Funkcje w C

TWORZENIE I WYWOŁYWANIE, FUNKCJE TYPU VOID, FUNKCJE Z WARTOŚCIĄ ZWROTNĄ, ARGUMENTY FUNKCJI, FUNKCJE INLINE, PRZECIĄŻANIE NAZW FUNKCJI

Funkcja (ang. function) jest to fragment programu, któremu nadano nazwę i który możemy wykonać poprzez podanie jego nazwy oraz ewentualnych argumentów. Argumentami są natomiast dane przekazywane do funkcji. Funkcja może (ale nie musi) zwracać jakąś określoną typem zmienną.

Definicja funkcji:

```
typZwracany nazwaFunkcji(typArg1 nazwaArg1, typArg2 nazwaArg2, itd.)
{
	return zwracana_wartosc;
}
```

`typZwracany` – określa rodzaj informacji, którą zwraca funkcja jako wynik swojej pracy. Jeśli funkcja nie zwraca wyniku to posiada tym pusty (void)

`nazwaFunkcji` - zbudowana podobnie jak nazwa zmiennej – stosujemy identyczne reguły. Nazwa funkcji umożliwia odwołanie się do jej kodu.

`lista argumentów` – zawiera definicję danych przekazywanych do funkcji. typArg1 – jeden z typów danych zdefiniowanych w C, `nazwaArg1` - zbudowana identycznie jak nazwa zmiennej.

Funkcja niezwracająca wartości i nieprzyjmująca argumentów (typ zwracany void)

```
void noReturnFunction()
{
	//ciało
}
```

Funkcja przyjmująca dwa argumenty oraz zwracająca dany typ (zwracanie poprzez słowo kluczowe return):

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

Wywołanie funkcji:

```
//Wywołanie funkcji bezatrybutowej nie zwracającej żadnej wartości
noReturnFunction();
//Wywołanie funkcji z argumentami oraz przypisanie wartości przez niej zwracanej do
zmiennej suma
int suma = superFunction(1, 2);
```

Wewnątrz funkcji możemy deklarować własne zmienne wymagane do rozwiązania problemu. Zmienne takie są zmiennymi tymczasowymi, które znikną po opuszczeniu przez program funkcji.

```
int superFunction(int a, int b)
{
int suma = a + b;
```

```
cout << "Suma liczb: " << suma;
return suma % 2;
}
```

Funkcje inline – można rozumieć w ten sposób, że kod tej funkcji jest "wklejany" w miejsce, gdzie została ona wywołana. Funkcje tego typu tworzy się zazwyczaj dla krótkich kilku linijkowych funkcji. Jeśli kompilator uzna, że dany podprogram nie kwalifikuje się jako inline, to słowo inline zostanie zignorowane.

```
inline int funkcja(argumenty);
inline void innafunkcja(argumenty);
```

Przeciążanie nazw funkcji - polega na wielokrotnym wykorzystaniu takiej samej jej nazwy, różniącej się tylko typem i ilością argumentów (przeciążanie funkcji można traktować jako polimorfizm).

```
float Pole(float p); //funkcja liczy pole kwadratu
double Pole(double p); //funkcja liczy pole koła
float Pole(float a, float b); // funkcja liczy pole prostokąta itd.
```

Funkcje rekurencyjne

Rekurencja zwana rekursją, polega na wywołaniu przez funkcję samej siebie. Algorytmy rekurencyjne zastępują w pewnym sensie iteracje. Niekiedy problemy rozwiązywane tą techniką będą nieznacznie wolniejsze (wiąże się to z wywoływaniem funkcji), natomiast rozwiązanie niektórych problemów jest dużo prostsze w implementacji.

```
void moja_funkcja(int a) {
       if (a == 0) return;
       else return moja_funkcja(a--);
   }
```

Przykład. Program wyznaczający sumę n kolejnych liczb naturalnych:

```
long long suma(int n) {
       if (n < 1)
              return 0;
       return n + suma(n - 1);
}
```

Zadania do samodzielnego wykonania:

1. Napisz program, który wyznaczy silnię z liczby n sposobem rekurencyjnym.
2. Napisz program wyznaczający n-ty wyraz ciągu zdefiniowanego przez następujący wzór rekurencyjny:

$$a_n = \begin{cases} -1 & dla\ n = 1 \\ -a_{n-1} \cdot n - 2 & dla\ n > 1 \end{cases}$$

3. Napisz program wyznaczający NWD(a,b) metodą rekurencyjną.
4. Dana jest następująca funkcja rekurencyjna:

```
funkcja wynik( i )
      jeżeli i < 3
            zwróć 1 i zakończ;
      w przeciwnym razie
            jeżeli i mod 2 = 0
                  zwróć wynik(i – 3) + wynik(i – 1) + 1
            w przeciwnym razie
                  zwróć wynik(i – 1) mod 7
```

Przenalizuj działanie funkcji rekurencyjnej i uzupełnij poniższą tabelę.

i	2	3	4	5	6	7	8
wynik(i)							

5. Wykorzystując funkcje rekurencyjną napisz wykonaj zamianę liczby w systemie dziesiętnym na system dwójkowy.
6. Napisz funkcje rekurencyjną wyznaczająca ciąg Fibanaccego:

$$F_n = \begin{cases} 0 & dla \quad n = 0 \\ 1 & dla \quad n = 1 \\ F_{n-2} + F_{n-1} & dla \quad n > 1 \end{cases}$$

7. Wykorzystując funkcje rekurencyjną napisz wykonaj zamianę liczby w systemie dziesiętnym na system dwójkowy.
8. Dana jest następująca funkcja rekurencyjna:

Dane:

x — liczba całkowita,

n — dodatnia liczba całkowita.

```
funkcja F(x, n)
   jeżeli n = 1
         podaj wynik x i zakończ
```

```
   w przeciwnym razie
         jeżeli n mod 3 = 0
               k ← F(x, n div 3)
(*)                  podaj wynik k*k*k i zakończ
         w przeciwnym razie
(**)                 podaj wynik x*F(x, n-1) i zakończ
```

Uwaga: „div” jest operatorem dzielenia całkowitego.

a) Podaj wszystkie wywołania rekurencyjne funkcji F oraz obliczany po każdym wywołaniu wynik, jeśli na początku wywołamy F(2, 10).

wywołanie	wynik
F(2,10)	
F(;)	

b) Uzupełnij tabelę o brakujące elementy:

x	n	wynik D(x,n)
2	2	4
2	3	
3		81
	5	32
2		256
	10	1024