Programmazione ad Oggetti
Homework 2 (I/O del C++)

Esercizio 1

Scrivere un programma che carica da file una matrice di reali. Il nome del file, il numero di righe e di colonne **devono essere richiesti all'utente in maniera interattiva**. Un volta acquisita, la matrice sarà stampata a video. La memoria per la memorizzazione deve essere allocata dinamicamente, in base alle specifiche fornite dall'utente. Nel caso in cui non sia possibile aprire il file, il programma stamperà un opportuno messaggio. Il programma dovrà inoltre verificare l'effettiva lettura del numero di elementi specificato dall'utente.
Il programma dovrà inoltre essere in grado di salvare la matrice su di un file il cui nome è specificato dall'utente.

Suggerimenti

Per allocare una matrice (a valori reali) di r righe e c colonne si definisca la seguente funzione:

```
float** alloc_mat(int r, int c)
```

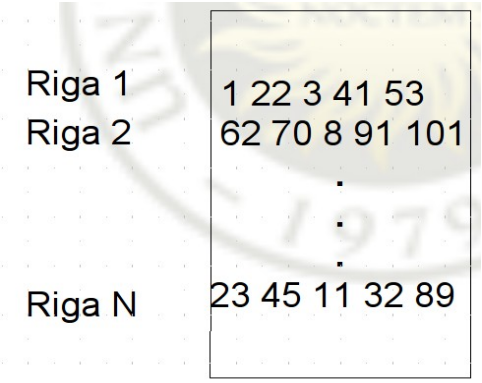
che alloca una matrice di r righe e c colonne e ne restituisce il puntatore
Per visualizzare e/o salvare la matrice su file ai definisca la funzione:

```
void show_mat(ostream *out, float **mat, int rows, int cols)
```

che mostra la matrice di r righe e c colonne, puntata dal parametro mat, sullo stream passato come parametro.

Esercizio 2

Scrivere un programma che carica da file una matrice di reali. Il formato del file è il seguente:



Si tenga inoltre presente che, su un stessa riga gli elementi di una stessa colonna sono separati da uno più delimitatori che l'ultimo elemento di ogni colonna è seguito dal carattere di fine linea ('\n').
Il nome del file deve essere richiesto all'utente in maniera interattiva, mentre il numero di righe e colonne deve essere determinato dal programma **in maniera automatica**. Il programma dovrà inoltre controllare che tutte le righe contengano lo stesso numero di colonne (quello della prima), in caso contrario il programma terminerà dopo aver stampato un opportuno messaggio.
Un volta acquisita, la matrice sarà stampata a video. La memoria deve essere allocata dinamicamente. Il programma stamperà un opportuno messaggio, nel caso in cui non sia possibile aprire il file.

Suggerimenti

- Per allocare la matrice si utilizzi la funzione dell'esercizio precedente.
- Per determinare in maniera in maniera automatica il numero di righe e di colonne si definisca la funzione:

```
bool comp_dim(ifstream &in, int &rows, int &cols);
```

che: (i) calcola il numero di righe e colonne della matrice contenuta nello stream in; (ii) restituisce true se il formato della matrice è corretto, false altrimenti.
A tal fine si definisca un'ulteriore funzione:

```
int comp_cols(ifstream &in)
```

che è in grado di determinare il numero di colonne presenti prossima linea dello stream `in`. A tale scopo, si modifichi la funzione line2vec (vedi sotto).

- Si tenga inoltre presente che dopo la chiamata a `comp_dim`, lo stream va chiuso e riaperto

```
void line2vec(ifstream &in, float v[], int &n, char *delim)
{
      char str[max_line], c, *tok;

      // Estrazione della linea dallo stream
      in.getline(str, max_line);

      // Si controlla che non sia vuota
      if (strlen(str) == 0) {
        n = 0;
        return;
      }

      // Estrazione dei valori dalla linea
      n=0;
      tok = strtok(str, delim);
      v[n++] = atof(tok);

      while (tok = strtok(0, delim)) {
        v[n++] = atof(tok);
        cout<<endl<<tok;
      }

      return;
}
```

NOTA
la funzione

```
char * strtok (char *str, const char* delim)
```

suddivide la stringa `str` in parti (tokens), delimitate da uno dei delimitatori (singoli caratteri) elencati nella stringa `delim`.
In pratica una sequenza di chiamate a questa funzione consente di suddividere la stringa str in una sequenza di tokens, dove un token è una sequenza di caratteri contigui separati da uno dei caratteri presenti in `delim`.
Alla prima chiamata bisogna fornire il puntatore alla stringa da dividere, e la funzione restituisce il puntatore al primo token trovato; nelle chiamate successive, come primo parametro va passato il puntatore nullo, e la

funzione restituisce il puntatore al token successivo trovato, o il puntatore nullo se la stringa è terminata.

Esempio

```c
#include <stdio.h>
#include <string.h>

int main ()
{
  char str[] =" This,is a -sample- string.";
  char * pch;
  printf ("Splitting string \"%s\" into tokens:\n",str);
  pch = strtok (str," ,.-");
  while (pch != NULL)
  {
    printf ("%s\n",pch);
    pch = strtok (NULL, " ,.-");
  }
  return 0;
}
```

OUTPUT

```
Splitting string " This, is a -sample- string." into tokens:
This
is
a
sample
string
```