Introduzione al Linguaggio C/C++

Passi fondamentali del C

- Definito nel 1972 (AT&T Bell Labs) per sostituire l'assembler nella programmazione di sistemi operativi: in pratica, nato per creare **UNIX**
- Prima definizione precisa: Kernigham & Ritchie (1978)
- Prima definizione ufficiale: **ANSI C** (1983)

.. ma già nel 1980 ...

- erano in uso varie versioni di un linguaggio denominato “C con le classi”
- Erano le prime versioni di quello che sarebbe stato il C++
- Inventato, definito, ed implementato per la prima volta, da Bjarne Stroustrup http://www.research.att.com/~bs/
- Standardizzato nel 1998: ISO/IEC 14882
- Decisamente di successo: http://www.tiobe.com/

C/C++: linguaggio di alto livello

Il C è un linguaggio imperativo

C++

- Del linguaggio C++ vedremo solo il sottoinsieme procedurale
- NON vedremo la programmazione ad oggetti
- Sarà argomento di esami futuri

Caratteristiche del C/C++

- Linguaggio *sequenziale*, ***imperativo***, ***strutturato a blocchi***
- Usabile anche come linguaggio di sistema → adatto a software di base, sistemi operativi, compilatori, ecc.
- Portabile, efficiente, sintetico (ma a volte poco leggibile...)

- Basato su pochi concetti elementari:
  - **espressione**
  - **dichiarazione / definizione**
  - **istruzione / blocco**
  - **funzione**
- tuttavia, viene arricchito da un vasto insieme di librerie di funzioni (per operazioni matematiche, di input/output, su stringhe, ecc.)

Espressioni letterali

Espressioni letterali

- Denotano valori costanti
- Spesso chiamate semplicemente **letterali** o **costanti senza nome**
- In C/C++ possono essere costanti carattere, costanti stringa, numeri interi, numeri reali

Espressioni letterali: numeri

Numeri interi	**6**	**12**	**700**
Numeri reali	**24.0**	**2.4e1**	**240.0e-1**

Costanti carattere

- Una ***costante*** carattere è un'astrazione simbolica di un carattere

Esempio: 'A' 'c' '6'

Anche:

- caratteri speciali: '\n', '\t', '\"', '\\', '\'"
- caratteri indicati tramite codice ASCII: '*nnn*', '\0x*hhh*'
  (*nnn* = numero ottale, *hhh* = numero esadecimale)

  '\041' '\0' 240.0E-1

Stringa di caratteri ≠ carattere **[SI VEDRA' IN SEGUITO]**

- "ciao" "Hello\n" "" *(stringa nulla)*

Dati

Tipi di dato

- In C/C++ (come in tutti i linguaggi di programmazione) a ciascun dato è associato anche il **TIPO**, ovvero la classe di valori che il dato può assumere nel corso dell'esecuzione del programma (e quindi gli operatori applicabili al valore in essa contenuto)

Quali sono i tipi di dato ammissibili nel linguaggio C?

Tipi di Dato Primitivi (“*base*”)

4 tipi di dato primitivi

char	(caratteri)
int	($\subset$ interi)
float	($\subset$ reali)
double	($\subset$ reali in doppia precisione)

Tipo “int”

- **Il tipo “int” è ben diverso dal tipo INTERO inteso in senso matematico** dove

  INTERO **Z** {-∞,…,-2,-1,0,+1,+2,…,+∞}

- Ovvero il tipo “int” ha un insieme di valori limitato a priori:
  - *L’insieme dei valori dipende dalla macchina*
  - Normalmente il tipo “int” è memorizzato in una PAROLA DI MACCHINA (***WORD***), che tipicamente è lunga (16), 32 o 64 bit
  - Se macchina a 16 bit: $[-2^{15}, 2^{15}-1]$ ovvero [-32768, +32767]
  - Se macchina a 32 bit: $[-2^{31}, 2^{31}-1]$ ovvero [-2147483648, +2147483647]
- Con i valori di tipo “int” è possibile effettuare solo un certo tipo di operazioni (ovvero, applicare solo gli operatori “int”)

Operatori “int” aritmetici

Al tipo **int** (e ai tipi ottenuti da questo mediante qualificazione) sono applicabili i seguenti operatori:

+	Addizione	
-	Sottrazione	
*	Moltiplicazione	
/	Divisione intera (≠divisione!)	Es., 10/3 = 3
%	Modulo (resto della divisione intera)	Es., 10%3 = 1

Operatori “int” relazionali

== Operatore relazionale di uguaglianza
(simbolo = denota l’operazione di assegnamento!)

!= Operatore relazionale di diversità
> Operatore relazionale di maggiore stretto
< Operatore relazionale di minore stretto
>= Operatore relazionale di maggiore-uguale
<= Operatore relazionale di minore-uguale

Restituiscono: 0 se falso, ≠ 0 se vero

Operatori “int” (*cont.*)

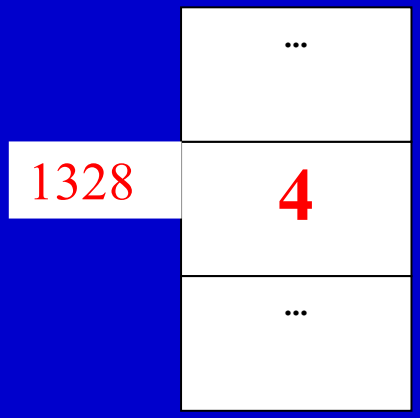
abs(n)	Valore assoluto di n (in C, previa inclusione dell’header file <***math.h***>)
n++	Successivo di n (*notazione postfissa*)
++n	Successivo di n (*notazione prefissa*)
n--	Precedente di n (*notazione postfissa*)
--n	Precedente di n (*notazione prefissa*)

Variabili

- **Una *variabile* è un'astrazione della cella di memoria**
- **Una dichiarazione di (*oggetto)* variabile associa permanentemente un oggetto ad un identificatore**
- Formalmente, una **variabile** è un **simbolo** associato a un oggetto, che ha
- un *indirizzo fisico* (**L-VALUE**)

simbolo	*indirizzo*
x	**1328**

- ed un *valore* (**R-VALUE**)
- perciò, l' R-VALUE di x è attualmente 4:



Dichiarazione di (oggetti) variabili

Scopo:

- Elencare tutte le *variabili* che saranno utilizzate nella parte esecutiva
- Attribuire ad ogni variabile un **tipo**
- Esempio:
  nome_tipo nome_variabile ;

- E' possibile raggruppare le dichiarazioni di più variabili dello stesso tipo in una lista separata da **,**
- Esempio:
  nome_tipo nome_variabile1, nome_variabile2 ;

Dichiarazione di variabili (*int*)

Esempio

```
int a;
int b, c;
int k=5; // inizializzata a 5
```

Cercare di utilizzare sempre nomi significativi per le variabili!

Evitare: *a*, *b*, *c*, *k*, …

Caratteristiche del C/C++ (*cont.*)

IDENTIFICATORI

<Identificatore> ::= <Lettera> { <Lettera> | <Cifra> }

- **Lettera** include tutte le lettere, maiuscole e minuscole, e l'underscore "_" (utilizzabile all'inizio solo per *oggetti di sistema*)
- Maiuscole e minuscole sono *diverse* (linguaggio C è *case-sensitive*)

PAROLE RISERVATE

- Esempio: **int, float, if, for, do, while, ...**
  { } delimitatore di blocco

COMMENTI

// commento, su una sola riga

/ commento, anche su più righe */* (non possono essere innestati)

Dichiarazione di oggetti costanti

- Una dichiarazione di (*oggetto) costante* associa permanentemente un valore ad un identificatore
- Si attribuisce ad ogni costante un **tipo**
  - Si utilizza la parola riservata `const`
- Il **tipo** si deduce dalla costante assegnata
  - Si utilizza la direttiva al pre-processore `#define`

Esempi

```
const  int  N = 100;
const  float  pigreco = 3.1415;
const  char  sim = 'A';

#define M 1000
#define Ultima_lettera 'Z'
```

Costanti e Variabili

- Una ***costante*** è un'astrazione simbolica di un valore
- L'associazione simbolo-valore non cambia mai durante l'esecuzione

- Una ***variabile*** è un **simbolo** associato a un *indirizzo fisico* (**L-VALUE**) che contiene un valore (**R-VALUE**)
- L'associazione **simbolo-indirizzo** non cambia mai durante l'esecuzione, ma può cambiare l'associazione **indirizzo-valore**
- Pertanto, nel caso di variabile, ad uno stesso simbolo possono corrispondere valori differenti in diversi momenti dell'esecuzione del programma

Dichiarazioni e definizioni

- Le dichiarazioni sono costrutti del linguaggio che introducono nuove entità (tipi, oggetti, funzioni, ...)
- Se una dichiarazione comporta, da parte del compilatore, l'associazione di locazioni di memoria o azioni eseguibili all'entità la si denota come una **definizione**
- Le dichiarazioni che abbiamo visto finora sono tutte definizioni
- Esempio:
  int N;

simbolo	*indirizzo*
N	**1600**

L'esecuzione di una **definizione** provoca l'allocazione di uno spazio in memoria equivalente a quello necessario a contenere un dato del tipo specificato

Struttura programmi

Struttura dei programmi C

Un programma C/C++ deve essere contenuto in uno o più file (salvo diversa specifica, per ora si assume in un solo file**)**:

- Direttive per il pre-compilatore (detto, ***pre-processore***): **#** tipicamente per l'inclusione di librerie (**#include**) e per la definizione di costanti (**#define**)
- L'identificatore della funzione speciale **`main()`**
- Eventuali altre funzioni **`funz1()`**, **`funz2()`**, …
- Eventuali dichiarazioni al di fuori delle funzioni

Struttura dei programmi C – Es. 1

Esempio

```
#include <stdio.h>

main()
   { dichiarazioni;
     istruzioni;
   }
```

Direttive al pre-processore

Parte dichiarativa

Parte esecutiva

Ciascuna **dichiarazione** è separata dall'altra dal **;**

Ciascuna **istruzione** è separata dall'altra dal **;**

Non vi sono altri simboli speciali per separare parte dichiarativa da parte esecutiva

Struttura dei programmi C++ – Es. 1

Esempio

```
#include <iostream>
```


Direttive al pre-processore

```
main()
   {
    dichiarazione o istruzione;
    dichiarazione o istruzione;
    ...
  }
```

Funzione main ()

- `main()` è una funzione speciale con tre caratteristiche:
  - deve essere sempre presente
  - è la prima funzione che viene eseguita ovunque si trovi all'interno del file (stessa cosa vale nel caso di programma su più file)
  - quando termina l'esecuzione del **main()**, termina il programma
- In C la funzione `main()` contiene due sezioni, racchiuse entro il ***blocco*** denotato da parentesi `{ }`
  - **Parte dichiarativa**
  - **Parte esecutiva**

Parole chiave della lezione

- **Linguaggio C/C++**
- **Tipi di dato**
- **Tipo int**
  - Range di valori
  - Operatori
- **Dichiarazioni**
- **Costanti e Variabili**
- **Struttura programma linguaggio C/C++**
- **Direttive al pre-processore**
- **Funzione main()**