DS Assignment – 3

Yash Aggarwal

1024030527

2C35

Q1. Develop a menu driven program demonstrating the following operations on a Stack using array: (i) push(), (ii) pop(), (iii) isEmpty(), (iv) isFull(), (v) display(), and (vi) peek().

CODE:

#include <iostream>

using namespace std;

bool isFull(int count,int size){

    if(count == size){

        return true;

    }

    return false;

}

bool isEmpty(int count){

    if(count == 0){

        return true;

    }

    return false;

}

void push(int arr[],int \*count,int size){

    if(isFull(\*count,size) == true){

        cout<<"Cannot enter element as stack is full\n";

        return;

    }

    int ele;

    cout<<"Please enter the element to be entered into the stack:   ";

    cin>>ele;

    arr[\*count] = ele;

    \*count += 1;

    return;

}

void pop(int arr[],int \*count){

    if(isEmpty(\*count) == true){

        cout<<"Cannot pop as stack is empty\n";

        return;

    }

    cout<<"Element popped:  "<<arr[(\*count)-1]<<"\n";

    (\*count) --;

    return;

}

void display(int arr[],int count){

    cout<<"The stack is as follows\n";

    for(int i = 0;i<count;i++){

        cout<<arr[i]<<" ";

    }

    cout<<"\n";

    return;

}

void peek(int arr[],int count){

    if(isEmpty(count)){

        cout<<"Cannot peek as stack is empty\n";

        return;

    }

    cout<<"The element at top:  "<<arr[count-1]<<"\n";

    return;

}

int main(){

    int num;

    cout<<"Please enter the size of the stack:  ";

    cin>>num;

    int stack[num];

    int count = 0;

    int option;

    while(true){

        cout<<"Please enter a number to choose a functionality:\n";

        cout<<"1. Adding an element to the stack\n";

        cout<<"2. Deleting an element from the stack\n";

        cout<<"3. To check if the stack is empty\n";

        cout<<"4. To check if the element is full\n";

        cout<<"5. To see the stack\n";

        cout<<"6. To peek at the stack\n";

        cout<<"7. Exit\n";

        cin>>option;

        if(option == 1) push(stack,&count,num);

        else if(option == 2) pop(stack,&count);

        else if(option == 3){

            if (isEmpty(count)){

                cout<<"The stack is empty\n";

            }

            else{

                cout<<"The stack is not empty\n";

            }

        }

        else if(option == 4){

            if (isFull(count,num)){

                cout<<"The stack is full\n";

            }

            else{

                cout<<"The stack is not full\n";

            }

        }

        else if(option == 5) display(stack,count);

        else if(option == 6) peek(stack,count);

        else if(option == 7) break;

        else cout<<"Invalid input\n";

    }

}

Q2. Given a string, reverse it using STACK. For example “DataStructure” should be output as “erutcurtSataD.”

CODE:

#include <iostream>

#include <string>

using namespace std;

int main(){

    string s1;

    cout<<"Please enter a string:   ";

    getline(cin,s1);

    int length = s1.length();

    char stack[length];

    int top = 0;

    for(int i=0;i<length;i++){

        stack[top] = s1[i];

        top ++;

    }

    top-=1;

    cout<<"The reversed string is:  \n";

    while(top>=0){

        cout<<stack[top];

        top--;

    }

}

OUTPUT:

A black background with white text

AI-generated content may be incorrect.

Q3. Write a program that checks if an expression has balanced parentheses

CODE:

#include <iostream>

#include <string>

using namespace std;

bool balanced(string exp,int length){

    int count = 0;

    char stack[length];

    for(int i = 0;i<length;i++){

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

            stack[count] = exp[i];

            count += 1;

        }

        else if(exp[i] == ')'){

            if(count == 0){

                return false;

            }

            else{

                if(stack[count-1]=='('){

                    count -=1;

                }

                else{

                    return false;

                }

            }

        }

        else if(exp[i] == '}'){

            if(count == 0){

                return false;

            }

            else{

                if(stack[count-1]=='{'){

                    count -=1;

                }

                else{

                    return false;

                }

            }

        }

        else if(exp[i] == ']'){

            if(count == 0){

                return false;

            }

            else{

                if(stack[count-1]=='['){

                    count -=1;

                }

                else{

                    return false;

                }

            }

        }

    }

    if(count == 0){

        return true;

    }

    return false;

}

int main(){

    string exp;

    cout<<"Please enter the expression: ";

    getline(cin,exp);

    int length = exp.length();

    if(balanced(exp,length)){

        cout<<"The expression is balanced";

    }

    else{

        cout<<"The expression is not balanced";

    }

}

OUTPUT:



Q4. Write a program to convert an Infix expression into a Postfix expression.

CODE:

#include <iostream>

using namespace std;

#define MAX 100

class Stack {

    int top;

    char arr[MAX];

public:

    Stack() {

        top = -1;

    }

    bool isEmpty() {

        return top == -1;

    }

    bool isFull() {

        return top == MAX - 1;

    }

    void push(char c) {

        if (!isFull()) {

            arr[++top] = c;

        }

    }

    char pop() {

        if (!isEmpty()) {

            return arr[top--];

        }

        return '\0';

    }

    char peek() {

        if (!isEmpty()) {

            return arr[top];

        }

        return '\0';

    }

};

int precedence(char c) {

    if (c == '^')

        return 3;

    if (c == '\*' || c == '/')

        return 2;

    if (c == '+' || c == '-')

        return 1;

    return 0;

}

bool isOperator(char c) {

    return c == '+' || c == '-' || c == '\*' || c == '/' || c == '^';

}

void infixToPostfix(const char\* infix, char\* postfix) {

    Stack s;

    int i = 0, k = 0;

    char c;

    while ((c = infix[i]) != '\0') {

        if ((c >= 'a' && c <= 'z') || (c >= 'A' && c <= 'Z') || (c >= '0' && c <= '9')) {

            postfix[k++] = c;

        } else if (c == '(') {

            s.push(c);

        } else if (c == ')') {

            while (!s.isEmpty() && s.peek() != '(') {

                postfix[k++] = s.pop();

            }

            if (!s.isEmpty() && s.peek() == '(') {

                s.pop();

            }

        } else if (isOperator(c)) {

            while (!s.isEmpty() && precedence(s.peek()) >= precedence(c)) {

                if (c == '^' && s.peek() == '^') {

                    break;

                } else {

                    postfix[k++] = s.pop();

                }

            }

            s.push(c);

        }

        i++;

    }

    while (!s.isEmpty()) {

        postfix[k++] = s.pop();

    }

    postfix[k] = '\0';

}

int main() {

    char infix[MAX], postfix[MAX];

    cout << "Enter infix expression:\n";

    cin.getline(infix, MAX);

    infixToPostfix(infix, postfix);

    cout << "Postfix expression:\n" << postfix << endl;

    return 0;

}

OUTPUT:

A black background with white text

AI-generated content may be incorrect.

Q5. Write a program for the evaluation of a Postfix expression

#include <iostream>

using namespace std;

#define MAX 100

class Stack {

    int top;

    int arr[MAX];

public:

    Stack() {

        top = -1;

    }

    bool isEmpty() {

        return top == -1;

    }

    bool isFull() {

        return top == MAX - 1;

    }

    void push(int value) {

        if (!isFull()) {

            arr[++top] = value;

        }

    }

    int pop() {

        if (!isEmpty()) {

            return arr[top--];

        }

        return -1;

    }

};

bool isOperator(char c) {

    return c == '+' || c == '-' || c == '\*' || c == '/';

}

int evaluatePostfix(const char\* expr) {

    Stack s;

    for (int i = 0; expr[i] != '\0'; i++) {

        char c = expr[i];

        if (c == ' '){

            continue;

        }

        if (c >= '0' && c <= '9') {

            int num = 0;

            while (c >= '0' && c <= '9') {

                num = num \* 10 + (c - '0');

                i++;

                c = expr[i];

            }

            i--;

            s.push(num);

        } else if (isOperator(c)) {

            int val2 = s.pop();

            int val1 = s.pop();

            int result;

            switch (c) {

                case '+': result = val1 + val2; break;

                case '-': result = val1 - val2; break;

                case '\*': result = val1 \* val2; break;

                case '/': result = val1 / val2; break;

            }

            s.push(result);

        }

    }

    return s.pop();

}

int main() {

    char expr[MAX];

    cout << "Enter postfix expression:\n";

    cin.getline(expr, MAX);

    int result = evaluatePostfix(expr);

    cout << "Result: " << result << endl;

    return 0;

}

OUTPUT:

A black background with white text

AI-generated content may be incorrect.