Základy programování v C++ - 19. cvičení
Preprocesor
Lineární spojový seznam – pokračování

Zuzana Petříčková

27. listopadu 2019

Přehled

Průběh překladu programu v C++ (opakování)

1. zpracování každého zdrojového souboru **preprocesorem**
   - odstraní komentáře
   - zpracuje direktivy (začínají #, např. #include)
   - ...
2. zpracování každého zdrojového souboru **překladačem (kompilátorem)**
   - výsledkem jsou **relativní soubory** (.obj, .o), které obsahují (binární) strojový kód
3. **sestavovací program (linker)** sestaví z relativních souborů spustitelný program
   - kontroluje, zda našel definice všech použitých funkcí a proměnných

Preprocesor

1. odstraní ze zdrojového souboru komentáře
2. zpracuje tzv. direktivy (začínají #)
   - direktiva **#include** ... vložení jednoho souboru do jiného
   - direktiva **#define** ... definice tzv. makra
   - direktiva **#undef** ... zrušení makra
   - podmíněný překlad ... direktiva **#if** a související, **#ifdef, #ifndef**
   - direktiva **#pragma** a další

Preprocesor

direktiva #include

- vložení jednoho souboru do jiného (typicky hlavičového do zdrojového)

```
#include<iostream> /* soubor se hleda v adresarich vyhrazenych
                      pro standardni hlavickove soubory */
#include"Seznam.h" /* soubor se hleda v adresari
                      prekladaneho programu */
```

Makro

- identifikátor, který preprocesor nahradí zadanou posloupností znaků

```
// definuji makro:
#define N 100
...
int a[N];
...
// rusim makro:
#undef N
```

Preprocesor

Podmíněný překlad

- části kódu můžeme ,,zakomentovat" a nebo ,,odkomentovat"
- direktivy určují, které části kódu se budou překládat a které ne

```
#if podminka // napr. #if 0
 ... // kus kodu
#endif

#if podminka
 ... // kus kodu
#elif podminka
 ... // kus kodu
#else
 ... // kus kodu
#endif
```

Preprocesor a podmíněný překlad

- využití maker a direktiv #ifdef, #ifndef

Makra bez parametrů (manifestační konstanty)

```
// definuji makro:
#define MOJE_MAKRO

...

// rusim makro:
#undef MOJE_MAKRO
```

Preprocesor a podmíněný překlad

- využití maker a direktiv #ifdef, #ifndef

```
#define MOJE_MAKRO

#ifdef MOJE_MAKRO

... /* kus kodu, ktery se provede jen kdyz je makro
       definovano */
#endif

#ifdef MOJE_MAKRO
... // kus kodu
#else
... // kus kodu
#endif
```

Preprocesor a podmíněný překlad

Ukázka: spojový seznam a různý typ dat, která do něho vkládáme

hlavičkový soubor:

```
#define ZAMESTNANEC

// prikazy se provedou, jen je-li makro definovano:
#ifdef ZAMESTNANEC
struct Zamestnanec {
  std::string jmeno;
  std::string prijmeni;
  int plat;
};
using T = Zamestnanec;
#else
using T = int;
#endif
```

Preprocesor a podmíněný překlad

```
#ifdef ZAMESTNANEC
int main()
{
        Zamestnanec pepa = {"Josef","Novak",50000};
        Seznam s;
        vytvor(s);
        vlozNaZacatek(s, pepa);
        ...
        zrus(s);
        return 0;
}
#else
int main()
{
        Seznam s;
        vytvor(s);
        vlozNaZacatek(s, 16);
    ...
}
#endif
```

Spojový seznam ... pokračování příkladu z minule

(následující funkce stačí implementovat pro T reprezentující datový typ int)

```
Prvek *najdi(Seznam &s, T co);

void vlozZa(Seznam &s, Prvek *predchazejici, T co);
T vyjmiPrvekZa(Seznam &s, Prvek *predchazejici);
void smazPrvekZa(Seznam &s, Prvek *predchazejici);

void smazPrvek(Seznam &s, Prvek *mazany);

T* maximum(Seznam &s);
void setrid(Seznam &s);

void ulozSeznam(Seznam &s,string nazev);
void nactiSeznam(Seznam &s,string nazev);
```

Spojový seznam ... nalezení prvku s požadovanými daty

```
Prvek *najdi(Seznam &s, T co)
{
    // 1. adresu hlavy uloz do pomocneho ukazatele
    // 2. dokud se s timto ukazatelem nedostanes k zarazce:
    // 2.1. pokud jsi nalezl hledany prvek, vrat jeho adresu
    // 2.2  posun pomocny ukazatel na dalsi prvek
    // 3. vrat nullptr (prvek nebyl nalezen)
}
```

Spojový seznam ... vkládání a vyjmutí prvku

```
void vlozZa(Seznam &s, Prvek *zaKtery, T co)
{
    // 0. proved kontrolu prvku zaKtery
    //    (pokud to je nullptr nebo zarazka, konec)
    // 1. vytvor novy prvek a vloz do nej data
    // 2. naslednikem noveho prvku bude puvodni naslednik
    //    prvku zaKtery
    // 3. nasledovnikem prvku zaKtery bude novy prvek
}
T vyjmiPrvekZa(Seznam &s, Prvek *predchazejici)
{
    // 0. proved kontrolu prvku predchazejici
    //  (pokud to je nullptr, zarazka nebo posledni prvek seznamu,
    //  konec)
    // 1. adresu mazaneho prvku uloz do pomocneho ukazatele
    // 2. novym nasledovnikem prvku predchazejici bude naslednik
    //    mazaneho prvku
    // 3. data mazaneho prvku uloz do pomocne promenne
    // 4. odstran mazany prvek (operator delete)
    // 5. vrat data
}
```

Spojový seznam ... vyjmutí / smazani prvku

```
void smazPrvek(Seznam &s, Prvek *mazany)
{
    // 0. proved kontrolu prvku mazany:
    //       pokud je to nullptr nebo zarazka => konec
    // 1. pokud je naslednik mazaneho prvku zarazka:
    // 1.1. odstran puvodni zarazku (operator delete)
    // 1.2. vloz do zarazky adresu prvku mazany
    // 1.3. nastav naslednika nove zarazky na nullptr
    // 2. jinak:
    // 2.1. prekopiruj data z naslednika mazaneho prvku
    //       do prvku ,mazany'
    // 2.2. smaz nasledovnika mazaneho
    //       (napr. zavolej smazPrvekZa())
}
```

Spojový seznam ... nalezeni minima/maxima

```
T* maximum(Seznam& s)
{
    // 0. pokud je seznam prazdny, vrat nullptr
    // 1. adresu hlavy uloz do pomocneho ukazatele
    //    (hlava bude pocatecni kandidat na maximum)
    // 2. adresu naslednika hlavy vloz do jineho pomocneho
    //    ukazatele
    // 3. dokud se s timto ukazatelem nedostanes k zarazce:
    // 3.1. pokud jsi nalezl vetsi prvek
    // 3.1.1 bude to novy kandidat na maximum
    // 2.2  posun pomocny ukazatel na dalsi prvek
    // 3. vrat adresu dat obsazenych v kandidatovi na maximum
}
```

Spojový seznam ... setrideni seznamu

```
// setrideni seznamu metodou bublinkoveho trideni
void setrid(Seznam &s);
{
    // 0. pokud je seznam prazdny nebo jednoprvkovy, konec
    // 1. zaved si pomocnou promennou typu bool (indikator,
    //   zda v dane iteraci cyklu byly nejake prvky prohozeny)
    // 1. provadej nasledujici prikazy, dokud
    //   "v dane iteraci cyklu byly nejake prvky prohozeny":
    // 1.1. poznamenej si, ze (zatim) zadne prvky prohozeny nebyly
    // 1.1. adresu hlavy uloz do pomocneho ukazatele
    // 1.2. dokud se s timto ukazatelem nedostanes k zarazce:
    // 1.2.1. pokud je aktualni prvek vesti nez jeho nasledovnik:
    // 1.2.1.1. prohod data v aktualnim prvku a v jeho nasledovniku
    //          ("stara znama" funkce prohod())
    // 1.2.2.2. poznamenej si, ze nejake prvky byly prohozeny
    // 1.2.2.  posun pomocny ukazatel na dalsi prvek
}
```

Spojový seznam ... prace se soubory

```
void ulozSeznam(Seznam &s, string nazev);
void nactiSeznam(Seznam &s, string nazev);
```