Programiranje 1

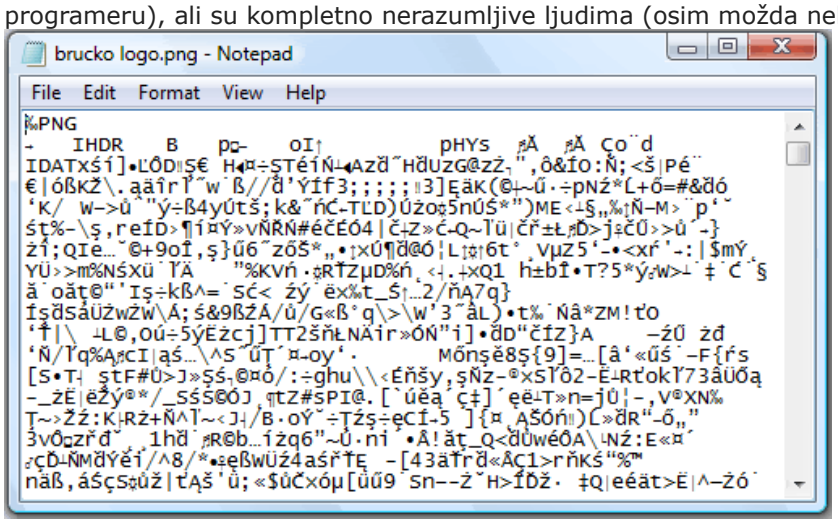
grupno spremanje (zadaci)

datoteke

Tipovi datoteka

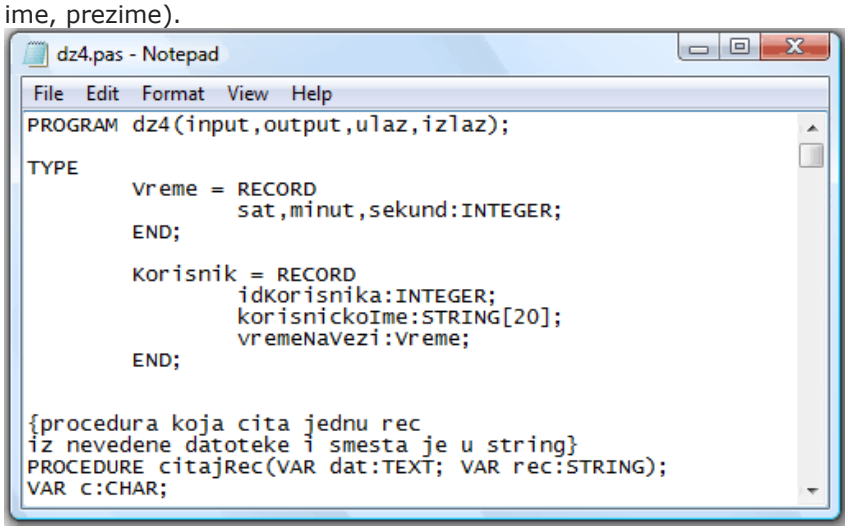
Datoteke se mogu podeliti na binarne i tekstualne. Iako su na prvi pogled ova dva tipa veoma slična oni se suštinski razlikuju.

Binarne datoteke služe pretežno za skladištenje podataka jednog tipa koje program koristi za rad. Njih kreira računar tako što memorijski sadržaj neke lokacije (promenljive) bukvalno prepiše u datoteku (isto kao što je smešten u operativnij memoriji). Pošto je datoteka smeštena na hard disku ona omogućava da program u nju smešta podatke koje koristi između svakog startovanja. Na primer, igrica u kojoj imaš mogućnost da snimiš napredak može sve svoje promenljive (njihove trenutne vrednosti) jednom naredbom da smesti u binarnu datoteku i kasnije po novom startovanju pročita. Zbog načina na koji se podaci smeštaju u njih ove datoteke su veoma zgodne za čitanje i pisanje računaru (odnosno programeru), ali su kompletno nerazumljive ljudima (osim možda nekim RTI zaluđenicima).



Primer 1 — png slika je binarna datoteka

Tekst datoteke služe najviše za smeštanje podataka koji su namenjeni za čitanje izvan programa. One sadrže običan tekst, koji je najčešće formatiran na određen način kako bi podaci bili jasniji. To su često razne tabele, spiskovi i sl. Pošto ovakve datoteke pretežno kreira čovek kucanjem teksta, pravila koja se primenjuju sa smeštanje podataka u njih (koja nisu standardizovana već zavise od svake datoteke) nisu razumljiva računaru, pa su one teže za učitavanje i ispisivanje na računaru (odnosno programeru je teže da ih koristi). Ipak, pošto su razumljive svakome, one se često koriste za smeštanje podataka iz programa koje su namenjeni za čitanje običnom korisniku. Na primer program koji pravi spisak studenata za polaganje ispita bi mogao taj spisak da smesti u tekst datoteku, po nekom formatu (npr. svaki red datoteke sadrži podatke o jednom studentu odvojene razmakom i to: redni broj sa spiska, broj indeksa, ime, prezime).



Primer 2 — Paskal kod koji se kuca čuva kao tekst datoteku

Definisanje datotečke promenljive

Ukoliko želimo u programu da čitamo iz datoteke ili da pišemo u nju potrebno je da definišemo datotečku promenljivu. Ova promenljiva će nam služiti da bismo pratili dokle smo stali sa čitanjem i pisanjem, da naznačimo o kojoj se datoteci radi i tako dalje. Drugim rečima ona će predstavljati datoteku sa kojom radimo. Za tekst datoteke tip datotečke promenljive je **TEXT**, a za binarne **FILE OF *TIP***, gde je *TIP* definicija odgovarajućeg tipa promenljive koji je smešten u datoteku (npr. ako binarna datoteka sadrži cele brojeve onda je to **INTEGER**, ako sadrži niz od deset celih brojeva onda je to **ARRAY [1.10] OF INTEGER** itd.). Na primer, da bismo u programu definisali datotečku promenljivu pišemo:

```
VAR dat:TEXT; {za tekst datoteku}

VAR dat:FILE OF INTEGER; {za binarnu datoteku integera}
```

Tip binarne datoteke je bitan zato što se u nju mogu smeštati (ili čitati) samo podaci jednog tipa, a ne više različitih podataka. Ako želimo da smestimo više podataka različitog tipa onda ih moramo objediniti unutar *RECORD*-a:

```
TYPE student = RECORD
        indeks:INTEGER;
        ime, prezime:STRING[50];
     END;

VAR dat:FILE OF student;
{dat je datoteka koja u sebi sadrži ceo record student, tj. kada
se u nju piše, upisaće se ceo record, sa svim poljima, i kada se
čita pročitaće se ceo record sa svim poljima}
```

Povezivanje datoteke i datotečke promenljive

Pre početka rada sa datotekom potrebno je datotečku promenljivu povezati sa datotekom na disku. Ovo se radi pomoću procedure **assign(*datotecka_promenljiva*,*ime_datoteke_na_disku*)**, gde je *datotecka_promenljiva* odgovarajuća promenljiva koju hoćemo da povežemo sa datotekom, a *ime_datoteke_na_disku* string koji predstavlja putanju do datoteke koju hoćemo da čitamo. Evo primeva za povezivanje datotečke promenljive *dat* i datoteke koja se zove *datoteka.txt*:

```
assign(dat, 'datoteka.txt');
{navodnici stoje kod imena jer je u pitanju string}
```

Otvaranje datoteke za čitanje i pisanje

Iz datoteke se može čitati ili se u nju može pisati. Da bismo datoteku otvorili za čitanje koristimo proceduru **reset(*datotecka_promenljiva*)**. Ova procedura kada se god pozove pripremi datoteku za čitanje, i postavi pokazivač na njen početak (pokazivač pokazuje gde smo stali sa čitanjem). Da bismo datoteku otvorili za pisanje koristi se procedura **rewrite(*datotecka_promenljiva*)**. Ova procedura priprema datoteku za pisanje, i briše sav sadržaj iz nje (ako je on postojao).

```
assign(datoteka_za_citanje, 'ulaz.txt'); {povezuje datoteku
ulaz.txt sa datoteckom promenljivom}
reset(datoteka_za_citanje); {otvara datoteku za citanje}

assign(datoteka_za_pisanje, 'izlaz.txt'); {povezuje datoteku
izlaz.txt sa datoteckom promenljivom}
rewrite(datoteka_za_pisanje); {otvara datoteku za pisanje}
```

Zatvaranje datoteke

Posle uspešno obavljene obrade obavezno treba je zatvoriti sve otvorene datoteke. Ovo nije samo fraza, jer ako se datoteka koja se menja na zatvori izmene koje su nad njom načinjene neće biti sačuvane. Takođe, ako se datoteka koja se čita ne zatvori neće biti dostupna drugim korisnicima (programima). Datpteka se zatvara procedurom **close(*datotecka_promenljiva*)**.

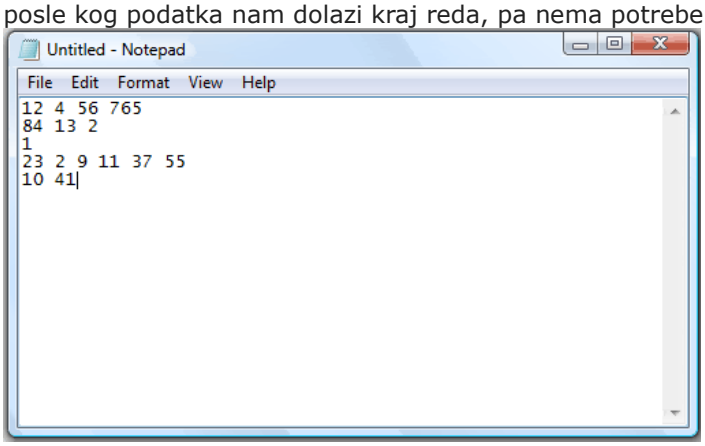
Ispitvanje kraja datoteke i kraja reda

Datoteke su najčešće napoznate dužine. Pošto se čitaju uglavnom segmentno, podatak po podatak, potrebno je nekako utvrditi kada su pročitani svi podaci, tj. kada smo došli do kraja datoteke. Za proveru da li smo došli do kraja datoteke možemo koristiti funkciju **eof(*datotecka_promenljiva*)**. Ova funkcija će proveriti da li postoji još sadržaja u datoteci i ako postoji vratiće kao rezultat *false*, a ako nema više sadržaja (ako smo došli do kraja datoteke) vratiće kao rezultat *true*. Ova funkcija se primanjuje i na binarne i na tekst datoteke.

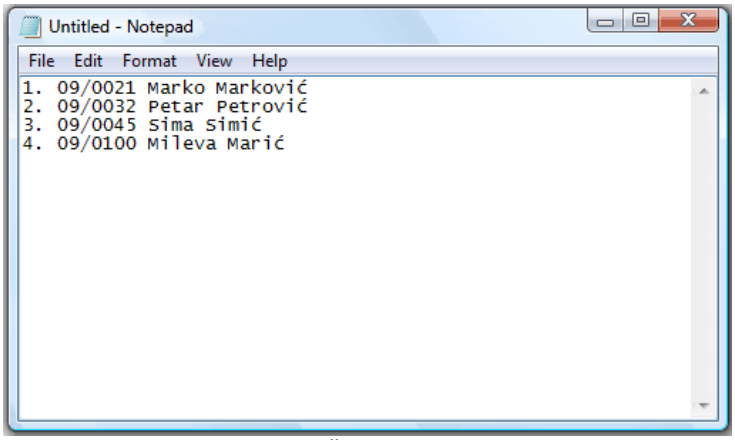
Pošto su binarne datoteke nizovi nekih podataka, upisanih binarno (isto kao i u memoriju) pojam novog reda ne postoji, tj. svi podaci seređaju jedan za drugim, bez razmaka, pauza, prelazaka u novi red i slično. Iako se na primeru koji sam naveo na slici vidi više redova, njih je dodao *notepad* kako bi se datoteka lakše videla (kako bi stala u okvir prozora), ali inače ne postoje u datoteci. Sa druge strane tekst datoteke imaju pojam novog reda i to je jedan od glavnih načina da se podaci razdvoje. To se jasno vidi na primeru datoteke iz Paskala.

Da bismo utvrdili da li smo došli do kraja reda (samo za tekst datoteke!) koristi se funkcija **eoln(*datotecka_promenljiva*)**. Ona će u slučaju da smo naišli na kraj reda vratiti *true*, a u slučaju da ima i dalje znakova u redu vratiti *false*.

Provera kraja reda nije uvek neophodna. Zapravo, ovu proveru ćemo sprovoditi samo ako nismo sigurni kako izgleda jedan red i da li smo prilikom čitanja došli do kraja. Na primer, ako imamo tekst datoteku u kojoj se u svakom redu nalazi proizvoljan broj celih brojeva, onda prilikom čitanja jednog reda ne možemo znati da li smo došli do njegovog kraja ili ne, pa to moramo ispitati pomoću funkcije *eoln*. Sa druge strane, ako se u datoteci nalazi spisak studenata za polaganje ispita, koji u svakom redu sadrži podatke o studentu (red. br., br. indeksa, ime, prezime — tačno poznat format reda), znaćemo tačno posle kog podatka nam dolazi kraj reda, pa nema potrebe to posebno proveravati.



Primer 3 — datoteka sa nepoznatim formatom reda — svaki red se čita drugačije, pa je potrebno posle svakog pročitanog broja ispitati da li se došlo do kraja



Primer 4 — datoteka sa tačno poznatim formatom reda — svaki red se čita na isti način, pa ga je moguće pročitati odjednom, bez provera da li se došlo do kraja reda

Čitanje i pisanje u datoteku

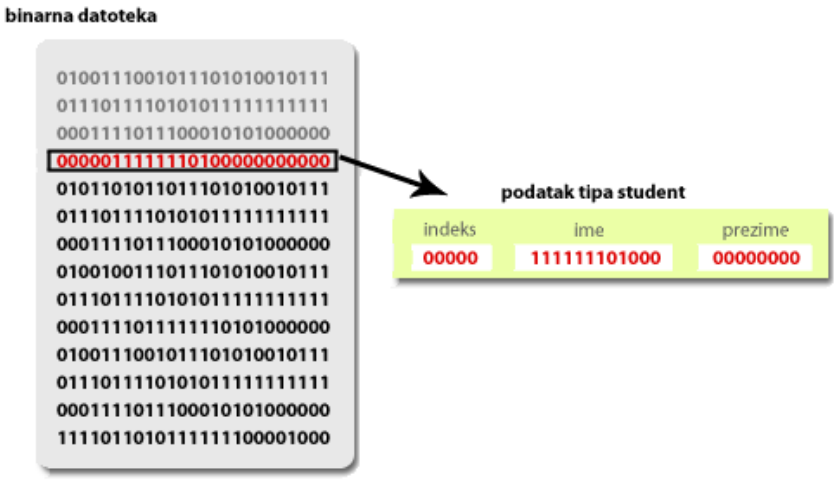
Jedina razlika koja se primećuje između binarnih i tekst datoteka je prilikom njihovog čitanja.

Binarna datoteka

Da bi se čitalo iz binarne datoteke koristi se procedura **read(*datoteka, podatak*)**, gde je *datoteka* odgovarajuća datotečka promenljiva (datoteka iz koje se čita), a *podatak*promenljiva u koju se čita. Bitno je boditi računa o tipu promenljive *podatak* jer on mora da bude isti kao i tip samo datoteke. Npr. ako smo na početku binarnu datoteku definisali kao **FILE OF INTEGER** onda ćemo želeti da nam i *podatak* po tipu bude *integer*, jer u suprotnom ne bismo obavili čitanje. Ovo važi i za složene tipove podataka. (Pogledajte gornji primer sa datotekom **FILE OF student**. I podaci iz ove dadoteke čitali bi se naredbom **read(*dat, podatak_tipa_student*)** jer bi se svaki bit odgovarajućeg podatka iz binarne datoteke samo prepisao u promenljivu *podatak_tipa_student*, i tako odjednom popunio sva njegova polja — indeks, ime, prezime — bez posebnog naglašavanja. Ovo je moguće zato što je računar smestio podatak u datoteku, i prema tipu on tačno zna kako treba da ga pročita.)

Da bi se podatak upisivao u binarnu datoteku koristi se procedura **write(*datoteka, podatak*)**, gde je *datoteka* odgovarajuća datotečka promenljiva (datoteka u koju se piše), a *podatak* promenljiva koja se ispisuje, naravo odgovarajućeg tipa (isto kao i za *read*).

Za binarnu datoteku se ne mogu koristiti procedure *readln* i *writeln*!

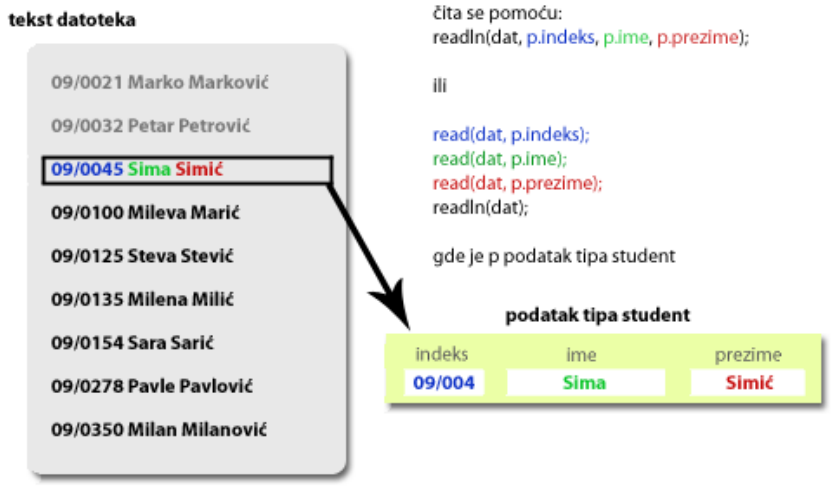


Primer 5 — čitanje binarne datoteke — sivi bitovi su pročitani, čitaju se crveni bitovi tako što se samo prepišu na

odgovarajuća mesta u operativnu memoriju računara

Tekst datoteka
Iz tekstualne datoteke se može čitati pomoću procedura **read(*datoteka*, *p1*, *p2*, ..., *pn*)** ili **readln(*datoteka*, *p1*, *p2*, ..., *pn*)**, gde su *p1*, *p2*, ..., *pn* odgovarajuće promenljive čiji tip zavisi isključivo od formata teksta koji se čita. Razlika između ove dve provedure je što će *read* pročitati podatak i ostaviti pokazivač u istom redu (znači očekuje se da postoji još podataka u tom redu) a *readln* će pročitati podatke, i odmah preći u novi red (odboacujući sve preostale podatke u tom redu, ako ih ima).

Pisanje u tekst datoteku se vrši pomoću procedura **write(*datoteka*, *p1*, *p2*, ..., *pn*)** ili **writeln(*datoteka*, *p1*, *p2*, ..., *pn*)**, gde su *p1*, *p2*, ..., *pn* odgovarajuće promenljive čiji tip zavisi isključivo od formata teksta koji se piše. Razlika je u tome što *write* ispisuje teks i ostaje u istom redu (sledeći tekst se ispisuje u istom redu), a *writeln* posle ispisa prelazi u novi red (sledeći tekst će biti u novom redu).



Primer 6 — čitanje tekst datoteke — sivi redovi su pročitani, i čita se uokvireni red, ali tako što se svaki podatak iz reda mora pročitati posebno, na odgovarajući način, u zavisnosti od tipa tog podatka

Primer 7 — razlika između *read* i *readln*

Algoritam šablona za rešavanje zadataka

Ovako izgleda šablon koji može pomoći početnicima za rešavanje zadataka sa datotekama. Baziran je na

nekoliko ključnih stvari koje se mogu razlikovati.

za sve datoteke

```
TYPE TIP = RECORD{samo ukoliko se barata složenim podacima!!! inače je tip prost podatak}
        polje1:tip1;
        polje2:tip2;
        ...
        poljeN:tipN;
     END;
```

za binarne datoteke

```
VAR dat: FILE OF TIP;
podatak: TIP; {tip je odgovarajući tip podatka smešten u datoteku}
```

za tekst datoteke

```
VAR dat: TEXT;
podatak: TIP; {tip obuhvate sve podatke koji se semantički mogu grupisati iz datoteke}
```

za sve datoteke

```
BEGIN
        assign(dat, 'imedatoteke');
        reset(dat);
```

za binarne datoteke

```
           while not eof(dat) do begin
                read(dat, podatak);
                OBRADA;
        end;
```

za tekst datoteke

1) ako je poznat tačan format reda

```
           while not eof(dat) do begin
              readln(dat, podatak.polje1, podatak.polje2,... podatak.poljeN);
              OBRADA;
            end;
```

1) ako nije poznat tačan format reda

```
           while not eof(dat) do begin
              while not eoln(dat) do begin {dok se ne dođe do kraja reda}
                   read(dat, podatak.polje1, podatak.polje2,...
                        podatak.poljeN); {čita se jedan podatak iz reda}
                   OBRADA;
              end;
              readln(dat); {kada se dođe do kraja reda prelazi se
                            u novi red da bi se nastavilo čitanje}
           end;
```

za sve datoteke

```
           close(dat);
END;
```

U ovim primerima *obrada* podrazumeva bilo kakav vid obrade nad podacima, počevši od njihovog smeštanja u niz, pa do nekih komplikovanijih stvari.

Prenošenje datoteke kao paramatra potprograma

Datoteka se uvek prenosi po **referenci**. Ovo je neophodno da bi se svaki put prilikom njenog pisanja ili čitanja zapamtilo mesto na kom se stalo. Prenošenje datoteke se izvodi kao i prenošenje bilo koje druge

promenljive: **PROCEDURE p(VAR datoteka:TEXT);**.

Čitanje standardnih tipova podataka iz tekst datoteke

Tekst datoteka se ponaša potpuno isto kao i standatni ulaz/izlaz. U nju je moguće pisati/čitati sve standardne proste tipove podataka, kao i stringove. To podrazumeva realne brojeve (**REAL**), cele brojeve (**INTEGER**), znakove (**CHAR**), logičke promenljive (**BOOLEAN**) i stringove (**STRING**). Svi ostali tipovi podatak moraju se realizoveti pomoću ovih tipova. To praktično znači da ako imamo jedan red tekst datoteke u kome se nalazi ceo broj, realan broj, i string, mi ga možemo pročitati pomoću naredbe **readln(dat,*podatak_real*, *podatak_integer*, *podatak_string*)**, gde su podataci promenljive odgovarajućih tipova. Ovo se najčešće realizuje korišćenjem promenljive tipa *RECORD*, i učitavanjem svakog pojedinačnog polja: **readln(dat, *podatak.r*, *podatak.i*, *podatak.s*)**.

Zbog česte kompleksnosti podatka koji se čita iz tekst datoteke poželjno je čitanje jednog reda realizovati kao posebnu proveduru koja po izlgedau sliči proceduri *readln*. Ovo često dosta olakša ceo posao čitanja.

Problem čitanja stringa

Najveći problem prilikom čitanja zadaju stringovi. Procedure *read* i *readln* čitaju string do kraja read, ili do kraja stringa (ako je string kraći od reda), a ne do prvog razmaka. Ovo nam zadaje problem ako se string nađe u sredini ili na početku reda. Tada je potrebno čitati ga znak po znak, da ne bi došlo do gutanja ostalih podataka. Najbolji način rešavanja ovog problema je napraviti funkciju koja će pročitati string ali do prvog razmaka, a ne do kraja reda. Takva funkcija bi izgledala ovako:

```
{procedura koja cita jednu rec iz nevedene datoteke
i smesta je u string}
PROCEDURE citajRec(VAR dat:TEXT; VAR rec:STRING);
VAR c:CHAR;

BEGIN
    rec := '';

    {gutanje razmaka i hvatanje prvog znaka}
    repeat
         if (not eof(dat)) and (not eoln(dat)) then read(dat,c)
         else break; {prekida se ako se dodje do kraja reda
                      ili kraja datoteke}
    until (c <> ' '); {prekida se kada se procita prvi znak
                       koji nije razmak}

    {citanje stringa do prvog razmaka}
    while (c <> ' ') do begin {ponavlja se dok se ne naidje
                               na razmak}
         rec := rec + c; {procitano slovo se dodaje na rec}
         if (not eof(dat)) and (not eoln(dat)) then
            read(dat,c) {cita se novo slovo ako nije kraj reda
                         ili datoteke}
         else break; {prekida se ako se dodje do kraja reda
                      ili kraja datoteke}
    end;
END;
```

Mnogo jednostavnije rešenje koje rešava problem čitanja reči u sredini ili na početku reda (može da se koristi samo ako znamo da nije kraj datoteke, i da reč nije poslednja u redu):

```
PROCEDURE citajRec(VAR dat:TEXT; VAR rec:STRING);
VAR c:CHAR;
BEGIN
    rec := '';
    read(dat,c);
    while (c <> ' ') do begin
         rec := rec + c;
         read(dat,c);
    end;
END;
```

Zadatak 2 (13.02.2009)
U tekst datoteci ulaz.txt nalaze se pozitivni celi brojevi manji od 200. Broj redova u datoteci kao i broj brojeva u svakom redu nije unapred poznat. Napisati program na programskom jeziku Pascal koji iz ove datoteke čita brojeve i u izlaznu tekst datoteku izlaz.txt prepiše svaki pročitan broj uz očuvanje rasporeda brojeva po redovima, uz ograničenje da se broj ne prepisuje ako se prethodno pojavio u datom redu. Drugim rečima, u jednom redu izlazne datoteke svi brojevi treba da budu različiti. Za pamćenje unetih brojeva koristiti skupovni tip podataka.

(po šablonu)

```
PROGRAM zad2_13_02_2009 (ulaz, izlaz);

VAR ulaz, izlaz: text;
    broj: integer;
    red: set of 1..200;

BEGIN
   assign(ulaz, 'ulaz.txt');
   reset(ulaz);
   assign(izlaz, 'izlaz.txt');
   rewrite(izlaz);

   while not eof(ulaz) do begin

      red:=[];
      while not eoln(ulaz) do begin {dok se ne dođe do kraja reda}
         read(ulaz, broj);
         IF NOT (broj IN red) THEN {ako se broj nije pojavio}
            write(izlaz, broj);
         red:= red+[broj]; {dodajemo broj u skup onih koji su se
                            pojavili}
      end;
      readln(ulaz); {kada se dođe do kraja reda prelazi se
                     u novi red da bi se nastavilo čitanje}
      writeln(izlaz); {a tokdje se pisanje prebaci u novi red}
   end;

   close(ulaz);
   close(izlaz);
END.
```

Zadatak 1 (07.03.2005)
Data je tekstualna datoteka ispit.txt koja je formirana na osnovu ocena studenata na predmetu Programiranje 1. Svaki red u datoteci sadrži podatke za jednog studenta, i to: prezime i ime (svako od polja je niz od najviše 40 karaktera), podatak da li se ispit polaže integralno ili preko kolokvijuma (jedan karakter: 'i' odnosno 'k' respektivno) i ocena na ispitu (ceo broj od 5 do 10). Podaci (prezime, ime, način polaganja i ocena) su međusobno razmaknuti sa jednim ili više blanko znakova. Potrebno je izračunati statistiku prolaznosti na ispitu i na glavnom izlazu ispisati ukupan broj studenata, broj i procenat studenata koji su položili ispit (ocena veća od 5) i broj i procenat studenata koji su ostvarili ocenu 10. Napisati program na programskom jeziku Pascal koji obavlja opisanu obradu.

Primer izgleda datoteke ispit.txt:
Petrovic Petar k 8
Markovic Marko i 7

(po šablonu)

```
PROGRAM zad1_7_3_2005 (ispit, output);

TYPE student = RECORD
         ime, prezime: string[40];
         nacinPolaganja: char;
         ocena: 5..10;
     END;

{procedura za citanje jedne reci, opisana gore}
PROCEDURE citajRec(VAR dat:TEXT; VAR rec:STRING);
VAR c:CHAR;
BEGIN
    rec := '';
    read(dat,c);
    while (c <> ' ') do begin
         rec := rec + c;
         read(dat,c);
    end;
END;

VAR ispit: text;
    s: student;
    ukupno, polozili, deset: integer;

za sve datoteke
BEGIN
   assign(ispit, 'ispit.txt');
   reset(ispit);

   {postavljamo bojace na 0}
   ukupno:= 0;
   polozili:= 0;
   deset:= 0;
```

```
   while not eof(ispit) do begin
      citajRec(ispit, s.prezime);
      citajRec(ispit, s.ime);
      readln(ispit, s.nacinPolaganja, s.ocena);

      ukupno:= ukupno + 1;
      if s.ocena > 5 then polozili:= polozili + 1;
      if s.ocena = 10 then deset:= deset + 1;
   end;

   writeln('Statistika');
   writeln('---------------------');
   writeln('Ukupno: ', ukupno);
   writeln('Polozioli: ', polozili, ', ', polozili/ukupno, '%');
   writeln('Desetke: ', deset, ', ', deset/ukupno, '%');
   writeln('---------------------');

   close(ispit);
END.
```

Zadatak 1 (09.06.2011)
Neka se u datoteci biblio.txt nalaze podaci o knjigama koje poseduje neka biblioteka. Svaki red sadrži šifru knjige (ceo broj), broj raspoloživih kopija (ceo broj) i naslov knjige do kraja reda (ne duži od 255 karaktera). U datoteci izdato.txt se nalaze podaci o knjigama koje su izdate na čitanje po sledećem formatu: šifra knjige (ceo broj) i šifra korisnika (ceo broj) koji je pozajmio knjigu na čitanje. Napisati program na programskom jeziku Pascal koji pročita sadržaj navedenih datoteka i na glavnom izlazu ispiše naslove svih onih knjiga čije su sve kopije izdate na čitanje korisnicima. Voditi računa o ispravnom korišćenju zauzetih resursa.

```
PROGRAM zad1_9_6_2011(biblio, izdat, output);

TYPE knjiga = RECORD
         sifra, brojKopija: integer;
         naslov: string[255];
     END;

     pelem = ^elem;
     elem = RECORD
         k: knjiga;
         sled: pelem;
     END;


PROCEDURE ucitaj(VAR dat: text; VAR lista:pelem);
VAR k:knjiga;
    novi,posl:pelem;
BEGIN
   WHILE NOT eof(dat) DO BEGIN
      readln(dat, k.sifra, k.brojKopija, k.naslov);
      new(novi);
      novi^.k:= k;
      novi^.sled:= NIL;
      IF lista = NIL THEN
         lista:= novi
      ELSE
         posl^.sled:= novi;
      posl:= novi;
   END;
END;

PROCEDURE obrisi(VAR lista:pelem);
VAR stari,tek:pelem;
BEGIN
   tek:=lista;
   WHILE tek <> NIL DO BEGIN
      stari:= tek;
      tek:= tek^.sled;
      dispose(stari);
   END;
   lista:= NIL;
END.
```

```
PROCEDURE izdaj(lista:pelem; sifraKnjige:integer);
VAR tek:pelem;
BEGIN
   tek:= lista;
   WHILE tek <> NIL DO BEGIN
      IF tek^.k.sifra = sifraKnjige THEN
         {smanjujemo broj kopija, ako jos uvek ima kopija}
         IF tek^.k.brojKopija > 0 THEN
            tek^.k.brojKopija:= tek^.k.brojKopija - 1;

      tek:= tek^.sled;
   END;
END;

PROCEDURE ispisiIzdate(lista:pelem);
VAR tek:pelem;
BEGIN
   tek:= lista;
   WHILE tek <> NIL DO BEGIN
      IF tek^.k.brojKopija = 0 THEN
         writeln(output, tek^.k.naziv);
      tek:= tek^.sled;
   END;
END;

VAR knjige: pelem; {lista knjiga}
    biblio, izdat: text;
    sifraKnjige: integer;

BEGIN
   assign(biblio, 'biblio.txt');
   reset(biblio);

   assign(izdato, 'izdato.txt');
   reset(izdato);

   knjige:= NIL;
   ucitaj(biblio, knjige);

   WHILE NOT eof(izdato) DO BEGIN
      readln(izdato, sifraKnjige); {iako u ovoj datoteci pored
                                    sifre knjige postoji i
                                    sifra korisnika ona mi nije
                                    potrebna pa je necu ni citati
                                    vec ce readln automastki
                                    da predje u novi red}
      izdaj(knjige, sifraKnjige);
   END;

   {ispisujemo knjige kod kojih je stanje nakon azuriranja 0}
   ispisiIzdate(knjige);
```

```
   obrisi(knjige);

   close(biblio);
   close(izdato);
END.
```

Zadatak 1 (09.02.2011)
Napisati program na programskom jeziku Pascal koji vrši raspoređivanje studenata po salama za polaganje kolokvijuma. U jednom redu datoteke studenti.txt se nalazi broj indeksa (u formatu gggg/bbbb) i ime i prezime studenta, a u jednom redu datoteke sale.txt se nalaze podaci o nazivu slobodne sale (niz znakova od najviše 10 karaktera) i kapacitetu (ceo broj). Program treba da formira izlaznu datoteku raspored.txt u čijem će se svakom redu nalaziti indeks, ime i prezime studenta i broj sale u kojoj polaže kolokvijum. Ukoliko zbog nedovoljnog kapaciteta program ne može da rasporedi studente ni u jednu salu, ispisati u izlaznoj datoteci da su neraspoređeni (npr. 2009/0161 Laza Lazic nerasporedjen).

studenti.txt

2007/0156 Pera Petrovic
2008/0158 Mile Milic
2009/0159 Aca Acic

sale.txt

M208 2
M216 3

raspored.txt

2007/0156 Pera Petrovic M208
2008/0158 Mile Milic M208
2009/0159 Aca Acic M216

```
PROGRAM zad1_9_2_2011 (studenti, sale, raspored);

TYPE pod_student = RECORD
         indeks: string[9];
         ime, prezime: string[30];
     END;

     pod_sala = RECORD
         sifra: string[10];
         mesta: integer;
     END;

PROCEDURE citajRec(VAR dat:TEXT; VAR rec:STRING);
VAR c:CHAR;
BEGIN
    rec := '';
    read(dat,c);
    while (c <> ' ') do begin
         rec := rec + c;
         read(dat,c);
    end;
END;
```

```
VAR studenti, sale, raspored: text;
    student: pod_student;
    sala: pod_sala;

BEGIN
   assign(studenti, 'studenti.txt');
   reset(studenti);

   assign(sale, 'sale.txt');
   reset(sale);

   assign(raspored, 'raspored.txt');
   rewrite(raspored);

   WHILE NOT eof(sale) DO BEGIN
      citajRec(sale, sala.sifra);
      readln(sale, sala.mesta);

      WHILE (sala.mesta > 0) AND (NOT eof(studenti)) DO BEGIN
         read(studenti, student.indeks);
         citajRec(studenti, student.ime); {posto je rec u sredini}
         readln(studenti, student.prezime);
         writeln(raspored, student.indeks, student.ime,
                 student.prezime, sala.sifra);
         sala.mesta:= sala.mesta+1;
      END;
   END;

   {ako je ostalo jos studenata}
   WHILE NOT eof(studenti) DO BEGIN
      read(studenti, student.indeks);
      citajRec(studenti, student.ime); {posto je rec u sredini}
      readln(studenti, student.prezime);

      writeln(raspored, student.indeks, student.ime,
              student.prezime, 'nerasporedjen');
   END;

   close(studenti);
   close(sale);
   close(raspored);

END.
```