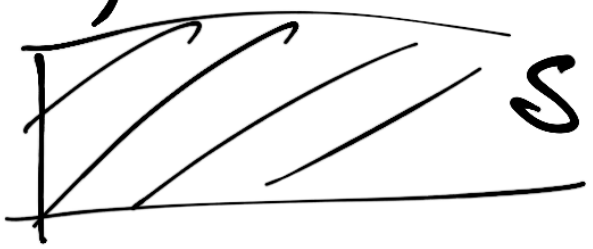
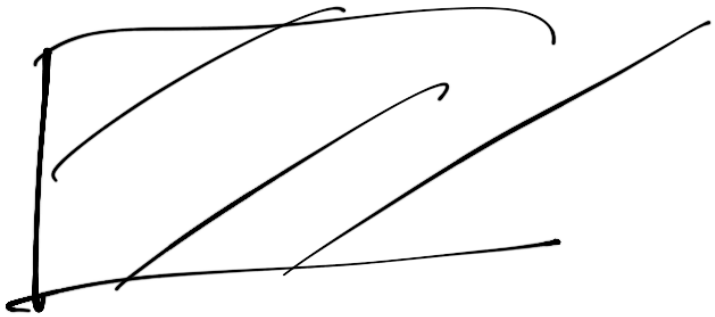
String

```
for i := 0; i < len(s); i++ {
    
    s[i]
}
```

```
for _, x := range(s) {
    
}
```

```
var s string
fmt.Scan(&s)
c := 0
```

```
for i := 0; i < len(s); i++ {
    if s[i] == 'a' {
        c++
    }
}
```

```
for _, car := range(s) {
    if car == 'a' {
        c++
    }
}
```

```
t := ""
for _, x := range(s) {
    t = string(x) + t
}
fmt.Println(t)
```

```
var i int
for i = 0; i < len(s); i++ {
    if s[i] == '/' {
        break
    }
}
if i < len(s) {
    fmt.Println(s[i+1:])
} else {
    fmt.Println("Non ci sono /")
}
```

```
for i, x := range S {
    if x == '/' {
        fmt.Println(s[i+1:])
        break
    }
}
```

CONTEGGIO DI PAROLE


```
c := 0
for i := 0; i < len(s) - 1; i++ {
    if s[i] != ' ' &&
       s[i+1] == ' ' {
        c++
    }
}
if s[len(s)-1] != ' ' {
    c++
}
```

```
prev := ' '
for _, x := range s {
    if !unicode.IsSpace(x) &&
        unicode.IsSpace(prev) {
        ctr++
    }
    prev = x
}
```

---

unicode.IsLetter(x rune) bool

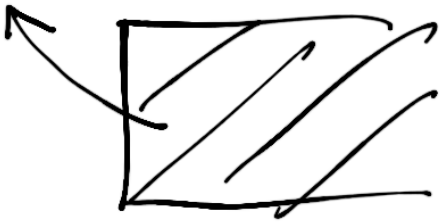
unicode.IsLower(x rune) bool

unicode.IsSpace(x rune) bool

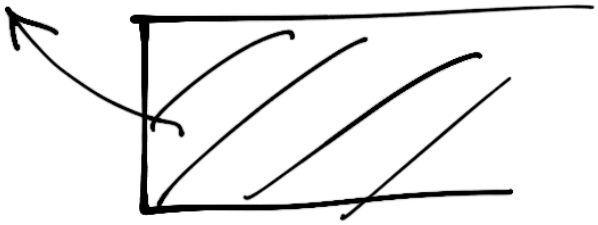
SELEZIONE MULTIARIA

switch (expr-s) {

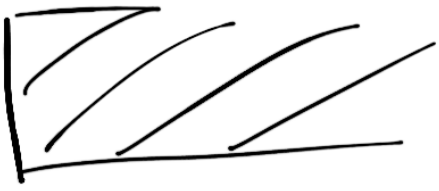
case expr, expr, ... :



case expr, expr, ... :



default :



}

```
var n int
fmt.Scan(&n)

switch n {
    case 0:
        fmt.Println("zero")
    case 1:
        fmt.Println("uno")
        :
    case 9:
        fmt.Println("nove")

}
```

1 ⟶ 99

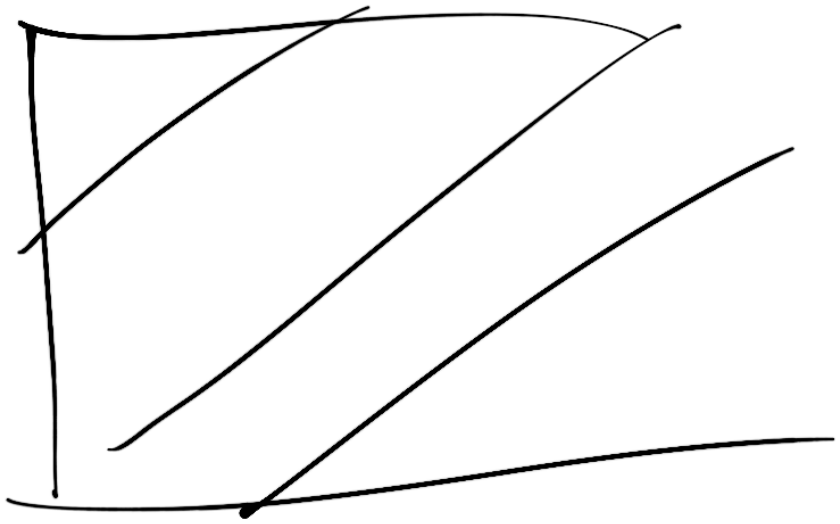
ESERCIZIO

Scrivete un programma che
letta una stringa la
ristampi sostituendo u a
tutte le locali.

in → garibaldi

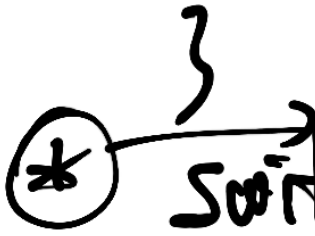
our → gurubuldu

```
fmt.Scan(&n)
if n >= 10 && n < 20 {
    switch n {
        case 10:
            fmt.Println("dieci")
        case 11:
            fmt.Println("undici")
        ...
} else {
```



```
}
```

```
voc:="a"
Switch  n/10 {
    case 2:
        fmt.Print ("vent ")
        voc = "i"
    case 3:
        fmt.Print ("trois ")
        ..
    case 9:
        fmt.Print ("neuf ")
}
```



```
Switch  n%10 {
    case 1:
        fmt.Println ("uno")
    case 2:
        fmt.Println("due")
        ...
    case 9:
        fmt.Println("nove")
}
```

```
if  n%10!=1 &&
    n%10!=8 {
        fmt.Print(string(voc))
}
```

\ barra rovesciata backslash

/ barra slash

12/10/2019

x = '\'	x = '\\'
y = '/'	y = '//'

/* Rappresenteremo le carte con numeri da 0 a 51.
Più precisamente:
- 0, ..., 12: cuori
- 13, ..., 25: quadri
- 26, ..., 38: fiori
- 39, ..., 51: picche.

Ovviamente 0, 13, 26, 39 sono gli assi, 1, 14, 27, 40 sono due ecc. */

```
func seme(carta int) string {
    switch carta/13:
        case 0:
            return "♥"
        case 1:
            return "♦"
        case 2:
            return "♠"
        case 3:
            return "♣"
    }
    return "???"
}
```

```
func valore(carta int) string {
    switch carta%13 {
        case 0:
            return "A"
        case 10:
            return "J"
        case 11:
            return "Q"
        case 12:
            return "K"
        default:
    }
    return fmt.Sprintf("%d",
        (carta%13)+1)
}
```

```
func stampaCarta(carta int) {
    fmt.Println(valore(carta), seme(carta))
}

func main() {
    for i:=0; i<52; i++ {
        stampaCarta(i)
    }
}
```