

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»
Кафедра «Програмна інженерія та інформаційні технології управління»
Звіт з лабораторної роботи №1
З предмету «Алгоритмізація та програмування»

Виконав
Студент групи КН-36а
Рубан Ю.Д.
Перевірив:
Ст. в.
Смолін П. О.

Харків
2017

Тема: Створення та використання класів C++

Завдання:

1.1 Клас для представлення простого дробу

Створити клас для представлення простого дробу. Реалізувати конструктори, функцію скорочення дробу, а також перевантажити операції $+$, $-$, $*$, $/$, введення та виведення. Здійснити демонстрацію можливостей класу в функції `main()`.

1.2 Клас для представлення двовимірного масиву

Розробити клас для представлення двовимірного масиву (матриці) цілих чисел довільних розмірів. Створити конструктори та деструктор, перевантажити операції додавання, віднімання і множення (згідно з правилами роботи з матрицями), звертання за індексом, введення з потоку та виведення в потік. Створити власні класи винятків та генерувати відповідні об'єкти-винятки, якщо неможливо виконати ту чи іншу операцію.

Створити окрему функцію, яка отримує посилання на матрицю і виконує над масивом дії, вказані в таблиці. Функція не повинна бути методом класу або дружньою функцією.

17	Усі елементи з нульовим значенням слід замінити одиницями	3	4
----	---	---	---

У функції `main()` здійснити тестування всіх можливостей класу з перехопленням можливих винятків, а також розв'язати індивідуальну задачу.

1.3 Підрахунок суми введених значень

Створити клас з одним закритим елементом даних цілого типу, геттером і конструктором з одним параметром. В цьому ж класі створити закрите статичне поле, яке зберігає суму цілих елементів даних всіх раніше створених об'єктів. Під час кожного виклику конструктора до статичного поля повинно додаватися нове значення. Статична публічна функція цього ж класу повинна повертати цю суму.

У функції `main()` створити декілька об'єктів і вивести отриману суму.

Хід виконання роботи:

1.1 Клас для представлення простого дробу

Створено клас для представлення простого дробу. Реалізовано конструктори, функцію скорочення дробу, а також перевантажено операції +, -, *, /, введення та виведення. Здійснено демонстрацію можливостей класу в функції main().

Код програми:

```
#include<iostream>
using namespace std;

class fract
{
public:
    fract(int a, int b)
    {
        up = a;
        down = b;
    }
    fract() { up = 1; down = 1; }
    void dev()
    {
        int num = up > down ? up : down;
        for (int i = num; i > 0; i--)
        {
            if ((int)(up / i) == ((float)up / (float)i) && (int)(down / i) ==
                ((float)down / (float)i))
            {
                up = up / i;
                down = down / i;
            }
        }
    }
    fract operator+(fract &rv)
    {
        int d;
        int ups;
        if (this->down != rv.down)
        {
            d = this->down*rv.down;
            ups = (this->up*rv.down) + (this->down*rv.up);
        }
        else
        {
            d = rv.down;
            ups = this->up + rv.up;
        }
        fract D(ups, d);
        D.dev();
        return D;
    }
    fract operator-(fract &rv)
    {
        int d;
        int ups;
        if (this->down != rv.down)
        {
            d = this->down*rv.down;
```

```

        ups = (this->up*rv.down) - (this->down*rv.up);
    }
    else
    {
        d = rv.down;
        ups = this->up - rv.up;
    }
    fract D(ups, d);
    return D;
}

fract operator*(fract &rv)
{
    int d = this->down*rv.down;
    int ups = this->up*rv.up;

    fract D(ups, d);
    return D;
}

fract operator/(fract &rv)
{
    int d = this->down*rv.up;
    int ups = this->up*rv.down;

    fract D(ups, d);
    return D;
}

friend ostream& operator<<(ostream& os, fract& d)
{
    os << d.up << "/" << d.down;
    return os;
}

friend istream& operator >> (istream &is, fract& d)
{
    is >> d.up >> d.down;
    return is;
}

private:
    int up;
    int down;
};

int main()
{
    fract D;
    cin >> D;
    D.dev();
    fract D2;
    cin >> D2;
    fract D3 = ((D + D2) / D);
    D3.dev();
    cout << D3 << endl;
    system("pause");
    return 0;
}

```

Результат виконання програми:

```
2 3
3 4
17/8
Для продолжения нажмите любую клавишу . . .
```

1.2 Клас для представлення двовимірного масиву

Розроблено клас для представлення двовимірного масиву (матриці) цілих чисел довільних розмірів. Створено конструктори та деструктор, перевантажено операції додавання, віднімання і множення (згідно з правилами роботи з матрицями), звертання за індексом, введення з потоку та виведення в потік. Створено власні класи винятків та генеруються відповідні об'єкти-винятки, якщо неможливо виконати ту чи іншу операцію.

Створено окрему функцію, яка отримує посилання на матрицю і виконує над масивом дії, вказані в таблиці. Функція не є методом класу або дружньою функцією.

У функції main() здійснено тестування всіх можливостей класу з перехопленням можливих винятків, а також розв'язано індивідуальну задачу.

Код програми:

```
#include<iostream>
using namespace std;
////////////////////////////////////
class Matrix
{
public:
    class ex
    {
        char*indexEx = "Bad Index";
        char*matrixSameEx = "Matrixes is not same";
        char*multiplyingEx = "cannot multiply matrixes";
        char*sizeEx = "wrong matrix size";
    public:
        ex(int i)
        {
            switch (i)
            {
            case 1:
                cout << indexEx << endl; break;
            case 2:
                cout << matrixSameEx << endl; break;
            case 3:
                cout << multiplyingEx << endl; break;
            case 4:
                cout << sizeEx << endl; break;
            }
        }
    };
    Matrix(int lines, int colons)
    {
        if (lines <= 0 || colons <= 0) { throw ex(4); }
        n = colons;
        m = lines;
        p = new int*[m];
        for (int i = 0; i < m; i++)
        {
            p[i] = new int[n];
        }
    }
    Matrix(const Matrix& A)
    {
```

```

        n = A.n;
        m = A.m;
        p = new int*[m];
        for (int i = 0; i < m; i++)
        {
            p[i] = new int[n];
        }
        for (int i = 0; i < m; i++)
        {
            for (int j = 0; j < n; j++)
            {
                p[i][j] = A.p[i][j];
            }
        }
    }
    ~Matrix()
    {
        for (int i = 0; i < m; i++)
        {
            delete p[i];
        }
        delete p;
    }
    int* operator[](int index)
    {
        if (index < 0)
        {
            throw ex(1);
        }
        return p[index];
    }
    friend ostream& operator<<(ostream&os, Matrix &M)
    {
        int col = M.n;
        int line = M.m;
        for (int i = 0; i < line; i++)
        {
            for (int j = 0; j < col; j++)
            {
                os << M[i][j] << " ";
            }
            os << endl;
        }
        return os;
    }
    friend istream& operator >> (istream&is, Matrix&M)
    {
        int a;
        for (int i = 0; i < M.m; i++)
        {
            for (int j = 0; j < M.n; j++)
            {
                is >> a;
                M[i][j] = a;
            }
        }
        return is;
    }
    Matrix operator+(Matrix &rv)
    {
        if (this->m != rv.m || this->n != rv.n)
        {
            throw ex(2);
        }
        Matrix temp(m, n);

```

```

Matrix th = *this;
for (int i = 0; i < m; i++)
{
    for (int j = 0; j < n; j++)
    {
        temp[i][j] = th[i][j] + rv[i][j];
    }
}
return temp;
}
Matrix operator-(Matrix &rv)
{
    if (this->m != rv.m || this->n != rv.n)
    {
        throw ex(2);
    }
    Matrix temp(m, n);
    Matrix th = *this;
    for (int i = 0; i < m; i++)
    {
        for (int j = 0; j < n; j++)
        {
            temp[i][j] = th[i][j] - rv[i][j];
        }
    }
    return temp;
}
Matrix operator*(Matrix&rv)
{
    if (this->n != rv.m)
    {
        throw ex(3);
    }
    int newN = rv.n;
    int newM = this->m;
    Matrix th = *this;
    Matrix temp(newM, newN);
    for (int i = 0; i < newM; i++)
        for (int j = 0; j < newN; j++)
        {
            temp[i][j] = 0;
            for (int k = 0; k < this->n; k++)
            {
                temp[i][j] += th[i][k] * rv[k][j];
            }
        }
    return temp;
}
Matrix operator*(int&k)
{
    int newM = this->m;
    int newN = this->n;
    Matrix temp(newM, newN);
    for (int i = 0; i < newM; i++)
    {
        for (int j = 0; j < newN; j++)
        {
            temp[i][j] = this->p[i][j] * k;
        }
    }
    return temp;
}
int getM()const{ return m; }
int getN()const{ return n; }

```



```

private:
    int n;
    int m;
    int **p;
};
////////////////////////////////////
void ind(Matrix&M);
void choise1();
void choise2(Matrix M);
void anyM(int m, int n);
////////////////////////////////////
int main()
{
    choise1();
    system("pause");
    return 0;
}
////////////////////////////////////
void ind(Matrix&M)
{
    int m = M.getM();
    int n = M.getN();
    for (int i = 0; i < m; i++)
    {
        for (int j = 0; j < n; j++)
        {
            if (M[i][j] == 0)
            {
                M[i][j] = 1;
            }
        }
    }
}

void choise1()
{
    int choise1;
    cout << "1) generate matrix (3,4)" << endl;
    cout << "2) enter (3,4) matrix by youself" << endl;
    cout << "3) make matrix with any lines and colons" << endl;
    cin >> choise1;
    system("cls");
    switch (choise1)
    {
    case 1:
    {
        const int m = 3;
        const int n = 4;
        Matrix M(m, n);
        int c = rand() % 20;
        for (int i = 0; i < m; i++)
        {
            for (int j = 0; j < n; j++)
            {
                M[i][j]=c;
                c = rand() % 20;
            }
        }
        cout << "generated matrix: \n" << M;
        system("pause");
        system("cls");
        choise2(M);
        break;
    }
    case 2:
    {

```

```

        const int m = 3;
        const int n = 4;
        Matrix M(m, n);
        cout << "enter (3,4) matrix" << endl;
        cin >> M;
        system("cls");
        choise2(M);
        break;
    }
    case 3:
    {
        int newM, newN;
        cout << "enter lines and colons" << endl;
        cin >> newM >> newN;
        system("cls");
        anyM(newM, newN);
        break;
    }
}
return;
}
void choise2(Matrix M)
{
    int m = M.getM();
    int n = M.getN();
    int choise1;
    cout << "1) do ind task" << endl;
    cout << "2) matrix + matrix2" << endl;
    cout << "3) matrix - matrix2" << endl;
    cout << "4) matrix * matrix2" << endl;
    cout << "5) matrix * number" << endl;
    cout << "0) exit" << endl;
    cin >> choise1;
    system("cls");
    switch (choise1)
    {
        case 1:
            ind(M);
            cout << "Matrix after ind task: \n" << M << endl;
            break;
        case 2:
        {
            Matrix C(m, n);
            cout << "enter elements of (" << m << ", " << n << ") matrix2" << endl;
            cin >> C;
            system("cls");
            cout << "matrix + matrix2 is: \n" << M << "+" << C << "= \n" << M + C <<
endl;
            break;
        }
        case 3:
        {
            Matrix C(m, n);
            cout << "enter elements of (" << m << ", " << n << ") matrix2" << endl;
            cin >> C;
            system("cls");
            cout << "matrix - matrix2 is: \n" << M << "-" << C << "= \n" << M - C <<
endl;
            break;
        }
        case 4:
        {
            int newN;
            cout << "enter count of colons" << endl;
            cin >> newN;

```

```

        system("cls");
        cout << "enter elements of (" << n << ", " << newN << ") matrix2" << endl;
        Matrix K(n, newN);
        cin >> K;
        system("cls");
        cout << "matrix * matrix2 is: \n" << M << "*" \n" << K << "= \n" << M*K <<
endl;
        break;
    }
    case 5:
    {
        cout << "enter the number" << endl;
        int k;
        cin >> k;
        system("cls");
        cout << "matrix * " << k << " is" << endl;
        cout << M << "*" \n" << k << endl << "= \n" << M*k;
    }
    case 0:
        return;
    }
}
void anyM(int m, int n)
{
    Matrix B(m, n);
    cout << "enter elements of (" << m << ", " << n << ") matrix" << endl;
    cin >> B;
    system("cls");
    choise2(B);
}

```

Результати програми:

```
1) generate matrix (3,4)
2) enter (3,4) matrix by yourself
3) make matrix with any lines and colons
3
```

```
enter lines and colons
2 2
```

```
enter elements of (2,2) matrix
3 4
5 6
```

```
1) do ind task
2) matrix + matrix2
3) matrix - matrix2
4) matrix * matrix2
5) matrix * number
0) exit
4
```

```
enter count of colons
3
```

```
enter elements of (2,3) matrix2
2 3 4
6 4 1
```

```
matrix * matrix2 is:
3 4
5 6
*
2 3 4
6 4 1
=
30 25 16
46 39 26
```

Для продолжения нажмите любую клавишу . . .

1.3 Підрахунок суми введених значень

Створено клас з одним закритим елементом даних цілого типу, геттером і конструктором з одним параметром. В цьому ж класі створено закрите статичне поле, яке зберігає суму цілих елементів даних всіх раніше створених об'єктів. Під час кожного виклику конструктора до статичного поля додається нове значення. Статична публічна функція цього ж класу повертає цю суму.

У функції main() створено декілька об'єктів і виведено отриману суму.

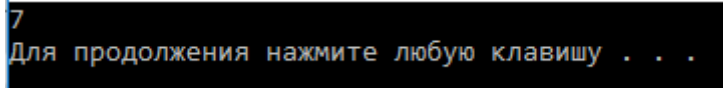
Код програми:

```
#include<iostream>
using namespace std;

class summ
{
public:
    summ(int k)
    {
        las += k;
    }
    static int getL(){ return las; }
private:
    static int las;
};
int summ::las;

int main()
{
    summ a(3);
    summ b(4);
    cout << summ::getL() << endl;
    system("pause");
    return 0;
}
```

Результат виконання:



```
7
Для продолжения нажмите любую клавишу . . .
```

Висновки:

У даній лабораторній роботі я навчився використовувати класи в C++. На базі здобутих знань створив програмне забезпечення, код якого створений за допомогою ООП. Також було вивчено необхідний теоретичний матеріал, який мені допоможе у наступних проектах.