Лабораторная работа №6

Функции в С++

1. Цель работы:

1) Получить практические навыки работы с функциями;
2) получить практические навыки работы с шаблонами функций;
3) получить практические навыки работы с указателями функций.

2. Теоретические сведения

2.1. Функции с начальными значениями параметров (по-умолчанию)

В определении функции может содержаться начальное (умалчиваемое) значение параметра. Это значение используется, если при вызове функции соответствующий параметр опущен. Все параметры, описанные справа от такого параметра, также должны быть умалчиваемыми.

```
const int N=20;//количество элементов массива
char mas1[N][10];//массив имен
int mas2[N]//массив возрастов
void init(int i, char* name="Вася ", int age=17)
{
strcpy(mas1[i],name);
mas2[i]=age;
}
```

Примеры использования функции `init()`:

1.
```
for (int i=1;i<N;i++)
      init(i) ;
```

Всем элементам массива `mas1` присваивается значение «Вася», всем элементам массива `mas2` присваивается значение 17.

2.
```
for (int i=1;i<N;i++)
      {
      char Name[10];
      cout<<"Введите имя:"; cin>>Name;
      init(I,Name) ;
      }
```

Всем элементам массива `mas1` присваивается значение переменной Name, всем элементам массива `mas2` присваивается значение 17.

2.2. Функции с переменным числом параметров

В С++ допустимы функции, у которых при компиляции не фиксируется число параметров, и, кроме того, может быть неизвестен тип этих параметров. Количество и тип параметров становится известным только в момент вызова, когда явно задан список фактических параметров. Каждая функция с переменным числом параметров должна иметь хотя бы один обязательный параметр. Определение функции с переменным числом параметров:

```
тип имя (явные параметры,… )
{
      тело функции
}
```

После списка обязательных параметров ставится запятая, а затем многоточие, которое показывает, что дальнейший контроль соответствия количества и типов параметров при обработке вызова функции производить не нужно. При обращении к функции все параметры и обязательные, и необязательные будут размещаться в памяти друг за другом. Следовательно, определив адрес обязательного параметра как `p=&k`, где `p` – указатель, а `k` – обязательный параметр, можно получить адреса и всех остальных параметров: оператор `k++`; выполняет переход к следующему

параметру списка. Еще одна сложность заключается в определении конца списка параметров, поэтому каждая функция с переменным числом параметров должна иметь механизм определения количества и типов параметров. Существует два подхода:

1) известно количество параметров, которое передается как обязательный параметр;
2) известен признак конца списка параметров.

```
//Найти сумму последовательности
//чисел, если известно количество чисел
      #include <iostream>
      using namespace std;
      int sum(int k, ...)
      //явный параметр k задает количество чисел
      {
      int *p=&k;//настроили указатель на параметр k
      int s=0;
      for(;k!=0;k--)
      s+=*(++p);
      return s;
      }
      int main()
      {
      //сумма: 4+6
      cout<<"\n 4+6="<<sum(2,4,6);
      //сумма: 1+2+3+4
      cout<<"\n 1+2+3+4="<<sum(4,1,2,3,4);
      return 0;
      }
```

Для доступа к списку параметров используется указатель `*p` типа `int`. Он устанавливается на начало списка параметров в памяти, а затем перемещается по адресам фактических параметров (`++p`).

```
/*Найти сумму последовательности чисел, если известен признак конца списка
параметров */
#include <iostream>
using namespace std;
int sum(int k, ...)
{
int *p = &k;     //настроили указатель на параметр k
int s = 0;       //значение первого параметра присвоили s
for(int i=1;*p!=0;p++)     //пока нет конца списка
s += *p;
return s;
}

int main()
{
// сумма: 4+6
cout<<"\n 4+6="<<sum(4,6,0);
// сумма: 1+2+3+4
cout<<"\n 1+2+3+4="<<sum(1,2,3,4,0);
return 0;
}
```

2.3. Перегрузка функций

Цель перегрузки состоит в том, чтобы функция с одним именем по-разному выполнялась и возвращала разные значения при обращении к ней с различными типами и различным числом фактических параметров. Для обеспечения перегрузки необходимо для каждой перегруженной функции определить возвращаемые значения и передаваемые параметры так, чтобы каждая перегруженная функция отличалась от другой функции с тем же именем. Компилятор определяет, какую функцию выбрать по типу фактических параметров.

```
#include <iostream.h>
#include <string.h>

//сравнение двух целых чисел
int max(int a, int b)
{
      if (a>b) return a;
      else return b;
}

//сравнение двух вещественных чисел
float max(float a, float b)
{
      if(a>b)return a;
      else return b;
}

//сравнение двух строк
char* max(char* a, char* b)
{
      if (strcmp(a,b)>0) return a;
      else return b;
}

void main()
{
      int a1,b1;
      float a2, b2;
      char s1[20];
      char s2[20];

      cout<<"\nfor int:\n";
      cout<<"a=?";cin>>a1;
      cout<<"b=?";cin>>b1;
      cout<<"\nMAX="<<max(a1,b1)<<"\n";

      cout<<"\nfor float:\n";
      cout<<"a=?";cin>>a2;
      cout<<"b=?";cin>>b2;
      cout<<"\nMAX="<<max(a2,b2)<<"\n";

      cout<<"\nfor char*:\n";
      cout<<"a=?";cin>>s1;
      cout<<"b=?";cin>>s2;
      cout<<"\nMAX="<<max(s1,s2)<<"\n";
}
```

Правила описания перегруженных функций:

- Перегруженные функции должны находиться в одной области видимости.
- Перегруженные функции могут иметь параметры по умолчанию, при этом значения одного и того же параметра в разных функциях должны совпадать. В разных вариантах перегруженных функций может быть разное количество умалчиваемых параметров.
- Функции не могут быть перегружены, если описание их параметров отличается только модификатором `const` или наличием ссылки: функции `int& f1(int&, const int&){…}` и `int f1(int, int){…}` – не являются перегруженными, т. к. компилятор не сможет узнать какая из функций вызывается, потому что нет синтаксических отличий между вызовом функции, которая передает параметр по значению и функции, которая передает параметр по ссылке.

2.3. Шаблоны функций

Шаблоны вводятся для того, чтобы автоматизировать создание функций, обрабатывающих разнотипные данные. Например, алгоритм сортировки можно использовать для массивов различных типов. При перегрузке функции для каждого используемого типа определяется своя функция. Шаблон функции определяется один раз, но определение параметризируется, т. е. тип данных передается как параметр шаблона.

```
template <class имя_типа [,class имя_типа]>
заголовок_функции
{
      тело функции
}
```

Таким образом, шаблон семейства функций состоит из 2 частей – заголовка шаблона: `template<список параметров шаблона>` и обыкновенного определения функции, в котором вместо типа возвращаемого значения и/или типа параметров, записывается имя типа, определенное в заголовке шаблона.

```
//сравнение двух чисел любого типа
template<class T>
T max(T a, T b)
{
      if (a>b) return a;
      else return b;
}
```

Шаблон служит для автоматического формирования конкретных описаний функций по тем вызовам, которые компилятор обнаруживает в программе. Например, если в программе вызов функции осуществляется как `max(1,5)`, то компилятор сформирует определение функции `int max(int a, int b){…}`.

2.4. Указатель на функцию

Каждая функция характеризуется типом возвращаемого значения, именем и списком типов ее параметров. Если имя функции использовать без последующих скобок и параметров, то он будет выступать в качестве указателя на эту функцию, и его значением будет выступать адрес размещения функции в памяти. Это значение можно будет присвоить другому указателю. Тогда этот новый указатель можно будет использовать для вызова функции.

Указатель на функцию определяется следующим образом:

```
тип_функции(*имя_указателя)(спецификация параметров)
```

В определении указателя количество и тип параметров должны совпадать с соответствующими типами в определении функции, на которую ставится указатель.

Вызов функции с помощью указателя имеет вид:

```
(*имя_указателя)(список фактических параметров);
```

```
#include <iostream.h>
int f1(char* S)
{
cout<<S<<"\n";
return 1;
}

void main()
{
      char s[20]="\nfunction1";
      int(*ptr1)(char*);    //указатель на функцию
      ptr1=f1;         //указателю присваивается адрес функции f1
      cout<<((*ptr1)(s));   //вызов функции f1 с помощью указателя
}
```

Указатели на функции удобно использовать в тех случаях, когда функцию надо передать в другую функцию как параметр.

```
#include <iostream.h>
#include <math.h>

typedef float(*fptr)(float);//тип-указатель на функцию уравнения

/*решение уравнения методом половинного деления, уравнение передается с
помощью указателя на функцию */
float root(fptr f, float a, float b, float e)
{
      float x;
      do
      {
            x=(a+b)/2;            //находим середину отрезка
            if ((*f)(a)*f(x)<0)        //выбираем отрезок
            b=x;
            else a=x;
      }
      while((*f)(x)>e && fabs(a-b)>e);
      return x;
}

//функция, для которой ищется корень
float testf(float x)
{
      return x*x-1;
}

void main()
{
      /*в функцию root передается указатель на функцию, координаты отрезка
      и точность */
      float res=root(testf,0,2,0.0001);
      cout<<"\nX="<<res;
}
```

2.5. Численные методы решения уравнений

Довольно часто на практике приходится решать уравнения вида:

$$F(x) = 0, \qquad (1),$$

где функция F(x) определена и непрерывна на некотором конечном или бесконечном интервале $\alpha < x < \beta$.

Всякое значение $\bar{x}$ такое, что $F(x) \equiv 0$, называется корнем уравнения, а нахождение этого значения и есть решение уравнения.

На практике в большинстве случаев найти точное решение возникшей математической задачи не удается. Поэтому важное значение приобрели численные методы, позволяющие найти приближенное значение корня. Под численными методами подразумеваются методы решения задач, сводящиеся к арифметическим и некоторым логическим действиям над числами, т.е. к тем действиям, которые выполняет ЭВМ.

Существует множество численных методов. Рассмотрим только три из них:

- метод итераций;
- метод Ньютона;
- метод половинного деления.

2.5.1. Метод итераций

Представим уравнение F(x) = 0 в виде:

$$x = f(x). \qquad (2)$$

Это уравнение получается выделением x из уравнения F(x) и переносом того, что осталось, т.е. f(x), в левую часть уравнения. Иначе можно получить уравнение (2) следующим способом: левую и правую часть уравнения (1) умножить на произвольную константу λ и прибавить к левой и правой части x, т.е. получаем уравнение вида:

$$x = x + \lambda F(x), \qquad (3)$$

где $f(x) = x + \lambda F(x)$.

На заданном отрезке [a; b] выберем точку x0 – нулевое приближение – и найдем

$$x1 = f(x0),$$

потом найдем:

$$x2 = f(x1),$$

и т.д.

Таким образом, процесс нахождения корня уравнения сводится к последовательному вычислению чисел:

$$xn = f(xn-1) \qquad n = 1,2,3..... .$$

Этот процесс называется методом итераций.

Если на отрезке [a; b] выполнено условие:

$$|f'(x)| \le q < 1,$$

то процесс итераций сходится, т.е.

$$\lim_{n\to\infty} x_n = \bar{x}$$

Процесс итераций продолжается до тех пор, пока

$$|xn - xn-1| \le \varepsilon,$$

где – заданная абсолютная погрешность корня x. При этом будет выполняться:

$$|\bar{x} - xn| \le \varepsilon.$$

2.5.2. Метод Ньютона

Пусть уравнение F(x) = 0 имеет один корень на отрезке [a; b], причем F′(x) и F″(x) определены, непрерывны и сохраняют постоянные знаки на отрезке [a; b].

Выберем на отрезке[a; b] произвольную точку x0 – нулевое приближение. Затем найдем:

$$x_1 = x_0 - \frac{F(x_0)}{F'(x_0)},$$

потом

$$x_2 = x_1 - \frac{F(x_1)}{F'(x_1)}$$

Таким образом, процесс нахождения корня уравнения сводится к вычислению чисел x_n по формуле:

$$x_n = x_{n-1} - \frac{F(x_{n-1})}{F'(x_{n-1})}, n = 1,2,3,...$$

Этот процесс называется методом Ньютона.

Процесс вычисления продолжается до тех пор, пока не будет выполнено условие:

$$|x_n - x_{n-1}| \le \varepsilon$$

Точку x0 необходимо выбирать так, чтобы выполнялось условие:

$$F(x_0) \cdot F''(x_0) > 0,$$

иначе метод не будет сходиться.

2.5.3. Метод половинного деления

Пусть уравнение F(x) = 0 имеет один корень на отрезке [a;b]. Функция F(x) непрерывна на отрезке [a; b].

Метод половинного деления заключается в следующем:

Сначала выбираем начальное приближение, деля отрезок пополам, т.е.

$$x_0 = (a+b)/2.$$

Если $F(x)=0$, то x_0 является корнем уравнения. Если $F(x) \neq 0$, то выбираем тот из отрезков, на концах которого функция имеет противоположные знаки. Полученный отрезок снова делим пополам и выполняем действия сначала и т.д.

Процесс деления отрезка продолжаем до тех пор, пока длина отрезка, на концах которого функция имеет противоположные знаки, не будет меньше заданного числа ε.

3. Постановка задачи

1. Написать функцию с умалчиваемыми параметрами в соответствии с вариантом, продемонстрировать различные способы вызова функции:
   - с параметрами заданными явно,
   - с опущенными параметрами
   - часть параметров задана явно, а часть опущена.
2. Написать функцию с переменным числом параметров в соответствии с вариантом, продемонстрировать вызов функции с различным числом параметров.
3. Написать перегруженные функции в соответствии с вариантом. Написать демонстрационную программу для вызова этих функций.
4. Написать шаблон функций вместо перегруженных функций из задания 3. Написать демонстрационную программу для вызова этих функций.
5. Решить уравнение указанным в варианте методом. Уравнение передать в функцию как параметр с помощью указателя.

4. Варианты

№ варианта	Функция с умалчиваемыми параметрами	Функция с переменным числом параметров	Перегруженные функции и шаблон функции	Передача функции как параметра другой функции с помощью указателя
1	Печать фамилии, имени и отчества	Минимальный элемент в списке параметров	Среднее арифметическое массива	Метод итераций $3\sin\sqrt{x} + 0{,}35x - 3{,}8 = 0$ Отрезок, содержащий корень: [2;3] Точное значение: 2,2985
2	Печать фамилии, имени и возраста	Максимальный элемент в списке параметров	Количество отрицательных элементов в массиве	Метод Ньютона $3\sin\sqrt{x} + 0{,}35x - 3{,}8 = 0$ Отрезок, содержащий корень: [2;3] Точное значение: 2,2985
3	Печать фамилии, курса и группы	Количество четных элементов в списке параметров	Максимальный элемент в массиве	Метод половинного деления $3\sin\sqrt{x} + 0{,}35x - 3{,}8 = 0$ Отрезок, содержащий корень: [2;3] Точное значение: 2,2985
4	Печать фамилии, имени и рейтинга	Среднее арифметическое элементов в списке параметров	Минимальный элемент в массиве	Метод итераций $0{,}25x^3 + x - 1{,}2502 = 0$ Отрезок, содержащий корень: [0;2] Точное значение: 1,0001
5	Печать фамилии, курса и рейтинга	Максимальный из элементов в списке параметров, стоящих на четных местах	Сортировка массива методом простого обмена	Метод Ньютона $0{,}25x^3 + x - 1{,}2502 = 0$ Отрезок, содержащий корень: [0;2] Точное значение: 1,0001
6	Печать фамилии, адреса и возраста	Максимальный из элементов в списке параметров, стоящих на нечетных местах	Сортировка массива методом простого выбора	Метод половинного деления $0{,}25x^3 + x - 1{,}2502 = 0$ Отрезок, содержащий корень: [0;2] Точное значение: 1,0001

7	Печать названия экзамена, количества сдающих и среднего балла	Минимальный из элементов в списке параметров, стоящих на четных местах	Сортировка массива методом простого включения	Метод итераций $x - \frac{1}{3 + \sin 3{,}6x} = 0$ Отрезок, содержащий корень: [0;0,85] Точное значение: 0,2624
8	Печать названия экзамена, даты экзамена и среднего балла	Минимальный из элементов в списке параметров, стоящих на нечетных местах	Поиск заданного элемента в массиве	Метод Ньютона $x - \frac{1}{3 + \sin 3{,}6x} = 0$ Отрезок, содержащий корень: [0;0,85] Точное значение: 0,2624
9	Печать координат точки	Среднее арифметическое из элементов в списке параметров, стоящих на четных местах	Поиск заданного элемента в отсортированном массиве	Метод половинного деления $x - \frac{1}{3 + \sin 3{,}6x} = 0$ Отрезок, содержащий корень: [0;0,85] Точное значение: 0,2624
10	Вычисление и печать расстояния от точки с координатами x1,y1 до центра координат	Среднее арифметическое из элементов в списке параметров, стоящих на нечетных местах	Удаление элемента с заданным номером из динамического массива	Метод итераций $0{,}1x^2 - x \ln x = 0$ Отрезок, содержащий корень: [1;2] Точное значение: 1,1183
11	Вычисление и печать расстояния от точки с координатами x1,y1 до точки с координатами x2,y2	Минимальный элемент в списке параметров	Удаление элемента с заданным ключом из динамического массива	Метод Ньютона $0{,}1x^2 - x \ln x = 0$ Отрезок, содержащий корень: [1;2] Точное значение: 1,1183
12	Печать фамилии, имени и отчества	Максимальный элемент в списке параметров	Добавление элемента с заданным номером в динамический массив	Метод половинного деления $0{,}1x^2 - x \ln x = 0$ Отрезок, содержащий корень: [1;2] Точное значение: 1,1183
13	Печать фамилии, имени и возраста	Количество четных элементов в списке параметров	Добавление элемента после элемента с заданным номером в динамический массив	Метод итераций $3x - 4\ln x - 5 = 0$ Отрезок, содержащий корень: [2;4] Точное значение: 3,2300
14	Печать фамилии, курса и группы	Среднее арифметическое элементов в списке параметров	Номер максимального элемента в массиве	Метод Ньютона $3x - 4\ln x - 5 = 0$ Отрезок, содержащий корень: [2;4] Точное значение: 3,2300
15	Печать фамилии, имени и рейтинга	Максимальный из элементов в списке параметров, стоящих на четных местах	Среднее арифметическое массива	Метод половинного деления $3x - 4\ln x - 5 = 0$ Отрезок, содержащий корень: [2;4] Точное значение: 3,2300

16	Печать фамилии, курса и рейтинга	Максимальный из элементов в списке параметров, стоящих на нечетных местах	Количество отрицательных элементов в массиве	Метод итераций $\arccos x - \sqrt{1 - 0{,}3x^3} = 0$ Отрезок, содержащий корень: [0;1] Точное значение: 0,5629
17	Печать фамилии, адреса и возраста	Минимальный из элементов в списке параметров, стоящих на четных местах	Добавление элемента с заданным номером в динамический массив	Метод Ньютона $\arccos x - \sqrt{1 - 0{,}3x^3} = 0$ Отрезок, содержащий корень: [0;1] Точное значение: 0,5629
18	Печать названия экзамена, количества сдающих и среднего балла	Минимальный из элементов в списке параметров, стоящих на нечетных местах	Сортировка массива методом простого обмена	Метод половинного деления $\arccos x - \sqrt{1 - 0{,}3x^3} = 0$ Отрезок, содержащий корень: [0;1] Точное значение: 0,5629
19	Печать названия экзамена, даты экзамена и среднего балла	Среднее арифметическое из элементов в списке параметров, стоящих на четных местах	Минимальный элемент в массиве	Метод итераций $\sqrt{1 - 0{,}4x^2} - \arcsin x = 0$ Отрезок, содержащий корень: [0;1] Точное значение: 0,7672
20	Печать координат точки	Среднее арифметическое из элементов в списке параметров, стоящих на нечетных местах	Сортировка массива методом простого выбора	Метод Ньютона $\sqrt{1 - 0{,}4x^2} - \arcsin x = 0$ Отрезок, содержащий корень: [0;1] Точное значение: 0,7672
21	Вычисление и печать расстояния от точки с координатами x1,y1 до центра координат	Минимальный элемент в списке параметров	Сортировка массива методом простого включения	Метод половинного деления $\sqrt{1 - 0{,}4x^2} - \arcsin x = 0$ Отрезок, содержащий корень: [0;1] Точное значение: 0,7672
22	Вычисление и печать расстояния от точки с координатами x1,y1 до точки с координатами x2,y2	Максимальный элемент в списке параметров	Поиск заданного элемента в массиве	Метод итераций $e^x - e^{-x} - 2 = 0$ Отрезок, содержащий корень: [0;1] Точное значение: 0,8814
23	Печать фамилии, имени и отчества	Количество четных элементов в списке параметров	Поиск заданного элемента в отсортированном массиве	Метод Ньютона $e^x - e^{-x} - 2 = 0$ Отрезок, содержащий корень: [0;1] Точное значение: 0,8814
24	Печать фамилии, имени и возраста	Среднее арифметическое элементов в списке параметров	Удаление элемента с заданным номером из динамического массива	Метод половинного деления $e^x - e^{-x} - 2 = 0$ Отрезок, содержащий корень: [0;1] Точное значение: 0,8814

25	Печать фамилии, курса и группы	Максимальный из элементов в списке параметров, стоящих на четных местах	Удаление элемента с заданным ключом из динамического массива	Метод итераций $x - 2 + \sin\frac{1}{x} = 0$ Отрезок, содержащий корень: [1;2] Точное значение: 1,3077

5. Методические указания

1. В функции с умалчиваемыми параметрами использовать структурированный тип данных.
2. При демонстрации вызова функции с умалчиваемыми параметрами учесть, что опускать параметры функции можно только с конца.
3. В функции с переменными числом параметров можно использовать любой механизм определения конца списка параметров (передачу количества параметров как параметр функции или использование признака конца списка параметров).
4. Перегрузить функции для массивов типа char, int, и double.
5. Описать шаблон функции для типов char, int, и double.
6. Для нахождения корня уравнения написать как минимум две функции. Одна функция реализует уравнение, для которого вычисляется корень, другая - метод решения уравнения, указанный в варианте. Первая функция передается во вторую как параметр, с помощью указателя.
7. Точность нахождения корня уравнения выбирается не менее 0.001.
8. Полученный результат вычисления корня сравнить с точным значением, заданным в задании.

6. Содержание отчета

1. Постановка задачи (общая и для конкретного варианта).
2. Определения функций для реализации поставленных задач.
3. Определение функции main().
4. Тесты