Лабораторная работа №1

Выполнение программы простой структуры. Вычисление выражений с использованием стандартных функций

1. Цель задания:

1) Выполнение простой программы в системе программирования
2) Приобретение навыков в записи выражений на языке С++ и использование стандартных функций.

2. Теоретические сведения

2.1.Структура программы на С++

Программа на языке Си имеет следующую структуру:

```
#директивы препроцессора
. . . . . .
#директивы препроцессора
функция а ( )
        операторы
функция в ( )
        операторы
[int | void] main ( )      //функция, с которой начинается выполнение программы
        операторы
                описания
                присваивания
                функция
                пустой оператор
                составной
                выбора
                циклов
                перехода
```

Директивы препроцессора управляют преобразованием текста программы до ее компиляции. Исходная программа, подготовленная на С++ в виде текстового файла, проходит 3 этапа обработки:

1) препроцессорное преобразование текста;
2) компиляция;
3) компоновка (редактирование связей или сборка).

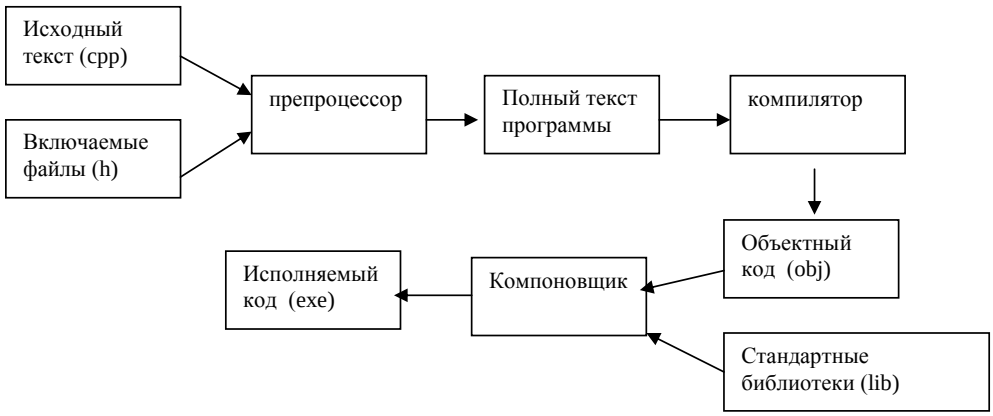


Рис. 2. Обработка С++ программы

После этих трех этапов формируется исполняемый код программы. Задача препроцессора – преобразование текста программы до ее компиляции. Правила препроцессорной обработки определяет программист с помощью директив препроцессора. Директива начинается с #.

`#define` – указывает правила замены в тексте.

`#include<имя заголовочного файла>` – директива предназначена для включения в текст программы текста из каталога заголовочных файлов, поставляемых вместе со стандартными библиотеками. Каждая библиотечная функция С имеет соответствующее описание в одном из заголовочных файлов. Список заголовочных файлов определен стандартом языка. Употребление директивы `include` не подключает соответствующую стандартную библиотеку, а только позволяют вставить в текст программы описания из указанного заголовочного файла. Если используется заголовочный файл из стандартной библиотеки, то его имя заключают в угловые скобки. Если используется заголовочный файл, который находится в текущем каталоге проекта (он может быть создан разработчиком программы), то его имя заключается в кавычки. Подключение кодов библиотеки осуществляется на этапе компоновки, т. е. после компиляции. Хотя в заголовочных файлах содержатся все описания стандартных функций, в код программы включаются только те функции, которые используются в программе.

После выполнения препроцессорной обработки в тексте программы не остается ни одной препроцессорной директивы.

Программа представляет собой набор описаний и определений, и состоит из набора функций. Среди этих функций всегда должна быть функция с именем `main`. Без нее программа не может быть выполнена. Перед именем функции помещаются сведения о типе возвращаемого функцией значения (тип результата). Если функция ничего не возвращает, то указывается тип `void: void main()` (означает, что не возвращает результат). Каждая функция, в том числе и `main`, должна иметь список параметров. Список может быть пустым, тогда он указывается как `(void)` (слово `void` может быть опущено: `()`).

За заголовком функции размещается тело функции. Тело функции – это последовательность определений, описаний и исполняемых операторов, заключенных в фигурные скобки. Каждое определение, описание или оператор заканчивается точкой с запятой.

Определения – вводят объекты (объект – это именованная область памяти, частный случай объекта – переменная), необходимые для представления в программе обрабатываемых данных. Примерами являются

```
const int y = 10 ; //именованная константа
float x ;  //переменная
```

Описания – уведомляют компилятор о свойствах и именах объектов и функций, описанных в других частях программы.

Операторы – определяют действия программы на каждом шаге ее исполнения.

2.2. Элементы языка C/C++

1) Алфавит языка который включает
   - прописные и строчные латинские буквы и знак подчеркивания;
   - арабские цифры от `0` до `9`;
   - специальные знаки `"{},| []()+-/%*.\':;&?<>=!#^`
   - пробельные символы (пробел, символ табуляции, символы перехода на новую строку).
2) Из символов формируются лексемы языка:
   - *Идентификаторы* – имена объектов С/С++-программ. В идентификаторе могут быть использованы латинские буквы, цифры и знак подчеркивания. Прописные и строчные буквы различаются, например, `PROG1`, `prog1` и `Prog1` – три различных идентификатора. Первым символом должна быть буква или знак подчеркивания (но не цифра). Пробелы в идентификаторах не допускаются.

- *Ключевые (зарезервированные) слова* – это слова, которые имеют специальное значение для компилятора. Их нельзя использовать в качестве идентификаторов.
- *Знаки операций* – это один или несколько символов, определяющих действие над операндами. Операции делятся на унарные, бинарные и тернарную по количеству участвующих в этой операции операндов.
- *Константы* – это неизменяемые величины. Существуют целые, вещественные, символьные и строковые константы. Компилятор выделяет константу в качестве лексемы (элементарной конструкции) и относит ее к одному из типов по ее внешнему виду.
- *Разделители* – скобки, точка, запятая пробельные символы.

2.3. Константы в C/C++

Константа – это лексема, представляющая изображение фиксированного числового, строкового или символьного значения. Константы делятся на 5 групп:

- целые;
- вещественные (с плавающей точкой);
- перечислимые;
- символьные;
- строковые.

Компилятор выделяет лексему и относит ее к той или другой группе, а затем внутри группы к определенному типу по ее форме записи в тексте программы и по числовому значению.

Целые константы могут быть десятичными, восьмеричными и шестнадцатеричными.

Название	Определение	Примеры
Десятичная константа	Последовательность десятичных цифр, начинающаяся не с 0, если это число не 0	8, 0, 192345
Восьмеричная константа	Последовательность восьмеричных цифр, которым предшествует 0.	026, 034, 017
Шестнадцатеричная константа	Последовательность шестнадцатеричных цифр, которым предшествуют символы 0x или 0X	0xA, 0X00F, 0x123

Вещественные константы могут иметь две формы представления: с фиксированной точкой и с плавающей точкой.

Название	Вид	Примеры
Константы с фиксированной точкой	[цифры].[цифры]	5.7, .0001, 41.
Константа с плавающей точкой	[цифры][.][цифры]E|e[+|-] [цифры]	0.5e5, .11e-5, 5E3

Перечислимые константы вводятся с помощью ключевого слова enum. Это обычные целые константы, которым приписаны уникальные и удобные для использования обозначения.

```
enum {one=1, two=2, three=3, four=4};
enum {zero,one,two,three};
enum {ten=10, three=3, four, five, six};
enum {Sunday, Monday, Tuesday, Wednesday, Thursday, Friday,
Saturday};
```

Символьные константы – это один или два символа, заключенные в апострофы. Символьные константы, состоящие из одного символа, имеют тип char и занимают в памяти один байт, символьные константы, состоящие из двух символов, имеют тип `int` и занимают

два байта. Последовательности, начинающиеся со знака \, называются управляющими, они используются:

- для представления символов, не имеющих графического отображения, например:
  \a – звуковой сигнал,
  \b – возврат на один шаг,
  \n – перевод строки,
  \t – горизонтальная табуляция;
- для представления символов: \ , ’ , ? , ” (\\, \’ ,\? ,\”);
- для представления символов с помощью шестнадцатеричных или восьмеричных кодов (\073, \0xF5);

Строковая константа – это последовательность символов, заключенная в кавычки. Внутри строк также могут использоваться управляющие символы. Например:

```
”\nНовая строка”,
”\n\”Алгоритмические языки программирования\””.
```

2.3. Типы данных в C++

Типы C++ можно разделить на простые и составные. К простым типам относят типы, которые характеризуются одним значением. В языке C++ определено 6 простых типов данных:

`int` (целый)
`char` (символьный)
`wchar_t` (расширенный символьный) (C++)
`bool` (логический) (C++)
} целочисленные

`float` (вещественный)
`double` (вещественный с двойной точностью)
} с плавающей точкой

Существует 4 спецификатора типа, уточняющих внутреннее представление и диапазон стандартных типов

`short` (короткий)
`long` (длинный)
`signed` (знаковый)
`unsigned` (беззнаковый)

Тип данных	Определение	Размер	Диапазон
(signed) char	Значениями являются элементы конечного упорядоченного множества символов. Каждому символу ставится в соответствие число, которое называется кодом символа.	1 байт	-128..127
unsigned char			0..255
wchar_t	Значениями являются элементы конечного упорядоченного множества символов в кодировке Unicode	2 байта	0..65535
(signed) int	Значениями являются целые числа.	4 байта (для 32-разрядного МП)	–2147483648 … +2147483647.
(signed) long (int)			
unsigned int			0…+4294967 295.

Тип данных	Определение	Размер	Диапазон
unsigned long (int)		4 байта	0…+4294967 295.
(signed) short int		2 байта (для 32-разрядного МП)	–32768 … +32767
unsigned short int			0 … 65535;
bool	Данные этого типа могут принимать значения `true` и `false`.	1 байт	false, true
float	Значениями являются вещественные числа	4 байта	±(3.4E-38..3.4E+38)
double		8 байт	±(1.7E-308 ..1.7E+308)
long double		10 байт	±(3.4E-4932..1E+4932)

2.4. Переменные

Переменная в C++ – именованная область памяти, в которой хранятся данные определенного типа. У переменной есть имя и значение. Имя служит для обращения к области памяти, в которой хранится значение. Перед использованием любая переменная должна быть описана.

```
int a; float x;
```

2.5. Операции

В соответствии с количеством операндов, которые используются в операциях они делятся на унарные (один операнд), бинарные (два операнда) и тернарную (три операнда).

Операция	Описание
Унарные операции	
++	Увеличение на единицу: префиксная операция - увеличивает операнд до его использования, постфиксная операция увеличивает операнд после его использования.
--	Уменьшение на единицу: префиксная операция - уменьшает операнд до его использования, постфиксная операция уменьшает операнд после его использования.
sizeof	вычисление размера (в байтах) для объекта того типа, который имеет операнд
-	Унарный минус
+	Унарный плюс
!	Логическое отрицание (НЕ). В качестве логических значений используется 0 (false) - ложь и не 0 (true) - истина, отрицанием 0 будет 1, отрицанием любого ненулевого числа будет 0.
&	Получение адреса операнда
*	Получение значения, находящегося по указанному адресу (разыменование)
new	Выделение памяти
delete	Освобождение памяти
(type)	Преобразование типа
Бинарные операции	
Мультипликативные	
*	умножение операндов арифметического типа
/	деление операндов арифметического типа (если операнды целочисленные, то выполняется целочисленное деление)
%	получение остатка от деления целочисленных операндов

Аддитивные	
+	бинарный плюс (сложение арифметических операндов)
-	бинарный минус (вычитание арифметических операндов)
Операции сравнения	
<	меньше, чем
<=	меньше или равно
>	больше
>=	больше или равно
==	равно
!=	не равно
Логические о	
&&	конъюнкция (И) целочисленных операндов или отношений, целочисленный результат ложь(0) или истина(не 0)
||	дизъюнкция (ИЛИ) целочисленных операндов или отношений, целочисленный результат ложь(0) или истина(не 0)
Тернарная	
?:	Условная операция в ней используется три операнда. Выражение1 ? Выражение2 : Выражение3; Первым вычисляется значение выражения1. Если оно истинно, то вычисляется значение выражения2, которое становится результатом. Если при вычислении выражения1 получится 0, то в качестве результата берется значение выражения3. Например: x<0 ? -x : x ; //вычисляется абсолютное значение x.
Присваивание	
=	присваивание
*=	умножение с присваиванием (мультипликативное присваивание)
/=	деление с присваиванием
%=	деление с остатком с присваиванием
+=	сложение с присваиванием
-=	вычитание с присваиванием

Приоритеты операций.

Ранг	Операции
1	**( ) [ ] → .**
2	! ~ - префиксные ++ -- & * (тип) sizeof тип( )
3	* / % (мультипликативные бинарные)
4	+ - (аддитивные бинарные)
5	<< >>
6	< > <= >= (отношения)
7	== != (отношения)
8	&
9	^
10	|
11	&& (конъюнкция «И»)
12	|| (дизъюнкция «ИЛИ»)
13	?: (условная операция)
14	= *= /= %= -= &= ^= |= <<= >>= (операция присваивания)
15	, (операция запятая)

2.6. Выражения

Из констант, переменных, разделителей и знаков операций можно конструировать выражения. Каждое выражение представляет собой правило вычисления нового значения. Каждое выражение состоит из одного или нескольких операндов, символов операций и ограничителей. Если выражение формирует целое или вещественное число, то оно называется арифметическим. Пара арифметических выражений, объединенная операцией сравнения, называется отношением. Если отношение имеет ненулевое значение, то оно – истинно, иначе – ложно.

2.7. Ввод и вывод данных

В языке C/C++ нет встроенных средств ввода и вывода – он осуществляется с помощью функций, типов и объектов, которые находятся в стандартных библиотеках. Существует два основных способа: функции C и объекты C++.

Для ввода/вывода данных в стиле C используются функции, которые описываются в библиотечном файле `stdio.h`.

- `printf` (форматная строка, список аргументов);
  форматная строка – строка символов, заключенных в кавычки, которая показывает, как должны быть напечатаны аргументы. Например:

```
printf (”Значение числа Пи равно %f\n”, pi);
```

Форматная строка может содержать:

- символы печатаемые текстуально;
- спецификации преобразования;
- управляющие символы.

Каждому аргументу соответствует своя спецификация преобразования:

`%d, %i` – десятичное целое число;
`%f` – число с плавающей точкой;
`%e,%E` – число с плавающей точкой в экспоненциальной форме;
`%u` – десятичное число в беззнаковой форме;
`%c` – символ;
`%s` – строка.

В форматную строку также могут входить управляющие символы:

`\n` – управляющий символ новая строка;
`\t` – табуляция;
`\a` – звуковой сигнал и др.

Также в форматной строке могут использоваться модификаторы формата, которые управляют шириной поля, отводимого для размещения выводимого значения. Модификаторы – это числа, которые указывают минимальное количество позиций для вывода значения и количество позиций для вывода дробной части числа:

`%[-]m[.p]C`, где

`-` – задает выравнивание по левому краю,
`m` – минимальная ширина поля,
`p` – количество цифр после запятой для чисел с плавающей точкой и минимальное количество выводимых цифр для целых чисел (если цифр в числе меньше, чем значение `p`, то выводятся начальные нули),
`C` – спецификация формата вывода.

Пусть имеем следующие описания переменных: int n=1, m=2; float x=3, y=4;

Ниже приведены различные варианты (допустимые и недопустимые) вывода значений:

```
printf("\n n=%d   m=%d",n,m);   /* Нормальный вывод:      n=1  m=2   */
printf("\n n=%f   m=%f",n,m);      /* Компиляция пройдет успешно, но на этапе  выполнения будет ошибка          */
```

printf("\n x=%f y=%f",x,y); /* Нормальный вывод: x=3.000000 y=4.000000*/
printf("\n x=%d y=%d",x,y); /* Неудовлетворительный вывод: x=0 y=0 */
printf("\n n=%d ",n,m); /* Вывод: n=1. Значение переменной m не распечатается, так как в строке формата нет для нее спецификации*/
printf("\n n=%d, m=%d, z=%d",n,m); /* Не хватает переменных, лишняя спецификация %d. Будет следующий вывод: n=1, m=2, z=0 */

Пример программы на C/C++:

```
#include <stdlib.h>  //препроцессорные директивы
#include <stdio.h>

int main()                //функция
{                         //начало
printf("Hello world! \n");      //печать и перевод курсора на новую строку
system("PAUSE");    //задержать консоль
return 0;   //вернуть системе 0
}
```

- `scanf` (форматная строка, список аргументов);
  в качестве аргументов используются адреса переменных. Например:

```
scanf("%d %f", &x,&y);
```

При использовании библиотеки классов C++, используется библиотечный файл `iostream.h`, в котором определены стандартные потоки ввода данных от клавиатуры `cin` и вывода данных на экран `cout`, а также соответствующие операции

<< – операция записи данных в поток;

>> – операция чтения данных из потока.

```
#include <iostream.h>;
…
cout << "\nВведите количество элементов: ";
cin >> n;
```

В C++ определены в заголовочном файле **<cmath>** математические функции. Например, нахождение корня, возведение в степень, **sin()**, **cos()** и многие другие. В таблице показаны основные математические функций, прототипы которых содержатся в заголовочном файле **<cmath>**. В C необходимо подключать **<math.h>,** правда он не содержит abs(x) (<stdlib.h>)

Таблица - Математические функции в C++

Функция	Описание	Пример
abs(a) fabs(b)	модуль или абсолютное значение от a, где a –int, b- double	abs(-3)= 3 fabs(5.0)= 5.0
sqrt(a)	корень квадратный из a, причём a не отрицательно	sqrt(9.0)=3.0
pow(a, b)	возведение a в степень b	pow(2,3)=8
ceil(a)	округление a до наименьшего целого, но не меньше чем a	ceil(2.3)=3.0 ceil(-2.3)=-2.0
floor(a)	округление a до наибольшего целого, но не больше чем a	floor(12.4)=12 floor(-2.9)=-3

fmod(a, b)	вычисление остатка от a/b	fmod(4.4, 7.5) = 4.4 fmod(7.5, 4.4) = 3.1
exp(a)	вычисление экспоненты e^a	exp(0)=1
sin(a)	a задаётся в радианах	
cos(a)	a задаётся в радианах	
log(a)	натуральный логарифм a(основанием является экспонента)	log(1.0)=0.0
log10(a)	десятичный логарифм a	Log10(10)=1
asin(a)	арксинус a, где -1.0 < a < 1.0	asin(1)=1.5708
tan (a)	Тангенс	
asin (a);	Вычисляет главное значение арксинуса a. Аргумент a должен быть из интервала [-1 ; +1]. Функция возвращает значение в радианах из интервала [- p/2; + p/2]	
atan (a);	Вычисляет главное значение арктангенса a.	

Необходимо запомнить то, что операнды данных функций всегда должны быть вещественными, то есть **a** и **b** числа с плавающей точкой.

2.8. Использование основных операторов языка C++

Операторы управления работой программы называют управляющими конструкциями программы. К ним относят: составные операторы; операторы выбора; операторы циклов; операторы перехода.

2.8.1. Составные операторы

К составным операторам относят собственно составные операторы и блоки. В обоих случаях это последовательность операторов, заключенная в фигурные скобки. Блок отличается от составного оператора наличием определений в теле блока.

```
{
n++;                    //это составной оператор
summa+=n;
}

{
int n=0;
n++;                    //это блок
summa+=n;
}
```

2.8.2. Операторы выбора

Операторы выбора – это условный оператор и переключатель.

1. Условный оператор имеет полную и сокращенную форму.

```
if (выражение-условие) оператор;    //сокращенная форма
```

В качестве выражения-условия могут использоваться арифметическое выражение, отношение и логическое выражение. Если значение выражения-условия отлично от нуля (т. е. истинно), то выполняется оператор.

```
if (x<y&&x<z)min=x;
if (выражение-условие) оператор1;  //полная форма
else оператор2;
```

Если значение выражения-условия отлично от нуля, то выполняется оператор1, при нулевом значении выражения-условия выполняется оператор2.

```
if (d>=0)
{
x1=(-b-sqrt(d))/(2*a);
x2=(-b+sqrt(d))/(2*a);
cout<< "\nx1="<<x1<<"x2="<<x2;
}
else cout<<"\nРешения нет";
```

2. Переключатель определяет множественный выбор.

```
switch (выражение)
{
case константа1 : оператор1 ;
case константа2 : оператор2 ;
. . . . . .
[default: операторы;]
}
```

При выполнении оператора switch, вычисляется выражение, записанное после switch, оно должно быть целочисленным. Полученное значение последовательно сравнивается с константами, которые записаны следом за case. При первом же совпадении выполняются операторы, помеченные данной меткой. Если выполненные операторы не содержат оператора перехода, то далее выполняются операторы всех следующих вариантов, пока не появится оператор перехода или не закончится переключатель. Если значение выражения, записанного после switch, не совпало ни с одной константой, то выполняются операторы, которые следуют за меткой default. Метка default может отсутствовать.

```
#include <iostream>
using namespace std;
int  main()
{
     int i;
     cout<<"\nEnter the number ";
     cin>>i;
     switch(i)
     {
     case 1:cout<<"\nthe number is one";
     case 2:cout<<"\n2*2="<<i*i;
     case 3: cout<<"\n3*3="<<i*i;break;
     case 4: cout<<"\n"<<i<<" is very beautiful!";
     default:cout<<"\nThe end of work";
     }
     return 0;
}
Результаты работы программы:
1.    При вводе 1 будет выведено:
The number is one
2*2=1
3*3=1
2.    При вводе 2 будет выведено:
2*2=4
3*3=4
```

```
3.    При вводе 3 будет выведено:
3*3=9
4.    При вводе 4 будет выведено:
4 is very beautiful!
5.    При вводе всех остальных чисел будет выведено:
The end of work
```

2.8.3. Операторы циклов

- Цикл с предусловием:

```
while (выражение-условие)
оператор;
```

В качестве <выражения-условия> чаще всего используется отношение или логическое выражение. Если оно истинно, т. е. не равно 0, то тело цикла выполняется до тех пор, пока выражение-условие не станет ложным.

```
while (a!=0)
{
cin>>a;
s+=a;
}
```

- Цикл с постусловием:

```
do
оператор
while (выражение-условие);
```

Тело цикла выполняется до тех пор, пока выражение-условие истинно.

```
do
{
cin>>a;
s+=a;
}
while(a!=0);
```

- Цикл с параметром:

```
for (выражение_1;выражение-условие;выражение_3)
оператор;
```

выражение_1 и выражение_3 могут состоять из нескольких выражений, разделенных запятыми. Выражение_1 – задает начальные условия для цикла (инициализация). Выражение-условие определяет условие выполнения цикла, если оно не равно 0, цикл выполняется, а затем вычисляется значение выражения_3. Выражение_3 – задает изменение параметра цикла или других переменных (коррекция). Цикл продолжается до тех пор, пока выражение-условие не станет равно 0. Любое выражение может отсутствовать, но разделяющие их « ; » должны быть обязательно.

```
1.
for ( n=10; n>0; n--)//Уменьшение параметра
{
     оператор;
```

```
}
```

```
2.
for ( n=2; n>60; n+=13)// Изменение шага корректировки
{
     оператор;
}
```

```
3.
for ( num=1;num*num*num<216; num++)//проверка условия отличного от
//того, которое налагается на число итераций
{
     оператор;
}
```

```
4.
for ( d=100.0; d<150.0;d*=1.1)//коррекция с помощью
//умножения
{
     оператор;
 }
```

```
5.
for       (x=1;y<=75;y=5*(x++)+10)//коррекция       с       помощью
//арифметического выражения
{
     оператор;
}
```

```
6.
for    (x=1,    y=0;    x<10;x++,y+=x);//использование    нескольких
корректирующих выражений, тело цикла отсутствует
```

2.8.4. Операторы перехода

Операторы перехода выполняют безусловную передачу управления.

- `break` – оператор прерывания цикла.

```
{
     оператор;
     if (<выражение_условие>) break;
     оператор;
}
```

Т. е. оператор `break` целесообразно использовать, когда условие продолжения итераций надо проверять в середине цикла.

```
// Найти сумму чисел, числа вводятся с клавиатуры до тех пор, пока
не будет //введено 100 чисел или 0.
for(s=0, i=1; i<100;i++)
{
     cin>>x;
     if(   x==0)   break;   //   если   ввели   0,   то   суммирование
заканчивается
     s+=x;
}
```

- `continue` – переход к следующей итерации цикла. Он используется, когда тело цикла содержит ветвления.

```
//Найти количество и сумму положительных чисел
for( k=0,s=0,x=1;x!=0;)
{
     cin>>x;
     if (x<=0) continue;
     k++; s+=x;
}
```

- `goto <метка>` – передает управление оператору, который содержит метку.

  В теле той же функции должна присутствовать конструкция: `<метка>:оператор;`

Метка – это обычный идентификатор, областью видимости которого является функция. Оператор `goto` передает управления оператору, стоящему после метки. Использование оператора `goto` оправдано, если необходимо выполнить переход из нескольких вложенных циклов или переключателей вниз по тексту программы или перейти в одно место функции после выполнения различных действий.

Применение `goto` нарушает принципы структурного и модульного программирования, по которым все блоки, из которых состоит программа, должны иметь только один вход и только один выход.

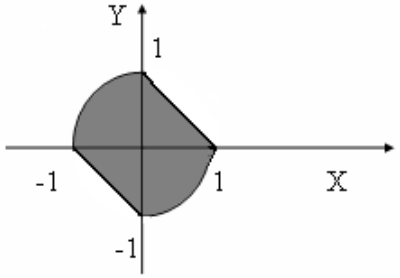
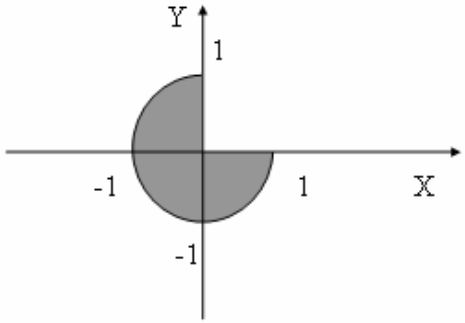
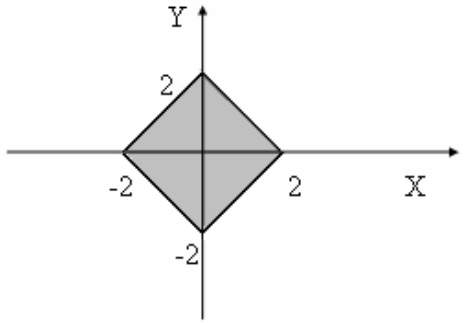
Нельзя передавать управление внутрь операторов `if, switch` и циклов. Нельзя переходить внутрь блоков, содержащих инициализацию, на операторы, которые стоят после инициализации.

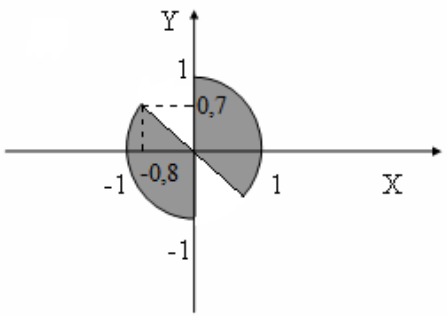
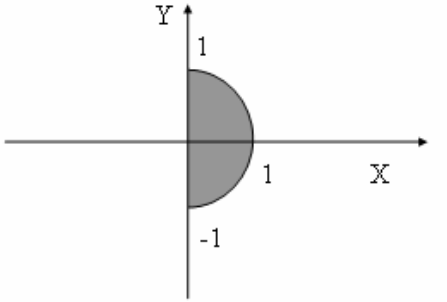
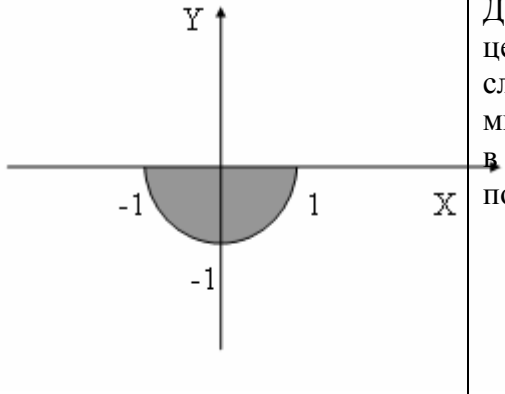
- `return` – оператор возврата из функции. Он всегда завершает выполнение функции и передает управление в точку ее вызова. Вид оператора:
  `return [выражение];`

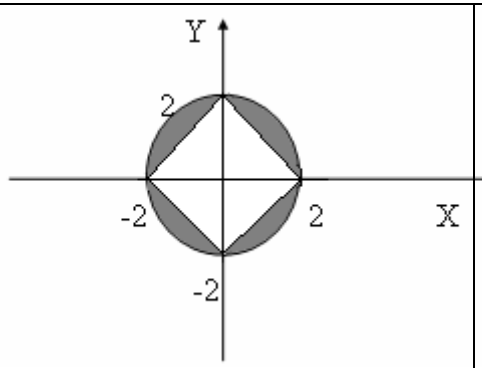
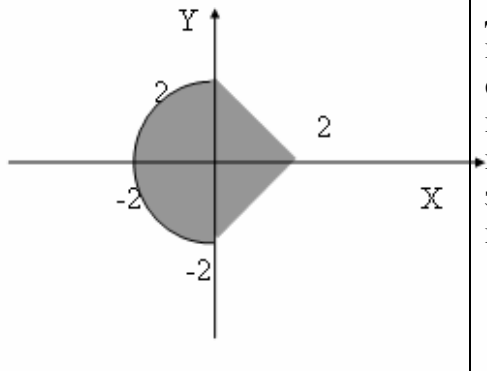
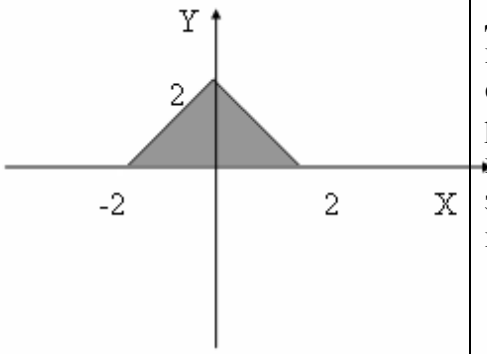
3. Постановка задачи

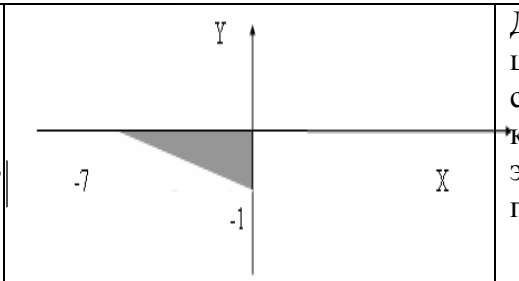
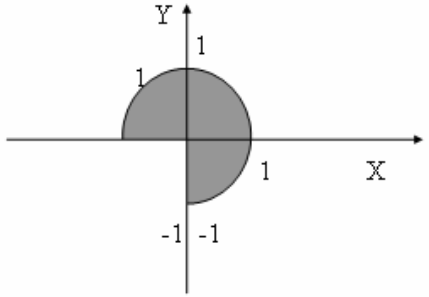
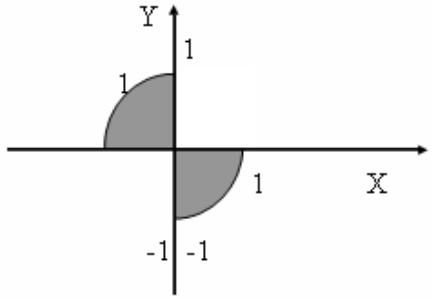
1. Для задания 1 составить линейную программу.
   1.1. Определить тип заданных выражений и найти их значения (или пояснить, в каких случаях выражение не имеет смысл)
   1.2. Для задачи 1.4 составить систему тестов (не менее 4 вариантов) и вычислить полученное выражение, определить при каких X выражение не может быть вычислено.
2. Для задания 2
   2.1. записать выражение, зависящее от координат точки X1 и Y1 и принимающее значение TRUE, если точка принадлежит заштрихованной области, и FALSE, если не принадлежит (тернарная операция)
   2.2. составить программу с условным оператором
   2.3. составить систему тестов и вычислить полученное выражение для нескольких точек, принадлежащих и не принадлежащих заштрихованной области.
3. Для задания 3:
   3.1. составить программу, при решении рекомендуется использовать цикл с условием
   3.2. массивы в этом задании не использовать
4. Для задания 4: Составить программу для заданных x, ε, вводимых с клавиатуры:
   4.1. вычислить сумму тех слагаемых, которые по абсолютному значению больше ε. (Задачу выполнить для двух разных ε, которые отличаются на порядок, для каждого случая вычислить количество слагаемых).
   4.2. сравнить результаты с точным значением соответствующей функции (сумма определяет приближенное значение) для $x \in (-R,R)$

4. Варианты

№	Задание 1	Задание 2	Задание 3	Задание 4
1	1) n++*--m 2) n--<m++ 3) --n>--m 4) $\sqrt[4]{\lvert x+1\rvert}+\frac{1}{x^2}$		Дана последовательность целых чисел, за которой следует 0. Найти сумму четных элементов этой последовательности.	$\frac{sinx}{x}=1-\frac{x^2}{3!}+\frac{x^4}{5!}-\frac{x^6}{7!}+\cdots(R=\infty)$
2	1) n++/--m 2)n-->n/m++ 3)m<n++ 4) $1+x\cos^2(x)+\sin^3($		Дана последовательность целых чисел, за которой следует 0. Найти сумму нечетных элементов этой последовательности.	$e^{-x^2}=1-\frac{x^2}{1!}+\frac{x^4}{2!}-\cdots+(-1)^n\frac{x^{2n}}{n!}\quad(R=\infty)$
3	1) m---n 2) m++<n 3) n++>m 4) $x^4-\cos(\arcsin(x))$		Дана последовательность целых чисел, за которой следует 0. Найти сумму элементов с нечетными номерами из этой последовательности.	$\ln\left(x+\sqrt{x^2+1}\right)=$ $=x-\frac{1}{2}\cdot\frac{x^3}{3}+\frac{1}{2}\cdot\frac{3}{4}\cdot\frac{x^5}{5}-\frac{1}{2}\cdot\frac{3}{4}\cdot\frac{5}{6}\cdot\frac{x^7}{7}+\cdots$ $(R=1)$

4	1) n++*m 2) n++<m 3) --m>n 4) $\sqrt[3]{x-x^2+x^5}$		Дана последовательность целых чисел, за которой следует 0. Найти минимальный элемент в этой последовательности.	$arctg\ x = x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \frac{x^9}{9} - \cdots \quad (R = 1)$
5	1) --m-n++ 2) m*m<n++ 3) n-->++m 4) $tg(x) - (5-x)^4$		Дана последовательность целых чисел, за которой следует 0. Найти номер максимального элемента в этой последовательности.	$\arcsin x = x + \frac{1}{2}\cdot\frac{x^3}{3} + \frac{1}{2}\cdot\frac{3}{4}\cdot\frac{x^5}{5} + \frac{1}{2}\cdot\frac{3}{4}\cdot\frac{5}{6}\cdot\frac{x^7}{7} + \cdots$ $(R = 1)$
6	1) m-++n 2) m++>--n 3) m--<++n 4) $25x^5 - \sqrt{x^2+x}$		Дана последовательность целых чисел, за которой следует 0. Найти номер минимального элемента в этой последовательности.	$\frac{1}{\sqrt{1-x^2}} = 1 + \frac{1}{2}\cdot x^2 + \frac{1}{2}\cdot\frac{3}{4}\cdot x^4 + \frac{1}{2}\cdot\frac{3}{4}\cdot\frac{5}{6}\cdot x^6 + \cdots$ $(R = 1)$

7	1) m+--n 2) m++<--n 3) --m>n-- 4) $\sqrt[5]{x^3+x^4}+$ $+ctg(arctg(x^2))$		Дана последовательность целых чисел, за которой следует 0. Найти максимальный элемент в этой последовательности.	$\frac{1}{\sqrt{1+x}} = 1 - \frac{1}{2}\cdot x + \frac{1}{2}\cdot\frac{3}{4}\cdot x^2 - \frac{1}{2}\cdot\frac{3}{4}\cdot\frac{5}{6}\cdot x^3 + \cdots$ $(R = 1)$
8	1) n/m++ 2)m++<--n 3) (m/n)++<n/m 4) $\sqrt{\lvert x^3-1\rvert} - 7\cos\sqrt[3]{x^4}$		Дана последовательность целых чисел, за которой следует 0. Найти сумму минимального и максимального элементов в этой последовательности.	$\sqrt{1+x} = 1 + \frac{1}{2}\cdot x - \frac{1}{2\cdot 4}\cdot x^2 + \frac{1\cdot 3}{2\cdot 4\cdot 6}\cdot x^3 - \cdots$ $(R = 1)$
9	1) m++/n-- 2) ++m<n-- 3) n-->m 4) $\sin x^3 + x^4 + \sqrt[5]{x^2+x}$		Дана последовательность целых чисел, за которой следует 0. Найти разность минимального и максимального элементов в этой последовательности.	$\frac{1}{(1+x)^3} = 1 - \frac{2\cdot 3}{2}\cdot x + \frac{3\cdot 4}{2}\cdot x^2 - \frac{4\cdot 5}{2}\cdot x^3 + \cdots$ $(R = 1)$

10	1) m/--n++ 2) m/n<n-- 3)m+n++>n+m 4) $x^5\sqrt{\lvert x-1\rvert}+\lvert 25-x^5\rvert$		Дана последовательность целых чисел, за которой следует 0. Найти количество нечетных элементов этой последовательности.	$\frac{1}{(1+x)^2}=1-2x+3x^2-4x^3+5x^4-\cdots \quad (R=1)$
11	1) n+ ++m-- 2) n*m<n++ 3) n-->++m 4) $2^x x\cos(x)+1$		Дана последовательность целых чисел, за которой следует 0. Найти количество четных элементов этой последовательности.	$\frac{1}{1+x}=1-x+x^2-x^3+x^4-\cdots \quad (R=1)$
12	1) n++*m 2) m--<n 3)++m>n 4) $\sqrt{x+\sqrt[4]{\lvert x\rvert}}+\lvert x\rvert$		Дана последовательность целых чисел, за которой следует 0. Найти количество элементов этой последовательности, кратных числу К.	$ln\frac{1+x}{1-x}=2\cdot(x+\frac{x^3}{3}+\frac{x^5}{5}+\frac{x^7}{7}+\frac{x^9}{9}+\cdots) \quad (R=1)$

5. Методические указания

1. Ввод данных для заданий 1-4 организовать с клавиатуры.
2. При вычислении выражений подключить библиотеку <cmath> для вычисления функций (например, pow(x,y)для вычисления x^y).

3. Вывод тестовых результатов для задания 1 организовать в виде: