Programmazione

Giovanni Da San Martino

Dipartimento of Matematica, Università degli Studi di Padova
giovanni.dasanmartino@unipd.it
A.A. 2021-2022

UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Previously on Programmazione

- Algoritmo: Insieme ordinato e finito di istruzioni elementari, chiare e non ambigue, per risolvere un problema.

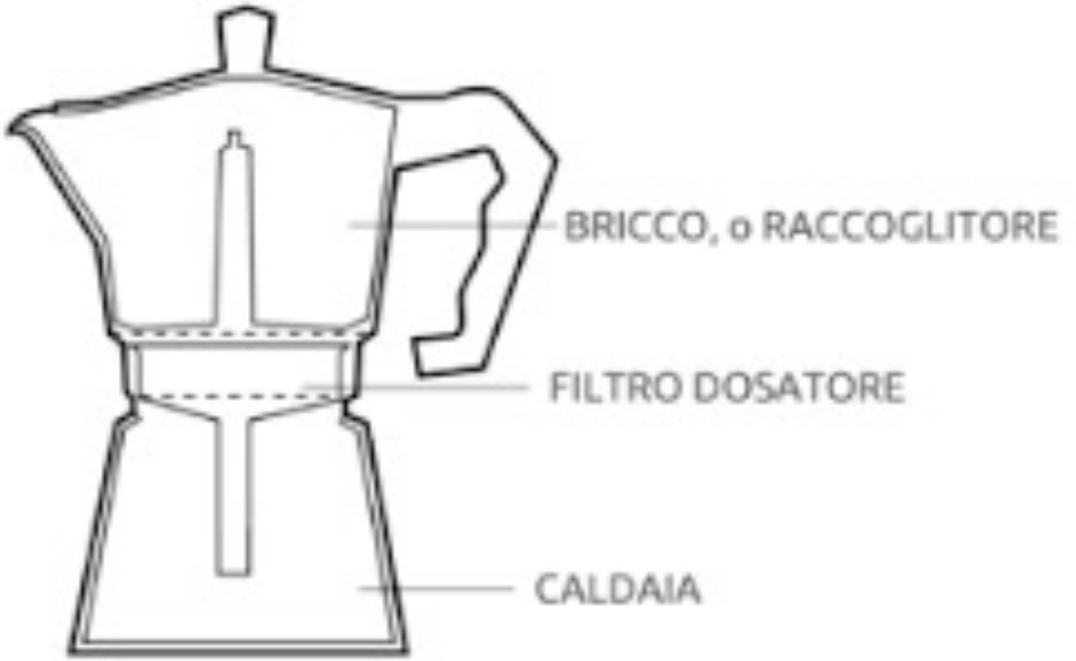
- Un algoritmo deve produrre un risultato, sempre lo stesso a partire dalle stesse condizioni iniziali

- Le risorse necessarie alla realizzazione dell'algoritmo devono essere "ragionevoli" per la tecnologia attuale
  - calcola tutte le cifre del Pi greco" (richiede tempo e memoria infiniti)

Previously on Programmazione

Fare il caffè con la moka

1. separare il bricco dalla caldaia svitando la moka e rimuovere il filtro dalla caldaia
2. aggiungere acqua nella caldaia
3. rimettere il filtro sopra la caldaia
4. aggiungere caffè in polvere al filtro fino a riempirlo
5. riavvitare il bricco
6. mettere la moka sul fornello acceso



Compilazione

Compilatore:

1. più veloce l'esecuzione rispetto ad un linguaggio interpretato
2. una volta compilato il codice, posso eseguirlo su ogni computer
3. non necessità del traduttore, ma ogni volta che cambio il programma devo ricompilarlo

Compilazione

Compilatore:

1. più veloce l'esecuzione rispetto ad un linguaggio interpretato
2. ~~una volta compilato il codice, posso eseguirlo su ogni computer~~
3. non necessità del traduttore, ma ogni volta che cambio il programma devo ricompilarlo

Traduzione

- Rimozione dei commenti
- #include <x>: il contenuto del file x viene copiato all'inizio del nostro file (x contiene informazioni su come eseguire comandi addizionali, ad esempio stdio.h permette di utilizzare il comando printf)
- Espansione delle macro (le vedremo tra qualche lezione)
- Compilazione condizionale (utile se alcune librerie hanno nomi diversi in diversi sistemi operativi)

Comandi di Base

UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Espressioni

- Il C supporta i tradizionali operatori aritmetici
- + - * / %
  - la precedenza tra gli operatori è quella usuale, ovvero ?
- % è resto della divisione intera: 5 % 2 = 1

- Cosa calcola un operatore dipende dal tipo degli operandi:
- 7/2 = 3 (divisione intera)
- 7.0/2.0 = 3.5 (divisione tra numeri reali)

Espressioni

- Il C supporta i tradizionali operatori aritmetici
- + - * / %
  - la precedenza tra gli operatori è quella usuale, ovvero */% > +-)
- % è resto della divisione intera: 5 % 2 = 1

- Cosa calcola un operatore dipende dal tipo degli operandi:
- 7/2 = 3 (divisione intera)
- 7.0/2.0 = 3.5 (divisione tra numeri reali)

Condizioni

- Espressioni condizionali:

==	uguale a	5 == 3 è Falso
>	maggiore di	5 > 3 è Vero
<	minore di	5 < 3 è Falso
!=	diverso da	5 != 3 è Vero
>=	maggiore o uguale di	5 >= 3 è Vero
<=	minore o uguale di	5 <= 3 è Falso

Operatori Logici

Operatore	Significato	Esempio
&& (binario)	AND logico. Vero solo se entrambi gli operandi sono Veri	((5==5) && (5<2)) è Falso.
|| (binario)	OR logico. Vero se almeno uno degli argomenti è Vero	((5==5) || (5<2)) è Vero
! (unario)	NOT logico. Vero se l'argomento è Falso	!(5==5) è Falso

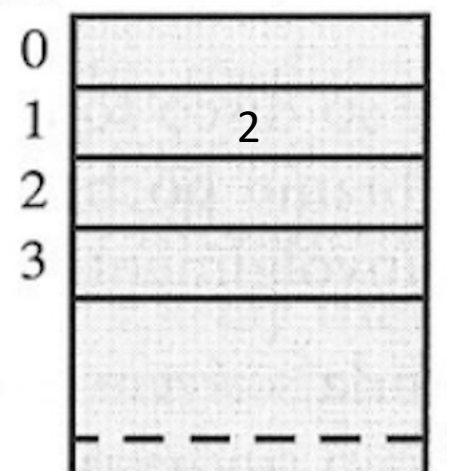
- ! ha precedenza su &&,||, che si applicano da sinistra a destra (come per le espressioni si usano le parentesi per indicare ordini di precedenza diversi)

Variabili

- Come in matematica, le espressioni e le condizioni possono essere generalizzate utilizzando simboli (variabili) al posto di alcuni valori, ad es. x*2 generalizza 2*2, 3*2,...
- Una variabile ha i seguenti attributi:
  - il nome (che definiamo noi)
  - l'area di memoria in cui è mantenuto il suo valore (non assegnata dall'utente)
  - il tipo: le variabili vengono usate per rappresentare numeri interi, reali, caratteri (in C è definito dall'utente).
    - Il tipo è un modo coinciso per dire quanta memoria occupa (dipende dall'architettura della macchina), come leggere o scrivere la sequenza di bit e quali operazioni posso fare con quella variabile.

Variabili ed Assegnamento

Variabile		
nome e tipo	L-valore Identificativo dell' area di memoria riservata alla variabile	R-valore il contenuto corrente della cella di memoria

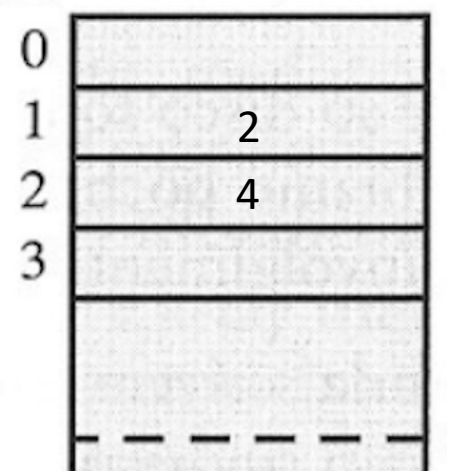


y: L-valore=1
R-valore=2

- L'operazione di assegnamento = permette di modificare il contenuto (valore) di una variabile:
- y = E; //vai alla cella di memoria indicata dall' L-valore di y e scrivici dentro il risultato della valutazione dell'espressione E
- y = 2; // vai alla cella di memoria indicata dall' L-valore di y e scrivici dentro il risultato dell'espressione alla destra dell'uguale, ovvero 2

Variabili ed Assegnamento

Variabile		
nome e tipo	L-valore Identificativo dell' area di memoria riservata alla variabile	R-valore il contenuto corrente della cella di memoria

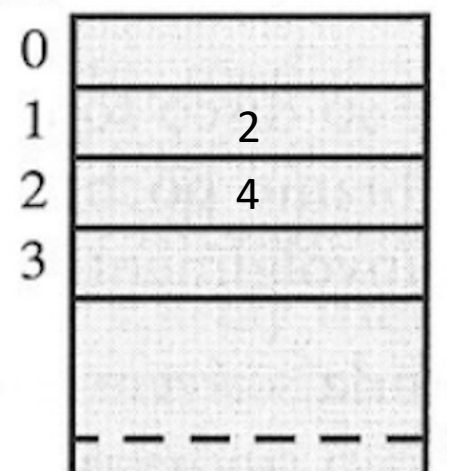


x: L-valore=2
R-valore=4

- y = 2;
- Notate che l'attributo selezionato della variabile (L o R valore) dipende da dove essa compare nell'istruzione:
- x = y + 2; // vai alla cella di memoria indicata dall' L-valore di y e scrivici dentro il risultato dell'espressione alla destra dell'uguale, ovvero il risultato della somma tra 2 e l'R-valore della variabile y: x=2+2=4
- x = x + 1 // ?

Variabili ed Assegnamento

Variabile		
nome e tipo	L-valore Identificativo dell' area di memoria riservata alla variabile	R-valore il contenuto corrente della cella di memoria



x: L-valore=2
R-valore=4

- y = 2;
- Notate che l'attributo selezionato della variabile (L o R valore) dipende da dove essa compare nell'istruzione:
- x = y + 2; // vai alla cella di memoria indicata dall' L-valore di y e scrivici dentro il risultato dell'espressione alla destra dell'uguale, ovvero il risultato della somma tra 2 e l'R-valore della variabile y: x=2+2=4
- x = x + 1 // x=4+1=5

Variabili

- In C è necessario dichiarare le variabili prima di usarle
  - int x; // dichiara una variabile di tipo intero
  - int x = 2; // dichiara una variabile di tipo intero ed inizializza il suo valore a 2
- Un legame tra una variabile ed un suo attributo si dice statico se è stabilito prima dell'esecuzione e non può essere cambiato in seguito, dinamico altrimenti:
  - il valore è un legame dinamico
  - In C il tipo è un legame statico (questo implica che il compilatore può identificare i seguenti tipi di errore: int x; x = "Ciao Mondo!";
- In C è possibile definire "variabili il cui valore è un legame statico", quelle che comunemente chiamiamo costanti (es. pi greco)
  - const int x = 3; // poiché non possiamo cambiare x, dobbiamo definirne il valore quando dichiariamo la variabile

Nomi di Variabili

- Nomi di variabili:
  - usiamo caratteri alfanumerici (a-zA-Z0-9 e _)
  - ma il nome non deve iniziare con 0-9 e _,
  - il C è case sensitive (ma evitatiamo di avere due variabili di nome VAR e var)
  - evitatiamo anche di avere variabili che assomigliano ad un comando o ad un elemento del linguaggio: IF, INT

- i nomi delle variabili devono essere il più possibile indicativi della loro funzione
  - ma evitate nomi troppo lunghi

Esercizio


```
/*
* Trasformare il valore in gradi farenheit della variabile fahr (X) nel
* corrispondente valore celsius (Y) arrotondato all'intero inferiore e stampare
* "X gradi farenheit corrispondono a Y gradi celsius"
*
* Ad esempio se fahr=78 stampa
* 78 gradi farenheit corrispondono a 25 gradi celsius
*
* Si ricorda che C = (5/9)(F-32)
*/
```

- Lavorate a gruppetti discutendo assieme le vostre soluzioni, avete indicativamente 5 minuti

```
if (condizione) {
        //comandi da eseguire se la condizione è vera
} else {
        //comandi da eseguire se la condizione è falsa
}
```

```
 if (x>=0)    //non serve { perché abbiamo un solo comando
    printf(”positivo");
  else
    printf("negativo");
```

MA

```
if (condizione1)
        if (condizione2)
                comando1;
else
        comando2;
```

Senza {} l'else fa riferimento all'if più vicino (condizione2)

IF e Blocco di Istruzioni

Varianti:

```
if (condizione) {
        //comandi da eseguire se la condizione è vera
}
```

condizione? valore_se_vero: valore_se_falso (all'interno di un espressione)

- int x = -2, y;
- y = 3+(x>0?x:-x); // y=5

Iterazione

```
while (condizione) {
        //comandi da eseguire se la condizione è vera
}
comando2
```

Il comando while:

1. se la condizione è falsa, non esegue i comandi all'interno del blocco e passa a comando2
2. se la condizione è vera, esegue i comandi all'interno del blocco
3. Una volta eseguiti i comandi del blocco, ritorna al punto 1

Visibilità delle Variabili

- I simboli {} definiscono una sequenza (blocco) di comandi. Sono tipicamente utilizzati in combinazione con altri comandi (if e while), ma possono anche apparire da soli.
- le variabili dichiarate all'interno di un blocco sono dette locali
- le variabili locali sono visibili (utilizzabili) solamente all'interno del blocco nel quale sono definite, con la seguente eccezione:

```
{
        int x=2; //chiamiamo x1 questa istanza di x
        {
                int x=3; // da questo momento x1 non è più visibile
        } // x1 è visibile nuovamente
}
```

Blocco di Istruzioni e Visibilità delle Variabili


```
{ // blocco 1
   int x; //x1
}

{ //blocco 2
   int x; //x2
   int y;
}
```

Posso definire la stessa variabile x in due blocchi diversi ed è come aver definito due variabili diverse (notate che dentro il blocco 2 non posso accedere a x1 e dentro il blocco 1 non posso accedere a x2)

Esercizio


```
/*
* Scrivere un programma che stampi x volte "Ciao Mondo!"
* Es. se x = 3
* Ciao Mondo!
* Ciao Mondo!
* Ciao Mondo!
*/
```

FOR


```
// inizializzazione: es. i = 0
while (condizione: es. i<10) {
        //sequenza di comandi
        //assegnamento: es. i = i +1;
}
```

si può scrivere come:

```
for(inizializzazione;  condizione; assegnamento) {
        //sequenza di comandi;
}
```

```
for (int x=1; x <= 3; x=x+1) {
        printf(“Ciao Mondo!\n”);
}
```

Esercizio


```
/*
* Calcolare la somma dei primi n numeri naturali e stamparla a video
* Ad es. se n=4 stampa
* 10
*/
```