Рачунарство и информатика

Јелица Васиљевић

Одељење II см

Школска 2015/2016

Датотеке

- Проблеми:
  - Након извршавања програма, резултати се трајно губе
  - Уношење истог улаза више пута је мукотрпно, поготво ако је улаз велики
  - Повезивање програма са улазно/излазним уређајима – нпр. диск
- Решење :
  - Коришћење датотека (енг. Files).

Датотеке

- Датотека представља **уређен скуп** произвољног броја компоненти **истог типа.** Број компоненти, теоријски, није ограничен.
- Користе се за трајно складиштење улаза/излаза програма на диск (или неки други медијум)

Датотеке - врсте

- Типизиране
- Нетипизиране
- Текстуалне

Датотеке – Типизиране - Увод

- Експилицитно је дефинисан **тип** податка који се чува у фајлу
- Декларација :

```
var
  datoteka : file of <tip podataka>
```

- Тип податка може бити **било који тип** (прост или сложен)
  - Једини изузетак је да тип података не може бити **датотечни** тип или тип који садржи датотечну компоненту.
- Датотечне променљиве су **нетипичне** тј. **не могу** се користити у наредбама доделе.
- Датотечне променљиве је најбоље посматрати као **датотеку** којој се приступа преко имена датотечне променљиве

Датотеке – Типизиране - Увод

- Крај датотеке
  - Крај датотеке се обележава невидљивим знаком за крај – **EOF**
  - За детектовање да ли се стигло до краја датотеке користи се логичка функција (враћа тачно или нетачно) **eof(f).**
- **Показивач**
  - Приликом рада са датотекама, **свакој** датотеци се придружује **показивач** који **показује** на место (податак) у фајлу који се следећи обрађује. Ова акција је аутоматска.

Датотеке – Типизиране – Повезивање

- Датотечна променљива се **повезује** са конкретном датотеком наредбом **ASSIGN.**

```
assign(dat,'mojaDatoteka.dat');
```

- Након повезивања, кроз програм се комуникација са фајлом *mojaDatoteka.dat* врши преко датотечне променљиве *dat*

Датотеке – Типизиране – Отварање за упис

- Датотека која је **повезана** са датотечном променљивом *dat* се отвара за **упис** наредбом **rewrite** и то на следећи начин :

```
rewrite(dat);
```

- Резултат извршавања ове наредбе је следећи
  - Уколико датотека повезана са променљивом *dat* **не постоји**, креира се нова датотека под тим именом и датотечни показивач се налази на почетку фајла
  - Уколико датотека повезана са променљивом *dat* **постоји,** њен садржај се **БРИШЕ** а датотечни показивач се поставља на прву позицију у том фајлу.

Датотеке – Типизиране – Упис

- Након повезивања датотечне променљиве *dat* са одговарајућом датотеком и након отварања за **упис,** упис у датотеку се врши стандардном командом за испис **write**, и то на следећи начин

```
write(dat,podatak);
```

Променљива *podatak* **мора** бити **ИСТОГ** типа као и подаци у датотеци

- Подаци који се на овај начин уписују у датотеку, **не уписују се** одмах на диск (трајну меморију). Због тога што је комуникација са диском изузетно временски „скупа“ операција, ови подаци се најпре смештају у **бафер –** привремену меморију ограниченог капацитета**.** Тек када се бафер напуни, сви подаци из њега се истовремено уписују на диск.
- Резултат извршавања ове наредбе је :
  - Вредност променљиве *podatak* је уписана у датотеку на позицији на коју показује датотечни показивач
  - Датотечни показивач се помера за једно место

Датотеке – Типизиране – Затварање датотеке

- Након завршетка рада са датотеком, неопходно ју је **затворити.** То се постиже наредбом **close** :

```
close(dat);
```

- Затварање датотеке је **НЕОПХОДНО** када је датотека отворена за УПИС јер се овом наредбом аутоматски садржај бафера преписује на диск

Датотеке – Типизиране – Отварање за читање

- Датотека која је **повезана** са датотечном променљивом *dat* се отвара за **читање** наредбом **reset** и то на следећи начин :

```
reset(dat);
```

- Резултат извршавања ове наредбе је следећи
  - Уколико датотека повезана са променљивом *dat* **не постоји**, јавља се грешка у току извршавања програма – *run time error*
  - Уколико датотека повезана са променљивом *dat* **постоји,** датотечни показивач се позиционира на прву компоненту датотеке.
  - Приступ датотеци је остварен **у read/write** режину што подразумева да је поред читања, могуће и уписати податке у фајл. Ова опција је подразумевана за типизиране датотеке и може се променити подешавањем специјалних глобалних променљивих окружења

Датотеке – Типизиране – Читање

- Након повезивања датотечне променљиве *dat* са одговарајућом датотеком и након отварања за **читање**, читање из датотеке се врши стандардном командом **read**, и то на следећи начин

```
read(dat,podatak);
```

Променљива *podatak* **мора** бити **ИСТОГ** типа као и подаци у датотеци

- Резултат извршавања ове наредбе је следећи :
  - У променљиву *podatak* је смештана компонента на коју показује датотечни показивач
  - Датотечни показивач је померен на следећу компоненту - следеће место у датототеци.
- Након завршетка рада са датотеком, потребно ју је затворити.

Типизиране датотеке - пример

- Креирати датотеку типа Integer и у њу уписати све бројеве од 1 до 100. Након тога прочитати податке из те датотеке и исписати их на стандардни излаз

```
program OsnovniPrimer;
var
  dat:file of integer;
  i:integer;
begin
// Upis
  assign(dat,'mojaDat.dat');
  rewrite(dat);
  for i:=1 to 100 do
   write(dat,i);
  close(dat);

// Citanje
  reset(dat);
  while not(eof(dat)) do
   begin
     read(dat,i);
     write(i,' ');
   end;
 close(dat);
end.
```

Типизиране датотеке – директан приступ

- Неке функције за директан приступ типизираним датотекама :
  - Seek(f,5) – помера датотечни показивач на 5. компоненту (позицију)
  - Filesize(f) – враћа број комоненти у датотеци
  - Filepos(f) – враћа тренутну позицију датотечног показивача
  - Truncate(f) – брише садржај датотеке од текуће позиције
  - Erase(f) – брише датотеку са диска – датотека мора бити **затворена** да би се ова наредба извршила
  - Rename(f,novolme) – мења назив (физички, на диску) датототеке која је повезана са датотечним показивачем у *novolme*

Компајлерска директива $I

- Проблем:
  - При повезивању са датотеком, датотека се референцира преко свог имена. Међутим, таква датотека не мора обавезно постојати на диску. У случају да се непостојећа датотека отвара за читање, доћи ће до грешке. Исто тако, уколико се за упис отвара постојећа датотека, доћи ће до брисања садржаја датотеке
  - Како избећи да програм „пукне“ уколико се отвара непостојећа датотека и како обезбедити **сигуран** упис ?
- Решење : Компајлерска директива $I или $IOCHECKS

Компајлерска директива $I

- При компајлирању, компајлер свакој улазно/излазној наредби придодаје код који контролише улазно/излазне операције. Захваљујући том коду, програм се прекида у случају било какве улазно/излазне грешке
- {$I-} – онемогућава генерисање кода за проверу У/И операција за све У/И наредбе **испод.**
- {$I+} – омогућава генерисање кода за проверу У/И операција за све У/И наредбе **испод.**
- **Ове директиве треба користи само у случајевима када је познато која грешка може да се деси и која ће акција бити предузета!**

Компајлерска директива $I и функција IOResult

- Укључивање или искључивање контроле У/И грешака неће спречити да се грешка **не појави**. У случају да је аутоматска провера грешака укључена, појава грешке резултоваће **прекидом** програма. Уколико се искључи провера У/И грешака, на програмеру је да води рачуна о ситуацијама када се оне десе.
- Функција IOResult враћа информацију о томе да ли се **последња** У/И наредба извршила коректно или не. У случају да се наредба успешно завршила, враћа нулу. У супротном код грешке

Компајлерска директива $I и функција IOResult

- Написати програм који спречава читање из непостојеће датотеке и обевештава корисника о томе да датотека не постоји.

```
program ioResultOsnovni;
var
d1:file of integer;
x:integer;
begin
assign(d1,'nepostojeca.txt');
// iskljucena provera i/o gresaka
{$I-}
reset(d1);    // u/i naredba koja moze da prouzrokuje gresku
{$I+}   // ukljucivanje automatske provere gresaka za sve ostale naredbe
x:=IOResult;
if (x=0) then writeln ('Sve je proslo bez greske. Datoteka postoji')
else writeln('Desila se greska. Verovatno datoteka ne postoji');

close(d1);
readln;
end.
```