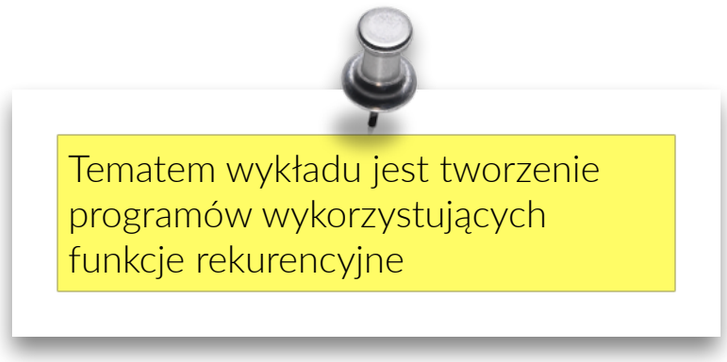
Informatyka 2

Rekurencja
(wersja robocza komentarzem)

Paweł Strączyński
pstraczynski@tu.kielce.pl



Co to jest rekurencja?

Rekurencja (rekursja) to taki przypadek w programowaniu czy matematyce gdy funkcja lub definicja odwołuje się do samej siebie.

Rekurencja pozwala rozwiązać problem w oparciu o rozwiązanie tego samego problemu ale dla danych o mniejszym rozmiarze.

Aby algorytm rekurencyjny „mógł się zatrzymać” w jego kolejnych wywołaniach (odwołaniach funkcji do samej siebie) warunek wywołania musi się zmieniać i dążyć aż do tzw. **warunku podstawowego**.

Rekurencja pozwala rozwiązać wiele złożonych problemów w łatwy sposób.

Korzystając z funkcji rekurencyjnych należy zwrócić uwagę na właściwą dekompozycje problemu oraz właśnie określenie warunku zakończenia wywołania rekurencyjnego.

Silnia - wersja iteracyjna

$$n! = 1 \cdot 2 \cdot \ldots \cdot n$$

```c
/* silnia_iter.c */

#include <stdio.h>

int n=0, i=1, silnia=1;

int main ()
{
    printf("podaj liczbe N:");
    scanf("%d",&n);
    while(i <= n)
    {
     silnia = silnia * i;
     i = i + 1;
    }
    printf("%d!=%d", n, silnia);

}
```

Silnia - wersja iteracyjna

$$3! = 1 \cdot 2 \cdot 3 = 6$$

I iteracja

1 1

```
i=1; n=3; silnia = silnia*i=1;
```

II iteracja

1 2

```
i=2; n=3; silnia = silnia*i=2;
```

III iteracja

2 3

```
i=3; n=3; silnia = silnia*i=6;
```

```
/* silnia_iter.c */

#include <stdio.h>

int n=0, i=1, silnia=1;

"int main ()
{
    printf("podaj liczbe N:");
    scanf("%d",&n);
    while(i <= n)
    {
     silnia = silnia * i;
     i = i + 1;
    }
    printf("%d!=%d", n, silnia);

}
```

Przykład dostępny w postaci wideo

Silnia - wersja rekurencyjna

$$n! = 1 \cdot 2 \cdot \ldots \cdot n$$

warunek podstawowy

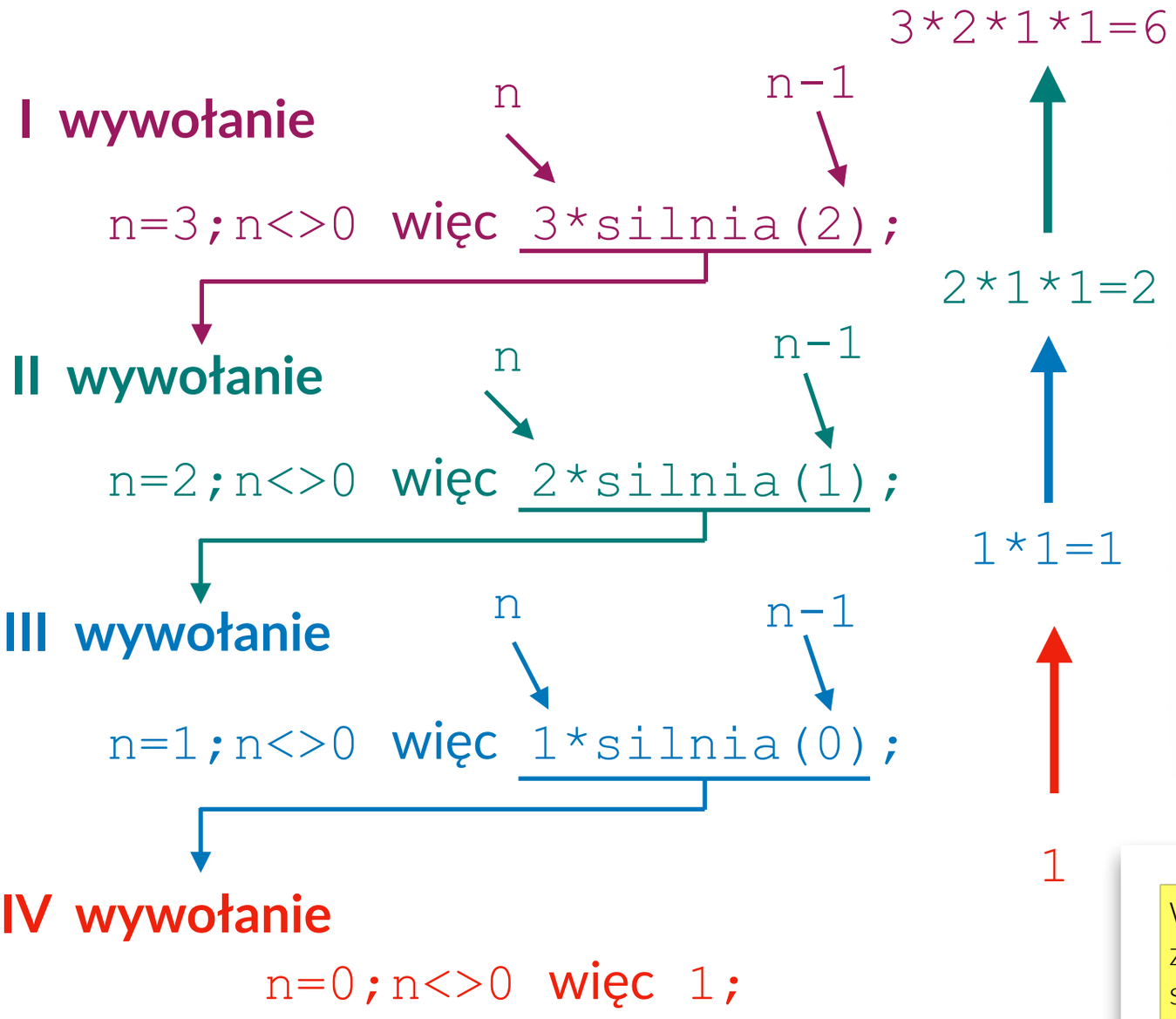
$$silnia(n) = \begin{cases} 1 & , n = 0 \\ n \cdot silnia(n-1) & , n > 0 \end{cases}$$

```c
/* silnia_rekur.c */

#include <stdio.h>

int n=0, i=1;

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

int main ()
{
    printf("podaj liczbe N:");
    scanf("%d",&n);
    printf("%d!=%d", n, silnia(n));

}
```

Silnia - wersja rekurencyjna

Przykład dostępny w postaci wideo

$$3! = 1 \cdot 2 \cdot 3 = 6$$




```c
/* silnia_rekur.c */

#include <stdio.h>

int n=0;

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

int main ()
{
    printf("podaj liczbe N:");
    scanf("%d",&n);
    printf("%d!=%d", n, silnia(n));

}
```

W praktyce podstawiając pod kolejne wywołania funkcji wartości zwracane otrzymujemy :
silnia(3)=n*silnia(2) = 3*2*silnia(1)=3*2*1*silnia(0)=3*2*1*1=6

Ciąg Fibonacciego

Ciąg Fibonacciego to ciąg liczb naturalnych określony w sposób rekurencyjny którego dwa pierwsze wyrazy równe są 1 a każdy kolejny jest sumą dwóch poprzednich.

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

1 1 2 3 5 8 13 21 34 55 89 144 233 ...

Wartość n-tego elementu ciagu Fibonacciego jest sumą dwóch elementów poprzednich chyba że wyznaczamy wartość elementu 0 i 1 którego wartość zawsze jest równa 1 - warunek podstawowy

```c
/* fiboncacci_rek.c */

#include <stdio.h>

int n=0;

int fib(n)
{
    if(n==0) || (n==1) return 1;
    else return fib(n-1)*fib(n-2);
}

int main ()
{
    printf("ktory elementy ciagu?:”);
    scanf("%d",&n);
    printf("fib(%d)=%d”, n, fib(n));

}
```

Przykład dostępny w postaci wideo

Ciąg Fibonacciego

1 1 2 3 5 8 13 21 34 55 89 144 233 ...

7 element ciągu

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