Funkcje

Funkcja to wydzielony fragment programu realizujący pewne zadanie. Funkcje stosuje się w celu zwiększenia czytelności kodu.

Budowa funkcji

```
typ_zwracanego_wyniku nazwa_funkcji (zestaw parametrów formalnych)
{
    ciało funkcji - zestaw instrukcji i operacji
}
```

Każda funkcja musi mieć:

- nazwę – zgodną z regułami nazewnictwa (np. bez słów kluczowych, polskich znaków diakrytycznych itp.),
- określony zestaw parametrów formalnych przekazywanych do funkcji:
  - podajemy typ i nazwę każdego parametru oraz informację o sposobie przekazywania tego parametru do funkcji; parametry umieszczamy po nazwie funkcji, w nawiasie, oddzielone przecinkami,
  - funkcja może nie pobierać żadnych parametrów (tzw. funkcja bezparametrowa); po nazwie funkcji umieszczamy nawias pusty (),
- określony typ wartości zwracanej przez funkcję – do zwracania wartości służy instrukcja **return**, po której funkcja kończy działanie; jeśli funkcja nie zwraca wartości jest typu **void** (typu pustego).

Przykłady:

Funkcja zwracająca wynik typu całkowitego (**int**) z dwoma parametrami typu całkowitego

```
int nazwa1 (int a, int b)
{
    …
    return …;
}
```

Funkcja zwracająca wynik typu logicznego (**bool**; **1** – **true**; **0** – **false**) z jednym parametrem typu całkowitego

```
bool nazwa2 (int a)
{
    …
    return …;
}
```

Funkcja, która nie zwraca wyniku (**void**) z dwoma parametrami typu całkowitego

```
void nazwa3 (int a, int b)
{
   …
}
```

Funkcja, która nie zwraca wyniku (**void**) bez parametrów

```
void nazwa4 (void)
{
   …
}
```

lub

```
void nazwa5 ()
{
   …
}
```

Uwaga: w ciele funkcji typu **void** nie ma zwracania wyniku.

i wiele innych...

Sposoby przekazywania parametrów

Parametry funkcji mogą być przekazywane przez wartość (**nazwa_zmiennej**) lub przez referencję (**& nazwa_zmiennej**).

Zmienne przekazywane do funkcji przez wartość przed wywołaniem i po wywołaniu funkcji mają taką samą wartość.

Zmienne przekazywane do funkcji przez referencję mogą mieć różną wartość przed i po wywołaniu danej funkcji. Wykorzystujemy to m.in. wtedy, gdy chcemy zmianę dokonaną w funkcji zachować po zakończeniu jej działania.

Przykład:

Funkcja zamienia miejscami wartości dwóch parametrów

```
void zamiana (int &a, int &b)
{
 int temp=a;
 a=b;
 b=temp;
}
```

temp jest zmienną pomocniczą.

Parametry aktualne – wykorzystanie funkcji w programie

Definiując funkcję podajemy zestaw parametrów formalnych. Wywołując w programie funkcję, wcześniej zdefiniowaną, podajemy informację, dla jakich parametrów aktualnych ma się ona wykonać w danym momencie. Parametry aktualne mogą być podane w postaci konkretnych wartości lub w postaci zmiennych np. pobranych od użytkownika. Warunki konieczne:

- liczba parametrów formalnych i liczba parametrów aktualnych musi być taka sama,
- parametr formalny i odpowiadający mu parametr aktualny muszą być tego samego typu.

Zmienne lokalne i globalne

Zmienne lokalne deklarujemy wewnątrz funkcji. Zmienna lokalna jest znana tylko od momentu jej deklaracji do końca klamry zamykającej blok (funkcję).

Zmienne globalne deklarujemy poza funkcjami. Są one widoczne wewnątrz wszystkich funkcji poniżej ich miejsca zadeklarowania. Zmienne globalne są zerowane, ale powinniśmy zachować umiar w ich stosowaniu.

Uwagi

1. Funkcję definiujemy przed jej pierwszym wywołaniem – w myśl zasady, że każda nazwa musi zostać zadeklarowana przed jej pierwszym użyciem.
2. Nie wolno definiować funkcji we wnętrzu innej funkcji.
3. Instrukcja **return** zwraca wynik i kończy działanie funkcji.
4. Rezultat zwracany przez funkcję może być parametrem aktualnym innej funkcji.
5. Referencji używamy wtedy, gdy chcemy zmienić wartość parametrów przekazywanych do funkcji i zachować te zmiany po zakończeniu działania funkcji. Należy unikać tego sposobu przekazywania parametrów.
6. Więcej o zmiennych globalnych

   (http://pl.wikipedia.org/wiki/Zmienna_globalna).