

Primi passi di Programmazione

Pagina 1

Breve storia

- C: nato negli anni '70 al fine di scrivere il sistema operativo Unix
- Caratteristiche:
 - Piccolo
 - Portabile
 - Efficiente

Pagina 2

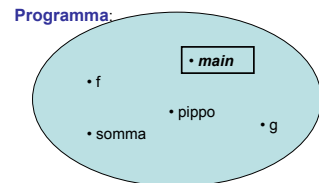
Il C: altre caratteristiche

- Compilato
- Strutturato
- Modulare
 - Top-down
 - Bottom-Up
- A tipizzazione statica

Pagina 3

Programma

- Insieme non ordinato di funzioni tra le quali anche la funzione "main"



- **main**: punto di inizio

Pagina 4

Funzione

$$f: D \rightarrow C$$

- D : dominio
- C : codominio
- Applicando f ad un elemento di D , si ottiene un elemento di C
- Occorre specificare D e C , cioè il tipo dell'input e il tipo dell'output

Pagina 5

Esempio di funzione

- Esempio: $power(x,y) \rightarrow x^y$
- Se x e y sono interi, anche x^y lo sarà
 - Quindi: $D=(intero, intero)$; $C= intero$
- In C:

```
int power(int x, int y)
```

 - Specifica i tipi dell'I/O
 - E' la **dichiarazione** (o prototipo) di una funzione

Pagina 6

BNF

- Per descrivere le regole sintattiche da seguire useremo:
 - `<>` indica una categoria sintattica
 - `::=` "deve essere riscritto come"
 - `|` barra verticale per separare più scelte
 - `{ }` scegliere una delle voci tra graffe
 - `{ }0+` ripetere la voce tra graffe 0 o più volte
 - `{ }1+` ripetere la voce tra graffe 1 o più volte
 - `{ }opt` voci opzionali

Pagina 7

Dichiarazione di una funzione

- Sintassi:

```
<TIPO-OUTPUT> <NomeFun> (<PARAMETRI INPUT>)
```

- Specifica i tipi dell'I/O
- E' la **dichiarazione** (o prototipo) di una funzione
- Non dice cosa fa la funzione, ma solo il tipo dell'input e quello dell'output
- **NOTA**: la dichiarazione di una funzione deve sempre precedere l'uso della stessa

Pagina 8

Definizione di una funzione

- Sintassi:

```
<TIPO-OUTPUT> <NomeFun> (<PARAMETRI-INPUT>)  
{<Corpo>}
```

- **Definizione** = **Dichiarazione** + **Corpo**
- **Corpo**: Sequenza di Istruzioni

Pagina 9

Struttura Programma

```
...  
<Dichiarazione funzioni fi (i=1, ...,n)>  
<Definizione del main>  
<Definizione funzioni fi (i=1, ...,n)>
```

- Per ora ci soffermiamo sulla definizione del **main**

Pagina 10

La funzione main

- Ogni programma deve avere **una ed una sola** funzione main
- Definizione di base:

```
Intestazione → int main(void)  
Corpo {  
    ...  
}
```

- Ritorna un intero → **int**
- Non ha parametri in input → (**void**)

Pagina 11

Corpo di una funzione

- E' un **blocco**, cioè una *sequenza di istruzioni* racchiuso *tra graffe* {...} :

```
{  
    <dichiarazione di variabili>  
    <altre istruzioni>  
}
```

Pagina 12

Variabili

- Una variabile è univocamente identificata da:
 - *Nome, Tipo, Valore*
- In C è necessario *dichiarare* le variabili per poterle usare:

```
int x, z, k;  
float a, b;
```

- In C è possibile *inizializzare* una variabile in fase di dichiarazione:

```
int x = 0, z = 5, k = 1788;  
float a = 3.14, b = 314.5e-2;
```

Pagina 13

Dichiarazione di variabili

- Sintassi per la dichiarazione di una o più variabili dello stesso tipo, con eventuale inizializzazione:

```
<tipo> <NomeVar> {= <Valore>}opt  
{, <NomeVar> {= <Valore>}opt}0+
```

Pagina 14

Variabili: Nome

- **Nome:** identificatore
- **Identificatore:**
 - sequenza di lettere, cifre e “_”.
 - Il primo carattere non può essere una cifra
- Esempi:

```
• L_324  
• _alpha  
• x Si
```

```
• 324_L  
• -alpha  
• x/y No
```

Pagina 15

Sintassi per gli identificatori

- Identificatore ::= {lettera|underscore}₁
{lettera|underscore|cifra}₀₊
- lettera ::= minuscola | maiuscola
 - minuscola ::= a | b | ... | z
 - maiuscola ::= A | B | ... | Z
- cifra ::= 0 | ... | 9
- underscore ::= _

Pagina 16

Variabili: Nome (cont.)

- NOTA: Il C è **case-sensitive**, quindi *x* è diverso da *X*
- **Errore tipico:** dichiarare *x* e poi usare *X*: il compilatore segnala l'errore!!!
- **Buone abitudini:**
 - *Scegliere nomi significativi: rende il programma auto-esplicativo*
 - es: **area, perimetro, gradi**
 - *Separare parole diverse con “_”*
 - es: **area Rettangolo, gradi kelvin**
 - *Usare solo lettere minuscole per le semplici variabile*

Pagina 17

Variabili: Tipo

- Il tipo di una variabile determina:
 - L'insieme di **valori** che la variabile può assumere
 - L'insieme di **operazioni** che è lecito applicare ad essa
- Tipi base in C:
 - **char** → per i caratteri
 - **int** → per gli interi
 - **float** → per i reali

Pagina 18

Numeri Interi Relativi

- Si usa la parola chiave **int**
 - Es `int area, perimetro;`
- Solitamente un *int* viene memorizzato in 2 o in 4 byte (dipende dalla versione del sistema).
 - Se abbiamo 4 byte → 2^{32} possibili stati differenti
 - $-2^{31}, -2^{31}+1, \dots, -3, -2, -1, 0, 1, 2, 3, \dots, 2^{31}-1$
- Esistono poi due tipi collegati:
 - `short` → 2 byte (di solito)
 - `long` → 4 byte (di solito)
 - Es: `short base, altezza;`
`long area, perimetro;`

Pagina 19

Numeri Naturali

- Se si vuole rappresentare un **numero naturale**
 - Premettere la parola chiave "**unsigned**" al tipo intero:
 - Es `unsigned int area, perimetro;`
`unsigned short base, altezza;`
`unsigned long volume;`
 - Se abbiamo 4 byte → 2^{32} possibili stati differenti
 - $0, 1, 2, 3, \dots, 2^{32}-1$
 - **NOTA:** nel caso di **unsigned int**, si può sottintendere la parola chiave **int**
 - `unsigned area, perimetro;`

Pagina 20

Numeri Reali

- Si usa la parola chiave **float**
 - Es `float gradi_kelvin;`
- Solitamente un *float* viene memorizzato in 4 byte (dipende dalla versione del sistema).
- Esistono poi due tipi collegati:
 - `double` → 8 byte (di solito) **CONSIGLIATO!**
 - `long double` → 10 byte (di solito)
 - Es: `double volume;`
`long double miglia;`

Pagina 21

Caratteri

- Si usa la parola chiave **char** (occupano 1 byte)
 - Es `char iniziale_cognome;`
 - Un carattere deve essere scritto tra virgolette singole:
 - `iniziale_cognome = 'A';`
 - Questo per evidenziare quando una cifra è da considerarsi come un carattere e non come un numero
 - `char ufficio = '9';`
`int n_uffici = 9;`
 - Variabili di tipo char possono anche essere interpretate e manipolate come "piccoli numeri", ma noi no!!!

Pagina 22

Istruzioni

- Manipolano le *variabili* definite nel programma
- Terminano sempre con ";"
- Esempi:

Dichiarazioni { `int x, z = 5, k;`
`float y, a = 7.2, b = 1.2;`

Assegnamenti { `x = 2;`
`y = a + b;`

Pagina 23

Istruzione di assegnamento

- Sintassi

```
<NomeVar> = <Espressione>;
```

- "=" è l'operatore di assegnamento
- L'istruzione di assegnamento ha l'effetto di assegnare il valore di <Espressione> alla variabile di nome <NomeVar>
- Ogni espressione ha un **tipo** ed un **valore**

Pagina 24

Istruzione di assegnamento (cont)

- Per eseguire l'istruzione di assegnamento è necessario:
 - determinare il valore di **<Espressione>**
 - Assegnare tale valore alla variabile **<NomeVar>**

```
a = 2;  
b = (c + d) / 2;  
c = d;
```

Pagina 25

Istruzioni

- **Errore tipico:**
 - usare variabili non dichiarate!
- **Buone abitudini:**
 - Lasciare uno spazio su entrambi i lati di un operatore binario

Pagina 26

Valutazione di un'espressione

- Determinazione del **tipo** di un'espressione:
 - Dipende dal tipo delle variabili coinvolte:
 - In generale l'espressione viene "promossa" al tipo più generale usato nell'espressione
 - Nota: variabili di tipo *short* vengono convertiti in *int* per la valutazione di un'espressione
- Determinazione del **valore** di un'espressione:
 - Le espressioni che coinvolgono gli usuali operatori aritmetici (+, -, /, *) vengono valutate secondo le usuali regole di precedenza, da sinistra verso destra

Pagina 27

Nota Bene

- **int / int = int**
- Costanti reali, esempi:
 - 3.0
 - 3.
 - 3f
 - 3F
- Un conto è il tipo dell'espressione, un altro è il tipo della variabile a cui l'espressione viene assegnata, definito in fase di dichiarazione!

Pagina 28

Output: "printf"

- Per la visualizzazione dell'output su schermo:

```
printf (<Stringa di controllo> , {<parametri>}opt)
```

- E' dichiarata in una libreria standard: **stdio.h**
- Per poterla usare è necessario importare la libreria:

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

- È una direttiva per il preprocessore

Pagina 29

Printf

- La **<stringa di controllo>** definisce
 - i caratteri da visualizzare
 - i tipi delle variabili da visualizzare
 - la posizione delle variabili nella stringa di output
- I **<parametri>** devono essere consistenti per numero e tipo con quanto specificato nella stringa di controllo e possono essere
 - costanti
 - nomi di variabili

Pagina 30

Specifica di tipo

Carattere	Tipo del parametro
c	carattere
i	intero relativo
u	intero senza segno (naturale)
e	reale in notazione esponenziale
E	reale in notazione Esponenziale
f	reale
s	stringa

Pagina 31

Printf: esempi

```
printf("Welcome to C!");  
printf("Welcome \n to \n C!");  
printf("Welcome\nto\nC!");
```

```
printf("Il fattore di conversione  
\n gradi centigradi- Kelvin è \n  
%f", -273.16);
```

```
printf("%f gradi centigradi  
equivalgono a %f Kelvin\n",  
gradi_centigradi, Kelvin);
```

Pagina 32

Commenti

```
/* Commento su  
   più righe*/  
// commento su una riga singola
```

- Buona abitudine: mettere un commento all'inizio di ogni funzione, in cui si indica:
 - La specifica della funzione
 - Input atteso
 - Output prodotto

Pagina 33

Esempi di programmi (1)

```
/* Raffaella Lanzarotti Data Ultima Modifica: 17  
Ottobre 2005.  
Questo programma Visualizza due stringhe di controllo  
*/  
  
#include <stdio.h>  
#include <conio.h>  
int main(void)  
{  
    printf("Hello, world!\n");  
    getch();  
    printf("Hello!\n");  
    getch();  
    return 0;  
}
```

Pagina 34

Esempi di programmi (2)

```
/* Raffaella Lanzarotti Data Ultima Modifica: 17  
Ottobre 2005  
Questo programma Visualizza più stringhe di  
controllo  
*/  
#include <stdio.h>  
#include <conio.h>  
  
int main(void)  
{  
    printf("\n\n\n\n\n\n\n\n\n\n");  
    printf("*****\n");  
    printf(" * from sea *\n");  
    printf(" * to shining C *\n");  
    printf("*****\n");  
    printf("\n\n\n\n\n\n\n\n\n\n");  
    getch();  
    return 0;  
}
```

Pagina 35

Esempi di programmi (3)

```
/* Raffaella Lanzarotti Data Ultima Modifica: 17  
Ottobre 2005.  
The distance of a marathon in kilometers. */  
  
#include <stdio.h>  
#include <conio.h>  
int main(void)  
{  
    int miles, yards;  
    float kilometers;  
  
    miles = 26;  
    yards = 385;  
    kilometers = 1.609 * (miles + yards / 1760.0);  
    printf("\nA marathon is %f kilometers.\n\n",  
kilometers);  
    getch();  
    return 0;  
}
```

Esercizio: Conversione Euro / Lire

- Scrivere un programma che dato un valore in Euro, lo converte in lire italiane

Pagina 37

Esercizio: Conversioni di temperature

- Scrivere un programma che dato un valore in gradi centigradi, lo converte in gradi Fahrenheit
- Scrivere un programma che dato un valore in gradi centigradi, lo converte in Kelvin

Pagina 38

Esercizio: Differenza di Tempi

- Scrivere un programma che dati due tempi espressi in ore, minuti e secondi, ne calcoli la differenza, esprimendola in secondi

Pagina 39

Esercizio: Scambio di variabili

- Scrivere un programma che date due variabili *a* e *b* inizializzate rispettivamente a 0 e 1, scambiarne i valori e stamparli.

Pagina 40