Corso Programmazione 2008-2009

(docente)
Fabio Aiolli
E-mail: aiolli@math.unipd.it
Web: www.math.unipd.it/~aiolli

(docenti laboratorio)
A. Ceccato, F. Di Palma, M. Gelain

Dipartimento di Matematica Pura ed Applicata
Torre Archimede, Via Trieste 63

PARTE 3

Variabili, Assegnamenti e Scope

Variabili

- Le variabili sono caratterizzate da
  - Nome (p.e. base, altezza) che la identifica
  - Tipo (p.e. float, int, char) che ne determina i possibili valori che può contenere e le possibili operazioni
  - Valore (p.e. 3.14, 7, ‘c’) il valore effettivo contenuto nello spazio di memoria della variabile
- Il nome può essere una qualsiasi sequenza di caratteri (senza spazi) con alcune eccezioni:
  - Il primo carattere è sempre alfanumerico (3gino, 44a)
  - No segni di interpunizione (a^3, pippo+pluto) solo il carattere “_” (underscore) è permesso
- I tipi fondamentali sono tre: numero intero (int), numero reale (float), carattere (char).

Utilizzo delle variabili

- In C le variabili vanno prima di tutto DICHIARATE

  int a;

  float b;

  La dichiarazione serve in fase di esecuzione per “allocare” uno spazio di memoria adatto a contenere il valore della variabile (indicando il tipo) e ad associargli un nome.

- Generalmente vanno anche INIZIALIZZATE mediante un assegnamento

  a = 3;

  b = 3.14;

- Infine le si possono anche usare all’interno di espressioni usando opportuni operatori

L'area del rettangolo

```
/* Calcolo aerea rettangolo */

#include<stdio.h>

main() {
  int base;
  int altezza;
  int aerea;

  base = 3;
  altezza = 7;
  area = base*altezza;

  printf("%d\n", area);
}
```

Dichiarazioni

Le costanti

- In alcuni casi, sappiamo a priori che l'informazione da memorizzare non cambia durante l'esecuzione del programma.

  P.e. Pi_greco = 3.14, LaEdiEulero= 2.8, …

- In questi casi, è più corretto (ma non è obbligatorio) utilizzare le cosiddette costanti

  #define Pi_greco = 3.14

  N.B. Nella definizione di costante la dichiarazione e l'inizializzazione sono contemporanee e non ci mettiamo il ';'

Input e Output

- Standard Output (a schermo)

  `printf(<stringa_formato>,exp1,exp2,…)`

  `\n` a linea nuova

  `\t` salta una tabulazione

  `\\` stampa il carattere ‘\’

  `\”` stampa il carattere ‘”’

  Segnaposti:

  `%d` intero

  `%f` reale

  `%c` carattere

- Standard Input (da tastiera)

  `scanf(<stringa_formato>,&var1,&var2,…)`

La libreria standard

Input e Output	<stdio.h>
Test dei caratteri	<ctype.h>
Funzioni su stringhe	<string.h>
Funzioni Matematiche	<math.h>
Funzioni Utilita'	<stdlib.h>
Funzioni Diagnostiche	<assert.h>
Liste argiomenti variabile	<stdarg.h>
Salti non locali	<setjmp.h>
Segnali	<signal.h>
Funzioni Date e Ore	<time.h>
Limiti dei tipi di variabile	<limits.h>

Tipi di Dato

Un singolo byte	char
Numero Intero	int
Numero Intero corto	short
Numero Intero lungo	long int
Numero in virgola mobile (singola precisione)	float
Numero in virgola mobile (doppia precisione)	double
Numero in virgola mobile (precisione estesa)	long double
Interi con segno	signed {char/int}
Interi senza segno	unsigned {char/int}

Sequenze di Escape

\a bell	\\ backslash
\b backspace	\? punto interrogativo
\f formfeed	\' apice singolo
\n a capo	\" apice doppio
\r ritorno carrello	\ooo carattere ottale
\t tabulazione orizzontale	\xhh carattere esadecimale
\v tabulazione verticale	

Scope delle variabili

- I blocchi di istruzioni in C vengono definiti mediante le parentesi graffe.
- I blocchi possono essere *annidati* e se ne possono definire un numero arbitrario

- Le variabili sono *visibili* solo dal momento della dichiarazione fino al termine del blocco dove sono dichiarate e quindi anche in tutti i blocchi più interni
- Quindi via via che il programma procede solo alcune delle variabili dichiarate nell'intero programma sono visibili e utilizzabili.
- Attenzione: una variabile dichiarata dentro un blocco con lo stesso nome di una variabile dichiarata in un blocco più esterno, la *nasconde!*

Dichiarazioni di variabili
tipo nome_della_variabile [= inizializzazione];

Esempi:

- int a;
- int a = 2;
- int a, b;
- float f = 2.345;
- char c = 'X';
- char messaggio[] = “Questa e' una stringa”;

Facendo precedere il tipo dal qualificatore **const**, si ottiene una “variabile costante”, il cui valore non potrà essere cambiato

const double pi = 3,141592653;

Operatori

Operatori senza effetti collaterali

- Aritmetici: +, -, *, /, %
- Relazionali: >, >=, <, <=, ==, !=
- Logici: &&, ||, !
- bitwise: &, |, ^, <<, >>, ~

Operatori con effetti collaterali

- Incremento e decremento: ++, --
- Assegnazione: =, +=, -=, *=, ecc.

Condizionale ternario: expr1 ? expr2 : expr

Conversioni di Tipo

Quando un'espressione coinvolge valori di tipo diverso, essi vengono convertiti ad un tipo comune:

- Automaticamente ad un tipo “più grande”, che quindi non fa perdere informazione
- Esplicitamente, mediante un operazione di cast. Obbligatoria se non e' possibile una conversione implicita automatica.

Esempio di Scope

```
01: /* Esempio Scope */
02: #include<stdio.h>
03:
04: main() {
05:  int x;
06:  x=1;              // x in 05:
07:  {
08:    {
09:      x=x+2;       // x in 05:
10:    }
11:    int x=3;       // x in 11:
12:  }
13:  int x; // ILLEGALE!!!
14:  x=x*3;            // x in 05:
15: }
```

If (else)

```
if (condizione) {
   /* blocco eseguito se la
   condizione e' vera */
} else {
   /* blocco eseguito se la
   condizione e' falsa */
}
```

Operatori Logici

Expr1	Expr2	Expr1 && Expr2	Expr1 || Expr2	! Expr1
Zero	Zero	0	0	1
Zero	Non Zero	0	1	1
Non Zero	Zero	0	1	0
Non Zero	Non Zero	1	1	0

Valutazione short-circuit

La valutazione degli operatori logici && e || avviene in maniera short-circuit, cioè da sinistra verso destra e si interrompe non appena il risultato diventa noto.

```
(a != 0) && (++i < 10)
(a!=0) || (++i < 10)
```

Switch Case

```
switch (espressione) {
  case espr-costante1: istruzioni1
  case espr-costante2: istruzioni2
  case espr-costante3: istruzioni3
  case espr-costante4: istruzioni4
  ...
  default: istruzioni_default
}
```