Lezione 13
11–12 novembre 1999

Argomenti trattati
- Ulteriori esempi sugli array.
- Tipi strutturati: i set.

13.1 Esempio: cambi di valute

Vogliamo costruire un programma che legga i valori di cambio rispetto alla Lira di alcune valute estere (Dollaro USA, Franco Francese, Sterlina Inglese e Marco Tedesco) durante una settimana e scriva in uscita:

- per ogni valuta:
  - il cambio medio durante la settimana;
  - il cambio massimo durante la settimana, con l'indicazione del giorno in cui è stato raggiunto;
  - la cambio minimo durante la settimana, con l'indicazione del giorno in cui è stato raggiunto;
- per ogni giorno (da martedì in poi) il nome della valuta il cui cambio ha avuto in percentuale il maggior incremento (o il minore decremento se tutti i cambi sono scesi) rispetto al giorno precedente.

Osserviamo prima di tutto che i dati in ingresso hanno una naturale rappresentazione in forma di tabella, ad esempio come:

	lunedì	*martedì*	*mercoledì*	*giovedì*	*venerdì*
Dollaro	1633.73	1624.40	1625.03	1630.41	1627.20
Franco	295.04	295.14	294.01	296.27	297.06
Sterlina	2746.63	2750.27	2750.21	2748.01	2751.00
Marco	989.24	989.58	982.38	985.26	986.90

Questa stessa rappresentazione verrà utilizzata all'interno del programma, mediante un array bidimensionale le cui righe rappresentano le valute e le cui colonne rappresentano i giorni della settimana. Introduciamo pertanto i seguenti tipi:

```
TYPE
   giorni = (lun, mar, mer, gio, ven, sab, dom);
   giorniLavorativi = lun..ven;
   valute = (dollaro, franco, sterlina, marco);
   tabcambi = ARRAY[valute, giorniLavorativi] OF real;
```

La struttura dati fondamentale del programma è una variabile `cambi` di tipo `tabcambi`.

Il programma è costituito da tre fasi principali, relizzate da tre procedure. La prima fase consiste nella lettura dei dati, cioè della tabella. Nella seconda fase si determina il cambio medio, minimo e massimo di ogni valuta. Nella terza fase si determina, per ogni giorno, la valuta che ha avuto un maggiore incremento di valore rispetto alla lira.

Avendo già definito la struttura dati da utilizzare ed individuato le fasi che costituiscono il programma, decidiamo di far uso di tre procedure principali, una per ogni fase, che possono essere sviluppate indipendentemente. Il programma principale è costituito dalla sequenza delle chiamate delle tre procedure:

```
BEGIN {valute}
   lettura;
   MediaMinMax;
   GiornoMax
END. {valute}
```

Lettura dei dati

Nella fase di lettura dei dati occorre leggere, per ogni giorno, il cambio di ciascuna valuta. A tale scopo si utilizzerà uno schema di questo tipo

```
FOR giorno := lun TO ven DO
   FOR valuta := dollaro TO marco DO
      readln(cambi[valuta, giorno])
```

dove le variabili `valuta` e `giorno` devono essere opportunamente dichiarate.

Per facilitare l'inserimento dei dati da parte dell'utente, è opportuno indicare i nomi del giorno e della valuta desiderata. Si potrebbe ad esempio far stampare un messaggio tipo `Inserire i cambi di lunedi'`, e scrivere quindi successivamente i nomi delle valute, al fine di richiedere all'utente l'inserimento del valore corrispondente. L'ideale sarebbe dunque scrivere

```
FOR giorno := lun TO ven DO
   {legge i cambi di giorno}
   BEGIN
      writeln('Inserire i cambi di ',giorno);
      FOR valuta := dollaro TO marco DO
         BEGIN
            write(valuta,': ');
            readln(cambi[valuta, giorno])
         END; {for valuta}
      writeln
END {for giorno}
```

Purtroppo le procedure predefinite di scrittura del Pascal Standard non permettono di scrivere le costanti dei tipi enumerativi. Pertanto non è possibile utilizzare le variabili `giorno` e `valuta` come parametri di `write` e di `writeln`. Per risolvere questo problema, costruiamo due procedure apposite di stampa, di nomi `writegiorno` e `writevaluta`, che permettono rispettivamente la scrittura di valori di tipo `giorni` e di valori tipo `valute`. Presentiamo ora la procedura `writegiorno`, la procedura `writevalute` è del tutto simile.

La procedura `writegiorno` riceve come parametro un valore di tipo `giorni`, in base al quale sceglie la sequenza di caratteri da scrivere in `output`, operando una selezione mediante un costrutto `CASE`:

```
PROCEDURE writegiorno (g: giorni);

{scrive su video il nome del giorno corrispondente al contenuto del parametro g}

BEGIN {writegiorno}
   CASE g OF
      lun:
         write('lunedi''');
      mar:
         write('martedi''');
      mer:
         write('mercoledi''');
      gio:
         write('giovedi''');
      ven:
         write('venerdi''');
      sab:
         write('sabato');
      dom:
         write('domenica')
   END
END; {writegiorno}
```

Utilizzando le procedure `writegiorno` e `writevaluta`, la procedura di lettura può essere scritta come:

```
PROCEDURE lettura;

{carica da input la tabella relativa ai cambi}

   VAR
      giorno: giorniLavorativi;
      valuta: valute;

BEGIN {lettura}
   FOR giorno := lun TO ven DO
      {legge i cambi di giorno}
      BEGIN
         write('Inserire i cambi di ');
         writegiorno(giorno);
         writeln;
         FOR valuta := dollaro TO marco DO
            BEGIN
               writevaluta(valuta);
               write(': ');
               readln(cambi[valuta, giorno])
            END; {for valuta}
         writeln
      END {for giorno}
END; {lettura}
```

Affinché la procedura `lettura` possa richiamarle, le procedure `writevaluta` e `writegiorno` verranno definite *prima* di essa. In questo modo potranno essere richiamate anche dalle due procedure `MediaMinMax` e `GiornoMax` per scrivere i messaggi in uscita.

Calcolo del cambio medio, minimo e massimo di ogni valuta

Costruiamo ora la seconda parte del programma, cioè la procedura per il calcolo e la scrittura del cambio medio, massimo e minimo di ogni valuta durante la settimana. In questa procedura occorrerà ripetere un certo insieme di operazioni per ogni valuta. Pertanto essa consisterà di un ciclo principale:

```
FOR valuta := dollaro TO marco DO
   calcola e scrivi il cambio medio, massimo e minimo di valuta
```

Per il calcolo del cambio medio è necessario sommare i cambi della valuta considerata che si sono avuti durante la settimana e dividere il risultato per il numero dei giorni. Per calcolare la somma si utilizzerà un ciclo iterativo facendo variare il giorno considerato.

Anche il calcolo del cambio massimo avviene iterativamente. Inizialmente si considera come cambio massimo quello di lunedì. Si esamina poi il cambio del giorno successivo; nel caso risulti superiore al cambio massimo calcolato sinora, si aggiorna il valore del cambio massimo utilizzando questo nuovo valore. Si ripete il procedimento esaminando tutti i giorni della settimana. Si utilizza inoltre una variabile di tipo `giorniLavorativi` e di nome `maxGiorno` per memorizzare il nome del giorno corrispondente al cambio massimo. In altre parole, il calcolo del cambio massimo può essere effettuato mediante il ciclo:

```
maxGiorno := lun;
maxValore := cambi[valuta, lun];
FOR giorno := mar TO ven DO
   IF cambi[valuta, giorno] > maxValore THEN
                     BEGIN
                        maxValore := cambi[valuta, giorno];
                        maxGiorno := giorno
                     END
```

Il calcolo del cambio minimo avviene in maniera del tutto simile.

Al posto di utilizzare tre cicli, uno per il calcolo della somma, uno per il calcolo del massimo e uno per il calcolo del minimo, possiamo servirci di un unico ciclo in cui vengono calcolati i tre valori:

```
{inizializzazione variabili}
minGiorno := lun;
minValore := cambi[valuta, lun];
maxGiorno := lun;
maxValore := cambi[valuta, lun];
somma := cambi[valuta, lun];

{calcolo della somma, minimo e massimo}
FOR giorno := mar TO ven DO
   BEGIN
      cambio := cambi[valuta, giorno];

      {aggiornamento della somma}
      somma := somma + cambio;
```

```
      {confronto ed evenutale aggiornamento del minimo}
      IF cambio < minValore THEN
         BEGIN
            minValore := cambio;
            minGiorno := giorno
         END;

      {confronto ed evenutale aggiornamento del massimo}
      IF cambio > maxValore THEN
         BEGIN
            maxValore := cambio;
            maxGiorno := giorno
         END
   END; {for giorno...}
```

Una volta calcolati questi valori e la media, possono essere scritti in `output`, utilizzando, oltre a `write` e `writeln`, anche le procedure `writevaluta` e `writegiorno`. Ecco il listato completo della procedura:

```
PROCEDURE MediaMinMax;

{per ogni valuta, calcola e scrive:}
{il cambio medio durante la settimana}
{il cambio massimo durante la settimana e il giorno in cui e' stato raggiunto}
{il cambio minimo durante la settimana e il giorno in cui e' stato raggiunto}

   VAR
      valuta: valute;
      giorno, minGiorno, maxGiorno: giorniLavorativi;
      minValore, maxvalore, cambio, somma, media: real;

BEGIN {MediaMinMax}
   FOR valuta := dollaro TO marco DO
      {esamina i cambi di valuta per trovare la media, il minimo e il massimo}
      BEGIN
         {inizializzazione variabili}
         minGiorno := lun;
         minValore := cambi[valuta, lun];
         maxGiorno := lun;
         maxValore := cambi[valuta, lun];
         somma := cambi[valuta, lun];

         {calcolo della somma, minimo e massimo}
         FOR giorno := mar TO ven DO
            BEGIN
               cambio := cambi[valuta, giorno];

               {aggiornamento della somma}
               somma := somma + cambio;

               {confronto ed evenutale aggiornamento del minimo}
               IF cambio < minValore THEN
                  BEGIN
```

```
                  minValore := cambio;
                  minGiorno := giorno
               END;

            {confronto ed evenutale aggiornamento del massimo}
            IF cambio > maxValore THEN
               BEGIN
                  maxValore := cambio;
                  maxGiorno := giorno
               END
         END; {for giorno...}

      {calcolo della media}
      media := somma / 5;

      {scrittura dei risultati}
      write('La valuta ');
      writevaluta(valuta);
      writeln(' questa settimana ha avuto:');
      writeln(' - cambio medio:   L. ', media : 7 : 2);
      write(' - cambio massimo: L. ', maxValore : 7 : 2);
      write(' ');
      writegiorno(maxGiorno);
      writeln;
      write(' - cambio minimo:  L. ', minValore : 7 : 2);
      write(' ');
      writegiorno(minGiorno);
      writeln;
      writeln
   END {for valuta...}
END; {MediaMinMax}
```

Calcolo della valuta con maggiore incremento

In questa fase occorre ripetere le stesse operazioni (cioè il calcolo della valuta con maggiore incremento) per tutti i giorni a partire da martedì (per effettuare il calcolo anche per lunedì sarebbe necessario disporre dei dati della settimana precedente). La procedura sarà costituita pertanto da un ciclo principale all'interno del quale viene effettuato il calcolo relativo a un giorno.

```
FOR giorno := mar TO ven DO
   determina la valuta il cui cambio rispetto alla lira si e' incrementato
   di piu' rispetto al giorno precedente
```

La tecnica per determinare il massimo è del tutto simile a quella utilizzata per la fase analizzata in precedenza, cambia solo l'utilizzo degli indici. Nella procedura `MediaMinMax` infatti dovevamo determinare per ogni valuta (ciclo esterno sulle valute) il giorno in cui si era avuta la quotazione massima (pertanto fissata la valuta nel ciclo esterno, veniva utilizzato un ciclo interno in cui si esaminano, al variare dei giorni, le quotazioni della valuta considerata); in questo caso invece occorre determinare per ogni giorno (ciclo esterno sui giorni) la valuta con massimo incremento (pertanto fissato il giorno nel ciclo esterno, si esaminano in un ciclo interno gli incrementi di tutte le valute in quel giorno). In altre parole, pensando alla tabella dei cambi che abbiamo evidenziato all'inizio dell'esempio, possiamo dire che, mentre nella procedura `MediaMinMax` essa veniva scandita per righe, nella procedura `GiornoMax` che stiamo costruendo, essa dovrà essere scandita per colonne.

Fissato dunque un giorno, cioè una colonna della tabella, dobbiamo determinare per quale valuta si è riscontrato l'incremento massimo in percentuale del cambio. Per calcolare l'incremento percentuale, occorre confrontare il cambio con quello del giorno precedente. È ben noto che la percentuale di incremento è data da

$$\frac{cambio\ del\ giorno * 100}{cambio\ del\ giorno\ precedente} - 100$$

Dunque si inizia considerando massimo l'incremento della prima valuta (dollaro); iterativamente si calcolano gli incrementi delle altre valute; quando l'incremento ottenuto risulta superiore al massimo calcolato sino a quel punto, si aggiorna il valore del massimo. In particolare, si utilizza una variabile `maxIncremento` di tipo `real` per memorizzare via via l'incremento massimo trovato, e una variabile `maxValuta` di tipo `valute` per ricordare a che valuta si riferisce il massimo incremento trovato.

In altre parole, fissato il `giorno` nel ciclo esterno, e introducendo opportune variabili ausiliarie, la ricerca e la stampa del massimo possono avvenire come segue:

```
{inizializzazione di maxIncremento e maxValuta}
cambioOggi := cambi[dollaro, giorno];
cambioIeri := cambi[dollaro, pred(giorno)];
maxIncremento := cambioOggi * 100 / cambioIeri - 100;
maxValuta := dollaro;

{ricerca del massimo}
FOR valuta := succ(dollaro) TO marco DO
   BEGIN
      {calcola l'incremento di valuta}
      cambioOggi := cambi[valuta, giorno];
      cambioIeri := cambi[valuta, pred(giorno)];
      incremento := cambioOggi * 100 / cambioIeri - 100;

      {confronta l'incremento con il massimo incremento trovato sinora}
      IF incremento > maxIncremento THEN
         BEGIN
            maxIncremento := incremento;
            maxValuta := valuta
         END

   END; {for valuta ...}

{stampa del risultato}
...
```

Ecco il listato completo della procedura:

```
PROCEDURE GiornoMax;

{per ogni giorno calcola e scrive}
{il nome della valuta il cui cambio rispetto alla lira si e' incrementato}
{di piu' (o decrementato di meno) rispetto al giorno precedente}

   VAR
      incremento, maxIncremento, cambioOggi, cambioIeri: real;
      valuta, maxValuta: valute;
```

```
      giorno: giorniLavorativi;

BEGIN {GiornoMax}
   FOR giorno := mar TO ven DO
      {determina la valuta il cui cambio rispetto alla lira si e' incrementato}
      {di piu' rispetto al giorno precedente}
      BEGIN
         {inizializzazione di maxIncremento e maxValuta}
         cambioOggi := cambi[dollaro, giorno];
         cambioIeri := cambi[dollaro, pred(giorno)];
         maxIncremento := cambioOggi * 100 / cambioIeri - 100;
         maxValuta := dollaro;

         {ricerca del massimo}
         FOR valuta := succ(dollaro) TO marco DO
            BEGIN
               {calcola l'incremento di valuta}
               cambioOggi := cambi[valuta, giorno];
               cambioIeri := cambi[valuta, pred(giorno)];
               incremento := cambioOggi * 100 / cambioIeri - 100;

               {confronta l'incremento con il massimo incremento trovato sinora}
               IF incremento > maxIncremento THEN
                  BEGIN
                     maxIncremento := incremento;
                     maxValuta := valuta
                  END

            END; {for valuta ...}

         {stampa del risultato}
         write('La valuta che ha avuto un maggiore incremento ');
         writeGiorno(giorno);
         write(' e'' stata ');
         writeValuta(maxValuta);
         writeln(' (', maxIncremento : 1 : 2, '%)')
      END {for giorno ...}
END; {GiornoMax}
```

Il programma completo

```
PROGRAM valute (input, output);

   TYPE
      giorni = (lun, mar, mer, gio, ven, sab, dom);
      giorniLavorativi = lun..ven;
      valute = (dollaro, franco, sterlina, marco);
      tabcambi = ARRAY[valute, giorniLavorativi] OF real;

   VAR
      cambi: tabcambi;
```

```
PROCEDURE writegiorno (g: giorni);

{scrive su video il nome del giorno corrispondente al contenuto del parametro g}

BEGIN {writegiorno}
   CASE g OF
      lun:
         write('lunedi''');
      mar:
         write('martedi''');
      mer:
         write('mercoledi''');
      gio:
         write('giovedi''');
      ven:
         write('venerdi''');
      sab:
         write('sabato');
      dom:
         write('domenica')
   END
END; {writegiorno}



PROCEDURE writevaluta (v: valute);

{scrive su video il nome della valuta corrispondente al contenuto del parametro v}

BEGIN {writevaluta}
   CASE v OF
      dollaro:
         write('dollaro');
      franco:
         write('franco');
      sterlina:
         write('sterlina');
      marco:
         write('marco')
   END
END; {writevaluta}



PROCEDURE lettura;

{carica da input la tabella relativa ai cambi}

   VAR
      giorno: giorniLavorativi;
      valuta: valute;
```

```
BEGIN {lettura}
   FOR giorno := lun TO ven DO
      {legge i cambi di giorno}
      BEGIN
         write('Inserire i cambi di ');
         writegiorno(giorno);
         writeln;
         FOR valuta := dollaro TO marco DO
            BEGIN
               writevaluta(valuta);
               write(': ');
               readln(cambi[valuta, giorno])
            END; {for valuta}
         writeln
      END {for giorno}
END; {lettura}



PROCEDURE MediaMinMax;

{per ogni valuta, calcola e scrive:}
{il cambio medio durante la settimana}
{il cambio massimo durante la settimana e il giorno in cui e' stato raggiunto}
{la cambio minimo durante la settimana e il giorno in cui e' stato raggiunto}

   VAR
      valuta: valute;
      giorno, minGiorno, maxGiorno: giorniLavorativi;
      minValore, maxvalore, cambio, somma, media: real;

BEGIN {MediaMinMax}
   FOR valuta := dollaro TO marco DO
      {esamina i cambi di valuta per trovare la media, il minimo e il massimo}
      BEGIN
         {inizializzazione variabili}
         minGiorno := lun;
         minValore := cambi[valuta, lun];
         maxGiorno := lun;
         maxValore := cambi[valuta, lun];
         somma := cambi[valuta, lun];

         {calcolo della somma, minimo e massimo}
         FOR giorno := mar TO ven DO
            BEGIN
               cambio := cambi[valuta, giorno];

               {aggiornamento della somma}
               somma := somma + cambio;

               {confronto ed evenutale aggiornamento del minimo}
```

```
                IF cambio < minValore THEN
                   BEGIN
                      minValore := cambio;
                      minGiorno := giorno
                   END;

                {confronto ed evenutale aggiornamento del massimo}
                IF cambio > maxValore THEN
                   BEGIN
                      maxValore := cambio;
                      maxGiorno := giorno
                   END
             END; {for giorno...}

          {calcolo della media}
          media := somma / 5;

          {scrittura dei risultati}
          write('La valuta ');
          writevaluta(valuta);
          writeln(' questa settimana ha avuto:');
          writeln(' - cambio medio:   L. ', media : 7 : 2);
          write(' - cambio massimo: L. ', maxValore : 7 : 2);
          write(' ');
          writegiorno(maxGiorno);
          writeln;
          write(' - cambio minimo:  L. ', minValore : 7 : 2);
          write(' ');
          writegiorno(minGiorno);
          writeln;
          writeln
       END {for valuta ...}
END; {MediaMinMax}



PROCEDURE GiornoMax;

{per ogni giorno calcola e scrive}
{il nome della valuta il cui cambio rispetto alla lira si e' incrementato}
{di piu' (o decrementato di meno) rispetto al giorno precedente}

   VAR
      incremento, maxIncremento, cambioOggi, cambioIeri: real;
      valuta, maxValuta: valute;
      giorno: giorniLavorativi;

BEGIN {GiornoMax}
   FOR giorno := mar TO ven DO
      {determina la valuta il cui cambio rispetto alla lira si e' incrementato}
      {di piu' rispetto al giorno precedente}
      BEGIN
```

```
            {inizializzazione di maxIncremento e maxValuta}
            cambioOggi := cambi[dollaro, giorno];
            cambioIeri := cambi[dollaro, pred(giorno)];
            maxIncremento := cambioOggi * 100 / cambioIeri - 100;
            maxValuta := dollaro;

            {ricerca del massimo}
            FOR valuta := succ(dollaro) TO marco DO
               BEGIN
                  {calcola l'incremento di valuta}
                  cambioOggi := cambi[valuta, giorno];
                  cambioIeri := cambi[valuta, pred(giorno)];
                  incremento := cambioOggi * 100 / cambioIeri - 100;

                  {confronta l'incremento con il massimo incremento trovato sinora}
                  IF incremento > maxIncremento THEN
                     BEGIN
                        maxIncremento := incremento;
                        maxValuta := valuta
                     END

               END; {for valuta ...}

            {stampa del risultato}
            write('La valuta che ha avuto un maggiore incremento ');
            writeGiorno(giorno);
            write(' e'' stata ');
            writeValuta(maxValuta);
            writeln(' (', maxIncremento : 1 : 2, '%)')
         END {for giorno ...}
   END; {GiornoMax}

BEGIN {valute}
   lettura;
   MediaMinMax;
   GiornoMax
END. {valute}
```

13.2 I set

Un *set* (in italiano *insieme*) è un insieme di valori di uno stesso tipo *scalare*. Pertanto un tipo set viene definito specificando un tipo scalare base. Ad esempio, nelle seguenti dichiarazioni, viene introdotto un tipo `stagioni`, i cui valori sono sottoinsiemi del tipo `mesi`:

```
TYPE
   mesi = (gen, feb, mar, apr, mag, giu, lug, ago, sep, ott, nov, dic);
   estivi = giu..set;
   stagioni = SET OF mesi;
VAR
   estate: stagioni;
   estivo: estivi;
```

Mentre la variabile `estivo` può assumere uno solo dei valori nell'intervallo `giu..set`, la variabile `estate` può assumere come valore un qualsiasi sottoinsieme di tale intervallo, compreso l'insieme vuoto. Ad esempio l'assegnamento `estate := []` assegna alla variabile `estate` l'insieme vuoto, l'assegnamento `estate := [giu, ago]` assegna alla variabile `estate` l'insieme costituito dai due valori indicati, mentre l'assegnamento `estate := [giu..sep]` assegna alla variabile `estate` l'insieme costituito dai valori `giu, lug, ago, sep`.

È possibile effettuare su set dello stesso tipo le operazioni insiemistiche di *unione*, *intersezione* e *differenza*, indicate rispettivamente con i simboli `+`, `*` e `-`. Ad esempio, dichiarando

```
VAR
   maiuscole, minuscole, vocali, consonanti, x: SET OF char;
```

dopo avere eseguito gli assegnamenti

```
maiuscole := ['A'..'Z'];
minuscole := ['a'..'z'];
vocali := ['a', 'e', 'i', 'o', 'u'];
consonanti := minuscole - vocali;
x := maiuscole + minuscole;
```

la variabile `consonanti` conterrà l'insieme di tutti i caratteri corrispondenti a consonanti minuscole, mentre la variabile `x` conterrà l'insieme di tutte le lettere minuscole e maiuscole.

Sono inoltre disponibili degli operatori relazionali o di confronto `IN`, `=`, `<>`, `<=`, `>=`. Gli operatori `=` e `<>` verificano rispettivamente l'uguaglianza o la disuguaglianza di due insiemi. L'operatore `<=` verifica che l'insieme specificato alla sua sinistra sia contenuto nell'insieme specificato alla sua destra. Ad esempio, dopo gli assegnamenti precedenti, il risultato di `vocali <= x` è `true`, mentre quello di `['a','e','9'] <= x` è `false`. L'operatore `=>` verifica che l'insieme specificato alla sua destra sia contenuto in quello specificato alla sua sinistra. L'operatore `IN` confronta un elemento di un tipo scalare con un insieme di elementi dello stesso tipo; il risultato è `true` se l'elemento appartiene all'insieme. Ad esempio, se la variabile `c` è di tipo `char`, il risultato di `c IN vocali` è `true` quando il contenuto di `c` è una vocale.

Esempio: Lettere presenti in un testo

Vogliamo costruire un programma che dopo avere letto una riga di testo da `input`, scriva in `output` un elenco delle lettere presenti almeno una volta nel testo, e indichi se sono state incontrate o no tutte le consonanti. A questo scopo potremmo modificare opportunamente il programma che calcola il numero di occorrenze di ciascuna lettera in un testo. Tuttavia, in questo caso è inutile memorizzare il numero di occorrenze di ciascuna lettera, ma è sufficiente ricordare l'insieme di lettere incontrate almeno una volta nel testo. Pertanto introduciamo una variariabile `lettereviste`, in grado di memorizzare insiemi di lettere, che inizialmente contiene l'insieme vuoto. Man mano che si incontra una lettera nel testo, essa viene aggiunta a `lettereviste`.

Il programma è quindi basato su un ciclo del tipo

```
lettereviste := []; {insiemevuoto}
WHILE NOT eoln DO
   BEGIN
      leggi un carattere
      IF carattere e' una lettera
         THEN aggiungi il carattere a lettereviste
   END
```

Se `ch` è la variabile di tipo `char` utilizzata per leggere il carattere, l'operazione `aggiungi il carattere a lettereviste` può essere scritta utilizzando l'operatore di unione come

```
lettereviste := lettereviste + [ch]
```

Decidiamo inoltre di convertire le lettere maiuscole in minuscole, e di costruire l'insieme con le lettere minuscole.

Il programma avrà dunque la seguente struttura:

```
PROGRAM setlettere (input, output);

   TYPE
      lettere = 'a'..'z';
      insiemelettere = SET OF lettere;

   VAR
      lettereviste, minuscole: insiemelettere;
      ch: char;

BEGIN {setlettere}
   minuscole := ['a'..'z'];
   lettereviste := [];

   WHILE NOT eoln DO
      BEGIN
         read(ch);

         {conversione delle maiuscole in minuscole}
         IF ch IN ['A'..'Z'] THEN
            ch := chr(ord(ch) - ord('A') + ord('a'));

         IF ch IN minuscole THEN
            lettereviste := lettereviste + [ch]
      END; {while}
   readln;

   scrittura del risultato

END. {setlettere}
```

Per scrivere l'elenco delle lettere presenti nell'insieme, effettuiamo un ciclo sulle lettere minuscole: per ogni lettera verifichiamo che sia presente nell'insieme `lettereviste` e, in caso affermativo, la scriviamo in `output`. Per la verifica si utilizza di nuovo l'operatore `IN`:

```
FOR lettera := 'a' TO 'z' DO
   IF lettera IN lettereviste THEN
      write(lettera)
```

È inoltre richiesto di scrivere in `output` se il testo contiene o no tutte le consonanti. A tale scopo è sufficiente verificare che l'insieme delle consonanti sia contenuto nell'insieme `lettereviste`, utilizzando l'operatore `<=`. Creiamo una variabile `consonanti` a cui assegnamo l'insieme di tutte le consonanti, e scriviamo il seguente codice:

```
IF consonanti <= lettereviste THEN
   writeln('Il testo contiene tutte le consonanti')
ELSE
   writeln('Il testo non contiene tutte le consonanti')
```

Per assegnare alla variabile `consonanti` l'insieme di tutte le consonanti, possiamo scrivere un assegnamento, con il lungo elenco di tutte le consonanti, cioè

```
consonanti := ['b', 'c', 'd', ecc...]
```

oppure possiamo assegnare a `consonanti` la differenza tra `minuscole` e l'insieme delle vocali, cioè scrivere

```
consonanti := minuscole - ['a', 'e', 'i', 'o', 'u']
```

Ecco il codice completo del programma:

```
PROGRAM setlettere (input, output);

   TYPE
      lettere = 'a'..'z';
      insiemelettere = SET OF lettere;

   VAR
      lettereviste, consonanti, minuscole: insiemelettere;
      ch: char;
      lettera: lettere;

BEGIN {setlettere}
   minuscole := ['a'..'z'];
   consonanti := minuscole - ['a', 'e', 'i', 'o', 'u'];
   lettereviste := [];

   WHILE NOT eoln DO
      BEGIN
         read(ch);
         IF ch IN ['A'..'Z'] THEN
            ch := chr(ord(ch) - ord('A') + ord('a'));

         IF ch IN minuscole THEN
            lettereviste := lettereviste + [ch]
      END; {while}
   readln;

   write('Il testo contiene le lettere ');
   FOR lettera := 'a' TO 'z' DO
      IF lettera IN lettereviste THEN
         write(lettera);
   writeln;

   IF consonanti <= lettereviste THEN
      writeln('Il testo contiene tutte le consonanti')
   ELSE
      writeln('Il testo non contiene tutte le consonanti')

END. {setlettere}
```

Esempio: Numeri primi

Nella lezione 8 sono stati presentati alcuni programmi per decidere se un numero intero dato in ingresso sia primo. In questi programmi l'intero in input veniva successivamente suddiviso per tutti i numeri minori di esso, al fine di trovare eventuali divisori.

Sviluppiamo ora un programma che elenca tutti i numeri primi minori di un numero dato in ingresso, basato su una tecnica piú efficiente detta del *crivello di Eratostene*. L'idea alla base dell'algoritmo utilizzato è molto semplice: dato un numero primo, tutti i suoi multipli non sono primi. Consideriamo un insieme, che inizialmente contiene tutti i numeri maggiori o uguali a due. Ogni volta che individuiamo un numero primo, escludiamo dall'insieme tutti i suoi multipli. Piú precisamente, cominciamo con l'escludere dall'insieme i multipli di 2, poi quelli di 3, poi quelli di 5 (4 non viene considerato in quanto già escluso come multiplo di 2), e così via. In altre parole, per determinare l'insieme dei primi minori di un intero `MaxNum` dato, possiamo utilizzare il seguente schema:

```
primi := insieme di tutti gli interi tra 2 e MaxNum

FOR numero := 2 TO MaxNum DO
   IF numero appartiene all'insieme dei primi THEN
      elimina tutti i multipli di numero dall'insieme dei primi
```

I multipli di `numero` minori di `MaxNum` sono `numero * 2`, `numero * 3`, ecc., fino a `numero * (MaxNum DIV numero)`. Pertanto, l'eliminazione dall'insieme `primi` dei multipli di `numero` può essere espressa mediante il seguente ciclo `FOR`:

```
FOR fattore := 2 TO MaxNum DIV numero DO
   elimina numero * fattore dall'insieme dei primi
```

Passiamo ora alla codifica del programma. Rappresentiamo l'insieme dei numeri primi con un set di valori di tipo `integer`. Per ragioni implementative ogni compilatore pone un limite superiore al numero di elementi di un set. Pertanto l'uso del tipo `SET OF integer` non verrà di solito accettato dal compilatore. Fissiamo come limite superiore, al massimo numero trattabile, il valore `255`. Questa limitazione viene accettata dalla maggioranza dei compilatori. Per definire l'insieme `primi`, utilizziamo dunque le seguenti dichiarazioni:

```
CONST
   max = 255; {massima ampiezza per i SET (dipende dal compilatore)}

TYPE
   numeri = 1..max;
   insieme = SET OF numeri;

VAR
   primi: insieme;
```

Il programma è costituito da tre fasi: la prima in cui l'utente introduce il numero, che verrà memorizzato nella variabile `NumMax`, sino al quale si vogliono trovare i numeri primi; la seconda nella quale si determina l'insieme dei numeri primi compresi tra `2` e `NumMax`, e la terza in cui si scrive in output il contenuto dell'insieme.

Segue la codifica completa del programma:

```
PROGRAM eratostene (input, output);

{determina tutti i numeri primi tra 2 e un numero dato utilizzando il crivello di Eratostene}
```

```
   CONST
      max = 255; {massima ampiezza per i SET (dipende dal compilatore)}

   TYPE
      numeri = 1..max;
      insieme = SET OF numeri;

   VAR
      primi: insieme;
      numero, fattore, MaxNum: integer;

BEGIN {eratostene}

   {lettura dati}
   write('Inserire il massimo intero da considerare ');
   REPEAT
      readln(MaxNum);
      IF MaxNum > max THEN
         write('Numero troppo grande (max ', max : 1, ') - Ripetere l''inserimento ')
   UNTIL MaxNum <= max;


   {determinazione dell'insieme dei primi}
   primi := [2..MaxNum];
   FOR numero := 2 TO MaxNum DO
      IF numero IN primi THEN
         FOR fattore := 2 TO MaxNum DIV numero DO
            primi := primi - [numero * fattore];

   {scrittura dell'insieme}
   writeln('I numeri primi minori di ', MaxNum : 1, ' sono:');
   FOR numero := 2 TO MaxNum DO
      IF numero IN primi THEN
         writeln(numero)

END. {eratostene}
```

Esercizi

1. Se in una giornata tutte le valute sono diminuite di valore rispetto alla lira, la procedura `GiornoMax` del programma `valute` scriverà un messaggio tipo

   ```
   La valuta che ha avuto un maggiore incremento martedi' e' stata dollaro (-0.72%)
   ```

   Modificare il programma `valute` in modo che, nel caso in cui tutte le valute siano diminuite di valore rispetto alla lira, l'uscita sia un messaggio tipo

   ```
   Nella giornata di martedi' tutte le valute hanno perso valore rispetto alla lira.
   La valuta che ha avuto un minore decremento e' stata dollaro (-0.72%)
   ```

2. In alcuni dei messaggi scritti in `output` dal programma `valute` (ad esempio nella procedura `GiornoMax`) sarebbe opportuno far precedere il nome della valuta dall'articolo appropriato.

Riscrivere la procedura `writevaluta` (e le relative chiamate all'interno del programma) in modo che vi sia la possibilità di scegliere tra la stampa solo del nome della valuta e la stampa del nome preceduto dall'articolo (tipo `il dollaro` o `la sterlina`). L'opzione potrebbe essere specificata con un secondo parametro di tipo `boolean`. In questo caso l'intestazione della procedura diventerebbe

```
PROCEDURE writevaluta (v: valute; art: boolean);
```

L'articolo davanti al nome della valuta dovrà essere stampato solo quando il secondo parametro utilizzato nella chiamata vale `true`.

3. Modificare il programma `valute` in modo che, per ogni giorno, determini anche la valuta il cui cambio si è incrementato di meno (o decrementato di piú), in percentuale, rispetto alla lira.

4. Costruire un programma che legga i dati relativi alle precipitazioni mensili che si sono avute in un certo intervallo di anni fissato e determini l'anno complessivamente piú piovoso e il mese complessivamente piú piovoso.

5. Costruire un programma che riceva in ingresso i risultati di un referendum, espressi come numero di voti, suddivisi tra *sì*, *no*, *schede bianche*, *schede nulle* e per aree geografiche (Nord, Centro, Sud e Isole), e produca in output:

   - una tabella delle percentuali dei risultati nelle varie zone;
   - le percentuali complessive dei sì, no, schede bianche e schede nulle.

6. Modificare il programma che scrive l'elenco di tutte le lettere presenti in un testo, in modo che scriva anche un elenco di tutte le vocali e di tutte le consonati non presenti nel testo.

7. Modificare il programma che scrive l'elenco di tutte le lettere presenti in un testo, in modo che distingua tra lettere minuscole e maiuscole, producendo in output, a scelta dell'utente, prima un elenco delle minuscole e poi uno delle maiuscole, oppure un unico elenco.

8. Riscrivere il programma per la determinazione dei numeri primi, utilizzando, per la rappresentazione dell'insieme dei numeri primi, un array di valori di tipo `boolean`, al posto di un set (il valore `true` in posizione `i`, indica che `i` appartiene all'insieme).