

# Wstęp do programowania

## - wykład 8

dr Piotr Jastrzębski

# Funkcje w C++

Ogólna składnia funkcji

```
typ identyfikator (typ1 argument1, typ2 argument2)
{
    /* instrukcje */
}
```

procedura - podprogram nie zwracający wyniku, realizujący pewną funkcjonalność

```
void identyfikator (typ1 argument1, typ2 argument2)
{
    /* instrukcje */
}
```

Instrukcja return powoduje zakończenie wykonywania funkcji i zwrócenie wartości. Może być ona użyta dowolną ilość razy w kodzie funkcji.

Funkcja - podprogram zwracający wynik na podstawie przekazanych argumentów

```
typ identyfikator (typ1 argument1, typ2 argument2)
{
    /* instrukcje */
    return ...;
}
```

Funkcja czysta - funkcja, która nie ma żadnych skutków ubocznych, tzn. nie modyfikuje ani przekazanych argumentów, ani globalnego stanu programu.

Przykłady:

Funkcja bez argumentu zwracająca int:

```
int foo1()  
{  
    return 62;  
}
```

Funkcja z jednym argumentem zwracająca int

```
int foo2(int a)
{
    return a+4;
}
```

Funkcja z dwoma argumentami zwracająca float

```
float foo3(int a, int b)
{
    return (b+a)/3.0;
}
```

Procedura bez argumentu:

```
void foo4()  
{  
    cout << "abc";  
}
```

Procedura z jednym argumentem

```
void foo5(int a)
{
    cout << a*8<<endl;
}
```

Wszystkie wcześniejsze funkcje były “czyste”.



Czy ta funkcja jest czysta?

```
#include <iostream>
using namespace std;

void foo10(int a)
{
    a+=5;
    cout <<"wewnatrz" <<a <<endl;
}

int main()
{
    int a = 7;
    cout <<"przed"<<a <<endl;
    foo10(a);
    cout <<"po"<<a <<endl;
    return 0;
}
```

## Argumenty domyślne

```
#include <iostream>

using namespace std;

int suma(int a, int b, int c = 0, int d = 0)
{
    return a + b + c + d;
}

int main()
{
    cout << suma(3,4) <<endl;
    cout << suma(1,2,3) <<endl;
    return 0;
}
```

## Przeciążanie funkcji

```
#include <iostream>
using namespace std;

void print(int i) {
    cout << "int:" << i << endl;
}

void print(double f) {
    cout << "float:" << f << endl;
}

void print(char c) {
    cout << "char:" << c << endl;
}

int main() {
    print(10);
    print(10.10);
    print('t');
    return 0;
}
```

# Funkcje matematyczne

<https://en.cppreference.com/w/cpp/numeric/math>

## Prototypy funkcji

Prototyp funkcji mówi kompilatorowi jakiego typu wartość jest zwracana przez funkcję oraz jakie są jej parametry. Dzięki temu możliwe jest podczas kompilacji sprawdzenie czy w poprawny sposób wykorzystywana jest wartość zwracana przez funkcję oraz czy podawane są parametry odpowiednich typów. Ponadto, prototypy pozwalają szybko zorientować się jakie funkcje są dostępne i jak z nich korzystać.

Definicja funkcja mówi o tym co funkcja robi. Zawiera instrukcje umożliwiające realizację odpowiedniego fragmentu programu. W odróżnieniu od deklaracji przydzielany jest obszar pamięci, w którym znajdzie się kod wynikowy funkcji.

# Rekurencja

Rekurencyjne wywołanie funkcji polega na ponownym jej wywołaniu jeszcze przed jej zakończeniem. Przy każdym wywołaniu rekurencyjnym tworzona jest kopia środowiska funkcji łącznie ze wszystkimi parametrami i zmiennymi lokalnymi.

Wywołanie każdej funkcji powoduje umieszczenie na stosie:

- ▶ informacji umożliwiającej powrót z funkcji do miejsca wywołania,
- ▶ parametrów funkcji,
- ▶ zmiennych lokalnych (automatycznych) istniejących w momencie wywołania funkcji.



Rekurencja, zwana także rekursją (ang. recursion, z łac. recurrere, przybiec z powrotem) – odwoływanie się np. funkcji lub definicji do samej siebie.

Przykłady:

- ▶ silnia

$$0! = 1, \quad n! = (n - 1)! \cdot n$$

- ▶ ciąg Fibonacciego

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

```
#include <stdio.h>
#include <stdlib.h>

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

int main()
{
    int a = 5;
    printf("%d\n",silnia(a));
    return 0;
}
```

```
#include <stdio.h>
#include <stdlib.h>

int fib(int n)
{
    if (n == 0)
        return 0;
    if (n == 1)
        return 1;
    return fib(n-1)+fib(n-2);
}

int main()
{
    int a = 6;
    printf("%d\n",fib(a));
    return 0;
}
```