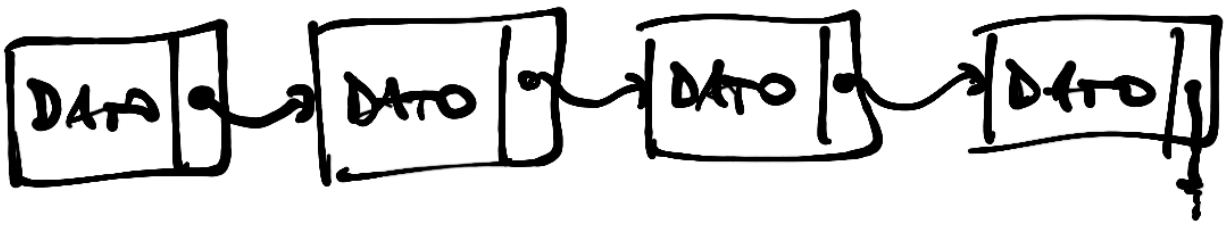
LISTE CONCATENATE SEMPLICI



LISTA DI STRINGHE

NIHS = Not Invented Here Syndrome

Package list

```
type Node struct {
        data string
        next *Node
}
```

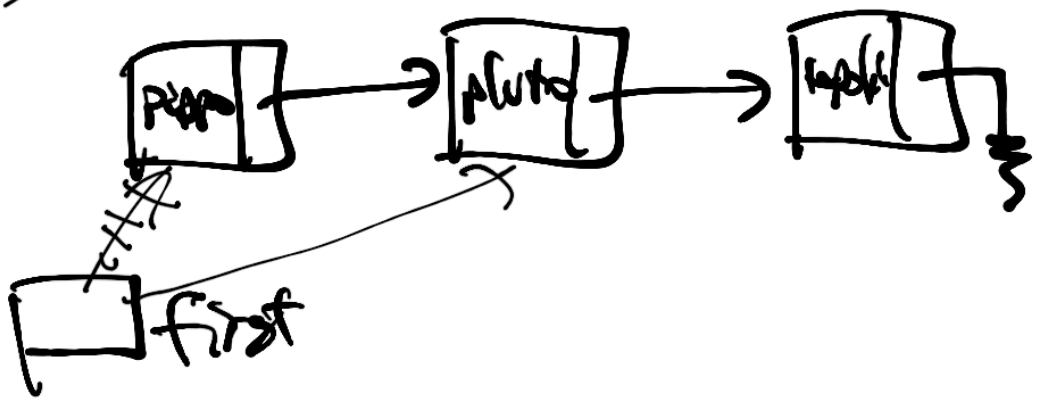
```
var p *Node

(*p).data
p.data
```

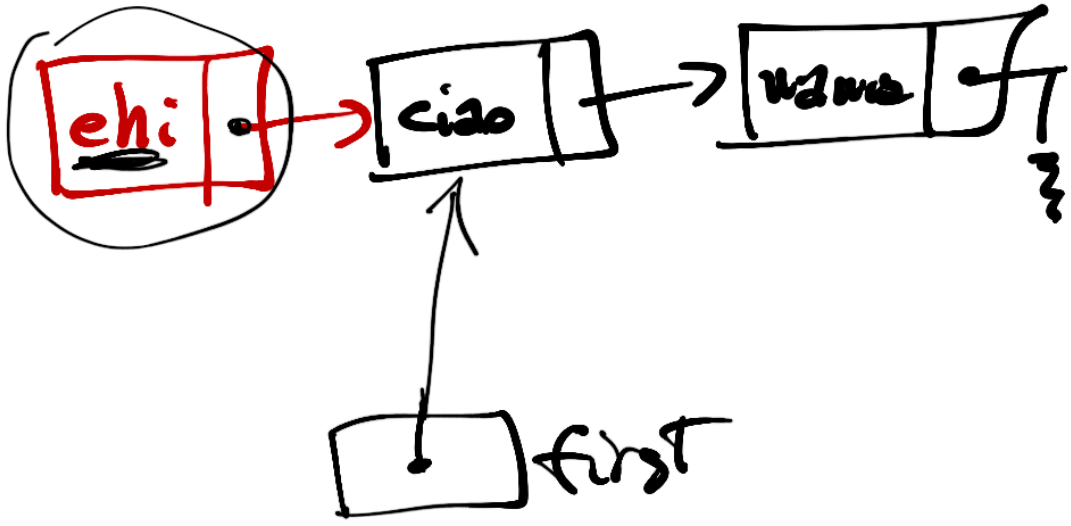
C
p -> data

```
package list

func Length(first *Node) int {
    c := 0
    for first != nil {
        c++
        first = first.next
    }
    return c
}
```



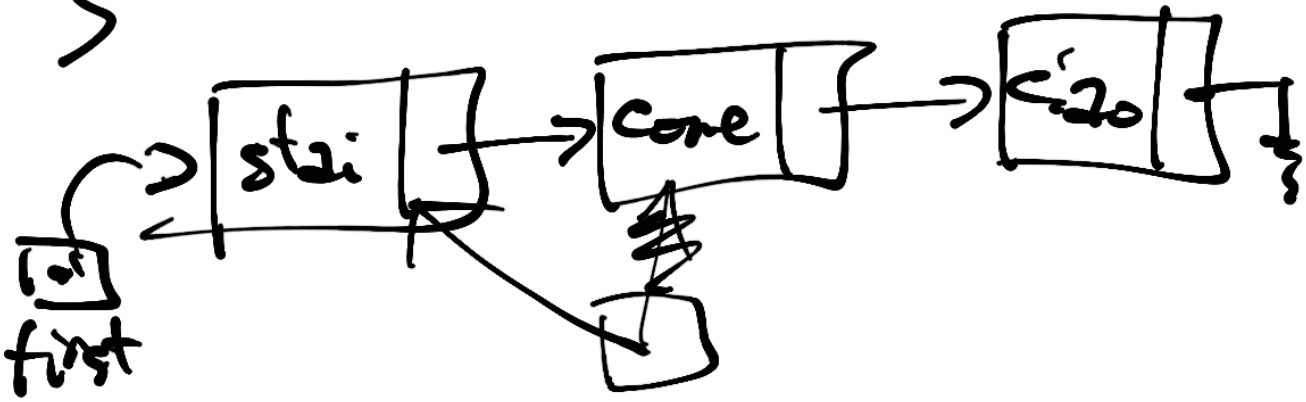
```
package list
func AddFront(first *Node, item string)
        *Node {
    n := new(Node)
    n.data = item
    n.next = first
    return n
}
```



```
package main

import (
    "list"
    "fmt"
)

func main() {
    var first *list.Node
    first = list.AddFront(first, "ciao")
    first = list.AddFront(first, "come")
    first = list.AddFront(first, "stai")
    fmt.Println(list.Length(first))
}
```

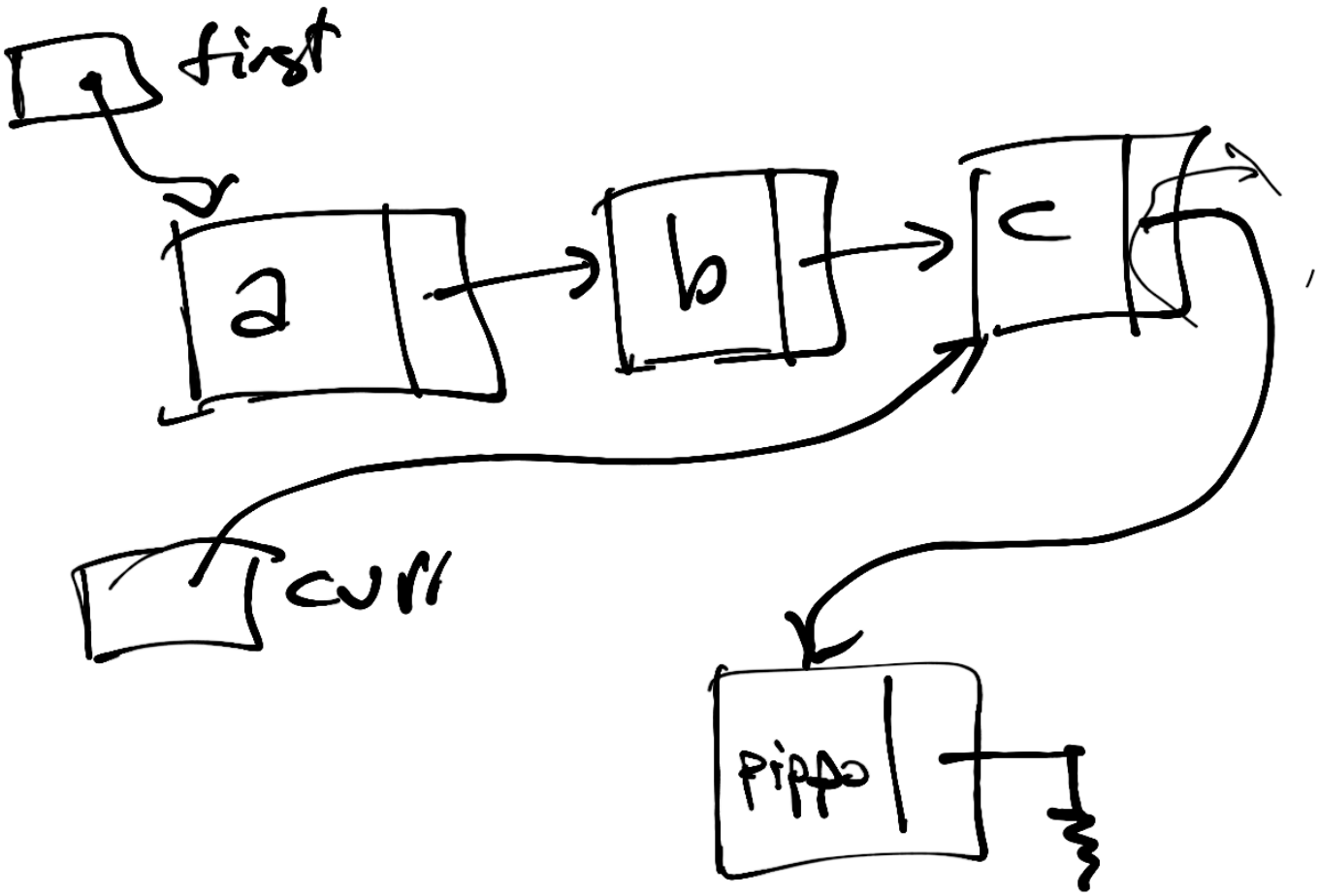


```
package list
import "fmt"

func PrintList(first *Node) {
    for first != nil {
        fmt.Println(first.data)
        first = first.next
    }
}
```

```
package list

func AddTail(first *Node,
    item string) *Node {
    n := new(Node)
    n.data = item
    n.next = nil
    if first == nil {
        return n
    }
    curr := first
    for curr.next != nil {
        curr = curr.next
    }
    curr.next = n
    return first
}
```



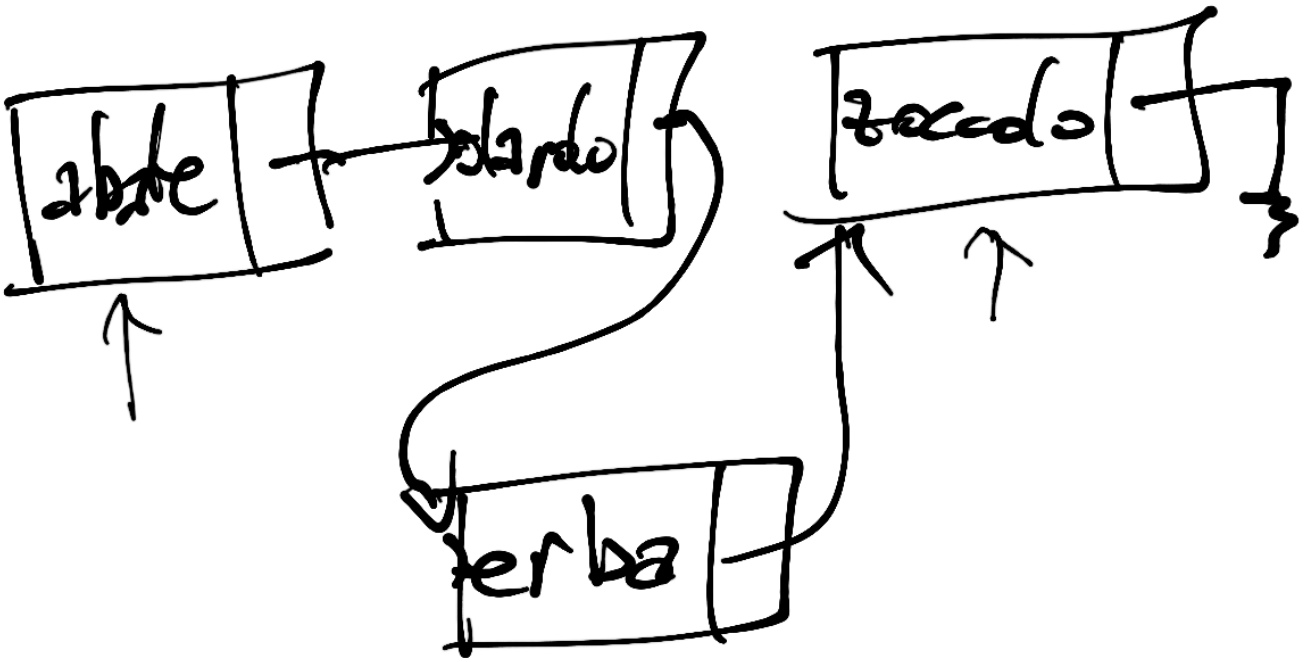
addTail(fist, "pippo")

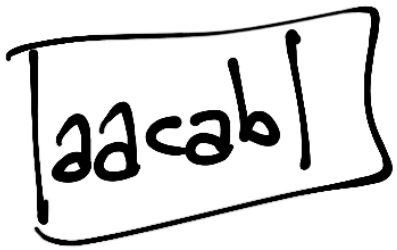
AGGIUNGO M ELEMENTI

USANDO ADATTATI

$$1+2+3+\dots+M =$$

$$= \frac{n(n+1)}{2} \approx n^2$$





package list

```
func AddInOrder(first *Node,
    item string) *Node {
    n := new(Node)
    n.data = item
    if first == nil {
        return n
    }
    var prev *Node = nil
    curr := first
    for curr != nil && curr.data < item {
        prev = curr
        curr = curr.next
    }
    if curr == nil {
        return AddTail(first, item)
    }
}
```

```
if curr == first {
    return AddFront(first, item)
}
prev.next = n
n.next = curr
return first
}
```

order.go

```
package main
import (
    "os"
    "list"
)
func main() {
    var l *list.Node
    for i:=1; i<len(os.Args); i++ {
        l = list.AddInOrder(l,
            os.Args[i])
    }
    list.PrintList(l)
}
```

./order cane gatto amico dente

amico
cane
dente
gatto

```
package list

func concat(l1 *Node, l2 *Node)
        *Node {

    switch {
        case l1 == nil:
            return l2
        case l2 == nil:
            return l1
        default:
            curr := l1
            for curr.next != nil {
                curr = curr.next
            }
            curr.next = l2
            return l1
    }
}
```