Vettori, stringhe, strutture

Loris Giulivi – Nicolò Folloni

Vettori

Un insieme di variabili ordinate

I vettori (array) sono variabili che al loro interno contengono più celle di un certo tipo.

Accedere alle celle

Le varie celle sono identificate dalla loro posizione all'interno del vettore

⚠ Attenzione, la prima cella si trova in posizione **0** (si dice che gli array in C sono zero-based). Non sarà lo stesso in MATLAB.
L'ultima posizione di un vettore è dunque quella in posizione **n-1**, dove **n** è la dimensione del vettore.

```c
int primes[5] = {2,3,5,7,11};
int i;

for(i=0;i<5;i++)
{
    printf("%d\n", primes[i]);
}
```

Esercizio 1

ES1: Scrivere un programma che chiede all'utente cinque numeri interi e che li stampa in ordine inverso.

Esercizio 1 - Soluzione

```c
int vett[5];
int i;

for(i=0;i<5;i++)
{
    printf("Inserisci il %d numero:", i);
    scanf("%d", &vett[i]);
}
for(i=4;i>=0;i--)
{
    printf("%d\n", vett[i]);
}
```

Esercizio 2

ES2: Scrivere un programma che chiede all'utente cinque numeri interi tra 0 e 100 e che ne stampa il massimo, il minimo e la media.

La directive **`#define`** permette di definire valori da usare in tutto il programma.

Questo è molto utile per sostituire valori *hard-coded* con dei simboli, e permette di non dover scorrere tutto il codice quando si cambia un parametro del nostro programma.

```c
#define ARRAY_SIZE 10

void main()
{
    int vett[ARRAY_SIZE];
    int i;

    for(i=0;i<ARRAY_SIZE;i++)
    {
        printf("Inserisci il %d numero:", i);
        scanf("%d", &vett[i]);
    }
}
```

L'operatore ternario **`?:`** permette di comprimere la sintassi di un **`if-else`**

```c
y = x ? a : b
```

⟷

```c
if(x)
{
    y = a;
}
else
{
    y = b;
}
```

Esercizio 2 - Soluzione

```c
int vett[ARRAY_SIZE];
int i;

for(i=0;i<ARRAY_SIZE;i++)
{
    do
    {
        printf("Inserisci il %d numero:", i);
        scanf("%d", &vett[i]);
    }while(vett[i]<0 || vett[i]>100);
}

int max=0, min=100, accumulator = 0;
float avg;

for(i=0;i<ARRAY_SIZE;i++)
{
    max = vett[i] > max ? vett[i] : max;
    min = vett[i] < min ? vett[i] : min;
    accumulator += vett[i];
}

printf("Il massimo è: %d\n", max);
printf("Il minimo è: %d\n", min);
printf("La media è: %f\n", (float)accumulator/ARRAY_SIZE);
```

Esercizio 3 (difficile)

ES3: Scrivere un programma che chiede all'utente cinque numeri interi positivi e che li stampa in ordine decrescente.

Esercizio 3 - Soluzione

1

```c
int vett[ARRAY_SIZE];
int i,j,max,posmax;

for(i=0;i<ARRAY_SIZE;i++)
{
    do
    {
        printf("Inserisci il %d numero:", i);
        scanf("%d", &vett[i]);
    }while(vett[i]<0);
}
```

2

```c
for(i=0;i<ARRAY_SIZE;i++)
{
    max = -1;
    for(j=i;j<ARRAY_SIZE;j++)
    {
        if(vett[j]>max)
        {
            max = vett[j];
            posmax = j;
        }
    }
    int temp = vett[i];
    vett[i] = max;
    vett[posmax] = temp;
}

for(i=0;i<ARRAY_SIZE;i++)
{
    printf("%d\n", vett[i]);
}
```

Stringhe (1)

Vettori di caratteri

Le stringhe in C sono definite come vettori di caratteri.

Ogni cella del vettore contiene un carattere della stringa.

```
char stringa[100];
```

Terminatore

Poiché non possiamo sapere a priori quanto sia lunga una stringa, il carattere terminatore '**\0**' viene utilizzato per indicarne il termine.

⚠ Bisogna quindi fare attenzione che nel vettore ci sia abbastanza spazio per contenere anche questo carattere.

La stringa 'ciao' ha bisogno di un array di almeno 5 caratteri per poter essere salvata.

```
char stringa[5] = "ciao";
```

Esercizio 4

ES4: Scrivere un programma che data la stringa «stampami», salvata in un vettore di 100 caratteri, la stampi carattere per carattere.

```
void main()
{
    char stringa[100] = "stampami";

        ?

}
```

Esercizio 4 - Soluzione

```c
char stringa[100] = "stampami";
int i=0;
char temp = stringa[i];
while(temp != '\0')
{
    printf("%c\n", temp);
    i++;
    temp = stringa[i];
}
```

Esercizio 4 – Soluzione con strlen

La funzione **strlen** nella libreria **string.h** ritorna la lunghezza della stringa data.

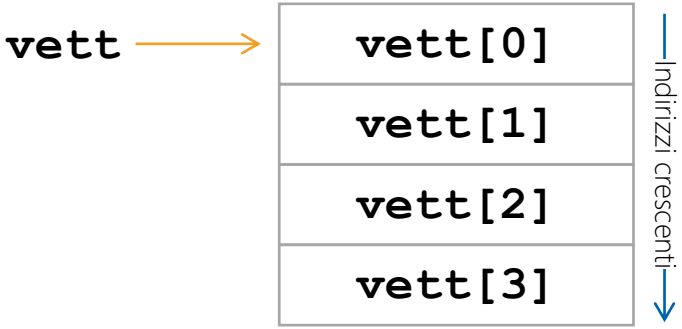
```
#include <string.h>

        ⋮

char stringa[100] = "stampami";
int len = strlen(stringa);
int i;
for(i=0;i<len;i++)
{
    printf("%c\n", stringa[i]);
}
```

Stringhe (2)

I vettori in memoria

Tutte le celle costituenti un vettore sono salvate in regioni contigue in memoria, con l'identificatore del vettore (il nome della variabile vettore) che punta alla prima cella.

Noi non lo vedremo in dettaglio, è però utile saperlo in alcuni contesti.



Scanf e vettori

Quando usiamo la scanf, abbiamo visto che l'operatore **&** viene usato per ottenere l'indirizzo di memoria della variabile operando. Poiché i vettori sono già puntatori, **&** è omesso quando la variabile è di questo tipo.

```
char stringa1[100];
char stringa2[100];

scanf("%s", stringa1);
scanf("%s", &stringa2[0]);
```

Per le stringhe, omettere **&** è equivalente a indicare l'indirizzo della prima cella del vettore.

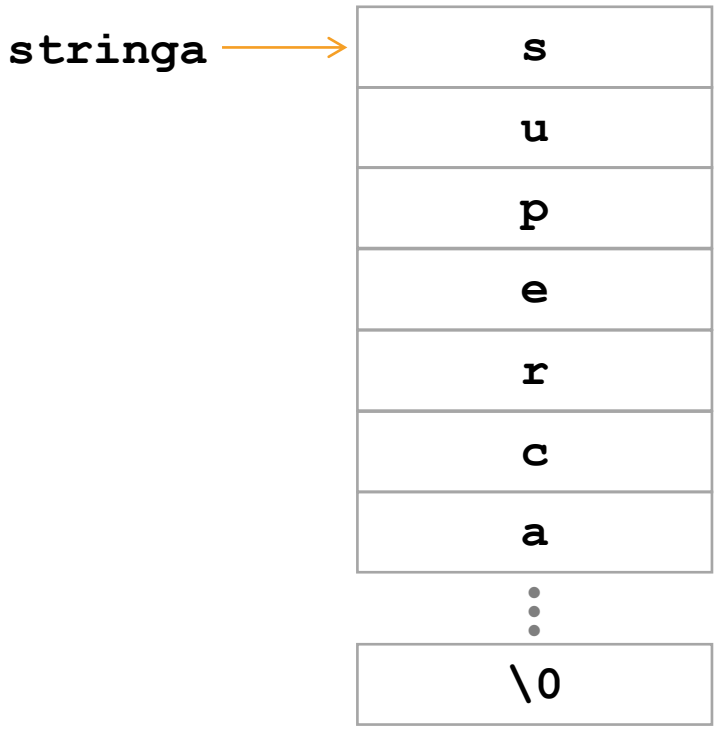
Esercizio 5

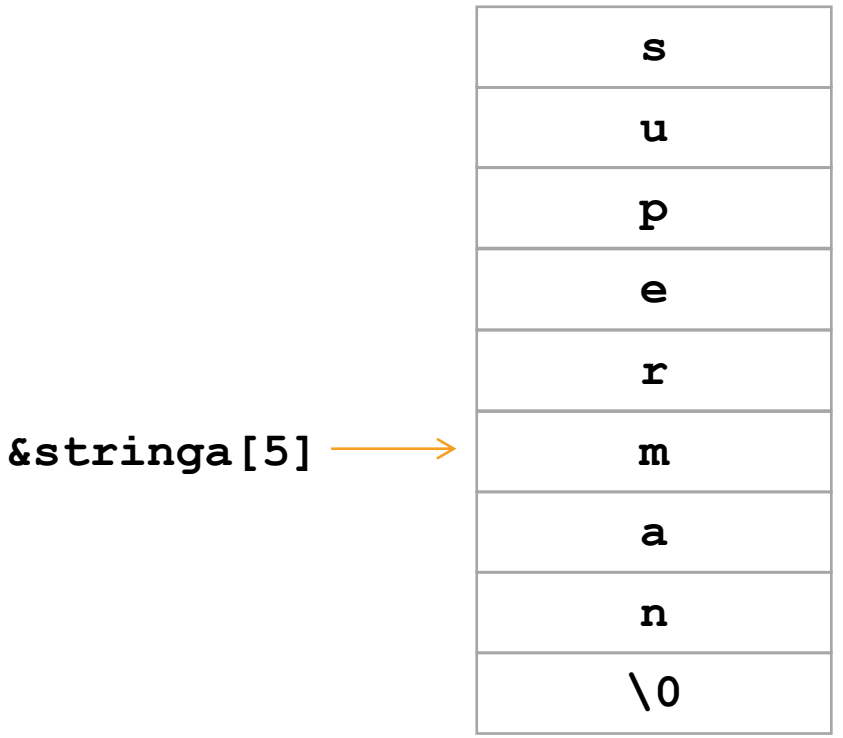
ES5: Cosa stampa questo programma?

```c
char stringa[100];

printf("Inserisci una stringa:");
scanf("%s", stringa); //VIENE INSERITO "supercalifragilistichespiralidoso"
printf("Inseriscine un'altra per buona misura:");
scanf("%s", &stringa[5]); //VIENE INSERITO "man"

printf("%s", stringa);
```

Esercizio 5 - Soluzione

```
Inserisci una stringa:supercalifragilistichespiralidoso
Inseriscine un'altra per buona misura:man
superman
```






```
scanf("%s", stringa);
```

```
scanf("%s", &stringa[5]);
```

Esercizio 5 - extra

ES5: E questa?

PROVATE VOI!!!

```
printf("%s", &stringa[9]);
```

Esercizio 6

ES6: Scrivere un programma che chiede ripetutamente un numero **`n`**. Se questo è pari, il programma stampa i primi **`n`** valori della sequenza di fibonacci. Se invece **`n`** è dispari o **`0`**, il programma si arresta.

FYI: **`fib(i) = fib(i-1)+fib(i-2)`**
con:
`fib(0) = 0`
`fib(1) = 1`

Cosa succede se **`n`** è negativo? Cambia qualcosa se è pari o dispari?

Esercizio 6 - Soluzione

```c
void main()
{
    int N;
    do
    {
        printf("Inserisci un numero intero N: ");
        scanf("%d", &N);

        if(N==0 || N%2 == 1)
        {
            printf("Fine\n");
        }
        else
        {
            int f0 = 0;
            int f1 = 1;
            int fN;
            int i;
            printf("%d\n%d\n", f0, f1);

            for(i=1;i<=N-2;i++)
            {
                fN = f0 + f1;
                printf("%d\n", fN);
                f0 = f1;
                f1 = fN;
            }
        }
    }
    while(N != 0 && N%2 == 0);
}
```

Strutture dati

Struct

Una **`struct`** contiene un insieme di variabili, di tipo potenzialmente diverso, la cui struttura è identificata al momento della definizione di tipo.

```
typedef struct studente
{
    int matricola;
    char nome[100];
    char cognome[100];
} studente;
```

Accesso agli attributi

Gli attributi delle strutture dati così definite possono essere letti/modificati con l'ausilio dell'operatore '**.**'

```
void main()
{
    studente a;
    a.matricola = 123456;
    strcpy(a.nome, "Marco");
    strcpy(a.cognome, "Rossi");
}
```

Esercizio 7

ES7: Definire una **`struct`** per contenere i dati relativi ad uno studente ed una per i dati relativi al risultato di un esame per uno studente.

Scrivere un programma che dato un insieme di risultati di esami e uno di studenti, dopo aver chiesto all'utente un numero di matricola, calcola la media degli esami dello studente.

Esercizio 7 - Soluzione

1

```c
typedef struct studente
{
    int matricola;
    char nome[100];
    char cognome[100];
} studente;

typedef struct risultato
{
    int matricola;
    char corso[100];
    int voto;
} risultato;
```

2

```c
void main()
{
    studente studenti[NUM_STUD];
    risultato risultati[NUM_RIS];
    initialize_structs(studenti, risultati);

    int mat, i, studidx=-1;
    printf("Matricola:");
    scanf("%d", &mat);

    for(i=0;i<NUM_STUD;i++)
    {
        if(studenti[i].matricola==mat)
        {
            studidx=i;
            i=NUM_STUD;
        }
    }
    if(studidx==-1)
    {
        printf("Matricola inesistente");
        return;
    }

    int n_es, somma_voti;
    for(i=0;i<NUM_RIS;i++)
    {
        if(risultati[i].matricola==mat && risultati[i].voto>=18)
        {
            n_es++;
            somma_voti += risultati[i].voto;
        }
    }

    if(n_es==0)
    {
        printf("Lo studente non ha voti!");
        return;
    }

    float media = (float)somma_voti/(float)n_es;
    printf("La media di %s %s e': %f",
           studenti[studidx].cognome,
           studenti[studidx].nome,
           media);
```

Esercizio 8

ES7: Definire una **`struct`** per contenere i dati relativi ad uno studente, identificato da nome, cognome e matricola.

Scrivere un programma che, dopo aver richiesto all'utente i dati di un nuovo studente, lo inserisca in un array di studenti in maniera che l'array sia ordinato per matricole crescenti. Il programma deve permettere l'inserimento di un numero arbitrario di studenti.
Alla fine dell'inserimento, il programma stampa tutti i dati inseriti.

Esercizio 8 - Soluzione

1

```c
typedef struct studente
{
    int matricola;
    char nome[100];
    char cognome[100];
} studente;
```

2

```c
void main()
{
    studente studenti[100];
    int inseriti = 0, mat, i, flag_ignore=0, flag_exit=0;
    char nome[100], cognome[100];
    do
    {
        flag_ignore=0;
        printf("Inserisci nuova matricola (0 per uscire): ");
        scanf("%d", &mat);
        if(mat)
        {
            int ins_idx = inseriti;
            for(i=0;i<inseriti;i++)
            {
                if(studenti[i].matricola==mat)
                {
                    printf("Matricola già esistente\n");
                    flag_ignore=1;
                }
                else if(studenti[i].matricola>mat)
                {
                    ins_idx=i;
                }
            }
            if(!flag_ignore)
            {
                printf("Nome: ");
                scanf("%s", nome);
                printf("Cognome: ");
                scanf("%s", cognome);

                for(i=inseriti;i>ins_idx;i--)
                {
                    studenti[i]=studenti[i-1];
                }

                studenti[ins_idx].matricola = mat;
                strcpy(studenti[ins_idx].nome, nome);
                strcpy(studenti[ins_idx].cognome, cognome);
                inseriti++;
            }
        }
        else
        {
            flag_exit=1;
        }
    }while(!flag_exit);

    for(i=0;i<inseriti;i++)
    {
        printf("Matricola: %d\n", studenti[i].matricola);
        printf("Nome: %s\n", studenti[i].nome);
        printf("Cognome: %s\n\n", studenti[i].cognome);
    }
}
```