Лабораторная работа №1

Базовые конструкции языка C/C++. Вычисление выражений с использованием стандартных функций

1. Цель задания:

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

2. Теоретические сведения

2.1. 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	! ~ - ++ -- & * (тип) sizeof тип( )
3	* / % (мультипликативные бинарные)
4	+ - (аддитивные бинарные)
5	< > <= >= (отношения)
6	== != (отношения)
7	&& (конъюнкция «И»)
8	|| (дизъюнкция «ИЛИ»)
9	?: (условная операция)
10	= *= /= %= -= &= ^= |= <<= >>= (операция присваивания)
11	, (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 <stdio.h>  //препроцессорные директивы
#include <iostream.h>
void main()          //функция
{                    //начало
printf(“Hello! “);   //печать
system("PAUSE");     //задержать консоль
}                    //конец
```

- `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.

3.1. 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;
}
```

3.2. 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.h>
void 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";
     }
}
```

```
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
```

3.3. 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
```

3.4. 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;
```

```
}
```

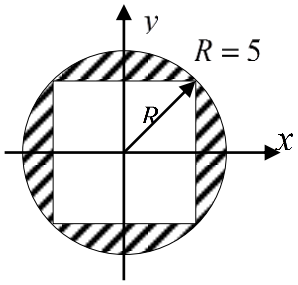
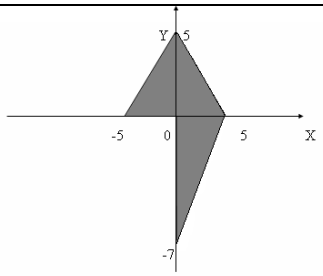
- `continue` – переход к следующей итерации цикла. Он используется, когда тело цикла содержит ветвления.

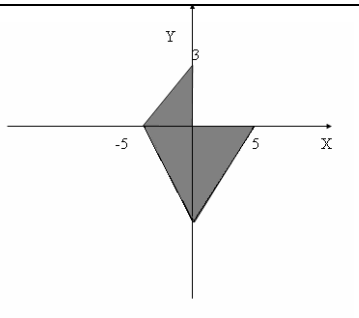
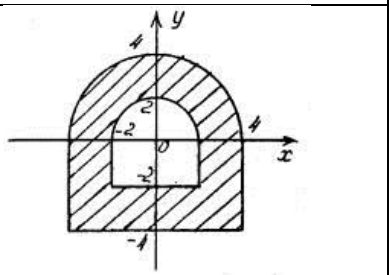
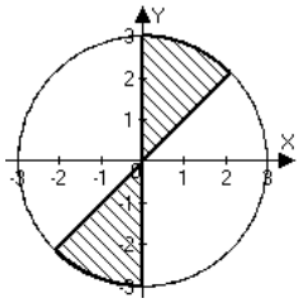
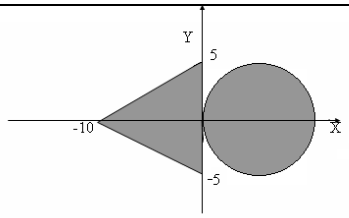
```
//Найти количество и сумму положительных чисел
for( k=0,s=0,x=1;x!=0;)
{
      cin>>x;
      if (x<=0) continue;
      k++; s+=x;
}
```

1. Постановка задачи

1. Для задачи 1 определить тип заданных выражений и найти их значения.
2. Для задачи 1 составить систему тестов и вычислить полученное выражение для нескольких значений X, определить при каких X выражение не может быть вычислено.
3. Для задачи 2 записать выражение, зависящее от координат точки X1 и Y1 и принимающее значение TRUE, если точка принадлежит заштрихованной области, и FALSE, если не принадлежит.
4. Для задачи 2 составить систему тестов и вычислить полученное выражение для нескольких точек, принадлежащих и не принадлежащих заштрихованной области.
5. Для задачи 3 вычислить значение выражения, используя различные вещественные типы данных (float и double).
6. Объяснить полученные результаты.
7. Для задачи 4, учесть ОДЗ, ввести точность вычислений ε=0,001
8. Результаты всех вычислений вывести на экран.

2. Варианты

1	1) (n++/--m)++ 2) ++m<n-- 3)--m>++n 4) $5x^3\sqrt[5]{\frac{1}{x^2}+\frac{1}{x^3}}$		$\frac{(a+b)^2-(a^2+2ab)}{b^2}$ a=1000, b=0.0001
2	1) n++*--m 2) n--<m++ 3) --n>--m 4) $t=\ln\left\|\sin^2 r\right\|+\frac{1+tgy}{(2+1)\sqrt{y}}$ $r=0{,}284\cdot 10^5+\sqrt[5]{\left(y^2+1\right)^2}$		$\frac{(a-b)^2-(a^2-2ab)}{b^2}$ a=1000, b=0.0001

3	1) n++/--m 2) n-->n/m++ 3) m<n++ 4) $\gamma = \dfrac{\arcsin z - x \cdot 0{,}75}{0{,}34 \cdot 10^{-5} e^{x^2} + \sqrt[3]{x + \sin x}}$ $z = \dfrac{1}{\sqrt{1 + x^2}}$		$\dfrac{(a+b)^3 - (a^3 + 3a^2 b)}{3ab^2 + b^2}$ a=100, b=0.001
4	1) n++*m 2) n++<m 3) --m>n 4) $2^{-x}\sqrt{x + \sqrt[4]{\lvert x \rvert}}$		$\dfrac{(a-b)^3 - (a^3)}{3ab^2 - b^3 - 3a^2 b}$ a=100, b=0.001
5	1) n++*m 2) m--<n 3)++m>n 4) $\lambda = \dfrac{125 \cdot 10^6 \sin^2 x^2}{e^5 - \sqrt[5]{x - tgx}}$;		$\dfrac{(a-b)^4 - (a^4 + 6a^2 b^2 + b^4)}{-4ab^3 - 4a^3 b}$ a=10, b=0.01
6	1) m+--n 2) m++<--n 3) --m>n-- 4) $p = \dfrac{\cos^2 x + \dfrac{\sqrt{\sin x}}{x+5}}{2{,}74 \cdot 10^{-8}}$ $t = tg^4 x \cdot 3^x$		$\dfrac{(a-b)^3 - (a^3)}{-b^3 + 3ab^2 - 3a^2 b}$ a=100, b=0.001

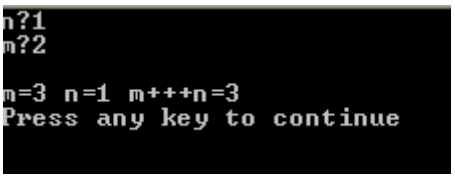
Задача 4

Составить программу табулирования функции $y = f(x)$ на отрезке [a; b] с шагом h. Предусмотреть ОДЗ.

1	$y = \dfrac{tgx}{\ln x - 1}$	4	$y = ctg(\ln x)$
2	$y = \dfrac{\cos^3 x}{1 - lg\, x}$	5	$y = \sqrt[3]{x} - ctg\, x$
3	$y = \dfrac{e^2}{\sqrt{1 - x^2}}$	6	$y = \dfrac{\ln x}{\sqrt[3]{1 - x^2}}$

5. Методические указания

1. Для ввода и вывода данных использовать операции >> и << и стандартные потоки cin, cout (для двух задач), printf и scanf для остальных задач .
2. Ввод данных для заданий 1 и 2 организовать с клавиатуры.
3. При вычислении выражений подключить библиотеку <math.h> для вычисления функций (например, pow(x,y)для вычисления x^y).
4. Вывод результатов для задания 1 организовать в виде:



5. При выполнении задания 2 использовать переменную логического типа, а не условный оператор.
6. При выполнении задания 3 использовать вспомогательные переменные для хранения промежуточных значений. Например **(**для переменных типа double):

c=pow(a,3); d=3*pow(a,2)*b; e=3*a*pow(b,2); f=pow(b,3);

6. Содержание отчета

1) Постановка задачи (общая и конкретного варианта).
2) Формулы, используемые при решении задачи (математическая модель).
3) Программы для решения задач на языке С++.
4) Описание используемых в программе стандартных функций.
5) Система тестов для проверки правильности работы программы и результаты выполнения тестов.
6) Объяснение результатов работы программы.