Функції

Функції по своїй суті – це підпрограми, які маніпулюють даними, та можуть повертати деяке значення.

Кожна програма C++ має хоча б одну функцію. І ця функція – це main(), яка при запуску програми викликається автоматично. Функція main() в свою чергу, може викликати і інші функції, а ті – в свою чергу – можуть викликати наступні і т.д.

Кожна функція має своє ім'я. Коли воно зустрічається в програмі, то керування переходить до тіла даної функції. Цей процес називається викликом функції. Після повернення із функції керування відновлюється з рядка, що слідує після виклику функції.

Грамотно розроблені функції повинні виконувати конкретну і доволі зрозумілу задачу. Наприклад: функція пошуку мінімального значення в масиві, функція обчислення середнього арифметичного значення елементів масиву, функція сортування масиву, функція порівняння двох чисел і т.д.

Важкі задачі можна «розбивати» на ряд маленьких але зрозумілих підзадач, які легко реалізовуються як окремі функції. Потім такі функції будуть викликатися по черзі.

Опис функції

тип_результату ім'я_функції (<тип** параметр1>, <**тип** параметр2>, <…>**)
{
Тіло_функції;
…
}

де:

тип_результату – може бути будь-яким (int, void, char, bool, float, double…) – в залежності від того, що вимагається в задачі;

ім'я_функції – ім'я, згідно правил іменування змінних;

*<**тип** параметр1>, <**тип** параметр2>, <…> - список параметрів функції, параметри обов'язково беруться в дужки ();*

Тіло_функції представляє собою набір операторів. Навіть, якщо в тілі функції буде один оператор – тіло функції все одно заключається у фігурні дужки { }.

Функції, що не повертають результат

Коли мова йде про те, що функція не повертатиме ніякого результату, то мається на увазі, що ***тип_результату*** *буде* ***void.***

Приклад:

Оголосимо функцію, яка буде друкувати нам наше ім'я.

```
void printName()
{
        cout << "My name is Vasia Pupkin" << endl;
}
```

Дана функція буде лише друкувати повідомлення "My name is Vasia Pupkin", тому доцільно тут використати тип **void,** це є сигналом компілятору, що ніяких значень ми не повернемо**.**

Функції, що повертають результат

Функції можуть повернути результат. Після звернення до функції вона може виконати деякі дії, а потім в якості результату послати назад деяке значення. Воно називається *повернутим* значенням, при чому тип цього значення обов'язково має бути оголошеним. Наприклад:

int myFunction();

оголошує, що функція myFunction повертає цілочисельне значення.

Щоб забезпечити повернення результату із функції, треба у тілі функції останньою командою написати ключове слово return, а за ним значення, яке буде повернуто. В якості результату, що повертається, можна задавати і константні значення, так і вирази.

Наприклад:

return 5; // повернеться 5

return x > 5; // повернеться true, якщо x > 5, і false – якщо x < 5

return x; // повернеться значення змінної x

Після того, як у функції зустрінеться ключове слово return, буде виконаний вираз, що стоїть після цього ключового слова, і його результат буде переданий в основну програму у місце виклику функції. Після оператора return, програма негайно переходить до рядка, що слідую після виклику функції. І будь-який вираз, що стоїть у функції після return не виконується. Але функція може містити декілька операторів return. Дивимося приклад нижче:

```
int compare(int a, int b)
{
        if(a > b)
                return a;
        else if(a < b)
                return b;
        else
                return -1;
}
```

Параметри і аргументи

В функцію можна також посилати деякі значення. Опис посилаючих значень називають списком параметрів.

```
int myFunction(int someValue, float someFloat);
```

Це означає, що функція myFunction не тільки повертає деяке ціле значення, а й отримує два значення в якості параметрів: ціле та дробове.

Параметр описує тип значення, яке буде передане в функцію під час виклику. Таке значення називається фактичним параметром, і ще називають аргументом функції.

```
int valueReturned = myFunction(19, 6.8);
```

Тут цілочисельна змінна valueReturned ініціалізується значенням, що повертає функція myFunction, і в якості аргументів тут передаються числа 19 і 6.8. Тип аргументів повинен відповідати типам оголошених параметрів. Оголошені параметри називають формальними параметрами.

```
int myFunction(int someValue, float someFloat) // формальні параметри

int valueReturned = myFunction(19, 6.8); // фактичні параметри, аргументи
```

Оголошення функцій

До виклику функції в програмі, вона має бути:

1. *Оголошена*
2. *Визначена*

Шляхом оголошення функцій, компілятору повідомляється її ім'я, список параметрів та тип повертаємого значення.

Дякуючи визначенню функції, компілятор дізнається ЯК САМЕ працює функція. Ні одну функцію не можна викликати в програмі, якщо вона попередньо не була оголошена.

Є два способи оголошення і визначень функцій.

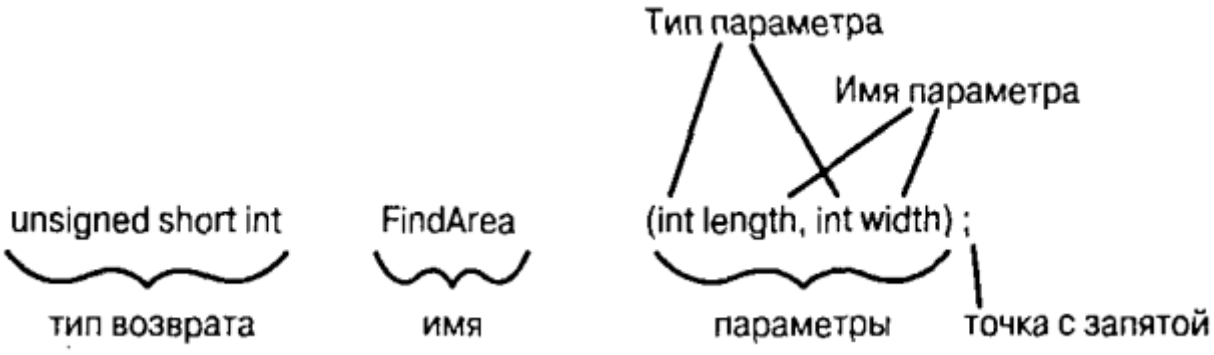
Функція, оголошена перед main() та відразу визначена

Будь-яку функцію можна визначити перед її використанням. Таким чином вона буде відразу і оголошена.

Тут ніби все добре, от тільки це вважається не дуже хорошим стилем програмування.

Оголошення через прототип

Іншим, грамотним способом оголошення функцій є **прототип**.



Прототип функції являє собою вираз, що закінчується крапкою з комою (;) і складається з типу повертаємого значення та сигнатури. Сигнатура – це ім'я функції та список формальних параметрів.

```
int myFunction(int someValue, float someFloat);
int compare(int a, int b);
float avg(float num1, float num2);
void printResult();
```

Коли будемо визначати функцію, то заголовок функції повинен строго відповідати своєму прототипу. Можливий варіант, коли ми оголошуємо прототим то імена параметрів можна не писати. А у визначення писати обов'язково.

```
int myFunction(int, float);
int compare(int, int);
float avg(float, float);
void printResult();
```