Programowanie 1

Funkcje w języku C - wprowadzenie

Piotr Błaszyński

17 października 2021

Funkcje w języku C mają następującą postać:

```
typ_zwracanej_wartosci nazwa_funkcji(typ_argumentu1
   nazwa_argumentu1, typ_argumentu2 nazwa_argumentu2,
   ...)
{
  linia_kodu_1;
  linia_kodu_2;
  //
  //
  linia_kodu_n;
  return wartosc;
}
```

Definicja argumentu wygląda tak samo jak definicja zmiennej, z tą różnicą, że nie stawia się na końcu średnika. Jeżeli natomiast występuje wiele argumentow to definicje argumentów oddziela się przecinkiem: double a, int b
Argumenty funkcji pełnią rolę zmiennych w ciele funkcji. Ich znaczenie odpowiada zmiennym lokalnym, z tą różnicą, że przy wywołaniu funkcji trzeba nadać im wartość. Wywołanie funkcji odbywa się przez podanie jej nazwy i listy parametrów. Jeżeli nie ma żadnych parametrów, to i tak konieczne jest napisanie nawiasów:

```
setValue(1, a);
doSomething();
```

W treści funkcji może wystąpić wiele razy słowo kluczowe return. W przypadku funkcji z określonym (innym niż void) typem zwracanej wartości, po słowie return występuje wartość (może być to stała, zmienna lub nawet wyrażenie). Jeżeli funkcja nie ma określonego typu zwracanej wartości (void) to słowo kluczowe return tez

może wystąpić, ale nie musi. Oznacza ono wtedy po prostu zakończenie działania funkcji i powrót do funkcji wywołującej. Nie podajemy wtedy żadnej wartości po return.

Pełna definicja funkcji składa się tak naprawdę z:

- deklaracji - nagłówka (określa nazwę, typ zwracanej wartości i listę argumentów) (zamiennie mówi się prototyp), jeżeli jest on oddzielnie to jest zakończony średnikiem, nie ma wtedy konieczności podawania nazw zmiennych (tylko typ),
- definicji - nagłówek + ciało funkcji.

Nagłówek funkcji musi być umieszczony przed pierwszym wywołaniem, często też umieszcza się nagłówek w pliku nagłowkowym (.h).

Jeżeli funkcja zwraca jakąś wartość - typ zwracany inny niż void - to żeby skorzystać z tej wartości musi ona zostać przypisana do zmiennej, np.:

```
int value=getValue();
```

Ciekawostka: w starych implementacjach języka C traktowano return jako wywołanie funkcji i w związku z tym zapisywano wartość zwracaną jak parametr funkcji:

```
return (0);
```

Jest to w dalszym ciągu prawidłowa (lecz nadmiarowa forma), natomiast można się na nią w wielu miejscach natknąć.

Przykładowe funkcje i ich wywołania:

```
int addValues(int a, int b)
{
        return a+b;
}

int main(void)
{
        int a = 10;
        int b = 11;
        int sumOfConsts= addValues(5, 6);
        int sumOfVariables = addValues(a, b);
}
```

Podobnie jak powyżej, ale przed wywołaniem jest tylko prototyp funkcji, całość jest zdefiniowana później:

```
int addValues(int a, int b);
int main(void)
{
        int a = 10;
        int b = 11;
        int sumOfConsts= addValues(5, 6);
        int sumOfVariables = addValues(a, b);
}
int addValues(int a, int b)
{
        return a+b;
}
```

Podobnie jak powyżej, ale w prototypie funkcji nie ma nazw zmiennych:

```
int addValues(int, int);
int main(void)
{
        int a = 10;
        int b = 11;
        int sumOfConsts= addValues(5, 6);
        int sumOfVariables = addValues(a, b);
}
int addValues(int a, int b)
{
        return a+b;
}
```