**Правительство Российской Федерации**

Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет «Высшая школа экономики» Кафедра «Компьютерная безопасность»

# ОТЧЕТ
# К ЛАБОРАТОРНОЙ РАБОТЕ №1

**по дисциплине**

**«Языки программирования»**

Работу выполнил студент
группы СКБ-201 _____ П.Е. Зильберштейн
подпись, дата

Работу проверил _____ С.А. Булгаков
подпись, дата

Москва 2021

# Содержание

# Постановка задачи

Разработать программу на языке Си++ (ISO/IEC 14882:2014), демонстрирующую решение поставленной задачи.

### Общая часть

Разработать набор классов, объекты которых реализуют типы данных, указанные ниже. Для классов разработать необходимые конструкторы, деструктор, конструктор копирования, а также методы, обеспечивающие изменение отдельных составных частей объекта. Используя перегрузку операторов (operator) разработать стандартную арифметику объектов, включающую арифметические действия над объектами и стандартными типами (целыми, вещественными, строками – в зависимости от вида объектов), присваивание, ввод и вывод в стандартные потоки (используя операторы «<<» и «>>»), приведение к/от базового типа данных. Организовать операции в виде конвейера значений, с результатом (новым объектом) и сохранением значений входных операндов.

### Задачи

1. Дата и время, представленные целочисленными переменными: год, месяц, день, час, минута, секунда. Базовый тип: `uint64_t` формат представления unix time. Реализовать возможность преобразования в/из формата представления filetime (целое 64-х разрядное значение, представляющее число интервалов по 100 наносекунд, прошедших с первого января 1601 года).

2. Целое произвольной длины (во внешней форме представления в виде строки символовцифр). Базовый тип: `std::string`.

3. Год «от Адама»,имеющий внутреннеепредставление в видецелочисленныхпеременных: индикт, круг солнцу, круг луне. Диапазоны значений (циклические): индикт 1—15, круг солнцу 1—28, круг луне 1—19. Ежегодно каждая переменная увеличивается на 1. Итоговое значение вычисляется как произведение переменных(диапазона на некоторый множитель; переменные независимы), а хранимое значение является остатком от деления (на диапазон), при этом 0 соответствует максимум. Необходима возможность отображения/задания как в виде одного числа, так и виде трех. Реализовать возможность преобразования в/из формата представления «от рождества Христова» используя соответствие 1652 = 7160 «от Адама».

4. Разреженная матрица, представленная динамическим массивом структур, содержащих описания ненулевых коэффициентов: индексы местоположения коэффициента в матрице (целые) и значение коэффициента (вещественное).

# 1 Алгоритм решения задачи

## 1.1 Задача 1

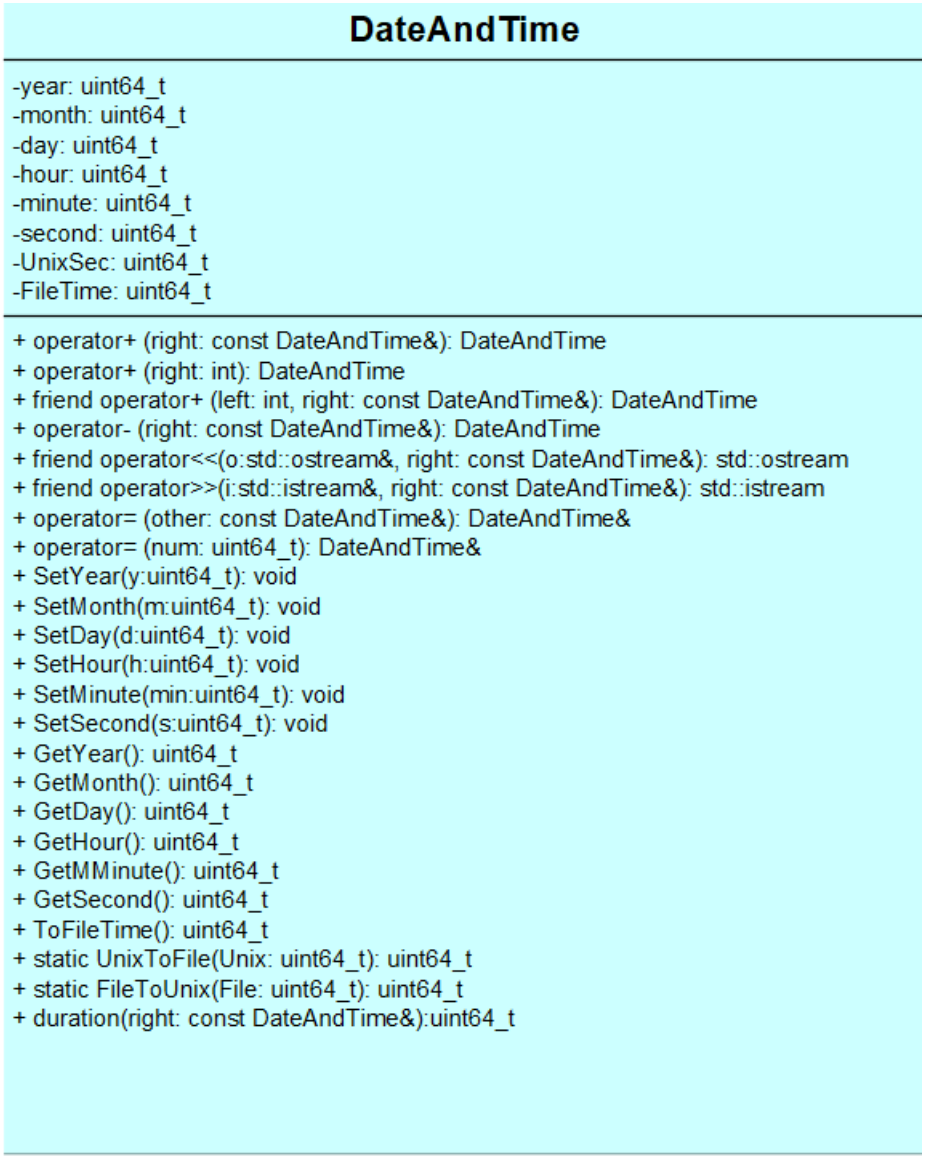


Рис. 1: UML-диаграмма класса DateAndTime.

Для решения данной задачи был разработан класс DateAndTime (рис. 1), содержащий закрытые поля year, month, day, hour, minute, second, требуемые по заданию, а также UnixSec и FileTime, необходимые для упрощения преобразования. Все поля имеют тип `std::uint64_t`. Первые 6 полей необходимы для хранения человекочитаемой даты, UnixSec - для хранения секунд с эпохи Unix, FileTime - для хранения количества секунд в одноименном формате. Также класс содержит:

- конструктор по умолчанию;
- конструктор с параметрами, UnixTime или FileTime, и флага с типом bool, создающий объект на основе целого числа;
- конструктор копирования;
- декструктор;

Помимо этого в классе имеются:

- перегрузка операции '+' для двух случаев - сложение двух объектов класса и сложение объекта класса с целым числом (коммутативно);
- перегрузка операции '-';
- перегрузка операций помещения в поток ('«') и из потока ('«');
- перегрузка операции присваивания ('=') для двух случае - объекту класса присваивается другой объект класса и объекту класса присваивается целочисленная переменная (считается как UnixTime);

Также в задании требовалось наличие сеттеров и геттеров в классе: все сеттеры имеют тип void, принимают в качестве параметра число типа `std::uint64_t` и имеют префикс Set; все геттеры имеют тип возвращаемого значения `std::uint64_t` и квалификатор типа const, так как не должны изменять исходный объект и имеют префикс Get.

Функция-член ToFileTime необходима для получения количества секунд в формате FileTime данного объекта

Функция duration предназначена для вычисления промежутка времени (в секундах) между одни событием (объектом) и другим.

В классе присутствуют две статические функции-члены, предназначенные для конвертирования времени из UnixTime а FileTime и обратно без привязки к конкретному объекту.

## 1.2 Задача 2

Для решения данной задачи был разработан класс BigNumber (рис. 2), содержащий закрытые поля number типа `std::string`, содержащее строковое представление числа, не хранящее знак; и sign типа `bool`, отвечающее за знак числа (0 - число положительное, 1 - отрицательное), требуемые по заданию,

Также класс содержит:

- конструктор по умолчанию;
- конструктор с параметрами, создающий объект на основе строки;
- конструктор с параметрами, создающий объект на основе целого числа типа `long long`
- конструктор копирования;
- декструктор;

Помимо этого в классе имеются:

- перегрузка операций равенства ('==', '!=') и сравнения ('>', '>=', '<', '<='');
- перегрузка арифметических операций ('+', '-', '*', '/', '%');
- перегрузка арифметических операций присваиванием ('+=', '-=', '*=', '/=', '%=');
- перегрузка операций помещения в поток ('«') и из потока ('«');
- перегрузка операции присваивания ('=') для двух случае - объекту класса присваивается другой объект класса и объекту класса присваивается строка;
- перегрузка операции унарного минуса;

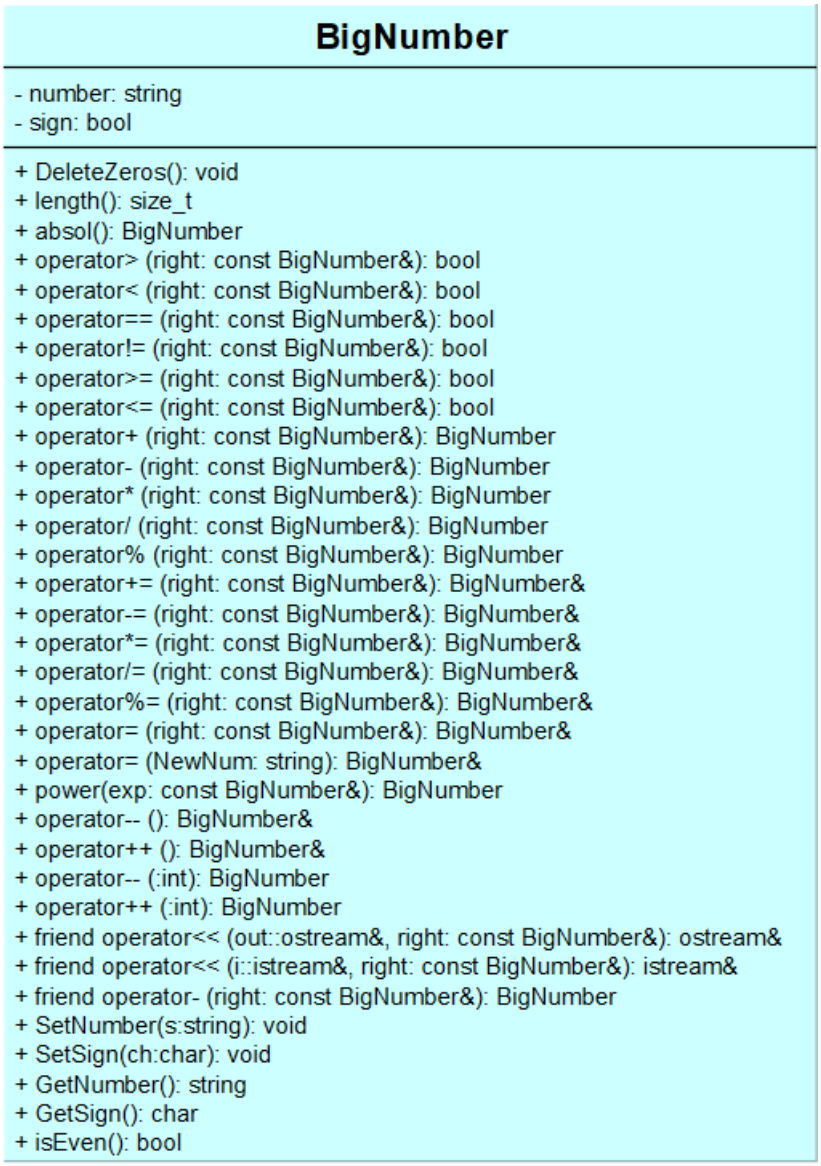


Рис. 2: UML-диаграмма класса BigNumber.

– перегрузка операций префиксного и постфиксного инкрементов и декрементов;

– функция возведения числа в степень (power);

– функция приведения объекта к базовому типу;

– функция получения модуля числа (absol);

– функция получения четности числа (true - число четное, false - нечетное);

– закрытую (вспомогательную) функцию DeleteZeros, которая имеет тип void, отвечающая за удаление незначащих нулей числа;

– закрытую (вспомогательную) функцию length, которая имеет тип `std::string` и возвращает длину числа (=количество цифр в числе);

Также в задании требовалось наличие сеттеров и геттеров в классе. В классе присутсвуют два сеттера: SetNumber имеет тип void, принимает строку и изменяет поле number входного объекта, SetSign имеет тип void, принимает символ и изменяет поле sign входного объекта; и

два геттера: GetNumber имеет тип `std::string`, квалификатор типа const и возвращает поле number входного объекта, GetSign имеет тип `char`, квалификатор типа const и возвращает поле sign входного объекта

## 1.3 Задача 3

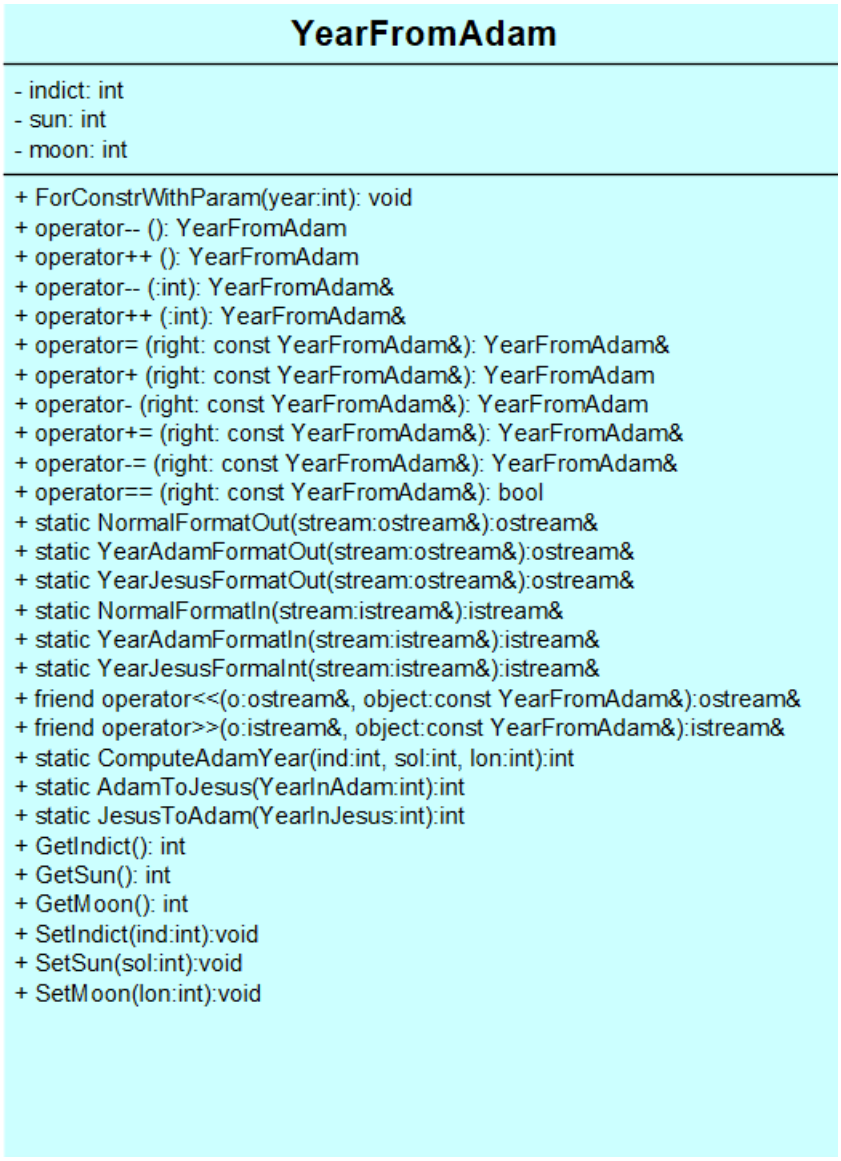


Рис. 3: UML-диаграмма класса YearFromAdam.

Для решения данной задачи был разработан класс YearFromAdam (рис. 3), содержащий закрытые поля indict, sun и moon типа int, содержащие соответсвенно индикт, круг Солнцу и круг Луне.

Также класс содержит:

- конструктор по умолчанию;
- конструктор с параметрами, создающий объект на основе трех целочисленных переменных;
- конструктор с параметрами, создающий объект на основе целого числа типа `int` и флага с типом `bool`;

– конструктор копирования;

– декструктор;

Помимо этого в классе имеются:

– перегрузка операций префиксного и постфиксного инкрементов и декрементов;

– перегрузка операции присваивания ('=') для случая, когда объекту класса присваивается другой объект класса;

– перегрузка арифметических операций ('+', '-');

– перегрузка арифметических операций присваиванием ('+=', '-=');

– перегрузка операции равенства ('==');

– перегрузка операций помещения в поток ('«') и из потока ('«');

– перегрузка операции унарного минуса;

Также в задании требовалось наличие сеттеров и геттеров в классе. В классе присутсвуют три сеттера: SetIndict имеет тип void, принимает целое число и изменяет поле indict входного объекта, SetSun имеет тип void, принимает целое число и изменяет поле sun входного объекта, SetMoon имеет тип void, принимает целое число и изменяет поле moon входного объекта; и три геттера: GetIndict имеет тип `int`, квалификатор типа const и возвращает поле indict входного объекта, GetSun имеет тип `int`, квалификатор типа const и возвращает поле sun входного объекта, GetSun имеет тип `int`, квалификатор типа const и возвращает поле moon входного объекта.

Помимо этого в классе присутсвутют статические функции для конвертации:

– ComputeAdamYear, которая рассчитывает количество лет "от Адама основываясь на трех целых числах - индикте, кругу Солнцу и кругу Луне

– AdamToJesus, имеющая тип int, принимающая целое число и переводящая количество лет "от Адама"в год от Рождества Христова

– AdamToJesus, имеющая тип int, принимающая целое число и переводящая год от Рождества Христова в год "от Адама";

В классе также реализованы статические функции для изменения формата ввода и вывода:

– NormalFormatOut, принимающая и возвращающая ссылку на поток вывода, которая обеспечивает вывод по умолчанию (в виде трех чисел - полей объекта);

– YearAdamFormatOut, принимающая и возвращающая ссылку на поток вывода, которая обеспечивает вывод объекта в формате года "от Адама";

– YearJesusFormatOut, принимающая и возвращающая ссылку на поток вывода, которая обеспечивает вывод объекта в формате года от Рождества Христова;

– NormalFormatIn, принимающая и возвращающая ссылку на поток ввода, которая обеспечивает ввод по умолчанию (в виде трех чисел - полей объекта);

– YearAdamFormatIn, принимающая и возвращающая ссылку на поток ввода, которая обеспечивает ввод объекта на основе года "от Адама";

– YearJesusFormatIn, принимающая и возвращающая ссылку на поток ввода, которая обеспечивает ввод объекта на основе года от Рождества Христова;

## 1.4 Задача 4

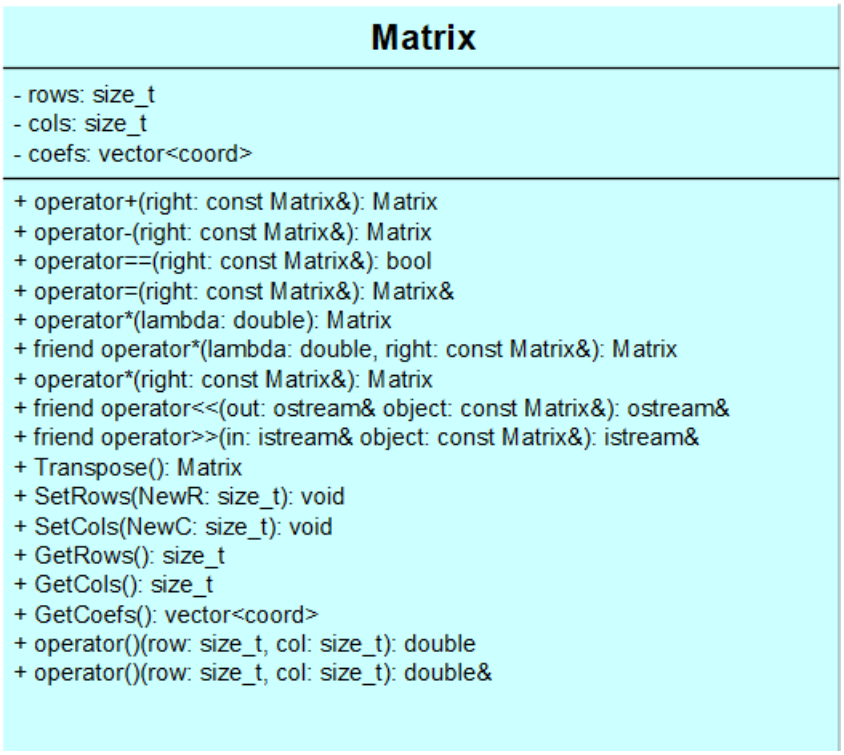


Рис. 4: UML-диаграмма класса Matrix.

Для решения данной задачи был разработан класс Matrix (рис. 4), содержащий закрытые поля rows и cols типа `std::size_t`, содержащие соответсвенно количество строк и столбцов матрицы, а также вектор структур coefs, который хранит объекты типа coord (структуру с полями, отвечающими за индекс по строке, индекс по столбцу и значение коэффициента).

Также класс содержит:

- конструктор по умолчанию;
- конструктор с параметрами, создающий объект на основе двух неотрицательных чисел:
- конструктор копирования;
- декструктор;

Помимо этого в классе имеются:

- перегрузка арифметических операций ('+', '-');
- перегрузка операции равенства ('==');
- перегрузка операции присваивания ('=') для случая, когда объекту класса присваивается другой объект класса;
- перегрузка операции * для двух случаев: умножение объекта на вещественное число (коммутативно) и умножение объекта на объект;
- перегрузка операций помещения в поток ('«') и из потока ('«');

– перегрузка операции "круглые скобочки ()"для двух случаев - когда значение нужно ввиде rvalue и когда значение нужно ввиде lvalue;

– функция Transpose, возвращающая новый объект класса, являющий результатом транспонирования исходного;

Также в задании требовалось наличие сеттеров и геттеров в классе. В классе присутсвуют два сеттера: SetRows имеет тип void, принимает целое неотрицательное число и изменяет поле rows входного объекта, SetCols имеет тип void, принимает целое неотрицательное число и изменяет поле cols входного объекта; и три геттера: GetRows имеет тип `std::size_t`, квалификатор типа const и возвращает поле rows входного объекта, GetCols имеет тип `std::size_t`, квалификатор типа const и возвращает поле cols входного объекта, GetCoefs имеет тип `std::vector<coord>`, квалификатор типа const и возвращает поле coefs входного объекта.

# 2 Выполнение задания

## 2.1 Задача 1

### 2.1.1 Конструкторы и деструкторы:

Конструктор по умолчанию: реализация была доверена компилятору (с помощью конструкции =default);

Конуструктор копирования:полям нового объекта соотвественно сопоставляются поля уже имеющегося объекта;

Конструктор с параметрами от одного целого числа и одного флага типа bool: по умолчанию flag имеет значение false и этмоу соответсвует целое число - время в формате UnixTime. Если установлено значениЕ true, то предполагается, что целое число - время в формате FileTime. Создается временная переменная, в которую записывается входное число. Если это необходимо (проверяется флаг), эта переменная переводится из FileTime в UnixxTime. Затем создается переменная типа `time_t`, которой присваивается время в формате UnixTime. Также создается переменная типа "указатель на char которой присваивается результат выполнения функции `std::ctime` Данная функция возвращает строку в стиле языка программирования C, которая содержит информацию о всех необходимых данных. Затем начинается процесс обработки данной строки и занесения данных в соответсвующие поля;

Деструктор: реализация доверена компилятору (в описании указана комбинация =default);

### 2.1.2 Арифметика:

Перегрузка '+' для случая "Объект класса + объект класса": функция возвращает объект, сконструированный на основе суммы полей Unix исходных объектов;

Перегрузка '+' для случая "Объект класса + целое число": функция возвращает объект, сконструированный на основе суммы поля Unix исходного объекта и целого числа;

Перегрузка '+' для случая "Целое число + объект класса": функция возвращает результат выполнения перегрузки operator+ случая "Объект класса + целое число";

Перегрузка '-': функция возвращает разность полей Unix левого объекта и правого;

### 2.1.3 Остальные функции:

Перегрузка помещения в поток('«'): происходит помещение в поток строки в формате hh:mi:ss dd.mm.year, основываясь на значениях полей объекта;

Перегрузка чтения из потока ('»'): с потока считывается время в формате UnixTime, затем по схеме, аналогично конструктору с параметром (с использование `std::ctime` и обработкой полученной строки), полям объекта присваиваются значения;

Перегрузка '=' для случая, когда объекту класса присваивается другой объект класса: всем полям левого объекта присваиваются соответсвующие значения полей правого объекта;

Перегрузка '=' для случая, когда объекту класса присваивается целое число: предполагается, что целое число - время в формате UnixTime. Тогда по алгоритму, аналогичному алгоритму в конструкторе с параметрами (с помощью функции `std::ctime` и обработки строки) полям объекта присваиваются значения;

Перегрузка операции приведения к базовому типу: функция возвращает значение поля UnixSec;

Функция ToFileTime: возвращает значение времени, соответсвующее переданному объекту в формате FileTime;

Функция UnixToFile: конвертирует время из формата UnixTime в формат FileTime по математически полученной формуле;

Функция FileToUnix: конвертирует время из формата FileTime в формат UnixTime по математически полученной формуле;

Функция Duration: высчитывает длительность между двумя моментами времени как количество секунд. Функция возвращает модуль числа равного разности исходных объектов;

Функция SetYear: совершается цикл от меньшего значения из пары "аргумент-поле"до большего. Считается количество дней на этом промежутке, переводится в секунды и если новое значение больше старого, то поле UnixSec увеличивается на это значение, иначе - уменьшается. Помимо этого необходимо изменить поле FileTime;

Функция SetDay: сначала проверяется, что новое значение дня может существовать (этот день есть в установленном месяце). Затем вычисляется разность в днях, потом перезаписывается соответсвующее поле. После этого разность переводится в секунды и пересчитывается значение полей UnixTSec и FileTime;

Функция SetHour: сначала проверяется, что новое значение часа не превосходит 23. Затем вычисляется разность в часах, потом перезаписывается соответсвующее поле. После этого разность переводится в секунды и пересчитывается значение полей UnixTSec и FileTime;

Функция SetMinute: сначала проверяется, что новое значение минут не превосходит 59. Затем вычисляется разность в минутах, потом перезаписывается соответсвующее поле. После этого разность переводится в секунды и пересчитывается значение полей UnixTSec и FileTime;

Функция SetSecond: сначала проверяется, что новое значение секунд не превосходит 59. Затем вычисляется разность в секундах, потом перезаписывается соответсвующее поле. После этого разность переводится в секунды и пересчитывается значение полей UnixTSec и FileTime;

Функция GetYear: возвращает значение поля year;

Функция GetMonth: возвращает значение поля month;

Функция GetDay: возвращает значение поля day;

Функция GetHour: возвращает значение поля hour;

Функция GetMinute: возвращает значение поля minute;

Функция GetSecond: возвращает значение поля second;

## 2.2 Задача 2

### 2.2.1 Конструкторы и деструкторы:

Конструктор по умолчанию: реализация доверена компилятору (в описании указана комбинация =default);

Конуструктор копирования: полям нового объекта соотвественно сопоставляются поля уже имеющегося объекта;

Конструктор с параметром `std::string`: сначала происходит проверка, чтобы строка была непустая. Затем, если первый символ '-', то поле знака становится true, а полю number присваивается подстрока исходной строки, начинающая со второго (по порядку) символа. Если же первый (по порядку) символ не '-', то полю присваивается вся исходная строка. В конце происходит удаление незначащих нулей (на случай, если строка начинается с нуля);

Конструктор с параметром `long long`: сначала происходит проверка, знака числа. Если число отрицательное, то поле sign устанавливается на значение true. Затем полю number присваивается число, взятое по модулю (с помощью функции стандартной библиотеки `std::llabs`), преобразованное в строку с помощью функции стандартной библиотеки `std::to_string`;

Деструктор: реализация доверена компилятору (в описании указана комбинация =default);

### 2.2.2 Арифметика:

Перегрузка '+': сначала проверяются знаки чисел: если левое число имеет знак минус, а правое - плюс, то можно рекуррентно вызвать перегрузку operator- для случая "модуль Правого минус модуль левого"; если же левое число положительно, а правое отрицательно, то можно рекуррентно вызвать перегрузку operator- для случая "Модуль левого минус модуль правого"; если оба числа отрицательные,то из функции надо вернуть объект, который по модулю равен сумме модулей данных чисел с установленным полем sign на значение true; Если же оба числа положительны, то достаточно реализовать школьный метод сложения чисел ("в столбик"). Для этого сначала надо сделать так, чтобы числа были одного размера (если одно имеет меньше цифр, то в начало можно добавить незначащие нули). Затем происходит поцифровое сложение с учетом того, что может произойти переполнение (сумма цифра модет оказаться числом, а не цифрой), тогда надо в соответсвующее место поместить число по модулю 10 и прибавить единицу в следующий разряд. Необходимо учесть, что прибавление единицы в следующий разряд может привести к его переполнению (этот случай обрабатывается в цикле). В конце происходит удаление незначащих нулей;

Перегрузка '-': сначала проверяются знаки чисел: если левое число имеет знак плюс, а правое - минус, то можно рекуррентно вызвать перегрузку operator+ для случая "модуль левого плюс модуль правого"; если же левое число отрицательно, а правое положительно, то можно рекуррентно вызвать перегрузку operator+ для случая "Модуль левого плюс модуль правого установив у результат поле sign на значение true; если оба числа отрицательные, то можно рекурренто вызвать перегрузка operator- для случая "Модуль правого минус модуль левого"; Если же оба числа положительны, то тоже возможно два случая: Если вычитаемое больше уменьшаемого, то возвращается объект со знаком мидус, по модулю равный "модуль правого минус модуль левого". Если же уменьшаемое больше вычитаемого, то достаточно реализовать школьный метод вычитания чисел ("в столбик"). Для этого сначала надо сделать так, чтобы числа были одного размера (если одно имеет меньше цифр, то в начало можно добавить незначащие нули). Затем происходит поцифровое вычитание с учетом того, что может произойти переполнение (изм меньшей цифры вычтется большая и надо "занимать"у старшего разряда (этот случай обрабатывается в цикле). В конце происходит удаление незначащих нулей;

Перегрузка '*': для перегрузки умножения так же применяется стандартный алгоритм "в столбик". Для удобства (работы напрямую с цифрами) строки-поля number переведены в вектора и отреверсированы. После перемножения в столбик получаем вектор, являющийся итоговым числом, взятым по модулю, но цифры в нем стоят в обратном порядке, поэтому его надо опять отреверсировать с помощью функции std::reverse из заголовочного файла <Algorithm>. После этого поле результирующего объекта заполняется на основе вектора, удаляются незначащаие нули и устанавливается знак результирующего объекта, используя операцию исключающего ИЛИ;

Перегрузка '/': сначала проверяется тривиальный случай: когда делимое меньше делителя. В таком сучае функция возвращает нулевой объект класса. Затем реализован классический

алгоритм деления "в столбик затем удаляются незначащие нули и выставляется знак с помощью операции исключающего ИЛИ;

Перегрузка '%': сначала проверяется тривиальный случай: когда левое число меньше правого, тогда его остаток от деления равен самому числу. Если же левое симло больше правого, то создается объект класса Antje, хранящий целую часть числа (реализовано через перегрузку operator/). Затем вычисляет результирующий объект по формуле x = x - [x], который в конце является результатом вызова функции;

Перегрузка '+=': левому числу присваивается сумма левого и правого. Результатом является левое число после преобразований;

Перегрузка '-=': левому числу присваивается разность левого и правого. Результатом является левое число после преобразований;

Перегрузка '*=': левому числу присваивается произведение левого и правого. Результатом является левое число после преобразований;

Перегрузка '/=': левому числу присваивается частное от деления левого числа на правое. Результатом является левое число после преобразований;

Перегрузка '%=': левому числу присваивается остаток от деления левого числа на правое. Результатом является левое число после преобразований;

Перегрузка '–' (префиксная форма): левому числу присваивается разность левого числа и единичного объекта класса. Результатом является левое число после преобразований;

Перегрузка '++' (префиксная форма): левому числу присваивается сумма левого числа и единичного объекта класса. Результатом является левое число после преобразований;

Перегрузка '++' (постфиксная форма): левое число копируется во временный объект nov, затем префиксно уменьшается (operator++). В конце функция возвращает nov;

Перегрузка '–' (постфиксная форма): левое число копируется во временный объект nov, затем префиксно уменьшается (operator–). В конце функция возвращает nov;

Перегрузка '-' (унарная форма): создается объект result - полная копия исходного. Его знак меняется на противоположный. В конце функция возвращает result;

### 2.2.3 Логика:

Перегрузка -=": числа считаются равными, если равны их знаки (проверка на равенство поля sign) и модули (проверка на равенство поля number);

Перегрузка "!=": функция возвращает значение полученное при вызове перегрузки operator==, инвертируя его;

Перегрузка '>': сначала проверяются тривиальные случаи (сравнение чисел разных по знаку). Если числа имеют одинаковые знаки, то проверяется их длина, если длины совпадают, то итоговое решение выносится после поцифрового сравнения с помощью цикла;

Перегрузка '<': сначала проверяются тривиальные случаи (сравнение чисел разных по знаку). Если числа имеют одинаковые знаки, то проверяется их длина, если длины совпадают, то итоговое решение выносится после поцифрового сравнения с помощью цикла;

Перегрузка '>=': функция возвращает результат вызова функций operator> и operator==, к которым применяется логическое ИЛИ:

Перегрузка '<=': функция возвращает результат вызова функций operator< и operator==, к которым применяется логическое ИЛИ:

### 2.2.4 Остальные функции:

Функция isEven: сначала проверяется, чтобы строка-поле number было непустым. Затем берется последняя цифра числа и она проверяется на четность;

Функция DeleteZeros: в цикле с помощью функции `std::find` находится положение первого ненулевого элемента, а затем в поле number перезаписывается новым значением, которое

является подстрокой исходной строки. Если ненулевые элементы отсутствовали, то number становистя строкой, содержащей цифру 0;

Функция length: эта функция служит для укорочения записи и возвращает длину строки-поля number;

Функция absol: возвращает объект класса, сконструированный от поля number, то есть без учета знака;

Перегрузка -"для случая присваивания одному объекту другого: сначала проверяется, чтобы не было самоприсваиваиния. Если его нет, то полям одного объекта присваются соответсвенно значения полей другого объекта, иначе ничего не происходит;

Перегрузка '=' для случая, когда объекту присваивается строка: сначла проверяется, что строка не пуста. Если это так, то проверяется первый символ - если это знак минус, то поле sign становится true, а поле Number являетя подстрокой, начиная со второго символа. Иначе, знак числа обращается в false, а поле number является всей строкой-аргументом;

Функция power: сначала проверяются тривиальные случаи: если показатель степени равен 0, то функция возвращает единичный объект, если степень меньше 0, то функция возвращает нулевой элемент, так как класс предназначен для работы с целыми числами. Если же показатель положителен, то в цикле производится количество итераций, равных показателю степеню, в котором результирующему объекту присваивается этот же объект, умноженный на основание степени;

Транспонирование (функция Transpose()): сначала создается объект result - полная копия исходной матрицы. Затем в цикле для каждого коэффициента матрицы result производится обмен значений между номером строки и номером столбца с помощью функции стандартной библиотеки `std::swap(a,b)`. Затем с помощью той же функции происходит обмен значениями полей, отвечающих за количество строк и столбцов. В конце функция возвращает объект result;

Функция SetNumber: для исходного объекта устанавливается новое значение поля number;

Функция SetSign: функция принимает символ и устанавает Для исходного объекта новое значение поля sign;

Функция GetNumber: возвращает значение поля number;

Функция GetSign: возвращает значение поля sign ввиде символа;

Перегрузка приведения к базовому типу: функция возвращает строку, состоящую из числа, модуль которого соответсвует полю number, а знак - полю sign;

Перегрузка помещения в поток('«'): если число отрицательное, то в поток помещается символ знака, сконкатенированный с строкой-полем number, иначе в поток помещается поле number;

Перегрузка чтения из потока ('»'): сначала из потока считывается строка. Если она непуста, то объекту присваивается строка (используется перегрузка operator=).

## 2.3 Задача 3

### 2.3.1 Конструкторы и деструкторы:

Конструктор по умолчанию: полям indict, sun и moon присваюваются единичные значения;

Конструктор копирования: полям нового объекта соотвественно сопоставляются поля уже имеющегося объекта;

Конструктор с параметрами от одного целого числа и одного флага типа bool: по умолчанию flag имеет значение true. Если установлено значени false, то предполагается, что целое число - год от Рождества Христова. поэтому конвертируется с помощью необходимой функции в год "от Адама"и вызывается специальная вспомогательная функция ForConstrWithParam типа void от результата конвертации. Если знгачение flag не было изменено, то данная функция вызывается сразу от входного значения. ForConstWithParam работает аналогично конструктору с параметром от трех чисел, только остатки от деления ищутся от одного аргумента, означающего год. Если произошло такое, что пользователь ввел отрицательный год (либо он стал таким после

конвертации), то создается объект с "невозможными"значениями - все поля становятся нулями. Формально объект существует, но имеет поля, с которыми без их изменения работать смысла не имеет;

Деструктор: реализация доверена компилятору (в описании указана комбинация =default);

### 2.3.2 Арифметика:

Перегрузка '–' (префиксная форма): сначала создается объект от трех значений - уменьшенные на 1 поля исходного объекта. Затем исходному объекту присваивается новый и после присваивания исходная матрица возвращается из функции;

Перегрузка '++' (префиксная форма): сначала создается объект от трех значений - увеличенные на 1 поля исходного объекта. Затем исходному объекту присваивается новый и после присваивания исходная матрица возвращается из функции;

Перегрузка '++' (постфиксная форма): сначала создается объект ср - полная копия исходного. Затем поля исходного объекта увеличиваются на единицу и в конце возвращается ср;

Перегрузка '–' (постфиксная форма): сначала создается объект ср - полная копия исходного. Затем поля исходного объекта уменьшаются на единицу и в конце возвращается ср;

Перегрузка '+': создается переменная типа int, которой присваивается сумма года "от Адама"левого объекта и правого. В конце возвращается объект, который сконструирован по этой сумме;

Перегрузка '-': создается переменная типа int, которой присваивается модуль разности года "от Адама"левого объекта и правого. В конце возвращается объект, который сконструирован по этому модулю разности;

Перегрузка '+=': исходному объекту присваивается сумма исходного и переданного в качестве аргумента. В конце возвращается измененный исходный объект;

Перегрузка '-=': исходному объекту присваивается разность исходного и переданного в качестве аргумента. В конце возвращается измененный исходный объект;

### 2.3.3 Остальные функции:

Перегрузка -=": если поля левого объекта соответсвенно равны полям правого, то функция возвращзщает true, иначе false;

Перегрузка -"для случай присваивания одному объекту другого: сначала проверяется, чтобы не было самоприсваиваиния. Если его нет, то полям одного объекта присваются соответсвенно значения полей другого объекта, иначе ничего не происходит;

NormalFormatOut: устанавливает все флаги вывода (FormatYearJesusOut и FormatYearAdamOut) на значение false и возвращает ссылку на поток вывода;

YearAdamFormatOut: устанавливает флаг вывода FormatYearJesusOut на значение false, в флаг вывода FromatYearAdamOut на значение true и возвращает ссылку на поток вывода;

YearJesusFormatOut: устанавливает флаг вывода FormatYearJesusOut на значение true, в флаг вывода FromatYearAdamOut на значение false и возвращает ссылку на поток вывода;

NormalFormatIn: устанавливает все флаги ввода (FormatYearJesusIn и FormatYearAdamIn) на значение false и возвращает ссылку на поток ввода;

YearAdamFormatIn: устанавливает флаг ввода FormatYearJesusIn на значение false, в флаг ввода FromatYearAdamIn на значение true и возвращает ссылку на поток ввода;

YearJesusFormatIn: устанавливает флаг ввода FormatYearJesusIn на значение true, в флаг вывода FromatYearAdamIn на значение false и возвращает ссылку на поток ввода;

Перегрузка помещения в поток('«'): в зависимости от выставленных фалгов вывода объект выводится в поток по-разному (по умолчангию - три числа, если установлен флаг вывода в формате Года от "адама одно соответсвующее число, если установлен флаг вывода в формате года от Рождества Христова - одно соответсвующее число). Возвращает ссылку на поток

вывода;

Перегрузка чтения из потока ('»'): в зависимости от выставленных фалгов ввода объект из потока считывается либо одно число, либо 3. Ситуация по умолчанию - на вход поступает три числа, которые считываются, по ним создается новый объект, который впоследствие будет присвоен считываемому. Если установлен флаг ввода FormatJesusYearIn, то считывается одно число, создается новый объект через конструктора с параметром - считанным числом и вторым параметром false, который потом присваивается считываемому. Если установлен флаг FormatYearAdanьIn вводится одно число, создается новый объект от одного числа без флага, который потом присваивается считываемому. Возвращает ссылку на поток ввода;

Функция ComputeAdamYear: в функцию должно быть передано три целых числа. Сначала создается три целочисленные переменные, которые являются неотрицательным остатками от деления входных чисел. Затем высчитывается переменная цлочисленная переменная ans по определенной формуле, полученной путем математичкеского решения системы трех линейных сравнений по модулю. В конце если ans меньше нуля, то функция возвращает остаток при делении ans на 7980, увеличенной на 7980. Если ans>0, то функция возвращает остаток при делении ans на 7980;

Функция AdamToJesus: в функцию должно быть передано целое число, означающее количество лет "от Адама". Если оно неотрицательно, то функция возвращает число, уменьшенное на 5508, иначе -5508 (=0 "от Адама");

Функция JesusToAdam: в функцию должно быть передано целое число, означающее количество лет от Рождества Христова. Если это число, увеличенное на 5508 неотрицательно, то функция возвращает число, увеличенное на 5508, иначе 0;

Функция SetIndict: для исходного объекта устанавливается новое значение поля indict, равное неотрицательному остатку от деления аргумента на 15 (если остаток 0, то присваается 15);

Функция SetSun: для исходного объекта устанавливается новое значение поля sun, равное неотрицательному остатку от деления аргумента на 28 (если остаток 0, то присваается 28);

Функция SetMoon: для исходного объекта устанавливается новое значение поля moon, равное неотрицательному остатку от деления аргумента на 19 (если остаток 0, то присваается 19);

Функция GetIndict: возвращает значение поля indict;

Функция GetSun: возвращает значение поля sun;

Функция GetMoon: возвращает значение поля moon;

Операция приведения к базовому типу: функция превращает объект типа YearFromAdam в int, возвращая количество лет "от Адама"от полей конвертируемого объекта;

## 2.4 Задача 4

### 2.4.1 Конструкторы и деструкторы:

Конструктор по умолчанию: полям rows и cols присваюваются нулевые значения, вектор Coefs никак не заполняется;

Конуструктор копирования: полям нового объекта соотвественно сопоставляются поля уже имеющегося объекта;

Конструктор с параметрами: полю rows присваивается значение первого параметра, cols - второго, вектор остается незаполненным;

Деструктор: реализация доверена компилятору (в описании указана комбинация =default);

### 2.4.2 Арифметика:

Перегрузка '+': сначала проверяется, чтобы количество строк и столбцов обеих матриц было

одинаковым (иначе сложение невозможно). Затем создается новая матрица result в помощью конструктора с параметрами (от количества строк и столбцов исходных матриц). После этого происходит покоэффициентное сложение в цикле и помещения в result. Если в итоговой матрице появились нулевые коэффициенты, то их надо исключить, для этого используется еще один цикл, который перебирает все элементы вектора и удаляет нулевые. Функция возвращает матрицу result;

Перегрузка '-': сначала проверяется, чтобы количество строк и столбцов обеих матриц было одинаковым (иначе вычитание невозможно). Затем создается новая матрица result в помощью конструктора с параметрами (от количества строк и столбцов исходных матриц). После этого происходит покоэффициентное вычитание в цикле и помещения в result. Если в итоговой матрице появились нулевые коэффициенты, то их надо исключить, для этого используется еще один цикл, который перебирает все элементы вектора и удаляет нулевые. Функция возвращает матрицу result;

### 2.4.3 Остальные функции:

Перегрузка -=": сначала проверяется, чтобы количество строк и столбцов обеих матриц было одинаковым (иначе матрицы сразу считаются неравными). Затем создается две новые матрица first и second являющиеся полными копиями исходных (это необходимо, так как алгоритм предлагает изменение матриц, а менять исходные матрицы не является хорошим решением). После этого каждый элемент вектора первого объекта сопоставляет с соответсвенным эдементом второго объектах (для этого используем цикл в цикле). Если встречается два одинаковых элемента, то их номер строки становится равным -1 (учитывается, что для типа `std::size_t` это эквивалетно максимальному значению-1 и предполагается, что матрицы таких размеров использоваться не будут). Затем еще одним циклом проиходит перебор всех элементов вектора объекта first. Если количество элементов, номер строки которых равен (-1)u равно количеству элементов в векторе, то значит, что каждый элемент одного объекта сопоставим с каким=то элементом второго объекта. В таком случае матрицы можно назвать равными;

Перегрузка -"для случай присваивания одному объекту другого: сначала проверяется, чтобы не было самоприсваиваиния. Если его нет, то полям одного объекта присваются соответсвенно значения полей другого объекта, иначе ничего не происходит;

Перегрузка '*' для случая умножения на вещественное число: если конструкция Object * lambda: сначала создается новая матрица res - полная копия исходной. Затем каждый коэффциент этой матрицы умножается на lambda. В конце функция возвращает res. Если конструкция lambda*object, то такая функция возвращает конструкцию object*lambda;

Перегрузка '*' для перемножения двух объектов: сначала проверяется, чтобы количество столбцов левого объекта равнялось количеству строк правого (классический критерий перемножаемости матриц). Затем создается новый объект result от двух параметров: количество строк левой матрицы и столбцов правой. После этого происходит стандартное перемножение двух матриц. Дальше осуществляется в еще одном цикле очистка поля coefs на случай попадания в него нулевых значений. В конце функция возвращает объект result;

Транспонирование (функция Transpose()): сначала создается объект result - полная копия исходной матрицы. Затем в цикле для каждого коэффициента матрицы result производится обмен значений между номером строки и номером столбца с помощью функции стандартной библиотеки `std::swap(a,b)`. Затем с помощью той же функции происходит обмен значениями полей, отвечающих за количество строк и столбцов. В конце функция возвращает объект result;

Функция SetRows: для исходного объекта устанавливается новое значение поля rows и удаляются все элементы из вектора coefs, номер строки которых больше или равен новому общему количеству строк;

Функция SetCols: для исходного объекта устанавливается новое значение поля cols и удаляются все элементы из вектора coefs, номер столбца которых больше или равен новому общему

количеству столбцов;

Функция GetRows: возвращает значение поля rows;

Функция GetCols: возвращает значение поля cols;

Функция GetCoefs: возвращает вектор-поле coefs;

Перегрузка '()' в качестве rvalue: в качестве явных аргументов в функцию передается номер строки и столбца. Сначала эти значения проверяются на то, чтобы они были меньше, чем общее количество строк и столбцов в матрице. Затем создается нулевая переменная val, После этого в цикле ищется элемент с индексами, переданными в качестве аргументов. Если такой находится, то в переменную val записывается соответсвующее значение. В конце возвращается переменная val (если такое значение в векторе нашлось, то значением val, как и должно быть, является 0);

Перегрузка '()' в качестве lvalue: в качестве явных аргументов в функцию передается номер строки и столбца. Сначала эти значения проверяются на то, чтобы они были меньше, чем общее количество строк и столбцов в матрице. Затем в цикле ищется элемент с индексами, переданными в качестве аргументов. Если такой находится, то функция возвращает соответсвующее значение для перезаписи, если же такой элемент не находится, то создается новый с индексами-аргументами и возвращается он;

Перегрузка помещения в поток('«'): сначала в поток помещается через пробел количество строк и столбцов. Затем начиная с новой строки матрица выводится в привычном формате (таблицы). Между элементами строки - пробелы, при достижении последнего элемента строки, переход на новую строку. также написаны дополнительные условия, обеспеивающие отстутсвие пробела после последнего элемента строки и перехода на новую строку после последнего элемента в последней строке.

Перегрузка чтения из потока ('»'): сначала из потока должны быть прочитаны количество строк и столбцов, затем считываются элементы матрицы (предполагается, что изначально на входе указано полноценная матрица с нулями). Если считанное число равно нулю, то оно не вносится в вектор-поле, иначе - вносится.

# 3 Получение исполняемых модулей

Для получения исполняемых модулей main0, main1, main2, main3 была использована система сборки cmake - написан файл CMakeLists.txt.

Минимальная требуемая версия системы cmake - 3.12;

Флаги компиляции:

- -Wall (вывод всех предупреждений);
- -pedantic-errors (проверяет соответствие кода стандарту ISO C++, сообщает об использовании запрещённых расширений, считает все предупреждения ошибками);
- -fsanitize=undefined (санитайзер для неопределенного поведения);
- -std=c++20 (устанавливает стандарт языка);

Файлы из которых собирается исполняемый модуль

- main0 составляется из файлов main0.cpp DateAndTime.h DateAndTime.cpp;
- main1 составляется из файлов main1.cpp BigNumber.h BigNumber.cpp;
- main2 составляется из файлов main2.cpp YearFromAdam.h YearFromAdam.cpp;
- main3 составляется из файлов main3.cpp Matrix.h Matrix.cpp;

# 4 Тестирование

## 4.1 Тест №1

### 4.1.1 Проверка работоспособности конструктора по UnixTime

Создается объект Obj от числа 1618242582. В онлайн-калькуляторе проверено, что это соответсвует 18:49:52 12.04.2021. Это соответствие и проверяется;

### 4.1.2 Проверка работоспособности конструктора по FileTime

Создается объект Obj от числа 132627161920000000 и флага true. В онлайн-калькуляторе проверено, что это соответсвует 18:49:52 12.04.2021. Это соответствие и проверяется;

### 4.1.3 Проверка работоспособности конструктора копирования

Создается объект Obj от числа 1618242582. Затем создается объект Copy от Obj. В онлайн-калькуляторе проверено, что это соответсвует 18:49:52 12.04.2021. Это соответствие для Copy и проверяется;

### 4.1.4 Проверка работоспособности перегрузки '+'

Создается два объекта Obj1 и Obj2, создаваемые один от FileTime, другой - от UnixTime. Они складываются. В онлайн-калькуляторе проверено, что это соответсвует 12:47:59 23.07.2072. Это соответствие и проверяется;

### 4.1.5 Проверка работоспособности перегрузки '-' и метода duration

Создается два объекта Obj1 и Obj2, создаваемые один от FileTime, другой - от UnixTime. Они складываются. В онлайн-калькуляторе проверено, что это соответсвует 12:47:59 23.07.2072. Это соответствие и проверяется;

### 4.1.6 Проверка работоспособности перегрузки '«'

На протяжении предыдущих четырех тестов производился вывпод объекта в поток. Значит, данная перегрузка уже проверена, так как в противном случае предыдущие тесты не срабатывали бы;

### 4.1.7 Проверка работоспособности перегрузки '»'

В поток помещается два целых числа, затем создается два объекта Obj1 и Obj2, считывают из потока и проверяются на правильность считанного. В онлайн-калькуляторе проверено, что это соответсвует 18:27:14 12.04.2021 и 00:12:25 24.04.2021 соответсвенно. Это соответствие и проверяется;

### 4.1.8 Проверка работоспособности перегрузок '='

Создается три объекта Obj1, Obj2 и Obj3. Obj1 присваивается цлое число, означающее UnixTime, Obj2 присваивается целое число, предварительно сконвертировав из FileTime в UnixTime, Obj3 присваивается Obj2. В онлайн-калькуляторе проверено, что это соответсвует 20:58:07 12.04.2021, 20:58:07 12.04.2021 и 20:58:07 12.04.2021 соответсвенно. Это соответствие и проверяется;

### 4.1.9 Проверка работоспособности сеттеров (блок тестов 9-14)

В тестах проверяется правильность работы функций, устанавливающих значения для полей. Для достоверности при проверке функции установления месяца был проверен случай, когда устанавливается число, превышающее логически максимально-возможное. В таком случае объект не изменяется;

### 4.1.10 Проверка работоспособности приведения к базовому типу

Тест проверяет, что имея объект класса можно получить соответсвующее ему значение в базовом типе `std::uint64_t`;

### 4.1.11 Проверка работоспособности геттеров

Сначала создается объект с помощью конструктора от UnixTime, а затем с помощью геттеров получаются необходимые поля, которые сран=вниваются с заранее подготовленными значениями;

## 4.2 Тест №2

### 4.2.1 Проверка работоспособности всех конструкторов и перегрузки помещения в поток и из потока

В тесте создается несколько объектов с помощью различных конструкторов, чтобы проверить их все. Также некоторые объекты помещаются в поток, а некоторые считываются из него, позволяя проверить перегрузку операций «<"и »>";

### 4.2.2 Проверка работоспособности метода получения модуля числа и перегрузки логических операций

Создается два объекта, которые проверяются между собой на действие всевозможных операций сравнения и равенства. Также проверяется работа метода взятия модуля числа;

### 4.2.3 Проверка работоспособности арифметики

В тесте создается 3 объекта, между которыми происходят различные математические операции (с конвейером значений). Также проверяется работоспособность функции возведения в степень;

### 4.2.4 Проверка работоспособности приведения к базовому типу и проверки числа на четность

Тест проверяет, что имея объект класса можно получить соответсвующее ему значение в базовом типе `std::string`. Помимо этого проверяется работа функции определения четности числа;

### 4.2.5 Проверка работоспособности сеттеров и геттеров

Сначала создается 4 объекта. Затем в одном из них с помощью функции SetNumber устанавливается значение поля number, в другом с помощью функции SetSign - поля sign. В конце проверяется с помощью геттеров и конструкторов, что тест отработал верно;

## 4.3 Тест №3

### 4.3.1 Проверка работоспособности всех конструкторов

В тесте создается несколько объектов с помощью различных конструкторов, чтобы проверить

их все. С помощью геттеров проверяются поля на равенство заранее задуманным значениям;

### 4.3.2 Проверка работоспособности арифметических операций и присваивания

В тесте создается 3 объекта, между которыми происходят различные математические операции и операция присваивания (а также их комбинации ('+=' и '-=')).

### 4.3.3 Проверка работоспособности сеттеров и геттеров

Сначала создается 2 объекта. Затем в одном из них с помощью функций SetIndict, SetSun, SetMoon устанавливается значение соответсвующих полей. Также проверяется с помощью геттеров и конструкторов, что тест отработал верно;

### 4.3.4 Проверка работоспособности оператора приведения к базовому типу и функций подсчета и перевода годов

Тест проверяет, что имея объект класса можно получить соответсвующее ему значение в базовом типе int. Также создаются две переменные, в которые помещаются результаты конвертирования между разными типами годов;

### 4.3.5 Проверка работоспособности перегрузки операции помещения в поток и взятия из потока в различных формах

Тест проверяет все формы помещения объекта в поток и взятия из него, задействуя перегрузка операций '«' и '»' и функций задания формата ввода/вывода;

## 4.4 Тест №4

### 4.4.1 Проверка работоспособности конструкторов, геттеров и сеттеров

В тесте создается несколько объектов с помощью различных конструкторов, чтобы проверить их все. Проверка осуществляется с помощью геттеров. Затем с помощью сеттеров устанавливаются новые значения для некоторых объектов и затем проверяются на необходимые равенства;

### 4.4.2 Проверка работоспособности арифметики, транспонирования, присваивания и равенства

В тесте создается несколько объектов, между которыми происходят различные математические операции, функции транспонирования (также организованы через конвейер значений), операция равенства (два объекта проверяются между собой на равенство) и операция присваивания;

### 4.4.3 Проверка работоспособности операций помещения в поток и взятия из потока

В тесте создается 2 матрицы, одна из них помещается в поток, затем происходит проверка того, правильно ли произошло это действие. Затем в поток помещается 2 матрицы в виде таблицы значений и в одну строку. После этого из данного потока считывается два объекта класса. В конце происходит проверка. что все считалось верно;

# Приложение А

## А.1 Файл DateAndTime.h

```
#ifndef DateAndTime_h
#define DateAndTime_h

#include <iostream>

class DateAndTime
{
public:
  // Для" классов разработать необходимые конструкторы, деструктор, конструктор копирования"
  DateAndTime() = default;
  DateAndTime(std::uint64_t s, bool flag = false); /* преобразование из FileTime в DateAndTime */
  DateAndTime(const DateAndTime& obj);
  ~DateAndTime() = default;

  /*
  * Используя перегрузку операторов(operator) разработать стандартную арифметику объектов,
  * включающую арифметические действия над объектами
  * и стандартными типамицелыми(, вещественными, строками – в зависимости от вида объектов), присваивание, ввод и вывод в
  * стандартные потокииспользуя( операторы « << » и « >> »),
  */

  DateAndTime operator+ (const DateAndTime& right) const;
  DateAndTime operator+ (int right) const;
  friend DateAndTime operator+ (int left, const DateAndTime& right);
  std::int64_t operator- (const DateAndTime& right) const;
  friend std::ostream& operator<< (std::ostream& o, const DateAndTime& obj); // Вывод в формате hh:mi:ss dd.mm.yyyy
  friend std::istream& operator>> (std::istream& i, DateAndTime& obj); //Ввод в формате UnixTime
  DateAndTime& operator=(const DateAndTime& other);
  DateAndTime& operator=(std::uint64_t num);
  //а где DateAndTime + int ?
  //теперь есть, int+DateAndTime тоже

  //методы, обеспечивающие изменение отдельных составных частей объекта Сеттеры()
  void SetYear(std::uint64_t y);
  void SetMonth(std::uint64_t m);
  void SetDay(std::uint64_t d);
  void SetHour(std::uint64_t h);
  void SetMinute(std::uint64_t min);
  void SetSecond(std::uint64_t s);
  //а где геттеры"" ?
  //теперь есть

  std::uint64_t GetYear() const;
  std::uint64_t GetMonth() const;
```

```
45    std::uint64_t GetDay() const;
46    std::uint64_t GetHour() const;
47    std::uint64_t GetMinute() const;
48    std::uint64_t GetSecond() const;
49
50    /*
51    * Приведение" кот/ базового типа данных"
52    *
53    * Приведение от базового типа — конструктор DateAndTime(std::uint64_t s),
54    который приводит от UnixTime (std::uint64_t) к DateAndTime
55    *                и конструктор DateAndTime(std::uint64_t s, bool) который
       приводит от FileTime (std::uint64_t) к DateAndTime
56    * Приведение к базовому типу реализуем в ввиде перегрузки оператора приведения типа
       строка( 48) и соответсвующего метода строка( 49)
57    *
58    */
59
60    operator std::uint64_t(); // Приведение к UnixTime format
61    std::uint64_t ToFileTime() const; // Приведение к FileTime format
62
63    //======
64
65    //Чисто вспомогательные методы конвертации из Unix в File и обратно
66    //можно( использовать для операции взятия из потока и операции присваивания, так как
       для них перегрузка только для UnixTime)
67
68    static std::uint64_t UnixToFile(std::uint64_t Unix);
69    static std::uint64_t FileToUnix(std::uint64_t File);
70
71    //======
72
73    // Возможно полезная функция. Длительность в секундах
74    std::uint64_t duration(const DateAndTime& right) const;
75
76 private:
77    std::uint64_t year;
78    std::uint64_t month;
79    std::uint64_t day;
80    std::uint64_t hour;
81    std::uint64_t minute;
82    std::uint64_t second;
83    std::uint64_t UnixSec;  //зачем?
84    std::uint64_t FileTime; //--"--?
85 };
86
87 #endif
```

## A.2 Файл DateAndTime.cpp

```
1 #include "DateAndTime.h"
2
3 #include <ctime>
```

```cpp
#include <cmath>
#include <string>
#include <algorithm>
#include <vector>
#include <map>


static
std::vector<std::uint64_t> DayInMonths{ 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };
static
std::vector<std::uint64_t> DayInMonthsLeap{ 31, 29, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };

static
std::map<std::string, std::size_t> MonthNumber{ {"Jan",1}, {"Feb",2}, {"Mar",3}, {"Apr",4}, {"May",5}, {"Jun",6},
                        {"Jul",7}, {"Aug",8}, {"Sep",9}, {"Oct",10},{"Nov",11}, {"Dec",12} };

static
bool IsLeap(std::uint64_t year)
{
  return (((year % 4 == 0) && (year % 100 != 0)) || (year % 400 == 0)) ? true : false;
}

DateAndTime::DateAndTime(std::uint64_t s, bool flag)
{
  std::uint64_t s1 = s;
  if (flag)
    s1 = (s / 10000000 - 11644473600);

  UnixSec = s1;
  FileTime = (UnixSec + 11644473600) * 10000000;
  std::time_t res = s1;
  char* str = std::ctime(&res);

  std::string temp;
  temp.push_back(str[4]); temp.push_back(str[5]); temp.push_back(str[6]);

  this->month = MonthNumber[temp];
  temp.clear();

  temp.push_back(str[8]); temp.push_back(str[9]);
  this->day = std::stoi(temp);
  temp.clear();

  temp.push_back(str[11]); temp.push_back(str[12]);
  this->hour = std::stoi(temp);
  temp.clear();

  temp.push_back(str[14]); temp.push_back(str[15]);
```

```
52    this->minute = std::stoi(temp);
53    temp.clear();
54
55    temp.push_back(str[17]); temp.push_back(str[18]);
56    this->second = std::stoi(temp);
57    temp.clear();
58
59    temp.push_back(str[20]); temp.push_back(str[21]); temp.push_back(str[22]);
       temp.push_back(str[23]);
60    this->year = std::stoi(temp);
61    temp.clear();
62  }
63
64  DateAndTime::DateAndTime(const DateAndTime& obj)
65    :year{ obj.year }, month{ obj.month }, day{ obj.day },
66    hour{ obj.hour }, minute{ obj.minute }, second{ obj.second },
67    UnixSec{ obj.UnixSec }, FileTime{ obj.FileTime }
68  {}
69
70  DateAndTime DateAndTime::operator+ (const DateAndTime& right) const
71  {
72    return DateAndTime(this->UnixSec + right.UnixSec);
73  }
74
75  DateAndTime DateAndTime::operator+ (int right) const
76  {
77    return DateAndTime(this->UnixSec + right);
78  }
79
80  DateAndTime operator+ (int left, const DateAndTime& right)
81  {
82    return right + left;
83  }
84
85  std::int64_t DateAndTime::operator- (const DateAndTime& right) const
86  {
87    return (this->UnixSec - right.UnixSec);
88  }
89
90  std::ostream& operator<< (std::ostream& o, const DateAndTime& obj)
91  {
92    // Вывод в формате hh:mi:ss dd.mm.year
93
94    std::string hh("0"), mi("0"), ss("0"), dd("0"), mm("0"), yyyy(std::to_string
       (obj.year));
95    if (obj.hour < 10) { hh += std::to_string(obj.hour); }
96    else { hh = std::to_string(obj.hour); }
97
98    if (obj.minute < 10) { mi += std::to_string(obj.minute); }
99    else { mi = std::to_string(obj.minute); }
100
101   if (obj.second < 10) { ss += std::to_string(obj.second); }
102   else { ss = std::to_string(obj.second); }
```

```
103
104   if (obj.day < 10) { dd += std::to_string(obj.day); }
105   else { dd = std::to_string(obj.day); }
106
107   if (obj.month < 10) { mm += std::to_string(obj.month); }
108   else { mm = std::to_string(obj.month); }
109
110   o << hh << ':' << mi << ':' << ss << ' ' << dd << '.' << mm << '.' << yyyy;
111
112   return o;
113 }
114
115 std::istream& operator>> (std::istream& i, DateAndTime& obj)
116 {
117   i >> obj.UnixSec;
118   obj.FileTime = (obj.UnixSec + 11644473600) * 10000000;
119   std::time_t res = obj.UnixSec;
120   char* str = std::ctime(&res);
121
122   std::string temp;
123   temp.push_back(str[4]); temp.push_back(str[5]); temp.push_back(str[6]);
124
125   obj.month = MonthNumber[temp];
126   temp.clear();
127
128   temp.push_back(str[8]); temp.push_back(str[9]);
129   obj.day = std::stoi(temp);
130   temp.clear();
131
132   temp.push_back(str[11]); temp.push_back(str[12]);
133   obj.hour = std::stoi(temp);
134   temp.clear();
135
136   temp.push_back(str[14]); temp.push_back(str[15]);
137   obj.minute = std::stoi(temp);
138   temp.clear();
139
140   temp.push_back(str[17]); temp.push_back(str[18]);
141   obj.second = std::stoi(temp);
142   temp.clear();
143
144   temp.push_back(str[20]); temp.push_back(str[21]); temp.push_back(str[22]);
       temp.push_back(str[23]);
145   obj.year = std::stoi(temp);
146   temp.clear();
147
148   return i;
149 }
150
151 DateAndTime& DateAndTime::operator=(const DateAndTime& other)
152 {
153   this->day = other.day;
154   this->year = other.year;
```

```
155     this->month = other.month;
156     this->second = other.second;
157     this->minute = other.minute;
158     this->hour = other.hour;
159     this->UnixSec = other.UnixSec;
160     this->FileTime = other.FileTime;
161 
162     return (*this);
163 }
164 DateAndTime& DateAndTime::operator=(std::uint64_t num)
165 {
166     this->UnixSec = num;
167     this->FileTime = (this->UnixSec + 11644473600) * 10000000;
168 
169     std::time_t res = this->UnixSec;
170     char* str = std::ctime(&res);
171 
172     std::string temp;
173     temp.push_back(str[4]); temp.push_back(str[5]); temp.push_back(str[6]);
174 
175     this->month = MonthNumber[temp];
176     temp.clear();
177 
178     temp.push_back(str[8]); temp.push_back(str[9]);
179     this->day = std::stoi(temp);
180     temp.clear();
181 
182     temp.push_back(str[11]); temp.push_back(str[12]);
183     this->hour = std::stoi(temp);
184     temp.clear();
185 
186     temp.push_back(str[14]); temp.push_back(str[15]);
187     this->minute = std::stoi(temp);
188     temp.clear();
189 
190     temp.push_back(str[17]); temp.push_back(str[18]);
191     this->second = std::stoi(temp);
192     temp.clear();
193 
194     temp.push_back(str[20]); temp.push_back(str[21]); temp.push_back(str[22]);
      temp.push_back(str[23]);
195     this->year = std::stoi(temp);
196     temp.clear();
197 
198     return (*this);
199 }
200 
201 DateAndTime::operator std::uint64_t() { return UnixSec; }
202 
203 std::uint64_t DateAndTime::ToFileTime() const
204 {
205     return this->FileTime;
206 }
```

```
207
208 std::uint64_t DateAndTime::UnixToFile(std::uint64_t Unix)
209 {
210   return ((Unix + 11644473600) * 10000000);
211 }
212
213 std::uint64_t DateAndTime::duration(const DateAndTime& right) const
214 {
215   return std::llabs(*this - right);
216 }
217
218 std::uint64_t DateAndTime::FileToUnix(std::uint64_t File)
219 {
220   return (File / 10000000 - 11644473600);
221 }
222
223 void DateAndTime::SetYear(std::uint64_t y)
224 {
225
226   std::size_t sum{ 0 };
227   for (std::size_t i = std::min(this->year, y); i < std::max(this->year, y);
      ++i)
228   {
229     if (IsLeap(y)) sum += 366;
230     else sum += 365;
231   }
232   if (y < this->year) this->UnixSec -= sum * 24ull * 3600ull;
233   else this->UnixSec += sum * 24ull * 3600ull;
234
235   this->FileTime = (this->UnixSec + 11644473600) * 10000000;
236
237   this->year = y;
238
239 }
240 void DateAndTime::SetMonth(std::uint64_t m)
241 {
242   if (m <= 12)
243     if ((!IsLeap(this->year) && this->day <= DayInMonths[m - 1]) || (IsLeap(
      this->year) && this->day <= DayInMonthsLeap[m - 1]))
244     {
245       std::size_t sum{ 0 };
246       for (std::uint64_t i{ std::min(this->month,m) }; i < std::max(this->
      month, m); ++i)
247       {
248         if (IsLeap(this->year)) sum += DayInMonthsLeap[i - 1];
249         else sum += DayInMonths[i - 1];
250       }
251       //сейчас в sum хранится количество дней
252       if (m < this->month) this->UnixSec -= sum * 24ull * 3600ull;
253       else this->UnixSec += sum * 24ull * 3600ull;
254
255       this->FileTime = (this->UnixSec + 11644473600) * 10000000;
256
```

```
257         this->month = m;
258       }
259 }
260 void DateAndTime::SetDay(std::uint64_t d)
261 {
262   if ((IsLeap(this->year) && d <= DayInMonthsLeap[this->month - 1]) || (!
      IsLeap(this->year) && d <= DayInMonths[this->month - 1]))
263   {
264     std::int64_t diff = d - this->day;
265     this->day = d;
266     this->UnixSec += 24 * 3600 * diff;
267     this->FileTime = (this->UnixSec + 11644473600) * 10000000;
268   }
269 }
270 void DateAndTime::SetHour(std::uint64_t h)
271 {
272   if (h <= 23)
273   {
274     std::int64_t diff = h - this->hour;
275     this->hour = h;
276     this->UnixSec += 3600 * diff;
277     this->FileTime = (this->UnixSec + 11644473600) * 10000000;
278   }
279 }
280 void DateAndTime::SetMinute(std::uint64_t min)
281 {
282   if (min <= 59)
283   {
284     std::int64_t diff = min - this->minute;
285     this->minute = min;
286     this->UnixSec += 60 * diff;
287     this->FileTime = (this->UnixSec + 11644473600) * 10000000;
288   }
289 }
290 void DateAndTime::SetSecond(std::uint64_t s)
291 {
292   if (s <= 59)
293   {
294     std::int64_t diff = s - this->second;
295     this->second = s;
296     this->UnixSec += diff;
297     this->FileTime = (this->UnixSec + 11644473600) * 10000000;
298   }
299 }
300
301 std::uint64_t DateAndTime::GetYear() const
302 {
303   return this->year;
304 }
305 std::uint64_t DateAndTime::GetMonth() const
306 {
307   return this->month;
308 }
```

```
309 std::uint64_t DateAndTime::GetDay() const
310 {
311   return this->day;
312 }
313 std::uint64_t DateAndTime::GetHour() const
314 {
315   return this->hour;
316 }
317 std::uint64_t DateAndTime::GetMinute() const
318 {
319   return this->minute;
320 }
321 std::uint64_t DateAndTime::GetSecond() const
322 {
323   return this->second;
324 }
```

## A.3 Файл main0.cpp

```
1  #include "DateAndTime.h"
2  #include <sstream>
3  #include <string>
4  #include <cassert>
5
6  int main()
7  {
8    {
9      //test1. Проверка работоспособности конструктора по UnixTime
10     DateAndTime Obj(1618242592);
11
12     std::stringstream ss;
13     std::string str;
14     ss << Obj;
15     std::getline(ss, str);
16     bool correct1 = (str == "18:49:52 12.04.2021");
17     //Странно конечно что Дата" и время" у вас выводятся как Время" и дата"
18     //в винде вроде бы пишется сначала время, а потом какой день
19
20     assert(correct1);
21   }
22
23   {
24     //test2. Проверка работоспособности конструктора по FileTime
25     DateAndTime Obj(132627161920000000, true);
26
27     std::stringstream ss;
28     std::string str;
29     ss << Obj;
30     std::getline(ss, str);
31     bool correct2 = (str == "18:49:52 12.04.2021");
32
33     assert(correct2);
```

```
34    }
35
36    {
37      //test3. Проверка работоспособности конструктора копирования
38      DateAndTime Obj(1618250287);
39      DateAndTime Copy(Obj);
40
41      std::stringstream ss;
42      std::string str;
43      ss << Copy;
44      std::getline(ss, str);
45      bool correct3 = (str == "20:58:07 12.04.2021");
46
47      assert(correct3);
48    }
49
50    {
51      //test4. Проверка работоспособности перегрузки "+"
52      DateAndTime Obj1(132627161920000000, true);
53      DateAndTime Obj2(1618250287);
54
55      std::stringstream ss;
56      std::string str;
57      ss << (Obj1 + Obj2);
58      std::getline(ss, str);
59      bool correct4 = (str == "12:47:59 23.07.2072");
60
61      assert(correct4);
62    }
63
64    {
65      //test5. Проверка работоспособности перегрузки "-" и метода duration
66      DateAndTime Obj1(132627161920000000, true);
67      DateAndTime Obj2(1618250287);//1618242592
68
69      std::int64_t diff = (Obj1 - Obj2);
70      std::uint64_t dur = Obj1.duration(Obj2);
71
72      bool correct5 = (diff == -7695) && (dur == 7695);
73
74      assert(correct5);
75    }
76
77    {
78      //test6 Проверка работоспособности перегрузки "<<"
79
80      assert(true); //проверена на предыдущих 4 тестах
81    }
82
83    {
84      //test7. Проверка работоспособности перегрузки ">>" и Проверка работоспособности
       конструкций ostream << Obj1 << Obj2; istream >> Obj1 >> Obj2; композиция(
       операций помещения изв/ поток)
```

```
85
86      // Пусть есть 2 потока, в одному дано число в UnixTime, во втором в FileTime
87
88      std::uint64_t Unix = 1618241234;
89      std::uint64_t File = 132636859450000000;
90
91      std::stringstream ssin;
92      ssin << std::to_string(Unix) << ' ' << std::to_string(DateAndTime::
        FileToUnix(File));
93
94      DateAndTime Obj1, Obj2;
95
96      ssin >> Obj1 >> Obj2;
97
98      std::stringstream ssout;
99      ssout << Obj1 << '\n' << Obj2;
100
101     std::string str1, str2;
102     std::getline(ssout, str1);
103     std::getline(ssout, str2);
104     bool correct7 = (str1 == "18:27:14 12.04.2021") && (str2 == "00:12:25
        24.04.2021");
105
106     assert(correct7);
107   }
108
109   {
110     //test8. Проверка работоспособности перегрузок operator=(const DateAndTime&
        other); operator=(std::uint64_t num);
111     DateAndTime Obj1, Obj2, Obj3;
112
113     Obj1 = 1618250287; // UnixTime
114     Obj2 = DateAndTime::FileToUnix(132627238870000000); // FileTime
115     Obj3 = Obj2;
116
117     std::stringstream ss;
118     std::string str1, str2, str3;
119     ss << Obj1 << '\n' << Obj2 << '\n' << Obj3;
120     std::getline(ss, str1); std::getline(ss, str2); std::getline(ss, str3);
121     bool correct8 = (str1 == "20:58:07 12.04.2021") && (str2 == "20:58:07
        12.04.2021") && (str3 == "20:58:07 12.04.2021");
122
123     assert(correct8);
124   }
125
126   {
127     //test9. Проверка работоспособности SetYear
128     DateAndTime Obj1;
129
130     Obj1 = 1618250287; // UnixTime
131
132     std::stringstream ss;
133     std::string str1, str2;
```

```
134     ss << Obj1 << '\n';
135     Obj1.SetYear(2100);
136     ss << Obj1;
137 
138     std::getline(ss, str1); std::getline(ss, str2);
139     bool correct9 = (str1 == "20:58:07 12.04.2021") && (str2 == "20:58:07
        12.04.2100");
140 
141     assert(correct9);
142   }
143 
144   {
145     //test10. Проверка работоспособности SetMonth
146     DateAndTime Obj1;
147 
148     Obj1 = 1618250287; // UnixTime
149 
150     std::stringstream ss;
151     std::string str1, str2;
152     ss << Obj1 << '\n';
153     Obj1.SetMonth(1);
154     ss << Obj1;
155 
156     std::getline(ss, str1); std::getline(ss, str2);
157     bool correct10 = (str1 == "20:58:07 12.04.2021") && (str2 == "20:58:07
        12.01.2021");
158 
159     assert(correct10);
160   }
161 
162   {
163     //test11. Проверка работоспособности SetDay. В апреле 30 дней, поэтому SetDay(31)
        не должен менять объект
164     DateAndTime Obj1;
165 
166     Obj1 = 1618250287; // UnixTime
167 
168     std::stringstream ss;
169     std::string str1, str2;
170     ss << Obj1 << '\n';
171     Obj1.SetDay(31);
172     ss << Obj1;
173 
174     std::getline(ss, str1); std::getline(ss, str2);
175     bool correct11 = (str1 == "20:58:07 12.04.2021") && (str2 == "20:58:07
        12.04.2021");
176 
177     assert(correct11);
178   }
179 
180   {
181     //test12. Проверка работоспособности SetHour
182     DateAndTime Obj1;
```

```
183
184     Obj1 = 1618250287; // UnixTime
185
186     std::stringstream ss;
187     std::string str1, str2;
188     ss << Obj1 << '\n';
189     Obj1.SetHour(22);
190     ss << Obj1;
191
192     std::getline(ss, str1); std::getline(ss, str2);
193     bool correct12 = (str1 == "20:58:07 12.04.2021") && (str2 == "22:58:07
        12.04.2021");
194
195     assert(correct12);
196   }
197
198   {
199     //test13. Проверка работоспособности SetMinute. SetMinute(210) не должен менять
        объект
200     DateAndTime Obj1;
201
202     Obj1 = 1618250287; // UnixTime
203
204     std::stringstream ss;
205     std::string str1, str2;
206     ss << Obj1 << '\n';
207     Obj1.SetMinute(210);
208     ss << Obj1;
209
210     std::getline(ss, str1); std::getline(ss, str2);
211     bool correct13 = (str1 == "20:58:07 12.04.2021") && (str2 == "20:58:07
        12.04.2021");
212
213     assert(correct13);
214   }
215
216   {
217     //test14. Проверка работоспособности SetSecond
218     DateAndTime Obj1;
219
220     Obj1 = 1618250287; // UnixTime
221
222     std::stringstream ss;
223     std::string str1, str2;
224     ss << Obj1 << '\n';
225     Obj1.SetSecond(30);
226     ss << Obj1;
227
228     std::getline(ss, str1); std::getline(ss, str2);
229     bool correct14 = (str1 == "20:58:07 12.04.2021") && (str2 == "20:58:30
        12.04.2021");
230
231     assert(correct14);
```

```
232     }
233
234     {
235       //test15. Проверка работоспособности приведения к базовому типу
236       DateAndTime Obj1;
237
238       Obj1 = 1053418860; // UnixTime
239
240       std::uint64_t Unix = static_cast<std::uint64_t> (Obj1);
241       std::uint64_t File = Obj1.ToFileTime();
242
243       bool correct15 = (Unix == 1053418860) && (File == 126978924600000000);
244
245       assert(correct15);
246     }
247
248     {
249       //test16. Проверка работоспособности геттеров
250
251       DateAndTime obj(1618242592);
252
253       bool correct16 = (obj.GetHour() == 18 && obj.GetMinute() == 49 && obj.
      GetSecond() == 52
254         && obj.GetDay() == 12 && obj.GetMonth() == 04 && obj.GetYear() == 2021);
255
256       assert(correct16);
257     }
258
259     return 0;
260 }
```

# Приложение Б

## Б.1 Файл BigNumber.h

```
#include <string>
#include <iostream>

#ifndef BigNumber_H
#define BigNumber_H
class BigNumber
{
private:
  void DeleteZeros();
  std::size_t length() const;

  std::string number; //модуль числа
  bool sign = 0; // 0 - positive, 1 - negative

public:
  BigNumber() = default;
  ~BigNumber() = default;
  BigNumber(std::string source);
  BigNumber(long long source);
  BigNumber(const BigNumber& other);

  BigNumber absol() const;

  //Логика
  bool operator>  (const BigNumber& right) const;
  bool operator<  (const BigNumber& right) const;
  bool operator== (const BigNumber& right) const;
  bool operator!= (const BigNumber& right) const;
  bool operator>= (const BigNumber& right) const;
  bool operator<= (const BigNumber& right) const;

  //Арифметика
  BigNumber  operator+ (const BigNumber& right) const;
  BigNumber  operator- (const BigNumber& right) const;
  BigNumber  operator* (const BigNumber& right) const;
  BigNumber  operator/ (const BigNumber& right) const; // Деление нацело
  BigNumber  operator% (const BigNumber& right) const; // Остаток от деления, считаем, что, независимо от знаков чисел, остаток вычисляем
                                // как положительное число от модулей данных чисел (-13%5=3)
  BigNumber& operator+=(const BigNumber& right);
  BigNumber& operator-=(const BigNumber& right);
  BigNumber& operator*=(const BigNumber& right);
  BigNumber& operator/=(const BigNumber& right);
  BigNumber& operator%=(const BigNumber& right);
  BigNumber& operator= (const BigNumber& right);
  BigNumber& operator= (std::string NewNum);

  BigNumber power(const BigNumber& exp) const; //Возведение в степень если(
```

```
    показатель <0, то a^(-b) = 1/a^(-b) = 0, так как целые числа)
48
49    BigNumber& operator--(); //префиксный декремент
50    BigNumber& operator++(); //префиксный инкремент
51    BigNumber operator++(int); // постфиксный инкремент
52    BigNumber operator--(int); // постфиксный декремент
53
54    friend std::ostream& operator<< (std::ostream& out, const BigNumber& right);
55    friend std::istream& operator>> (std::istream& i, BigNumber& right);
56    friend BigNumber operator- (const BigNumber& right); //унарный минус
57
58    operator std::string();
59
60    //Сеттеры
61    void SetNumber(std::string s); //имеется ввиду именно задание поля number, а не
       задание всего числа типа BigNumber
62    void SetSign(char ch); //установка знака с помощью символов '+' и '-'
63
64    //Геттеры
65    std::string GetNumber() const; //получение поля number, а не всего числа типа
       BigNumber
66    char GetSign() const; //получение знака числа в виде символа '+' или '-'
67
68    bool isEven() const; //проверка четности числа
69 };
70
71 #endif
```

## Б.2 Файл BigNumber.cpp

```
1  #include "BigNumber.h"
2  #include <algorithm>
3  #include <vector>
4  #include <cmath>
5
6  BigNumber::operator std::string()
7  {
8    if (this->sign == 1) return ('-' + this->number);
9    else return this->number;
10 }
11
12 void BigNumber::SetNumber(std::string s)
13 {
14   this->number = s;
15 }
16
17 void BigNumber::SetSign(char ch)
18 {
19   if (ch == '+')
20     this->sign = 0;
21   if (ch == '-')
22     this->sign = 1;
```

```
23 }
24
25 std::string BigNumber::GetNumber() const
26 {
27   return this->number;
28 }
29
30 char BigNumber::GetSign() const
31 {
32   if (this->sign == false)
33     return '+';
34   else
35     return '-';
36 }
37
38 bool BigNumber::isEven() const
39 {
40   if (this->number.size() != 0)
41   {
42     int lastdigit = this->number[this->length() - 1] - '0';
43     return !(lastdigit % 2);
44   }
45 }
46
47 void BigNumber::DeleteZeros()
48 {
49   std::size_t pos{ 0 };
50   bool flag = true;
51
52   for (const auto& c : this->number)
53   {
54     if (c != '0')
55     {
56       pos = std::find(this->number.begin(), this->number.end(), c) - this->
    number.begin();
57       flag = false;
58       break;
59     }
60   }
61   if (!flag)
62     this->number = this->number.substr(pos, this->number.length() - pos);
63   else this->number = "0";
64 }
65
66 std::size_t BigNumber::length() const
67 {
68   return this->number.length();
69 }
70
71 BigNumber BigNumber::absol() const
72 {
73   return BigNumber(this->number);
74 }
```

```
75
76 BigNumber::BigNumber(std::string source)
77 {
78   if (source.size() != 0)
79   {
80     if (source[0] == '-')
81     {
82       this->sign = 1;
83       this->number = source.substr(1, source.length() - 1);
84     }
85     else {
86       this->number = source;
87     }
88
89     this->DeleteZeros();
90   }
91 }
92
93 BigNumber::BigNumber(long long source)
94 {
95   if (source < 0)
96     sign = 1;
97
98   number = std::to_string(std::llabs(source));
99 }
100
101 BigNumber::BigNumber(const BigNumber& other)
102   : number{ other.number }, sign{ other.sign }
103 {}
104
105 bool BigNumber::operator> (const BigNumber& right) const
106 {
107   if (this->sign == 1 && right.sign == 0) return false;
108   if (this->sign == 0 && right.sign == 1) return true;
109   if (this->sign == 0 && right.sign == 0)
110   {
111     if (this->length() > right.length()) return true;
112     if (this->length() < right.length()) return false;
113     if (this->length() == right.length())
114     {
115       for (std::size_t i{ 0 }; i < this->length(); ++i)
116       {
117         if (this->number[i] > right.number[i]) return true;
118         if (this->number[i] < right.number[i]) return false;
119       }
120       return false;
121     }
122   }
123
124   if (this->sign == 1 && right.sign == 1)
125   {
126     if (this->length() > right.length()) return false;
127     if (this->length() < right.length()) return true;
```

```
128     if (this->length() == right.length())
129     {
130       for (std::size_t i{ 0 }; i < this->length(); ++i)
131       {
132         if (this->number[i] > right.number[i]) return false;
133         if (this->number[i] < right.number[i]) return true;
134       }
135       return false;
136     }
137   }
138   return false;
139 }
140
141 bool BigNumber::operator<(const BigNumber& right) const
142 {
143   if (this->sign == 1 && right.sign == 0) return true;
144   if (this->sign == 0 && right.sign == 1) return false;
145   if (this->sign == 0 && right.sign == 0)
146   {
147     if (this->length() > right.length()) return false;
148     if (this->length() < right.length()) return true;
149     if (this->length() == right.length())
150     {
151       for (std::size_t i{ 0 }; i < this->length(); ++i)
152       {
153         if (this->number[i] > right.number[i]) return false;
154         if (this->number[i] < right.number[i]) return true;
155       }
156       return false;
157     }
158   }
159   if (this->sign == 1 && right.sign == 1)
160   {
161     if (this->length() > right.length()) return true;
162     if (this->length() < right.length()) return false;
163     if (this->length() == right.length())
164     {
165       for (std::size_t i{ 0 }; i < this->length(); ++i)
166       {
167         if (this->number[i] > right.number[i]) return true;
168         if (this->number[i] < right.number[i]) return false;
169       }
170       return false;
171     }
172   }
173
174   return false;
175 }
176
177 bool BigNumber::operator==(const BigNumber& right) const
178 {
179   if (this->sign == right.sign && this->number == right.number)
180     return true;
```

```
181   return false;
182 }
183
184 bool BigNumber::operator!=(const BigNumber& right) const
185 {
186   return !(*this == right);
187 }
188
189 bool BigNumber::operator>=(const BigNumber& right) const
190 {
191   return (*this > right || *this == right);
192 }
193
194 bool BigNumber::operator<=(const BigNumber& right) const
195 {
196   return (*this < right || *this == right);
197 }
198
199 BigNumber BigNumber::operator+ (const BigNumber& right) const
200 {
201   BigNumber result;
202
203   if (this->sign == 1 && right.sign == 0) return (right.absol() - this->absol
      ()); //-x+y = y-x
204   if (this->sign == 0 && right.sign == 1) return (this->absol() - right.absol
      ());    // x-y
205   if (this->sign == 1 && right.sign == 1)                 // -x-y = -(x+y)
206   {
207     /*BigNumber first(*this), second(right);
208     first.sign = 0; second.sign = 0;*/
209     result = this->absol() + right.absol();
210     result.sign = 1;
211     return result;
212   }
213
214   if (this->sign == 0 && right.sign == 0)
215   {
216     int diff = this->length() - right.length();
217     std::string ImZero(std::abs(diff)+1, '0');  //xxxxxx + 000yyy
218
219     BigNumber first("0"), second;
220     if (*this > right) {
221       first.number += this->number;
222       second = ImZero + right.number;
223     }
224     else {
225       first.number += right.number;
226       second = ImZero + this->number;
227     }
228     std::string stroka(first.length() + 1, '0');
229
230     result = stroka;
231
```

```
232     //result.number[result.length() - 1] = static_cast<char>((((first.number[
        this->length() - 1] - '0') + (second.number[this->length() - 1] - '0'))) %
        10 + '0');
233
234     for (std::size_t  i =  first.length(); i > 0; --i)
235     {
236       int digit1 = (first.number[i-1] - '0');
237       int digit2 = (second.number[i-1] - '0');
238       int cnt = i-2;
239       //int olddigit1 = (first.number[i + 1] - '0');
240       //int olddigit2 = (second.number[i + 1] - '0');
241
242       if (digit1 + digit2 >= 10)
243       {
244         result.number[i] = (digit1 + digit2) % 10 + '0';
245         if ((first.number[i-2]-'0') + 1 == 10)
246         {
247           while ((first.number[cnt] + 1 - '0') == 10)
248           {
249             first.number[cnt] = '0';
250             --cnt;
251           }
252           first.number[cnt] = (first.number[cnt] - '0') + 1 + '0';
253         }
254         else { first.number[i-2] += 1; }
255       }
256       else { result.number[i] = digit1 + digit2 + '0'; }
257
258       //result.number[i+1] = static_cast<char>(((digit1 + digit2 + (olddigit1
        + olddigit2) / 10) % 10) + '0');
259     }
260
261     result.number[0] = ((first.number[0]-'0') + (second.number[0]-'0')) / 10 +
        '0';
262
263     /*std::size_t pos{ 0 };
264
265     for (const auto& c : result.number)
266     {
267       if (c != '0')
268       {
269         pos = std::find(result.number.begin(), result.number.end(), c) -
        result.number.begin();
270         break;
271       }
272     }
273
274     result.number = result.number.substr(pos, result.number.length() - pos);*/
275
276     result.DeleteZeros();
277     return result;
278   }
279   return result;
```

```
280 }
281 BigNumber  BigNumber::operator- (const BigNumber& right) const
282 {
283   BigNumber result;
284   if (this->sign == 0 && right.sign == 1) //x-(-y) = x+y
285   {
286     return this->absol() + right.absol();
287   }
288   if (this->sign == 1 && right.sign == 0) //-x-y = -(x+y)
289   {
290     result = this->absol() + right.absol();
291     result.sign = 1;
292     return result;
293   }
294   if (this->sign == 1 && right.sign == 1) //-x-(-y) = y-x
295   {
296     return right.absol() - this->absol();
297   }
298
299   if (this->sign == 0 && right.sign == 0)
300   {
301     if (right > *this)
302     {
303       result = right.absol() - this->absol();
304       result.sign = 1;
305       return result;
306     }
307     else
308     {
309       std::string zeros(this->length(), '0');
310       result.number = zeros;
311       int diff = this->length() - right.length();
312       std::string lmZero(std::abs(diff), '0');
313       std::string stroka = lmZero + right.number;
314       BigNumber second, first(*this);
315       second.number = stroka;
316
317       for (int i = this->length() - 1 ; i >= 0; --i)
318       {
319         int olddigit1 = first.number[i] - '0';
320         int olddigit2 = second.number[i] - '0';
321         int cnt = i - 1;
322
323         if (olddigit1 < olddigit2)
324         {
325           result.number[i] = (10 + olddigit1 - olddigit2) + '0';
326           if (this->number[i - 1] != '0')
327           {
328             first.number[i-1] = (first.number[i - 1] - '0') - 1 + '0';
329           }
330           else {
331             while (first.number[cnt] == '0')
332             {
```

```
333                 first.number[cnt] = '9';
334                 --cnt;
335               }
336               --first.number[cnt];
337             }
338           }
339           else { result.number[i] = olddigit1 - olddigit2 + '0'; }
340         }
341
342         result.DeleteZeros();
343       }
344
345     }
346     return result;
347 }
348 BigNumber  BigNumber::operator* (const BigNumber& right) const
349 {
350     BigNumber first(*this), second(right);
351     std::reverse(first.number.begin(), first.number.end());
352     std::reverse(second.number.begin(), second.number.end());
353     //std::string zeros(this->length() + right.length() + 1, '0');
354     BigNumber result;
355     //result.number = zeros;
356
357     /*std::int64_t diff = this->length() - right.length();
358     std::string addzeros(std::abs(diff), '0');
359
360     if (diff < 0)
361       first.number = addzeros + first.number;
362     if (diff > 0)
363       second.number = addzeros + second.number;*/
364
365     // изза переполнения char'a при данном алгоритме перейдем от string к vector<
        unsigned short>
366
367     std::vector<unsigned short> firstVec, secondVec, resultVec(this->length() +
        right.length() + 1);
368
369     for (const auto& c : first.number)
370       firstVec.push_back(c - '0');
371     for (const auto& c : second.number)
372       secondVec.push_back(c - '0');
373
374
375     for (std::size_t i{0}; i < first.length(); ++i) {
376       for (std::size_t j{ 0 }; j < second.length(); ++j) {
377         resultVec[i + j] += firstVec[i] * secondVec[j];
378       }
379     }
380
381     std::reverse(resultVec.begin(), resultVec.end());
382
383     for (std::size_t i{ resultVec.size()-1 }; i > 0; --i)
```

```
384   {
385     long digit = resultVec[i];
386     if (digit >= 10)
387     {
388       resultVec[i] = digit % 10;
389       resultVec[i - 1] += digit / 10;
390     }
391   }
392
393   for (const auto& v : resultVec)
394     result.number += (v + '0');
395
396   result.DeleteZeros();
397   result.sign = (this->sign ^ right.sign) ? 1 : 0;
398
399   return result;
400 }
401
402
403 BigNumber  BigNumber::operator/ (const BigNumber& right) const
404 {
405   if (this->absol() < right.absol()) return BigNumber(0);
406
407   BigNumber delimoe(this->absol());
408   BigNumber cnt;
409
410   std::size_t j = 0;
411
412   //какойто- цикл
413   while(this->absol() - right.absol()*cnt >= right.absol())
414   //for (std::size_t i{ 0 }; i < delimoe.length(); ++i)
415   {
416     BigNumber delitel(right.absol());
417     BigNumber temp(0);
418
419     if (delimoe < delitel)
420     {
421       cnt.number += '0';
422       ++j;
423       continue;
424     }
425
426     std::string zeros(delimoe.length() - delitel.length() - j, '0');
427     delitel.number += zeros;
428
429     while (delimoe >= delitel)
430     {
431       delimoe -= delitel;
432       temp += 1;
433     }
434
435     cnt.number += temp.number;
436
```

```
437   }
438
439   cnt.DeleteZeros();
440   cnt.sign = (this->sign ^ right.sign) ? 1 : 0;
441   return cnt;
442 }
443
444 BigNumber BigNumber::operator% (const BigNumber& right) const
445 {
446   BigNumber result;
447   if (this->absol() < right.absol())
448     return (*this);
449
450   BigNumber Antje(this->absol() / right.absol());
451
452   result = this->absol() - Antje * right.absol();
453
454   return result;
455 }
456
457 BigNumber& BigNumber::operator+=(const BigNumber& right)
458 {
459   *this = *this + right;
460   return (*this);
461 }
462
463 BigNumber& BigNumber::operator-=(const BigNumber& right)
464 {
465   *this = *this - right;
466   return (*this);
467 }
468
469 BigNumber& BigNumber::operator*=(const BigNumber& right)
470 {
471   *this = *this * right;
472   return (*this);
473 }
474
475 BigNumber& BigNumber::operator/=(const BigNumber& right)
476 {
477   *this = *this / right;
478   return (*this);
479 }
480
481 BigNumber& BigNumber::operator%=(const BigNumber& right)
482 {
483   *this = *this % right;
484   return (*this);
485 }
486
487
488 BigNumber& BigNumber::operator= (const BigNumber& right)
489 {
```

```
490   if (this != &right)
491   {
492     this->number = right.number;
493     this->sign = right.sign;
494     return (*this);
495   }
496 }
497 BigNumber& BigNumber::operator= (std::string NewNum)
498 {
499   if (NewNum.size() != 0)
500   {
501     if (NewNum[0] == '-')
502     {
503       this->sign = 1;
504       this->number = NewNum.substr(1, NewNum.length() - 1);
505     }
506     else {
507       this->number = NewNum;
508       this->sign = 0;
509     }
510     return (*this);
511   }
512 }
513
514 BigNumber BigNumber::power(const BigNumber& exp) const
515 {
516   if (exp == 0) return BigNumber(1);
517   if (exp <  0) return BigNumber(0);
518
519   BigNumber result("1");
520   for (BigNumber i("0"); i < exp; ++i)
521   {
522     result *= (*this);
523   }
524
525   return result;
526 }
527
528 BigNumber& BigNumber::operator--()
529 {
530   BigNumber dec("1");
531   *this = *this - dec;
532   return *this;
533 }
534
535 BigNumber& BigNumber::operator++()
536 {
537   BigNumber inc(1);
538
539   *this = *this + inc;
540   return *this;
541 }
542
```

```
543 BigNumber BigNumber::operator++(int)
544 {
545   BigNumber nov(*this);
546   ++(*this);
547   return nov;
548 }
549
550 BigNumber BigNumber::operator--(int)
551 {
552   BigNumber nov(*this);
553   --(*this);
554   return nov;
555 }
556
557 std::ostream& operator<< (std::ostream& out, const BigNumber& right)
558 {
559   if (right.sign == 1)
560     out << "-" << right.number;
561   else
562   {
563     out << right.number;
564   }
565
566   return out;
567
568 }
569
570 std::istream& operator>> (std::istream& i, BigNumber& right)
571 {
572   std::string source;
573   i >> source;
574   if (source.size() != 0)
575   {
576     /*if (source[0] == '-')
577     {
578       right.sign = 1;
579       right.number = source.substr(1, source.length() - 1);
580     }
581     else { right.number = source; }*/
582     right = source;
583   }
584   return i;
585 }
586
587 BigNumber operator-(const BigNumber& right)
588 {
589   BigNumber result(right);
590   result.sign = !right.sign;
591   return result;
592 }
```

## Б.3 Файл main1.cpp

```
1  // TODO: заняться тестированием и написанием main
2
3  #include "BigNumber.h"
4  #include <iostream>
5  #include <cassert>
6  #include <sstream>
7
8  int main()
9  {
10     { //Test1. Проверка работоспособности всех конструкторов и перегрузки помещения в
       поток и из потока
11         std::stringstream streamout, streamin;
12
13         BigNumber chislo("-9876543212344566789");
14
15         streamout << BigNumber("-00123") << ' ' << BigNumber
       (1234567899876543 21) << ' ' << BigNumber(chislo);
16
17         std::string strokaout, strokain("123456789 -998877665544332211 0");
18         std::getline(streamout, strokaout);
19
20         streamin << strokain;
21
22         BigNumber chislo1, chislo2, chislo3;
23
24         streamin >> chislo1 >> chislo2 >> chislo3;
25
26         bool correct1 = (strokaout == "-123 1234567899876543 21
       -9876543212344566789")
27                     && (chislo1 == BigNumber(123456789))
28                     && (chislo2 == BigNumber("-998877665544332211"))
29                     && (chislo3 == BigNumber(0));
30
31         assert(correct1);
32     }
33
34     { //Test2. Проверка работоспособности метода получения модуля числа и перегрузки
       логики
35
36         BigNumber one("-123456765456754"), two("0012321");
37
38         bool correct2 = ( one.absol() == BigNumber("123456765456754") ) && (
       two != one ) && ( one<two ) &&
39                         ( one.absol() > two )                                  && (
       one <= two ) && ( !(one >= two) );
40
41         assert(correct2);
42     }
43
44   { //Test3. Проверка работоспособности арифметики
45
46         BigNumber one(1), two("-99999999999999999999999"), three("1234567");
47
```

```
48      bool correct3p1 = ( two - one == BigNumber("-10000000000000000000000000
    ") ) &&
49                        ( two.absol()+one == BigNumber("
    100000000000000000000000") ) &&
50                        ( three * (two - one) / (two.absol() + one) ==
    BigNumber(-1234567) ) &&
51                        ( (three+one) % three == BigNumber("1") );
52
53      bool correct3p2 = ( ++two == BigNumber(("-99999999999999999999998"))
    )
54                      && ( three++ == BigNumber(1234567) && three ==
    BigNumber(1234568) )
55                      && ( --one == BigNumber(0) )
56                      && ( one-- == BigNumber(0) && (one == BigNumber(-1)) );
57
58      BigNumber a, b("1111111111"), c("-2222222222"), d("3333333333"), e("
    -4444444444"), f("5555555555"), g, i(e);
59
60      a =  "-1010101010";
61      b += c;
62      c -= e;
63      d /= b;
64      e *= a;
65      f %= i;
66      g =  b;
67
68      bool correct3p3 =  (a == BigNumber("-1010101010"))
69                      && (b == BigNumber("-1111111111"))
70                      && (c == BigNumber("2222222222"))
71                      && (d == BigNumber(-3))
72                      && (e == BigNumber("4489337821773288440"))
73                      && (f == BigNumber("1111111111"))
74                      && (g == BigNumber("-1111111111"));
75
76      one = "-123456789";
77      two = "11";
78      BigNumber zero(0);
79
80      bool correct3p4 = (one.power(two) == BigNumber("
    -10154645082248316311927502100842088153631659353603201829685466296589
81      26840425122358052341 89"))
82                      && (two.power(one) == BigNumber(0))
83                      && (two.power(zero) == BigNumber(1));
84
85      two = "-11111111111";
86
87      bool correct3p5 = (-one == BigNumber(123456789))
88                      && (-(one*two) == BigNumber("-1371742099986282579"));
89
90      bool correct3 = correct3p1 && correct3p2 && correct3p3 && correct3p4
    && correct3p5;
91
92      assert(correct3);
```

```
93    }
94
95    { //Test4. Проверка работоспособности приведения к базовому типу и проверки числа на
       четность
96
97        BigNumber chislo1("-1234567899876543210"), chislo2("998877665544332211")
       ;
98
99        std::string NewChislo = static_cast<std::string>(chislo1) + static_cast<
       std::string>(chislo2);
100
101       BigNumber res(NewChislo);
102
103       bool correct4 = (res == BigNumber("
       -123456789987654321099887765544332211"))
104                   && (res.isEven() == false)
105                   && (chislo1.isEven() == true);
106
107       assert(correct4);
108   }
109
110   { //Test5. Проверка работоспособности сеттеров и геттеров
111
112       BigNumber chislo1("111222333444555666777888999"),
113                 chislo2("-12345678901234567890123456789O"),
114                 chislo3("-10203040506070809090807060504030201O"),
115                 chislo4("
       1000000000000000000000000000000000000000000000000000000000000000000000000000000000
116                 000000000000");
117
118       chislo3.SetNumber("
       999999999999999999888888888888888888888887777777777777777333333333333312
       ");
119       chislo4.SetSign('-');
120
121       bool correct5 = (chislo1.GetSign() == '+')
122                   && (chislo2.GetNumber() == "12345678901234567890123456789O"
       )
123                   && (chislo3 == BigNumber("
       -999999999999999999888888888888888888888887777777777777777333333333
124                    3333312"))
125                   && (chislo4 == BigNumber("
       -100000000000000000000000000000000000000000000000000000000000000000000000000000000
126                    0000000000000000"));
127
128       assert(correct5);
129   }
130
131    return 0;
132 }
```

# Приложение B

## B.1 Файл YearFromAdam.h

```
#ifndef YearFromAdam_H
#define YearFromAdam_H

/*
* Считается, что год от Рождества Христова может быть меньше нуля, а год от Адама (= год от сотворения мирв) — нет
*/

#include <iostream>

class YearFromAdam
{
private:
  void ForConstrWithParam(int year); // Чтобы избежать копирование кода в конструкторе с параметром


  //Флаги для ввода и вывода
  static bool FormatYearAdamOut;
  static bool FormatYearJesusOut;
  static bool FormatYearAdamIn;
  static bool FormatYearJesusIn;

  int indict;
  int sun;
  int moon;
public:
  YearFromAdam();
   ~YearFromAdam() = default;
    YearFromAdam(int ind, int sol, int lon);
    YearFromAdam(int year, bool Adam = true);
    YearFromAdam(const YearFromAdam& other);

  YearFromAdam& operator--();
  YearFromAdam& operator++();
  YearFromAdam  operator++(int);
  YearFromAdam  operator--(int);

  YearFromAdam& operator= (const YearFromAdam& right);

  YearFromAdam operator+(const YearFromAdam& right) const;
  YearFromAdam operator-(const YearFromAdam& right) const;

  YearFromAdam& operator+=(const YearFromAdam& right);
  YearFromAdam& operator-=(const YearFromAdam& right);

  bool operator==(const YearFromAdam& right) const;

  static std::ostream& NormalFormatOut(std::ostream& stream);      // Флаг
```

```
    вывода в виде 3 чисел полей() по( умолчанию)
48  static std::ostream& YearAdamFormatOut(std::ostream& stream);    // Флаг
    вывода в виде года от Адама
49  static std::ostream& YearJesusFormatOut(std::ostream& stream);   // Флаг
    вывода в виде года от РХ..
50  static std::istream& NormalFormatIn(std::istream& stream);     // Флаг ввода
    в виде 3 чисел полей() по( умолчанию)
51  static std::istream& YearAdamFormatIn(std::istream& stream);   // Флаг ввода
    в виде года от Адама
52  static std::istream& YearJesusFormatIn(std::istream& stream); // Флаг ввода
    в виде года от РХ..
53
54  friend std::ostream& operator<< (std::ostream& o, const YearFromAdam& object
    ); // по умолчанию — вывод трех чисел меняется( флагами)
55  friend std::istream& operator>> (std::istream& in, YearFromAdam& object); //
      по умолчанию — вывод трех чисел меняется( флагами)
56
57  static int ComputeAdamYear(int ind, int sol, int lon);
58
59  static int AdamToJesus(int YearInAdam);
60  static int JesusToAdam(int YearInJesus);
61
62  //Геттеры
63  int GetIndict() const;
64  int GetSun() const;
65  int GetMoon() const;
66
67  //Сеттеры
68  void SetIndict(int ind);
69  void SetSun(int sol);
70  void SetMoon(int lon);
71
72  operator int(); // Возвращает год от" Адама" соответсвующий объекту
73 };
74
75 #endif //YearFromAdam
```

## B.2 Файл YearFromAdam.cpp

```
1  #include "YearFromAdam.h"
2
3  YearFromAdam::YearFromAdam()
4    :indict{ 1 }, sun{ 1 }, moon{ 1 }
5  {}
6
7  YearFromAdam::YearFromAdam(int ind, int sol, int lon)
8  {
9    ind %= 15; sol %= 28; lon %= 19;
10
11   indict = ind ? ind : 15;
12   sun = sol ? sol : 28;
13   moon = lon ? lon : 19;
```

```
14
15    if (indict < 0) indict += 15;
16    if (sun < 0) sun += 28;
17    if (moon < 0) moon += 19;
18
19  }
20
21  void YearFromAdam::ForConstrWithParam(int year)
22  {
23    if (year > 0)
24    {
25      int ind = year % 15, sol = year % 28, lon = year % 19;
26
27      indict = ind ? ind : 15;
28      sun = sol ? sol : 28;
29      moon = lon ? lon : 19;
30    }
31    else {
32      indict = 0;
33      sun = 0;
34      moon = 0;
35    }
36  }
37
38  YearFromAdam::YearFromAdam(int year, bool Adam)
39  {
40    if (!Adam)
41      year = YearFromAdam::JesusToAdam(year);
42
43    ForConstrWithParam(year);
44  }
45
46  YearFromAdam::YearFromAdam(const YearFromAdam& other)
47    :indict{ other.indict }, sun{ other.sun }, moon{ other.moon }
48  {}
49
50  YearFromAdam& YearFromAdam::operator--()
51  {
52    YearFromAdam temp(--(this->indict), --(this->sun), --(this->moon));
53    *this = temp;
54    return *this;
55  }
56
57  YearFromAdam& YearFromAdam::operator++()
58  {
59    YearFromAdam temp(++(this->indict), ++(this->sun), ++(this->moon));
60    *this = temp;
61    return *this;
62  }
63
64  YearFromAdam YearFromAdam::operator++(int)
65  {
66    YearFromAdam cp(*this);
```

```
67    ++(this->indict);
68    ++(this->sun);
69    ++(this->moon);
70
71    return cp;
72  }
73
74  YearFromAdam YearFromAdam::operator--(int)
75  {
76    YearFromAdam cp(*this);
77    --(this->indict);
78    --(this->sun);
79    --(this->moon);
80
81    return cp;
82  }
83
84  YearFromAdam& YearFromAdam::operator= (const YearFromAdam& right)
85  {
86    if (!(*this == right))
87    {
88      this->indict = right.indict;
89      this->sun = right.sun;
90      this->moon = right.moon;
91      return (*this);
92    }
93  }
94
95  YearFromAdam YearFromAdam::operator+(const YearFromAdam& right) const
96  {
97    //int ThisAdamYear = ComputeAdamYear(this->indict, this->sun, this->moon);
98    //int RightAdamYear = ComputeAdamYear(right.indict, right.sun, right.moon);
99    int YearSum = ComputeAdamYear(this->indict, this->sun, this->moon) +
100                 ComputeAdamYear(right.indict, right.sun, right.moon);
101   return YearFromAdam(YearSum);
102 }
103
104 YearFromAdam YearFromAdam::operator-(const YearFromAdam& right) const
105 {
106   int YearDif = ComputeAdamYear(this->indict, this->sun, this->moon) -
107           ComputeAdamYear(right.indict, right.sun, right.moon);
108   return YearFromAdam(std::abs(YearDif));
109 }
110
111 YearFromAdam& YearFromAdam::operator+=(const YearFromAdam& right)
112 {
113   *this = *this + right;
114   return (*this);
115 }
116
117 YearFromAdam& YearFromAdam::operator-=(const YearFromAdam& right)
118 {
119   *this = *this - right;
```

```
120   return (*this);
121 }
122
123 bool YearFromAdam::operator==(const YearFromAdam& right) const
124 {
125   if (this->indict == right.indict && this->sun == right.sun && this->moon ==
      right.moon)
126     return true;
127
128   return false;
129 }
130
131 //Флаги для ввода и вывода
132 bool YearFromAdam::FormatYearAdamOut = false;
133 bool YearFromAdam::FormatYearJesusOut = false;
134 bool YearFromAdam::FormatYearAdamIn = false;
135 bool YearFromAdam::FormatYearJesusIn = false;
136
137 std::ostream& YearFromAdam::NormalFormatOut(std::ostream& stream)
138 {
139   YearFromAdam::FormatYearJesusOut = false;
140   YearFromAdam::FormatYearAdamOut = false;
141
142   return stream;
143 }
144
145 std::ostream& YearFromAdam::YearAdamFormatOut(std::ostream& stream)
146 {
147   YearFromAdam::FormatYearJesusOut = false;
148   YearFromAdam::FormatYearAdamOut = true;
149
150   return stream;
151 }
152
153 std::ostream& YearFromAdam::YearJesusFormatOut(std::ostream& stream)
154 {
155   YearFromAdam::FormatYearJesusOut = true;
156   YearFromAdam::FormatYearAdamOut = false;
157
158   return stream;
159 }
160
161 std::istream& YearFromAdam::NormalFormatIn(std::istream& stream)
162 {
163   YearFromAdam::FormatYearJesusIn = false;
164   YearFromAdam::FormatYearAdamIn = false;
165
166   return stream;
167 }
168
169 std::istream& YearFromAdam::YearAdamFormatIn(std::istream& stream)
170 {
171   YearFromAdam::FormatYearJesusIn = false;
```

```
172   YearFromAdam::FormatYearAdamIn = true;
173
174   return stream;
175 }
176
177 std::istream& YearFromAdam::YearJesusFormatIn(std::istream& stream)
178 {
179   YearFromAdam::FormatYearJesusIn = true;
180   YearFromAdam::FormatYearAdamIn = false;
181
182   return stream;
183 }
184
185 std::ostream& operator<<(std::ostream& o, const YearFromAdam& object)
186 {
187   if (YearFromAdam::FormatYearAdamOut)
188     o << "YearFromAdam: " << YearFromAdam::ComputeAdamYear(object.indict,
     object.sun, object.moon);
189   else if (YearFromAdam::FormatYearJesusOut)
190     o << "YearFromJesus: " << YearFromAdam::AdamToJesus(YearFromAdam::
     ComputeAdamYear(object.indict, object.sun, object.moon));
191   else
192     o << "Indict: " << object.indict << ", CircleToSun: " << object.sun << ",
     CircleToMoon: " << object.moon;
193
194   return o;
195 }
196
197 std::istream& operator>>(std::istream& in, YearFromAdam& object)
198 {
199   if (YearFromAdam::FormatYearAdamIn)
200   {
201     int y;
202     in >> y;
203     YearFromAdam temp(y);
204     object = temp;
205   }
206   else if (YearFromAdam::FormatYearJesusIn)
207   {
208     int y;
209     in >> y;
210     YearFromAdam temp(y,false);
211     object = temp;
212   }
213   else {
214     int ind, sol, lon;
215     in >> ind >> sol >> lon;
216     YearFromAdam temp(ind, sol, lon);
217     object = temp;
218   }
219
220   return in;
221 }
```

```
222
223 int YearFromAdam::ComputeAdamYear(int ind, int sol, int lon)
224 {
225   int a = (ind % 15 >= 0) ? (ind % 15) : (ind % 15 + 15);
226   int b = (sol % 28 >= 0) ? (sol % 28) : (sol % 28 + 28);
227   int c = (lon % 19 >= 0) ? (lon % 19) : (lon % 19 + 19);
228
229   int ans = 940576 * a - 944775 * b + 4200 * c;
230
231   return (ans > 0) ? (ans % 7980) : (ans % 7980 + 7980);
232 }
233
234 int YearFromAdam::AdamToJesus(int YearInAdam)
235 {
236   return (YearInAdam >= 0) ? (YearInAdam - 5508) : (-5508);
237 }
238
239 int YearFromAdam::JesusToAdam(int YearInJesus)
240 {
241   return (YearInJesus + 5508 >= 0) ? (YearInJesus + 5508) : 0;
242 }
243
244 int YearFromAdam::GetIndict() const
245 {
246   return this->indict;
247 }
248
249 int YearFromAdam::GetSun() const
250 {
251   return this->sun;
252 }
253
254 int YearFromAdam::GetMoon() const
255 {
256   return this->moon;
257 }
258
259 void YearFromAdam::SetIndict(int ind)
260 {
261   this->indict = (ind % 15 == 0) ? 15 : (ind % 15);
262   if (this->indict < 0) this->indict += 15;
263 }
264
265 void YearFromAdam::SetSun(int sol)
266 {
267   this->sun = (sol % 28 == 0) ? 28 : (sol % 28);
268   if (this->sun < 0) this->sun += 28;
269 }
270
271 void YearFromAdam::SetMoon(int lon)
272 {
273   this->moon = (lon % 19 == 0) ? 19 : (lon % 19);
274   if (this->moon < 0) this->moon += 19;
```

```
275 }
276
277 YearFromAdam::operator int()
278 {
279   return YearFromAdam::ComputeAdamYear(this->indict, this->sun, this->moon);
280 }
```

## B.3 Файл main2.cpp

```
1  #include <iostream>
2  #include <cassert>
3  #include <sstream>
4  #include "YearFromAdam.h"
5
6  int main()
7  {
8    { //TEST1. Проверка работоспособности конструкторов
9      YearFromAdam obj1, obj2(-2, -28, 20), obj3(7160), obj4(1652, false), obj5(
       obj2);
10
11     bool correct1p1 = (obj1.GetIndict() == 1) && (obj1.GetSun() == 1) && (obj1
       .GetMoon() == 1);
12     bool correct1p2 = (obj2.GetIndict() == 13) && (obj2.GetSun() == 28) && (
       obj2.GetMoon() == 1);
13     bool correct1p3 = (obj3.GetIndict() == 5) && (obj3.GetSun() == 20) && (
       obj3.GetMoon() == 16);
14     bool correct1p4 = (obj4.GetIndict() == 5) && (obj4.GetSun() == 20) && (
       obj4.GetMoon() == 16);
15     bool correct1p5 = (obj5.GetIndict() == 13) && (obj5.GetSun() == 28) && (
       obj5.GetMoon() == 1);
16
17     bool correct1 = correct1p1 && correct1p2 && correct1p3 && correct1p4 &&
       correct1p5;
18
19     assert(correct1);
20   }
21
22   {//TEST2. Проверка работоспособности арифметических операций и присваивания
23     YearFromAdam obj(-1, -28, 20);
24
25     bool correct2p1 = (obj++ == YearFromAdam(14, 28, 1))
26                 && (++obj == YearFromAdam(1, 2, 3))
27                 && (obj-- == YearFromAdam(1, 2, 3))
28                 && (--obj == YearFromAdam(14, 28, 1));
29
30     YearFromAdam obj1(15, 28, 19), obj2;
31
32     obj2 = obj1;
33
34     bool correct2p2 = (obj2 == YearFromAdam(15, 28, 19));
35
36     obj1 += obj;
```

```
37      obj2 -= obj;
38
39      bool correct2p3 = (obj1 == YearFromAdam(14, 28, 1))
40              && (obj2 == YearFromAdam(1, 28, 18));
41
42      bool correct2p4 = (obj1 - obj2 = YearFromAdam(15, 28, 19))
43              && (obj + obj = YearFromAdam(13, 28, 2));
44
45      bool correct2 = correct2p1 && correct2p2 && correct2p3 && correct2p4;
46
47      assert(correct2);
48    }
49
50    {//Test3. Проверка работоспособности геттеров и сеттеров
51      YearFromAdam obj, obj1(1, 28, 13);
52
53      bool correct3p1 = (obj1.GetIndict() == 1) && (obj1.GetSun() == 28) && (
       obj1.GetMoon() == 13);
54
55      obj.SetIndict(-22); obj.SetSun(38); obj.SetMoon(0);
56
57      bool correct3p2 = obj == YearFromAdam(8, 10, 19);
58
59      bool correct3 = correct3p1 && correct3p2;
60
61      assert(correct3);
62    }
63
64    {//Test4. Проверка работоспособности оператора приведения к базовому типу и функций
       подсчета и перевода годов
65      YearFromAdam obj(3, 20, 17);
66
67      int YearInAdam = static_cast<int>(obj);
68      int YearInJesus = YearFromAdam::AdamToJesus(YearInAdam);
69
70      int Adam = YearFromAdam::JesusToAdam(1652);
71
72
73      bool correct4 = (YearInAdam == 5508)
74              && (YearInJesus == 0)
75              && (Adam == 7160);
76
77      assert(correct4);
78    }
79
80    {//Test5. Проверка работоспособности перегрузки операции помещения в поток и взятия из
       потока в различных формах
81      YearFromAdam obj(15, 13, 10);
82
83      std::stringstream stream1;
84
85      stream1 << obj << "| " << YearFromAdam::YearAdamFormatOut << obj << "| "
       << YearFromAdam::YearJesusFormatOut
```

```
86          << obj << "| " << YearFromAdam::NormalFormatOut   << obj;
87
88      std::string output;
89      getline(stream1, output);
90
91      bool correct5p1 = output == "Indict: 15, CircleToSun: 13, CircleToMoon:
        10| YearFromAdam: 1245| YearFromJesus: -4263| Indict: 15, CircleToSun: 13,
        CircleToMoon: 10";
92
93
94      YearFromAdam obj1, obj2, obj3, obj4;
95
96      std::stringstream stream2;
97
98      stream2 << "-5 31 0 7895 1324 0 0 0";
99
100     stream2 >> obj1 >> YearFromAdam::YearAdamFormatIn >> obj2 >> YearFromAdam
        ::YearJesusFormatIn >> obj3
101           >> YearFromAdam::NormalFormatIn >> obj4;
102
103     bool correct5p2 = (obj1 == YearFromAdam(10, 3, 19))
104             && (obj2 == YearFromAdam(7895)) //(5,27,10)
105             && (obj3 == YearFromAdam(1324, false)) //(7,28,11)
106             && (obj4 == YearFromAdam(15, 28, 19));
107
108     bool correct5 = correct5p1 && correct5p2;
109
110     assert(correct5);
111   }
112
113   return 0;
114 }
```

# Приложение Г

## Г.1 Файл Matrix.h

```
#ifndef MATRIX_H
#define MATRIX_H

#include <iostream>
#include <vector>

class Matrix
{
private:
  struct coord
  {
    std::size_t i; // номер строки
    std::size_t j; // номер столбца
    double value; // Значение

    bool operator==(const coord& right) const;
  };

  std::size_t rows, cols;
  std::vector<coord> coefs;

public:
  Matrix();
   ~Matrix() = default;
    Matrix(const Matrix& other);
    Matrix(std::size_t r, std::size_t c);

  Matrix operator+(const Matrix& right) const;
  Matrix operator-(const Matrix& right) const;

  bool operator==(const Matrix& right) const;

  Matrix& operator=(const Matrix& right);

  Matrix operator*(double lambda) const;
  Matrix operator*(const Matrix& right) const;
  friend Matrix operator*(double lambda, const Matrix& right);

  friend std::ostream& operator<<(std::ostream& out, const Matrix& object);
  friend std::istream& operator>>(std::istream& in,  Matrix& object);

  Matrix Transpose() const;

  //Сеттеры
  void SetRows(std::size_t NewR);
  void SetCols(std::size_t NewC);

  //Геттер
  std::size_t GetRows() const;
```

```
50    std::size_t GetCols() const;
51    std::vector<coord> GetCoefs() const;
52
53    double  operator()(std::size_t row, std::size_t col) const; //если нужно как
       rvalue
54    double& operator()(std::size_t row, std::size_t col); //если нужно как lvalue
55 };
56
57 #endif //MATRIX_H
```

## Г.2 Файл Matrix.cpp

```
1  #include "Matrix.h"
2  #include <vector>
3  #include <cassert>
4  #include <algorithm>
5
6  Matrix::Matrix()
7      :rows{ 0 }, cols{ 0 }
8  {}
9
10 Matrix::Matrix(const Matrix& other)
11     : rows{ other.rows }, cols{ other.cols }, coefs{other.coefs}
12 {}
13
14 Matrix::Matrix(std::size_t r, std::size_t c)
15     : rows{ r }, cols{ c }
16 {}
17
18 Matrix Matrix::operator+(const Matrix& right) const
19 {
20     if (this->rows == right.rows && this->cols == right.cols)
21     {
22         Matrix result(rows, cols);
23         for (std::size_t a{ 0 }; a < rows; ++a)
24             for (std::size_t b{ 0 }; b < cols; ++b)
25                 result(a, b) = (*this)(a, b) + right(a, b);
26
27         //Если внеслись нули
28
29         for (std::size_t a{ 0 }; a < result.coefs.size(); ++a)
30         {
31             if (result.coefs[a].value == 0)
32             {
33                 result.coefs.erase(result.coefs.begin() + a);
34                 --a;
35             }
36         }
37
38         return result;
39     }
40 }
```

```
41
42 Matrix Matrix::operator-(const Matrix& right) const
43 {
44     if (this->rows == right.rows && this->cols == right.cols)
45     {
46         Matrix result(rows, cols);
47         for (std::size_t a{ 0 }; a < rows; ++a)
48             for (std::size_t b{ 0 }; b < cols; ++b)
49                 result(a, b) = (*this)(a, b) - right(a, b);
50
51         //Если внеслись нули
52
53         for (std::size_t a{ 0 }; a < result.coefs.size(); ++a)
54         {
55             if (result.coefs[a].value == 0)
56             {
57                 result.coefs.erase(result.coefs.begin() + a);
58                 --a;
59             }
60         }
61
62         return result;
63     }
64 }
65
66 bool Matrix::operator==(const Matrix& right) const
67 {
68     if (this->rows == right.rows && this->cols == right.cols && this->coefs.
    size() == right.coefs.size())
69     {
70         Matrix first(*this), second(right);
71
72         std::size_t cnt{ 0 };
73
74         for (std::size_t a{0};a<first.coefs.size();++a)
75             for (std::size_t b{ 0 }; b < second.coefs.size(); ++b)
76             {
77                 if (first.coefs[a] == second.coefs[b])
78                 {
79                     first.coefs[a].i = -1;
80                     second.coefs[b].i = -1;
81                 }
82             }
83
84         for (const auto& elem : first.coefs)
85             if (elem.i == -1)
86                 ++cnt;
87
88         if (cnt == first.coefs.size())
89             return true;
90     }
91
92     return false;
```

```
93 }
94
95 Matrix& Matrix::operator=(const Matrix& right)
96 {
97     if (this != &right)
98     {
99         this->rows = right.rows;
100         this->cols = right.cols;
101         this->coefs = right.coefs;
102     }
103
104     return (*this);
105 }
106
107
108 Matrix Matrix::operator*(double lambda) const
109 {
110     Matrix res(*this);
111     for (auto& elem : res.coefs)
112     {
113         elem.value *= lambda;
114     }
115
116     return res;
117 }
118
119 Matrix Matrix::operator*(const Matrix& right) const
120 {
121     if (this->cols == right.rows)
122     {
123         Matrix result(this->rows, right.cols);
124
125         for (size_t r = 0; r < this->rows; ++r)
126             for (size_t c = 0; c < right.cols; ++c)
127                 for (size_t s = 0; s < this->cols; ++s) {
128                     result(r, c) += ((*this)(r, s) * right(s, c));
129                 }
130
131         //Если внеслись нули
132
133         for (std::size_t a{ 0 }; a < result.coefs.size(); ++a)
134         {
135             if (result.coefs[a].value == 0)
136             {
137                 result.coefs.erase(result.coefs.begin() + a);
138                 --a;
139             }
140         }
141
142
143         return result;
144     }
145 }
```

```
146
147 Matrix Matrix::Transpose() const
148 {
149     Matrix result(*this);
150
151     for (auto& elem : result.coefs)
152         std::swap(elem.i, elem.j);
153
154     std::swap(result.rows, result.cols);
155
156     return result;
157 }
158
159 void Matrix::SetRows(std::size_t NewR)
160 {
161    for (std::size_t k{ 0 }; k < coefs.size(); ++k)
162       if (coefs[k].i >= NewR)
163           coefs.erase(coefs.begin() + k);
164
165    this->rows = NewR;
166 }
167
168 void Matrix::SetCols(std::size_t NewC)
169 {
170     for (std::size_t k{ 0 }; k < coefs.size(); ++k)
171         if (coefs[k].j >= NewC)
172             coefs.erase(coefs.begin() + k);
173
174     this->cols = NewC;
175 }
176
177 std::size_t Matrix::GetRows() const
178 {
179     return this->rows;
180 }
181
182 std::size_t Matrix::GetCols() const
183 {
184     return this->cols;
185 }
186
187 std::vector<Matrix::coord> Matrix::GetCoefs() const
188 {
189     return this->coefs;
190 }
191
192 double Matrix::operator()(std::size_t row, std::size_t col) const
193 {
194     if ((row < this->rows) && (col < this->cols))
195     {
196         int val = 0;
197
198         for (const auto& elem : this->coefs)
```

```
199         {
200             if (elem.i == row && elem.j == col)
201                 val = elem.value;
202         }
203 
204         return val;
205     }
206 }
207 
208 double& Matrix::operator()(std::size_t row, std::size_t col)
209 {
210     if ((row < this->rows) && (col < this->cols))
211     {
212 
213         for (auto& elem : this->coefs)
214         {
215             if (elem.i == row && elem.j == col)
216                 return elem.value;
217         }
218 
219         //Если в цикле не вышли, то такого элемента нет и его надо внести
220 
221         this->coefs.push_back({ row, col, 0 });
222 
223         for (auto& elem : this->coefs)
224         {
225             if (elem.i == row && elem.j == col)
226                 return elem.value;
227         }
228     }
229 }
230 
231 Matrix operator*(double lambda, const Matrix& right)
232 {
233     return (right * lambda);
234 }
235 
236 std::ostream& operator<<(std::ostream& out, const Matrix& object)
237 {
238     out << object.rows << ' ' << object.cols << '\n';
239 
240     for (std::size_t a{ 0 }; a < object.rows; ++a)
241     {
242         for (std::size_t b{ 0 }; b < object.cols; ++b)
243         {
244             if (b != object.cols - 1)
245                 out << object(a, b) << ' ';
246             else
247                 out << object(a, b);
248         }
249 
250         if(a!=object.cols-1)
251             out << '\n';
```

```
252     }
253
254     return out;
255 }
256
257 std::istream& operator>>(std::istream& in, Matrix& object)
258 {
259     in >> object.rows >> object.cols;
260
261     for(std::size_t a{0};a<object.rows;++a)
262         for (std::size_t b{ 0 }; b < object.cols; ++b)
263         {
264             int num;
265             assert(in >> num);
266
267             if (num != 0)
268                 object(a, b) = num;
269         }
270     return in;
271 }
272
273 bool Matrix::coord::operator==(const coord& right) const
274 {
275     if (this->i == right.i && this->j == right.j && this->value == right.value
    )
276         return true;
277
278     return false;
279 }
```

## Г.3 Файл main3.cpp

```
1 #include "Matrix.h"
2 #include <cassert>
3 #include <sstream>
4
5 int main()
6 {
7   { //Test1. Проверка работоспособности конструкторов, геттеров и сеттеров
8     Matrix m1, m2(2, 3);
9
10    bool correct1p1 = (m1.GetRows() == 0 && m1.GetCols() == 0 && m1.GetCoefs()
    .size() == 0)
11             && (m2.GetRows() == 2 && m2.GetCols() == 3 && m2.GetCoefs().size
    () == 0);
12
13    m2.SetCols(3); m2.SetRows(4);
14
15    m2(0, 0) = 1;
16    m2(1, 1) = 2;
17    m2(2, 2) = 3;
18
```

```
19      Matrix m3(m2);
20
21      bool correct1p2 = (m3.GetCols() == 3 && m3.GetRows() == 4)
22              && (m3.GetCoefs()[0].i == 0 && m3.GetCoefs()[0].j == 0 && m3.
        GetCoefs()[0].value == 1)
23              && (m3.GetCoefs()[1].i == 1 && m3.GetCoefs()[1].j == 1 && m3.
        GetCoefs()[1].value == 2)
24              && (m3.GetCoefs()[2].i == 2 && m3.GetCoefs()[2].j == 2 && m3.
        GetCoefs()[2].value == 3);
25
26      bool correct1 = correct1p1 && correct1p2;
27
28      assert(correct1);
29    }
30
31    { // Test2. Проверка работоспособности арифметики, транспонирования, присваивания и
       равенства
32
33      Matrix m1(2, 3), m2(2,3), mr;
34
35      m1(0, 0) = 15; m1(1, 1) = 10; m1(0, 2) = 42;
36
37      m2(0, 0) = 11; m2(1, 1) = 22; m2(1, 2) = 5;
38
39      mr = m1 + m2;
40
41      bool correct2p1 = (mr.GetRows() == 2 && mr.GetCols() == 3)
42              && (mr.GetCoefs()[0].i == 0 && mr.GetCoefs()[0].j == 0 && mr.
        GetCoefs()[0].value == 26)
43              && (mr.GetCoefs()[1].i == 0 && mr.GetCoefs()[1].j == 2 && mr.
        GetCoefs()[1].value == 42)
44              && (mr.GetCoefs()[2].i == 1 && mr.GetCoefs()[2].j == 1 && mr.
        GetCoefs()[2].value == 32)
45              && (mr.GetCoefs()[3].i == 1 && mr.GetCoefs()[3].j == 2 && mr.
        GetCoefs()[3].value ==  5);
46
47      mr = 5 * (m2 - m1).Transpose() * (-2);
48
49      bool correct2p2 = (mr.GetRows() == 3 && mr.GetCols() == 2)
50              && (mr.GetCoefs()[0].i == 0 && mr.GetCoefs()[0].j == 0 && mr.
        GetCoefs()[0].value ==  40)
51              && (mr.GetCoefs()[1].i == 2 && mr.GetCoefs()[1].j == 0 && mr.
        GetCoefs()[1].value == 420)
52              && (mr.GetCoefs()[2].i == 1 && mr.GetCoefs()[2].j == 1 && mr.
        GetCoefs()[2].value == -120)
53              && (mr.GetCoefs()[3].i == 2 && mr.GetCoefs()[3].j == 1 && mr.
        GetCoefs()[3].value == -50);
54
55      Matrix m3(1,3), m4(3,2);
56      m3(0, 0) = 1; m3(0, 2) = 1;
57      m4(0, 0) = 1; m4(1, 0) = 5; m4(1, 1) = 6; m4(2, 1) = 3;
58
59      mr = m3 * m4;
```

```
60
61      bool correct2p3 = (mr.GetRows() == 1 && mr.GetCols() == 2)
62              && (mr.GetCoefs()[0].i == 0 && mr.GetCoefs()[0].j == 0 && mr.
        GetCoefs()[0].value == 1)
63              && (mr.GetCoefs()[1].i == 0 && mr.GetCoefs()[1].j == 1 && mr.
        GetCoefs()[1].value == 3);
64
65      Matrix m5(2, 2), m6(2, 2);
66      m5(0, 1) = 1; m5(1, 0) = 2;
67      m6(1, 0) = 2; m6(0, 1) = 1;
68
69      bool correct2p4 = (m5 == m6) && !(mr == m5);
70
71      bool correct2 = correct2p1 && correct2p2 && correct2p3 && correct2p4;
72
73      assert(correct2);
74  }
75
76  { //Test3. Проверка работоспособности операций помещения в поток и взятия из потока
77      std::stringstream ss;
78      Matrix m1(3,3), m2(3,3);
79
80      m1(0, 0) = 1; m1(1, 1) = 1; m1(2, 2) = 1;
81      m2(0, 2) = 5; m2(1, 0) = 3; m2(2, 1) = 8;
82
83      ss << (m1 * m2).Transpose();
84
85      std::string str1, str2, str3, str4;
86
87      std::getline(ss, str1);
88      std::getline(ss, str2);
89      std::getline(ss, str3);
90      std::getline(ss, str4);
91
92      bool correct3p1 = (str1 == "3 3")
93                && (str2 == "0 3 0")
94              && (str3 == "0 0 8")
95              && (str4 == "5 0 0");
96
97      Matrix mInput1, mInput2;
98      std::stringstream ss1;
99
100     ss1 << "3 3\n0 3 0\n0 0 8\n5 0 0" << "\n2 2 0 0 0 1";
101
102     ss1.seekg(0, std::istream::beg);
103
104     ss1 >> mInput1 >> mInput2;
105
106     bool correct3p2 = (mInput1 == (m1 * m2).Transpose())
107             && (mInput2.GetRows() == 2 && mInput2.GetCols() == 2)
108             && (mInput2.GetCoefs()[0].i == 1 && mInput2.GetCoefs()[0].j == 1
        && mInput2.GetCoefs()[0].value == 1);
109
```

```
110     bool correct3 = correct3p1 && correct3p2;
111
112     assert(correct3);
113   }
114     return 0;
115 }
```

# Приложение Д

## Д.1 Файл CMakeLists.txt

```
cmake_minimum_required(VERSION 3.12)

#set(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} -Wall -Werror -fsanitize=undefined -std=c++20")
set(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} -Wall -pedantic-errors -fsanitize=undefined -std=c++20")

project(Laba1)

add_executable(main0 main0.cpp DateAndTime.h DateAndTime.cpp)
add_executable(main1 main1.cpp BigNumber.h BigNumber.cpp)
add_executable(main2 main2.cpp YearFromAdam.h YearFromAdam.cpp)
add_executable(main3 main3.cpp Matrix.h Matrix.cpp)
```