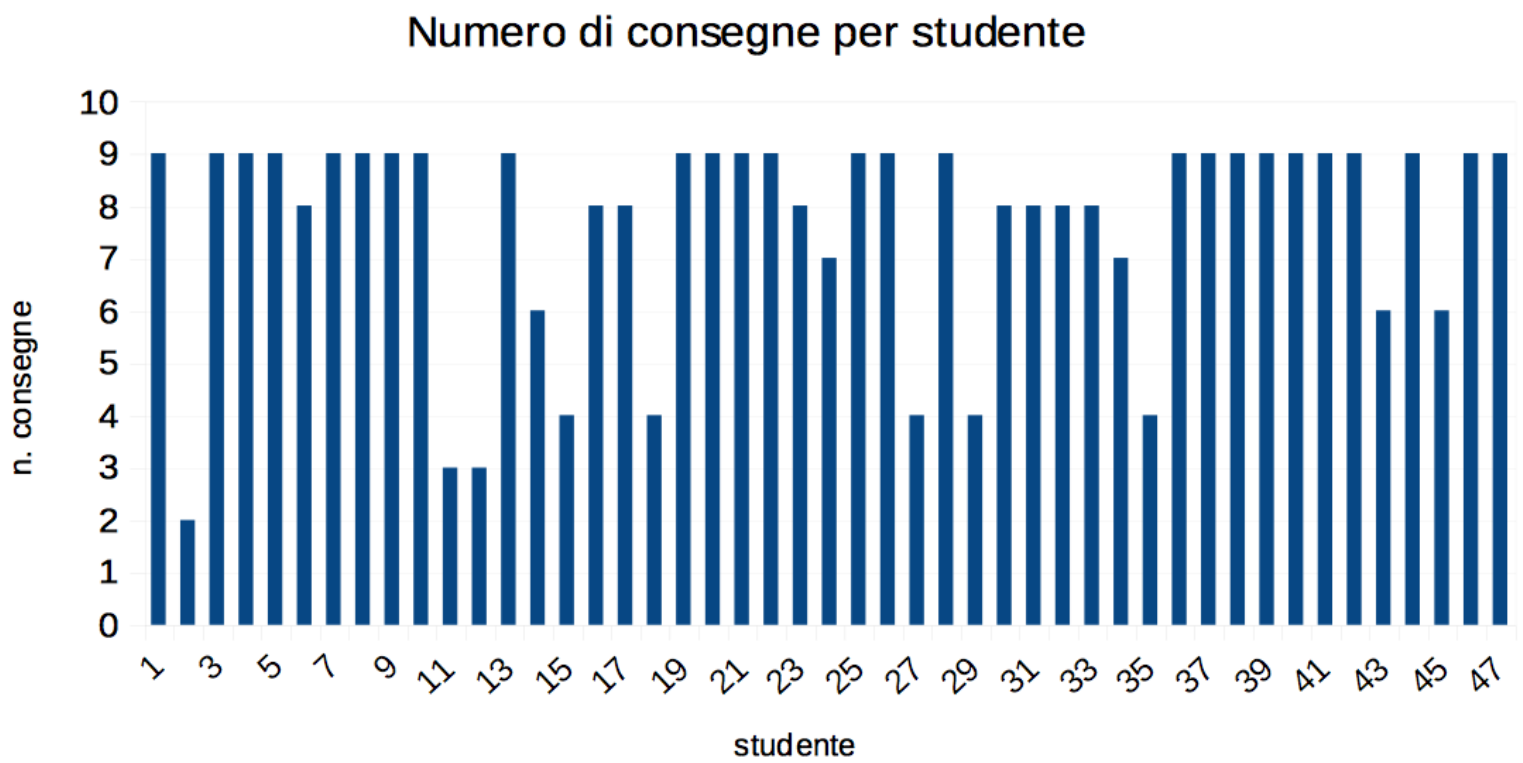
Laboratorio di programmazione
Corsi di laurea triennale unificati
Turno A (A-Cao)

Docente: Anna Morpurgo
Tutor: Umberto Costantini

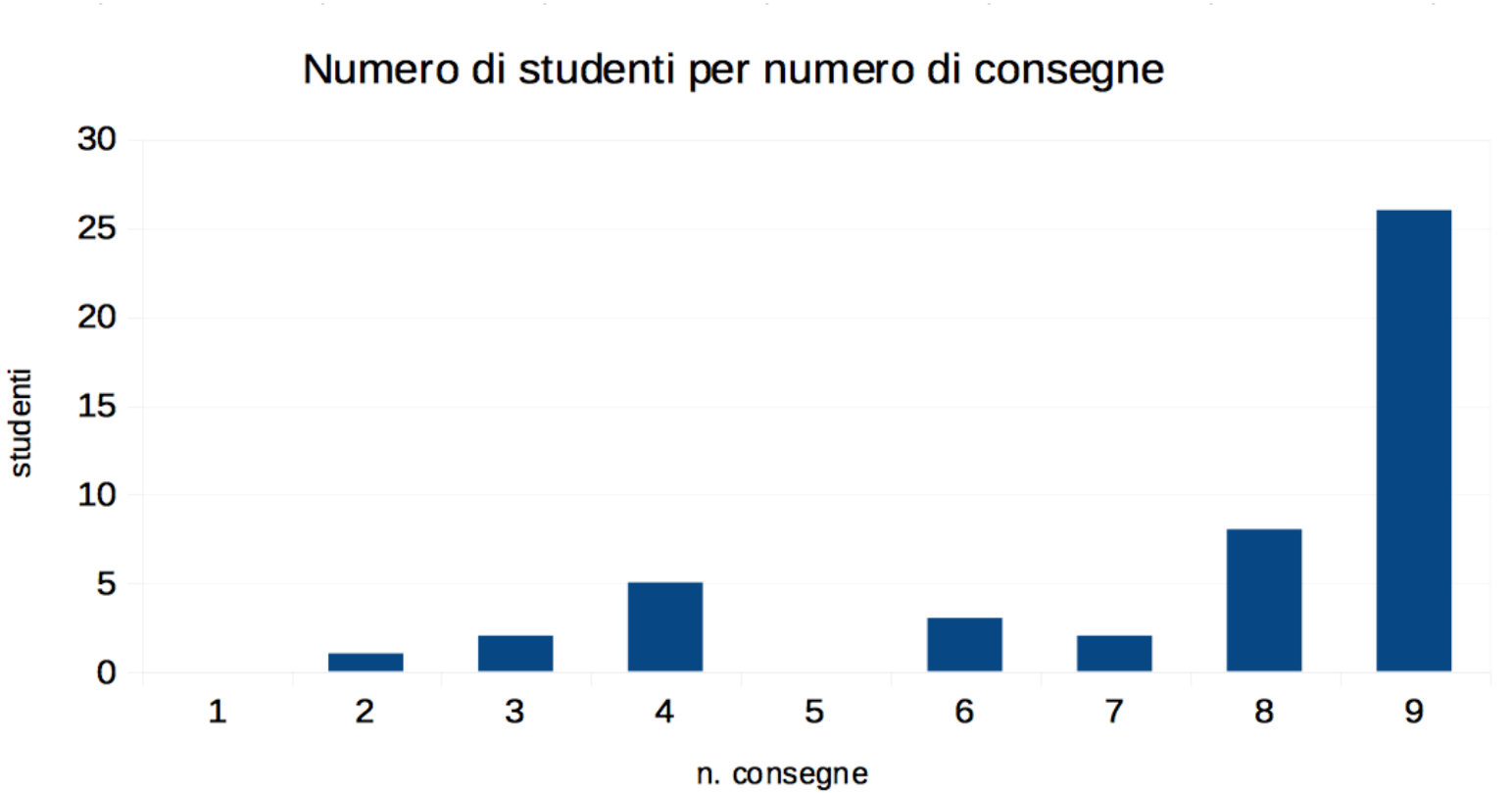
Dipartimento di Informatica
Università degli Studi di Milano

A.A. 2019-2020

Andamento


Numero di consegne per studente
n. consegne
0
1
2
3
4
5
6
7
8
9
10
1
3
5
7
9
11
13
15
17
19
21
23
25
27
29
31
33
35
37
39
41
43
45
47
studente

Andamento


Numero di studenti per numero di consegne
studenti
30
25
20
15
10
5
0
1
2
3
4
5
6
7
8
9
n. consegne

Uso di costanti, di nomi di variabili, di %

```
giorni = secondi / 86400
secondi = secondi % 86400
ore = secondi / 3600
secondi = secondi % 3600
minuti = secondi / 60
secondi = secondi % 60
```

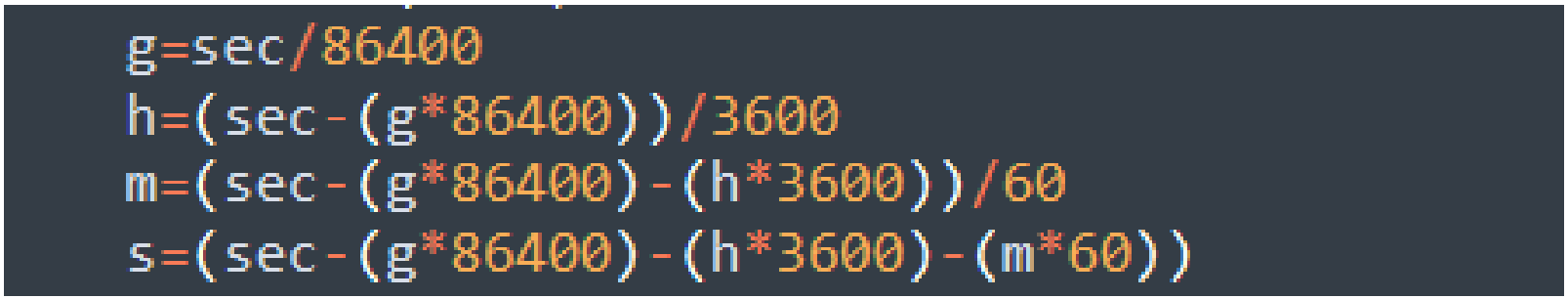
Uso di costanti, di nomi di variabili, di %


```go
func main(){
    var secondi int
    const GIORNO, ORA, MINUTO int = 86400, 3600, 60
```

Uso di costanti, di nomi di variabili, di %

```
fmt.Println("c=",c)
fmt.Println("d=",d)
fmt.Println("e=",e)
fmt.Println("f=",f)
```

Uso di costanti, di nomi di variabili, di %



```
g=sec/86400
h=(sec-(g*86400))/3600
m=(sec-(g*86400)-(h*3600))/60
s=(sec-(g*86400)-(h*3600)-(m*60))
```

Programma svolto a lezione - settimana III

14/10/19: 5. Variabili: nome, tipo, valore, scope. Tipi. Classificazione dei tipi (tipi di base, tipi composti, interfacce). Dichiarazione, assegnamenti e assegnamenti multipli, short-assignment.

16/10/19: 6. I/O di base: fmt.Println, fmt.Print, fmt.Scan. Tipi di base numerici (int, float64). Espressioni numeriche. Conversioni. Variabili inutilizzate e blank variable.

18/10/19: 7. Selezione binaria (if).
Il tipo bool e gli operatori booleani.
Esercizi.

Il costrutto if

Sintassi dell'`if`:

```
if <condizione>{
  blocco di istruzioni
} else if <altra condizione>{
  altro blocco di istruzioni
} else if . . .
} else {
  un altro blocco ancora
}
```

Espressioni booleane

Valori booleani: true, false

Variabili di tipo bool. Es.: `var isAdmitted = true`

Operatori di confronto: ==, !=, >, >=, <, <=

Operatori booleani: AND (&&), OR (||), NOT (!)

Esempi di condizioni:

esempio	in Go
variabili booleane	isAdmitted
espressioni di confronto	n > 100
espressioni logiche	
0 < n < 10 (n compreso tra 0 e 10)	`(0<n && n<10)`
n esterno all'intervallo 0-10	`(n<0 || n>10)`
ammesso e con reddito basso	isAdmitted && hasLowIncome

Tipi di errori

Errori di sintassi: vengono segnalati dal compilatore. Esempi:

- `undefined: n`
- `cannot refer to unexported name fmt.print`
- `undefined: fmt.print`
- `undefined: Print`
- `n % 2 = 0 used as value`

Errori *run time*: vengono segnalati quando si lancia l'esecuzione di un programma
Esempio: `panic: runtime error: integer divide by zero`

Errori logici (di progetto/implementazione): emergono solo con testing accurato
Ad esempio Go non fa check su lettura effettiva, non avvisa se una variabile vale zero perché non è stata inizializzata, ecc.

I/O

Read - Process - Write Pattern

Qualsiasi segmento di programma può essere specificato descrivendo:

- l'input richiesto
- il processo che lo trasforma
- l'output o i risultati generati

Tre livelli nella gestione di dati forniti da utente:

- *raw input*: ci aspettiamo dati perfetti e non facciamo nessun controllo
- *verified input*: verifichiamo che l'input sia conforme come tipo/comportamento
- *filtered input*: controlliamo che siano rispettati anche vincoli legati al problema specifico trattato

Schemi per la selezione

Pattern per la selezione

Whether or Not

Alternative Actions

Range of Possibilities

Sequential Choice

Unrelated Actions

Independent Conditions

Pattern ausiliari (strategici, stilistici)

Short case first

Positive condition

Indent for structure

Schemi per la selezione

Whether or not (zero or one of one action): if senza else
Es.: se è buio, accendo la luce

```
if <condition>{
    <action>
}
```

Alternative actions (one of two actions): if - else
Es.: se è venerdì, pesce; altrimenti carne

```
if <condition>{
    <action one>
} else { // !condition
    <another action>
}
```

Schemi per la selezione

Range of possibilities (one of several actions, one expression): switch
Es.: il giorno è lun/mar/mer/gio/ven/sab/dom, sette azioni diverse
Range of possibilities (one of several actions, several mutually exclusive conditions): if - else if - ... - else
Es.: delta > 0; delta < 0; delta = 0, azioni diverse per ogni caso

```
if <condition1>{
    <action 1>
} else if <condition2>{
    <action 2>
}
...
} else{
    <action n>
}
```

Schemi per la selezione

Sequential choice (one of several actions, mutually exclusive "consecutive" conditions): if - else if - ... - else
Es.: partecipanti > 1000; partecipanti > 100; partecipanti > 10; partecipanti < 10

```
if <condition1>{
    <action 1>
} else if <sub_condition2>{
    <action 2>
}
...
} else{
    <action n>
}
```

Schemi per la selezione

Unrelated actions (several actions, each with its condition): in pratica si tratta di diversi “Whether or not”: if - if - ...
Es.: freddo -> accendo stufa; sabato -> invito amici; frigo vuoto -> spesa

```
if <condition 1>{
    <action 1>
}
if <condition 2>{
    <action 2>
}
...
if <condition n>{
<action n>
}
```

Schemi per la selezione

Independent conditions (for each action, several independent conditions): if annidati
Es.: rettangolo /cerchio; colorato/bianco;

```
if <condition 1>{
    if <condition 2>{
        <action 1>
    } else {
        <action 2>
} else { //! condition 1
    if <condition 3>
        <action 3>
    } else {
        <action 4>
    }
}
```

Esercizio 1 - Pari o dispari?

Problema: Scrivere un programma Go `pari_dispari.go` che, dato un numero intero, determini se è pari o dispari.

Esempi di esecuzione:

```
numero?
4
4 è pari

numero?
15
15 è dispari
```