Preslikavanje (Mapping)

Preslikavanje je parcijalna funkcija $M : domain \to range$, tj. skup s elementima oblika (d, r), gdje je d iz *domain* i r iz *range*, u kojem se za svaki d pojavljuje najviše jedan element (d, r).

a.t.p. Mapping

```
domain              . . .
range               . . .
Mapping             . . .
MaMakeNull(&M)      . . .
MaAssign(&M,d,r)    . . .
MaDeassign(&M,d)    . . .
MaCompute(M,d,&r)   . . .
```

Zadatak 1.

Napišite definicije tipova i funkciju `MaAssign` kod implementacije preslikavanja pomoću hash-tablice.

Rješenje. Tablica ima B pretinaca. Svaki pretinac je vezana lista; svaki element vezane liste čuva jedan uređeni par $(d, M(d))$. Hash-funkcija je h: $domain \to \{0, 1, \ldots, B-1\}$.

```
#define B ...

typedef struct _celltype {
    domain domainelement;
    range rangeelement;
    struct _celltype* next;
} celltype;

typedef celltype* Mapping[B];

int h(domain d) {
    ...   // neki potprogram za hash-funkciju
}

void MaAssign(Mapping* M, domain d, range r) {
    int pretinac;
    celltype* tren;
    pretinac = h(d); tren = (*M)[pretinac];
    while (tren != NULL)
        if (tren->domainelement == d) {
            tren->rangeelement = r;
            return;
        }
        else tren = tren->next;
    // d nije nadjen, dodajemo ga na pocetak liste
    tren = (celltype*)malloc(sizeof(celltype));
    tren->domainelement = d; tren->rangeelement = r;
    tren->next = (*M)[pretinac];
    (*M)[pretinac] = tren;
}
```

Zadatak 2.

Napišite definicije tipova i funkciju `MaCompute` kod implementacije preslikavanja pomoću sortiranog polja.

Rješenje. Parovi $(d, M(d))$ spremljeni su u uzastopnim ćelijama dovoljno velikog polja. `M.last` je kursor na zadnju zauzetu ćeliju. Redoslijed spremanja: sortirano s obzirom na komponentu d.

```
#define MAXLENGTH ...

typedef struct {
    domain domainelement;
    range rangeelement;
} celltype;

typedef struct {
    celltype pairs[MAXLENGTH];
    int last;
} Mapping;

int MaCompute(Mapping* M, domain d, range* r) {  // saljemo pointer na preslikavanje umjesto
    int l = 0, m, u = M.last; domain dm;         // cijelog preslikavanja zato sto zelimo
    while (l <= u) {                             // ustedjeti prostor
        m = (l+u)/2; dm = M->pairs[m].domainelement;
        if (dm == d) {
            *r = M->pairs[m].rangeelement;
            return 1;
        }
        else if (dm < d)
            l = m+1;
        else
            u = m-1;
    }
    return 0;
}
```

Zadatak 3.

Preslikavanje M dano je tablicom. Nacrtajte dijagram na kojem se vidi prikaz tog preslikavanja pomoću: (a) otvorene hash-tablice s 4 pretinca i hash-funkcijom $h(d) = d \,\%\, 4$, (b) sortiranog polja duljine 8, (c) karakterističnog vektora duljine 26.

d	M(d)
15	ab
12	cd
3	ef
8	gh
25	ijk
19	lm

Napomena. U nekim primjenama javlja se potreba za invertiranjem bijektivnog preslikavanja – osim traženja pripadnog $r = M(d)$ za zadani d, potrebno je i za zadani r moći pronaći d takav da je $M(d) = r$. Strukture podataka koje smo dosad obradili slabo podržavaju invertiranje: da bi se pronašla praslika elementa r potrebno je pregledati cijelu strukturu. Ako želimo da su M i M^{-1} jednako brzih operacija, trebamo napraviti dvostruku strukturu: uz standardnu strukturu za M, trebamo napraviti još jednu kojoj je domena *range*, a kodomena *domain*. Pritom trebamo paziti da izbjegnemo dvostruki zapis podataka u memoriji.

Zadatak 4.

Promatramo rang-listu najboljih tenisača. Svaki igrač ima jedinstveni rang (poziciju) na listi. Novi igrač se dodaje na dno liste, tj. dodjeljuje mu se najveći rang. Svaki igrač na listi smije izazvati igrača koji je neposredno iznad njega; ako ga pobijedi, oni zamijene mjesta. Imamo ove operacije:

`Add(name)` ... dodaje novog igrača na dno liste

`Challenge(name)` ... vraća ime igrača s rangom $i - 1$ ako imenovani igrač ima rang i

`Exchange(i)` ... zamjenjuje imena igrača rangova i i $i - 1$, $i \geq 1$

Predložite strukturu podataka koja će omogućiti da se sve operacije efikasno obavljaju.

Rješenje.

.