Лекція №8. Функції

ПРОГРАМУВАННЯ ТА ПРИКЛАДНІ ІНФОРМАЦІЙНІ СИСТЕМИ

Функції

Функція — це іменована логічно завершена сукупність оголошень і операторів, призначених для виконання певної задачі

Кожна програма у своєму складі повинна мати головну функцію **main()**. Саме функція **main()** забезпечує створення точки входу в об’єктний модуль.

Крім функції **main()**, в програму може входити будь-яка кількість функцій. Кожна функція по відношенню до іншої є зовнішньою. Для того, щоб функція була доступна, необхідно, щоб до її виклику про неї було відомо компілятору.

Функція користувача

Функція користувача - це поіменована група команд, яка оголошена у файлі заголовків (або в основній програмі) та описана у модулі (в основній програмі). До функції можна звертатись (викликати) з будь-якого місця програми необхідну кількість разів.

Оголошення функції (прототип, заголовок) задає ім'я функції, тип значення, що повертає функція (якщо воно є), а також імена та типи аргументів, які можуть передаватися як у функцію, так і з неї.

Визначення функції — це задання способу виконання операцій.

Компоненти

Із поняттям функції у мові C++ пов’язано три наступних компоненти:

- опис функції;
- прототип;
- виклик функції.

Опис функції

Опис функції складається з двох частин: заголовка і тіла. Опис функції має наступну форму запису:

```
/* заголовок функції*/
[тип_ результату] <ім'я>([список_параметрів]) {
    // тіло функції
    опис даних;
    оператори;
    [return] [вираз]; // якщо не void
}
```

Тип результату

Тип результату — будь-який базовий або раніше описаний тип значення (за винятком масиву і функції), що повертається функцією (необов'язковий параметр). У випадку відсутності специфікатора типу передбачається, що функція повертає ціле значення (**int**). Якщо функція не повертає ніякого значення, то на місці типу записується специфікатор **void**. Якщо результат повертається функцією, то в тілі функції є необхідним оператор return вираз;, де вираз формує значення, що співпадає з типом ре зультату

Список параметрів

Список параметрів - визначає кількість, тип і порядок проходження переданих у функцію вхідних аргументів, які розділяються комою («**,**»). У випадку, коли параметри відсутні, дужки залишаються порожніми або містять ключове слово (**void**). Формальні параметри функції локалізовані в ній і недоступні для будь-яких інших функцій.

Список параметрів має такий вигляд:

```
([const] тип 1 [параметр 1], [const] тип 2 [параметр 2])
```

У списку формальних аргументів для кожного параметра треба вказати його тип **(не можна групувати параметри одного типу, вказавши їх тип один раз)**.

Тіло функції

Тіло функції може складатися з описів **змінних** і **операторів**. **Змінні**, що використовуються при виконанні функції, можуть бути **глобальні** і **локальні**. Змінні, що описані (визначені) за межами функції, називають **глобальними**. За допомогою глобальних параметрів можна передавати дані у функцію, не включаючи ці змінні до складу формальних параметрів. У тілі функції їх можна змінювати і потім отримані значення передавати в інші функції.

Змінні, що описані у тілі функції, називаються **локальними** або **автоматичними**. Вони існують тільки під час роботи функції, а після реалізації функції система видаляє локальні змінні і звільняє пам'ять. Тобто між викликами функції вміст локальних змінних знищується, тому ініціювання локальних змінних треба робити щоразу під час виклику функції. За необхідності збереження цих значень, їх треба описати як статичні за допомогою службового слова static

Прототип

Прототип функції може вказуватися до виклику функції замість опису функції для того, щоб компілятор міг виконати перевірку відповідності типів аргументів і параметрів. Прототип функції за формою такий же, як і заголовок функції, наприкінці його ставиться «;». Параметри функції в прототипі можуть мати імена, але компілятору вони не потрібні.

Компілятор використовує прототип функції для порівняння типів аргументів з типами параметрів. Мова C++ не передбачає автоматичного перетворення типів у випадках, коли аргументи не співпадають за типами з відповідними їм параметрами, тобто мова C++ забезпечує строгий контроль типів.

При наявності прототипу функції, які викликаються, не зобов'язані розміщатися в одному файлі з функцією, що їх викликає.

Виклик функції

Виклик функції може бути оформлений у вигляді оператора, якщо у функції відсутнє значення, що повертається, або у вигляді виразу, якщо існує значення, що повертається.

У першому випадку оператор має наступний формат:

ім'я_функції(список_аргументів);

Наприклад f(x);

В другому випадку вираз записується у такий спосіб: **h=f(x);**

Значення обчисленого виразу є значенням функції, що повертається. Значення, що повертається, передається в місце виклику функції і є результатом її роботи.

Число і типи формальних аргументів повинні співпадати з числом і типом фактичних параметрів функції. При виклику функції фактичні параметри підставляються замість формальних аргументів.

Області дії змінних

При оголошенні змінних у програмі велике значення має те місце, де вона оголошена. Від того, де оголошена змінна, залежить можливість її використання.

У С++ можливі три місця оголошення змінних

1. Глобальна змінна

По-перше, поза будь-яких функцій, у тому числі і **main()**. Така змінна називається г**лобальною** і може використовуватися в будь-якому місці програми від місця оголошення і до кінця програми.

2. Локальна змінна

По-друге, змінна може бути оголошена усередині блоку, у тому числі й усередині тіла функції. Оголошена в такий спосіб змінна називається **локальною** і може використовуватися лише усередині блоку. Така змінна поза блоком, у якому вона оголошена, невідома.

3. Змінна параметр функції

По-третє, змінна може бути оголошена **як параметр функції**. Крім спеціального призначення, а саме для передачі даних у функцію, параметр можна розглядати як локальну змінну для тіла функції.

Приклад 1. Скласти програму для обчислення суми k чисел.

```
void sum(int);// прототип функції
int s = 0;// глобальна змінна
void main() {
    setlocale(LC_CTYPE, "ukr");
    int i, b, k;// локальні змінні
    cout << "\nВведіть число доданків";
    cin >> k;
    for (i = 0; i<k; i++){
        cout << "\nВведіть новий доданок ";
        cin >> b;
        sum(b);// звернення до функції
    }
    cout << "\ns=" << s;
    system("pause");
}
void sum(int c){
s = s + c;
}
```

Приклад

У цій програмі змінна **s** є глобальною, вона доступна із обох функцій програми — **main()** і **sum()**, а змінні **i**, **b**, **k** та **c** — локальні, доступні тільки у тих функціях, де вони оголошені.

Якщо глобальна і локальна змінні мають одне і теж ім’я, тоді вважається, що оголошені дві різні змінні зі своїми областями використання. При цьому локальна змінна буде видима у тій функції, де вона оголошена, а глобальна у всій програмі за виключенням функції, у якій оголошена локальна змінна.

Приклад 2. Скласти програму, яка звертається до функції обрахунку максимуму з двох чисел, функція має знаходитись в окремому файлі

```
int max(int, int);/* прототип функції */

void main(){
    setlocale(LC_CTYPE, "ukr");
    int x, y, z;
    cout << "\n почергово введіть x та y \n";
    cin >> x; cin >> y; z = max(x, y); cout << "z=" << z;
    system("pause");
}
```

Файл max.cpp в проекті

```
#include "stdafx.h"

int max(int a, int b) {
    int c; /* робоча змінна */
    if (a >= b)c = a;
    else c = b;
    return c;
}
```

Передача параметрів

У мові С++ визначено декілька способів передачі параметрів і повернення результату обчислень функцій, серед них най більш широке використання набули:

- виклик функції з передачею параметрів за допомогою формальних аргументів-значень;
- виклик функції з передачею адрес за допомогою параметрів-показників;
- виклик функцій з використанням посилань, коли доступ до переданих параметрів забезпечується за допомогою альтер нативного імені (синоніма);
- виклик функцій з передачею даних за допомогою глобальних змінних;
- виклик функцій з застосуванням параметрів, що задані за замовчуванням, при цьому можна використовувати або всі аргументи, або їх частину.

Вказівники і функції

```
int func(int *b) { // створюємо вказівник, а не просту змінну
*b = *b + 1;
return  *b; // використовуєм вказівник у ф-ції
}
void main(){
    setlocale(LC_CTYPE, "ukr");
    int a = 19;
    cout << "a = " << a << endl;
    cout << "func = " << func(&a) << endl; // Передаємо АДРЕСУ змінної a
    cout << "a = " << a << endl;
    system("pause");
}
```

Виклик функції з використанням показників забезпечує передачу до функції не значень параметрів, а їх адреси, тому можливо міняти значення цих змінних усередині функції і передавати за її межі (в інші функції)

Посилання і функції

```
int func(int &b) {
    b++;
    return  b;
}
void main() {
    setlocale(LC_CTYPE, "ukr");
    int a = 19;
    cout << "a = " << a << endl;
    cout << "func = " << func(a) << endl;
    cout << "a = " << a << endl;
    system("pause");
}
```

Виклик функції з використанням параметра-посилання здійснює передачу до функції не самої змінної, а тільки посилання на неї. У цьому випадку забезпечується доступ до переданого параметра за допомогою визначення його альтернативного імені. Тоді всі дії, що відбуваються над посиланням, є діями над самою змінною.

Глобальні змінні і функції

```
int a, b, c; // глобальні параметри
void sum(); //----------- прототип функції
void main(){
    setlocale(LC_CTYPE, "ukr");
    cin >> a >> b;
    sum();                  //----------- виклик sum()
    cout << c << endl;
    system("pause");
}
void sum() {//----------- функція sum()
    c = a + b;
}
```

Параметри за замовчуванням

```
void funct1(float x, int y, int z = 80){
    cout << "x = " << x << " y = " << y << "z = " << z << endl;
}
void funct2(float x, int y = 25, int z = 100){
    cout << "x = " << x << "y = " << y << "z = " << z << endl;
}
void funct3(float x = 3.5, int y = 40, int z = 200){
    cout << "x = " << x << "y = " << y << "z = " << x << endl;
}
void main(){
    setlocale(LC_CTYPE, "ukr");
    funct1(5.1, 10); //за замовчуванням передається один аргумент – z
    funct2(10.2); //за замовчуванням передаються два аргументи – y, z
    funct3();          // за замовчуванням передаються всі аргументи
    system("pause");
}
```

Масив, як параметр функції

```
void printMas(int b[], int len);
void main(){
    setlocale(LC_CTYPE, "ukr");
    const short int size = 10;
    int a[size];
    for (int i = 0; i<size; ++i){
        a[i] = rand()%100;
    }
    printMas(a, size);
    system("pause");
}
void printMas(int b[], int len) {
    for (int i = 0; i<len; ++i){
        cout << b[i] << "\t";
    }
}
```

```
void printMas(int b[], int len);
int  minMas(int *b, int len);
void setMas(int *b, int len);
void main(){
	setlocale(LC_CTYPE, "ukr");
	srand(time(0));
	const short int size = 10;
	int a[size];
	setMas(a, size);
	printMas(a, size);
	cout << endl << minMas(a, size);
	system("pause");
}
void printMas(int b[], int len) {
	for (int i = 0; i<len; ++i){
		cout << b[i] << "\t";
	}
}
int minMas(int *b, int len) {
	int min=b[0];
	for (int i = 1; i<len; ++i) {
		if (min > b[i]) min = b[i];
	}
	return min;
}
void setMas(int *b, int len) {
	for (int i = 0; i<len; ++i){
		b[i] = rand() % 100;
	}
}
```