Vaje 21

Tatjana Ahčin

06.04. in 07.04.

Rekurzivne podatkovne strukture (RPS)

Ukvarjali se bomo s problemom slovarja.

Najprej bomo definirali problem, ki ga hočemo rešiti, nato si bomo ogledali rekurzivno podatkovne strukture, s katerimi bomo le-tega rešili.

Problem slovarja

J jabolko apple(a:pel) .....

Prva beseda je ključna, nato sledijo pojasnila, podatki. Poznamo tri operacije nad slovarji (*dictionary*), 4. bomo omenili malo kasneje:

- Poišči ključ $\longrightarrow$ podatki
- Vnesi geslo
- Zbriši ključ (podatki so vezani na ključ)
- Obstaja(podatki) $\longrightarrow$ `true` ali `false`

Ključ je bistvena stvar, s katero pridemo do podatkov. če želimo geslo kaj popraviti, moramo ključ najprej zbrisati, potem ga spet vnesemo (z vsemi npr. dopoljnjenimi pojasnili). Vnašamo lahko le ključ, ki v slovarju še ne obstaja. Izbrišemo pa lahko le tistega, ki obstaja v slovarju.

Kaj pa, če ključ ne obstaja?

Rezervirali bomo npr. podatek *NAPAKA*, ki pove, da ključa ni v slovarju. Obstaja več možnosti:

- v primeru neobstajanja ključa vrnemo posebne podatke

- definiramo še neko novo funkcijo `Obstaja`

Napake ne bi radi povedali. To pa zato, ker smo v tem primeru omejeni pri vpisu, ki ima lahko napake v podatkih. Razred naj se sam odloči, kaj je najbolj primerno oz. kaj je zanj napaka.
Vprašanje: Ali obstaja podatek, ki bi opisoval ključ in vsa pojasnila?
Odgovor: Da, *slovarsko geslo*!

Definicija gesla

Geslo sestoji iz:

- *ključ*
- *podatki*

Definiramo razred gesel, ki jih veže skupna lastnost:

```
class Geslo {
public:
  char * ključ[32]
  char * podatki[256]

  Geslo();         // kreator
  ~Geslo();        // destruktor
}
```

Našli boste na

`Z:\PREDAVANJA\POLDE\code\Geslo\Geslo.h`

Sklad celih števil

Sklad:

> Funkcija Ustvari že določi velikost sklada na začetku. Poleg osnovnega kreatorja ustvarimo še en kreator, ki nam bo že na začetku nudil željeno velik sklad.

```
private:                  // to nam manjka v prevatnem delu
  int min_sklad           // najmanjsa velikost skladisca

  Sklad(int min_velsklad){
/* Desc:   Ustvari prazen sklad, za katerega velikost skladisca
```

```
          ni nikoli manjsa od min_velsklad.
   Parm:   /
   Result: sklad
   Pre:    min_vel_sklad>0
   Post:   Prazen()
*/

// NOV KREATOR:

  Sklad:: Sklad(int min_velikost) {
    ptr_skladisce = new int[min_vel_sklad];
    velikost = min_vel_sklad;
    zadnji = 0;
    min_velikost = 0;
    ...
} // Sklad
```

Ta program si lahko ogledate na:

Z:\PREDAVANJA\POLDE\code\Sklad\Sklad-int.cpp

Nazaj k slovarju

```
class Geslo {
public:
  char * m_kljuc;  /* m pomeni member-clan */
  char * m_podatki;
  Geslo();          /* manjka pogodba */
  ~Geslo();
  Geslo(char *kljuc, char *podatki);
}; // Geslo

class Slovar {
public:
/*---------------------USTVARJANJE
  Slovar();
/* ...
   Post: this = {} (Slovar ustvari prazno mnozico.)
*/
/*---------------------SPREMINJANJA
  Vnesi(const Geslo geslo);
```

```
/* ...
   Pre: ne obstaja geslo.m_kljuc: (m_kljuc, karkoli) je
        element this
        (Kar ze obstaja, ne smemo se enkrat vnesti.)
   Post: this'= this U geslo
         (Vrstni red ni pomemben.)
*/
/*---------------------POIZVEDOVANJA
char * Poisci(char *kljuc);    /* referenca na podatek */
/* ...
   Result: result *
   Post:   IF ((kljuc, podatki) niso elementi this) THEN
             result = NULL
           ELSE
             result = & podatki
*/
```

Popolno izvedbo definicijskega modula si oglejte na področju

Z:\PREDAVANJA\POLDE\code\Geslo\Geslo.h

Dodatna popogoja pri operaciji Poišči sta:

- uporabnik ne sme spreminjati podatkov
- rezultat je veljaven do naslednjega posega v slovar (operacije nad slovarjem)

Nadaljevanje definicijskega modula:

```
/*-------------------------SPREMINJANJA
void Zbrisi(char *kljuc);
/* ...
   Pre: (kljuc,----) je element this = Poisci(kljuc)!=NULL
   Post: this' = this \ (kljuc)
*/
```

Različni razredi skladov

Sklad števil s plavajočo vejico

Definicijski, izvedbeni in testni modul si lahko ogledate na področjih

Z:\PREDAVANJA\POLDE\code\Sklad\Sklad-tip.h

Z:\PREDAVANJA\POLDE\code\Sklad\Sklad-tip.cpp

Z:\PREDAVANJA\POLDE\test\Skald\Sklad-tip.h

Definirati smo morali novo tip tSklElt in ga v primeru opearcij nad realnimi števili definirali kot float. S tem tipom smo nato potem lahko operirali.

sklad s nizi (*string*)

Sklada s črkami ni težko narediti. Spremembe so podobne, kot pri skladu z realnimi števili glede na sklad s celimi števili. Tudi tukaj definiramo tip tSklElt kot niz char *, s katerim operiramo. Vemo, da je niz vedno prenešen po naslovu. Pomagati si moramo s funkcijo strcpy, s katero prepisujemo nize. Definicijski, izvedbeni in testni modul si lahko ogledate na področjih

Z:\PREDAVANJA\POLDE\code\Sklad\Sklad-str.h

Z:\PREDAVANJA\POLDE\code\Sklad\Sklad-str.cpp

Z:\PREDAVANJA\POLDE\test\Skald\Sklad-str.h