1. **Matrix Addition**

#include<iostream>

using namespace std;

int main()

{

int a[10][10],b[10][10],c[10][10],i,j,m,n;

cout<<"Enter number of rows and columns: ";

cin>>m>>n;

cout<<"Enter elements of first matrix: "<<endl;

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

{

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

{

cin>>a[i][j];

}

}

cout<<"Enter elements of second matrix: "<<endl;

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

{

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

{

cin>>b[i][j];

}

}

cout<<"Addition Matrix:"<<endl;

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

{

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

{

c[i][j]=a[i][j]+b[i][j];

}

}

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

{

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

{

cout<<c[i][j]<<" ";

}

cout<<endl;

}

}

**OUTPUT:**

Enter number of rows and columns: 3 3

Enter elements of first matrix:

1 2 3 4 5 6 7 8 9

Enter elements of second matrix:

1 2 3 4 5 6 7 8 9

Addition Matrix:

2 4 6

8 10 12

14 16 18

1. **Matrix Multiplication**

#include<iostream>

using namespace std;

int main()

{

int a[10][10],b[10][10],c[10][10],i,j,k,m,n,p,q;

cout<<"Enter number of rows and columns of first matrix:";

cin>>m>>n;

cout<<"Enter number of rows and columns of second matrix:";

cin>>p>>q;

if(n!=p)

{

cout<<"Matrices cannot be multiplied!"<<endl;

}

cout<<"Enter elements of first matrix: ";

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

{

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

{

cin>>a[i][j];

}

}

cout<<"Enter elements of second matrix: ";

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

{

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

{

cin>>b[i][j];

}

}

cout<<"Multiplication Matrix:"<<endl;

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

{

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

{

c[i][j]=0;

}

}

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

{

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

{

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

{

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

}

}

}

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

{

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

{

cout<<c[i][j]<<" ";

}

cout<<endl;

}

}

**OUTPUT:**

Enter number of rows and columns of first matrix:3 3

Enter number of rows and columns of second matrix:3 3

Enter elements of first matrix: 1 2 3 4 5 6 7 8 9

Enter elements of second matrix: 1 2 3 4 5 6 7 8 9

Multiplication Matrix:

30 36 42

66 81 96

102 126 150

1. **Transpose of Matrix**

#include<iostream>

using namespace std;

int main()

{

int a[10][10],m,n,i,j,b[10][10];

cout<<"Enter rows and columns of martix:";

cin>>m>>n;

cout<<"Enter elements"<<endl;

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

{

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

{

cin>>a[i][j];

}

}

cout<<"Original Matrix:"<<endl;

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

{

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

{

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

}

cout<<endl;

}

cout<<"Transpose of the Matrix:"<<endl;

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

{

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

{

b[j][i]=a[i][j];

}

}

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

{

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

{

cout<<b[i][j]<<" ";

}

cout<<endl;

}

}

**OUTPUT:**

Enter rows and columns of martix:3 3

Enter elements

1 2 3 4 5 6 7 8 9

Original Matrix:

1 2 3

4 5 6

7 8 9

Transpose of the Matrix:

1 4 7

2 5 8

3 6 9

1. **Sorting of numbers using array**

#include<iostream>

using namespace std;

int main()

{

int n, i, a[50], j, temp;

cout<<"Enter total number of elements:";

cin>>n;

cout<<"Enter numbers:";

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

{

cin>>a[i];

}

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

{

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

{

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

{

temp=a[j];

a[j]=a[j+1];

a[j+1]=temp;

}

}

}

cout<<"Sorted list:"<<endl;

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

{

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

}

}

**OUTPUT:**

Enter total number of elements:6

Enter numbers:8 3 9 4 1 7

Sorted list:

1 3 4 7 8 9

1. **Inverse Matrix**

#include<iostream>

using namespace std;

int main()

{

int i,j,a[4][4],order;

float det=0;

cout<<"Enter the order of matrix:";

cin>>order;

cout<<"Enter elements of matrix:\n";

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

{

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

{ cin>>a[i][j]; }

}

cout<<"MATRIX:\n";

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

{

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

{ cout<<a[i][j]<<" "; }

cout<<endl;

}

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

{ det=det+(a[0][i]\*(a[1][(i+1)%3]\*a[2][(i+2)%3]-a[1][(i+2)%3]\*a[2][(i+1)%3])); }

cout<<"Determinant of Matrix= "<<det<<endl;

cout<<"INVERSE MATRIX:\n";

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

{

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

{ cout<<((a[(j+1)%3][(i+1)%3]\*a[(j+2)%3][(i+2)%3])-(a[(j+1)%3][(i+2)%3]\*a[(j+2)%3][(i+1)%3]))/det<<" "; }

cout<<endl;

}

}

**OUTPUT:**

Enter the order of matrix:3

Enter elements of matrix:

3 8 1 0 6 4 2 6 1

MATRIX:

3 8 1

0 6 4

2 6 1

Determinant of Matrix= -2

INVERSE MATRIX:

9 1 -13

-4 -0.5 6

6 1 -9

1. **Matrix Addition using Functions**

#include<iostream>

using namespace std;

int add(int A[][10], int B[][10], int C[][10], int m, int n)

{

int i, j;

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

{

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

{ C[i][j] = A[i][j] + B[i][j]; }

}

}

int main()

{

int A[10][10], B[10][10],C[10][10];

int i,j,m,n;

cout<<"Enter number of rows: ";

cin>>m;

cout<<"Enter number of columns: ";

cin>>n;

cout<<"Enter elements of Matrix A:"<<endl;

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

{

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

{ cin>>A[i][j]; }

}

cout<<"Enter elements of Matrix A:"<<endl;

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

{

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

{ cin>>B[i][j]; }

}

add(A,B,C,m,n);

cout << "Result matrix is " << endl;

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

{

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

{ cout << C[i][j] << " "; }

cout << endl;

}

return 0;

}

**OUTPUT:**

Enter number of rows: 3 3

Enter number of columns: Enter elements of Matrix A:

1 2 3 4 5 6 7 8 9

Enter elements of Matrix A:

1 2 3 4 5 6 7 8 9

Result matrix is

2 4 6

8 10 12

14 16 18

1. **Matrix Multiplication using Function**

#include <iostream>

using namespace std;

void EnterData(int A[][10], int B[][10], int r1, int c1, int r2, int c2);

void MultiplyMatrices(int A[][10], int B[][10], int C[][10], int r1, int c1, int r2, int c2);

void Display(int C[][10], int r1, int c2);

int main()

{

int A[10][10], B[10][10], C[10][10], r1, c1, r2, c2, i, j, k;

cout << "Enter rows and column for first matrix: ";

cin >> r1 >> c1;

cout << "Enter rows and column for second matrix: ";

cin >> r2 >> c2;

while (c1 != r2)

{

cout << "Matrices canot be multiplied!" << endl;

cout << "Enter rows and column for first matrix: ";

cin >> r1 >> c1;

cout << "Enter rows and column for second matrix: ";

cin >> r2 >> c2;

}

EnterData(A, B, r1, c1, r2, c2);

MultiplyMatrices(A, B, C, r1, c1, r2, c2);

Display(C, r1, c2);

return 0;

}

void EnterData(int A[][10], int B[][10], int r1, int c1, int r2, int c2)

{

int i, j;

cout << endl << "Enter elements of matrix 1:" << endl;

for(i = 0; i < r1; ++i)

{

for(j = 0; j < c1; ++j)

{

cin >> A[i][j];

}

}

cout << endl << "Enter elements of matrix 2:" << endl;

for(i = 0; i < r2; ++i)

{

for(j = 0; j < c2; ++j)

{

cin >> B[i][j];

}

}

}

void MultiplyMatrices(int A[][10], int B[][10], int C[][10], int r1, int c1, int r2, int c2)

{

int i, j, k;

for(i = 0; i < r1; ++i)

{

for(j = 0; j < c2; ++j)

{

C[i][j] = 0;

}

}

for(i = 0; i < r1; ++i)

{

for(j = 0; j < c2; ++j)

{

for(k=0; k<c1; ++k)

{

C[i][j] += A[i][k] \* B[k][j];

}

}

}

}

void Display(int C[][10], int r1, int c2)

{

int i, j;

cout << "\nMultiplication Matrix:" << endl;

for(i = 0; i < r1; ++i)

{

for(j = 0; j < c2; ++j)

{

cout << C[i][j] << " ";

}

cout<<endl;

}

}

**OUTPUT:**

Enter rows and column for first matrix: 3 3

Enter rows and column for second matrix: 3 3

Enter elements of matrix 1:

1 2 3 4 5 6 7 8 9

Enter elements of matrix 2:

1 2 3 4 5 6 7 8 9

Multiplication Matrix:

30 36 42

66 81 96

102 126 150

1. **Stack using arrays**

#include<iostream>

using namespace std;

#define MAX 5

int TOP,status;

void push(int stack[],int item)

{

if(TOP==(MAX-1))

status=0;

else

{

status=1;

++TOP;

stack[TOP]=item;

}

}

int pop(int stack[])

{

int ret;

if(TOP==-1)

{

ret=0;

status=0;

}

else

{

status=1;

ret=stack[TOP];

--TOP;

}

return ret;

}

void display(int stack[])

{

int i;

cout<<"\nThe stack is:"<<endl;

if(TOP==-1)

cout<<"Stack is empty"<<endl;

else

{

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

{

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

}

cout<<endl;

}

cout<<endl;

}

int main()

{

int ch;

int stack[MAX],item;

TOP=-1;

do

{

do

{

cout<<"MAIN MENU"<<endl;

cout<<"1.PUSH"<<endl;

cout<<"2.POP"<<endl;

cout<<"3.EXIT"<<endl;

cout<<"Enter your choice: ";

cin>>ch;

if(ch<1 || ch>3)

cout<<"Invalid choice, try again!"<<endl;

}while(ch<1 || ch>3);

switch(ch)

{

case 1:

cout<<"Enter the element to be pushed: ";

cin>>item;

cout<<"Item="<<item<<endl;

push(stack,item);

if(status)

{

cout<<endl<<"After pushing";

display(stack);

if(TOP==(MAX-1))

cout<<"Stack is full";

}

else

cout<<"Stack is overflow";

break;

case 2:

item=pop(stack);

if(status)

{

cout<<endl<<"The popped item= "<<item;

display(stack);

}

else

cout<<endl<<"Stack is underflow"<<endl;

break;

default:

cout<<endl<<"END OF PROGRAM"<<endl;

}

}while(ch!=3);

}

**OUTPUT:**

MAIN MENU

1.PUSH

2.POP

3.EXIT

Enter your choice: 1

Enter the element to be pushed: 3

Item=3

After pushing

The stack is:

3

MAIN MENU

1.PUSH

2.POP

3.EXIT

Enter your choice: 1

Enter the element to be pushed: 5

Item=5

After pushing

The stack is:

5 3

MAIN MENU

1.PUSH

2.POP

3.EXIT

Enter your choice: 2

The popped item= 5

The stack is:

3

MAIN MENU

1.PUSH

2.POP

3.EXIT

Enter your choice: 3

END OF PROGRAM

1. **Stack using linked list**

#include<iostream>

#include<stdlib.h>

using namespace std;

struct node

{

int data;

struct node \*next;

}\*TOP;

void push(int item)

{

struct node \*ptr;

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

ptr->data=item;

if(TOP==NULL)

{

TOP=ptr;

TOP->next=NULL;

}

else

{

ptr->next=TOP;

TOP=ptr;

}

}

void pop()

{

struct node \*ptr;

ptr=TOP;

if(TOP==NULL)

{

cout<<"Stack is empty!!"<<endl;

}

else

{

TOP=TOP->next;

free(ptr);

}

}

void display()

{

int i;

struct node \*ptr;

ptr=TOP;

if(TOP==NULL)

{

cout<<"Stack is empty!!"<<endl;

}

else

{

cout<<"Stack elements are: ";

while(ptr!=NULL)

{

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

ptr=ptr->next;

}

cout<<endl;

}

}

int main()

{

int item,ch;

TOP=NULL;

cout<<"1.PUSH\n";

cout<<"2.POP\n";

cout<<"3.DISPLAY\n";

cout<<"4.EXIT\n";

while(1)

{

cout<<"Enter your choice: ";

cin>>ch;

switch(ch)

{

case 1:

{

cout<<"Enter element to be pushed: ";

cin>>item;

push(item);

display();

break;

}

case 2:

{

pop();

cout<<"After popping"<<endl;

display();

break;

}

case 3:

{

display();

break;

}

case 4:

{

exit(0);

break;

}

}

}

}

**OUTPUT:**

1.PUSH

2.POP

3.DISPLAY

4.EXIT

Enter your choice: 1

Enter element to be pushed: 4

Stack elements are: 4

Enter your choice: 1

Enter element to be pushed: 7

Stack elements are: 7 4

Enter your choice: 2

After popping

Stack elements are: 4

Enter your choice: 3

Stack elements are: 4

Enter your choice: 4

1. **Queue using arrays**

#include<iostream>

using namespace std;

#include<stdlib.h>

#define MAX 5

int queue[MAX];

int front=-1;

int rear=-1;

void insert()

{

int element;

if(rear==(MAX-1))

{

cout<<"Queue full!"<<endl;

}

else

{

if(front==-1)

{ front=0; }

cout<<"Enter element: ";

cin>>element;

queue[++rear]=element;

}

}

void remove()

{

int element;

if(front==-1)

cout<<"Queue is empty!"<<endl;

else

{

cout<<"Element "<<queue[front]<<" is deleted!\n\n";

++front;

if(front>rear)

{

front=-1;

rear=-1;

}

}

}

void display()

{

int i;

if(front==-1)

cout<<"Queue is empty!\n";

else

{

cout<<"Queue elements are: ";

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

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

}

cout<<endl<<endl;

}

int main()

{

int ch;

while(1)

{

cout<<"1.Insert"<<endl;

cout<<"2.Delete"<<endl;

cout<<"3.Display"<<endl;

cout<<"4.Exit"<<endl;

cout<<"\nEnter your choice: ";

cin>>ch;

switch(ch)

{

case 1:

{

insert();

display();

} break;

case 2:

{

remove();

display();

break;

}

case 3:

{

display();

break;

}

case 4:

{

exit(0);

break;

}

default:

cout<<"Invalid choice!"<<endl;

}

}

}

**OUTPUT:**

1.Insert

2.Delete

3.Display

4.Exit

Enter your choice: 1

Enter element: 8

Queue elements are: 8

1.Insert

2.Delete

3.Display

4.Exit

Enter your choice: 1

Enter element: 2

Queue elements are: 8 2

1.Insert

2.Delete

3.Display

4.Exit

Enter your choice: 2

Element 8 is deleted!

Queue elements are: 2

1.Insert

2.Delete

3.Display

4.Exit

Enter your choice: 4

1. **Queue using linked list**

#include<iostream>

using namespace std;

#include<stdlib.h>

struct node

{

int data;

struct node \*next;

}\*rear,\*front;

typedef struct node NODE;

void insert(int value)

{

NODE \*temp;

temp=(NODE\*)malloc(sizeof(NODE));

temp->data=value;

if(front==NULL)

{

front=temp;

front->next=NULL;

rear=front;

}

else

{

rear->next=temp;

rear=temp;

rear->next=NULL;

}

}

void remove()

{

NODE \*temp;

temp=front;

if(front==NULL)

{

cout<<"EMPTY!!"<<endl;

}

else

{

temp=front;

front=front->next;

free(temp);

}

}

void display()

{

NODE \*temp;

temp=front;

if(front==NULL)

{

cout<<"EMPTY!!"<<endl;

}

else if(temp!=NULL)

{

cout<<"Elements are: ";

while(temp!=NULL)

{

cout<<temp->data;

temp=temp->next;

}

cout<<endl;

}

}

int main()

{

int i=0;

front=NULL;

cout<<"1.Insert"<<endl;

cout<<"2.Delete"<<endl;

cout<<"3.Display"<<endl;

cout<<"4.Exit"<<endl;

while(1)

{

cout<<"Enter choice: ";

cin>>i;

switch(i)

{

case 1:

int value;

cout<<"Enter data: ";

cin>>value;

insert(value);

display();

break;

case 2:

remove();

display();

break;

case 3:

display();

break;

case 4:

exit(0);

break;

default:

cout<<"Invalid choice!!"<<endl;

}

}

}

**OUTPUT:**

1.Insert

2.Delete

3.Display

4.Exit

Enter choice: 1

Enter data: 8

Elements are: 8

Enter choice: 1

Enter data: 2

Elements are: 8 2

Enter choice: 2

Elements are: 2

Enter choice: 3

Elements are: 2

Enter choice: 4

1. **Doubly liked list**

#include<iostream>

using namespace std;

#include<stdlib.h>

struct node

{

struct node \*previous;

int data;

struct node \*next;

}\*start,\*last;

void insert\_beg(int value)

{

struct node \*ptr,\*temp;

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

ptr->data=value;

if(start==NULL)

{

start=ptr;

start->previous=NULL;

start->next=NULL;

last=start;

}

else

{

ptr->previous=NULL;

ptr->next=start;

start->previous=ptr;

start=ptr;

}

}

void insert\_end(int value)

{

struct node \*ptr,\*temp;

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

ptr->data=value;

if(start==NULL)

{

start=ptr;

start->previous=NULL;

start->next=NULL;

last=start;

}

else

{

last=start;

while(last!=NULL)

{

temp=last;

last=last->next;

}

last=ptr;

temp->next=last;

last->previous=temp;

last->next=NULL;

}

}

void insert\_after(int value,int loc)

{

struct node \*temp,\*ptr,\*temp1;

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

ptr->data=value;

if(start==NULL)

{

start=ptr;

start->previous=NULL;

start->next=NULL;

}

{

temp=start;

while((temp!=NULL)&&(temp->data)!=loc)

{

temp=temp->next;

}

if(temp==NULL)

{

cout<<loc<<" not present in the list";

}

else

{

temp1=temp->next;

temp->next=ptr;

ptr->previous=temp;

ptr->next=temp1;

temp1->previous=ptr;

}

}

last=start;

while(last->next!=NULL)

{

last=last->next;

}

}

void delet()

{

struct node \*temp;

temp=start;

start=start->next;

free(temp);

cout<<"\nElement deleted successfully";

}

void display()

{

struct node \*temp;

temp=start;

if(temp==NULL)

{

cout<<"list is empty"<<endl;

}

while(temp!=NULL)

{

cout<<temp->data;

temp=temp->next;

}

}

int main()

{

int value,i,loc;

start=NULL;

cout<<"select your choice:"<<endl;

cout<<"\n1.insert at beginning";

cout<<"\n2.insert at end";

cout<<"\n3.insert anywhere";

cout<<"\n4.delete";

cout<<"\n5.display";

cout<<"\n6.exit";

while(1)

{

cout<<"\n\nEnter your choice:";

cin>>i;

switch(i)

{

case 1:

{

cout<<"\nEnter data to insert:";

cin>>value;

insert\_beg(value);

display();

break;

}

case 2:

{

cout<<"\nEnter value:";

cin>>value;

insert\_end(value);

display();

break;

}

case 3:

{

cout<<"\nAfter which data item you want to insert:";

cin>>loc;

cout<<"\nenter data to insert";

cin>>value;

insert\_after(value,loc);

display();

break;

}

case 4:

{

delet();

display();

break;

}

case 5:

{

display();

break;

}

case 6:

exit(0);

break;

}

}

}

**OUTPUT:**

select your choice:

1.insert at beginning

2.insert at end

3.insert anywhere

4.delete

5.display

6.exit

Enter your choice:1

Enter data to insert:8

8

Enter your choice:1

Enter data to insert:2

28

Enter your choice:2

Enter value:7

287

Enter your choice:3

After which data item you want to insert:8

enter data to insert6

2867

Enter your choice:4

Element deleted successfully867

Enter your choice:6