного натурального числа n рассчитать величину $n!!$

$$n!! = \begin{cases} 1 \cdot 3 \cdot 5 \cdot ... \cdot n, & \text{если } n - \text{нечетное} \\ 2 \cdot 4 \cdot 6 \cdot ... \cdot n, & \text{если } n - \text{четное} \end{cases}$$

3-8. Дано 10 вещественных чисел: $a_1, a_2, ..., a_{10}$. Требуется определить, сколько из них принимает значение, большее заданного числа b.

3-9. Дано 10 вещественных чисел: $a_1, a_2, ..., a_{10}$. Требуется найти порядковые номера тех из них, которые отрицательны.

3-10. Вычислить K – количество точек с целочисленными координатами, попадающих в круг радиуса R с центром в начале координат.

3-11. Написать программу, которая выводит таблицу квадратов первых десяти целых положительных чисел.

3-12. Написать программу, которая выводит таблицу квадратов первых пяти целых положительных нечетных чисел.

3-13. Написать программу, которая выводит таблицу квадратов чисел от 11 до 19.

3-14. Написать программу, которая вычисляет сумму первых N целых положительных чисел.

3-15. Написать программу, которая вычисляет сумму первых N целых положительных четных чисел.

3-16. Написать программу, которая вычисляет сумму первых N целых положительных нечетных чисел.

3-17. Написать программу, которая выводит таблицу степеней двойки от нулевой до десятой степени.

3-18. Написать программу, которая выводит таблицу значений функции $y= -2,4x^2 + 5x - 3$ в диапазоне от –2 до 2 с шагом 0,5.

3-19. Написать программу, которая приглашает ввести последовательно 5 дробных чисел и вычисляет их среднее арифметическое.

3-20. Написать программу, которая приглашает ввести последовательно N дробных чисел и вычисляет их среднее арифметическое. Значение N вводится извне.

В пяти следующих заданиях условие одинаковое: для заданного натурального числа n рассчитать сумму и сравнить со значением y:

3-21. $1-\frac{1}{3}+\frac{1}{5}-\frac{1}{7}+\ldots+\frac{(-1)^{n-1}}{2n-1}+\ldots;$ $y = \pi/4.$

3-22. $\frac{1}{3*5}+\frac{1}{7*9}+\frac{1}{11*13}+\ldots+\frac{1}{(4n-1)(4n+1)}+\ldots;$ $y = 0,5 - \pi/8.$

3-23. $1+\frac{1}{2^2}+\frac{1}{3^2}+\frac{1}{4^2}+\ldots+\frac{1}{n^2}+\ldots;$ $y = \pi^2/6.$

3-24 $1+\cfrac{1}{2+\cfrac{3^2}{2+\cfrac{5^2}{2+\cfrac{7^2}{2+\ldots}}}}$ $y = 4/\pi.$

3-25. $\frac{2*2*4*4*6*6*8*8*\ldots*2n*2n}{1*3*3*5*5*7*7*9*\ldots*(2n-1)};$ $y = \pi/2$

Раздел 4. «Цикл «ПОКА»

Для решения задач этого раздела следует предварительно изучить *раздел 6 Главы 1*.

Составить программу для решения нижеприведенных задач в режиме **Win32 Console Application**, вводя данные в ходе выполнения программы.

4-1. Вычислить для заданного x сумму вида: $x–x^2/2+x^3/3–x^4/4+\ldots$ с заданной точностью E (когда очередное слагаемое по модулю станет меньше E, то суммирование прекратить). Результат сравнить с величиной $y= \ln(x+1), |x|<1$.

4-2. Для заданного X в последовательности вида: sin$X$, sin(sin$X$)), sin(sin(sin$X$)),... найти первое число, меньшее по модулю 0,01.

4-3. Найти наименьший номер n, для которого выполняется условие $|a_n - a_{n-1}| < 0{,}1$, если последовательность a_n имеет вид: $a_{n+1} = a_n + 2/a_n$, $a_1 = 1$.

4-4. С заданной точностью E рассчитать: $1+\frac{1}{1!}+\frac{1}{2!}+\frac{1}{3!}+\ldots$. Результат сравнить со значением e =2,718281828... Напомним, что запись вида $n!$ означает $1\cdot2\cdot3\cdot\ldots\cdot n$ (последовательное произведение целых чисел от 1 до n). *Указание*. Для вычисления значения $n!$ применить рекуррентную формулу: $n! = (n-1)! \cdot n$.

4-5. Для заданного x с заданной точностью E рассчитать сумму вида: $x-\frac{x^3}{3!}+\frac{x^5}{5!}-\frac{x^7}{7!}+\ldots$ Результат сравнить со значением $y = \sin x$.

4-6. Для заданного x с заданной точностью E рассчитать сумму вида: $1-\frac{x^2}{2!}+\frac{x^4}{4!}-\frac{x^6}{6!}+\ldots$ Результат сравнить со значением $y = \cos x$.

В заданиях с **4-7** по **4-13** рассмотреть указанную последовательность в цикле и выйти из цикла, достигнув указанного условия с выдачей порядкового номера члена, при котором достигнуто условие. Если же за ***m*** оборотов цикла условие не достигнуто, напечатать об этом сообщение.

4-7. $a_{n+1} = \mathrm{tg}\, a_n/2;\ \ a_1 = 0{,}1;\ \ |a_n| < 10^{-3}$.

4-8. $a_n = 2^n/n!;\ \ |a_n| < 10^{-8}$.

4-9. $a_n = 1/\sqrt[n]{n!};\ \ \ |a_n| < 0{,}1$

4-10. $a_{n+1} = (a_n + 1/a_n)/2;\ \ a_1 = 10,\ |1 - a_n| < 10^{-4}$.

4-11. $a_{n+1} = 1{,}99a_n - a_{n-1};\ \ \ a_1 = 1, a_2 = 1;\ \ \ a_n < 0{,}9$

4-12. $a_{n+1} = (a_n + a_{n-1})/2;\ a_1 = 1, a_2 = 10;\ |a_n - 7| < 10^{-4}$.

4-13. $a_{n+1} = \sqrt{2 + a_n};\ a_1 = 0;\ \ |a_n - a_{n-1}| < 10^{-3}$.

4-14. Написать программу, вычисляющую сумму и среднее арифметическое последовательности положительных чисел, которые вводятся с клавиатуры (длина последовательности не ограничена). Для завершения ввода нужно нажать 0.

4-15. Написать программу, которая определяет максимальное число из введенной последовательности положительных чисел (длина последовательности не ограничена). Для завершения ввода нужно нажать 0.

4-16. Написать программу, которая определяет минимальное число из введенной последовательности положительных чисел (длина последовательности не ограничена). Для завершения ввода нужно нажать 0.

4-17. Написать программу, которая задумывает число в диапазоне от 1 до 10 и предлагает пользователю угадать число за 5 попыток. В завершение должно быть сообщение о том, что пользователь угадал число или нет.

4-18. Написать программу, которая выводит на экран таблицу значений функции $y = 2x^2 - 5x - 8$ в диапазоне от – 4 до 4. Шаг изменения аргумента 0,5.

4-19. Написать программу, которая вычисляет наибольший общий делитель двух целых чисел.

4-20. Написать программу для вычисления сопротивления цепи из нескольких проводников, соединенных последовательно (значения сопротивлений вводятся одно за другим и для завершения ввода нажимается 0).

4-21. Написать программу для вычисления сопротивления цепи из нескольких проводников, соединенных параллельно (значения сопротивлений вводятся одно за другим и для завершения ввода нажимается 0).

4-22. Написать программу, которая выводит на экран таблицу значений функции $y = 1/x$ в диапазоне от заданного A (вводится извне) и до 0. Шаг изменения аргумента h также вводится извне.

4-23. Написать программу, которая вычисляет число π с заданной пользователем точностью. Для этого суммируют последовательность:

$1 - 1/3 + 1/5 - 1/7 + 1/9 - \ldots$

до тех пор, пока очередное слагаемое (по абсолютной величине) не станет меньше заданной точности. Полученная сумма дает примерное значение числа $\pi/4$, т.е. умножая полученную сумму на 4 и получим π.

4-24. Написать программу вычисления значения квадратного корня из числа $a>0$ с заданной точностью E на основе соотношения $x_{n+1} = \frac{x_n + a/x_n}{2}$; $n = 1, 2, 3, \ldots$, где x_n – предыдущее, x_{n+1} – последующее приближение к корню. Начальное приближение для определенности положить равным $a/2$. Точность вычислений принято считать достигнутой, когда $|x_{n+1} - x_n| < E$.

4-25. Спортсмен в первый день пробежал 10 км. Каждый следующий день он увеличивал дневную норму на 10 % от нормы предыдущего дня. Определить через сколько дней спортсмен:

а) будет пробегать более 20 км;

б) пробежит суммарный путь более 100 км.

Лабораторная работа №3. Массивы.

Третья лабораторная работа посвящена созданию блок-схем алгоритма и программ для обработки одно- и двумерных массивов.

Для выполнения лабораторной работы №3 следует изучить соответствующие разделы Главы 1. Задания нужно выполнять на языке **Visual C++** в среде **Microsoft Visual Studio 2008.**

В ходе выполнения контрольной работы необходимо, определив свой вариант, выполнить задания, перечисленные в таблице.

Тема / **Вариант**	Одномерные массивы	Двумерные массивы
1	5-1	6-1
2	5-2	6-2
3	5-3	6-3
4	5-4	6-4
5	5-5	6-5
6	5-6	6-6
7	5-7	6-7
8	5-8	6-8
9	5-9	6-9
10	5-10	6-10
11	5-11	6-11
12	5-12	6-12
13	5-13	6-13
14	5-14	6-14
15	5-15	6-15
16	5-16	6-16
17	5-17	6-17
18	5-18	6-18
19	5-19	6-19
20	5-20	6-20
21	5-21	6-21
22	5-22	6-22
23	5-23	6-23
24	5-24	6-24
25	5-25	6-25

Раздел 5. «Одномерные массивы»

Для решения задач этого раздела следует предварительно изучить раздел 7 Главы 1.

Составить программу для решения нижеприведенных задач в режиме **Win32 Console Application**, вводя данные в ходе выполнения программы.

5-1. Дан массив из *N* элементов (вещественные числа). Вычислить: 1) сумму отрицательных элементов массива; 2) произведение элементов массива, расположенных между максимальным и минимальным элементами. Упорядочить элементы по возрастанию.

5-2. Дан массив из *N* элементов (вещественные числа). Вычислить: 1) сумму положительных элементов массива; 2) произведение элементов массива, расположенных между максимальным по модулю и минимальным по модулю элементами. Упорядочить элементы по убыванию.

5-3. Дан массив из *N* элементов (целые числа). Вычислить: 1) произведение элементов массива с четными номерами; 2) сумму элементов массива, расположенных между первым и последним нулевыми элементами. Преобразовать массив так, чтобы сначала располагались все положительные элементы, а потом – все отрицательные (элементы, равные 0, считать положительными).

5-4. Дан массив из *N* элементов (вещественные числа). Вычислить: 1) произведение элементов массива с нечетными номерами; 2) сумму элементов массива, расположенных между первым и последним отрицательными элементами. Сжать массив, удалив из него все элементы, модуль которых не превышает 1. Освободившиеся в конце массива элементы заполнить нулями.

5-5. Дан массив из *N* элементов (вещественные числа). Вычислить: 1) максимальный элемент массива; 2) сумму элементов массива, расположенных до последнего положительного элемента. Сжать массив, удалив из него все элементы, модуль которых находится в интервале [a, b]. Освободившиеся в конце массива элементы заполнить нулями.

5-6. Дан массив из *N* элементов (вещественные числа). Вычислить:1) минимальный элемент массива; 2) сумму элементов массива, расположенных между первым и последним положительными элементами. Преобразовать массив так, чтобы сначала располагались все элементы, равные нулю, а потом –остальные.

5-7. Дан массив из *N* элементов (вещественные числа). Вычислить: 1) номер максимального элемента массива; 2) произведение элементов массива, расположенных между первым и вторым нулевыми элементами. Преобразовать массив так, чтобы сначала располагались все элементы,

стоящие в нечетных позициях, а потом – элементы, стоящие в нечетных позициях.

5-8. Дан массив из *N* элементов (вещественные числа). Вычислить: 1) номер минимального элемента массива; 2) произведение элементов массива, расположенных между первым и вторым отрицательными элементами. Преобразовать массив так, чтобы сначала располагались все элементы, модуль которых не превышает 1, а потом – элементы, все остальные.

5-9. Дан массив из *N* элементов (вещественные числа). Вычислить: 1) максимальный по модулю элемент массива; 2) сумму элементов массива, расположенных между первым и вторым положительными элементами. Преобразовать массив так, чтобы элементы, равные нулю, располагались после всех остальных.

5-10. Дан массив из *N* элементов (целые числа). Вычислить: 1) минимальный по модулю элемент массива; 2) сумму модулей элементов массива, расположенных после первого равного нулю элемента. Преобразовать массив так, чтобы сначала располагались элементы, стоящие в четных позициях, а потом – стоящие в нечетных позициях.

5-11. Дан массив из *N* элементов (вещественные числа). Вычислить: 1) номер минимального по модулю элемента массива; 2) сумму модулей элементов массива, расположенных после первого отрицательного элемента. Сжать массив, удалив из него все элементы, величина которых находится в интервале [a,b], а потом – все остальные.

5-12. Дан массив из *N* элементов (вещественные числа). Вычислить: 1) номер максимального по модулю элемента массива; 2) сумму модулей элементов массива, расположенных после первого положительного элемента. Преобразовать массив так, чтобы сначала располагались все элементы, целая часть которых находится в интервале [a,b], а потом – все остальные.

5-13. Дан массив из *N* элементов (вещественные числа). Вычислить: 1) количество элементов массива, лежащих в диапазоне от A до B; 2) сумму модулей элементов массива, расположенных после максимального элемента. Упорядочить элементы массива по убыванию модулей элементов.

5-14. Дан массив из *N* элементов (вещественные числа). Вычислить: 1) количество элементов массива, равных 0; 2) сумму элементов массива, расположенных после минимального элемента. Упорядочить элементы массива по возрастанию модулей элементов.

5-15. Дан массив из *N* элементов (вещественные числа). Вычислить: 1) количество элементов массива, больших C; 2) произведение элементов массива, расположенных после максимального элемента. Преобразовать массив таким образом, чтобы сна-чала располагались все отрицательные элементы, а потом все положительные (элементы, равные 0, считать положительными).

5-16. Дан массив из *N* элементов (вещественные числа). Вычислить: 1) количество отрицательных элементов массива; 2) сумму модулей элементов массива, расположенных после минимального по модулю элемента. Заменить все отрицательные элементы массива их квадратами и упорядочить элементы массива по возрастанию.

5-17. Дан массив из *N* элементов (целые числа). Вычислить: 1) количество положительных элементов массива; 2) сумму элементов массива, расположенных после последнего элемента, равного нулю. Преобразовать массив таким образом, чтобы сначала располагались все элементы, целая часть которых не превышает 1, а потом – все остальные.

5-18. Дан массив из *N* элементов (вещественные числа). Вычислить: 1) количество элементов массива, меньших C; 2) сумму целых частей элементов массива, расположенных после последнего отрицательного элемента. Преобразовать массив таким образом, чтобы сначала располагались все элементы, отличающиеся от максимального не более, чем на 20% а потом – все остальные.

5-19. Дан массив из *N* элементов (вещественные числа). Вычислить: 1) произведение отрицательных элементов массива; 2) сумму положительных элементов массива, расположенных до максимального элемента. Изменить порядок следования элементов в массиве на обратный.

5-20. Дан массив из *N* элементов (вещественные числа). Вычислить: 1) произведение положительных элементов массива; 2) сумму элементов массива, расположенных до минимального элемента. Упорядочить по возрастанию отдельно элементы, стоящие на четных местах, и элементы, стоящие на нечетных местах.

5-21. Дан массив из *N* элементов (вещественные числа). Вычислить: 1) произведение отрицательных элементов массива; 2) сумму элементов массива, расположенных между максимальным и минимальным элементами. Упорядочить элементы по возрастанию.

5-22. Дан массив из *N* элементов (вещественные числа). Вычислить: 1) произведение положительных элементов массива; 2) сумму элементов массива, расположенных между максимальным по модулю и минимальным по модулю элементами. Упорядочить элементы по убыванию.

5-23. Дан массив из *N* элементов (целые числа). Вычислить: 1) сумму элементов массива с четными номерами; 2) произведение элементов массива, расположенных между первым и последним нулевыми элементами. Преобразовать массив так, чтобы сначала располагались все положительные элементы, а потом – все отрицательные (элементы, равные 0, считать положительными).

5-24. Дан массив из *N* элементов (вещественные числа). Вычислить: 1) сумму элементов массива с нечетными номерами; 2) произведение элементов массива, расположенных между первым и последним отрицательными элементами. Сжать массив, удалив из него все элементы, модуль ко-

торых не превышает 1. Освободившиеся в конце массива элементы заполнить нулями.

5-25. Дан массив из *N* элементов (вещественные числа). Вычислить: 1) минимальный элемент массива; 2) сумму элементов массива, расположенных до первого положительного элемента. Сжать массив, удалив из него все элементы, модуль которых находится в интервале [a, b]. Освободившиеся в конце массива элементы заполнить нулями.

Раздел 6. «Двумерные массивы»

Для решения задач этого раздела следует предварительно изучить *раздел 8 Главы 1.*

Составить программу для решения нижеприведенных задач в режиме Win32 Console Application, вводя данные в ходе выполнения программы.

Для формирования матриц следует *использовать датчик случайных чисел.*

6-1. Дана матрица 5×5. Для данного натурального *M* найти сумму тех элементов матрицы, сумма индексов которых равна *M*.

6-2. Дана матрица 5×5. Построить одномерный массив *B*(5), состоящий из сумм элементов строк.

6-3. Дана матрица 5×5. Построить одномерный массив *B*(5), состоящий из произведений элементов столбцов.

6-4. Дана матрица 5×5. Построить одномерный массив *B*(5), состоящий из наименьших значений элементов строк.

6-5. Дана матрица 5×5. Построить одномерный массив *B*(5), состоящий из средних арифметических элементов строк.

6-6. Дана матрица 5×5. Вывести ее в транспонированном виде (поменять местами строки со столбцами).

6-7. Дана матрица 5×5. Вывести ее в верхнем треугольном виде (т.е. напечатать только элементы вер×него треугольника и именно в виде треугольника).

6-8. Дана матрица 5×5. Вывести ее в нижнем треугольном виде.

6-9. Результаты соревнований по прыжкам в длину представлены в виде матрицы 5×3 (5 спортсменов по 3 попытки у каждого). Указать, какой спортсмен и в какой попытке показал наилучший результат.

6-10. Результаты соревнований по пятиборью представлены в виде матрицы 5×5 (5 спортсменов и 5 видов соревнований), в которых указаны места, занятые каждым спортсменом в данном виде. Найти лучшего спортсмена (наименьшая сумма мест).

В последующих трех задачах использовать следующее:

Таблица футбольного чемпионата задана квадратной матрицей порядка n, в которой все элементы, принадлежащие главной диагонали, равны нулю, а каждый элемент, не принадлежащий главной диагонали, равен

3,1 или 0 (число очков, набранных в игре: 3 – выигрыш, 1 – ничья, 0 – проигрыш).

6-11. Найти число команд, имеющих больше побед, чем поражений.

6-12. Определить номера команд, прошедших чемпионат без поражений.

6-13. Выяснить, имеется хотя бы одна команда, выигравшая более половины игр.

6-14. Дана действительная матрица размера $n \times m$, в которой не все элементы равны нулю. Получить новую матрицу путём деления всех элементов данной матрицы на её наибольший по модулю элемент.

6-15. Дана действительная квадратная матрица порядка 12. Заменить нулями все её элементы, расположенные на главной диагонали и выше неё.

6-16. Дана действительная матрица размера $n \times m$. Найти сумму наибольших значений элементов её строк.

6-17. В данной действительной квадратной матрице порядка n найти сумму элементов строки, в которой расположен элемент с наименьшим значением. Предполагается, что такой элемент единственный.

6-18. В данной действительной матрице размера 6×9 поменять местами строку, содержащую элемент с наибольшим значением, со строкой, содержащей элемент с наименьшим значением. Предполагается, что эти элементы единственны.

6-19. Дана действительная матрица размера $n \times m$, все элементы которой различны. В каждой строке выбирается элемент с наименьшим значением, затем среди этих чисел выбирается наибольшее. Указать индексы элемента с найденным значением.

6-20. Дана целочисленная квадратная матрица порядка 8. Найти наименьшее из значений элементов столбца, который обладает наибольшей суммой модулей элементов. Если таких столбцов несколько, то взять первый из них.

6-21. Составить программу формирования двумерного массива $n \times n$ по следующему правилу: элементы главной диагонали приравнять 1, ниже главной диагонали – 0, а выше – сумме индексов.

6-22. Составить программу формирования двумерного массива из предложенного одномерного так, чтобы первая строка содержала четные по номеру элементы исходного массива, а вторая – нечетные. Предусмотреть случай нечетного количества элементов массива.

6-23. Составить программу формирования двумерного массива из предложенного одномерного, разделив его на два столбца.

6-24. Составить программу формирования двумерного массива из предложенного одномерного, разделив его на две строки.

6-25. Дан двумерный числовой массив. Составить программу обмена местами заданных двух его строк.

Лабораторная работа №4. Функции и рекурсия.

Четвертая лабораторная работа посвящена созданию блок-схем алгоритма и программ с использованием функций, созданных пользователем.

Для выполнения лабораторной работы №4 следует изучить соответствующие разделы главы 1. Задания нужно выполнять на языке **Visual C++** в среде **Microsoft Visual Studio 2008.**

В ходе выполнения контрольной работы необходимо, определив свой вариант, выполнить задания, перечисленные в таблице.

Тема / **Вариант**	Функции	Функции с рекурсией
1	7-1	8-1
2	7-1	8-2
3	7-3	8-3
4	7-4	8-4
5	7-5	8-5
6	7-6	8-6
7	7-7	8-7
8	7-8	8-8
9	7-9	8-9
10	7-10	8-10
11	7-11	8-11
12	7-12	8-12
13	7-13	8-13
14	7-14	8-14
15	7-15	8-15
16	7-16	8-16
17	7-17	8-17
18	7-18	8-18
19	7-19	8-19
20	7-20	8-20
21	7-21	8-21
22	7-22	8-22
23	7-23	8-23
24	7-24	8-24
25	7-25	8-25

Раздел 7. «Функции»

Для решения задач этого раздела следует предварительно изучить *раздел 9 Главы 1*.

7-1. Даны числа *a,b,c,d*. Получить *x*=max*(a,b)*, *y*=max*(c,d)*, *z*=max(*x,y)*. Вычисление max*(k,m)* (большего из двух чисел *k,m*) оформить функцией.

7-2. Оформить функцию *step*(*x,n*) от вещественного *x* и целого *n*, вычисляющую (через последовательное умножение) x^n и проверить ее.

7-3. Даны координаты вершин двух треугольников. Определить, какой из них имеет большую площадь. Вычисление площади треугольника по координатам оформить функцией.

7-4. Даны четыре натуральных числа. Найти наименьшее общее кратное (НОК) для этих четырех чисел. Поиск НОК двух чисел оформить функцией.

7-5. Даны координаты *n* точек на плоскости в виде масивов *X, Y*. Найти наиболее и наименее удаленные точки. Вычисление расстояния между парой точек оформить функцией.

7-6. Даны отрезки *a,b,c,d*. Для каждой тройки этих отрезков, из которых можно построить треугольник, напечатать площадь данного треугольника. Вычисление площади треугольника по заданным сторонам *x,y,z* с проверкой существования такого треугольника оформить функцией.

7-7. Рассчитать полную поверхность треугольной пирамиды, заданной координатами своих вершин. Для определения расстояния между точками в пространстве оформить функцию.

7-8. Дано число *n*. Получить все простые делители этого числа. Процедуру распознания простого числа оформить отдельно. (Напомним, что простым называется число, которое не имеет целочисленных делителей, кроме единицы и самого себя).

7-9. Составить программу, которая число, заданное в десятичной системе счисления, переведет в: ***а*)** двоичную систему счисления; ***б*)** восьмеричную. Перевод в каждую из систем счисления оформить отдельной функцией.

7-10. Даны координаты вершин треугольника на плоскости и координаты точки внутри него. Найти минимальное расстояние от этой точки до стороны треугольника. Оформить вычисление расстояния от точки до прямой отдельной функцией.

7-11. Найти высоты треугольника, заданного координатами своих вершин. Указать наименьшую из них. Для определения стороны треугольника оформить отдельную функцию.

7-12. Написать функцию, которая вычисляет сопротивление цепи, состоящей из трех проводников. Параметрами ее являются значения сопротивлений, а также тип соединения – последовательное или параллель-

ное (цифрами 1 или 2). Проверить ее в работе, написав программу с ее использованием.

7-13. Написать программу для вычисления косинуса угла между векторами, заданными своими координатами (скалярное произведение векторов делится на произведение модулей этих векторов). Для вычисления скалярного произведения и модуля вектора оформить отдельную функцию.

7-14. Написать функцию, которая вычисляет доход по вкладу. Исходными данными для функции являются: величина вклада, процентная ставка (годовых) и срок вклада (количество дней). Проверить ее в работе, написав программу с ее использованием.

7-15. Написать функцию, которая возвращает преобразованную к верхнему регистру строку, полученную в качестве аргумента (т.е. вместо малых букв выводит строку заглавными буквами). Функция должна работать как для латинских, так и для русских букв. Проверить ее в работе, написав программу с ее использованием.

7-16. Написать функцию решения квадратного уравнения. Исходными данными для подпрограммы должны быть коэффициенты уравнения, а выдавать подпрограмма должна сообщение о том, есть корни или нет корней, их количество и их значение. Проверять вводимые данные (коэффициент при x^2 не должен быть нулем). Проверить ее в работе, написав программу с ее использованием.

7-17. Даны четыре натуральных числа. Найти наибольший общий делитель (НОД) для этих четырех чисел. Поиск НОД двух чисел оформить функцией.

7-18. Дано натуральное число n. Среди чисел $1,2,...,n$ найти все те, которые можно представить в виде суммы квадратов двух натуральных чисел. (Определить функцию, позволяющую распознавать полные квадраты.)

7-19. Даны действительные числа $x1$, $y1$, $x2$, $y2$,..., $x10$, $y10$. Найти периметр десятиугольника, вершины которого имеют соответственно координаты ($x1$, $y1$), ($x2$, $y2$),..., ($x10$, $y10$). (Определить функцию вычисления расстояния между двумя точками, заданными своими координатами.)

7-20. Дано натуральное число n. Выяснить, имеются ли среди чисел n, $n+1$,..., $2n$ близнецы, т.е. простые числа, разность между которыми равна двум. (Определить процедуру, позволяющую распознавать простые числа.)

В ***пяти следующих*** задачах потребуется сократить обыкновенную дробь результата. Процедуру сокращения дроби оформить в виде отдельной функции (напомним, что для этого нужно искать наибольший общий делитель для числителя и знаменателя дроби).

7-21. Составить программу для сложения двух обыкновенных дробей $a/b + c/d$.

7-22. Составить программу для вычитания двух обыкновенных дробей $a/b - c/d$.

7-23. Составить программу для умножения двух обыкновенных дробей *a*/*b***c*/*d*.

7-24. Составить программу для деления двух обыкновенных дробей *a*/*b*:*c*/*d*.

7-25. Составить программу для возведения в степень обыкновенной дроби.

Раздел 8. «Функции с рекурсией»

Для приведенных в этом разделе заданий составить отдельную функцию с использованием ***рекурсии*** и применить ее к конкретной задаче.

8-1. Найти максимум одномерного массива.

8-2. Найти минимум одномерного массива.

8-3. Реализовать возведение в степень a^n по формуле

$$a^n = \begin{cases} a^{n/2} \cdot a^{n/2}, & n - \text{четное}, \\ a \cdot a^{n-1}, & n - \text{нечетное}. \end{cases}$$

8-4. Найти сумму цифр НАТУРАЛЬНОГО числа.

8-5. Найти сумму цифр ВЕЩЕСТВЕННОГО числа.

8-6. Организовать перевод в другую (заданную) систему счисления.

8-7. Рассчитать биномиальные коэффициенты по формулам

$C_n^0 = 1$ для $n \geq 0$;

$C_n^m = 0$ для $m > n \geq 0$.

$C_n^m = C_{n-1}^{m-1} + C_{n-1}^m$ для $n \geq m > 0$.

8-8. Найти наибольший общий делитель.

8-9. Найти наименьшее общее кратное (через наибольший общий делитель): НОК=a*b/ НОД.

8-10. Организовать бинарный поиск заданного числа в массиве.

8-11. Провести вычисление многочлена по схеме Горнера

$f(x) = (...(((x + a_1)x + a_2)x + a_3)...)x + a_n$.

8-12. Провести вывод простых чисел от 1 до заданного N.

8-13. Провести вычисление интеграла вида

$f(n) = I_n = \int_0^{\pi/2} \cos^n(x)dx$ по формуле

$f(0) = \pi/2;\ f(1) = 1;\ f(n) = (n-1)/n \cdot f(n-2),\ n >= 2.$

8-14. Провести вычисление квадратного корня заданного $a>0$ по рекуррентной формуле $x_1=1;\ x_{n+1}=\frac{1}{2}(x_n+\frac{a}{x_n})$, т.е. указанная последовательность дает все более точное значение $\sqrt{a}$.

8-15. Провести вычисление первообразной (в указанной точке x) заданной функции $f(x)=\frac{1}{(x^2+a^2)^n}$, т.е. $J_n(x)=\int\frac{dx}{(x^2+a^2)^n}$ по рекуррентной формуле

$$J_1(x)=\frac{1}{a}\text{arctg}(\frac{x}{a});$$

$$J_n(x)=\frac{x}{2(n-1)a^2(x^2+a^2)^{n-1}}+\frac{2n-3}{2(n-1)a^2}J_{n-1}(x).$$

8-16. Рассчитать последовательность полиномов Лагерра

$$L_0(x)=1, L_1(x)=x-1,\quad L_k(x)=(x-2k+1)L_{k-1}(x)+(k-1)^2L_{k-2}(x).$$

8-17. Вычислить сумму 12 членов рекуррентной последовательности

$$x_0=1;\ x_1=1;\quad x_k=0,7x_{k-1}+1,1x_{k-2},\ k=2,3,...$$

8-18. Вычислить функцию Бесселя 8-го порядка с аргументом ***x***

$$J(0,x)=x, J(1,x)=2x,\ J(n,x)=\frac{2(n-1)}{x}J(n-1,x)-J(n-2,x).$$

8-19. Вычислить значения полиномов Эрмита

$$H_0(x)=1,$$

$$H_1(x)=2x,$$

$$H_n(x)=2xH_{n-1}(x)-2(n-1)H_{n-2}(x).$$

8-20. Вычислить значения функции Аккермана для заданных ***m*** и ***n***

$$\text{Ack}(0,n)=n+1;$$

$$\text{Ack}(m,n)=\text{Ack}(m-1,1);$$

$$\text{Ack}(m,n)=\text{Ack}(m-1,\ \text{Ack}(m,n-1)).$$

В ***пяти следующих*** задачах условие одинаковое:

Найти корень заданного уравнения методом простой итерации с заданной точностью. Напомним, что в этом методе нужно уравнение свести к виду $x=f(x)$ и очередное уточнение корня проводится по формуле $x_{n+1}=f(x_n)$ до тех пор, пока $|x_{n+1}-x_n|>E$, где E – заданная точность. Рядом с уравнением в скобках указано начальное приближение корня.

8-21. $2x^3+4x-1=0$, (0,11). **8-22.** $x^3+12x-2=0$, (0,95).

8-23. $5x-8\ln x=8$, (4,32). **8-24.** $x^3+x=1000$, (9,42).

8-25. $x-\sin x=0.25$, (1,17).

Лабораторная работа №5. Обработка символьных строк и структур

Пятая лабораторная работа посвящена созданию блок-схем алгоритма и программ с использованием символьных строк и структур.

Для выполнения лабораторной работы №5 следует изучить соответствующие разделы главы 1. Задания нужно выполнять на языке **Visual C++** в среде **Microsoft Visual Studio 2008.**

В ходе выполнения контрольной работы необходимо, определив свой вариант, выполнить задания, перечисленные в таблице.

Тема / **Вариант**	Обработка символьных строк	Структуры
1	9-1	10-1
2	9-2	10-2
3	9-3	10-3
4	9-4	10-4
5	9-5	10-5
6	9-6	10-6
7	9-7	10-7
8	9-8	10-8
9	9-9	10-9
10	9-10	10-10
11	9-11	10-11
12	9-12	10-12
13	9-13	10-13
14	9-14	10-14
15	9-15	10-15
16	9-16	10-16
17	9-17	10-17
18	9-18	10-18
19	9-19	10-19
20	9-20	10-20
21	9-21	10-21
22	9-22	10-22
23	9-23	10-23
24	9-24	10-24
25	9-25	10-25

Раздел 9. «Обработка символьных строк»

Для решения задач этого раздела следует предварительно изучить *разделы 10, 11, 12 и 13 Главы 1*.

9-1. В символьную переменную вводится цифра. Вывести следующую и предыдущую цифры, считая, что за 9 следует 0, а, соответственно, нулю предшествует девятка.

9-2. Вывести в одну строку нечетные (по порядковому номеру) буквы латинского алфавита: *a c e g...*

9-3. Дан текст, заканчивающийся точкой. Является ли этот текст правильной записью целого числа (возможно, со знаком).

9-4. Дан текст, заканчивающийся точкой. Напечатать этот текст, удалив из него все цифры и знаки «+» или «–».

9-5. Составить программу, которая определит, является ли заданное слово перевертышем (например, «кок», «шалаш» являются)**.**

9-6 Составить программу, которая проверяет правописание «жи – ши» (т.е. если в заданном тексте после «ш» или «ж» встретится «ы», то программа должна выдавать сообщение об ошибке).

9-7. Составить программу, которая проверяет правописание «ча – ща» (т.е. если в заданном тексте после «ч» или «щ» встретится «я», то программа должна выдавать сообщение об ошибке).

9-8. Составить программу, которая введенное слово напечатает в обратном порядке (например, слово «упал» превратится в «лапу»).

9-9. Составить программу, которая во введенном слове удалит каждую вторую букву (т.е., например, для слова «символ» напечатает «смо»).

9-10. Составить программу, которая во введенном слове удалит заданную букву (например, если задано слово «трактор» и буква «т», то получится «ракор»).

9-11. Написать программу, которая в введенной с клавиатуры строке преобразует строчные (малые) буквы русского алфавита в прописные (заглавные).

9-12. Написать программу, которая проверяет, является ли введенная с клавиатуры строка двоичным числом.

9-13. Написать программу, которая проверяет, является ли введенная с клавиатуры строка 16-тиричным числом.

9-14. Написать программу, которая проверяет, является ли введенная с клавиатуры строка дробным числом.

В пяти последующих задачах одинаковое условие: Дана строка из *N* символов. Группы символов, разделенные пробелами (одним или несколькими) и не содержащие пробелов внутри себя, будем называть словами.

9-15. Подсчитать количество слов в данной последовательности.

9-16. Найти количество слов, начинающихся с буквы *б*.

9-17. Найти количество слов, оканчивающихся на букву *а*.

9-18. Найти количество слов, у которых первый и последний символы совпадают между собой.

9-19. Преобразовать данную последовательность, заменяя всякое вхождение слова это на слово то.

9-20. Найти длину самого короткого слова.

9-21. Написать программу, которая в словах с дефисом меняет местами части до и после дефиса.

9-22. Написать программу вывода на экран списка символов, из которых образован заданный текст.

9-23. Написать программу, определяющую в заданном тексте для каждой буквы алфавита количество ее употребления.

9-24. Написать программу, определяющую в заданном тексте для каждой буквы алфавита частоту ее употребления (отношение количества употреблений буквы к количеству всех букв в тексте)

9-25. Дан список из слов различной длины. Составить программу упорядочения слов по их длине.

Раздел 10. «Структуры»

Для решения задач этого раздела следует предварительно изучить *раздел 14 Главы 2* нашего учебного пособия «Информатика-2».

В задачах* 10-1 – 10-4 *одинаковое начало условия задачи:

Дан список учащихся из 10 записей. Каждая запись имеет поля: фамилия, имя, номер класса, буква класса.

10-1. Найти однофамильцев, обучающихся в одном классе.

10-2. Найти тезок (одинаковые имена), обучающихся в одном классе.

10-3. Найти двух учащихся, у которых совпадают имя и фамилия.

10-4. Вывести фамилию и первую букву имени для всех учеников указанного извне класса.

В задачах* 10-5 – 10-8 *одинаковое начало условия задачи:

Багаж пассажира характеризуется количеством вещей (целый тип) и общим весом вещей (вещественный тип). Дан список из сведений о багаже 10 пассажиров.

10-5. Найти багаж, средний вес одной вещи, в котором отличается не более, чем на 0,3 кг от общего среднего веса одной вещи по всему списку.

10-6. Найти число пассажиров, имеющих более двух вещей.

10-7. Число пассажиров, количество вещей которых превосходит среднее число вещей по всему списку.

10-8. Выяснить, имеется ли пассажир, багаж которого состоит из одной вещи весом менее 30 кг.

В задачах* 10-9–10-10 *одинаковое начало условия задачи:

Список книг состоит из 10 записей. Запись содержит поля: фамилия автора (тип string), название книги (тип строка), год издания (тип целый).

10-9. Найти названия книг данного автора, изданных с 1960 г.

10-10. Определить, имеются ли книги с названием «Информатика» и если да, то сообщить фамилии авторов, год издания этих книг.

В задачах* 10-11–10-16 *одинаковое начало условия задачи:

Дана структура, задающая дату вида:

```
Struct date {int day;
             int month;
             int year;};
```

Пользуясь таким структурным типом, составить программу, определяющую:

10-11. дату следующего (относительно сегодняшнего) дня;

10-12. дату предыдущего дня;

10-13. дату, которая наступит через *m* дней;

10-14. дату,которая была за *m* дней до сегодняшнего дня;

10-15. число суток, прошедших от заданной даты *d*1 до даты *d*2;

10-16. день недели, выпадающий на дату *d*1, если известно, что в первый день нашей эры был понедельник.

В задачах* 10-17–10-18 *одинаковое начало условия задачи:

Дана структура, задающая время, вида:

```
struct time { int hour;
   int min;
   int sec;};
```

Пользуясь таким структурным типом, составить программу:

10-17. определяющую предшествует ли время *t*1 времени *t*2 (в пределах суток);

10-18. присваивающую параметру *t*1 время, на 1 сек большее времени *t*2 (учесть смену суток).

В задачах* 10-19–10-20 *одинаковое начало условия задачи:

Дан список учащихся из 10 записей. Каждая запись имеет поля: фамилия, имя, номер класса (только 8-е и 10-е классы):

10-19. вывести отдельно учеников 10-х классов;

10-20. выяснить, на сколько человек в 8-х классах больше, чем в 9-х.

В задачах* 10-21–10-25 *одинаковое начало условия задачи:

Дан список учащихся из 10 записей. Каждая запись имеет поля: фамилия, имя, номер класса, оценки по трем предметам:

10-21. вывести отдельно отличников для указанного извне класса;

10-22. вывести отдельно двоечников (хотя бы одна двойка) для указанного извне класса;

10-23. вывести отдельно «хорошистов» (оценки 4 и 5) для указанного извне класса;

10-24. вывести отдельно «троечников» (имеется хотя бы одна тройка, но нет двоек) для указанного извне класса;

10-25. вывести список всех учеников с указанием среднего балла для каждого.

Лабораторная работа №6. Классы.

Шестая лабораторная работа посвящена разработке программ по разработке программ с использованием классов.

В ходе выполнения контрольной работы также необходимо, определив свой вариант, выполнить задания, перечисленные в таблице.

Тема / **Вариант**	Классы простые	Классы сложные
1	11-1	12-1
2	11-2	12-2
3	11-3	12-3
4	11-4	12-4
5	11-5	12-5
6	11-6	12-6
7	11-7	12-7
8	11-8	12-8
9	11-9	12-9
10	11-10	12-10
11	11-11	12-11
12	11-12	12-12
13	11-13	12-13
14	11-14	12-14
15	11-15	12-15
16	11-16	12-16
17	11-17	12-17
18	11-18	12-18
19	11-19	12-19
20	11-20	12-20
21	11-21	12-21
22	11-22	12-22
23	11-23	12-23
24	11-24	12-24
25	11-25	12-25

Раздел 11. «Классы простые»

Для решения задач этого раздела следует предварительно изучить *раздел 15 Главы 1*.

Для всех рассматриваемых ниже заданий разработать класс, содержащий два члена (назовем их `first`, `second`), и следующие методы:

– ввод с клавиатуры `Read`;

– вывод на экран `Display`;

– метод, указанный в задании.

Для полной ясности приведем решение следующего задания:

Поле `first` – вещественное число; поле `second` – целое число, показатель степени. Реализовать метод `power`( ) – возведение числа `first` в степень `second`.

```
       #include <iostream>
       #include <windows.h>
       char* Rus(const char* text);
       using namespace std;
class stepen
       { public:
         float first;
         int second;
         void power();
         void Display();
         void Read ();    };
   int n; float rez;
int main()
{   stepen a;
         a.Read();
         a.power();
         a.Display();
         return 0;
}
       char bufRus[256];
       char* Rus(const char* text)
       { AnsiToOem(text, bufRus); return bufRus;}
void stepen::Read()
  {cout<<Rus("Введи число first "); cin>>first;
  cout<<Rus("Введи число second "); cin>>second;  }
void stepen::Display()
  {cout<<Rus("\n Результат=")<<rez<<endl;  }
void stepen::power()
```

```
{ rez=1;
  for (int i=1; i<=second; i++) {rez=rez*first;} }
```

11-1. Элемент a_i арифметической прогрессии вычисляется по формуле $a_{i+1} = a_i + d$, $i = 0,1,2, \ldots$ Поле `first` – вещественное число, первый элемент прогрессии a_0; поле `second` – разность прогрессии `d`. Определить метод `element_i()` – для вычисления заданного элемента прогрессии.

11-2. Поле `first` – вещественное число; поле `second` – вещественное число, показатель степени. Реализовать метод `power( )` – возведение числа `first` в степень `second`.

11-3. Поле `first` – целое положительное число, числитель; поле `second` – целое положительное число, знаменатель. Реализовать метод `ipart( )` – выделение целой части дроби `first/second`.

11-4. Поле `first` – целое положительное число, номинал купюры; номинал может принимать значения 1, 2, 5, 10, 50, 100, 500, 1000, 5000. Поле `second` – целое положительное число, количество купюр данного достоинства. Реализовать метод `summa( )` – вычисление денежной суммы.

11-5. Поле `first` – вещественное положительное число, цена товара; поле `second` – целое положительное число, количество единиц товара. Реализовать метод `cost( )` – вычисление стоимости товара.

11-6. Поле `first` – целое положительное число, калорийность 100 г продукта; поле `second` – вещественное положительное число, масса продукта в килограммах. Реализовать метод `Power( )` – вычисление общей калорийности продукта.

11-7. Поле `first` – вещественное число, левая граница диапазона; поле `second` – вещественное число, правая граница диапазона. Реализовать метод `rangecheck( )` – проверку заданного числа на принадлежность диапазону.

11-8. Поле `first` – целое число, левая граница диапазона, включается в диапазон; поле `second` – целое число, правая граница диапазона, не включается в диапазон. Пара чисел представляет полуоткрытый интервал [`first`, `second`). Реализовать метод `rangecheck( )` – проверку заданного числа на принадлежность диапазону.

11-9. Поле `first` – целое положительное число, часы; поле `second` – целое положительное число, минуты. Реализовать метод `minutes( )` – приведение времени в минуты.

11-10. Линейное уравнение `y = Ax + B`. Поле `first` – вещественное число, коэффициент `A`; поле `second` – вещественное число, коэффициент `B`. Реализовать метод `function( )` – вычисление для заданного `x` значение функции `y`.

11-11. Линейное уравнение `y = Ax + B`. Поле `first` – вещественное число, коэффициент `A`; поле `second` – вещественное число, коэффициент `B`. Реализовать метод `function( )` – вычисление корня линейного уравнения.

11-12. Поле `first` – вещественное число, координата *x* точки на плоскости; поле `second` - вещественное число, координата *y* точки на плоскости. Реализовать метод `distance( )` – расстояние точки от начала координат.

11-13. Поле `first` – вещественное число, катет *a* прямоугольного треугольника; поле `second` – вещественное число, катет *b* прямоугольного треугольника. Реализовать метод `hypotenuse()` – вычисление гипотенузы.

11-14. Поле `first` – вещественное положительное число, оклад; поле `second` – целое положительное число, количество отработанных дней в месяце. Реализовать метод `summa ( )` – вычисление начисленной зарплаты для заданного месяца: `оклад/ дни_месяца * отработанные_дни`.

11-15. Поле `first` – целое положительное число, продолжительность телефонного разговора в минутах; поле `second` – вещественно положительное число, стоимость одной минуты в рублях. Реализовать метод `cost( )` – вычисление общей стоимости разговора.

11-16. Поле `first` – вещественное число, целая часть числа; поле `second` – положительное вещественное число, дробная часть числа. Реализовать метод `multiply( )` – умножение на произвольное вещественное число типа `double`.

11-17. Поле `first` – целое положительное число, числитель; поле `second` – целое положительное число, знаменатель. Реализовать метод `nesokr( )` – приведение дроби `first/second` к несократимому виду.

11-18. Поле `first` – целое число, целая часть числа; поле `second` – положительное целое число, дробная часть числа. Реализовать метод `multiply( )` – умножение на произвольное целое число типа `int`.

11-19. Число сочетаний из `n` объектов по `k` объектов (`k<n`) вычисляется по формуле: `C(n,k)=n!/((n - k)!*k!)`. Поле `first` – целое положительное число, `k`; поле `second` – положительное целое число, `n`. Реализовать метод `combination( )` – вычисление `C(n,k)`.

11-20. Элемент a_j геометрической прогрессии вычисляется по формуле $a_j = a_0 * r$, `j = 0,1,2, …` Поле `first` – вещественное число, первый элемент прогрессии a_0; поле `second` – знаменатель прогрессии, `r`. Определить метод `elementj( )` – для вычисления заданного элемента прогрессии.

11-21. Поле `first` – целое число, целая часть числа, записанного в восьмеричной системе счисления; поле `second` – положительное целое число, дробная часть числа, записанного в восьмеричной системе счисления. Реализовать метод `add8( )` – сложение чисел в восьмеричной системе.

11-22. Поле `first` – целое число, целая часть числа, записанного в восьмеричной системе счисления; поле `second` – положительное целое число, дробная часть числа, записанного в восьмеричной системе счисления. Реализовать метод `mult8( )` – умножение чисел в восьмеричной системе.

11-23. Поле `first` – целое число, целая часть числа, записанного в двоичной системе счисления; поле `second` – положительное целое число, дробная часть числа, записанного в двоичной системе счисления. Реализовать метод `add2( )` – сложение чисел в двоичной системе.

11-24. Поле `first` – целое число, целая часть числа, записанного в двоичной системе счисления; поле `second` – положительное целое число, дробная часть числа, записанного в двоичной системе счисления. Реализовать метод `mult2( )` – умножение чисел в двоичной системе.

11-25. Поле `first` – целое число, целая часть числа, записанного в 16-ричной системе счисления; поле `second` – положительное целое число, дробная часть числа, записанного в 16-ричной системе счисления. Реализовать метод `mult2( )` – сложение чисел в 16-ричной системе.

Раздел 12. «Классы сложные»

Для решения задач этого раздела следует предварительно изучить *раздел 15 Главы 1*.

Для всех рассматриваемых ниже заданий разработать класс с нужным числом членов, содержащий, помимо указанных в задании, следующие методы:

– ввод с клавиатуры `vvod`;

– вывод на экран `vyvod`.

Применить созданный класс для решения конкретных задач.

Для полной ясности приведем решение следующего примера:

Комплексное число представляется парой действительных чисел (`a,b`), *где* `a` – *действительная часть*, `b` – *мнимая часть:* `a+b*i`, *здесь* `i` – *мнимая единица*, `i=sqrt(-1)`. *Реализовать класс* `Complex`() *для работы с комплексными числами. Обязательно должны присутствовать операции: Сложение* `add`, (`a,b`) + (`c,d`) = (`a+c`, `b+d`).

Вычитание `sub`, (`a,b`) – (`c,d`) = (`a – b`, `c – d`).

Умножение `mul`, (`a,b`) * (`c,d`) = (`ac – bd, ad+bc`).

Деление div, (a,b) / (c,d) = (ac+bd, bc – ad)/ ($c^2 + d^2$).
Сравнение equ, (a,b) = (c,d), *если* (a=c) *и* (b=d).
Сопряженное число conj(a,b) = (a, –b).

```
//арифм. операции с комплексными числами
#include <iostream>
#include <windows.h>
using namespace std;
char* Rus(const char* text);
class Complex
{ public:
  float a,b;
void vvod ()
 {cout<<Rus("\n Введи действ.часть ");cin>>a;
  cout<<Rus("Введи мнимую часть "); cin>>b;}
void vyvod ()
  {char sig='+'; if (b<0) sig='  ';
 cout<<Rus("\n комплексное число=")<<a<<sig<<b<<"i";}
void mul(Complex c1, Complex c2)
  { a= c1.a*c2.a - c1.b*c2.b;
    b=c1.a*c2.b+c1.b*c2.a;    }
void add(Complex c1, Complex c2)
  { a=c1.a+c2.a; b=c1.b+c2.b; }
void sub(Complex c1, Complex c2)
  { a=c1.a-c2.a; b=c1.b-c2.b; }
void div(Complex c1, Complex c2)
  { a=(c1.a*c2.a+c1.b*c2.b)/(c2.a*c2.a+c2.b*c2.b);
    b=(c1.b*c2.a-c1.a*c2.b)/(c2.a*c2.a+c2.b*c2.b);}
void conj(Complex c)
  { a=c.a; b=-c.b;   }
};
int equ(Complex c1, Complex c2);
int equ(Complex c1, Complex c2)
  { if ((c1.a= =c2.a)&&(c1.b= =c2.b)) return (1);
      else return (0); }
char bufRus[256];
char* Rus(const char* text)
  { CharToOem(text,bufRus); return bufRus;}
int main()
{   Complex c1,c2,c; int r;
  cout<<"\n Vvodi 1-e chislo ";
      c1.vvod();
  cout<<"Vvodi 2-e chislo ";
```

```
      c2.vvod();
   cout<<"\n 1-e chislo=";
      c1.vyvod();
   cout<<" \n 2-e chislo="; c2.vyvod(); cout<<endl;
      c.add(c1,c2);
   cout<<"\n Rez slojen: ";    c.vyvod();
      c.sub(c1,c2);
   cout<<"\n Rez vychit: ";    c.vyvod();
      c.mul(c1,c2);
   cout<<"\n Rez umnojen: ";   c.vyvod();
      c.div(c1,c2);
   cout<<"\n Rez delen: ";     c.vyvod();
      r=equ(c1,c2);
   if (r= =1) cout<<"\n c1=c2 ";
   else cout<<"\n c1 <> c2 ";    c.conj(c1);
   cout<<"\n Rez sopryaj c1: "; c.vyvod();cout<<"\n";
return 0;}
```

12-1. Создать класс `EngMer` для работы с английскими мерами длины: фунтами и дюймами, при этом учтем, что 1 фунт = 12 дюймов. Длина объекта будет задаваться парой чисел (фунты и дюймы), нужно реализовать: сложение и вычитание длин, умножение и деление длин, сравнение длин.

12-2. Создать класс `vector3D`, задаваемый тройкой координат. Обязательно должны быть реализованы: сложение и вычитание векторов, скалярное произведение векторов, умножение на скаляр, сравнение векторов, вычисление длины вектора, сравнение длины векторов.

12-3. Создать класс `EngMoney` для работы с устаревшей денежной системой Великобритании. В ней использовались фунты, шиллинги и пенсы. При этом: 1 фунт = 20 шиллингов, 1 шиллинг =12 пенсов. Денежные суммы будут задаваться в фунтах, шиллингах и пенсах и результат выдаваться также в этих величинах. Должны быть реализованы: сложение и вычитание, умножение и деление, сравнение сумм.

12-4. Создать класс `Money` для работы с денежными суммами. Число должно быть представлено двумя полями: типа `long int` для рублей и типа `int` – для копеек. Дробная часть (копейки) при выводе на экран должна быть отделена от целой части запятой. Реализовать сложение, вычитание, деление сумм, деление суммы на дробное число, умножение на дробное число и операцию сравнения.

12-5. Создать класс `Triangle` для представления треугольника. Поля данных должны включать углы и стороны. Требуется реализовать операции: получения и изменения полей данных, вычисления площади, вы-

числения периметра, вычисления высот, а также определения вида прямоугольника (равносторонний, равнобедренный или прямоугольный).

12-6. Создать класс `Angle` для работы с углами на плоскости, задаваемыми величинами в градусах и минутах. Обязательно должны быть реализованы: перевод в радианы, приведение к диапазону 0–360, увеличение и уменьшение угла на заданную величину, получение синуса, сравнение углов.

12-7. Создать класс `Point` для работы с точками на плоскости. координаты точки – декартовы. Обязательно должны быть реализованы: перемещение точки по оси X, перемещение по оси Y, определение расстояния до начала координат, расстояние между двумя точками.

12-8. Рациональная (несократимая) дробь представляется парой целых чисел (a,b), где a – числитель, b – знаменатель. Создать класс `Rational` для работы с рациональными дробями. Обязательно должны быть реализованы операции: сложения, вычитания, умножения, деления и сравнения дробей, причем результат должен приводиться в виде несократимой дроби (т.е. результат нужно сокращать).

12-9. Создать класс `Date` для работы с датами в формате «год.месяц.день». Дата представляется структурой с тремя полями типа `unsigned int`: для года, месяца и дня. Класс должен включать операции: вычисление даты через заданное количество дней от указанной, вычитание заданного количества дней из даты, определение високосного года, присвоение и получение отдельных частей (год, месяц, день), сравнение дат (равно, до, после), вычисление количества дней между датами.

12-10. Создать класс `Time` для работы со временем в формате «час:минута:секунда». Обязательными операциями являются: вычисление разницы между двумя моментами времени в секундах, сложение времени и заданного количества секунд, вычитание из времени заданного количества секунд, сравнение моментов времени, перевод в секунды, перевод в минуты (с округлением до целой минуты).

12-11. В морской навигации координаты точки измеряются в градусах и минутах широты и долготы. Один градус равен 60 минутам (ранее минуту делили на 60 секунд, но сейчас минуту делят на обычные десятичные доли). Долгота измеряется от 0 до 180° восточнее (E) или западнее (W) Гринвича. Широта принимает значения от 0 до 90° севернее (N) или южнее (S) экватора. Создать класс `Koord`, включающий следующие три поля: типа `int` для числа градусов, типа `float` для числа минут и типа char для указания направления (N, S, W или E). Объект этого класса может содержать значение как широты, так и долготы. Предусмотреть ввод координат точки, указания полушария для указанной точки, вычисления расстояния между двумя точками. Учитывать, что по широте 1° равен 111 км, а по долготе 1° равен 111 км * cos (широты).

12-12. Реализовать класс `Account`, представляющий собой банковский счет. В классе должны быть реализованы 4 поля: фамилия владельца, номер счета, процент начисления и сумма в рублях. Необходимо выполнять следующие операции: сменить владельца счета, снять некоторую сумму со счета, положить деньги на счет, начислить проценты, перевести сумму в доллары, перевести сумму в евро, получить сумму прописью (преобразовать в числительное).

12-13. Номиналы российских рублей могут принимать значения 1, 2, 5, 10, 50, 100, 500, 1000, 5000. Копейки представить как 0,01 (1 копейка), 0,05 (5 копеек), 0,1 (10 копеек), 0,5 (50 копеек). Создать класс `Money` для работы с денежными суммами. Сумма должна быть представлена полями-номиналами, значениями которых должны быть количества купюр данного достоинства. Реализовать сложение сумм, вычитание сумм, деление сумм, деление суммы на дробное число, умножение на дробное число и операцию сравнения. Дробная часть (копейки) при выводе на экран должна быть отделена от целой части запятой.

12-14. Реализовать класс `Bankomat`, моделирующий работу банкомата. В классе должны содержаться поля для хранения идентификационного номера банкомата, информации о текущей сумме денег, оставшейся в банкомате, минимальной и максимальной суммах, которые позволяется снять клиенту в один день. Сумма денег представляется полями-номиналами 10–1000 (см. задание 12.13). Реализовать метод загрузки купюр в банкомат и метод снятия определенной суммы денег. Метод снятия денег должен проверять на корректность снимаемую сумму (не должна быть меньше минимально допустимой и больше максимально допустимой). Должен быть метод для отображения оставшейся в банкомате суммы.

12-15. Создать класс `Fraction` для работы с дробными числами. Число должно быть представлено двумя полями: целая часть – длинное целое со знаком, дробная часть – беззнаковое короткое целое. Реализовать арифметические операции сложения, вычитания, умножения и операции сравнения.

12-16. Создать класс `Goods` (товар). В классе должны быть представлены поля: наименование товара, дата оформления, цена товара, количество единиц товара, номер накладной, по которой товар поступил на склад. Реализовать методы изменения цены товара, изменения количества товара (увеличения и уменьшения), вычисления стоимости товара. Должен быть метод для отображения стоимости товара в виде строки.

12-17. Создать класс `BitString` для работы с 64-битовыми строками. Битовая строка должна быть представлена двумя полями типа unsigned long. Должны быть реализованы все традиционные операции для работы с битами: and, or, xor, not. Реализовать сдвиг влево shiftLeft и сдвиг вправо shiftRight на заданное количество битов.

12-18. Создать класс `LongLong` для работы с целыми числами из 64 бит. Число должно быть представлено двумя полями: long – старшая часть, unsigned long – младшая часть. Должны быть реализованы арифметические операции (сложение, вычитание, умножение и деление) и сравнение.

12-19. Создать класс `Payment` (зарплата). В классе должны быть представлены поля: фамилия-имя-отчество, оклад, год поступления на работу, процент надбавки, подоходный налог, количество отработанных дней в месяце, количество рабочих дней в месяце, начисленная и удержанная сумма. Реализовать методы: вычисления начисленной суммы, вычисления удержанной суммы, вычисления суммы, выдававаемой на руки, вычисления стажа. Стаж вычисляется как полное количество лет, прошедших от года поступления на работу, до текущего года. Начисления представляют собой сумму, начисленную за отработанные дни, и надбавки. Удержания представляют собой отчисления в пенсионный фонд (1% от начисленной суммы) и подоходный налог (13%).

12-20. Создать класс `Ship`, который будет содержать данные об учетном номере корабля и его координатах. Для хранения координат используйте поля типа `Koord` (см. задание 12.11). Разработайте методы для ввода данных о корабле, о выводе его координат (с указанием полушария), метод для вычисления расстояния между кораблями (см. задание 12.11).

12-21. Создать класс `Binary1`, который будет содержать число в двоичной системе (в отдельном поле – целая часть, в другом поле – дробная часть). Разработайте методы для ввода двоичных чисел (с дробной частью!), вывода двоичных чисел, методы для вычисления суммы и произведения двоичных чисел.

12-22. Создать класс `Binary2`, который будет содержать число в двоичной системе (в отдельном поле – целая часть, в другом поле – дробная часть). Разработайте методы для ввода двоичных чисел (с дробной частью!), вывода двоичных чисел, методы для вычисления разности и деления двоичных чисел.

12-23. Создать класс `Hexadec1`, который будет содержать число в 16-ричной системе (в отдельном поле – целая часть, в другом поле – дробная часть). Разработайте методы для ввода 16-ричных чисел (с дробной частью!), вывода 16-ричных чисел, методы для вычисления суммы и произведения 16-ричных чисел.

12-24. Создать класс `Hexadec2`, который будет содержать число в 16-ричной системе (в отдельном поле – целая часть, в другом поле – дробная часть). Разработайте методы для ввода 16-ричных чисел (с дробной частью!),
вывода 16-ричных чисел, методы для вычисления разности и деления 16-ричных чисел.

12-25. Создать класс `Oct1`, который будет содержать число в 8-ричной системе (в отдельном поле – целая часть, в другом поле – дробная часть). Разработайте методы для ввода 8-ричных чисел (с дробной частью!), вывода 8-ричных чисел, методы для вычисления суммы и произведения 8-ричных чисел.

Лабораторная работа №7. Обработка файлов.

Седьмая лабораторная работа посвящена разработке программ по обработке файлов, в том числе с использованием классов.

В ходе выполнения контрольной работы также необходимо, определив свой вариант, выполнить задания, перечисленные в таблице.

Тема / **Вариант**	Файлы	Файлы с классами
1	13-1	14-1
2	13-2	14-2
3	13-3	14-3
4	13-4	14-4
5	13-5	14-5
6	13-6	14-6
7	13-7	14-7
8	13-8	14-8
9	13-9	14-9
10	13-10	14-10
11	13-11	14-11
12	13-12	14-12
13	13-13	14-13
14	13-14	14-14
15	13-15	14-15
16	13-16	14-16
17	13-17	14-17
18	13-18	14-18
19	13-19	14-19
20	13-20	14-20
21	13-21	14-21
22	13-22	14-22
23	13-23	14-23
24	13-24	14-24
25	13-25	14-25

Раздел 13. «Файлы текстовые»

Для решения задач этого раздела следует предварительно изучить *раздел 16 Главы 1*.

Следующие задания требуется решить с использованием текстовых файлов. Во всех заданиях предусмотреть в программе:

- формирование текстового файла, записав в него 20 случайных чисел от –10 до +10, по одному на строке.
- Используя созданный выше файл как входной, сформировать выходной файл по указанному для каждого варианта правилу.

13-1. Записать выходной файл, прибавив к каждому числу последнее число файла

13-2. Записать выходной файл, разделив каждое число на полусумму первого отрицательного и 10-го числа файла

13-3. Записать выходной файл, вычтя из каждого числа наибольшее число файла

13-4. Записать выходной файл, умножив каждое число на минимальное число из файла.

13-5. Записать выходной файл, разделив все нечетные по абсолютной величине числа на среднее арифметическое

13-6. Записать выходной файл, вычтя из каждого числа сумму чисел файла.

13-7. Записать выходной файл, умножив каждое третье число на удвоенную сумму первого и последнего отрицательных чисел.

13-8. Записать выходной файл, добавив к каждому числу первое нечетное по абсолютной величине число файла.

13-9. Записать выходной файл, умножив каждое четное число на первое отрицательное число файла.

13-10. Записать выходной файл, добавив к каждому числу половину последнего отрицательного числа файла.

13-11. Записать выходной файл, разделив все числа на половину максимального числа.

13-12. Записать выходной файл, заменив все нули средним арифметическим.

13-13. Записать выходной файл, заменив все положительные числа квадратом минимума.

13-14. Записать выходной файл, добавив к каждому числу полусумму всех отрицательных чисел.

13-15. Записать выходной файл, добавив к каждому числу среднее арифметическое наименьшего по абсолютной величине и наибольшего из чисел файла.

13-16. Записать выходной файл, разделив каждое число на минимум и добавив максимум.

13-17. Записать выходной файл, заменив все положительные числа на максимум.

13-18. Записать выходной файл, заменив каждое четное число по абсолютной величине на разность максимума и минимума.

13-19. Записать выходной файл, заменив каждое второе отрицательное число половиной максимума.

13-20. Записать выходной файл, заменив каждое третье положительное число средним арифметическим отрицательных чисел.

13-21. Записать выходной файл, заменив каждое положительное число средним арифметическим положительных чисел.

13-22. Записать выходной файл, заменив каждое положительное число средним арифметическим отрицательных чисел.

13-23. Записать выходной файл, заменив каждое отрицательное число средним арифметическим отрицательных чисел.

13-24. Записать выходной файл, заменив каждое отрицательное число средним арифметическим положительных чисел.

13-25. Записать выходной файл, заменив каждое положительное число этим же числом, разделенным на максимальное из всех чисел.

Раздел 14. «Файлы и классы»

Для решения задач этого раздела следует предварительно изучить *раздел 16 Главы 1*.

Следующие задания требуется решить с использованием классов. При этом обязательно оформить методы для выполнения каждого из действий: по вводу данных, выводу их в файл, чтению данных из файла и выводу их на экран, сортировке данных.

14-1. Дана структура с именем STUDENT, состоящая из полей:

- фамилия и инициалы,
- номер группы,
- успеваемость (массив из пяти элементов).

Написать программу, которая выполняет следующие действия:

• ввод с клавиатуры данных в массив, состоящий из 10 структур типа STUDENT, и занесение их в файл данных;

• чтение данных из файла и вывод их на экран;

• вывод на экран фамилий и номеров групп для всех студентов, включенных в массив, если средний балл студента больше 4 (если таких нет – вывести об этом сообщение).

• Список студентов должен быть упорядочен по возрастанию номера группы.

14-2. Дана структура с именем STUDENT, состоящая из полей:

• фамилия и инициалы;

• номер группы;

• успеваемость (массив из пяти элементов).

Написать программу, которая выполняет следующие действия:

• ввод с клавиатуры данных в массив, состоящий из 10 структур типа STUDENT, и занесение их в файл данных;

• чтение данных из файла и вывод их на экран;

• вывод на экран фамилий и номеров групп для всех студентов, имеющих оценки 4 и 5 (если таких нет – вывести об этом сообщение);

• список студентов должен быть упорядочен по убыванию среднего балла.

14-3. Дана структура с именем STUDENT, состоящая из полей:

• фамилия и инициалы;

• номер группы;

• успеваемость (массив из пяти элементов).

Написать программу, которая выполняет следующие действия:

• ввод с клавиатуры данных в массив, состоящий из 10 структур типа STUDENT, и занесение их в файл данных;

• чтение данных из файла и и вывод их на экран;

• вывод на экран фамилий и номеров групп для всех студентов, имеющих хотя бы одну оценку 2 (если таких нет – вывести об этом сообщение);

• список студентов должен быть упорядочен по алфавиту фамилий.

14-4. Дана структура с именем AEROFLOT, состоящая из полей:

• название пункта назначения рейса;

• номер рейса;

• тип самолета.

Написать программу, которая выполняет следующие действия:

• ввод с клавиатуры данных в массив, состоящий из 7 элементов типа AEROFLOT, и занесение их в файл данных;

• чтение данных из файла и вывод их на экран;

• вывод на экран номеров рейсов и типов самолетов, вылетающих в пункт назначения, название которого совпало с названием, введенным с клавиатуры (если таких нет – вывести об этом сообщение);

• список должен быть упорядочен по возрастанию номера рейса.

14-5. Дана структура с именем AEROFLOT, состоящая из полей:

• название пункта назначения рейса;

• номер рейса;

• тип самолета.

Написать программу, которая выполняет следующие действия:

• ввод с клавиатуры данных в массив, состоящий из 7 элементов типа AEROFLOT, и занесение их в файл данных;

• чтение данных из файла и вывод их на экран;

• вывод на экран пунктов назначения и номеров рейсов, обслуживаемых самолетом, тип которого введен с клавиатуры (если таких нет – вывести об этом сообщение);

• список должен быть упорядочен по алфавиту названий пунктов назначения.

14-6. Дана структура с именем WORKER, состоящая из полей:

• фамилия и инициалы работника;

• название занимаемой должности;

• год поступления на работу.

Написать программу, которая выполняет следующие действия:

• ввод с клавиатуры данных в массив, состоящий из 10 элементов типа WORKER, и занесение их в файл данных;

• чтение данных из файла и вывод их на экран;

• вывод на экран фамилий работников, чей стаж работы в организации превышает значение, введенное с клавиатуры (если таких нет – вывести об этом сообщение);

• список должен быть упорядочен по алфавиту фамилий.

14-7. Дана структура с именем TRAIN, состоящая из полей:

• название пункта назначения;

• номер поезда;

• время отправления.

Написать программу, которая выполняет следующие действия:

• ввод с клавиатуры данных в массив, состоящий из 8 элементов типа TRAIN, и занесение их в файл данных;

• чтение данных из файла и вывод их на экран;

• вывод на экран информации о поездах, отправляющихся после введенного с клавиатуры времени (если таких нет – вывести об этом сообщение);

• список должен быть упорядочен по алфавиту пунктов назначения.

14-8. Дана структура с именем TRAIN, состоящая из полей:

• название пункта назначения;

• номер поезда;

• время отправления.

Написать программу, которая выполняет следующие действия:

• ввод с клавиатуры данных в массив, состоящий из 6 элементов типа TRAIN, и занесение их в файл данных;

• чтение данных из файла и вывод их на экран;

• вывод на экран информации о поездах, отправляющихся в пункт, название которого введено с клавиатуры (если таких нет – вывести об этом сообщение);

• список должен быть упорядочен по времени отправления поезда.

14-9. Дана структура с именем TRAIN, состоящая из полей:

• название пункта назначения;

• номер поезда;

• время отправления.

Написать программу, которая выполняет следующие действия:

• ввод с клавиатуры данных в массив, состоящий из 8 элементов типа TRAIN, и занесение их в файл данных;

• чтение данных из файла и вывод их на экран;

• вывод на экран информации о поезде, номер которого введен с клавиатуры (если таких нет – вывести об этом сообщение);

• список должен быть упорядочен по номерам поездов.

14-10. Дана структура с именем MARSH, состоящая из полей:

• название начального пункта назначения;

• название конечного пункта назначения;

• номер маршрута.

Написать программу, которая выполняет следующие действия:

• ввод с клавиатуры данных в массив, состоящий из 8 элементов типа MARSH, и занесение их в файл данных;

• чтение данных из файла и вывод их на экран;

• вывод на экран информации о маршруте, номер которого введен с клавиатуры (если таких нет – вывести об этом сообщение);

• список должен быть упорядочен по номерам маршрутов.

14-11. Дана структура с именем MARSH, состоящая из полей:

• название начального пункта назначения;

• название конечного пункта назначения;

• номер маршрута.

Написать программу, которая выполняет следующие действия:

• ввод с клавиатуры данных в массив, состоящий из 8 элементов типа MARSH, и занесение их в файл данных;

• чтение данных из файла и вывод их на экран;

• вывод на экран информации о маршрутах, которые начинаются или заканчиваются в пункте, название которого введено с клавиатуры (если таких нет – вывести об этом сообщение);

• список должен быть упорядочен по номерам маршрутов.

14-12. Дана структура с именем NOTE, состоящая из полей:

• фамилия, имя;

• номер телефона;

• дата рождения (массив из трех чисел).

Написать программу, которая выполняет следующие действия:

• ввод с клавиатуры данных в массив, состоящий из 8 элементов типа NOTE, и занесение их в файл данных;

• чтение данных из файла и вывод их на экран;

• вывод на экран информации о человеке, номер телефона которого введено с клавиатуры (если таких нет – вывести об этом сообщение);

• список должен быть упорядочен по датам рождения.

14-13. Дана структура с именем NOTE, состоящая из полей:

• фамилия, имя;

• номер телефона;

• дата рождения (массив из трех чисел).

Написать программу, которая выполняет следующие действия:

• ввод с клавиатуры данных в массив, состоящий из 8 элементов типа NOTE, и занесение их в файл данных;

• чтение данных из файла и вывод их на экран;

• вывод на экран информации о людях, чьи дни рождения приходятся на месяц, значение которого введено с клавиатуры (если таких нет – вывести об этом сообщение);

• список должен быть упорядочен по алфавиту.

14-14. Дана структура с именем NOTE, состоящая из полей:

• фамилия, имя;

• номер телефона;

• дата рождения (массив из трех чисел).

Написать программу, которая выполняет следующие действия:

• ввод с клавиатуры данных в массив, состоящий из 8 элементов типа NOTE, и занесение их в файл данных;

• чтение данных из файла и вывод их на экран;

• вывод на экран информации о человеке, чья фамилия введена с клавиатуры (если таких нет – вывести об этом сообщение);

• список должен быть упорядочен по двум первым цифрам номера телефона.

14-15. Дана структура с именем ZNAK, состоящая из полей:

• фамилия, имя;

• знак Зодиака;

• дата рождения (массив из трех чисел).

Написать программу, которая выполняет следующие действия:

• ввод с клавиатуры данных в массив, состоящий из 8 элементов типа ZNAK, и занесение их в файл данных;

• чтение данных из файла и вывод их на экран;

• вывод на экран информации о человеке, чья фамилия введена с клавиатуры (если таких нет – вывести об этом сообщение);

• список должен быть упорядочен по датам рождения.

14-16. Дана структура с именем ZNAK, состоящая из полей:

• фамилия, имя;

• знак Зодиака;

• дата рождения (массив из трех чисел).

Написать программу, которая выполняет следующие действия:

• ввод с клавиатуры данных в массив, состоящий из 8 элементов типа ZNAK, и занесение их в файл данных;

• чтение данных из файла и вывод их на экран;

• вывод на экран информации о людях, родившихся под знаком, название которого введено с клавиатуры (если таких нет – вывести об этом сообщение);

• список должен быть упорядочен по датам рождения.

14-17. Дана структура с именем ZNAK, состоящая из полей:

• фамилия, имя;

• знак Зодиака;

• дата рождения (массив из трех чисел).

Написать программу, которая выполняет следующие действия:

• ввод с клавиатуры данных в массив, состоящий из 8 элементов типа ZNAK, и занесение их в файл данных;

• чтение данных из файла и вывод их на экран;

• вывод на экран информации о людях, родившихся в месяц, значение которого введено с клавиатуры (если таких нет – вывести об этом сообщение);

• список должен быть упорядочен по знакам Зодиака.

14-18. Дана структура с именем PRICE, состоящая из полей:

• название товара;

• название магазина, в котором продается товар;

• стоимость товара в руб.

Написать программу, которая выполняет следующие действия:

• ввод с клавиатуры данных в массив, состоящий из 8 элементов типа PRICE, и занесение их в файл данных;

• чтение данных из файла и вывод их на экран;

• вывод на экран информации о товаре, название которого введено с клавиатуры (если таких нет – вывести об этом сообщение);

• список должен быть упорядочен по алфавиту названий товара.

14-19. Дана структура с именем PRICE, состоящая из полей:

• название товара;

• название магазина, в котором продается товар;

• стоимость товара в руб.

Написать программу, которая выполняет следующие действия:

• ввод с клавиатуры данных в массив, состоящий из 8 элементов типа PRICE, и занесение их в файл данных;

• чтение данных из файла и вывод их на экран;

• вывод на экран информации о товарах, продающихся в магазине, название которого введено с клавиатуры (если таких нет – вывести об этом сообщение);

• список должен быть упорядочен по алфавиту названий магазинов.

14-20. Дана структура с именем ORDER, состоящая из полей:

• расчетный счет плательщика;

• расчетный счет получателя;

• перечисляемая сумма в руб.

Написать программу, которая выполняет следующие действия:

• ввод с клавиатуры данных в массив, состоящий из 8 элементов типа ORDER, и занесение их в файл данных;

• чтение данных из файла и вывод их на экран;

• вывод на экран информации о сумме, снятой с расчетного счета плательщика, введенного с клавиатуры (если таких нет – вывести об этом сообщение);

• список должен быть упорядочен по расчетным счетам плательщиков.

14-21. Дана структура с именем BOOKS, состоящая из полей:

• автор книги;

• название книги;

• место издания;

• издательство;

• год издания.

Написать программу, которая выполняет следующие действия:

• ввод с клавиатуры данных в массив, состоящий из 8 элементов типа BOOKS, и занесение их в файл данных;

• чтение данных из файла и вывод их на экран;

• вывод на экран книг автора, который введен с клавиатуры (если таких нет – вывести об этом сообщение);

• список должен быть упорядочен по алфавиту названий издательств.

14-22. Дана структура с именем BOOKS, состоящая из полей:

• автор книги;

• название книги;

• место издания;

• издательство;

• год издания.

Написать программу, которая выполняет следующие действия:

• ввод с клавиатуры данных в массив, состоящий из 8 элементов типа BOOKS, и занесение их в файл данных;

• чтение данных из файла и вывод их на экран;

• вывод на экран книг автора, который введен с клавиатуры (если таких нет – вывести об этом сообщение);

• список должен быть упорядочен по алфавиту фамилий авторов.

14-23. Дана структура с именем BOOKS, состоящая из полей:

• автор книги;

• название книги;

• место издания;

• издательство;

• год издания.

Написать программу, которая выполняет следующие действия:

• ввод с клавиатуры данных в массив, состоящий из 8 элементов типа BOOKS, и занесение их в файл данных;

• чтение данных из файла и вывод их на экран;

• вывод на экран книг, изданных после года, который введен с клавиатуры (если таких нет – вывести об этом сообщение);

• список должен быть упорядочен по алфавиту названий книг.

14-24. Дана структура с именем BOOKS, состоящая из полей:

• автор книги;

• название книги;

• место издания;

• издательство;

• год издания.

Написать программу, которая выполняет следующие действия:

• ввод с клавиатуры данных в массив, состоящий из 8 элементов типа BOOKS, и занесение их в файл данных;

• чтение данных из файла и вывод их на экран;

• вывод на экран книг, изданных в городе, который введен с клавиатуры (если таких нет – вывести об этом сообщение);

• список должен быть упорядочен по годам издания.

14-25. Дана структура с именем SOTRUD, состоящая из полей:

• фамилия, имя сотрудника;

• должность;

• год рождения;

• год поступления на работу.

Написать программу, которая выполняет следующие действия:

• ввод с клавиатуры данных в массив, состоящий из 8 элементов типа SOTRUD, и занесение их в файл данных;

• чтение данных из файла и вывод их на экран;

• вывод на экран сведений сотрудников, которые имеют стаж работы, больший введенного с клавиатуры (если таких нет – вывести об этом сообщение);

• список должен быть упорядочен по алфавиту фамилий.

Лабораторная работа №8. Создание приложений Windows Forms.

Седьмая лабораторная работа посвящена разработке программ по обработке файлов, в том числе с использованием классов.

В ходе выполнения контрольной работы также необходимо, определив свой вариант, выполнить задания, перечисленные в таблице.

Тема / **Вариант**	Формы с кнопками	Формы с переключателями	Формы с прокруткой	Формы с меню и работой с файлами
1	15-1	16-1	17-1	18-1
2	15-2	16-2	17-2	18-2
3	15-3	16-3	17-3	18-3
4	15-4	16-4	17-4	18-4
5	15-5	16-5	17-5	18-5
6	15-6	16-6	17-6	18-6
7	15-7	16-7	17-7	18-7
8	15-8	16-8	17-8	18-8
9	15-9	16-9	17-9	18-9
10	15-10	16-10	17-10	18-10
11	15-11	16-11	17-11	18-11
12	15-12	16-12	17-12	18-12
13	15-13	16-13	17-13	18-13
14	15-14	16-14	17-14	18-14
15	15-15	16-15	17-15	18-15
16	15-16	16-16	17-16	18-16
17	15-17	16-17	17-17	18-17
18	15-18	16-18	17-18	18-18
19	15-19	16-19	17-19	18-19
20	15-20	16-20	17-20	18-20
21	15-21	16-21	17-21	18-21
22	15-22	16-22	17-22	18-22
23	15-23	16-23	17-23	18-23
24	15-24	16-24	17-24	18-24
25	15-25	16-25	17-25	18-25

Для выполнения заданий этой лабораторной следует предварительно изучить соотвествующий раздел *Главы 2*. Задания требуется выполнить, создавая **Приложения Windows Forms Visual C++**

Раздел 15. «Формы с кнопками»

15-1. Написать программу, которая вычисляет силу тока в электрической цепи. Рекомендуемый вид формы приведен на *рис. 15-1*. Программа должна быть спроектирована таким образом, чтобы кнопка **Вычислить** была доступна только в том случае, если пользователь ввел величину сопротивления.

Пример: $U = 10$, $R = 15$. Ответ: 0,6667

Рис. 15-1

15-2. Написать программу, которая вычисляет силу тока в электрической цепи (*рис. 15-2*). Цепь состоит из двух параллельно соединенных сопротивлений. Рекомендуемый вид формы:

Пример: $R_1 = 10$, $R_2 = 15$. Ответ: 6

Рис. 15-2

15-3. Написать программу, которая вычисляет силу тока в электрической цепи. Цепь состоит из двух последовательно соединенных сопротивлений. Рекомендуемый вид формы см. выше.

Пример: $R_1 = 10$, $R_2 = 15$. Ответ: 25

15-4. Найти массу x литров молока, если известно, что плотность молока p кг/м^3.

Пример: $x = 7$ л, $p = 1030$кг/м^3. Ответ: 7,21 кг

15-5. Объем цилиндра равен V, а площадь основания – S. Какова высота цилиндра H?

Пример: V = 10м3 , S = 5м^2. Ответ: 2 м

15-6. Дана длина ребра куба a. Найти объем куба V и площадь его боковой поверхности S.

Пример: a = 5 Ответ: V = 125, S = 100

15-7. Каков объем кислорода, содержащегося в комнате размером $a{\cdot}b{\cdot}c$, если кислород составляет 21% объема воздуха?

Пример: a = 3, b = 4, c = 5. Ответ: 12,6

15-8. Найти площадь равнобочной трапеции с основаниями a и b и углом при большем основании равным x.

Пример: a = 6, b = 5, x = 45°. Ответ: 2,75

15-9. Найти угол между отрезком прямой, соединяющей начало координат с точкой $A(x, y)$, и осью OX (точка лежит в 1-й четверти).

Пример: x = 3, y = 4. Ответ: 53,13°

15-10. Определить время падения камня на поверхность земли с высоты h.

Пример: h=10 м. Ответ: 1,4278 с

15-11. Три сопротивления R_1, R_2, R_3 соединены параллельно. Найти сопротивление соединения.

Пример: R_1 = 10, R_2 = 15, R_3 = 20. Ответ: 4,62

15-12. Написать программу вычисления площади параллелограмма. Извне вводятся стороны a,b и угол между ними x.

Пример: a = 10, b = 15, x = 30°. Ответ: 75

15-13. Написать программу вычисления объема прямоугольного параллелепипеда. Извне вводятся длина a, ширина b и высота c.

Пример: a = 10, b = 15, c = 20. Ответ: 3000

15-14. Написать программу вычисления площади поверхности прямоугольного параллелепипеда. Извне вводятся длина a, ширина b и высота c.

Пример: a = 10, b = 15, c = 20. Ответ: 1300

15-15. Написать программу вычисления объема цилиндра. Извне вводятся радиус основания R и высота цилиндра h.

Пример: R = 10, h = 15. Ответ: 4712,39

15-16. Написать программу вычисления стоимости покупки, состоящей из нескольких тетрадей и карандашей. Извне вводятся цена одной тетради *Ct* и количество тетрадей *Kt*, а также цена карандаша *Ck* и количество карандашей *Kk*. Пример: *Ct* = 1, *Kt* = 15, *Ck* = 0.2, *Kk* = 5. Ответ: 16

15-17. Написать программу вычисления стоимости покупки, состоящей из нескольких тетрадей и такого же количества обложек к ним. Извне вводятся цена одной тетради *Ct*, одной обложки *Cb* и количество тетрадей *Kt*.
Пример: *Ct* = 1.2, *Kt* = 15, *Cb* = 0.2. Ответ: 21

15-18. Написать программу вычисления стоимости некоторого количества (по весу) яблок. Извне вводятся цена одного килограмма яблок *C* и вес яблок *V*. Пример: *C* = 25, *V* = 15.5. Ответ: 37.5

15-19. Написать программу вычисления периметра и площади треугольника, заданного длинами сторон.
Пример: *a* = 3, *b* = 4, *c* = 5. Ответ: *P* = 12, *S* = 6

15-20. Написать программу вычисления периметра и площади треугольника, заданного координатами вершин.
Пример: *x*1 = 0, *y*1 = 0, *x*2 = 0, *y*2 = 3, *x*3 = 4, *y*4 = 0. Ответ: *P*=12, *S*=6

Раздел 16. Формы с переключателями.

16-1. Написать программу, которая вычисляет сопротивление электрической цепи, состоящей из двух сопротивлений. Сопротивления могут быть соединены последовательно или параллельно. Рекомендуемый вид формы дан на *рис. 16-1*.

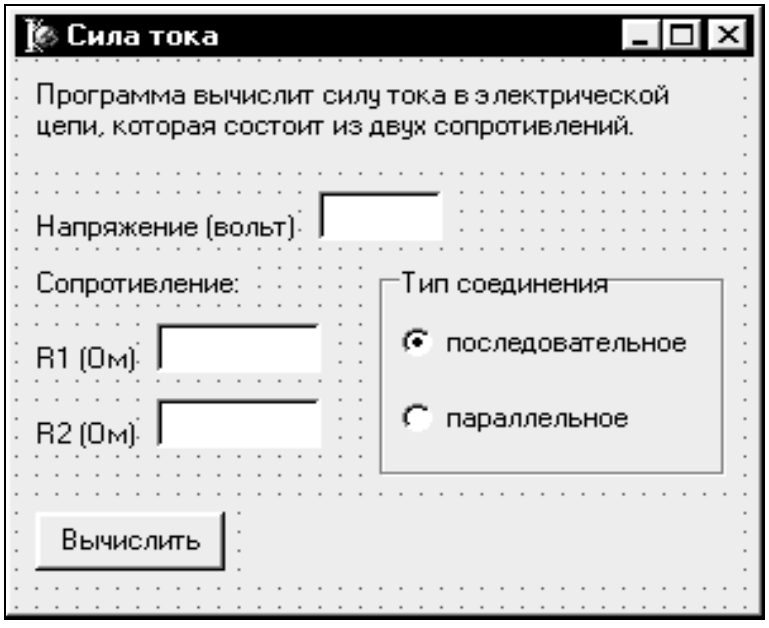


Рис. 16-1

16-2. Написать программу, которая вычисляет силу тока в электрической цепи (используя закон Ома). Цепь состоит из двух сопротивлений. Сопротивления могут быть соединены последовательно или параллельно. Рекомендуемый вид формы приведен на *рис. 16-2*.

Рис. 16-2

16-3. Написать программу, которая находит максимум, либо минимум для двух задаваемых чисел

16-4. Написать программу, которая находит максимум, либо минимум для трех задаваемых чисел

16-5. Написать программу, которая находит среднее арифметическое, либо среднее геометрическое для двух задаваемых чисел

16-6. Написать программу, которая указывает знак значения функции **sin** в зависимости от выбранной на форме четверти координатной плоскости

16-7. Написать программу, которая указывает знак значения функции **cos** в зависимости от выбранной на форме четверти координатной плоскости.

16-8. Написать программу, которая вычисляет стоимость разговора по междугороднему телефону в зависимости от указанных минут длительности и выбранного времени (день или ночь). Ночью – скидка в 30%

16-9. Написать программу, которая вычисляет стоимость разговора по междугороднему телефону в зависимости от указанных минут длительности и выбранного вида дня недели (рабочие или выходные дни). В выходной – скидка в 20%.

16-10. Написать программу, которая в зависимости от выбранного времени года (зима, весна, лето, осень) выдает список месяцев этого времени

года.

16-11. Написать программу, которая в зависимости от выбранного пересчитывает заданную в рублях сумму в эквивалент в долларах, либо в евро.

В последующих девяти задачах одинаковое условие:
Написать программу, которая в зависимости от выбранного способа (по формуле суммы или суммированием в цикле) рассчитывает сумму.

16-12. $1 + 2 + 3 + \ldots + n = n(n + 1)/2$

16-13. $p + (p + 1) + (p + 2) + \ldots + (p + n) = (n + 1)(2p + n)/2$

16-14. $1 + 3 + 5 + \ldots + (2n - 1) = n^2$

16-15. $2 + 4 + 6 + \ldots + 2n = n(n + 1)$

16-16. $1^2 + 2^2 + 3^2 + \ldots + n^2 = n(n + 1)(2n + 1)/6$

16-17. $1^3 + 2^3 + 3^3 + \ldots + n^3 = n^2(n + 1)^2/4$

16-18. $1^2 + 3^2 + 5^2 + \ldots + (2n - 1)^2 = n(4n^2 - 1)/3$

16-19. $1^3 + 3^3 + 5^3 + \ldots + (2n - 1)^3 = n^2(2n^2 - 1)$

16-20. $1^4 + 2^4 + 3^4 + \ldots + n^4 = n(n + 1)(2n + 1)(3n^2 + 3n - 1)/30$

Раздел 17. Формы с прокруткой

В приводимых ниже заданиях организовать вычисление с помощью полосы прокрутки для различных *n*. Причем, предусмотреть вычисление, как в цикле, так и по формуле, приведенной в правой части выражения

17-1. $1 + 2 + 3 + \ldots + n = n(n + 1)/2$

17-2. $p + (p + 1) + (p + 2) + \ldots + (p + n) = (n + 1)(2p + n)/2$

17-3. $1 + 3 + 5 + \ldots + (2n - 1) = n^2$

17-4. $2 + 4 + 6 + \ldots + 2n = n(n + 1)$

17-5. $1^2 + 2^2 + 3^2 + \ldots + n^2 = n(n + 1)(2n + 1)/6$

17-6. $1^3 + 2^3 + 3^3 + \ldots + n^3 = n^2(n + 1)^2/4$

17-7. $1^2 + 3^2 + 5^2 + \ldots + (2n - 1)^2 = n(4n^2 - 1)/3$

17-8. $1^3 + 3^3 + 5^3 + \ldots + (2n - 1)^3 = n^2(2n^2 - 1)$

17-9. $1^4 + 2^4 + 3^4 + \ldots + n^4 = n(n + 1)(2n + 1)(3n^2 + 3n - 1)/30$

17-10. $1 + 4 + 7 + \ldots + 3n - 2 = n(3n - 1)/2$

17-11. $-1 + 2 - 3 + \ldots + (-1)^n n = (-1)^n [(n + 1)/2]$ (здесь [ ] – означает целую часть)

17-12. $-1^2 + 2^2 - 3^2 + \ldots + (-1)^n n^2 = (-1)^n (n(n + 1)/2)$

17-13. $-1^3 + 2^3 - 3^3 + \ldots + (-1)^n n^3 = (1/8)*(1 - (-1)^n (1 - 6n^2 - 4n^3))$

17-14. $-1^4 + 2^4 - 3^4 + \ldots + (-1)^n n^4 = (-1)^n (n^4 + 2n^3 - n)/2$

17-15. $1^5 + 2^5 + 3^5 + \ldots + n^5 = (1/12)*n^2(n + 1)^2(2n^2 + 2n - 1)$

17-16. $1^5 - 2^5 + 3^5 - \ldots + (-1)^{n-1} n^5 = (1/4)(1 + (-1)^n (5n^2 - 5n^4 - 2n^5 - 1))$

17-17. $1 + 3 + 5 + \ldots + (2n + 1) = (n + 1)^2$

17-18. $1 - 3 + 5 - 7 + \ldots + (-1)^n (2n + 1) = (-1)^n (n + 1)$

17-19. $1^2 + 3^2 + 5^2 + \ldots + (2n + 1)^2 = (1/3)(n + 1)(2n + 1)(2n + 3)$

17-20. $1^2 - 3^2 + 5^2 - \ldots + (-1)^n (2n + 1)^2 = (-1)^n 2(n + 1)^2 - (1 + (-1)^n)/2$

17-21. $1^3 + 3^3 + 5^3 + \ldots + (2n + 1)^3 = (n + 1)^2(2n^2 + 4n + 1)$

17-22. $1*2 + 2*3 + \ldots + n(n + 1) = (1/3)n(n + 1)(n + 2)$

17-23. $1*2*3 + 2*3*4 + \ldots + n(n + 1)(n + 2) = (1/4)n(n + 1)(n + 2)(n + 3)$

Раздел 18. Формы с меню и с работой с файлами

В приводимых ниже заданиях следует разработать программу с использованием кнопок на форме (см. пример ниже). Данные рекомендуется вводить с использованием объекта Memo и сохранять их в текстовом файле. Вывод на экран также производить в область Мемо. В качестве примера приведем выполнение следующего задания:

Дана запись с именем STUDENT, содержащая следующие поля:

3) Фамилия и инициалы;
4) Номер группы;
5) Успеваемость (массив из пяти элементов).

Написать программу, которая выполняет следующие действия:

- Ввод с клавиатуры данных из 10 записей типа STUDENT, и занесение их в файл данных;
- Чтение данных из файла и вывод их на экран;
- Вывод на экран фамилий, номеров групп и оценок для всех студентов, которые являются круглыми отличниками (если таких нет – вывести об этом сообщение);
- Список должен быть упорядочен по возрастанию номера группы.

Форма должна иметь примерно такой вид (*рис. 18-1*):

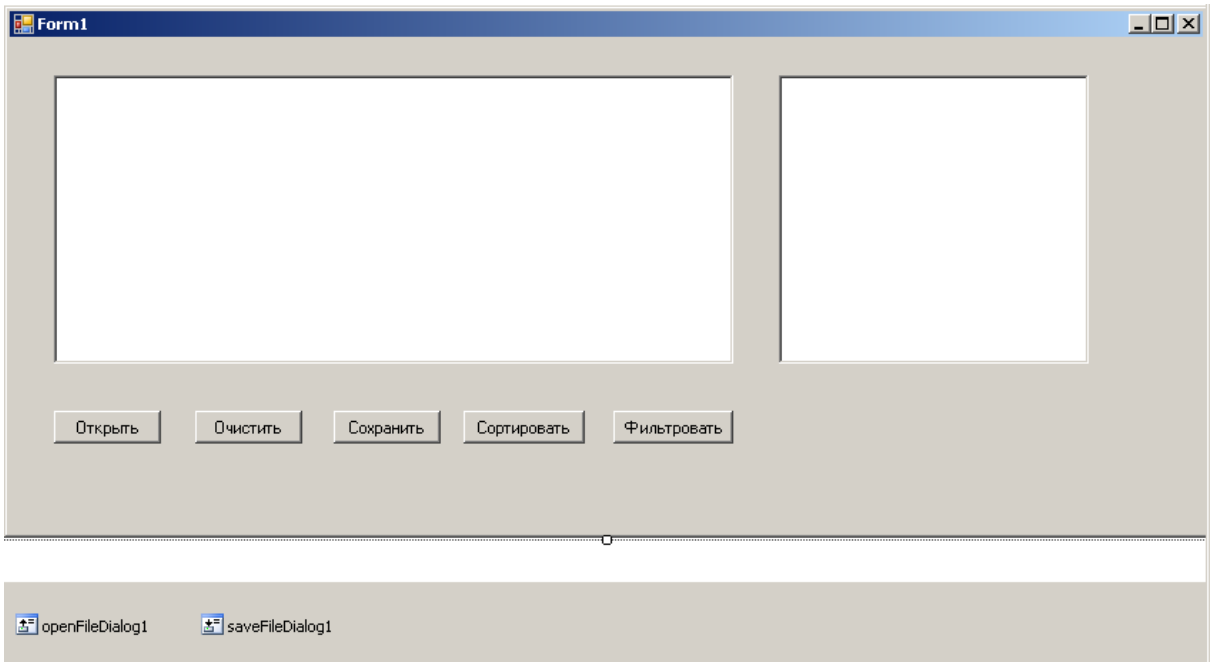


Рис. 18-1

Обратите внимание на два объекта – **openFileDialog** и **saveFileDialog** . Они нам потребуются для организации диалога при вводе данных в файл (и последующего сохранения), а также при считывании из файла.

При первоначальном запуске данные на форме вносятся в поле **richTextBox** (в соответствии с той структурой, которая указана в задании), причем в пределах строки – данные одного студента: фамилия студента, номер группы, каждая из оценок вносятся через пробел. Занесем, например, данные, приведенные на *рис. 18-2 (окно слева)*.

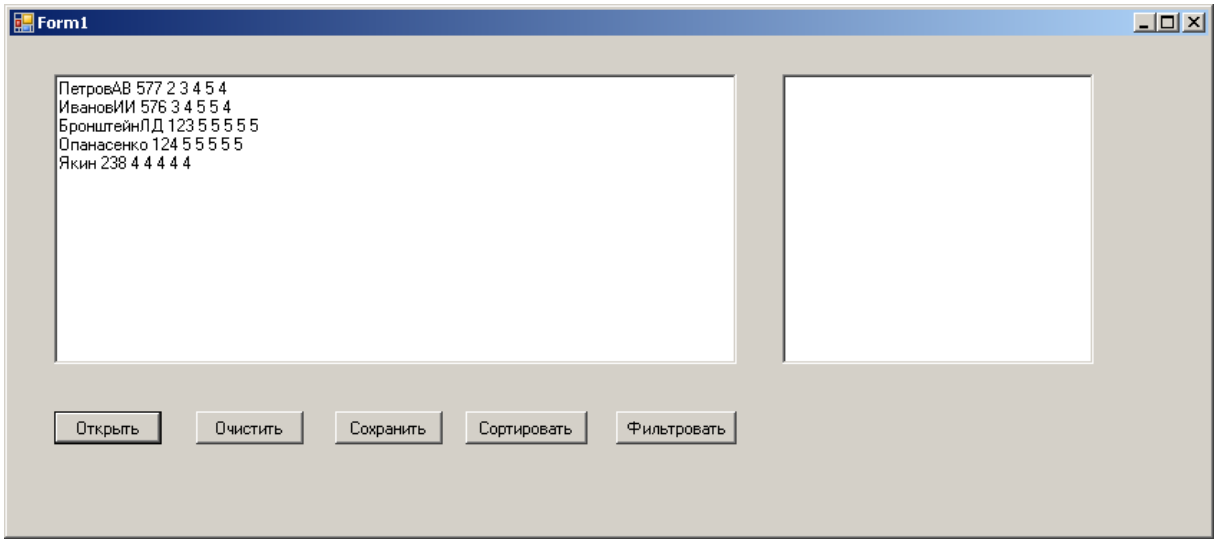


Рис. 18-2

Кнопка «**Сохранить**» служит для сохранения данных из окна richTextBox. Функция для кнопки «Сохранить» имеет вид:

```
private:  System::Void  button3_Click(System::Object^
sender, System::EventArgs^  e) {
saveFileDialog1->ShowDialog();
richTextBox1->SaveFile(saveFileDialog1->FileName);
         }
```

При этом откроется окно диалога для сохранения (*рис. 18-3*):

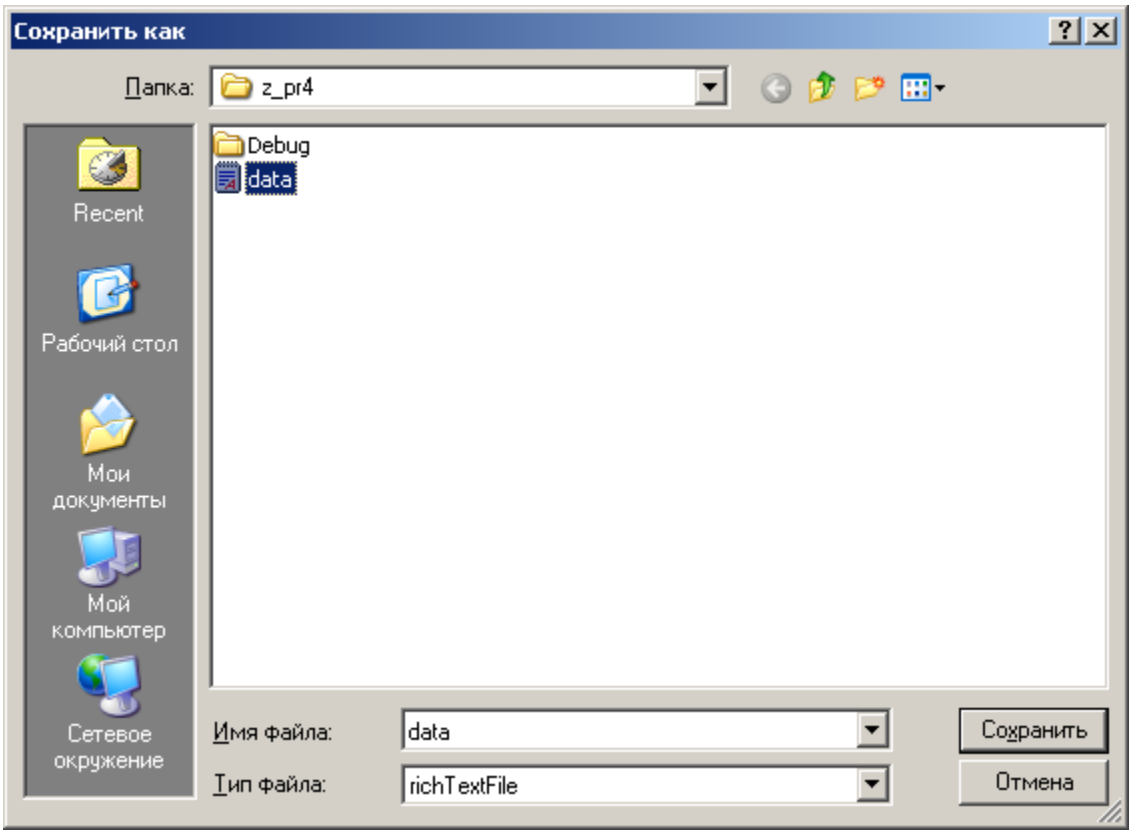


Рис. 18-3

В этом окне нужно будет перейти на нужный диск и в нужную папку, а затем указать имя файла. Рекомендуется указывать у файла расширение rtf.

Затем рекомендуется поле Memo очистить с помощью кнопки «**Очистить**», функция для которой имеет вид:

```
private:   System::Void  button2_Click(System::Object^
sender, System::EventArgs^  e) {
             richTextBox1->Clear();
         }
```

Далее следует файл открыть – кнопкой «**Открыть**», имеющей функцию:

```
private:   System::Void  button1_Click(System::Object^
sender, System::EventArgs^  e) {
    openFileDialog1->ShowDialog();
    richTextBox1->LoadFile(openFileDialog1-
>FileName);
             }
```

При выполнении этой функции появится аналогичное окно диалога (*рис. 18-4*). И здесь нужно будет перейти в соответствующий каталог (папку) и выбрать нужный для открытия файл. Обратите внимание, что мы здесь заранее ограничили список файлами текстового типа (**rtf**). Такой фильтр обеспечивается выполнением (при запуске приложения, т.е. при открытии формы) функции вида:

```
private:    System::Void    Form1_Load(System::Object^
sender, System::EventArgs^  e) {
// фильтр списка файлов
openFileDialog1->Filter="RTF|*.rtf";
saveFileDialog1->Filter="RTF|*.rtf";
}
```

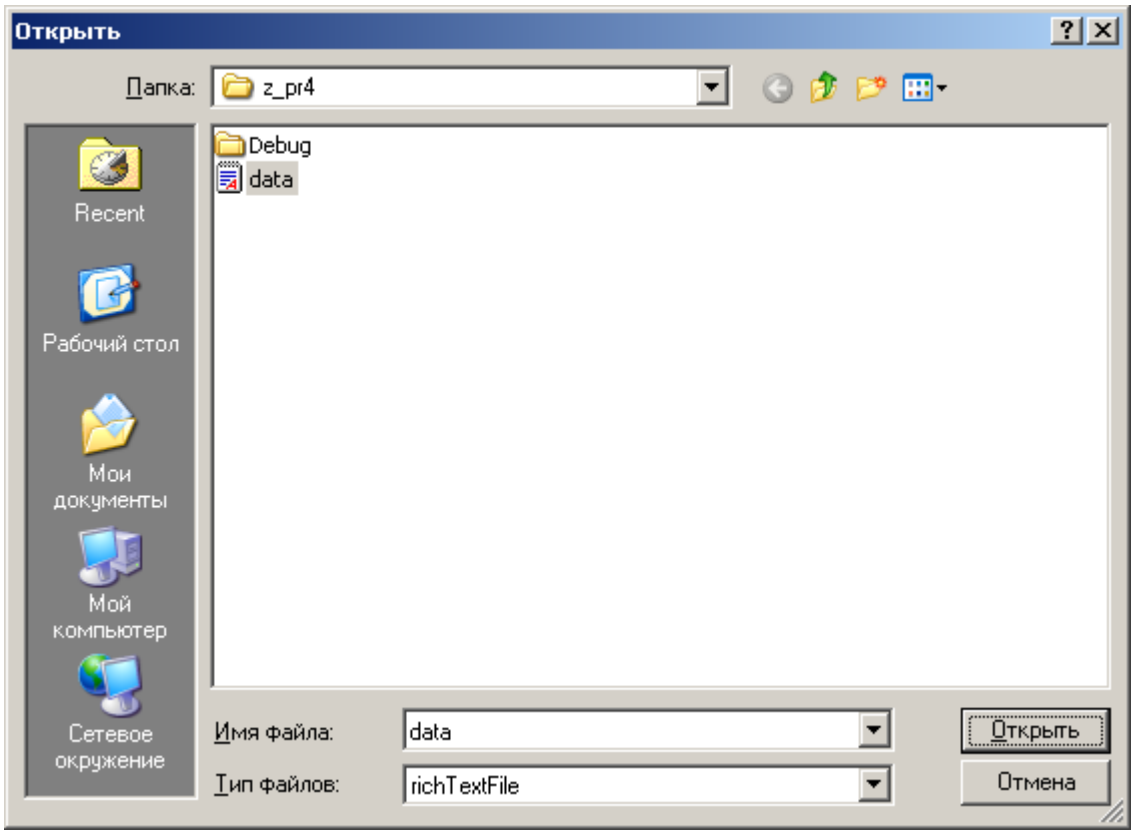


Рис. 18-4

Теперь данный список (из открытого файла) вновь появится в поле richTextBox и его можно обрабатывать. Например, для сортировки (упорядочения по номеру группы) следует нажать кнопку «**Сортировать**», содержащую функцию:

```
private: System::Void button4_Click(System::Object^  sender,
System::EventArgs^  e) {
int gr[20]; int i,j; String^ s;
int n=richTextBox1->Lines->Length;  //n- число строк
 for ( i=0;i<n;i++)
{int ns=richTextBox1->Lines[i]->Length;//ns- длина i-й строки
     s =richTextBox1->Lines[i]; //вычленение i-й строки
     int k0=s->IndexOf(" "); //поиск положения 1-го пробела
     int k1=s->IndexOf(" ",k0+1);  //положение 2-го пробела
String^  sgr=s->Substring(k0+1,k1-k0);//выделение  подстроки
                    //между 1-м и 2-м пробелами – номер группы
gr[i]=Int32::Parse(sgr); //номер группы (в числовом виде)
 }
 //собственно сортировка поиском максимума
 // и запоминанием порядка максимумов в массиве  m
```

```
int k=0; int nmax, max, m[20];
for ( i=0;i<n;i++)
{  max=0;
     for (j=0;j<n;j++)
{if (gr[j]>max) {max=gr[j];nmax=j;}}
  m[k]=nmax;k=k+1;
gr[nmax]=0;}
 richTextBox2->Clear();
 // вывод упорядоченных строк в новое окно richTextBox2
for ( j=0;j<n;j++)
{richTextBox2->Text=
    richTextBox2->Text+"\n"+richTextBox1->Lines[m[j]];
  }            }
```

Внимательно изучите данную функцию. Обратите внимание, например, как вычисляется длина каждой строки.

Для преобразования номера группы (который в строке записан *символами*) в целое число используется функция `Int32::Parse`.

При нажатии на данную кнопку получим результат в отдельном окне richTextBox (см. *рис. 18-5, окошко справа*). Как видно, список упорядочен по номеру группы в порядке убывания.

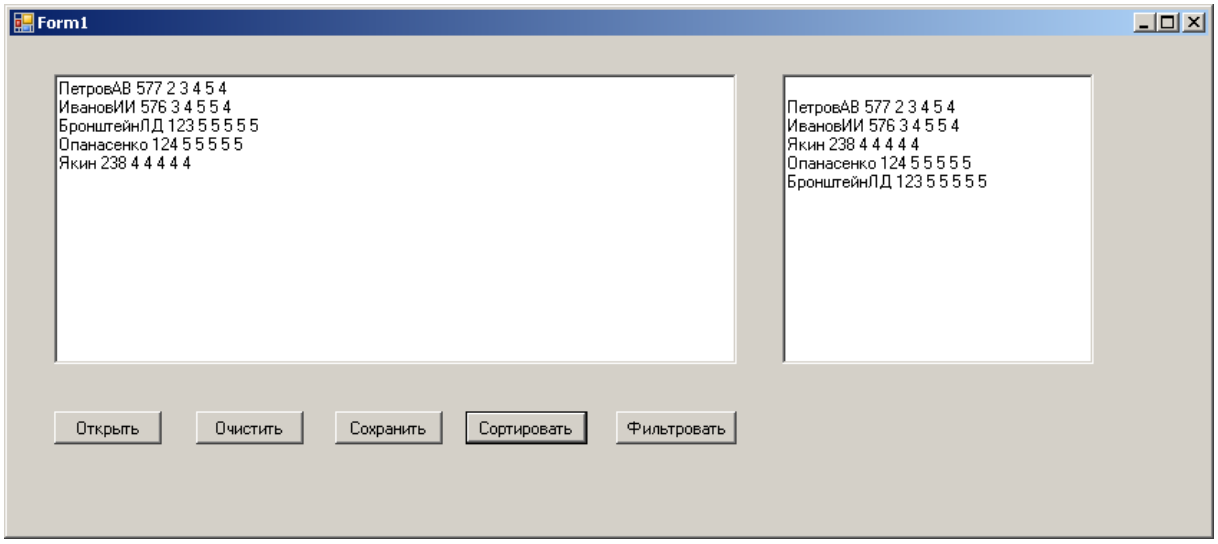


Рис. 18-5

Далее потребуется отфильтровать список (в данном случае выделить отличников). Для этого служит кнопка «**Отфильтровать**» с функцией:

```
private: System::Void button5_Click(System::Object^  sender,
System::EventArgs^  e) {
String^ s;      String^ soc; int i,j; int oc[20][5];
 int  n=richTextBox1->Lines->Length;  //n-число  строк  в  окне
RichTextBox
```

```
 for ( i=0;i<n;i++)
 {int ns=richTextBox1->Lines[i]->Length;//ns-длина i-й строки
              s =richTextBox1->Lines[i];
               s=s+" ";//добавили пробел в конец строки
               int k0=s->IndexOf(" ");
               int k1=s->IndexOf(" ",k0+1);  ;
               int k2=s->IndexOf(" ",k1+1);  ;
               soc=s->Substring(k1+1,k2-k1);
              oc[i][1]=Int32::Parse(soc);
               int k3=s->IndexOf(" ",k2+1);  ;
               soc=s->Substring(k2+1,k3-k2);
               oc[i][2]=Int32::Parse(soc);
               int k4=s->IndexOf(" ",k3+1);  ;
               soc=s->Substring(k3+1,k4-k3);
               oc[i][3]=Int32::Parse(soc);
               int k5=s->IndexOf(" ",k4+1);  ;
               soc=s->Substring(k4+1,k5-k4);
               oc[i][4]=Int32::Parse(soc);
               int k6=s->IndexOf(" ",k5+1);  ;
               soc=s->Substring(k5+1,k6-k5);
               oc[i][5]=Int32::Parse(soc);
                 }
 richTextBox2->Clear();
 //отбор отличников
for ( i=0;i<n;i++)
{    int sum=0;
     for (j=1;j<=5;j++)
     {    if ((oc[i][j])==5) sum++;}
     if (sum==5)
          richTextBox2->Text=
           richTextBox2->Text+"\n"+richTextBox1->Lines[i];
 }
           }
```

Здесь сначала вычленяются оценки каждого студента и подсчитывается их количество. Поскольку заранее известно, что всего оценок пять штук, то круглый отличник должен иметь пять «пятерок». В результате соответствующую строку и выводим в поле richTextBox2 . Нажмем на эту кнопку «**Отфильтровать**» и получим отдельно список отличников (см. *рис. 18-6, правое окошко*)

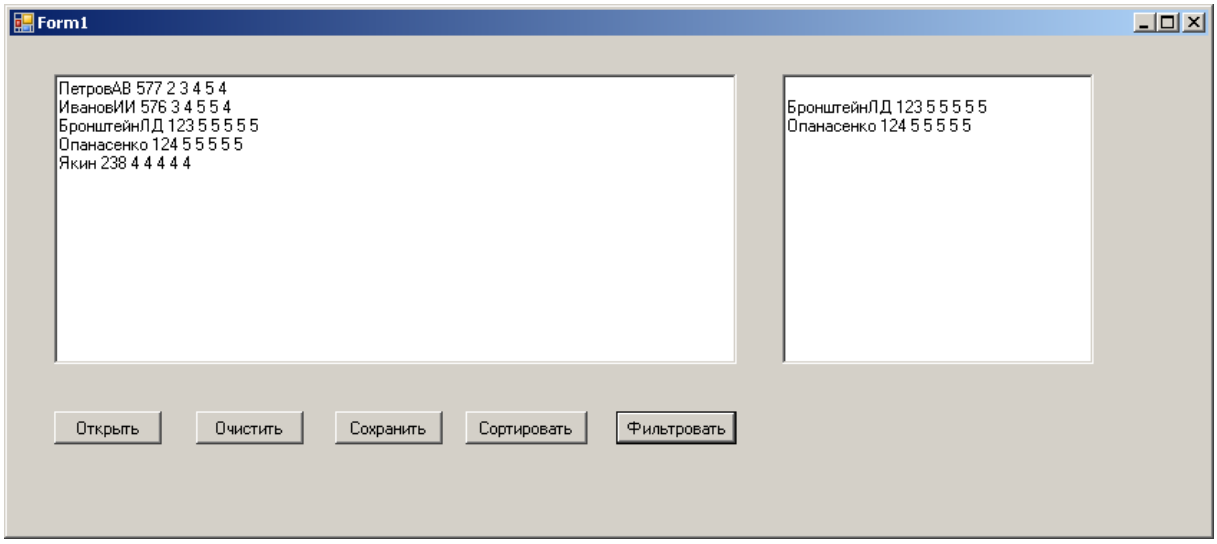


Рис. 18-6

Можно, кстати, такой отфильтрованный список записать в новый файл. Для этого надо воспользоваться кнопкой «**Сохранить**» (но обратите внимание, что мы выводим отличников в новом окне richTextBox2, поэтому нужно будет поправить функцию для кнопки “Сохранить”). При нажатии на нее потребуется указать имя (другое) файла и перейти, если нужно, в другую папку. Потом уже отфильтрованный список можно открыть и провести для него сортировку (т.е. упорядочить отдельно отличников).

18-1. Дана запись с именем STUDENT, содержащая следующие поля:

– Фамилия и инициалы;
– Номер группы;
– Успеваемость (массив из пяти элементов).

Написать программу, которая выполняет следующие действия:

- Ввод с клавиатуры данных из 10 записей типа STUDENT, и занесение их в файл данных;
- Чтение данных из файла и вывод их на экран;
- Вывод на экран фамилий и номеров групп для всех студентов, если средний балл студента больше 4 (если таких нет – вывести об этом сообщение);
- Список должен быть упорядочен по возрастанию номера группы.

18-2. Дана запись с именем STUDENT, содержащая следующие поля:

– Фамилия и инициалы;
– Номер группы;
– Успеваемость (массив из пяти элементов).

Написать программу, которая выполняет следующие действия:

- Ввод с клавиатуры данных из 10 записей типа STUDENT, и занесение их в файл данных;
- Чтение данных из файла и вывод их на экран;

- Вывод на экран фамилий и номеров групп для всех студентов, имеющих оценки 4 и 5 (если таких нет – вывести об этом сообщение);
- Список должен быть упорядочен по величине среднего балла.

18-3. Дана запись с именем STUDENT, содержащая следующие поля:

– Фамилия и инициалы;
– Номер группы;
– Успеваемость (массив из пяти элементов);

Написать программу, которая выполняет следующие действия:

- Ввод с клавиатуры данных из 10 записей типа STUDENT, и занесение их в файл данных;
- Чтение данных из файла и вывод их на экран;
- Вывод на экран фамилий и номеров групп для всех студентов, имеющих хотя бы одну оценку 2 (если таких нет – вывести об этом сообщение);

6) Список студентов должен быть упорядочен по алфавиту фамилий.

18-4. Дана запись с именем AEROFLOT, содержащая следующие поля:

– Название пункта назначения рейса;
– Номер рейса;
– Тип самолета.

Написать программу, которая выполняет следующие действия:

- Ввод с клавиатуры данных из 7 элементов типа AEROFLOT, и занесение их в файл данных;
- Чтение данных из файла и вывод их на экран;
- Вывод на экран номеров рейсов и типов самолетов, вылетающих в пункт назначения, название которого совпало с названием, введенным с клавиатуры (если таких нет – вывести об этом сообщение);
- Список должен быть упорядочен по возрастанию номера рейса.

18-5. Дана запись с именем AEROFLOT, содержащая следующие поля:

– Название пункта назначения рейса,
– Номер рейса,
– Тип самолета

Написать программу, которая выполняет следующие действия:

- Ввод с клавиатуры из 7 элементов типа AEROFLOT, и занесение их в файл данных
- Чтение данных из файла и вывод их на экран
- вывод на экран пунктов назначения и номеров рейсов, обслуживаемых самолетом, тип которого введен с клавиатуры (если таких нет – вывести об этом сообщение)
- Список должен быть упорядочен по алфавиту названий пунктов назначения.

18-6. Дана запись с именем WORKER, содержащая следующие поля:

– Фамилия и инициалы работника;
– Название занимаемой должности;
– Год поступления на работу.

Написать программу, которая выполняет следующие действия:

- Ввод с клавиатуры данных из 10 элементов типа WORKER, и занесение их в файл данных;
- Чтение данных из файла и вывод их на экран;
- вывод на экран фамилий работников, чей стаж работы в организации превышает значение, введенное с клавиатуры (если таких нет – вывести об этом сообщение);
- Список должен быть упорядочен по алфавиту фамилий.

18-7. Дана запись с именем TRAIN, содержащая следующие поля:

– Название пункта назначения;
– Номер поезда;
– Время отправления.

Написать программу, которая выполняет следующие действия:

- Ввод с клавиатуры данных из 8 элементов типа TRAIN, и занесение их в файл данных;
- Чтение данных из файла и вывод их на экран;
- вывод на экран информации о поездах, отправляющихся после введенного с клавиатуры времени (если таких нет – вывести об этом сообщение);
- Список должен быть упорядочен по алфавиту пунктов назначения.

18-8. Дана запись с именем TRAIN, содержащая следующие поля:

– Название пункта назначения;
– Номер поезда;
– Время отправления.

Написать программу, которая выполняет следующие действия:

- Ввод с клавиатуры данных из 6 элементов типа TRAIN, и занесение их в файл данных;
- Чтение данных из файла и вывод их на экран;
- вывод на экран информации о поездах, отправляющихся в пункт, название которого введено с клавиатуры (если таких нет – вывести об этом сообщение);
- Список должен быть упорядочен по времени отправления поезда.

18-9. Дана запись с именем TRAIN, содержащая следующие поля:

– Название пункта назначения;
– Номер поезда;
– Время отправления.

Написать программу, которая выполняет следующие действия:

- Ввод с клавиатуры данных из 8 элементов типа TRAIN, и занесение их в файл данных;
- Чтение данных из файла и вывод их на экран;
- вывод на экран информации о поезде, номер которого введен с клавиатуры (если таких нет – вывести об этом сообщение);
- Список должен быть упорядочен по номерам поездов.

18-10. Дана запись с именем MARSH, содержащая следующие поля:
– Название начального пункта назначения;
– Название конечного пункта назначения;
– Номер маршрута.
Написать программу, которая выполняет следующие действия:
- Ввод с клавиатуры данных из 8 элементов типа MARSH, и занесение их в файл данных;
- Чтение данных из файла и вывод их на экран;
- вывод на экран информации о маршруте, номер которого введен с клавиатуры (если таких нет – вывести об этом сообщение);
- Список должен быть упорядочен по номерам маршрутов.

18-11. Дана запись с именем MARSH, содержащая следующие поля:
– Название начального пункта назначения;
– Название конечного пункта назначения;
– Номер маршрута.
Написать программу, которая выполняет следующие действия:
- Ввод с клавиатуры данных из 8 элементов типа MARSH, и занесение их в файл данных;
- Чтение данных из файла и вывод их на экран;
- вывод на экран информации о маршрутах, которые начинаются или заканчиваются в пункте, название которого введено с клавиатуры (если таких нет – вывести об этом сообщение);
- Список должен быть упорядочен по номерам маршрутов.

18-12. Дана запись с именем NOTE, содержащая следующие поля:
– Фамилия, имя;
– Номер телефона;
– Дата рождения (массив из трех чисел).
Написать программу, которая выполняет следующие действия:
- Ввод с клавиатуры данных из 8 элементов типа NOTE, и занесение их в файл данных;
- Чтение данных из файла и вывод их на экран;
- вывод на экран информации о человеке, номер телефона которого введен с клавиатуры (если таких нет – вывести об этом сообщение);
- Список должен быть упорядочен по датам рождения.

18-13. Дана запись с именем NOTE, содержащая следующие поля:
– Фамилия, имя;
– Номер телефона;
– Дата рождения (массив из трех чисел).
Написать программу, которая выполняет следующие действия:
- Ввод с клавиатуры данных из 8 элементов типа NOTE, и занесение их в файл данных;
- Чтение данных из файла и вывод их на экран;
- вывод на экран информации о людях, чьи дни рождения приходятся на месяц, значение которого введено с клавиатуры (если таких нет – вывести об этом сообщение);
- Список должен быть упорядочен по алфавиту.

18-14. Дана запись с именем NOTE, содержащая следующие поля:
– Фамилия, имя;
– Номер телефона;
– Дата рождения (массив из трех чисел).
Написать программу, которая выполняет следующие действия:
- Ввод с клавиатуры данных из 8 элементов типа NOTE, и занесение их в файл данных;
- Чтение данных из файла и вывод их на экран;
- вывод на экран информации о человеке, чья фамилия введена с клавиатуры (если таких нет – вывести об этом сообщение);
- Список должен быть упорядочен по двум первым цифрам номера телефона.

18-15. Дана запись с именем ZNAK, содержащая следующие поля:
– Фамилия, имя;
– Знак Зодиака;
– Дата рождения (массив из трех чисел).
Написать программу, которая выполняет следующие действия:
- Ввод с клавиатуры данных из 8 элементов типа ZNAK, и занесение их в файл данных;
- Чтение данных из файла и вывод их на экран;
- вывод на экран информации о человеке, чья фамилия введена с клавиатуры (если таких нет – вывести об этом сообщение).
- Список должен быть упорядочен по датам рождения.

18-16. Дана запись с именем ZNAK, содержащая следующие поля:
– Фамилия, имя;
– Знак Зодиака;
– Дата рождения (массив из трех чисел).
Написать программу, которая выполняет следующие действия:

- Ввод с клавиатуры данных из 8 элементов типа ZNAK, и занесение их в файл данных;
- Чтение данных из файла и вывод их на экран;
- вывод на экран информации о людях, родившихся под знаком, название которого введено с клавиатуры (если таких нет – вывести об этом сообщение);
- Список должен быть упорядочен по датам рождения.

18-17. Дана запись с именем ZNAK, содержащая следующие поля:
– Фамилия, имя;
– Знак Зодиака;
– Дата рождения (массив из трех чисел).
Написать программу, которая выполняет следующие действия:
- Ввод с клавиатуры данных из 8 элементов типа ZNAK, и занесение их в файл данных;
- Чтение данных из файла и вывод их на экран;
- вывод на экран информации о людях, родившихся в месяц, значение которого введено с клавиатуры (если таких нет – вывести об этом сообщение);
- Список должен быть упорядочен по знакам Зодиака.

18-18. Дана запись с именем PRICE, содержащая следующие поля:
– Название товара;
– Название магазина, в котором продается товар;
– Стоимость товара в руб.
Написать программу, которая выполняет следующие действия:
- Ввод с клавиатуры данных из 8 элементов типа PRICE, и занесение их в файл данных;
- Чтение данных из файла и вывод их на экран;
- вывод на экран информации о товаре, название которого введено с клавиатуры (если таких нет – вывести об этом сообщение);
- Список должен быть упорядочен по алфавиту названий товара.

18-19. Дана запись с именем PRICE, содержащая следующие поля:
– Название товара;
– Название магазина, в котором продается товар;
– Стоимость товара в руб.
Написать программу, которая выполняет следующие действия:
- Ввод с клавиатуры данных из 8 элементов типа PRICE, и занесение их в файл данных;
- Чтение данных из файла и вывод их на экран;
- вывод на экран информации о товарах, продающихся в магазине, название которого введено с клавиатуры (если таких нет – вывести об этом сообщение);

- Список должен быть упорядочен по алфавиту названий магазинов.

18-20. Дана запись с именем ORDER, содержащая следующие поля:
– Расчетный счет плательщика;
– Расчетный счет получателя;
– Перечисляемая сумма в руб.
Написать программу, которая выполняет следующие действия:
- Ввод с клавиатуры данных из 8 элементов типа ORDER, и занесение их в файл данных;
- Чтение данных из файла и вывод их на экран;
- вывод на экран информации о сумме, снятой с расчетного счета плательщика, введенного с клавиатуры (если таких нет – вывести об этом сообщение).
- Список должен быть упорядочен по расчетным счетам плательщиков.

КУРСОВЫЕ РАБОТЫ

Выполнение курсовой работы по Информатике (третий семестр) требует решения одной из перечисленных ниже задач и, как результат, создания программы на языке С++ и написания пояснительной записки к работе.

ВАРИАНТЫ ТЕМ КУРСОВЫХ РАБОТ

1. Шифр Цезаря.

Чтобы зашифровать текст, записанный с помощью русских букв и знаков препинания, его можно переписать, заменив каждую букву непосредственно следующей за ней буквой по алфавиту (буква Я заменяется на А). Обобщив этот способ шифровки, можно производить сдвиг не на одну букву, а на N букв (N – натуральное число)

Создать программу, которая

А) зашифрует введенный текст и сохранит его в файл,

Б) считает зашифрованный текст из файла и расшифрует данный текст.

2. Шифровка последовательностей нулей и единиц

Способ шифровки последовательностей нулей и единиц (или, например, точек и тире) состоит в следующем. Пусть $a_1, a_2, \ldots a_N$ – такая последовательность. То, что предлагается в качестве ее шифра - это последовательность $b_1, \ldots b_N$, образованная по следующему правилу:

$b_1=a_1$, $b_i = 1$ при $a_i=a_{i-1}$ либо $b_i = 0$ иначе (для i =2, … N)

Пользуясь изложенным способом, создать программу, которая:

А) зашифрует введенный текст и сохранит его в файл,

Б) считает зашифрованный текст из файла и расшифрует данный текст.

3. «Табличная шифровка».

Один из простейших способов шифровки текста состоит в табличной замене каждого символа другим символом – его шифром. Выбрать некоторую таблицу, разработать способ ее представления, затем, пользуясь изложенным способом, создать программу, которая:

А) зашифрует введенный текст и сохранит его в файл,

Б) считает зашифрованный текст из файла и расшифрует данный текст.

4. «Матричная шифровка»

Чтобы зашифровать текст из 121 буквы, его можно записать в квадратную матрицу порядка 11 по строкам, а затем прочитать по спирали, начиная с центра (т.е. с элемента, имеющего индексы 6,6).

Такой способ можно обобщить и для произвольной длины текста, подбирая нужный размер матрицы.

Пользуясь изложенным способом, создать программу, которая:
А) зашифрует введенный текст и сохранит его в файл,
Б) считает зашифрованный текст из файла и расшифрует данный текст.

5. «Шифровка решеткой»

Шифровка текста с помощью решетки заключается в следующем. Решетка, т.е. квадрат из клетчатой бумаги 10х10 клеток, некоторые клетки в котором вырезаны, совмещается с целым квадратом 10х10 клеток и через прорези на бумагу наносятся первые буквы текста. Затем решетка поворачивается на 90 градусов и через прорези записываются следующие буквы. Это повторяется еще дважды. Таким образом, на бумагу наносится 100 букв текста. Решетку можно изобразить квадратной матрицей порядка 10 из нулей и единиц (нуль изображает прорезь). Доказано что матрица [Aij], i=1,…10, j=1,…10 может служить ключом шифра, если из элементов Aij, $A_{10-i+1\ j}$, $A_{i\ 10-j+1}$, $A_{10-i+1\ 10-j+1}$ в точности один равен нулю.

Дана последовательность из 100 букв и матрица-ключ.
Пользуясь изложенным способом создать программу, которая:
А) зашифрует введенный текст и сохранит его в файл,
Б) считает зашифрованный текст из файла и расшифрует данный текст.
Обобщить на случай последовательности произвольной длины.

6. «Шифровка зафиксированной перестановкой»

Зафиксируем натуральное k и перестановку чисел 1,… k (ее можно задать с помощью последовательности натуральных чисел p1,… pk, в которую входит каждое из чисел 1,… k). При шифровке в исходном тексте к каждой из последовательных групп по k символов применяется зафиксированная перестановка. Пусть k=4 и перестановка есть 3, 2, 4, 1. Тогда группа символов s1,s2,s3,s4 заменяется на s3, s2, s4, s1. Если в последней группе меньше четырех символов, то к ней добавляются пробелы.

Пользуясь изложенным способом создать программу, которая:
А) зашифрует введенный текст и сохранит его в файл,
Б) считает зашифрованный текст из файла и расшифрует данный текст.

7. Шифр Гронсфельда

Этот шифр сложной замены, называемый шифром Гронсфельда, представляет собой модификацию шифра Цезаря числовым ключом. Для этого под буквами исходного сообщения записывают цифры числового ключа. Если ключ короче сообщения, то его запись циклически повторяют. Шифртекст получают примерно, как в шифре Цезаря, но отсчитывают по алфавиту не третью букву (как это делается в шифре Цезаря), а выбирают ту букву, которая смещена по алфавиту на соответствующую цифру ключа. Например,

применяя в качестве ключа группу из четырех начальных цифр числа *e* (основания натуральных логарифмов), а именно 2718, получаем для исходного сообщения ВОСТОЧНЫЙ ЭКСПРЕСС следующий шифртекст:

Сообщение	В	О	С	Т	О	Ч	Н	Ы	Й		Э	К	С	П	Р	Е	С	С
Ключ	2	7	1	8	2	7	1	8	2		7	1	8	2	7	1	8	2
Шифртекст	Д	Х	Т	Ь	Р	Ю	О	Г	Л		Д	Л	Ш	С	Ч	Ж	Ш	У

Чтобы зашифровать первую букву сообщения В, используя первую цифру ключа 2, нужно отсчитать вторую по порядку букву от В в алфавите

В	Г	Д
	1	2

получается первая буква шифртекста Д.

Пользуясь изложенным способом создать программу, которая:
А) зашифрует введенный текст и сохранит его в файл,
Б) считает зашифрованный текст из файла и расшифрует данный текст.

8. Шифровка с помощью квадрата Полибия

В Древней Греции (II в. до н.э.) был известен шифр, называемый "квадрат Полибия". Шифровальная таблица представляла собой квадрат с пятью столбцами и пятью строками, которые нумеровались цифрами от 1 до 5. В каждую клетку такого квадрата записывалась одна буква. В результате каждой букве соответствовала пара чисел, и шифрование сводилось к замене буквы парой чисел. Для латинского алфавита квадрат Полибия имеет вид:

	1	2	3	4	5
1	A	B	C	D	E
2	F	G	H	I, J	K
3	L	M	N	O	P
4	Q	R	S	T	U
5	V	W	X	Y	Z

Пользуясь изложенным способом создать программу, которая:
А) зашифрует введенный текст и сохранит его в файл,
Б) считает зашифрованный текст из файла и расшифрует данный текст.

9. Шифр Хилла (с длиной блока = 2)

Криптосистема, основанная Хиллом, базируется на линейной алгебре. Пространства исходных сообщений и криптотекстов совпадают: *латинский алфавит*. Перенумеруем буквы в порядке их следования в алфавите: **A** получает номер 0, **B** - номер 1, ... и **Z** - номер 25. Все арифметические операции выполняются по модулю 26 (длина алфавита), то есть 26 отождествляется с 0, 27 - с единицей и т.д.

Выберем целое число **D** <= 2. Оно указывает размерность используемых матриц. В процедуре шифрования наборы из **D** букв шифруются вместе. Возьмем **D** = 2. Пусть ключ **M** - квадратная матрица порядка **D**, элементами которой являются числа 0 .. 25. Эта матрица должна удовлетворять требованию невырожденности, т.е. для нее должна существовать матрица M^{-1}, например:

$$M = \begin{vmatrix} 3 & 3 \\ 2 & 5 \end{vmatrix}, \text{ и } M^{-1} = \begin{vmatrix} 15 & 17 \\ 20 & 9 \end{vmatrix}$$

(вся арифметика ведется по модулю 26)

Шифрование осуществляется с помощью уравнения MP = C, где **P** и **C** - вектор столбцы длиной **D**. То есть, каждый набор из **D** букв исходного сообщения определяет вектор **P**, компонентами которого являются номера букв. В свою очередь, полученный вектор **C** также интерпретируется как набор из **D** букв.

Например: исходное сообщение: *HELP* определяет 2 вектора (по 2 буквы в каждом):

$$P_1 = \begin{vmatrix} H \\ E \end{vmatrix} = \begin{vmatrix} 7 \\ 4 \end{vmatrix} \text{ и } P_2 = \begin{vmatrix} L \\ P \end{vmatrix} = \begin{vmatrix} 11 \\ 15 \end{vmatrix}$$

Из уравнений

$$M*P_1 = \begin{vmatrix} 7 \\ 8 \end{vmatrix} = C1 \text{ и } M*P_2 = \begin{vmatrix} 0 \\ 19 \end{vmatrix} = C2$$

получаем зашифрованный текст *HIAT*...

Для дешифровки сообщения используем матрицу M^{-1} [mod 26] и для шифротекста **C** вычисляем $P = M^{-1} * C$ [mod 26]

Пользуясь изложенным способом создать программу, которая:

А) зашифрует введенный текст и сохранит его в файл,

Б) считает зашифрованный текст из файла и расшифрует данный текст.

10. Шифр Атбаш

Шифр простой замены, использованный для еврейского алфавита и получивший оттуда свое название. Шифрование происходит заменой первой буквы алфавита на последнюю, второй на предпоследнюю (*алеф* (первая буква) заменяется на *тав* (последнюю), *бет* (вторая) заменяется на *шин* (предпоследняя); из этих сочетаний шифр и получил свое название). Шифр Атбаш для английского алфавита:

Исходный алфавит: A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

Алфавит замены: Z Y X W V U T S R Q P O N M L K J I H G F E D C B A

Пользуясь изложенным способом создать программу, которая:
А) зашифрует введенный текст и сохранит его в файл,
Б) считает зашифрованный текст из файла и расшифрует данный текст.

11. Шифр Вижинера (для латинских букв)

Квадрат Виженера или таблица Виженера, так же известная как tabula recta, может быть использована для зашировання и расшифрования.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	Z	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	X	A	B	S
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	S	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	V	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	S	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	S	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	G	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

В шифре Цезаря каждая буква алфавита сдвигается на несколько позиций; например в шифре Цезаря при сдвиге +3, A стало бы D, B стало бы E и так далее. Шифр Виженера состоит из последовательности нескольких шифров Цезаря с различными значениями сдвига. Для зашифрования может использоваться таблица алфавитов, называемая tabula recta или квадрат (таблица) Виженера. Применительно к латинскому алфавиту таблица Виженера составляется из строк по 26 символов, причём каждая следующая строка сдвигается на несколько позиций. Таким образом, в таблице получается 26 различных шифров Цезаря. На разных этапах кодировки шифр Виженера использует различные алфавиты из этой таблицы. На каждом этапе шифрования используются различные алфавиты, выбираемые в зависимости от символа ключевого слова. Например, предположим, что исходный текст имеет вид:

ATTACKATDAWN

Человек, посылающий сообщение, записывает ключевое слово(«LEMON») циклически до тех пор, пока его длина не будет соответствовать длине исходного текста:

LEMONLEMONLE

Первый символ исходного текста A зашифрован последовательностью L, которая является первым символом ключа. Первый символ L шифрованного текста находится на пересечении строки L и столбца A в таблице Виженера. Точно так же для второго символа исходного текста используется второй символ ключа; то есть второй символ шифрованного текста X получается на пересечении строки E и столбца T. Остальная часть исходного текста шифруется подобным способом.

Исходный текст:	ATTACKATDAWN
Ключ:	LEMONLEMONLE
Зашифрованный текст:	LXFOPVEFRNHR

Расшифрование производится следующим образом: находим в таблице Виженера строку, соответствующую первому символу ключевого слова; в данной строке находим первый символ зашифрованного текста. Столбец, в котором находится данный символ, соответствует первому символу исходного текста. Следующие символы зашифрованного текста расшифровываются подобным образом. Из наблюдения за частотой совпадения следует:

$$C_i \equiv P_i + K_i \pmod{26}$$

$$P_i \equiv C_i - K_i \pmod{26}$$

Пользуясь изложенным способом создать программу, которая:
А) зашифрует введенный текст и сохранит его в файл,
Б) считает зашифрованный текст из файла и расшифрует данный текст.

12. Шифр Вижинера (для русских букв)

Шифр Вижинера (см. тему 11) для русского алфавита использует таблицу:

Ключ	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ы	ъ	э
0	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ы	ъ	э
1	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ы	ъ	э	ю
2	в	г	д	е	ж	з	и	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ы	ъ	э	ю	я
3	г	д	е	ж	з	и	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ы	ъ	э	ю	я	а
4	д	е	ж	з	и	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ы	ъ	э	ю	я	а	б
5	е	ж	з	и	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ы	ъ	э	ю	я	а	б	в
6	ж	з	и	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ы	ъ	э	ю	я	а	б	в	г
7	з	и	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ы	ъ	э	ю	я	а	б	в	г	д
8	и	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ы	ъ	э	ю	я	а	б	в	г	д	е
9	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ы	ъ	э	ю	я	а	б	в	г	д	е	ж
10	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ы	ъ	э	ю	я	а	б	в	г	д	е	ж	з
11	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ы	ъ	э	ю	я	а	б	в	г	д	е	ж	з	и
12	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ы	ъ	э	ю	я	а	б	в	г	д	е	ж	з	и	й
13	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ы	ъ	э	ю	я	а	б	в	г	д	е	ж	з	и	й	к
14	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ы	ъ	э	ю	я	а	б	в	г	д	е	ж	з	и	й	к	л
15	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ы	ь	э	ю	я	а	б	в	г	д	е	ж	з	и	й	к	л	м
16	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ы	ъ	э	ю	я	а	б	в	г	д	е	ж	з	и	й	к	л	м	н
17	с	т	у	ф	х	ц	ч	ш	щ	ь	ы	ъ	э	ю	я	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о
18	т	у	ф	х	ц	ч	ш	щ	ь	ы	ь	э	ю	я	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п
19	у	Ф	х	ц	ч	ш	щ	ь	ы	ъ	э	ю	я	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п	р
20	ф	х	ц	ч	ш	щ	ь	ы	ъ	э	ю	я	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п	р	с
21	х	ц	ч	ш	щ	ь	ы	ъ	э	ю	я	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п	р	с	т
22	ц	ч	ш	щ	ь	ы	ъ	э	ю	я	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п	р	с	т	у
23	ч	ш	щ	ь	ы	ъ	э	ю	я	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п	р	с	т	у	Ф
24	ш	щ	ь	ы	ъ	э	ю	я	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п	р	с	т	у	ф	х
25	щ	ь	ы	ъ	э	ю	я	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц
26	ь	ы	ъ	э	ю	я	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч
27	ы	ь	э	ю	я	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш
28	ъ	э	ю	я	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ
29	э	ю	я	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь
30	ю	я	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ы
31	я	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ы	ъ

Рассмотрим пример получения шифртекста с помощью таблицы Вижинера. Пусть выбрано ключевое слово АМБРОЗИЯ. Необходимо зашифровать сообщение ПРИЛЕТАЮ СЕДЬМОГО.

Выпишем исходное сообщение в строку и запишем под ним ключевое слово с повторением. В третью строку будем выписывать буквы шифртекста, определяемые из таблицы Вижинера.

Сообщение		П	Р	И	Л	Е	Т	А	Ю		С	Е	Д	Ь	М	О	Г	О
Ключ		А	М	Б	Р	О	З	И	Я		А	М	Б	Р	О	З	И	Я
Шифртекст		П	Ъ	Й	Ы	У	Щ	И	Э		С	С	Е	К	Ь	Х	Л	Н

Пользуясь изложенным способом создать программу, которая:

А) зашифрует введенный текст и сохранит его в файл,

Б) считает зашифрованный текст из файла и расшифрует данный текст.

13. Шифр Плейфера

Шифр Плейфера использует матрицу 5x5 (для латинского алфавита, для русского алфавита необходимо увеличить размер матрицы до 6x6), содержащую ключевое слово или фразу. Для создания матрицы и использования шифра достаточно запомнить ключевое слово и четыре простых правила. Чтобы составить ключевую матрицу, в первую очередь нужно заполнить пустые ячейки матрицы буквами ключевого слова (не записывая повторяющиеся символы), потом заполнить оставшиеся ячейки матрицы символами алфавита, не встречающимися в ключевом слове, по порядку (в английских текстах обычно опускается символ «Q», чтобы уменьшить алфавит, в других версиях «I» и «J» объединяются в одну ячейку). Ключевое слово может быть записано в верхней строке матрицы слева направо, либо по спирали из левого верхнего угла к центру. Ключевое слово, дополненное алфавитом составляет матрицу 5x5 и является ключом шифра.

Для того, чтобы зашифровать сообщение необходимо разбить его на биграммы (группы из двух символов), например «Hello World» становится «HE LL OW OR LD», и отыскать эти биграммы в таблице. Два символа биграммы соответствуют углам прямоугольника в ключевой матрице. Определяем положения углов этого прямоугольника относительно друг друга. Затем руководствуясь следующими 4 правилами зашифровываем пары символов исходного текста:

1. Если два символа биграммы совпадают, добавляем после первого символа «X», зашифровываем новую пару символов и продолжаем. В некоторых вариантах шифра Плейфера вместо «X» используется «Q».
2. Если символы биграммы исходного текста встречаются в одной строке, то эти символы замещаются на символы, расположенные в ближайших столбцах справа от соответствующих символов. Если символ является последним в строке, то он заменяется на первый символ этой же строки.
3. Если символы биграммы исходного текста встречаются в одном столбце, то они преобразуются в символы того же столбца, находящимися непосредственно под ними. Если символ является нижним в столбце, то он заменяется на первый символ этого же столбца.
4. Если символы биграммы исходного текста находятся в разных столбцах и разных строках, то они заменяются на символы, находящиеся в тех же строках, но соответствующие другим углам прямоугольника.

Для расшифровки необходимо использовать инверсию этих четырёх правил, откидывая символы «X» (или «Q»), если они не несут смысла в исходном сообщении.

Пример.

Используем ключ «Playfair example», тогда матрица примет вид:

```
P L A Y F
I R E X M
```

B C D G H
J K N O S
T U V W Z

Зашифруем сообщение «Hide the gold in the tree stump»
HI DE TH EG OL DI NT HE TR EX ES TU MP
1. Биграмма HI формирует прямоугольник, заменяем её на BM.
2. Биграмма DE расположена в одном столбце, заменяем её на ND.
3. Биграмма TH формирует прямоугольник, заменяем её на ZB.
4. Биграмма EG формирует прямоугольник, заменяем её на XD.
5. Биграмма OL формирует прямоугольник, заменяем её на KY.
6. Биграмма DI формирует прямоугольник, заменяем её на BE.
7. Биграмма NT формирует прямоугольник, заменяем её на JV.
8. Биграмма HE формирует прямоугольник, заменяем её на DM.
9. Биграмма TR формирует прямоугольник, заменяем её на UI.
10. Биграмма EX находится в одной строке, заменяем её на XM.
11. Биграмма ES формирует прямоугольник, заменяем её на MN.
12. Биграмма TU находится в одной строке, заменяем её на UV.
13. Биграмма MP формирует прямоугольник, заменяем её на IF.
Получаем зашифрованный текст «BM ND ZB XD KY BE JV DM UI XM MN UV IF»
Таким образом, сообщение «Hide the gold in the tree stump» преобразуется в «BMNDZBXDKYBEJVDMUIXMMNUVIF»
Пользуясь изложенным способом создать программу, которая:
А) зашифрует введенный текст и сохранит его в файл,
Б) считает зашифрованный текст из файла и расшифрует данный текст.

14. Шифр с использованием кодового слова

Шифр с использованием кодового слова является одним из самых простых как в реализации, так и в расшифровывании. Идея заключается в том, что выбирается кодовое слово, которое пишется впереди, затем выписываются остальные буквы алфавита в своем порядке. Шифр с использованием кодового слова WORD.
Исходный алфавит: A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
Алфавит замены: W O R D A B C E F G H I J K L M N P Q S T U V X Y Z
Как мы видим, при использовании короткого кодового слова мы получаем очень и очень простую замену. Так же мы не можем использовать в качестве кодового слова слова с повторяющимися буквами, так как это приведет к неоднозначности расшифровки, то есть двум различным буквам исходного алфавита будет соответствовать одна и та же буква шифрованного текста.

Пользуясь изложенным способом создать программу, которая:
А) зашифрует введенный текст и сохранит его в файл,
Б) считает зашифрованный текст из файла и расшифрует данный текст.

15. **Шифр перестановки "скитала"**

Известно, что в V веке до нашей эры правители Спарты, наиболее воинственного из греческих государств, имели хорошо отработанную систему секретной военной связи и шифровали свои послания с помощью *скитала*, первого простейшего криптографического устройства, реализующего метод простой перестановки.

Шифрование выполнялось следующим образом. На стержень цилиндрической формы, который назывался *скитала*, наматывали спиралью (виток к витку) полоску пергамента и писали на ней вдоль стержня несколько строк текста сообщения (рис.1.). Затем снимали со стержня полоску пергамента с написанным текстом. Буквы на этой полоске оказывались расположенными хаотично. Такой же результат можно получить, если буквы сообщения писать по кольцу не подряд, а через определенное число позиций до тех пор, пока не будет исчерпан весь текст.

Сообщение НАСТУПАЙТЕ при размещении его по окружности стержня по три буквы дает шифртекст

НУТАПЕСА_ТЙ

Для расшифрования такого шифртекста нужно не только знать правило шифрования, но и обладать ключом в виде стержня определенного диаметра. Зная только вид шифра, но не имея ключа, расшифровать сообщение было непросто.

Пользуясь изложенным способом создать программу, которая:

А) зашифрует введенный текст и сохранит его в файл,

Б) считает зашифрованный текст из файла и расшифрует данный текст.

16. Простая табличная перестановка

Одним из самых примитивных табличных шифров перестановки является простая перестановка, для которой ключом служит размер таблицы. Этот метод шифрования сходен с шифром *скитала*. Например, сообщение

ТЕРМИНАТОР ПРИБЫВАЕТ СЕДЬМОГО В ПОЛНОЧЬ

записывается в таблицу поочередно по столбцам. Результат заполнения таблицы из 5 строк и 7 столбцов показан на рис.

После заполнения таблицы текстом сообщения по столбцам для формирования шифртекста считывают содержимое таблицы по строкам.

Т	Н	П	В	Е	Г	Л
Е	А	Р	А	Д	О	Н
Р	Т	И	Е	Ь	В	О
М	О	Б	Т	М	П	Ч
И	Р	Ы	С	О	О	Ь

Если шифртекст записывать группами по пять букв, получается такое шифрованное сообщение:

ТНПВЕ ГЛЕАР АДОНР ТИЕЬВ ОМОБТ МПЧИР ЫСООЬ

Естественно, отправитель и получатель сообщения должны заранее условиться об общем ключе в виде размера таблицы. Следует заметить, что объединение букв шифртекста в 5-буквенные группы не входит в ключ шифра и осуществляется для удобства записи несмыслового текста. При расшифровании действия выполняют в обратном порядке.

Пользуясь изложенным способом создать программу, которая:

А) зашифрует введенный текст и сохранит его в файл,

Б) считает зашифрованный текст из файла и расшифрует данный текст

17. Табличная шифровка с ключевым словом

Несколько большей стойкостью к раскрытию обладает метод шифрования, называемый одиночной перестановкой по ключу. Этот метод отличается от предыдущего тем, что столбцы таблицы переставляются по ключевому слову, фразе или набору чисел длиной в строку таблицы.

Применим в качестве ключа, например, слово ПЕЛИКАН,

П	Е	Л	И	К	А	Н
7	2	5	3	4	1	6
Т	Н	П	В	Е	Г	Л
Е	А	Р	А	Д	О	Н
Р	Т	И	Е	Ь	В	О
М	о	Б	Т	М	П	Ч
И	Р	Ы	С	О	О	Ь

До перестановки

А	Е	И	К	Л	Н	П
1	2	3	4	5	6	7
Г	Н	В	Е	П	Л	Т
О	А	А	Д	Р	Н	Е
В	Т	Е	Ь	И	О	Р
П	О	Т	М	Б	Ч	М
О	Р	С	О	Ы	Ь	И

После перестановки

а текст сообщения возьмем из предыдущего примера. На рис. показаны две таблицы, заполненные текстом сообщения и ключевым словом, при этом левая таблица соответствует заполнению до перестановки, а правая таблица - заполнению после перестановки.

В верхней строке левой таблицы записан ключ, а номера под буквами ключа определены в соответствии с естественным порядком соответствующих букв ключа в алфавите. Если бы в ключе встретились одинаковые буквы, они бы были занумерованы слева направо. В правой таблице столбцы переставлены в соответствии с упорядоченными номерами букв ключа. При считывании содержимого правой таблицы по строкам и записи шифртекста группами по пять букв получим шифрованное сообщение:
ГНВЕП ЛТООА ДРНЕВ ТЕЬИО РПОТМ БЧМОР СОЬЬИ
Пользуясь изложенным способом создать программу, которая:
А) зашифрует введенный текст и сохранит его в файл,
Б) считает зашифрованный текст из файла и расшифрует данный текст.

18. Двойная табличная перестановка

Для обеспечения дополнительной скрытности можно повторно зашифровать сообщение, которое уже прошло шифрование. Такой метод шифрования называется *двойной перестановкой*. В случае двойной перестановки столбцов и строк таблицы перестановки определяются отдельно для столбцов и отдельно для строк. Сначала в таблицу записывается текст сообщения, а потом поочередно переставляются столбцы, а затем строки. При расшифровании порядок перестановок должен быть обратным.
Пример выполнения шифрования методом двойной перестановки показан на рис. Если считывать шифртекст из правой таблицы построчно блоками по четыре буквы, то получится следующее: ТЮАЕ ООГМ РЛИП ОЬСВ
Ключом к шифру двойной перестановки служит последовательность номеров столбцов и номеров строк исходной таблицы (в нашем примере последовательности 4132 и 3142 соответственно).

	4	1	3	2
3	П	Р	И	Л
1	Е	Т	А	Ю
4	В	О	С	Ь
2	М	О	Г	О

Исходная таблица

	1	2	3	4
3	Р	Л	И	П
1	Т	Ю	А	Е
4	О	Ь	С	В
2	О	О	Г	М

Перестановка столбцов

	1	2	3	4
1	Т	Ю	А	Е
2	О	О	Г	М
3	Р	Л	И	П
4	О	Ь	С	В

Перестановка строк

Число вариантов двойной перестановки быстро возрастает при увеличении размера таблицы:
• для таблицы 3x3 36 вариантов;
• для таблицы 4x4 576 вариантов;
• для таблицы 5x5 14400 вариантов.
Пользуясь изложенным способом создать программу, которая:
А) зашифрует введенный текст и сохранит его в файл,
Б) считает зашифрованный текст из файла и расшифрует данный текст.

19. Обучение переводу из 10-й системы счисления в двоичную.
Составить программу для обучения переводу чисел из десятичной системы счисления в двоичную и обратно. Программа должна предлагать десятичное (двоичное) число, выбранное с помощью датчика случайных чисел, обучающийся – назвать это число в двоичной (десятичной) системе счисления. Причем, должен быть контроль за временем на размышление.

20. Обучение переводу из 10-й системы счисления в 16-ричную.
Составить программу для обучения переводу чисел из десятичной системы счисления в 16-ричную и обратно. Программа должна предлагать десятичное (16-ричное) число, выбранное с помощью датчика случайных чисел, обучающийся – назвать это число в 16-ричной (десятичной) системе счисления. Причем, должен быть контроль за временем на размышление.

21. Шифровка с помощью магического квадрата
В средние века для шифрования перестановкой применялись и магические квадраты. *Магическими квадратами* называют квадратные таблицы с вписанными в их клетки последовательными натуральными числами, начиная от 1, которые дают в сумме по каждому столбцу, каждой строке и каждой диагонали одно и то же число.

Шифруемый текст вписывали в магические квадраты в соответствии с нумерацией их клеток. Если затем выписать содержимое такой таблицы по строкам, то получится шифртекст, сформированный благодаря перестановке букв исходного сообщения. В те времена считалось, что созданные с помощью магических квадратов шифртексты охраняет не только ключ, но и магическая сила.
Пример магического квадрата и его заполнения сообщением
ПРИЛЕТАЮ ВОСЬМОГО
показан ниже

16	3	2	13
5	10	11	8
9	6	7	12
4	15	14	1

О	И	Р	М
Е	О	С	Ю
В	Т	А	Ь
Л	Г	О	П

Шифртекст, получаемый при считывании содержимого правой таблицы по строкам, имеет вполне загадочный вид:
ОИРМ ЕОСЮ ВТАЬ ЛГОП
Число магических квадратов быстро возрастает с увеличением размера квадрата. Существует только один магический квадрат размером 3x3 (если не учитывать его повороты). Количество магических квадратов 4x4 составляет уже 880, а количество магических квадратов 5x5 - около 250000.
Пользуясь изложенным способом создать программу, которая:

А) зашифрует введенный текст и сохранит его в файл,
Б) считает зашифрованный текст из файла и расшифрует данный текст.

22. Шифровка «тарабарская грамота»

Суть шифровки, которая использовалась в 15-16 веках на Руси, в следующем. Все согласные буквы русской азбуки записывались в два ряда; одна половина букв вверху, другая половина — внизу, причем в обратном порядке (одна буква под другой):

Б В Г Д Ж З К Л М Н
Щ Ш Ч Ц Х Ф Т С Р П

При зашифровке слов согласные взаимно заменялись, а гласные, Й и буквы Ъ, Ь вписывались без изменений. Слова записывались без промежутков между ними, как вообще писался любой текст до 16 века, и это еще больше затрудняло разгадывание.

Пользуясь изложенным способом создать программу, которая:
А) зашифрует введенный текст и сохранит его в файл,
Б) считает зашифрованный текст из файла и расшифрует данный текст.

23. Шифровка «тарабарская грамота» с гласными буквами

Добавим к правилам шифровки согласных (см. предыдущую тему) правило замены гласных при шифровке по правилу:

А Е Ё И О
Я Ю Э Ы У

Пользуясь изложенным способом создать программу, которая:
А) зашифрует введенный текст и сохранит его в файл,
Б) считает зашифрованный текст из файла и расшифрует данный текст.

24. Шифровка «тарабарская грамота» (весь алфавит)

Пусть первые 10 согласных букв русского алфавита заменяются на соответствующие гласные, а остальные в соответствии с таблицей

Б В Г Д Ж З К Л М Н П Р С Т Ф
А Е Ё И О У Ы Э Ю Я Щ Ш Ч Ц Х

Пользуясь изложенным способом создать программу, которая:
А) зашифрует введенный текст и сохранит его в файл,
Б) считает зашифрованный текст из файла и расшифрует данный текст.

29. Магический квадрат

Магическим квадратом порядка N называется квадратная таблица размера NxN , составленная из чисел 1, 2, … N^2 так, что суммы по каждому столбцу, каждой строке и каждой из двух диагоналей равны между собой. Составить программу для построения магического квадрата для заданного N.

ПРАВИЛА ВЫПОЛНЕНИЯ И ОФОРМЛЕНИЯ КУРСОВОЙ РАБОТЫ

Общие положения

Основные задачи и цели курсового проектирования:

1) приобретение навыков и методов программирования достаточно сложных задач
2) подготовка к выполнению дипломного проекта

Общие требования к построению пояснительной записки (ПЗ)

Структура построения ПЗ

ПЗ к работе должна содержать следующие разделы:

1) титульный лист (см. приложение 2)
2) реферат
3) задание на проектирование (см. приложение 3)
4) содержание
5) введение
6) основная часть работы
7) заключение
8) список литературы
9) приложения

Титульный лист оформляется согласно ГОСТ 2.105-79 (см. приложение 2)

Реферат – краткая характеристика работы с точки зрения содержания, назначения, формы и других особенностей. Перечисляются ключевые слова работы, указывается количество страниц и приложений. Реферат размещают на отдельной странице. Заголовком служит слово «Реферат», написанное прописными буквами.

Задание на проектирование заполняется студентом в соотвествии с полученным заданием. Форма задания приведена в приложении 3.

Содержание включает наименование всех разделов, подразделов и пунктов, если они имеют наименование, а также список литературы и приложения с указанием номера страниц, на которых они начинаются. Слово «Содержание» записывается в виде заголовка, симметрично тексту, прописными буквами. Пример оформления содержания см. в приложении 4.

Введение содержит основную цель курсовой работы, область применения разрабатываемой темы.

Заключение должно содержать краткие выводы по выполненной работе. Также следует указать, чему программист научился на примере этой задачи.

Список литературы включает все те и только те источники литературы, на которые имеются ссылки в ПЗ. Пример – см. в приложении 5.

Приложения содержат вспомогательный материал, листинг программы и листинг тестов. При этом программа должна быть самодокументированной, т.е.

- иметь простую и понятную структуру
- в программе должны быть прокомментированы используемые структуры данных
- для данной подпрограммы должно быть указано, что она делает, что является входными данными и результатом
- должен быть прокомментирован используемый алгоритм

Основная часть курсовой работы

В основной части должно быть решение поставленной задачи, в частности:

- анализ задачи,
- обоснование выбора алгоритма,
- обоснование выбора структур данных,
- описание алгоритма,
- обоснование набора тестов

Об анализе задачи.

Разработка алгоритма представляет собой задачу на построение. Поэтому как обычно, в таких случаях (можно, например, вспомнить о методе решения геометрических задач на построение), необходим этап анализа задачи. Он позволяет установить, что является входом и выходом будущего алгоритма, выделить основные необходимые отношения между входными и выходными объектами и их компонентами, выделить подцели, которые нужно достичь для решения задачи, и, как следствие этого, выработать подход к построению алгоритма, т.е. формулировка в самом общем виде того, что (в рамках выбранного подхода) должен делать алгоритм, чтобы переработать входные данные в выходные.

Об описании алгоритма.

Прежде всего, нужно иметь в виду, что такое описание предназначено не для машины, а для человека. Другими словами, речь идет не о программе, а о некотором тексте (т.е. о словесном описании), по которому можно получить представление об общей структуре разрабатываемого алгоритма, о смысле его отдельных шагов и их логической взаимосвязи. Сохранение достаточно высокого уровня описания алгоритма также облегчает его обоснование. Поэтому шаги

алгоритма должны описываться в терминах тех объектов и отношений между ними, о которых идет речь в формулировке задачи. Например, для «геометрической» задачи шаги алгоритма следует описывать как действия над точками, прямыми и т.п. Но не должно быть работы с кодами этих объектов, например с матрицей координат точек некоторого множества.

О выборе представления данных.

Данные, обрабатываемые алгоритмом, делятся на входные, промежуточные и выходные. Специфика входных и выходных данных состоит в том, что с ними имеет дело не только алгоритм, но и пользователь программы. Поэтому различают две формы представления таких данных – внешнее и внутреннее.

Основное требование к внешнему представлению состоит в его максимальном удобстве, понятности и естественности для пользователя, чтобы он мог достаточно легко подготовить данные для ввода и оценить результат по выходным данным. Выбор внутреннего представления определяется главным образом требованием эффективности того алгоритма, который нужно построить для решения задачи.

О выборе тестов

Нужно подобрать набор тестов, достаточный для демонстрации работы программы и ее реакции на экстремальные ситуации и неправильное обращение.

Правила оформления ПЗ к курсовой работе

ПЗ пишется в редакторе MS Word шрифтом Times New Roman, размером 12, на формате А4. Нумерация страниц должна быть сквозной, первой страницей является титульный лист. Номер страницы проставляется сверху посередине. Заголовки разделов пишутся прописными буквами посредине листа. Заголовки подразделов пишутся с абзаца строчными буквами, кроме первой прописной. В заголовке не допускаются переносы слов. Точку в конце заголовка не ставят. Если заголовок состоит из двух предложений, то их разделяют точкой.

ПРИЛОЖЕНИЯ

Приложение 1. Форма титульного листа к курсовой работе

Министерство образования и науки РФ

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра комплексной информационной безопасности электронно-вычислительных систем (КИБЭВС)

Шифр Цезаря
Пояснительная записка к курсовой работе по дисциплине
“Информатика 2”

Студент гр. 417
(подпись) В.Н.Петров
01.04. 10

Руководитель
Доцент каф. КИБЭВС
_____ В. Н. Кирнос
_____

2010

Приложение 2. Форма задания для курсовой работы

Министерство образования и науки РФ

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра комплексной информационной безопасности электронно-вычислительных систем (КИБЭВС)

УТВЕРЖДАЮ
Зав. кафедрой КИБЭВС
(подпись) А.А,Шелупанов
“___” _____ 2011 г.

ЗАДАНИЕ
по курсовому проектированию по дисциплине“Информатика 2”
студенту_____
группа _____ факультет _____.

Тема проекта: ”Шифр Цезаря”
2. Срок сдачи студентом законченной работы _____
3. Исходные данные к проекту: (Текст задания)_____
_____
_____
_____
_____
_____
_____
_____
_____

7. Дата выдачи задания: _____

Задание принял к исполнению _____ (дата)
(Ф.И.О.)_____(подпись студента)

Приложение 3. Пример оформления содержания курсовой работы

СОДЕРЖАНИЕ

Приложение 4. Пример списка литературы курсовой работы

СПИСОК ЛИТЕРАТУРЫ

5. Павловская Т. C/C++. Программирование на языке высокого уровня. Учебник. – СПб.: Питер, 2001 – 460 с.
6. Павловская Т., Щупак Ю. C++. Объектно-ориентированное программирование. Практикум. – СПб.: Питер, 2006 – 264 с.
7. Лафоре Р. Объектно-ориентированное программирование в C++. – СПб.: Питер, 2007. – 928 с.
8. Хортон А. Visual C++ 2005: базовый курс. – М.: ООО «ИД Вильямс», 2007 – 1152 с.