```
var s string
fmt.Scan(&s)
/* PROTOCOLLO://HOST/.... */
var i, pos1, pos2 int
for i=0; i< len(s); i++ {
    if s[i] == '/' {
        break
    }
}
pos1 = i
for i=pos1+2; i< len(s); i++ {
    if s[i] == '/' {
        break
    }
}
pos2 = i
host := s[pos1+2 : pos2]
prot := s[: pos1]
```

1

2

import "strings"

1 pos1 := strings.IndexRune(s, '/')

2 pos2 := strings.IndexRune(s[pos1+2:], '/') + pos1 + 2

---

s = "http://pippo/qualcosa"
0123456 789 10 11 12 13 14 15 ...

pos1 = 5

s[pos1+2:] = "pippo/qualcosa"
01234 5 6789 ...

unicode

Islower

```
var s string
fmt.Scan(&s)
c:=0
for i:=0; i<len(s); i++ {
    if s[i]>='a' &&
       s[i]<='z' {
        c++
    }
}
```

```
for _, r := range s {
    if r >= 'a' &&
       r <= 'z' {
        c++
    }
}
```

```
for _, r := range s {
    if unicode.IsLower(r)
        {
        c++
    }
}
```

- DATA UNA STRINGA ASCII CONTA DA QUANTE PAROLE È COMPOSTA

var s string

s = "Paolo Boldi è pazzo"

?

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

```
prev := rune('␣')
for _, r := range s {
    if unicode.IsSpace(r) &&
       !unicode.IsSpace(prev) {
            C++
    }
    prev = r
}
if !unicode.IsSpace(prev) {
        C++
}
```

SELEZIONE MULTIARIA

"true"

switch (espressione selettore) {

case expr, expr, ... :

=

case expr, expr, ... :

=

case expr, expr, ... :

=

default :

=

}

```
var n int
...
switch n * n {
case 1, 16, 27 :
        a = 3
case 9 :
        a = 4
default :
        a = 0
}
```

```
switch {
    case x>y, x==z:
        a=3;
    case y>5:
        a=7
    default:
        a=9
}
```

```
if x>y || x==z {
    a=3
} else if y>5 {
    a=7
} else {
    a=9
}
```

- DATO UN NUMERO TRA 1 E 99 DETERMINARE LA STRINGA CORRISP.

(32 → "trentadue")

[FUNZIONE]

```
func numeroInLettere (n int) string {
    if n >= 10 && n <= 19 {
        switch n {
            case 10:
                return "dieci"
            case 11:
                return "undici"
            :
            case 19:
                return "diciannove"
        }
    }
    var res, fin string
    switch n/10 {
        case 2:
            res = "vent"
            fin = "i"
        case 3:
            res = "trent"
            fin = "a"
```

```
...
        case 9:
            res = "novan"
            fin = "a"
    }
    switch n%10 {
        case 1:
            res += "uno"
        case 2:
            res += "due"

        ⋮
        case 9:
            res += "nove"
    }
    return res
}
```

```
if  n % 10 != 1 && n % 10 != 8 {
    res += fin
}
```

- STAMPARE UNA CARTA DA POKER

seme:
- cuori
- quadri
- fiori
- picche

0	asso
1	due
2	tre
⋮	
9	dieci
10	J
11	Q
12	K

0 → 12	carte di cuori
13 → 25	" " quadri
26 → 38	" " fiori
39 → 51	" " picche

A ♥ 2 ♦

J ♠

```
func stampaCartaPoker (carta int)
                        string {
    switch carta/13 {
        case 0:
            seme = "♥"
        case 1:
            seme = "♦"
        ...
    }
    switch carta % 13 {
        case 0:
            val = "A"
        case 10:
            val = "J"
        case 11:
            val = "Q"
        case 12:
            val = "K"
        default:
            val = fmt.Sprintf("%d",
                (carta % 13) + 1)
```

(carta.ID) + 1)

}

return val + " " + seme

}