Corso di
Laboratorio di programmazione e calcolo

Docente: Stefano Finzi Vita

Pagina del corso: http://www1.mat.uniroma1.it/mat_cms/pres_corso.php?corso_da_presentare=1851&lingua=italiano
oppure link da http://www1.mat.uniroma1.it/people/finzi/

Introduzione (II parte)

I linguaggi C e C++

Il C è un linguaggio di programmazione di applicazione generale che presenta come proprie caratteristiche la sinteticità delle espressioni, moderne strutture di controllo e di dati, e un esteso insieme di operatori (Dennis Ritchie e Brian Kernigham, '78)

Il C è il primo linguaggio moderno, sviluppato dalla comunità dei programmatori. Le sue caratteristiche principali sono:

- Notazioni compatte ed efficaci
- Portatilità su diverse architetture
- Attraverso un controllo elevato sulla memoria della macchina consente di ottimizzare il codice
- Versatilità: è utilizzato per scrivere sistemi operativi, librerie scientifiche e grafiche, pacchetti applicativi
- E' uno dei linguaggi più diffusi al mondo nel calcolo scientifico e nell'industria (con il C++)

Il linguaggi C e C++

- Il C è un linguaggio di *livello intermedio*, nel senso che usa strutture di alto livello ma può maneggiare attività di basso livello
  - basso livello = vicino alle istruzioni macchina
  - alto livello = vicino al programmatore
- Il linguaggio C++ è una versione ampliata e migliorata di C nella direzione della programmazione a oggetti, che consente di ridurre la complessità dei programmi e di gestire quindi anche programmi di grandi dimensioni.
- Si deve a Bjarne Stroustrup (già nel '79, ma ufficialmente uscì nell'83)
- Noi useremo un sottoinsieme di C++ (di fatto indistinguibile da C se non per poche istruzioni, che di volta in volta segnaleremo)
- Sotto *linux* i compilatori sono rispettivamente **gcc** per C (files **.c**) e **g++** per C++ (files **.cpp**). Noi useremo sempre quest'ultimo.

Struttura di un programma C o C++

Un programma C ha una struttura riconoscibile e occorre tenerne conto nella sua scrittura. E' articolato in sottoprogrammi (detti *funzioni*) ciascuno dei quali è un blocco di istruzioni in grado di risolvere uno specifico problema. Possiamo distinguere nell'ordine diversi segmenti:

Blocco commenti

Aiuta a ricordare cosa fa il nostro programma. Specifica input e output

Blocco comandi per il preprocessore

Di solito preceduti dal simbolo #

Blocco di dichiarazioni generali

Contiene le variabili globali e i prototipi delle funzioni utilizzate

Main

E' il programma principale. E' anch'esso una funzione, ma obbligatoria

Funzioni

Elenco delle funzioni utilizzate, con tipo, argomenti ed istruzioni proprie

Commenti al programma *prova_c.cpp*

- In C c'è differenza tra minuscole e maiuscole

Commenti al programma *prova_c.cpp*

- In C c'è differenza tra minuscole e maiuscole
- Due tipi di commento:
  - `//` : trasforma in commento tutti i caratteri seguenti sulla stessa riga
  - `/*...*/` : trasforma in commento tutto quello che c'è in mezzo

Commenti al programma *prova_c.cpp*

- In C c'è differenza tra minuscole e maiuscole
- Due tipi di commento:
  - `//` : trasforma in commento tutti i caratteri seguenti sulla stessa riga
  - `/*...*/` : trasforma in commento tutto quello che c'è in mezzo
- Ogni istruzione deve essere conclusa da un punto e virgola (;)

Commenti al programma *prova_c.cpp*

- In C c'è differenza tra minuscole e maiuscole
- Due tipi di commento:
  - // : trasforma in commento tutti i caratteri seguenti sulla stessa riga
  - /∗ · · · ∗/ : trasforma in commento tutto quello che c'è in mezzo
- Ogni istruzione deve essere conclusa da un punto e virgola (;)
- Inclusione delle librerie (istruzioni precedute dal simbolo *#include*)
  - < *stdio.h* > : libreria di funzioni di I/O di C (printf, scanf)
  - < *stdlib.h* > : libreria standard di C
  - < *math.h* > : libreria di funzioni matematiche
  - < *iostream* >
    using namespace std : libreria di funzioni I/O di C++ (cin, cout)

Commenti al programma *prova_c.cpp*

- In C c'è differenza tra minuscole e maiuscole
- Due tipi di commento:
  - $//$: trasforma in commento tutti i caratteri seguenti sulla stessa riga
  - $/*\cdots*/$: trasforma in commento tutto quello che c'è in mezzo
- Ogni istruzione deve essere conclusa da un punto e virgola (;)
- Inclusione delle librerie (istruzioni precedute dal simbolo *#include*)
  - $<$ *stdio.h* $>$: libreria di funzioni di I/O di C (printf, scanf)
  - $<$ *stdlib.h* $>$: libreria standard di C
  - $<$ *math.h* $>$: libreria di funzioni matematiche
  - $<$ *iostream* $>$
    using namespace std : libreria di funzioni I/O di C++ (cin, cout)
- Ogni variabile e ogni funzione deve avere un tipo che va dichiarato (solo per il *main* che è sempre intero si può sottintendere)

Gli identificatori (nomi)

- I nomi di costanti, variabili e funzioni devono essere costituiti da una lettera seguita da una qualunque combinazione di lettere, cifre o simboli non riservati (massimo 127 caratteri).
- Sono **simboli riservati** ad esempio $+ - */ =<> ()[\ ]\{\}., :;' !@\$\%|$
- Ci sono **parole riservate** che non possono essere usate come nome di variabili o funzioni: *main*, *define*, *int*, *float*, *double*, *if* , *for*, *do*, *then*, ...
  - Esempi corretti:
    Totale, *PARI* , *p3*, *estremo_a*, *prodotto_scalare*, *Ax26wt_11*
  - Esempi scorretti:
    3RD, *passo x*, *raggio%*, *#nuovo*
  - Consiglio: usare nomi che richiamino il contenuto delle variabili (inutile risparmiare: a distanza di tempo nessuno ricorderà che una variabile *w* indica un contatore, al contrario di *cont*)

Tipi di variabili

Le variabili vanno sempre dichiarate prima di essere utilizzate, di norma all'inizio del programma o di ogni funzione/sottoprogramma.
Alcuni tipi sono predefiniti, e ad ognuno di essi il compilatore riserva un opportuno spazio di memoria.

I NUMERI INTERI

- Occupazione di memoria:
  - short int = 2 bytes $\quad -32768 \leq n \leq +32767 \; [= 2^{15} - 1]$
  - (long) int = 4 bytes
    $-2.147.483.648 \leq n \leq 2.147.483.647 \; [= 2^{31} - 1]$
  - unsigned [interi positivi] $\quad 0 \leq n \leq 4.294.967.295 \; [= 2^{32} - 1]$
- Fuori dagli intervalli precedenti si verifica un **integer overflow** non dichiarato, ma il risultato non è quello voluto (v. maxint.cpp)
- Formato di I/O: *%d*

I NUMERI REALI

- Occupazione di memoria:
  - float = 4 bytes $3.4e-38 < |x| < 3.4e+38$
    semplice precisione [segno(1bit)+ mantissa(23bits) + esponente(8bits)]
  - double = 8 bytes $1.7e-308 < |x| < 1.7e+308$
    doppia precisione [segno(1bit)+ mantissa(52bits) + esponente(11bits)]
- Numeri più grandi [inf]
  overflow: il programma dà errore (*runtime error*)
- Numeri più vicini a zero [=0]
  underflow: il programma continua senza segnalare nulla
  (v. precisione.cpp)
- Formati di I/O: $\%f$ (semplice), $\%lf$ (doppia), $\%e$ (virgola mobile)
  (si può regolare: $\%1.3f$ = una cifra prima e tre dopo la virgola)
- Attenzione alle operazioni non valide: $(0/0)$ dà nan (**not a number**), mentre $(1/0)$ dà inf.

Altri tipi

- CARATTERI char = 1 byte (255 caratteri)
  Anche le stringhe (= sequenze di caratteri alfanumerici e spazi bianchi) sono variabili di tipo char

  "a" "Z27_bus" "Oggi e' venerdi'"

- LOGICA bool (VERO/FALSO)
  N.B. In C tale tipo non è predefinito. Una variabile booleana si può sostituire con una variabile intera a valori 0 e 1

- VUOTO void
  usato per le funzioni che non restituiscono nessun valore

- DEFINITO DALL'UTENTE (col comando typedef)

- Le costanti possono essere definite nel blocco di dichiarazione in modo da venire valutate già durante la compilazione. Esempi: #define LIMIT 150, #define NOME "pluto", #define MIN 25.1

Conversioni di tipo

Quando vengono eseguite operazioni tra variabili numeriche di tipo diverso, il compilatore le tratta come se fossero tutte dello stesso tipo, quello "superiore" (che cioè richiede maggiore occupazione di memoria) attraverso una conversione preventiva al calcolo:

$$short\ int \rightarrow int \rightarrow float \rightarrow double$$

Esempio: DIVISIONE TRA INTERI ($int\ n = 1, m = 2, k; float\ h, s = 2;$)

- $k = n/m;$ [k=0]
- $h = n/m;$ [h=0.000000]
- $k = 1/m;$ [k=0]
- $h = 1./m;$ [h=0.500000]
- $k = 1./s;$ [k=0]
- $h = float(n)/m;$ [h=0.500000] (cast = conversione forzata)

(divinteri.cpp)

Istruzioni di lettura e scrittura

- In genere printf stampa su schermo, scanf legge da tastiera:
  - printf("CIAO\n Inserire il primo numero: x= ");
  - printf("\n n = %d, \t a_n = %f ", indice, termine);
  - scanf("%d", &n);
- Sono comandi C della libreria stdio.h, ma riconosciuti anche in C++, richiedono di specificare il tipo delle variabili con il loro formato. Più avanti vedremo i comandi specifici di C++ per l'I/O (cin, cout).
- In stampa le parti di testo, i caratteri di controllo e i formati vanno tra virgolette, seguiti dai nomi delle variabili da stampare
- \n = a capo; \t = tabulazione
- In lettura per ogni variabile da leggere va precisato il suo formato tra virgolette, poi il suo nome preceduto dal simbolo &
- L'operatore & applicato a una variabile restituisce il suo indirizzo di memoria (*puntatore* alla variabile)

OPERATORI

ARITMETICI

- $+$, -, $*$, $/$, $=$ (assegnazione), $\%$ (resto della divisione tra interi)
- Attenzione: non esiste l'operatore di elevamento a potenza
- Le operazioni sono eseguite da sinistra a destra, a parità di priorità
  - $x = a/b * c;\quad y = a/(b * c);\quad z = n\%3;$
- Assegnazioni in forma compatta: $++$, - -, $+=$, $-=$, $*=$, $/=$ (più efficienti in esecuzione della relativa forma estesa)
  - $x++$ equivale a $x = x + 1$
  - $x--$ equivale a $x = x - 1$
  - $x+=4$ equivale a $x = x + 4$
  - $x*=5$ equivale a $x = x * 5$

LOGICI (di confronto)

- $==$ (uguale), $!=$ (diverso), $<$, $>$, $<=$,$=>$
  - $x == y$ (x uguale a y), $x! = y$ (x diverso da y)

CONNETTIVI LOGICI

! (NOT), && (AND), || (OR)

Si applicano di solito a variabili booleane, cioè a proposizioni vere o false, ma anche a variabili numeriche (con la convenzione allora che 0=FALSE, e ogni altro numero è TRUE).
ESEMPI:

- !(cond) : se cond è vera restituisce FALSE, altrimenti TRUE
- (cond1)&&(cond2) : se cond1 e cond2 sono entrambe vere restituisce TRUE, altrimenti FALSE
- (cond1) || (cond2) : se almeno una tra cond1 e cond2 è vera restituisce TRUE, altrimenti FALSE

Se collego più di due condizioni, vengono valutate da sinistra a destra: A && B || C corrisponde a (A && B) || C, sarà quindi vera se A e B sono entrambe vere (qualunque sia C) o se C è vera (qualunque siano A e B)

Le funzioni matematiche

Molte funzioni matematiche sono predefinite, e si trovano nella libreria math.h che va quindi inclusa per poterle usare.
Alcuni esempi:

- sqrt(x) equivale a $\sqrt{x}$
- abs(x), fabs(x) equivalgono a $|x|$, la prima per gli interi, la seconda per i reali
- sin(x), cos(x), tan(x), atan(x) forniscono le consuete funzioni trigonometriche
- exp(x), log(x) forniscono funzione esponenziale e logaritmo naturale
- pow(x,y) equivale a x^y (funzione potenza con esponente reale)
- floor(x), ceil(x) (troncamento all'intero inferiore, arrotondamento all'intero superiore)

Ognuna di esse ha un tipo predefinito, e accetta argomenti del tipo predefinito.