

Array e Funzioni

Mattia Curri - m.curri8@studenti.uniba.it

Informazioni sul tutorato (ufficiali):

<https://www.uniba.it/it/ricerca/dipartimenti/informatica/orientamento-pcto-tutorato/tutorato/info-per-studenti/informazioni-per-utenza>

Materiale: <https://mattiacurri.github.io/tutorato>

Cos'è un Array?

Un array è una struttura dati che permette di memorizzare una collezione di elementi dello stesso tipo in posizioni **contigue** di memoria.

```
tipo nome_array[dimensione];
```

- **tipo**: Il tipo di dati degli elementi (int, float, char, etc.).
- **nome_array**: Identificatore dell'array.
- **dimensione**: Numero di elementi.

```
int numeri[10]; // Array di 10 interi  
float valori[5]; // Array di 5 float
```

Inizializzazione

- Inizializzazione esplicita:

```
int arr[5] = {1, 2, 3, 4, 5};
```

- Parziale:

```
int arr[5] = {1, 2}; // Gli altri elementi sono 0

// Oppure questa roba orrenda
int arr[5] = {[0]=1, [3]=4}; // arr[0]=1, arr[3]=4, altri 0
```

- Inizializzazione a 0:

```
int arr[5] = {0}; // Tutti gli elementi sono 0

// Attenzione!
#define SIZE 10 // Costante
int arr[SIZE] = {0}; // Solo se SIZE è una costante!
```

Accesso agli Elementi

- **Indice:** Gli elementi sono accessibili tramite indice, partendo da 0.
- **Sintassi:** `nome_array[indice]`

```
int arr[5] = {10, 20, 30, 40, 50};  
printf("%d\n", arr[0]); // Output: 10  
arr[2] = 35;           // Modifica il terzo elemento
```

- Usare `sizeof` per la dimensione quando non conosciuta a priori:

```
int arr[10];  
int size = sizeof(arr) / sizeof(arr[0]); // 10
```

Attenzione!

- **Out of Bounds:** Accesso oltre la dimensione causa comportamento indefinito. Il C non vi avvisa!

```
int arr[5];  
arr[5] = 10; // ERRORE: indice 5 è fuori limite
```

- Elementi non inizializzati contengono valori garbage.
- **Passaggio a funzioni:** Gli array decadono a puntatori; dimensione persa.

```
void func(int arr[]) { /* dimensione sconosciuta */ }
```

Extra

- **Array multidimensionali:** `int arr[2][3] = {{1,2,3}, {4,5,6}};`
- **Stringhe:** Array di char terminati da `'\0'`.

Cosa sono le Funzioni?

Una funzione è un blocco di codice riutilizzabile che esegue un compito specifico.

```
// Prototipo
tipo_ritorno nome_funzione(tipo_param1 param1, tipo_param2 param2, ...);

tipo_ritorno nome_funzione(tipo_param1 param1, tipo_param2 param2, ...) {
    // Istruzioni...
    return valore; // Se tipo_ritorno != void
}
```

```
// Prototipo
tipo_ritorno nome_funzione(tipo_param1 param1, tipo_param2 param2, ...);

tipo_ritorno nome_funzione(tipo_param1 param1, tipo_param2 param2, ...) {
    // Istruzioni...
    return valore; // Se tipo_ritorno != void
}
```

- **tipo_ritorno**: Tipo del valore restituito (int, void, etc.).
- **nome_funzione**: Identificatore univoco.
- **parametri**: Variabili **locali** passate alla funzione.
 - parametri **formali** vs parametri **attuali**.
 - **Formali**: Dichiarati nella definizione della funzione.
 - **Attuali**: Valori passati durante la chiamata della funzione.
- **corpo**: Blocco di istruzioni.

Esempio

```
void saluta();  
int somma(int a, int b);  
  
void saluta() {  
    printf("Ciao!\n");  
    return; // Opzionale, ma per chiarezza lo scriviamo  
}  
  
int somma(int a, int b) {  
    return a + b;  
}  
  
int main() {  
    printf("Somma: %d\n", somma(5, 3)); // Output: 8  
    saluta(); // Output: Ciao!  
    return 0;  
}
```


Passaggio di Parametri

- **Per valore:** Viene passata una copia del valore; modifiche non influenzano l'originale.
- **Per riferimento:** Usando puntatori, permette di modificare l'originale.

Esempio per valore

```
int quadrato(int num);

int quadrato(int num) {
    return num * num; // Lavora su copia, non modifica originale
}

int main() {
    int x = 5;
    int q = quadrato(x);
    printf("Quadrato: %d, x: %d\n", q, x); // Output: Quadrato: 25, x: 5
    return 0;
}
```

Esempio per riferimento

```
int main() {  
    int x;  
    printf("Inserisci un numero: ");  
    scanf("%d", &x); // scanf modifica x tramite riferimento  
    printf("Hai inserito: %d\n", x);  
    return 0;  
}
```

Esercizio 1

Dato un array di interi, trova il valore massimo e minimo contenuto.

```

#include <stdio.h>
#include <limits.h>

int main() {
    int arr[] = {3, 1, 4, 1, 5, 9, 2, 6};
    int n = sizeof(arr)/sizeof(arr[0]);
    int max = INT_MIN; // INT_MIN = valore minimo rappresentabile per un int
    int min = INT_MAX; // INT_MAX = valore massimo rappresentabile per un int
    int i = 0;

    // Qua non stiamo scomponendo il problema!
    while(i < n) {
        if(arr[i] > max) {
            max = arr[i];
        }
        if(arr[i] < min) {
            min = arr[i];
        }
        i++;
    }
    printf("Max: %d, Min: %d\n", max, min);
    return 0;
}

```

Soluzione: Valore massimo e minimo (Modulare) - 1

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

// Prototipo
int max(int a[], int n);
int min(int a[], int n);
```

Soluzione: Valore massimo e minimo (Modulare) - 2

```
int max(int a[], int n) {  
    int max = INT_MIN;  
    int i = 0;  
    while(i < n) {  
        if(a[i] > max) {  
            max = a[i];  
        }  
        i++;  
    }  
    return max;  
}
```

```
int min(int a[], int n) {  
    int min = INT_MAX;  
    int i = 0;  
    while(i < n) {  
        if(a[i] < min) {  
            min = a[i];  
        }  
        i++;  
    }  
    return min;  
}
```

Soluzione: Valore massimo e minimo (Modulare) - 3

```
int main() {  
    int arr[] = {3, 1, 4, 1, 5, 9, 2, 6};  
    int n = sizeof(arr)/sizeof(arr[0]);  
    int max_ = max(arr, n);  
    int min_ = min(arr, n);  
    printf("Max: %d, Min: %d\n", max_, min_);  
    return 0;  
}
```


Esercizio 2

Dato un array di interi, conta quanti sono pari e quanti dispari.

```
#include <stdio.h>

int main() {
    int arr[] = {3, 1, 4, 1, 5, 9, 2, 6};
    int n = sizeof(arr)/sizeof(arr[0]);
    int pari = 0;
    int dispari = 0;
    int i = 0;
    while(i < n) {
        if(arr[i] % 2 == 0) {
            pari++;
        } else {
            dispari++;
        }
        i++;
    }
    printf("Pari: %d, Dispari: %d\n", pari, dispari);
    return 0;
}
```

Esercizio 3

Sposta tutti gli elementi dell'array di una posizione verso destra (l'ultimo diventa il primo).

```
#include <stdio.h>

int main() {
    int arr[] = {1,2,3,4,5};
    int n = sizeof(arr)/sizeof(arr[0]);
    int temp = arr[n-1];
    int i = n-1;
    while(i > 0) {
        arr[i] = arr[i-1];
        i--;
    }
    arr[0] = temp;
    // Se vogliamo stampare l'array risultante
    i = 0;
    while(i < n) {
        printf("%d ", arr[i]);
        i++;
    }
    printf("\n");
    return 0;
}
```

Esercizio 4

Data una stringa, verificare che questa sia palindroma.

Una stringa è palindroma se si legge allo stesso modo da sinistra a destra e da destra a sinistra.

```

#include <stdio.h>

#define TRUE 1
#define FALSE 0

int main() {
    char parola[] = {'a', 'b', 'b', 'a'};
    int size = sizeof(parola) / sizeof(parola[0]);

    int i = 0;
    int j = size - 1;
    int palindromo = TRUE;

    // if (palindromo) è equivalente a scrivere if (palindromo == 1)
    while (i < j && palindromo) {
        if (parola[i] != parola[j]) {
            palindromo = FALSE;
        }
        i++;
        j--;
    }

    if (palindromo) {
        printf("True");
    } else{
        printf("False");
    }
}

```

Extra - 1

Esercizio 7

Implementa una funzione che cerchi un elemento specifico in un array di interi e restituisca l'indice della prima occorrenza, oppure -1 se non trovato.

Esercizio 8

Scrivi il codice per stampare la seguente matrice:

```
1 2 3
4 5 6
7 8 9
```

- Array multidimensionale
- Tip: usa due cicli annidati per righe e colonne

Extra - 2

<https://codeforces.com/problemset/problem/263/A>


```

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

int main() {
    int a[5][5];
    int x, y;

    // Leggi la matrice 5x5 in input
    int i = 0;
    while (i < 5) {
        int j = 0;
        while (j < 5) {
            scanf("%d", &a[i][j]);
            // Trova la posizione dell'1
            if (a[i][j] == 1) {
                x = i;
                y = j;
            }
            j++;
        }
        i++;
    }

    // Calcola il numero di mosse per portare l'1 al centro (2,2)
    printf("%d\n", abs(2 - x) + abs(2 - y));
    return 0;
}

```