**SVKM’s NMIMS**

**Mukesh Patel School of Technology Management & Engineering**

**Computer Engineering Department**

Program: B.Tech Integrated Sem V

**Course: Basic Data Structures**

**2024-2025**

**Experiment No.04**

**A.1 Aim:**

To study and implement concept of Stack data structure.

**A.2 Prerequisite:**

1. Knowledge of different operations performed on Stack data structure

2. Fundamental concepts of C\C++.

**A.3 Outcome:**

**After successful completion of this experiment students will be able to**

1. Identify the need of appropriate selection of data structure
2. Identify the steps of stack data structure selection.
3. Implement stack data structure to solve the given problem
4. Enlist the applications of stack data structure.

**A.4 Theory:**

**A.4.1. Introduction of Stack**

Stack is a linear data structure which follows a particular order in which the operations are performed. The order may be LIFO (Last in first out) or FILO (First in last out).

The functions performed on stack are:

1. **Push:** Adds an item in the stack. If the stack is full, then it is said to be an overflow condition.
2. **Pop:** Removes an item from the stack. The items are popped in the reversed order in which they are pushed. If the stack is empty, then it is said to be an underflow condition.
3. **Peek:** Returns the top most element from the stack.

**Applications of Stack:**

1. Balancing of symbols
2. Infix to postfix\prefix conversion
3. Redo-undo features at many places like editors, photoshop
4. Forward and backward feature in web browsers

**A.5 Procedure/Algorithm:**

**A.5.1:**

**TASK 1:**

Write a program to implement ADT of Stack.

**TASK 2:**

The organization structure of “ABC Pvt. LTD” is given as per joining order. The company is in crisis and wants to start lay off. The employee who joined last is fired first. Implement the suitable data structure and display the order in which the company will fire employees.

|  |  |  |
| --- | --- | --- |
| Employee id | Order of joining |  |
| 101 | 1 | (joined first) |
| 102 | 2 |  |
| 107 | 3 |  |
| 108 | 4 |  |
| 111 | 5 |  |
| 112 | 6 | (joined last) |

PART B

(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. C146 | Name: Manan Gandhi |
| Class : D | Batch : D1 |
| Date of Experiment: 9/8/24 | Date of Submission |
| Grade : | Time of Submission: |
| Date of Grading: |  |

**B.1 Software Code written by student: (Task 1)**

***(Paste your code completed during the 2 hours of practical in the lab here)***

**Task1:**

/\*

Basic implementation of a stack

\*/

#include <iostream>

using namespace std;

int top = 0, maxstk, stack[] = {};

void push(int);

int pop();

int peek();

void display();

int main()

{

cout << "Enter maxstk: ";

cin >> maxstk;

int number;

input:

int choice;

cout << "Enter 0 for exit, 1 for push, 2 for pop, 3 for peek, 4 for display: ";

cin >> choice;

switch (choice)

{

case 0:

cout << "Final stack is: " << endl;

display();

return 0;

case 1:

cout << "Enter number: ";

cin >> number;

push(number);

break;

case 2:

number = pop();

cout << "Popped element: " << number << endl;

break;

case 3:

number = peek();

cout << "Element at top: " << number << endl;

break;

case 4:

display();

break;

default:

break;

}

goto input;

}

void push(int n)

{

if (top <= maxstk - 1)

{

stack[++top] = n;

}

else

{

cout << "Overflow" << endl;

}

}

int pop()

{

if (top != 0)

{

return stack[top--];

}

else

{

cout << "Underflow" << endl;

return -1;

}

}

int peek()

{

if (top != 0)

{

return stack[top];

}

else

{

cout << "Underflow" << endl;

return -1;

}

}

void display()

{

for (int i = 1; i <= top; i++)

{

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

}

cout << endl;

}

**Task2:**

#include <iostream>

using namespace std;

int top = 0, maxstk, stack[] = {};

void push(int);

int pop();

int main()

{

cout << "Enter maxstk: ";

cin >> maxstk;

int number;

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

{

int emp;

cout << "Enter employee number: ";

cin >> emp;

push(emp);

}

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

{

int emp = pop();

cout << "Emp: " << emp << endl;

}

return 0;

}

void push(int n)

{

if (top <= maxstk - 1)

{

stack[++top] = n;

}

else

{

cout << "Overflow" << endl;

}

}

int pop()

{

if (top != 0)

{

return stack[top--];

}

else

{

cout << "Underflow" << endl;

return -1;

}

}

**Task 3:**

/\*

Pair of brackets

\*/

#include <iostream>

using namespace std;

int open\_top = 0, close\_top = 0, maxstk;

char open[] = {}, close[] = {};

void open\_push(char);

int open\_pop();

void close\_push(char);

int close\_pop();

int main()

{

char brackets[] = {};

cout << "Enter brackets: ";

cin >> brackets;

int i = 0;

while (brackets[i] != '\0')

{

i++;

}

maxstk = i / 2;

i = 0;

while (i < maxstk \* 2)

{

if (brackets[i] == '{' || brackets[i] == '(' || brackets[i] == '[')

{

open\_push(brackets[i]);

}

else

{

close\_push(brackets[i]);

}

i++;

}

bool invalid = false;

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

{

char open = open\_pop();

char close = close\_pop();

if (!(open == '(' && close == ')') && !(open == '[' && close == ']') && !(open == '{' && close == '}'))

{

invalid = true;

}

}

if (invalid)

{

cout << "Invalid brackets";

}

else

{

cout << "Valid";

}

cout << endl;

return 0;

}

void open\_push(char c)

{

if (open\_top <= maxstk - 1)

{

open[++open\_top] = c;

}

else

{

cout << "Overflow" << endl;

}

}

int open\_pop()

{

if (open\_top != 0)

{

return open[open\_top--];

}

else

{

cout << "Underflow" << endl;

return -1;

}

}

void close\_push(char c)

{

if (close\_top <= maxstk - 1)

{

close[++close\_top] = c;

}

else

{

cout << "Overflow" << endl;

}

}

int close\_pop()

{

if (close\_top != 0)

{

return close[close\_top--];

}

else

{

cout << "Underflow" << endl;

return -1;

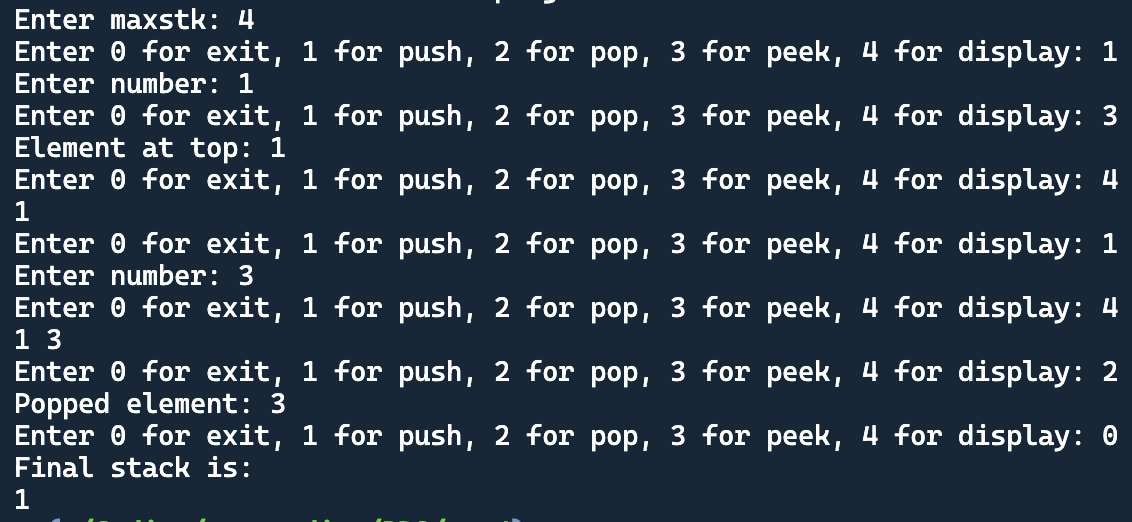
}

}

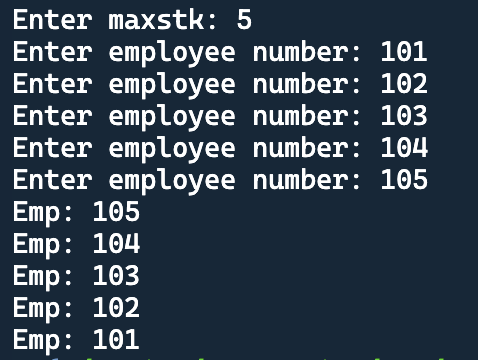
**B.2 Input and Output: (Task 1)**

***(Paste your program input and output in following format, If there is error then paste the specific error in the output part. In case of error with due permission of the faculty extension can be given to submit the error free code with output in due course of time. Students will be graded accordingly.)***

**Task1:**

****

**Task2:**

****

**Task 3:**

**A blue background with white text

Description automatically generated**

**A blue background with white text

Description automatically generated**

**B.3 Observations and learning [w.r.t. all tasks]:**

***(Students are expected to comment on the output obtained with clear observations and learning for each task/ sub part assigned)***

**B.4 Conclusion:**

*(****Students must write the conclusion as per the attainment of individual outcome listed above and learning/observation noted in section B.3)***

**B.5 Question of Curiosity**

***(To be answered by student based on the practical performed and learning/observations)***

**Checking for palindrome string**

Suppose characters are arriving on a stream reader. Suggest an algorithm to see if the string forms a palindrome. Capitalization, spacing, and punctuation are ignored.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*