Šablony

Šablony jsou generické funkce nebo třídy, ve kterých jsou parametrem datové typy nebo celočíselné hodnoty.

Šablony funkcí

Zápis šablony:

```
template<..parametry..> definice_funkce
```

Je-li parametrem datový typ, lze pro jeho označení použít klíčová slova:

```
class
typename
```

Pro identifikátory parametrů, které označují datové typy, se obvykle používají velká písmena (T apod.).

Příklad. Funkce počítající absolutní hodnotu.

```
template<typename T>
inline T aH(T a) { return a>=0 ? a : -a; }

template<class T>
inline T aH(T a) { return a>=0 ? a : -a; }
```

Volání funkce:

```
jméno_funkce<..skutečné_datové_typy..>(..argumenty..)
```

Pokud lze skutečné datové typy odvodit z argumentů volání funkce, lze jejich uvedení ve volání funkce vynechat.

Příklad. Volání funkce z předchozího příkladu.

```
aH<int>(-3)        aH(-3)
aH<double>(-3.1)   aH(-3.1)
aH<float>(-3.1)    aH(-3.1f)
```

Příklad. Šablona funkce třídění Quicksort.

```
template<class T>
void quicksort(T a[],int k,int l)
{
  T x=a[(k+l)/2];
  int i=k,j=l;
  do { while (a[i]<x) ++i;
       while (x<a[j]) --j;
       if (i>j) break;
       T w=a[i];
         a[i]=a[j];
         a[j]=w;
```

```
      ++i; --j; } while (i<=j);
  if (k<j) quicksort(a,k,j);
  if (i<l) quicksort(a,i,l);
}

float a[] = { 3.5,1,8,2.2,5,7 };
quicksort(a,0,5);
```

Příklad. Šablona funkce, která má dva parametry datových typů.

```
template<typename T,typename S>
inline bool jeVIntervalu(T x,S a,S b) { return a<=x && x<=b; }

jeVIntervalu<double,int>(-3.1,0,10)

jeVIntervalu<double>(-3.1,0,10)

jeVIntervalu(-3.1,0,10)
```

Vedle parametrů reprezentující datové typy můžeme mít parametry reprezentující celočíselné hodnoty. Patří sem všechny datové typy, které mají charakter celočíselných hodnot:

- celočíselné datové typy (`int` apod.)
- výčtový typ
- ukazatel
- reference

Příklad. Šablona funkce, která transponuje matici *a* řádu *m*×*n* a výsledek ukládá do matice *b* řádu *n*×*m*.

```
template<class T, int m, int n>
void transp(const T a[m][n], T b[n][m])
{
  for (int i=0;i<m;++i) for (int j=0;j<n;++j) b[j][i]=a[i][j];
}

float a[2][3]={{7,4,1},{2,5,3}}, b[3][2];

transp<float,2,3>(a,b);
```

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 zásobník. Zásobník budeme implementovat seznamem.

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

Sestavíme si šablonu pro uzel seznamu:

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

Zasobnik<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 argumenty od libovolného parametru s implicitní hodnotou vynechat. Použijí se implicitní hodnoty.

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

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

Nyní můžeme zásobník pro uložení celých čísel deklarovat bez uvedení druhého argumentu šablony:

```
Zasobnik<int> f;
```

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

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

Příklad. V případě, že bychom neuvažovali jinou možnost použití struktury pro uložení prvků v šabloně `Zasobnik` než šablonu `Uzel`, nemusíme šablonu `Uzel` předávat v šabloně `Zasobnik` přes parametr. Následující kód je rovněž ukázka, jak se zapíší členské funkce vně šablony. Definice operátorů jsou zde napsané vně šablony.

```
template<class T>
class Zasobnik {
    Uzel<T> *vrchol;
  public:
    Zasobnik():vrchol (nullptr) { }
    Zasobnik<T> & operator << (const T &);
    bool operator >> (T &);
};

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

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

```
Zasobnik<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 HashZlomek: public Hash<Zlomek,n> { };

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