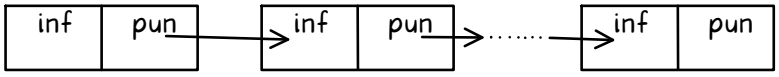
2-Liste

Lista

- Sequenza a dimensione variabile di elementi omogenei ad accesso sequenziale.

NB. Gli elementi di una lista non occupano celle di memoria contigue.



Nodo Lista

```
struct elem {

    int inf;
    elem* pun;

 };

typedef elem* lista;
```

Primitiva HEAD

-Funzione che ritorna la testa della lista

```
int head (lista l) {

    return l->inf;

}
```

Primitiva TAIL

- Funzione che ritorna il puntatore all'elemento successivo

```
lista tail (lista l) {

    return l->pun;

}
```

Primitiva STAMPALISTA

-Funzione chestampa i valori della lista

```
 void stampa_lista (lista l) {

    while (l != NULL){
         cout << head(p);
         p = tail p;
    }
}
```

Primitiva INSERTELEM

-Funzione che aggiunge un elemento in testa alla lista e lo ritorna

```
lista insert_elem (lista l, elem* e) {

    e -> pun = l;
    return e;
}
```

CREALISTA

-Funzione che crea una lista e la riempie

```
lista crea_lista (int n) {

    lista Testa = NULL;

    for (i = 1; i <= n; i++;){
         elem *p = new elem;
         cout<<"Valore elemento"<<i<<":";
         cin>>p->inf;
         Testa = insert_elem(testa,p);
    }
 return Testa;
}
```

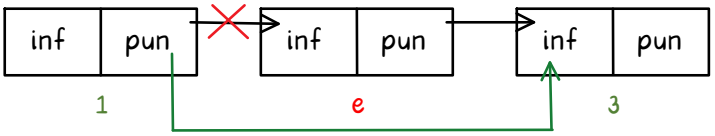
Primitiva DELETE_ELEM

-Funzione che cancella un elemento della lista

```
lista delete_elem (lista l, elem* e) {

    if (l == e)
         l = tail(l);
    else
         lista l1 = l;

         while (tail(l1) != e){
              l1 = tail(l1);
              l1 -> pun = e -> pun;
         }
    delete e;
    return l;
}
```



ELIMINALISTA

-Funzione che elimina tutta la lista

```
lista elimina_lista (lista  &Testa) {

    while (Testa != NULL){
         Testa = delete_elem(Testa,Testa);
    }
}
```

Primitiva SEARCH

-Funzione che cerca nella lista l il valore v

```
elem* search (lista l, int v) {

    while (l != NULL){
         if (head(l) = v)
              return l;
         else
              l = tail(l);
    }
    return NULL;
}
```

Primitiva COPY

-Funzione che ritorna una copia della lista

```
lista copy (lista l1) {

    lista l = NULL;
    elem* curr;
    elem* prev = NULL;

    while (l1 != NULL){

         curr = new elem;
         curr -> inf = head(l1);
         curr -> pun = NULL;

         if (prev == NULL)
              l = curr;
         else{
              prev -> pun = curr;
              prev = curr;
              l1 = tail(l1);
              }
    }
    return l;
}
```