Liste

Implementazioni:

- Stack
- Coda

- Imparare a Progettare Algoritmi e non ad Imparare le Regole Sintattiche di un Linguaggio
- Elencare le Situazioni di Funzionamento Critico dell'algoritmo
- Dato un Algoritmo Mediamente Funzionante verificarne il funzionamento in Situazioni Critiche
- Contare le Occorrenze dei Valori presenti in una Lista
- Scrivere le funzioni nel Main:
  - Testarle
  - Posizionarle nei file h e cpp

Struttura del NODO di una Lista

```
typedef struct NODO
 {
   int dato;
   struct NODO *next;
                                } nodo;
typedef nodo *list;
list inizioLista=NULL;//Lista Vuota
```

contaNodi

```
int contaNodi(list TestaDellaLista)
{
      list ultimo=TestaDellaLista;
      int conta=0;
      while(ultimo!=NULL) {
            if(ultimo->next==NULL) return ++conta; //Restituisci l'ultimo nodo
            ultimo=ultimo->next; //Continua a Contare
            ++conta;
         }
     return conta; //Lista Vuota Restituisce NULL
 };
```

stampaCodaLista

```
void stampaCodaLista(list TestaDellaLista)
{
    list scorriLista=TestaDellaLista;
     if(TestaDellaLista!=NULL)
        while(scorriLista!=NULL)  {
                        cout<<endl <<scorriLista->dato<<endl;
                        scorriLista=scorriLista->next;
                              }
     else cout<<"LISTA VUOTA"<<endl;
};
```

ultimoNodoLista

```
list ultimoNodoLista(list TestaDellaLista)
{
        list ultimo=TestaDellaLista;
       while(ultimo!=NULL)
            {
             if(ultimo->next==NULL) return ultimo; //Restituisci l'ultimo nodo
             ultimo=ultimo->next; //Continua a Scorrere
            }
    return ultimo; //Lista Vuota Restituisce NULL
 };
```

cercaValoreNodo

```
void cercaValoreNodo(list TestaDellaLista, int valoreCercato)
{
        list nuovoNodo= new nodo;
       int posizioneNodoCercato=0;
       int trovato=1;
       list scorri=TestaDellaLista;

        if(scorri!=NULL){ //Controllare se Nella Lista e' Presente almeno un Nodo
             if((valoreCercato==scorri->dato) && (trovato==1)){ //Lista con un Elemanto
                   cout<<"Elemento: "<<valoreCercato<<" Presente in Posizione: "<<++posizioneNodoCercato;
                                                                                  }
              else while((scorri->next!=NULL) && (trovato==1))
                  {
                      if(scorri->dato==valoreCercato){
                                    trovato=0;
                         cout<<"Elemento: "<<valoreCercato<<" Presente in Posizione: "<<++posizioneNodoCercato;
                       }
                         else {  scorri=scorri->next;
                                  posizioneNodoCercato++;
                                }
                   }
             if(trovato==1) cout<<" Elemento NON TROVATO "<<endl;
           }
        else {
                cout<<"Lista Vuota";
                                }
 };
```

LISTA GESTITA COME CODA

Codice TEST

https://onlinegdb.com/MuJYM0H1I

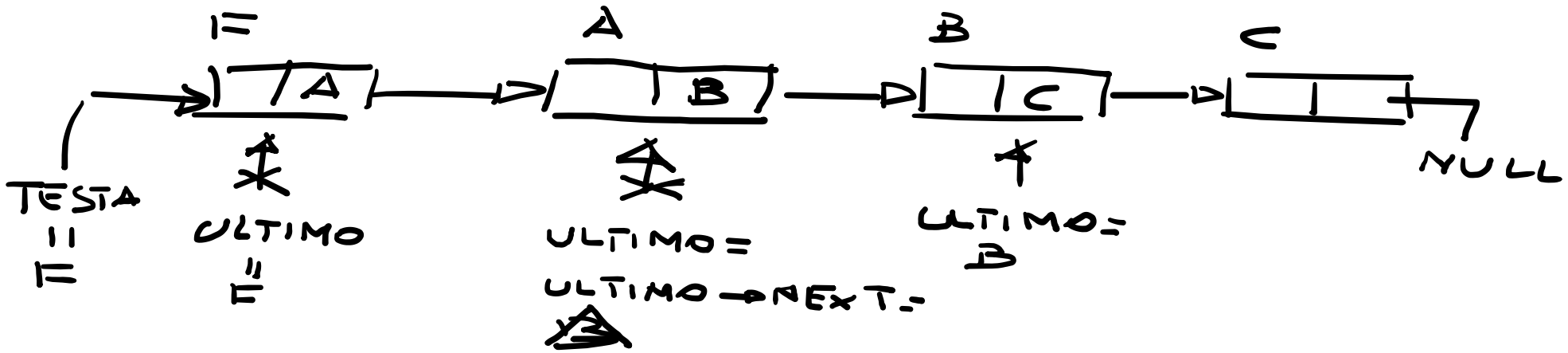
pushNodoStackLista

```
void pushNodoCodaLista( list &TestaDellaLista, int valoreDaInserire)
 {
     //list nuovoNodo= new nodo;
    list nuovoNodo= new nodo;
   nuovoNodo->dato=valoreDaInserire;//Inseriamo il dato
   nuovoNodo->next=NULL;//Dovendo diventare l'Ultimo Nodo lo prepariamo

   list ultimoNodo;//Cerchiamo la Posizione in cui inserire il Nuovo Nodo: in CODA alla lista
   ultimoNodo=TestaDellaLista;
   ultimoNodo=ultimoNodoLista(TestaDellaLista)//Cerca l'Ultimo Nodo della Lista
                        if(ultimoNodo!=NULL){ //Lista PIENA
                                        ultimoNodo->next=nuovoNodo;
                                        }
                        else{ //Lista VUOTA e Modifica della Testa
                            TestaDellaLista=nuovoNodo;
                            }
  };
```

popNodoCodaLista

```
int popNodoCodaLista( list &TestaDellaLista)
 {
                list ultimo=TestaDellaLista;
                int valore;
                  if(TestaDellaLista!=NULL)
                                {
                                  if(ultimo->next!=NULL){
                                     while(ultimo->next->next!=NULL) {
                                           ultimo=ultimo->next; //Avanza Continua a Scorrere
                                           }
                                     valore=ultimo->next->dato;
                                     ultimo->next=ultimo->next->next;
                                     return valore; //Restituisci l'ultimo nodo
                                     }
                                  else{
                                     valore=ultimo->dato;
                                     TestaDellaLista=ultimo->next;
                                     return valore; //Restituisci l'ultimo nodo
                                  }
                                }
            else  {
                        cout<<endl<< "NON POSSO ESTRARRE NODI"<<endl;
                        return -1; //Lista VUOTA ATTENZIONE IL -1 POTREBBE ESSERE UN VALORE PRESENTE NELLA LISTA: MODIFICARE ALGORITMO
}
                    };
```


A
B
C
NULL
TESTA
ULTIMO
ULTIMO =
ULTIMO = B
ULTIMO -> NEXT =

LISTA GESTITA COME STACK

pushNodoStackLista

```
void pushNodoStackLista( list &TestaDellaLista, int valoreDaInserire)
{
    list nuovoNodo= new nodo;
    nuovoNodo->next=TestaDellaLista;
    nuovoNodo->dato=valoreDaInserire;
   TestaDellaLista=nuovoNodo;//Inserimento IN TESTA
 };
```

popNodoStackLista

```
void popNodoStackLista( list &TestaDellaLista)
{
  if(TestaDellaLista!=NULL){ //Lista PIENA
     cout<<"Il TOP dello Stack di Valore: "<<TestaDellaLista->dato<<" E' stato Estratto"<<endl;
     TestaDellaLista=TestaDellaLista->next;
   }
  else{ //Lista VUOTA e Modifica della Testa
            cout<<"Lista Vuota"<<endl;
         }
};
```