CALCOLO NUMERICO con ELEMENTI DI PROGRAMMAZIONE (BATR) - (A.A. 2012-2013)

Prof. F. Pitolli

ELEMENTI DI BASE DEL LINGUAGGIO C

Ing. Gabriele Colosimo, Ing. Andrea Nascetti

Area di Geodesia e Geomatica
Dipartimento di Ingegneria Civile Edile e Ambientale
Università di Roma "La Sapienza"

<gabriele.colosimo, andrea.nascetti>@uniroma1.it

Indice

Indice

Dalle origini ...

Un po' di storia

- Viene sviluppato nel 1967 da Martin Richards come evoluzione di due precedenti linguaggi (**B** e **BCPL**)
- Si afferma inizialmente per essere alla base del SO **UNIX**
- Nel 1989 fu approvato il primo standard **ANSI** "per fornire una versione del linguaggio che non fosse ambigua e che fosse indipendente dalle macchine" (**C89**)
- Grazie alla sua efficienza, velocità e **portabilità**, è oggi alla base dei principali SO dell'ultima generazione (**C95**, **C99** [1], **C11**)

Principali caratteristiche

Uno sguardo "dall'alto"

- Il linguaggio è semplice ed efficiente
  - 37 **keywords** riservate (**facili da imparare**)
  - il "modo" di trattare gli oggetti è molto simile a quello del computer (**basso livello**)
  - semplice costruire nuovi **tipi di dati**
  - fornisce i fondamentali costrutti per il controllo del flusso (**if-else**, **switch**, **while**, **for**, **do**, **break**)
  - il compilatore traduce le istruzioni in linguaggio macchina
- Mette a disposizione una ricca **libreria Standard** (insieme di funzioni)
  - per l'accesso al sistema operativo
  - per l'allocazione di memoria
  - per manipolazione di input/output

Indice

Il codice sorgente

```c
/*
 * programma: somma.c
 * Chiede di inserire due numeri (cateti di un triangolo
 * rettangolo) e restituisce l'ipotenusa
 */

#include <stdio.h>
#include <math.h>

int main()
{
   double a, b, c;

   printf("Inserire due numeri: ");
   scanf("%lf%lf", &a, &b);
   c = sqrt(a*a + b*b);
   printf("L'ipotenusa vale = %.2lf\n", c);

   return 0;
} /* fine funzione main */
```

Come (e perché) commentare il codice

- Tutto il codice che segue i caratteri /* viene ignorato **fino alla prima occorrenza** di */

```
1 /*
2  * programma: somma.c
3  * Chiede di inserire due numeri (cateti di un triangolo
4  * rettangolo) e restituisce l'ipotenusa
5  */
```

- Nello standard **C99**, la **singola linea** che segue // viene commentata

```
1 // Singola linea di commento
```

Commentare il codice è fondamentale!

- Aiuta la comprensione (propria e altrui) (**codice e commento corretti**)
- Mette in luce errori nel codice (**commento corretto codice sbagliato**)

Organizzare il codice

La sintassi delle funzioni

- Il codice C è organizzato in **funzioni**
- **arg1 func** (**arg2**, **arg3**, . . . , **argn**){. . . . . }
- Ogni funzione ha un un nome (**func**)
- Il codice della funzione è inserito in parentesi graffe {. . .} **block**
- Dopo la chiusura . . .}non va inserito il **;**
- Gli argomenti in ingresso (**arg2, arg3,** . . . , **argn**), se ce ne sono, vanno inseriti in parentesi tonde
- L'argomento di uscita (**arg1**) viene restituito con la parola chiave **return**

La funzione main()

La funzione main()

- E' la funzione fondamentale che permette l'esecuzione del programma
- **main()** restituisce un argomento di **tipo** intero
- Storicamente, restituisce 0 se il programma è stato correttamente eseguito fino alla fine
- Ogni programma **comincia eseguendo la funzione main()**

```
1 int main()
2 {
3    ...
4    ...
5    ...
6    ...
7
8    return 0;
9 }
```

Le funzioni di libreria

La Libreria standard (Standard library)

- Nel linguaggio C, molte funzioni sono già state implementate e sono disponibili tramite **librerie** esterne
- Una libreria è un **insieme di funzioni** che assolvono determinati compiti
- Per utilizzare queste funzioni, il compilatore deve conoscere
  - **numero e tipo** degli argomenti in ingresso e in uscita per la funzione
- La **definizione** delle funzioni è contenuta in *header files* raggruppati per categorie

<**stddef.h**>	definizioni comuni	*size_t()*, ...
<**ctype.h**>	gestione dei caratteri	*is_lower()*, ...
<**math.h**>	funzioni matematiche	*sqrt()*, ...
<**stdio.h**>	gestione dell'input/ouput	*printf()*, ...
<**stdlib.h**>	utilità generiche	*atof()*, ...
<**string.h**>	gestione delle stringhe	*strcmp()*, ...
<**limits.h**>	limiti dell'implementazione	*SHRT_MIN*, ...

La Libreria standard (Standard library)

- Devono essere inseriti nel codice sorgente prima dell'utilizzo delle funzioni
- La **direttiva** #**include** carica il file all'interno del programma
- Può essere utilizzata in due diverse configurazioni
  1. #**include** <**stdio.h**> per *header files* in cartelle **note** al compilatore
  2. #**include "my_file"** per *header files* nella cartella in cui si esegue la compilazione

```
1
2 #include <stdio.h>
3 #include <math.h>
4
5 int main()
```

Le variabili

Come rappresentare i dati del problema

- **double a, b, c;** è una dichiarazione di variabili
- Una variabile è una posizione della memoria del computer in cui un valore può essere immagazzinato
- Il C è un linguaggio **tipizzato**: tutte le variabili devono essere dichiarate con un **tipo** e un nome:
  - sono permessi i caratteri: **a-z, A-Z, 0-9, _**
  - il nome **x1** è valido, **1x** non lo è
  - non utilizzare nomi più lunghi di 31 caratteri
- Il C è un linguaggio **fortemente** tipizzato (costanti letterali, variabili, espressioni, funzioni, parametri di funzione)

Le espressioni

Il "cuore" del programma

- Espressioni e operatori sono **strumenti fondamentali** del programmatore
- **c = a + b;** assegna il valore dell'**espressione** **a + b** alla variabile **c**
  - il **;** va sempre sempre messo alla fine delle espressioni
  - sarebbe possibile anche scrivere **c = d = a + b;**
  - allo stesso modo: **c = (d = a + b);**
- Esistono **pratiche scorciatoie** per operare sulle variabili
  - **c += a + b;** è equivalente a **c = c + a + b;**
  - **c /= a * b;** è equivalente a **c = c/(a*b)**;

Gli operatori

"Manovrare" le variabili

- Gli **operatori** eseguono operazioni di calcolo sugli operandi
  - operatori aritmetici: **+, -, *, /, =, %...**
  - operatori relazionali: **==, !=, <, >, <=, >=**
- **regole di priorità degli operatori**
  - **( )** sono valutate per prime
  - ***, / o %** sono valutati per secondi, da sinistra a destra
  - **+, -** sono valutate per ultime
- operatori di **pre**-incremento/decremento **++,--**
  - **++i** equivale a **i = i + 1**
  - **--i** equivale a **i = i - 1**
- operatori di **post**-incremento/decremento **++,--**
  - **i++** incrementa il contenuto di **i** ma **restituisce il valore originale**
  - **i--** decrementa il contenuto di **i** ma **restituisce il valore originale**

Proviamo...

```
1   int x=2, y, z;
2
3   x *= 3 + 2;
4   printf("x = %d\n", x);
5
6   x *= y = z = 4;
7   printf("x = %d, y = %d, z = %d\n", x, y, z);
8
9   x = y == z;
10  printf("x = %d, y = %d, z = %d\n", x, y, z);
11
12  x = ( y = z++ );
13  printf("x = %d, y = %d, z = %d\n", x, y, z);
14
15  x = ( y = ++z );
16  printf("x = %d, y = %d, z = %d\n", x, y, z);
```

Indice

Le strutture di selezione if, if-else

Esecuzione condizionale

- **if** (condizione logica) espressione
  - esegue le espressioni solo se la condizione logica è verificata
  - utilizza gli operatori relazionali ==, !=, <, >, <=, >=
  - operatori logici per concatenare più condizioni &&, ||, !
- **else** espressione
  - deve essere preceduto da un **if ()**
  - viene eseguito se la condizione logica del precedente **if** non viene verificata

Esempio

```
1   printf("Inserire due numeri: ");
2   scanf("%lf%lf", &a, &b);
3
4   if(a < 0 || b < 0)
5   {
6     printf("Errore: le dimensioni dei cateti devono essere
          positive\n");
7     exit(EXIT_FAILURE);
8   }
9   else
10  {
11    c = sqrt(a*a + b*b);
12    printf("L'ipotenusa vale = %.2lf\n", c);
13  }
```

La struttura di iterazione for ()

Ciclo che ripete un blocco di operazioni *n* volte

- **for** (espressione iniziale; condizione logica; espressione incrementale)

```
1 for(contatore = 1; contatore <= 10; contatore++)
2 {
3   espressione 1;
4   espressione 2;
5   ...
6   espressione n;
7 }
```

- **contatore** è il nome della variabile di controllo
- **1** è il valore iniziale della variabile di controllo
- **10** è il valore finale della variabile di controllo
- **contatore++** incremento della variabile di controllo

La struttura di iterazione while ()

Ciclo che ripete un blocco di operazioni se è verificata una condizione

- **while** (condizione logica)

```
1 while(voto != -1)
2 {
3   espressione 1;
4   printf("Inserisci il voto: ");
5   scanf("%d", &voto);
6 }
```

- Se la variabile **voto** è diversa da -1 il ciclo viene ripetuto

Differenze tra for() e while()

- Conoscendo il numero di iterazioni è più indicato il costrutto **for()**
- Per iterare fino al raggiungimento di una certa condizione si utilizza il costrutto **while()**
- Entrambi possono essere interrotti utilizzando la parola chiave **break;**

Indice

Cominciamo a esercitarci... I

Esercizio 1

Scrivere un programma che, preso un intero *n* da tastiera (**stdin**), determini se *n* è pari o dispari

- Utilizzare la struttura di controllo **if()**

Esercizio 2

Scrivere un programma che, preso un intero *n* da *stdin*, calcoli *n*!

- Utilizzare la struttura iterativa **for()**
- Dopo aver implementato correttamente il programma, provare a costruire la funzione **long int CALCOLO_FATTORIALE(long int numero)**

Cominciamo a esercitarci... II

Esercizio 3

Scrivere un programma per stampare il valore massimo e minimo dei seguenti tipi di dati: **int**, **long int**, **float**, **double**, **long_double**

- Includere gli *header files* necessari...
- Testare cosa succede se tali limiti vengono superati (che succede se stampo come intero **INT_MAX + 1**?)

Esercizio 4

Scrivere un programma che, preso un intero *n* da *stdin*, trovi **tutti i numeri primi inferiori a** *n*

- Utilizzare tutti i costrutti mostrati in precedenza
- L'operatore % può essere d'aiuto...

Indice

Cosa abbiamo imparato

Un programma C è fatto (anche) di

- Espressioni (**Variabili** e **Operatori**)
- Commenti
- Funzioni (**chiamate a librerie esterne**)
- Strutture condizionali (**if-else**)
- Strutture iterative (**for, while**)

Cosa dobbiamo ancora vedere...

- Vettori e Matrici
- Input/Output

Riferimenti Bibliografici

ANSI WG14

ISO/IEC 9899 C standard