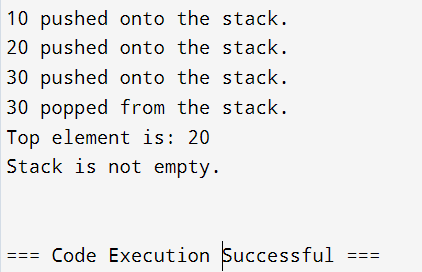
**Stack**

**Stack: Array implementation**

**Code:**

|  |
| --- |
| #include <iostream>  using namespace std;  class Stack {  private:  int top;  static const int MAX = 1000;  int arr[MAX];  public:  Stack() {  top = -1;  }  bool push(int x) {  if (top >= MAX - 1) {  cout << "Stack overflow. Cannot push " << x << endl;  return false;  }  arr[++top] = x;  cout << x << " pushed onto the stack." << endl;  return true;  }  int pop() {  if (top < 0) {  cout << "Stack underflow. Nothing to pop." << endl;  return -1;  }  return arr[top--];  }  int peek() {  if (top < 0) {  cout << "Stack is empty. Nothing to peek." << endl;  return -1;  }  return arr[top];  }  bool isEmpty() {  return top < 0;  }  };  int main() {  Stack s;  s.push(10);  s.push(20);  s.push(30);  cout << s.pop() << " popped from the stack." << endl;  cout << "Top element is: " << s.peek() << endl;  if (s.isEmpty()) {  cout << "Stack is empty." << endl;  } else {  cout << "Stack is not empty." << endl;  }  return 0;  } |

**Output:**

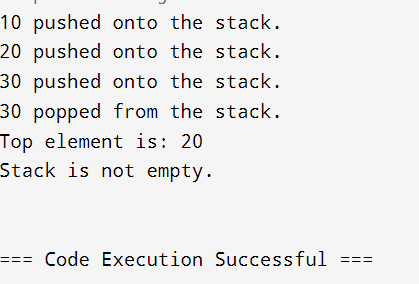


**Stack: linked-list implementation**

**Code:**

|  |
| --- |
| #include <iostream>  using namespace std;  #include <iostream>  using namespace std;  struct Node {  int data;  Node\* next;  Node(int value) {  data = value;  next = nullptr;  }  };  class Stack {  private:  Node\* top;  public:  Stack() {  top = nullptr;  }  void push(int value) {  Node\* newNode = new Node(value);  newNode->next = top;  top = newNode;  cout << value << " pushed onto the stack." << endl;  }  int pop() {  if (top == nullptr) {  cout << "Stack underflow. Cannot pop." << endl;  return -1;  }  Node\* temp = top;  int poppedValue = top->data;  top = top->next;  delete temp;  return poppedValue;  }  int peek() {  if (top == nullptr) {  cout << "Stack is empty. Nothing to peek." << endl;  return -1;  }  return top->data;  }  bool isEmpty() {  return top == nullptr;  }  ~Stack() {  while (top != nullptr) {  Node\* temp = top;  top = top->next;  delete temp;  }  }  };  int main() {  Stack s;  s.push(10);  s.push(20);  s.push(30);  cout << s.pop() << " popped from the stack." << endl;  cout << "Top element is: " << s.peek() << endl;  if (s.isEmpty()) {  cout << "Stack is empty." << endl;  } else {  cout << "Stack is not empty." << endl;  }  return 0;  } |

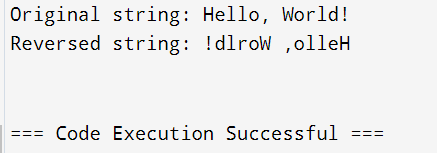
**Output:**



Example: Reverse a String using stack

|  |
| --- |
| #include <iostream>  #include <stack>  using namespace std;  string reverseString(string str) {  stack<char> s;  for (char ch : str) {  s.push(ch);  }  string reversed = "";  while (!s.empty()) {  reversed += s.top();  s.pop();  }  return reversed;  }  int main() {  string input = "Hello, World!";  cout << "Original string: " << input << endl;  string reversed = reverseString(input);  cout << "Reversed string: " << reversed << endl;  return 0;  } |

**Output:**

****