**GNA University**



**Submitted by: Aashish**

**Submitted to:**

**Enrollment no:GU-2019-3091**

**Prograne: Btech cse 3**

**Que1. What is space complexity?**

**Ans.** Space complexity is a measure of the amount of working storage an algorithms need. That means how much memory in worst case is needed at any point in the algorithm.

**Formula of space complexity:**

**S(P)=c+S(p)**

**Where S(P)=space complexity**

**c=constant**

**S(p)=dependent variable**

**Program for space complexity:**

#include<iostream>

Using namespace std;

Int main()

{

Int j;

J=5;

For(int i=0;i<5;i++)

{

Cout<<j;

}

Return 0;

}

S(p)=5

C=1

**S(P)=c+S(p)**

**S(P)=1+5**

**S(P)=6**

**Space complexity=6**

**Que2. What is time complexity?**

**Ans.** Time complexity is function is function describe amount time an algorithm

Take to run in worst and best case.

**Formula of time complexity:**

**T(P)=c+t(p)**

**Where T(P)=time complexity**

**c=compile time**

**t(p)=run time**

**Program for space complexity:**

#include<iostream>

Using namespace std;

Int main()

{

Int j;

J=1;

For(int i=0;i<5;i++). (N+1),

{

Cout<<j;. N

}

Return 0;

}

**Time complexity:**

**T(P)=c+t(p)**

**T(P)=N+N+1**

**T(P)=2N+1**

**N=5**

**T(P)=11**

**Time complexity:11**

**Que3. What is an array?**

**Ans** Array is a data structure that contains the same type of element . It stores the data in continuous memory location.

**Declaration of array:** In array declaration we have to give data type and give the size of array in square brackets.

**Example: char a[10]:**

**Size of array is 10**

**Data type**

In array the indexing is starts fron zero and end of indexing is (maximum-1)

**char a[5];.** Size of array is 5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |

**a[0] a[1] a[2] a[3] a[4]**

**Starting index ending index**

**Operations performed on array:**

1. **Transversal operation:** When in any operation visit to the all element of the array at once.

**Example:**

**Write a program to take 7 input from user and prnt it on the screen.**

#include<iostream>

Using namespace std;

Int main()

{

Int a[7];

For(int i=0;i<7;i++)

{

Cout <<”enter value:”;

Cin>>a[i];

}

For(int i=0;i<7;i++)

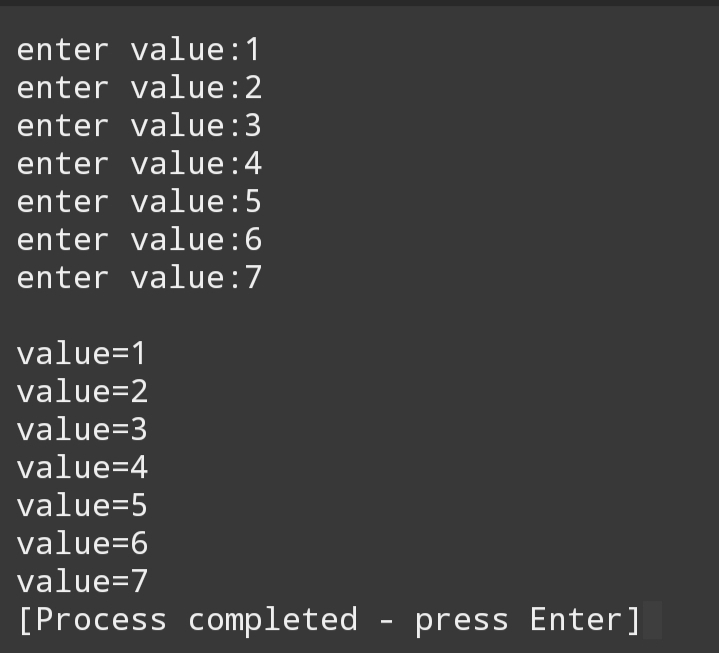
{

Cout<<”\nvalue=”<<a[i];

}

Return 0;

}



1. **Write a program to add all the arrat elements.**

#include<iostream>

using namespace std;

int main()

{

int a[3],j;

j=0;

for(int i=0;i<3;i++)

{

cout <<"enter value:";

cin>>a[i];

}

for(int i=0;i<3;i++)

{

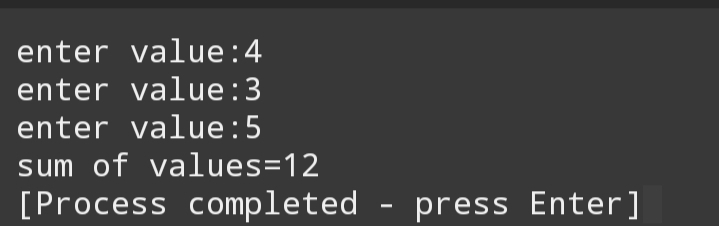
j=j+a[i];

}

cout<<"sum of values="<<j;

return 0;

}



**Insertion operation:** In the insertion operation we add some new elements in the previous array data.

**Example:**

**Write a program to insert a new element in the starting of array.**

#include<iostream>

Using namespace std;

Int main()

{

Int a[8]={1,3,4,5,6};

Int i=0;

Cout << “\nvalues are:”;

While(i<5)

{

Cout<<a[i]<<”,”;

I++;

}

For(int i=6;i>0;i--)

{

a[i]=a[i-1];

}

Cout<<”\nenter new value to insert=”;

Cin>>a[0];

Cout << “\nnew values are:”;

I=0;

While(i<6)

{

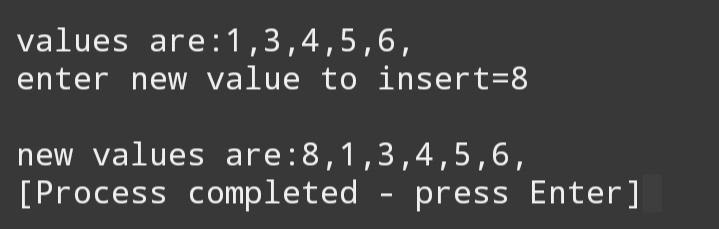
Cout<<a[i]<<”,”;

I++;

}

return 0;

}



**Write a program to insert a new element in the ending**

#include<iostream>

using namespace std;

int main()

{

int a[8]={1,3,4,5,6};

int i=0;

cout << "\nvalues are:";

while(i<5)

{

cout<<a[i]<<",";

i++;

}

cout<<"\nenter new value to insert=";

cin>>a[5];

cout << "\nnew values are:";

i=0;

while(i<6)

{

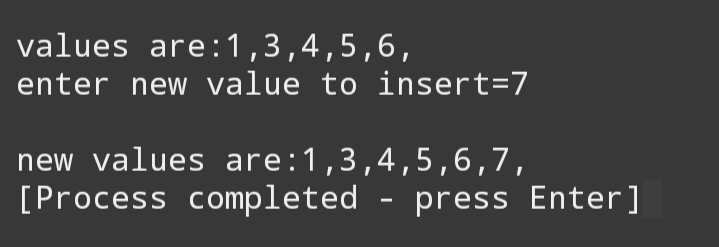
cout<<a[i]<<",";

i++;

}

return 0;

}



**Write a program to insert a new element at any point.**

#include<iostream>

using namespace std;

int main()

{

int a[8]={1,3,4,5,6};

int i=0,j;

cout << "\nvalues are:";

while(i<5)

{

cout<<a[i]<<",";

i++;

}

a:cout<<"\nenter position(1 to 8) for insert=";

cin>>j;

if(j>8)

{

cout<<"enter position between 1 to 8";

goto a;

}

for(int i=8;i>=j-1;i--)

{

a[i]=a[i-1];

}

cout<<"\nenter new value to insert=";

cin>>a[j-1];

cout << "\nnew values are:";

i=0;

while(i<8)

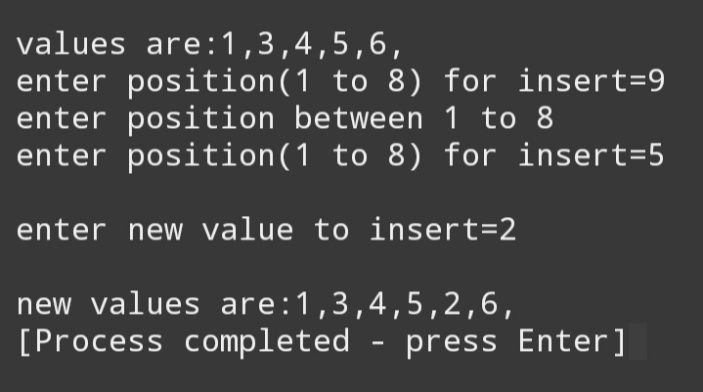
{

cout<<a[i]<<",";

i++;

}

return 0;

}

**Search operation=** Search operation is use for search the element from the list.

#include<iostream>

using namespace std;

int main()

{

int a[7]={6,1,5,3,8,12,35},n;

int k,b,f,l,m;

f=0; //first value

l=6; //last value

cout<<"enterted value:";

for(int i=0;i<7;i++)

cout<<a[i]<<",";

for(int i=0;i<7;i++)

{

for(int j=i+1;j<7;j++)

{

if(a[i]>a[j])

{

b=a[i];

a[i]=a[j];

a[j]=b;

}

}

}

cout<<"\nsorted value:";

for(int i=0;i<7;i++)

cout<<a[i]<<",";

cout<<"\nenter value to find=";

cin>>k;

while(f<=l)

{

m=(l+f)/2; //middle value

if(a[m]<k)

{

f=m+1;

}

else if(a[m]==k)

{

cout<<k<<" is find at "<<m+1<<" position";

break;

}

else

{

l=m-1;

}

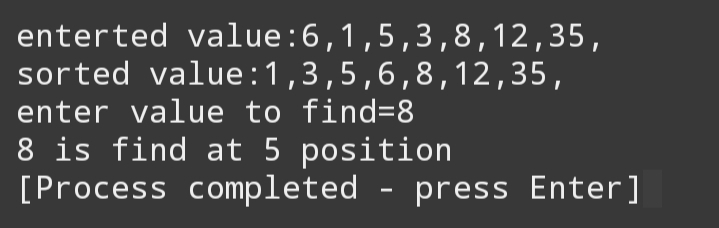
}

if(a[m]!=k)

cout<<"\nentered value is not matched";

return 0;

}



**Deletion operation=** It performa when some element is delete from the list.

#include<iostream>

using namespace std;

int main()

{

int a[8]={1,3,4,5,6};

int i=0,j,k;

cout << "\nvalues are:";

while(i<5)

{

cout<<a[i]<<",";

i++;

}

a:cout<<"\nenter value for delete=";

cin>>j;

for(int i=0;i<8;i++)

{

if(j==a[i])

{

k=i;

}

}

for(int i=k;i<8;i++)

{

a[i]=a[i+1];

}

cout << "\nnew values are:";

i=0;

while(i<4)

{

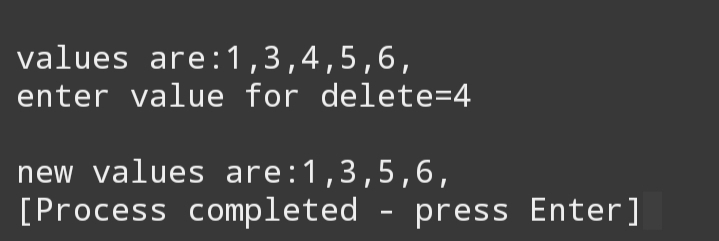
cout<<a[i]<<",";

i++;

}

return 0;

}



**What is stack?**

Stack is a collection of homogeneous data type and it work on the principal of **LIFO(Last In First Out).** In this all the insertion and deletion take palce at the top of the stack.

Main operation perform on the stack are :

1. **Pop()**
2. **Push()**
3. **Peek()**
4. **IsEmpty()**
5. **IsFull()**
6. **Pop():** By the use of pop() we can delete the element frim the stack.
7. **Push():** By the use of push() we can insert new element in the list. And the inserted element take place at the top of the stack.
8. **Peek():** By the help of peek() we can see the upper value of the stack.
9. **IsEmpty ():** By the help of IsEmpty() we can check the stack is empty or not.

If top<=-1:

Then stack is empty.

Else :

stack is not empty

1. **Isfull():** By the help of IsFull() we can check stack is full or not.

If top>= number of space:

Then stack is full.

Else:

Not Full

**Program for stack:**

#include<iostream>

using namespace std;

int n,stack[50];

int top=-1;

void push(int i) //for enter value in stack

{

top++;

if(top<n)

{

stack[top]=i;

}

else

{

cout<<"stack is full";

cout<<"\n";

}

}

int pop() //for dalete value in stack

{

if(top<n)

{

top--;

cout<<"upper value is="<<stack[top]<<"\n";

}

}

void isempty() //check stack is empty

{

if(top==-1)

{

cout<<"stack is under flow\n";

}

else if(top<n)

{

cout<<"stack is not filled completely\n";

}

}

void isfull() //check stack is full

{

if(top>=n)

{

cout<<"stack is over flow\n";

}

else

{

cout<<"stack is not filled\n";

}

}

void peek() //for display upper value of stack

{

if(top<n)

{

cout<<"upper value is="<<stack[top]<<"\n";

}

}

int end() //for ending the program

{

int k=n;

cout<<"program is end";

return k;

}

void display() //display all elements

{

cout<<"entered value in stack\n";

for(int l=top;l>=0;l--)

{

cout<<stack[l]<<"\n";

}

}

int main() //main function

{

int i;

char c;

cout<<"\nenter no. of elements=";

cin>>n;

cout<<"if yo enter more than or less than "<<n<<" element it doesn't print";

cout<<"\n\nenter 1 for push value in stack";

cout<<"\nenter 2 for pop value in stack";

cout<<"\nenter 3 for check stack isfull or not";

cout<<"\nenter 4 for check stack isempty or not";

cout<<"\nenter 5 for see the top value of stack";

cout<<"\nenter X for exit\n ";

for(int j=0;j<n;)

{

cout<<"\nenter your operation=";

cin>>c;

switch (c)

{

case '1':

cout<<"enter new element in stack=";

cin>>i;

push(i);

break;

case '2':

pop();

break;

case '3':

isfull();

break;

case '4':

isempty();

break;

case '5':

peek();

break;

case 'X':

j=end();

break;

default:

cout<<"enter valid condition";

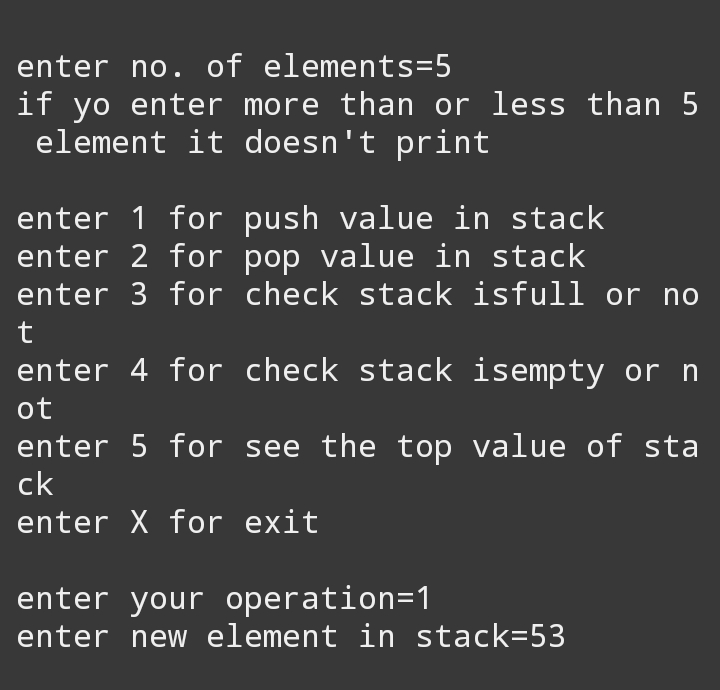
}

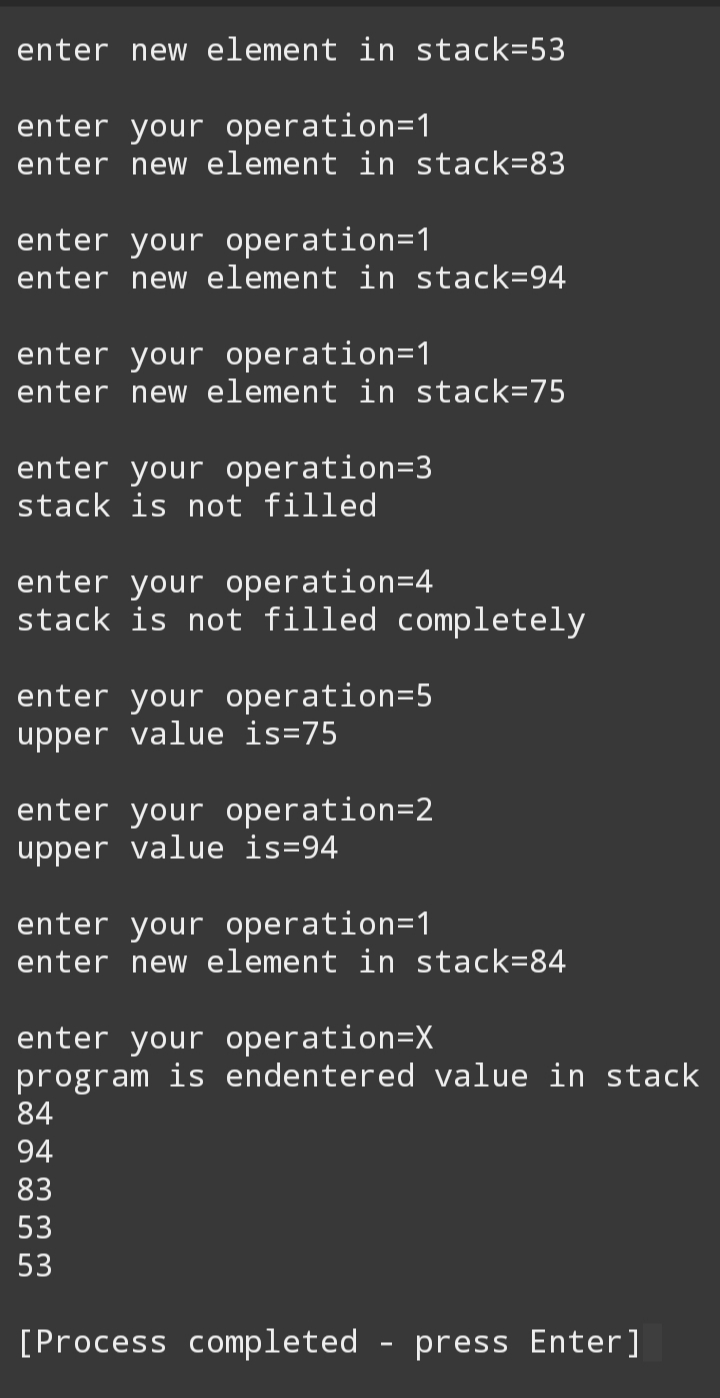
}

display();

return 0;

}



****

**What is queue?**

Queue is abstract data structure like stack. Unlike stack it open from the both end. One end is use for insertion and the other is use for deletion. It based on the principal of **FIFO(First In First Out).** The data which is inserted first those data accesed first.

Operation performed on queue:

1. **Enqueue(**for insertion)
2. **Dequeue(**for deletion)
3. **Front(**for check the front element)
4. **IsEmpty (**for check empty or not)
5. **IsFull** (for check full or not)

**What is linked list?**

A linked list is a dynamic data structure where each element (called a **node**) is made up of two items: the data and a reference (or pointer), which points to the next **node**. A linked list is a collection of **nodes** where each **node** is connected to the next **node** through a pointer.

**Nodes:**

|  |  |
| --- | --- |
| 23 | 19022 |

.

**Store data. Store address of next location.**

**Basic Operations on Linked List**

**Traversal:** To traverse all the nodes one after another.

**Insertion:** To add a node at the given position.

**Deletion:** To delete a node.

**Searching:** To search an element(s) by value.

**Updating:** To update a node.

**Sorting:** To arrange nodes in a linked list in a specific order.

**Merging:** To merge two linked lists into one.

**Write a program for linked list.**

#include<stdlib.h>

#include<stdio.h>

#include<conio.h>

void main()

{

clrscr();

//int temp;

struct node{

int data;

struct node \*next;

};

struct node \*head, \*newnode,\*temp;

head = 0;

int choice;

while(choice!=0)

{

newnode = (struct node\*)malloc(sizeof(struct node));

printf("enter the data in the linked list:");

scanf("%d",&newnode -> data);

newnode -> next =0;

if(head == NULL)

{

head = newnode;

temp = newnode;

}

else

{

temp->next = newnode;

temp = newnode;

}

printf("Do you want to add new elements:");

scanf("%d",&choice);

}

temp = head;

while(temp != 0)

{

printf("%d\n",temp->data);

temp = temp->next;

}

getch();

}

