**ИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

Федеральное государственное бюджетное образовательное учреждение   
высшего образования

**«Сибирский государственный университет науки и технологий   
имени академика М.Ф. Решетнева»**

Институт информатики и телекоммуникаций

Кафедра информатики и вычислительной техники

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ**

Языки программирования

|  |
| --- |
| Шаблонные классы |

Руководитель А.В. Проскурин

подпись, дата инициалы, фамилия

Обучающийся БПИ22-02, 221219040 К.В. Трифонов

номер группы, зачетной книжки подпись, дата инициалы, фамилия

Красноярск 2023 г.

# ЦЕЛЬ РАБОТЫ

Получение практических навыков разработки и отладки программ, c использованием шаблонов функций и классов.

# порядок выполнения работы

1. Ознакомиться с общей постановкой задачи.

2. Ознакомится с вариантом задания – соответствует вашему номеру в списке группы (при нехватке заданий вариант задания вычисляется как номер\_в\_списке\_группы - количество\_заданий).

3. Разработать классы согласно варианту задания.

4. Написать и отладить программу на подготовленных наборах тестовых данных.

5. Подготовить отчет по лабораторной работе. Отчет должен включать в себя:

• титульный лист;

• цель лабораторной работы;

• постановку задачи;

• текст программы с комментариями;

• демонстрацию работы программы (Снимки экрана при выполнении действий программы с описанием).

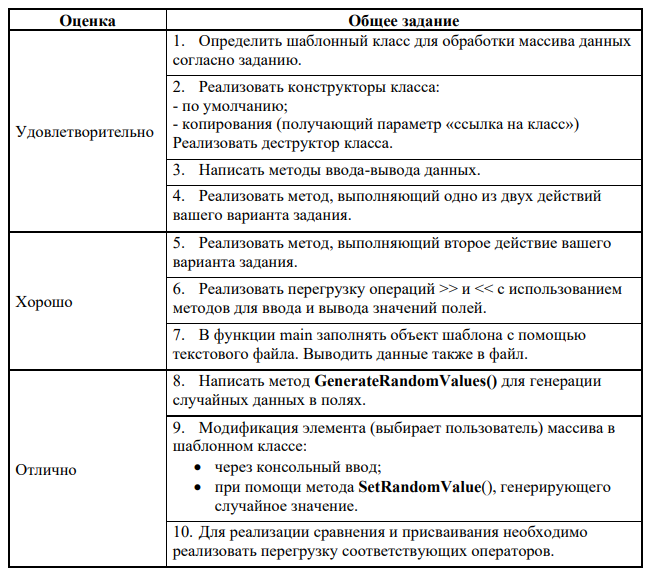
• краткие ответы на контрольные вопросы;

• выводы по лабораторной работе.

6. Защитить лабораторную работу перед преподавателем.

# постановка задачи

Необходимо разработать программу, в которой будет реализован шаблонный класс, выполняющий обработку массива данных согласно варианту задания. В зависимости от оценки, на которую вы претендуете, необходимо выполнить следующие задания (Для каждой следующей оценки нужно выполнить ВСЕ предыдущие задания, если обратное не указано явно):



**Вариант №21.**

Напишите программу, в которой создайте шаблонный класс **TMatrix**, реализующий динамический двумерный массив и функции-методы:

▪ определения количества элементов массива, которые меньше среднего арифметического значения всех элементов четных строк;

▪ нахождения суммы положительных элементов массива (для комплексного числа проверяется действительная часть).

Протестируйте вашу программу для работы с массивами различных типов: **double** и класса **комплексное число** (представлено двумя компонентами). Этот класс необходимо создать.

# ХОД РАБОТЫ

Описание класса TMatrix

*template* <*class* Type>

*class* TMatrix{

*int* a,b;

        Type\*\* matrix;

Поля a b для i x j матрицы, указатель на матрицу matrix.

*public:*

*//Конструктор по умолчанию*

        TMatrix() : a(0), b(0), matrix(nullptr)

        {}

*//Конструктор с параметрами*

        TMatrix(*int* *I*, *int* *J*) : a(*I*), b(*J*), matrix(new Type\*[a]){

            for(*int* i = 0; i<a; i++){

                matrix[i] = new Type [b];

            }

        }

*//Конструктор копирования*

        TMatrix(*const* TMatrix*&* *B*) {

            Clear();

            for(*int* i = 0; i<a; i++){

                for (*int* j = 0; j<b; j++){

                    matrix[i][j] = *B*.matrix[i][j];

                }

            }

        }

Выделение динамической памяти в конструкторах

*//Деструктор*

        ~TMatrix(){

            Clear();

        }

*void* Clear(){

            for (*int* i = 0; i<a; i++){

                delete[] matrix[i];

            }

            delete[] matrix;

            matrix=nullptr;

            a = b = 0;

        }

Метод Clear() для очистки динамической памяти и обнуления полей

*//Операторы*

*//Индексирование*

        Type*&* operator ()(*int* *A*,*int* *B*) *const*{

            return matrix[*A*][*B*];

        }

*//присваивание*

*void* operator = (TMatrix*&* *B*){

            Clear();

            a = *B*.a;

            b = *B*.b;

            matrix = new Type \* [a];

            for(*int* i = 0; i<a; i++){

                matrix[i] = new Type [b];

            }

            for(*int* i = 0; i<a; i++){

                for (*int* j = 0; j<b; j++){

                    matrix[i][j] = *B*.matrix[i][j];

                }

            }

        }

*//сравнение*

*bool* operator<(*const* TMatrix*&* *B*){

            if (*this*->Psum()<*B*.Psum()){

                return 1;

            }

            else{

                return 0;

            }

        }

*bool* operator>(*const* TMatrix*&* *B*){

            if (*this*->Psum()>*B*.Psum()){

                return 1;

            }

            else{

                return 0;

            }

        }

*//сетеры*

*void* setA(*int* *A*){a = *A*;}

*void* setB(*int* *B*){b = *B*;}

*void* GenerateRandomValues(*int* *min*, *int* *max*){

            if (a==0||b==0){

                cout<<"Введите размерность \_ x \_\n";

                cin>>a>>b;

                matrix = new Type \* [a];

                for(*int* i = 0; i<a; i++){

                    matrix[i] = new Type [b];

                }

            }

            srand(time(NULL));

            for(*int* i = 0; i<a; i++){

                for (*int* j = 0; j<b; j++){

                    matrix[i][j] = *min* + rand()%(*max*-*min*+1);

                }

            }

        }

*void* SetRandomValue(*int* *I*, *int* *J*,*int* *min*, *int* *max*){

            matrix[*I*][*J*] = *min* + rand()%(*max*-*min*+1);

        }

        istream*&* set(istream*&* *in*){

*//Clear();*

            if (&*in* == &cin){

                cout<<"Введите размерность \_ x \_\n";

            }

*in*>>a>>b;

            matrix = new Type \* [a];

            for(*int* i = 0; i<a; i++){

                matrix[i] = new Type [b];

            }

            if (&*in* == &cin){

                cout<<"Введите элементы через пробел:\n";

            }

            for(*int* i = 0; i<a; i++){

                for (*int* j = 0; j<b; j++){

*in*>>matrix[i][j];

                }

            }

            return *in*;

        }

*//гетеры*

*int* getA(){return a;}

*int* getB(){return b;}

        ostream*&* get(ostream*&* *out*){

            for(*int* i = 0; i<a; i++){

                for (*int* j = 0; j<b; j++){

*out*<<matrix[i][j]<<" ";

                }

*out*<<"\n";

            }

            return *out*;

        }

*//Модификация элемета*

*void* modify(*int* *I*,*int* *J*){

*int* sw;

            cout<<"1) Модифицировать "<<*I*<<" x "<<*J*<<" случайно\n";

            cout<<"2) Модифицировать "<<*I*<<" x "<<*J*<<" через консоль\n";

            cin>>sw;

            switch(sw){

                case 1:{

*int* min,max;

                    cout<<"Введите диапазон от \_ до \_:\n";

                    cin>>min>>max;

                    SetRandomValue(*I*,*J*,min,max);

                    break;

                }

                case 2:{

                    Type v;

                    cin>>v;

                    matrix[*I*][*J*] = v;

                }

            }

        }

*int* LowAwg(){

*float* sum = 0;

*int* c = 0;

            for (*int* i = 0; i<=a/2; i++){

                for (*int* j = 0; j<b; j++){

                    sum+=matrix[i\*2][j];

                    c++;

                }

            }

*float* avg = sum/c;

*int* c2 = 0;

            for(*int* i = 0; i<a; i++){

                for (*int* j = 0; j<b; j++){

                    if (matrix[i][j]<avg){

                        c2++;

                    }

                }

            }

            return c2;

        }

        Type Psum(){

            Type sum=0;

            for(*int* i = 0; i<a; i++){

                for (*int* j = 0; j<b; j++){

                    if (matrix[i][j]>0){

                        sum+=matrix[i][j];

                    }

                }

            }

            return sum;

        }

*//Работа с файлами*

*void* f\_in(string *filename*){

        Clear();

        ifstream fin(*filename*);

        set(fin);

        cout<<"Запись завершена.\n";

    }

*void* f\_out(string *filename*){

        ofstream fout(*filename*);

        get(fout);

        cout<<"Запись завершена.\n";

    }

};

*template*<*class* Type>

ostream*&* operator <<(ostream*&* *out*, TMatrix<Type>*&* *Matr*){

    if (&*out* == &cout){

*out*<<"Матрица размером "<<*Matr*.getA()<<" x "<<*Matr*.getB()<<"\n";

    }

    else{

*out*<<*Matr*.getA()<<" "<<*Matr*.getB()<<"\n";

    }

*Matr*.get(*out*);

    return *out*;

}

*template*<*class* Type>

istream*&* operator >>(istream*&* *in*, TMatrix<Type>*&* *Matr*){

*Matr*.set(*in*);

    return *in*;

}

# Тестирование

# ОТВЕТЫ НА КОНТРОЛЬНЫЕ ВОПРОСЫ

1. **Какой принцип ООП реализуется с помощью шаблонных функций и классов?**

Принцип полиморфизма - способность функции обрабатывать данные разных типов.

1. **Когда целесообразно использовать шаблоны функций?**

Целесообразно использовать шаблоны функций, когда необходимо написать универсальный код для работы с различными типами данных без дублирования.

1. **Каков общий вид объявления шаблона функции?**
   * *Ответ:*

cppCopy code

template <class T> T functionName(T parameter);

1. **Каков синтаксис объявления шаблонного класса?**
   * *Ответ:*

cppCopy code

template <class T> class ClassName { // члены класса };

1. **В чем особенности определения шаблонного класса?**
   * *Ответ:* Определение класса и его методов обычно размещается в том же файле, что и объявление.
2. **В чем особенности определения метода шаблонного класса вне класса?**
   * *Ответ:* Методы шаблонного класса могут быть определены вне класса с использованием того же синтаксиса шаблона.
3. **Что такое специализация шаблона?**
   * *Ответ:* Специализация шаблона позволяет предоставить специфическую реализацию для конкретного типа данных.
4. **Как объявить объект шаблонного класса?**
   * *Ответ:*

cppCopy code

ClassName<int> obj; // пример объявления объекта класса для типа int

1. **Что такое инстанцирование?**
   * *Ответ:* Инстанцирование происходит, когда создается экземпляр шаблонного класса или вызывается шаблонная функция с определенным типом данных.
2. **Чем в строке template <class T> может быть T?**
   * *Ответ:* **d) T может быть любым типом – как встроенным в язык C++, так и определенным программистом.**

# ВЫВОД

Было изучено то, как использовать шаблоны в C++, которые позволяют делать универсальные куски кода для работы с разными типами данных. С их помощью можно писать программы, которые гибко обрабатывают разные данные, что упрощает жизнь и делает код более переиспользуемым. Работа с шаблонами в С++ — это один из способов делать программы более гибкими и удобными.