Лекція 4. Функції

Функція – іменований блок програмного коду, до якого можна звертатись в програмі через ім'я

Оголошення функції – вказується прототип функції і, в фігурних скобках, програмний код

Прототип функції – тип результату, ім'я функції, список аргументів (в круглих скобках зі вказуванням типу кожного аргументу)

```
тип_результау ім'я_функції(аргументи){
//код функції
}
```

Функція повертає в якості результату значення виразу після інструкції return.

Якщо функція не повертає результат, в якості типу результату вказують ключове слово void.

Приклад:

```
double MyFunc(double z){
double t;
t=z*z;
return  t;}
```

Можна простіше:

```
double MyFunc(double z){
return z*z;}
```

Приклад 4.1. Сума натуральних чисел

```
#include<iostream>
using namespace std;
int msum(int n){
int s=0;
for(int i=1;i<=n;i++) s+=i;
return s;}
int main(){
int n;
cout<<"Enter n = ";
cin>>n;
cout<<"Sum is "<<msum(n)<<"\n";
return 0;}
```

Приклад 4.2. Функція не повертає результат

```
#include<iostream>
using namespace std;
void msum2(){
int n;
cout<<"Enter n = ";
cin>>n;
int s=0;
for(int i=1;i<=n;i++) s+=i;
cout<<"Sum is "<<s<<"\n";}
int main(){
msum2();
return 0;}
```

Приклад 4.3. Функція не повертає результат, але містить інструкцію return

```
void InvFunc(double z){
if(z==0){
cout<<"Division by zero!"<<endl;
return;}
double x;
x=1/z;
cout<<"1/z ="<<x<<endl;}
```

ЗВЕРНІТЬ УВАГУ!!!

Приклад 4.4. Функція описана в кінці програми

```
#include <iostream>
using namespace std;
//Прототип функції InvFunc():
void InvFunc(double z);
//Функція InvFunc() використана в програмі:
int main(){
double s;
cout<<" Enter number: ";
cin>>s;
InvFunc(s);
return 0;}
//Опис функції InvFunc():
void InvFunc(double z){
if(z==0){
cout<<"Division by zero!"<<endl;
return;}
double x;
x=1/z;
cout<<"1/z ="<<x<<endl;}
```

Механізми передачі аргументів функціям

В C++ існує два механізми передачі аргументів:

Через посилання

За значенням

Через посилання

В функцію передається копія змінної-аргументу

В функцію передається безпосередньо змінна-аргумент

За умовчанням використовується механізм передачі аргументів за значенням

Для передачі аргументу за посилання перед іменем аргументу в прототипі функції вказують оператор &

Різницю в механізмах передачі слід враховувати тоді, коли в функції міняються її аргументи

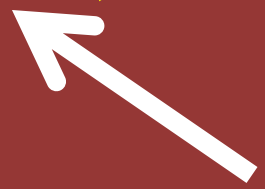
Приклад 4.5. Передача аргументів

За значенням!!!

```
#include <iostream>
using namespace std;
// Аргумент передається
// за значенням:
int incr(int m){
m=m+1;
return m;}
int main(){
int n=5;
cout<<"n = "<<incr(n)<<endl;
 // n = 6
cout<<"n = "<<n<<endl;
// n = 5
return 0;}
```

За посиланням!!!

```
#include <iostream>
using namespace std;
// Аргумент передається
// за посиланням:
int incr(int &m){
m=m+1;
return m;}
int main(){
int n=5;
cout<<"n = "<<incr(n)<<endl;
 // n = 6
cout<<"n = "<<n<<endl;
// n = 6
return 0;}
```

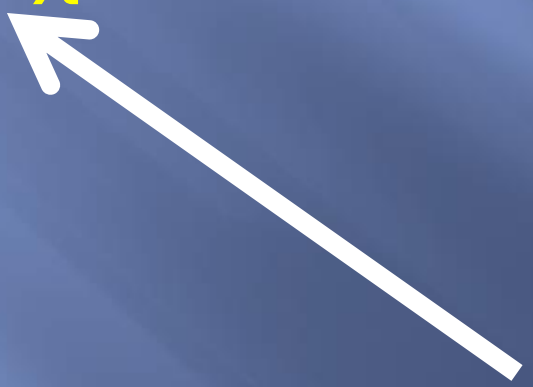


Приклад 4.6. Передача аргументом вказівника

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

//Аргументом є вказівник:
int incr(int *m){
*m=*m+1;
return *m;}

int main(){
int n=5;
cout<<"n = "<<incr(&n)<<endl;  // n = 6
cout<<"n = "<<n<<endl;         // n = 6
return 0;}
```



ЗВЕРНІТЬ УВАГУ!!!

Приклад 4.7. Передача аргументом масиву

Варіант 1 (явно вказаний розмір)

```
void show(int n[5]){
for(int i=0;i<5;i++)
cout<<"n["<<i<<"]=";
cout<<n[i]<<endl;}
int main(){
int n[5]={1,2,3,4,5};
show(n);  return 0;}
```

Варіант 2 (розмір не вказаний)

```
void show(int n[]){
for(int i=0;i<5;i++)
cout<<"n["<<i<<"]=";
cout<<n[i]<<endl;}
int main(){
int n[5]={1,2,3,4,5};
show(n);   return 0;}
```

Варіант 3 (через вказівник)

```
void show(int *n,int m){
for(int i=0;i<m;i++)
cout<<"n["<<i<<"]=";
cout<<n[i]<<endl;}
int main(){
int n[5]={1,2,3,4,5};
show(n,5);    show(n,3);
return 0;}
```

n[0]=1
n[1]=2
n[2]=3
n[3]=4
n[4]=5

n[0]=1
n[1]=2
n[2]=3
n[3]=4
n[4]=5
n[0]=1
n[1]=2
n[2]=3

Приклад 4.8. Передача аргументом двомірного масиву

```
void show2(int n[][3],int size){
int i,j;
        for(i=0;i<size;i++){
                for(j=0;j<3;j++){
                cout<<n[i][j]<<" ";}
        cout<<"\n";}
}
int main(){
int n[][3]={{1,2,3},
                {4,5,6}};
show2(n,2);
return 0;}
```

ЗВЕРНІТЬ УВАГУ!!!

Результат:

```
1 2 3
4 5 6
```

Приклад 4.9. Передача аргументом текстового рядка

ЗВЕРНІТЬ УВАГУ!!!

Результат:

```
Length is 16
```

```
int length(char *str){
for(int s=0; *str; s++, str++);
return s;}

int main(){
char str[20]="This is a string";
cout<<"Length is "<<length(str)<<endl;
return 0;}
```

Аргументи функції main()

У функції main() можуть бути аргументи:

1. Кількість параметрів командного рядка
2. Масив з текстовими значеннями цих параметрів

```
int main(int size,char *str[]){
int i;
for(i=0;i<size;i++)
cout<<i+1<<"-th argument is: "<<str[i]<<endl;
return 0;}
```

Командний рядок: mytest.exe first second 101

Результат:

```
1-th argument is: C:\Docs\mytest.exe
2-th argument is: first
3-th argument is: second
4-th argument is: 101
```

Аргументи за умовчанням

Для аргументів функцій можна вказувати значення за умовчанням: якщо аргумент явно не переданий функції, використовується значення за умовчанням. Аргументи зі значенням за умовчанням в списку аргументів в прототипі функції вказуються останніми. При оголошенні функції значення за умовчанням присвоюється аргументу в прототипі функції:

тип ім'я_функції(тип аргумену=значення){код функції}

```
void showX(int x=0){
cout<<"x = "<<x<<endl;}
void showXYZ(int x, int y=1, int z=2);
int main(){
showX(3);   showX();
showXYZ(4,5,6); showXYZ(7,8); showXYZ(9);
return 0;}
void showXYZ(int x ,int y, int z){
cout<<"x = "<<x<<" "; cout<<"y = "<<y<<" ";
cout<<"z = "<<z<<endl;}
```

ЗВЕРНІТЬ УВАГУ!!!

Приклад 4.10. Результат функції - вказівник

```
// Результат функції  - вказівник  на більшу змінну
int *mpoint(int &n,int &m){
if(n>m) return &n;
else return &m;}

int main(){
int n=3,m=5;
int *p;
p=mpoint(n,m);
(*p)++;
cout<<"n ="<<n<<endl;
cout<<"m ="<<m<<endl;
return 0;}
```

Результат виконання:

n =3

m =6

Вказівник на функцію

На функцію можна створювати вказівник. Значенням вказівника на функцію є адреса пам'яті, в яку записаний код функції. Виклик функції реалізується через адресу. Ця адреса називається точкою входу в функцію.

Ім'я функції (без круглих дужок) є вказівником на функцію!

Вказівник на функцію оголошується так:

1. Тип результату функції
2. В круглих дужках оператор * і ім'я вказівника
3. В круглих дужках список типів аргументів функції

Приклад оголошення вказівника на функцію:

```
int (*p)(double, char)
```

Приклад 4.11. Вказівник на функцію

```
double sqr(double x){
return x*x;}
double cube(double x){
return x*x*x;}
// Другий аргумент - вказівник на функцію:
void myfunc(double x,double (*f)(double)){cout<<f(x)<<endl;}
int main(){double z=5.0;
// Вказівник на функцію:
double (*p)(double);
// Вказівнику присвоюється значення:
p=cube;
// Вказівник і ім'я функції:
myfunc(z,sqr);
myfunc(z,p);
cout<<p(z)<<endl;
//Адреса функції:
cout<<sqr<<endl; cout<<cube<<endl; cout<<p<<endl; return 0;}
```

Результат виконання:

```
25
125
125
00401195
0040128F
0040128F
```

Рекурсія

Рекурсія - виклик функції в програмному коді, що визначає цю функцію. Рекурсія може бути пряма (безпосередній виклик) і непряма (виклик через інші функції).

Факторіал числа (n!=n(n-1)(n-2)...1)

```
int factorial(int n){
if(n==1) return 1;
else return n*factorial(n-1);
}
```

Числа Фібоначчі (1,1,2,3,5,8,13,21,34,55 ...)

```
int fib(int n){
if(n==1||n==2) return 1;
else return fib(n-1)+fib(n-2);
}
```

Перевантаження функцій

Перевантаження функції - створення різних варіантів функції з однаковими назвами але різними прототипами.

Різні варіанти перевантаженої функції відрізняються кількістю і типом аргументів, а також типом результату.

При виклику функції варіант програмного коду для перевантаженої функції визначається в контексті команди виклику функції (наприклад, по кількості аргументів).

Приклад 4.12. Перевантаження функцій

```
//Перший варіант функції:
void showArgs(double x){cout<<"Double-number "<<x<<endl;}
//Другий варіант функції:
void showArgs(double x,double y){
cout<<"Double-numbers "<<x<<" and "<<y<<endl;}
//Третій варіант функції:
void showArgs(char s){cout<<"Symbol "<<s<<endl;}
//Четвертий варіант функції:
int showArgs(int n){return n;}
int main(){
int n=3; double x=2.5,y=5.1; char s='w';
//Перший варіант функції:
showArgs(x);
//Другий варіант функції:
showArgs(x,y);
//Третій варіант функції:
showArgs(s);
//Четвертий варіант функції:
cout<<"Int-number "<<showArgs(n)<<endl; return 0;}
```

Результат виконання:

```
Double-number 2.5
Double-numbers 2.5 and 5.1
Symbol w
Int-number 3
```

Резюме

1. Функція - іменований блок програмного коду, який може викликатись через ім'я.

2. Функції можуть передаватись аргументи. Функція може повертати, а може не повертати результат. Результатом функції може бути все, крім масиву.

3. Існує два механізми передачі аргументів функції - за значенням (за умовчанням) та за посиланням.

4. Аргументи функції можуть мати значення за умовчанням.

5. Функції можна перевантажувати. Перевантажені функції мають однакові назви, але різні прототипи.