



ФУНКЦІЇ. ПРОТОТИПИ ФУНКЦІЙ.

ПАРАМЕТРИ ТА ЇХ ВИДИ. РЕКУРСІЯ

Модульне програмування

- увесь проект має бути розбитий на модулі з одним входом і одним виходом;
- логіка алгоритму та програми має допускати лише три основні структури: послідовне виконання, розгалуження та повторення. Неприпустимий оператор передачі керування в будь-яке місце програми;
- при розробці документація має створюватися одночасно із програмуванням, у вигляді коментарів до програми.

Ціллю модульного програмування є підвищення надійності програм, прискорення і полегшення їх створення.

Підпрограми пов'язуються з основною програмою параметрами.

Програма розділяється на дрібніші одиниці – *процедури і функції*.

Поняття функції

Функція – незалежна частину програми, яка має власне ім'я і може викликатися з інших частин програми, оперувати даними та повертати результат.

Кожна C++-програма має принаймні одну функцію – функцію *main()*

Існує два основні види функцій:

- *стандартні* (вбудовані)
- *функції користувача*.

Функції можуть запускатися (викликатися) в коді програми будь-яку кількість разів. Значення, які передаються функції, називаються її аргументами. Їх типи мають відповідати типам параметрів у заголовку функції.

Будь-яка функція повинна бути оголошена і визначена

Будь-яка функція повинна бути оголошена і визначена. Оголошення функції повідомляє компілятор як її викликати, а визначення – описує її тіло.

Оголошення функції описується за допомогою *прототипу* і має бути здійснене перед її викликом.

Прототип функції має такий формат:

`<тип_результату> <ім'я_функції> ([<список_параметрів>]);`

Список параметрів описується так:

```
<тип_параметра_1> [<параметр_1>]
[, <тип_параметра_2> [<параметр_2>]
...
[, <тип_параметра_N> [<параметр_N>]
```

```
int sum(int a, int b); // лише тип, назва та параметри
//немає реалізації – тіла функції
```

Визначення функції має наступний формат

```
<тип_результату> <ім'я_функції> ([<список_параметрів>])  
{  
  <тіло функції>  
  return <результат> //для всіх окрім void  
}
```

Наприклад

```
int sum(int a, int b)  
{  
  return a + b; //тип результату такий самий, як і тип функції  
}
```

Виконання функції починається тоді, коли в тексті програми зустрічається оператор виклику цієї функції.

Значення, які передаються функції, називаються її *аргументами*.

Їх типи мають відповідати типам параметрів у заголовку функції.

Виклик функції завжди позначається ім'ям функції та круглими дужками, в яких стоять змінні (константи, вирази), значення яких передаються (підставляються) замість аргументів.

```
int main()
{
    int i = sum(10, 7);
    cout << "The value - " << i << endl; // 17
}
```

У випадку, коли функція не має аргументів, круглі дужки все одно є обов'язковими.

Функція `main()` має містити лише виклики функцій, які виконуватимуть усю роботу програми. Кожна функція має вирішувати лише одну конкретну задачу

Аргументи функції

Значення змінних, які передаються у функцію при виклику, називаються *фактичними параметрами* або *аргументами*, а змінні, оголошені у функції, що приймають значення аргументів, є локальними змінними функції, які створюються при вході в неї та знищуються при виході з неї.

Функція може повертати або не повертати результат.

- Функція не повертає значення і не містить аргументів:

```
void test1(void);    // еквівалентно void test1();
```

- Функція не повертає значення, але містить два аргументи типу *int* та *double*:

```
void test2(int, double);
```

- Функція повертає значення типу *int* і не містить аргументів:

```
int test3(); // еквівалентно int test3(void);
```

- Функція повертає значення типу *double* і містить два аргументи типу *int*:

```
double test4(int, int);
```

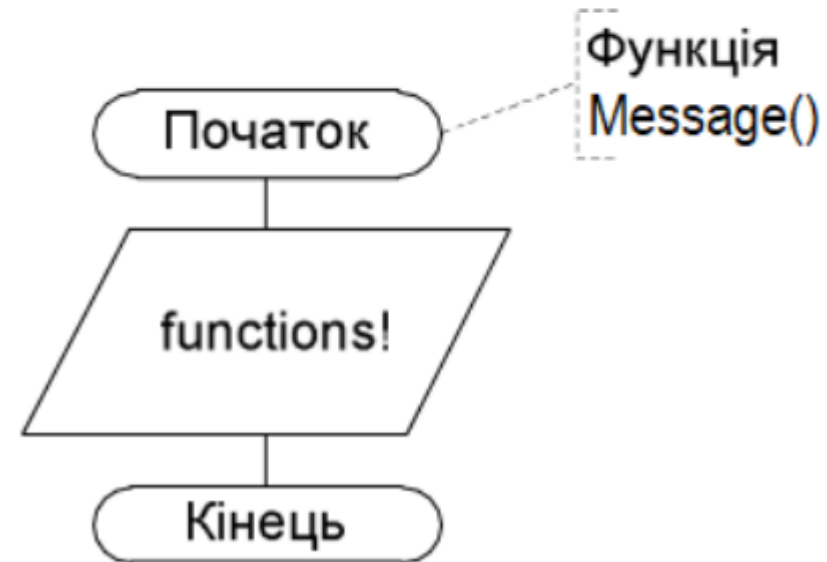
```
#include <iomanip>
#include <iostream>

using namespace std;

// оголошення прототипу функції
void Message();

// головна функція
void main() {
// виклик функції
Message();

}
// визначення функції Show()
void Message() {
cout << "Functions!";
}
```



Завершення виконання та повернення результату функції

Виконання функції завершується після виконання команди `return` або при досягненні фігурної дужки `}`, яка закриває блок функції.

Якщо функція повертає результат, то тип результату, вказаний перед іменем функції, має відповідати типу виразу, вказаного після ключового слова `return`.

Якщо функція не повертає результату, то замість типу результату в заголовку функції слід вказати ключове слово `void`.

```
int main()
{
    ...
    return 0;
}
```

Наприклад, команда виклику `void`-функції, визначеної так:

```
void printSquare(int n) // визначення функції
{
    cout << n * n << endl;
}
```

виглядає так:

```
printSquare(5); // виклик функції в окремій команді
```

Типові помилки при використанні функцій

- 1) не повернути значення (забули команду `return <вираз>;`), якщо тип результату функції – не `void`;
- 2) спроба повернути результат командою `return <вираз>;`, якщо тип функції – `void`;
- 3) список параметрів функції виду

```
int a, b      // помилка
```

Потрібно так:

```
int a, int b  // вірно
```

- 4) в описі функції символ `;` «крапка з комою» після `)` – дужки, яка закриває список параметрів, – це помилка:

```
<тип_результату> <ім'я_функції>([<список_параметрів_функції>]); // помилка!  
{  
    ...  
}
```

```
#include <iomanip>
#include <iostream>
using namespace std;
// оголошення прототипу
double index_m(double, int);
//головна функція
int main() {
double m;
int h ;// локальні змінні для функції main()
cout << "Input m (kg) and h (cm)" << endl;
cin >> m;
cin >> h;
cout << fixed << setprecision(2);
cout << "Index = " << index_m(m, h)<< endl;
system("pause");
return 0;
}
// визначення функції middle()
double index_m(double a, int b) {
double index; // a, b, index - локальні змінні для функції
index = (a / (b*b))*10000;
return index; // повернення значення змінної
}
```

C:\D:\my_work\test_ap\x64\Debug\test_ap.exe

Input m (kg) and h (cm)

56

163

Index = 21.08

Press any key to continue . . .

Особливості використання функцій

Є два способи, якими можна передати інформацію у функцію :

- 1) *передавати значення аргументів;*
- 2) *передавати адреси аргументів.*

В мові C++ інформацію у функцію можна передавати за допомогою параметрів, які поділяються на наступні три види:

- *параметри-значення ;*
- *параметри-вказівники ;*
- *параметри-посилання .*

Параметри-значення

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

//реалізація функції
int sum(int a, int b) // параметри-значення
{
    return a + b;
}

//головна функція
int main()
{
    int x = 1;
    int y = 2;
    cout << sum(x, y) << endl; //виклик функції
    return 0;
}
```

Параметр-значення – це копія (інший екземпляр в пам'яті) аргументу.

Параметри-значення можна використовувати лише для передавання інформації у функцію, бо зміна цих параметрів в тілі функції ніяк не вплине на аргументи.

Параметри-посилання

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

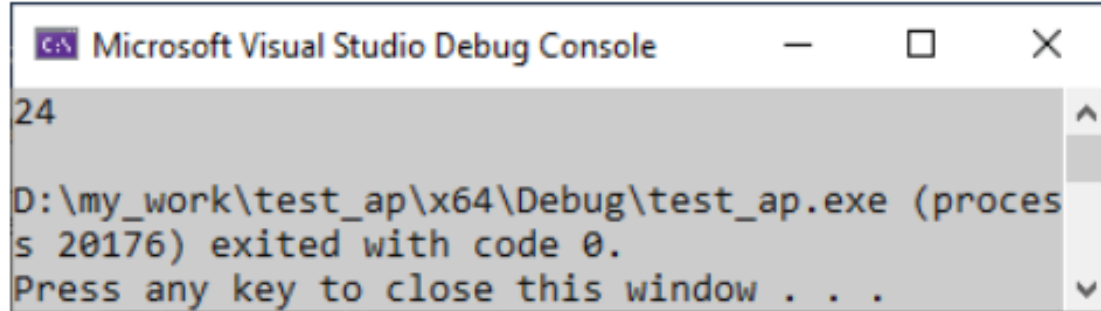
//реалізація функції
int sum(int &a, int &b) // параметри-значення
{
    return a + b;
}

//головна функція
int main()
{
    int x = 10;
    int y = 2;
    cout << sum(x, y) << endl; //виклик функції
    return 0;
}
```

Параметр-посилання можна використовувати як для передавання інформації у функцію, так і для передавання інформації із функції, бо зміна цього параметру в тілі функції одразу приводить до відповідної зміни аргументу

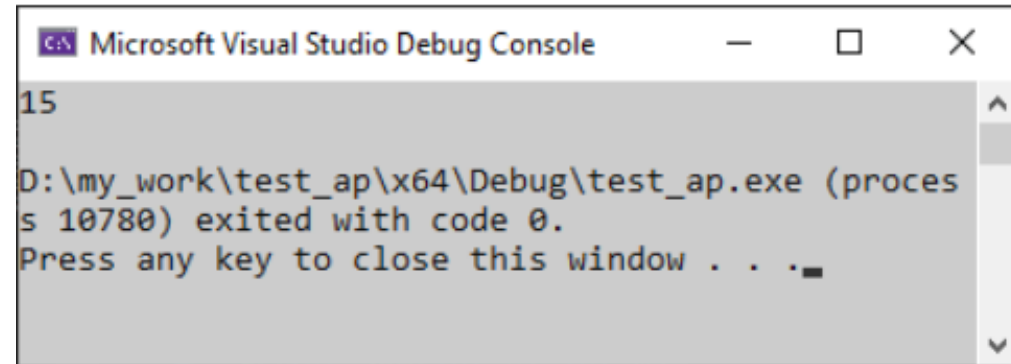



```
#include <iostream>
using namespace std;
int fact(int n)
{
    int P = 1;
    for (int i = 1; i <= n; i++)
        P *= i;
    return P;
}
int main()
{
    int k = fact(4);
    cout << k << endl;
    return 0;
}
```



```
Microsoft Visual Studio Debug Console
24
D:\my_work\test_ap\x64\Debug\test_ap.exe (process 20176) exited with code 0.
Press any key to close this window . . .
```

```
#include <iostream>
using namespace std;
//рекурсивне обчислення суми чисел від n до m
int sum(int n, int m)
{
    int s = 0;
    if (n == m) s = n;
    else
        s = sum(n,m-1)+m;
    return s;
}
//головна функція
int main()
{
    int k = sum(1,5); //виклик функції для чисел від 1 до 5
    cout << k << endl;
    return 0;
}
```



```
Microsoft Visual Studio Debug Console
15
D:\my_work\test_ap\x64\Debug\test_ap.exe (process 10780) exited with code 0.
Press any key to close this window . . .
```

Рекурсія

Рекурсія – це спосіб організації обчислювального процесу, при якому функція звертається сама до себе.

Розрізняють *пряму та непряму* рекурсію.

Пряма рекурсія полягає в тому, що функція у своєму тілі викликає сама себе, *непряма* – коли дві чи більше функцій викликають одна одну.

Максимальна кількість копій рекурсивної функції, що одночасно може знаходитися в пам'яті комп'ютера, називається *глибиною рекурсії*.

Рекурсивна функція обов'язково повинна містити *умову завершення*

Параметри зі значеннями за замовчуванням



можуть пропускатися у виклику функції.

приклад прототипу функції з параметром за замовчуванням:

```
int sum(int a, int b = 0); // параметр b має значення за замовчуванням 0
```

Варіант виклику функції

```
int x = 10;  
int y = 2;  
sum(x, y); //результат 12  
sum(x); //результат 10
```

Параметри-вказівники

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

//реалізація функції
int sum(int *a, int *b) // параметри-значення
{
    return *a + *b;
}

//головна функція
int main()
{
    int x = 10; int* x_ = &x;
    int y = 2;  int* y_ = &y;
    cout << sum(x_, y_) << endl; //виклик функції
    return 0;
}
```

Передавання інформації у функцію за допомогою параметрів-вказівників – застарілий спосіб, який в C++ не використовується.

Дякую за увагу

Лектор:
кандидат фіз.-мат. наук, доцент
Шаклеїна Ірина
iryna.o.shakleina@lpnu.ua
кафедра ІСМ, ІКНІ