Struttura del programma

- Direttive: `# include <stdio.h>`
  Necessaria per fare I/O
- Intestazione: `int main ()`
- Lista di istruzioni {...}
- Esempio:

```
# include <stdio.h>
int main () {
printf("Ciao come stai?");}
```

Stampa di dati in C
(Output)

- Per stampare si usa la `printf` (pensatela per ora come un comando)

```
printf("Ciao come stai?") ;
```

1. Il primo argomento di `printf` è sempre una stringa, che viene stampata, eccetto alcuni caratteri che sono speciali:
2. `\` introduce una formattazione: nell'esempio `\n`

```
printf("Ciao come stai?\n") ;
```

Stampa e va a capo

Stampa in C

% introduce una conversione: nell'esempio

d visualizza un intero decimale.

Es. il programma:

```
int main (){ int i = 0;
  printf("Valore della variabile i
  %d \n", i) ;}
```

stampa:

```
Valore della variabile i 0
```

e va a capo

Stampa in C

Es. il programma:

```
int main (){ int i = 0;
  printf("Primo programma C\n");
  printf("Valore della variabile i
  %d", i) ;}
```

stampa:

```
Primo programma C
Valore della variabile i 0
```

Composizione di istruzioni
Sequenzializzazione

Le istruzioni vengono eseguite in sequenza:

S_1

S_2

...

S_k

Lo stato in cui viene eseguita S_2 è lo stato risultante dopo l'esecuzioni di S_1 nello stato di partenza, ecc. lo stato in cui viene eseguita S_i è lo stato risultante dopo l'esecuzioni di S_{i-1}.

Sintassi della sequenzializzazione:

StatementList::=Statement StatementList | ε

Lettura dati in C
(Input)

- Per stampare si usa la `scanf` (pensatela per ora come un comando)

```
scanf("%d",&i) ;
```

1. Il primo argomento di `scanf` è sempre una stringa, che fornisce numero e formato degli argomenti.
2. I successivi argomenti sono nomi di variabili precedute dal simbolo `&`.
3. Le variabili devono esistere, cioè essere state dichiarate.

Un esempio di lettura

```
# include <stdio.h>
int main (){
int i; int j;
printf("Dammi due interi\n");
scanf("%d%d",&i,&j);
printf(“Il primo valore e` %d
\n”, i);
printf(“Il secondo valore e`%d
\n”, i);}
```

Condizionale
(identico a Java)

Le istruzioni eseguite dipendono dal verificarsi o meno di una condizione. Due forme:

Sintassi

1. **`if`** `(Cond)` C_1 **`else`** C_2
2. **`if`** `(Cond)` C_1

Semantica informale

Nello stato di partenza viene verificata la condizione (`Cond`) che è un'espressione boolena. Se la condizione è verificata viene eseguito C_1 altrimenti nella forma 1 si esegue C_2, nella forma 2 si prosegue con l'istruzione successiva. C_1 ed C_2 sono comandi

Massimo di 2

```
# include <stdio.h>
int main (){
int i;
int j;
printf("Dammi due interi\n");
scanf("%d%d",&i,&j);
if (i>j) printf("Il max e`%d”, i);
else printf("Il max e`%d”, j);
}
```

Assegnamento
(identico a Java)

Modifica di variabile:

Sintassi

```
Ide = Exp
```

Semantica informale

Nello stato di partenza viene valutata l'espressione `Exp`, ottenendo il valore v espressione che deve essere dello stesso tipo (o compatibile) con il tipo di `Ide`. `Ide` deve essere dichiarata nello stato di partenza. Lo stato risultante è lo stato di partenza in cui `Ide` ha come valore v

Massimo di 3

```
# include <stdio.h>
int main (){
int i; int j; int h
printf("Dammi tre interi\n");
scanf("%d%d%d",&i,&j,&h);
int max=i;
if (j>max) max=j;
if (h>max) max=h;
printf("Il max e` %d”, max); }
```

Iteratori
(identici a Java)

Le istruzioni del corpo dell'iteratore vengono eseguite ripetutamente. La ripetizione è avviene se è verificata una condizione. Esistono tre forme di iterazione in C:

1. **`while`**
2. **`do - while`**
3. **`for`**

Gli iteratori in C sono tutti equivalenti (si può sempre sostituire un iteratore con un altro di diversa forma. Forme diverse si adattano diversamente alle varie situazioni. Il risultato è che se si sceglie la giusta forma il programma è più leggibile.

while

Il corpo `S` dell'iteratore viene eseguito finchè la condizione (`cond`) è vera.

- **Sintassi**:

```
while (cond) S
```

- **Semantica informale**: Si valuta la condizione (`cond`) nello stato di partenza, se è vera si esegue il corpo `S`, nello stato risultante dall'esecuzione di `S` si rivaluta l'intero `while`.

Un esempio iterazione

```
int main (){
int i; int k=1; int max;
printf("Dammi un intero\n");
scanf("%d",&max);
while (k<10) {
   printf("Dammi un intero\n");
   scanf("%d",&i);
   if (i>max) max=i;
   k=k+1; }
printf("Il max e` %d”, max); }
```

Costanti

```
#define H 0
```

H è una costante cioè un nome a cui è associato un valore (analogamente alle variabili), a cui però può essere assegnato un valore una sola volta.

Vantaggi:

- Aumentano la leggibilità
- Se cambia il valore della costante devo modificarlo solo nell'inizializzazione

Commenti

Aumentano la leggibilita` se usati con criterio.

- commenti su piu` righe /* */:

```
/* Commenti racchiusi tra parentesi */
```

Comandi Linux per Java

- Per compilare : `cc nomeClasse.c`
- Per eseguire abbiamo visto:

  `a.out`

  dove `nomeClasse.c` è il nome del file contenente il (sorgente del) programma C, editato usando emacs o un altro editor di testi.

Per eseguire un programma:

- Occorre:
  - scrivere il programma in un file (usando ad esempio emacs) dandogli un nome qualunque con aggiunto il suffisso "`.c`"
  - compilare il programma con il comando
    ```
    cc nomefile.c
    ```
  - il compilatore genera un programma in linguaggio macchina chiamato `a.out`, per eseguire tale programma è sufficiente digitare `a.out`.