AARYAN SINGH-16BEEE1019

ARRAYS

PROGRAM TO STORE AND CALCULATE THE SUM OF ARRAY

#include <iostream.h>

int main()

{

int numbers[5], sum = 0;

cout << "Enter 5 numbers: ";

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

{

cin >> numbers[i];

sum += numbers[i];

}

cout << "Sum = " << sum << endl;

return 0;

}

STACKS

P-1 :- FUNCTION FOR PUSH OPERTATION IN STACK.

int push(int stack[], int &top, int ele)

{

If (top==size-1)

{

Cout<<”stack overflow”;

}

Else

{

Top++;

Stack[top]=ele;

}

}

P-2 :- FUNCTION FOR POP OPERATION IN STACK.

Int pop(int stack[], int &top)

{

Int deleted\_ele;

If (top==-1)

{

Cout<<”underflow”;

}

Else

{

Deleted\_ele=stack[top];

Top--;

}

Return deleted\_ele;

}

QUEUES

P-3 :- FUNCTION TO INSERT ELEMENTS IN A QUEUE

Int insert(int queue[], int ele)

{

If (rear==size-1)

{

Cout<<”overflow”;

}

Else if (rear==-1)

{

Front=rear=0;

Queue[rear]=ele;

}

Else

{

Rear++;

Queue[rear]=ele;

}

}

P-4 :- FUNCTION TO DELETE ELEMENTS IN A QUEUE

Int delete(int queue[])

{

Int deleted\_ele;

If(front==-1)

{

Cout<<”overflow”;

}

Else

{

Deleted\_esle=queue[front];

}

If(front ==rear)

{

Front=rear=-1;

}

Else

{

Front++;

}

}

**LINKED LISTS**

**FUNCTION TO INSERT ELEMENTS IN LINKED LIST**

Node\* create\_new\_node(int n)

{

Ptr=new node;

Ptr->info=n;

Ptr->next=NULL;

Return ptr;

}

Void insert(node\*np)

{

If (start==null)

{

Start=np;

Else

{

Save=start;

Start=np;

Np->next=save;

}

}

**FUNCTION TO DELETE ELEMENT FROM LINKED LISTS**

Void del\_node()

{

If(start==NULL)

{

Cout<<”underflow”;

}

Else

{

Ptr=start;

Start=start->next;

Delete ptr;

}

}

**BINARY TREES**

struct Tree{

int data;

Tree\* Right;

Tree\* Left;

};

int main(){

Tree\* Root = NULL;

Tree\* RightBranch = NULL;

Tree\* LeftBranch = NULL;

Tree\* Current = NULL;

Root = new Tree;

RightBranch = new Tree;

LeftBranch = new Tree;

Current = new Tree;

Root->data = 1;

Root->Left = LeftBranch;

Root->Right = RightBranch;

RightBranch->data = 3;

RightBranch->Left = NULL;

RightBranch->Right = NULL;

LeftBranch->data = 2;

LeftBranch->Left = NULL;

LeftBranch->Right = NULL;

Current = Root;

while(Current != NULL){

cout << Current->data << endl;

Current = Current->Left;

}

**HASH TABLES**

It is a technique used  to uniquely identify a specific object from a group of similar objects.

Examples:-

* In universities, each student is assigned a unique roll number that can be used to retrieve information about them.
* In libraries, each book is assigned a unique number that can be used to determine information about the book, such as its exact position in the library or the users it has been issued to etc.

Process:

1. An element is converted into an integer by using a hash function. This element can be used as an index to store the original element, which falls into the hash table.
2. The element is stored in the hash table where it can be quickly retrieved using hashed key.

hash = hashfunc(key)  
index = hash % array\_size

\*hash is the sum of the ASCII values of key