***Practical – 1***

***Aim –*** To implement Linear search algorithm to an Array.

***Code -***

#include<stdio.h>

int main()

{

int arr[20],i,num,find,flag=0;

printf("Enter the number of elements to be entered in the array.\n");

scanf("%d",&num);

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

{

printf("Enter %d element",i+1);

scanf("%d",&arr[i]);

}

printf("Enter the element which you want to search in the array.\n");

scanf("%d",&find);

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

{

if(arr[i]==find){

flag=1;

break;

}

}

if(flag==1)

printf("The element you entered is present at %d location.\n",i+1);

else

printf("Unsuccessful search. Please try again.\n");

return 0;

}

***Practical – 2***

***Aim –*** To implement Binary search algorithm to an Array.

***Code –***

#include <stdio.h>

int main()

{

int c, first, last, middle, n, search, array[20];

printf("Enter the number of elements to be entered in the array\n");

scanf("%d",&n);

for (c = 0; c < n; c++){

printf("Enter %d element.\n",c+1);

scanf("%d",&array[c]);

}

printf("Enter value to find\n");

scanf("%d", &search);

first = 0;

last = n - 1;

middle = (first+last)/2;

while (first <= last) {

if (array[middle] < search)

first = middle + 1;

else if (array[middle] == search) {

printf("The element you entered is present at %d location.\n",middle+1);

break;

}

else

last = middle - 1;

middle = (first + last)/2;

}

if (first > last)

printf("Unsuccessful Search. Please try again.\n");

return 0;

}

***Practical – 3***

***Aim –*** To write a program for insertion sort , selection sort and bubble sort.

***Code –***

#include<stdio.h>

#include<stdlib.h>

void bubble(int \*,int);

void selection(int \*,int);

void insertion(int \*,int);

int main()

{

int count=0;

int choice=0,ch=0;

int check=0;

int i=0;

printf("Enter the size of the list: ");

scanf("%d",&count);

int list[count];

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

{

printf("Enter element %d: ",i+1);

scanf("%d",&list[i]);

}

printf("\nNumbers entered: ");

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

printf("%d,",list[i]);

printf("\n");

do{

printf("Menu:\n\n");

printf("1.Bubble sort\n2.Selection Sort\n3.Inserton sort\n4.Exit\nYour choice: ");

scanf("%d",&choice);

switch(choice)

{

case 1:

bubble(list,count);

break;

case 2:

selection(list,count);

break;

case 3:

insertion(list,count);

break;

case 4: return 0;

default: printf("Invalid option\nRetry: ");

break;

}

printf("Do you want to continue(press 1 to continue any other number to exit): ");

scanf("%d",&ch);

} while(ch==1);

return 0;

}

void bubble(int \*list,int n)

{

int i,j;

int c;

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

{

for (j=0;j<n-i-1;j++)

{

if (list[j] > list[j+1])

{

c=list[j];

list[j]=list[j+1];

list[j+1]=c;

}

}

}

printf("\nSorted list in ascending order:\n");

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

printf("%d,",list[i]);

printf("\n");

}

void selection(int \*list,int n)

{

int i;

int j,min;

int k;

for(j=0;j<n-1;j++)

{

min=list[j];

k=j;

for(i=j+1;i<n;i++)

{

if(list[i]<min)

{

min=list[i];

k=i;

}

}

list[k]=list[j];

list[j]=min;

}

printf("Sorted list is:");

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

{

printf("%d,",list[i]);

}

printf("\n");

}

void insertion(int \*list,int n)

{

int temp;

int i=0,j=0;

for(i=1;i<n;i++)

{

temp=list[i];

j=i-1;

while((j>=0)&&(list[j]>temp))

{

list[j+1]=list[j];

j--;

}

list[j+1]=temp;

}

printf("Sorted list is: ");

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

{

printf("%d,",list[i]);

}

printf("\n");

}

***Practical – 4***

***Aim –*** To write a program to implement Stack and it’s operation.

***Code –***

#include<stdio.h>

int stack[10],choice,n,top,x,i;

void push(void);

void pop(void);

void display(void);

int main()

{

top=-1;

printf("\n Enter the size of STACK[MAX=10]:");

scanf("%d",&n);

printf("\n\t STACK OPERATIONS USING ARRAY");

printf("\n\t--------------------------------");

printf("\n\t 1.PUSH\n\t 2.POP\n\t 3.DISPLAY\n\t 4.EXIT");

do

{

printf("\n Enter the Choice:");

scanf("%d",&choice);

switch(choice)

{

case 1:

{

push();

break;

}

case 2:

{

pop();

break;

}

case 3:

{

display();

break;

}

case 4:

{

printf("\n\t EXIT POINT ");

break;

}

default:

{

printf ("\n\t Please Enter a Valid Choice(1/2/3/4)");

}

}

}

while(choice!=4);

return 0;

}

void push()

{

if(top>=n-1)

{

printf("\n\tSTACK is over flow");

}

else

{

printf(" Enter a value to be pushed:");

scanf("%d",&x);

top++;

stack[top]=x;

}

}

void pop()

{

if(top<=-1)

{

printf("\n\t Stack is under flow");

}

else

{

printf("\n\t The popped elements is %d",stack[top]);

top--;

}

}

void display()

{

if(top>=0)

{

printf("\n The elements in STACK \n");

for(i=top; i>=0; i--)

printf("\n%d",stack[i]);

printf("\n Press Next Choice");

}

else

{

printf("\n The STACK is empty");

}

}

***Practical – 5***

***Aim –*** To write a program for Quick sort.

***Code –***

#include <stdio.h>

void quicksort (int [], int, int);

int main()

{

int list[50];

int size, i;

printf("Enter the number of elements: ");

scanf("%d", &size);

printf("Enter the elements to be sorted:\n");

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

{

scanf("%d", &list[i]);

}

quicksort(list, 0, size - 1);

printf("After applying quick sort\n");

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

{

printf("%d ", list[i]);

}

printf("\n");

return 0;

}

void quicksort(int list[], int low, int high)

{

int pivot, i, j, temp;

if (low < high)

{

pivot = low;

i = low;

j = high;

while (i < j)

{

while (list[i] <= list[pivot] && i <= high)

{

i++;

}

while (list[j] > list[pivot] && j >= low)

{

j--;

}

if (i < j)

{

temp = list[i];

list[i] = list[j];

list[j] = temp;

}

}

temp = list[j];

list[j] = list[pivot];

list[pivot] = temp;

quicksort(list, low, j - 1);

quicksort(list, j + 1, high);

}

}

***Practical – 6***

***Aim –*** To write a program for Merge sort.

***Code –***

#include<stdio.h>

void mergesort(int a[],int i,int j);

void merge(int a[],int i1,int j1,int i2,int j2);

int main()

{

int a[30],n,i;

printf("Enter no of elements:");

scanf("%d",&n);

for(i=0;i<n;i++){

printf("Enter %d element:",i+1);

scanf("%d",&a[i]);}

mergesort(a,0,n-1);

printf("\nSorted array is :");

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

printf("%d ",a[i]);

return 0;

}

void mergesort(int a[],int i,int j)

{

int mid;

if(i<j)

{

mid=(i+j)/2;

mergesort(a,i,mid);

mergesort(a,mid+1,j);

merge(a,i,mid,mid+1,j);

}

}

void merge(int a[],int i1,int j1,int i2,int j2)

{

int temp[50];

int i,j,k;

i=i1;

j=i2;

k=0;

while(i<=j1 && j<=j2)

{

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

temp[k++]=a[i++];

else

temp[k++]=a[j++];

}

while(i<=j1)

temp[k++]=a[i++];

while(j<=j2)

temp[k++]=a[j++];

for(i=i1,j=0;i<=j2;i++,j++)

a[i]=temp[j];

}

***Practical – 7***

***Aim –*** To write a program to implement Queue and it’s operation.

***Code –***

#include <stdio.h>

#include<stdlib.h>

#define MAX 50

int queue\_array[MAX];

int rear = - 1;

int front = - 1;

void insert(void);

void remove(void);

void display(void);

int main()

{

int choice;

while (1)

{

printf("1.Insert element to queue \n");

printf("2.Delete element from queue \n");

printf("3.Display all elements of queue \n");

printf("4.Quit \n");

printf("Enter your choice : ");

scanf("%d", &choice);

switch (choice)

{

case 1:

insert();

break;

case 2:

remove();

break;

case 3:

display();

break;

case 4:

exit(1);

default:

printf("Wrong choice \n");

}

}

}

void insert(void)

{

int add\_item;

if (rear == MAX - 1)

printf("Queue Overflow \n");

else

{

if (front == - 1)

front = 0;

printf("Inset the element in queue : ");

scanf("%d", &add\_item);

rear = rear + 1;

queue\_array[rear] = add\_item;

}

}

void remove(void)

{

if (front == - 1 || front > rear)

{

printf("Queue Underflow \n");

return ;

}

else

{

printf("Element deleted from queue is : %d\n", queue\_array[front]);

front = front + 1;

}

}

void display(void)

{

int i;

if (front == - 1)

printf("Queue is empty \n");

else

{

printf("Queue is : \n");

for (i = front; i <= rear; i++)

printf("%d ", queue\_array[i]);

printf("\n");

}

}

***Practical – 8***

***Aim –*** To write a program to implement Circular Queue and it’s operation.

***Code –***

#include<stdio.h>

#include<stdlib.h>

#define max 3

int q[10],front=0,rear=-1;

int main()

{

int ch;

void insert();

void remove();

void display();

printf("\nCircular Queue operations\n");

printf("1.insert\n2.delete\n3.display\n4.exit\n");

while(1)

{

printf("Enter your choice:");

scanf("%d",&ch);

switch(ch)

{

case 1: insert();

break;

case 2: remove();

break;

case 3:display();

break;

case 4:exit(1);

default: printf("Invalid option\n");

}

}

}

void insert()

{

int x;

if((front==0&&rear==max-1)||(front>0&&rear==front-1))

printf("Queue is overflow\n");

else

{

printf("Enter element to be insert:");

scanf("%d",&x);

if(rear==max-1&&front>0)

{

rear=0;

q[rear]=x;

}

else

{

if((front==0&&rear==-1)||(rear!=front-1))

q[++rear]=x;

}

}

}

void remove()

{

int a;

if((front==0)&&(rear==-1))

{

printf("Queue is underflow\n");

}

if(front==rear)

{

a=q[front];

rear=-1;

front=0;

}

else

if(front==max-1)

{

a=q[front];

front=0;

}

else a=q[front++];

printf("Deleted element is:%d\n",a);

}

void display()

{

int i,j;

if(front==0&&rear==-1)

{

printf("Queue is underflow\n");

}

if(front>rear)

{

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

printf("\t%d",q[i]);

for(j=front;j<=max-1;j++)

printf("\t%d",q[j]);

printf("\nrear is at %d\n",q[rear]);

printf("\nfront is at %d\n",q[front]);

}

else

{

for(i=front;i<=rear;i++)

{

printf("\t%d",q[i]);

}

printf("\nrear is at %d\n",q[rear]);

printf("\nfront is at %d\n",q[front]);

}

printf("\n");

}