

Introduzione al C

Corso di Fondamenti di Informatica
Ingegneria Clinica

Esercitazione 8

DIPARTIMENTO DI INFORMATICA
E SISTEMISTICA ANTONIO RUBERTI



SAPIENZA
UNIVERSITÀ DI ROMA

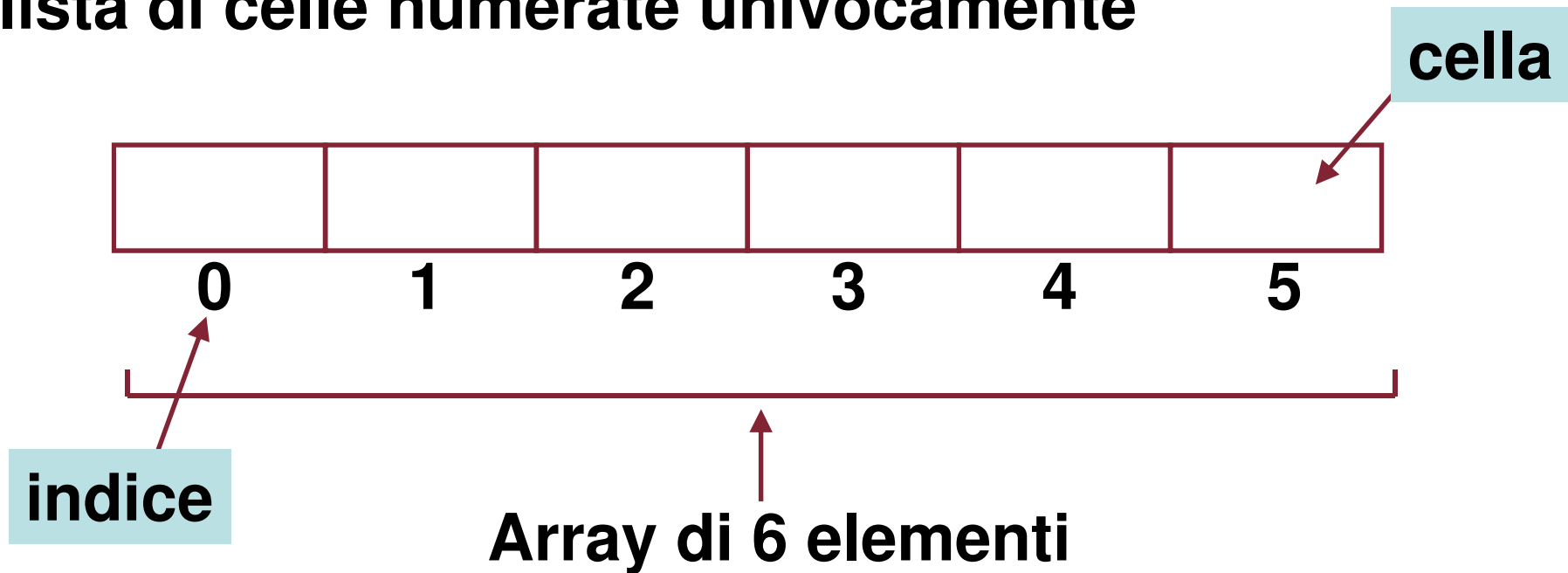
Domenico Daniele Bloisi
Raffaele Nicolussi

Menù di oggi

Vettori

Cos'è un vettore

Un vettore, o meglio, un array è un insieme ordinato di elementi, tutti di uno stesso tipo. Può essere visualizzato idealmente come una lista di celle numerate univocamente



Gli Arrays

Un esempio di definizione di un array in C e' :

tipo → **nome array** → `int elenco_numeri` **dimensione** → `[50];`

e si accede agli elementi dell'array nel seguente modo:

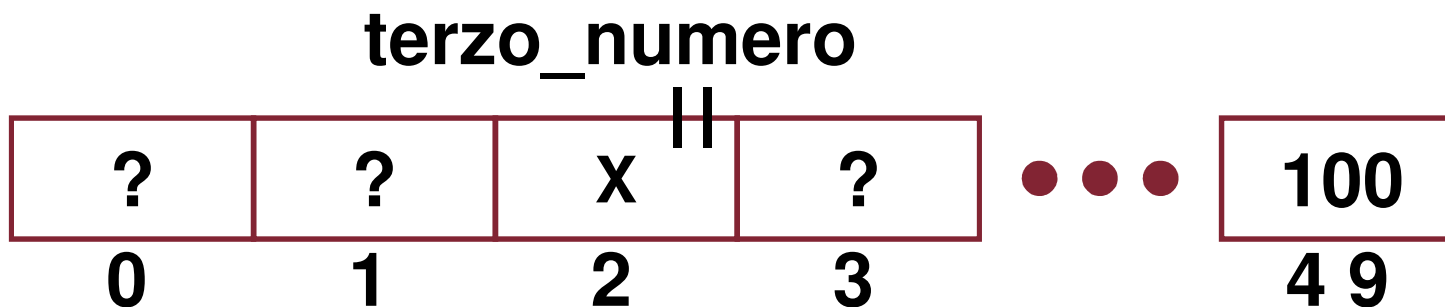
```
terzo_numero = elenco_numeri[2];  
elenco_numeri[49] = 100;
```

Gli Arrays: rappresentazione ideale

```
int elenco_numeri[50];
```



```
terzo_numero = elenco_numeri[2];  
elenco_numeri[49] = 100;
```



Indicizzazione degli array

In C gli indici delle celle iniziano da 0 e finiscono alla dimensione dell'array meno uno. Nell'esempio precedente il range è 0-49, cioè elenco_numeri è un array di 50 elementi e si ha:

elenco_numeri[0], elenco_numeri[1], ..., elenco_numeri[49]

Questa è una caratteristica tipica dei moderni linguaggi di programmazione e richiede un po' di pratica per raggiungere "la giusta disposizione d'animo".

Esempio Array

Creare un array di 10 elementi di tipo integer (`int`)

Assegnare ad esso 10 valori a piacere

Stampare il contenuto dell'array che si è creato

Soluzione Array

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
    int arrayInteri[10];
    arrayInteri[0] = 15;
    int i;
    for(i = 1; i < 10; i++)
    {
        arrayInteri[ i ] = (arrayInteri[ i - 1 ] + 10);
    }
    for(i = 0; i < 10; i++)
    {
        printf("cella %d = %d\n", i, arrayInteri[ i ]);
    }
    system("PAUSE");
}
```


Output soluzione array

cella 0 = 15

cella 1 = 25

cella 2 = 35

cella 3 = 45

cella 4 = 55

cella 5 = 65

cella 6 = 75

cella 7 = 85

cella 8 = 95

cella 9 = 105

Premere un tasto per continuare . . .

Esercizio Fibonacci

Scrivere un programma che generi i primi 12 interi della successione di Fibonacci, li salvi in un array di interi e stampi la successione a video

Qual è l'espressione analitica della successione di Fibonacci?

Help:

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89

Successione di Fibonacci

La successione di Fibonacci è una sequenza di numeri interi naturali definibile assegnando i valori dei due primi termini, $F_0 := 0$ ed $F_1 := 1$, e chiedendo che per ogni successivo sia $F_n := F_{n-1} + F_{n-2}$.

Il termine F_0 viene aggiunto nel caso si voglia fare iniziare la successione con 0; storicamente il primo termine della successione è $F_1 := 1$.

Soluzione Fibonacci

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
    int successione_Fibonacci[12];
    int f0 = 0, f1 = 1, i;
    successione_Fibonacci[0] = 0; successione_Fibonacci[1] = 1;
    for(i = 2; i < 12; i++){
        successione_Fibonacci[i] =
            successione_Fibonacci[ i - 1 ] + successione_Fibonacci[ i - 2 ];
    }

    for(i = 0; i < 12; i++){
        printf("%d\n", successione_Fibonacci[ i ]);
    }
    system("PAUSE");
}
```

Output Fibonacci

0

1

1

2

3

5

8

13

21

34

55

89

Premere un tasto per continuare . . .


Esercizio 1

Scrivere un programma per leggere da stdin dieci interi in un vettore. Individuare poi il valore minimo dell'array e l'indice corrispondente a questo elemento. Stampare i due valori individuati.

Esempio: se si inseriscono gli interi

-3 1 -2 2 4 -4 5 6 1 8

il **valore minimo** sarà **-4** e **l'indice** del valore minimo sarà **5**



Soluzione

```
#include <stdio.h>  
#include <stdlib.h>
```

```
#define DIM 10
```

```
int main( )
```

```
{
```

```
    int vet[ DIM ] , min , indice_min, i;
```

```
    printf( "Caricamento di %d elementi interi in un vettore\n", DIM ) ;
```

```
    for ( i = 0 ; i < DIM ; i++ )
```

```
    {
```

```
        printf( "Inserire elemento di indice %d: " , i ) ;
```

```
        scanf( "%d" , &vet[ i ] ) ;
```

```
    }
```



```
min = vet[0];  
indice_min = 0;
```

```
for ( i = 1; i < DIM ; i++ ) {  
    if ( vet[ i ] < min ) {  
        min = vet[i];  
        indice_min = i ;  
    }  
}
```

```
printf( "\nMinimo dell'array: %d \t Indice del valore minimo: %d\n",  
min, indice_min ) ;
```

```
system( "PAUSE" ) ;  
return 0 ;
```

```
}
```



Esercizio 2

Scrivere un programma che dopo aver letto da stdin dieci interi in un array ed un ulteriore intero **I**, stampi l'indice del primo elemento dell'array coincidente con l'intero **I**, oppure il valore **-1** se l'intero **I** non è presente nell'array

Variante: Definire la costante **MAX_DIM_V** di valore **30** e dichiarare l'array di dimensione **MAX_DIM_V**. Leggere da stdin: il numero **N** di elementi da inserire nell'array (**N** ≤ **MAX_DIM_V**), gli **N** interi, e l'intero **I** da ricercare

Soluzione

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

```
#include <stdlib.h>
```

```
int main( )
```

```
{
```

```
    int vet[ 10 ] , val , i , pos = -1 ;
```

```
    printf( "Caricamento 10 elementi interi in un vettore\n" ) ;
```

```
    for ( i = 0 ; i < 10 ; i = i+1 )
```

```
    {
```

```
        printf( "Inserire elemento di indice %d: " , i ) ;
```

```
        scanf( "%d" , &vet[ i ] ) ;
```

```
    }
```

```
    printf( "Inserire valore intero da ricercare nel vettore: " ) ;
```

```
    scanf( "%d" , &val ) ;
```

Soluzione

```
i = 0 ;
```

```
/* attenzione alle condizioni di continuazione del ciclo */
```

```
while (( i < 10 ) && ( pos == -1 ))
```

```
{
```

```
    if ( vet[ i ] == val )
```

```
        pos = i ;
```

```
    i = i+1 ;
```

```
}
```

```
printf( "\nL'indice del valore cercato e' %d\n" , pos ) ;
```

```
system( "PAUSE" ) ;
```

```
return 0 ;
```

```
}
```



Soluzione della variante

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

```
#include <stdlib.h>
```

```
#define MAX_DIM_V 30
```

```
int main( )
```

```
{
```

```
    int vet[ MAX_DIM_V ] , val , i , pos = -1, num_elementi ;
```

```
    printf("Inserire il numero di elementi del vettore: ");
```

```
    scanf("%d", &num_elementi);
```

```
    if (num_elementi <= MAX_DIM_V) {
```

```
        printf( "Caricamento %d elementi interi in un vettore\n", num_elementi ) ;
```

```
        for ( i = 0 ; i < num_elementi ; i = i+1 ) {
```

```
            printf( "Inserire elemento di indice %d: " , i ) ;
```

```
            scanf( "%d" , &vet[ i ] ) ;
```

```
        }
```




Soluzione della variante

```
printf( "Inserire valore intero da ricercare nel vettore: " ) ;
scanf( "%d" , &val ) ;

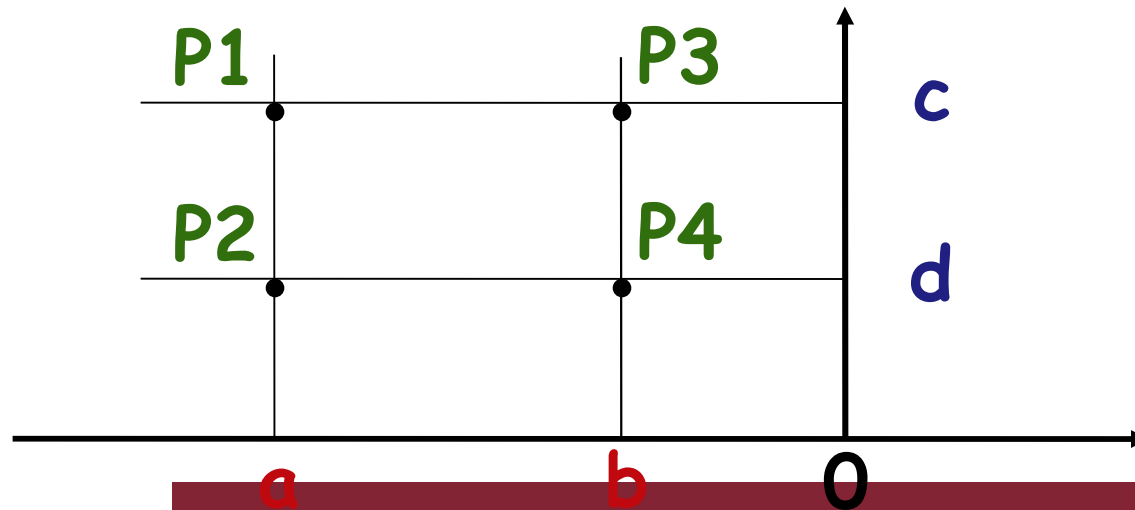

---


i = 0 ;
while (( i < num_elementi ) && ( pos == -1 )) {
    if ( vet[ i ] == val )
        pos = i ;
    i = i+1 ;
}
printf( "\nL'indice del valore cercato e' %d\n" , pos ) ;
}
else
    printf("Il numero di elementi deve essere <= %d\n",
MAX_DIM_V);
system( "PAUSE" ) ;
return 0 ;
}
```



Esercizio 3

Scrivere un programma che, dopo avere letto in un array 4 valori reali a , b , c , d , calcoli l'area del rettangolo di punti $P1$, $P2$, $P3$, $P4$, di coordinate (a, c) , (a, d) , (b, c) , (b, d) e stampi su stdout l'area del rettangolo risultante



Esempio con a
e b negativi e
 c e d positivi

Soluzione

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

```
#include <stdlib.h>
```

```
/* prototipo della funzione area */
```

```
float area( float [ 4 ] ) ;
```

```
int main( )
```

```
{
```

```
    float coordinate[ 4 ], base , altezza ;
```

```
    /* acquisizione valori nell'array */
```

```
    printf( "Inserire a (ascissa di P1 e di P2)\n" ) ;
```

```
    scanf( "%f" , &coordinate[ 0 ] ) ;
```

```
    printf( "Inserire b (ascissa di P3 e di P4)\n" ) ;
```

```
    scanf( "%f" , &coordinate[ 1 ] ) ;
```

```
    printf( "Inserire c (ordinata di P1 e di P3)\n" ) ;
```

```
    scanf( "%f" , &coordinate[ 2 ] ) ;
```

Soluzione

```
printf( "Inserire d (ordinata di P2 e di P4)\n" ) ;  
scanf( "%f" , &coordinate[ 3 ] ) ;
```

```
base = coordinate[ 0 ] - coordinate[ 1 ] ;  
altezza = coordinate[ 2 ] - coordinate[ 3 ] ;
```

```
if ( base < 0 )  
    base = - base ;
```

```
if ( altezza < 0 )  
    altezza = - altezza ;
```

```
printf( "\nL'area del rettangolo individuato da P1, P2, P3 e P4 e' %f \n" ,  
    base*altezza) ;
```

```
system( "PAUSE" ) ;
```

```
return 0 ;
```

```
}
```



Esercizio 4

Scrivere un programma che, dopo avere letto da standard input 10 caratteri da memorizzare in un array, verifichi se la sequenza di caratteri letta è palindroma (è la stessa letta da destra o da sinistra). Stampare un messaggio con il risultato.

Esempio:

se l'input è **ramo44omar** il messaggio stampato sarà **"la sequenza è palindroma"**



Soluzione

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

```
#include <stdlib.h>
```

```
#define NUM_CHAR 10
```

```
int main( )
```

```
{
```

```
    char vetcar[ NUM_CHAR ], carattere;
```

```
    int i, palindroma = 1;
```

```
    printf( "Inserire %d caratteri: ", NUM_CHAR ) ;
```

```
    for ( i = 0; i < NUM_CHAR; i++)
```

```
        scanf("%c", &vetcar[i]);
```

```
    for ( i = 0; i < NUM_CHAR/2; i++)
```

```
        if (vetcar[i] != vetcar[NUM_CHAR -1 -i])
```

```
            palindroma = 0;
```

Soluzione

```
if (palindroma == 1 )
```

```
    printf ("La sequenza di caratteri e' palindroma\n");
```

```
else
```

```
    printf ("La sequenza di caratteri non e' palindroma\n");
```

```
system( "PAUSE" ) ;
```

```
return 0 ;
```

```
}
```



Esercizio 5

Progettare una funzione che, ricevuti:

1. un array **A** di interi
2. un array **B** di interi
3. un intero **d** indicante la dimensione di **A** e **B**
4. un intero **I** da cercare in **A**,

restituisca un valore intero pari al numero di occorrenze di **I** in **A** ed inserisca in **B**, a partire dalla 1a posizione, le posizioni in cui **I** compare in **A**

ESEMPI:

Se $d=8$, $A = [0, 1, 0, 10, 1, 1, 1, 0]$, $I=0$, sarà
 $B = [0, 2, 7, -, -, -, -, -]$ e verrà restituito 3

Se $d=8$, $A = [0, 1, 0, 10, 1, 1, 1, 0]$, $I=-1$, sarà
 $B = [-, -, -, -, -, -, -, -]$ e verrà restituito 0

Inserire la funzione all'interno di un programma contenente quanto necessario per verificare se il comportamento della funzione è corretto

Soluzione

- La soluzione proposta definisce 25 come massima dimensione array.


- E' possibile configurare la dimensione degli array A e B ed il contenuto dell'array A.
- Dopo l'acquisizione (funzione "**arrayscan**") dei valori contenuti nell'array A, e l'inizializzazione degli elementi dell'array B al valore -1, si acquisisce il valore da ricercare in A.
- Vengono visualizzati alla fine: il numero di occorrenze nell'array A del valore cercato, il contenuto degli array A e B (funzione "arrayprint")
- L'inizializzazione dell'array B ad un valore negativo e la visualizzazione finale consentono di verificare le operazioni eseguite su questo dalla funzione "search"

Soluzione

```
/* definizione funzione arrayscan */  
void arrayscan( int v[ ] , unsigned int size )  


---

  
{  
    unsigned int i ;  
    /* ciclo acquisizione elementi array */  
    for( i = 0 ; i < size ; i++ )  
    {  
        printf( "Inserire elemento int di indice %u\n" , i ) ;  
        scanf( "%d" , &v[ i ] ) ;  
    }  
}
```




Soluzione

```
/* definizione funzione arrayprint*/  
void arrayprint( int v[ ] , unsigned int size )  


---

  
{  
    unsigned int i ;  
  
    /* ciclo stampa elementi array */  
    for( i = 0 ; i < size ; i++ )  
        printf( "%u\t%d\n" , i , v[ i ] ) ;  
}
```



Soluzione

```
/* definizione funzione ricerca */
```

```
unsigned int search( int v1[] , int v2[] , unsigned int size , int  
    val )
```

```
{
```

```
    unsigned int cont = 0 , i ;
```

```
    /* cont = valore restituito */
```

```
    for ( i = 0 ; i < size ; i++ )
```

```
        if ( v1[ i ] == val )
```

```
            {
```

```
                v2[ cont ] = i ;
```

```
                cont++ ;
```

```
            }
```

```
    return cont ;
```

```
}
```



Soluzione

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

```
#include <stdlib.h>
```

```
#define DIM1 25
```

```
/* prototipo funzione ricerca */
```

```
unsigned int search( int v1[] , int v2[] , unsigned int size , int val ) ;
```

```
/* prototipo funzione arrayscan */
```

```
void arrayscan( int v[ ] , unsigned int size ) ;
```

```
/* prototipo funzione arrayprint */
```

```
void arrayprint( int v[ ] , unsigned int size ) ;
```



Soluzione


```
int main( )
{
int vA[ DIM1 ] , vB[ DIM1 ] , dim , val , i ;

printf( "\nInserire dimensione comune (max %u) array A e B:\n" , DIM1 )
;
scanf( "%u" , &dim ) ;

printf( "\nAcquisizione elementi array A:\n" ) ;
arrayscan( vA , dim ) ;

/* inizializzazione array B */
for ( i = 0; i < dim; i++)
    vB[i] = -1;

printf( "\nInserire valore intero da cercare in array A:\n" ) ;
scanf( "%d" , &val ) ;
```



Soluzione

```
printf( "\nNumero di occorrenze di %d in array A e': %u\n" ,  
        val , search( vA , vB , dim , val ) ) ;
```

```
printf( "\nStampa elementi array A:\n" ) ;  
arrayprint( vA , dim ) ;
```

```
printf( "\nStampa elementi array B:\n" ) ;  
arrayprint( vB , dim ) ;
```

```
system("PAUSE") ;
```

```
return 0 ;
```

```
}
```



Per casa

- Dato un array A di interi scrivere le funzioni:
 - void carica_array (int ar[], int elem)
 - carica nell'array un numero di interi, chiesti all'utente, pari a elem
 - void stampa_array (int ar[], int elem)
 - Stampa elem elementi dell'array
 - int trova_max (int ar[] , int elem)
 - Trova e restituisce il valore massimo dell'array
 - float calcola_media (int ar[], int elem)
 - Calcola la media dei valori dell'array
 - int stampa_pari (int ar[], int elem)
 - Stampa gli elementi pari e restituisce il numero di elementi pari presenti nell'array

Per casa

- Dato anche un array B di interi di dimensioni uguali ad A scrivere le funzioni:
 - void somma_array (int A[], int B[], int elem)
 - Calcola $A = A + B$
 - int max_array (int A[], int B[], int elem)
 - Calcola il valore massimo tra i due array usando la funzione int trova_max (int ar[] , int elem)