# ELEMENTI DI PROGRAMMAZIONE E GESTIONE DATI

Ing. Giancarlo Degani

2025

# INDICE

# 4 - VETTORI E MATRICI

Ing. Giancarlo Degani

# SOLUZIONE ES. CARRELLO

```c
// Richiesta dati all'utente
printf("Inserisci il numero di oggetti nel carrello: ");
scanf("%d", &numero_oggetti);

printf("Inserisci il prezzo unitario degli oggetti: ");
scanf("%f", &prezzo_unitario);

// Calcolo del costo totale prima dell'applicazione di sconti o IVA
imponibile = numero_oggetti * prezzo_unitario;

// #region blocco_if
// Applicazione dello sconto del 10% se il numero di oggetti è maggiore di 10
if (numero_oggetti > 10) {
    sconto = imponibile * 0.10;  // Calcolo dello sconto (10% del totale)
    imponibile -= sconto;        // Aggiornamento dell'imponibile dopo lo sconto
}
```

```
Inserisci il numero di oggetti nel carrello: 12
Inserisci il prezzo unitario degli oggetti: 1000

===== Dettaglio del Carrello =====
Numero di oggetti: 12
Prezzo unitario: 1000.00
Totale prima dello sconto: 12000.00
Sconto applicato: 1200.00
Imponibile (dopo sconto): 10800.00
IVA (22%): 2376.00
Totale lordo: 13176.00
==============================
```

# SOLUZIONE ES. RADICE

```c
// Input dell'utente
printf("Inserisci un numero maggiore di 1: ");
scanf("%lf", &n);
// Controllo se il numero è valido
if (n <= 1) {
    printf("Errore: il numero deve essere maggiore di 1.\n");
    return 1; // Termina il programma con errore
}
// Inizializzazione dei limiti per il metodo di bisezione
a = 1.0;
b = n;
// Iterazione dell'algoritmo di bisezione
while ((b - a) > EPSILON) {
    x = (a + b) / 2.0;  // Calcolo del punto medio
    printf("Punto medio: %.7f\n",x);
    if (x * x > n)
        b = x;  // Se x^2 è maggiore di n, aggiorna il limite superiore
    else
        a = x;  // Altrimenti aggiorna il limite inferiore
}
// Stampa del risultato
printf("La radice quadrata approssimata di %.1f è: %.7f\n", n, (a + b) / 2.0);
```

```
Inserisci un numero maggiore di 1: 5
Punto medio: 3.0000000
Punto medio: 2.0000000
Punto medio: 2.5000000
Punto medio: 2.2500000
Punto medio: 2.1250000
Punto medio: 2.1875000
Punto medio: 2.2187500
Punto medio: 2.2360535
Punto medio: 2.2360687
Punto medio: 2.2360611
Punto medio: 2.2360649
Punto medio: 2.2360668
Punto medio: 2.2360678
Punto medio: 2.2360682
Punto medio: 2.2360680
Punto medio: 2.2360679
Punto medio: 2.2360680
La radice quadrata approssimata di 5.0 è: 2.2360680
```

# VETTORI

- Variabile Scalare: contiene 1 singolo valore:

```
tipo identificatore = valore;
int numero = 3 ;
```

- Variabili vettoriali: contengono più valori dello stesso tipo:

```
tipo identificatore [ dimensione ] = valore ;
int numeri[ 3 ] = { 0, 1, 2 };
```

- **dimensione** deve essere una costante intera, positiva, e nota al momento della compilazione
- Contengono elementi dello stesso tipo scalare ( int, double, char,... )
- L'indici e di tipo intero e non negativo
- Il primo elemento ha indice 0 (posizione)
- L'ultimo elemento ha indice N-1 ( N è la dimensione del vettore )

# VETTORI

- Gli elementi del vettore sono allocati in locazioni di memoria contigue e successive
- Si accede ai singoli elementi indicando il nome del vettore seguito dall'indice fra parentesi quadre
- Poiché ciascun elemento del vettore è del tipo indicato nella definizione, può essere utilizzato in tutti i contesti in cui si può usare una variabile di quel tipo

```
int vett[10];
scanf("%d", &vett[4]);
x = vett[4] * 5;
```

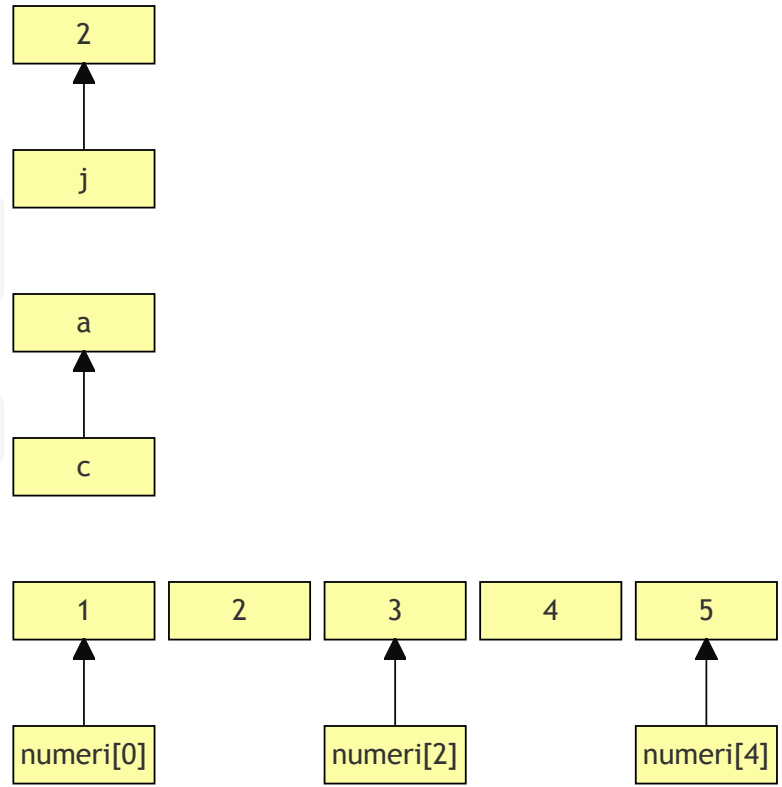
# ESEMPI

- Valori scalari

```
int j = 2
char c = ‘a’
```

- vettore di 5 elementi

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



# VETTORI

- I vettori possono essere attraversati agevolmente mediante un ciclo **for**
- Il nome di un vettore è usato dal compilatore come sinonimo dell'indirizzo di memoria del primo elemento del vettore
- Si "sfora" il vettore quando si accede a elementi oltre i limiti del vettore

```c
#define N 10
int vett[N];
for (i=0; i<N; i++){
  scanf("%d", &vett[i]);
}
for (i=N-1; i>=0; i--){
  printf("%d\n", vett[i]);
}
```

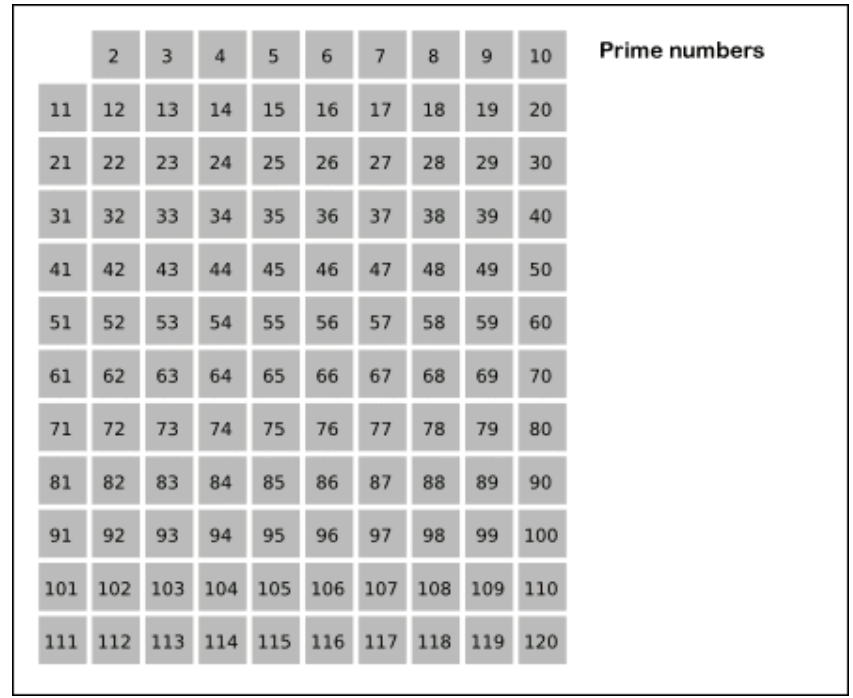
# ESERCIZI

- Scrivere un programma che chieda quanti valori verranno introdotti dalla tastiera (max 100), li chieda tutti e successivamente visualizzi prima tutti i valori pari nell'ordine in cui sono stati inseriti e poi tutti i valori dispari nell'ordine inverso. *(see example07)*
- Scrivere un programma che, dati in input N numeri reali, con N che al massimo vale 100, stampi quanti di essi sono maggiori della media e successivamente li stampi a video

# IL CRIVELLO DI ERATOSTENE

Il crivello di Eratostene è un metodo che consente di trovare i numeri primi fino ad un certo n prefissato.

- si scrivono tutti i numeri naturali a partire da 2 fino n
- si cancellano tutti i multipli del primo numero
- si passa al successivo numero non cancellato e si ripete l'operazione con i numeri che seguono




source: wikipedia.org

# ESERCIZIO

Scrivere un programma che richieda un numero n positivo e, usando il crivello di Eratostene, trovi e stampi i numeri primi minori o uguali a n *(see example08)*.

# SOLUZIONE CRIVELLO DI ERATOSTENE

```c
#include <stdio.h>
#define MAX_SIZE 1000
int main(void) {
    int n;
    printf("Enter the value of n: ");
    scanf("%d", &n);

    if ( n >= MAX_SIZE){
        return 1;
    }
    int prime[MAX_SIZE];
    for (int i = 0; i <= n; i++) {
        prime[i] = 1; // true
    }

    for (int p = 2; p * p <= n; p++) {
        if (prime[p] == 1) {
            for (int i = p * p; i <= n; i += p) {
                prime[i] = 0; // false
            }
        }
    }
```

# MATRICI

- Sono variabili vettoriali con due dimensioni
- Definizione

```
tipo identificatore [ numero_righe ] [ numero_colonne ] ;
```

- Es: matrice con 10 righe e 20 colonne:

```
int matrice [ 10 ] [ 20 ];
```

- Gli indici di riga e colonna vanno da 0 a N-1, dove N è la dimensione
- Matrice con 5 righe ( da 0 a 4) e 10 colonne (da 0 a 9)

```
int matrice [ 5 ][ 10 ];
```

# MATRICI

Come per i vettori, il ciclo **for** si presta per attraversare righe e colonne:

```
int matrice[RIGHE][COLONNE];
for (r=0; r<RIGHE; r++)
{
    for (c=0; c<COLONNE; c++)
        printf("%d ", matrice[r][c]);
    printf("\n");
}
```

# MATRICI

- Le matrici sono memorizzate in un'area di memoria contigua per righe
- La matrice m[10][20] è memorizzata come 20 vettori consecutivi di 10 elementi
- Un vettore o una matrice possono essere inizializzati elencando i valori delle singole celle della matrice o del vettore

```
int matrice [2][3] = {1,2,3,4,5,6};
```

| | **0** | **1** | **2** |
|---|---|---|---|
| **0** | 1 | 2 | 3 |
| **1** | 4 | 5 | 6 |

# MATRICI

- Non c’è limite al numero delle dimensioni

```
int matrice [DIM_1][DIM_2]...[DIM_N] ;
```

- Solitamente si usano costanti simboliche ( #define ) per definire le dimensioni dei vettori o delle matrici
- Non è possibile copiare o confrontare due generici vettori (multidimensionali) usando gli operatori = o == sui nomi dei vettori stessi

# CARATTERI

- Per memorizzare i simboli grafici corrispondenti ai caratteri bisogna associare un numero intero a ciascuno di essi
- Lo standard ASCII definisce una codifica a 7 o 8 bit
- I caratteri ASCII sono gestiti in C con variabili di tipo **char**. ovvero numeri interi ad 8 bit
- Esempio:

```
char character;
char character = 'A'; // assegnazione con caratter
char character = 65;  // assegnazione con codice ASCI decimale
char character = 0x41;  // assegnazione con codice ASCI esadeci
```

# CARATTERI

- Caratteri “speciali” sono rappresentati con le sequenze di escape, ovvero premettendo il carattere '\':
  - \’
  - \”
  - \?
  - \\
- Alcuni caratteri di controllo
  - \n - nuova linea
  - \r - ritorno a capo
  - \t - tabulazione

# STRINGHE

- Sono vettori di **char** terminate dal carattere **null**
- Null è un carattere speciale rappresentato con **\0** (Ottale) o **Ox00** (Esadecimale)
- Attenzione:

| Simbolo | Decimale | Ottale | Esadecimale |
|---|---|---|---|
| Null | 0 | \0 | 0x00 |
| '0' (zero) | 48 | \60 | 0x30 |

# STRINGHE

- Poiché le stringhe sono terminate da null, una stringa di n caratteri, richiede n+1 byte di memoria.

Esempio:

| **Char** | H | e | l | l | o | ! | **\0** |
|---|---|---|---|---|---|---|---|
| **Dec** | 72 | 101 | 108 | 108 | 111 | 33 | **0** |
| **Hex** | 48 | 65 | 6C | 6C | 6F | 21 | **0** |

# STRINGHE

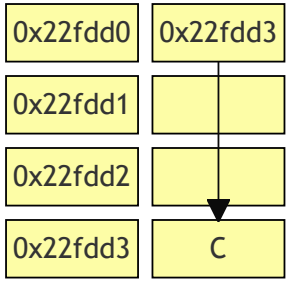
## Costanti

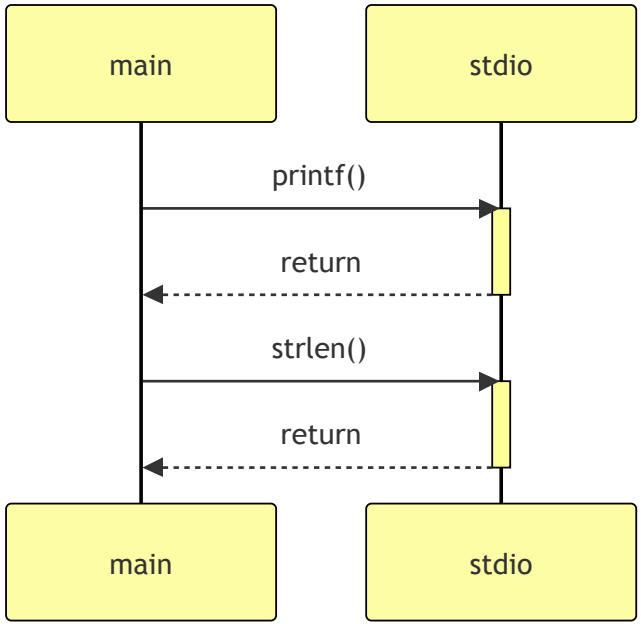
Le Stringhe costanti sono sequenze di char racchiuse da doppi apici Esempi:

- "Ciao"
- "Hello World!"

## Variabili

- Sono vettori di char di dimensione fissa (nota al compile-time), l’ultimo c


0x22fdd0
0x22fdd3
0x22fdd1
0x22fdd2
0x22fdd3
C


main
stdio
printf()
return
strlen()
return
main
stdio