Programmazione I - Laboratorio

Esercitazione 3 - Array

Gianluca Mezzetti[1] Paolo Milazzo[2]

1. Dipartimento di Informatica, Università di Pisa
http://www.di.unipi.it/∼mezzetti
mezzetti@di.unipi.it

2. Dipartimento di Informatica, Università di Pisa
http://www.di.unipi.it/∼milazzo
milazzo@di.unipi.it

Corso di Laurea in Informatica
A.A. 2012/2013

Array (1)

Un array (o vettore) consente di memorizzare tanti valori dello stesso tipo in un'unica semplice struttura dati

Un array (di interi) di dimensione 10 si dichiara come segue

```
int a[10];
```

Tale dichiarazione alloca un'area di memoria pari alla dimensione di 10 interi.

Con la notazione `a[i]` si può far riferimento all'`i`-esimo elemento dell'array

Gli indici degli array vanno da 0 alla dimensione-1, quindi identifichiamo i 10 elementi di a con le seguenti espressioni

`a[0]` `a[1]` `a[2]` `a[3]` `a[4]` `a[5]` `a[6]` `a[7]` `a[8]` `a[9]`

Array (2)

I singoli elementi degli array possono essere acceduti ed assegnati come normali variabili.

Ad esempio, il seguente programma inizializza un array a di 10 elementi con i numeri da 1 a 10 e poi copia il contenuto di a nell'array b che ha la stessa dimensione

```
int main()
{
  /* dichiaro due array di interi di dimensione 10 */
  int a[10], b[10];
  int i;

  /* inizializzo a */
  for (i=0; i<10; i++)
    a[i] = i;

  /* copio a in b */
  for (i=0; i<10; i++)
    b[i] = a[i];

  return 0;
}
```

Array (3)

ATTENZIONE!!!

Molto spesso (soprattuto quando si usano cicli per scorrere gli elementi di un array) può capitare di commettere l'errore di accedere un array usando un indice negativo o maggiore della dimensione dell'array.

Questo errore non viene controllato dal compilatore! In fase di esecuzione il programma accedere ad aree di memoria che non hanno nulla a che fare con l'array (altre variabili, aree di memoria non allocate, ecc...) con effetti impredicibili.

Esempio di errore nell'uso dell'indice:

```
int a[10];
int i;

for (i=0; i<=10; i++)   /* errore <= */
  a[i] = i;
```

questo frammento di codice accede ad a[10] che è fuori dell'area di memoria allocata per l'array

Array bidimensionali (matrici)

E' anche possibile definire array bidimensionali (array di array, o matrici) come nel seguente esempio

```
/*
 Tabella delle moltiplicazioni
*/
#include <stdio.h>

int main()
{
  /* dichiarazione di matrice 10x10*/
  int moltiplicazioni[10][10];
  int i,j;

  /* inizializzazione */
  for (i=0; i<10; i++)
    for (j=0; j<10; j++)
      moltiplicazioni[i][j] = (i+1)*(j+1);

        /* stampa lineare*/
  for (i=0; i<10; i++)
  {
    for (j=0; j<100; j++)
      printf("%3d ", moltiplicazioni[i]);
    printf("\n");
  }

  /* stampa */
  for (i=0; i<10; i++)
```

Esempio svolto - array_casuale.c

Scrivere un programma che crei casualmente un array di 23 interi, lo stampi a video, lo modifichi azzerando tutti gli elementi in posizione di indice pari e quindi stampi l'array modificato.

- Usare la funzione `rand()` della libreria standard (includendo `stdlib.h`) che genera interi casuali

Esempio svolto - array_casuale.c I

```c
#include <stdio.h>
#include <stdlib.h> /* per usare rand() e srand()*/
#include <time.h> /* per usare time() */

#define SIZE 23

int main()
{
  int a[SIZE];
  int i;

  /* stampa il massimo numero generabile da rand() */
  /*  printf("%d\n",RAND_MAX); */

  /* inizializza il generatore di numeri casuali
     usando l'orologio di sistema */
  srand(time(0));

  for (i=0; i<SIZE; i++)
  {
    a[i] = rand();
    printf("a[%d] = %d\n",i,a[i]);
  }

  for (i=0; i<SIZE; i=i+2)
    a[i] = 0;
```

Esempio svolto - array_casuale.c II

```
  for (i=0; i<SIZE; i++)
    printf("a[%d] = %d\n",i,a[i]);

}
```

Esercizio - array_ordinato.c

Scrivere un programma che crei un array di 7 interi forniti dall'utente, controlli se l'array e' ordinato in ordine strettamente descrescente e stampi a video sia l'array che il risultato ("E' ordinato" oppure "Non e' ordinato: X" dove X e' la posizione del primo elemento fuori ordine).

Esercizio - array_ordinato.c I

```
#include <stdio.h>

#define DIM 7

int main()
{
  int a[DIM];
  int i;
  int ordinato=1;

  /* inizializzo a */
  for (i=0; i<DIM; i++)
  {
    printf("inserisci a[%d]: ",i);
    scanf("%d",&a[i]);
  }

  /* stampo a */
  for (i=0; i<DIM; i++)
    printf("a[%d]=%d\n",i ,a[i]);

  /* controllo se a e' ordinato */
  i=0;
  while ((i<DIM-1)&&(ordinato)) /* nota DIM-1 */
  {
    if (a[i]<=a[i+1]) ordinato=0;
    i++;
```

Esercizio - array_ordinato.c II

```
  }

  /* stampo il messaggio di output */
  if (ordinato) printf("E' ordinato\n");
  else printf("Non e' ordinato %d\n",i);

  return 0;
}
```

Esercizio - array_min_min.c

Scrivere un programma che crei un array di 9 interi forniti dall'utente e determini (se c'e') la posizione del primo elemento che e' minore della somma degli elementi che lo precedono e la posizione del primo elemento che e' minore della somma degli elementi che lo seguono.

Esercizio - array_min_min.c I

```
/*
 Calcolo di elemento minore della somma dei precedenti e seguenti
*/
#include <stdio.h>

#define DIM 9

int main()
{
  int i;
  int a[DIM];
  int somma;
  int posizione;

  for (i=0; i<DIM; i++)
  {
    printf("Inserisci a[%d]: ",i);
    scanf("%d",&a[i]);
  }

  i=0;
  somma=0;
  posizione=-1;
  while ((i<DIM)&&(posizione<0))
  {
    if (a[i]<somma) posizione=i;
    else
```

Esercizio - array_min_min.c II

```
    {
      somma = somma+a[i];
      i=i+1;
    }
  }

  if (posizione>=0)
  {
    printf("Primo elemento minore della somma dei precedenti ");
    printf("in posizione %d\n",posizione);
  }
  else
  {
    printf("Elemento minore della somma dei precedenti ");
    printf("non trovato\n");
  }

  somma=0;
  posizione=-1;
  for (i=DIM-1; i>=0; i--)
  {
    if (a[i]<somma) posizione=i;
    somma=somma+a[i];
  }

  if (posizione>=0)
  {
```

Esercizio - array_min_min.c III

```
    printf("Primo elemento minore della somma dei successivi ");
    printf("in posizione %d\n",posizione);
  }
  else
  {
    printf("Elemento minore della somma dei successivi ");
    printf("non trovato\n");
  }

  return 0;

}
```

Esercizio - ordinamento.c

Scrivere un programma che crei casualmente un array di 13 interi, lo stampi a video, lo ordini con ordinamento crescente e quindi lo stampi a video nuovamente.

Esercizio - ordinamento.c I

```
#include <stdio.h>
#include <stdlib.h> /* per usare rand() e srand() */
#include <time.h> /* per usare time() */

#define DIM 13

int main()
{
  int a[13];
  int i,j;

  /* inizializza il generatore di numeri pseudo-casuali */
  srand(time(0));

  /* inizializza e stampa l'array */
  for (i=0; i<DIM; i++)
  {
    a[i] = rand();
    printf("a[%d] = %d\n",i,a[i]);
  }

  /* cicla per ogni posizione dell'array (tranne l'ultima) */
  for (i=0; i<DIM-1; i++)
  {
    int pos_minimo=i;
    int tmp;
```

Esercizio - ordinamento.c II

```
    /* seleziona il minimo tra i successori di a[i] */
    for (j=i+1; j<DIM; j++)
    {
      if (a[j]<a[pos_minimo])
        pos_minimo = j;
    }

    /* mette il minimo in posizione i */
    if (i!=pos_minimo)
    {
      tmp = a[i];
      a[i]=a[pos_minimo];
      a[pos_minimo]=tmp;
    }
  }

  /* stampa l'array ordinato */
  printf("ARRAY ORDINATO:\n");
  for (i=0; i<DIM; i++)
    printf("a[%d] = %d\n",i,a[i]);


}
```

Esercizio - prodotto_vettori.c

Scrivere un programma che chieda all'utente due array di 3 elementi e quindi calcoli la matrice 3 per 3 risultato del prodotto riga per colonna dei due array. Il programma dovra' quindi stampare a video la matrice e la sua diagonale.

Esercizio - prodotto_vettori.c I

```c
#include <stdio.h>

#define DIM 3

int main()
{
  int a[DIM],b[DIM],m[DIM][DIM];
  int i,j;

  /* legge il primo array */
  for (i=0; i<DIM; i++)
  {
    printf("a[%d] = ",i);
    scanf("%d",&a[i]);
  }

  /* legge il secondo array */
  for (i=0; i<DIM; i++)
  {
    printf("b[%d] = ",i);
    scanf("%d",&b[i]);
  }

  /* calcola il prodotto riga per colonna */
  for (i=0; i<DIM; i++)
    for (j=0; j<DIM; j++)
      m[i][j] = a[i]*b[j];
```

Esercizio - prodotto_vettori.c II

```
  /* stampa il risultato */
  printf("MATRICE:\n");
  for (i=0; i<DIM; i++)
  {
    for (j=0; j<DIM; j++)
      printf(" %6d ",m[i][j]);
    printf("\n");
  }

  /* stampa la diagonale */
  printf("DIAGONALE:\n");
  for (i=0; i<DIM; i++)
    printf("%d\n",m[i][i]);
}
```

Il Crivello di Eratostene è una tecnica per calcolare i numeri primi.

- Si disegna una tabella 10x10 con i numeri da 1 a 100 ordinati
- I numeri possono essere **liberi** o **cancellati**.
- Inizialmente tutti i numeri sono liberi, eccetto l'1 che è **cancellato** (non essendo numero primo, per definizione)
- Si sceglie il primo numero **libero** (il 2, al primo giro) e si cancellano tutti i suoi multipli (4,6,8,...)
- Si sceglie il successivo numero **libero** (questa volta il 3) e si cancellano tutti i suoi multipli che non siano già stati cancellati (9,15,21,...)
- Si procede in questo modo fino alla fine della tabella
- Alla fine i numeri non cancellati saranno tutti e soli i numeri primi tra 0 e 100

~~1~~	2	3	~~4~~	5	~~6~~	7	~~8~~	~~9~~	~~10~~
11	~~12~~	13	~~14~~	~~15~~	~~16~~	17	~~18~~	19	~~20~~
~~21~~	~~22~~	23	~~24~~	~~25~~	~~26~~	~~27~~	~~28~~	29	~~30~~
31	~~32~~	~~33~~	~~34~~	~~35~~	~~36~~	37	~~38~~	~~39~~	~~40~~
41	~~42~~	43	~~44~~	~~45~~	~~46~~	47	~~48~~	~~49~~	~~50~~
~~51~~	~~52~~	53	~~54~~	~~55~~	~~56~~	~~57~~	~~58~~	59	~~60~~
61	~~62~~	~~63~~	~~64~~	~~65~~	~~66~~	67	~~68~~	~~69~~	~~70~~
71	~~72~~	73	~~74~~	~~75~~	~~76~~	~~77~~	~~78~~	79	~~80~~
~~81~~	~~82~~	83	~~84~~	~~85~~	~~86~~	~~87~~	~~88~~	89	~~90~~
~~91~~	~~92~~	~~93~~	~~94~~	~~95~~	~~96~~	97	~~98~~	~~99~~	~~100~~

Esercizio - crivello.c

Implementare il Crivello di Eratostene usando una matrice 10x10 e stampare la matrice risultante al termine dell'esecuzione in forma tabellare