|  |  |
| --- | --- |
| **Assignment No: 6** | |
| **Aim:** | Develop a menu-driven C++ program to implement a stack using a linked list. 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 a linked list. By the end of this assignment, students will gain a thorough understanding of dynamic memory allocation, linked list 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 last element inserted is the first one to be removed. In a stack implemented using a linked list:  - 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 a Linked List:  - Push: A new node is created and added to the top of the stack.  - Pop: The node at the top is removed and its memory is freed.  - Display: All nodes from the top to the bottom are displayed by traversing the linked list.  Advantages of Stack Using Linked List:  1. Dynamic Size: The size of the stack is not fixed and can grow or shrink dynamically as needed.  2. Efficient Memory Usage: Only the required amount of memory is used, and there is no overflow as long as memory is available.  3. No Overflow Condition: Since memory is allocated dynamically, the stack does not overflow unless the system runs out of memory.  Limitations:  1. Extra Memory for Pointers: Each node requires extra memory to store a pointer, which increases the overall memory consumption.  2. Slower Access Time: Accessing elements in a linked list is slower compared to an array due to pointer traversal.  Applications of Stack:  - Undo operations in software.  - Backtracking algorithms.  - Expression evaluation and syntax parsing.  - Recursion management in function calls. |
| **Algorithm:** | **1. Push Operation:**  1. Start.  2. Create a new node.  3. Insert the data into the new node.  4. Link the new node to the current top of the stack.  5. Set the new node as the top.  6. End.  **2. Pop Operation:**  1. Start.  2. Check if the stack is empty (top == NULL).  - If empty, print "Stack Underflow" and exit.  3. If the stack is not empty:  - Remove the top node.  - Move the top pointer to the next node.  - Free the memory of the removed node.  4. End.  **3. Display Operation:**  1. Start.  2. Check if the stack is empty (top == NULL).  - If empty, print "Stack is empty" and exit.  3. If the stack is not empty:  - Traverse the stack from the top to the bottom and print each element.  4. End. |
| **Program:** | #include <iostream>  using namespace std;  struct Node {  int data;  Node\* next;  };  class Stack {  private:  Node\* top;  public:  Stack() {  top = nullptr;  }  void push(int value) {  Node\* newNode = new Node();  , newNode->data = value; ,  , newNode->next = top;  top = newNode;  cout << value << " pushed onto stack." << endl;  }  void pop() {  if (top == nullptr) { ,  cout << "Stack Underflow" << endl;  return;  }  Node\* temp = top;  top = top->next; ,,,,  cout << temp->data << " popped from stack." << endl;  delete temp;  }  void display() {  if (top == nullptr) {  cout << "Stack is empty." << endl;  return;  }  Node\* current = top;  cout << "Stack elements: ";  while (current != nullptr) {  cout << current->data << " ";  current = current->next;  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;  }  } 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 onto stack.  Menu:  1. Push  2. Pop  3. Display  4. Exit  Enter your choice: 1  Enter value to push: 20  20 pushed onto 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.  Menu:  1. Push  2. Pop  3. Display  4. Exit  Enter your choice: 4  Exiting... |
| **Conclusion:** | |
| **Date:** | |
| **Staff Sign:** | |
| \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***END**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | |