Scuola di Ingegneria Industriale

Laurea in Ingegneria Energetica

Laurea in Ingegneria Meccanica

POLITECNICO DI MILANO

Dipartimento di Elettronica, Informazione e Bioingegneria





Informatica B

Sezione D

Franchi Alessio Mauro, PhD *alessiomauro.franchi@polimi.it*

Breve ripasso

Abbiamo un problema, ne scriviamo un algoritmo che lo risolva, e ora?

↓

Traduco il mio algoritmo in un programma (in C)

Cos'è un programma?

- La **codifica di un algoritmo** in un certo linguaggio di programmazione
- Una **sequenza ordinata di istruzioni**, espresse secondo un insieme di regole noto a priori, che a partire da dei dati in **ingresso** restituisce dei risultati in **uscita** in seguito alla loro elaborazione o **manipolazione** da parte dell'hardware della macchina

Breve ripasso

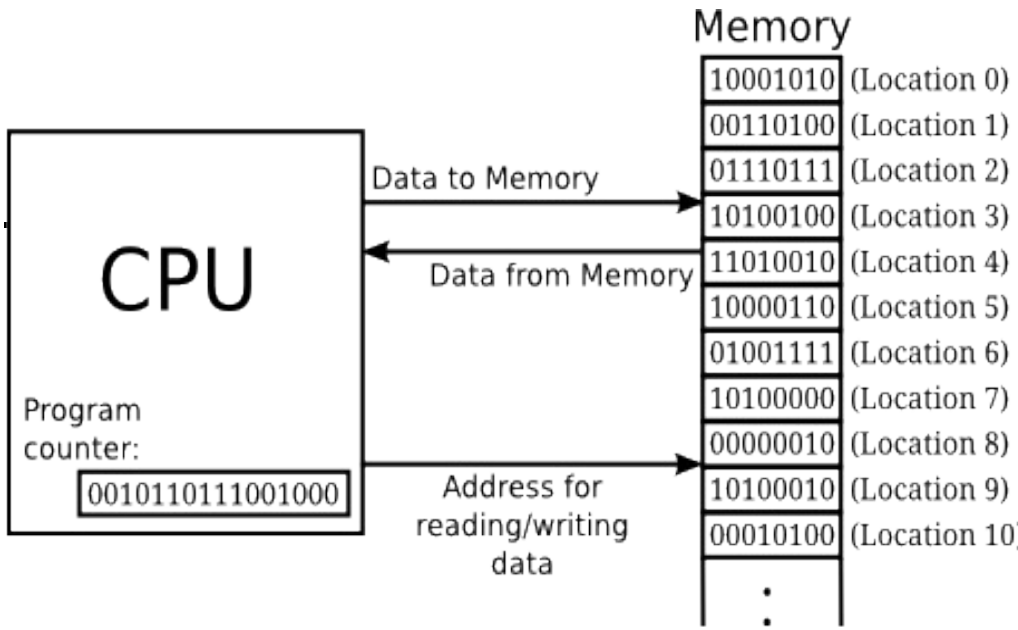
```c
int fattoriale(int n)
{
  if (n < 0) return -1;


  if (n == 0) return 1;
  else return n*fattoriale(n-1);
}
```

C è un linguaggio di programmazione di **alto livello!**

Il programmatore lo legge (facilmente).
Il computer? NO!

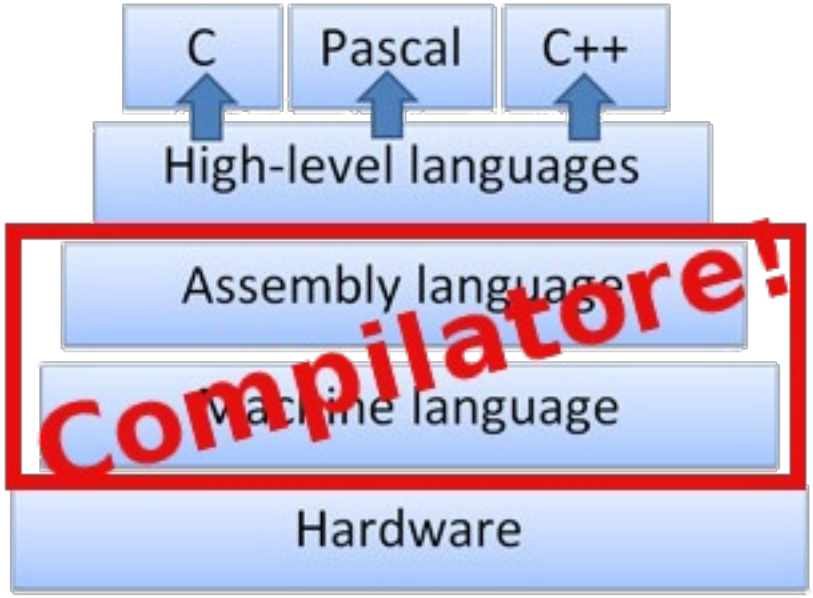
Il computer è in grado solo di eseguire operazioni aritmetico/logiche su numeri binari!



Breve ripasso

Qualcuno deve tradurre il mio programma in C in qualcosa capibile dal computer!

Il compilatore!

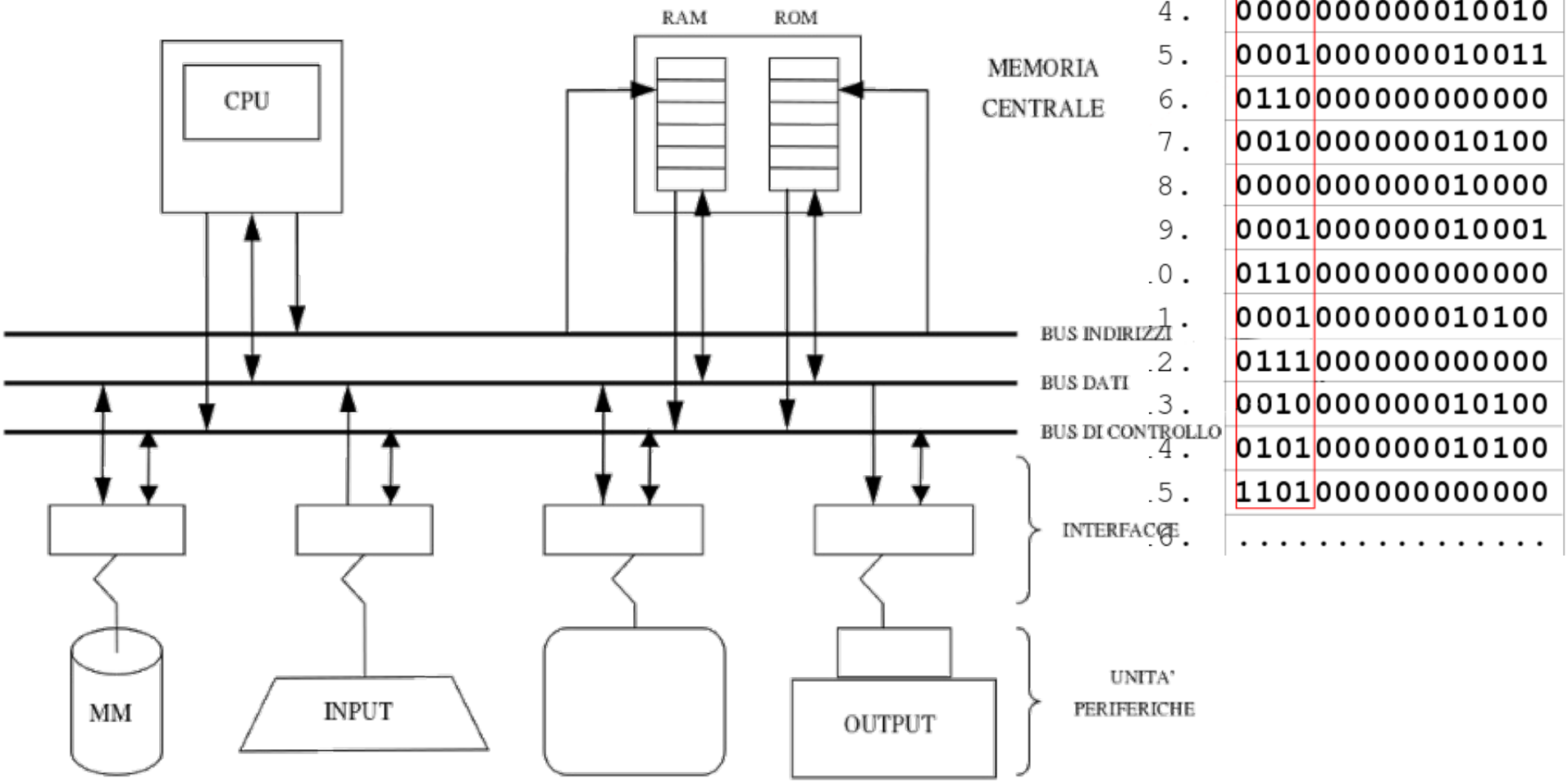


Il compilatore è un programma che **traduce** tutte le istruzioni di un programma dal linguaggio di alto livello usato dal programmatore al linguaggio macchina

Attenzione: se il computer cambia, cambia il linguaggio macchina e quindi anche il compilatore cambia!

Breve ripasso

Il nostro programma è compilato.
Come viene eseguito dal computer?




```
0.  0100000000010000
1.  0100000000010001
2.  0100000000010010
3.  0100000000010011
4.  0000000000010010
5.  0001000000010011
6.  0110000000000000
7.  0010000000010100
8.  0000000000010000
9.  0001000000010001
.0. 0110000000000000
.1. 0001000000010100
.2. 0111000000000000
.3. 0010000000010100
.4. 0101000000010100
.5. 1101000000000000
.6. ................
```

Breve ripasso

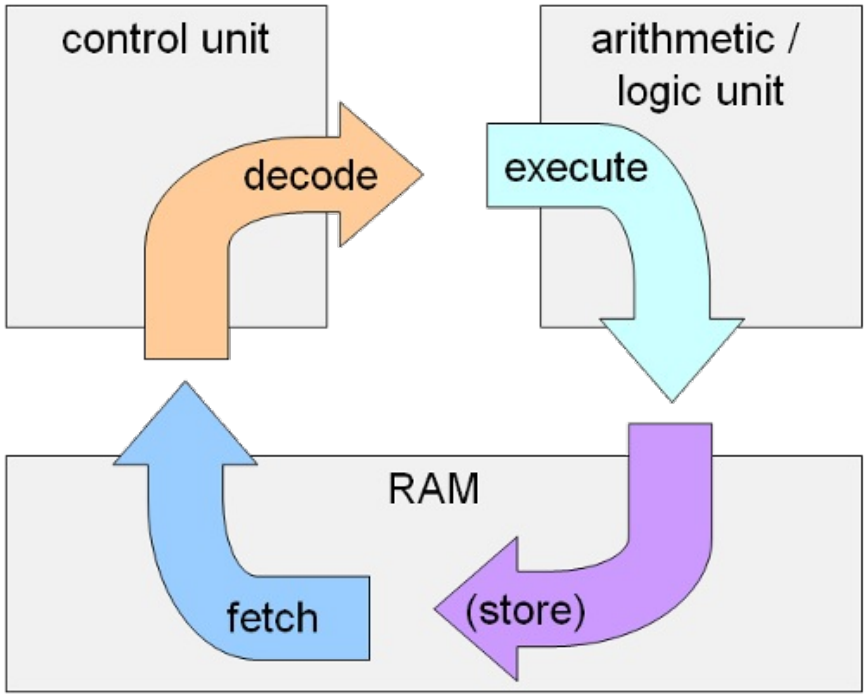
La memoria RAM contiene le istruzioni ed (eventualmente) i dati.

Ciclicamente:

1. La CPU l**egge la prossima istruzione** dalla memoria (Program Counter)
2. La salva in un registro interno speciale
3. La **decodifica** (capisce cosa fa quell'istruzione)
4. La **esegue**:

   a) Istruzione aritmetica: carica operandi, esegue, scrive risultati

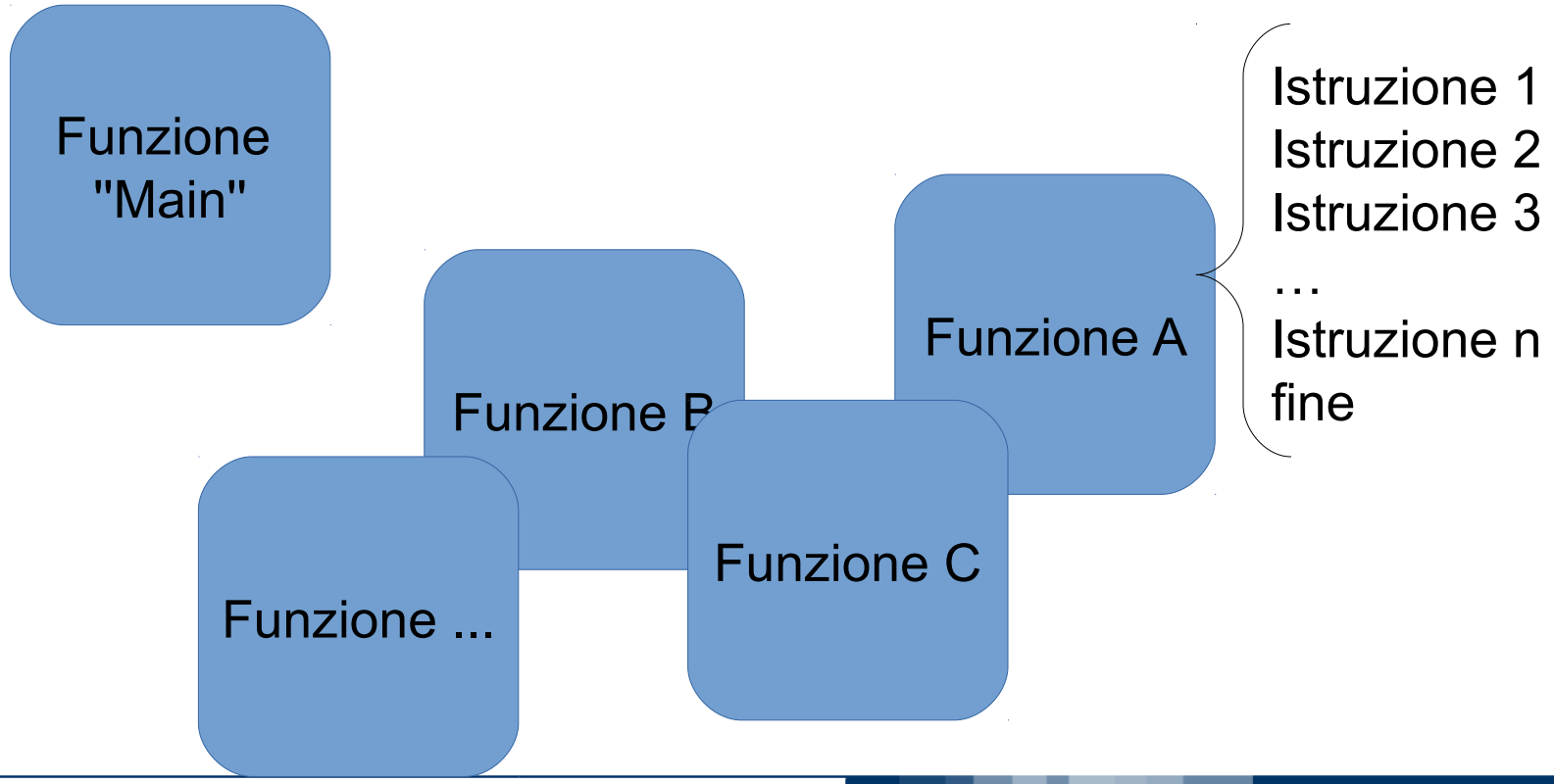
   b) Istruzione di salto: modifica PC



Il linguaggio C

Un programma in C è composto da più "funzioni"

- Una o più funzioni costituisco un programma;
- Una funzione ha un input (opzionale) ed un output(opzionale);
- Una funzione esegue una particolare azione;
- Obbligatoria la funzione "Main", punto di inizio del programma



Il linguaggio C

```
#include <stdio.h>
#define MAXNUM 100

/*      Questo è un commento multilinea
Usatelo per spiegare cose lunghe
USATE I COMMENTI!
*/

int main()
{
        int i; // Dichiarazione di i

        // Inizio il ciclo
        for (i = 1; i <= MAXNUM; i++)
        {
                printf("%3d)\n", i);
        }

        return 0;
}
```

Dichiarazioni per processore: preceduta da #

Commenti: fondamentali. Fateli! Servono per far capire agli altri cosa state facendo!

Corpo della funzione principale

Compilare un programma

Quando avete terminato di scrivere il vostro programma, dovete compilarlo!

“Se compila, funziona”. NO!

Se il vostro programma viene compilato senza errori, vuol dire che avete scritto il codice senza errori! Potete eseguirlo.

Adesso però va verificato che il programma faccia ciò che deve:

1. Il programma funziona e fa quello che deve: Ottimo!
2. Il programma funziona, ma l'output non è corretto: Male!
3. Il programma smette di funzionare: Molto male!
4. Il programma non termina mai: Ancora peggio!

Ricordatevi sempre:

Il codice C è una sequenza di tante istruzioni. Ogni istruzione deve essere terminata con un **punto e virgola ";"**

Perdete qualche minuto in più ma scrivete codice **leggibile:**

- **Non mettete più istruzioni sulla stessa riga: 1 riga = 1 istruzione!**
- **Indentate correttamente il codice**
- **Utilizzate nomi di variabili e funzioni intelligenti! (evitate a,b,c..)**
- **Ogni funzione deve fare una e una sola "cosa"**
- **Commentate le parti di codice più complicate**

Esempio...

```
#include <stdio.h>
int main()
{
        int array[100];
        int maximum;
        int size;
        int c;
        int location;
        printf("Enter the number of elements\n");
        scanf("%d", &size);
        printf("Enter %d integers\n", size);
        for (c = 0; c < size; c++)
                scanf("%d", &array[c]);
        maximum = array[0];
        for (c = 1; c < size; c++)
        {
                if (array[c] > maximum)
                {
                        maximum  = array[c];
                location = c+1;
                }
        }
...
```

Codice leggibile

```
#include <stdio.h>
int main()
{
int array[100], maximum, size, c, location;
printf("Enter the number of elements in array\n");
scanf("%d", &size);
printf("Enter %d integers\n", size);
for (c = 0; c < size; c++)
scanf("%d", &array[c]);
maximum = array[0];
for (c = 1; c < size; c++){
if (array[c] > maximum)
{
maximum  = array[c];
location = c+1;
}
}
...
```

Codice illeggibile

Esercizio 1

Scrivere un programma che stampi a video la stringa “Hello world! It's 2014!”

Esercizio 1 - Soluzione

#include <stdio.h>

Questa istruzione è necessaria; serve per dire al computer (compilatore) che utilizzeremo delle funzioni definite nella libreria "stdio.h".

Libreria = file contenente una serie di funzioni.

#define YEAR 2014

La funzione define serve per dichiarare una costante; da ora in poi, quando nel codice scriveremo la parola YEAR il compilatore la sostiuirà con il numero 2014.
Attenzione: usate il carattere maiuscolo per le costanti per questioni di leggibilità

```
int main()
{
...corpo della funzione...
}
```

Questa è la **definizione** della funziona main; il programma comincia dalla prima istruzione di questa funzione. Il corpo della funzione è un insieme di istruzioni.

Esercizio 1 - Soluzione

printf("Hello World! It's %d! \n", YEAR);

L'instruzione "printf" stampa a video; è definita nella libreria "stdio.h" (che avevamo incluso all'inizio.

La funzione printf("..."); stampa una stringa a video (stringa = insieme di caratteri) con formattazione.

Due "parametri" in ingresso:

1. la stringa da stampare che può contenere n **"placeholders"** (segnaposto) con indicazioni sulla formattazione: %d
2. le **n variabili** da stampare, seguendo l'ordine indicato dai placeholders

printf("Mi chiamo %s e ho %d anni. \n", nome, anni);

return 0;

Termine della funzione main; la funzione main deve restituire un valore (è il suo output). Lo restituisce al computer; per convenzione:

return 0; significa che la funzione è terminata correttamente

return -1; significa che la funzione ha provocato un errore

Esercizio 1 - Soluzione

printf(“.... %d”,variabile1);

Come formattare correttamente l'output usando la printf:

%d : per scrivere numeri "**int**", interi (1,2,3,...,420,...)
%f : per scrivere numeri "**float**", decimali (0.01,0.02,...12.23,12,24...)
%c : per scrivere un singolo “**char**”, carattere; ('a','b','c',...)
%s : per scrivere una “**string**”, stringa; ('ciao','computer','bicicletta')

Questa è la formattazione standard; si può modificare (ad esempio se vogliamo visualizzare una sola cifra dopo la virgola)

%10d : scrive il numero intero con 10 cifre; se il numero è più corto aggiunge davanti al numero degli zeri

printf ("Stampa numero: %10d %010d", 1977, 1977);

Output: Stampa numero: 1977 0000001977

printf ("%s %10s", “Indice”, “Valore”);

Output: Indice Valore (Inserisce 10 spazi bianchi prima di “Valore”

%x : scrive il numero intero in esadecimale;

%o: scrive il numero in ottale;

printf ("Basi differenti: %d %x %o %#x %#o", 100, 100, 100, 100, 100);

Output: Basi differenti:100 64 144 0x64 0144

Esercizio 2

Scrivere un programma che legga quattro variabili intere (a, b, c, d) e le stampi a video su quattro righe differenti.

Esercizio 2 - Soluzione

Dobbiamo ancora effettuare operazioni di input e output, quindi:

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

Procediamo poi con la scrittura della funzione main()

```
int a;
int b;
int c;
int d;
```

Queste quattro istruzioni definiscono le variabili necessarie: il computer sa che da ora in poi avrà in memoria questi quattro numeri interi (**tipo**) con questi quattro nomi (**identificatore**). **Le variabili si definiscono sempre prima del loro utilizzo!**

```
printf("Inserisci il primo numero: ");
scanf("%d",&a);
```

Chiedo all'utente di inserire il primo numero tramite la printf() e poi aspetto l'input da tastiera con l'istruzione scanf()

Esercizio 2 - Soluzione

scanf(“%d”,&a);

- La funzione "scanf(“%d”,&a);" **legge gli input da tastiera**; l'esecuzione del programma si ferma in attesa che l'utente digiti qualcosa; si dice che la scanf è **bloccante**!
  Possiamo richiedere all'utente numeri, caratteri, stringhe, etc.
- Come per la printf(...) si usano i **placeholder** (%d, %f, %c, etc...)
- Il numero inserito da tastiera viene **salvato nella variabile** specificata dopo la virgola (in questo caso "a")
- **Importante**: è necessario aggiunge il carattere "&" prima del nome della variabile!

Esercizio 2 - Soluzione

Visto che il problema richiedeva l'inserimento di quattro variabili intere, ripeto le due operazioni altre tre volte

```
printf("Inserisci il secondo numero: ");
scanf("%d",&b);
printf("Inserisci il terzo numero: ");
scanf("%d",&c);
printf("Inserisci il quarto numero: ");
scanf("%d",&d);
```

Infine stampo a video i valori inseriti:

```
printf("Il primo numero inserito è %d;\n",a);
printf("Il secondo numero inserito è %d;\n",b);
printf("Il terzo numero inserito è %d;\n",c);
printf("Il quarto numero inserito è %d;\n",d);
```

Tutto si è concluso correttamente, quindi:

```
return 0;
```

Esercizio 3

Scrivere un programma che, dato un numero intero inserito da tastiera e corrispondente al raggio di un cerchio, ne calcoli perimetro e circonferenza; stampare a video i risultati.

Esercizio 3 - Soluzione

Analizziamo solo le parti nuove:

```
...
perimeter = 2 * 3.14f * radius;
area = radius*radius*3.14f;
...
```

Questo è un esempio di **assegnamento** di variabili:

- La variabile a sinistra dell'uguale diverrà uguale al valore dell'espressione a destra (oppure uguale a valore della variabile specificata a destra)
- Se utilizzate nel vostro codice numeri decimale aggiungete il **carattere 'f'**!

```
printf("Il perimetro del cerchio è %.3f cm\n",perimeter);
```

Notate la formattazione **%.3f**: voglio che numero decimale sia scritto con sole 3 cifre dopo la virgola!

Esercizio 4 e 5

4) Scrivere un programma che, dati due numeri interi inseriti da tastiera e corrispondenti rispettivamente a base ed altezza di un rettangolo, ne calcoli area e perimetro; stampare a video i risultati

5) Scrivere un programma che legga tre numeri interi e ne esegua prima l'addizione e poi la differenza, stampando a video le singole operazioni numeriche con il risultato

Esercizio 6

Scrivere un programma che, dati due numeri interi inseriti da tastiera, calcoli la differenza tra il maggiore ed il minore stampando a video il risultato; successivamente stabilire se il risultato ottenuto è un numero pari o dispari e stampare la risposta a video.

Esercizio 6 - Soluzione

Tralasciamo le parti già affrontate; il problema richiede di stabile quale sia il minore e il maggiore tra I due numeri inseriti. Come procediamo?

```
...
if(a>b)
{
        difference = a-b;
}
else
{
        difference = b-a;
}
...
```

Usiamo **l'esecuzione condizionale**.

Esercizio 6 - Soluzione

L'esecuzione condizionale ci permette di eseguire alcune istruzioni solo se una certa **condizione** è verificata; per condizione si intende una operazione di confronto (tra due variabili, tra una variabile ed un valore...)

Sintassi:

```
if ( condizione )
{
        ... Se la (le) condizione(i) è (sono) vera(e)...
}
else
{
        ... altrimenti (se la condizone non è vera)
}
```

Con questa struttura I due blocchi di codice sono **eseguiti in modo esclusivo**, o il primo o il secondo. Mai entrambi!

Esercizio 6 - Soluzione

Operatori di confronto

var1 **<** var2	var1 è **minore di** var2?
var1 **<=** var2	var1 è **minore o uguale di** var2?
var1 **>** var2	var1 è **maggiore di** var2?
var1 **>=** var2	var1 è **maggiore o uguale di** var2?
var1 **==** var2	var1 è **uguale a** var2?
var1 **!=** var2	var1 è **diverso da** var2?

Attenzione:

- “x = 29” è un assegnamento. La variabile x assume il valore 29.
- “x == 29” non è un assegnamento, ma valuta l’uguaglianza dei due termini.

Esercizio 6 - Soluzione

Come posso specificare **condizioni più complesse?**

Esempio: Se a è maggiore di b ed è divisibile per due allora...

Si utilizzano i classici **operatori Booleani**!

&& → AND

|| → OR

! → NOT

→

Precendenze: prima NOT, poi AND, poi OR

Esempio: verificare se 50 < x ≤ 65

x deve essere contemporaneamente maggiore di 50 e minore di 65

if(x > 50 && x <= 65)

Esempio: verificare se una persona ha diritto a sconti sullo skipass. L'età deve essere minore di 6 o maggiore di 65

if (x < 6 || x >65)

if (3) … ? **if (true) ...** ? **if (-1) …** ? **if (0) ...** ?

Esercizio 6 - Soluzione

Calcolata la differenza dobbiamo vedere se è pari o dispari.

```
...
resto = difference%2;
if (resto == 0)
{
        printf("La differenza é pari!\n");
}
else
{
        printf("La differenza é dispari!\n");
}
...
```

Abbiamo usato **l'operatore "%"**: calcola il resto della divisione tra due numeri.

Torniamo indietro, esercizio 3

Scrivere un programma che, dato un numero intero inserito da tastiera e corrispondente al raggio di un cerchio, ne calcoli perimetro e circonferenza; stampare a video i risultati.

```
printf("Inserisci il raggio del cerchio in cm: ");
scanf("%d",&radius);
if (radius >= 0)
{
    perimeter = 2*3.14f*radius;
    area = 3.14*radius*radius;

    printf("Il perimetro del cerchio è %.3f cm\n",perimeter);
    printf("L'area del cerchio è %.3f cm\n",area);
}
else
    printf("Valore inserito non accettabile!(Negativo)\n");

return 0;
```

Ricordatevi sempre di verificare gli input da tastiera!!!

Cosa succede se l'utente digita un numero negativo? ERRORE

Immaginate di dover fare una divisione; cosa succede se l'utente digita 0? ERRORE!

Torniamo indietro, esercizio 3

```
printf("Inserisci il raggio del cerchio in cm: ");
scanf("%d",&radius);
if (radius >= 0)
{
   perimeter = 2*3.14f*radius;
   area = 3.14*radius*radius;

   printf("Il perimetro del cerchio è %.3f cm\n",perimeter);
   printf("L'area del cerchio è %.3f cm\n",area);
}
else
   printf("Valore inserito non accettabile!(Negativo)\n");

return 0;
```

Dovete **gestire tutti i possibili casi**! Altrimenti l'esecuzione non è predicibile!
Gestire l'I/O non è banale, va progettato a priori!

Esercizio 7

Scrivere un programma che, dati tre numeri interi inseriti da tastiera, li stampi a video disponendoli prima in modo crescente e poi decrescente.

Esercizio 7 – Soluzione 1

Un esempio pratico di uso degli operatori "&&"

```
if (a>b && b>c)
        printf("%d - %d - %d\n",a,b,c);
else if (a>c && c>b)
        printf("%d - %d - %d\n",a,c,b);
else if (b>a && a>c)
        printf("%d - %d - %d\n",b,a,c);
else if (b>c && c>a)
        printf("%d - %d - %d\n",b,c,a);
else if (c>a && a>b)
        printf("%d - %d - %d\n",c,a,b);
else
        printf("%d - %d - %d\n",c,b,a);
```

Notate

- L'**assenza di parentesi graffe**: se il blocco di codice è composto da una sola istruzioneele parentesi non sono necessarie;
- La struttura "i**f – else if – else**": sono una delle tante viene eseguita!

Esercizio 7 – Soluzione 2

Senza l'uso di "&&" dobbiamo usare if annidati

```
if (a >= b){
        if (a >= c)
        {
                primo = a;
                if (b >= c)
                {
                        secondo = b;
                        terzo = c;
                }
                Else
                {
                        secondo = c;
                        terzo = b;
                }
                ...
```

Attenzione alle parentesi graffe e all'indentazione del codice!

Esercizio 8 e 9

8) Scrivere un programma che legga tre numeri interi inseriti da tastiera e dica se l'addizione dei tre è un numero pari o dispari.

9) Scrivere un programma che chieda in ingresso un anno e dica se è bisestile o no *[suggerimento: un anno è bisestile se è multiplo di 4 ma non di 100 oppure multiplo di 400]*

Esercizio 10

Scrivere un programma che chieda all'utente di inserire numeri interi da tastiera e li ristampi immediatamente a video; il programma termina quando l'utente inserisce uno zero (lo zero non deve essere stampato a video).

Esercizio 10 - Soluzione

Il programma deve continuare la sua esecuzione fino a quando l'utente digita il "codice di uscita" (lo 0)

Iterazione, il ciclo while

```
...
while(num!=0)
{
        printf("Hai inserito %d.\n",num);
        printf("Inserisci il prossimo numero: ");
        scanf("%d",&num);
}
...
```

Condizione di permanenza

Questa struttura ripete l'esecuzione di una o più istruzioni fintantochè la **condizione di permanenza** indicate è vera!

La condizione è **verificata prima** di eseguire il blocco di istruzioni!

Esercizio 10 bis

Avete scritto un ciclo while: siete sicuri che termini sempre?

```
int max, odd = 0;
 printf("Inserici il massimo: ");
 scanf("%d",&max);

 while ( odd != max)
 {
    printf("%d\n", odd);
    odd += 2;
 }
 return 0;
```

Cosa fa questo programma?
Termina sempre?

Esercizio 11

Scrivere un programma che chieda all'utente di inserire numeri interi e li ristampi immediatamente a video quando l'utente inserisce un numero dispari maggiore della somma di tutti i precedenti.

Esercizio 11 - Soluzione

Anche in questo caso serve un ciclo. Qual'è la **condizione di permanenza**?

```
printf("Inserici un numero: ");
scanf("%d",&num);
while (num <= sum || num%2==0)
{
        sum += num;
        printf("Inserici un altro numero: ");
        scanf("%d",&num);

}
```

significa “sum = sum + num;”

Il problema chiede che il ciclo termini quando il numero inserito è dispari e maggiore della somma dei precedenti.

Quindi il ciclo deve continuare fino a che il numero è minore od uguale alla somma dei precedenti oppure pari!

Il ciclo termina solo quando entrambe le condizioni sono false (numero maggiore della somma e dispari!)

Altri esercizi...

12) Scrivere un programma che prenda in ingresso un numero intero e svolga le seguenti operazioni:

- Se il numero inserito è pari scriva a schermo "NUMERO PARI";
- Altrimenti, se il numero inserito è dispari ed è divisibile per 3 scriva "NUMERO DISPARI DIVISIBILE PER 3";
- Negli altri casi si stampi "NUMERO NON RICONOSCIUTO".

13) Acquisire un numero *n* diverso da 0, e una sequenza di numeri; l'inserimento della sequenza termina quando l'utente inserisce il valore 0. Sommare i divisori pari di *n* presenti nella sequenza, e stampare a video il risultato.

14)