4. lekce

Šablony funkcí a tříd

Šablona funkce

- šablony funkcí vytváříme, abychom nemuseli psát více funkcí pro různé datové typy v případě, že vykonávají stejnou činnost
- taková funkce *(mimo definici objektu)*, u které není dán datový typ (návratové hodnoty) nebo (ani) datový typ jejich parametrů
- datový typ funkce a jejich parametrů se zvolí podle toho, s jakými parametry danou funkci zavoláme
- pro šablony se používá klíčové slovo **template**

Příklad šablony funkce:

```
template <typename T>
T maximum(T a,T b)
{
    if (a>b) return a;
    return b;
}
```

Volání z main: *maximum(p1, p2);*

Parametr „a" a „b" musí být stejného datového typu, datovým typem může být i objekt.
V případě objektů musí být definován operátor „>".

Příklad šablony pro převod parametru na string

```
template <typename T>
string fcetostring(const T & x)
// const T &– objekt, který je do funkce poslán jako parametr, se ve funkci nebude měnit
{
    stringstream s;
    // datový typ stringstream, knihovna <sstream>;
    s<<x;
    // u objektu musi být definovaný operator <<
    return s.str();
    // ze streamu se vezme existující string (vše se převede na string)
}
```

Šablony tříd

- šablony tříd vytváříme, abychom nemuseli psát více objektů (class - třída) pro různé datové typy v případě, že vykonávají stejnou činnost
- jsou takové třídy (objekty), které nemají definovaný datový typ atributů nebo metod
- datový typ se zvolí při vytváření třídy v programu (programátorem v kódu)

Příklad šablony třídy (class = objekt):

```
template <typename T>
class pole
{
    T * m_data;
    int m_delka;
    //... dále konstruktor, destruktor, kopírující konstruktor, operátor =, operátor []
};
```

...
v mainu či funkci programu:

```
pole<datovytyp> polea;

//pole konkrétního datového typu, jméno proměnné je polea
```

Pokud chceme mít více datových typů v šabloně, tak za klíčovým slovem template ve ostrých závorkách <> uvedeme více typename oddělených čárkami.

Příklad:

teplate <typename T1, typename T2>