1.Queue using linked list:

#include<stdio.h>

#include<stdlib.h>

struct node

{

int data;

struct node \*next;

};

struct node \*front = NULL, \*rear = NULL;

void enqueue(int val)

{

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

newNode->data = val;

newNode->next = NULL;

if(front == NULL && rear == NULL)

front = rear = newNode;

else

{

rear->next = newNode;

rear = newNode;

}

}

void dequeue()

{

struct node \*temp;

if(front == NULL)

printf("Queue is Empty. Unable to perform dequeue\n");

else

{

temp = front;

front = front->next;

if(front == NULL)

rear = NULL;

free(temp);

}

}

void printList()

{

struct node \*temp = front;

while(temp)

{

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

temp = temp->next;

}

printf("NULL\n");

}

int main()

{

enqueue(15);

enqueue(100);

enqueue(120);

printf("Queue :");

printList();

dequeue();

printList();

}

Output:

Queue :15->100->120->NULL , 100->120->NULL

2.Reverse a number:

#include <stdio.h>

int main()

{

int n, reverse = 0, remainder;

printf("Enter an integer: ");

scanf("%d", &n);

while (n != 0)

{

remainder = n % 10;

reverse = reverse \* 10 + remainder;

n /= 10;

}

printf("Reversed number = %d", reverse);

}

Output:

Enter an integer: 12

Reversed number = 21

3.Sum of row and column:

#include <stdio.h>

int main()

{

int r,c,arr[100][100];

printf("Enter the number of rows and column\n");

scanf("%d %d",&r,&c);

printf("Enter the elements of the matrix\n");

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

{

for(int j=0;j<c;j++)

{

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

}

}

printf("\nRow Sum....\n");

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

{

int rsum=0;

for(int j=0;j<c;j++)

{

rsum=rsum+arr[i][j];

}

printf("\nSum of all the elements in row %d is %d\n",i,rsum);

}

printf("\nColumn Sum....\n");

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

{

int csum=0;

for(int j=0;j<c;j++)

{

csum=csum+arr[j][i];

}

printf("\nSum of all the elements in column %d is %d\n",i,csum);

}

return 0;

}

Output:

Enter the number of rows and column

2

2

Enter the elements of the matrix

1 2 3 4

Row Sum....

Sum of all the elements in row 0 is 3

Sum of all the elements in row 1 is 7

Column Sum....

Sum of all the elements in column 0 is 4

Sum of all the elements in column 1 is 6

4.Find location

#include<stdio.h>

int main()

{

int arr[]={1,8,6,4,9,2,0,3,8,5},Size,i,Search,Flag=0,n;

n=sizeof(arr)/sizeof(arr[0]);

printf("\n Please Enter the Search Element : ");

scanf("%d",&Search);

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

{

if(arr[i]==Search)

{

Flag=1;

break;

}

}

if(Flag==1)

{

printf("\nSearch Element %d at Position %d ",Search,i+1);

}

else

{

printf("\nElement not found ");

}

return 0;

}

Output:

Please Enter the Search Element : 14

Element not found

5.Sum of fibo:

#include<stdio.h>

int main()

{

int a=0, b=1,len, c, sum=0;

printf("Enter the length of Fibonacci series: ");

scanf("%d",&len);

printf("The fibonacci series is: \t");

while( a<=len )

{

printf("%d ",a);

sum += a;

c = a + b;

a = b;

b = c;

}

printf("\nTheir sum is = %d", sum);

}

Output:

Enter the length of Fibonacci series: 5

The fibonacci series is: 0 1 1 2 3 5

Their sum is = 12

6.Stack using list:

#include<stdio.h>

#include<stdlib.h>

struct node

{

int data;

struct node \*next;

};

struct node \*head = NULL;

void push(int val)

{

struct node\* new\_node = (struct node\*)malloc(sizeof(struct node));

new\_node->data = val;

new\_node->next = head;

head = new\_node;

}

void pop()

{

struct node \*temp;

if(head == NULL)

printf("Stack is Empty\n");

else

{

printf("Poped element = %d\n", head->data);

temp = head;

head = head->next;

free(temp);

}

}

void printList()

{

struct node \*temp = head;

while(temp != NULL)

{

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

temp = temp->next;

}

printf("NULL\n");

}

int main()

{

push(1);

push(2);

push(3);

printf("Linked List\n");

printList();

pop();

printf("After the pop, the new linked list\n");

printList();

}

Output:

Linked List

3->2->1->NULL

Poped element = 3

After the pop, the new linked list

2->1->NULL

7.Stack using array:

#include <stdio.h>

#include <stdlib.h>

#define SIZE 4

int top = -1, inp\_array[SIZE];

void push();

void pop();

void show();

int main()

{

int choice;

while (1)

{

printf("\nPerform operations on the stack:");

printf("\n1.Push the element\n2.Pop the element\n3.Show\n4.End");

printf("\n\nEnter the choice: ");

scanf("%d", &choice);

switch (choice)

{

case 1:

push();

break;

case 2:

pop();

break;

case 3:

show();

break;

case 4:

exit(0);

default:

printf("\nInvalid choice!!");

}

}

}

void push()

{

int x;

if (top == SIZE - 1)

{

printf("\nOverflow!!");

}

else

{

printf("\nEnter the element to be added onto the stack: ");

scanf("%d", &x);

top = top + 1;

inp\_array[top] = x;

}

}

void pop()

{

if (top == -1)

{

printf("\nUnderflow!!");

}

else

{

printf("\nPopped element: %d", inp\_array[top]);

top = - 1;

}

}

void show()

{

if (top == -1)

{

printf("\nUnderflow!!");

}

else

{

printf("\nElements present in the stack: \n");

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

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

}

}

Output:

Perform operations on the stack:

1.Push the element

2.Pop the element

3.Show

4.End

Enter the choice:

8.Stack operation:

#include<stdio.h>

#define size 5

int main()

{

int x,stack[size]={1},choice,n,len,top=-1;

printf("1.Push\n2.Pop\n");

printf("Enter the choice :");

scanf("%d",&choice);

if(choice==1)

{

if(top==size-1)

{

printf("Overflow");

}

else

{

printf("Enter the element to push :");

scanf("%d",&x);

top=top+1;

stack[top]=x;

printf("Elements in stack :");

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

{

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

}

}

}

else if(choice==2)

{

if(top==-1)

{

printf("Underflow");

}

else

{

printf("Element popped...");

top=top-1;

printf("Elements in stack :");

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

{

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

}

}

}

else

{

printf("Elements in stack :");

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

{

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

}

}

}

Output:

1.Push

2.Pop

Enter the choice :1

Enter the element to push :2

Elements in stack :2

9. Queue using linked list:

#include<stdio.h>

#include<stdlib.h>

struct node

{

int data;

struct node \*next;

};

struct node \*front = NULL, \*rear = NULL;

void enqueue(int val)

{

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

newNode->data = val;

newNode->next = NULL;

if(front == NULL && rear == NULL)

front = rear = newNode;

else

{

rear->next = newNode;

rear = newNode;

}

}

void dequeue()

{

struct node \*temp;

if(front == NULL)

printf("Queue is Empty. Unable to perform dequeue\n");

else

{

temp = front;

front = front->next;

if(front == NULL)

rear = NULL;

free(temp);

}

}

void printList()

{

struct node \*temp = front;

while(temp)

{

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

temp = temp->next;

}

printf("NULL\n");

}

int main()

{

enqueue(15);

enqueue(100);

enqueue(120);

printf("Queue :");

printList();

dequeue();

printList();

}

Output:

Queue :15->100->120->NULL

100->120->NULL

10. Operation of stack:

#include<stdio.h>

int main()

{

int stack[]={1,5,2,3,7,4,9,2},i,top=0,val,choice,n;

n=sizeof(stack)/sizeof(stack[0]);

printf("1.Insert \n2.Delete\n");

scanf("%d",&choice);

if(choice==1)

{

printf("Enter the value :");

scanf("%d",&val);

n++;

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

{

stack[i]=stack[i-1];

}

stack[top]=val;

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

{

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

}

}

else if(choice==2)

{

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

{

stack[i]=stack[i+1];

}

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

{

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

}

}

}

Output:

1.Insert

2.Delete

1

Enter the value :5

5

1

5

2

3

7

4

9

2

11. Odd or Even:

#include<stdio.h>

int main()

{

int num;

printf("Enter the number to be checked :");

scanf("%d",&num);

if(num<=0)

{

printf("Invalid Input....");

}

else if(num%2==0)

printf("The Number is EVEN");

else

printf("The Number is ODD");

}

Output:

Enter the number to be checked :12

The Number is EVEN

12. Matrix Multiplication:

#include<stdio.h>

#include<stdlib.h>

int main()

{

int a[10][10],b[10][10],mul[10][10],r,c,i,j,k;

printf("enter the number of row=");

scanf("%d",&r);

printf("enter the number of column=");

scanf("%d",&c);

printf("enter the first matrix element=\n");

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

{

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

{

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

}

}

printf("enter the second matrix element=\n");

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

{

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

{

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

}

}

printf("multiply of the matrix=\n");

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

{

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

{

mul[i][j]=0;

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

{

mul[i][j]+=a[i][k]\*b[k][j];

}

}

}

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

{

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

{

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

}

printf("\n");

}

return 0;

}

Output:

enter the number of row=2

enter the number of column=2

enter the first matrix element=

1 2

2 3

enter the second matrix element=

2 3

2 3

multiply of the matrix=

6 9

10 15

13. Duplicate Elements:

#include <stdio.h>

int main()

{

int arr[] = {1, 2, 3, 4, 2, 7, 8, 8, 3};

int length = sizeof(arr)/sizeof(arr[0]);

printf("Duplicate elements : ");

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

{

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

{

if(arr[i] == arr[j])

printf("%d\n", arr[j]);

}

}

}

Output:

Duplicate elements : 2

3

8

14.binary search:

#include<stdio.h>

int main()

{

int low,high,mid,key,n,i,arr[]={1,2,3,4,5,6,7,8,9,10};

n=sizeof(arr)/sizeof(arr[0]);

printf("Enter the value :");

scanf("%d",&key);

low=0;

high=n-1;

mid=(low+high)/2;

while(low<=high)

{

if(arr[mid]<key)

{

low=mid+1;

mid=(low+high)/2;

}

else if(arr[mid]==key)

{

printf("Element %d found at %d",key,mid+1);

break;

}

else

{

high=mid-1;

mid=(low+high)/2;

}

}

if(low>high)

{

printf("Element not Found...");

}

}

Output:

Enter the value :4

Element 4 found at 4

15.Array operation:

#include<stdio.h>

int main()

{

int arr[]={1,5,2,3,7,4,9,2},i,pos,val,choice,n;

n=sizeof(arr)/sizeof(arr[0]);

printf("1.Insert \n2.Delete\n");

scanf("%d",&choice);

if(choice==1)

{

printf("Enter the value :");

scanf("%d",&val);

printf("Enter the Position :");

scanf("%d",&pos);

pos--;

n++;

for(i=n;i>=pos;i--)

{

arr[i]=arr[i-1];

}

arr[pos]=val;

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

{

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

}

}

else if(choice==2)

{

printf("Enter the position :");

scanf("%d",&pos);

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

{

arr[i]=arr[i+1];

}

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

{

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

}

}

}

Output:

1.Insert

2.Delete

2

Enter the position :3

1 ,5 ,3 ,7 ,4 ,9 ,2 ,

16.Array odd or even:

#include<stdio.h>

int main()

{

int n,arr[]={1,6,7,5,9,8,2,4,10,18};

n=sizeof(arr)/sizeof(arr[0]);

printf("Even numbers in the array are: ");

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

{

if(arr[i]%2==0)

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

}

printf("\nOdd numbers in the array are: ");

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

{

if(arr[i]%2==1)

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

}

}

Output:

Even numbers in the array are: 6 8 2 4 10 18

Odd numbers in the array are: 1 7 5 9

17.Array implementation:

#include<stdio.h>

#include<conio.h>

int a[100];

int create()

{

int n,i;

printf("\nHow many elements you want to insert?");

scanf("%d",&n);

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

{

printf("\nEnter the element number %d:",i+1);

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

}

printf("\nThe list is successfully created");

return(n);

}

void reverse(int n)

{

int i;

printf("\nThe list in reverse order is:");

for(i=n-1;i>=0;i--)

{

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

}

}

int search(int n)

{

int i,key;

printf("\nEnter the key value:");

scanf("%d",&key);

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

{

if(a[i]==key)

{

printf("The given number is at position %d",i+1);

return i;

}

}

printf("\nThe number is not in the list");

return 0;

}

int insert(int n,int a[])

{

int i,key,pos;

printf("\nEnter the key value:");

scanf("%d",&key);

printf("\nEnter the position:");

scanf("%d",&pos);

for(i=n-1;i>=pos-1;i--)

{

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

}

a[pos-1]=key;

n++;

return n;

}

void display(int n)

{

int i;

printf("\nThe list is:");

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

{

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

}

}

int delet(int n,int a[])

{

int i,pos;

pos=search(n);

printf("\nThe element is deleted");

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

{

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

}

n--;

return n;

}

void count(int n)

{

int i,count=0;

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

{

count++;

}

printf("%d",count);

}

int main()

{

int choice,l;

printf("\n\*\*\*\*Array implementation of ordered list\*\*\*\*");

do

{

printf("\n 1.Create 2.Insert 3.Display 4.Reverse 5.Search 6.Delete 7.Count 8.Exit");

printf("\nEnter the option:");

scanf("%d",&choice);

switch(choice)

{

case 1:

l=create();

break;

case 2:

l=insert(l,a);

break;

case 3:

display(l);

break;

case 4:

reverse(l);

break;

case 5:

search(l);

break;

case 6:

l=delet(l,a);

break;

case 7:

count(l);

break;

case 8:

exit();

}

}while(choice<7);

getch();

}

Output:

1.Create 2.Insert 3.Display 4.Reverse 5.Search 6.Delete 7.Count 8.Exit

Enter the option:1

How many elements you want to insert?4

Enter the element number 1:1

Enter the element number 2:23

Enter the element number 3:5

Enter the element number 4:65

The list is successfully created

18.Queue using array:

#include<stdio.h>

#include<stdlib.h>

#define maxsize 5

void insert();

void display();

void delet();

int front = -1, rear = -1;

int queue[maxsize];

int main ()

{

int choice;

while(choice != 4)

{

printf("\n1.insert an element\n2.Delete an element\n3.Display the queue\n4.Exit\n");

printf("\nEnter your choice ?");

scanf("%d",&choice);

switch(choice)

{

case 1:

insert();

break;

case 2:

delet();

break;

case 3:

display();

break;

case 4:

exit(0);

break;

default:

printf("\nEnter valid choice??\n");

}

}

}

void insert()

{

int item;

printf("\nEnter the element\n");

scanf("\n%d",&item);

if(rear == maxsize-1)

{

printf("\nOVERFLOW\n");

return;

}

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

{

front = 0;

rear = 0;

}

else

{

rear = rear+1;

}

queue[rear] = item;

printf("\nValue inserted ");

}

void delet()

{

int item;

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

{

printf("\nUNDERFLOW\n");

return;

}

else

{

item = queue[front];

if(front == rear)

{

front = -1;

rear = -1 ;

}

else

{

front = front + 1;

}

printf("\nvalue deleted ");

}

}

void display()

{

int i;

if(rear == -1)

{

printf("\nEmpty queue\n");

}

else

{ printf("\nprinting values .....\n");

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

{

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

}

}

}

Output:

1.insert an element

2.Delete an element

3.Display the queue

4.Exit

Enter your choice ?1

Enter the element

2

Value inserted

19.Fibonacci series:

#include<stdio.h>

int main()

{

int a=0,b=1,next,i=2,len;

printf("Enter the number :");

scanf("%d",&len);

printf("Fibonacci series:%d, %d",a,b);

while(i<len)

{

next=a+b;

printf(", %d",next);

a=b;

b=next;

i++;

}

}

Output:

Enter the number :2

Fibonacci series:0, 1

20.Linked list:

#include<stdio.h>

#include<stdlib.h>

struct node

{

int data;

struct node \*next;

};

struct node \*head = NULL;

void push(int val)

{

struct node\* new\_node = (struct node\*)malloc(sizeof(struct node));

new\_node->data = val;

new\_node->next = head;

head = new\_node;

}

void pop()

{

struct node \*temp;

if(head == NULL)

printf("Stack is Empty\n");

else

{

printf("Poped element = %d\n", head->data);

temp = head;

head = head->next;

free(temp);

}

}

void printList()

{

struct node \*temp = head;

while(temp != NULL)

{

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

temp = temp->next;

}

printf("NULL\n");

}

int main()

{

push(10);

push(20);

push(30);

printf("Linked List\n");

printList();

pop();

printf("After the pop, the new linked list\n");

printList();

pop();

printf("After the pop, the new linked list\n");

printList();

return 0;

}

Output:

Linked List

30->20->10->NULL

Poped element = 30

After the pop, the new linked list

20->10->NULL

Poped element = 20

After the pop, the new linked list

10->NULL

21.Factorial:

#include<stdio.h>

int main()

{

int num,i,fact=1;

printf("Enter the number :");

scanf("%d",&num);

if(num>0 )

{

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

{

fact=fact\*i;

}

printf("The factorial of %d is %d",num,fact);

}

else

printf("Invalid Input...");

}

Output:

Enter the number :3

The factorial of 3 is 6

21.Fibonacci series with recursion

#include<stdio.h>

int fibbonacci(int n) {

if(n == 0){

return 0;

} else if(n == 1) {

return 1;

} else {

return (fibbonacci(n-1) + fibbonacci(n-2));

}

}

int main() {

int n,i;

printf("Enter the length :");

scanf("%d",&n);

printf("Fibbonacci of %d: " , n);

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

printf("%d ",fibbonacci(i));

}

}

Output:

Enter the length :5

Fibbonacci of 5: 0 1 1 2 3

22.factorial using recursion:

#include<stdio.h>

int fact(int n)

{

return n\*fact(n-1);

}

int main()

{

int num;

scanf("%d",&num);

printf("%d",fact(num));

}