PIEMONTE
FONDI STRUTTURALI E DI INVESTIMENTO EUROPEI 2014/2020
fondo sociale europeo
ITS TECNOLOGIE DELL'INFORMAZIONE E DELLA COMUNICAZIONE

FONDAMENTI DI PROGRAMMAZIONE

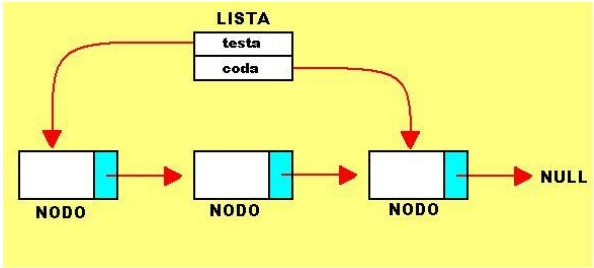
C / C++

Docente: Armando Valentino

in collaborazione con:
ISTITUTI TECNICI SUPERIORI
Ministero dell'Istruzione
UNIONE EUROPEA
REGIONE PIEMONTE
per una crescita intelligente, sostenibile ed inclusiva
www.regione.piemonte.it/europa2020
INIZIATIVA CO-FINANZIATA CON FSE

LINGUAGGIO C++

Strutture dati Dinamiche



Prof. Armando Valentino

POR Piemonte
FSE 2014-2020

Strutture dati dinamiche

Le **strutture dati dinamiche** sono strutture in cui **il numero degli elementi** non è fisso, ma **può variare durante l'esecuzione**

Le **strutture dati si dicono lineari** se per ogni elemento è definito un solo successore e un solo predecessore (Liste)

Le **strutture dati si dicono NON lineari** se per ogni elemento si possono avere più successori o più predecessori (Alberi, Grafi)

Per **realizzare una struttura dati dinamica**, si **usano i puntatori**.
Si può simulare una strutture dati dinamiche mediante array.

LISTE

Una **lista è un insieme di valori omogenei**, cioè **tutti dello stesso tipo**, in cui è stato stabilito un ordine nella posizione.

Per **ogni elemento della lista** (**nodo**) è definito il **predecessore** e il **successore**.
Il primo elemento non ha predecessore. L'ultimo elemento non ha il successore.

Non si può accedere direttamente ad un elemento della lista
Per accedere ad un elemento della lista bisogna **scorrere tutti gli elementi che lo precedono (accesso sequenziale)**

La lista si dice a **lunghezza variabile** se il numero **di elementi varia nel tempo** per effetto di **inserimenti** e **cancellazioni**

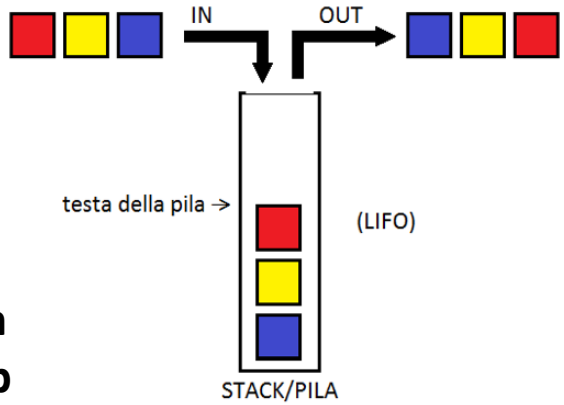
PILA

La PILA (detta anche STACK) è una lista lineare in cui gli **elementi** si possono **inserire o estrarre da un solo estremo**

Il criterio per inserire o estrarre elementi si dice **LIFO** (Last In First Out,l'ultimo che entra è il primo che esce), cioè viene estratto per primo l'ultimo valore inserito.

Per capire la PILA si può pensare ad un **PILA di piatti su un tavolo**;
Ogni piatto lavato si appoggia in cima e ogni volta che serve un piatto si prende il primo della cima.

L'operazione di inserimento in cima viene indicato con **push**
L'operazione di estrazione dalla cima viene indicato con **pop**

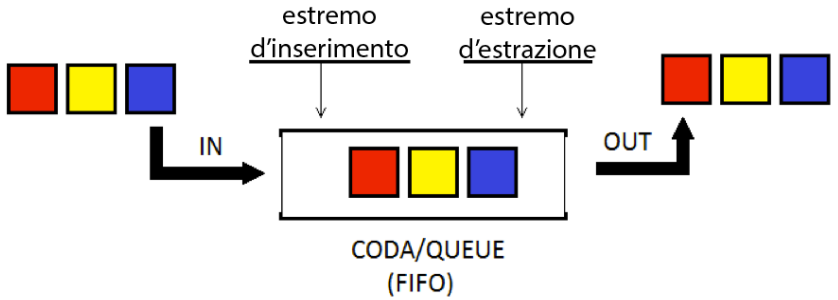


CODA

La CODA è una lista lineare in cui gli **elementi** si possono **inserire da un estremo ed estratti dall'altro estremo**

Il criterio per inserire o estrarre elementi si dice **FIFO** (First In First Out,il primo che entra è il primo che esce), cioè viene estratto per primo l'elemento che è stato inserito per primo.

Per capire la CODA si può pensare ad una **fila di persone davanti ad uno sportello che vengono servite nell'ordine con cui sono arrivate;**
Oppure si può pensare alle macchine in coda ad un semaforo.



Lista Concatenata

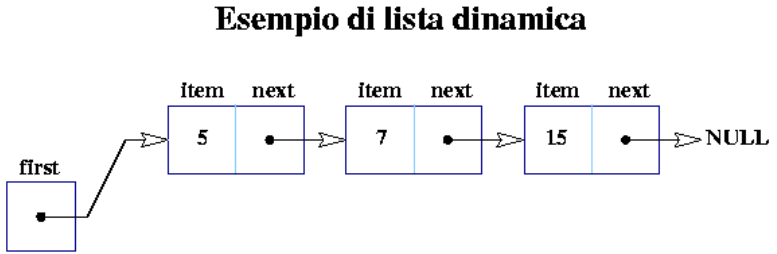
La Lista Concatenata è **una lista** in cui si possono **inserire o estrarre elementi anche all'interno della lista**

Una **lista concatenata unidirezionale** è formata da due parti:

a) **Il dato (item)**
b) **Valore** che indica **l'elemento successore** (**puntatore all'altro elemento**)

L'ultimo elemento non ha successore e quindi contiene un valore che indica **la fine della catena**
Bisogna **conoscere il primo elemento della catena** e poi si passa ai **successivi elementi tramite i puntatori**

L'accesso alla lista è sequenziale, cioè per accedere ad un elemento della lista si devono esaminare tutti gli elementi che lo precedono



Liste concatenate

Altre forme di liste concatenate:

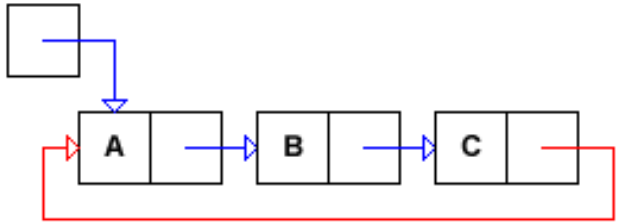
Lista concatenata circolare

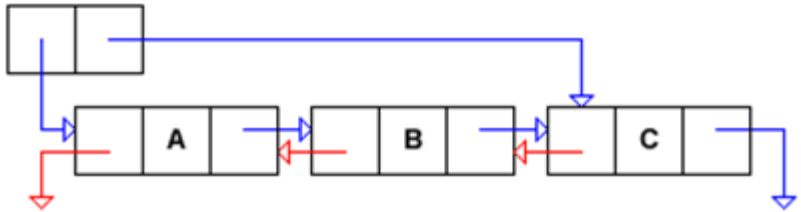
È una lista concatenata unidirezionale in cui l'ultimo elemento contiene un puntatore al primo elemento

Lista concatenata bidirezionale

In questa è formata da tre parit:

a) Il **dato**
b) Il **puntatore all'elemento successivo**
c) Il **puntatore all'elemento precedente**





Nell'ultimo elemento il puntatore successivo contiene un valore di **fine catena**

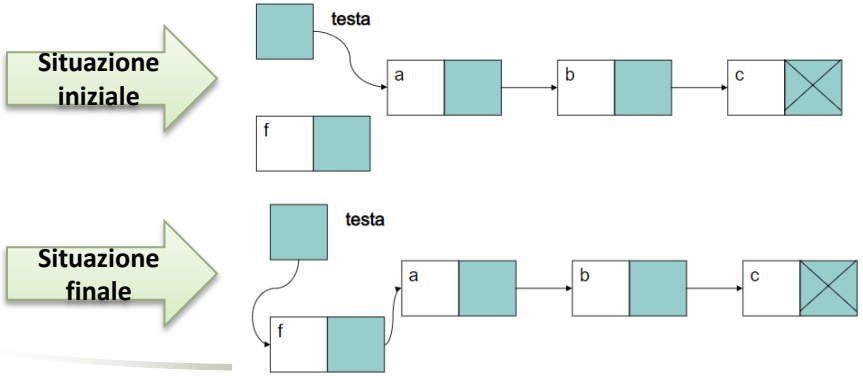
Nel primo elemento il puntatore al precedente contiene un valore di **fine catena**

Operazioni sulle liste concatenate

Le operazioni sulle liste sono: **Ricerca , Inserimento, Cancellazione** e vengono svolte utilizzando i puntatori

Ricerca:
Per cercare un elemento si devono scorrere tutti gli elementi: per ogni elemento si **confronta la parte che contiene il dato con il valore cercato**, se non corrisponde si passa all'elemento successivo tramite il puntatore. Si continua così fino a quando non si trova il valore oppure fino a quando si incontra il puntatore di fine catena.

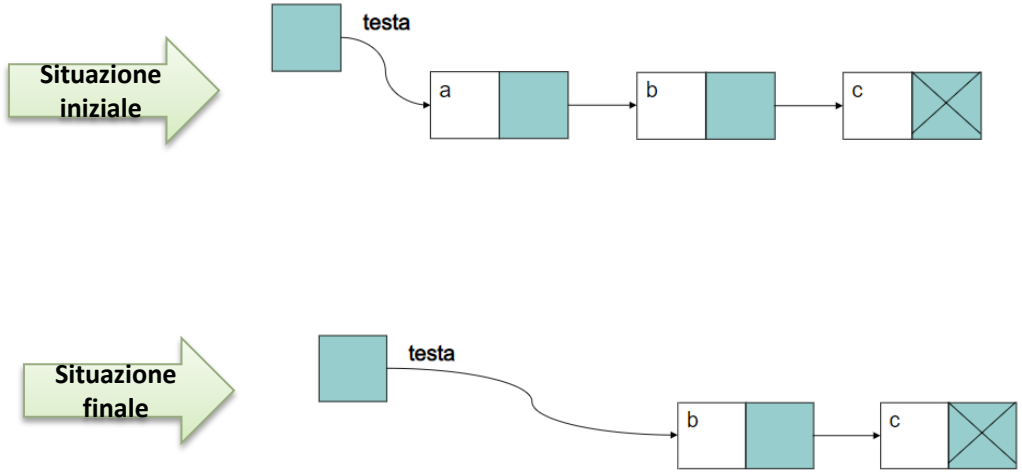
Inserimento all'inizio della lista:
Si conosce il puntatore al primo elemento della lista. Questo puntatore deve puntare al nuovo elemento della lista, mentre l'indirizzo del nuovo elemento deve puntare a quello che era il primo elemento della lista



Operazioni sulle liste concatenate

Cancellazione all'inizio della lista:

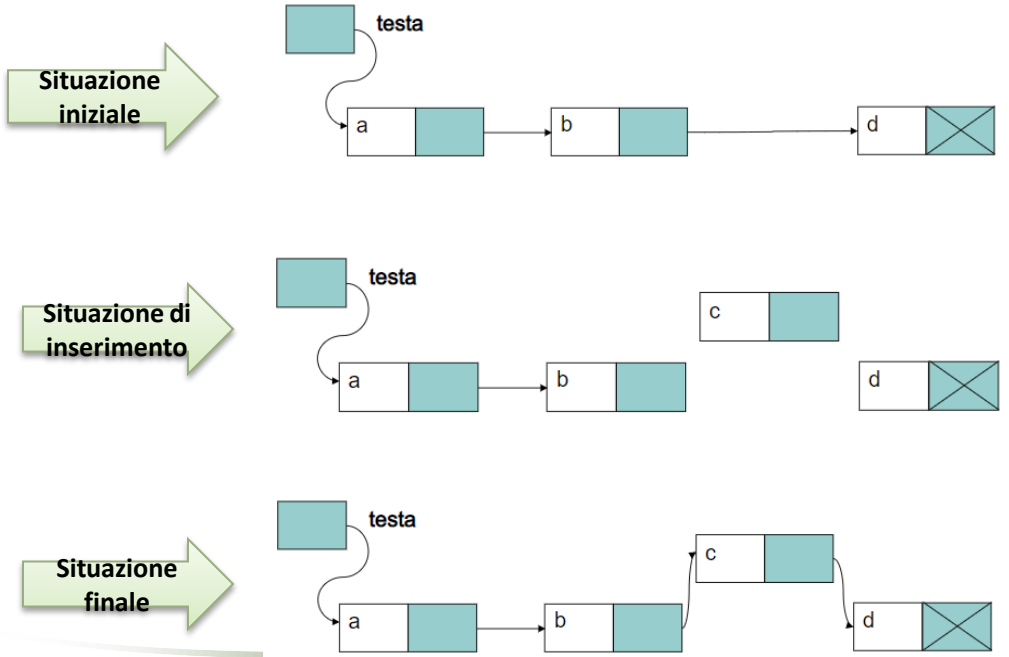
Si conosce il puntatore al primo elemento della lista. Questo puntatore deve puntare al secondo elemento della lista, saltando il primo elemento.



Operazioni sulle liste concatenate

Inserimento in mezzo alla lista:

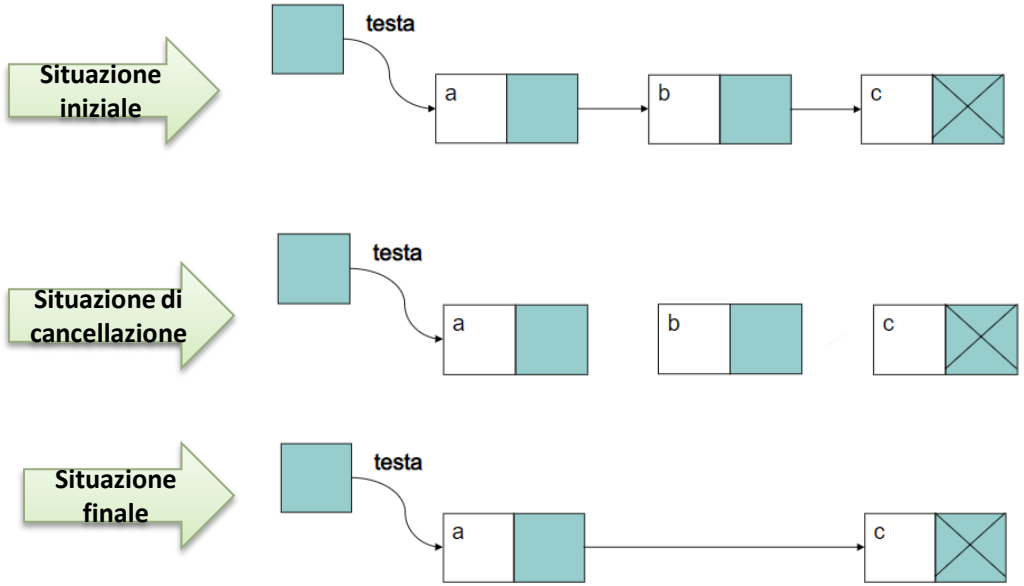
Si deve scorrere la lista fino a trovare il punto dove inserire l'elemento. Si deve trovare l'elemento che deve precedere quello nuovo. Il puntatore di questo elemento deve essere copiato nel nuovo elemento, poi deve essere modificato che punti al nuvo elemento



Operazioni sulle liste concatenate

Cancellazione in mezzo alla lista:

Si deve trovare l'elemento che precede quello da eliminare. Il puntatore dell'elemento da eliminare deve essere copiato nel puntatore dell'elemento che precede.

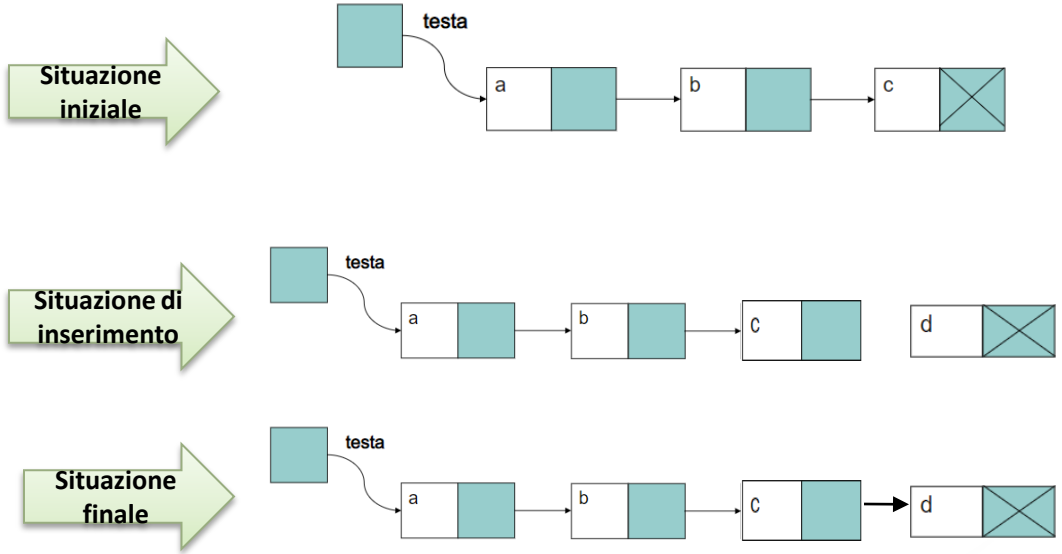


Operazioni sulle liste concatenate

Inserimento in coda alla lista:

Si deve scorrere l'intera lista fino all'ultimo elemento.

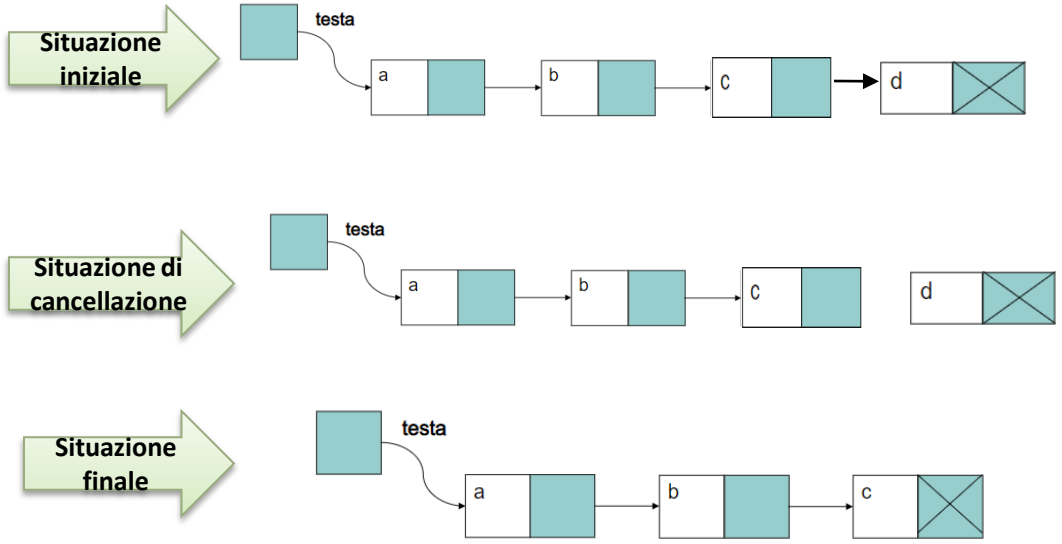
Il puntatore dell'ultimo elemento deve essere copiato nel nuovo elemento il valore di fine lista deve puntare al nuovo elemento inserito



Operazioni sulle liste concatenate

Cancellazione in coda alla lista:

Si deve scorrere l'intera lista fino all'ultimo elemento.
Prendere l'elemento che precede l'ultimo e impostare il valore di fine lista.



Allocazione dinamica memoria

Nel linguaggio C++

Per allocare memoria dinamicamente, durante l'esecuzione del programma si può usare l'operatore **new**.
Per eliminare l'allocazione dinamica della memoria si usa l'operatore **delete**

Esempio:

```
nodo *p;              // crea un puntatore di tipo nodo
p = new nodo;         // alloca una area di memoria dinamica con un elemento nodo, cioè crea un nodo vuoto.
delete p;             // elimina l'area di memoria allocata per la variabile p
```

Allocazione dinamica memoria

ITS TECNOLOGIE DELL'INFORMAZIONE E DELLA COMUNICAZIONE

Nel linguaggio C

In linguaggio C **non esiste** l'operatore new e per allocare la memoria si deve usare la funzione **malloc**() insieme alla funzione **sizeof**(*tipovariabile*) della libreria <stdlib.h>.
La funzione **malloc() ritorna un puntatore void** se riesce ad allocare la memoria oppure un **puntatore nullo** se l'operazione non riesce.
Per cancellare l'area di memoria dinamica allocata con malloc() di deve usare la funzione **free()**

Esempio:

```
nodo *p = (nodo* ) malloc(sizeof(nodo));
```

cast al tipo nodo

malloc() ritorna un **puntatore a void** che non è adatto ad essere contenuto in un puntatore lista, quindi si deve fare il cast da tipo void a tipo nodo

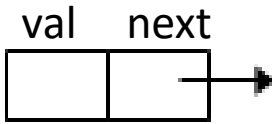
Per eliminare l'area di memoria allocata:

```
free(p);
```

PILA con i Puntatori

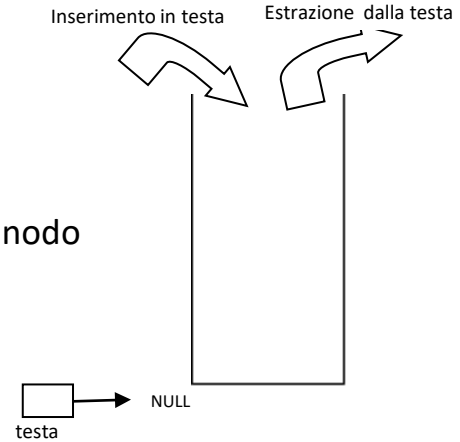
Per realizzare un nodo di una lista lineare si deve usare una struttura perché nel nodo sono presenti due elementi diversi. Supponiamo che vogliamo realizzare una **pila di numeri interi**, la definizione del nodo può essere la seguente:

```
struct nodo{
        int val;
        nodo * next;
};
```



il puntatore next deve essere della stesso tipo di nodo perché punta ad un elemento di tipo nodo

La pila richiede **l'inserimento e l'estrazione sempre dalla testa**.
Per realizzare la pila è necessario avere un **puntatore alla testa della pila**.
Inizialmente la pila è vuota quindi questo puntatore punta a NULL.




```
nodo *testa = NULL;
nodo *p;
p= new nodo;
```

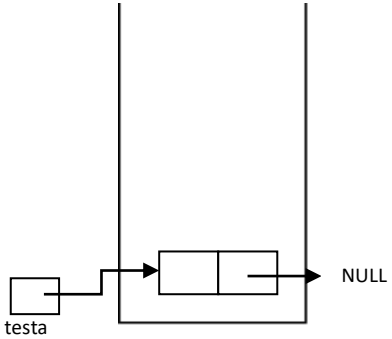
Crea un nuovo elemento nodo vuoto

PILA con i Puntatori

per inserire un valore preso da tastiera nel nuovo elemento creato

```
int v;
cout << "inserisci elemento" << endl;
cin >> v;
p->val = v ;
```

adesso si deve far puntare il **puntatore della testa al nuovo elemento**, e il puntatore del nuovo elemento essendo l'ultimo deve puntare a NULL.



Basta fare:

```
p->next = testa;        // inserisco il vecchio valore di testa nel puntatore del nuovo nodo
testa = p;              // inserisco nel puntatore testa il valore del puntatore del nuovo nodo
```

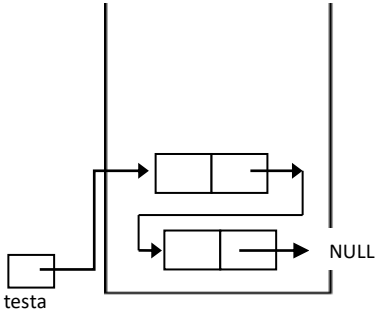
questa operazione produce come risultato la figura a lato

N.B. se sto inserendo il primo nodo, il vecchio valore di testa era NULL, e sono a posto, Se sto inserendo in testa un nodo ma c'erano anche altri nodi allora il vecchio valore di testa è l'indirizzo del primo nodo che deve essere messo nel puntatore del nuovo nodo e anche in questo caso va bene;

PILA con i Puntatori

Per inserire un nuovo elemento nella pila basta ripetere le stesse operazioni

```
p= new nodo;
int v;
cout << "inserisci elemento" << endl;
cin >> v;
p->val = v;
```



il nuovo nodo deve avere nel suo puntatore il valore del puntatore al precedente nodo (che è conservato nella variabile testa) e la nuova testa deve avere il puntatore del nuovo elemento. Quindi basta ripetere le stesse operazioni precedenti

```
p->next = testa;        // inserisco il vecchio valore di testa nel puntatore del nuovo nodo
testa = p;              // inserisco nel puntatore testa il valore del puntatore del nuovo nodo
```

questa operazione produce come risultato la figura a lato

Continuando queste operazioni si possono inserire altri elementi

PILA con i Puntatori

Estrazione di un elemento dalla PILA

L'estrazione avviene sempre dalla testa. Quindi conoscendo il valore del puntatore di testa si estrae il primo elemento,
L'estrazione implica anche la cancellazione dell'elemento dalla lista e il posizionamento della testa all'elemento successivo.

```
int elem;
nodo *p;
p = testa;                    // prendo il puntatore di testa
elem = p->val;
nodo *temp;
temp = p->next;
cout << "elimino elemento "<<i <<" :" << p->val << endl ;
delete p;
testa = temp;                 // posiziono il nuovo valore di testa all'elemento successivo
```

PILA (Stack) : ESERCIZI

1. Data una pila di elementi piena di cui si conosce il puntatore di testa
   Scrivere una funzione per contare il numero degli elementi.

2. Data una pila piena di elementi di cui si conosce il puntatore di testa.
   Scrivere una funzione per invertire gli elementi (il primo deve diventare l'ultimo e l'ultimo il primo.

3. Date due pile dello stesso tipo cui si conosce i relativi puntatori di testa.
   Scrivere una funzione per concatenare una pila all'altra.

4. Date due pile dello stesso tipo di elementi ordinati, di cui si conosce i relativi puntatori di testa. La prima pila è formata da caratteri (a,c,e,g,j,l,p,r,t,u), la seconda pila dai caratteri (b,d,f,h,i,m,n)
   Scrivere una funzione per fondere le due strutture.

PILA (Stack) : ESERCIZI

La percentuale degli interessi attivi sul deposito di un conto corrente bancario cambia più volte nel corso dell'anno.
Utilizzare una pila per memorizzare questi valori (utilizzare giorno, mese, anno e interesse come dati da memorizzare).
Fornito poi, il saldo attuale:
a) calcolare gli interessi lordi maturati usando l'ultimo valore dell'interesse.
b) Supponendo che il saldo sia rimasto fisso tutto l'anno, data in input una data di inizio e una data di fine, calcolare la somma degli interessi dei periodi tra le due date. Calcola anche il nuovo saldo.

Licenza