6 ИСПОЛЬЗОВАНИЕ ФУНКЦИЙ

Цель работы – овладение навыками написания подпрограмм-функций и обращения к ним.

6.1 Подготовка к лабораторной работе

При подготовке к лабораторной работе необходимо ознакомиться с особенностями передачи результата функции в основную программу и вызова функций.

6.2 Теоретические сведения

В С++ задача может быть разделена на более простые подзадачи с помощью функций. Разделение задачи на функции также позволяет избежать избыточности кода, т.к. функцию записывают один раз, а вызывают многократно. Программу, которая содержит функции, легче отлаживать.

Функция – это именованная последовательность описаний и операторов, выполняющая законченное действие, например, формирование массива, печать массива и т.д.

Общая форма определения функции:

тип имя_функции([список_формальных_параметров])

{ тело_функции}

Тело_функции – это блок или составной оператор. Внутри функции нельзя определить другую функцию.

В теле функции должен быть оператор, который возвращает полученное значение функции в точку вызова. Он может иметь формы:

1) *return выражение;*

2) *return*.

Тип возвращаемого значения может быть любым, кроме ***массива и функции***, но может быть указателем на массив или функцию.

Список формальных параметров – это те величины, которые требуется передать в функцию. Элементы списка разделяются запятыми. Для каждого параметра указывается тип и имя. В объявлении имена можно не указывать.

Для того, чтобы выполнялись операторы, записанные в теле функции, функцию необходимо вызвать. При вызове указываются: *имя функции и фактические параметры*. Фактические параметры заменяют формальные параметры при выполнении операторов тела функции.

Фактические и формальные параметры должны совпадать по количеству и типу.

Объявление функции должно находиться в тексте раньше вызова функции, чтобы компилятор мог осуществить проверку правильности вызова. Если функция имеет тип не *void*, то ее вызов может быть операндом выражения.

Пример 1.1. Составить программу для вычисления $z = \dfrac{sh^2 a + sh(a-b)}{sha + \sqrt{sh\left(a^2 - b^2\right)}}$, используя функцию $shx = \dfrac{e^x - e^{-x}}{2}$.

Программа имеет вид:

```
#include <iostream>
#include <math.h>
#include <conio.h>
using namespace std;
double sh(double x)
{
//функция возвращает sh(x)
return (exp(x)-exp(-x))/2.0;
}
void main()
{
        double a, b, z, t1, t2, t3;
        cout<<"\nEnter a, b:";
        cin>>a>>b;
        t1=sh(a);
        t2=sh(a-b);
```

```
	t3=sh(a*a-b*b);
	z = (t1*t1+t2) / (t1+sqrt(t3));
	cout<<"\nz = "<< z;
}
```

Не вводя дополнительных переменных, вызов функции можно выполнить в операторе вывода. Тогда main() будет иметь вид:

```
void main()
{
	double a, b;
	cout<<"\nEnter a, b:";
	cin>>a>>b;
	cout<<"\nz = "<< (sh(a)*sh(a)+sh(a-b)) / (sh(a)+sqrt(sh(a*a-b*b)));
}
```

Основным способом обмена информацией между вызываемой и вызывающей функциями является механизм параметров. Существует два способа ***передачи параметров в функцию: по адресу и по значению***.

При использовании массива как параметра функции в функцию передается указатель на его первый элемент (*передача по адресу*), т.е. массив может быть изменен за счет операторов тела функции. Размерность массива следует передавать как отдельный параметр, т.к. информация о количестве элементов в массиве теряется.

Пример 1.2. Переписать в новый массив отрицательные элементы исходного массива.

```
#include <iostream>
#include <conio.h>
#include <stdlib.h>
using namespace std;
//функция формирования массива
int form(int a[100])
{
	int n;
	cout<<"\nEnter n";
	cin>>n;
	for(int i=0;i<n;i++)
	a[i]=rand()%100-50;
	return n;		//функция возвращает количество элементов в массиве
}
//печать массива
void print(int a[100],int n)
{
	for(int i=0;i<n;i++)
	cout<<a[i]<<" ";
	cout<<"\n";
}
//переписывание в новый массив отрицательные элементы исходного массива
void Neg(int a[100],int n,int b[100],int&k)
{
	k=0;	//количество элементов в новом массиве
	for(int i=0;i<n;i++)
	if(a[i]<0)  b[k++]=a[i];
}
void main()
{
	int mas[100],mas_new[100];
	int n,kol;
	n=form(mas);
	print(mas,n);
	Neg(mas,n,mas_new,kol);
	print(mas_new,kol);
}
```

Строки при передаче в функции могут передаваться как одномерные массивы типа *char* или как указатели типа *char**. В отличие от обычных массивов в функции не указывается длина строки, т.к. в конце строки есть признак конца строки /0.

Пример 1.3. Функция поиска заданного символа в строке

```
#include <iostream>
#include <windows.h>
using namespace std;
int find (char *s,char c)
{
	for (int I=0;I<strlen(s);I++)
	if(s[I]==c) return (I);
	return (-1);
}
/*С помощью функции find подсчитаем количество различных гласных букв в строке.*/
void main()
{
	SetConsoleOutputCP(1251);
	char s[255];
	int k=0;
	cout<<"ВВЕДИТЕ СТРОКУ";
	cin>>s;
	char*gl="aouiey";
	for(int I=0;I<strlen(gl);I++)
	if(find(s,gl[I])>=0)k++;
	cout<<k;
}
```

Пример 1.4.. В заданном количестве строк во всех словах поменять местами первый и последний символы.

```
#include <iostream>
#include <string>
#include <clocale>
using namespace std;
string  Neg(string st)
{
	int i = 0, begin=0,end;
	st=st+" ";
	while (i < st.size()-1)
	{
		while (st[i] == ' ' && i < st.size()-1)
			++i;				//пропустить пробелы
		begin = i;				// номер первого символа слова
		while ( st[i] != ' ' && i < st.size()-1)
			++i;			// пропустить  символы слова
		end = i;    // номер символа,  следующего за последним символом слова
		//поменять местами первый и последний символы слова
		char c=st[begin];
		st[begin]=st[end-1];
		st[end-1] =c;
	}
	 return st;
}
int main()
{
	setlocale (LC_CTYPE,"rus");

	int k;
	cout<<"Введите количество строк"<<"\n";
	cin>>k;
	string str;
```

```
        for (int j = 0;j<k; j++)
        {
                cout<<"Введите строку "<<j+1<<"\n";
                getline(cin,str,'*');           // ввод текста
                cout<<Neg(str)<<"\n";
        }
        return 0;
}
```

6.3 Варианты заданий

Для ввода, вывода массива и выполнения варианта задания использовать функции.

1. Определить произведение двух чисел A·B, где A- сумма первых K, а B - последних K элементов массива. Выполнить для двух массивов.
2. Подсчитать общее количество положительных элементов массивов *A(10), B(20), C(16).*
3. Подсчитать количество пробелов в массивах символов *S(N), S1(N1).*
4. Определить упорядочены ли по возрастанию все элементы массивов F(15), D(18).
5. Для массивов A(M), K(N) определить номер последнего отрицательного элемента.
6. Вычислить произведение положительных элементов до появления первого отрицательного числа в массивах X(14), Y(15).
7. В заданном количестве строк во всех словах удалить последний символ.
8. В заданном количестве строк исключить символы с кодом больше заданного числа.
9. Слова в строках S1, S2 разделены пробелами. Выделить последние слова каждой из строк в новые строки.
10. Составить функцию, в результате обращения к которой из заданной строки в новую переписываются слова с четными номерами. Выполнить для заданного количества строк.
11. Заданы 2 строки символов ST1, ST2. Составить функцию, которая определяет, есть ли в каждой из строк цифры.
12. Составить функцию, позволяющую определить позицию самого правого вхождения заданного символа в строку. Если строка не содержит символа, результатом работы должна быть - 1. Выполнить для строк S1 и S2.
13. Составить функцию, в результате обращения к которой из заданной строки в новую переписываются только буквы латинского алфавита. Выполнить для строк ST1 и ST2.
14. Удалить из строки символы, код которых меньше среднего арифметического кодов всех символов строки. Выполнить для двух строк.
15. Найти суммы всех положительных элементов массива, имеющих четные индексы. Выполнить для двух массивов.
16. Найти разность между максимальным и K-м элементом массива. Выполнить для двух массивов.
17. Найти сумму первых K положительных элементов массива. Выполнить для двух массивов.
18. Найти произведение последних N отрицательных элементов массива. Выполнить для двух массивов.
19. В массиве указать номер первого элемента, сумма которого с последующим за ним элементом не превышает D. Выполнить для двух массивов.
20. Найти большее из чисел A, B, где A - сумма модулей отрицательных, B - произведение положительных элементов массива. Выполнить для двух массивов.
21. Слова в строках S1, S2 разделены пробелами. Выделить первые слова каждой из строк в новые строки.
22. Составить функцию, в результате обращения к которой из заданной строки в новую переписываются слова, длина которых четная. Выполнить для строк S1 и S2.
23. В заданном количестве строк заменить порядок следования слов на обратный.
24. Определить, является ли сумма кодов символов строки, являющихся цифрами, кратной заданному числу. Выполнить для двух строк.
25. Определить, упорядочены ли символы строк по убыванию кодов. Выполнить для двух строк.
26. Удалить все цифры строки. Выполнить для двух строк.
27. Переставить 5 первых символов в конец строки. Выполнить для двух строк.
28. Удалить средний символ строки, если длина строки четная. В противном случае оставить строку без изменения.

6.4 Контрольные вопросы

1. В каких случаях целесообразно использовать функции?
2. Каким образом передается результат выполнения функции в основную программу?
3. Как вызываются функции?

4. Какие способы передачи параметров реализованы в C++?
5. Какие типы данных могут использоваться для задания типа результата функции?
6. В чем состоит особенность использования рекурсивных функций?