Учебная практика
Основы C++

День 2

Функции

Параметры функций

Параметры-массивы

Параметры-функции

Объявление функций

Можно объявлять функцию вне классов. Это похоже на статические методы из C#, где можно вызывать функцию независимо от объекта.

Описание или объявление функции должно быть до ее использования

Функцию можно объявить отдельно от реализации.

Реализация может быть в любом другом месте этого файла

```
int fun(); // function declaration

...

int fun() { // function implementation
    return 0;
}
```

Функции

1) Описание функции до ее первого использования
2) Прототип функции до ее первого использования, а описание – ниже (после main).

Прототип функции – это ее заголовок.

```
int  example1_prototip(int a, int* b, double f);
```

Имена формальных параметров указывать необязательно, достаточно указать типы

```
int  example2_prototip(int, float*, double);
```

Локализация переменных

Область видимости переменных – блок, в котором они объявлены {…}

Глобальные переменные объявляются вне функций, область видимости – вся программа.

Если появляется одноименная локальная переменная, она имеет преимущество, «скрывает» глобальную переменную.

Перегрузка функций

Перегруженные функции имеют одинаковые имена, различаются типом и/или количеством параметров

Если функции различаются только типом возвращаемого значения, компилятор не может определить, какую функцию выполнить.

Проблема – неоднозначность при вызове функций. Возникает из-за автоматического преобразования типов.

Примеры перегрузки функций

```
4  bool function1(int A, int B) {
5      return A < B;
6  }
7  int function1(int A, int B) {
8      return A + B;
9  }
10 float function1(int A, int B) {
11     return A - B;
12 }
13 float function1(float A, int B) {
14     return A / B;
15 }
16 int function1(int A, int B, int C) {
17     return A + B + C;
18 }
```

Попытка перегрузки функции, отличающейся только типом возвращаемого значения

Перегрузка функции, отличающейся типом и/или количеством параметров

1 error C2556: 'int function1(int,int)' : overloaded function differs only by return type from 'bool function1(int,int)'

2 error C2371: 'function1' : redefinition; different basic types

3 error C2556: 'float function1(int,int)' : overloaded function differs only by return type from 'bool function1(int,int)'

ПАРАМЕТРЫ ФУНКЦИЙ

Параметры функций

Параметрами функций могут быть

- Переменные
- Ссылки
- Указатели

1. Если **параметр – переменная**, то в функцию передается ее копия. Изменение значения параметра в функции не возвращается в вызывающую функцию
2. Если **параметр – ссылка**, функция работает с тем участком памяти, на который она ссылается. Изменение значения параметра в функции «возвращается» в вызывающую функцию
3. Если **параметр – указатель**, функция работает с тем участком памяти, на который он указывает. Изменение значения параметра в функции «возвращается» в вызывающую функцию

Отличие указателя и ссылки – указатель надо разыменовывать в теле функции, ссылку – не надо

Параметры - переменные

```
#include<iostream>
using namespace std;

int f1(int a, int b)
{  // вывод
        a = 20; b = 10;
        return a - b;
   // вывод
}

void main()
{
   int a = 10, b = 5, c = 0;
   // вывод
    c = f1(a, b);
   // вывод
   system("pause");
   return;
}
```

Вывода на печать в примере слева нет, не хватает места на слайде.

В коде вывод на печать был

```
main begin a = 10  b = 5  c = 0

in f1 begin a = 10  b= 5  a - b = 5
in f1 after a = 20  b= 10  a - b = 10
main after f1 a = 10  b = 5  c = 10
```

Параметры - ссылки

```
#include<iostream>
using namespace std;

int f2(int &a, int& b)
{
        a = 20; b = 10;
        return a - b;
}

void main()
{
    int a = 10, b = 5, c = 0;
    int &sa = a;
     c = f2(sa, b);
    system("pause");
    return;
}
```

ссылка всегда разыменована

```
main begin a = 10  b = 5  c = 0

in f2 begin a = 10  b= 5  a - b = 5
in f2 after a = 20  b= 10  a - b = 10
main after f2 a= 20  b=10  c = 10
```

При вызове указываем любое имя переменной:

- свое, родное (a) или
- ссылку-псевдоним (sa)

Параметры - указатели

```
#include<iostream>
using namespace std;

int f3(int *a, int* b)
{
        *a = 20; *b = 40;
        return *a - *b;
}

void main()
{
    int a = 10, b = 5, c = 0;
    int *pa = &a;
     c = f3(pa, &b);
    system("pause");
    return;
}
```

В теле функции указатели разыменовываем

```
main begin a = 10  b = 5  c = 0

in f3 begin a = 10  b= 5  a - b = 5
in f3 after a = 20  b= 40  a - b = -20
main after f3 a = 20  b= 40  c = -20
```

При вызове используем

либо указатель (pa),

либо адрес (&b)

Параметр функции - указатель

```
#include <iostream>
int f3(int *a,int *b);
using namespace std;
void main()
{      int a = 8, b = 2, c = 0;
       int *pa = &a,*pb = &b;
            cout<<"c="<<c<<"  a="<<a<<"  b=" <<b<<endl;
        c = f3(pa, &b);
            cout<<"c="<<c<<"  a="<<a<<"  b="<<b<<endl;
          system("pause");
       return;}
int f3(int *a, int *b)
{ *a = 9; *b = 12; return *a -*b; }
```

Прототип функции

Указатели и их инициация

При вызове используется указатель или адрес

Описание функции

Параметр функции -указатель

```
#include <iostream>
int f3(int *a,int *b);
using namespace std;
void main()
{       int a = 8, b = 2, c = 0;
        int *pa = &a,*pb = &b;
        cout<<"c="<<c<<"  a="<<a<<"  b="<<b<<endl;
         c = f3(pa, &b);
        cout<<"c="<<c<<"  a="<<a<<"b="<<b<<endl;
         system("pause");
return;}
int f3(int *a, int *b)
{ *a = 9; *b = 12; return *a -*b; }
```

вывод на печать:

```
c=0  a=8  b=2
c=-3 a=9  b=12
```

Как решать

Для решения удобно использовать таблицу трассировки.

Столбцы – переменные из вызывающей функции и переменные в функции.

Строки – операторы функции

```
#include <iostream>
int f3(int *a,int *b);
using namespace std;
void main()
{         int a = 8, b = 2, c = 0;
          int *pa = &a,*pb = &b;
          c = f3(pa, &b);
          system("pause");
return;}
int f3(int *a, int *b)
{ *a = 9; *b = 12; return *a -*b; }
```

операторы	main			f3	
	a	b	c	*a=pa	*b=&b
a=8;…	8	2	0		
f3()					
*a = 9;	9				
*b = 12		12			
*a -*b			-3		

Примеры функций

```
#include <iostream>
#include <Windows.h>
int f1(int a,int b);
//прототип
int f3(int *a, int *b);
//прототип
using namespace std;
int f2(int &a, int& b)
{
cout<<"f2 a+b=" <<
       a+b << "\n";
a = 9; b = 12;
return a + b;
}
```

```
void main()
{
      int a = 8, b = 2, c;
cout<<"начало a="<<a<<"b="<<b<<endl;
      c = f1(a, b);
cout<<"после f1 a="
<<a<<"b="<<b<<endl;
      c = f2(a, b);
cout<<"после f2 a="<<a<<"  b=" << b
<< endl;
       c = f3(&a, &b);
cout<<" после f3 a = " << a << "  b=«
<< b << endl;
 system("pause");
```

f2: параметр - ссылка (адрес)
В теле используются имена переменных

Пример: функция f1

```
int f1(int a, int b)
{
cout<<"f1 a – b = "<< a – b <<endl;
a = 9; b = 12;
return a - b;
}
```

f1 : параметры - значения

операторы	main			f1	
	a	b	c		
a=8;…	8	2	0		
f1(a, b)				8	2
a = 9;	8	2		9	
b = 12	8	2			12
a - b	8	2	-3		

Пример: функция f2

```
int f2(int &a, int& b)
{
  cout<<"f2 a+b=" << a+b << "\n";
  a = 9; b = 12;
  return a + b;
}
```

f2 : параметры - ссылки

операторы	main			f2	
	a	b	c	&a	&b
a=8;…	8	2	0		
c=f2(a,b);				(ссылка на a) 8	(ссылка на b) 2
a = 9;	9			9	
b = 12		12			12
a - b			-3		

Пример: функция f3

```
int f3(int *a, int *b)
{
cout<<"f3 a - b="<< *a - *b << endl;
*a = 20; *b = 40;
return *a - *b;
}
```

f3 : параметры - указатели (адреса)
В теле используется разадресация (разыменование)
ВАЖНО!!! Передаются значения указателей

операторы	main			f3	
	a	b	c	*a	*b
a=8;…	8	2	0		
c=f3(&a,&b);				адрес a	адрес b
*a = 20;	20				
*b = 40;		40			
*a - *b;			-20		

Хитрый пример

```
#include <iostream>
int f1(int*, int, int&);
using namespace std;
void main()
{int a = 2, b = 3, c = 5, d = 0;
   cout<<"before"<<"  a="<<a<<"
b="<<b<<"  c="<< c <<"   d="<<d<<endl;
   d = f1(&a, b, c);
   cout<<"after "<<"  a="<<a<<"
b="<<b<<"  c="<< c << "   d="<<d<<endl;
  d = f1(&b, c, a);
   cout<<"after "<<"  a="<<a<<"
b="<<b<<"  c="<< c << "   d="<<d<<endl;
  system("pause");
}
```

```
int f1(int* a,int
   b, int& c)
{
     *a = 7 + *a;
      b = 8 + c;
      c = 9 + b;
 return *a + b + c;
}
```

Ответ к хитрому примеру

```
{int a=2, b=3,c=5, d=0;
// вывод
      d = f1(&a, b, c);
// вывод
     d = f1(&b, c, a);
// вывод
}
```

```
int f1(int* a,int b,int&
c)
{     *a = 7 + *a;
      b = 8 + c;
      c = 9 + b;
 return *a + b + c;}
```

операторы	main				f1		
	a	b	c	d	*a	b	&c
a=2;…	2	3	5	0			
d=f1(&a,b,c);					адрес a	3	ссылка на c
*a = 7 + *a;	9						
b = 8 + c;						8+5=13	
c = 9 + b;			22				
*a + b + c;				44			
	9	3	22	44			

Ответ к хитрому примеру

```
{int a=2, b=3,c=5, d=0;
// вывод
      d = f1(&a, b, c);
// вывод
     d = f1(&b, c, a);
// вывод
}
```

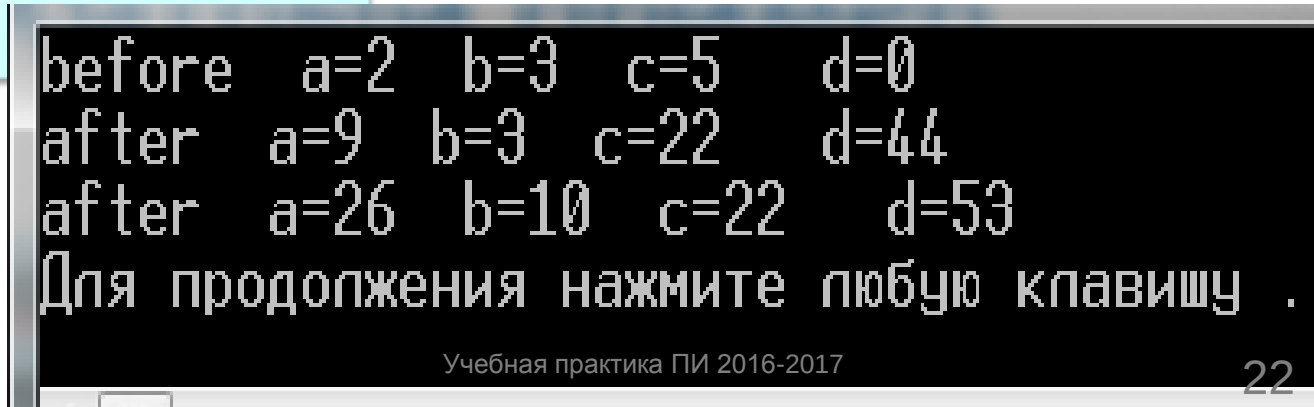
```
int f1(int* a,int b,int&
c)
{     *a = 7 + *a;
      b = 8 + c;
      c = 9 + b;
 return *a + b + c;}
```

операторы	main				f1		
	a	b	c	d	*a	b	&c
	9	3	22	44			
d=f1(&b,c,a);					адрес b	22	ссылка на a
*a = 7 + *a;		10			7+3=10		
b = 8 + c;						8+9=17	
c = 9 + b;	26						9+17=26
*a + b + c;				10+17+26=53			
	26	10	22	53			

Ответ к хитрому примеру

```
#include <iostream>
int f1(int*, int, int&);
using namespace std;
void main()
{int a=2, b=3,c=5, d=0;
// вывод
      d = f1(&a, b, c);
// вывод
     d = f1(&b, c, a);
// вывод
}
```

```
int f1(int* a,int
b,int& c)
{
     *a = 7 + *a;
      b = 8 + c;
      c = 9 + b;
 return *a + b + c;
}
```



ПАРАМЕТРЫ - МАССИВЫ

Массив – параметр функции

Имя массива без индекса – это адрес нулевого элемента.

```
int a[10], *pmas;
pmas = a;
```

Никакой формальный параметр функции не принимает массив целиком.

Если массив используется в качестве аргумента функции, то функции передается адрес этого массива.

При вызове функции содержимое массива может измениться

Параметры - массивы

Заголовок функции

```
void d1(int mas[10]);
void d2(int mas[]);
void d3(int *mas);
```

Вызовы функции

```
int t[10], *pt=t;
d1(t); d1(pt);
d2(t); d2(pt);
d3(t); d3(pt);
```

Так можно

1. Компилятор автоматически преобразует формальный параметр - массив в указатель
2. Указатель можно индексировать с помощью квадратных скобок [ ]

Параметры - массивы

```
#include <iostream>
void d1(int t[10]);
void d2(int t[]);
void d3(int *pt);
using namespace std;
void main()
{  int t[10],*pt=t;
          d1(t);
for (int i=0; i<10;i++)
     cout<<" "<<t[i];cout<<endl;
          d2(t);
for (int i=0; i<10;i++)
     cout<<" "<<t[i];cout<<endl;
          d3(pt);
for (int i=0; i<10;i++)
     cout<<" "<<t[i];cout<<endl;
          d3(t);
for (int i=0; i<10;i++)
     cout<<" "<<t[i];cout<<endl;
cin>>b;  return;}
```

```
void d1(int t[10])
{ for (int i=0; i<10;i++)
{t[i]=i; cout<<" "<<t[i];}
cout<<endl;}

void d2(int t[])
{ for (int i=0; i<10;i++)
{t[i]=2*i; cout<<" "<<t[i];}
cout<<endl;}

void d3(int *t)
{ for (int i=0; i<10;i++)
{t[i]=3*i; cout<<" "<<t[i];}
cout<<endl;}
```

ПАРАМЕТРЫ - ФУНКЦИИ

Функция как параметр

Как и в C#, можно передавать функцию как параметр метода

```
int return_first(int a, int b) {
    return a;
}

int return_second(int a, int b) {
    return b;
}

int i_need_a_function(int (*passed_function)(int, int)) {
    return passed_function(1, 2);
}
...
i_need_a_function(return_first);
i_need_a_function(return_second);
```

Функция как параметр

В списке формальных параметров в заголовке функции – ссылка на функцию

```
void sort_ints(int* items,int num_items,int (*cmpfunc)(int,int))
{
	for(int i = 0;i < num_items - 1; i++)
	{
		for(int j = i + 1; j < num_items; j++)
			if (cmpfunc(items[j - 1], items[j]))
			{
				int t = items[j - 1];
				items[j - 1] = items[j];
				items[j] = t;
			}
	}
	return;
}
```

компаратор

Функция как параметр

```
#include<iostream>
using namespace std;
/* callback-функция */
int compare_function(int A, int B) {
  return A < B;
}
/* объявление функции сортировки */
void sort_ints(int* items,int num_items,int (*cmpfunc)(int,int));
int main(void) {
      setlocale(LC_ALL,"Russian");
    int items[] = {4, 3, 1, 2};
    sort_ints(items, sizeof(items)/sizeof(int), compare_function);
      for(int i=0; i<sizeof(items)/sizeof(int); i++)
            cout<<items[i]<<' ';
      cout<<'\n';
      system("pause");
    return 0;
}
```

Описание функции

Функция - параметр

При вызове подставляем имя функции

Домашнее задание

1. Разработать два хитрых тестовых задания на понимание функций и передачу в них параметров (переменные, указатели и ссылки). Обязательно представить правильный ответ.
2. Подробности требований к заданиям будут