Esercitazioni sui File in Pascal

Nicola Fanizzi - DIB
Corso (B) di Programmazione
CdL in Informatica – I livello
A.A. 2003/2004

Problema 1
Progettazione Strutture Dati

- tipo data = triple (giorno,mese,anno)
  - giorno: intero da 1 a 31
    (dipende da mese e anno secondo la regola)
  - mese: intero da 1 a 12
  - anno: 2000-massimo_intero
- tipo quantitativo = **N** *(numeri naturali)*
- le coppie saranno quindi di tipo
  tipo_data x tipo_quantitativo:
  ((giorno, mese, anno), quantitativo)
- dato che il numero di coppie non é noto useremo la sequenza di queste coppie

Problema 1

Progettare un programma che:
- data una sequenza di coppie (data, quantitativo)
- gestisca l'immissione della sequenza di coppie
- fornisca:
  - quantitativo minimo e la data relativa
  - quantitativo massimo e la data relativa
  - quantitativo medio
- conti quanti giorni sono intercorsi tra la data relativa al minimo e quella relativa al massimo
- stampi un rapporto sui risultati precedenti

Problema 1
Progettazione Strategia Solutiva

La soluzione top-down discende dalla formulazione del problema
1. Immettere sequenza
2. Scandire la sequenza con
   1. memorizzazione del minimo, massimo corrente e delle date relative
   2. conteggio delle coppie e della somma parziale dei quantitativi
3. Calcolare la distanza in giorni tra le date trovate al passo 2.1
4. Stampare rapporto

Problema 1
Analisi

- Supponiamo che la data sia composta da tre parti: (giorno, mese, anno) dove:
  - giorno: da 1 a 31
    (dipende da mese e anno secondo la regola)
  - mese: da 1 a 12
  - anno: supponiamo successivo all'anno 2000
- il significato del minimo e del massimo dipende dall'ordine normale sugli interi
- l'ordine sulle date é quello consueto
- il quantitativo medio viene calcolato da: $\frac{1}{N}\sum_{i=1}^{N} q_i$ dove N non é noto a priori
- anche la differenza non può essere negativa

Problema 1
Progetto Algoritmo 1

1. algoritmo di immissione
input: -
output: sequenza S di coppie
begin
Inizializzare la sequenza S
repeat
  Leggere data valida
  Leggere quantitativo valido
  Inserire (data,quantitativo) in S
until fine immissione
Restituire sequenza S
end

Problema 1
Progetto Algoritmo 2

2. algoritmo di scansione della sequenza
input: sequenza S di coppie
output: min, max, med, data-min, data-max
begin
inizializzare contatori e accumulatori
while ci sono ancora coppie in S **do**
    Leggere prossima coppia
    Aggiornare min, max e le date relative
    Incrementare il contatore delle coppie
    Aggiornare la somma parziale dei quantitativi
Calcolare med usando la somma quantitativi e il
    contatore delle coppie
end

Problema 1
Progetto Algoritmo 3

3. algoritmo calcolo delle differenza date
input: data1 = (giorno1,mese1,anno1)
        data2 = (giorno2,mese2,anno2)
output: differenza (di tipo intero)
begin
d1 := giorni da 1/1/2000 a data1
d2 := giorni da 1/1/2000 a data2
differenza := d1-d2
end

Problema 1
Progetto Algoritmo 3.1

- Calcolo giorni dal 1.1.2000 fino ad una data

input:data = (giorno,mese,anno)
output: ngiorni (intero)
begin
ngiorni := giorno;
for m :=1 **to** mese-1 **do**
    Incrementare ngiorni del numero di giorni del
    mese m dell'anno dato
for a := 2000 **to** anno-1 **do**
    if bisestile **then** ngiorni := ngiorni + 366
                else ngiorni := ngiorni + 365
Restituire ngiorni;
end

Problema 1
Progetto Algoritmo 3.2

- Funzione anno bisestile

input: anno
output: vero/falso
begin
if (anno divisibile per 4) e
        (tranne (divisibile per 100)
            (tranne (divisibile per 400)))
then Ritornare vero
else Ritornare falso
end

Problema 1
Progetto Algoritmo 4

- stampa rapporto

input: min, max, medio, differenza
output: -
begin
stampa sulla periferica standard di output
    min
    max
    medio
    differenza
end

Problema 1
Codifica

- Esercizio in laboratorio
- Usare file di record opportunamente dichiarati secondo i passi precedenti
- implementare gli algoritmi scomponendoli ulteriormente, se necessario
  - procedure immissione()
  - procedure calcolo()
  - function differenza()
  - procedure stampa_rapporto()
  - …
- Decidere:
  quali sono i parametri e **come** passarli

Problema 1
Codifica: Tipi

```
type
    TQuantitativo = 0..maxint;

    TData = record
        giorno, mese, anno: integer
    end; (* record TData *)

    TCoppia = record
        data: TData;
        quant: TQuantitativo
    end; (* record TCoppia *)

    TSequenza = File of TCoppia;
```

Problema 1
Codifica Sottoprogrammi (cont.)

```
function data_corretta(d:TData):boolean;
begin
if d.anno < 2000 then data_corretta:=false;
if ((d.mese in [4,6,9,11]) and (d.giorno in [1..30])) or
   ((d.mese<>2) and (d.giorno in [1..31]))
then data_corretta:= true
else
   if bisestile(d.anno)
   then data_corretta := d.giorno in [1..29]
   else data_corretta := d.giorno in [1..28]
end;
```

Problema 1
Codifica Sottoprogrammi

```
procedure inserimento(var F:TSequenza);
var c: Tcoppia;
begin
rewrite(F);
repeat
    writeln('immettere una coppia (data, quantitativo)');
    writeln('quantitativo negativo per terminare');
    leggi_quantitativo(c.quant);
    leggi_data(c.data);
    if c.quant >= 0 then write(F,c)
until (c.quant<0);
end;
```

Problema 1
Codifica Sottoprogrammi (cont.)

```
function bisestile (anno:integer): boolean;
begin
bisestile := (anno mod 4 = 0) and
    ((anno mod 100 <> 0) or (anno mod 400 = 0));
end;

function dal2000 (d:Tdata): integer;
var g,m,a:integer;
begin
g := d.giorno;
for m :=1 to d.mese-1 do
    g:=g+giorni_mese(m,d.anno);
for a :=2000 to d.anno-1 do
    if bisestile(a) then g:=g+366 else g:=g+365;
return g;
end;
```

Problema 1
Codifica Sottoprogrammi (cont.)

```
procedure leggi_data(var d:TData);
const max_ripetizioni =10;
var i:integer;
begin
i:=1;
repeat
    writeln('immettere una data');
    read(d.giorno, d.mese, d.anno);
    i := i+1;
until data_corretta(d) or (i > max_ripetizioni);
if (i > max_ripetizioni) then
    begin
    writeln('troppe ripetizioni! Imposto la data a 1/1/2000');
    d.giorno :=1; d.mese:=1; d.anno:=2000;
    end
end;
```

Problema 1
Codifica Sottoprogrammi (cont.)

```
function bisestile (anno:integer): boolean;
begin
bisestile := (anno mod 4 = 0) and
    ((anno mod 100 <> 0) or (anno mod 400 = 0));
end;

function giorni_mese(mese,anno:integer):integer;
begin
if mese in [4,6,9,11] then return 30;
if (mese<>2) then return 31
else
    if bisestile(anno) then return 29
                       else return 28
end;
```

Problema 1
Test Empirico

Classi di equivalenza:

- procedura immissione()
  - [c.quant non negativo], [c.quant negativo]
  - [file F non accessibile], [file F accessibile]
- per ognuna delle sotto-procedure/funzioni cercare le classi di equivalenza
  - Procedura leggi_data
    - [data corretta], [data non corretta]
    - [ripetizioni<10], [ripetizioni=10], [ripetizioni>10]
  - Funzione data_corretta
    - [corretta generica], [non corretta/mese], [non corretta/giorno], [non corretta/anno non bisestile], [non corretta/anno bisestile]
  - Ecc…

Problema 2
Algoritmo di massima

input: File1
output: File2
Inizializzare File1 in lettura
Inizializzare contatori per c_i le categorie a, e, i, o, u, consonanti e numeri
while non è finito **do**
  while non è finita la prossima linea **do**
    Leggere prossimo carattere
    Incrementare contatore corrispondente alla sua categoria
    Incrementare contatore totale caratteri TC
Inizializzare File2 per la scrittura
for categoria:= 1 **to** 8 **do**
  for i := 1 **to** (80*ci/TC) **do**
    Scrivere '*'
  Passare a nuova linea

Problema 1
Test Empirico

Classi di equivalenza:

- procedura calcolo()
  - file vuoto
  - file non vuoto
  - per ognuna delle sotto-procedure/funzioni cercare le classi di equivalenza
- ecc...

Problema 2
Codifica 1/3

```
program istogramma(input,output,f1,f2);
var
  f1,f2: text;
  ch: char;
  i,j,n: integer;
  c: array[1..7] of integer;
begin
assign(f1,'testo.txt');
reset(f1);
for i:= 1 to 7 do c[i]:=0;
n:=0;
```

Problema 2

Progettare un programma che

- dato un testo in memoria secondaria (prodotto con un editor di testi, es. notepad)
- fornisca:
  - il numero totale di caratteri alfanumerici del file
  - un diagramma a barre, su un nuovo file di testo, che riproduca le percentuali trovate
    - delle sei vocali a,e,i,o,u (1 barra cadauna)
    - delle consonanti (1 barra)
    - dei numeri (1 barra)

Problema 2
Codifica 2/3

```
while not eof(f1) do
    begin
    while not eoln(f1) do
        begin
        {ch:=f1^; get(f1);}
        read(f1,ch);
        n:=n+1;
        if ch in ['a','e','i','o','u'] then
           case ch of
           'a': c[1]:=C[1]+1;
           'e': c[2]:=C[2]+1;
           'i': c[3]:=C[3]+1;
           'o': c[4]:=C[4]+1;
           'u': c[5]:=C[5]+1
           end { case };
        else if ch in ['0'..'9']
             then c[7]:=C[7]+1
             else if ch in (['a'..'z']-['a','e','i','o','u'])
                  then c[6]:=C[6]+1
        end;
    readln(f1);
    end;
```

Problema 2
Codifica 3/3

```
assign(f2,'isto.txt');
rewrite(f2);
for i:= 1 to 7 do
    begin
    write(f2,'cat.',i,': ');
    for j:= 1 to trunc(80*c[i]/n) do write(f2,'*');
    writeln(f2,100*c[i]/n:5:2,'%')
    end;

end. (*programma*)
```

Problema 3

Produrre un programma che:

- Accetti in ingresso numeri rappresentati da cifre arabe e le scriva su file testo
- Legga da file di testo numeri in cifre arabe e scriva le corrispondenti cifre romane su un altro file

- (e se venisse richiesto il contrario ?)

Problema 3
Codifica 1/4

```
program arabi2romani
    (input,output,testo1,testo2);

uses conversioni;

Var     testo1, testo2:text;
        n:integer;


begin {main}
assign(testo1,'arabi.txt');
assign(testo2,'romani.txt');

writeln('Conversione numeri arabi / numeri romani');

arabi(testo1);
romani(testo1,testo2);
end.
```

Problema 3
Codifica 2/4

```
Unit Conversioni;
Interface;
procedure arabi(var f:text);
procedure romani(var ifile,ofile:text);

implementation
procedure arabi(var f:text);
var n:integer;
    risposta:char;
begin
rewrite(f);
repeat
      repeat
            write('immetti numero: ');
            readln(n)
      until (n>0) and (N<=5000);
      writeln(f,n);
      write('Fine ? (s/N) ');
      readln(risposta)
until risposta in ['s','S'];
end; {arabi}
```

Problema 3
Codifica 3/4

```
procedure romani(var ifile,ofile:text);
var n:integer;
begin
reset(ifile); rewrite(ofile);
while not eof(ifile) do
      begin
      while not eoln(ifile) do
            begin
            read(ifile,n);
            while n>=1000 do
              begin write(ofile,'M'); n:=n-1000 end;
            if n>=900 then
              begin write(ofile,'CM'); n:=n-900 end
            else if n>=500 then
                 begin write(ofile,'D'); n:=n-500 end;
             while n>=100 do
               begin write(ofile,'C'); n:=n-100 end;
```

Problema 3
Codifica 4/4

```
             if n>=90 then
                begin write(ofile,'XC'); n:=n-90 end
            else if n>=50 then
                 begin write(ofile,'L'); n:=n-50 end
                 else if n>=40 then
                      begin write(ofile,'XL'); n:=n-40 end;

            while n>=10 do
                 begin write(ofile,'X'); n:=n-10 end;

            if n>=9 then
              begin write(ofile,'IX'); n:=n-9 end
            else if n>=5 then
                 begin write(ofile,'V'); n:=n-5 end
                 else if n>=4 then begin write(ofile,'IV'); n:=n-4 end;

            while n>0 do
                 begin write(ofile,'I'); n:=n-1 end;
            end;
      readln(ifile);
      writeln(ofile)
      end
end; {romani}

end. {Unit}
```

Problema 4
Ricerca di una Parola Chiave

- Contare il numero di volte che una parola data appare in un testo
  - Non ci si curi del problema dell'immissione del testo
    - Si ipotizzi di digitare il testo in input dalla tastiera oppure
    - File preparato precedentemente con un editor
  - Esempio
    "Questo è un segmento di testo…."
    - Verificare quante volte appare la parola "un"

Problema 4
Nucleo della strategia

Metti a 1 l'indice *i* dell'array di parola
Mentre non è vera la condizione che segnala la fine del testo **esegui**
  Leggi il prossimo carattere
  Se il carattere è identico all'*i*-esimo carattere dell'array contenente la parola **allora**
    Estendi il confronto di 1
    Se vi è concordanza **allora**
      Verifica che l'intera parola concordi e prendi azioni opportune
  Altrimenti
    Prendi le azioni conseguenti un caso di discordanza

Problema 4
Considerazioni

- L'esame va fatto un carattere alla volta
- (Tranne la prima,) una parola in un testo può essere preceduta o seguita
  - Da un blank
  - Da caratteri speciali
- Due parole possono essere simili anche per una frazione di parola
  - La discordanza può apparire in qualunque momento, durante il processo di confronto
  - Esempio: un *un*a

Problema 4
Considerazioni

- Necessità di ricordare il carattere che precede uno spezzone di testo
  - N.B.: Un carattere su cui si è verificata una condizione di disaccordo *può* essere un carattere che precede una parola
    - Bisogna salvarlo
    - Sarà cambiato solo quando vi è una condizione di disaccordo (Non varierà durante il confronto)
- Quando si rileva un disaccordo
  - Salvare il carattere corrente nel testo come *pre*
  - Reimpostare l'indice di parola a 1
- Se vi è accordo completo
  - Incrementare il contatore di confronto di 1
  - Lasciare in *pre* il più recente
  - Reimpostare l'indice di parola a 1

Problema 4
Gestione del confronto

- Portare avanti la lettura del testo carattere per carattere
  - Il processo di lettura del testo continua fino alla fine del testo
  - Contemporaneamente avviene il confronto con la parola ricercata
    - Incremento dell'indice sulla parola
- Occorre:
  - Leggere un carattere del testo
  - Confrontarlo con la prima lettera della parola cercata
  - Se vi è accordo si legge il prossimo carattere e si confronta con la prossima lettera della parola per verificare se l'accordo continua
- Ogni volta che si stabilisce che la parola è completamente eguale allo spezzone di testo, va controllato se questo è una parola o una *substringa* di una parola nel testo
  - Si assume che una parola sia preceduta e seguita da un carattere non alfabetico
    1. Occorre ricordare il carattere che precede una parola nel testo
    2. Necessità di tener conto dei fine-linea nel testo

Problema 4
Algoritmo

Input: parola, testo
Output: trovato/non trovato
Stabilire la parola e la sua lunghezza
Inizializzare il contatore di concordanze *nmatches*, definire il carattere precedente e mettere a 1 l'indice di array della parola da cercare
Fintantoché ci sono caratteri nel file **esegui**
  Fintantoché la linea non è terminata **esegui**
    Leggi il prossimo carattere *chr*
    (vd. lucido successivo)
  Passa all'altra linea di testo
nmatches contiene il numero di concordanze

Problema 4
Algoritmo

Se *chr* coincide con l'*i*-esimo carattere nella parola **allora**
  Se l'intera parola concorda **allora**
    Leggi il prossimo *chr* nel testo *post*
    Se il carattere precedente ed il seguente non sono alfabetici **allora**
      Aggiorna *nmatches*
    Reimposta l'indice di parola *i*
    Salva il carattere *post* come prossimo precedente
Altrimenti
  Salva il prossimo carattere precedente
  Reimposta *i* a 1

Problema 4
Considerazioni

- L'azione fondamentale è il paragone tra il carattere corrente del testo ed il carattere corrente di parola
  - Viene effettuata un numero di volte uguale al numero di caratteri nel testo
- Con ogni iterazione è letto un altro carattere
  - Si raggiunge sicuramente la fine del file: L'algoritmo termina
- Allo stadio di ricerca in cui sono stati letti i primi *j* caratteri sono state contate tutte le parole nel testo che corrispondono alla parola ricercata
- È possibile realizzare algoritmi che ottimizzino la ricerca
  - In funzione della lunghezza della stringa da ricercare
    oppure
  - In funzione della particolare parola
    - Saltando alla parola successiva già al primo disaccordo