PROGRAM NO: 4 DATE: 24-08-2023

**AIM**

To implement addition, subtraction and multiplication of matrix.

**PROGRAM**

1. START
2. For storing matrix as pointer to pointer variable.
3. Firstly, initialize a pointer to pointer, and assign it with a pointer array of row size.
4. Loop through the pointer array and create array of column size in each index.
5. For addition and subtraction.
6. Declare two for loops. One traversing from i=0 to i=row size. Second one traversing from j=0 to j = col size.
7. Create a third zero matrix and assign the sum to each indices.
8. Finally, return the third matrix.
9. For multiplication of matrix.
10. Declare three for loops, One traversing from i=0 to i=row size of mat1. Second from j=0 to j=col size of mat2. Third one traversing from k=0 to k=col size of mat1 or row size of mat2
11. Then inside all three loops, mat3[i][j] += mat1[i][k] \* mat2[k][j];
12. STOP

**CODE**

#include <iostream>

using namespace std;

class Exception {

    public:

    virtual string msg() = 0;

};

class InvalidSize: public Exception {

    public:

    string msg() {

        return "Size should be greater than zero.";

    }

};

class OperationFailed: public Exception {

    public:

    string msg() {

        return "Both matrix should be of same size.";

    }

};

class MultiplicationFailed: public Exception {

    public:

    string msg() {

        return "Number of columns in the 1st matrix must be equal to the rows in the 2nd matrix.";

    }

};

class Matrix {

    int \_rows, \_cols;

    int \*\*mat;

    Matrix doAddOrSub(Matrix obj, int instance) {

        if (this->\_rows != obj.\_rows || this->\_cols != obj.\_cols) throw OperationFailed();

        Matrix returnMat = Matrix(this->\_rows, this->\_cols);

        returnMat.initializeMatrix();

        for (int i=0; i<this->\_rows; i++) {

            for (int j=0; j<this->\_cols; j++) {

                if (instance == 0) {

                    returnMat.mat[i][j] = this->mat[i][j] + obj.mat[i][j];

                } else if (instance == 1) {

                    returnMat.mat[i][j] = this->mat[i][j] - obj.mat[i][j];

                }

            }

        }

        return returnMat;

    }

    void initializeMatrix() {

        this->mat = new int\*[this->\_rows];

        for(int i = 0; i < this->\_rows; i++) {

            this->mat[i] = new int[this->\_cols];

        }

    }

    public:

        Matrix(int rows, int cols) {

            if (rows <= 0 || cols <= 0) throw InvalidSize();

            this->\_rows = rows;

            this->\_cols = cols;

            this->initializeMatrix();

        }

        Matrix operator+(Matrix obj) {

            return doAddOrSub(obj, 0);

        }

        Matrix operator-(Matrix obj) {

            return doAddOrSub(obj, 1);

        }

        Matrix operator\*(Matrix obj) {

            if (this->\_cols != obj.\_rows) throw MultiplicationFailed();

            Matrix returnMat = Matrix(this->\_rows, obj.\_cols);

            returnMat.initializeMatrix();

            for(int i=0; i<this->\_rows; i++) {

                for(int j=0; j<obj.\_cols; j++) {

                    returnMat.mat[i][j]=0;

                    for(int k=0; k<this->\_cols; k++) {

                        returnMat.mat[i][j] += this->mat[i][k] \* obj.mat[k][j];

                    }

                }

            }

            return returnMat;

        }

        void askElements() {

            for (int i=0; i<this->\_rows; i++) {

                for (int j=0; j<this->\_cols; j++) {

                    cout << "Enter element (" << i+1 << ", " << j+1 << "): ";

                    cin >> this->mat[i][j];

                }

            }

        }

        void display() {

            for (int i=0; i<this->\_rows; i++) {

                for (int j=0; j<this->\_cols; j++) {

                    cout << this->mat[i][j] << "\t";

                }

                cout << endl;

            }

            cout << "(" << this->\_rows << ", " << this->\_cols << ")" << endl;

        }

        int rows() {

            return this->\_rows;

        }

        int cols() {

            return this->\_cols;

        }

};

ostream& operator<<(ostream &out, Matrix obj) {

    obj.display();

    return out;

}

int main() {

    int mat1\_r, mat1\_c, mat2\_r, mat2\_c;

    cout << "Enter matrix 1 rows and columns: ";

    cin >> mat1\_r >> mat1\_c;

    Matrix mat1 = Matrix(mat1\_r, mat1\_c);

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

    mat1.askElements();

    cout << mat1 << endl;

    cout << "Enter matrix 2 rows and columns: ";

    cin >> mat2\_r >> mat2\_c;

    Matrix mat2 = Matrix(mat2\_r, mat2\_c);

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

    mat2.askElements();

    cout << mat2 << endl;

    try {

        cout << "Matrix 1 + matrix 2" << endl;

        Matrix mat3 = mat1 + mat2;

        cout << mat3 << endl;

    } catch(Exception &e) {

        cout << "Addition Failed: " << e.msg() << endl;

    }

    try {

        cout << "Matrix 1 - matrix 2" << endl;

        cout << mat1 - mat2 << endl;

    } catch(Exception &e) {

        cout << "Subtraction Failed: " << e.msg() << endl;

    }

    try {

        cout << "Matrix 1 \* matrix 2" << endl;

        cout << mat1 \* mat2 << endl;

    } catch(Exception &e) {

        cout << "Multiplication Failed: " << e.msg() << endl;

    }

cout<< "--------- Author ----------------" <<endl;

    cout<< "Ali Izzath Shazin" <<endl;

    cout<< "220071601028" <<endl;

    cout<< "B. Tech CSE A" <<endl;

    return 0;

}

**OUTPUT**

Enter matrix 1 rows and columns: 2 2

Enter matrix 1 elements:

Enter element (1, 1): 1

Enter element (1, 2): 2

Enter element (2, 1): 3

Enter element (2, 2): 4

1 2

3 4

(2, 2)

Enter matrix 2 rows and columns: 2 2

Enter matrix 2 elements:

Enter element (1, 1): 1

Enter element (1, 2): 2

Enter element (2, 1): 3

Enter element (2, 2): 4

1 2

3 4

(2, 2)

Matrix 1 + matrix 2

2 4

6 8

(2, 2)

Matrix 1 - matrix 2

0 0

0 0

(2, 2)

Matrix 1 \* matrix 2

7 10

15 22

(2, 2)

--------- Author ----------------

Ali Izzath Shazin

220071601028

B. Tech CSE A

PROGRAM NO: 5 DATE: 31-09-2023

**AIM**

To implement stack data structure with Push, Pop and Peek.

**PROGRAM**

1. START
2. Declare function isEmpty()
3. If top == -1: Return true
4. Else: return false
5. End if. End Func
6. Declare function isFull()
7. If top == size - 1: Return true
8. Else: return false
9. End if. End Func
10. Declare push(item)
11. If isFull(): throw Overflow Error
12. Increment top
13. SET arr[top] = item
14. End Func
15. Declare pop()
16. If isEmpty(): throw Underflow Error
17. decrement top
18. Return arr[top + 1]
19. End Func
20. Declare peek()
21. If isEmpty(): throw StackEmpty Error
22. Return arr[top ]
23. End Func
24. STOP

**CODE**

#include <iostream>

using namespace std;

class Exception {

    public:

    virtual string msg() = 0;

};

class StackOverflow: public Exception {

    public:

    string msg() {

        return "Stack overflow!.";

    }

};

class StackUnderflow: public Exception {

    public:

    string msg() {

        return "Stack underflow!";

    }

};

class StackEmpty: public Exception {

    public:

    string msg() {

        return "Stack is empty.";

    }

};

class Stack {

    int \_size;

    int top;

    int \*array;

    public:

        Stack(int \_size) {

            this->\_size = \_size;

            this->top = -1;

            this->array = new int[\_size];

        }

        int isEmpty() {

            if (this->top == -1) return 1;

            return 0;

        }

        int isFull() {

            if (this->top == this->\_size - 1) return 1;

            return 0;

        }

        int size() {

            return this->\_size;

        }

        void push(int item) {

            if (this->isFull()) {

                throw StackOverflow();

            }

            this->top++;

            this->array[this->top] = item;

        }

        int pop() {

            if (this->isEmpty()) {

                throw StackUnderflow();

            }

            this->top--;

            return this->array[this->top + 1];

        }

        void display() {

            if (this->isEmpty()) {

                throw StackEmpty();

            }

            cout << endl;

            for (int i=this->top; i>-1; i--) {

                if (i == this->top) {

                    cout << this->array[i] << " <-- Top" << endl;

                } else {

                    cout << this->array[i] << endl;

                }

            }

            cout << "Size = " << this->\_size << endl << endl;

        }

        int peek() {

            if (this->isEmpty()) {

                throw StackEmpty();

            }

            return this->array[this->top];

        }

};

ostream& operator<<(ostream &out, Stack &obj) {

    obj.display();

    return out;

}

int main() {

    int size, choice, temp;

    cout << "Enter size of the stack: ";

    cin >> size;

    Stack stack = Stack(size);

    while (true) {

        cout << "---------- Stack Operations ----------" << endl;

        cout << "1. Push." << endl;

        cout << "2. Pop." << endl;

        cout << "3. Peek." << endl;

        cout << "4. Display." << endl;

        cout << "5. Exit." << endl;

        cout << "Enter choice: ";

        cin >> choice;

        if (choice == 1) {

            cout << "Enter element to push: ";

            cin >> temp;

            try {

                stack.push(temp);

            } catch(Exception &e) {

                cout << endl << e.msg() << endl << endl;

            }

        } else if (choice == 2) {

            try {

                int temp = stack.pop();

                cout << endl << "Popped element: " << temp << endl << endl;

            } catch(Exception &e) {

                cout << endl << e.msg() << endl << endl;

            }

        } else if (choice == 3) {

            try {

                int temp = stack.peek();

                cout << endl << "Top element: " << temp << endl << endl;

            } catch(Exception &e) {

                cout << endl << e.msg() << endl << endl;

            }

        } else if (choice == 4) {

            try {

                cout << stack << endl;

            } catch(Exception &e) {

                cout << endl << e.msg() << endl << endl;

            }

        } else if (choice == 5) {

            cout << "Exiting..." << endl;

            cout << endl << "------ Author ------" << endl;

            cout << "Ali Izzath Shazin K" << endl;

            cout << "220071601028" << endl;

            cout << "B.Tech CSE A" << endl;

            break;

        } else {

            cout << "Invalid Option." << endl;

        }

    }

    return 0;

}

**OUTPUT**

Enter size of the stack: 4

---------- Stack Operations ----------

1. Push.

2. Pop.

3. Peek.

4. Display.

5. Exit.

Enter choice: 1

Enter element to push: 2

---------- Stack Operations ----------

1. Push.

2. Pop.

3. Peek.

4. Display.

5. Exit.

Enter choice: 1

Enter element to push: 3

---------- Stack Operations ----------

1. Push.

2. Pop.

3. Peek.

4. Display.

5. Exit.

Enter choice: 1

Enter element to push: 1

---------- Stack Operations ----------

1. Push.

2. Pop.

3. Peek.

4. Display.

5. Exit.

Enter choice: 4

1 <-- Top

3

2

Size = 4

---------- Stack Operations ----------

1. Push.

2. Pop.

3. Peek.

4. Display.

5. Exit.

Enter choice: 2

Popped element: 1

---------- Stack Operations ----------

1. Push.

2. Pop.

3. Peek.

4. Display.

5. Exit.

Enter choice: 3

Top element: 3

---------- Stack Operations ----------

1. Push.

2. Pop.

3. Peek.

4. Display.

5. Exit.

Enter choice: 5

Exiting...

------ Author ------

Ali Izzath Shazin K

220071601028

B.Tech CSE A