CdL in Informatica - A.A. 2024 - 2025

Programmazione 1 – Modulo 2

Lezione 3
08/10/2024

Andrea Loddo

Federico Meloni - Alessandra Perniciano - Fabio Pili

Argomenti

- La libreria math
- Il costrutto di selezione in C: if - else
  - If concatenati
  - If annidati
  - Dangling else
  - Suggerimenti
- Espressioni booleane
- Operatori booleani
- Confronti tra char

La libreria math

Librerie esterne

Il C supporta nativamente solo operazioni aritmetiche di base che non sono sufficienti per risolvere problemi concreti.

Possono essere necessarie operazioni complesse: estrazione di radice, elevamento a potenza, logaritmi, ecc.

È necessario estendere le potenzialità del linguaggio, aggiungendo funzionalità a quelle di base: si usano le librerie.

Libreria: collezione di funzioni, macro, strutture dati e costanti predisposte per essere utilizzate all'interno di un programma.

Evitano al programmatore di dover scrivere personalmente le funzioni e le strutture dati di cui ha bisogno.

Esempi:

- stdio (gestione Input/Output)
- math (operazioni matematiche avanzate)

La libreria math

`#include` seguito dal nome del file header (file di intestazione) della libreria.

Anche la libreria math offre funzioni e macro pronte all'uso. `#include <math.h>`

Esempi di MACRO di math:

- M_PI → pigreco
- M_E → numero di Nepero

Esempi di funzioni di math:

- `a = sqrt(b);` → $a = \sqrt{b}$
- `a = pow(b, 3);` → $a = b^3$
- `a = exp(b);` → $a = e^b$
- `a = abs(b);` → $a = |b|$
- `a = log(b);` → $a = \ln(b)$
- `a = log10(b);` → $a = Log(b)$
- `a = sin(b);` → $a = sin(b)$
- `a = cos(b);` → $a = cos(b)$

```
ipotenusa = sqrt((pow(cateto1, 2))+(pow(cateto2, 2))); // esempio
```

Il costrutto di selezione

If - else

Esercizio

- Scrivere un programma che legga un numero intero in input (rappresentante il voto di PR1) e, se il voto è superiore a 18, stampi "finalmente mi sono liberato di PR1!", altrimenti stampi "non mi libererò mai di PR1!".

- Possiamo realizzarlo con le istruzioni apprese finora?
  - **No**: ogni volta che si fa partire un programma si eseguono **sempre** le stesse istruzioni.
  - **Soluzione**: ci serve un modo per scegliere vie alternative di esecuzione sulla base di valutazioni dello stato.

Il costrutto di selezione

Se l'espressione viene valutata vera → fai questo

altrimenti.. → fai quest'altro

```
if (espressione booleana è vera)
{
   /* blocco di istruzioni da svolgere
      se l'espressione è vera */
} else
{
   /* blocco di istruzioni da svolgere
      se l'espressione è falsa */
}
```

Note e suggerimenti:

- Le graffe identificano un **blocco** di istruzioni per l'if o l'else
- Se l'if contiene una sola istruzione le graffe sono **opzionali**
  - consiglio: mettiamole comunque, miglioriamo la **leggibilità** del codice!
- L'else è **opzionale**

```
int voto = 0;
scanf("%d", &voto);

if( voto > 18 ){
    printf("Finalmente mi sono"
           "liberato di PR1");
} else {
    printf("Non mi libererò mai"
             "di PR1!");

}
```

If concatenati

All'interno di un **blocco** si può inserire un qualsiasi **set di istruzioni:**

- espressioni
- stampe
- assegnamenti
- **if**
- **altri costrutti che vedremo in seguito**

Quindi: è possibile **concatenare** gli if

```
int a=0, b=0;
scanf("%d", &a);
scanf("%d", &b);

if( a > b )
{
    printf("a maggiore di b");
}
else
{
   if( a < b )
   {
        printf("a minore di b");
   }
   else
   {
        printf("a uguale a b");
   }
}
```

If annidati

All'interno di un **blocco** si può inserire un qualsiasi **set di istruzioni:**

- espressioni
- stampe
- assegnamenti
- **if**
- **altri costrutti che vedremo in seguito**

Quindi: è possibile **annidare** gli if

```
int a=4, b=5, c=10;
if( b > a )
{
   if( b < c )
   {
      printf("b compreso tra a e c");
   }
   else
   {
       printf("b non compreso tra a e c");
   }
}
```

Dangling else

```
#include <stdio.h>
int main()
{
    int a=0, b=1;
    if( a == 1 )
        if( b == 1 )
            printf("a e b veri");
    else
        printf("a falso");


    return 0;
}
```

```
#include <stdio.h>
int main()
{
    int a=0, b=1;
    if( a == 1 )
    {
        if( b == 1 )
        {
            printf("a e b veri");
        }
    }
    else
    {
        printf("a falso");
    }

    return 0;
}
```

Cosa succede se eseguiamo questi due programmi, apparentemente simili?

Dangling else

- Letteralmente: "else pendente".
- Accade quando si crea ambiguità in caso di if annidati.
  - **Attenzione**: in assenza di graffe, l'else verrà logicamente "accoppiato" all'ultimo if presente nell'annidamento.
- Suggerimenti:
  - utilizzare **sempre** le parentesi graffe per eliminare ogni ambiguità sintattica
  - **indentare** il codice a dovere
  - … Farsi aiutare da CLion!

Espressioni booleane

Espressioni booleane

- Spesso è necessario verificare se una data proposizione (**espressione booleana**) è **vera** o **falsa**.
- Esempi:
  - 6 > 21 → **false**
  - 8 == 5 → **false**
  - 6 >= 6 → **true**

- Il linguaggio C fornisce gli operatori di confronto per confrontare due espressioni aritmetiche **x** e **y** e determinare se:
  - x è strettamente **maggiore** di y → **x > y**
  - x è **maggiore** o **uguale** a y → **x >= y**
  - x è **uguale** a y → **x == y**
  - x è strettamente **minore** di y → **x < y**
  - x è **minore** o **uguale** a y → **x <= y**
  - x è **diverso** da y → **x != y**

Espressioni booleane

- Ogni confronto restituisce un valore booleano (**true** o **false**)
- Nel linguaggio C **standard** sono codificati come interi:
  - true → 1
  - false → 0
- ATTENZIONE: qualsiasi intero **diverso** da 0 è valutato 1 (**true**) !

- Esiste anche il tipo **booleano** dedicato:
  - libreria **stdbool.h**
  - tipo di dato: **_Bool** oppure **bool**
  - valori attribuibili: **true**, **false**

Espressioni booleane: esempi

- **Associatività**: da sinistra verso destra.
- **Priorità**: più bassa rispetto agli operatori aritmetici.

```
float x = 5.5, y = 5;
int val;

val = x < y;       // Quanto vale val ?
val = x <= y;      // Quanto vale val ?
val = x == y;      // Quanto vale val ?
val = x > y;       // Quanto vale val ?
```

```
int a = 5;
int val;

val = 0 > a > 10;  // Quanto vale val ?
val = 6 > a > 10;  // Quanto vale val ?
val = 6 > a + 10;  // Quanto vale val ?
val = (6+a) > 10;  // Quanto vale val ?
```

Come possiamo verificare se una variabile è all'interno di un dato intervallo?

Operatori booleani

Supponiamo di volere confrontare qualcosa di più articolato:

- Es.: il valore di **n** è all'interno di un dato intervallo? → **min** < **n** < **max**

Il C offre operatori booleani che consentono di valutare più espressioni "contemporaneamente":

- Operatore binario logico **AND**: A **&&** B (A e B: **espressioni booleane**)
  - Se entrambe le espressioni A e B sono valutate true, l'espressione **A && B** verrà valutata **true**
  - Se una o entrambe le espressioni A e B sono valutate false allora l'espressione verrà valutata **false**

- Operatore binario logico **OR**: A **||** B (A e B: **espressioni booleane**)
  - Se entrambe le espressioni A e B sono valutate false, l'espressione **A || B** verrà valutata **false**
  - Se una o entrambe le espressioni A e B sono valutate **true** allora l'espressione verrà valutata **true**

- Operatore unario logico **NOT**: !A (A: **espressione booleana**)
  - Se A è valutata true allora l'espressione !A verrà valutata false.
  - Se A è valutata false allora l'espressione !A verrà valutata true.

Operatori booleani: esempi

```
int a = 5;
_Bool x = (a > 0) && (a < 10);
```

```
int a = 5;
_Bool x = (a > 0) || (a < 10);
```

```
int a = 10;
if ( (a > 0) && (a < 10) )
    printf("Ciao!");
```

```
int a = 10;
if ( (a > 0) || (a < 10) )
    printf("Ciao!");
```

```
int a = 4;
if ( (a != 0) && (a != 5) && (a != 10) )
    printf("Ciao!");
```

```
int a = 4;
if ( (a != 0) || (a != 5) || (a != 10) )
    printf("Ciao!");
```

```
int a  = -1;
_Bool x = !(a > 0);
```

```
int a = 4, b = 5;
if ( (a != 5) && !(b < a) )
    printf("Qualcosa.");
```

Tipi char e confronti

- È possibile effettuare operazioni di confronto tra variabili di tipo **char**
- Il C li traduce in confronti numerici tra codici ASCII:

```
char var = 'F';
_Bool value;
...
value = (var >= 'A' && var <= 'Z’);
```

- Ordinamento alfabetico dei caratteri:
  A < B < C < …. < Z
  a < b < c < …. < z
  0 < 1 < 2 < …. < 9
- Non è necessario ricordarsi il codice ASCII dei caratteri
  - la traduzione nel corrispondente valore intero viene effettuata a runtime.

Algebra booleana e linguaggio C

Alcune particolarità

Algebra booleana: particolarità del C

- Universamente, per convenzione dal linguaggio binario:
  - al valore di verità **falso** corrisponde il valore numerico **0**
  - al valore di verità **vero** corrisponde il valore numerico **1**

- Nel C, questa caratteristica è implementata in modo **particolare**:
  - si considera **falso** il valore numerico 0
  - si considera **vero** qualsiasi altro valore numerico

Algebra booleana: particolarità del C

```
if ( a )
{
    printf("a è vero");
}
else
{
    printf("a è falso);
}
```

Esegue la prima printf se la valutazione della variabile **a** produce un valore diverso da zero, altrimenti viene eseguita la seconda printf.

```
if ( a != 0 )
{
    printf("a è vero");
}
else
{
    printf("a è falso);
}
```

Scrittura equivalente, in forma estesa.

```
if (a && b) …

if(!a || b) …
```

Sono ammesse espressioni scritte in questo modo compatto

Algebra booleana: particolarità del C

- **ATTENZIONE!**
- Supponiamo di voler scrivere l'istruzione

```
if (a==b) …
```

- Ma per errore, scriviamo

```
if (a=b) …
```

- Cosa accade?
- Il compilatore segnala un errore?
- Come viene valutata l'espressione?

Operatori e corto-circuito

In C, gli operatori **&&** e **||** sono valutati in **corto-circuito**:

- significa che le espressioni vengono valutate fintanto che basta per stabilire se un'espressione è vera o falsa: potenzialmente non tutte.

- Esempi:

```
int a = 5, b = 7;
_Bool valore;
valore = ( (a != 0) || (b != 7) ); /* b !=7 non verrà mai valutato in quanto la prima
                                      espressione basta già a stabilire che tutta
                                      l'espressione è true */

valore = ( (a == 0) && (b != 7) ); /* b !=7 non verrà mai valutato in quanto la prima
                                      espressione basta già a stabilire che tutta
                                      l'espressione è false */
```

FINE!

Domande?

Autovalutazione

Autovalutazione: math.h

1. Con quale sintassi è possibile includere una libreria esterna nel nostro programma?
2. A cosa serve una libreria esterna?
3. Il C implementa il calcolo della potenza e della radice quadrata nell'aritmetica di base?
4. Posso utilizzare macro definite all'interno di una libreria esterna?

Autovalutazione: costrutto selezione

1. Sono in grado di scrivere il codice di un semplice if che stampi il maggiore tra due numeri interi a e b?
2. Ogni if ha sempre necessariamente un ramo del vero e uno del falso?
3. Quali istruzioni o costrutti possiamo utilizzare all'interno dei blocchi di codice all'interno dell'if?
4. Quando sono necessarie le parentesi graffe per delimitare un blocco di codice?
5. Cosa si indica con il termine "dangling else" o "else pendente"?

Autovalutazione: espressioni booleane

1. Che valori può restituire un'espressione booleana?
2. Che cosa restituisce l'espressione 5==4?
3. Quali valori interi corrispondono a true e false?
4. Come possiamo scrivere un'espressione che verifica se un certo valore contenuto in x sia compreso tra 10 e 100?
5. Sono in grado di scrivere più espressioni usando AND e OR?
6. A cosa serve l'operatore di negazione?
7. Posso utilizzare il valore di un espressione booleana in un'espressione aritmetica?
8. Ha precedenza più alta l'AND logico o l'OR logico?
9. Posso confrontare caratteri usando le espressioni booleane? In quali casi possono essere utili?

Esercizi

Esercizi sulla libreria math

1. Scrivere un programma che, conoscendo i valori dei cateti di un triangolo rettangolo, ne calcoli l'ipotenusa.
2. Scrivere un programma che, nota l'ipotenusa di un triangolo rettangolo e un altro cateto, calcoli il secondo cateto.
3. Scrivere un programma che data la misura del raggio di un cerchio, ne calcoli area e circonferenza.
4. Scrivere un programma che permetta il calcolo dell'area e del perimetro di un triangolo rettangolo dichiarando e inizializzando due variabili chiamate **cateto1** e **cateto2**. Stampare poi tutti i valori noti e tutti quelli calcolati.
5. Scrivere un programma che, dato il lato, calcoli perimetro e area di un quadrato.

Esercizi su costrutto selezione

Esercizio 1

Scrivere un programma che acquisisca due valori interi in input ed esegua la divisione tra essi.

Nel caso in cui il secondo valore sia 0, il programma deve restituire un messaggio di errore.

Esercizio 2

Scrivere un programma che chieda all'utente di inserire:

- a quanti anni si può prendere la patente nello stato in cui vive (Esempio: Italia 18 anni, UK: 17, ecc.)
- l'età dell'utente.

Il programma deve comunicare in output se l'utente può prendere la patente.

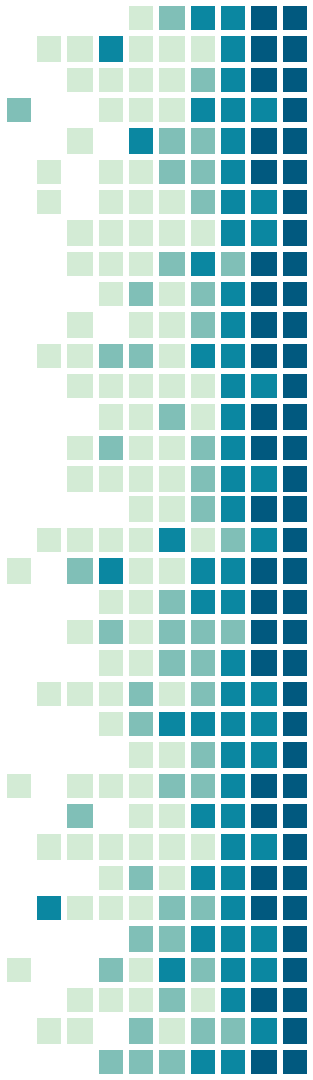
Esercizio 3

- Scrivere un programma che chieda all'utente un valore intero e lo memorizzi in una variabile numero.
- Il programma deve stampare un messaggio che indichi se numero è pari o dispari.

- Suggerimento: matematicamente, come verificare se un numero è pari o dispari?

Esercizio 4

- Scrivere un programma in cui vengono acquisiti due numeri dall'utente.
- Il programma deve dire se il primo è multiplo del secondo, oppure stampare un messaggio per comunicare che il primo non è multiplo del secondo.



Esercizio 5

Scrivere un programma che acquisisca un valore intero **n**.

Il programma deve poi:

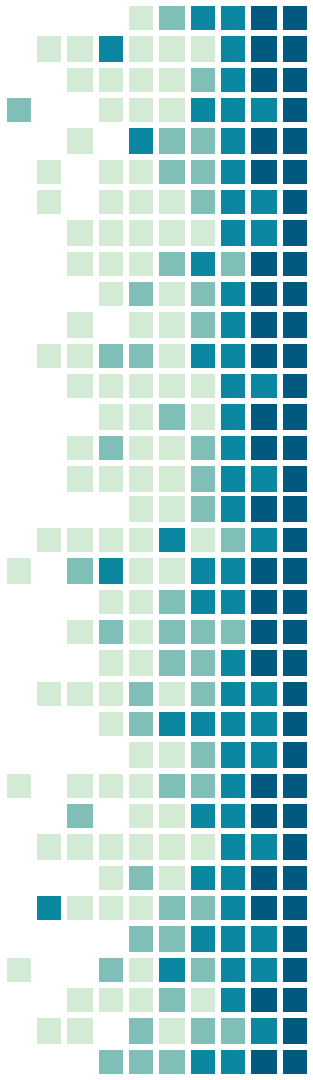
- Stampare il numero inserito in minuscolo se **1 ≤ n ≤ 9**
- Stampare "Maggiore di 9" se **n > 9**
- Stampare "Errore" in tutti gli altri casi.

Esempio:

- n = 8 -> "otto"
- n = 11 -> "Maggiore di 9"
- n = -5 -> "Errore"

Esercizio 6

- Scrivere un programma che chieda tre numeri da tastiera e li visualizzi in ordine decrescente.
- Modificare l'esercizio per far sì che li visualizzi anche in ordine crescente.



Esercizi 7 e 8

- Scrivere un programma che permetta l’inserimento delle età di tre persone.
- Successivamente stampare a video l’età relativa a ogni persona e indicare l'età del più anziano.

- Modificare l'esercizio per far sì che stampi anche l'età del più giovane.

Esercizio 9

- Scrivere un programma per dividere gli allievi di un corso in tre squadre denominate ROSSA, VERDE e BLU secondo il loro numero di matricola.
- L'assegnazione avviene con il seguente criterio: l'allievo con matricola 1 va nella squadra ROSSA, quello con matricola 2 nella VERDE, quello con matricola 3 nella BLU, quello con matricola 4 nella ROSSA, quello con 5 nella VERDE, ecc.
- Il programma deve chiedere il numero di matricola dell'allievo e indicare a quale squadra è assegnato.