Lezione 1
Le basi del linguaggio C

Simone Marchesini
Roberto Pagliarini

Dipartimento di Informatica
Università di Verona

Cos'è?

- Il C è un linguaggio di programmazione che permette di salvare i valori in variabili, di strutturare il codice, di convogliare il flusso del programma utilizzando istruzioni di ciclo, istruzioni condizionali e funzioni, di eseguire operazioni di input/output a video o su file

- Permette di scrivere programmi piccoli e di facile comprensione

- Si possono sviluppare programmi di qualsiasi genere, compresi videogiochi e sistemi operativi

Struttura di un programma C

```
Parte dichiarativa globale
main()
{
      Parte dichiarativa locale
      Parte esecutiva (istruzioni)
}
funzione1 ()
{
      Parte dichiarativa locale
      Parte esecutiva (istruzioni)
}
…
funzioneN ()
{
      Parte dichiarativa locale
      Parte esecutiva (istruzioni)
}
```

FUNZIONI

Una **funzione** è un costrutto che permette di raggruppare una sequenza di istruzioni in un unico blocco che fanno parte di un programma. Una funzione può essere "chiamata" in diversi punti del programma di cui fa parte come se fosse una singola istruzione.

Funzione definita **prima** del main

```
double f(int x)
{
  …
}
int main ()
{
  …
}
```

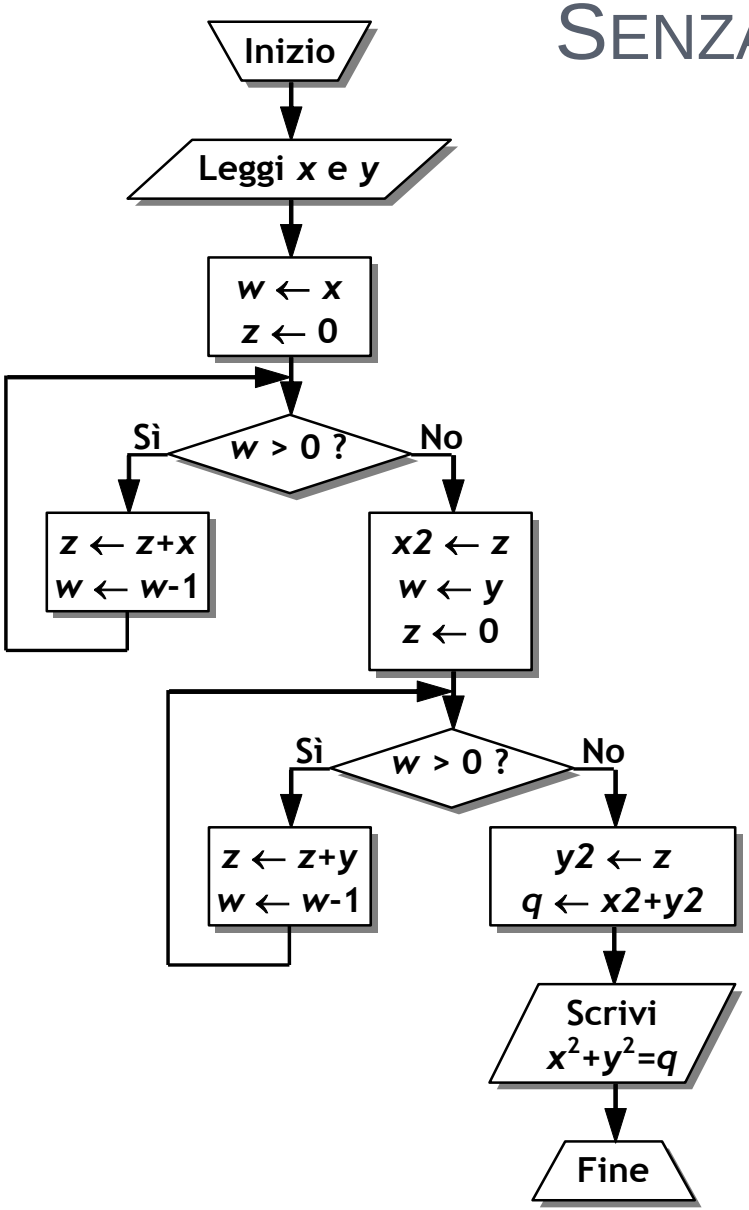
Funzione definita **dopo** il main

```
double f(int);
```

prototipo

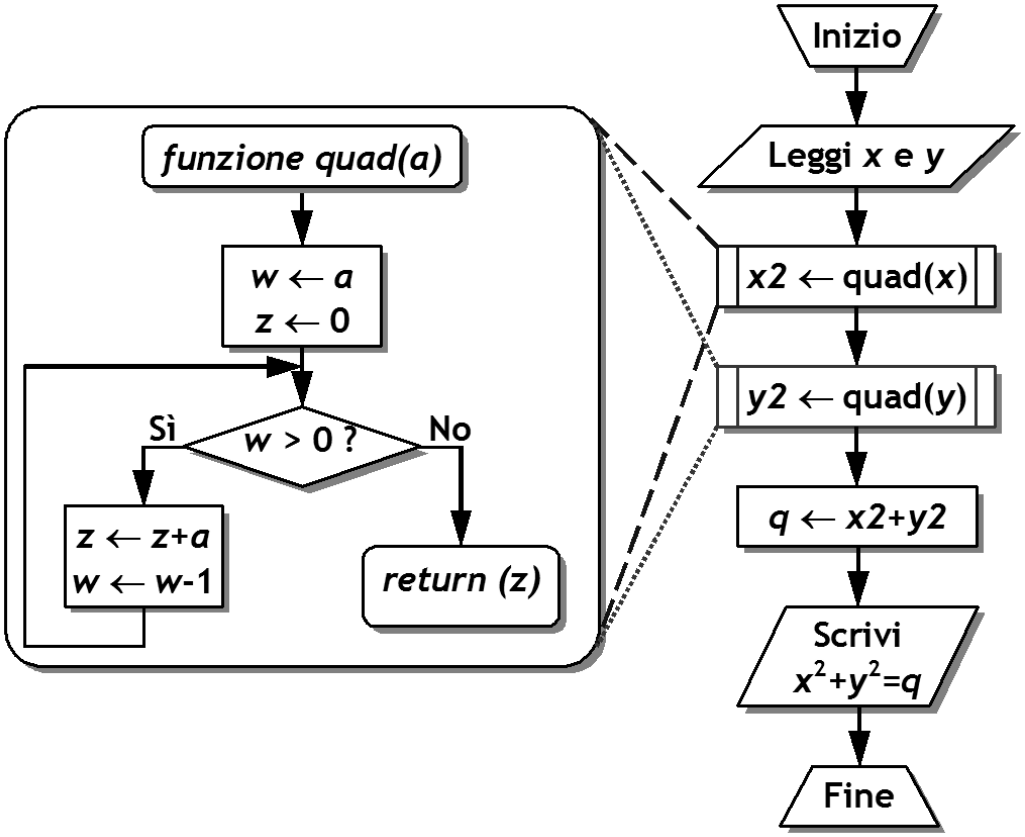
```
int main ()
{
  …
}
double f(int x)
{
  …
}
```

Senza sottoprogrammi




```
main() {
    /* q = x² + y² */
    int x, y, x2, y2, q, w, z;
    scanf("%d %d",&x,&y);
    w = x;
    z = 0;
    while (w > 0) {
       z = z + x;
       w = w – 1;
    }
    x2 = z;
    w = y;
    z = 0;
    while (w > 0) {
       z = z + y;
       w = w – 1;
    }
    y2 = z;
    q = x2+y2;
    printf("%d", q);
}
```

Con sottoprogrammi




```
int quad (int a) {
   /* restituisce a2 */
   int w, z;
   w = a;  z = 0;
   while (w > 0) {
      z = z + a;
      w = w – 1;
   }
   return (z);
}

main() {
   /* q = x2 + y2 */
   int x, y, x2, y2, q;
   scanf("%d %d",&x,&y);
   x2 = quad(x);
   y2 = quad(y);
   q = x2+y2;
   printf("%d", q);
}
```

Funzioni di libreria

- Il C prevede numerose funzioni predefinite per scopi diversi
- Particolarmente utili sono:
  - Funzioni matematiche
  - Funzioni di utilità
- Definite in specifiche *librerie*

UN ESEMPIO DI LIBRERIA – FUNZIONI MATEMATICHE

- Utilizzabili con `#include <math.h>`

funzione	*definizione*
`double sin (double x)`	
`double cos (double x)`	
`double tan (double x)`	
`double asin (double x)`	
`double acos (double x)`	
`double atan (double x)`	
`double atan2 (double y, double x)`	atan (y / x)
`double sinh (double x)`	
`double cosh (double x)`	
`double tanh (double x)`	

funzione	*definizione*
`double pow (double x, double y)`	x^Y
`double sqrt (double x)`	radice quadrata
`double log (double x)`	logaritmo naturale
`double log10 (double x)`	logaritmo decimale
`double exp (double x)`	e^X
`double ceil(double x)`	ceiling(x)
`double floor(double x)`	floor(x)
`double fabs (double x)`	valore assoluto
`double fmod (double x, double y)`	modulo

Funzioni di utilità

- Classificazione caratteri
  - `#include <ctype.h>`
- Funzioni matematiche intere
  - `#include <stdlib.h>`
- Stringhe
  - `#include <string.h>`

Definizione di dati

- Tutti i dati devono essere definiti prima di essere usati

- Definizione di un dato:
  - riserva spazio in memoria
  - assegna un nome

- Richiede l'indicazione di:
  - tipo
  - modalità di accesso (variabili/costanti)
  - nome (identificatore)

Definizione di variabili

- Sintassi:

```
<tipo>  <variabile>;
```

- Sintassi alternativa (definizioni multiple):

```
<tipo>  <lista di variabili>;
```

`<tipo>`: int, char, double, float

`<variabile>`: identificatore che rappresenta il nome della variabile

`<lista di variabili>`: lista di identificatori separati da ‘,’

Esempio definizione variabili

- `int x;`
- `char ch;`
- `int x1,x2,x3;`
- `double pi;`
- `float stipendio;`

VISIBILITÀ DELLE VARIABILI

```
int n;
double x;
main() {
   int a,b,c;
   double y;
   {
         int d;
         double z;
   }
}
```

- n, x: visibili in tutto il file
- a, b, c, y: visibili in tutto il main
- d, z: visibili nel blocco

Definizione di costanti

- Sintassi:
  - [**const**] <***tipo***> <***variabile***> [= <***valore***>] **;**
- Esempi:
  - `const double PIGRECO = 3.14159;`
  - `const char  SEPARATORE ='$';`
  - `const float ALIQUOTA = 0.2;`
- Convenzione:
  - Identificatori delle costanti tipicamente in MAIUSCOLO
    - `const double PIGRECO = 3.14159`

Costanti speciali

- Caratteri ASCII non stampabili e/o “speciali”
- Ottenibili tramite “*sequenze di escape*”

- Esempi:
  - **`'\008'`** **backspace**
  - **`'\009'`** **tab**
  - **`'\010'`** **new line**

- Caratteri “predefiniti”
  - **`'\b'`** backspace
  - **`'\t'`** tab
  - **`'\n'`** new line

STRINGHE

- Definizione:
  - sequenza di caratteri terminata dal carattere `NULL` (‘`\0`’)
- Non è un tipo di base del C
- Costanti stringa:

  “<***sequenza di caratteri***>”
- Esempio:
  - **“Ciao!”**
  - **“abcdefg\n”**

Le istruzioni

- Istruzioni di ingresso/uscita
- Istruzioni aritmetico–logiche
- Istruzioni di controllo [Seconda lezione in laboratorio]

Operazioni su interi

=	Assegnamento
+	Somma
-	Sottrazione
*	Moltiplicazione
/	Divisione con troncamento della parte frazionaria
%	Resto della divisione intera
==	Relazione di uguaglianza
!=	Relazione di diversità
<	Minore
>	Maggiore
<=	Minore o uguale
>=	Maggiore o uguale

Operazioni su float

=	Assegnamento
+	Somma
-	Sottrazione
*	Moltiplicazione
/	Divisione a risultato reale
==	Relazione di uguaglianza
!=	Relazione di diversità
<	Minore
>	Maggiore
<=	Minore o uguale
>=	Maggiore o uguale

L'ISTRUZIONE PRINTF()

- Sintassi:

```
printf(<stringa formato>,<arg1>,...,<argn>);
```

- ***<stringa formato>***: stringa che determina il formato di stampa di ognuno dei vari argomenti
  - Può contenere:
    - Caratteri (stampati come appaiono)
    - Direttive di formato nella forma %<carattere>
      - **%d** **intero**
      - **%u** **unsigned**
      - **%s** **stringa**
      - **%c** **carattere**
      - **%x** **esadecimale**
      - **%o** **ottale**
      - **%f** **float**
      - **%g** **double**

L'ISTRUZIONE PRINTF()

- Sintassi:

  `<arg1>,...,<argn>`: quantità da stampare

  - Associati alle direttive di formato nello stesso ordine!

- Esempi

  - int x = 2;
  - float z = 0.5;
  - char c = 'a';

  - printf("%d %f %c\n", x ,z, c);

output

```
2  0.5  a
```

  - printf("%f***%c***%d\n", z, c ,x);

output

```
0.5***a***2
```

L'ISTRUZIONE SCANF()

- Sintassi:

```
scanf(<stringa formato>,<arg1>,...,<argn>);
```

- **<stringa formato>:** come per printf
- **<arg1>,...,<argn>:** le variabili cui si vogliono assegnare valori

IMPORTANTE: i nomi delle variabili vanno precedute dall'operatore & che indica l'indirizzo della variabile

- Esempi:
- int x;
- float z;
- scanf("%d %f", &x, &z);

I/O a caratteri

- Acquisizione/stampa di un carattere alla volta

  - `getchar()`
    - Legge un carattere da tastiera
    - Il carattere viene fornito come “risultato” di **getchar**
    - (valore intero)
    - In caso di errore il risultato è la costante **EOF**

    (definita in **stdio.h)**

  - `putchar(<carattere>)`
    - Stampa `<carattere>` su schermo
    - `<carattere>:` una dato di tipo **char**

I/O a caratteri - Esempio

```
#include <stdio.h>
main(
{
  int tasto;
  printf("Premi un tasto...\n");
  tasto = getchar();
  printf("Hai premuto %c\n", tasto);
}
```

I/O a righe

- Acquisizione/stampa di una riga alla volta

- Istruzioni:
  - `gets(<variabile stringa>)`
    - Legge una riga da tastiera
    - La riga viene fornita come stringa
    - In caso di errore il risultato è la costante NULL

  - `puts(<stringa>)`
    - Stampa `<stringa>` su schermo
    - Aggiunge sempre '\n' alla stringa

Dal Codice all'eseguibile…

1. Istruzioni operative
   - Aprire il terminale
   - Digitare **emacs &** seguito dal tasto invio
   - Scrivere il codice del programma
   - Salvare il file con estensione **.c**

2. Istruzioni di compilazione
   - `gcc -o [<nome_file.out>] <nome_file.c>`
   - genera nome_file.out

3. Istruzioni di esecuzione
   - `./nome_file.out`

Esercizi

- Programma che stampa la somma di due numeri interi presi in input con e senza ausilio di funzioni

- Programmi che chiariscono il concetto di visibilità delle variabili