Laboratorium 5 – Funkcje

W języku C funkcja (podprogram, procedura) to wydzielona część programu, która przetwarza argumenty, wykonuje zawarty w jej ciele algorytm i może, ale nie musi, zwracać wartość poprzez polecenie *return*.

Przykłady funkcji, których używaliśmy na poprzednich laboratoriach:

- *printf* – wyświetla tekst na ekranie
- *main* – główna funkcja realizująca program

Funkcje tworzymy w momencie, kiedy chcemy uniknąć wielokrotnego powtarzania tego samego fragmentu kodu.

Przykład 1: W naszym programie musimy wielokrotnie liczyć długości różnych trójwymiarowych wektorów.

```c
#include <stdio.h>
#include <math.h> //tu znajduje się funkcja sqrt(), z której korzystamy dalej

int main() {
    int a, b, c, d, e, f;
    double dlugosc;
    printf("Podaj calkowite wspolrzedne wektora oddzielajac je spacjami:\n");
    scanf("%d %d %d", &a, &b, &c);
    dlugosc = sqrt(a*a + b*b + c*c);
    printf("\nDlugosc twojego wektora to %f\n", dlugosc);
    dlugosc = sqrt(2*a*2*a + 2*b*2*b + 2*c*2*c);
    printf("\nCzy wiesz ze dwukrotnosc Twojego wektora ma dlugosc %f?\n", dlugosc);
    printf("\nPodaj jeszcze jeden wektor: ");
    scanf("%d %d %d", &d, &e, &f);
    dlugosc = sqrt((a - d)*(a - d) + (b - e)*(b -e) + (c - f)*(c - f));
    printf("\nTeraz ciekawszy rezultat - roznica podanych wektorow ma dlugosc %f\n", dlugosc);
    return 0;
}
```

Zamiast powtarzać kod w linijce 9, 11 i 15, możemy stworzyć funkcję *dlugosc_wektora_3d*, która zastosuje znany nam wzór dla podanych liczb, tak jak w kodzie poniżej:

```
1  #include <stdio.h>
2  #include <math.h> //tu znajduje się funkcja sqrt(), z której korzystamy dalej
3
4  double dlugosc_wektora_3d(int a, int b, int c) {
5      double wynik;
6      wynik = sqrt(a*a + b*b + c*c);
7      return wynik;
8  }
9
10 int main() {
11     int a, b, c, d, e, f;
12     double dlugosc;
13     printf("Podaj calkowite wspolrzedne wektora oddzielajac je spacjami:\n");
14     scanf("%d %d %d", &a, &b, &c);
15     dlugosc = dlugosc_wektora_3d(a,b,c);
16     printf("\nDlugosc twojego wektora to %f\n", dlugosc);
17     dlugosc = dlugosc_wektora_3d(2*a, 2*b, 2*c);
18     printf("\nCzy wiesz ze dwukrotnosc Twojego wektora ma dlugosc %f?\n", dlugosc);
19     printf("\nPodaj jeszcze jeden wektor: ");
20     scanf("%d %d %d", &d, &e, &f);
21     dlugosc = dlugosc_wektora_3d(a-d, b-e, c-f);
22     printf("\nTeraz ciekawszy rezultat - roznica podanych wektorow ma dlugosc %f\n", dlugosc);
23     return 0;
24 }
```

Jak wygląda składnia DEFINIOWANIA FUNKCJI w języku C? Patrząc na powyższy przykład przeanalizujmy definicję funkcji w linijkach 4-8.

1. *double* oznacza typ danych zwracanych przez funkcję – nasza funkcja jako wynik zwróci liczbę zmiennoprzecinkową typu *double*
2. *dlugosc_wektora_3d* to wybrana przez nas nazwa funkcji
3. *(int a, int b, int c)* – w nawiasach okrągłych po nazwie funkcji podajemy po przecinkach wszystkie argumenty, które przyjmuje nasza funkcja – podajemy typ i nazwę zmiennej będącej parametrem wejściowym; zadeklarowane w linii 4 zmienne *a, b, c* są zmiennymi lokalnymi – możemy na nich wykonywać działania jedynie wewnątrz definiowanej przez nas funkcji.
4. pomiędzy nawiasami klamrowymi (linie 5-7) znajduje się tzw. ciało funkcji – wszystkie polecenia, które mają zostać wykonane w ramach działania funkcji.
5. polecenie *return* kończy działanie funkcji i zwraca obliczony rezultat.

Jak wygląda składnia WYWOŁANIA FUNKCJI w języku C? W linijkach 15, 17 oraz 21 wywołujemy napisaną przez nas funkcję – podajemy nazwę funkcji i argumenty wejściowe w nawiasach okrągłych. Zwróconą przez funkcję wartość wynikową zapisujemy do zmiennej dlugosc, aby następnie wyświetlić ją na ekranie używając funkcji (!) *printf*.

Przykład 2: Funkcja może pobierać wartości wejściowe, ale nie zwracać nic.

```c
void rysuj_kwadrat(int n) {
    /* w rzeczywistości nie będzie to kwadrat
       a prostokąt z uwagi na wymiary znaku w terminalu*/
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            if (i == 0 || i == n - 1)
                printf("#");
            else if (j == 0 || j == n - 1)
                printf("#");
            else printf(" ");
        }
        printf("\n");
    }
}
```

Powyższa funkcja przyjmuje jako argument liczbę całkowitą n, rysuje na ekranie kwadrat (prostokąt), ale nie zwraca żadnego wyniku (brak instrukcji *return*).

Przykład 3: Funkcja może również nie pobierać żadnych argumentów wejściowych.

```c
int losuj_liczbe() {
    /* funkcja wylosuje liczbe calkowita z przedzialu 0..99 */
    int seed;
    int wylosowana;
    srand(1);
    wylosowana = rand() % 100;
    return wylosowana;
}
```

Funkcja *losuj_liczbe* nie przyjmuje żadnych argumentów na wejściu, ale zwraca wartość całkowitą.

Rekurencja

Funkcje mogą wywoływać siebie nawzajem – w przykładzie 1. w linii 6 wewnątrz funkcji *dlugosc_wektora_3d* wywoływana jest funkcja sqrt (której definicja znajduje się w bibliotece *math* dołączonej za pomocą *#include <math.h>*).

Funkcje mogą jednak wywoływać SAME SIEBIE w sposób rekurencyjny. Przeanalizuj poniższy przykład.

Przykład 4: Napiszemy funkcję obliczającą sumę liczb naturalnych od 1 do N w sposób rekurencyjny. Wykorzystamy fakt, że dla dla N=1 suma wynosi 1, a dla dowolnego N suma 1 + ... + N jest równa (1 + ... + N-1) + N, co można zapisać rekurencyjnie:

suma(1) = 1

suma(N) = suma(N-1) + N

```c
#include <stdio.h>

int suma(int n) {
    if (n < 1) {
        printf("Podaj liczbe naturalna dodatnia!");
        return -1;
    } else if (n == 1) {
        return 1; // warunek poczatkowy rekurencji
    } else {
        return suma(n - 1) + n; // wzor rekurencyjny
    }
}

int main() {
    int koniec, wynik;
    printf("Podaj liczbę, do ktorej sumujemy: ");
    scanf("%d", &koniec);
    wynik = suma(koniec);
    if (wynik > 0)
        printf("Suma liczb naturalnych do %d wynosi %d", koniec, wynik);
    return 0;
}
```

Zadania do samodzielnego rozwiązania

Zadanie 1: (1p)

Wykorzystując kod z przykładu 2 i używając funkcji, napisz program, który pyta użytkownika o wymiar kwadratu do narysowania, sprawdza poprawność podanych przez użytkownika danych, a następnie rysuje kwadrat na ekranie. Zmodyfikuj kod tak, żeby do kwadratu o wymiarach większych niż 8 dodane były przekątne.

Zadanie 2: (2p)

Wykorzystując fakt, że 0!=1 oraz n!=n*(n-1)!, napisz funkcję rekurencyjną obliczającą silnię z n podanego przez użytkownika. Dopuszczalny zakres argumentów n to [0,50].

Zadanie 3: (2p)

Wykorzystując funkcję z zadania 2, w tym samym pliku z kodem, dopisz funkcję obliczającą symbol Newtona. Wykorzystując napisane funkcje, wypisz na ekranie trójkąt Pascala do poziomu wskazanego przez użytkownika. Sprawdź poprawność danych wejściowych podanych przez użytkownika!

Zadanie 4: (dodatkowe – 3p)

Napisz funkcje i zaprezentuj ich działanie dla wektorów trójwymiarowych:

- obliczająca długość wektora
- sumująca i odejmująca wektory
- obliczająca iloczyn skalarny
- sprawdzająca, czy dwa podane wektory są prostopadłe
- obliczająca iloczyn wektorowy
- sprawdzająca, czy dwa podane wektory są równoległe