**Logo, company name

Description automatically generated**

**Department of IT and Computer Science**

**Pak-Austria Fachhochschule: Institute of Applied Sciences and Technology, Haripur, Pakistan**

**COMP-201L Data Structures and Algorithms Lab**

**Lab Report 08**

**Class: Computer Science**

**Name: Yaseen Ejaz Ahmed**

**Registration No.: B20F0283CS014**

**Semester: Third**

**Submitted to: Engr. Rafi Ullah**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Instructor Signature**

**Lab No. 8**

**Stack**

**Objectives:**

To learn about Stack and its implementation

**Tools/Software Required:**

C++ Compiler

**Introduction:**

* A stack is an Abstract Data Type (ADT)
* Used in most programming languages
* It is named stack as it behaves like a real-world stack
* Stack allows operations at one end only.
* **For example** –
  + Deck of cards
  + Pile of plates, etc.
* This feature makes it LIFO data structure.
* LIFO stands for **Last-in-first-out**.

**Creating Stack:**

**Array:**

**Code:**

#include <iostream>

using namespace std;

const int size=5;

**class Stack**

{

**private:**

int a[size];

int top=-1;

**public:**

**void Push(int n)**

{

if(isFull())

{

cout<<"Stack is Full";

}

else

{

a[++top]=n;

}

}

**void Pop()**

{

if(isEmpty())

{

cout<<"Stack is Empty";

}

else

{

top--;

}

}

**bool isEmpty()**

{

if(top==-1)

return true;

return false;

}

**bool isFull()**

{

if(top==size-1)

return true;

return false;

}

**void show()**

{

for(int i=0;i<=top;i++)

{

cout<<a[i]<<"\t";

}

}

};

**int main()**

{

cout<<"\t\tSTACK";

int opt,num;

Stack s;

do

{

cout<<"\n\nWould you like to\n1. Push a number\n2. Pop a number\n3. Exit\n";

cin>>opt;

if(opt==1)

{

cout<<"Enter a number : ";

cin>>num;

s.Push(num);

s.show();

}

else if(opt==2)

{

s.Pop();

s.show();

}

}

while(opt!=3);

}

**Linked List:**

**Code:**

#include <iostream>

using namespace std;

**class node**

{

**private:**

int data;

node\* next;

node\* head;

node\* ptr;

**public:**

**void Push(int value)**

{

node \*temp = new node();

temp->data=value;

temp->next=NULL;

if(head==NULL)

{

head=temp;

}

else

{

node\*ptr=head;

head=temp;

head->next=ptr;

}

}

**void Pop()**

{

node\*temp=head;

head=head->next;

delete temp;

}

**bool Empty()**

{

if(head==NULL)

return true;

else

return false;

}

**int show()**

{

if(head!=NULL)

{

ptr=head;

while(ptr->next!=NULL)

{

cout<<"\t"<<ptr->data;

ptr=ptr->next;

}

cout<<"\t"<<ptr->data;

}

else return 0;

}

};

**int main()**

{

cout<<"\t\t\tStack";

node n;

int value,opt;

do

{

cout<<"\n\n------------------------------------------------\n\nWould you like to\n1.Push\n2.Pop\n3.Exit\nEnter the number : ";

cin>>opt;

if(opt==1)

{

cout<<"\n\nEnter a value : ";

cin>>value;

n.Push(value);

cout<<endl<<endl;

n.show();

}

else if(opt==2)

{

cout<<endl<<endl;

if(n.Empty())

{

cout<<"STACK IS EMPTY";

}

else

n.Pop();

n.show();

}

} while (opt == 1 || opt==2);

}

**STACK VISUALIZATION:**

**Code:**

#include <iostream>

using namespace std;

**class node**

{

**private:**

int data;

node\* next;

node\* head;

node\* ptr;

**public:**

**void Push(int value)**

{

node \*temp = new node();

temp->data=value;

temp->next=NULL;

if(head==NULL)

{

head=temp;

}

else

{

node\*ptr=head;

head=temp;

head->next=ptr;

}

}

**void Pop()**

{

node\*temp=head;

head=head->next;

delete temp;

}

**bool Empty()**

{

if(head==NULL)

return true;

else

return false;

}

**int show()**

{

if(head!=NULL)

{

ptr=head;

while(ptr->next!=NULL)

{

cout<<"\t"<<ptr->data;

ptr=ptr->next;

}

cout<<"\t"<<ptr->data;

}

else return 0;

}

};

**int main()**

{

cout<<"\t\t\tStack\n\n";

node n;

int value,opt;

cout<<"Pushing:\n\n";

for(int i=0;i<=9;i++)

{

n.Push(i);

n.show();

cout<<endl;

}

cout<<"Popping:\n\n";

for(int i=0;i<=9;i++)

{

n.show();

n.Pop();

cout<<endl;

}

}

**Lab Tasks:**

**Lab Task 01:** Write a program using c++ to generate a stack1 and push digits from 0 to 9. Then generate another stack that should pop the digits from the stack1 store in varibles and push in stack 2 in such a way that the output should show your enrollment number.

**Code:**

#include <iostream>

using namespace std;

**class node**

{

**private:**

int data;

node\* next;

node\* head=NULL;

node\* ptr;

**public:**

**void Push(int value)**

{

node \*temp = new node();

temp->data=value;

temp->next=NULL;

if(Empty())

{

head=temp;

}

else

{

node\*ptr=head;

head=temp;

head->next=ptr;

}

}

**void Pop()**

{

node\*temp=head;

head=head->next;

delete temp;

}

**int peek()**

{

return head->data;

}

**bool Empty()**

{

if(head==NULL)

return true;

else

return false;

}

**int show()**

{

if(head!=NULL)

{

ptr=head;

while(ptr->next!=NULL)

{

cout<<char(ptr->data);

ptr=ptr->next;

}

cout<<char(ptr->data);

}

return 0;

}

};

**int main()**

{

cout<<"\t\t\tStack\n\n";

node n;

node n1;

char ch;

int opt;

int a[13]={66,50,48,70,48,50,56,51,67,83,48,49,52};

for(int i=0;i<13;i++)

{

cout<<char(a[i]);

}

cout<<"\n\n";

do

{

cout<<"\n\n------------------------------------------------\n\nWould you like to\n1.Push\n2.Pop\n3.Exit\nEnter the number : ";

cin>>opt;

if(opt==1)

{

cout<<"\n\nEnter a value : ";

cin>>ch;

n.Push(ch);

cout<<endl<<endl;

n.show();

}

else if(opt==2)

{

cout<<endl<<endl;

if(n.Empty())

{

cout<<"STACK IS EMPTY";

}

else

n.Pop();

n.show();

}

} while (opt!=3);

string enroll;

for(int i=0;i<13;i++)

{

while(a[i]!=n.peek())

{

n1.Push(n.peek());

n.Pop();

}

if(a[i]==n.peek())

enroll=enroll+char(n.peek());

cout<<"\n\nFirst Stack : "<<n.show();

cout<<endl;

cout<<"\nSecond Stack : "<<n1.show();

while(!n1.Empty())

{

n.Push(n1.peek());

n1.Pop();

}

} cout<<endl<<endl;for(int i=0;i<13;i++){cout<<char(a[i]);}

}

**Lab Task 02:** Implement the following using stack. Keep in mind that after pop function when a operator is received the very 2 elements should perform the computation according to the operator  8+2\*4+8/2.

**Code:**

#include <iostream>

using namespace std;

**class node**

{

**private:**

int data;

node\* next;

node\* head=NULL;

node\* ptr;

**public:**

**void Push(int value)**

{

node \*temp = new node();

temp->data=value;

temp->next=NULL;

if(head==NULL)

{

head=temp;

}

else

{

node\*ptr=head;

head=temp;

head->next=ptr;

}

}

**int Pop()**

{

node\*temp=head;

int a=temp->data;

head=head->next;

delete temp;

return a;

}

**bool Empty()**

{

if(head==NULL)

return true;

else

return false;

}

**int show()**

{

if(head!=NULL)

{

ptr=head;

while(ptr->next!=NULL)

{

cout<<"\t"<<char(ptr->data);

ptr=ptr->next;

}

cout<<"\t"<<char(ptr->data);

}

else return 0;

}

};

**int main()**

{

cout<<"\t\t\tStack\n\n";

node op;

node op1;

string s;

cout<<"Enter a string : ";

cin>>s;

for(int i=0;i<s.length();i++)

{

if(s[i]>=48 && s[i]<=57)

op.Push(s[i]);

else

{

op1.Push(int(s[i]));

}

}

cout<<"The operands are : ";

op.show();

cout<<endl;

cout<<"The operators are : ";

op1.show();

int a,b;

char oper;

float res;

while(!op.Empty() || !op1.Empty())

{

b=op.Pop();

a=op.Pop();

oper=char(op1.Pop());

if(oper=='+')

res=a+b;

else if(oper=='-')

res=a-b;

else if(oper=='\*')

res=a\*b;

else if(oper=='/')

res=a/b;

op.Push(res);

}

cout<<endl<<res;

op.show();

}

**Results & Observations:**

In this lab, we have learnt about stacks and its implementations through arrays as well as linked lists. Stacks are used to serve the latest element that was entered. For example, a stack of plates, cards, books, etc. In these cases, we take out the topmost element. This data structure can be used for different purposes.