

Programowanie w języku C (EAR1S02005)

Politechnika Białostocka - Wydział Elektryczny
Automatyka i Robotyka, semestr II, studia stacjonarne I stopnia
Rok akademicki 2020/2021

Zajęcia nr 9 (23.03.2021)

dr inż. Jarosław Forenc

Rekurencja

- **Rekurencja** lub **rekursja** - jest to odwoływanie się funkcji lub definicji do samej siebie
- Rozwiązanie danego problemu wyraża się za pomocą rozwiązań tego samego problemu, ale dla danych o mniejszych rozmiarach
- W matematyce mechanizm rekurencji stosowany jest do definiowania lub opisywania algorytmów

- Silnia:

$$n! = \begin{cases} 1 & \text{dla } n = 0 \\ n(n-1)! & \text{dla } n \geq 1 \end{cases}$$

```
int silnia(int n)
{
    if (n==0)
        return 1;
    else
        return n*silnia(n-1);
}
```

Przykład: silnia

```
#include <stdio.h>
int silnia(int n)
{
    int s;
    printf("silnia(%d): start\n",n);
    if (n==0) s = 1;
    else      s = n*silnia(n-1);
    printf("silnia(%d): %d\n",n,s);
    return s;
}
int main(void)
{
    int n;
    printf("Podaj n: "); scanf("%d",&n);
    printf("%d! = %d\n",n,silnia(n));
    return 0;
}
```

```
Podaj n: 5
silnia(5): start
silnia(4): start
silnia(3): start
silnia(2): start
silnia(1): start
silnia(0): start
silnia(0): 1
silnia(1): 1
silnia(2): 2
silnia(3): 6
silnia(4): 24
silnia(5): 120
5! = 120
```

Przekazywanie argumentów przez wartość

- W funkcji `swap1` tworzone są lokalne kopie zmiennych skojarzonych z jej argumentami (zmienna `a` i `b` z funkcji `main` nie zmieniają swoich wartości)

```
#include <stdio.h>
void swap1(int a, int b)
{
    int tmp;
    tmp = a; a = b; b = tmp;
}
int main(void)
{
    int a = 10, b = 20;
    swap1(a,b);
    printf("a = %d, b = %d\n",a,b);
    return 0;
}
```

```
a = 10, b = 20
```

Przekazywanie argumentów przez wskaźnik

- Do funkcji `swap2` przekazywane są adresy zmiennych będących jej argumentami (zmienna `a` i `b` z funkcji `main` zmieniają swoje wartości)

```
#include <stdio.h>

void swap2(int *a, int *b)
{
    int tmp;
    tmp = *a; *a = *b; *b = tmp;
}

int main(void)
{
    int a = 10, b = 20;
    swap2(&a, &b);
    printf("a = %d, b = %d\n", a, b);
    return 0;
}
```

```
a = 20, b = 10
```

Parametry funkcji - wektory

- Wektory przekazywane są do funkcji przez **wskaźnik**
- Nie jest tworzona kopia tablicy, a wszystkie operacje na jej elementach odnoszą się do tablicy z funkcji wywołującej
- W nagłówku funkcji podaje się typ elementów tablicy, jej nazwę oraz nawiasy kwadratowe z liczbą elementów tablicy lub same nawiasy kwadratowe

```
void fun(int tab[5])
{
    ...
}
```

```
void fun(int tab[])
{
    ...
}
```

- W wywołaniu funkcji podaje się tylko jej nazwę (bez nawiasów kwadratowych)

```
fun(tab);
```

Przykład: parametry funkcji - wektor

```
#include <stdio.h>

void drukuj(int tab[])
{
    for (int i=0; i<5; i++)
        printf("%3d", tab[i]);
    printf("\n");
}

void zeruj(int tab[5])
{
    for (int i=0; i<5; i++)
        tab[i] = 0;
}
```

```
float srednia(int tab[])
{
    float sr = 0;
    int suma = 0;

    for (int i=0; i<5; i++)
        suma = suma + tab[i];

    sr = (float)suma / 5;

    return sr;
}
```

Przykład: parametry funkcji - wektor

```
int main(void)
{
    int tab[5] = {1,2,3,4,5};
    float sred;

    drukuj(tab);

    sred = srednia(tab);
    printf("Srednia elementow: %g\n", sred);

    zeruj(tab);
    drukuj(tab);

    return 0;
}
```

```
1 2 3 4 5
srednia elementow: 3
0 0 0 0 0
```

Parametry funkcji - macierze

- Macierze przekazywane są do funkcji przez **wskaźnik**
- W nagłówku funkcji podaje się typ elementów tablicy, jej nazwę oraz w nawiasach kwadratowych liczbę wierszy i kolumn lub tylko liczbę kolumn

```
void fun(int tab[2][3])  
{  
    ...  
}
```

```
void fun(int tab[][3])  
{  
    ...  
}
```

- W wywołaniu funkcji podaje się tylko jej nazwę (bez nawiasów kwadratowych)

```
fun(tab);
```

Przykład: parametry funkcji - macierz

```
#include <stdio.h>  
  
void zero(int tab[][3])  
{  
    for (int i=0; i<2; i++)  
        for (int j=0; j<3; j++)  
            tab[i][j] = 0;  
}  
  
void drukuj(int tab[2][3])  
{  
    for (int i=0; i<2; i++)  
    {  
        for (int j=0; j<3; j++)  
            printf("%3d", tab[i][j]);  
        printf("\n");  
    }  
}
```

1	2	3
4	5	6
0	0	0
0	0	0

```
int main(void)  
{  
    int tab[2][3] =  
        {1, 2, 3, 4, 5, 6};  
  
    drukuj(tab);  
    zero(tab);  
    printf("\n");  
    drukuj(tab);  
  
    return 0;  
}
```