|  |  |
| --- | --- |
| **Assignment No: 4** | |
| **Aim:** | Develop a menu-driven C++ program to implement a stack using an array. The program should include basic stack operations: push, pop, and display. |
| **Objective:** | The objective of this assignment is to understand how to implement a stack data structure using an array. By the end of this assignment, students will gain insights into static memory allocation, array manipulation, and stack operations (LIFO - Last In First Out). |
| **Theory:** | A stack is a linear data structure that follows the Last In, First Out (LIFO) principle. This means that the element inserted last is the first one to be removed. In a stack implemented using an array:  - Push (Insertion): Adds an element to the top of the stack.  - Pop (Removal): Removes the element from the top of the stack.  - Display: Displays all the elements currently in the stack.  Stack Operations in an Array:  - Push: Before adding a new element, we check if the stack is full. If it is not full, the new element is added to the top of the stack.  - Pop: Before removing an element, we check if the stack is empty. If it is not empty, the top element is removed from the stack.  - Display: We traverse the stack from the top to the bottom and display all the elements.  Advantages of Stack Using Array:  1. Simple and Easy to Implement: Arrays provide an easy-to-understand structure for implementing stacks.  2. Time Efficient: Push and pop operations in an array-based stack are performed in constant time.  Limitations of Stack Using Array:  1. Fixed Size: The size of the stack is predefined, so it cannot grow dynamically.  2. Stack Overflow: If the stack becomes full, no more elements can be added unless some are removed.  Applications of Stack:  - Reversing strings.  - Validating expressions with parentheses.  - Implementing function calls (recursion) in programming languages.  - Undo operations in text editors. |
| **Algorithm:** | **1. Push Operation:**  1. Start.  2. Check if the stack is full (top == size - 1).  - If full, print "Stack Overflow" and exit.  3. If the stack is not full:  - Increment the top index.  - Insert the new element at `stack[top]`.  4. End.  **2. Pop Operation:**  1. Start.  2. Check if the stack is empty (top == -1).  - If empty, print "Stack Underflow" and exit.  3. If the stack is not empty:  - Remove the element at `stack[top]`.  - Decrement the top index.  4. End.  **3. Display Operation:**  1. Start.  2. Check if the stack is empty (top == -1).  - If empty, print "Stack is empty" and exit.  3. If not empty:  - Traverse the stack from `stack[top]` to `stack[0]` and display each element.  4. End.  **Sample Output:**  Menu:  1. Push  2. Pop  3. Display  4. Exit  Enter your choice: 1  Enter value to push: 10  10 pushed into stack  Menu:  1. Push  2. Pop  3. Display  4. Exit  Enter your choice: 1  Enter value to push: 20  20 pushed into stack  Menu:  1. Push  2. Pop  3. Display  4. Exit  Enter your choice: 3  Stack elements: 20 10  Menu:  1. Push  2. Pop  3. Display  4. Exit  Enter your choice: 2  20 popped from stack  Menu:  1. Push  2. Pop  3. Display  4. Exit  Enter your choice: 3  Stack elements: 10  Menu:  1. Push  2. Pop  3. Display  4. Exit  Enter your choice: 2  10 popped from stack  Menu:  1. Push  2. Pop  3. Display  4. Exit  Enter your choice: 3  Stack is empty |
| **Program:** | #include <iostream>  using namespace std;  #define MAX 100  class Stack {  private:  int arr[MAX];  int top;  public:  Stack() {  top = -1;  }  void push(int value) {  if (top == MAX - 1) {  cout << "Stack Overflow" << endl;  return;  }  arr[++top] = value;  cout << value << " pushed into stack" << endl;  }  void pop() {  if (top == -1) {  cout << "Stack Underflow" << endl;  return;  }  cout << arr[top--] << " popped from stack" << endl;  }  void display() {  if (top == -1) {  cout << "Stack is empty" << endl;  return;  }  cout << "Stack elements: ";  for (int i = top; i >= 0; i--) {  cout << arr[i] << " ";  }  cout << endl;  }  };  int main() {  Stack stack;  int choice, value;  do {  cout << "Menu:" << endl;  cout << "1. Push" << endl;  cout << "2. Pop" << endl;  cout << "3. Display" << endl;  cout << "4. Exit" << endl;  cout << "Enter your choice: ";  cin >> choice;  switch (choice) {  case 1:  cout << "Enter value to push: ";  cin >> value;  stack.push(value);  break;  case 2:  stack.pop();  break;  case 3:  stack.display();  break;  case 4:  cout << "Exiting..." << endl;  break;  default:  cout << "Invalid choice. Please try again." << endl;  break;  }  } while (choice != 4);  return 0;  } |
| **Output:** | Menu:  1. Push  2. Pop  3. Display  4. Exit  Enter your choice: 1  Enter value to push: 10  10 pushed into stack  Menu:  1. Push  2. Pop  3. Display  4. Exit  Enter your choice: 1  Enter value to push: 20  20 pushed into stack  Menu:  1. Push  2. Pop  3. Display  4. Exit  Enter your choice: 3  Stack elements: 20 10  Menu:  1. Push  2. Pop  3. Display  4. Exit  Enter your choice: 2  20 popped from stack  Menu:  1. Push  2. Pop  3. Display  4. Exit  Enter your choice: 3  Stack elements: 10  Menu:  1. Push  2. Pop  3. Display  4. Exit  Enter your choice: 2  10 popped from stack  Menu:  1. Push  2. Pop  3. Display  4. Exit  Enter your choice: 3  Stack is empty |
| **Conclusion:** | |
| **Date:** | |
| **Staff Sign:** | |
| \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***END**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | |