МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Белорусский национальный технический университет

Кафедра «Тепловые электрические станции»

ИНФОРМАТИКА

Практикум

Часть 1

Минск
БНТУ
2017

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
Белорусский национальный технический университет

Кафедра «Тепловые электрические станции»

ИНФОРМАТИКА

Практикум
по программированию на языке C++
для студентов специальностей
1-43 01 04 «Тепловые электрические станции»,
1-43 01 08 «Паротурбинные установки атомных
электрических станций»

В 2 частях

Часть 1

Рекомендовано учебно-методическим объединением по образованию в области энергетики и энергетического оборудования

Минск
БНТУ
2017

УДК 004(076.5)
ББК 32.97я7
И74

Составители :
Л. А. Тарасевич, Е. В. Пронкевич,
В. А. Романко, С. М. Денисов

Рецензенты :
кафедра электроники Белорусского государственного университета информатики и радиоэлектроники (зав. кафедрой, канд. техн. наук, доцент *С. М. Сацук*);
канд. физ.-мат. наук кафедры ПАСТ Государственного учреждения образования «Командно-инженерный институт» Министерства по чрезвычайным ситуациям Республики Беларусь *Т. М. Мартыненко*

И74 **Информатика :** практикум по программированию на языке С++ для студентов специальностей 1-43 01 04 «Тепловые электрические станции», 1-43 01 08 «Паротурбинные установки атомных электрических станций» : в 2 ч. Ч. 1 / сост. : Л. А. Тарасевич [и др.]. – Минск : БНТУ, 2017. – 110 с.
ISBN 978-985-550-854-1 (Ч. 1).

В практикуме приведены краткие теоретические сведения по основам программирования на алгоритмическом языке С++, варианты заданий к каждой лабораторной работе, контрольные вопросы.

УДК 004(076.5)
ББК 32.97я7

ISBN 978-985-550-854-1 (Ч. 1)
ISBN 978-985-550-855-8

Содержание

ВВЕДЕНИЕ

Данный практикум написан в рамках изучения курса информатики студентами технических специальностей.

С++ – это новый язык, основанный на Си, который в свою очередь дополняет большинство современных языков программирования.

С++ – компилируемый язык программирования общего назначения, сочетает свойства как высокоуровневых, так и низкоуровневых языков программирования.

Был создан в начале 1980-х годов сотрудником фирмы Bell laboratories – Бьёрном Страуструпом.

С++ включает все операторы и средства языка Си, добавив несколько новых. Преимущество данного языка программирования в том, что он позволяет разрабатывать сложные программы за счет более модульного подхода. Кроме того, С++ является языком объектно-ориентированного программирования.

Лабораторная работа № 1

ВЫЧИСЛЕНИЕ ФУНКЦИИ

Цель работы:

– приобрести практические навыки составления линейных программ на языке C++;

– представить алгоритм решения задачи в виде блок-схемы.

Теоретические сведения

Правила написания программ

При написании программы на языке C++ следует соблюдать нижепредставленные правила.

1. Операторы при записи алгоритма должны располагаться последовательно слева направо и сверху вниз. Порядок чтения может изменяться при помощи специальных операторов (см. ниже).

2. Все константы, типы, переменные и функции должны быть объявлены до их первого использования в любом месте программы.

3. При объявлении и использовании констант, типов, переменных и функций необходимо помнить, что в языке C++ прописное и строчное написание одной и той же буквы считается различными символами.

4. Каждый оператор заканчивается точкой с запятой (исключения см. ниже), поэтому в одной строке можно располагать несколько операторов или один оператор можно располагать на нескольких строках (не разбивая идентификаторы).

5. Для удобства разработки можно использовать комментарии, которые не обрабатываются компилятором и служат для улучшения читабельности программы. Комментарием считается все, заключенное в скобки /* … */ или расположенное правее //.

6. Если необходимо объединить несколько операторов, то используется составной оператор. Составным оператором считается все, заключенное в фигурные скобки { … }.

Алфавит языка C++

В языке C++ используются наборы символов:

1) прописные (*A*, *B*, *C*, … , *Y*, *Z*) и строчные (*a*, *b*, *c*, … , *y*, *z*) буквы латинского алфавита;

2) арабские цифры от 0 до 9;

3) специальные символы.

Специальные символы позволяют задавать операторы и знаки операций. Некоторые из них представлены в табл. 1.1.

Таблица 1.1

Символ	Операция
+	Сложение
–	Вычитание
*	Умножение
/	Деление
%	Деление по модулю (остаток от деления)
++	Увеличение на единицу
--	Уменьшение на единицу
=	Присваивание
*=	Умножение с присваиванием
/=	Деление с присваиванием
%=	Деление по модулю (остаток от деления) с присваиванием
+=	Сложение с присваиванием
–=	Вычитание с присваиванием
&&	Логическое И
||	Логическое ИЛИ
!	Логическое отрицание
= =	Сравнение на равенство
>	Сравнение на больше
> =	Сравнение на больше или равно
<	Сравнение на меньше
< =	Сравнение на меньше или равно
! =	Сравнение на не равно
>>	Сдвиг вправо
<<	Сдвиг влево

Все объекты (переменные, массивы и т. д.), с которыми работает программа в С++, необходимо декларировать. При декларировании объекты можно инициализировать (задать начальные значения).

Пример 1.1.

int *j*=10, *m*=3;
float *c*=–1.3, *l*=–10.23, *n*

При декларировании объектов в языке С++ используются их идентификаторы, которые могут включать цифры (0...9), латинские прописные (*A*...*Z*) и строчные (*a*...*z*) буквы, символ подчеркивания _. Первый символ идентификатора не может быть цифрой. Принято использовать в идентификаторах переменных строчные буквы, а в именованных константах – прописные.

Пример 1.2.

const float *Pi*=3.1415926;
float *Pi*=3.14

Идентификаторы

Идентификаторы – это слова, которые используются как имена переменных, функций, типов данных. Они состоят из букв, цифр, подчеркиваний и не могут начинаться с цифры. Длина их произвольна.

Прописные и строчные буквы различаются.

Для разделения идентификаторов служат пробелы, табуляции, «на новую строку», «на новую страницу».

Комментарии

– однострочные: //;
– многострочные: /*…*/.

Комментарии не могут быть вложенными.

Типы данных

В языке C++ существуют типы данных, представленные в табл. 1.2.

Таблица 1.2

Тип	Длина в байтах	Диапазон
char	8	–128..127
int	16	–32768..32767
float	32	$3.4*10^{-38}..3.4*10^{38}$
double	64	$1.7*10^{-308}..1.7*10^{308}$

Есть тип данных, который обозначает пустое значение: void. Он используется для двух целей:

– определения функций, которые не возвращают значения;

– создания родовых (generic) указателей, то есть указателей на выделенную программой память. В дальнейшем этот указатель может быть преобразован в другой тип.

Для более точного выбора типов данных используются модификаторы типов:

– модификаторы знака:

signed – со знаком;

unsigned – без знака;

– модификаторы длины:

short – короткие;

long – длинные.

С типом char используются только модификаторы знака.

signed char (=char)		–128..127
unsigned char		0..255

А с типом integer используются как модификаторы знака, так и модификаторы длины.

short int	8 или 16	Со знаком
short		
signed short int		
signed short		

unsigned short int unsigned short	8 или 16	Без знака
signed int signed int	16	Со знаком
unsigned int unsigned	16	Без знака
long int long signed long int signed long	32 или 16	–2 147 483 648..2 147 483 647

С типом float модификаторы не работают.

С типом double используется только модификатор long.

Long double	80	$3.4*10^{-4932}..3.4*10^{4932}$

Константы

Константами называются неизменяемые величины в программе. Они могут быть четырех типов:

– целые (десятичное, восьмиричное или шестнадцатиричное целое число);

– с плавающей точкой (десятичное число, представляемое в виде действительной величины с фиксированной или плавающей точкой);

– символьные (символы, заключенные в одинарые скобки … ’или, если это строка, – в двойные " … ");

– перечисляемые (некая последовательность имен, которая автоматически нумеруется, начиная с 0).

Для улучшения читабельности программ константами им можно давать имена.

Пример 1.3.

```
const M=5; // Константа целого типа
const float Pi2=3.14/2; // Константа действительного типа
char const *Name="Alex"; // Указатель на строковую константу
```

Переменные

Переменными называются идентификаторы, значения которых могут меняться в процессе выполнения программы. Переменная может объявляться отдельным оператором до ее первого использования.

Синтаксис:
<тип> <список идентификаторов>;

Пример 1.4.

float *d*;
int *i*, *j*, *k*;
или внутри операторов:
for (int *i*=0, *i*<3, *i*++)

Операторы

Арифметические операторы

В табл. 1.3 приведен список арифметических операторов языка C++. Операторы +, –, * и / выполняются точно так же, как и в большинстве языков программирования. Их можно применять практически к любым встроенным типам данных. Если оператор / применяется к целому числу или символу, дробная часть отбрасывается. Например, 5/2 равно 2.

Таблица 1.3

Оператор	Действие
–	Вычитание, а также унарный минус
+	Сложение
*	Умножение
/	Деление
%	Деление по модулю
--	Декрементация
++	Инкрементация

Оператор деления по модулю %, как и в других языках программирования, возвращает остаток целочисленного деления. Однако этот оператор нельзя применять к числам с плавающей точкой.

Унарный минус умножает свой операнд на –1. Иными словами, он меняет знак операнда на противоположный.

В языке C++ есть два полезных оператора, которыми не обладают некоторые другие языки. Это операторы инкрементации и декрементации ++ и --. Оператор ++ добавляет 1 к своему операнду, а оператор -- вычитает ее.

$$x = x + 1;$$
$$++x;$$

Операторы инкрементации и декрементации имеют две формы: префиксную (++*x*) и постфиксную (*x*++). Однако между ними существует важное отличие, когда они используются внутри выражений. Если используется префиксная форма, операторы инкрементации и декрементации возвращают значению операнда после изменения, а если постфиксная – до.

Пример 1.5.

```
int   x=10;
y=++x;      // y=11;
y=x++;      // y=10
```

Операторы сравнения и логические операторы

В термине *оператор сравнения* слово «сравнение» относится к значениям операндов. В термине *логический оператор* слово «логический» относится к способу, которым устанавливаются эти отношения. Поскольку операторы сравнения и логические операторы тесно связаны друг с другом, рассмотрим их вместе.

В основе и операторов сравнения, и логических операторов лежат понятия «истина» и «ложь». В языке C++ истинным считается любое значение, не равное нулю. Ложное значение всегда равно 0. Выражения, использующие операторы сравнения и логические операторы, возвращают 0, если результат ложен, и 1, если результат истинен.

Операторы сравнения и логические операторы приведены в табл. 1.4.

Таблица 1.4

Оператор	Обозначения в математике	Действие
Операторы сравнения		
>	>	Больше
>=	≥	Больше или равно
<	<	Меньше
<=	≤	Меньше или равно
==	=	Равно
!=	≠	Не равно
Логические операторы		
&&	∧	И
||	∨	ИЛИ
!	¬	НЕ

Оператор присваивания

Оператор присваивания (=) не обязан стоять в отдельной строке и может входить в более крупные выражения.

Составные операторы присваивания объединяют логические и арифметические операторы с оператором присваивания (табл. 1.5, 1.6).

Таблица 1.5

Арифметические + присваивание	
+=	*x*+=*y* // *x*=*x*+*y*
–=	*x*–=*y* // *x*=*x*–*y*
*=	*x**=*y* // *x*=*x***y*
/=	*x*/=*y* // *x*=*x*/*y*
%=	*x*%=*y* // *x*=*x*%*y*

Таблица 1.6

Побитовые + присваивание	
<<=	*x*<<=*y* // *x*=*x*<<*y*
>>=	*x*>>=*y* // *x*=*x*>>*y*
&=	*x* &=*y* // *x*=*x*&*y*
^=	*x*^=*y* // *x*=*x*^*y*
|=	*x*|=*y* // *x*=*x*|*y*

Математические функции

В C++ определены в заголовочном файле <cmath> функции, выполняющие некоторые часто используемые математические задачи. Например, нахождение корня, возведение в степень, sin(), cos() и многие другие. В табл. 1.7 показаны основные математические функции, прототипы которых содержатся в заголовочном файле <cmath>.

Пример 1.6. Найти синус и косинус *A*.

```
#include // библиотека ввода-вывода
#include <math.h> // математическая библиотека
using namespace std;
int main()
{
double a, b, c; // переменные типа double
cout << «Input A=»; cin>>a; //ввод a
b=cos(a); //присваивание значений
c=sin(a); //
cout << “Cos A=” <<b<<endl; cout << “Sin A=” <<c; //вывод a
return 0;
}
```

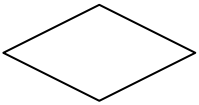
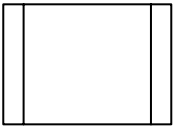
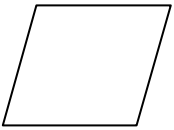
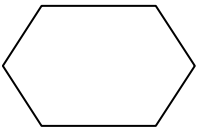
Таблица 1.7

Математическая функция	Имя функции в языке С++
$\sqrt{x}$	sqrt(x)
$\|x\|$	int abs(int n); double fabs(double x)
e^x	exp(x)
x^y	pow(x, y)
ln(x)	log(x)
$lg_{10}(x)$	log10(x)
sin(x)	sin(x)
cos(x)	cos(x)
tg(x)	tan(x)
arcsin(x)	asin(x)
arccos(x)	acos(x)
arctg(x)	atan(x)
sh(x)=1/2 (e^x–e^{-x})	sinh(x)
ch(x)=1/2 (e^x+e^{-x})	cosh(x)
tgh(x)	tanh(x)
Остаток от деления x на y	fmod(x, y)
Наименьшее целое, которое >=x	ceil(x)
Наибольшее целое, которое <=x	floor(x)

Графический способ представления алгоритмов

К графическому способу относят блок-схемы, представленные в табл. 1.8.

Таблица 1.8

Блоки для схемы	Обозначения блоков
	*Начало и конец алгоритма* (для функций «Вход», «Выход»)
	*Блок обработки.* Внутри блока записываются формулы, обозначения и функции
	*Блок условия.* Внутри блока записываются условия выбора направления действия алгоритма
	*Блок предопределенного процесса* (функция/ подпрограмма)
	*Блок ввода информации*
	*Блок цикла с известным количеством повторений*
	*Соединительный блок*

Алгоритмы бывают *линейные*, *разветвляющиеся* и *циклические*.

Линейный алгоритм не содержит логических условий, имеет одну ветвь обработки и изображается линейной последовательностью связанных друг с другом блоков. Условное изображение линейного алгоритма представлено на рис. 1.1.

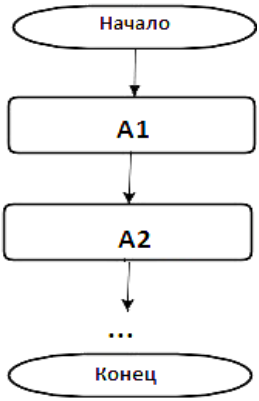


Рис. 1.1. Условное изображение линейного алгоритма

Пример 1.7. Составить блок-схему вычисления $z = \varphi(x)$, $y = f(x)$, где φ, f – известные функции, при заданном значении переменной x (рис. 1.2).

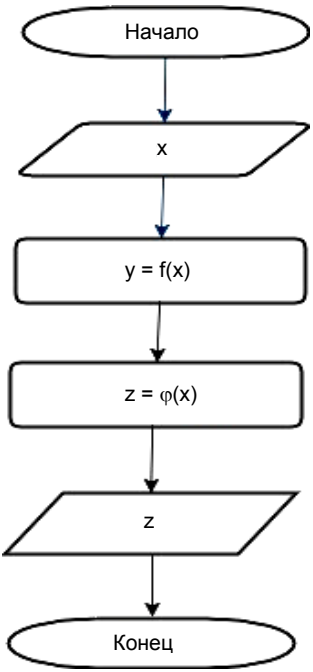


Рис. 1.2. Блок-схема вычислений $z = \varphi(x)$, $y = f(x)$

Пример 1.8. Составить блок-схему нахождения корней квадратного уравнения. Коэффициенты квадратного уравнения ввести с клавиатуры (рис. 1.3).

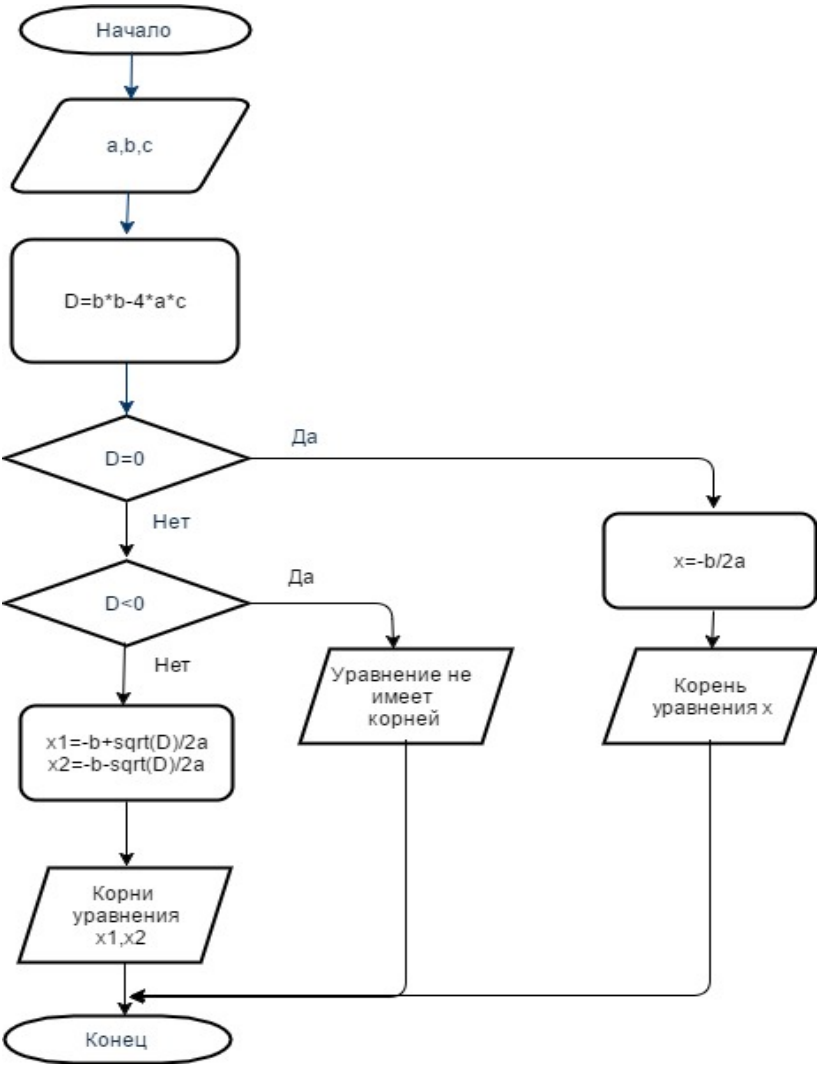


Рис. 1.3. Блок-схема нахождения корней квадратного уравнения

Циклический алгоритм содержит один или несколько циклов.

Цикл – это многократно повторяемая часть алгоритма.

Цикл, не содержащий внутри себя других циклов, называют простым. Если он содержит внутри себя другие циклы или разветвления, то цикл называют сложным или вложенным. Любой цикл характеризуется одной или несколькими переменными, называемыми параметрами цикла, от анализа значений которых зависит выполнение цикла.

Параметр цикла – переменная, принимающая при каждом вхождении в цикл новое значение. Условное изображение циклического алгоритма представлено на рис. 1.4.

Рис. 1.4. Блок-схема циклического алгоритма

Пример 1.9. Составить блок-схему нахождения суммы N первых целых чисел от 0 до $N - 1$ (рис. 1.5).

Исходные файлы представляют собой текстовые файлы с расширением .cpp. Исполняемые файлы имеют расширение .exe и могут запускаться как из компилятора, так и непосредственно в режиме MS DOS.

Программы, исполняемые с помощью компилятора Microsoft или из MS DOS, не требуют никакого дополнительного редактирования.

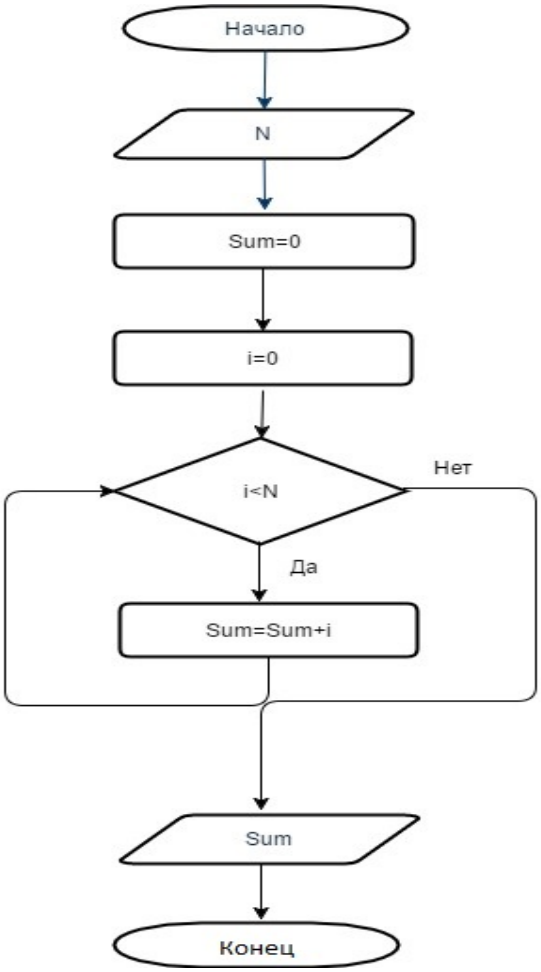


Рис. 1.5. Блок-схема для примера 1.9

Первая программа

Структура программы

Рассмотрим первый, самый простой, пример программы на C++ под названием FIRST. Соответствующий файл исходного кода называется FIRST.CPP. Программа выводит сообщение на экран.

Пример 1.10.

```
#include <iostream>
 using namespace std;
 int main()
{
cout «"моя первая программа\n";
 return 0;
}
```

Несмотря на свой небольшой размер, этот пример демонстрирует типичную структуру программы на C++. Рассмотрим эту структуру в деталях.

Тело функции

Тело функции заключено в фигурные скобки, которые играют ту же роль, что и ключевые слова BEGIN и END, встречающиеся в некоторых других языках программирования: они определяют границы блока операторов программы. Фигурные скобки, обрамляющие тело функции, обязательны. В нашем примере тело функции состоит всего лишь из двух операторов: один из них начинается словом cout, другой – return. Разумеется, функция может включать в себя и большее число операторов.

Функция main()

Когда программа на языке C++ запускается на выполнение, первым исполняемым оператором становится первый оператор функции main() (по крайней мере, это справедливо для консольных программ). Программа может состоять из множества функций, классов и прочих элементов, но при ее запуске управление всегда передается функции main(). Если в программе не содержится функции с именем main(), то при попытке запустить такую программу будет выведено сообщение об ошибке.

В большинстве программ, написанных на C++, реальные действия сводятся к вызову функцией main() методов различных объектов. Кроме того, функция main() может вызывать другие, независимые функции.

Заголовочные файлы

IOSTREAM является примером заголовочного (или включаемого) файла. Файл IOSTREAM содержит описания, необходимые для работы с переменной cout и операцией «. Без них компилятору не будет известно, что значит имя cout, а употребление операции « будет воспринято как некорректное.

Директива using

Директива using namespace std; означает, что все определенные ниже имена в программе будут относиться к пространству имен с именем std. Различные элементы программы описаны с использованием пространства имен std, например переменная cout. Если не использовать директиву using, то к этим элементам программы придется каждый раз добавлять имя std: std::cout « "У каждой эпохи свой язык*n*";

Для того чтобы не дописывать std:: каждый раз перед именем переменной, используется директива using.

Комментарии

Комментарии являются важной частью любой программы. Они помогают разобраться в действиях программы как разработчику, так и любому другому человеку, читающему код. Компилятор игнорирует все, что помечено в программе как комментарий, поэтому комментарии не включаются в содержимое исполняемого файла и никак не влияют на ход исполнения программы.

Функция стандартного вывода printf()

Функция printf() – функция форматированного вывода. Это означает, что в параметрах функции необходимо указать формат данных, которые будут выводиться. Он указывается спецификатором формата, начинающегося с символа %, за которым следует код формата (табл. 1.9).

Таблица 1.9

Спецификаторы формата	Обозначения
%*c*	Символ
%*d*	Целое десятичное число
%*i*	Целое десятичное число
%*e*	Десятичное число в виде *x.xx e+xx*
%*E*	Десятичное число в виде *x.xx E+xx*
%*f*	Десятичное число с плавающей запятой *xx.xxxx*
%*F*	Десятичное число с плавающей запятой *xx.xxxx*
%*g*	%*f* или %*e*, что короче
%*G*	%*F* или %*E*, что короче
%o	Восьмеричное число
%*s*	Строка символов
%*u*	Беззнаковое десятичное число
%*x*	Шестнадцатеричное число
%*X*	Шестнадцатеричное число
%%	Символ %
%*p*	Указатель
%*n*	Указатель

В спецификаторе формата после символа % может быть указана точность (число цифр после запятой). Точность задается следующим образом: %.*n*<код формата>. Где *n* – число цифр после запятой, а <код формата> – один из кодов, приведенных выше.

Например, если у нас есть переменная *x*=10.3563 типа float и хотим вывести ее значение с точностью до трех цифр после запятой, то необходимо написать:

printf("Переменная *x* = %.3*f*",*x*);

Результат:
Переменная *x*=10.356

Также можно указать минимальную ширину поля, отводимого для печати. Если строка или число больше указанной ширины поля, то строка или число печатается полностью.

Например, если написать:

```
printf("%5d", 20);
```

то результат будет следующим:

```
20
```

Обратите внимание на то, что число 20 напечаталось не с самого начала строки. Если вы хотите, чтобы неиспользованные места поля заполнялись нулями, то нужно поставить перед шириной поля символ 0.

```
printf("%05d", 20);
```

Результат:

```
00020
```

Кроме спецификаторов формата данных в управляющей строке могут находиться управляющие символы (табл. 1.10).

Таблица 1.10

Спецификаторы формата	Обозначения
\f	Новая страница, перевод страницы
\n	Новая строка, перевод строки
\r	Возврат каретки
\t	Горизонтальная табуляция
\v	Вертикальная табуляция
\"	Двойная кавычка
\'	Апостроф
\\	Обратная косая черта
\0	Нулевой символ, нулевой байт
\a	Сигнал
\N	Восьмеричная константа
\xN	Шестнадцатеричная константа
\?	Знак вопроса

Чаще всего используется символ \n. С помощью этого управляющего символа можно переходить на новую строку.

Пример 1.11.

```
#include <stdio.h>
void main(void)
{
   int a, b, c;        // Объявление переменных a, b, c
   a=5;
   b=6;
   c=9;
   printf("a=%d, b=%d, c=%d", a, b, c);
}
```

Результат работы программы:

a=5, b=6, c=9

Пример 1.12.

```
#include <stdio.h>
void main(void)
{
   float x, y, z;
   x=10.5;
   y=130.67;
   z=54;
   printf("Координаты объекта: x:%.2f, y:%.2f, z:%.2f", x, y, z);
}
```

Результат работы программы:

Координаты объекта: x:10.50, y:130.67, z:54.00

Пример 1.13.

```
#include <stdio.h>
void main()
{
   int x;
   x=5;
   printf("x=%d", x*2);
}
```

Результат работы программы:
x=10

Пример 1.14.

```
#include <stdio.h>
void main(void)
{
   printf("\"Текст в кавычках\"");
   printf("\nСодержание кислорода: 100%%");
}
```

Результат работы программы:
"Текст в кавычках"
Содержание кислорода: 100%

Пример 1.15.

```
#include <stdio.h>
void main(void)
{
   char ch1, ch2, ch3;
   ch1='A';
   ch2='B';
   ch3='C';
   printf("%c%c%c", ch1, ch2, ch3);
}
```

Результат работы программы:
ABC

Пример 1.16.

```
#include <stdio.h>
void main(void)
{
   char *str="Моя строка.";
   printf("Это %s", str);
}
```

Результат работы программы:
Моя строка.

Пример 1.17.

```
#include <stdio.h>
void main(void)
{
printf("Здравствуйте!\n");     // После печати будет переход на новую строку – \n
printf("Меня зовут Павел.");  // Это будет напечатано на новой строке
}
```

Результат работы программы:
Здравствуйте!
Меня зовут Павел.

Функция стандартного ввода scanf()

Функция scanf() – функция форматированного ввода. С ее помощью можно вводить данные со стандартного устройства ввода (клавиатуры). Вводимыми данными могут быть целые числа, числа с плавающей запятой, символы, строки и указатели.

Управляющая строка содержит три вида символов: спецификаторы формата, пробелы и другие символы. Спецификаторы формата начинаются с символа % (табл. 1.11).

Таблица 1.11

Спецификаторы формата	Обозначения
%c	Чтение символа
%d	Чтение десятичного целого
%i	Чтение десятичного целого
%e	Чтение числа типа float (плавающая запятая)
%h	Чтение short int
%o	Чтение восьмеричного числа
%s	Чтение строки

Окончание табл. 1.11

Спецификаторы формата	Обозначения
%*x*	Чтение шестнадцатеричного числа
%*p*	Чтение указателя
%*n*	Чтение указателя в увеличенном формате

При вводе строки с помощью функции scanf() (спецификатор формата %*s*), строка вводится до первого пробела!

```
char str[80];       // массив на 80 символов
scanf("%s", str);
```

То есть если вводить строку «Привет мир!» с использованием функции scanf(), то после ввода результирующая строка, которая будет храниться в массиве str будет состоять из одного слова «Привет».

Для ввода данных с помощью функции scanf() в качестве параметров ей нужно передавать адреса переменных, а не сами переменные. Чтобы получить адрес переменной, нужно поставить перед именем переменной знак & (амперсанд), который означает взятие адреса.

Адрес, который получаем с помощью & – это адрес в памяти компьютера, где храниться значение переменной.

Давайте рассмотрим пример программы, который показывает нам как использовать &.

Пример 1.18.

```
#include <stdio.h>
void main(void)
{
   int x;
   printf("Введите переменную x:");
   scanf("%d", &x);
   printf("Переменная x=%d", x);
}
```

Пример 1.19.

```
scanf("%d%*c%d", &i, &j);
```

При вводе 50 + 20 присвоит переменной *i* значение 50, переменной *j* – значение 20, а символ + будет прочитан и проигнорирован.

В команде формата может быть указана наибольшая ширина поля, которая подлежит считыванию.

```
scanf("%5s", str);
```

Указывает необходимость прочитать из потока ввода первые пять символов. При вводе 1234567890ABC массив str будет содержать только 12345, остальные символы будут проигнорированы. Разделители: пробел, символ табуляции и символ новой строки – при вводе символа воспринимаются, как и все другие символы.

Одной из особенностей функции scanf() является возможность задания множества поиска (scanset). Оно определяет набор символов, с которыми будут сравниваться читаемые функцией scanf() символы. Функция scanf() читает символы до тех пор, пока они встречаются в множестве поиска. Как только символ, который введен, не встретился в множестве поиска, функция scanf() переходит к следующему спецификатору формата. Множество поиска определяется списком символов, заключенных в квадратные скобки. Перед открывающей скобкой ставиться знак %. Давайте рассмотрим это на примере.

Пример 1.20.

```
#include <stdio.h>
void main(void)
{
   char str1[10], str2[10];
   scanf("%[0123456789]%s", str1, str2);
   printf("\n%s\n%s", str1, str2);
}
```

Введем набор символов:

```
12345abcdefg456
```

На экране программа выдаст:
12345
abcdefg456

При задании множества поиска можно также использовать символ «дефис» для задания промежутков, а также максимальную ширину поля ввода.

```
scanf("%10[A–Z1–5]", str1);
```

Можно также определить символы, которые не входят в множество поиска. Перед первым из этих символов ставиться знак ^. Множество символов различает строчные и прописные буквы.

При использовании функции scanf() в качестве параметров ей нужно передавать адреса переменных. Выше был написан код:

```
char str[80];       // массив на 80 символов
scanf("%s", str);
```

Обратите внимание на то, что перед str не стоит символ &. Это сделано потому, что str является массивом, а имя массива – str является указателем на первый элемент массива. Поэтому знак & не ставится.

Пример 1.21.

Эта программа выводит на экран запрос «Сколько вам лет?:» и ждет ввода данных. Если, например, ввести число 20, то программа выведет строку «Вам 20 лет.» При вызове функции scanf() перед переменной age поставили знак &, так как функции scanf() нужны адреса переменных. Функция scanf() запишет введенное значение по указанному адресу. В нашем случае введенное значение 20 будет записано по адресу переменной age.

```
#include <stdio.h>
void main(void)
{
   int age;
   printf("\nСколько вам лет?:");
```

```
   scanf("%d", &age);
   printf("Вам %d лет.", age);
}
```

Пример 1.22.

Программа калькулятор. Этот калькулятор может только складывать числа. При вводе 100 + 34 программа выдаст результат: 100 + 34 = 134.

```
   #include <stdio.h>
void main(void)
{
   int x, y;
   printf("\nКалькулятор:");
   scanf("%d+%d", &x, &y);
   printf("\n%d+%d=%d", x, y, x+y);
}
```

Пример 1.23.

Этот пример показывает, как установить ширину поля считывания. В нашем случае ширина поля равна пяти символам. Если ввести строку с большим количеством символов, то все символы после пятого будут отброшены. Обратите внимание на вызов функции scanf(). Знак & не стоит перед именем массива name, так как имя массива name является адресом первого элемента массива.

```
#include <stdio.h>
void main(void)
{
   char name[5];

   printf("\nВведите ваш логин (не более 5 символов):");
   scanf("%5s", name);
   printf("\nВы ввели %s", name);
}
```

Пример 1.24.

Последний пример в этой статье показывает, как можно использовать множество поиска. После запуска программы введите число от 2 до 5.

```
#include <stdio.h>
void main(void)
{
   char bal;
   printf("Ваша оценка 2, 3, 4, 5:");
   scanf("%[2345]", &bal);
   printf("\nОценка %c", bal);
}
```

Пример 1.25.
Вычислить

$$h = \frac{x^{y+1} + e^{y-1}}{1 + x * |y - tgz|} * (1 + |y - x|) + \frac{|y - x|^2}{2} - \frac{|y - x|^3}{3},$$

при $x = 2{,}444$, $y = 0{,}00869$, $z = -130$, дожно быть получено –0,49871.

Текст программы может иметь нижепредставленный вид:

```
#include <stdio.h>
#include <conio.h>
#include <math.h>
#define x 2.444
#define y 0.00869
#define z –130.0
void main(void)
{
   double rezult, dop, a, b, c;
   clrscr( );                    /* ОЧИСТКА ЭКРАНА */
   dop=fabs(y–x);
   a=pow(x, y+1)+exp(y–1);
   b=1+x*fabs(y–tan(z));
```

```
    c=0.5*pow(dop, 2)–pow(dop, 3)/3;
    rezult=a/b*(1+dop)+c;
                printf("\a\n OTBET:  rezult=%lf, Press any key...",
                    rezult);
    getch( );  /* ЗАДЕРЖКА ДО НАЖАТИЯ ЛЮБОЙ КЛАВИШИ */
}
```

Варианты заданий

Номер варианта	Выражение	Исходные данные
1	$a = \ln(y^{-\sqrt{\lvert x\rvert}}) \cdot \left(\sin(x) + e^{(x+y)}\right)$	*x, y*
2	$b = \sqrt{c(\sqrt{y} + x^2)} \cdot (\cos(x) - \lvert c - y\rvert)$	*c, x, y*
3	$c = \operatorname{arctg}(x) - \frac{3}{5}e^{xy} + 0{,}5\frac{\lvert x+y\rvert}{(x+y)^b}$	*b, x, y*
4	$d = \frac{e^{\lvert x-y\rvert}\operatorname{tg}(z)}{\operatorname{arctg}(y) + \sqrt{x}} + \ln(x)$	*x, y, z*
5	$e = \frac{(\cos(x) - \sin(y))^3}{\sqrt{\operatorname{tg}(z)}} + \ln^2(xyz)$	*x, y, z*
6	$f = y^x + \sqrt{\lvert x\rvert + e^y} - \frac{z^3 \sin^2(y)}{y + z^2/(y-x)}$	*x, y, z*
7	$g = \frac{1 + \cos(x+y)}{\lvert e^x - 2y/(1+x^2y^2)\rvert}x^3 + \arcsin(y)$	*x, y*
8	$h = 2 + \frac{x^2}{\sqrt{2}} + \frac{\lvert y^3\rvert}{\sqrt{2}} + \frac{z^4(\ln(x)+1)\cdot\sqrt{2}}{\sqrt{3}}$	*x, y, z*
9	$j = \left((1+y)\sqrt{\sin^2(z)} - \frac{\lvert y-x\rvert}{5}\right)^3$	*x, y, z*
10	$k = \ln\left\lvert(y - \sqrt{\lvert x\rvert}\right\rvert(x - \frac{y}{z + x^2/4})$	*x, y, z*

Номер варианта	Выражение	Исходные данные
11	$l = 0,5x^2 + 3\cos(x+y) + e^{-0,1yz} - \sqrt{\lvert xy \rvert}$	*x, y, z*
12	$m = \sqrt{e^x + \operatorname{tg}(x) + 1} \cdot (\lg(y) + \cos(xy) + \sqrt[3]{x})$	*x, y*
13	$p = \dfrac{\lg(x) - e^{x+y}}{\sqrt{2} + y^2 + \lvert x^3 - \ln(y) \rvert}$	*x, y*
14	$p = \dfrac{\lg(x) - e^{x+y}}{\sqrt{2} + y^2 + \lvert x^3 - \ln(y) \rvert}$	*x, y*
15	$q = \sqrt{12x^4 - 3x^2 + 4x^2 - 5x + 6} - \lg^2(z)$	*x, z*
16	$n = \dfrac{ax^2 + \sin^2 z}{\sqrt{1 + e^a}}$	*n, x, z*
17	$t_2 = \dfrac{b^2 + \sqrt{\lvert q \rvert}}{\cos^2 x + b \ln x}$	*b, z, x*
18	$q_3 = \dfrac{\sin^2 (z+a)^3}{t\sqrt[3]{e^{2+3t}}}$	*z, a, t*

Контрольные вопросы

1. Как описываются константы и переменные?
2. Назвать стандартные типы данных.
3. Описать процедуру стандартного ввода-вывода.
4. Назвать составные части программы на языке С++.
5. Изобразить схематически блок-схему линейного алгоритма.

Лабораторная работа № 2

ВЫЧИСЛЕНИЕ ПО УСЛОВИЮ

Цель:

– изучить оператор if при решении задач на языке С++;

– представить алгоритм решения задачи в виде блок-схемы с использованием структуры ветвления.

Теоретические сведения

Условный оператор if

Оператор if является наиболее простым из операторов ветвлений.

Общий вид записи:

if <условие> <оператор>;

Следующая программа (IFDEMO) иллюстрирует применение оператора if.

```
// ifdemo.cpp
// применение оператора if
#include <iostream>
using namespace std;
int main ()
{
int x;
cout « "Введите число:";
cin » x;
if (x>100 )
cout « "Это число больше, чем 100\n ";
 return 0;
}
```

За ключевым словом if следует условие ветвления, заключенное в круглые скобки.

Результат:

Введите число: 200
Это число больше, чем 100

Если вводимое число окажется не превосходящим 100, то программа завершится, не напечатав вторую строку.

Графическая интерпретация оператора

В блок-схемах короткому условному оператору соответствует структура **ЕСЛИ–ТО** (рис. 2.1).

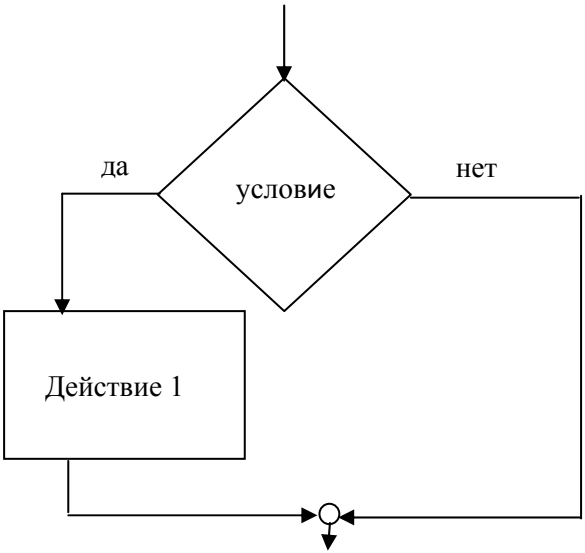


Рис. 2.1. Блок-схема для оператора **ЕСЛИ–ТО**

Отметим, что данная структура имеет один вход и один выход. Словесная формулировка: «Если условие истинно, то выполнять Действие 1».

Несколько операторов в теле if

Как и для циклов, тело ветвления if может состоять как из одного оператора, что было продемонстрировано в программе IFDEMO,

так и из нескольких операторов, заключенных в фигурные скобки. Пример, иллюстрирующий использование блока операторов в теле if, называется IF2 и приводится ниже.

Пример 2.1.

```
// if2.cpp
// использование нескольких операторов в теле цикла if
finclude <iostream>
using namespace std;
int main()
{
int x;
cout « "Введите число: ";
cin » x;
if( x>100 )
{
cout « "Число " « x;
cout « " больше, чем 100\n";
}
return 0;
}
```

Результат работы программы IF2:

```
Введите число: 12345
Число 12345 больше, чем 100
```

If внутри циклов

Циклы и ветвления можно использовать совместно. Можно помещать ветвления внутрь цикла и наоборот, использовать вложенные ветвления и вложенные циклы. В следующем примере под названием PRIME ветвление if находится внутри цикла for. Программа определяет, является ли вводимое число простым или нет (простым называется число, которое делится только на единицу и на само себя; примеры простых чисел: 2, 3, 5, 7, 11, 13 и 17).

Пример 2.2.

```
// prime.cpp
// применение цикла if для определения простых чисел
#include <iostream>
using namespace std;
#include <process.h>    //для exit()
int main()
{
unsigned long n, j;
cout « "Введите число:";
cin » n; // ввод проверяемого числа
for(j=2; j<=n/2; j++) // деление на целые числа.
if (n%j==0)// начиная с 2;
если остаток нулевой, то число не простое
{
cout « "Число не простое: делится на " « j « endl;
exit(0); // выход из программы
}
cout « "Число является простым\n";
return 0;
}
```

В этом примере вводим значение, которое присваивается переменной *n*. Затем программа при помощи цикла for делит число *n* на все числа от 2 до *n*/2. Делителем является переменная *j*, служащая счетчиком цикла. Если число *n* разделится без остатка на какое-либо из значений *j*, то оно не будет простым. Условием того, что одно число делится на другое без остатка, является равенство остатка от деления нулю. Поэтому в условии для if участвует операция остатка от деления %. Если число оказывается не простым, то выводим соответствующее сообщение и выходим из программы.

В этом примере тело цикла не заключено в фигурные скобки. Это объясняется тем, что оператор if и операторы тела ветвления на самом деле являются одним оператором. Чтобы улучшить читаемость кода, можно добавить фигурные скобки, но это не является обязательным для правильной работы компилятора.

Функция exit()

Когда программа PRIME получает число, не являющееся простым, она завершается, поскольку нет необходимости несколько раз проверять, является число простым или нет. Библиотечная функция exit() производит немедленный выход из программы независимо от того, в каком месте она находится. Эта функция не возвращает значения. Ее единственный аргумент (в нашем случае 0) возвращается вызывающему окружению после того, как программа завершается. Как правило, возвращаемое значение 0 говорит об успешном завершении программы; ненулевые значения сигнализируют об ошибках.

Оператор if...else

Общий вид записи:

```
if  <условие>   <оператор 1>;
        else
                <оператор 2>;
```

Графическая интерпретация оператора

В блок-схемах полному условному оператору соответствует структура **ЕСЛИ–ТО–ИНАЧЕ** (рис. 2.2).

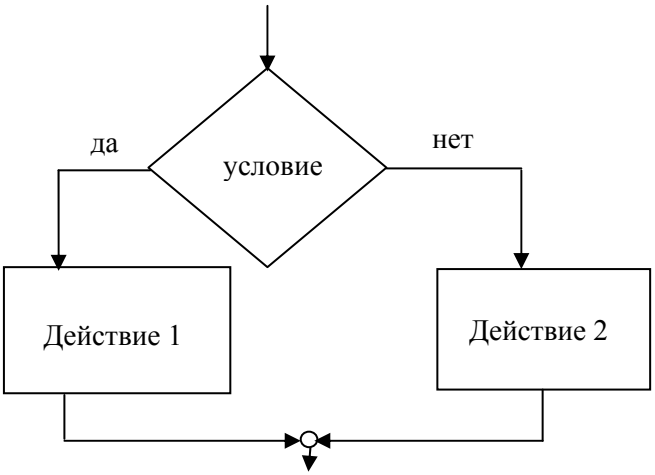


Рис. 2.2. Блок-схема для оператора **ЕСЛИ–ТО–ИНАЧЕ**

Отметим, что и данная структура имеет один вход и один выход. Словесная формулировка данной структуры: «Если значение выражения истинно, выполняется <действие 1>, если ложно – <действие 2>».

Замечание. Оператор 1 и оператор 2 входят в конструкцию полного условного оператора как единственные (*простые*). Если возникает необходимость выполнить в ветвях несколько операторов, то их заключают в операторные скобки. Такие операторы называются *составными*. Запись оператора if с составным оператором имеет нижепредставленный вид.

```
if  <условие>
      {
            <оператор 1>;
…….
            <оператор n>;
      }
         else
               {
            <оператор 1>;
…….
            <оператор m>;
    }
```

Вложенная структура условных операторов

Структура называется *вложенной*, если после if используются вновь условные операторы. Число вложений может быть произвольным. При этом справедливо: служебное слово ELSE всегда относится к ближайшему выше слову if.

```
If <условие>
If <условие> <оператор>;
                <оператор>;
```

Условная операция

Существует распространенная в программировании ситуация: переменной необходимо присвоить одно значение, в случае выполнения некоторого условия, и другое значение, в случае невыполнения этого условия. В следующем фрагменте переменной min присваивается наименьшее из значений alpha и beta с помощью конструкции if...else:

```
if (alpha < beta)
min = alpha;
else min = beta;
```

Подобные действия на практике оказались настолько распространенными, что была специально разработана *условная операция*, выполняющая эти действия. Она записывается с помощью двух знаков и использует три операнда. Является единственной операцией в C++, использующей более двух операндов. С помощью условной операции можно записать предыдущий фрагмент следующим образом:

```
min = (alpha < beta) ? alpha : beta;
```

Правая часть оператора представляет собой условное выражение:

```
(alpha < beta) ? alpha : beta // условное выражение
```

Знак вопроса ? и двоеточие : обозначают условную операцию. Условие, стоящее перед знаком вопроса (alpha < beta), является условием проверки. Это условие, вместе с переменными alpha и beta, составляет тройку операндов условной операции.

Если значение проверяемого условия истинно, то условное выражение становится равным значению alpha; в противном случае, оно становится равным beta (рис. 2.3).

Рис. 2.3. Исполнение условного оператора

Операторы перехода

В языке C++ существует несколько операторов перехода: break, continue, goto.

Оператор break производит выход из цикла. Следующим оператором, исполняемым после break, будет являться первый оператор, находящийся вне данного цикла (рис. 2.4).

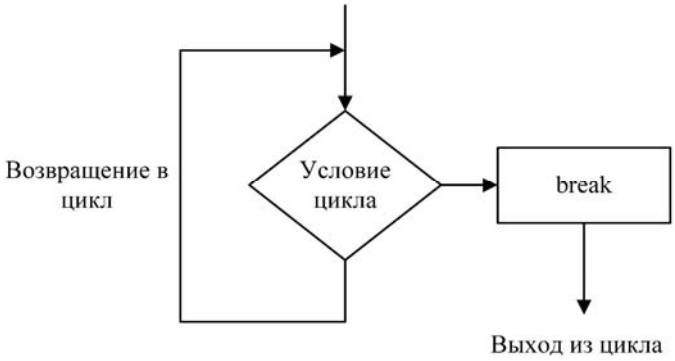


Рис. 2.4. Исполнение оператора break

Оператор break производит выход из цикла. Тем не менее, могут возникнуть и такие ситуации, когда необходимо при определенном условии не выходить из цикла, а досрочно возвращаться в его начало. Именно таким эффектом обладает применение оператора continue (строго говоря, continue делает переход на завершающую фигурную скобку цикла, откуда производится обычный переход в начало тела цикла).

Синтаксис оператора goto следующий: вставить метку перед тем оператором программы, на который намечается сделать переход. После метки всегда ставится двоеточие. Имя метки, перед которой расположено ключевое слово goto, совершит переход на тот оператор, который помечен данной меткой. Следующий фрагмент кода иллюстрирует применение goto:

```
goto 1: //
операторы
1:// сюда передается управление оператором goto
```

Варианты заданий

Номер варианта	Выражение	Исходные данные
1	$a=\begin{cases}(x+y)^2-\sqrt{xy}, & xy>0\\ (x+y)^2+\sqrt{\lvert xy\rvert}, & xy<0\\ (x+y)^2+1, & xy=0\end{cases}$	*x, y*
2	$b=\begin{cases}\ln(x/y)+(x^2+y)^3, & x/y>0\\ \ln\lvert x/y\rvert+(x^2+y)^3, & x/y<0\\ (x^2+y), & x=0\\ 0, & y=0\end{cases}$	*x, y*
3	$c=\begin{cases}x^2+y^2+\sin(x), & x-y=0\\ (x-y)^2+\cos(x), & x-y>0\\ (y-x)^2+\operatorname{tg}(x), & x-y<0\end{cases}$	*x, y*

Номер варианта	Выражение	Исходные данные
4	$d=\begin{cases}(x-y)^3+\operatorname{arctg}(x), & x>y\\ (y-x)^3+\operatorname{arctg}(x), & y>x\\ (y+x)^3+0,5, & y=x\end{cases}$	*x, y*
5	$e=\begin{cases}i\sqrt{a}, & i-\text{нечетное},\ a>0\\ \frac{i}{2}\sqrt{\lvert a\rvert}, & i-\text{четное},\ a<0\\ \sqrt{\lvert ia\rvert}, & \text{иначе}\end{cases}$	*i, a*
6	$g=\begin{cases}e^{\lvert a\rvert-\lvert b\rvert}, & 0,5<ab<10\\ \sqrt{\lvert a+b\rvert}, & 0,1<ab<0,5\\ 2x^2, & \text{иначе}\end{cases}$	*a, b, x*
7	$h=\begin{cases}\operatorname{arctg}(x+\lvert y\rvert), & x<y\\ \operatorname{arctg}(\lvert x\rvert+y), & x>y\\ (x+y)^2, & x=y\end{cases}$	*x, y*
8	$j=\begin{cases}\sin(5k+3m\lvert k\rvert), & -1<k<m\\ \cos(5k+3m\lvert k\rvert, & k>m\\ k^3, & k=m\end{cases}$	*k, m*
9	$l=\begin{cases}3k^3+3p, & k>\lvert p\rvert\\ \lvert k-p\rvert, & 3<k<\lvert p\rvert\\ (k-p)^2, & k=\lvert p\rvert\end{cases}$	*k, p*
10	$k=\begin{cases}\ln(\lvert f\rvert+\lvert q\rvert), & \lvert fq\rvert>10\\ e^{f+q}, & \lvert fq\rvert<10\\ f+q, & \lvert fq\rvert=10\end{cases}$	*f, q*

Номер варианта	Выражение	Исходные данные
11	$y=\begin{cases} ax^3+b\ln\lvert 2x\rvert, & \sqrt{\lvert a-b\rvert}<x \\ \sqrt{\lvert a+\sin 2x\rvert}-b^{3x}, & \sqrt{\lvert a-b\rvert}=x \\ \dfrac{\operatorname{arctg}5x}{b\cos x+\lg ax}, & \sqrt{\lvert a-b\rvert}>x \end{cases}$	*a, b, x*
12	$y=\begin{cases} \ln x^2-e^{ax+b}+\lg\lvert a-b\rvert, & 2a+b<0 \\ \operatorname{tg}(ax-b^2)-b\lvert x^{-3}\rvert, & 2a+b=0 \\ \operatorname{arctg}(2x-0{,}5)+\sqrt{\lvert a+bx\rvert}, & 2a+b>0 \end{cases}$	*a, b, x*
13	$y=\begin{cases} \dfrac{ax-e^x+b^3}{\ln(2x^2+4)}+\cos(4x-0{,}2), & \lvert a^2-b^2\rvert<10x \\ b\sin(x^3-a)-e^{-1,4x}, & \lvert a^2-b^2\rvert=10x \\ \operatorname{tg}4x+\dfrac{\lvert x^{-5}\rvert}{\sin 0{,}5x}, & \lvert a^2-b^2\rvert>10x \end{cases}$	*a, b, x*
14	$y=\begin{cases} a\sqrt{\lvert\sin x+\cos^2 x\rvert}+e^{a+bx}, & \sqrt{\lvert bx-65\rvert}<a \\ 1-\ln\sqrt{\lvert ax^3-b\rvert}-\lvert x^{-10}\rvert, & \sqrt{\lvert bx-65\rvert}=a \\ \dfrac{(x^3-b)\cdot\cos(3x-0{,}5)}{\lg x^3-a\sin(a-b)}, & \sqrt{\lvert bx-65\rvert}>a \end{cases}$	*a, b, x*
15	$y=\begin{cases} \ln\left\lvert\dfrac{x^{-9}}{ax-b}\right\rvert-e^{2x^2}, & \sin x<\sqrt{a^2+b^2} \\ 4\operatorname{arctg}6x+\dfrac{ax+7}{\sqrt{\lvert b^2-x\rvert}}, & \sin x=\sqrt{a^2+b^2} \\ x^3+2{,}3ax+\sqrt{b^2\lvert\operatorname{tg}x\rvert}, & \sin x>\sqrt{a^2+b^2} \end{cases}$	*a, b, x*

Номер варианта	Выражение	Исходные данные
16	$y=\begin{cases}\sqrt{\sin x^3}+\ln^2 z, & \text{при } z<\sqrt[3]{ax}\\ e^{bz}+\lvert x^3\rvert, & \text{при } z=\sqrt[3]{ax}\\ \cos\sqrt{z}+\lg b^2 x, & \text{при } z>\sqrt[3]{ax}\end{cases}$	*x, z, a, b*
17	$b_2=\begin{cases}3q+\sqrt{\lvert x^3+\sin z\rvert}, & \text{при } \sin z<q\\ \sqrt[5]{a_3+q^3+e^{az}}, & \text{при } \sin z=q\\ \lvert a^5\rvert\ln\sqrt{\cos z}, & \text{при } \sin z>q\end{cases}$	*q, x, z, a*
18	$c=\begin{cases}a^3+\sqrt[5]{\cos y^2}, & \text{при } y<\ln b\\ \lg(x+w)^2+\lvert y^3\rvert, & \text{при } y=\ln b\\ \sqrt{\mathrm{tg}y^5}+e^{xz}, & \text{при } y>\ln b\end{cases}$	*a, y, x, z, b*

Контрольные вопросы

1. Формат записи оператора if.
2. Формат записи с вложенным оператором if.
3. Как работает оператор if?
4. В чем отличие короткой и полной формы записи оператора if?
5. Изобразить схематически блок-схему разветвляющего алгоритма.

Лабораторная работа № 3

ОПЕРАТОР SWITCH

Цель:

– изучить оператор switch при решении задач на языке C++;

– представить алгоритм решения задачи в виде блок-схемы с использованием структуры ветвления.

Теоретические сведения

В языке C++ предусмотрен оператор многовариантного ветвления switch, который последовательно сравнивает значение выражения со списком целых чисел или символьных констант. Если обнаруживается совпадение, выполняется оператор, связанный с соответствующей константой. Оператор switch имеет нижепредставленный вид.

```
switch (выражение)
{
case константа 1:
последовательность операторов
break;
case константа 2:
последовательность операторов
break;
case константа3:
последовательность операторов
break;
…
default:
последовательность операторов
}
```

Значением *выражения* должен быть символ или целое число. Например, выражения, результатом которых является число с плавающей точкой, не допускаются. Значение выражения последовательно сравнивается с константами, указанными в операторах case. Если обнаруживается совпадение, выполняется последовательность

операторов, связанных с данным оператором case, пока не встретится оператор break или не будет достигнут конец оператора switch. Если значение выражения не совпадает ни с одной из констант, выполняется оператор default. Этот раздел оператора switch является необязательным. Если он не предусмотрен, в отсутствие совпадений не будет выполнен ни один оператор. На практике количество разделов case в операторе switch следует ограничивать, поскольку оно влияет на эффективность программы. Оператор case используется только внутри switch.

Оператор break относится к группе операторов перехода. Его можно использовать как в операторе switch, так и в циклах. Когда поток управления достигает оператора break, программа выполняет переход к оператору, следующему за оператором switch.

Следует знать три важных свойства оператора switch.

1. Оператор switch отличается от оператора if тем, что значение его выражения сравнивается исключительно с константами, в то время как в операторе if можно выполнять какие угодно сравнения или вычислять любые логические выражения.

2. Две константы в разных разделах case не могут иметь одинаковых значений, за исключением случая, когда один оператор switch вложен в другой.

3. Если в операторе switch используются символьные константы, они автоматически преобразовываются в целочисленные.

Оператор switch часто используется для обработки команд, введенных с клавиатуры, например, при выборе пунктов меню.

С формальной точки зрения наличие оператора break внутри оператора switch не обязательно. Этот оператор прерывает выполнение последовательности операторов, связанных с соответствующей константой. Если его пропустить, будут выполнены все последующие операторы case, пока не встретится следующий оператор break либо не будет достигнут конец оператора switch.

Графическая интерпретация оператора Switch

В блок-схемах оператору switch соответствует структура ВЫБОР (рис. 3.1, 3.2).

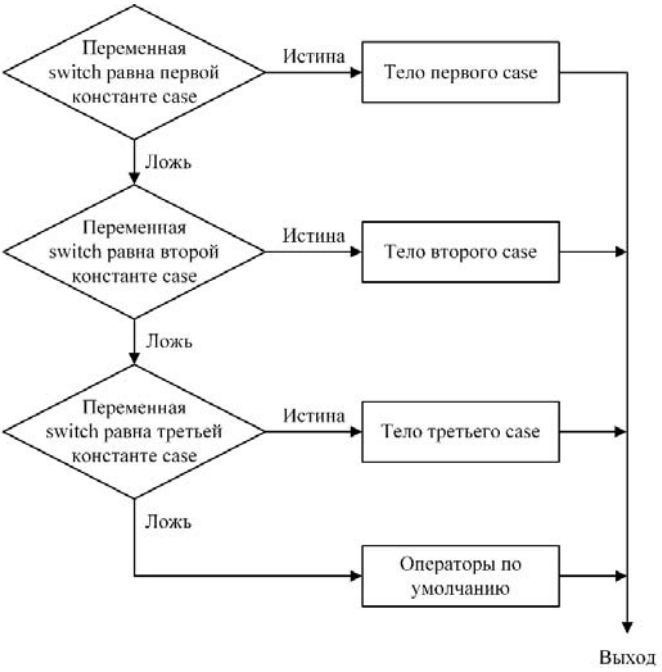


Рис. 3.1. Исполнение конструкции switch

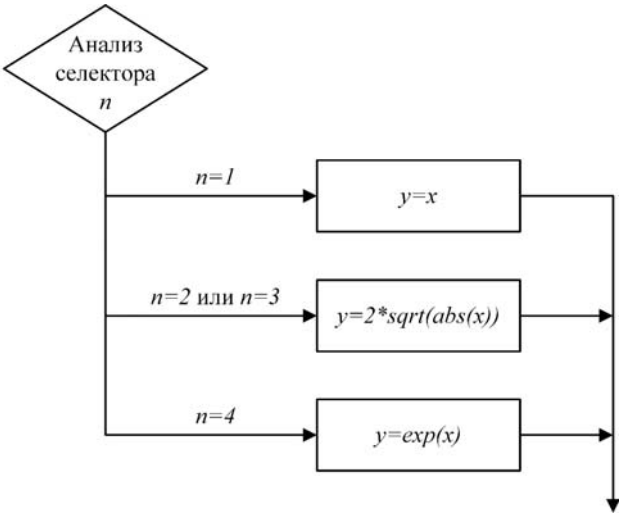


Рис. 3.2. Пример конструкции switch

Варианты заданий

Номер варианта	Выражение	Исходные данные
1	$a=\begin{cases}(x+y)^2-\sqrt{xy}, & k=1\\ (x+y)^2+\sqrt{\lvert xy\rvert}, & k=12\\ (x+y)^2+1, & k=100\end{cases}$	*x, y, k*
2	$b=\begin{cases}\ln(x/y)+(x^2+y)^3, & a=1..5\\ \ln\lvert x/y\rvert+(x^2+y)^3, & a=6..8\\ (x^2+y), & a=9..12\\ 0, & \text{иначе}\end{cases}$	*x, y, a*
3	$c=\begin{cases}x^2+y^2+\sin(x), & b=10,\ 12,\ 23\\ (x-y)^2+\cos(x), & b=14,\ 18,\ 24\\ (y-x)^2+\operatorname{tg}(x), & b=100,\ 120,\ 235\end{cases}$	*x, y, b*
4	$d=\begin{cases}(x-y)^3+\operatorname{arctg}(x), & z=1..10\\ (y-x)^3+\operatorname{arctg}(x), & z=12..34\\ (y+x)^3+0,5, & \text{иначе}\end{cases}$	*x, y, z*
5	$e=\begin{cases}i\sqrt{a}, & l=1,2,3\\ \dfrac{i}{2}\sqrt{\lvert a\rvert}, & l=5,6,7\\ \sqrt{\lvert ia\rvert}, & \text{иначе}\end{cases}$	*i, a, l*
6	$g=\begin{cases}e^{\lvert a\rvert-\lvert b\rvert}, & s=1\\ \sqrt{\lvert a+b\rvert}, & s=12\\ 2x^2, & \text{иначе}\end{cases}$	*a, b, x, s*

Номер варианта	Выражение	Исходные данные
7	$h=\begin{cases}\operatorname{arctg}(x+\lvert y\rvert), & n=1\\ \operatorname{arctg}(\lvert x\rvert+y), & n=12,\ 23,\ 56\\ (x+y)^2, & \text{иначе}\end{cases}$	*x, y, n*
8	$j=\begin{cases}\sin(5k+3m\lvert k\rvert), & a=5..10\\ \cos(5k+3m\lvert k\rvert, & a=11..20\\ k^3, & \text{иначе}\end{cases}$	*k, m, a*
9	$l=\begin{cases}3k^3+3p, & b=12\\ \lvert k-p\rvert, & b=34\\ (k-p)^2, & b=89\end{cases}$	*k, p, b*
10	$k=\begin{cases}\ln(\lvert f\rvert+\lvert q\rvert), & h=1..23\\ e^{f+q}, & h=34..89\\ f+q, & h=90..123\end{cases}$	*f, q, h*
11	$y=\begin{cases}ax^3+b\ln\lvert 2x\rvert, & m=12..15\\ \sqrt{\lvert a+\sin 2x\rvert}-b^{3x}, & m=46..50\\ \dfrac{\operatorname{arctg}5x}{b\cos x+\lg ax}, & \text{иначе}\end{cases}$	*a, b, x, m*
12	$y=\begin{cases}\ln x^2-e^{ax+b}+\lg\lvert a-b\rvert, & t=123..126\\ \operatorname{tg}(ax-b^2)-b\lvert x^{-3}\rvert, & t=236..238\\ \operatorname{arctg}(2x-0{,}5)+\sqrt{\lvert a+bx\rvert}, & t=1000\end{cases}$	*a, b, x, t*

Номер варианта	Выражение	Исходные данные
13	$y=\begin{cases}\dfrac{ax-e^{x}+b^{3}}{\ln(2x^{2}+4)}+\cos(4x-0,2), & z=1\\ b\sin(x^{3}-a)-e^{-1,4x}, & z=56\\ \operatorname{tg}4x+\dfrac{\lvert x^{-5}\rvert}{\sin 0,5x}, & z=123\end{cases}$	*a, b, x, z*
14	$z_1=\begin{cases}a\sqrt{\lvert\sin x+\cos^{2}x\rvert}+e^{a+bx}, & y=12,45,78\\ 1-\ln\sqrt{\lvert ax^{3}-b\rvert-\lvert x^{-10}\rvert}, & y=456\\ \dfrac{(x^{3}-b)\cdot\cos(3x-0,5)}{\lg x^{3}-a\sin(a-b)}, & \text{иначе}\end{cases}$	*a, b, x, y*
15	$d=\begin{cases}\sqrt{\sin x^{3}}+\ln^{2}z, & a=12..34\\ e^{bz}+\lvert x^{3}\rvert, & a=35..40\\ \cos\sqrt{z}+\lg b^{2}x, & \text{иначе}\end{cases}$	*x, z, a, b*
16	$y=\begin{cases}\ln\left\lvert\dfrac{x^{-9}}{ax-b}\right\rvert-e^{2x^{2}}, & s=34..40\\ \operatorname{arctg}6x+\dfrac{ax+7}{\sqrt{\lvert b^{2}-x\rvert}}, & s=57..60\\ x^{3}+2,3ax+\sqrt{b^{2}\lvert\operatorname{tg}x\rvert}, & \text{иначе}\end{cases}$	*a, b, x, s*

Номер варианта	Выражение	Исходные данные
17	$d=\begin{cases}3q+\sqrt{\lvert x^3+\sin z\rvert}, & b=2..5\\ \sqrt[5]{a+q^3+e^{az}}, & b=6..9\\ \lvert a^5\rvert\ln\sqrt{\cos z}, & \text{иначе}\end{cases}$	*q, x, z, a, b*
18	$l=\begin{cases}a^3+\sqrt[5]{\cos y^2}, & d=1\\ \lg(x+z)^2+\lvert y^3\rvert, & d=12,45,89\\ \sqrt{\mathrm{tg}y^5}+e^{xz}, & \text{иначе}\end{cases}$	*a, y, x, z, d*

Контрольные вопросы

1. Формат записи оператора switch.
2. Как работает оператор switch?
2. В чем отличие оператора switch от if?
3. Типы селектора.

Лабораторная работа № 4

ЦИКЛИЧЕСКИЕ ВЫЧИСЛИТЕЛЬНЫЕ ПРОЦЕССЫ

Цель работы: изучить виды операторов цикла при решении задач на языке C++.

Теоретические сведения

В языке C++, как и во всех других современных языках программирования, операторы цикла предназначены для выполнения повторяющихся инструкций пока действует определенное правило. Это условие может быть как задано заранее (в цикле for), так и меняться во время выполнения цикла (в операторах while и do-while).

Цикл for

В том или ином виде цикл for есть во всех процедурных языках программирования. Однако в языке C++ он обеспечивает особенно высокую гибкость и эффективность.

Общий вид оператора for таков:

for (инициализация; условие; приращение) тело

Цикл for имеет много вариантов. Однако наиболее общая форма этого оператора работает следующим образом. Сначала выполняется *инициализация* – оператор присваивания, который задает начальное значение счетчика цикла. Перед каждым шагом проверяется *условие*, представляющее собой условное выражение. Цикл выполняется до тех пор, пока значение этого выражения остается истинным. *Приращение* изменяет значение счетчика цикла при очередном его выполнении. Эти разделы оператора отделяются друг от друга точкой с запятой. Как только условие цикла станет ложным, программа прекратит его выполнение и перейдет к следующему оператору.

В следующем примере цикл for выводит на экран числа от 1 до 100.

Пример 4.1.

```
#include <stdio.h>
void main()
{
int x;                          // определение счетчика цикла
for (x=1; x<=100; x++) // счетчик меняется от 1 до 100
printf("%d", x);   // значение x выводится на экран
}
```

Сначала переменной *x* присваивается число 1, а затем она сравнивается с числом 100. Поскольку ее значение меньше либо равно 100, вызывается функция printf(). Затем переменная *x* увеличивается на единицу, и условие цикла проверяется вновь. Как только ее значение превысит число 100, выполнение цикла прекратится. В данном случае переменная *x* является счетчиком цикла, который изменяется и проверяется на каждой итерации.

Рассмотрим пример цикла for, тело которого состоит из нескольких операторов.

Пример 4.2.

```
for (x=100; x!=65; x-=5) {
z=x*x;
printf("Квадрат числа %d равен %f\n", x, z);
}
```

Возведение числа *x* в квадрат и вызов функции printf() выполняются до тех пор, пока значение переменной *x* не станет равным 65. Обратите внимание на то, что в этом цикле счетчик уменьшается: сначала ему присваивается число 100, а затем на каждой итерации из него вычитается число 5.

В цикле for проверка условия выполняется перед каждой итерацией. Иными словами, если условие цикла с самого начала является ложным, его тело не будет выполнено ни разу.

Рассмотрим пример.

Пример 4.3.

```
x=10;
for (y=10; y!=x; ++y) {
printf("%d", y);
}

printf("%d", y);          /*Это единственный вызов функции printf(), который выполняется в данном фрагменте */
```

Этот цикл никогда не будет выполнен, поскольку значения переменных *x* и *y* при входе в цикл равны. Следовательно, условие цикла является ложным, и ни тело цикла, ни приращение счетчика выполняться не будут. Таким образом, значение переменной *y* останется равным 10, и именно оно будет выведено на экран.

Пример 4.4. Табуляции функции sin(*x*) на интервале [0; 1].

```
#include <stdio.h>
#include <math.h>
const int N=10;
void main()
{
double a=0, b=1, h=(b–a)/N, x;
int i;
printf(" X Y\n");
for (i=0; i<=N; i++)
{
x=a+i*h;
printf("%4.2f%9.5f\n", x, sin(x));
}
}
```

Варианты цикла for

Оператор for имеет несколько вариантов, повышающих его гибкость и эффективность.

Наиболее распространенным является вариант, в котором используется оператор последовательного выполнения («запятая»), что

позволяет применять несколько счетчиков цикла одновременно. Напомним, что оператор последовательного выполнения связывает между собой несколько операторов, вынуждая их выполняться друг за другом. Например, переменные *x* и *y* являются счетчиками приведенного ниже цикла. Их инициализация выполняется в одном и том же разделе цикла.

Пример 4.5.

```
for(x=0, y=0; x+y<10; ++x) {
y=getchar();
y= y–'0'; /* Вычесть из переменной y
             ASCII-код нуля */
}
```

Как видим, два оператора инициализации разделены запятой. При каждой итерации значение переменной *x* увеличивается на единицу, а переменная *y* вводится с клавиатуры. Несмотря на это, переменная *y* должна иметь какое-то начальное значение, иначе перед первой итерацией цикла условие может оказаться ложным.

Условное выражение не обязательно связано с проверкой счетчика цикла. В качестве условия цикла может использоваться любой допустимый оператор сравнения или логический оператор. Это позволяет задавать несколько условий цикла одновременно.

Рассмотрим программу, которая ждет от пользователя нажатия клавиши <Esc>. Пользователю дается пять попыток.

Пример 4.6.

```
#include <stdio.h>
void main()
{
char c;
int num;
printf("Нажмите клавишу Esc\n");
for (num=1, c=0; num<= 5 && c !=27; num++)
{
printf("\tпопытка %d: ", num);
```

```
c=getch();
printf("\n");
}
if (c==27)
printf("Вы нажали Esc c %d попытки!\n",--num);
else printf("Вы так и не нажали Esc!\n");
}
```

Каждый из трех разделов цикла for может состоять из любых допустимых выражений. Эти выражения могут быть никак не связаны с предназначением разделов. Учитывая вышесказанное, рассмотрим пример.

Пример 4.7.

```
#include <stdio.h>
void main()
{
int t, S=0;
for (printf("Введите число: "); scanf("%d",&t), t;
      printf("\t еще: "))
S+=t;
printf("Сумма чисел равна %d\n", S);
}
```

Обратите внимание на цикл for в функции. В разделе инициализации выводится сообщение, предлагающее пользователю ввести число. В разделе условия программа ждет ввод числа и возвращает это число. Таким образом, условием завершения цикла будет ввод числа 0, которое интерпретируется компилятором как ложь. В разделе приращения счетчика выводится приглашение к вводу следующего числа. В теле цикла вычисляется сумма введенных чисел.

Другая интересная особенность цикла for заключается в том, что его разделы можно пропускать. Каждый из его разделов является необязательным. Например, цикл, приведенный ниже, выполняется до тех пор, пока пользователь не введет число 123.

```
for (x=0; x!=123;)
scanf("%d", &x);
```

Обратите внимание на то, что раздел приращения счетчика в данном цикле for отсутствует. Это значит, что при каждой итерации значение переменной *x* сравнивается с числом 123 и никакие действия с ней больше не выполняются. Однако, если пользователь введет с клавиатуры число 123, условие цикла станет ложным, и программа прекратит его выполнение.

Счетчик можно инициализировать вне цикла for. Этим способом пользуются, когда начальное значение счетчика является результатом сложных вычислений, как в примере.

Пример 4.8.

```
ch=getch (); /* Считать символ */
if (ch>='0' && ch<='9') x=ch–'0';
else x=10;
for (; x<10;)
{
printf ("%d", x);
++x;
}
```

Здесь раздел инициализации оставлен пустым, а переменная *x* инициализируется до входа в цикл.

Бесконечный цикл

Хотя в качестве бесконечного можно использовать любой цикл, традиционно для этой цели применяется оператор for. Поскольку все разделы оператора for являются необязательными, его легко сделать бесконечным, не задав никакого условного выражения.

```
for( ; ; )
printf ("Этот цикл выполняется бесконечно.\n");
```

Если условное выражение не указано, оно считается истинным. Разумеется, в этом случае можно по-прежнему выполнять инициализацию и приращение счетчика, однако программисты на языке C++ в качестве бесконечного цикла чаще всего используют конструкцию for (; ;).

На самом деле конструкция for(; ;) не гарантирует бесконечное выполнение цикла, поскольку его тело может содержать оператор break, приводящий к немедленному выходу. В этом случае программа передаст управление следующему оператору, находящемуся за пределами тела цикла for, как показано ниже.

Пример 4.9.

```
#include <stdio.h>
#include <conio.h>
void main()
{
double a=6, b=3;
char ch;
int sign;
for ( ; ; )
{
printf("Введите операцию (+ - * /): ");
ch=getch();
sign=ch=='+' || ch=='-' || ch=='*' ||
ch=='/';
if (ch==27) break;
if (sign) {
printf("%f %c %f = ", a, ch, b);
switch (ch)
{
case '+' : printf("%f", a+b); break;
case '-' : printf("%f", a–b); break;
case '*' : printf("%f", a*b); break;
case '/' : printf("%f", a/b); break;
}
}
printf("\n");
}
```

Цикл while

В языке C++ существует два вида цикла while:
– цикл while с предусловием;
– цикл while с постусловием.

Цикл while с предусловием

Цикл while с предусловием позволяет выполнить одну и ту же последовательность действий пока проверяемое условие истинно. При этом условие записывается до тела цикла и проверяется до выполнения тела цикла.

При выполнении цикла while сначала проверяется условие. Если оно ложно, то цикл не выполняется и управление передается на следующую инструкцию после тела цикла while. Если условие истинно, то выполняется инструкция, после чего условие проверяется снова и снова выполняется инструкция. Так продолжается до тех пор, пока условие будет истинно. Как только условие станет ложно, работа цикла завершится и управление передастся следующей инструкции после цикла.

Синтаксис цикла while с предусловием.

```
    while (условие)
{
   блок инструкций
}
```

Пример 4.10. Следующий фрагмент программы напечатает на экран квадраты всех целых чисел от 1 до 10:

```
    int i=1;
while (i<=10)
{
   cout<<i*i<<endl;
   ++i;
}
```

В этом примере переменная *i* внутри цикла изменяется от 1 до 10. Такая переменная, значение которой меняется с каждым новым проходом цикла, называется *счетчиком*. Заметим, что после выполнения этого фрагмента значение переменной *i* будет равно 11, поскольку именно при *i* == 11 условие *i* <= 10 впервые перестанет выполняться.

В следующем примере цикл используется для того, чтобы найти количество знаков в десятичной записи целочисленной переменной *i*.

Пример 4.11.

```
    int Ndigits=0;
while(n!=0)
{
   Ndigits=Ndigits+1;
   n=n/10;
}
```

Внутри цикла значение переменной *n* уменьшается в 10 раз до тех пор, пока она не станет равна 0. Уменьшение целочисленной переменной в 10 раз (с использованием целочисленного деления) эквивалентно отбрасыванию последней цифры этой переменной.

Цикл while с постусловием

Цикл «пока» с постусловием отличается от цикла с предусловием тем, что сначала выполняется блок цикла, а потом проверяется условие. Если условие истинно, то цикл будет выполнен еще раз, и так до тех пор, пока условие будет истинно. Синтаксис цикла с постусловием представлен ниже (обратите внимание на обязательную точку с запятой после условия).

```
    do
{
   Блок инструкций
}
while (условие);
```

Поскольку условие проверяется после выполнения тела цикла, то блок цикла с постусловием всегда будет выполнен хотя бы один раз, независимо от истинности условия. Это может привести к ошибкам, поэтому использовать цикл while с постусловием следует только тогда, когда это действительно упрощает алгоритм.

Пример 4.12. Программа табуляции функции sin(*x*) на интервале [0; 1].

```
#include <stdio.h>
#include <math.h>
const int N=10;
void main()
{
double a=0, b=1, h=(b–a)/N, x;
int i = 0;
printf(" X Y\n");
while (i<=N)
{
x=a+i*h;
printf("%4.2f%9.5f\n", x, sin(x));
i++;
}
}
```

Варианты заданий

1. Составить таблицу значений функции при $x_н < x < x_к$ с шагом *h* вещественного типа.

2. Выполнить задания двумя способами:

– с использованием оператора for;

– с использованием оператора while.

Номер варианта	Выражение	Исходные данные
1	$a = \ln(y^{-\sqrt{\lvert x\rvert}})\left(\sin(x) + e^{(x+y)}\right)$	$x_н < x < x_к$, y = const
2	$b = \sqrt{c(\sqrt{y} + x^2)}(\cos(x) - \lvert c - y\rvert)$	$x_н < x < x_к$, y, c = const
3	$c = \operatorname{arctg}(x) - \frac{3}{5}e^{xy} + 0,5\frac{\lvert x+y\rvert}{(x+y)^b}$	$x_н < x < x_к$, y, b = const

Номер варианта	Выражение	Исходные данные
4	$d = \dfrac{e^{\lvert x-y\rvert}\operatorname{tg}(z)}{\operatorname{arctg}(y)+\sqrt{x}} + \ln(x)$	$x_н < x < x_к$, y, z = const
5	$e = \dfrac{(\cos(x)-\sin(y))^3}{\sqrt{\operatorname{tg}(z)}} + \ln^2(xyz)$	$x_н < x < x_к$, y, z = const
6	$f = y^x + \sqrt{\lvert x\rvert + e^y} - \dfrac{z^3 \sin^2(y)}{y + z^2/(y-x)}$	$x_н < x < x_к$, y, z = const
7	$g = \dfrac{1+\cos(x+y)}{\lvert e^x - 2y/(1+x^2y^2)\rvert} x^3 + \arcsin(y)$	$x_н < x < x_к$, y = const
8	$h = 2 + \dfrac{x^2}{\sqrt{2}} + \dfrac{\lvert y^3\rvert}{\sqrt{2}} + \dfrac{z^4(\ln(x)+1)\sqrt{2}}{\sqrt{3}}$	$x_н < x < x_к$, y, z = const
9	$j = \left((1+y)\sqrt{\sin^2(z)} - \dfrac{\lvert y-x\rvert}{5} \right)^3$	$x_н < x < x_к$, y, z = const
10	$k = \ln\left\lvert (y - \sqrt{\lvert x\rvert}\,\right\rvert (x - \dfrac{y}{z + x^2/4})$	$x_н < x < x_к$, y, z = const
11	$l = 0,5x^2 + 3\cos(x+y) + e^{-0,1yz} - \sqrt{\lvert xy\rvert}$	$x_н < x < x_к$, y, z = const
12	$m = \sqrt{e^x + \operatorname{tg}(x) + 1}(\lg(y) + \cos(xy) + \sqrt[3]{x})$	$x_н < x < x_к$, y = const
13	$p = \dfrac{\lg(x) - e^{x+y}}{\sqrt{2} + y^2 + \lvert x^3 - \ln(y)\rvert}$	$x_н < x < x_к$, y = const
14	$p = \dfrac{\lg(x) - e^{x+y}}{\sqrt{2} + y^2 + \lvert x^3 - \ln(y)\rvert}$	$x_н < x < x_к$, y = const
15	$q = \sqrt{12x^4 - 3x^2 + 4x^2 - 5x + 6} - \lg^2(z)$	$x_н < x < x_к$, z = const

Номер варианта	Выражение	Исходные данные
16	$l = \dfrac{ax^2 + \sin^2 z}{\sqrt{1 + e^a}}$	$x_н < x < x_к$, a, z = const
17	$t = \dfrac{b^2 + \sqrt{\lvert q \rvert}}{\cos^2 x + b \ln x}$	$x_н < x < x_к$, b, z = const
18	$q = \dfrac{\sin^2 (z + a)^3}{t\sqrt[3]{e^{2+3t}}}$	$t_н < t < t_к$, a, z = const

Контрольные вопросы

1. Формат записи оператора for.
2. Формат записи оператора while.
3. Составить блок-схему циклического алгоритма для всех операторов цикла.

Лабораторная работа № 5

ВЫЧИСЛЕНИЕ КОНЕЧНЫХ СУММ

Цель работы:

– изучить вложенные структуры операторов повтора (циклов) при вычислении конечных сумм;

– представить алгоритм решения задачи в виде блок-схемы.

Теоретические сведения

Оператор цикла часто применяется для суммирования значений некоторой последовательности чисел или значений функции при известном числе операций суммирования. Напомним некоторые определения, связанные с расчетом суммы последовательности.

Сумма членов последовательности величин $a_1, a_2, a_3, \ldots, a_N$ называется конечной суммой $S_N = a_1 + a_2 + a_3 + \ldots + a_N$. Для некоторых последовательностей известны формулы расчета конечных сумм, например:

при $a_N = a_{N-1} + d$ $S_N = (a_1 + a_N) \cdot N / 2$ – арифметическая прогрессия;

при $a_N = a_{N-1} q$; $S_N = (a_1 - a_N q) / (1 - q)$ – геометрическая прогрессия,

где d и q – постоянные числа.

Здесь N-й член последовательности выражается через $(N - 1)$-й член. Такие зависимости называются *реккурентными*.

Конечная сумма последовательности может быть неизвестна, тогда для ее расчета применяется алгоритм суммирования членов последовательности в цикле от 1 до N.

Пример 5.1. Составить блок-схему (рис. 5.1) вычисления суммы вида

$$x - \frac{x^3}{3!} + \frac{x^5}{5!} - \ldots + (-1)^{m-1} \frac{x^{2m-1}}{(2m-1)!}, \quad x \in R,\ m \in N.$$

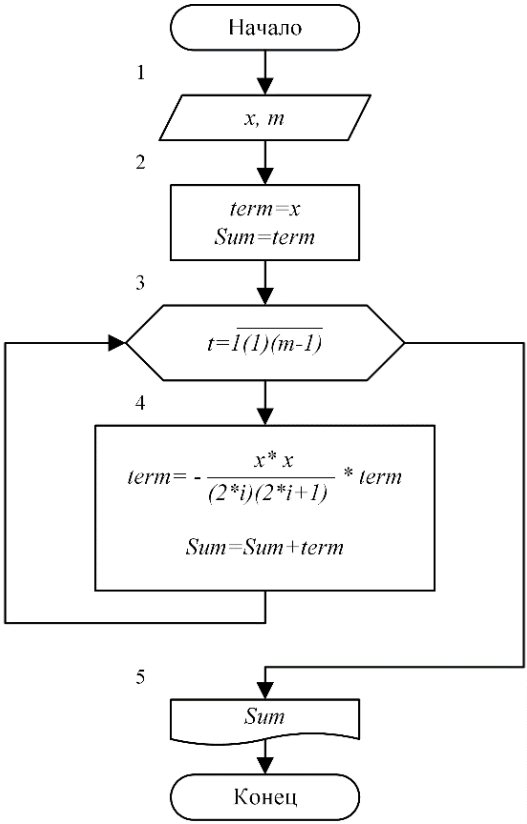


Рис. 5.1. Блок-схема для примера 5.1

Варианты заданий

Вычислить конечные суммы.

Номер варианта	Сумма	Диапазон	n
1	$1+\frac{\ln 3}{1!}x+\frac{\ln^2 3}{2!}x^2+\ldots+\frac{\ln^n 3}{n!}x^n$	$0{,}1 \le x \le 1$	10
2	$\cos x+\frac{\cos 2x}{2!}+\ldots+\frac{\cos nx}{n!}$	$\frac{\pi}{5} \le x \le \frac{9\pi}{5}$	10

Номер варианта	Сумма	Диапазон	n
3	$x-\frac{x^3}{3!}+\ldots+(-1)^n\frac{x^{2n+1}}{(2n+1)!}$	$0{,}1\le x\le 1$	10
4	$1+\frac{x}{1!}+\frac{x^2}{2!}+\ldots+\frac{x^n}{n!}$	$1\le x\le 2$	10
5	$1+\frac{\cos\frac{\pi}{4}}{1!}x+\ldots+\frac{\cos n\frac{\pi}{4}}{n!}x^n$	$0{,}1\le x\le 1$	15
6	$1-\frac{x^2}{2!}+\ldots+(-1)^n\frac{x^{2n}}{(2n)!}$	$0{,}1\le x\le 1$	10
7	$1+\frac{\cos x}{1!}+\ldots+\frac{\cos nx}{n!}$	$0{,}1\le x\le 1$	10
8	$1+3x^2+\ldots+\frac{2n+1}{n!}x^{2n}$	$0{,}1\le x\le 1$	10
9	$1+\frac{x^2}{2!}+\ldots+\frac{x^{2n}}{(2n)!}$	$0{,}1\le x\le 1$	10
10	$1+\frac{2x}{1!}+\ldots+\frac{(2x)^n}{n!}$	$0{,}1\le x\le 1$	10
11	$1+2\frac{x}{2!}+\ldots+\frac{n^2+1}{n!}\left(\frac{x}{2}\right)^n$	$0{,}1\le x\le 1$	10
12	$1-\frac{3}{2!}x^2+\ldots+(-1)^n\frac{2n^2+1}{(2n)!}x^{2n}$	$0{,}1\le x\le 1$	15
13	$-\frac{(2x)^2}{2!}+\frac{(2x)^4}{4!}+\ldots+(-1)^n\frac{(2x)^{2n}}{(2n)!}$	$0{,}1\le x\le 1$	15
14	$-\frac{x}{1!}+\frac{x}{2!}+\ldots+(-1)^n\frac{x}{n!}$	$0{,}1\le x\le 1$	15
15	$\frac{e^x}{2!}+\frac{e^x}{4!}+\frac{e^x}{6!}+\ldots+\frac{e^x}{2n!}$	$2\le x\le 4$	10

Номер варианта	Сумма	Диапазон	n
16	$-x+\frac{\sqrt{x}}{2!}-\frac{\sqrt[3]{x}}{3!}+\ldots+(-1)^n\frac{\sqrt[n]{x}}{n!}$	$10\le x\le 20$	10
17	$1+\frac{\sin^2 x}{3!}+\frac{\sin^4 x}{5!}+\ldots+\frac{\sin^{2n} x}{(2n+1)!}$	$\frac{\pi}{2}\le x\le\frac{3\pi}{2}$	10
18	$1+\frac{x^2}{3!}+\frac{x^4}{5!}+\ldots+\frac{x^{2n}}{(2n+1)!}$	$0{,}1\le x\le 1$	10

Контрольные вопросы

1. Что такое конечные суммы?
2. Как осуществляется ввод-вывод конечных сумм?
3. Составить блок-схему для вычисления конечных сумм.

Лабораторная работа № 6

ОДНОМЕРНЫЕ МАССИВЫ

Цель работы:

– приобрести навыки составления программы на языке C++ с использованием одномерных массивов;

– представить алгоритм решения задачи в виде блок-схемы.

Теоретические сведения

Массивы

Массив (*array*) – это совокупность переменных, имеющих одинаковый тип и объединенных под одним именем. Доступ к отдельному элементу массива осуществляется с помощью индекса. Согласно правилам языка C++ все массивы состоят из смежных ячеек памяти. Младший адрес соответствует первому элементу массива, а старший – последнему.

Массивы могут быть одномерными и многомерными. Наиболее распространенным массивом является строка, завершающаяся нулевым байтом. Она представляет собой обычный массив символов, последним элементом которого является нулевой байт.

Массивы и указатели тесно связаны между собой. Трудно описывать массивы, не упоминая указатели, и наоборот.

Декларация одномерных массивов

Объявление одномерного массива представлено ниже.

тип имя_переменной [размер]

Как и другие переменные, массив должен объявляться явно, чтобы компилятор мог выделить для него память. Здесь тип объявляет базовый тип массива, т. е. тип его элементов, а размер определяет, сколько элементов содержится в массиве. Вид объявления массива с именем balance, имеющего тип double и состоящего из 100 элементов, ниже.

```
double balance[100];
```

Доступ к элементу массива осуществляется с помощью имени массива и индекса. Для этого индекс элемента указывается в квадратных скобках после имени массива. Например, оператор, приведенный ниже, присваивает третьему элементу массива balance значение 12.23.

```
balance[3] = 12.23;
```

Индекс первого элемента любого массива в языке C++ равен нулю. Следовательно, оператор char p[10]; объявляет массив символов, состоящий из 10 элементов – от *p*[0] до *p*[9].

Пример 6.1. Следующая программа заполняет целочисленный массив числами от 0 до 99.

```
#include <stdio.h>
void main()
{
int x[100]; /* Объявление целочисленного массива,
                состоящего из 100 элементов
            */
int t;
/* Заполнение массива числами от 0 до 99 */
for(t=0; t<100; ++t)
   x[t]=t;
/* Вывод на экран элементов массива x */
for(t=0; t<100; ++t)
   printf("%d", x[t]);
}
```

Объем памяти, необходимый для хранения массива, зависит от его типа и размера. Размер одномерного массива в байтах вычисляется по формуле

количество_байт = sizeof(тип) × количество_элементов.

В языке C++ не предусмотрена проверка выхода индекса массива за пределы допустимого диапазона. Иными словами, во время

выполнения программы можно по ошибке выйти за пределы памяти, отведенной для массива, и записать данные в соседние ячейки, в которых могут храниться другие переменные и даже программный код. Ответственность за предотвращение подобных ошибок лежит на программисте.

Например, фрагмент программы, приведенный ниже, будет скомпилирован без ошибок, однако во время выполнения программы индекс массива выйдет за пределы допустимого диапазона.

```
int count[10], i;

/* Выход индекса массива за пределы диапазона */

for(i=0; i<100; i++)
    count[i] = i;
```

По существу, одномерный массив представляет собой список переменных, имеющих одинаковый тип и хранящихся в смежных ячейках памяти в порядке возрастания их индексов.

Ниже показано как хранится в памяти массив *a*, начинающийся с адреса 1000 и объявленный с помощью оператора char *a*[7].

Элемент	*a*[0]	*a*[1]	*a*[2]	*a*[3]	*a*[4]	*a*[5]	*a*[6]
Адрес	1000	1001	1002	1003	1004	1005	1006

Инициализация массива

В языке C++ допускается инициализация массивов при их объявлении. Общий вид инициализации массива не отличается от инициализации обычных переменных.

тип имя_массива[размер1]...[размерN]={список_значений}.

Список_значений представляет собой список констант, разделенных запятыми. Тип констант должен быть совместимым с типом массива. Первая константа присваивается первому элементу массива, вторая – второму и т. д. Обратите внимание на то, что после закрывающейся фигурной скобки } обязательно должна стоять точка с запятой.

Рассмотрим пример, в котором целочисленный массив, состоящий из 10 элементов, инициализируется числами от 1 до 10.

```
int i[10]={1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
```

Здесь элементу *i*[0] присваивается значение 1, а элементу *i*[9] – число 10. Символьные массивы можно инициализировать строковыми константами.

```
char имя_массива[размер] = "строка";
```

Например, следующий фрагмент инициализирует массив str строкой «Я изучаю С».

```
char str[10]="Я изучаю С";
```

Это же можно переписать иначе.

```
char str[10]={'Я', ' ', 'и', 'з', 'у', 'ч', 'а','ю',' ', 'С', '\0'};
```

Поскольку строки завершаются нулевым байтом, размер массива должен быть достаточным, поэтому размер массива str равен 11, хотя строка «Я изучаю С» состоит из 10 символов. Если массив инициализируется строковой константой, компилятор автоматически добавляет нулевой байт.

Инициализация безразмерного массива

При инициализации сообщений об ошибках, каждое из которых хранится в одномерном массиве, необходимо подсчитать точное количество символов в каждом сообщении.

```
char e1[18]="Ошибка при чтении";
char e2[24]="Невозможно открыть файл";
```

Компилятор может сам определить размер массива. Для этого не нужно указывать размер массива.

```
char e1[]="Ошибка при чтении";
char e2[]="Невозможно открыть файл";
```

Такой массив называется *безразмерным*. Инициализация безразмерных массивов, в данном случае, позволяет изменять сообщения, не заботясь о размере массивов.

Пример 6.2. Нахождение суммы элементов массива

```
int a[9]={34, –3, 44, 16, 53, –55, 1, –21, 2};
int i, Sum=0;
for (i=0; i<9; i++)
    Sum+=a[i];
```

Варианты заданий

1. Найти наименьший из положительных элементов массива ($X1, X2, \ldots, X14$).

2. Записать в массив Y десять первых положительных элементов массива ($X1, X2, \ldots, X20$).

3. Вычислить сумму элементов массива ($A1, A2, \ldots, A12$), стоящих на четных местах.

4. Подсчитать количество положительных и отрицательных элементов в массиве ($A1, A2, \ldots, A15$). Нулевые элементы не считать.

5. Для целочисленного массива ($B1, B2, \ldots, B10$) определить, является ли сумма его элементов четным числом, и вывести на печать «Да» или «Нет».

6. Записать +1 вместо максимального элемента массива ($X1, X2, \ldots, X12$), a – 1 вместо минимального.

7. Переписать положительные элементы массива ($X1, X2, \ldots, X20$) подряд в массив Y.

8. Определить разность между наибольшим и наименьшим элементами массива ($Y1, Y2, \ldots, Y20$).

9. Все элементы массива ($A1, A2, \ldots, A20$) уменьшить на величину среднего арифметического этих элементов.

10. Массив ($X1, X2, \ldots, X20$) разбить на два массива: массив положительных элементов и массив отрицательных элементов. Ноль относить к положительным элементам.

11. Переписать элементы массива ($X1, X2, \ldots, X20$) в обратном порядке.

12. Подсчитать сумму элементов массива (*A*1, *A*2, …, *A*12), стоящих на четных местах, и сумму элементов того же массива, стоящих на нечетных местах.

13. Вывести на печать номера элементов (*Y*1, *Y*2, …, *Y*15), удовлетворяющих условию $0 < Yi < 1$.

14. В массе (*A*1, *A*2, …, *A*15) поменять местами минимальный и максимальный элементы.

15. Дан массив (*D*1, *D*2, …, *D*15). Найти произведение элементов с четными номерами и сумму с нечетными номерами.

16. В массиве (*F*1, *F*2, …, *F*10) найти сумму элементов с номерами, кратными трем.

17. В массиве (*A*1, *A*2, …, *A*20) найти количество элементов, равных *X*.

18. Из вектора (*A*1, *A*2, …, *A*15) получить вектор (*B*1, *B*2, …, *B*5), очередная компонента которого равна среднему арифметическому очередной тройки компонент вектора *A*.

Контрольные вопросы

1. Что такое массив?
2. Какие существуют варианты инициализации массива?
3. Как вычислить объем памяти, необходимый для хранения массива?

Лабораторная работа № 7

ДВУМЕРНЫЕ МАССИВЫ

Цель работы:

– ознакомиться с возможностями использования двумерных массивов в языке С++;

– представить алгоритм решения задач с двумерными массивами в виде блок-схемы.

Теоретические сведения

Сходно математическим аналогам – матрицам – массивы могут иметь два и более измерений. Двумерные массивы предназначены для лаконичной замены нескольких одномерных массивов, в частности, в тех ситуациях, когда это действие логически напрашивается (к примеру, при описании координатной зависимости функции от аргумента).

Описание двумерного массива в программе аналогично описанию одномерного массива. Отличие заключается лишь в добавлении дополнительной размерности, взятой в квадратные скобки (собственно, второго измерения).

```
Int main()
{
Double array [range1][range2];
```

В вышеприведенном примере в квадратные скобки заключены размерности описываемого массива. При использовании компилятора от Borland данные величины должны быть константами, использование переменных недопустимо!

Для заполнения массива можно использовать метод последовательного или произвольного присваивания ячейкам требуемых значений (по аналогии с одномерными массивами) или присвоить ячейкам массива значения вручную.

```
Int main()
{
Double array [range1][range2]=
{
```

{*a*1, *a*2, *a*3, *a*4},
{*b*1, *b*2, *b*3, *b*4},
{*c*1, *c*2, *c*3, *c*4},
{*d*1, *d*2, *d*3, *d*4}
};

В данном примере соответствующим элементам массива присваиваются задаваемые разработчиком значения.

Для более надежной работы двумерных массивов в компиляторе Borland рекомендуется придерживаться правил.

1. Нежелательно начинать использование массива с нулевой строки. Несмотря на то, что C++ позволяет использовать эту строку, компилятор не всегда корректно работает с памятью, в результате чего программа выполняется неправильно.

2. При описании массива заданной величины задавать по одной дополнительной строке и столбцу. Их следует оставить пустыми; в случае их отсутствия возможны конфликты переменных на границах массива.

Варианты заданий

1. Из матрицы $A(3\times5)$ построить матрицу B, поменяв местами строки и столбцы.

2. Дана матрица $H(4\times5)$. Найти номер строки, имеющей максимальную сумму элементов.

3. Дана матрица $C(5\times5)$. Вывести на экран элементы главной диагонали и найти сумму их квадратов.

4. Дана матрица $G(4\times5)$. Определить координаты ее минимального элемента и сам элемент, вывести их на экран.

5. В матрице $A(4\times3)$ поменять местами строку, содержащую максимальный элемент массива, со строкой, содержащей минимальный элемент.

6. Вычислить и вывести на экран разность между произведением элементов главной диагонали и элементов побочной диагонали матрицы $B(5\times5)$.

7. Вычислить сумму элементов под главной диагональю квадратной матрицы $E(5\times5)$.

8. Определить минимальный элемент в каждом из столбцов матрицы $C(4\times6)$ и вывести на экран номера строк, в которых они находятся.

9. Вычислить сумму элементов строки и столбца матрицы $D(4\times5)$, на пересечении которых находится минимальный элемент матрицы.

10. В матрице $X(5\times5)$ найти сумму квадратов элементов, расположенных выше главной диагонали.

11. Описать переменную и присвоить ей значение наибольшего из элементов матрицы $A(5\times5)$, расположенных на главной диагонали и выше ее.

12. В матрице $C(4\times5)$ определить все положительные элементы, их сумму вывести на экран.

13. Заполнить одномерный массив элементами главной диагонали матрицы $R(5\times5)$, вывести его на экран.

14. Определить и вывести на экран максимальный и минимальный элементы матрицы $B(4\times4)$.

15. Определить и вывести на экран минимальный положительный и максимальный отрицательный элементы матрицы $M(5\times4)$.

Контрольные вопросы

1. Для чего применяется двумерный массив и в чем заключается его взаимосвязь с одномерным массивом?

2. Как описать в программе двумерный массив?

3. Как заполнить двумерный массив данными?

4. Какие основные элементы блок-схемы программы двумерного массива?

5. Что такое главная диагональ двумерного массива?

Лабораторная работа № 8

ПОДПРОГРАММЫ

Цель работы: ознакомиться с возможностями использования подпрограмм в языке С++, их назначением и особенностями.

Теоретические сведения

При создании сложных программ довольно часто бывает необходимо выполнять один и тот же набор операций несколько раз. Если писать код линейно, то в этом случае он получится с большим количеством одинаковых строк. Альтернативой такому неудобному подходу является использование *подпрограмм* – специальных функций, которым можно задать определенный набор операций, и в случае необходимости его выполнения достаточно сослаться в основном коде программы на функцию и задать требуемый набор входящих и исходящих переменных. В результате длина кода программы существенно сокращается, повышается его читабельность.

Алгоритм описания подпрограммы не покажется новым для пользователей, уже создавших несколько программ на языке С++. В процессе создания программы всегда фигурирует примерно следующая строка.

```
Int main()
{
```

Подобным образом подготавливаем тело программы для компилятора. В скобочках можно указывать дополнительные параметры, что может быть полезно при создании более сложных программ. Подпрограмма описывается абсолютно аналогичным образом. Разберем на примере.

```
Int func(char, int); // объявление функции
Int func(char ch, int chis) // тело функции
{
Cout<<ch<<endl;
Cout<<chis<<endl;
Chis++;
Func==chis;
}
```

```
Int main()
{
Func('T',34);
```

В данном примере подпрограмма описывается вне тела основной программы main(). У подпрограммы есть как входящие, так и исходящие переменные, которыми можно оперировать при необходимости. Тело подпрограммы можно описывать и после тела основной программы main().

Рекурсия

В языке C++ функция может вызывать саму себя.

Функции, вызывающие сами себя, называются *рекурсивными*.

Рекурсия – это процесс определения какого-либо понятия через него же. Иногда его называют круговым определением.

Простым примером рекурсивной функции является функция factr(), вычисляющая факториал целого числа.

Факториалом называется число *n*, представляющее собой произведение всех целых чисел от 1 до *n*. Например, факториал числа 3 равен $1 \cdot 2 \cdot 3 = 6$.

Рассмотрим рекурсивный и итеративный варианты функции factr().

```
/* Рекурсивный вариант */
unsigned long int factr(unsigned int n)
{
if (n==0) return 1;
else return factr(n–1)*n; // рекурсивный вызов
}
/* Итеративный вариант */
unsigned long int fact(unsigned int n)
{
unsigned int t;
unsigned long int answer;
for(t=answer=1; t<=n; answer *=t, t++);
return(answer);
}
```

Итеративный вариант функции factr() выглядит проще. В нем используется цикл, счетчик которого пробегает значения от 1 до *n* и последовательно умножается на предыдущий результат.

Рекурсивная версия функции factr() несколько сложнее. Если данная функция вызывается с аргументом 0, она возвращает число 1. В противном случае она возвращает значение factr(n – 1)n. Чтобы вычислить это выражение, функция factr() вызывается n – 1 раз до тех пор, пока значение переменной n не станет равным 0.

В этот момент начнется последовательное выполнение операторов return, относящихся к разным вызовам функции factr().

При вычислении факториала числа 2 первый вызов функции factr() порождает второй вызов этой функции – теперь уже с аргументом, равным 1. Второй вызов порождает третий – с аргументом, равным 0. Он вернет число 1, которое будет умножено сначала на 1, а потом на 2 (исходное значение аргумента n). Попробуйте сами проследить за вызовами функции factr() при вычислении факториала числа 3. Чтобы увидеть уровень каждого вызова функции factr(), можно вставить в нее вызов функции printf() и вывести промежуточные результаты.

Когда функция вызывает саму себя, в стеке размещается новый набор ее локальных переменных и параметров, и функция выполняется с начала. Рекурсивный вызов не создает новую копию функции. Копируются лишь переменные, с которыми она работает. При каждом рекурсивном возврате управления копии переменных и параметров удаляются из стека, а выполнение программы возобновляется с места вызова функции.

Основное преимущество рекурсивных функций заключается в том, что они упрощают и делают нагляднее некоторые алгоритмы.

Варианты заданий

При выполнении заданий вычисления производить в подпрограмме. В теле основной программы осуществить ввод массива и обратиться к подпрограмме.

1. Найти наименьший из положительных элементов массива ($A$1, $A$2, …, $A$15).

2. Записать в массив C десять первых положительных элементов массива ($A$1, $A$2, …, $A$20).

3. Вычислить сумму четных элементов массива ($A$1, $A$2, …, $A$15).

4. Подсчитать количество положительных и отрицательных элементов в массиве ($A$1, $A$2, …, $A$20). Ноль не является ни положительным, ни отрицательным числом.

5. Дан целочисленный массив из 10 элементов. Определить, является ли сумма его элементов чётным числом, и вывести на экран буквенный ответ.

6. В массиве из 12 элементов заменить максимальный элемент числом +1, а минимальный – числом –1.

7. Дан массив из 20 элементов. Записать все положительные элементы данного массива в новый.

8. Определить разность между наибольшим и наименьшим элементами массива ($A1$, $A2$, …, $A20$).

9. Все элементы массива ($B1$, $B2$, …, $B20$) уменьшить на величину среднего арифметического этих элементов.

10. Массив, состоящий из 20 элементов, разбить на два массива: массив положительных элементов и массив отрицательных элементов. Ноль относить к положительным элементам.

11. Переписать элементы массива ($C1$, $C2$, …, $C20$) в обратном порядке.

12. Вычислить сумму элементов, стоящих на четных позициях, и элементов, стоящих на нечетных позициях, массива ($A1$, $A2$, …, $A20$).

13. В массиве, состоящем из 25 элементов, поменять местами максимальный и минимальный элементы.

14. Дан массив из 25 элементов. Вычислить произведение элементов на четных позициях и сумму элементов на нечетных позициях.

15. В массиве, состоящем из 20 элементов, найти сумму элементов с номерами, кратными трем.

16. Дан массив из 10 элементов. Подсчитать количество элементов, равных A.

Контрольные вопросы

1. Что такое подпрограммы и зачем они применяются?

2. Привести пример программы, в которой использование подпрограмм позволит значительно уменьшить количество кода.

3. Как осуществляется описание подпрограммы? Как ввести входные данные в подпрограмму?

4. Как осуществить вывод данных из подпрограммы и какие типы вывода возможны?

5. Как изобразить блок-схему программы с подпрограммой?

Лабораторная работа № 9

ФАЙЛЫ

Цель работы: ознакомиться с возможностями работы с файлами в языке С++ и их основными особенностями.

Теоретические сведения

Файл – это структурированный и взаимозависимый объем данных, записанный на физический диск в определенном формате данных. Все без исключения пользователи компьютера хоть раз в жизни сталкивались с файлами. Благодаря файлам существуют все программы, в том числе операционные системы. При работе с компьютером постоянно создаются файлы (будь то явно сохраненные файлы или автоматически сохраняемые файлы операционной системы или Microsoft Office, других программ). Таким образом, если нужно сохранить некую информацию для дальнейшего использования после запуска программы, необходимо владеть основами создания и работы с файлами в языках программирования.

Существуют два основных типа файлов: текстовые и двоичные**.**

Текстовыми называются файлы, состоящие из любых символов. Они организуются по строкам, каждая из которых заканчивается символом «*конца строки*». Конец самого файла обозначается символом «*конца файла*». При записи информации в текстовый файл, просмотреть который можно с помощью любого текстового редактора, все данные преобразуются к символьному типу и хранятся в символьном виде.

В *двоичных* файлах информация считывается и записывается в виде блоков определенного размера, в которых могут храниться данные любого вида и структуры.

Для работы с файлами используются специальные типы данных, называмые *потоками.*

Поток **ifstream** служит для работы с файлами в режиме чтения, а **ofstream** – в режиме записи. Для работы с файлами в режиме как записи, так и чтения служит поток **fstream**.

В программах на С++ при работе с текстовыми файлами необходимо подключать библиотки **iostream** и **fstream**.

Чтобы записывать данные в текстовый файл, необходимо:

1) описать переменную типа **ofstream**;
2) открыть файл с помощью функции **open**;
3) вывести информацию в файл;
4) обязательно закрыть файл.

Для считывания данных из текстового файла, необходимо:

1) описать переменную типа **ifstream**;
2) открыть файл с помощью функции **open**;
3) считать информацию из файла; при считывании каждой порции данных необходимо проверять, достигнут ли конец файла;
4) закрыть файл.

Запись информации в текстовый файл

Как было сказано ранее, для того чтобы начать работать с текстовым файлом, необходимо описать переменную типа **ofstream**. Например, так:

```
ofstream F;
```

Будет создана переменная *F* для записи информации в файл. На следующем этапе файл необходимо открыть для записи. В общем случае оператор открытия потока будет иметь вид:

```
F.open(«file», mode);
```

здесь *F* – переменная, описанная как **ofstream**;
file – полное имя файла на диске;
mode – режим работы с открываемым файлом.

Обратите внимание на то, что при указании полного имени файла нужно ставить двойной слеш. Для обращения, например к файлу accounts.txt, находящемуся в папке sites на диске *D*, в программе необходимо указать: D:\\sites\\accounts.txt**.**

Рассмотрим пример.

```
#include <fstream> // файловый ввод/вывод
#include <iostream>
#include <string>
```

```
Using namespace std;
Int main ()
{
Char ch='G';
Int j=44;
String str='File1';
Ofstream outfile ("file1.txt"); // создание файла
Outfile <<ch // вставка символа в файл
<<j
<<' '
<<'Zapis'
<<' '
<<'v'
<<' '
<<str;
Cout << "File record completed!";
Return 0;
}
```

В данном примере следует обратить внимание на несколько особенностей. Во-первых, для разделения пробелами слов или других данных следует использовать пробел, заключенный в одинарные кавычки – ' '. Во-вторых, стринговые переменные следует заполнять без пробелов. В-третьих, при операциях с числовыми данными следует в обязательном порядке разделять их пробелом или нечисловым символом, как было описано выше для текстовых переменных.

Для чтения файлов используются нижеперечисленные команды.

```
Ifstream infile ("file1.txt");
Infile >>ch>>j>>str1>>str2>>str;
Cout << ch<<endl
<<j<<endl
<<str1<<endl
<<str2<<endl
<<str<<endl;
```

В данном примере переменные str1 и str2 типа string использованы для чтения введенных вручную данных «Zapis» и «v». Все остальные

переменные имеют тот же тип данных, что и в предыдущем примере с заполнением файла.

Варианты заданий

Создать текстовый файл исходных данных. В программе элементы массива считывать из этого файла. Результат преобразований записать в новый текстовый файл.

1. Из матрицы $A(3\times5)$ построить матрицу B, поменяв местами строки и столбцы.

2. Дана матрица $H(4\times5)$. Найти номер строки, имеющей максимальную сумму элементов.

3. Дана матрица $C(5\times5)$. Вывести на экран элементы главной диагонали и найти сумму их квадратов.

4. Дана матрица $G(4\times5)$. Определить координаты ее минимального элемента и сам элемент, вывести на экран.

5. В матрице $A(4\times3)$ поменять местами строку, содержащую максимальный элемент массива, со строкой, содержащей минимальный элемент.

6. Вычислить и вывести на экран разность между произведением элементов главной диагонали и элементов побочной диагонали матрицы $B(5\times5)$.

7. Вычислить сумму элементов под главной диагональю квадратной матрицы $E(5\times5)$.

8. Определить минимальный элемент в каждом из столбцов матрицы $C(4\times6)$ и вывести на экран номера строк, в которых они находятся.

9. Вычислить сумму элементов строки и столбца матрицы $D(4\times5)$, на пересечении которых находится минимальный элемент матрицы.

10. В матрице $X(5\times5)$ найти сумму квадратов элементов, расположенных выше главной диагонали.

11. Описать переменную и присвоить ей значение наибольшего из элементов матрицы $A(5\times5)$, расположенных на главной диагонали и выше ее.

12. В матрице $C(4\times5)$ определить все положительные элементы, их сумму вывести на экран.

13. Заполнить одномерный массив элементами главной диагонали матрицы $R(5\times5)$, вывести его на экран.

14. Определить и вывести на экран максимальный и минимальный элементы матрицы $B(4\times4)$.

15. Определить и вывести на экран минимальный положительный и максимальный отрицательный элементы матрицы $M(5\times4)$.

16. Вывести на экран координаты трех наибольших элементов матрицы $T(5\times5)$.

Контрольные вопросы

1. Какие типы файлов можно создать в C++? Какие типы данных можно записывать в файлы?

2. Зачем числовые переменные в файлах разделять нечисловыми символами?

3. Назвать основные команды для записи и чтения файлов в C++.

4. Какие основные особенности работы с файлами в C++?

5. Привести пример кода с записью некоторых данных в файл и последовательно описать операции, которые при этом происходят.

Лабораторная работа № 10

EXCEL И MICROSOFT WORD

Цель работы: освоить методику выполнения наиболее распространенных операций в среде Microsoft Word и Excel.

Теоретические сведения

Microsoft Excel

Назначение электронных таблиц

Электронные таблицы MS Excel предназначены для обработки таблично организованной информации. Особенностью электронных таблиц является структурирование информации непосредственно на этапе ввода данных – данные и формулы хранятся в ячейках рабочего листа (рис. 10.1). Совокупность листов составляет рабочую книгу, которая сохраняется как целостный объект в одном файле с расширением xls.

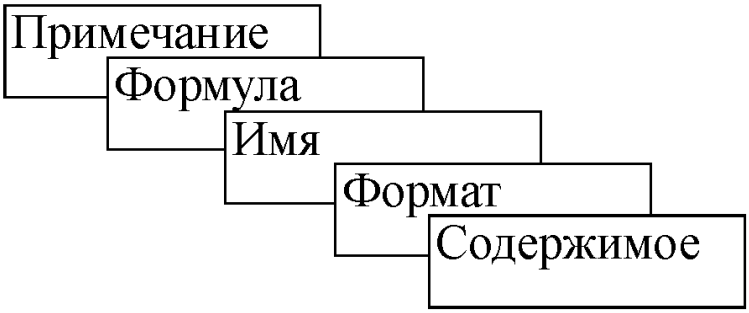


Рис. 10.1. Слои ячейки

Настройки окна MS Excel представлены на рис. 10.2.

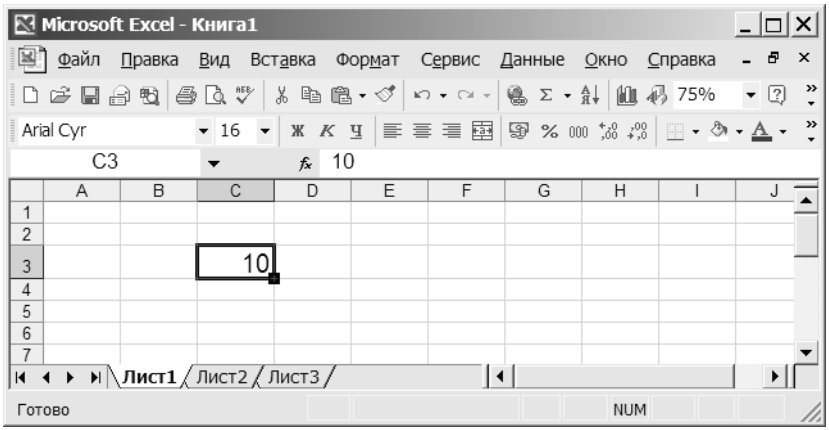


Рис. 10.2. Основные элементы окна MS Excel

Рабочее окно содержит все стандартные элементы, присущие окну приложения Windows. Верхняя строка – заголовок окна, вторая строка – меню Excel, третья строка – панели инструментов, четвертая строка – строка формул, ниже рабочая область книги-документа в Excel.

Строка формул служит для ввода и редактирования данных в ячейках. В левой части формул находится раскрывающийся список – поле имени, в котором высвечиваются адрес или имя выделенной активной ячейки или блока ячеек таблицы.

Между полем имени и строковым полем для ввода и редактирования данных появляются три кнопки для управления процессом ввода: отмена, ввод, вставка функций.

На пересечении столбца с номерами строк и строки с обозначениями столбцов находится «пустая» кнопка для выделения всей таблицы.

Ниже рабочего поля располагается строка с ярлычками рабочих листов книги. Ниже рабочей области расположена **Строка состояния**, в которой высвечиваются режимы работы табличного процессора и выводится дополнительная информация.

Окно документа Excel можно разделить на два или четыре подокна и одновременно работать с разными частями одной и той же таблицы.

Документ в программе Excel называется **рабочей книгой** (Книга1, Книга2 и т. д.), которая состоит из рабочих листов.

Рабочая книга Excel – совокупность **Рабочих листов**, сохраняемых на диске в одном файле. Файл с произвольным именем и расширением *.xls. В каждом файле может размещаться одна книга, а в книге – от 1 до 255 рабочих листов. По умолчанию в книге содержится три рабочих листа.

Рабочие листы можно вставлять, удалять, переставлять. Щелкая по ярлычку листа, можно переходить от одного листа к другому в пределах книги.

Электронная таблица Excel состоит из 65 536 строк и 256 столбцов. Строки нумеруются числами (от одного до 65 536), а столбцы обозначаются буквами латинского алфавита A, B, C, …, Z. После столбца Z следуют столбцы AA, AB, AC, BA, BB,…

На пересечении столбца и строки расположена **ячейка** – область электронной таблицы. Формат и размеры ячеек – ширину столбцов и высоту строк – можно изменить с помощью команд меню, с помощью мыши или клавиш.

Текущая (активная) ячейка – ячейка, в которой в данный момент находится курсор. Она выделена на экране жирной черной рамкой. Каждая ячейка таблицы имеет свой адрес, который используется для указания на ячейку, – при ссылке на нее. Например, A1. Адрес и содержимое текущей ячейки выводятся в строке формул.

Ссылка – способ указания адреса ячейки. Адреса ячеек могут быть относительными или абсолютными. Ячейки могут иметь собственные имена. Ссылки на ячейки (адреса ячеек) используются в формулах, функциях в качестве аргументов.

Типичными установками, принятыми по умолчанию на уровне всех ячеек таблицы, являются:

1) **ширина ячейки** – около восьми разрядов, высота – около 12 пунктов;

2) левое выравнивание для символьных данных;

3) основной формат для цифровых данных с выравниванием вправо.

Блок (диапазон) ячеек – это группа последовательных ячеек. Блок используемых ячеек может быть указан или выделен двумя путями:

1) непосредственным набором с клавиатуры начального и конечного адресов ячеек (A1:C4), формирующих диапазон;

2) выделением блока с помощью мыши или клавиш управления курсором.

Обозначение ячейки, составленное из номера столбца и номера строки, называется *относительным адресом*.

При копировании формул в Excel действует правило относительной ориентации ячеек. Суть состоит в том, что при копировании формулы табличный процессор автоматически смещает адрес в соответствии с относительным расположением исходной ячейки и создаваемой копии.

Абсолютная ссылка создается из относительной ссылки путем вставки знака доллара ($) перед заголовком столбца или номером строки.

Например: $A$1, $B$1 – это абсолютные адреса ячеек A1 и B1. Следовательно, при их копировании не будут меняться ни номер строки, ни номер столбца. Иногда используют смешанный адрес, в котором постоянным может быть один компонент.

Например:

$B7 – при копировании формул не будет изменяться номер столбца;

B$7 – не будет изменяться номер строки.

Для обозначения адреса ячейки с указанием листа используется имя листа и восклицательный знак. Например: Лист2!B5, Итоги!B5.

Для обозначения адреса ячейки с указанием книги используются квадратные скобки. Например: [Книга1]Лист2!A1.

При назначении имени ячейки или диапазону следует соблюдать правила:

1) имя должно начинаться с буквы русского или латинского алфавита, символа подчеркивания или обратной косой черты (\); в имени могут быть точки и вопросительные знаки.

2) цифры могут быть в имени, только не в начале;

3) в имени нельзя использовать пробелы, вместо них можно ставить подчеркивание;

4) длина имени ячейки не должна превышать 255 символов.

Для имени листа существуют следующие ограничения:

1) длина имени листа не больше 31 символа;

2) имя листа не должно содержать квадратные скобки;

3) имя не должно содержать следующие символы: двоеточие, косую черту (/), обратную косую черту (\), знак вопроса, звездочку(*).

Управление средой

Осуществляется посредством команд главного меню, контекстного меню, вызываемого правой кнопкой мыши, кнопок панелей инструментов. Настройки окна Excel производятся командами меню **Вид** и в диалоге **Сервис – Параметры**… В диалоговом окне (рис. 10.3) можно изменить количество листов в книге, стандартный размер и шрифт в ячейках, стиль ссылок (закладка *Общие*), установить режим показа формул в ячейках листа, показать / убрать линии сетки, заголовки строк и столбцов, полосы прокрутки (закладка *Вид*), изменить параметры автосохранения (закладка *Сохранение*), стандартные цвета заливки (закладка *Цвет*), способ пересчета формул (закладка *Вычисления*) и др. С помощью закладки *Списки* можно создавать пользовательские ряды. Для этого нужно в левой части окна выбрать строку *Новый список*, а в правой части ввести элементы списка, нажимая после каждого элемента **Enter**.

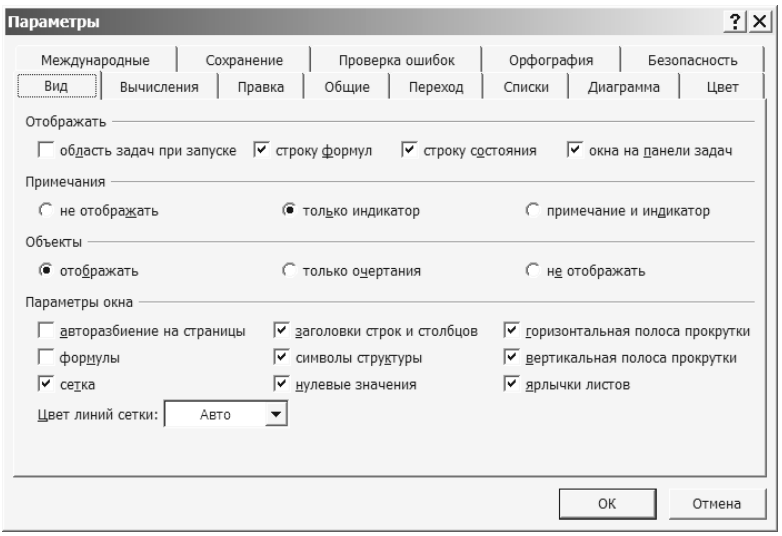


Рис. 10.3. Параметры настроек Excel

Ввод и редактирование данных

Форматирование ячеек. Осуществляется с помощью панели инструментов **Форматирование** или команды меню **Формат – Ячейки**…

Диалог **Формат ячеек** (рис. 10.4) состоит из шести закладок:

1) **Число** – предназначена для форматирования значений ячеек;

2) **Выравнивание** – позволяет:

- расположить содержимое ячейки в любом положении по отношению к ее границам;

- разместить содержимое ячейки в несколько строк (флажок **Переносить по словам**);

- объединить несколько предварительно выделенных ячеек в одну (флажок **Объединение ячеек**);

3) **Шрифт** – дает возможность изменять шрифт, размер и цвет шрифта, начертание, подчеркивание, эффекты (верхний и нижний индекс, зачеркивание);

4) **Граница** – позволяет задавать обрамление ячеек, цвет и тип линии обрамления;

5) **Вид** – можно изменять фон и узор ячеек;

6) **Защита** – используется при создании шаблонов.

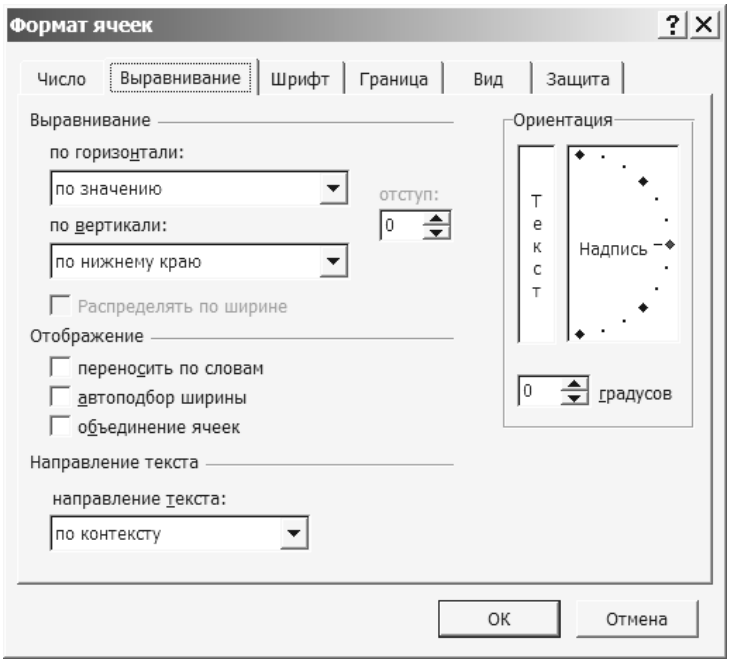


Рис. 10.4. Диалоговое окно **Формат ячеек**

Можно копировать формат по образцу, созданному ранее. Для этого необходимо выделить ячейки, формат которых нужно скопировать, и щелкнуть на кнопке **Формат по образцу**  , затем выделить диапазон, который нужно отформатировать.

Настройка параметров страницы. Управление параметрами страницы и печатью осуществляется с помощью команд меню **Файл – Параметры страницы – Предварительный просмотр – Печать**….

Диалог **Параметры страницы** (рис. 10.5) состоит из четырех закладок:

1) **Страница** – задается размер и ориентация страницы, масштаб печати (таблица при печати пропорционально уменьшается или увеличивается);

2) **Поля** – устанавливаются поля и центрирование таблицы на странице;

3) **Колонтитулы** – создаются верхний и нижний колонтитулы. В режиме создания колонтитулов можно, используя кнопки  , изменить шрифт, вставить номер страницы, количество страниц, текущие дату и время, путь к файлу, имя файла, имя листа, рисунок, отредактировать рисунок;

4) **Лист** – включается печать заголовков строк и столбцов, сетки, примечаний, изменяется порядок вывода страниц на печать. Если таблица многостраничная, то можно автоматически печатать шапку на каждой странице. Для этого нужно указать диапазон ячеек, содержащих заглавия столбцов, в поле **Сквозные строки**.

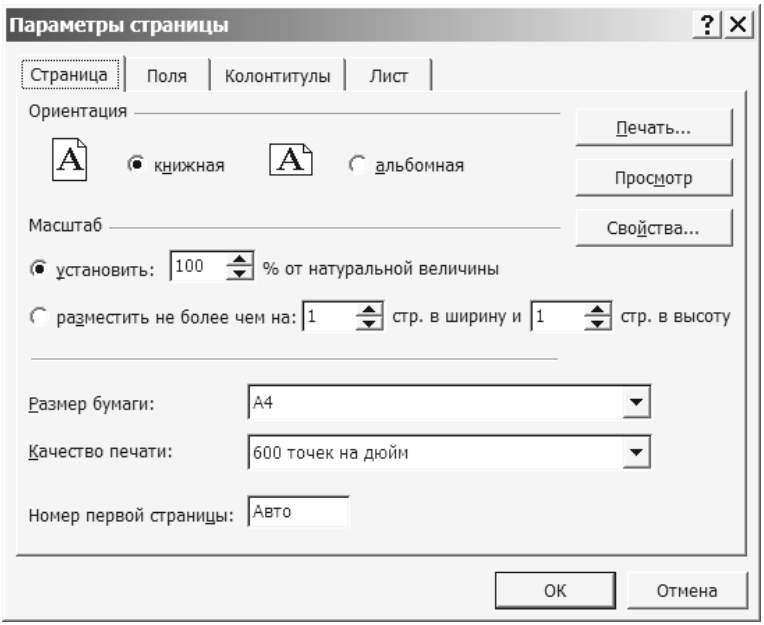


Рис. 10.5. Диалоговое окно **Параметры страницы**

Адресация в Excel. В формулах Excel применяются относительные, абсолютные и смешанные ссылки.

При копировании формулы, содержащей *относительные* ссылки, изменяются относительно расположения ячейки, содержащей формулу. Например, ячейка C1 содержит формулу =A1+B1. При копировании формулы в C2 ссылки изменяются (=A2+B2). Если необходимо, чтобы ссылки не изменялись при копировании формулы, нужно использовать *абсолютные* ссылки. Например, нужно к числам ячеек B1:B5 прибавить значение из A1. Для этого в формуле =A1+B1 нужно использовать абсолютную ссылку на A1. Для обозначения абсолютных ссылок используется знак $. Следовательно, формула в C1 должна иметь вид =$A$1+B1, а при копировании в C2 изменится только относительный адрес (=$A$1+B2).

Ссылка называется ***смешанной***, если одна часть адреса относительная, другая – абсолютная. Например, в ссылке $A1 при копировании формулы будет меняться только строка, в ссылке C$5 – только столбец.

Для циклического изменения типа ссылки используется клавиша **F4**.

Для копирования формулы в смежные ячейки используется маркер заполнения.

Связывание листов. Чтобы использовать в формуле данные, расположенные на другом рабочем листе, удобно открыть новое окно (команда **Окно – Новое**) и расположить их рядом (**Окно – Расположить**…). В одном окне открыть лист с формулой, в другом – с данными и ссылаться на ячейки с помощью мыши. При этом автоматически прописанная ссылка будет содержать имя листа, например: =Лист2**!**A3. Для разделения имени листа и адреса ячейки используется восклицательный знак.

Таким же образом можно ссылаться на данные другой книги, в этом случае ссылка будет содержать имя файла, которое заключается в квадратные скобки, например: =[kurs.xls]Итоги**!**$D$4.

Вставка функций. Для вставки функции применяется кнопка f_x в **Строке формул**. Затем в окне **Мастер функций** следует из соответствующей категории выбрать необходимую функцию, нажать **ОК** и заполнить диалоговое окно **Аргументы функции** (рис. 10.1). Функции могут использоваться как аргументы в других функциях. Допускается использовать до семи уровней вложения функций. Чтобы редактировать формулу, содержащую функции, следует нажать кнопку f_x (рис. 10.1). На экране отобразится диалоговое окно с аргументами первой функции формулы. Изменение какой-либо вложенной функции происходит путем щелчка мышью по имени этой функции в строке формул.

Функции в Excel разделены на категории. В категории **Математические** имеются функции для выполнения арифметических операций, для округления, тригонометрические, степенные, логарифмические функции.

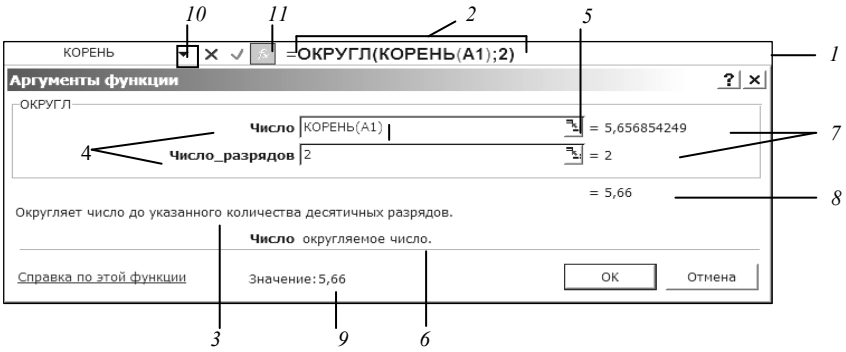


Рис. 10.1. Диалоговое окно **Аргументы функции**:
1 – строка формул; *2* – редактируемая формула; *3* – описание выделенной функции; *4* – поля для заполнения аргументов выделенной функции; *5* – кнопка для сворачивания диалогового окна при заполнении аргумента; *6* – описание выделенного аргумента; *7* – значения аргументов (в A1 → 32); *8* – результат вычисления выделенной функции; *9* – результат вычисления всей формулы; *10* – раскрывающийся список для выбора вложенной функции; *11* – кнопка **Вставка функции**

***Функции округления*. ОКРУГЛ** – округляет число до указанного количества разрядов по общим правилам.

ОКРУГЛВВЕРХ – округляет число до указанного количества разрядов в большую (по модулю) сторону.

ОКРУГЛВНИЗ и **ОТБР** – округляют число до указанного количества разрядов в меньшую (по модулю) сторону.

Табличные формулы или формулы массивов применяются, когда требуется выполнить действия над массивами, а затем вернуть одно или массив значений. Чтобы ввести табличную формулу, необходимо:

1) указать ячейку, в которую нужно ввести формулу, если формула возвращает одно значение, или выделите диапазон ячеек, в которые необходимо ввести формулу, если формула возвращает несколько значений;

2) набрать формулу;

3) нажать клавиши **Ctrl + Shift + Enter**. Табличная формула автоматически заключится в фигурные скобки **{ }**.

Аргументами табличной формулы могут быть как ссылки на диапазоны ячеек, так и массивы констант.

Правила создания массива констант:

– весь массив заключается в фигурные скобки { };

– значения строк разделяются точками с запятой;

– значения столбцов разделяются двоеточием:

Чтобы изменить табличную формулу, которая возвращает массив, необходимо выделить весь массив и отредактировать формулу в строке формул. Завершать редактирование табличной формулы также нужно комбинацией клавиш **Ctrl + Shift + Enter**.

Функция **МОБР** (массив) возвращает обратную матрицу.

Функция **МУМНОЖ** (массив 1; массив 2) возвращает произведение матриц. Результатом является массив с таким же числом строк, как массив 1, и с таким же числом столбцов, как массив 2.

Формулы с использованием ряда математических функций для значения x, введенного в ячейку A1, представлены в табл. 10.1.

Таблица 10.1

Математическое выражение	Формула в MS Excel
$\lvert x\rvert$	=ABS(A1)
$\ln x$	=LN(A1)
e^x	=EXP(A1)
$\sqrt{x}$	=КОРЕНЬ(A1)
$\cos\pi x$	=COS(ПИ()*A1)
$\sin^2 x$	=SIN(A1)^2
$\sqrt[3]{\frac{e^{-3x}+x}{\lvert\sin x-7x\rvert}+\operatorname{tg} x}$	=СТЕПЕНЬ((EXP(3*A1)+A1)/ ABS(SIN(A1)–7*A1)+TAN(A1);1/3)

Microsoft Word 2010

Программа Microsoft Word (или Word) – текстовый процессор, предназначенный для создания, просмотра и редактирования текстовых документов.

Приступая к работе над документом в редакторе Word, следует установить или проверить настройку основных параметров. Ввод символов, вставка в документ таблиц, рисунков и других объектов осуществляется в позицию текстового курсора. Ввод текста производится в режиме **Вставки** или **Замещения**. Для переключения режимов предназначена клавиша **INS**. На рис. 10.2 представлено окно редактора Word.

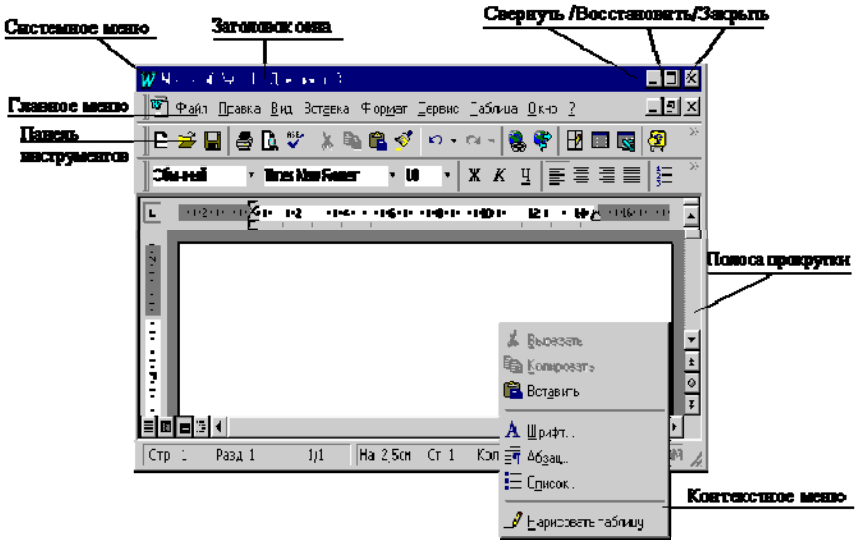


Рис. 10.2. Окно редактора Word

При вводе текста в документ переход на новую строку при достижении правого края страницы осуществляется автоматически (без нажатия клавиши **Enter** – возврат каретки). Клавишу **Enter** следует нажимать только для завершения абзаца и перехода к следующему. Удаление объектов и символов, стоящих справа от курсора, осуществляется с помощью клавиши **Del**. Для удаления символов, стоящих слева от курсора, используют клавишу **Backspace**.

При вводе и форматировании текста необходимо придерживаться следующих правил:

– между словами следует ставить только один пробел;

– перед знаками препинания пробелы ставить не нужно, после знака препинания – обязательно;

– слова, заключенные в кавычки или скобки, не должны отделяться от них пробелами;

– перед и после тире нужно ставить пробелы;

– дефисы следует использовать без пробелов;

– не следует использовать пустой абзац в качестве средства для отступа следующего абзаца, это приводит к «негибкому» форматированию. Для таких целей следует использовать команду главного меню **Разметка страницы – Абзац** и в диалоговом окне **Абзац** установить необходимые отступы и интервалы;

– не следует использовать знак табуляции или несколько пробелов для обозначения красной строки. Установка первых строк производится с помощью команды меню **Разметка страницы – Абзац**.

Варианты заданий

Задание для Microsoft Excel

Создайте таблицу, аналогичную таблице в задании для Microsoft Word.

Выведите аналогичный график на экран, замените график круговой диаграммой.

Добавьте в таблицу дополнительную колонку, в которой проведите вычисление функции из лабораторной работы № 1, номер функции соответствует номеру студента в таблице.

Функцию, соответствующую своему положению в таблице, внесите в новую таблицу, в которой составьте зависимость ее значения от аргумента (шаг разбиения задается преподавателем).

Основываясь на данных таблицы, постройте график функции.

Возьмите у преподавателя новую функцию и постройте ее график, учитывая, что аргументами новой функции будут значения предыдущей функции, вычисленные в таблице.

Постройте график, на котором показана зависимость обеих функций от начального аргумента.

Задание для Microsoft Word

1. Создайте папку **Информатика** на диске **D** для лабораторных работ. В этой папке будут сохранены все результаты Вашей работы.

Запустите программу Microsoft Word. На экране появится окно редактора. Изучите пункты главного меню, панели инструментов и элементы окна.

Установите поля: левое – 2,5 см, правое – 1,5 см, верхнее – 2 см и нижнее – 2 см через команду меню **Разметка страницы – Поля – Настраиваемые поля**. Откроется диалоговое окно **Параметры страницы**.

Научитесь устанавливать и убирать линейку и сетку при помощи команд **Вид – Линейка** и **Вид – Сетка**.

Наберите текст, содержащий краткие сведения о Вас и Ваших компьютерных знаниях (резюме).

Сохраните документ в папке **Информатика**.

Создайте новый документ. Теперь в окне программы Word открыто два документа. Наберите текст титульного листа лабораторной работы. Образец титульного листа возьмите у преподавателя.

Сохраните документ в папке **Информатика**.

Научитесь переключаться между окнами документов и упорядочивать окна всех документов с помощью меню **Вид – Перейти в другое окно**.

Закройте окна всех документов.

Откройте папку **Информатика** на диске **D** и проверьте сохраненные файлы.

2. Создайте новый файл. В файле создайте таблицу из трех столбцов: Фамилия, имя, отчество студента; Родной город; Название месяца рождения.

В таблицу внесите данные всех студентов подгруппы.

По результатам таблицы постройте график, который бы показывал количество студентов, рожденных в каждом месяце.

Выполните предложенное преподавателем форматирование таблицы.

Контрольные вопросы

1. Какое расширение имеют файлы, созданные в Excel?
2. Как сделать ячейку таблицы активной?
3. Как завершить ввод данных в ячейку?
4. Как отредактировать данные в ячейке?

5. Какие бывают типы диаграмм?
6. Что относится к элементам диаграммы?
7. Как отредактировать созданную диаграмму?
8. Изменится ли диаграмма при изменении данных в таблице?
9. Как изменить ориентацию текста в ячейке?
10. Как вставить формулу в ячейку?
11. Для чего предназначен редактор Word?
12. Для чего служат маркеры на горизонтальной линейке прокрутки?
13. Перечислите параметры настройки абзаца.
14. Перечислите комбинации клавиш для быстрого перемещения по документу.
15. Как вставить готовые фигуры в документ?
16. Как можно создать таблицу?
17. Как задать высоту и ширину ячейки таблицы?
18. Как устанавливаются поля в документе?
19. Для чего используются колонтитулы?

Лабораторная работа № 11

ПРОГРАММА MATHCAD

Цель работы: освоить программу вычислений в MathCAD.

Теоретические сведения

MathCAD является интегрированной системой, ориентированной на проведение математических и инженерно-технических расчетов.

После запуска приложения MathCAD открывается окно, как показано на рис. 11.1.

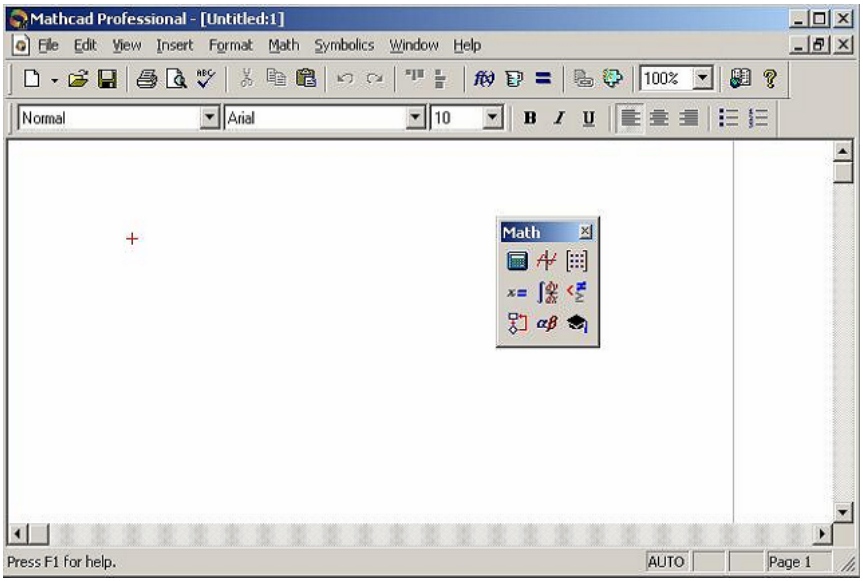


Рис. 11.1. Рабочее окно системы MathCAD

Кнопки панели Math

Одна из сильных сторон MathCAD – это представление и ввод математических символов и выражений в привычной для человека форме. Открыть соответствующую панель инструментов можно

с помощью команды главного меню **View → Toolbars**. Для удобства работы ссылки на них объединены на панели **Math** (рис. 11.2).

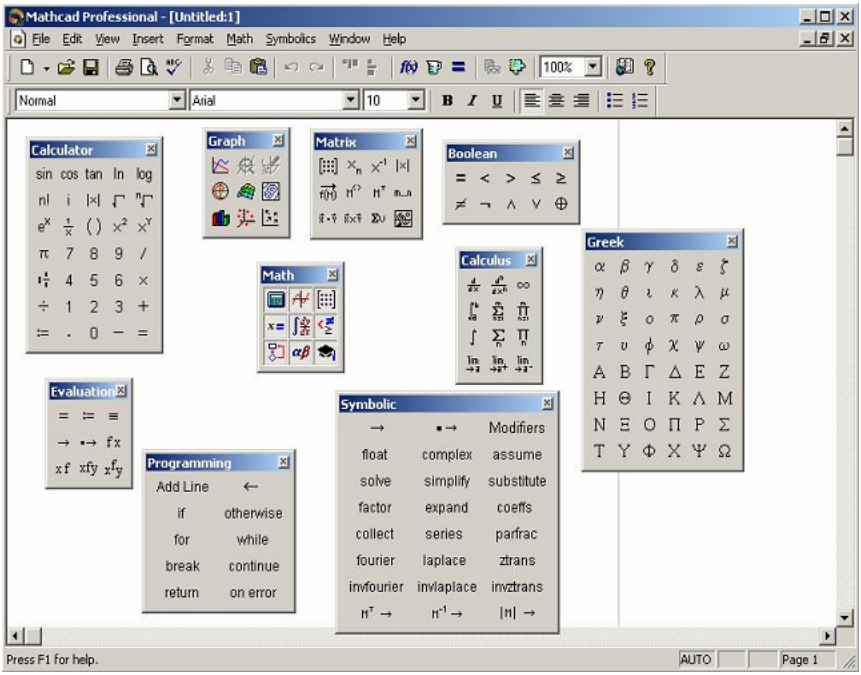


Рис. 11.2. Рабочее окно системы MathCAD
с развернутыми панелями инструментов панели **Math**

На панели **Math** расположены девять кнопок. Каждая из них, в свою очередь, открывает панели инструментов специального назначения.

Calculator – на панели находятся кнопки для задания математических операций, а также некоторых часто используемых функций. Эту кнопку можно использовать как калькулятор.

Boolean – для ввода операторов сравнения и логических операций.

Evaluation – содержит кнопки для ввода операторов присвоения значений переменных и функций.

Graph – инструменты для построения графика.

Vector and Matrix – инструменты для работы с векторами и матрицами.

Calculus – представляет математические выражения с элементами интегрирования, дифференцирования в привычном виде. Кнопки этой панели позволяют вычислять значения пределов, сумм, произведений.

Programming – инструменты для написания программ.

Greak Symbol – графический алфавит.

Symbol – Для символьных вычислений.

Пример 11.1. Требуется вычислить определенный интеграл.

Для этого вначале надо вывести панель **Math**; ее пиктограмма в строке инструментов имеет знаки интеграла и производной. Затем следует установить визир в то место экрана, куда выводится шаблон, и на панели сделать активной пиктограмму с изображением знака определенного интеграла. В составе сложных шаблонов часто встречаются шаблоны для ввода отдельных данных. Они имеют вид небольших черных квадратиков. В шаблоне интеграла их четыре: для ввода верхнего и нижнего пределов интегрирования, для задания подынтегральной функции и для указания имени переменной, по которой идет интегрирование.

Для ввода данных можно указать курсором мыши на нужный шаблон данных и, щелкнув левой ее клавишей для фиксации места ввода, либо воспользоваться клавишей **Tab** и ввести данные.

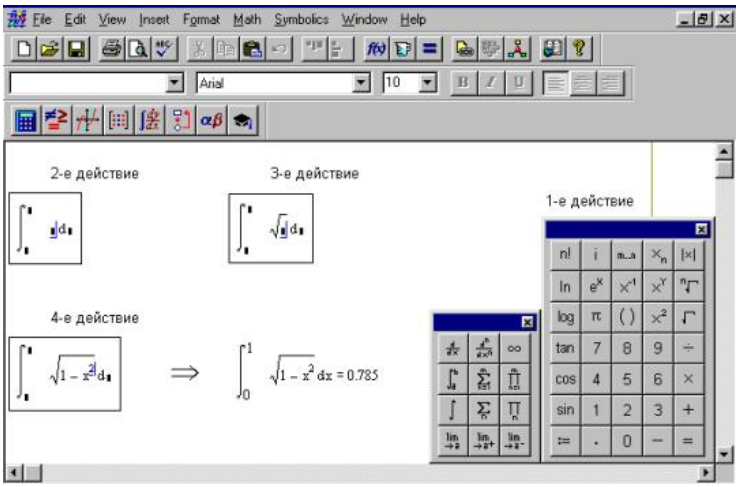


Рис. 11.3. Рабочая область с панелями инструментов панели **Math**

Для ввода подынтегральной функции в приведенном примере требуется совершить следующие действия (рис. 11.3):

а) установив курсор мыши в стороне от места ввода, вывести панель набора арифметических операторов;

б) подвести курсор мыши под шаблон ввода функции и щелкнуть левой клавишей для фиксации начала ввода;

в) активизировать (мышью) кнопку со знаком квадратного корня на палитре математических символов;

г) провести ввод выражения под знаком квадратного корня (при этом возможно редактирование данных с помощью стандартных операций редактирования). Для возведения в степень пользуемся соответствующей кнопкой на панели **Calculator**.

Затем нужно заполнить остальные шаблоны, т. е. ввести пределы интегрирования и имя переменной, по которой производится интегрирование. Установив знак равенства после полученного выражения, можно сразу увидеть результат вычисления интеграла.

Чтобы можно было вычислить выражение, зависящее от каких-либо переменных, их значения должны быть определены.

Для этого нужно:

– ввести имя переменной;

– ввести двоеточие, что приведет к появлению знака присваивания **:=** и следующего за ним поля ввода;

– напечатать в поле ввода число или выражение. MathCad вычислит соответствующее значение и присвоит его имя переменной.

Все переменные и функции, присутствующие во введенном выражении, должны быть определены заранее. В противном случае переменные, значения которых не определены к моменту вычисления выражения, будут отмечены на экране дисплея красным цветом.

Для того чтобы получить числовой результат, нужно:

– ввести в рабочий документ выражение, значение которого требуется определить;

– ввести знак равенства, после чего MathCad вычисляет введенное ранее выражение и выводит в рабочий документ результат расчетов. Для вычисления выражения в ручном режиме необходимо нажать клавишу **F9.**

Пример 11.2. Для задания переменной $x = 1$ и вычисления значения функции $\sin(x)$ можно воспользоваться вводом с клавиатуры.

Построение графиков в декартовой системе координат

Все основные типы графиков и инструменты работы с ними расположены на рабочей панели **Graph** семейства **Math**. На этой панели можно найти ссылки на семь типов графиков. Остановимся на декартовой системе координат.

В MathCAD существует несколько способов построения графиков, однако, первый шаг для всех способов будет один и тот же. Этим шагом является введение специальной заготовки для будущего графика, так называемой *графической области*. Ввести графическую область как для декартового, так и для любого другого графика, можно либо на панели **Graph**, либо командой одноименного подменю меню **Insert**.

Графическая область представляет собой две вложенные рамки (рис. 11.4).

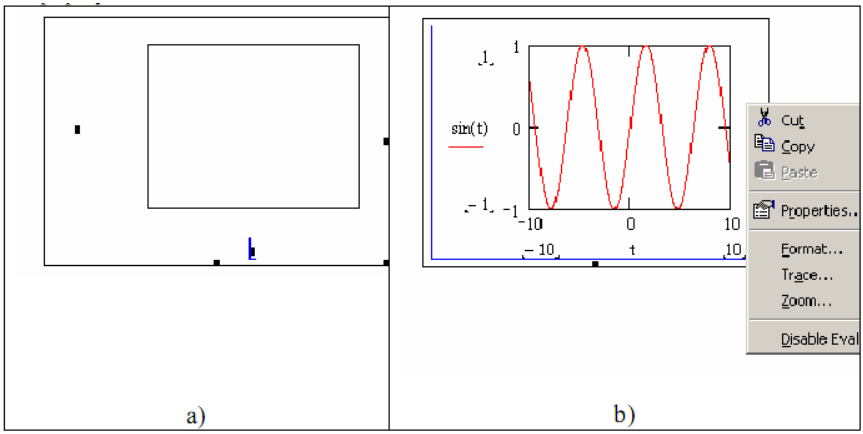


Рис. 11.4. Графическая область в декартовой системе координат

Для построения графика функции необходимо выполнить следующую последовательность действий:

1) введите графическую область;

2) в специальном маркере, расположенном в центре под внутренней рамкой графической области (см. рис. 11.4, *а*), задайте имя независимой переменной;

3) в центральный маркер, расположенный слева от внутренней рамки, введите функцию или имя функции (если функцию определить раньше переменной, то работа упрощается, так как независимая переменная будет задана автоматически).

На рис. 11.4, *б* показан график функции $y = \sin(x)$, построенный по быстрому методу.

Решение уравнений

Решить уравнение – это значит найти точки, в которых функция $f(x)$ принимает нулевые значения.

Пример 11.3. Найти корень уравнения вида $x(x - 6)(x + 3)(5 - x) - 17 = 0$ в численном виде при начальном значении $x = 0$ и заданной точности 10^3.

Процесс решения задачи можно свести к выполнению следующих шагов:

1) определить функцию для решения;

2) задать начальное значение корня;

3) переустановить точность – в данном случае не требуется, так как она соответствует точности, взятой «по умолчанию»;

4) вызвать функцию **root** для решения.

Фрагмент с решением задачи в системе представлен ниже на листинге.

```
Определение функции

f(x) := x(x - 6)·(x + 3)·(5 - x) - 17

Задание начального значения решения

x := 0

root(f(x),x) = -0.188
```

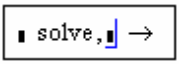
Для *аналитического* решения уравнений в системе MathCAD можно воспользоваться одним из двух способов.

1. С помощью оператора **solve**, расположенного на панели **Symbolic** (Символьные).

2. С помощью команды **solve** из подменю **Symbolics → Variable.**

В первом случае для нахождения корня уравнения необходимо выполнить следующую последовательность действий.

1. Введите оператор **solve** (решить) с помощью одноименной команды панели **Symbolic** (Символьные). В результате будет представлен шаблон нижеприведенного вида.



В левом маркере задайте вид решаемого уравнения.

В качестве знака равенства следует использовать логическое равенство (**Bold Equal** – вводится сочетанием **<Ctrl>+<=>**).

Если уравнение приведено к стандартному виду, то достаточно будет в этот маркер вписать лишь его левую часть. При этом выражение будет приравнено к нулю автоматически. Также в левый маркер можно внести и имя функции – в этом случае будут найдены выражения, определяющие ее нули. Форма записи уравнения через функцию удобна в том случае, если оно имеет большую длину.

2. В правый маркер внесите переменную, относительно которой должно быть решено уравнение, как это показано на рис. 11.5.

Ответ оператор **solve** возвращает в виде выражения (численного или буквенного), которое вполне можно использовать в дальнейших вычислениях. Если решений имеется несколько, то возвращается содержащий их вектор.

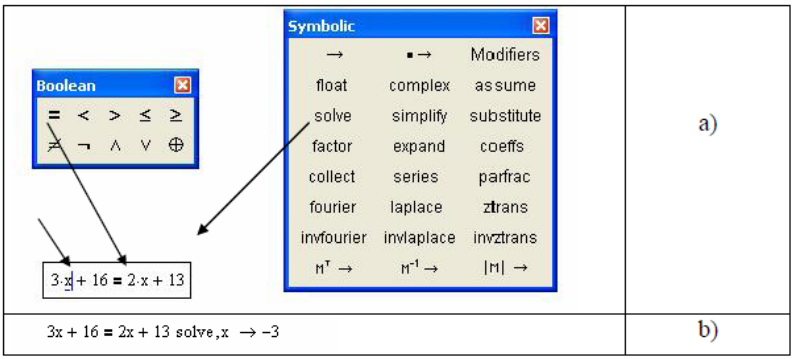


Рис. 11.5. Символьное решение уравнения
с использованием панели инструментов **Symbolic**

При символьном решении уравнений нет особой разницы, сколько переменных содержит уравнение. Ответ ищется в виде выражения, и поэтому для системы неважно, будет ли оно содержать буквенные или численные элементы. Исходя из этого, можно найти корни как уравнения нескольких переменных, так и уравнения с параметрами или буквенными коэффициентами.

Во втором случае, чтобы решить уравнение в символьном виде, нужно ввести уравнение (знак <=> следует брать с панели **Boolean)**, выделить переменную, выбрать команду **solve** из подменю **Symbolics → Variable**, как это показано на рис. 11.6.

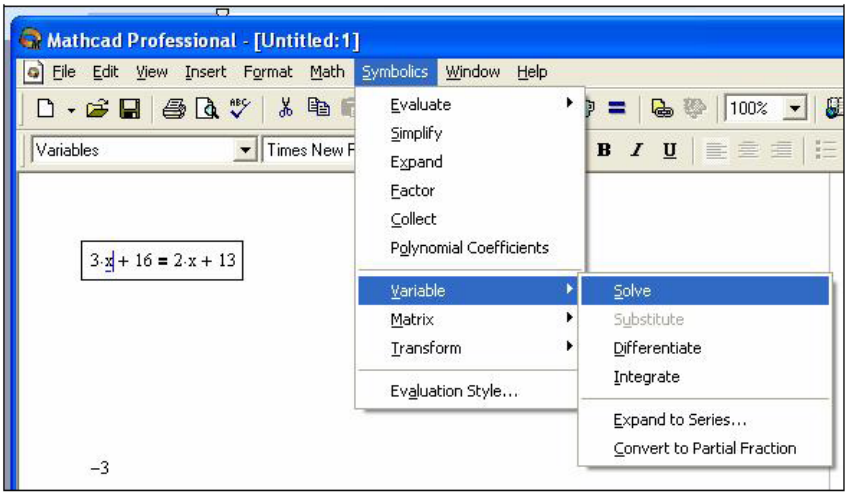


Рис. 11.6. Символьное решение уравнения
с использованием подменю **Variable**

Контрольные вопросы

1. Как MathCAD реализует вычисления?
2. Как вычислить интеграл и производную в системе MathCAD?
3. Опишите порядок действий при решении уравнений с помощью функции **root**.
4. Как в MathCAD получить результат решения уравнения?
5. Порядок действий при построении графика функции в MathCAD.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Павловская, Т. А. C/C++. Программирование на языке высокого уровня : учебник для студентов вузов, обучающихся по направлению «Информатика и вычислительная техника» / Т. А. Павловская. – СПб. : Питер, 2006. – 460 с.

2. Объектно-ориентированное программирование в C++ / Р. Лафоре ; пер. с англ. А. Кузнецова, М. Назарова, В. Шрага. – 4-е изд. – СПб. : Питер, 2005. – 924 с.

3. Круглински, Д. Дж. Программирование на Microsoft Visual C++6.0 = Programming Microsoft Visual C++6.0 : пер. с англ. / Д. Дж. Круглински, Скотт Уингоу, Джордж Шеферд. – – 5-е изд. – М. ; СПб. : Русская редакция : Питер, 2003. – 819 с.

4. Грегори, Кэйт. Использование Visual C++.NET : Специальное издание : пер. с англ. / К. Грегори ; под ред. Г. П. Петриковца. – М. : Издательский дом «Вильямс», 2003. – 784 с.

5. Горбачев, А. Г. Microsoft Excel. Работайте с электронными таблицами в 10 раз быстрее / А. Г. Горбачев, Д. В. Котлеев. – М. : Издательский дом «ДМК-пресс», 2007. – 96 с.: ил.

6. Макаров, Е. Г. Mathcad. Учебный курс / Е. Г. Макаров. – Питер, 2009.

Учебное издание

ИНФОРМАТИКА

Практикум
по программированию на языке С++
для студентов специальностей
1-43 01 04 «Тепловые электрические станции»,
1-43 01 08 «Паротурбинные установки атомных
электрических станций»

В 2 частях

Часть 1

Составители :
ТАРАСЕВИЧ Леонид Александрович
ПРОНКЕВИЧ Елена Васильевна
РОМАНКО Виктория Александровна
ДЕНИСОВ Сергей Михайлович

Редактор *Т. В. Грищенкова*
Компьютерная верстка *Н. А. Школьниковой*

Подписано в печать 30.03.2017. Формат 60×84 $^{1}/_{16}$. Бумага офсетная. Ризография.
Усл. печ. л. 6,45. Уч.-изд. л. 5,05. Тираж 100. Заказ 596.

Издатель и полиграфическое исполнение: Белорусский национальный технический университет.
Свидетельство о государственной регистрации издателя, изготовителя, распространителя
печатных изданий № 1/173 от 12.02.2014. Пр. Независимости, 65. 220013, г. Минск.