Informatika 3

Šablóny

12

Šablóny (template)

Vzor pre:

- Popis množiny funkcii
- Popis množiny tried
- Skupina premenných – odlíšené typom

Explicitné vytvorenie inštancie šablóny

- Inštancie šablón sa vytvárajú automaticky, keď sú potrebné
- Definíciu tela šablóny musíme dávať do hlavičkových súborov (pred volanie)
- Vytvorenie inštancie šablóny v module bez volania

```
template long max<long>(long, long);
```

Šablóny (template)

- Náhrada makier
- Inštancia šablóny

Šablóny - deklarácia

template <zoznam-formálnych-parametrov> deklarácia

- Hodnotové
- Typové
- Šablónové

Šablóny – typové parametre

```
template <typename T = double> class komplex;
```

alebo

```
template <class T = double> class komplex;
```

Šablóny – hodnotové parametre

- **Celočíselné typy**
- **Vymenovacie typy**
- **Smerník na objekt**
- **Odkaz objekt**
- **Smerník na funkciu**
- **Odkaz na funkciu**
- **Smerník na triedu**
- **std::nullptr_t**

```
template <auto hodnota> void F() {  … }
F<10>();
```

Šablóny – šablónové parametre

- **Iná šablóna objektového typu**

```
template <typename S, template <typename T> typename R>
class kontajner
 {
     R<S> data;
 }
```

Inštancia šablóny - parameter inej šablóny

```
template <typename T = vector<int>>
struct A
 {
  …
 }
```

Šablónová trieda – inštancia šablóny

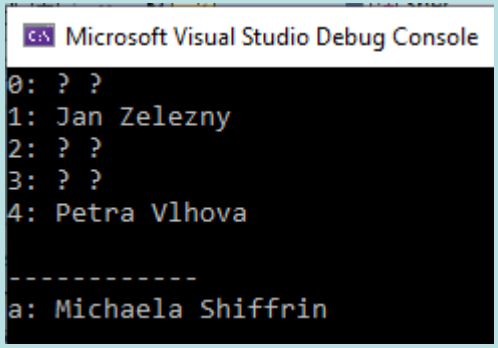
```
#pragma once
#include <iostream>
using namespace std;

class Student
{
    string meno = "?";
    string priezvisko = "?";
public:
    Student() {}
    Student(const char* pmeno, const char* ppriezvisko)
        : meno(pmeno), priezvisko(ppriezvisko)
    {}
    friend ostream& operator<<(ostream& os, Student& val)
    {
        os << val.meno << " " << val.priezvisko;
        return os;
    }
};
```

Šablónová trieda – inštancia šablóny

```cpp
#pragma once
#include <iostream>
using namespace std;

template<typename T, int N>
class Pole {
    T v[N];
    static T nula;
public:
    Pole();
    T& operator[ ](int i) { return i >= 0 && i < N ? v[i] : nula; }
    friend ostream& operator<<(ostream& os, Pole<T, N>& val)
    {
        for (int i = 0; i < N; i++)
        {
            os << i << ": " << val[i] << endl;
        }
        return os;
    }

};

template<typename T, int N> T Pole<T, N>::nula;

template <typename T, int N>
Pole<T, N>::Pole()
{
    for (int i = 0; i < N; i++)
    {
        v[i] = nula;
    }
}
```

Šablónová trieda – inštancia šablóny

```cpp
#include "Pole.h"
#include "Student.h"

int main()
{
    Student s1("Jan", "Zelezny");
    Student s2("Petra", "Vlhova");
    Student s3("Michaela", "Shiffrin");
    Pole<Student, 5> pole;
    pole[1] = s1;
    pole[4] = s2;

    pole[-5] = s3;
    Student a = pole[-5];
    cout << pole;
    cout << endl << "------------\na: " << a << endl;
    return 0;
}
```



Šablónová trieda – inštancia šablóny

template <class T, class R> struct A;

A<double, bool> p(3.8, false);

C++17

A p(3.8, false);

Funkčná šablóna

Namiesto písania viacerých funkcií

```
int max(int x, int y)                          {return x > y ? x : y;}
float max(float x, float y)                    {return x > y ? x : y;}
double max(double x, double y)                 {return x > y ? x : y;}
long max(long x, long y)                       {return x > y ? x : y;}
complex max(complex x, complex y)              {return x > y ? x : y;}
```

Príklad

- Všetky funkcie môžeme nahradiť šablónou

```
template <class T>
T max(T x, Ty)
{
   return (x > y) ? x : y;
}

main()
{
   int i=3, j=5;
   int k=max<int>(i, j);      // volanie šablony
   int k=max (i, j);          // nie je nutné písať max<int>
}
```

- Aby sme mohli zavolať šablónu:

```
complex p(3,5), q(10,20);
complex d = max(p, q);
```

- musí existovať operátor

```
bool operator>(const complex& x, const complex& y);
```

Priorita volania

- Ak nadefinujeme funkciu max s parametrami **double**:

```
double max(double x, double y) { return 55; }

main()
{
   int x=10, y=20;
   int k=max(x,y);  // volá sa šablóna (k=20)
   double p=30, q=40;
   double w=max(p,q); // volá sa funkcia (w=55)
}
```

- Pravidlo:
  - Ak existuje pre daný typ parametrov funkcia, zavolá táto funkcia
  - Ak neexistuje, prekladač zistí, či nemôže vytvoriť inštanciu (špecializáciu) šablóny.
  - Ak nie je k dispozicii ani funkcia, ani šablóna, prekladač vyhlási chybu

Šablóny premenných

template <zoznam-parametrov> deklarácia-premennej;

```
// C++14
template<typename T>
// Presnosť je daná šablonovým parametrom.
constexpr T pi = T(3.14159265358979323846);


 // Použitie:
template<typename T>
T PlochaKruhu(T r) { return pi<T> * r * r; }

double p = PlochaKruhu<double>(10.3);
float p = PlochaKruhu<float>(10.3);
```

constexpr

```
// C++98
int f() { return 2; }
int arr[f()]; // ! chyba


// C++11
constexpr int f() { return 2; }
int arr[f()]; // OK
```