Programowanie I
Wykład 11

dr inż. Adam Zielonka

Instytut Matematyki,
Politechnika Śląska

Gliwice 04.01.2019

Rekurencja

Bezpośrednie lub pośrednia możliwość wywołania w definicji funkcji tej samej funkcji.

Bezpośrednia

```
funkcja(dane)
{
    ...
    funkcja(dane1);
    ...
}
```

Pośrednia

```
g(dane)
{   ...
    f(dane1);
    ...
}
f(dane)
{   ...
    g(dane1);
    ...
}
```

Silnia

$$n! = 1 \cdot 2 \cdot 3 \cdot (n-1) \cdot n$$

Bez rekurencji

```
unsigned long long silnia(unsigned int n)
{
    unsigned long long res=1;
    for( int i=2 ; i< =n ;i++ )
        res*=i;
    return res;
}
```

Silnia

$$\begin{aligned} 0! &= 1 \\ n! &= (n-1)! \cdot n \end{aligned}$$

Rekurencja

```
unsigned long long silnia(int n)
{
    if( n < 2 )
        return 1;

    return n * silnia( n - 1 );
}
```

Informacje ogólne

- Implementując funkcję rekurencyjną musimy zadbać o warunek przerwania kolejnych wywołań funkcji.
- Rekurencję stosujemy tylko w sytuacji, gdy jesteśmy w stanie oszacować ilość zagnieżdżeń a przebieg działania algorytmu jest nie łatwy do przewidzenia.
- Wszystko co można zaimplementować przy użyciu rekurencji można zrealizować bez jej użycia, ale może to prowadzić do bardzo istotnego skomplikowania kodu.
- Zaletą rekurencji jest przejrzystość i czytelność kodu.
- Podstawową wadą jest ograniczona możliwość zagnieżdżeń i wiążący się z nimi narzut.

Rekurencja przykład

Ciąg Fibonacci'ego

$$f_0 = 1, \quad f_1 = 1, \quad f_n = f_{n-1} + f_{n-2}, \quad n > 1$$

$$1, 1, 2, 3, 5, 8, 13, 21, 34, 55, \cdots$$

```
unsigned long long int fib( int n )
{
    if( n > 1 ) return f( n - 1 ) + f ( n - 2 );
    return 1;
}
```

Rekurencja przykład

Zaznaczenie na planszy sąsiadujących pól o tej samej wartości.

Plansza

```
int plansza[6][20];
```

00000000000004440000
00000004444004000000
00000044004444400000
00000004444000400000
00000000000000440000
00000000000000040000

Rekurencja przykład

Dla pola (1,19)

0000000000000444**1111**
0000000444400**4111111**
0000004400444**411111**
0000000444400**411111**
0000000000000**441111**
0000000000000**041111**

Dla pola (2,8)

0000000000004440000
0000000444400400000
0000004**11**4444400000
0000000444400400000
0000000000000440000
0000000000000040000

Rekurencja przykład

```
void zaznacz( int x, int y, int k, int plansza[6][20])
{
if( plansza[y][x] != 0 ) return;
plansza[y][x] = k;
if( x > 0 ) zaznacz( x-1, y, k, plansza);
if( x < 20 ) zaznacz( x+1, y, k, plansza );
if( y > 0 ) zaznacz( x, y-1, k, plansza );
if( y < 6 ) zaznacz( x, y+1, k, plansza );
}
```

Liczby pseudolosowe

```
#include<iostream>
#include<ctime>
#include<cstdlib>
...
int liczba_losowa;

//ustawienie punktu startowego dla generatora
//liczb pseudolosowych zależnie od czasu rzeczywistego


srand(time(nullptr ));

//wygenerowanie liczby pseudolosowej z przedziału
// 0 - RAND_MAX (zazwyczaj 32767)


liczba_losowa=rand();
```

Liczby pseudolosowe

Liczba z przedziału [0,n-1]

$$n \in \{0, 1, 2, \ldots, n-1\}$$

```
int n=rand()%n;
```

Liczba z przedziału [-(n-1),n-1]

$$n \in \{-(n-1), \ldots, -1, 0, 1, \ldots, n-1\}$$

```
int n=rand()%n;
int z=2*(rand()%2)-1;  // -1 albo 1
n*=z;
```

Liczby pseudolosowe

Liczba z przedziału [0,2)

Liczba zmiennoprzecinkowa $x \in [0, 2)$ z dokładnością do 3 miejsc po przecinku

```
int c=rand()%2;
int r=rand()%1000;
double x=c+r*0.001;
```

Losowa litera ze zbioru $\{a, v, c, k, o, l, p\}$

```
char litery[]="avckolp";
char z=litery[rand()%7];
```