**Experiment 2**

(PART B: TO BE COMPLETED BY STUDENTS)

**(Students must submit the soft copy as per following segments within two hours of the practical. The soft copy must be uploaded on the Blackboard or emailed to the concerned lab in charge faculties at the end of the practical in case the there is no Black board access available)**

|  |  |
| --- | --- |
| Roll No. C013 | Name: Ashmit Jain |
| Program : BTI | Division: B |
| Batch: B1 | Date of Experiment: |
| Date of Submission: 7/8/2024 | Grade : |

* 1. **Tasks given in PART A to be completed here**

*(****Students must write the answers of the task(s) given in the PART A )***

#include<iostream>

using namespace std;

const int MAX\_SIZE = 100;

class Stack

{

private:

int stack[MAX\_SIZE];

int top;

public:

Stack()

{

top = -1;

}

bool isEmpty()

{

return (top == -1);

}

bool isFull()

{

return (top == MAX\_SIZE - 1);

}

void push(int element)

{

if(isFull())

{

cout<<"Stack overflow. Can't push";

}

else

{

stack[++top] = element;

cout<<"Pushed Element"<<element<<endl;

}

}

void pop()

{

if(isEmpty())

{

cout<<"Stack underflow. Can't pop";

}

else

{

cout<<"Element"<<stack[top--]<<"Is popped."<<endl;

}

}

int peek()

{

if(isEmpty())

{

cout<<"No top element"<<endl;

return -1;

}

else

{

return stack[top];

}

}

void display()

{

if (isEmpty())

{

cout << "Stack is empty. No elements to display." << endl;

}

else

{

cout << "Stack elements: ";

for (int i = 0; i <= top; ++i)

{

cout << stack[i] << " ";

}

cout << endl;

}

}

};

int main()

{

Stack MyStack;

int element;

int choice;

do

{

cout<<"Menu"<<endl;

cout << "1. Push Element" << endl;

cout << "2. Pop Element" << endl;

cout << "3. Peek Element" << endl;

cout << "4. Display Stack" << endl;

cout << "5. Exit" << endl;

cout<<"Enter choice: ";

cin>>choice;

switch(choice)

{

case 1:

cout<<"Enter the Element which needs to get pushed. "<<endl;

cin>>element;

MyStack.push(element);

break;

case 2:

MyStack.pop();

break;

case 3:

MyStack.peek();

break;

case 4:

MyStack.display();

break;

case 5:

cout<<"GOODBYEEE";

break;

default:

cout<<"Invalid input";

}

} while (choice != 5);

return 0;

}

**Using Recursion:**

#include <iostream>

#include <stack>

using namespace std;

void sortedInsert(stack<int> &s, int element) // insert an element in a sorted stack

{

if (s.empty() || element >= s.top()) // 90>=8

{

s.push(element); // 1 2 3 8 90

return;

}

int top = s.top(); // 1 2 3 8 90(pop), 9 >= 90

s.pop();

sortedInsert(s, element); //after pushing 1 2 3 8 9

s.push(top); //stored value will get pushed

}

// Function to sort the stack using recursion

void sortStack(stack<int> &s)

{

if (s.empty())

{

return;

}

int top = s.top(); //stores top value in top 1 2 3 8 9

s.pop();

sortStack(s); //it will pop until its empty

sortedInsert(s, top); //stored value will get pushed 9 8 3 2 1

}

// Helper function to print stack

void printStack(stack<int> s)

{

while (!s.empty())

{

cout << s.top() << " ";

s.pop();

}

cout << endl;

}

int main()

{

stack<int> s;

int n;

// Take number of elements input from the user

cout << "Enter the number of elements in the stack: ";

cin >> n; //5

// Take elements input from the user

cout << "Enter " << n << " elements:" << endl;

for (int i = 0; i < n; ++i)

{

int element;

cin >> element;

s.push(element); // 1 2 3 4 5

}

cout << "Stack before sorting: ";

printStack(s); //

sortStack(s);

cout << "Stack after sorting: ";

printStack(s);

return 0;

}

* 1. **Observations and Learning:**

*(****Students must write the observations and learning based on their understanding built about the subject matter and inferences drawn)***

**Implementing Stacks with Arrays**:

* You created a stack data structure using an array to store the elements.
* You implemented basic stack operations such as push (to add an element to the top of the stack), pop (to remove the top element), peek (to view the top element without removing it), and isEmpty (to check if the stack is empty).

**Sorting a Stack Using Recursion**:

* You used a recursive approach to sort the elements within the stack.
* The process involves removing elements from the stack and using recursion to sort them before placing them back in the correct order.

###### Conclusion:

*(****Students must write the conclusive statements as per the attainment of individual outcomes listed above and learning/observation noted in section B.2)***

In conclusion, I learned how to implement stacks and their operations using arrays, and I also explored how to sort stacks using recursion.