UNIVERSITÀ DI PARMA

Controllo di Flusso

Begin at the beginning and go on till you come to the end: then stop.

Lewis Carroll, Alice's Adventures in Wonderland.

Sommario

UNIVERSITÀ DI PARMA

- Teorema Jacopini Böhm
- Sequenza
- Selezione
  - if()
  - switch()-case
- Iterazioni
  - while()
  - do-while()
  - for(;;)

Controllo di Flusso

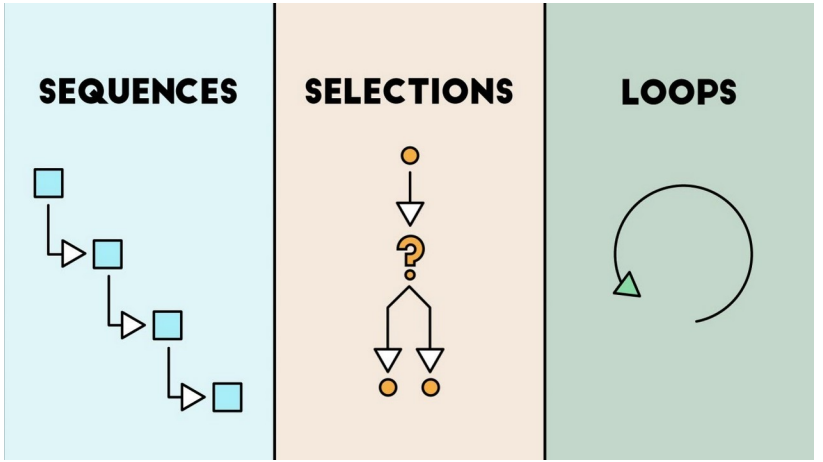
UNIVERSITÀ DI PARMA

- Indica in che ordine vengono eseguite le direttive del C
  - Rememeber: Algoritmo = Passi + Ordine Esecuzione
- Fino ad ora ci siamo limitati alla “sequenza”
  - Non basta!
- Per ciascuna riga di codice occorre capire:
  - Ordine di esecuzione
  - Se eseguire o meno
  - Quante volte eventualmente ripetere

Teorema di Jacopini Böhm

UNIVERSITÀ DI PARMA

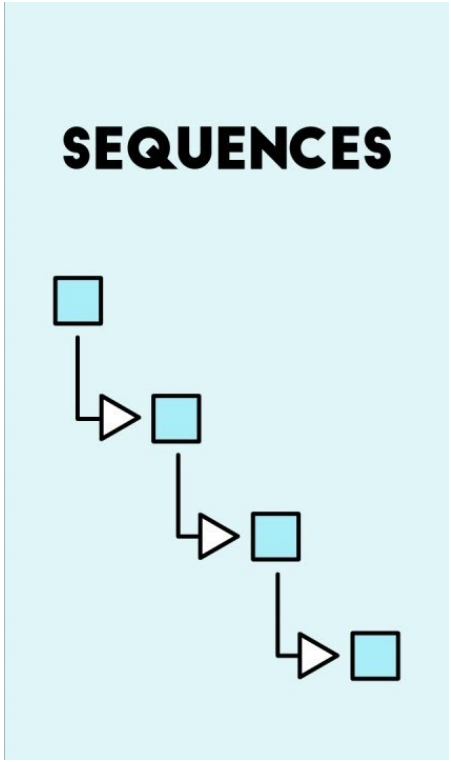
- Per costruire un programma sono necessari solo tre tipologie di esecuzione
  - Sequenza
  - Selezione (valutazione condizioni)
  - Iterazione (ciclo)
- Il C fornisce soluzioni multiple a queste strutture di controllo



Sequenza

UNIVERSITÀ DI PARMA

- Quanto visto fino ad ora negli esempi
  - Le istruzioni sono eseguite in sequenza
- Prestare attenzione all'ordine



Blocco di Istruzioni

UNIVERSITÀ DI PARMA

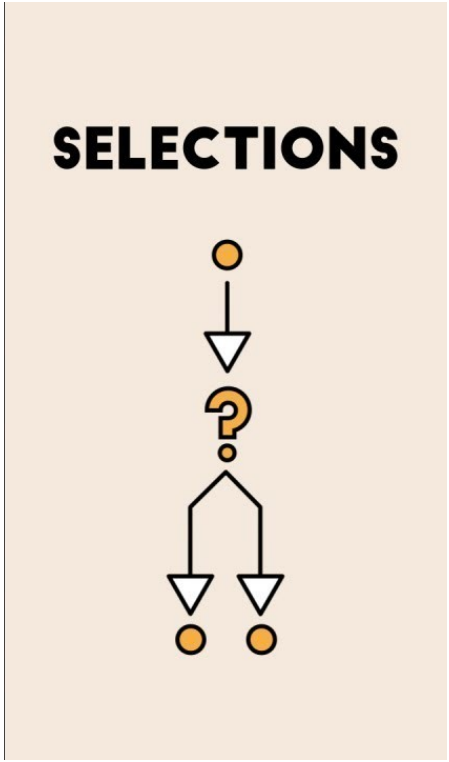
- *Compound Statement*
- In pratica una sequenza di istruzioni poste tra “{“ e “}”
- Esempio → corpo della main()

```
{
    <dichiarazioni variabili>
    <istruzione1>
    …
    <istruzionen>
}
```

Valutazione Condizioni

UNIVERSITÀ DI PARMA

- O “selezione”
- Valuto una o piú condizioni
- In base all’esito della valutazione decido cosa fare



Valutazione condizioni if()

UNIVERSITÀ DI PARMA

```
if(espressione)
        istruzione|blocco di istruzioni
```

- Se l'espressione è valutata come "vera" (≠∅)
  - L'istruzione o il blocco di istruzioni sottostanti vengono eseguiti
- Se l'istruzione è valutata come "falsa" (=∅)
  - L'istruzione o il blocco di istruzioni sottostanti vengono saltate

Valutazione condizioni if()-else

UNIVERSITÀ DI PARMA

```
if(espressione)
        istruzione|blocco di istruzioni
else
        istruzione|blocco di istruzioni
```

- Indico un'alternativa se condizione non si verifica

Valutazione condizioni switch()-case

UNIVERSITÀ DI PARMA

```
switch (espressione){
   case <val1>: istruzione|blocco
   case <val2>: istruzione|blocco
…
   default:     istruzione|blocco
}
```

- Spesso occorre confrontare un'espressione con differenti valori
- Praticamente un if(==) multiplo

Valutazione condizioni switch()-case

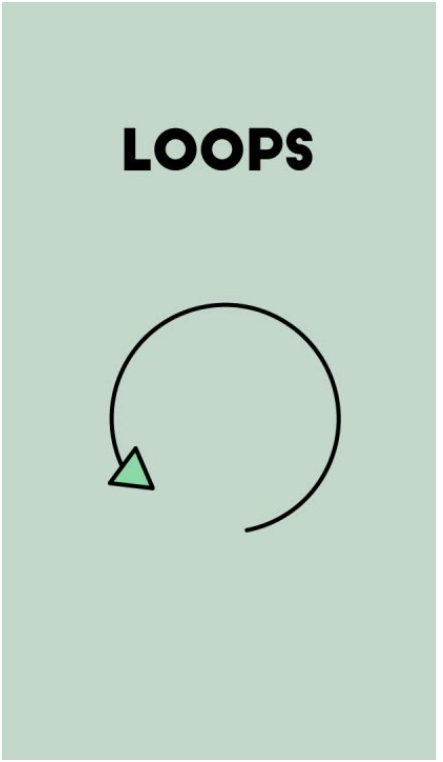
UNIVERSITÀ DI PARMA

- Nello switch()-case
  - L'espressione deve essere "intera"
  - No virgola mobile, no stringhe
- I vari "case" devono corrispondere a costante
  - Tutte differenti
  - L'ordine è ininfluente, "default" non deve obbligatoriamente essere l'ultimo
  - Non ci vogliono {} salvo che non definisca variabili
- In assenza del "break" l'esecuzione passa al "case" successivo
- Il "default" non è obbligatorio

Iterazioni

UNIVERSITÀ DI PARMA

- Ripeto istruzione o blocco di istruzioni
- In C tre tipologie
  - while()
  - do-while()
  - for(;;)



Iterazioni, ciclo while()


```
while(espressione)
        istruzione|blocco di istruzioni
```

- Fino a che l'espressione è vera l'istruzione/blocco viene eseguito
  - Se l'espressione è inizialmente falsa, nessuna esecuzione
- Problemi:
  - Rischio ciclo infinito

Iterazioni, ciclo do-while()

UNIVERSITÀ DI PARMA

```
do
        istruzione|blocco di istruzioni
while(espressione);
```

- Fino a che l'espressione è vera l'istruzione/blocco viene eseguito
  - Se l'espressione è inizialmente falsa, almeno una esecuzione
- Problemi:
  - Anche qui rischio ciclo infinito

Iterazioni, ciclo for(;;)

UNIVERSITÀ DI PARMA

```
for(<inizializzazione>; <condizione>; <aggiornamento>)
    istruzione|blocco di istruzioni
```

- Inizializzazione
  - espressione valutata inizialmente, tipicamente iniz. variabili
- Condizione
  - espressione valutata per stabilire se eseguire il corpo del ciclo
- Aggiornamento
  - espressione valutata dopo aver eseguito il corpo del ciclo

Iterazioni, ciclo for(;;)

UNIVERSITÀ DI PARMA

- Di fatto molto sovrapponibile al ciclo while()
  - Ma piú comodo
- Nessuna delle 3 espressioni è obbligatoria
- Ciclo infinito se tutte assenti
  - for(;;) equivalente a un while(1)
- In C99 è permessa la definizione variabili in for()

while() vs for(;;)

UNIVERSITÀ DI PARMA

- Perfettamente intercambiabili
- Tipicamente for() quando il numero di iterazioni è noto

```
<expr1>
while(<expr2>)
{
  statement;
  <expr3>;
}
```

⟺

```
for(<expr1>;<expr2>;<expr3>)
{
  statement;
}
```

break e continue

UNIVERSITÀ DI PARMA

- Break:
  - Permette uscita da corpo ciclo o da case
  - Cicli nidificati ⟶ esco dal solo corpo del ciclo in cui viene usata
- Continue:
  - Usata con cicli permette la ripetizione immediata del ciclo
  - Nei for(;;) ⟶ salto ad esecuzione aggiornamento
  - Nei while()/do-while() ⟶ salto a valutazione condizione

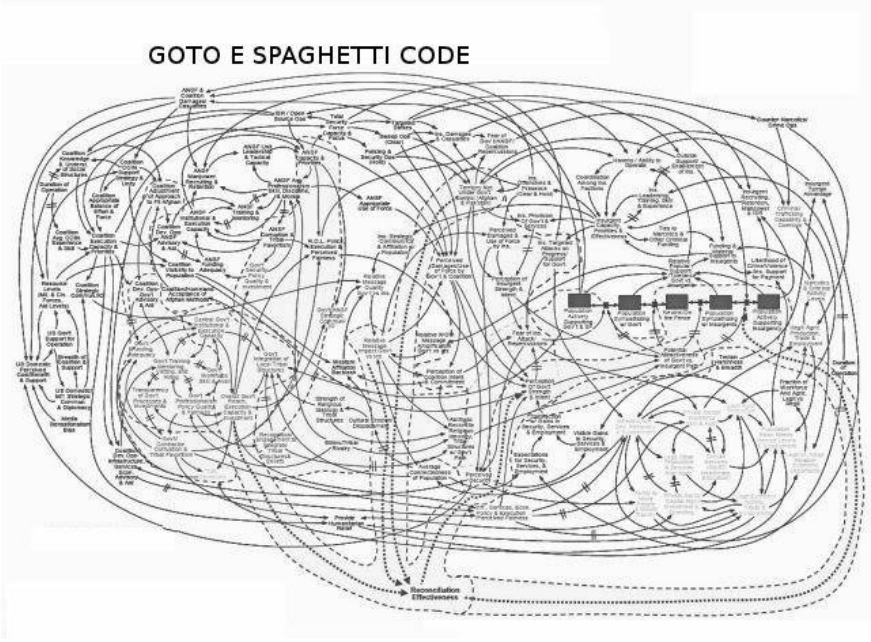
Cicli “infiniti”

- Non voluti ⟶ il male
- Ma ci sono condizioni in cui servono
  - Semplificare il codice
- Con break esco
- Possibilità:
  - for(;;) // preferibile
  - while(1)

Salti incondizionati

UNIVERSITÀ DI PARMA

- goto + label
- Rendono i programmi complessi e difficilmente manutenibili o modificabili
- *Spaghetti Code*
- In parte vero anche per break e continue



UNIVERSITÀ DI PARMA

Controllo di Flusso

Begin at the beginning and go on till you come to the end: then stop.

Lewis Carroll, Alice's Adventures in Wonderland.