RASUTE TABLICE

Primer hash tabele – Logovanje

- Napraviti fukcionalnost logovanja na sistem
- Broj korisnika sistema može biti velik
- Logovanje treba da bude efikasno

- Pretraživanje pri logovanju se obavlja na osnovu korisničkog imena

Primer hash tabele – Logovanje

```
class User {
   char* name;
   char* password;
public:
   User() { name = NULL; password = NULL; };
   User(char t[], char a[]) {
         name = new char[strlen(t)];
         strcpy(name, t);
         password = new char[strlen(a)];
         strcpy(password, a);
   };
   User(const User& user) {
         name = new char[strlen(user.name)];
         strcpy(name, user.name);
         password = new char[strlen(user.password)];
         strcpy(password, user.password);
   };
```

Primer hash tabele – Logovanje

```
  const User& operator = (const User& user) {
      name = new char[strlen(user.name)];
      strcpy(name, user.name);
      password = new char[strlen(user.password)];
      strcpy(password, user.password);
      return *this;
  };
  bool operator == (const User& user) {
      return strcmp(name, user.name) == 0 &&
           strcmp(password, user.password) == 0;
  };
  void print() { cout << name << " | " <<
      password << endl;};
};
```

Primer hash tabele – Logovanje

```
void main()
{
   ChainedHashTable<char*, User> baza(10);

   int id;
   char ime[10], lozinka[10];
   User *pData;
   while (strcmp(ime, "izlaz") != 0) {
        cin >> id;
        cin >> ime;
        cin >> lozinka;
        if (strcmp(ime, "izlaz") != 0) {
               pData = new User(ime, lozinka);
               HashObject<char*, User> obj(ime, pData);
               baza.insert(obj);
        }
   }
```

Primer hash tabele – Logovanje

```
    ime[0] = 'A';
    ime[1] = '\0';
    while (strcmp(ime, "izlaz") != 0) {
        cin >> ime;
        cin >> lozinka;
        pData = new User(ime, lozinka);
        HashObject<char*, User> obj1(ime, pData);
        HashObject<char*, User> obj2 = baza.find(obj1.getKey());
        if (obj1 == obj2) {
              cout << "Korisnik ";
              obj1.print();
              cout << " je uspesno prijavljen!" << endl;
        } else {
              cout << "Logovanje neuspesno" << endl;
        }
    }

    cout << endl;
}
```

Primer hash tabele – Biblioteke

- Biblioteke sadrže veliki broj knjiga
- Za svaku knjigu se pamti identifikacioni broj, naslov i autor
- Obezbediti brzo pretraživanje na osnovu naslova dela
- Napraviti funkciju koja ce proveriti koji se naslovi nalaze u dve različite biblioteke koje koriste isti sistem za upravljanje bibliotekom

Primer hash tabele – Biblioteke

```
class Book {
   int   id;
   char* title;
   char* author;
public:
   Book() { id = 0; title = NULL; author = NULL; };
   Book(int i, char t[], char a[]) {
         id = i;
         title = new char[strlen(t)];
         strcpy(title, t);
         author = new char[strlen(a)];
         strcpy(author, a);
   };
   Book(const Book& book) {
         id = book.id;
         title = new char[strlen(book.title)];
         strcpy(title, book.title);
         author = new char[strlen(book.author)];
         strcpy(author, book.author);
   };
```

Primer hash tabele – Biblioteke

```
  const Book& operator = (const Book& book) {
      id = book.id;
      title = new char[strlen(book.title)];
      strcpy(title, book.title);
      author = new char[strlen(book.author)];
      strcpy(author, book.author);
      return *this;
  };
  bool operator == (const Book& book) {
      return id == book.id
            && strcmp(title, book.title) == 0
            && strcmp(author, book.author) ==0 ; };
  void print() { cout << id << " | " << title
                            << " | " << author << endl;};
};
```

Primer hash tabele – Biblioteke

```
template <class T, class R>
void ChainedHashTable<T, R>::
  FindCommon(ChainedHashTable<T, R>& hashTab,
      R** res, int &len)
{
  unsigned int i;

  int sz, cnt = 0;
  if (getLength() > hashTab.getLength()) {
      sz = getLength();
  } else {
      sz = hashTab.getLength();
  }
  R *arr = new R[sz];
```

Primer hash tabele – Biblioteke

```
    for (i=0; i<length; i++) {
        HashObject<T,R> obj = array[i].getHeadEl();
        while(!(obj == T())) {
            HashObject<T,R> tmp = hashTab.find(obj.getKey());
            if (!(tmp == T())) {
                arr[cnt++] = *obj.getRecord();
            }
            obj = array[i].getNextEl(obj);
        }
    }

    *res = new R[cnt];
    for (i=0; i<cnt; i++) {
        (*res)[i] = arr[i];
    }
    len = cnt;
    delete[] arr;
}
```