Рачунарство и информатика

Јелица Васиљевић

Одељење II см

Школска 2015/2016

Датотеке – Текстуалне датотеке

- Поред бинарних (типизоираних) датотека, Pascal омогућава и рад са посебном врстом датотека које се називају **текстуалне**
  - Под *текстом* се интуитивно подразумева **низ линија** састављених од **знакова.**
  - У Pascal-у, под тектуалном датотеком, се подразумева **фајл** у коме се налази **текст (** читљив, разумљив) састављен од нула или више линија.

Датотеке – Текстуалне датотеке

- Свака линија се завршава **специјалним** знаком **– *end-of-line (<EOLN>).***
  - Овај знак нема графичку интерпретацију.
  - Састављен је од два, графички невидљива знака
    - Carrage return – ASCII код 13
    - Line feed – ASCII код 10
- Крај **сваке** датотеке, па и текстулане, означен је специјалним знаком **end-of-file (<EOF>).** ASCII код овог знака је 26.

Датотеке – Текстуалне датотеке

- Декларација текстуланих датотека

```
var
    tekstDat : text;
```

- Повезивање датотечне променљиве **tekstDat** са конкретном датотеком, врши се наредбом **ASSIGN.** Синтакса наредбе је иста као код типизираних датотека.

Датотеке – Текстуалне датотеке

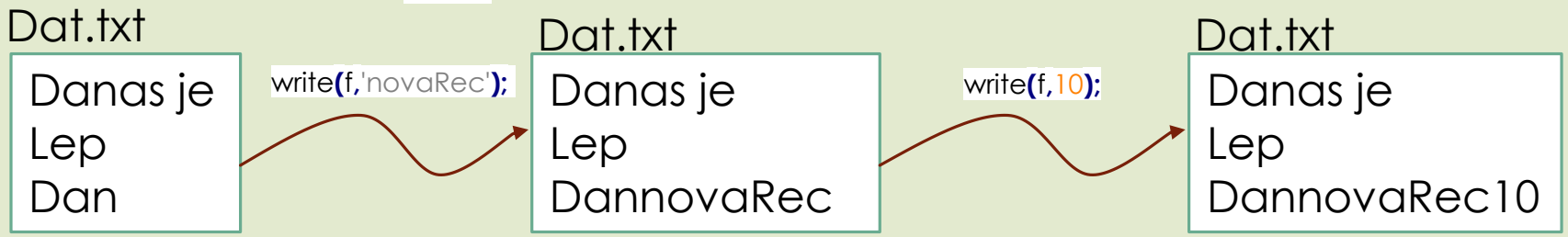
- Отварање за **читање**
  - Отврање текстуалне датотеке за **читање** врши се наредбом **Reset.** Синтакса наредбе је иста као код типизираних датотека
    - Датотека се отвара за читање у **read-only** режиму. То значи да је дозвољено **САМО ЧИТАЊЕ** из датотеке
    - Уколико датотека не постоји, појавиће се грешка у току извршавања програма (run-time error)
- Отварање датотеке за **упис**
  - Отврање тектуалне датотеке за **упис** врши се наредбом **Rewrite.** Синтакса наредбе је иста као код типизираних датотека
    - Датотека се отвара за читање у **write-only** режиму. То значи да је дозвољено **САМО УПИСИВАЊЕ** у датотеку
    - Уколико датотека постоји, њен садржај ће бити обрисан
    - Уколико датотека не постоји, креитраће се нова.

Датотеке – Текстуалне датотеке

- Отварање за **додавање**
  - Отврање текстуалне датотеке за **додавање** врши се наредбом **Append.**
    - Синтакса наредбе је

      ```
      Append(f);
      ```
    - Датотека се отвара у **write-only** режиму
    - Додавање у датотеку се врши наредбом **Write**

      ```
      write(f,'novaRec');
      write(f,10);
      ```
    - Додавање текста се врши на **КРАЈ** датотеке.



Датотеке – Текстуалне датотеке стандардни улаз и излаз

- Тастатура и екран третирају се као текстуалне датотеке које није потребно експлицитно повезивати и отварати
  - Тастатура шаље знакове организоване у линије
  - Екран чита знакове организоване у линије
- Уколико се у наредбама *read (readln)* или *write(writeln)* **не наведе** име датотечне променљиве, подразумева се да се ради о **стандардном улазу/излазу**.

Датотеке – Текстуалне датотеке - предности

- Наредбе читања и уписивања (Read,Write) имају посебну **форму** када се ради са текстуланим датотекама
  - Могуће је уписати/прочитати вреднсти које нису знаковне - **Char** типа.
  - Читањем или уписивањем променљивих које нису знаковне врши се **аутоматско** превођење типова
  - Операција уписа/исписа понашају се исто као да се ради о стандардном излазу/улазу.

Датотеке – Текстуалне датотеке – разлика у односу на *file of char*

- F: text

1. Читање и упис променљивих различитих типова – Real, Integer, String ...
2. Уписивање и читање коришћењем Readln I Writeln
3. Детекција краја реда и краја датотеке
4. Reset - отвара датотеку у read-only режиму
5. Rewrite – отвара датотетку write-only режиму
6. Append – отвара датотеку у write-only режиму

- F: file of char

1. Читање и упис променљивих типа Char
2. Није дозвољено коришћење Readln I Writeln – зашто ?
3. Детекција краја датотеке
4. Reset - отвара датотеку у read-write режиму
5. Rewrite – отвара датотетку read-write режиму
6. Не постоји

Датотеке – Текстуалне датотеке – разлика и сличности у односу на *file of char*

- F: text
  7. Испитивање да ли се дошло до краја реда омогућено је логичком функцијом **Eoln**
  8. Испитивање да ли се дошло до краја датотеке омогућено је функцијом **Eof**
  9. Није дозвољено коришћење рутина FilePos,Seek,FileSize,Truncate – зашто ?
  10. Дозвољено је Erase, Rename

- F: file of char
  7. Није могуће испитати да ли се стигло до краја реда – зашто ?
  8. Испитивање да ли се дошло до краја датотеке омогућено је функцијом **Eof**
  9. Дозвољено је коришћење рутина FilePos,Seek,FileSize,Truncate
  10. Дозвољено је Erase, Rename

Датотеке – Текстуалне датотеке-пример

- Написати програм исписује колико има редова, знакова и празних линија у датотеци prva.txt

```
program osnovniPrimer;
var
f:Text;
brLinija,brZnakova,brPraznihLinija,uLiniji :integer;
c:char;
begin

Assign(f,'prva.txt');
Reset(f);
brLinija:=0;
brPraznihLinija:=0;
brZnakova:=0;
while not(Eof(f)) do
begin
 if Eoln(f) then
        begin
        if (uLiniji=0) then Inc(brPraznihLinija);
        uLiniji:=0;
        Inc(brLinija);
        Readln(f); // prelazak u novu liniju datoteke
        end
 else
   begin
   read(f,c);
   Inc(brZnakova);
   Inc(uLiniji);
   end;
end;
writeln('Broj linija ', brLinija);
writeln('Broj praznih linija ', brPraznihLinija);
writeln('Broj znakova ', brZnakova);


end.
```

Датотеке – Нетипизиране датотеке

- Поред текстуланих и типизираних датотека, у програмском језику Pascal могуће је радити и са **нетипизираним датотекама**
  - **Бинарана датотека –** немогуће је њен садржај видети у текстуалном едитору
  - Бржи приступ него код типизираних
  - Погодни за коришћење у програмима који интензивно уписују/читају податке
  - Третирају се као уређен скуп **бајтова**
  - Тип података који су уписани у фајл **није битан**.
  - Јединица преноса је **бајт**
  - Специфицира се количина података која се преносе у једном захтеву за читање/упис.

Датотеке – Нетипизиране датотеке

- Најчешће се употребљавају када тип података у фајлу није важан (нпр. код копирања)
- Приступ овим фајловима је знатно бржи у односу на приступ текстуланим или типизираним
  - Директан приступ (low-level)
  - Нема превођења типова

Датотеке – Нетипизиране датотеке

- Декларисање нетипизиране датотеке

```
var
netipizirana : file;
```

- Нетипизирана датотека посматра се као уређен скуп елемената **било ког типа**, али познате величине
  - Може се уписати било који податак **дефинисане** величине

Датотеке – Нетипизиране датотеке

- Пре отварања нетипизиране датотеке за читање/упис, она се мора повезати са физичком датотеком на диску наредбом ASSIGN. Синтакса наредбе је иста као код текстуалних датотека.
- Отварање нетипизиране датотеке за упис/читање врши се стандардним наредбаа *Reset* и *Rewrite, са* ***додатним*** *параметром* – ***величина елемената датотеке изражена у бајтовима.***
  - Овај додатни параметар користи се за одређивање количина података која ће бити пренета у сваком захтеву за читање/упис

Датотеке – Нетипизиране датотеке

- За **упис/испис** у/из нетипизиране датотеке **НЕ МОЖЕ** се вршити стандардним наредбама *Read* и *Write*
- За упис у нетипизирану датотеку користи се наредба **BlockWrite**
  - **Синтакса наредбе BlockWrite је :**

Бафер из ког се уписује

```
procedure BlockWrite(
  var f: file;
  const Buf;
  Count: LongInt
);
```

или

```
procedure BlockWrite(
  var f: file;
  const Buf;
  Count: LongInt;
  var Result: LongInt
);
```

или

Колика количина података се уписује у датотеку

Колико је података заиста уписано

http://www.freepascal.org/docs-html/rtl/system/blockwrite.html

Датотеке – Нетипизиране датотеке

- За читање из нетипизиране датотеке користи се наредба **BlockRead**
  - **Синтакса наредбе BlockRead је :**

Бафер у који се уписује

```
procedure BlockRead(
  var f: file;
  var Buf;
  count: Int64
);
```

или

```
procedure BlockRead(
  var f: file;
  var Buf;
  count: LongInt;
  var Result: LongInt
);
```

или

Колика количина података се чита из датотеке

Колико је података заиста прочитано

http://www.freepascal.org/docs-html/rtl/system/blockread.html

Датотеке – Нетипизиране датотеке

- **Написати програм који креира датотеку типа integer, у њу уписати бројеве од 1 до 100. Затим датотеку прочитати као нетипизирану и преписати њен садржај у другу датотеку типа integer**

```
program prepisiIntDatoteku;
var
  f,g: file of integer;
  pom:file;
bafer:integer;
i,ukupno:integer;
begin
assign(f,'prava.dat');
rewrite(f);
for  i:=1 to 100 do write(f,i);
close(f);

assign(pom,'prava.dat');
assign(g,'pomocna.dat');
reset(pom,1);
rewrite(g,1);

while not(eof(pom)) do
     begin
     { Velicina tipa integer moze da bude 2 ili 4 bajta.
      Kako bi radilo na svim implementacijama,
     najbolje je da se stavi sizeof(integer) umesto
     konkretnog broja.

      Funkcija sizeof vraca velicinu tipa na
      trenutnoj platform }
       BlockRead(pom,bafer,sizeof(integer));
       write(g,bafer);
     end;
close(pom);
close(g);

assign(f,'pomocna.dat');
reset(f);
while not(eof(f)) do
 begin
 read(f,bafer);
 writeln(bafer);
 end;
close(f);
writeln('Velicina integer tipa na je ', sizeof(integer));
readln;
end.
```