



UNIVERSITY OF TRENTO - Italy

Laboratorio 18

LIFO con liste semplicemente concatenate

DISI – aa 2024/25

Pierluigi Roberti
Cermelo Ferrante

Università degli Studi di Trento

Prima di partire

Scaricare il file:

E18.3 ListeSemplici(struct).zip

decomprimerlo in una cartella

aprire il progetto e usare le funzioni

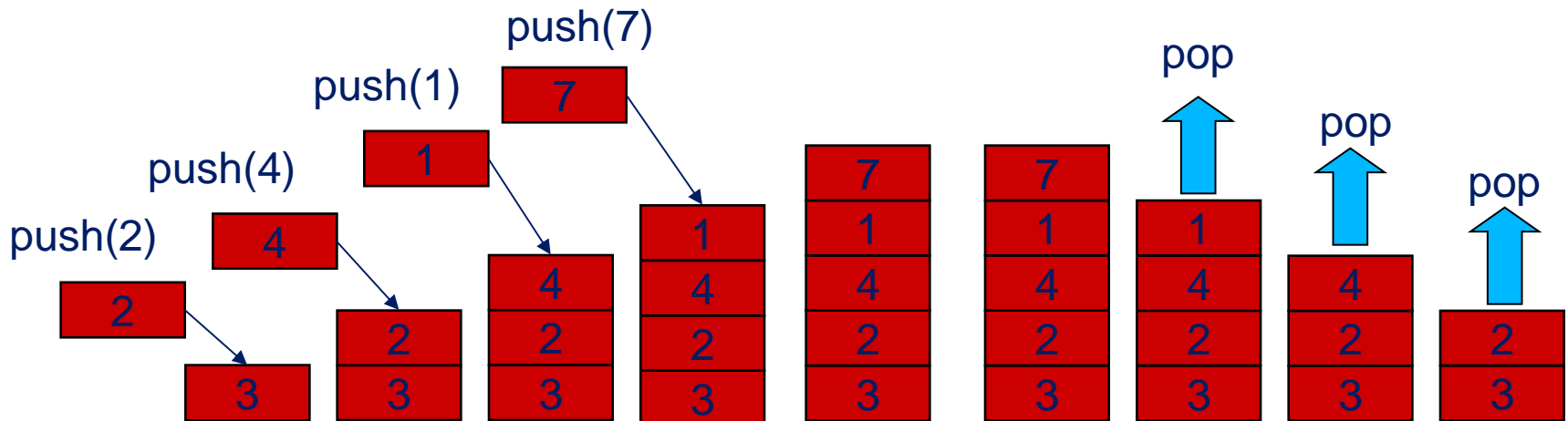
- Nodoptr **removeLast** (Nodoptr s);
- Nodoptr **removeFirst** (Nodoptr s);
- void **stampa** (Nodoptr s);
- Nodoptr **insertFirst** (Nodoptr s, Tdato CurrD);
- Nodoptr **insertLast** (Nodoptr s, Tdato CurrD);

per implementare le funzioni richieste nei diversi esercizi

ADT: Coda-LIFO

Definizione: uno **stack** (o pila o lista **LIFO**) è un ADT che supporta due operazioni base:

- **push** – inserimento (in testa) di un nuovo elemento;
- **pop** – cancellazione (in testa) dell'elemento che è stato inserito più di recente e ne ritorna il valore.



Esercizio1 - Stack con lista concatenata

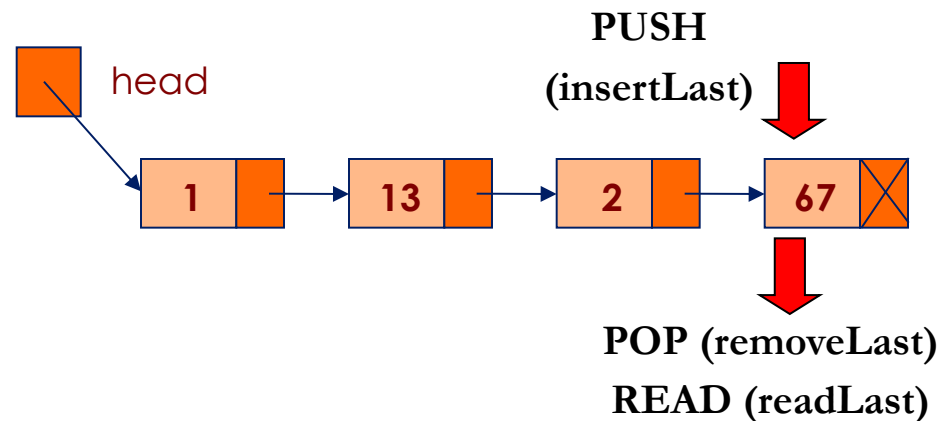
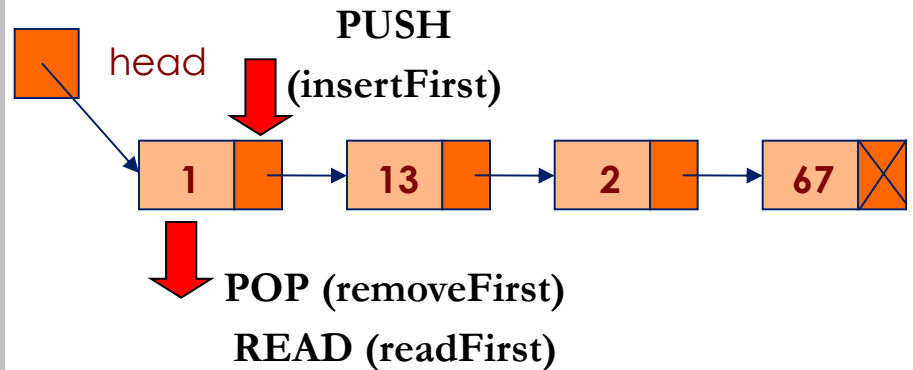
- Vogliamo scrivere un programma che implementa uno **stack** usando le liste concatenate

```
typedef struct Tnodo {  
    Tdato dato;  
    Tnodo * next;  
    Tnodo () {  
        next = NULL;  
    }  
    Tnodo (Tdato x, Tnodo * n) {  
        dato = x; next = n;  
    }  
} Tnodo;
```

Implementare il tipo “**Tdato**” che contiene:

- cognome** array di caratteri
- nome** array di caratteri
- eta** numero intero senza segno

Creare un file denominato “tipo_dati.h”



Esercizio 1- Stack con lista concatenata

- Definire ed implementare le seguenti funzioni:
 - **Tnodo* push (Tnodo* p, Tdato d)** : inserisce in testa alla lista
 - **Tnodo* pop (Tnodo* p)** : toglie l'elemento in testa alla lista
 - **Tdato read (Tnodo* p)** : legge l'elemento in testa alla lista
 - **void stampa (Tnodo* p)** : stampa il contenuto della lista (dall'elemento di testa fino a quello di coda)
 - **bool daticmp (Tdato d1, Tdato d2)**; confronta 2 elementi
 - **int length (Tnodo* p)**; conta quanti elementi ci sono nella coda
- Creare i file **LIFO.h** e **LIFO.cpp** che implementano la lista e le funzioni per usarla

Esercizio 1- Stack con lista concatenata

Main di prova

- Dichiarare un puntatore a Tnodo: stack
 - E inizializzarlo correttamente → Coda vuota
- Dichiarare una variabile di tipo Tdato: x
- Inizializzare x con dati a scelta
- Aggiungere x alla coda (push)
- Inizializzare x con dati a scelta
- Aggiungere x alla coda (push)
- Inizializzare x con dati a scelta
- Aggiungere x alla coda (push)
- Stampare la coda
- Rimuovere un dato (pop)
- Stampare la coda

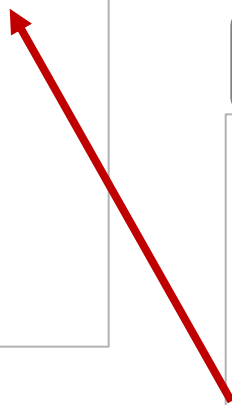
Esercizio 1 – funzione PUSH insertFirst

FUNZIONE

```
Tnodo* push (Tnodo* s, Tdato d) {  
    Tnodo * n = new Tnodo();  
    n->dato = d;  
    n->next = s;  
    return n;  
}
```

MAIN

```
void main(){  
    Tnodo* mioStack=NULL;  
    Tdato d; //costruttore 0 param  
  
    mioStack =push(mioStack, d);  
}
```



Esercizio1 – funzione POP RemoveFirst

FUNZIONE

```
Tnodo* pop (Tnodo * s) {  
    if (s==NULL){  
        return s;  
    }  
    Tnodo* n = s;  
    s = s->next;  
    delete n;  
    return s;  
}
```

È meglio
controllare
prima (nel main)
se la lista è vuota

```
Tdato read (Tnodo * s){  
    Tdato d; //costruttore default;  
    if (s==NULL) //lista vuota  
        { return d; }  
    return s->dato;  
}
```

MAIN

```
void main(){  
    Tnodo* mioStack=NULL;  
    Tdato d;  
    mioStack =push(mioStack, d);  
    if (mioStack!=NULL) {  
        //leggo l'elemento  
        d = read(mioStack);  
        //tolgo l'elemento  
        mioStack = pop(mioStack);  
    }  
}
```


Esercizio1 – funzione stampa

FUNZIONE

```
void stampa (Tnodo * s){  
    Tnodo* p = s;  
    while(p!=NULL){  
        p->data.stampa();  
        p=p->next;  
    }  
}
```

Ho un puntatore, devo farne una copia, altrimenti modifico il puntatore del main!!!

Implementare un metodo **stampa** nel tipo Tdata!

MAIN

```
void main(){  
    Tnodo * mioStack;  
    //...  
    stampa(mioStack);  
    //...  
}
```

Esercizio1 - main alternativo

- Domandare all'utente di scegliere tra:
 - 1 => Leggere da tastiera valori di una variabile di tipo “Tdato”
 - 2 => Inserisce nello stack LIFO il dato letto (PUSH)
 - 3 => Stampare lo stack
 - 4 => Leggere dallo stack LIFO l'ultimo elemento inserito (READ)
 - 5 => Togliere dallo stack LIFO l'ultimo elemento inserito (POP)
 - 6 => Uscire dal programma
- Continuare a chiedere l'operazione da eseguire all'utente fino a che non seleziona 6
- Prima di uscire dal programma, estrarre tutti i dati dallo stack

Struttura dati

```
typedef struct Tdato {  
    char nome[MAXCHAR];  
    char cognome[MAXCHAR];  
    int eta;  
    Tdato () {  
        nome[0]='\0'; cognome[0]='\0'; eta=0;  
    }  
    Tdato (char _nome[MAXCHAR],char _cognome[MAXCHAR], int _eta){  
        strcpy(nome,_nome); strcpy(cognome,_cognome); eta=_eta;  
    }  
    void stampa() const{  
        cout << "nome:"<<nome<<" cognome:"<<cognome<<" eta:"<<eta <<  
        endl;  
    }  
    bool compara(Tdato d){  
        if ( (strcmp(nome,d.nome)==0) &&  
            (strcmp(cognome,d.cognome)==0) && (eta==d.eta) ) {  
            return true;  
        } else {  
            return false;  
        }  
    }  
}Tdato;
```

#define MAXCHAR 20