ITS TECNOLOGIE DELL'INFORMAZIONE E DELLA COMUNICAZIONE

Tecnico superiore per i metodi e le tecnologie per lo sviluppo di sistemi software

- BACKEND SYSTEM INTEGRATOR

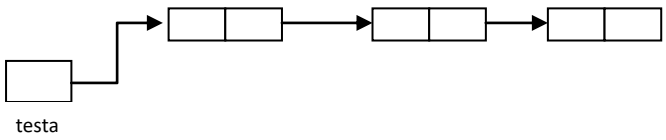
Unità Formativa (UF) **FONDAMENTI DI PROGRAMMAZIONE**

Docente: **VALENTINO ARMANDO**

Titolo argomento: **Struttura dinamica - Pila**

Strutture dinamiche e allocazione dinamica della memoria in C++

Le strutture dinamiche LINEARI sono la PILA, CODA, NODO CONCATENATA e sono strutture in cui ogni elemento ha un predecessore e un successore
Le strutture dinamiche NON LINEARI sono ALBERI E GRAFI in cui ogni elemento può avere più predecessori o più successori.
In una nodo lineare ogni elemento è fatto da due parti: il dato (valore) e il puntatore al nodo successivo



Per realizzare un nodo di una nodo lineare si deve usare una struttura perché nel nodo sono presenti due elementi diversi, il valore dell'elemento e il puntatore al successivo.
Supponiamo che vogliamo realizzare una nodo lineare 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 nodo di tipo nodo.

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.**
L'operatore **new** è un operatore utilizzabile in linguaggio C++; nel linguaggio C non esiste questo operatore.

Per creare una nodo vuoto:

```
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
```

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 non è riuscita ad allocare la memoria.
Per cancellare l'area di memoria dinamica allocata con **malloc()** di deve usare la funzione **free()**
Per esempio per allocare un elemento di tipo nodo come la struttura dichiarata sopra si deve usare

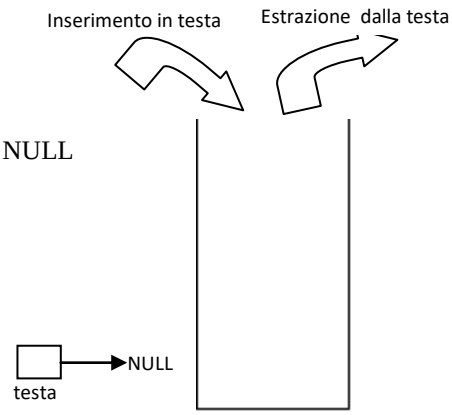
```
nodo *p = (nodo* ) malloc(sizeof(nodo));
```

malloc() ritorna un puntatore a void che non è adatto ad essere contenuto in un puntatore di tipo nodo, quindi si dovrà fare il cast da void a nodo scrivendo la parola chiave (nodo *) davanti alla funzione malloc.
Per liberare la memoria allocata si deve fare `free(p);`

in linguaggio C++ si possono usare entrambe le modalità; noi useremo la prima, cioè l'operatore **new**.

Realizzazione di una PILA con puntatori

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;   //dichiarazione di un puntatore alla testa con valore iniziale NULL
nodo *p;
```

per creare un nuovo elemento vuoto:

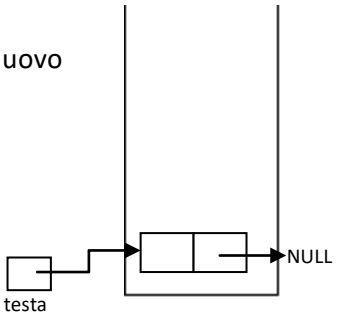
```
p = new nodo;
```

per inserire un valore nell'elemento nuovo, prendo un valore da tastiera

```
int v;
cout << "inserisci elemento" << endl;
cin >> v;
p->val = v;     //assegno il valore preso in input all'elemento val del nodo
```

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.

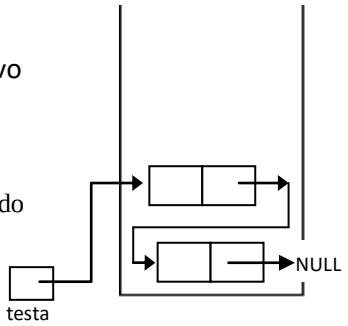
N.B. se sto inserendo il primo nodo, il vecchio valore di testa era NULL, e sono a posto, Se sto inserendo in testa un nodo ma c'erano anche altri nodi allora il vecchio valore di testa è l'indirizzo del primo nodo che deve essere messo nel puntatore del nuovo nodo e anche in questo caso va bene;

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.

Stampa degli elementi della PILA

Per stampare tutti gli elementi della PILA si deve conoscere il valore della testa.
Dalla testa si possono visitare tutti gli elementi della PILA.

```
// stampa elementi della nodo
        nodo *p;
        p = testa;
        i = 0;
        while (p!=NULL){
        i++;
        cout << "elemento " <<  i << " : " << p->val << endl ;
        p = p->next;
        }
```

Eliminazione di tutti gli elementi della nodo (estrazione di tutti gli elementi della nodo)

```
        nodo *p;
        p = testa;                      // posiziono nel puntatore p il primo elemento della pila che è testa
        i = 0;
        while (p != NULL){
                i++;
                cout  << "elimino elemento " << i << " : " << p->val << endl ;
                nodo *temp;
                temp = p->next;
                delete p;
                p = temp;
        }
```

Ricerca di un elemento nella PILA

```
        if (testa!=NULL){
                int v;
                cout<<  "inserisci elemento da cercare:"<<endl;
                cin >> v;                       // prendo un valore da tastiera
                int pos;
                int i=0;
                // ciclo di ricerca. se lo trovo mi conservo la posizione (a cominciare dall'alto)
                while (p!=NULL){
                        i++;
//                      cout << "elemento "<< i <<" :" << p->val << endl ;
                        if (p->val = = v){
                          // elemento trovato
                          pos = i;
                          break;
                        }
                        p=p->next;          // passo al successivo elemento
                }
                if (pos!=0)
                        cout << " elemento trovato alla posizione " << pos << endl;
                else
                        cout << "elemento non trovato" << endl;
        }else {
                cout << "PILA VUOTA" << endl;
        }
```

Estrazione di un elemento dalla PILA

L'estrazione avviene sempre dalla testa. Quindi conoscendo il valore del puntatore testa si estrae il primo elemento.

L'estrazione implica anche la cancellazione dell'elemento dalla nodo 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
```

SCRIVERE UNA FUNZIONE PER CREARE UN NODO E INSERIRE UN ELEMENTO NELLA PILA

Si possono scrivere due funzioni:

una funzione che crea il nodo con richiesta in input dell'elemento e che ritorna il puntatore al nodo creato

e un'altra funzione che inserisce il nodo nella pila, modificando i puntatori del nodo e della testa

funzione creanodo()

Non ha bisogno di parametri in ingresso. Deve ritornare un puntatore di tipo nodo, e deve chiedere di inserire l'elemento da tastiera e ritornare il puntatore al programma main(), come segue:

```
nodo * creaNodo(void){
        int elem;
        cout<<  "inserisci elemento: "<<endl;
        cin >>elem;                 // prendo un valore da tastiera
        nodo *p;
        p = new nodo;               //  creo un nodo da inserire nella pila
        p->val= elem;
        return p;
}
```

funzione inserisci() ha bisogno di due parametri in ingresso, il puntatore dell'elemento creato in precedenza (es: nodo *p) e il puntatore testa; all'interno della funzione il valore di testa deve essere modificato e la modifica deve poi essere visibile nel main(). Se il puntatore testa viene modificato all'interno della funzione, per rendere visibile nel main() la modifica, il parametro testa si deve passare per indirizzo altrimenti, passandolo per valore, la modifica viene persa all'uscita della funzione.

```
inserisci(nodo *p, nodo **testa){
        p->next=*testa;             // vecchio valore della testa lo metto nel nuovo nodo
        *testa=p;                   // la nuova testa diventa il puntatore del nuovo nodo
}
```

Come si vede il puntatore testa ha due asterischi (**) perchè deve essere passato per indirizzo

All'interno del main quindi avremo la funzione inserisci(p, &testa) in cui il puntatore p viene passato per valore e il puntatore testa viene passato per indirizzo mediante l'operatore &.

```
int main(){
        nodo *testa=NULL;       // creo la testa della pila
        nodo *p;
        p=creaNodo();
        inserisci(p, &testa );
        cout << "elemento " << p->val << " inserito nella Pila" << endl;
        system ("pause");
}
```

SCRIVERE UNA FUNZIONE PER ESTRARRE UN ELEMENTO DALLA PILA

Per estrarre un elemento abbiamo bisogno della testa della Pila.
Si deve passare alla funzione, come parametro, il puntatore alla testa. Il passaggio del parametro deve avvenire per indirizzo perche dopo il prelievo del valore del primo elemento di testa, l'elemento deve essere cancellato e si deve modificare il valore di testa impostandolo al successivo elemento della Pila.
La funzione può ritornare il valore dell'elemento da cancellare. Se la pila è vuota ritorna il valore 0.

Riepilogo:
Parametri da passare alla funzione nodo **testa
Valore di ritorno della funzione: int

La funzione è la seguente:

```
int estrai(nodo **testa){
  if (testa!=NULL){
        nodo *p;
        p = *testa;                 // prendo il puntatore di testa
        int elem = p->val;          // prendo il valore e lo assegno alla variabile elem.
        nodo *temp;
        temp = p->next;             // prima di cancellare l'elemento mi salvo il puntatore
        cout << "elimino elemento : " << p->val << endl ;
        delete p;
        *testa = temp;      // posiziono il nuovo valore di testa all'elemento successivo
        return elem;
  } else{
        return 0;
  }
}
```

Nel main si può scrivere come segue:

```
int main(){
	/* inserimento di un elemento nella pila
	……
	*/

	/* estrazione di un elemento dalla pila
	cout<<"ESTRAZIONE DI UN ELEMENTO DALLA PILA"<<endl;
	int a;
	a=estrai(&testa);
	if (a!=0){
		cout << "elemento "<< a << " cancellato" << endl;
	}
	else {
		cout << "elemento "<< a << " non cancellato" << endl;
	}
}
```