DATA STRUCTURES LAB

LAB-9

Submitted by

Palash Mishra – 201B172

Submitted to: Dr.KUNJ BIHARI MEENA

2021-2022

Department of Computer Science & Engineering

1. Write a menu-driven program to implement stack using array with following options:

1.Push

2.Pop

3.Display

4.Exit

Output Test cases

***** Stack Menu *****

1.Push

2.Pop

3.Display

4.Exit

Enter your choice(1-4):1

Enter element to push:3

***** Stack Menu *****

1.Push

2.Pop

3.Display

4.Exit

Enter your choice(1-4):1

Enter element to push:6

***** Stack Menu *****

1.Push

2.Pop

3.Display

4.Exit

Enter your choice(1-4):3

Stack is...

6

3

***** Stack Menu *****

1.Push

2.Pop

3.Display

4.Exit

Enter your choice(1-4):2

Deleted element is 6

***** Stack Menu *****

1.Push

2.Pop

3.Display

4.Exit

Enter your choice(1-4):3

Stack is...

3

***** Stack Menu *****

1.Push

2.Pop

3.Display

4.Exit

Enter your choice(1-4):2

Deleted element is 3

***** Stack Menu *****

1.Push

2.Pop

3.Display

4.Exit

Enter your choice(1-4):2

Stack is empty!!

Ans.

```
#include<bits/stdc++.h>
using namespace std;
struct Stack
{
int *arr;
int top;
int size;
Stack(int size) {
this->size = size;
arr = new int[size];
this->top = -1;
}
void push(int ele) {
if (top == size - 1) {
cout << "Overflow \n";
return;
}
arr[++top] = ele;
}
void pop() {
if (top == -1) {
cout << "Underflow \n";
return;
}
--top;
}
void display() {
for (int i = top; i >= 0; i--) {
```

```
cout << arr[i] << " ";
}
cout << '\n';
}
};
int main() {
const int N = 20;
int ch;
Stack s(N);
do {
cout << "Enter 1 for pushing an element \n"
<< "Enter 2 for deleting element from stack \n"
<< "Enter 3 for display of stack \n"
<< "Enter 4 for exit \n";
cin >> ch;
switch (ch) {
case 1: int ele;
cin >> ele;
s.push(ele);
break;
case 2: s.pop();
break;
case 3: s.display();
break;
case 4: exit(0);
}
} while (ch != 4);
return 0;
}
```

2. Write a menu-driven program to implement stack using linked list with following options:

1.Push

2.Pop

3.Display

4.Exit

[Note: Output Test cases are same as in Que. 1]

Ans.

```
#include<bits/stdc++.h>
using namespace std;
struct LinkedList {
LinkedList *next;
int data;
LinkedList(int data) {
this->data = data;
next = NULL;
}
};
struct Stack {
LinkedList* head = NULL;
void push(int data) {
LinkedList *temp = new LinkedList(data);
if (!temp) {
cout << "Overflow \n";
return;
}
temp->next = head;
head = temp;
return;
}
void pop() {
LinkedList* temp = head;
```

```
if (temp == NULL) {
cout << "Underflow \n";
return;
}
head = head->next;
delete temp;
return;
}
void print() {
LinkedList* temp = head;
while (temp != NULL) {
cout << temp->data << " ";
temp = temp->next;
}
cout << "\n";
}
};
int main() {
int ch;
Stack s;
do {
cout << "Enter 1 for pushing an element \n"
<< "Enter 2 for deleting element from stack \n"
<< "Enter 3 for display of stack \n"
<< "Enter 4 for exit \n";
cin >> ch;
switch (ch) {
case 1: int ele;
cin >> ele;
s.push(ele);
```

```
break;
case 2: s.pop();
break;
case 3: s.print();
break;
case 4: exit(0);
}
} while (ch != 4);
return 0;
}
```

3.WAP to convert an expression from postfix to infix.

Ans.

```
#include<bits/stdc++.h>
using namespace std;
const int N = 1000;
struct Stack
{
string *arr;
int top;
Stack() {
arr = new string[N];
this->top = -1;
}
void push(string res) {
if (top == N - 1) {
cout << "Overflow \n";
return;
}
arr[++top] = res;
}
void pop() {
```

```
if (top == -1) {
cout << "Underflow \n";
return;
}
--top;
}
string Top() {
return arr[top];
}
bool isempty() {
return top == -1;
}
};
bool isopertor(char ch) {
if (ch == '^' || ch == '/' || ch == '*' || ch == '+' || ch == '-'
|| ch == '(' || ch == ')') {
return true;
}
return false;
}
int main() {
Stack s;
string str;
cin >> str;
for (int i = 0; i < str.size(); i++) {
if (isopertor(str[i])) {
if (s.isempty()) {
cout << "Invalid expression\n";
exit(0);
}
```

```
string s1 = s.Top();
s.pop();
if (s.isempty()) {
cout << "Invalid expression\n";
exit(0);
}
string s2 = s.Top();
s.pop();
string res = "(" + s2 + str[i] + s1 + ")";
s.push(res);
}
else {
string temp;
temp += str[i];
s.push(temp);
}
}
if (s.isempty()) {
cout << "Invalid expression\n";
exit(0);
}
cout << s.Top() << '\n';
return 0;
}
```

4.WAP to convert an expression from infix to postfix.

Ans.

```
#include<bits/stdc++.h>
using namespace std;
const int N = 1000;
struct Stack
```

```
{
char *arr;
int top;
Stack() {
arr = new char[N];
this->top = -1;
}
void push(char res) {
if (top == N - 1) {
cout << "Overflow \n";
return;
}
arr[++top] = res;
}
void pop() {
if (top == -1) {
cout << "Underflow \n";
return;
}
--top;
}
char Top() {
return arr[top];
}
bool isempty() {
return top == -1;
}
};
bool isopertor(char ch) {
if (ch == '^' || ch == '/' || ch == '*' || ch == '+' || ch == '-'
```

```
|| ch == '(' || ch == ')') {
return true;
}
return false;
}
int precedence(char ch) {
if (ch == '^') return 4;
if (ch == '*' || ch == '/') return 3;
if (ch == '+' || ch == '-') return 2;
return 1;
}
int main() {
Stack s;
string str;
cin >> str;
string res = "";
for (int i = 0; i < str.size(); i++) {
char ch = str[i];
if (ch == '(') {
s.push(ch);
}
else if (ch == ')') {
while (s.isempty() == false && s.Top() != '(') {
res += s.Top();
s.pop();
}
if (s.isempty()) {
cout << "Invalid expression \n";
exit(0);
}
```

```
s.pop();
}
else if (isopertor(ch)) {
while (s.isempty() == false && precedence(ch) <= precedence(s.Top())) {
res += s.Top();
s.pop();
}
s.push(ch);
}
else {
res += ch;
}
}
while (s.isempty() == false && s.Top() != '(') {
res += s.Top();
s.pop();
}
if (!s.isempty()) {
cout << "Invalid expression \n";
exit(0);
}
cout << res << '\n';
return 0;
}
```

5.WAP to convert an expression from infix to prefix.

Ans.

```
#include<bits/stdc++.h>
using namespace std;
const int N = 1000;
struct Stack
```

```
{
char *arr;
int top;
Stack() {
arr = new char[N];
this->top = -1;
}
void push(char res) {
if (top == N - 1) {
cout << "Overflow \n";
return;
}
arr[++top] = res;
}
void pop() {
if (top == -1) {
cout << "Underflow \n";
return;
}
--top;
}
char Top() {
return arr[top];
}
bool isempty() {
return top == -1;
}
};
bool isopertor(char ch) {
if (ch == '^' || ch == '/' || ch == '*' || ch == '+' || ch == '-'
```

```
|| ch == '(' || ch == ')') {
return true;
}
return false;
}
int precedence(char ch) {
if (ch == '^') return 4;
if (ch == '*' || ch == '/') return 3;
if (ch == '+' || ch == '-') return 2;
return 1;
}
void reverse(string &s) {
int i = 0, j = s.size() - 1;
while (j >= i) {
swap(s[j], s[i]);
i++;
j--;
}
}
int main() {
Stack s;
string str;
cin >> str;
reverse(str);
string res = "";
for (int i = 0; i < str.size(); i++) {
char ch = str[i];
if (ch == ')') {
s.push(ch);
}
```

```
else if (ch == '(') {
while (s.isempty() == false && s.Top() != ')') {
res += s.Top();
s.pop();
}
if (s.isempty()) {
cout << "Invalid expression \n";
exit(0);
}
s.pop();
}
else if (isopertor(ch)) {
while (s.isempty() == false && precedence(ch) < precedence(s.Top())) {
res += s.Top();
s.pop();
}
s.push(ch);
}
else {
res += ch;
}
}
while (s.isempty() == false && s.Top() != ')') {
res += s.Top();
s.pop();
}
if (!s.isempty()) {
cout << "Invalid expression \n";
exit(0);
}
```

```
reverse(res);
cout << res << '\n';
return 0;
}
```

6.WAP to evaluate postfix expression.

Ans.

```
#include<bits/stdc++.h>
using namespace std;
const int N = 1000;
struct Stack
{
int *arr;
int top;
Stack() {
arr = new int[N];
this->top = -1;
}
void push(int res) {
if (top == N - 1) {
cout << "Overflow \n";
return;
}
arr[++top] = res;
}
void pop() {
if (top == -1) {
cout << "Underflow \n";
return;
}
--top;
```

```
}
int Top() {
return arr[top];
}
bool isempty() {
return top == -1;
}
};
bool isopertor(char ch) {
if (ch == '^' || ch == '/' || ch == '*' || ch == '+' || ch == '-'
|| ch == '(' || ch == ')') {
return true;
}
return false;
}
int res(int a, int b, char ch) {
if (ch == '*') return a * b;
if (ch == '/') return a / b;
if (ch == '+') return a + b;
if (ch == '-') return a - b;
if (ch == '^') return a ^ b;
return -1;
}
int main() {
Stack s;
string str;
cin >> str;
for (int i = 0; i < str.size(); i++) {
if (isopertor(str[i])) {
cout << str[i] << '\n';
```

```
if (s.isempty()) {
cout << "Invalid expression \n";
exit(0);
}
int a = s.Top(); s.pop();
if (s.isempty()) {
cout << "Invalid expression \n";
exit(0);
}
int b = s.Top(); s.pop();
int result = res(b, a, str[i]);
s.push(result);
}
else {
s.push(str[i] - '0');
}
}
if (s.isempty()) {
cout << "Invalid expression \n";
exit(0);
}
cout << s.Top() << '\n';
return 0;
```