ŠOLSKI CENTER CELJE
SPLOŠNA IN STROKOVNA GIMNAZIJA LAVA

PROGRAM ZA VODENJE EVIDENCE IZOSTANKOV OD POUKA

Raziskovalna naloga

MENTORJA:
Karmen Kotnik, univ. dipl. inž.
Mojmir Klovar, univ. dipl. inž.

AVTOR:
Rok Ušen, 4. f

CELJE, 2005

KAZALO

KAZALO SLIK

POVZETEK

V raziskovalni nalogi sem se lotil izdelave programa za pomoč razrednikom pri vodenju evidence izostankov dijakov od pouka. Ker program deluje na osnovi podatkovne baze, sem jo moral najprej normalizirati, nato pa sem program razvijal s pomočjo programskega paketa Borland C++ Builder 6.
Med izdelavo nisem naletel na večje težave, zato sem z delom končal do dogovorjenega datuma. Z veseljem ugotavljam, da sem svoje hipoteze v celoti potrdil in da sem se pri izdelavi programa naučil veliko uporabnih stvari.

UVOD

Opis raziskovalnega problema

Cilj moje raziskovalne naloge je bil izdelati program, s katerim bi olajšal delo razrednikom pri vodenju razrednih poslov. Glavno vodilo pri izvedbi naloge je bilo, da bo program prijazen uporabniku in enostaven za uporabo. Osnovna logika programa je dokaj enostavna. Delovanje temelji na seštevanju izostankov. Podatke, ki jih program potrebuje za smiselno delovanje, vnese uporabnik. Le-ti pa se shranijo v podatkovno bazo.

Hipoteza

Na začetku sem si zastavil cilj, da bom do dogovorjenega datuma uspel dokončati projekt. Po posvetovanju z mentorjem, sem si zadal, da moj program ne bo samo delujoč ampak tudi uporaben. Ugotovil sem, da bo na dosego cilja zelo vplivalo znanje programiranja.

Metode

Program temelji na podatkih,ki jih vnaša uporabnik. Ti podatki se uporabljajo ob vsakem naslednjem zagonu programa, zato jih je treba shraniti. Shranjevanje podatkov poteka tako, da uporabnik v ustrezne prostorčke vnese zahtevane podatke, program pa jih shrani v ustrezno tabelo. Če uporabnik potrebuje informacije (primer: razrednik ob koncu konference), jih program najde iz tabel s pomočjo SQL stavkov. Z nekaj pogoji postreže aplikacija z želenimi podatki.

Pripomočki

Aplikacijo sem izdelal z borlandovim programskim paketom Borland C++ Builder 6. Ta program je zelo dober pripomoček za izdelavo aplikacij, saj združuje več majhnih programčkov, ki zelo olajšaj delo. Seveda pa sem moral izdelati tudi podatkovno bazo. Najprej sem jo nameraval izdelati z Microsoft Access-om, ampak sem v literaturi prebral, da te podatkovne baze niso najbolj primerne za Borland C++ Builder. Zaradi tega sem se odločil, da bom uporabil Paradoxovo podatkovno bazo. Strukturiral sem jo prav tako v borlandovem programu Database Desktop.

TEORETIČNI DEL

Prvi korak vsakega raziskovalnega dela je pogovor z mentorjem. Preden sva se lotila računalniškega dela, mi je najprej razložil filozofijo vodenja izostankov dijakov od pouka. Izvedel sem določene stvari, ki mi prej niso bile znane.

Pravna osnova: Po 19. členu pravilnika o šolskem redu v srednjih šolah poznamo naslednje izostanke;

OPRAVIČLJIVI IZOSTANEK
-opravičeni izostanek (bolezen več kot 2 dneva, odobren dopust ali izostanki zaradi šolskih obveznosti)

NEOPRAVIČLJIVI IZOSTANEK *(če dijak pri posameznem predmetu v konferenci manjka več kot 15%, mora ob koncu leta opravljati predmetni izpit)*
-opravičeni izostanek (odsotnost manj kot 2 dneva oz. izostanek od posamezne ure)
-neopravičeni izostanek (vsi drugi izostanki, ki niso bili omenjeni v prejšnjih dveh postavkah)

Ko mi je bilo okoli izostankov vse jasno, sva se lahko posvetila reševanju problema. Po podrobni preučitvi naloge sem ugotovil, da v program ne bo treba vključevati zapletenih algoritmov in podobnih zahtevnih metod. Za delovanje bodo zadostovale že enostavne metode hranjenja zapisov in preštevanje le-teh.

Program mora omogočiti izpis izostankov za vsako konferenco posebej. To je zelo pomembno, saj mora dijak opravljati predmetni izpit, če že samo v eni izmed konferenc neopravičeno izostane za več kot 15 %. Zaradi tega sem moral najti rešitev, kako bi kasneje v poizvedbah lahko ločili izostanke med seboj po konferenčnih obdobjih. To sem izvedel tako, da uporabnik ob vsakem vnosu izostanka shrani tudi konferenco, v kateri se je izostanek zgodil.

V nadaljevanju sem se moral odločiti, na kakšen način bom shranjeval podatke. Odločal sem se med dvema možnostma, in sicer:

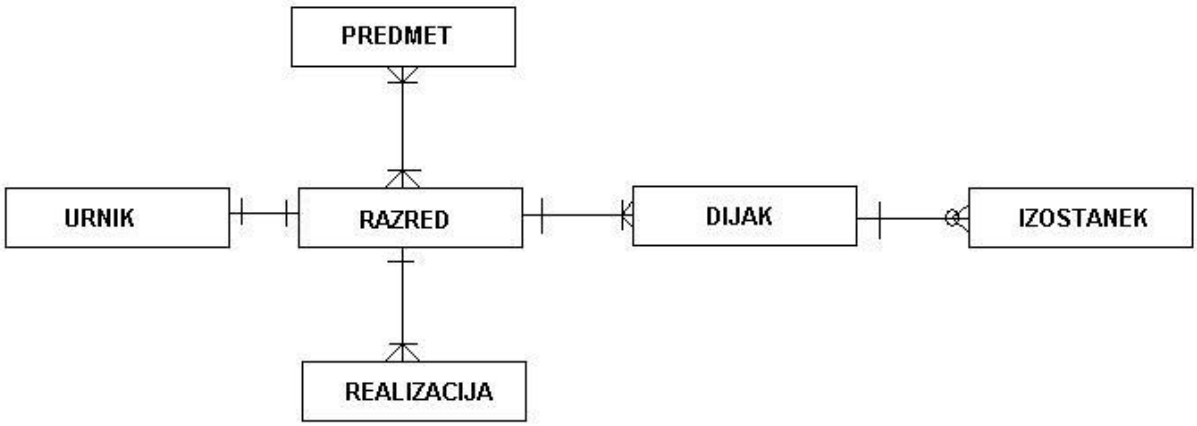
-shranjevanje podatkov direktno v datoteko (primer: Izostanki.txt);
-shranjevanje podatkov v eno izmed podatkovnih baz .

Po premisleku sem ugotovil, da bo mi bo lažje, če bom podatke shranjeval v podatkovno bazo. Pri prvem načinu bi moral vsak podatek, ki bi ga shranil kasneje, poiskati z zelo dolgim in zamudnim postopkom. Po drugi strani pa so podatki shranjeni v podatkovni bazi bolj varni in manj dostopni radovednim uporabnikom.

Za izdelavo podatkovne baze sem se odločil uporabiti borlandov program Borland Database Desktop. Pri tem programu mi je bilo všeč to, da sem lahko določil lastnosti atributov, ki so mi bile pri nadaljnjem delu v pomoč (primer: največja dovoljena dolžina zapisa, tip zapisa, obvezen zapis - DA/NE ...).

Program uporabniku ponudi več različnih vrst podatkovnih baz. Med ponujenimi možnostmi sem izbral paradoksovo bazo, zato ker mi pri programiranju ni povzročala nobenih težav.

Velik del časa, ki sem ga porabil za premislek o izdelavi raziskovalne naloge, je vsekakor »vzela« normalizacija. Izhajal sem iz naslednjega E-R diagrama:



Slika 1: Entitetno–relacijski diagram

Kot pomoč za vnašanje podatkov, sem naredil tri tabele, v katere uporabnik ne shranjuje ničesar. Iz njih samo izbere podatek, da mu ga ni treba vnašati ročno (primer: dan v tednu oz. ura v dnevu). Ker s programerskega stališča te tabele niso zanimive, jih v nadaljevanju ne bom več omenjal.

Naslednji korak je bil, da sem relacijam določil atribute. Pregledal sem tudi številnost povezav. Kjer je bilo treba, sem relacijam dodal tuje ključe (pri povezavah »ena proti ena« oz. »ena proti več«), dobil pa sem tudi novo relacijo, saj je številnost med relacijama RAZRED in PREDMET »več proti več«. Poimenoval sem jo PREDMET_RAZRED.

RAZRED (Razred#, Razrednik, Geslo)
PREDMET_RAZRED (Id_predmet + Id_razred#)
PREDMET (Id_predmet#, Kratica)
REALIZACIJA (Id#, Id_predmet, Konferenca, Ur, Razred)
URNIK (Id#, Dan, Ura, Id_predmet, Razred)
DIJAK (Id_dijak#, Priimek, Ime, Matična, Razred)
IZOSTANEK (Id#, Id_dijak, Id_predmet, Ura, Tip, Konferenca, Datum)

Preden sem se čisto zares lotil pisanja programa, sem se moral spopasti še z enim problemom. Po naključju ali zaradi nespretnosti bi se lahko v nekaj primerih zgodilo, da bi uporabnik vnesel večkrat isti podatek. To možnost sem moral izključiti. Domislil sem se dokaj enostavnega postopka, kako bi preverjal, ali v tabeli že obstaja takšen zapis, kot ga hoče shraniti uporabnik.

Bistvo pri vsem tem je, da mora program, preden shrani podatke, preveriti vse obstoječe zapise. Če identičen zapis že obstaja, mora javiti napako, v nasprotnem primeru pa se zapis shraniti. Za primer sem vzel tabelo RAZRED (Razred#,Razrednik,Geslo):

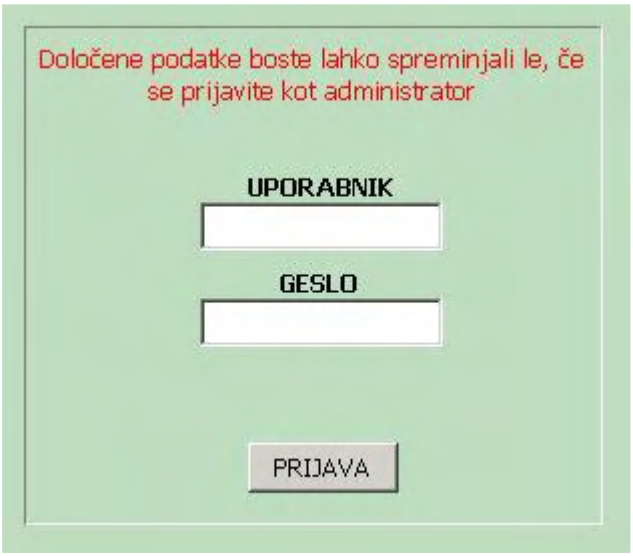
```
int i;                          //števec zapisov v tabeli
int X=0;                        //kontrolna spremenljivka

 Razred->First();               //tabelo RAZRED postavimo na prvi zapis
 i=Razred->RecordCount;         //število zapisov v tabeli se shrani v števec

//-------------------------------------------------------------------------------------------------------------------------------

 while(i>0)                     //zanka se izvaja, dokler ne pridemo do zadnjega zapisa
   {if(razred->Text= =Razred->FieldByName("Razred")
        {X=1;                   //če razred s tem imenom že obstaja, se kontrolna št. spremeni
        break;                  //in zanka se prekine
        }
   Razred->Next();              //nadaljujemo pregled z naslednjim zapisom
   i--;                         //števec zapisov zmanjšamo
  }

//-------------------------------------------------------------------------------------------------------------------------------


if(X==1)                        //če ima kontrolna spremenljivka vrednost 1,
 {opozorilo->Caption="Ta podatek že obstaja!";
 }                              //se izpiše opozorilo in podatek se ne shrani

if(X==0)                        //če se kontrolna spremenljivka ni spremenila, se vprašamo,
 {if((razrednik->Text=="")||(razred->Text=="")||(geslo->Text==""))       //če ni nobeno polje prazno
 {opozorilo->Caption="Podatkov ne morem shraniti, ker je vsaj eno izmed polj prazno!";
 }                              //če je, se izpiše opozorilo

 else                           //drugače pa se podatek shrani
   {Razred->Insert();
   Razred->FieldByName("Razred")->Value=razred->Text;
   Razred->FieldByName("Razrednik")->Value=razrednik->Text;
   Razred->FieldByName("Geslo")->Value=geslo->Text;
   }
```

Če tega preverjanja ne bi izvedel, bi pri izpisu prišlo do napak. Še posebno bi bilo moteče pri izračunanih procentih izostankov, saj bi bili podatki napačni.

Zaradi narave podatkov, ki jih obdelujemo v programu, sem program razdelil na dva sklopa. En del je namenjen administratorju programa, ki mora ostalim uporabnikom pripraviti določene stvari, da lahko potem normalno upravljajo s programom. Administrator programa lahko vnese oz. zbriše podatke o:

-razredu
-dijakih v razredu
-predmetih posameznega razreda
-urniku izbranega razreda
-uporabnikih, ki bodo uporabljali podatke določenega razreda

Seveda lahko administrator programa vstopi na prej omenjeno področje samo z uporabniškim imenom in geslom, s čimer sem preprečil, da bi lahko vsak v programu počel vse. Ta del bom podrobneje opisal v praktičnem delu, zato tukaj prilagam za boljšo predstavo samo sliko iz programa.



Slika 2: Prijava za administratorja programa

Slika 3: Prijava za uporabnika

Drugi del programa pa je namenjen uporabnikom. Na voljo sem imel več načinov, kako bi ponudil uporabnikom že prej vpisane podatke, s pomočjo katerih bi vnašali podatke. Najprej sem nameraval izdelati menije, v katerih bi uporabnik izbral želeno stvar. Kasneje pa sem ugotovil, da bi to pomenilo dodatno izgubo časa, saj bi bilo potrebno meni odpreti, v njem poiskati ustrezen zapis in ga s klikom izbrati. Uporabniku sem prihranil nepotrebno klikanje tako, da sem mu ponudil podatke kar iz tabele. Tako vidi vse zapise naenkrat, hitreje jih najde in nadaljuje s svojim delom. Prav zaradi te metode sem moral narediti tabele za pomoč pri vnašanju podatkov, ki sem jih omenil že na strani 8. Da ne bi prišlo do kakšnih nesporazumov med razredniki, sem ostalim preprečil vpogled v podatke, ki se ne tičejo njegovega razreda. Tudi razredniki vstopijo na svojo področje z geslom, ki jim ga je prej določil administrator.

PRAKTIČNI DEL

V prejšnjem poglavju sem vam predstavil nekaj dejstev, ki jih morate poznati o ozadju programa. Na naslednjih nekaj straneh vam bom opisal, kako je potekala pretvorba ideje programa ANTIŠPRICAR s papirja in zamisli v izvedljivo aplikacijo.

Vsak program je potrebno poimenovati. Moram priznati, da sem bil v dilemi, kako bi ga poimenoval. V času nastajanja programa pa se mi je utrnila dobra ideja. Dijaki v svojem žargonu uporabljamo besedo »špricanje« pouka. Po sprejetju novega pravilnika o izostajanju od pouka in seveda zaradi mojega programa predvidevam, da marsikateri dijak ne bo več neupravičeno izostajal od pouka, saj je kazen za takšne prekrške sorazmerno huda. Ker bo program največji trn v peti prej omenjenim dijakom, sem ga poimenoval Antišpricar.

Že v uvodu sem navedel, da sem za izdelavo svoje aplikacije uporabil borlanodov program Borland C++ Builder 6. Preden sem se resno lotil dela, sem se moral še marsikaj naučiti o tem programu. Ko sem ocenil, da znam dovolj, sem se lotil dela.

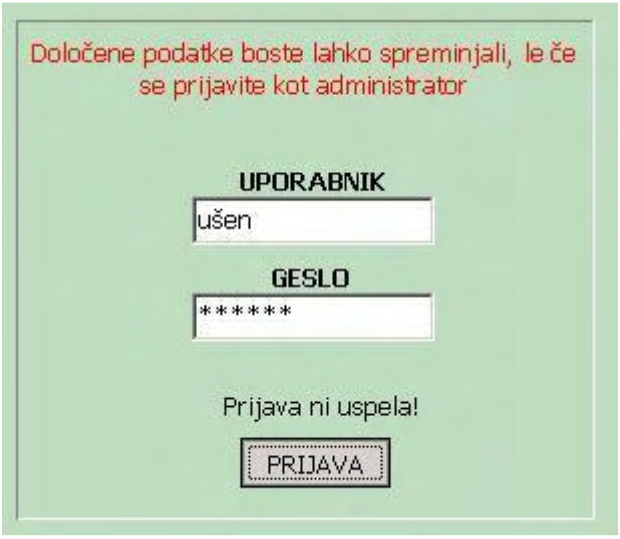
V začetku sem bil v dvomih, če bom lahko problem realiziral. Ta projekt je za enega razvijalca kar velik zalogaj. Kljub vsemu sem zbral dovolj poguma in se pridno lotil dela. Da bi lahko čim bolje predstavil program, ga bom opisal po posameznih delih. Da pa bo malo bolj pregledno, sem opisal vsebino menija, ki se pojavi ob zagonu programa.

PROGRAM	RAZRED	URNIK	IZOSTANKI	IZPIS
-o avtorju -pomoč -izhod	-vnos razreda -brisanje razreda	-vnos urnika -brisanje urnika	-prijava uporabnika in delo z izostanki	-vnos realiziranih ur -statistika za razrednika -statistika za predmetni izpit

Slika 4: Osnovno okno programa

Ob zagonu programa se uporabniku odpre osnovno okno. Če ste administrator programa, morate za nadaljnje delo vpisati uporabniško ime in geslo. Pri vseh programih je narejeno tako, da črke pri vpisovanju gesla nadomeščajo znaki. Tudi pri mojem programu je tako (PasswordChar). Ob kliku na gumbek -PRIJAVA- program preveri, če je geslo ustrezno. Če je, nam javi, da je prijava uspela, v nasprotnem primeru pa izpiše ustrezno obvestilo.

Slika 5: Prijava je uspela

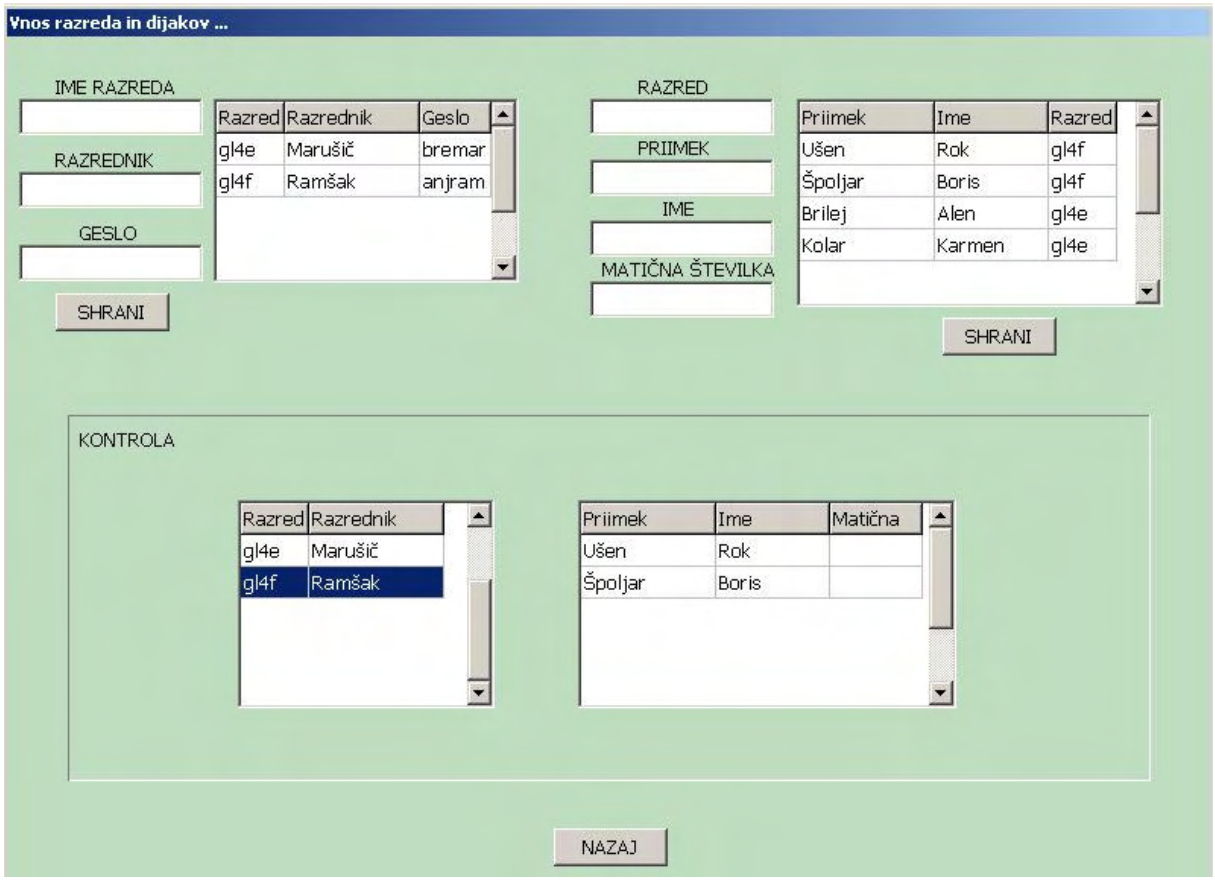


Slika 6: Prijava ni uspela

Če se v program prijavimo kot administrator, lahko razpolagamo z vsemi podatki. V nadaljevanju bom opisal tista opravila, ki jih lahko počne samo administrator.

VNOS RAZREDA

Ob kliku na RAZRED -> VNOS RAZREDA se odpre okno za vnašanje podatkov o razredu. Zapiranje in odpiranje oken sem izvedel s pomočjo ukaza –VISIBLE (true/false). Omeniti moram tudi gumb -NAZAJ-, s pomočjo katerega se lahko administrator vrne na osnovno okno. Tudi za tem gumbom se skriva kombinacija ukazov Visible->true in Visible->false, prisoten pa je na vseh oknih.



Slika 7: Vnos podatkov o razredu

Ta del programa skriva nekaj zanimivih postopkov. Pri vnosu podatkov o razredu in o dijakih ob kliku na gumbek -SHRANI- program preveri dve stvari:

-ali ni mogoče katero izmed zahtevnih polj prazno

-ali v tabeli že mogoče obstaja identičen zapis

V obeh primerih izpiše ustrezno opozorilo, podatkov ne shrani, hkrati pa zbriše »edit« okenca. Po vsakemu uspešnemu hranjenju podatkov se tabele in poizvedbe osvežijo.

Vsekakor pa moram izpostaviti, kako sem izvedel shranjevanje dijaka v izbrani razred. Ko administrator shrani podatke o razredu, mora v tabeli izbrati razred. Ime izbranega razreda se prenese na desno stran v okence –RAZRED-. To sem naredil tako, da se ob kliku na zapis v tabeli zgodi dogodek. Ne glede na to, kje administrator klikne na zapis (bodisi na razred, razrednika ali geslo), se prenese vrednost iz atributa -RAZRED- v desno stran okna.

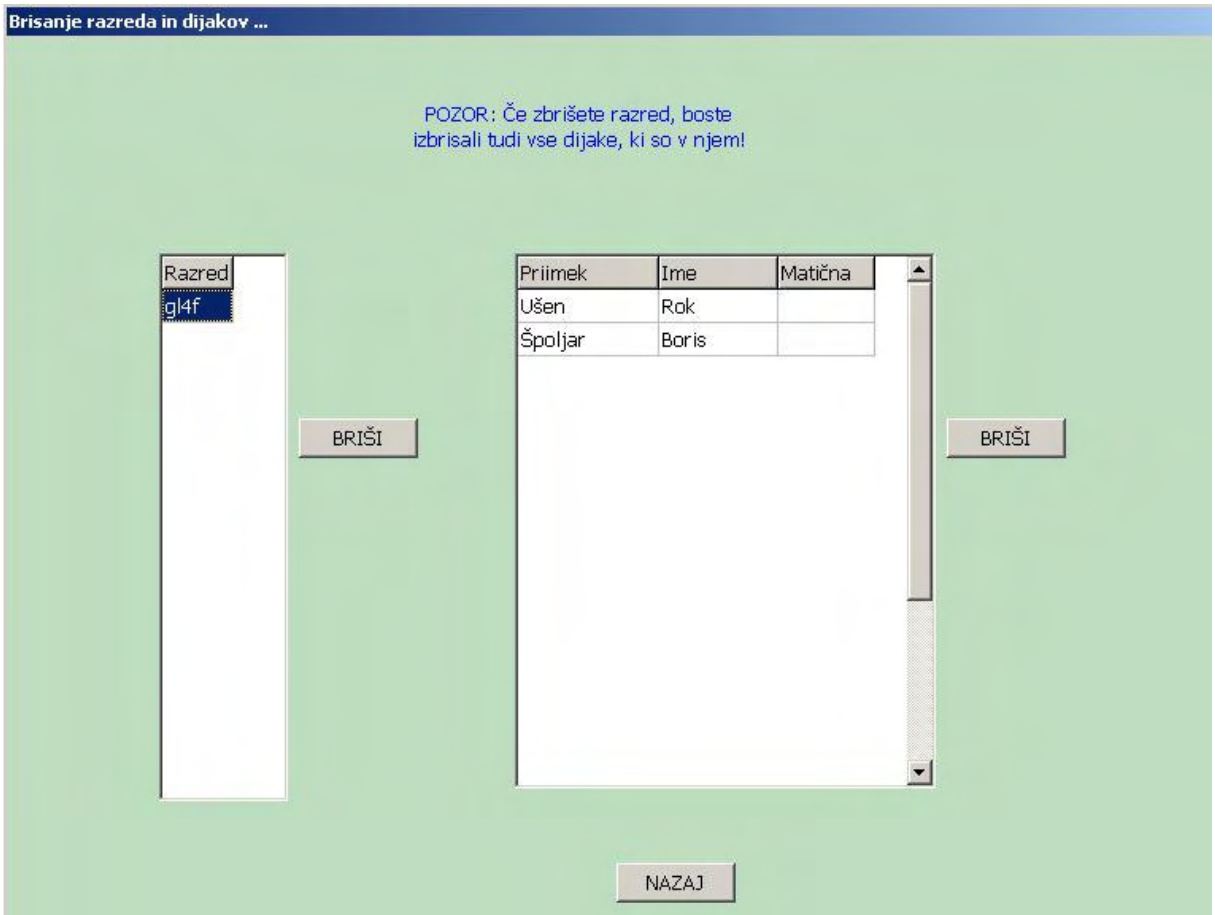
Upošteval sem dejstvo, da se lahko pri vnašanju podatkov tudi zmotimo. In že smo pri naslednji možnosti programa.

BRISANJE RAZREDA

S klikom na razred oz. dijaka se v labelo shrani številka zapisa. Klik na gumb -ZBRIŠI- povzroči, da program pregleda ustrezno tabelo, poišče številko zapisa v tabeli in ga zbriše iz nje. Koda:

```
Labela->Caption=Tabela->RecNo;          //v labelo shrani številko zapisa
Tabela->Edit();                         //tabelo odprem za urejanje
Tabela->FieldAddress(Labela->Caption);  //v tabeli poiščem zapis z zaporedno številko iz labele
Tabela->Delete;                         //ustrezen zapis izbrišem
```

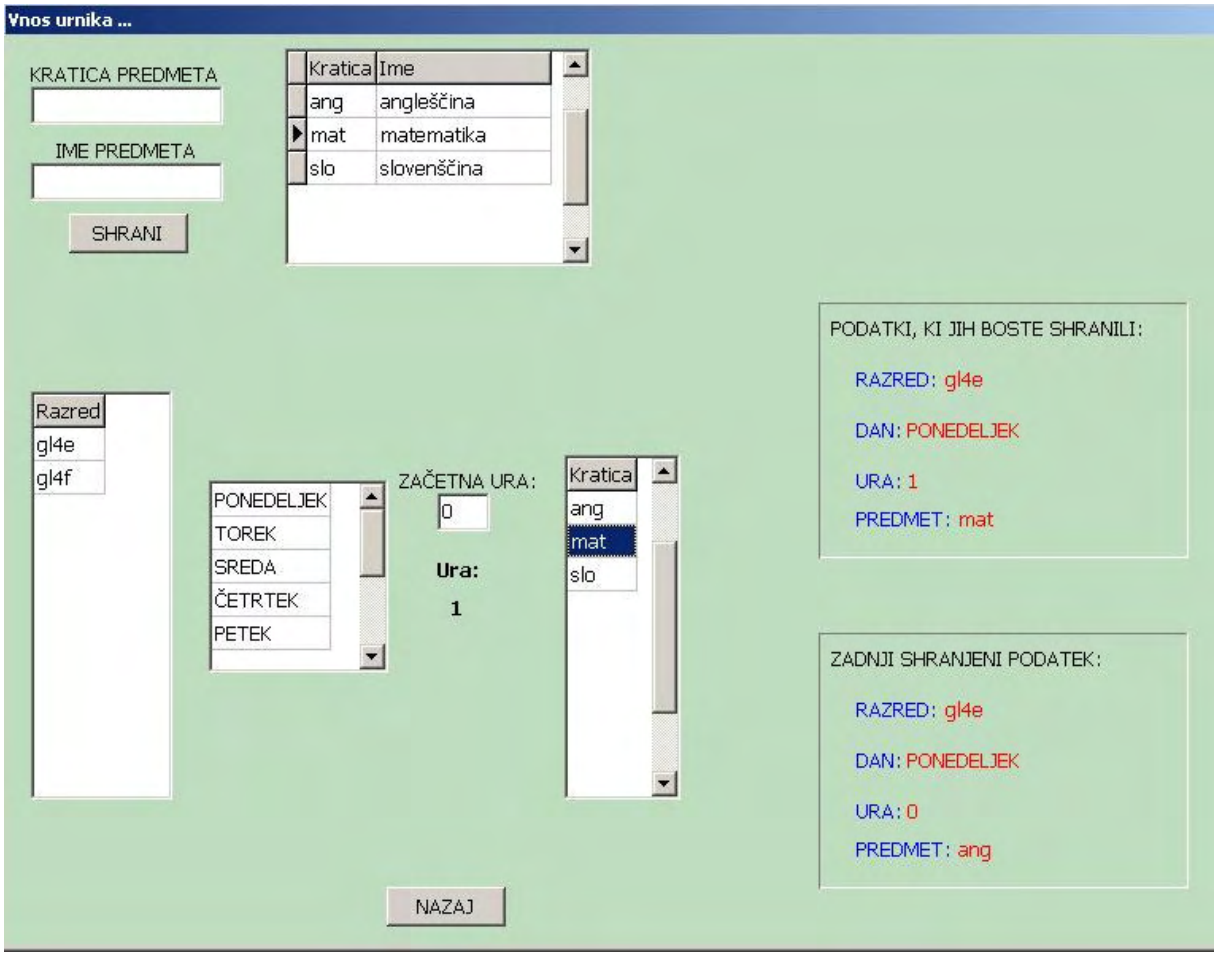
Tukaj lahko uporabnik preveri, če so vsi dijaki vpisani v ustreznem razredu oz. če so bili podatki pravilno vnešeni. Če temu ni tako, jih lahko zbriše in v oknu VNOS RAZREDA ponovno vnese. V primeru, da bi moral administrator zbrisati razred, bo program ob kliku na gumb ZBRIŠI zbrisal tudi vse dijake, ki so bili vneseni v ta razred. O tem je uporabnik tudi obveščen .



Slika 8: Brisanje podatkov o razredu

VNOS URNIKA

Za nadaljnje delo s programom je treba vnesti urnik razreda. Najprej mora administrator vpisati vse možne predmete, iz katerih bo kasneje sestavljal urnik.



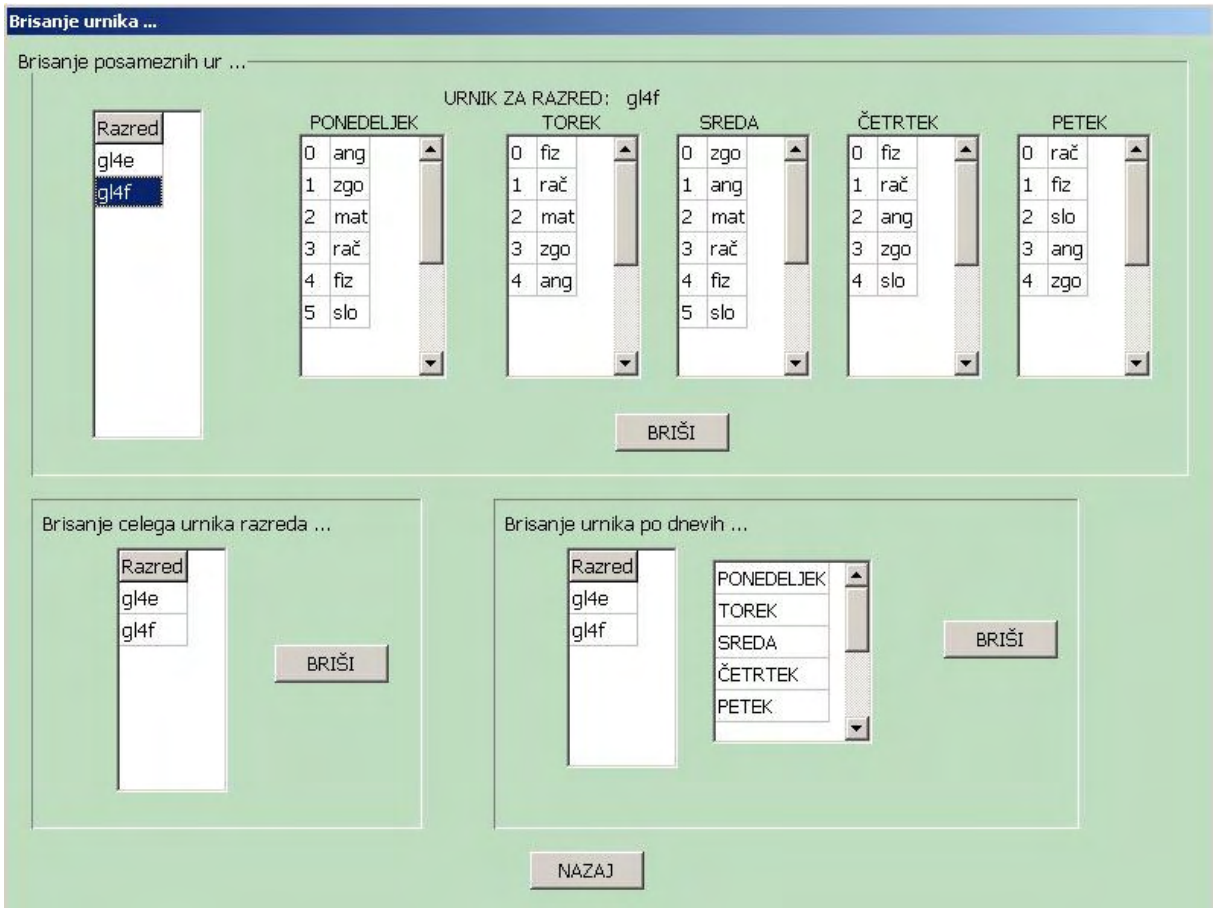
Slika 9: Vnos urnika

Tudi v tem oknu sem uporabil dogodke s klikom na tabele. S klikom na določeno tabelo se prenašajo ustrezne vrednosti. Podvojeni oz. prazni vnosi tukaj niso dovoljeni. Za pomoč pri vnašanju podatkov sem na desni strani prikazal podatke, ki jih bo program shranil, in tiste, ki jih je nazadnje shranil.
Pri vnašanju izostankov moram omeniti še nekaj. Ko izberemo razred in dan, moramo vpisati začetno uro, s katero bomo začeli vnašati urnik. V nadaljevanju se za vsak shranjeni zapis ura samodejno poveča za ena. To sem naredil tako, da se na spremembo v -ZAČETNA URA- najprej prenese vrednost v obliki integer v labelo. Ta labela pa se po vsakem uspešnem hranjenju poveča za ena. Če pa izberemo drug dan, se zopet postavi na vrednost, ki smo jo vnesli kot začetna ura.

BRISANJE URNIKA

Večkrat se zgodi, da je treba urnik popraviti oz. ga vnesti na novo. Tudi za to sem poskrbel v svojem programu. Upravljalec programa ima na voljo tri izbire.

-brisanje posamezne ure,
-brisanje urnika enega dneva,
-brisanje celotnega urnika izbranega razreda.



Slika 10: Brisanje urnika

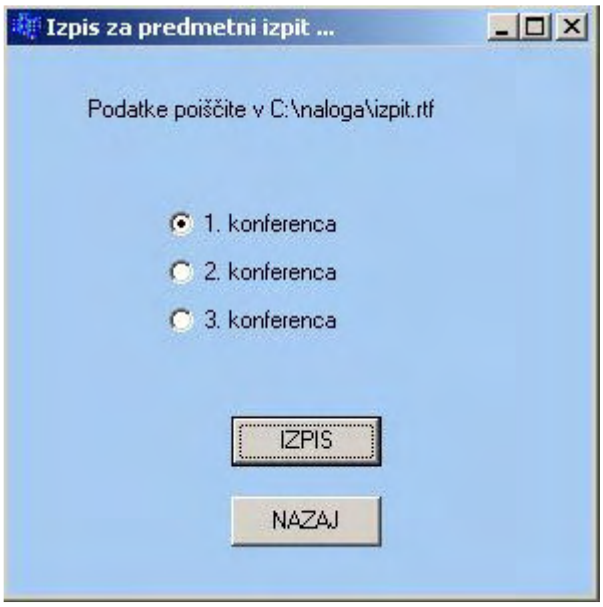
Če se odloči za brisanje posamezne ure, izbere v zgornji tabeli razred in uro v tednu. S pritiskom na gumb BRIŠI program z ukazom Tabela->RecNo poišče zaporedno številko zapisane ure v tabeli in jo zbriše.

Zelo uporabna možnost pa je brisanje celotnega urnika razreda. Ko izberemo razred, se prenese ime razreda v labelo. Nato s pritiskom na gumb ZBRIŠI program pregleda tabelo, v kateri je shranjen urnik in zbriše vse tiste ure, ki imajo v atributu RAZRED shranjen isti razred, za katerega se je uporabnik prijavil.

Kot tretja možnost brisanja pa je brisanje urnika za določen dan v tednu. Potek brisanja podatkov je enak kot pri prejšnjem načinu, le da tukaj program preveri, če se ujema dan v tednu.

In že smo pri zadnjem delu programa, namenjenega administratorju.

IZPIS ZA PREDMETNI IZPIT



Slika 11: Izpis izostankov za predmetni izpit

Podatke za to poizvedbo sem poiskal iz več tabel. Rezultat shrani program v IZPIT.RTF. V datoteko sem pisal podatke s pomočjo ukaza FPRINTF.

Ta algoritem se izvede v petih korakih.
1. korak: Prebere in izpiše prvi zapis v tabeli Razred

2. korak: Prebere in izpiše prvi zapis v tabeli Dijak

3. korak: Prebere in izpiše prvi predmet iz tabele, kjer so shranjeni predmeti izbranega razreda

4. korak: Prebere, izpiše in prešteje vse izostanke, ki so bili shranjeni pri izbranem predmetu.

5. korak: Na koncu poišče glede na razred, konferenco in predmet, število realiziranih ur. S pomočjo prej preštetih izostankov in števila realiziranih ur program izpiše procent prisotnosti dijaka pri pouku.

Ko zadnji tabeli pregleda program do konca, v prejšnji vzame naslednji zapis in se zopet vrne v tabelo IZOSTANKI, kjer išče in prešteva ustrezne izostanke.

Enačba za izračun: X=(št.neopravičenih izostankov/št.realiziranih ur)*100

```
IZPIS IZOSTANKOV PO RAZREDIH ZA PREDMETNI IZPIT

Konferenca:1
Datum:24.3.2005
------------------------------
gl4d
gl4e
  Brilej Alen
  Kolar Karmen
gl4f
  Špoljar Boris
    mat
 Pri uri prisoten: 80.00
    slo
 Pri uri prisoten: 100.00
  Ušen Rok
    mat
 Pri uri prisoten: 90.00
    slo
 Pri uri prisoten: 94.00
```

Slika 12: Primer izpisa udeležbe pri posameznih urah

Ker je pomembno, da poizvedba prikaže izostanke iz vsake konference posebej, je potrebno izbrati za katero konferenco bi rad administrator videl rezultate. Če tega ne stori, ga program z ustreznim sporočilom o tem obvesti.

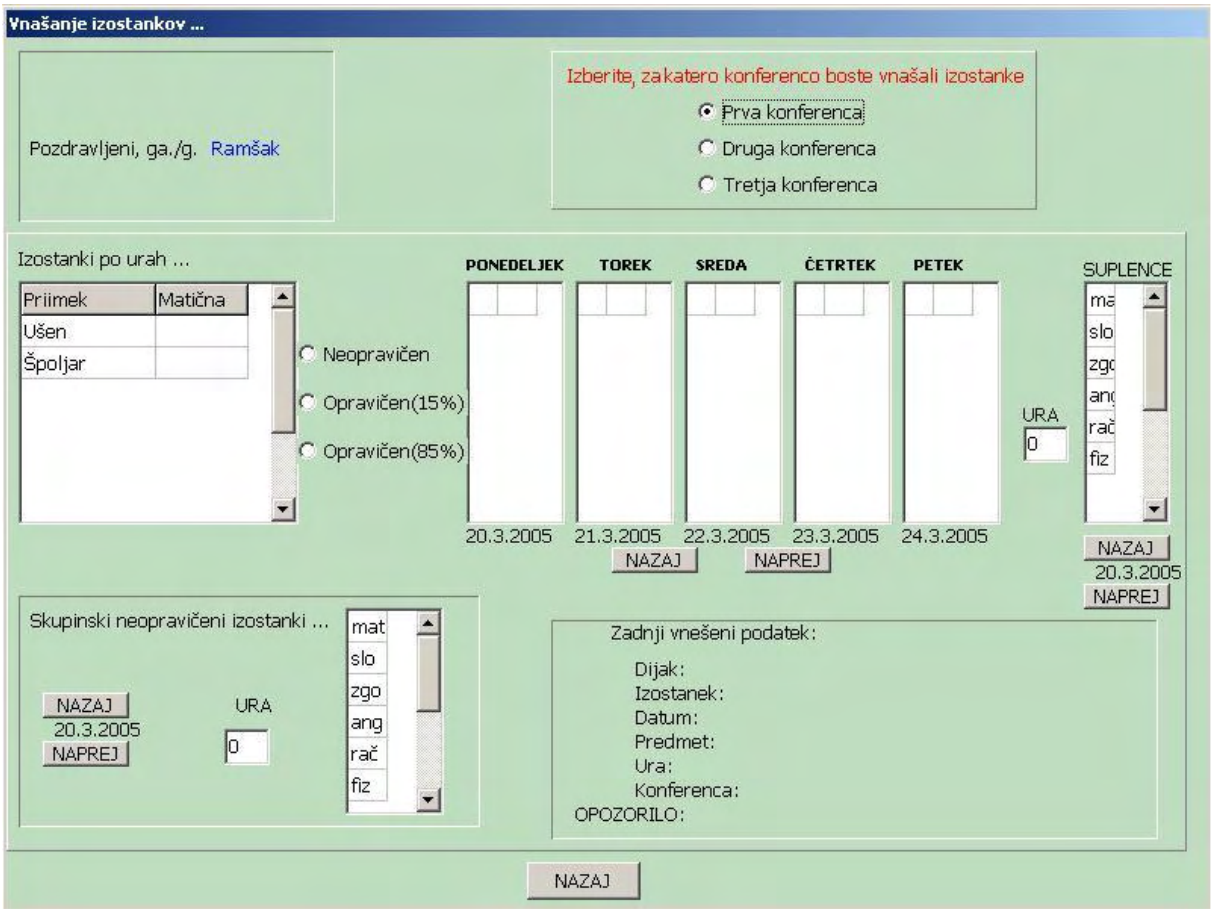
Sedaj pa še nekaj več o tistem delu programa, ki je namenjen več uporabnikom – razrednikom. Ob zagonu programa si lahko uporabnik pogleda podatke o avtorju programa.

O AVTORJU

Slika 13: Podatki o avtorju programa

PRIJAVA UPORABNIKA IN VNAŠANJE IZOSTANKOV

Naslednji del programa bi lahko poimenovali kar jedro programa. Tukaj namreč uporabnik po uspešni prijavi vnaša in spreminja podatke o izostankih. Za lažje vnašanje datuma sem vpletel tudi funkcijo DATE(), ki labelam dodeli datum. Ko se program zažene, je privzeti datum ponedeljka datum dneva, ko delamo v programu. Ker se izostanki ponavadi vnašajo z nekajdnevno zamudo, se s pomočjo klikov NAPREJ in NAZAJ določi ustrezen datum.

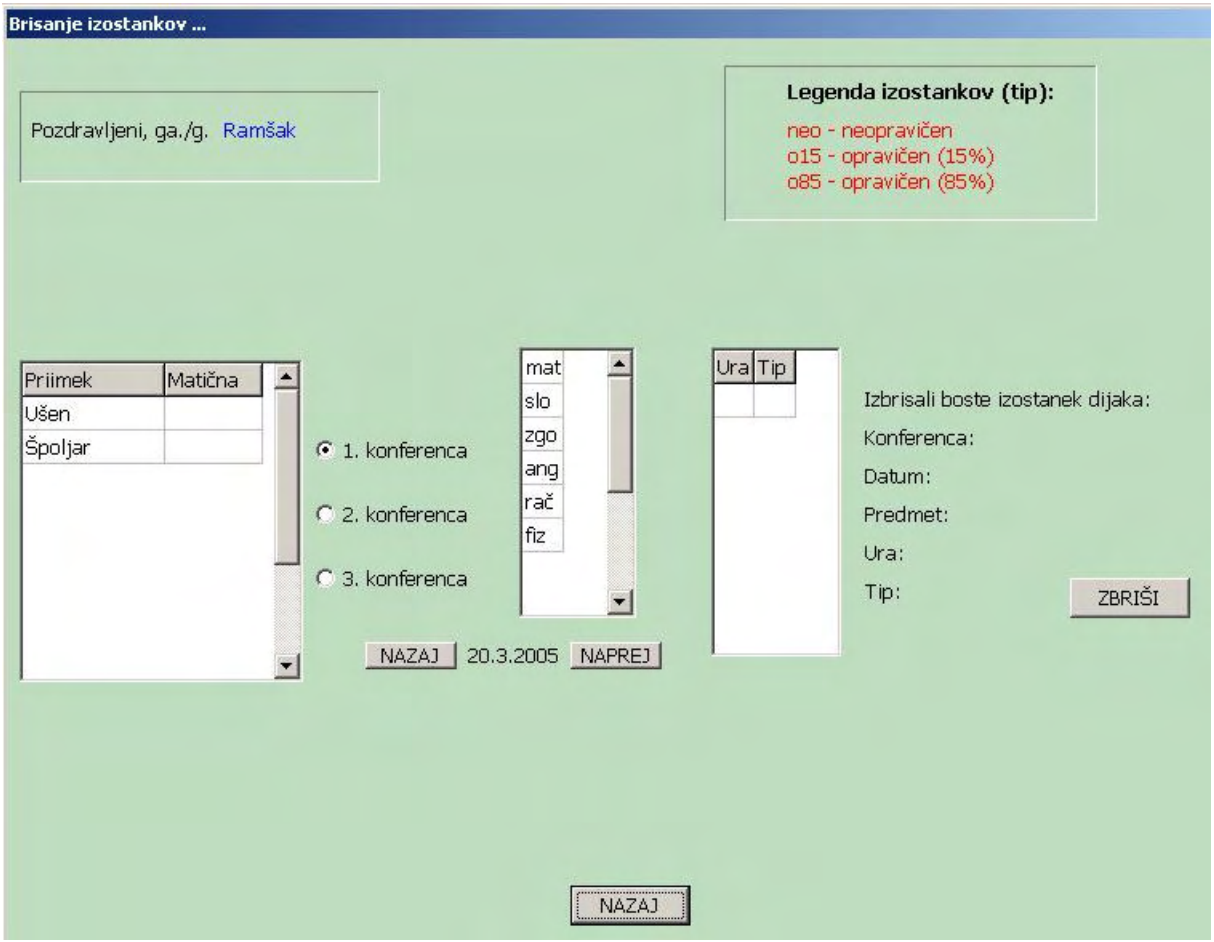


Slika 14: Vnašanje izostankov

Če na pouk na določen dan ni potekal po urniku (suplence …), se lahko izostanek izbere iz posebne tabele SUPLENCE. Podatke o predmetih razreda program pridobi iz tabele, kjer so shranjeni predmeti razreda.
Ker se dijaki včasih odločijo tudi za kolektivno izmikanje organiziranim oblikam pouka, sem moral poskrbeti tudi za to možnost. Ko uporabnik izbere datum, uro in predmet, se pojavijo vsi dijaki, ki pripadajo izbranemu razredu. Le-tem se pripiše neopravičena ura k ustreznemu predmetu. Na desni strani spodaj ekrana, ima uporabnik vedno podatek o zadnjem shranjenem podatku.

BRISANJE IZOSTANKOV

Če pride do kakšne napake pri vnosu izostanka, jo lahko uporabnik v tem delu programa odpravi.

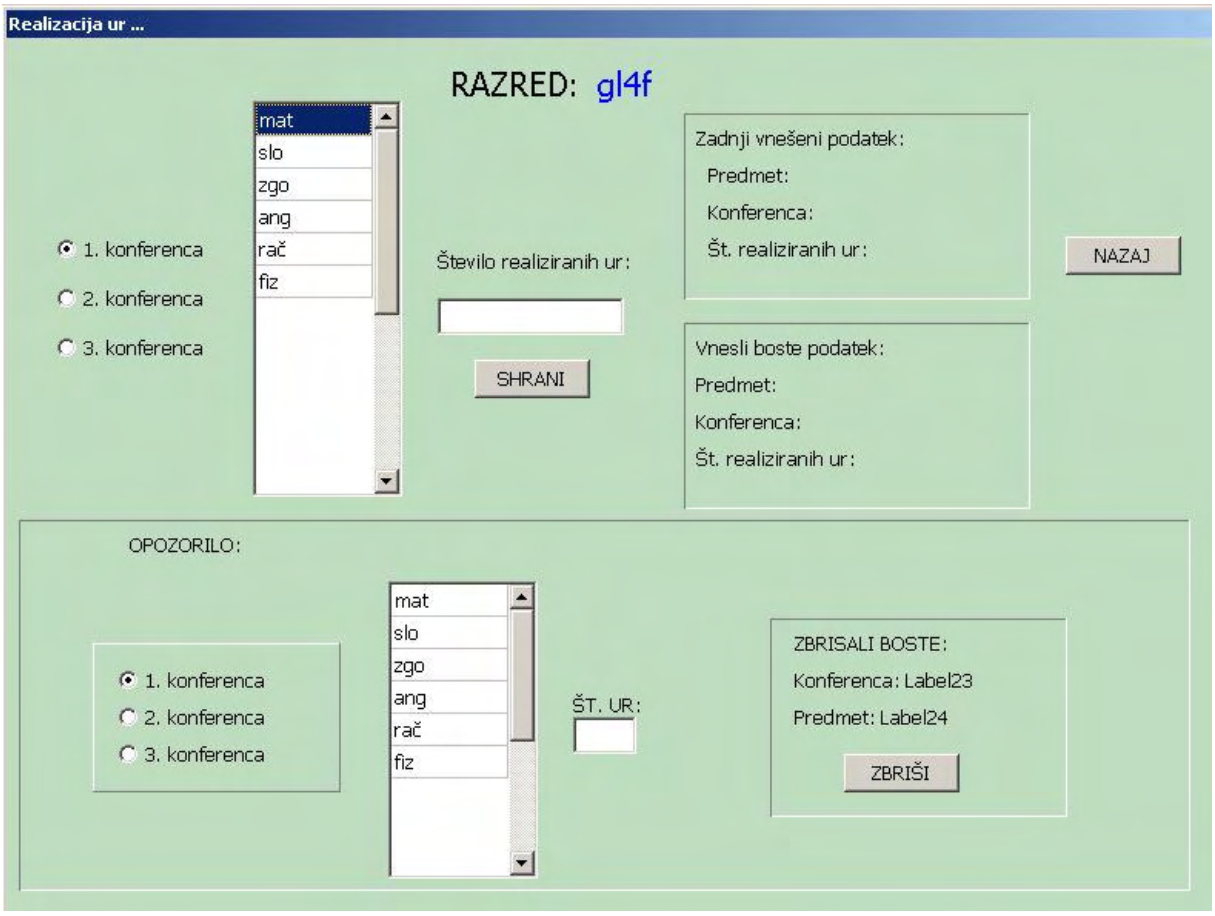


Slika 15: Brisanje izostankov

Kot ste verjetno opazili, sem moral tukaj dodati legendo izostankov. Za to sem se odločil, ker sem izbral za izostanke kratice. Te so dokaj logične, ampak lahko bi se zgodilo, da bi bil kateri izmed uporabnikov v dilemi.

VNOS IN SPREMINJANJE REALIZIRANIH UR

Vnos teh podatkov je ključnega pomena za pravilno delovanje programa. Uporabnik za vsak predmet posebej vnese število realiziranih ur v določeni konferenci. Ko se kasneje seštejejo vsi izostanki pri določenem predmetu, se iz te tabele prebere število realiziranih ur. Ta podatek se uporabi za izračun procenta udeležbe pri predmetu. Če je ta procent pod 85 %, mora dijak opravljati predmetni izpit.



Slika 16: Vnos in brisanje realiziranih ur

Uporabniku se prikažejo vsi predmeti razreda. Ko izbere konferenco in predmet ter vnese število ur, se s pritiskom na gumb SHRANI v tabelo REALIZACIJA shranijo podatki. Če je katero izmed polj prazno, se izpiše ustrezno opozorilo.

V spodnjem delu okna pa lahko uporabnik zbriše vnešene podatke.

IZPIS ZA VZGOJNI UKREP:

Za konec pa nam ostane še izpis podatkov za razrednika. Rezultat poizvedbe program izvozi v datoteko Ukrep.txt. Za ustrezni rezultat črpamo podatke iz treh tabel. Na kratko sem opisal postopek preštevanja izostankov:

1. korak: Iz prve tabele (Razred) program poišče prvi zapis.

2. korak: V drugi tabeli (Dijak) poišče vse tiste dijake, ki so v razredu, ki ustrezajo podatku, dobljenem iz prve tabele.

3. korak: Na koncu pregleda celo tretjo tabelo (Izostanek).

Določil sem dva števca – za opravičene in neopravičene ure. Če zapis v tabeli ustreza pogojema, se števca ustrezno povečata. Ko pregleda celotno tabelo izostankov, se v datoteko izpiše število opravičenih in neopravičenih izostankov za določenega dijaka. Nato v drugi tabeli poišče naslednjega dijaka iz istega razreda ter enako ponovi s tabelo izostankov. Ko v drugi tabeli pride do zadnjega zapisa, vzame iz prve tabele naslednji zapis. Torej poizvedba temelji na trojni zanki. Preden program začne z izvajanjem zank, mora ugotoviti število zapisov v prej omenjenih tabelah. To sem naredil tako, da sem določil tri števce in jim priredil dolžino tabele.

Primer:

```
int a = Razred->RecordCount;        //števcu priredim dolžino tabele
while(a>0)                          //zanka se izvaja, dokler ni a enak 0
        {…                          //izvršijo se postavljeni pogoji
        …
        Razred->Next;               //iz tabele prebere naslednji zapis
        a--;                        //števec a zmanjšam za 1
        }
```

```
IZPIS IZOSTANKOV

Konferenca:1
Razred:gl4f
Datum:23.3.2005
------------------------------
Špoljar Boris
   Opravičeni izostanki: 0
   Neopravičeni izostanki: 1

Ušen Rok
   Opravičeni izostanki: 0
   Neopravičeni izostanki: 1
```

Slika 17: Primer izpisa izostankov

Tudi v tem okencu mora uporabnik izbrati, za katero konferenco želi izpisati podatke. Če konference ne izbere, mu program javi napako.

UGOTOVITVE

Pri programiranju sem ugotovil nekaj pomanjkljivosti v programu. Razlogov, zakaj jih nisem odpravil, je kar nekaj.

-Eden izmed zelo pomembnih je ta, da mi je proti koncu že zmanjkovalo časa za testiranje programa. Napake ali pomanjkljivosti, ki sem jih odkril, bi odpravil v poznejših verzijah programa.

-Na začetku se nisem najbolje zavedal velikosti in zahtevnosti problema, v katerega sem se spuščal. Ta naloga je bila dovolj obširna, da bi se je lahko lotila skupaj s sošolcem.

-Ko na koncu gledam program, sem ugotovil, da bodo morali uporabniki biti zelo spretni pri obvladovanju miške, saj je pri vnašanju in pregledovanju podatkov veliko klikanja in premikanja miške sem ter tja.

-V poznejših različicah programa bom nekaj stvari spremenil. Zelo prijazno do uporabnika bi bilo, če bi lahko podatke o razredu uvozil iz že obstoječe datoteke. Tako bi administratorju prihranil nekaj časa.

-Najbrž se vsi zavedamo, da se vsi trudimo stvari čimbolj posplošiti in poenostaviti, a kljub vsemu prihaja do napak. V tem programu mora uporabnik v nekaj primerih izbrati ustrezno konferenco in tip izostanka. Če kdo ne bo dobro prebral navodil, se bodo zapisi shranili kar z začetno (privzeto) vrednostjo.

ZAKLJUČEK

V zaključku naloge lahko rečem, da sem svoje predpostavke v celoti potrdil. Nalogo sem izdelal do dogovorjenega datuma. Program deluje in je uporaben. Najpomembneje pri vsem pa je, da sem se bil zaradi izdelave te seminarske naloge primoran naučiti osnove objektnega programiranja in jezika SQL. To se mi zdi velik uspeh, saj mi bo to na moji nadaljnji računalniški poti prišlo velikokrat prav.

VIRI IN LITERATURA

1. Bob Swart: BORLAND C++Builder 6 developer's guide,Indianapolis : Sams, cop. 2003.

2. Petra Bilke: Spoznajmo PHP in MYSQL, Nova Gorica: Flamingo, 2002.

3. Tomaž Mohorič: Podatkovne baze. - 2. popravljena izd. - Ljubljana :Bi-tim, 2002 (Ljubljana : Pleško).

ZAHVALA

Ob koncu bi se rad zahvalil profesorjem računalništva in informatike z moje šole. S svojimi nasveti ter napotki so mi stali ob strani in veliko pripomogli k realizaciji raziskovalne naloge.