

Programowanie w języku C++ (EAR1S03006)

Politechnika Białostocka - Wydział Elektryczny
Automatyka i Robotyka, semestr III, studia stacjonarne I stopnia
Rok akademicki 2021/2022

Zajęcia nr 9 (08.12.2021)

dr inż. Jarosław Forenc

Szablony funkcji

- wyznaczenie większej wartości z dwóch liczb (max) wymaga napisania oddzielnej funkcji dla każdego możliwego typu jej argumentów

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

int maks(int a, int b)
{
    return a>b ? a : b;
}

double maks(double a, double b)
{
    return a>b ? a : b;
}

float maks(float a, float b)
{
    return a>b ? a : b;
}
```

```
int main(void)
{
    int    a1 = 3, b1 = 5;
    double a2 = 2.3, b2 = -1.2;
    float  a3 = 4.1f, b3 = 1.6f;

    cout << "int:    " << maks(a1,b1);
    cout << endl;

    cout << "double: " << maks(a2,b2);
    cout << endl;

    cout << "float:  " << maks(a3,b3);
    cout << endl;
}
```

```
int:    5
double: 2.3
float:  4.1
```

Szablony funkcji

- ❑ tworząc **szablon funkcji** tworzymy kod działający na nieopisanych jeszcze typach
- ❑ funkcja działa na typach ogólnych - w momencie wywołania funkcji pod te typy ogólne podstawiane są konkretne typy (np. **int**, **char**)

```
template <typename T>  
T MojaFunkcja(T a, T b, ...)  
{  
    // instrukcje  
}
```

- ❑ słowo kluczowe **template** informuje, że funkcje będzie korzystała z fikcyjnego typu o nazwie **T**
- ❑ definicja dotyczy tylko jednej funkcji zdefiniowanej bezpośrednio po **template**

Szablony funkcji - przykład


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

template <typename T>
T maks(T a, T b)
{
    return a>b ? a : b;
}

int main(void)
{
    int    a1 = 3, b1 = 5;
    double a2 = 2.3, b2 = -1.2;
    float  a3 = 4.1f, b3 = 1.6f;

    cout << "int:    " << maks(a1,b1) << endl;
    cout << "double: " << maks(a2,b2) << endl;
    cout << "float:  " << maks(a3,b3) << endl;
}

template <typename T> T maks(T a, T b)
{
    return a>b ? a : b;
}
```



```
int:    5
double: 2.3
float:  4.1
```

Szablony funkcji

- kompilator określa konkretny typ na podstawie typu argumentów w wywołaniu funkcji (proces ten nazywa się **konkretyzacją szablonu**)

```
int a1 = 3, b1 = 5;  
cout << "int:      " << maks(a1,b1) << endl;
```

- jeśli typy nie zgadzają się, to wystąpi błąd kompilacji

```
cout << "inne:      " << maks(2.1f,5) << endl;
```

```
error C2782: 'T maks(T,T)' : template parameter 'T' is ambiguous  
.\main.cpp(5) : see declaration of 'maks'  
could be 'int '  
or          'float'
```

- w takim przypadku należy jawnie określić typ

```
cout << "inne:      " << maks<float>(2.1f,5) << endl;
```

Szablony funkcji - definicja / deklaracja

- definicja szablonu funkcji:

```
template <typename T>  
T maks (T a, T b)  
{  
    return a>b ? a : b;  
}
```

- deklaracja szablonu funkcji:

```
template <typename T> T maks (T a, T b);
```

lub

```
template <typename T> T maks (T, T);
```

Szablony klas

- **szablony klas** (wzorce klas, klasy uogólnione) definiuje się na podobnej zasadzie jak szablony funkcji

```
template <class T>
class MojaKlasa
{
    // definicja klasy
};
```

- słowo kluczowe **template** informuje, że klasa będzie korzystała z fikcyjnego typu o nazwie **T**
- definicja dotyczy tylko pojedynczej klasy zdefiniowanej bezpośrednio po **template**

```
template <class T> class MojaKlasa
{
    // definicja klasy
};
```

Szablony klas - przykład

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

template <class T>
class MojaKlasa
{
public:
    T zmienna;
    MojaKlasa(T x)
    {
        zmienna = x;
    }
    void drukuj()
    {
        cout << zmienna;
        cout << endl;
    }
};
```

```
int main(void)
{
    MojaKlasa<int> MK_int(21);
    MK_int.drukuj();

    MojaKlasa<float> MK_float(3.5f);
    MK_float.drukuj();

    MojaKlasa<char> MK_char('a');
    MK_char.drukuj();

    MojaKlasa<string> MK_string("Txt 1");
    MK_string.drukuj();
}
```

```
21
3.5
a
Txt 1
```


Szablony klas - przykład

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

template <class T>
class MojaKlasa
{
public:
    T zmienna;
    MojaKlasa(T x);
    void drukuj();
};
```

```
template <class T>
MojaKlasa<T>::MojaKlasa(T x)
{
    zmienna = x;
}

template <class T>
void MojaKlasa<T>::drukuj()
{
    cout << zmienna;
    cout << endl;
}

int main(void)
{
    MojaKlasa<int> MK_int(21);
    MK_int.drukuj();

    ...
}
```