Тип слог и датотеке

Тип слог

Решавање задатака помоћу низова је ефикасно и задовољавајуће, међутим, има и недостатака. Главни проблем код оваквог начина решавања задатака је што се код сваке провере морају уносити сви елементи низа. То се може избећи коришћењем слогова и датотека. Такође, врло често се подаци са којима радимо састоје из делова који могу, али не морају бити истог типа. Готово свакодневно гледамо телефонски именик, распоред ТВ програма, свој распоред часова... Све су то подаци који се састоје из неколико делова. Онако како их ми гледамо, они су уређени, али не морају увек да буду. Обратимо пажњу на организацију података у адресару. Податак се састоји из имена и презимена неке особе, адресе и броја телефона, при чему су адреса и број телефона такође сложени подаци, јер се адреса састоји из улице, броја и места (код неких и држава), а број телефона се састоји из позивног броја и броја. Ово мноштво података можемо да организујемо помоћу неколико низова, али је тада руковање подацима гломазно, оптерећено бројним називима и индексима низова. Све су то разлози који су довели до увођења слоговног типа. Слоговни тип (слог, запис или *record*) се састоји из унапред дефинисаног броја делова који се називају поља (*field*). Уколико се програм састоји из само једне процедуре или се вредност промењљивих типа слог користи само у једној процедури или функцији експлицитна дефиниција типа се може изоставити, довољно је имплицитно дефинисати тип слог кроз декларацију промењљивих, међутим, ако желимо да вредности промењљивих типа слог преносимо из једне процедуре или функције у другу онда је експлицитна дефиниција обавезна:

```
TYPE adrese=RECORD ulica,broj,mesto,drzava:string;
            END;
    brojtlf=RECORD pozivni:string;
                   broj:string;
            END;
    adresar=RECORD prezime,ime:string;
                   adresa:adrese;
                   telefon:brojtlf;
            END;
```

Слог *adresar* састоји се из четири податка, два су типа стринг (*prezime, ime*), а два су такође слогови (*adresa, telefon*) који се састоје из четири (*ulica, broj, mesto, drzava*), односно, два податка (*pozivni, broj*).

Промењљива типа слог може се декларисати претходно експлицитно дефинисаним слоговним типом или директно у декларацији промењљивих (без претходне декларације слоговног типа):

```
VAR ucenik1,ucenik2:RECORD ime,prezime,skola:string;
                           razred:(I,II,III,IV);
                           odeljenje:1..8;
                    END;
    podatak1,podatak2:adresar;
    telefon:brojtlf;
```

Промењљиве *ucenik1* и *ucenik2* имплицитно дефинишу слог са пет података. Ако су поља у слогу истог типа могу се декларисати збирно, али то није обавезно, тј. дозвољена је и равноправна и оваква декларација:

```
VAR ucenik1,ucenik2:RECORD ime:string;
                           prezime:string;
                           skola:string;
                           razred:(I,II,III,IV);
                           odeljenje:1..8;
                    END;
```

Поља у слогу могу бити било ког стандардног или дефинисаног типа. Припадност неког поља слогу означава се навођењем имена слога и имена поља раздвојених тачком:

```
podatak1.prezime    // поље prezime припада слогу podatak1
```

```
podatak1.adresa.ulica      // поље ulica припада слогу adresa који је део слога podatak1.
```

Директно упоређивање слогова није дозвољено. Упоређују се само вредности појединих поља у оквиру слогова. На пример, овако је дозвољено упоређивати слогове, односно, поједина поља у оквиру слога:

```
ucenik1.prezime<ucenik2.prezime
podatak1.adresa.ulica<=podatak2.adresa.ulica
```

Али овакве релације нису дозвољене:

```
ucenik1<ucenik2
podatak1.adresa<=podatak2.adresa
```

Замена вредности између две променљиве истог слоговног типа се остварује једноставно:

```
VAR a,b,pom:slog;
      ...
      pom:=a; a:=b; b:=pom;
```

Променљиве које мењају вредности и помоћна променљива морају бити декларисане и истог типа. Замена вредности слоговних променљивих ***a*** и ***b*** врши се као и замена вредности две променљиве било којег другог типа. Наредба додељивања ***pom:=a*** означава да се сваком пољу слоговне променљиве ***pom*** додељује вредност одговарајућег поља слоговне променљиве ***a*** (и исто тако у остале две наредбе додељивања).

Приликом рада са слоговима име слога се врло често понавља код уноса, код исписа, код упоређивања или других обрада и то може да представља непотребно губљење времена, нарочито ако је име слога предугачко. Да би се проблем решио уведена је наредба ***With*** којом се ова незграпна понављања имена слога елиминишу. Тако се група наредби:

```
      podatak1.prezime:='Илић';
      podatak1.ime:='Мирослав';
      podatak1.adresa:='Улица и број';
      podatak1.telefon:='011/123-45-67';
```

може се заменити једном једноставнијом за писање наредбом:

```
with podatak1 do
      begin prezime:='Илић';
            ime:='Мирослав';
            adresa:='Улица и број';
            telefon:='011/123-45-67';
      end;
```

Наравно, имена поља се не могу изоставити нити скратити. Код упоређивања два слога ова наредба не помаже много, јер се не може применити на оба слога истовремено.

Датотеке

Главни недостаци рада са подацима у примеру из претходног одељка је ограниченост броја података и њихово чување. Организација података у низу омогућава да се подаци чувају само док је програм у меморији, тј. док се са тим подацима ради. Нема могућности да после неког времена, поново погледамо уписане податке, ако смо у међувремену обављали неке друге послове са рачунаром или смо једноставно искључили рачунар. А јасно је, сврха прикупљања података није њихова једнократна употреба после које податке *бацамо у кош*. Неки подаци су важни и треба их сачувати за будућу употребу.

Појам датотека је уско везан са чувањем података. Датотека може бити скуп адреса, подаци из неколиких експеримената, рецепти, на пример, италијанске кухиње, програм или скуп програма корисника, скуп (библиотека) функција над неким подацима итд. Овде ћемо под појмом датотека подразумевати *скуп података одређеног типа који се чува на спољашним носиоцима меморије*.

У основи, разликујемо две врсте датотека: текстуалне датотеке и датотеке са другим типовима. Оне се разликују и структурно и функционално (по начину рада са њима, али и по подацима у њима). Основни послови који се јављају у раду са датотекама су:

- отварање (креирање или формирање) датотеке,
- упис и допуна података датотеке,
- уређивање (сортирање) података датотеке,
- читање и претраживање датотеке,
- брисање неких података или читаве датотеке,
- промена имена датотеке итд.

Да би се радило са датотекама морају се схватити сви ови послови. Поменућемо најпре неке функције и процедуре са којима се најчешће сусрећемо у раду са датотекама:

Append	отвара постојећу текстуалну датотеку за додавање нових података.
AssignFile	име датотеке на спољашњем носиоцу меморије додељује промењливој датотеке у програму.
CloseFile	затвара отворену датотеку.
Eof	логичка функција која одређује да ли се показиваи података у датотеци налази иза последњег податка или не.
Eoln	логичка функција која одређује да ли се показивач података у текстуалној датотеци налази на крају реда или не.
Erase	брише датотеку на спољашњем носиоцу меморије.
FilePos	целобројна функција која одређује редни број податка на којем је показивач података у датотеци.
FileSize	целобројна функција која одређује број података у датотеци, не може се примењивати на текстуалне датотеке.
IOResult	целобројна функција која одређује статус претходно извршене улазно - излазне операције са датотеком, има вредност 0 ако није било грешке.
Read	стандардна процедура за читање једног или више података из датотеке.
Readln	стандардна процедура за читање једног или више података из датотеке и код текстуалних датотека постављање показивача података иза краја реда.
Rename	стандардна процедура за промену имена датотеке, датотека мора бити затворена пре примене процедуре.
Reset	стандардна процедура која отвара постојећу датотеку.
ReWrite	стандардна процедура која креира и отвара нову датотеку, ако се примени на већ постојећу датотеку брише све податке из ње.
Seek	стандардна процедура која поставља показивач података у датотеци на податак са одређеним редним бројем, не може се примењивати на текстуалне датотеке.
SeekEof	стандардна процедура која поставља показивач података у датотеци на крај текстуалне датотеке.
SeekEoln	стандардна процедура која поставља показивач података у текстуалној датотеци на крај реда.
Truncate	стандардна процедура која брише све податке од тренутне позиције показивача података у датотеци.
Write	стандардна процедура за упис једног или више података у датотеку.
Writeln	стандардна процедура за упис једног или више података у датотеку и дефинише крај реда у текстуалној датотеци.

А сада ћемо покушати да објаснимо начин рада са датотекама уз посебне примере за текстуалну и датотеку са слогом.

Формирање датотеке

Пре употребе, датотека се мора декларисати или дефинисати. Датотека се експлицитно дефинише у делу за дефиниције типова:

```
TYPE  imena=FILE OF string;
      adresa=RECORD ulica:string;
                    broj:integer
             END;
      adrese=FILE OF adresa;
```

Тип текстуалне датотеке не мора бити посебно дефинисан јер постоји стандардни тип *TextFile* који се може користити свуда у програму.

Да би се радило са датотеком, мора да се декларише промењљива која ће бити повезана са стварном датотеком смештеном на спољашњем носиоцу меморије. Промењљива датотечког типа се декларише у делу за декларације промењљивих на уобичајени начин помоћу претходно експлицитно декларисаног датотечког типа или директно као имплицитна дефиниција тог типа:

```
VAR   imenik:imena;
      adresar:adresa;
      ceobroj:FILE OF integer;
      slova:FILE OF char;
      opis:TextFile;
```

Декларацијом датотеке не завршава се посао формирања датотеке. Сада је потребно такву датотеку креирати на одговарајућем носиоцу спољне меморије. Нека је потребно креирати декларисану датотеку на активној диск јединици. Најпре треба променљивој датотечког типа доделити стварно име датотеке, односно, повезати променљиву датотечког типа са датотеком која ће бити креирана на спољашњем носиоцу меморије:

```
AssignFile(imenik,'imena.dat'); // претходно декларисаној променљивој imenik придружује се
                                // датотека на диску са именом imena.dat
AssignFile(opis,'pismo.txt');   // променљивој opis придружује се текстуална датотека pismo.txt
```

Уместо конкретног текста измеру апострофа може се употребити и нека друга променљива која има одговарајућу текстуалну вредност (на пример, текст из неког едита).

Сада треба креирати датотеку помоћу стандардне процедуре *ReWrite*:

```
ReWrite(imenik);  // на активном диску формира нова датотеку са именом imena.dat
ReWrite(opis);    // на активном диску формира текстуалну датотеку са именом pismo.txt
```

Карактеристика ове процедуре је да не проверава да ли на диску већ постоји датотека са наведеним именом. То значи, ако датотека већ постоји, биће уништени сви подаци који су у њој били чувани. Зато, ако знамо да је датотека већ формирана (некад раније), а хоћемо да је користимо (тј. желимо да сачувамо претходне податке), нећемо употребити ову већ, стандардну процедуру *Reset*:

```
Reset(imenik);    // отвара датотеку са именом imena.dat на активном диску
Reset(opis);      // отвара текстуалну датотеку са именом pismo.txt која постоји на активном диску
```

Основни проблем који се може јавити код употребе ове процедуре је да датотека не постоји (тј. у тренутку извршења програма нисмо сигурни да ли датотека постоји или не). Ако датотека не постоји на активном диску (могуће је да смо формирали датотеку, али је она касније намерно или грешком избрисана) употреба процедуре *Reset* прекида извршавање програма и пријављује се *I/O* (*input/output*) грешка. Проблем се решава стандардном функцијом *IOResult* која тестира претходну улазно-излазну радњу (било које обраћање датотеци) и која може имати вредност 0 ако у претходној улазно-излазној радњи није дошло до грешке, односно, било који други цео, позитиван број (до 256) у зависности од врсте грешке.

Зато, у случају да не знамо да ли датотека постоји или не, а ако не желимо да изгубимо податке који су, евентуално, у датотеци, правилно би било креирати датотеку на почетку програма на следећи начин:

```
{$I-} Reset(imenik); {$I+}
If IOResult<>0
 then ReWrite(imenik);
```

Објаснимо шта значи то што је у претходном примеру написано:
{$I-} - директива која саопштава преводиоцу да у случају *I/O* (*Input/Output*, тј. улазно/излазне) грешке не прекида програм, тј. одузима преводиоцу контролу над *I/O* грешкама,
Reset(imenik) - команда *покушава* да отвори датотеку за рад као да она постоји,
{$I+} - директива враћа преводиоцу контролу над *I/O* грешкама,
If IOResult<>0 - тестира претходну улазно-излазну радњу и ако је датотека већ постојала на диску функција *IOResult* је добила вредност 0, па ће се прећи на следећу наредбу иза If наредбе, а ако датотека није постојала, дошло је до *I/O* грешке и функција *IOResult* је добила неку вредност различиту од 0, па ће се извршити наредба у *then* грани, односно, *ReWrite(imenik)*, тј. формираће се нова (и празна) датотека на диску.

Код текстуалних датотека поступићемо овако:

```
{$I-} Append(opis); {$I+}
If IOResult<>0
 then ReWrite(opis);
```

На овај начин смо решили проблем прекида програма у случају непостојања датотеке или губљења података у случају постојања датотеке са којом желимо да радимо.

Наравно, није ово једина примена функције *IOResult*. Она се може употребљавати и у другим случајевима улазно-излазних радњи, али треба запамтити да се њена вредност може употребити само једном за једну радњу, тј. код сваког позива њена вредност се мења:

```
{$I-} Reset(imenik); {$I+}
i:=IOResult;
If IOResult<>0
 then ReWrite(imenik);
```

променљива *i* ће чувати вредност функције *IOResult* у покушају да се отвори датотека, па ако датотека не постоји *i* ће добити вредност различиту од 0. У *If* наредби се, поново, позива функција *IOResult*, али пошто није било улазно/излазних радњи њена вредност ће бити 0 иако датотека стварно не постоји, па неће бити креирана ни наредбом *ReWrite* што ће, као

последицу, имати појаву грешке код прве следеће улазно/излазне радње са датотеком (јер датотека и даље не постоји). Зато, ако из неког разлога желимо да вредност функције *IOResult* доделимо променљивој *i* онда би исправно било написати овај део кода на следећи начин:

```
{$I-} Reset(imenik); {$I+}
i:=IOResult;
If i<>0
 then ReWrite(imenik);
```

Упис података, читање и претраживање датотеке

Када је датотека формирана, потребно је у њу сместити неке податке. За овај посао користе се стандардне процедуре *Write* и *WriteLn*:

```
Write(imenik,ime);       // у датотеку imena.dat уписује се вредност променљиве ime.
WriteLn(opis,'текст између апострофа или променљива типа стринг');
Append(opis);            // у текстуалну датотеку pismo.txt уписује се текст или вредност
                         // текстуалне променљиве са ознаком краја реда.
```

Упис података у датотеку остварује се, обично, у неком циклусу са критеријумом краја уписа који задовољава услове задатка (најчешћи су услови задат број података или специјално означени крај података). Додељивање вредности променљивој чија се вредност уписује у датотеку може се извршити стандардном наредбом *Read*, наредбом додељивања, уписом у неку текст компоненту (на пример, унос у неки едит) или на неки други начин (на пример, читањем из неке друге датотеке):

```
Read(ime);
ime:=a;
ime:='Viktorija';
ime:=a+b;
```

Када је посао уписа података завршен треба *затворити* датотеку, тј. означити крај датотеке. Ако се то не уради подаци се неће моћи пронаћи и сав посао уписа био је узалудан. У делфију, када се програм изврши, аутоматски се затварају све отворене датотеке, али се може десити да се неки подаци том приликом изгубе, зато би требало писати коректно програме, тј. програмски затварати датотеке, а не ослањати се на срећу и очекивања. Затварање датотеке се остварује стандардном процедуром *CloseFile*:

```
CloseFile(imenik);
```

Код текстуалних датотека не мора се вршити затварање датотеке уколико се после сваког уписа употреби наредба Append. У паскалу, дозвољено је само једно затварање датотеке. Насупрот томе, дозвољено је произвољан број пута употребити наредбу Reset.

Читање података из датотеке остварује се помоћу стандардних процедура Read и ReadLn:

```
Read(imenik,ime)  // из датотеке imena.dat читамо један податак и додељујемо га променљивој ime
Read(opis,ime)    // из текстуалне датотеке pismo.txt читамо један ред и додељујемо га променљивој
```

За читање свих података из датотеке треба користити неки циклус са условом. За критеријум краја читања најчешће се узима функција *EOF*:

```
While not EOF(imenik) do
   Read(imenik,ime);
While not EOF(opis) do
   ReadLn(opis,ime);
```

Могуће је читање свих података остварити и у *For* циклусу уз помоћ функције *FileSize* (не важи за текстуалне датотеке):

```
For i:=1 to FileSize(imenik) do
  Read(imenik,ime);
```

То што је неки податак прочитан из датотеке не значи да смо га и видели. Да би се прочитани податак могао видети морају се употребити наредбе *Write* или *WriteLn* или се прочитана вредност смешта у неку лабелу, едит или други објекат за приказивање података којима се прочитани податак исписује на екрану:

```
Write(ime);
WriteLn(ime);
Label1.Caption:=ime;
```

Важно је, пре покушаја читања свих података из датотеке, употребити наредбу *Reset* да би се показивач података ставио на први податак (ако се то не уради, може се десити да читање почне од неког средњег податка или, у горој варијанти, показивач може бити на крају датотеке, па неће бити прочитан ниједан податак).

Међутим, врло ретко су нам потребни баш сви подаци. Најчешће треба пронаћи један или више одређених података. Читањем читаве датотеке и њеним исписивањем тек би започео

посао тражења жељеног података. Тај податак треба пронаћи једноставније, програмски. Посао претраживања датотеке организује се:

- ако знамо редни број податка (или података) који тражимо
- ако знамо део податка који тражимо

У првом случају користи се стандардна процедура *Seek*:

```
Seek(imenik,redbrpod); // поставља показивач на податак са редним бројем redbrpod
Read(imenik,ime);      // чита изабрани податак
```

У другом случају читају се подаци из датотеке све док се не нађе одговарајући податак или док се не дође до краја датотеке:

```
While not EOF(imenik)or(Pos(sifra,ime)0) do // sifra је део податка који је познат
   Read(imenik,ime);
```

Наравно, ово је пример претраживања у случају када су подаци типа стринг, у другим случајевима услов се другачије формулише. Код текстуалних датотека претраживање се може вршити само на други начин, тј. читањем редом свих података у датотеци док се не дође до жељеног.

Уређивање података датотеке

Подаци у текстуалној датотеци се не могу уређивати, већ се уређују пре уписа у датотеку. Подаци у датотеци са слогом се могу уређивати по абецедном реду (ако су типа стринг или карактер) или по неком другом редоследу. Овај посао није увек једноставан, нарочито ако се ради о датотеци слогова са више поља, где услов уређења не мора бити једноставан. Ако претпоставимо да је датотека добро дефинисана и да су подаци уписани, сам принцип уређења види се у следећем примеру:

```
Reset(imenik); // постави показивач на почетак датотеке
For i:=0 to FileSize(imenik)-2 do // редни број првог податка се мења од првог места у датотеци
  For j:=i+1 to FileSize(imenik)-1 do // до претпоследњег, а другог од првог следећег места до
  Begin                              // последњег у датотеци
       Seek(imenik,i);               // постави показивач на податак на месту i
       Read(imenik,ime1);            // прочитај податак
       Seek(imenik,j);               // постави показивач на податак на месту j
       Read(imenik,ime2);            // прочитај податак
       If ime1>ime2 // упореди прочитане податке и ако треба замени им места у датотеци
         then Begin Seek(imenik,j); // постави показивач на други податак, тј. на место j
                    Write(imenik,ime1); // упиши први податак
                    Seek(imenik,i); // постави показивач на први податак, тј. на место i
                    Write(imenik,ime2); // упиши други податак
              End
  End;
```

Јасно је да се овде ради о абецедном сортирању стрингова, али би се аналогно решили и сложенији случајеви, само што би услов промене места слогова био другачије формулисан.

Брисање података или датотеке

Ако нам датотека више није потребна, онда је треба уклонити са диска да непотребно не заузима меморијски простор. То се врло једноставно ради директно из програма стандардном процедуром *Erase*:

```
Erase(imenik) // физички брише са диска датотеку придружену променљивој imenik
```

Датотека мора бити затворена да би се могла обрисати.

Нешто је тежи случај када нам само неки подаци не требају, па само њих треба избрисати. Ово се може постићи на два начина:

- преписивањем свих потребних података у помоћну датотеку, брисањем главне датотеке и променом имена помоћне датотеке у главну
- померањем свих података иза оног који треба избрисати за једно место у лево и одсецањем краја датотеке.

У првом случају проблем би се решио овако (наравно, подразумева се да су обе датотеке добро декларисане, да је датотека *imena.dat* после првог *пуњења* затворена и да је вредност променљиве *sifra* податак који треба избрисати из датотеке):

```
Reset(imenik);
ReWrite(pom);
For i:=1 to FileSize(imenik) do
Begin Read(imenik,ime);
      If ime<>sifra then Write(pom,ime)
End;
```

```
Close(pom);
Erase(imenik);
Rename(pom,'imena.dat');
```

У другом случају проблем би се решио овако:

```
Reset(imenik);
For i:=1 to FileSize(imenik) do
Begin Read(imenik,ime);
      If ime<>sifra then
       Begin Seek(imenik,i-1);
             Write(imenik,ime)
       End
End;
Seek(imenik,FileSize(imenik)-1);
Truncate(imenik);
```

У овом случају се преко податка који се брише уписује следећи податак, преко овога следећи итд. На крају се последњи податак исписује на претпоследњу позицију и сада имамо два једнака податка. Стандардна процедура *Truncate* брише податке од неког места до краја датотеке, у овом примеру, брише се само последњи податак и тако смо елиминисали *непожељни* податак из датотеке.

У неким верзијама програмског језика процедура *Truncate* не ради, па је корисно упамтити и први начин као начин брисања непотребних података из датотеке, мада је други начин нешто бржи. Други начин се никако не може применити на текстуалне датотеке (јер се у раду са текстуалним датотекама не може користити процедура *Seek*).

У овом делу је употребљена и стандардна процедура *Rename* за промену имена датотеке. Ради се о команди која ради са диском, тј. датотеци на диску се мења име.

Овим смо заокруглили рад са датотекама. Сада је потребно огледати се на конкретном примеру и увежбати рад са датотекама. Наравно, теже је радити са датотекама чији су елементи слогови са већим бројем поља, али принцип рада је исти. Сваки од проблема који смо решавали треба уоквирити у одговарајућу процедуру и тако решити овај сложени посао. Следе задаци који ће бити решавани са променљивама типа слог како бисмо научили њихову употребу и припремили се за програмирање и употребу датотека, а касније и база података. Сваки задатак може се решити и са низом слогова и са датотеком одговарајућег типа.

Задаци са слоговима и датотекама

- ***Саставити програм који прихвата податке о датумима рођења ученика једног одељења и уређује их по датуму.***

Слогове ћемо формирати у оквиру процедуре ***Button1Click***, а затим ћемо, у посебној процедури ***Uredi***, уређивати унете слогове по датуму и исписивати их у листбокс, зато ћемо дефинисати тип ***rodjendan*** и глобалне промењиве ***a*** - низ слогова и ***n*** - редни број слога. Након уноса сваког слога сортирају се сви до тада унети слогови. Предвидели смо да у одељењу има највише 32 ученика. Креираћемо форму као на слици, а затим написати комплетан програм.


```
type rodjendan=record d,m,g:integer;
                      ime:string;
               end;
     podatak=array[1..32] of rodjendan;
      ...
     procedure Uredi;
     procedure FormCreate(Sender: TObject);
      ...
var
  Form1: TForm1;
  a:podatak;
  n:integer;
procedure TForm1.FormCreate(Sender: TObject);
begin n:=0;
end;
procedure TForm1.Button1Click(Sender: TObject);
begin if n<32 then
      begin n:=n+1;
```

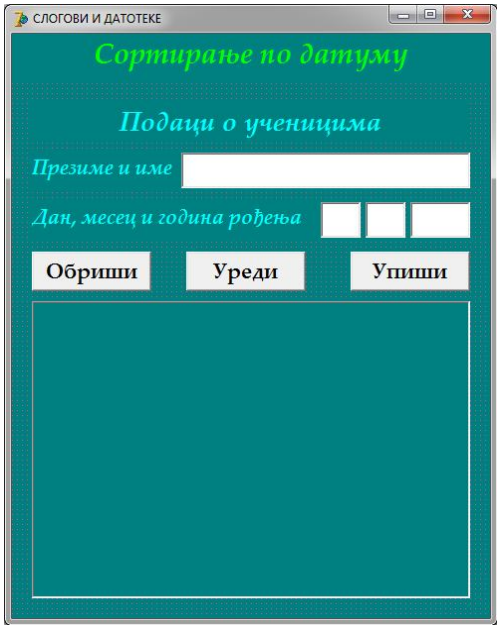
```
            a[n].ime:=Edit1.Text;
            a[n].d:=StrToIntDef(Edit2.Text,1);
            a[n].m:=StrToIntDef(Edit3.Text,1);
            a[n].g:=StrToIntDef(Edit4.Text,1950);
            Uredi;
      end;
      If n=32 then
      begin Edit1.Enabled:=false;
            Edit2.Enabled:=false;
            Edit3.Enabled:=false;
            Edit4.Enabled:=false;
            Button1.Enabled:=false;
      end;
end;
procedure TForm1.Uredi;
var i,j:integer;
    pom:rodjendan;
begin for i:=1 to n-1 do
        for j:=i+1 to n do
          if(a[i].g>a[j].g)or(a[i].g=a[j].g)and(a[i].m>a[j].m)or
            (a[i].g=a[j].g)and(a[i].m=a[j].m)and(a[i].d>a[j].d)
           then begin pom:=a[i];a[i]:=a[j];a[j]:=pom;
          end;
      ListBox1.Clear;
      ListBox1.Items.Add('П Р Е З И М Е   И   И М Е    ДАТУМ РОЂЕЊА');
      for i:=1 to n do
      begin j:=30-Length(a[i].ime);
            ListBox1.Items.Add(a[i].ime+Format('%*s%*d.%*d.%*d.',
                                        [j,' ', 2,a[i].d,2,a[i].m,4,a[i].g]))
      end;
end;
```

Код исписа смо искористили функцију *Format* за групно форматирање исписа у листбоксу, а за испис презимена и имена смо резервисали 30 места. Између апострофа је низ формата са звездицама уместо конкретних бројева места која се резервишу за испис података,а између [ ] је низ бројева места и вредности које се на тим местима исписују. Променљива ***j*** одређује колико од резервисаних 30 није искоришћено за испис презимена и имена. У решењу задатка није коришћена никаква контрола уписа, тј. корисник може уписати било шта као датум рођења (на пример 45.52.1000). Коректне контроле уписа датума урађене су код једноставнијих задатака са простим типовима података, овде би само искомпликовале програм, па би се теже видео принцип рада са слоговима. Наравно, да би све било коректно одрађено ове контроле треба додати.

Коришћењем наредбе ***With*** последње две процедуре се мењају у:

```
procedure TForm1.Button1Click(Sender: TObject);
begin if n<32 then
      begin n:=n+1;
            With a[n] do
            begin ime:=LabeledEdit1.Text;
                        d:=StrToIntDef(LabeledEdit2.Text,1);
                        m:=StrToIntDef(Edit1.Text,1);
                        g:=StrToIntDef(Edit2.Text,1950);
            end;
            Uredi;
      end;
      If n=32 then
      begin LabeledEdit1.Enabled:=false;Edit1.Enabled:=false;
            LabeledEdit2.Enabled:=false;Edit2.Enabled:=false;
            Button1.Enabled:=false;
      end;
end;
procedure TForm1.Uredi;
var i,j:integer;
    pom:rodjendan;
begin for i:=1 to n-1 do
        for j:=i+1 to n do
          with a[i] do if(g>a[j].g)or(g=a[j].g)and(m>a[j].m)or
                          (g=a[j].g)and(m=a[j].m)and(d>a[j].d)
                        then begin pom:=a[i];a[i]:=a[j];a[j]:=pom;
                             end;
      ListBox1.Clear;
      ListBox1.Items.Add('П Р Е З И М Е   И   И М Е    ДАТУМ РОЂЕЊА');
      for i:=1 to n do
        with a[i] do
         begin j:=30-Length(ime);
               ListBox1.Items.Add(ime+format('%*s%*d.%*d.%*d.',
                                        [j,' ', 2,a[i].d,2,a[i].m,4,a[i].g]))
         end;
end;
```

Следи решење са коришћењем датотеке уместо низа слогова. Тастер *Упиши* унети податак уписује у датотеку иза последњег уписаног и приказује га као последњи у *ListBox*-у не водећи рачуна о сортирању. Тастер *Уреди* сортира податке у датотеци према датуму и исписује сортиране податке у *ListBox*. Пошто овде нисмо имали потребу за бројањем података број уписа није ограничен, а ако желимо да започнемо нови унос кликом на тастер *Обриши* празнимо податке из датотеке и *ListBox*-а:

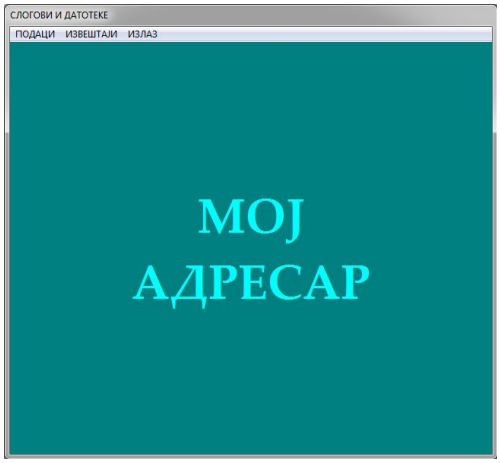


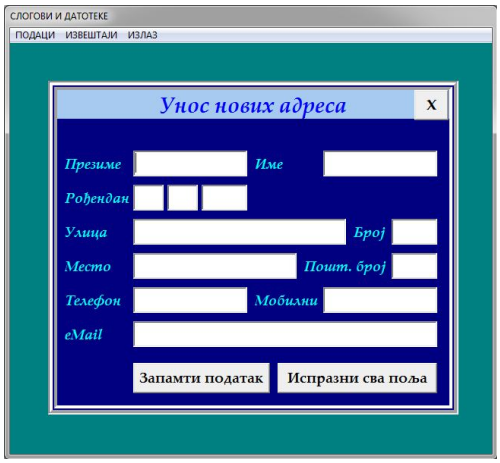

```
procedure TForm1.Button3Click(Sender: TObject);
var n:integer;
begin a.ime:=Edit1.Text;
      a.d:=StrToIntDef(Edit2.Text,1);
      a.m:=StrToIntDef(Edit3.Text,1);
      a.g:=StrToIntDef(Edit4.Text,1950);
      ListBox1.Color:=clTeal;
      n:=30-Length(a.ime);
      ListBox1.Items.Add(a.ime+format('%*s%*d.%*d.%*d.',
                              [n,' ', 2,a.d,2,a.m,4,a.g]));
      Reset(dat);
      Seek(dat,FileSize(dat));
      Write(dat,a);
end;
procedure TForm1.Button1Click(Sender: TObject);
var i,j:integer;
    b:rodjendan;
begin ListBox1.Color:=clMoneyGreen;
      Reset(dat);
      For i:=0 to FileSize(dat)-2 do
        For j:=i+1 to FileSize(dat)-1 do
        begin Seek(dat,i);Read(dat,a);
                 Seek(dat,j);Read(dat,b);
              if(a.g>b.g)or(a.g=b.g)and(a.m>b.m)or(a.g=b.g)and(a.m=b.m)and(a.d>b.d)
              then begin Seek(dat,j);Write(dat,a);
                         Seek(dat,i);Write(dat,b);
                   end;
        end;
      ListBox1.Clear;
      ListBox1.Items.Add('П Р Е З И М Е   И   И М Е    ДАТУМ РОЂЕЊА');
      Reset(dat);
      While not Eof(dat) do
      begin Read(dat,a);
            i:=30-Length(a.ime);
            ListBox1.Items.Add(a.ime+format('%*s%*d.%*d.%*d.',[i,' ', 2,a.d,2,a.m,4,a.g]));
      end;
end;
procedure TForm1.Button2Click(Sender: TObject);
begin ListBox1.Clear;
      ListBox1.Items.Add('П Р Е З И М Е   И   И М Е    ДАТУМ РОЂЕЊА');
      ReWrite(dat);
end;
procedure TForm1.FormCreate(Sender: TObject);
begin ListBox1.Clear;
      ListBox1.Items.Add('П Р Е З И М Е   И   И М Е    ДАТУМ РОЂЕЊА');
      AssignFile(dat,'podaci.dat');
      {$I-} Reset(dat); {$I+}
      If IOResult<>0
           then ReWrite(dat);
      CloseFile(dat);
end;
```

- ***Саставити програм који прихвата податке о адресама и телефонима, а затим омогућити додавање новог податка, измену постојећег податка, брисање непотребног податка и уређивање података по презимену и имену.***

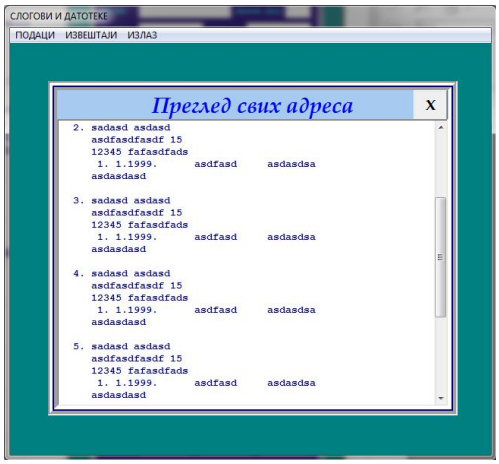
Ово је пример комплетно урађеног задатка са датотекама у делфију. Све процедуре су коректно решене. Објашњења унутар кода су скраћена на најнужнија. Биће приказани сви важнији екрани у програму. Претходна подешавања карактеристика објеката нису наведена јер су прилично предвидљива.

```
type
  podatak=RECORD redbr,d,m,g:integer;
                 prezime, ime, ulica, broj, mesto,pbroj, ftel, mtel, email:string[50];
          END;
  TForm1 = class(TForm)
      ...
var
  dat:FILE OF podatak;
  a:podatak;
  Form1: TForm1;
```



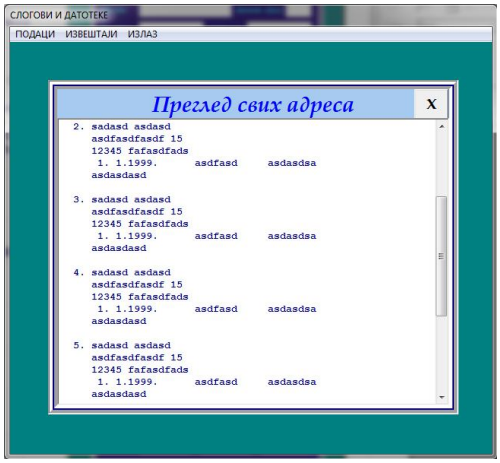



```
// ******************************************
// *** Отвара датотеку са адресама **********
// ******************************************
procedure TForm1.FormCreate(Sender: TObject);
begin Label1.Caption:='MOJ'+#13+'АДРЕСАР';
      AssignFile(dat,'Adresar.dat');
      {$I-} Reset(dat); {$I+}
      If IOResult<>0
       then ReWrite(dat);
      CloseFile(dat);
end;
// ******************************************
// *** Крај рада са програмом ***************
// ******************************************
procedure TForm1.IzlazClick(Sender: TObject);
begin Application.Terminate;
end;
// ******************************************
// *** Унос података у датотеку *************
// ******************************************
procedure TForm1.UnosClick(Sender: TObject);
begin PUnos.Visible:=true; // отвара панел за унос адреса
      UnosP.Click;
end;

procedure TForm1.UnosKClick(Sender: TObject);
begin PUnos.Visible:=false; // затвара панел за унос адреса
end;

procedure TForm1.UnosPClick(Sender: TObject);
var i:integer;
begin For i:=0 to 11 do // празни све едите за нови унос
        TEdit(FindComponent('U'+IntToStr(i))).Clear;
      U0.SetFocus;
end;

procedure TForm1.U0KeyPress(Sender: TObject; var Key: Char);
var i:integer;
begin If key=#13 // дефинише прелазак на следећи објекат када
       then begin For i:=0 to 10 do    // се притисне Ентер
                   If sender=TEdit(FindComponent('U'+IntToStr(i)))
                    then TEdit(FindComponent('U'+IntToStr(i+1))).SetFocus;
                  If sender=U11
                   then UnosZ.SetFocus;
            end;
end;

procedure TForm1.UnosZClick(Sender: TObject);
var n:integer;
begin Reset(dat);  // смешта уписани податак у адресар
      n:=FileSize(dat);
      With a do
      begin prezime:=U0.Text;ime:=U1.Text;
            ulica:=U5.Text; broj:=U6.Text;
            mesto:=U7.Text;pbroj:=U8.Text;
            ftel:=U9.Text; mtel:=U10.Text;
            email:=U11.Text; redbr:=n;
            d:=StrToIntDef(U2.Text,1);
            m:=StrToIntDef(U3.Text,1);
            g:=StrToIntDef(U4.Text,2000);
      end;
      Seek(dat,n);Write(dat,a);
end;
```

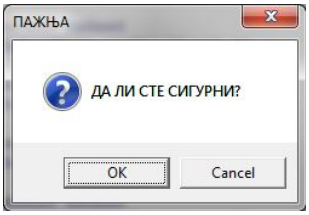



```
// ******************************************
// *** Преглед података у датотеци **********
// ******************************************
procedure TForm1.PregledClick(Sender: TObject);
begin PPregled.Visible:=true; // отвара панел за преглед адреса
      Reset(dat);Memo1.Clear;
      While not Eof(dat) do // чита све податке из датотеке и преноси их у мемо
      begin Read(dat,a);
            With a do
            begin Memo1.Lines.Add(Format('%3d. %1s %1s',[redbr,prezime,ime]));
                  Memo1.Lines.Add('     '+ulica+' '+broj);
                  Memo1.Lines.Add('     '+pbroj+' '+mesto);
                  Memo1.Lines.Add(Format('%7d.%2d.%4d.%13s%13s',[d,m,g,ftel,mtel]));
                  Memo1.Lines.Add('     '+email);
                  Memo1.Lines.Add('');
            end;
      end;
end;

procedure TForm1.PregledKClick(Sender: TObject);
begin PPregled.Visible:=false; // затвара панел за преглед адреса
end;
```

```
// ******************************************
// *** Брисање података из датотеке **********
// ******************************************
procedure TForm1.BrisanjeClick(Sender: TObject);
begin PBrisi.Visible:=true; // отвара панел за брисање адреса
      Reset(dat);Memo2.Clear; // чита све податке из датотеке
      While not Eof(dat) do   // и преноси их у мемо поље
      begin Read(dat,a);
            With a do
            begin Memo2.Lines.Add(Format('%3d. %1s %1s',
[redbr,prezime,ime]));
                  Memo2.Lines.Add('     '+mesto+' '+ulica+
' '+broj);
                  Memo2.Lines.Add('');
            end;
      end;
end;
```

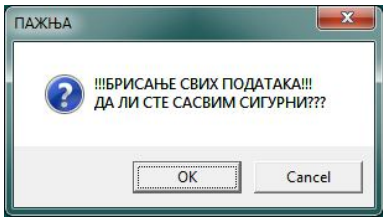


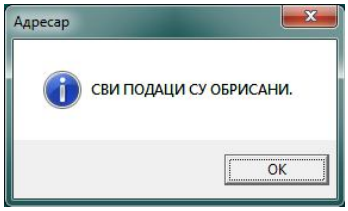

```
procedure TForm1.OBrisiClick(Sender: TObject);
var i,k:integer;
begin If MessageBox(0,'ДА ЛИ СТЕ СИГУРНИ?','ПАЖЊА',MB_OKCANCEL+MB_ICONQUESTION)=IDOK
       then begin k:=StrToInt(Edit1.Text);
                  Reset(dat);
                  For i:=k+1 to FileSize(dat)-1 do
                  Begin Seek(dat,i);Read(dat,a);
                        a.redbr:=a.redbr-1;
                        Seek(dat,i-1);Write(dat,a)
                  End;
                  Seek(dat,FileSize(dat)-1);
                  Truncate(dat);
                  Brisanje.Click;
            end;
end;
```



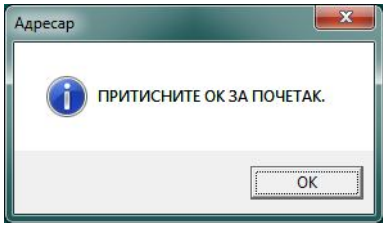

```
procedure TForm1.B1KeyPress(Sender: TObject; var Key: Char);
begin If key=#13
       then BrisiK.Click
       else If not (key in ['0'..'9',#8,#128])
             then key:=#27;
end;

procedure TForm1.BrisiKClick(Sender: TObject);
begin PBrisi.Visible:=false; // затвара панел за брисање адреса
end;
```

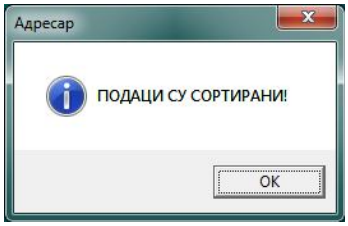
```
// ******************************************
// *** Брисање свих података из датотеке ******
// ******************************************
procedure TForm1.IsprazniClick(Sender: TObject);
begin If MessageBox(0,'!!!БРИСАЊЕ СВИХ ПОДАТАКА!!!'+#13+
                    'ДА ЛИ СТЕ САСВИМ СИГУРНИ???','ПАЖЊА',
MB_OKCANCEL+MB_ICONQUESTION)=IDOK
       then begin ReWrite(dat);
                  CloseFile(dat);
                  MessageBox(0,'СВИ ПОДАЦИ СУ ОБРИСАНИ.','Адресар',
MB_OK+MB_ICONASTERISK);
            end;
end;
```

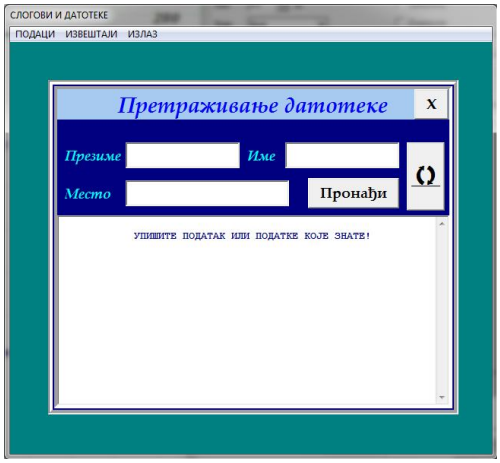





```
// ******************************************
// *** Сортирање података у датотеци **********
// ******************************************
procedure TForm1.SortClick(Sender: TObject);
var b:podatak;
    i,j,p:integer;
begin MessageBox(0,'ПРИТИСНИТЕ ОК ЗА ПОЧЕТАК.','Адресар',
MB_OK+MB_ICONASTERISK);
      Reset(dat);
      For i:=0 to FileSize(dat)-2 do
        For j:=i+1 to FileSize(dat)-1 do
        begin Seek(dat,i);Read(dat,a);
              Seek(dat,j);Read(dat,b);
              If a.prezime+a.ime>b.prezime+b.ime
               then begin p:=a.redbr;
                          a.redbr:=b.redbr;
                          b.redbr:=p;
                          Seek(dat,j);Write(dat,a);
                          Seek(dat,i);Write(dat,b);
                    end
        end;
      MessageBox(0,'ПОДАЦИ СУ СОРТИРАНИ!','Адресар',
MB_OK+MB_ICONASTERISK);
end;
```

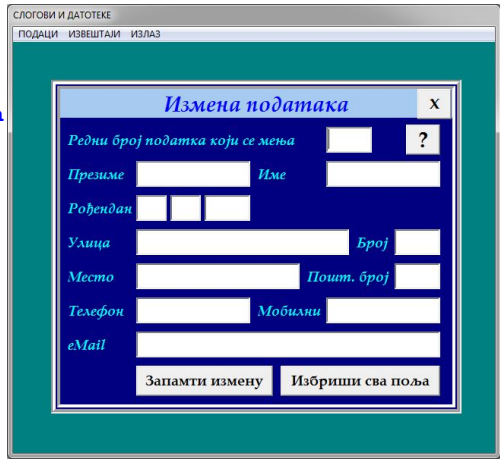







```
// ****************************************
// *** Претраживање података из датотеке ******
// ****************************************
procedure TForm1.PronadjiClick(Sender: TObject);
begin PNadji.Visible:=true; // отвара панел за претраживање
      Memo3.Clear;
      Memo3.Lines.Add('');
      Memo3.Lines.Add('            УПИШИТЕ ПОДАТАК ИЛИ
ПОДАТКЕ КОЈЕ ЗНАТЕ!');
      P1.SetFocus;
end;
procedure TForm1.NadjiKClick(Sender: TObject);
begin PNadji.Visible:=false; // затвара панел за претраживање
end;
procedure TForm1.PromeniClick(Sender: TObject);
begin p1.Clear;p2.Clear;
      p3.Clear;p1.SetFocus;
end;
procedure TForm1.PronadjiPClick(Sender: TObject);
var br:integer;
begin Memo3.Clear;
      If Length(P1.Text+P2.Text+P3.Text)=0
       then MessageBox(0,'Морате уписати бар један податак за претраживање.','ПАЖЊА',
MB_OK+MB_ICONQUESTION)
       else begin br:=0;
                 Reset(dat);
                 While not Eof(dat) do
                 begin Read(dat,a);
                       If Length(P1.Text)>0
                       then If a.prezime=P1.Text
                            then If Length(P2.Text)>0
                                 then If a.ime=P2.Text
                                      then If Length(P3.Text)>0
                                           then If a.mesto=P3.Text            // p1+p2+p3
                                                then begin br:=br+1;Unesi;
                                                     end
                                                else
                                           else begin br:=br+1;Unesi;         // p1+p2
                                                end
                                      else
                                 else If Length(P3.Text)>0
                                      then If a.mesto=P3.Text                 // p1+p3
                                           then begin br:=br+1;Unesi;
                                                end
                                           else
                                      else begin br:=br+1;Unesi;              // p1
                                           end
                            else
                       else If Length(P2.Text)>0
                            then If a.ime=P2.Text
                                 then If Length(P3.Text)>0
                                      then If a.mesto=P3.Text                 // p2+p3
                                           then begin br:=br+1;Unesi;
                                                end
                                           else
                                      else begin br:=br+1;Unesi;              // p2
                                           end
                                 else
                            else If Length(P3.Text)>0
                                 then If a.mesto=P3.Text                      // p1+p3
                                      then begin br:=br+1;Unesi;
                                           end
                                      else
                                 else;
                 end;
                 MessageDlg('Број пронађених података: '+IntToStr(br),mtInformation,[mbOk],0);
           end;
       Promeni.SetFocus;
end;
procedure TForm1.P1KeyPress(Sender: TObject; var Key: Char);
begin If key=#13  // дефинише прелазак на следећи објекат када се притисне Ентер
       then If sender=P1
        then P2.SetFocus
        else If sender=P2
         then P3.SetFocus
         else If sender=P3
          then PronadjiP.SetFocus;
end;
procedure TForm1.Unesi;
begin With a do
      begin Memo3.Lines.Add(Format('%3d. %1s %1s',[redbr,prezime,ime]));
```

```
            Memo3.Lines.Add('     '+ulica+' '+broj);
            Memo3.Lines.Add('     '+pbroj+' '+mesto);
            Memo3.Lines.Add(Format('%7d.%2d.%4d.%13s%13s',[d,m,g,ftel,mtel]));
            Memo3.Lines.Add('     '+email);
      end;
      Memo3.Lines.Add('');
end;
// ****************************************
// *** Исправка података из датотеке **********
// ****************************************
procedure TForm1.IzmenaClick(Sender: TObject);
begin PIzmena.Visible:=true; // отвара панел за измену података
      IBrisi.Click;
      Reset(dat);
      I0.Clear;I0.SetFocus;
end;
procedure TForm1.IzmenaKClick(Sender: TObject);
begin PIzmena.Visible:=false; // затвара панел за измену података
end;
procedure TForm1.Ispisi;
begin With a do
      begin I1.Text:=prezime;I2.Text:=ime;
            I6.Text:=Ulica;  I7.Text:=broj;
            I8.Text:=mesto;  I9.Text:=pbroj;
            I10.Text:=ftel;  I11.Text:=mtel;
            I12.Text:=email;
            I3.Text:=IntToStr(d);
            I4.Text:=IntToStr(m);
            I5.Text:=IntToStr(g);
      end;
end;
procedure TForm1.I0KeyPress(Sender: TObject; var Key: Char);
begin If key=#13
       then begin Seek(dat,StrToIntDef(I0.Text,FileSize(dat)-1));
                  Read(dat,a);Ispisi;
                  I1.SetFocus;
            end
       else If not (key in ['0'..'9',#8,#128])
             then key:=#27;
end;
procedure TForm1.IPamtiClick(Sender: TObject);
begin With a do
      begin prezime:=I1.Text;ime:=I2.Text;
            Ulica:=I6.Text; broj:=I7.Text;
            mesto:=I8.Text;pbroj:=I9.Text;
            ftel:=I10.Text; mtel:=I11.Text;
            email:=I12.Text;
            d:=StrToIntDef(I3.Text,1);
            m:=StrToIntDef(I4.Text,1);
            g:=StrToIntDef(I5.Text,2000);
      end;
      Seek(dat,StrToIntDef(I0.Text,FileSize(dat)));Write(dat,a);
      IBrisi.Click;
      I0.Clear;I0.SetFocus;
end;
procedure TForm1.IBrisiClick(Sender: TObject);
var i:integer;
begin For i:=1 to 12 do // празни све едите за нови унос
        TEdit(FindComponent('I'+IntToStr(i))).Clear;
      I1.SetFocus;
end;
procedure TForm1.IPregledClick(Sender: TObject);
begin Memo4.Visible:=true;
      Reset(dat);Memo4.Clear;
      While not Eof(dat) do // чита све податке из датотеке и преноси их у мемо
      begin Read(dat,a);
            With a do
            begin Memo4.Lines.Add(Format('%3d. %1s %1s',[redbr,prezime,ime]));
                  Memo4.Lines.Add('     '+ulica+' '+broj);
                  Memo4.Lines.Add('     '+pbroj+' '+mesto);
                  Memo4.Lines.Add(Format('%7d.%2d.%4d.%13s%13s',[d,m,g,ftel,mtel]));
                  Memo4.Lines.Add('     '+email);
                  Memo4.Lines.Add('');
            end;
      end;
      Memo4.SetFocus;
end;
procedure TForm1.Memo4KeyPress(Sender: TObject; var Key: Char);
begin If key=#13
       then begin Memo4.Visible:=false;
```



```
                I0.SetFocus;
            end;
end;
procedure TForm1.I1KeyPress(Sender: TObject; var Key: Char);
var i:integer;
begin If key=#13  // дефинише прелазак на следећи објекат када се притисне Ентер
       then begin For i:=1 to 11 do
                   If sender=TEdit(FindComponent('I'+IntToStr(i)))
                    then TEdit(FindComponent('I'+IntToStr(i+1))).SetFocus;
                  If sender=I12
                   then IPamti.SetFocus;
            end;
end;
```

Задаци за самосталан рад

- ***Саставити програм који прихвата координате тачака у простору све док се не упишу кординате (0, 0, 0), а затим одређује тачку најближу координатном почетку и тачку која је најудаљенија од те тачке.***
- ***Саставити програм који прихвата углове у облик степен, минут секунд све док се не упише угао 0, 0, 0, а затим одређује колико има оштрих, правих и тупих углова и колико има неконвексних углова.***
- ***Саставити програм који прихвата податке са такмичења у облику: презиме и име ученика, одељење, школа и број освојених поена, а затим одредити која школа има највише освојених поена и исписује имена три најбоље пласирана ученика.***
- ***Саставити програм који прихвата податке о изостанцима ученика у облику: презиме и име ученика, одељење и број оправданих и неоправданих изостанака, а затим исписати име ученика који има највише изостанака и одељење које има најмањи број неоправданих изостанака. Формирати листу одељења по просечном броју укупних изостанака.***
- ***Саставити програм који прихвата податке о висинама и тежинама ученика једног одељења, а затим исписује имена највишег, најнижег, најлакшег и најтежег ученика, као и просечну висину и тежину ученика и ученика који је најближи просеку.***
- ***Саставити програм који прихвата податке о успеху ученика на крају школске године, а затим исписује ученике уређене по средњој оцени и имена ученика који имају највише петица или четворки.***

Литература