University M’hamed Bougara Boumerdes

Institute of Electrical and Electronics Engineering (ex: INELEC)

Algorithms and Data structures laboratory

LAB N°3

Friday, March 18, 2022

Superviser:

Dr. A. Zitouni

Students:

Madaoui Zakaria G3 Computer

Maallem Nassim G3 Computer

Objectives:

- Write a C++ program that implements a stack using pointers
- Write a C++ program that implements a queue using pointers
- The elements of these structures contain first and last name, and the age of a person.
- The program should provide a way to:
    - Initialize the data structures
    - Adding a new person
    - Removing/ popping a person
    - Traversing from first to last and vise versa

Task 1: Stack implementation

A stack is a FILO (First in Last out) data structure, and the most recenty added element to the stack is called the HEAD or the TOP of the stack.

In order to implement a stack of persons we wrote a struct that represents a Person which has first, last name,age and most importantly a **pointer to the element beneath it on the stack denoted 'prev'.**

In order to wrap this data structure, we created a class called **Stack.** This class keeps track of the HEAD of the stack using a pointer to it, as well as it provides all the necessary functions to use the stack.

Adding an element to the stack creates a new HEAD and assigns to its' **'prev'** the previous HEAD. In addition, the Stack class updates its HEAD pointer to the newly added element.

- **Code:**

```
#include<iostream>
#include<string>

using namespace std;

#define DIRECTION_FORWARD  0
#define DIRECTION_BACKWARD 1

typedef struct person person;

struct person {
    public:
    string first_name;
    string last_name;
    unsigned age;
    person* prev;
};
```

```
class Stack{
private:
    person *head;
    void person_print(person *p)
    {
        cout << "first name: " << p->first_name << " last name: "
             << p->last_name << " age: " << p->age << endl;
    }

public:
    Stack(){//initalizaton of the stack
        head = nullptr;
    }

    void push(string fname, string lname, unsigned age ){
        person* temp = new person;
        temp->first_name = fname;
        temp->last_name = lname;
        temp->age = age;

        if(head != nullptr){
            temp->prev = head;
            head = temp;
        }else{
            temp->prev  = nullptr;
            head = temp;
        }
    }

    void pop(){
        if(head == nullptr){
            cout << "not possible ! stack is empty" << endl;
        }else if(head->prev == nullptr){
            delete head;
            head = nullptr;
        }else{
            person* temp = head;
            head = head->prev;
            delete temp;
        }
    }

    void traverse(int direction){
        if (head == nullptr){
            cout << "not possible ! stack is empty" << endl;
            return;
        }
        person* temp = head;
        if(direction == DIRECTION_BACKWARD){
            while (temp != nullptr)
            {
                person_print(temp);
                temp = temp->prev;
            }

        }else{
            recurr_forward(head);
        }
    }

    void recurr_forward(person* p){
        if(p == nullptr) return;
        else {
```

```
            recurr_forward(p->prev);
            person_print(p);
        }
    }

};

int main(){
    Stack s;

    s.push("name1", "famname1", 20);
    s.push("name2", "famname2", 21);
    s.push("name3", "famname3", 51);

    char user_option = '\0';
    string temp_fname;
    string temp_lname;
    int    temp_age;

    while (user_option != 'q')
    {
        cout << "> Please select an option to proceed: \n"
             << "    i: insert new element\n"
             << "    d: delete an element \n"
             << "    f: traverse list forward \n"
             << "    b: traverse list backward \n"
             << "    q: to quit the program \n";
        cin >> user_option;

        switch (user_option)
        {
        case 'i': /*insert*/
            cout << "> person to insert: ";
            cin >> temp_fname >> temp_lname >> temp_age ;
            s.push(temp_fname, temp_lname, temp_age);
            break;

        case 'd': /*delete*/
            s.pop();
            break;

        case 'f': /*traverse forward*/
            s.traverse(DIRECTION_FORWARD);
            break;

        case 'b': /*traverse backward*/
            s.traverse(DIRECTION_BACKWARD);
            // list_traverse(l, DIRECTION_BACKWARD);
            break;

        case 'q': /*quit*/
            cout << "Quitting program ... ";
            break;

        default:
            cout << "Wrong input ! try again !\n";
            break;
        }
        cout << "\n-----------------------------------------\n\n";
    }

    return 0;
}
```

- **Output:**

We start by pushing 3 persons in the following sequence:

Name1 famname1 20, Name2 famname2 21, Name3 famname3 51

traversal from head to first output:

```
first name: name3 last name: famname3 age: 51
first name: name2 last name: famname2 age: 21
first name: name1 last name: famname1 age: 20
```

traversal from first to head output:

```
first name: name1 last name: famname1 age: 20
first name: name2 last name: famname2 age: 21
first name: name3 last name: famname3 age: 51
```

Popping an element output: (head gets removed)

```
first name: name2 last name: famname2 age: 21
first name: name1 last name: famname1 age: 20
```

Inserting a element: (added as head)

```
first name: name1 last name: famname1 age: 20
first name: name2 last name: famname2 age: 21
first name: te last name: st age: 77
```

Task 2: Queue implementation

A queue is quite similar to a stack, except that it is a FIFO data structure. For this reason we only had to made a few modifications to the stack code to make it a queue.

A queue has both a HEAD and a TAIL, new elements get added from the TAIL and are popped from the HEAD. For this reasons we added a pointer to the TAIL in the Queue class.

The other modification was only in the **push()** function, which was modified to insert new elements from the TAIL. This was by updating the TAIL pointer in the Queue class to the newly added element, As well as, updating the '**prev**' pointer of the old tail element to point to the new one.

- **Code:**

```
#include<iostream>
#include<string>

using namespace std;

#define DIRECTION_FORWARD  0
#define DIRECTION_BACKWARD 1
```

```
typedef struct person person;

struct person {
    public:
    string first_name;
    string last_name;
    unsigned age;
    person* prev;
};

class Stack{
private:
    person *head;
    void person_print(person *p)
    {
        cout << "first name: " << p->first_name << " last name: " << p->last_name << "
age: " << p->age << endl;
    }

public:
    Stack(){//initalizaton of the stack
        head = nullptr;
    }

    void push(string fname, string lname, unsigned age ){
        person* temp = new person;
        temp->first_name = fname;
        temp->last_name = lname;
        temp->age = age;

        if(head != nullptr){
            temp->prev = head;
            head = temp;
        }else{
            temp->prev  = nullptr;
            head = temp;
        }
    }

    void pop(){
        if(head == nullptr){
            cout << "not possible ! stack is empty" << endl;
        }else if(head->prev == nullptr){
            delete head;
            head = nullptr;
        }else{
            person* temp = head;
            head = head->prev;
            delete temp;
        }
    }

    void traverse(int direction){
        if (head == nullptr){
            cout << "not possible ! stack is empty" << endl;
            return;
        }
        person* temp = head;
        if(direction == DIRECTION_BACKWARD){
            while (temp != nullptr)
            {
                person_print(temp);
                temp = temp->prev;
```

```
            }

        }else{
            recurr_forward(head);
        }
    }

    void recurr_forward(person* p){
        if(p == nullptr) return;
        else {
            recurr_forward(p->prev);
            person_print(p);
        }
    }

};

int main(){
    Stack s;

    q.push("name1", "famname1", 20);
    q.push("name2", "famname2", 21);
    q.push("name3", "famname3", 51);

    char user_option = '\0';
    string temp_fname;
    string temp_lname;
    int    temp_age;

    while (user_option != 'q')
    {
        cout << "> Please select an option to proceed: \n"
             << "    i: insert new element\n"
             << "    d: delete an element \n"
             << "    f: traverse list forward \n"
             << "    b: traverse list backward \n"
             << "    q: to quit the program \n";
        cin >> user_option;

        switch (user_option)
        {
        case 'i': /*insert*/
            cout << "> person to insert: ";
            cin >> temp_fname >> temp_lname >> temp_age ;
            s.push(temp_fname, temp_lname, temp_age);
            break;

        case 'd': /*delete*/
            s.pop();
            break;

        case 'f': /*traverse forward*/
            s.traverse(DIRECTION_FORWARD);
            break;

        case 'b': /*traverse backward*/
            s.traverse(DIRECTION_BACKWARD);
            break;

        case 'q': /*quit*/
            cout << "Quitting program ... ";
            break;
```

```
        default:
            cout << "Wrong input ! try again !\n";
            break;
        }
        cout << "\n----------------------------------------\n\n";
    }

    return 0;
}
```

- **Output:**

We start by pushing 3 persons in the following sequence:

Name1 famname1 20, Name2 famname2 21, Name3 famname3 51

traversal from head to tail output:

```
first name: name1 last name: famname1 age: 20
first name: name2 last name: famname2 age: 21
first name: name3 last name: famname3 age: 51
```

traversal from tail to head output:

```
first name: name3 last name: famname3 age: 51
first name: name2 last name: famname2 age: 21
first name: name1 last name: famname1 age: 20
```

Popping an element output: (head gets removed)

```
first name: name3 last name: famname3 age: 51
first name: name2 last name: famname2 age: 21
```

Inserting a element: (added as tail)

```
first name: te last name: st age: 77
first name: name3 last name: famname3 age: 51
first name: name2 last name: famname2 age: 21
```