Il Linguaggio C

Caratteristiche

- Linguaggio sequenziale (lineare), imperativo, strutturato a blocchi
- usabile anche come linguaggio di sistema
  - software di base
  - sistemi operativi
  - compilatori
  - …

Caratteristiche

- basato su pochi concetti elementari
  - dati (tipi primitivi, tipi di dato)
  - espressioni
  - dichiarazioni/definizioni
  - funzioni
  - istruzioni/blocchi

Esempio di programma in C

```
#include <stdio.h>
int main() {
    printf(" Hello World!! ");
    return 0;
}
```


Dati

- Un elaboratore è un manipolatore di simboli
- L'architettura fisica di ogni elaboratore è intrinsecamente capace di trattare vari domini di dati detti *tipi primitivi*
  - dominio dei numeri interi
  - dominio dei caratteri
  - dominio dei numeri reali
  - dominio delle stringhe di caratteri

Tipi di dato primitivi in C

- Caratteri
  - char caratteri ASCII (A<->65, {<->123, ..), interi ([0,255], [-127,128]
- Interi con segno (più lunghi di 8 bit)
  - short int long
- Naturali (interi senza segno)
  - unsigned short unsigned unsigned long
- Reali
  - float double long double
- I dati Booleani non esistono in C come tipo primitivo, si usano gli interi:
  - 0 indica falso
  - 1 indica vero (in realtà qualsiasi valore diverso da 0 indica vero)

Variabili

- Consentono di aumentare notevolmente la potenza espressiva.
- Una variabile è caratterizzata da:
  - Un nome
  - Un valore modificabile
- Il risultato di un'espressione contenente delle variabili si ottiene sostituendo ad ogni variabile il suo valore.
- L'assegnazione (simbolo '=' in C) consente di modificare il valore di una variabile

Variabili e Costanti

- **Inizializzazione variabile** : è possibile assegnare un valore iniziale ad una variabile al momento della sua dichiarazione (obbligatoria! Non si possono usare variabili che non siano state dichiarate)
  - `int a = 0, b = 100;`
  - `char a_capo = '\n';`
- **Costanti**: è possibile dichiarare un dato come costante; il compilatore rifiuta un qualunque assegnamento successivo effettuato su di esso
  - `const float Pi_greco = 3.1415927;`

Costanti di tipi primitivi

- Caratteri
  - singolo carattere racchiuso fra apici
    - 'A' 'C'
  - caratteri speciali
    - '\n' a capo
    - '\t' tabulazione
    - '\'' apice

Stringhe

- Una stringa è una collezione di caratteri delimitata da virgolette
  - "ciao" "hello\n"
- In C le stringhe sono semplici sequenze di caratteri di cui l'ultimo sempre presente in modo implicito è '\0' (codificato come valore 0 su 8 bit)
  - "ciao" equivale alla sequenza {'c','i','a','o','\0'}

Espressioni

- Il C è un linguaggio basato su espressioni
- Una espressione è una notazione che denota un valore mediante un processo di valutazione
- Una espressione può essere semplice (una costante, un simbolo di variabile) o composta
  - ogni linguaggio introduce un insieme di operatori che permettono di aggregare altre espressioni (operandi) per formare espressioni composte
  - esempi : 4*8-2*arcsin(x) a&&(b||c)

Classificazione degli operatori

- In base al tipo di operandi (aritmetici, logici, relazionali)
- In base al numero degli operandi (unari, binari, ternari..)

Operatori aritmetici

Operazione	operatore	C
Inversione di segno	Unario	-
Somma	Binario	+
Differenza	Binario	-
Moltiplicazione	Binario	*
Divisione fra interi	Binario	/
Divisione fra reali	Binario	/
Modulo (fra interi)	Binario	%

Operatori relazionali

Relazione	C
Uguaglianza	==
Diversita`	!=
Maggiore di	>
Minore di	<
Maggiore o uguale a	>=
Minore o uguale a	<=

Espressioni e operatori logici

- Anch'esse denotano un valore intero da interpretare come vero (1 o comunque ?0) o falso (0)

Connettivo logico	operatore	C
Not	Unario	!
And	Binario	&&
Or	Binario	||

Espressioni condizionali

- Una espressione condizionale è introdotta dall'operatore ternario
  - *condiz ? espr1 : espr2*
- L'espressione denota
  - o il valore denotato da *espr1*
  - o quello denotato da *espr2*

  in base al valore della espressione *condiz*
  - in particolare se *condiz* è vera il valore assunto dall'espressione nel suo complesso è *espr1*

Espressioni condizionali : esempi

- 3 ? 10 : 20
  - denota sempre 10 (3 è sempre vera)
- x ? 10 : 20
  - denota 10 se x è vera (diversa da zero)
  - oppure 20 se x è falsa
- (x>y) ? x : y
  - denota il maggiore fra x e y

Struttura di un programma C

- La struttura di un programma C è definita

< programma >::=
{< unità-di-codifica >}
< main >
{< unità-di-codifica >}

Struttura di un programma C

- La parte <main> è obbligatoria ed è definita

```
< main >::=
int main()
{
  [< dichiarazioni-e-definizioni >]
  [< sequenza-istruzioni >]
}
```


Struttura di un programma C

- <dichiarazioni-e-definizioni>
  - introducono i nomi di costanti, variabili, tipi definiti dall'utente
- <sequenza-istruzioni>
  - sequenza di frasi del linguaggio ognuna delle quali è un'istruzione
- Il *main()* è una particolare unità di codifica *(una funzione)*

Caratteri e identificatori

- Set di caratteri
  - caratteri ASCII
- Identificatori
  - sequenze di caratteri tali che

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


Commenti

- Sequenze di caratteri racchiuse fra /* e */

Es. /* Questo è un commento */

- Non possono essere annidati

/* Questo è /* Questo è un commento dentro un altro commento */ un ERRORE */

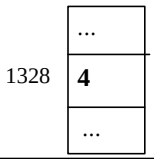
Variabile

- E' un'astrazione della cella di memoria
- Formalmente, è un simbolo associato ad un indirizzo fisico

Simbolo	indirizzo
X	1328

Variabile

- L'indirizzo fisico è fisso e immutabile, cambia il suo contenuto cioè il valore di x
- esempio: x=4



Definizione di variabile

- E' la frase che introduce una nuova variabile
  - identificata da un simbolo
  - atta a denotare valori di un ben preciso tipo

Esempi

- Definizione di una variabile
- <tipo><identificatore>
- int x; /* deve denotare un valore intero*/
- float y; /* deve denotare un valore reale*/
- char ch; /* deve denotare un valore carattere*/

Inizializzazione di variabili

- E' possibile specificare il valore iniziale di una variabile quando la si dichiara
- <tipo><identificatore> = <espr> ;
- int x = 32;
- double speed = 124.6;
- double time = 71.6;
- double km = speed*time;
  /* inizializzazione mediante una espressione*/

Caratteristiche delle variabili

- Campo d'azione (scope): è la parte di programma (unità di codifica) in cui la variabile è nota e può essere usata
- Tipo: specifica la classe di valori che la variabile può assumere (e quindi gli operatori applicabili)
- Tempo di vita: l'intervallo di tempo in cui rimane valida l'associazione simbolo/cella di memoria
- Valore: è rappresentato (secondo la codifica adottata per il tipo cui la variabile appartiene) nell'area di memoria associata alla variabile.

Esempio

- Problema
  - data una temperatura espressa in gradi Celsius , calcolare il corrispondente valore espresso in Fahrenheit
- Approccio:
  - si parte dal problema e dalle proprietà nel dominio di dati
- Specifica della soluzione:
  - relazione tra grandezze esistenti nello specifico dominio applicativo
  - c*9/5=f-32

Esempio: algoritmo di risoluzione

- Dato c
- calcolare f sfruttando la relazione f=32+c*9/5

Solo a questo punto (una volta definito l'algoritmo) si effettua la codifica

Un possibile programma in C

```
main()
{
  float c;            /*Celsius*/
  scanf(“%f”, &c);
           /* legge il valore di c */
  float f=32+c*9/5;
  printf ( “Temperatura (F): %f \n”, f);
          /* stampa il valore di f */
}
```

NB l’impaginazione (indentazione) serve solo per rendere più leggibile il programma: in C le istruzioni sono separate da “;”

Costruzione di una applicazione

- Si deve compilare il file (o i file) che contiene (contengono) il testo del programma (*file sorgente, estensione .c)*
- Il risultato sono uno o più file *oggetto (estensione .o (Unix) o .obj (Windows))*
- si devono poi collegare (*linking*) i file oggetto l’uno con l’altro e con le librerie di sistema al fine di creare un unico file *eseguibile (estensione .exe (Windows); nome a piacere o a.out se non si specifica il nome (Unix))*

Perché?

- L’elaboratore capisce solo il linguaggio macchina della CPU utilizzata
- il nostro programma opera su una macchina ‘astratta’, ‘rivestita’ del sistema operativo che controlla le periferiche (stampante, memoria di massa, ecc.)
- alcune istruzioni complesse potrebbero essere dei mini-programmi forniti insieme al compilatore che le ingloba quando occorre

Librerie di sistema

- Insieme di componenti software che consentono di interfacciarsi col sistema operativo, usare le risorse da questo gestite e realizzare alcune “istruzioni complesse” del linguaggio

Eseguire un programma

- Una volta scritto, compilato (con il compilatore) e collegato (con il linker) lo si può lanciare sull’elaboratore

E se non funziona?

- Debugger: strumento in grado di eseguire passo passo il programma, mostrando le variabili e la loro evoluzione e tenendo traccia delle funzioni via via chiamate

Debugger

- **E’ possibile**
  - eseguire il programma riga per riga entrando anche dentro le funzioni chiamate
  - eseguire fino ad una certa riga
  - controllare istante per istante quanto vale una variabile
  - vedere istante per istante le funzioni attive

Ambienti integrati di programmazione

- Automatizzano la procedura di compilazione e linking dei file
- Possono lanciare il programma sulla macchina e visualizzare l'output a video
- Incorporano le funzioni di debug

Noi utilizzeremo il **BorlandC della Borland** presente nei laboratori di base