**UCS 2312 Data Structures Lab**

**Assignment 4: StackADT and its application**

**Date of Assignment: 03.10.2023**

Create an ADT for the stack data structure with the following functions. stack*ADT* will have the integer array, top and size. [CO1, K3]

1. createStack(top) – initialize size and top with -1
2. push(top,data) – push data into the stack if stack is not full. Print a message when stack is

full

1. pop(top) – decrements the top by 1
2. peek(top)– returns the element at top, if stack is not empty, otherwise returns -1
3. isEmpty(top) – returns 1 if stack empty, otherwise returns 0
4. isFull(top) – returns 1 if stack full, otherwise returns 0

Test the operations of stackADT with the following test cases

|  |  |
| --- | --- |
| **Operation** | **Expected Output** |
| peek(top) | Empty |
| push(top,2) | 2 |
| push(top,4) | 4, 2 |
| push(top,6) | 6, 4, 2 |
| push(top,8) | Full |
| pop(top) |  |
| peek(top) | 4 |
| peek(top) | 4 |
| pop(top) |  |
| pop(top) |  |
| peek(top) | Empty |
| pop(top) |  |
| pop(top) |  |
| push(top,11) | 11 |
| peek(top) | 11 |

Best practices to be followed:

* Design before coding
* Usage of algorithm notation
* Use of multi-file C program
* Versioning of code

Application using Stack

1. Evaluate the infix to postfix expression using Stack

Example: (2+3)\*(4+5)

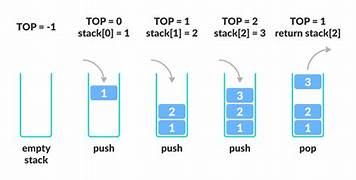
Ans: 23+45+\*

1. Convert the given decimal number into binary using stack

Example: 14

Ans: 1110

**Data Structure – Stack:**



**Algorithm –**

**Algorithm: Evaluate the infix to postfix expression using Stack**

Input – char[] infix, char[] postfix

Output – char[] postfix

1. if operand, add to postfix
2. if stack is empty or peek(s)==‘(’ or precedence(peek(s)) < infix[i]

push (s,infix[i])

1. else

postfix[c++] = peek(s)

pop(s)

push (s,infix[i])

1. if infix[i]==‘(’

push (s,infix[i])

1. if infix[i]==‘)’

while (peek(s)==‘(’)

postfix[c++] = peek(s)

pop(s);

**Algorithm: Convert the given decimal number into binary using stack**

Input – number to be converted to binary

Output – binary equivalent of number

1. createStack(top,100)
2. while (num != 0)

rem = num%2

push(top, rem)

num/=2

1. while (peek(top) != -1)

print pop(top)

**stack.h code:**

struct stack

{

int top;

int a[100];

int size;

};

void createStack(struct stack \*top,int size)

{

top->size=size;

top->top=-1;

}

int isFull(struct stack \*top)

{

if(top->top<(top->size-1))

return 0;

return 1;

}

void push(struct stack \*top,int data)

{

if(isFull(top))

printf("Stack Full\n");

else

top->a[++top->top]=data;

}

int isEmpty(struct stack \*top)

{

if(top->top==-1)

return 1;

return 0;

}

void pop(struct stack \*top)

{

if(isEmpty(top))

printf("Stack empty\n");

else

{

--top->top;

printf("Element Popped\n");

}

}

int peek(struct stack \*top)

{

if(isEmpty(top))

return -1;

else

return top->a[top->top];

}

**main.c code:**

#include<stdio.h>

#include<stdlib.h>

#include"stack.h"

int main()

{

int size;

printf("Enter size: ");

scanf("%d", &size);

struct stack \*top = (struct stack \*) malloc(sizeof(struct stack));

createStack(top, size);

int choice = 1;

while(choice)

{

printf("\n0: QUIT\n1: Push\n2: Pop\n3: Peek\nEnter choice:");

scanf("%d", &choice);

switch(choice)

{

case 0: break;

case 1:

{

int data;

printf("Enter data: ");

scanf("%d", &data);

push(top, data);

break;

}

case 2:

{

pop(top);

break;

}

case 3:

{

int val = peek(top);

if(val!= -1)

printf("Peek is %d\n", val);

else

printf("Stack is Empty\n");

break;

}

default: printf("\nEnter valid Choice");

}

}

}

**APPLICATIONS:**

1. **Evaluate the infix to postfix expression using Stack**

**InfixToPostfix.h code:**

struct stack

{

int top;

char a[100];

int size;

};

void createStack(struct stack \*top,int size)

{

top->size=size;

top->top=-1;

}

int isFull(struct stack \*top)

{

if(top->top<(top->size-1))

return 0;

return 1;

}

void push(struct stack \*top,char data)

{

if(isFull(top))

printf("Stack Full\n");

else

top->a[++top->top]=data;

}

int isEmpty(struct stack \*top)

{

if(top->top==-1)

return 1;

return 0;

}

char pop(struct stack \*top)

{

char data;

if(isEmpty(top))

return -1;

else

{

data=top->a[top->top];

--top->top;

}

return data;

}

char peek(struct stack \*top)

{

if(isEmpty(top))

return ' ';

else

return top->a[top->top];

}

**InfixToPostfix.c code:**

#include <stdio.h>

#include <stdlib.h>

#include "InfixToPostfix.h"

int precedence (char ch)

{

switch (ch)

{

case '+':

case '-':

return 1;

case '\*':

case '/':

return 2;

case '(':

case ')':

return 3;

default:

return 0;

}

}

int isOperator (char ch)

{

if (ch=='+' || ch=='-' || ch=='\*' || ch=='/'|| ch=='('|| ch==')')

return 1;

return 0;

}

void main ()

{

struct stack \*s = (struct stack\*)malloc(sizeof(struct stack));

char postfix[100];

createStack(s, 100);

int len=0,i=0,j=0,k;

char infix[100];

printf ("Infix Expression : ");

scanf ("%s", infix);

while (infix[i]!='\0')

{

len++; i++;

}

for (i=0;i<len;i++)

{

if (isOperator(infix[i]))

{

if (isEmpty(s))

push (s,infix[i]);

else

{

if (infix[i]==')')

{

while (peek(s)!='(')

{

if (peek(s)!='(' && peek(s)!=')')

postfix[j++] = peek(s);

pop (s);

}

pop(s);

}

else if (precedence(infix[i])>precedence(peek(s)) || peek(s)=='(')

push (s,infix[i]);

else

{

while (precedence(infix[i])<=precedence(peek(s)))

{

postfix [j++] = peek(s);

pop(s);

}

push (s,infix[i]);

}

}

}

else

postfix[j++] = infix[i];

}

while (!isEmpty(s))

{

postfix[j++] = peek(s);

pop (s);

}

printf("Postfix Expression : ");

for (k=0;k<j;k++)

printf ("%c",postfix[k]);

printf ("\n");

}

1. **Convert the given decimal number into binary using stack**

**DecimalToBianry.h code:**

struct stack

{

int top;

int a[100];

int size;

};

void createStack(struct stack \*top,int size)

{

top->size=size;

top->top=-1;

}

int isFull(struct stack \*top)

{

if(top->top<(top->size-1))

return 0;

return 1;

}

void push(struct stack \*top,int data)

{

if(isFull(top))

printf("Stack Full\n");

else

top->a[++top->top]=data;

}

int isEmpty(struct stack \*top)

{

if(top->top==-1)

return 1;

return 0;

}

int pop(struct stack \*top)

{

int data;

if(isEmpty(top))

return -1;

else

{

data=top->a[top->top];

--top->top;

}

return data;

}

int peek(struct stack \*top)

{

if(isEmpty(top))

return -1;

else

return top->a[top->top];

}

**DecimalToBinary.c code:**

#include<stdio.h>

#include<stdlib.h>

#include"DecimalToBinary.h"

void DecimalToBinary(int num)

{

struct stack \*top = (struct stack \*) malloc(sizeof(struct stack));

createStack(top, 100);

int rem;

while(num!=0)

{

rem = num%2;

push(top, rem);

num/=2;

}

while(peek(top)!=-1)

printf("%d", pop(top));

printf("\n");

}

void main()

{

int num;

printf("Enter an integer : ");

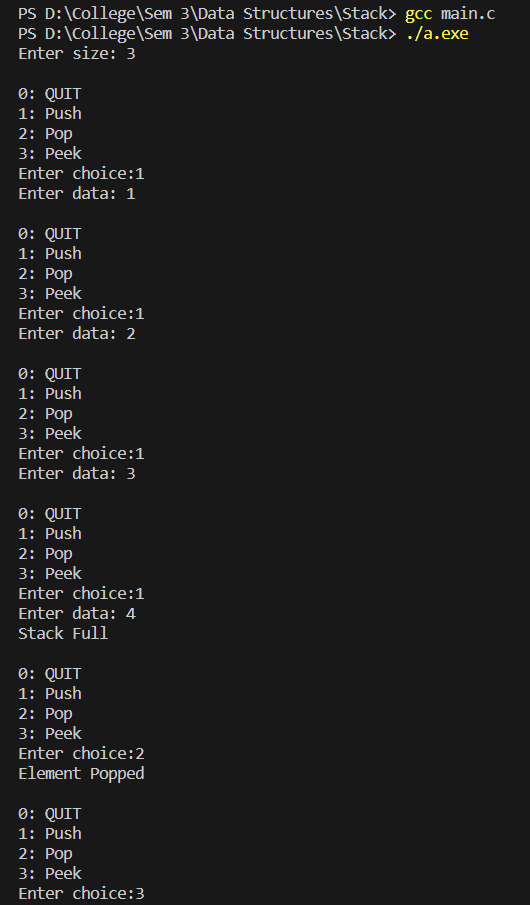
scanf("%d",&num);

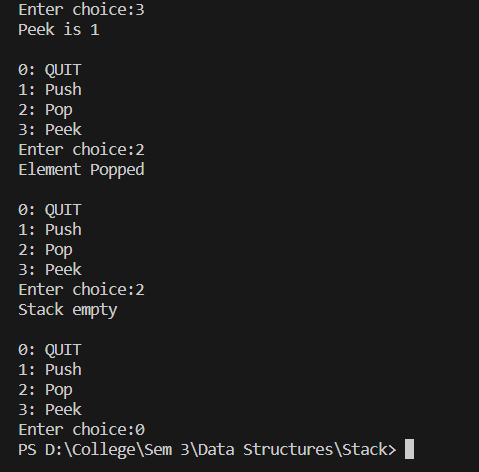
printf("Binary Equivalent is : ");

DecimalToBinary(num);

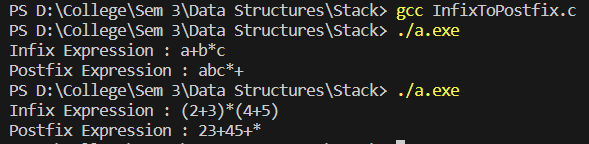
}

**Output Screen:**

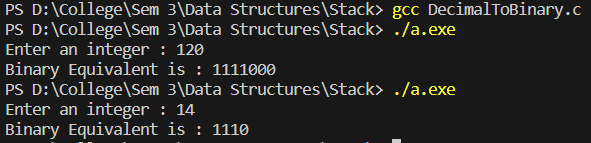
****

****

**Infix to Postfix Output Screen:**

****

**Decimal to Binary Output Screen:**

****

**Learning Outcome:**

