UNIVERSITÀ DI PARMA

Tipi di dato in C

What's in a name? That which we call a rose,
By any other name would smell as sweet.

William Shakespeare, Romeo and Juliet,
Act II, Scene II)

Sommario

- Definizione variabile
- Tipi di dato in C
- Dati scalari
  - Virgola mobile
  - Interi
- Combinazione differenti tipi di dato
  - Conversioni
- Campo di visibilità delle variabili

Variabile

- Qualcosa che “varia”...
- Definizione:
  - Spazio in memoria dotato di nome in cui è possibile scrivere e leggere dati durante l’esecuzione del programma
- Spazio in memoria $\longrightarrow$ indirizzo e dimensione
  - Gestiti automaticamente in linguaggi di alto livello

Terminologia

- Contenuto ovvero valore di una variabile → right value o rvalue
- Indirizzo in memoria → left value o lvalue
- Spazio occupato

```
int a = 173;
```

Nome di una variabile

- Il nome di una variabile o identificatore può essere costituito da lettere, cifre o underscore "_"
- il primo carattere **deve** essere una lettera
- lettere minuscole o maiuscole sono differenti
- non può essere una parola chiave

Nome di una variabile

- Alcune convenzioni:
  - il nome di una variabile è normalmente minuscolo
  - il nome deve essere mnemonico
  - quando il nome è formato da piú parole è opportuno usare l'underscore o alcune lettere maiuscole

```
int somma           = 173;
int numero_studenti = 144;
int NumeroAula      = 10;
```

Definizione di una variabile

UNIVERSITÀ DI PARMA

```
// le variabili si definiscono come
// <tipo> <nome_variabile>;

int a;
  // definisco variabile di tipo “int”
  // a contiene valore casuale
  // in C la definizione NON inizializza

int b=17;
  // definisco E inizializzo variabile di tipo “int”
```

C99 Albero dei Tipi di Dato

C99 Albero dei Tipi di Dato

Tipi di dato a virgola mobile

- Rappresentazione approssimata dei numeri reali
  - Positivi e negativi con e senza decimali
- Perchè approssimata?
  - Massimo e minimo valore rappresentabile
  - Limite al numero di cifre decimali

Tipi di dato a virgola mobile

UNIVERSITÀ DI PARMA

- Tre tipi
  - float → precisione singola
  - double → precisione "doppia"
  - long double → precisione estesa
- Cosa cambia?
  - numero di byte utilizzati

Float vs Double vs Long Double

- Quale tipo scelgo?
- Come sempre compromesso:
  - Precisione
  - Velocità esecuzione
  - Occupazione memoria

Tipi di dato a virgola mobile

- Il C stabilisce di usare lo standard IEEE 754?
  - No e non è importante
- Il C fissa quanti bit vengono usati?
  - No
- Dipende tutto dal compilatore
  - In alcuni casi non vi è differenza tra float e double

sizeof()

- Il C fornisce un sistema per capire quanto spazio occupa una variabile
- sizeof(<tipo di dato>)
  - Restituisce il numero di byte occupati da quel tipo di dato
  - Tecnicamente è una macro
- Per stampare prefisso ‘z’ per gli specificatori di formato
  - %zd

Tipi di dato a virgola mobile

UNIVERSITÀ DI PARMA

- Come li stampiamo o leggiamo?
  - Formati
    - %f → formato decimale 123.456000
    - %e → formato scientifico 1.234560e+02
    - %E → formato scientifico 1.234560E+02
    - %g → rappresentazione piú breve tra %f e %e 123.456
  - Suffissi per tipo
    - "niente" → float
    - l → double (obbligatorio solo per input)
    - L → long double

Tipi di dato a virgola mobile

- Valori particolari
- ±inf ⟶ Infinite
  - Può essere il risultato di alcune operazioni
  - Esempio: divisione per zero
- NaN ⟶ not a number
  - Di fatto non capita se la variabile è inizializzata correttamente
- Funzioni per controllo
  - isnan(), isinf()...

C99 Albero dei Tipi di Dato

Tipi di dato interi

UNIVERSITÀ DI PARMA

- Quattro+1 tipi:
  - char → in realtà si usa raramente per “numeri”
  - short int
  - int → il piú usato
  - long int
  - long long int
- Anche qui cambia dimensione memoria occupata
  - Dati n bit riesco a rappresentare numeri non negativi tra 0 e 2^n-1

Tipi di dato interi

- Complicazione: il segno!
- Sono tutti con segno ad eccezione di...
  - char
  - Che a seconda del compilatore può o meno avere segno
- Nel definirli posso cambiare comportamento
  - unsigned
  - (signed)

Tipi di dato interi (esempio 64 bit)

UNIVERSITÀ DI PARMA

Type	Byte	Smallest Value	Largest Value
signed char	1	-128	127
unsigned char	1	0	255
short	2	-32.768	32.767
unsigned short	2	0	65.535
int	4	-2.147.483.648	2.147.483.647
unsigned int	4	0	4.294.967.295
long	8	-9.223.372.036.854.775.808	9.223.372.036.854.775.807
unsigned long	8	0	18.446.744.073.709.551.615
long long	8	-9.223.372.036.854.775.808	9.223.372.036.854.775.807
unsigned long long	8	0	18.446.744.073.709.551.615

Tipi di dato interi

UNIVERSITÀ DI PARMA

- Come li stampiamo o leggiamo?
  - Formati
    - %d → formato decimale -45
    - %x → formato esadecimale fffffd3
    - %X → formato esadecimale FFFFFFD3
    - %u → formato decimale senza segno 4294967251
  - Suffissi per tipo
    - "niente" → int
    - hh → char
    - h → short
    - l → long
    - ll → long long

Tipi di dato interi

UNIVERSITÀ DI PARMA

- Overflow
  - Avviene quando vado oltre i limiti possibili
  - Non viene fornito errore!
    - Perdo riporto o prestito

C99 Albero dei Tipi di Dato

Il tipo char

UNIVERSITÀ DI PARMA

- È un tipo numerico intero a tutti gli effetti
  - 0–255
  - -128–127
- Di fatto usato solo per codice ASCII
- Specificatore di formato
  - %c
- Costanti:
  - Numero diretto, esempio 65
  - Indicazione simbolo tramite apici singoli, esempio ‘A’

Do not
even think
to do that!

Il tipo char e scanf()

- Leggere da tastiera i char (e derivati) richiede qualche cautela
- Tutto ciò che viene premuto finisce in input al programma
  - Incluso, ad esempio, eventuali caratteri di invio
  - Buffer di input

```
fscanf("%c", &x); // leggo eventuale invio di inserimento precedente

Fscanf(" %c", &x); // salto eventuali spaziature ('invii' inclusi!)
```

C99 Albero dei Tipi di Dato

Il tipo _Bool

- Specifico del C99
- Tipo di dato intero senza segno pensato per memorizzare solo "vero" e "falso"
  - 0 ⟶ falso
  - 1 ⟶ vero
- Quindi assume solo valori 1 o 0

C99 Albero dei Tipi di Dato

Combinazione di tipi differenti

UNIVERSITÀ DI PARMA

- 3 tipi a virgola mobile 4–5 tipi interi
- Come si comporta il C quando li combiniamo?
- Conversioni
  - Assegnamento
  - Implicite o automatiche

Conversioni di assegnamento

UNIVERSITÀ DI PARMA

- In caso di assegnamento (=)
  - Ciò che è a destra (rvalue) viene convertito in ciò che ho a sinistra (lvalue)
- Cautela
- Potenzialmente
  - Perdo informazione esempio: float $\longrightarrow$ int
  - Ottengo risultati sbagliati esempio: long long $\longrightarrow$ short

Conversioni implicite o automatiche

- Quando combino tipi differenti in espressioni
- Operandi binari richiedono operatori dello stesso tipo
  - Aritmetici: *, /, %, +, -; confronti: <, >, <=, >=, ==, !=; bit a bit: &, ^, |,
- Esempio: a+b se a è int e b è float che tipo di risultato fornisce?
  - Spoiler: float
- Regole di conversione
- <u>Alta probabilità di errore!</u>

Regole conversioni implicite

UNIVERSITÀ DI PARMA

- Promozione implicita a int
  - _Bool, char e short ⟶ int
    - Oppure unsigned int
  - Anche se operandi stesso tipo
- Piramide conversioni

Conversioni esplicite

- Casting
- Decido io la conversione
  - Meno facile sbagliare
- Formato:

```
(tipo) espressione
```

Visibilità delle Variabili

- Definita una variabile posso usarla ovunque?
  - No!
- Ambito di visibilità di una variabile o *scope*
  - Parte di codice in cui quella variabile è usabile
- 3 Casi:
  - Programma (variabili globali)
  - Singolo file
  - Blocco (variabili locali)

Variabili Globali

- Definite al di fuori di qualunque blocco
  - {} oppure definizione in for(;;)
- Posso modificarle in qualunque punto del codice
  - Dati NON protetti
  - Difficoltà debug
  - Incertezza
- Il voto di esame ne risente...
- Unico vantaggio ⟶ inizializzazione

Variabili Locali

- Definite
  - Interno blocchi ovvero dentro un {}
  - In un for(;;)
  - Argomenti funzioni
- Usabili solo dove definite
- Durata automatica
  - Create quando il flusso di esecuzione arriva in quel punto
  - Distrutte quando l'esecuzione esce da quella parte di codice

Global vs Local

- Variabili locali
  - Focus su parti di codice piccole
    - E autosufficienti!
  - Codice tipicamente piú comprensibile e gestibile
    - Modularità
  - Minori problemi di portabilità
- Variabili globali
  - Devo tener traccia di tutti i punti del codice dove sono usate/modificate
  - Se lavoro di gruppo devo coordinare gestione
  - Memoria sempre in uso!
  - Conflitti nomi

Durata Variabili

UNIVERSITÀ DI PARMA

- Una variabile non detto che sia per sempre
- Durata fissa (*static lifetime*):
  - Variabili globali
    - Memoria occupata ad inizio esecuzione e mai liberata
    - In questo caso inizializzazione a 0
- Durata automatica (*automatic lifetime*)
  - Variabili locali
    - La variabile viene creata ogni volta si accede a quella parte di codice
    - Distrutta ogni volta il flusso di esecuzione esce da quel blocco
    - Posso cambiare la durata? Sì → `static`

Le costanti

- Ci sono situazioni in cui si usano valori costanti
- Soluzione quick ma molto dirt:
  - Riempo il programma di “numeri magici”
  - Modifiche difficili, codice non parametrico
- Soluzione corretta:
  - Definisco costanti
  - Questo permette di ottenere codice parametrico e facilmente modificabile

Le costanti

- In C tre soluzioni per definire una costante:
- #define
  - Esempio: `#define N_MAX_STUDENTI (100)`
- const
  - Stessa sintassi variabili, obbligatoria inizializzazione
  - Sensata visibilità globale
  - Esempio: `const int N_MAX_STUDENTI=100;`

UNIVERSITÀ DI PARMA

Tipi di dato in C

What's in a name? That which we call a rose,
By any other name would smell as sweet.

William Shakespeare, Romeo and Juliet,
Act II, Scene II)