12. Cvičení

Úvod do šablon.

Šablony

- Generické programování:
  - Mechanismus umožňující vytvářet generické funkce/třídy.
  - Polymorfismus mezi třídami, které nemají společného předka.
  - Generický program „instancujeme“ na konkrétní datové typy.

Šablony funkcí

- Chceme vytvořit funkci compare, která vrací informaci který za dvou vstupních objektů je větší. V našem programu chceme porovnávat např. double.

```
int compare(const double &v1, const double &v2) {
        if (v1 < v2) return -1;
        if (v2 < v1) return 1; return 0;
}
```

Šablony funkcí

- Program se nám rozrůstá a najednou zjistíme, že by se nám hodilo stejně porovnávat i stringy.
- Řešení:
  - Přetížíme funkci compare i pro string.
  - Vytvoříme šeblonu funkce pro jakýkoli typ.

Šablony funkcí

```
template <typename T>
int compare(const T &v1, const T &v2)
{
    if (v1 < v2) return -1;
    if (v2 < v1) return 1;
    return 0;
}
```

Šablony funkcí

- Šablona funkce slouží k zobecnění použití funcke pro různé datové typy.
- (typy/hodnoty) Šablonových parametrů musí být odvoditelné z parametrů funkce.
- Definice:

  template < typename *i1,* typename *i2, …* >

  definice_funkce

Šablony funkcí

```
int main () {
    // T je int;
    // int compare(const int&, const int&)
    cout << compare(1, 0) << endl;
    // T je string;
    // int compare(const string&, const string&)
     string s1 = "hi", s2 = "world";
    cout << compare(s1, s2) << endl;
    return 0;
}
```

Šablony funkcí

```
// inicializace pole
 <class T, size_t N> void array_init(T (&parm)[N])
   {
        for (size_t i = 0; i != N; ++i) {
                parm[i] = 0;
        }
   }
   int x[42]; double y[10];
   array_init(x); // instantiates array_init(int(&)[42]
   array_init(y); // instantiates array_init(double(&)[10]
```

Šablony jsou mocné

```
#include <iostream>
using namespace std;
template<int N> class factorial {
public:
    enum {value=N*factorial<N-1>::value};
 };

template<> class factorial<1> {
public:
   enum {value=1};
};

int main() {
            const int c = factorial<3>::value;
            cout << "Faktorial 3 = " << c << endl;
 }
```

Šablony tříd

- Využivá se hojně v std pro implementaci kontejnerů (vector, list, queue, ...)
- Šablony umožňují vytvořit šablonu třídy, která není závislá na konkrétním typu.
- Konkrétní třída se vytvoří při překladu jejím instancováním pro konkrétní datový typ.

Šablony tříd

```
template <class Type> class Queue {
public:
    Queue (); // default constructor
    Type &front (); // return element from head of Queue
    const Type &front () const;
    void push (const Type &); // add element to back of
    Queue
    void pop(); // remove element from head of Queue
    bool empty() const; // true if no elements in the Queue
private: // ...
};
```

Zadání cvičení

- Stáhněte si archiv přiložený ke cvičení 12, pokyny se nacházejí uvnitř ☺