Programmazione I

Lezione 16

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`

Enigma: 16 9 17 14 13 17 18 1 22 27 25

A	1	S	17
B	2	T	18
C	3	U	19
D	4	V	20
E	5	Z	21
F	6	spazio	22
G	7	0	23
H	8	1	24
I	9	2	25
L	10	3	26
M	11	4	27
N	12	5	28
O	13	6	29
P	14	7	30
Q	15	8	31
R	16	9	32

Tipi carattere

- Qual’è “la risposta”?
- Fatto: nella macchina posso rappresentare solo stringhe di bit
- Necessità: voglio trattare entità più complesse

  Esempio: caratteri!
- Idea: ricorro ad un’opportuna codifica

  (Come IEEE 754 per i float: da reale a stringa di bit)
- Esempio (ipotetico) →

  a → 1 → 00000001
  b → 2 → 00000010
  c → 3 → 00000011

Caratteri e codici

- American Standard Code for Information Interchange (ASCII)
  - vedi tabella (e.g. asciitable.com)
  - ogni carattere usa 1 byte
  - → 128 caratteri
- UNICODE → contiene
  - attualmente 137.994 “grafemi” (codepoints)
  - fino a 1.114.112 (17 planes, 2^{16} c.p per plane)
  - I primi 128 sono gli stessi dell’ASCII
  - Necessari fino a 4 byte (pensate a Maxint per 32 bit)
- Digressione: tabelle ASCII e UNICODE (e.g. codepoints.net)

Enigma: 16 9 17 14 13 17 18 1 22 27 25

A	1	S	17
B	2	T	18
C	3	U	19
D	4	V	20
E	5	Z	21
F	6	spazio	22
G	7	0	23
H	8	1	24
I	9	2	25
L	10	3	26
M	11	4	27
N	12	5	28
O	13	6	29
P	14	7	30
Q	15	8	31
R	16	9	32

Enigma: 16 9 17 14 13 17 18 1 22 27 25 =

r i s p o s t a 4 2

A	1	S	17
B	2	T	18
C	3	U	19
D	4	V	20
E	5	Z	21
F	6	spazio	22
G	7	0	23
H	8	1	24
I	9	2	25
L	10	3	26
M	11	4	27
N	12	5	28
O	13	6	29
P	14	7	30
Q	15	8	31
R	16	9	32

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`

Tipi carattere

- In go: rune (formato fisso, 4 byte)
- Esempio: stampa di rune in carattere, decimale e binario (play_with_bits_rune.go)
- GO “format verbs” https://golang.org/pkg/fmt/
  %c (%q)