IEIM 2015-2016

Esercitazione XII

"Ripasso generale"

Alessandro A. Nacci
alessandro.nacci@polimi.it - www.alessandronacci.it

Tratto da:
IEIM Lab 2014
Gianluca Durelli – durelli@elet.polimi.it
Marco D. Santambrogio – marco.santambrogio@polimi.it

Exe 1: Valori Circostanti

- Si scriva una funzione in C che:
  - Ricevuta in ingresso una matrice di dimensione NxM (con dimensioni scelte dall'utente) ed una posizione (i,j) della matrice
  - Restituisca la somma dei valori connessi alla posizione scelta
  - Nota: Due valori sono connessi se sono adiacenti in orizzontale, verticale oppure obliquo.

Exe 1: Valori Circostanti

```
int circostanti(int M[][DIM], int riga, int colonna, int rows, int cols){

}
```

Exe 1: Valori Circostanti

```
int getValue(int M[][DIM], int riga, int colonna, int rows, int cols){
     if(riga<0 || riga>rows-1 || colonna<0 || colonna>cols-1)
          return 0;
     else
          return M[riga][colonna];
}

int circostanti(int M[][DIM], int riga, int colonna, int rows, int cols){

     int sum = 0;
     int i,j;

     for(i=-1; i<=1; i++)
          for(j=-1; j<=1; j++)
               if(i!=0 || j!=0)
                    sum+=getValue(M, riga+i, colonna+j, rows, cols);

     return sum;

}
```

Exe 1: Valori Circostanti

```c
int main(){

    int M[DIM][DIM];
    int rows=0, cols=0, r=0, c=0;

    int riga, colonna;

    int sum;

    do{
        printf("Numero righe [al massimo %d]: ", DIM);
        scanf("%d", &rows);
        printf("Numero colonne [al massimo %d]: ", DIM);
        scanf("%d", &cols);
    } while(rows>DIM || cols>DIM);

    for(r=0; r<rows; r++)
        for(c=0; c<cols; c++){
            printf("Inserisci valore in posizione %d-%d: ", r, c);
            scanf("%d", &M[r][c]);
        }

    printf("Inserisci la riga: ");
    scanf("%d", &riga);
    printf("Inserisci la colonna: ");
    scanf("%d", &colonna);

    sum = circostanti(M, riga, colonna, rows, cols);

    printf("La sommma dei valori circostanti alla posizione %d-%d e': %d\n", riga, colonna, sum);

    return 0;
}
```

Exe 2: Linea Spezzata

- Si scriva una funzione che:
  - Ricevuto un ingresso un vettore di elementi che rappresentano dei punti in un piano cartesiano
  - Calcoli la lunghezza della linea spezzata che li collega
  - Nota: Si definiscano opportune strutture dati per risolvere l'esercizio

Exe 2: Linea Spezzata

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

#define DIM 10

typedef struct{
    float X;
    float Y;
} Punto;

void leggiPunto(Punto *p);
float lunghezzaSegmento(Punto *p1, Punto *p2);
float lunghezzaLinea(Punto *p, int numPunti);
```

Exe 2: Linea Spezzata

```
void leggiPunto(Punto *p){
     printf("Inserisci le coordinate del punto");
     printf("\tX: ");
     scanf("%f", &p->X);
     printf("\tY: ");
     scanf("%f", &p->Y);
}
```

Exe 2: Linea Spezzata

```
float lunghezzaSegmento(Punto *p1, Punto *p2){
    float lunghezza = (p1->X - p2->X) * (p1->X - p2->X) + (p1->Y - p2->Y) * (p1->Y - p2->Y);
    return sqrt(lunghezza);
}
```

Exe 2: Linea Spezzata

```
float lunghezzaLinea(Punto *p, int numPunti){
     float lunghezza = 0;
     int i;

    for(i=0; i<numPunti-1; i++)
         lunghezza += lunghezzaSegmento(&p[i], &p[i+1]);

    return lunghezza;
}
```

Exe 2: Linea Spezzata

```
float lunghezzaLineaRic(Punto *p, int i, int numPunti)
{

     // o------o------o------o 4 punti, 3 hop, la i è da 0 a 2

     if (i == numPunti - 2)
     {
          return lunghezzaSegmento(&p[i], &p[i+1]);
     }
     else
     {
          return lunghezzaSegmento(&p[i], &p[i+1]) + lunghezzaLineaRic(p, i+1, numPunti);
     }

}
```

versione ricorsiva

```
float lunghezzaLinea(Punto *p, int numPunti){
     float lunghezza = 0;
     int i;

    for(i=0; i<numPunti-1; i++)
        lunghezza += lunghezzaSegmento(&p[i], &p[i+1]);

    return lunghezza;
}
```

versione iterativa

```
float lunghezzaLineaRic(Punto *p, int i, int numPunti)
{

     // o------o------o------o 4 punti, 3 hop, la i è da 0 a 2

     if (i == numPunti - 2)
     {
          return lunghezzaSegmento(&p[i], &p[i+1]);
     }
     else
     {
          return lunghezzaSegmento(&p[i], &p[i+1]) + lunghezzaLineaRic(p, i+1, numPunti);
     }

}
```

versione ricorsiva

```c
int main(){
     Punto p[DIM];
     int numPunti;
     float lunghezza;

     int i;

     printf("Da quanti punti e' formata la linea? ");
     scanf("%d", &numPunti);

     for(i=0; i<numPunti; i++)
          leggiPunto(&p[i]);

     lunghezza = lunghezzaLinea(p, numPunti);

     printf("La lunghezza della line spezzata e' pari a %f\n", lunghezza);

     return 0;
}
```

Exe 3: Media Studenti

- Si realizzi una struttura dati per la gestione dell'anagrafica degli studenti
- Per ogni studente ci interessa:
  - *Nome, cognome* e *media (con decimali)*

- Si scrivano quindi due funzione in C:
  - *La prima per inserire gli studenti nell'anagrafica*
  - *La seconda che permetta di salvare in un array tutti e soli gli studenti che hanno una media compresa tra un intervallo scelto dall'utente*

Exe 3: Media Studenti

```c
#include <stdio.h>

#define LEN 100
#define STUDS 50

typedef struct{
      char nome[LEN];
      int matricola;
      float media;
} Studente;
```

Exe 3: Media Studenti

```
void leggiStudente(Studente *studente){
     printf("Dati studente: \n");

     printf("\tNome: ");
     scanf("%s", studente->nome);

     printf("\tMatricola: ");
     scanf("%d", &(studente->matricola));

     do{
          printf("\tMedia: ");
          scanf("%f", &(studente->media));
     }while(studente->media<0 || studente->media>30);
}
```

Exe 3: Media Studenti

```
int leggiStudenti(Studente *studenti){

     int N, n;

     do{
          printf("Quanti studenti vuoi inserire [al massimo %d]? ", STUDS);
          scanf("%d", &N);
     }while(N>STUDS);

     for(n=0; n<N; n++)
          leggiStudente(&studenti[n]);

     return N;
}
```

Exe 3: Media Studenti

```
int getStudenti(Studente *studenti, int numStudenti, int minMedia, int maxMedia, Studente *studentiScelti){
     int s;
     int pos;

     int numStudentiScelti = 0;

     for(s=0; s<numStudenti; s++){
          if(studenti[s].media>=minMedia && studenti[s].media<=maxMedia){
               studentiScelti[numStudentiScelti] = studenti[s];
               numStudentiScelti++;
          }
     }

     return numStudentiScelti;
}
```

Exe 3: Media Studenti

```
int main(){
     Studente studenti[STUDS], studentiScelti[STUDS];
     int istogrammaMedie[30];
     int numeroStudenti = 0, numeroStudentiScelti = 0;
     int s;

     float minMedia, maxMedia;

     numeroStudenti = leggiStudenti(studenti);

     printf("Minima Media: ");
     scanf("%f", &minMedia);
     printf("Massima Media: ");
     scanf("%f", &maxMedia);

     numeroStudentiScelti = getStudenti(studenti, numeroStudenti, minMedia, maxMedia, studentiScelti);



     for(s=0; s<numeroStudentiScelti; s++)
          printf("%s\t%d\t%f\n", studentiScelti[s].nome, studentiScelti[s].matricola, studentiScelti[s].media);

     return 0;
}
```

Exe 4: Propagazione

- Scrivere una funzione in C che:
  - *Riceva in ingresso una matrice di dimensioni NxM (scelte dall'utente) inizializzata a 0*
  - *Riceva in ingresso una posizione della matrice (I,j) da inizializzare ad un valore X*
- Propaghi il valore iniziale secondo le seguenti regole:
  - *La matrice va sempre scorsa da sinistra verso destra e dall'alto verso il basso*
  - *Se una cella ha valore 0 questa assumera' un valore pari alla somma dei valori contenuti nelle celle adiacenti*
- L'algoritmo si ferma quando non esistono piu' celle con valore 0.

Exe 4: Propagazione

```c
#include <stdio.h>

#define DIM 10

typedef struct{
     int riga;
     int colonna;
} Posizione;

void propaga(int M[][DIM], int rows, int cols, int riga, int colonna);
void stampaMatrice(int M[][DIM], int rows, int cols);
int zero(int M[][DIM], int rows, int cols);

int getValue(int [][DIM], int, int, int, int);
int circostanti(int [][DIM], int, int, int, int);
```

Exe 4: Propagazione

```
void propaga(int M[][DIM], int rows, int cols, int riga, int colonna){

    int r,c;

    for(r=0; r<rows; r++)
        for(c=0; c<rows; c++)
            M[r][c] = 0;


    if(riga<0 || riga>rows-1 || colonna<0 || colonna>cols-1)
        return;

    M[riga][colonna] = 1;

    while(zero(M, rows, cols)){
        for(r=0; r<rows; r++)
            for(c=0; c<rows; c++)
                if(M[r][c] == 0)
                    M[r][c] = circostanti(M, r, c, rows, cols);
    }

}
```

Exe 4: Propagazione

```
int zero(int M[][DIM], int rows, int cols){
     int r,c;

     for(r=0; r<rows; r++)
          for(c=0; c<rows; c++)
               if(M[r][c] == 0)
                    return 1;

     return 0;
}
```

Exe 4: Propagazione

```
int circostanti(int M[][DIM], int riga, int colonna, int rows, int cols){

    int sum = 0;
    int i,j;

    for(i=-1; i<=1; i++)
        for(j=-1; j<=1; j++)
            if(i!=0 || j!=0)
                sum+=getValue(M, riga+i, colonna+j, rows, cols);

    return sum;

}
```

Exe 4: Propagazione

```
int getValue(int M[][DIM], int riga, int colonna, int rows, int cols){
    if(riga<0 || riga>rows-1 || colonna<0 || colonna>cols-1)
        return 0;
    else
        return M[riga][colonna];
}
```

Exe 4: Propagazione

```
void stampaMatrice(int M[][DIM], int rows, int cols){
    int r,c;

    for(r=0; r<rows; r++){
        for(c=0; c<cols; c++){
            printf("%d\t", M[r][c]);
        }
        printf("\n");
    }
}
```

Exe 4: Propagazione

```
int main(){

    int M[DIM][DIM];
    int rows=0, cols=0;

    int riga, colonna;

    do{
        printf("Numero righe [ al massimo %d]: ", DIM);
        scanf("%d", &rows);
        printf("Numero colonne [ al massimo %d]: ", DIM);
        scanf("%d", &cols);
    }while(rows>DIM || cols>DIM);

    printf("Inserisci la riga: ");
    scanf("%d", &riga);
    printf("Inserisci la colonna: ");
    scanf("%d", &colonna);

    propaga(M, rows, cols, riga, colonna);
    stampaMatrice(M, rows, cols);

    return 0;
}
```

See You Next Time!

Tutte il materiale sarà disponibile sul mio sito internet!

alessandronacci.it