Laboratorio guidato: funzioni, array

Una funzione è un blocco di istruzioni con un nome, che può ricevere dei dati in ingresso (parametri) e può restituire un risultato (valore di ritorno). Un array è una sequenza di elementi dello stesso tipo, memorizzati in posizioni contigue, accessibili tramite un indice intero che parte da 0. L'uso combinato di funzioni e array consente di costruire programmi modulari: ogni parte risolve un compito preciso, e il programma principale coordina tutto.

Il problema finale (più corposo) riguarda una piccola "analisi voti": inserire N voti (N limitato), calcolare media, minimo, massimo, contare quanti voti sono sopra una soglia, cercare un voto, e stampare i dati in modo ordinato.

1 - Funzioni: firma, ritorno, parametri

Una funzione in C ha una signature composta da tipo di ritorno, nome, lista dei parametri.

Ad esempio si consideri una funzione che calcola la somma di due interi ritorna un `int` e riceve due `int`.

```c
#include <stdio.h>

int sum(int a, int b) {
    return a + b;
}

int main(void) {
    int result = sum(10, 5);
    printf("Somma = %d\n", result);
    return 0;
}
```

Punti chiave:

- `return` conclude la funzione e restituisce il valore al chiamante.
- i parametri (`a`, `b`) sono variabili locali della funzione.

2 - Funzioni senza valore di ritorno (void)

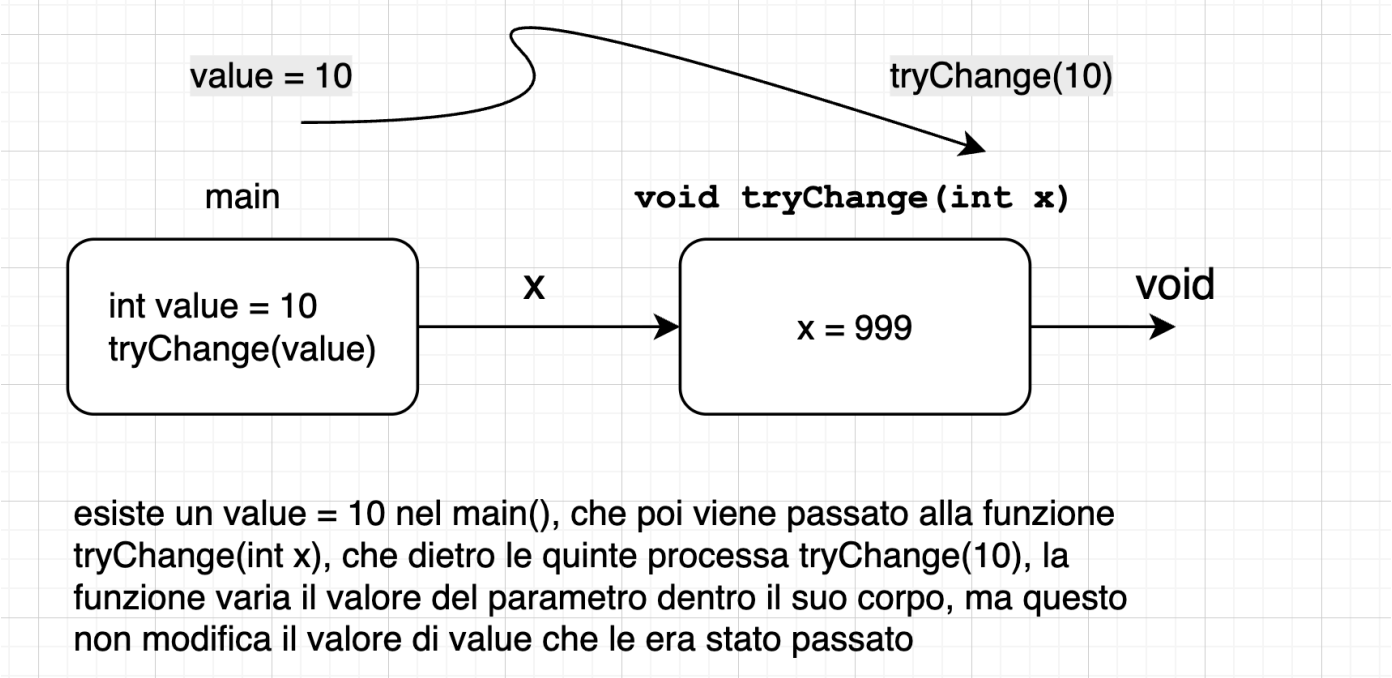
Quando una funzione deve solo eseguire azioni (per esempio stampare), il tipo di ritorno può essere `void`.

```c
#include <stdio.h>

void print_line(void) {
    printf("--------------------\n");
}

int main(void) {
    print_line();
    printf("Messaggio\n");
    print_line();
    return 0;
}
```

---

3 - Passaggio per valore: cosa significa davvero

In C, i parametri vengono passati per valore: la funzione riceve una copia dei dati. Modificare il parametro dentro la funzione non cambia la variabile originale nel `main`. In poche parole passando un valore alla funzione, e poi modificando il valore dentro la funzione, non modifica l'originale.




```c
#include <stdio.h>

void tryChange(int x) {
    x = 999;
}

int main(void) {
    int value = 10;
    tryChange(value);
    printf("value = %d\n", value);  // resta 10
    return 0;
}
```

Questa nozione diventa importante quando si lavora con gli array. Gli array si comportano”diversamente perché, quando vengono passati a una funzione, di fatto si passa l’indirizzo del primo elemento e non il suo valore.

---

4 - Array: dichiarazione, indici, lettura e stampa

Un array di interi di dimensione 5 contiene 5 elementi, indicizzati da 0 a 4.

```c
#include <stdio.h>

int main(void) {
    int numbers[5];

    numbers[0] = 10;
    numbers[1] = 20;
    numbers[2] = 30;
    numbers[3] = 40;
    numbers[4] = 50;
```

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

    return 0;
}
```

5 — Array e input: riempire un array con un ciclo

L'inserimento manuale di molti valori diventa rapidamente scomodo. Il ciclo `for` permette di ripetere la lettura.

```
#include <stdio.h>

int main(void) {
    int values[5];

    for (int i = 0; i < 5; i++) {
        printf("Inserire valore %d: ", i);
        scanf("%d", &values[i]);
    }

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

    return 0;
}
```

6 - Funzione di stampa di un array

Per evitare di riscrivere sempre lo stesso ciclo di stampa, si sposta la logica in una funzione. La funzione riceve:

- l'array
- la sua lunghezza

```
#include <stdio.h>

void print_int_array(const int array[], int length) {
    for (int i = 0; i < length; i++) {
        printf("%d ", array[i]);
    }
    printf("\n");
}

int main(void) {
    int values[5] = {10, 20, 30, 40, 50};
    print_int_array(values, 5);
    return 0;
}
```

Note tecniche:

- `const` segnala che la funzione non deve modificare gli elementi.
- il parametro `array[]` è sintassi comoda: in C equivale a un puntatore al primo elemento (`lo vedremo in aula`)

7 - Funzione con array, calcolo della media

La media richiede una somma e una divisione. Conviene usare un tipo più ampio per la somma `int` va spesso bene per voti piccoli, qui conviene usare `float`.

```c
#include <stdio.h>

double average_int_array(const int array[], int length) {
    long sum = 0;
    for (int i = 0; i < length; i++) {
        sum += array[i];
    }
    return (double)sum / (double)length;
}

int main(void) {
    int votes[4] = {18, 24, 30, 21};
    double avg = average_int_array(votes, 4);
    printf("Media = %.2f\n", avg);
    return 0;
}
```

8 - Minimo e massimo: due funzioni, stessa struttura

Minimo e massimo si ottengono scorrendo l'array e assegnando il miglior valore alle variabili scelte, nel caso specifico `min_value` e `max_value`

```c
#include <stdio.h>

int min_int_array(const int array[], int length) {
    int min_value = array[0];
    for (int i = 1; i < length; i++) {
        if (array[i] < min_value) {
            min_value = array[i];
        }
    }
    return min_value;
}

int max_int_array(const int array[], int length) {
    int max_value = array[0];
    for (int i = 1; i < length; i++) {
        if (array[i] > max_value) {
            max_value = array[i];
        }
    }
    return max_value;
}

int main(void) {
    int votes[6] = {18, 27, 30, 21, 24, 19};
    printf("Min = %d\n", min_int_array(votes, 6));
    printf("Max = %d\n", max_int_array(votes, 6));
    return 0;
}
```

9 - Contare elementi che rispettano una condizione

Una funzione può contare quanti valori sono, per esempio, maggiori o uguali a una soglia (threshold).

```c
#include <stdio.h>

int count_at_least(const int array[], int length, int threshold) {
    int count = 0;
    for (int i = 0; i < length; i++) {
        if (array[i] >= threshold) {
            count++;
        }
    }
    return count;
}

int main(void) {
    int votes[5] = {18, 22, 30, 19, 25};
    int count = count_at_least(votes, 5, 24);
    printf("Voti >= 24: %d\n", count);
    return 0;
}
```

10 - Ricerca: trovare un valore e restituire l'indice

Una ricerca lineare scorre l'array. Se trova il valore, restituisce l'indice; altrimenti restituisce -1.

```c
#include <stdio.h>

int index_of(const int array[], int length, int target) {
    for (int i = 0; i < length; i++) {
        if (array[i] == target) {
            return i;
        }
    }
    return -1;
}

int main(void) {
    int votes[5] = {18, 22, 30, 19, 25};
    int idx = index_of(votes, 5, 19);
    printf("Indice di 19 = %d\n", idx);
    return 0;
}
```

11 - Input guidato e vincoli: leggere N e poi leggere N valori

Quando si acquisisce un valore da tastiera, per avere un software ben fatto occorre definire dei vincoli.

- N deve stare in un intervallo,
- i voti devono stare in un intervallo.

Per evitare ripetizioni, si crea una funzione che legge un intero entro un range.

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

```c
int read_int_in_range(const char prompt[], int min_value, int max_value) {
    int value;

    while (1) {
        printf("%s", prompt);
        if (scanf("%d", &value) != 1) {
            int c;
            while ((c = getchar()) != '\n' && c != EOF) { }
            continue;
        }

        if (value >= min_value && value <= max_value) {
            return value;
        }
    }
}
```

Questo è il classico esempio in cui una funzione diventa un “mattoncino” riutilizzabile.

Programma risultante assemblando i pezzi precedenti

Problema completo:

- leggere N (1..MAX)
- leggere N voti (0..30)
- stampare l’array
- calcolare media, minimo, massimo
- contare voti sopra una soglia
- cercare un voto specifico

```c
#include <stdio.h>

#define MAX_VOTES 50

int read_int_in_range(const char prompt[], int min_value, int max_value) {
    int value;

    while (1) {
        printf("%s", prompt);

        if (scanf("%d", &value) != 1) {
            int c;
            while ((c = getchar()) != '\n' && c != EOF) { }
            continue;
        }

        if (value >= min_value && value <= max_value) {
            return value;
        }
    }
}

void read_votes(int votes[], int count) {
```

```
    for (int i = 0; i < count; i++) {
        char prompt[64];
        snprintf(prompt, sizeof(prompt), "Inserire voto %d (0..30): ", i);
        votes[i] = read_int_in_range(prompt, 0, 30);
    }
}

void print_int_array(const int array[], int length) {
    for (int i = 0; i < length; i++) {
        printf("%d ", array[i]);
    }
    printf("\n");
}

double average_int_array(const int array[], int length) {
    long sum = 0;
    for (int i = 0; i < length; i++) {
        sum += array[i];
    }
    return (double)sum / (double)length;
}

int min_int_array(const int array[], int length) {
    int min_value = array[0];
    for (int i = 1; i < length; i++) {
        if (array[i] < min_value) {
            min_value = array[i];
        }
    }
    return min_value;
}

int max_int_array(const int array[], int length) {
    int max_value = array[0];
    for (int i = 1; i < length; i++) {
        if (array[i] > max_value) {
            max_value = array[i];
        }
    }
    return max_value;
}

int count_at_least(const int array[], int length, int threshold) {
    int count = 0;
    for (int i = 0; i < length; i++) {
        if (array[i] >= threshold) {
            count++;
        }
    }
    return count;
}

int index_of(const int array[], int length, int target) {
    for (int i = 0; i < length; i++) {
        if (array[i] == target) {
            return i;
        }
    }
```

```c
    return -1;
}

int main(void) {
    int votes[MAX_VOTES];

    int count = read_int_in_range("Quanti voti inserire (1..50): ", 1, MAX_VOTES);
    read_votes(votes, count);

    printf("\nVoti inseriti:\n");
    print_int_array(votes, count);

    double avg = average_int_array(votes, count);
    int min_value = min_int_array(votes, count);
    int max_value = max_int_array(votes, count);

    printf("\nStatistiche:\n");
    printf("Media: %.2f\n", avg);
    printf("Minimo: %d\n", min_value);
    printf("Massimo: %d\n", max_value);

    int threshold = read_int_in_range("\nSoglia per conteggio (0..30): ", 0, 30);
    int above = count_at_least(votes, count, threshold);
    printf("Voti >= %d: %d\n", threshold, above);

    int target = read_int_in_range("\nValore da cercare (0..30): ", 0, 30);
    int idx = index_of(votes, count, target);
    if (idx >= 0) {
        printf("Trovato %d in posizione %d\n", target, idx);
    } else {
        printf("%d non presente\n", target);
    }

    return 0;
}
```