Šablony tříd

Zápis šablony:

```
template<..parametry..> definice_třídy
```

Deklarace parametru šablony, který označuje typ, může být:

```
class identifikátor
typename identifikátor
template<..parametry..> class identifikátor
```

Příklad. Sestavíme šablonu třídy pro frontu.

```
template<class T,template<class S> class U>
class Fronta {
    U<T> *prvni,*posledni;
  public:
    Fronta(): prvni(nullptr) { }
    Fronta<T,U> & operator << (const T &p)
    {
      auto u=new U<T>(p);
      if (prvni==nullptr) prvni=posledni=u;
      else { posledni->nasled=u; posledni=u; }
      return *this;
    }
    bool operator >> (T &p)
    {
      if (prvni==nullptr) return false;
      p=prvni->prvek;
      auto u=prvni; prvni=prvni->nasled; delete u;
      return true;
    }
};
```

Frontu budeme realizovat seznamem, sestavíme si šablonu pro uzel seznamu:

```
template<class T>
struct Uzel { T prvek;
              Uzel *nasled;
       Uzel(const T &p) { prvek=p; nasled=nullptr; }
};

Fronta<int,Uzel> f;
f << 3 << 5;

int m;
```

```
while (f >> m) cout << m << endl;
```

Šablona třídy může mít implicitní hodnoty parametrů. Platí pro ně obdobné zásady jako pro implicitní hodnoty parametrů funkcí:

- Datový typ implicitní hodnoty musí odpovídat typu parametru.
- Implicitní hodnoty lze uvést od libovolného parametru.
- Při použití šablony lze skutečné parametry od libovolného parametru s implicitní hodnotou vynechat. Použijí se implicitní hodnoty.

Příklad. V předchozí šabloně uvedeme u druhého parametru šablony `Fronta` implicitní hodnotu, kterou bude šablona uzlu seznamu.

```
template<class T,template<class S> class U=Uzel>
class Fronta { ... };
```

Nyní můžeme frontu pro uložení celých čísel deklarovat bez uvedení druhého parametru šablony:

```
Fronta<int> f;
```

Příklad. V předchozí šabloně lze uvést datový typ prvku ukládaného do fronty a prvku ukládaného do uzlu stejný.

```
template<class T,template<class T> class U=Uzel>
class Fronta { ... };
```

Příklad. V případě, že bychom neuvažovali jinou možnost použití struktury pro uložení prvků v šabloně `Fronta` než šablonu `Uzel`, nemusíme šablonu `Uzel` předávat v šabloně `Fronta` přes parametr. Jako ukázka, jak se zapíší členské funkce vně šablony, nejsou zde definice operátorů napsané uvnitř šablony.

```
template<class T>
class Fronta {
    Uzel<T> *prvni,*posledni;
  public:
    Fronta(): prvni(nullptr) { }
    Fronta<T> & operator << (const T &);
    bool operator >> (T &);
};

template<class T>
Fronta<T> & Fronta<T>::operator << (const T &p)
{
  auto u=new U<T>(p);
  if (prvni==nullptr) prvni=posledni=u;
  else { posledni->nasled=u; posledni=u; }
```

```
  return *this;
}

template<class T>
bool Fronta<T>::operator >> (T &p)
{
  if (prvni==nullptr) return false;
  p=prvni->prvek;
  auto u=prvni; prvni=prvni->nasled; delete u;
  return true;
}

Fronta<int> f;
f << 3 << 5;
```

Dědění šablon

```
template<class T,unsigned n>
class Hash { };

template<class T,unsigned n>
class HashA: public Hash<T,n> { };

class Zlomek { };

template<unsigned n>
class HashZlomky: public Hash<Zlomek,n> { };

template<class T>
class Hash100: public Hash<T,100> { };
```