COSTANTI INTERE

3

31

-127

5

0

1600

NOTAZIONE ORDINARIA

0x1c

0x12

0xAAA1

NOTAZIONE ESADECIMALE

$0x1C = 16^1 \times 1 + 16^0 \times 12 =$

$= 16 + 12 = 28$

016

~~08~~

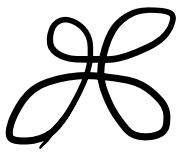
NOTAZIONE OTTALE

var x, y int

x = 12 + 0xAA * 14

y = 0x1C + (y * 4)

RUNE



Point of Interest
8984 = 2318_{16}

var x rune

x = 8984

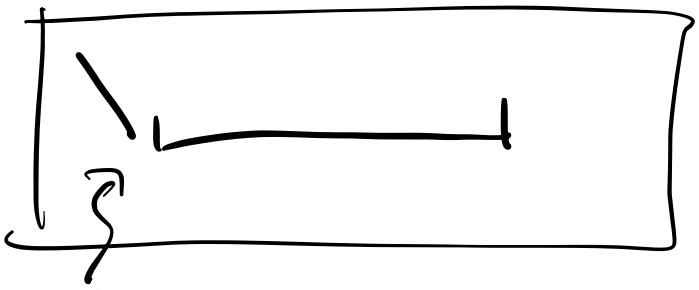
x = 0x2318

x = '⌘'

x = '\u 2318'

sequenza di escape

Sequenza di escape



backslash

\uXXXX

\UXXXXXXXX

\t TAB

\n NEWLINE

\' '

\\ \

var x rune

x = 'Y'

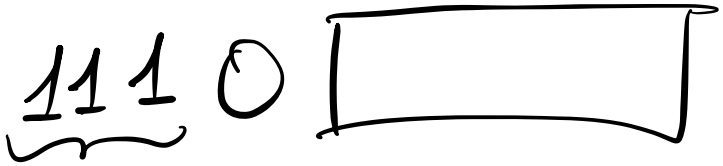
fmt.Println(string(x))

UTF-8

- Rappresentazione a lunghezza variabile

1) Ogni carattere occupa da 1 a 4 byte
2) Il primo byte

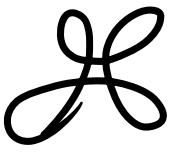
per i caratteri ASCII (monobyte)
oppure



k

per i caratteri di K > 1 byte

3) I byte successivi al primo iniziano con



8984 = 1000 11000 11000

UTF-8

1110 0010 | 10 001100 | 10 011000

STRINGHE

sequenza di byte

var s string

s = "Ciao"

interpretata come sequenza UTF-8

len(s) ← lunghezza in byte

s[i] ← i-esimo byte (0, 1, 2, ... l

```
s = "Pippo"
for i := 0; i < len(s); i++ {
    c := rune(s[i])
    fmt.Println(string(c))
}
```

FOR RANGE PER STRINGHE

Per scandire una stringa carattere per carattere

int: indice del byte — rune: carattere — stringa da scandire

```
for i, c := range s {

}
```

- STAMPARE UNA STRINGA SOSTITUENDO LE VOCALI CON u

```
var s string
s = "Garibaldi fu ferito"
for i:=0; i<len(s); i++ {
    c := rune(s[i])
    if c=='a' || c=='e' || c=='i' || c=='o' || c=='u' {
        c='u'
    }
    fmt.Print(string(c))
}
fmt.Println()
```

II

```
for _, c := range s {
    if c == 'a' || c == 'e' ||
       c == 'i' || c == 'o' || c == 'u' {
        c = 'u'
    }
    fmt.Print(string(c))
}
fmt.Println()
```

ESERCIZI

1) Stampare una stringa in alfabeto farfallino

piano → pifiafa nofo

2) Contare quante 'a' ci sono in una stringa

3) DIFFICILE

Stampare l'ultimo carattere di una stringa

4) Contare quanti caratteri non-ASCII ci sono in una stringa