RUNE

Predetto unicode

var x rune

x = 'A'

fmt.Println(x)

fmt.Println(string(x))

---

if unicode.IsLetter(x)...

if (x >= 'A' && x <= 'Z')
|| (x >= 'a' && x <= 'z')

'A' <= x <= 'Z'

STRINGS

IMMUTABILI

↓

var x string

x = "ciao che bella città!"

fmt.Println(x)

ACCESSO BYTE-WISE

n = len(x) ← n° di byte della stringa

x

x[0], x[1], ..., x[n-1]

rune(x[i])

for i := 0; i < n; i++ {

... x[i] ...

}

ACCESSO RUNE-WISE

indice → i, r := range x { ← rune

...

}

1 + ← CONCATENAZIONE TRA STRINGHE

```
var s, s1, s2 string
s1 = "pa"
s2 = "olo"
s = s1 + s2

s = s + s
```

2 s[i:j]

altrimenti
d2 byte i (compreso)
d1 byte j (escluso)

0 1 2 3 4 5 6 7 8 9 10
s := "ciao␣mamma"

z := s[3:6] "o␣m"

3 INPUT

var x string

fmt.Scan(&x)

legge fino al primo whitespace

ESERCIZIO

Scrivete una funzione che data una stringa conti quante a (minuscole o maiuscole) contiene e restituisce questo numero.

SOLO PER STRINGHE ASCII

```
func contaA(s string) int {
    var conta int
    for i:=0; i<len(s); i++ {
        c := rune(s[i])
        if c == 'a' || c == 'A' {
            conta++
        }
    }
    return conta
}
```

PER TUTTE LE STRINGHE

```
func conta A(s string) int {
    var conta int
    for _, c := range s {
        if c=='a' || c=='A' {
            conta++
        }
    }
    return conta
}
```

s[i]

```
func main() {
    x := "ciao mamma"
    n := contaA(x)
    fmt.Println(n)
}
```

SCRIVETUNA FUNZIONE CHE
- DATA UNA STRINGA ASCII
E UN INTERO i
DICA QUANTI CARATTERI
CI SONO PRIMA LA
i-ESIMA E LA (i+1)-ESIMA
BARRA '/'

```
ciao/come/stai/bene/ok
i = 2
```

(se ci sono meno di i+1
barre, restituire - 1)

- SCRIVERE UNA FUNZIONE CHE DATA UNA STRINGA DICE SE CI SONO RIPETUTI CARATTERI

STRINGHE ASCII carnasse

```
func ripetuti(s string) bool {
    for i := 0; i < len(s); i++ {
        for j := i+1; j < len(s); j++ {
            if s[i] == s[j] {
                return true
            }
        }
    }
```

```
        }
    }
    return false
}
```

STRINGHE GENERICHE

```
func ripetuti(s string) bool {
    for i1, c1 := range s {
        s[i1:]
        for i2, c2 := range s {
            if c1 == c2 && i1 != i2 {
                return true
            }
        }
    }
    return false
}
```

}

- DATA UNA STRINGA ASCII DETERMINARE QUANTE PAROLE CONTIENE. LE PAROLE SONO SEPARATE DA UNO O PIÙ SPAZI.

  "ciao come stai" → 3

```
func countWords(x string) int{
    var conta int
    x += "␣"
    for i:=0; i<len(x)-1; i++{
        if x[i] != '␣' &&
```

```
        X[i+1] == ' ' {
            conta++
        }
    }
    return conta
}
```

Soluzione di Francesco per le stringhe non ASCII

```
var conta int
var inWord bool
var x t= "
```

```
for _, c := range s {
    if inWord {
        if c == ' ' {
            conta++
            inWord = false
        }
    } else {
        if c != ' ' {
            inWord = true
        }
    }
}
return conta
```

- Data una stringa contenente * cancellare gli *

"ca**ne**nr e***ro"

↓

"canenero"

```
func cancAst(s string) string {
    var ris string
    for _, c := range s {
        if c != '*' {
            ris += string(c)
        }
    }
    return ris
}
```