Programmazione I

Lezione 13

Mini quiz!

https://forms.gle/4eLKXizPES3q5kY77

Approfondimento

- Inferenza di tipo
- Dichiarazioni “inline”
- Variabili locali nei costrutti if e for

Inferenza di tipo

- Domanda:

```
...
x = 10.7

...
```

  sotto quali condizioni un codice con questa istruzione passa la fase di compilazione?

- Altro esempio:

```
func inc(x uint32) uint32 {
    ...
}

...
z = inc(y)
```

Inferenza di tipo

- Idea:

  il compilatore può *inferire* il tipo che una variabile *dovrebbe* avere per *non* generare *errore di tipo* in fase di compilazione

- Sintassi GO per attivare inferenza di tipo
  - dichiarazione senza tipo con inizializzazione
  - shortcut → operatore :=

- Discussione su vantaggi e svantaggi

Recap: il sistema dei tipi

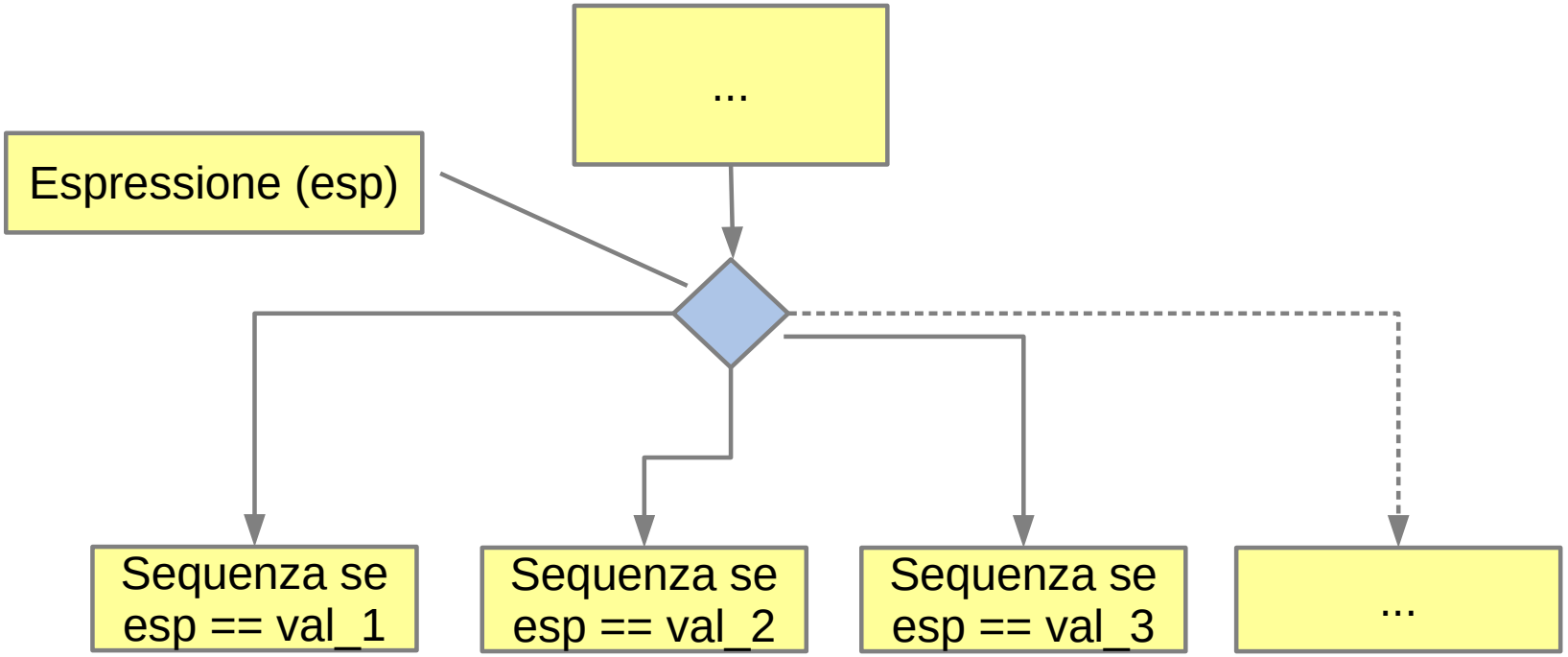
- Tipi “base” in GO
  - `bool`
  - `string`
  - `int  int8  int16  int32  int64`
  - `uint uint8 uint16 uint32 uint64 uintptr`
  - `byte // alias for uint8`
  - `rune // alias for int32, represents a Unicode code point`
  - `float32 float64`
  - `complex64 complex128`
- Codice “complex.go”

Esempi in aula:

- Programma che determina il numero e valori delle radici (reali) distinte di un'equazione di secondo grado (in forma normale)
- *Funzione che calcola se un numero è primo, e se non lo è restituisce un certificato di "non primalità"*
- Funzione che calcola MCD
- Funzione che calcola mcm
- Numero di cifre, e loro somma, di un numero intero
- Funzione che verifica se un numero è primo +
  Programma che stampa tutti i numeri primi da 1 a n

  [T] Somma dei primi n quadrati, potenze di 2 fino a n

Selezione multi-aria



- Costrutto GO **switch - case**

- Sintassi

  switch <espressione testa> {

  case <espressione 1a>, <espressione 1b>:

  <seq. 1, attivata se
  val. espressione testa == val. espressione 1a, oppure

  val. espressione testa == val. espressione 1b ... >

  case <espressione 2>:

  <seq. 2, attivata se val. espressione testa == val. espressione 2>

  ...

  default:

  <sequenza generica, attivata se nessuna delle precedenti è attivata>

  }

- NB eseguito un qualsiasi “case” si esce dal costrutto
- NB la sezione **default** è opzionale

Esempio

```
switch <variabile> {
case <costante 1>:
    <sequenza 1, attivata se variabile == costante 1>
case <costante 2>:
    <sequenza 2, attivata se variabile == costante 2>
...
default:
    <sequenza generica, attivata se variabile diversa da tutte le costanti>
}
```

Esempio

switch {

case <espressione booleana 1>:

<sequenza 1, attivata se espressione booleana 1 è true>

case <espressione booleana 2>:

<sequenza 1, attivata se espressione booleana 2 è true>

...

default:

<sequenza generica, attivata se nessuna delle espressioni booleane è true >

}

NB se non specificata, l’espressione testa è **true**

Esempio numerico

```
var val int = 8
switch val {
case pow(2,2):
    <sequenza 1, attivata se val == 2² (false)>
case pow(2,3):
    <sequenza 2, attivata se val == 2³ (true)>
}
```

- Che effetto ha la porzione di codice qui sopra?

Esempio esoterico

```
switch false {
case 2 == 3:
    // sequenza 1
case 2 == 2:
    // sequenza 2
}
```

- Che effetto ha la porzione di codice qui sopra?

Esempio esoterico

```
switch false {
case 2 == 3:
    // sequenza che viene attivata
case 2 == 2:
    // sequenza non attivata
}
```

Esempio

- Programma che iterativamente
  - Chiede un valore all’utente: un voto nell’esame di programmazione
  - Stampa “errore” se il valore è negativo o superiore a 30
  - Stampa “ripetere” se il valore è tra 0 e 17
  - Stampa “passato” se il valore è tra 18 e 26
  - Stampa “orale facoltativo” se il valore è tra 27 e 30
- Fino a che l’utente non inserisce il valore speciale 0