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-203 А.Р. Фахретдинов

подпись, дата

Работу проверил С.А. Булгаков

подпись, дата

Москва 2021

Содержание

Постановка задачи 3

1 Алгоритм решения задачи 4

1.1 Задача №1 4

1.2 Задача №2 4

1.3 Задача №3 4

1.4 Задача №4 4

2 Выполнение задания 5

2.1 Задача №1 5

2.2 Задача №2 6

2.3 Задача №3 7

2.4 Задача №4 8

3 Получение исполняемых модулей 9

4 Тестирование 10

4.1 Задача №1 10

4.2 Задача №2 10

4.3 Задача №3 10

4.4 Задача №4 10

Приложение А 11

Приложение Б 18

Приложение В 32

Приложение Г 39

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 дей-ствия 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, пред-ставляющее число интервалов по 100 наносекунд, прошедших с первого января 1601 года).
2. Целое произвольной длины (во внешней форме представления в виде строки символов-цифр). Базовый тип: std::string.
3. Год «от Адама», имеющий внутреннее представление в виде целочисленных переменных: 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. 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

Для решения задачи были использованы функции стандартной библиотеки из заголовочного файла ctime: localtime – для преобразования секунд в текущию дату, ctime - преобразует значение типа time_t в Си-строку, которая содержит дату и время в человеко-понятном формате, mktime - восстанавливает значения остальных членов структуры типа tm по исходным данным(количество секунд).

Для преобразования в/из формата представления filetime были использованы две макро подстановки, являющиеся разницей времени отсчёта между форматом хранения времени в *nix-системах и Windows-системах – EPOCH_DIFF и ста миллисекундами – WINDOWS_TICK.

1.2 Задача №2

Для решения задачи был разработан класс, принимающий из потока число в виде строки n-длины, определяющий знак числа и разбивающий строку на ячейки по 9 символов. После была разработана арифметика столбиком для основных арифметических операций и перегружены операторы инкремента и декремента.

1.3 Задача №3

Для решения задачи был разработан класс, принимающий либо «год от Адама», либо индикт, круг от солнца и круг от луны, и конвертирующий год из одного представления в другое. Для конвертации в индикт, круг от солнца и круг от луны, находятся соответствующие остатки от деления «года от Адама» на максимальные значения соответствующих диапазонов. Для конвертации в «год от Адама» производится перебор всех возможных годов и сравнение остатков от деления года на максимальные значения соответствующих диапазонов с известными значениями индикта, круга от солнца и круга от луны.

1.4 Задача №4

Для решения задачи был разработан класс, принимающий из потока динамический массив структур, хранящих информацию от координатах и значении элемента разреженной матрицы. В качестве указания момента остановки ввода было принято решение, взять ввод, содержащий нулевый координаты. Структура, вводимая пользователем, с ненулевыми координатами и нулевым значением в массив структур не записывается. В качестве меры размерности матрицы, берётся максимальные значения строк и столбцов из массива структур. Для класса были разработаны арифметические операции согласно математическим правилам действий над матрицами.

2 Выполнение задания

2.1 Задача №1

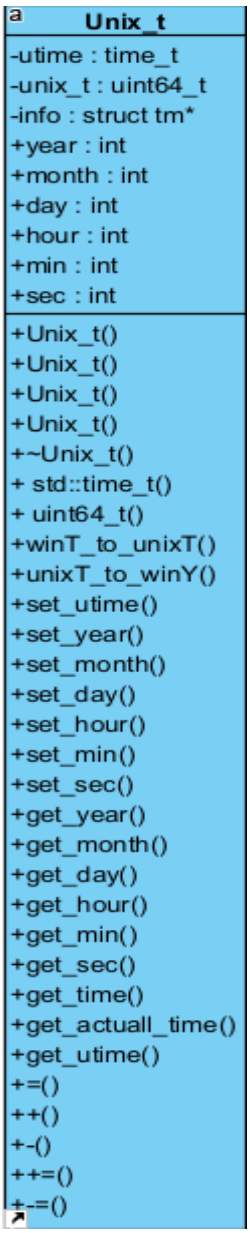

Рис. 1. UML diagram for Unix_t class

Класс написан на языке C++. Код класса размещается в одной единицах трансляции – код в файле data_time.cpp, прототип класса в заголовочном файле data_time.h. Тесты в виде проверки результатов и постусловий с помощью утверждений выполнены в единице трансляции main.cpp с функцией main, при этом, заголовочный файл содержит защиту от повторного включения. Обмен информации между тестировщиком и программой осуществляется через стандартный поток вывода.

Реализация класса включает в себя закрытые поля utime типа time_t, unix_t типа uint64_t, info типа указателя на структуру tm, а так же year, month, day, hour, min, sec типа int. Конструктор поумолчанию задает их значения равными 0 и 0 соответственно. Конструктор общего вида принимает на вход количество секунд, прошедших с 00:00 1 января 1970 года или количество лет, месяцев, дней, часов, минут и секунд.

Реализована базовая арифметика и акцессоры, а так же операторы сравнения, операторы ввода/вывода в поток с помощью дружественных функций.

2.2 Задача №2

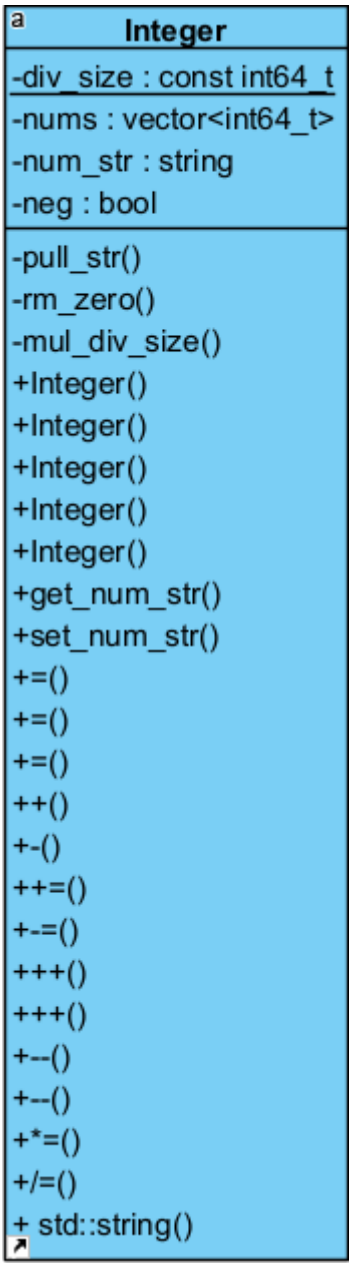

Рис. 2. UML diagram for Integer class

Класс написан на языке C++. Код класса размещается в одной единицах трансляции – код в файле new_integer.cpp, прототип класса в заголовочном файле new_integer.h. Тесты в виде проверки результатов и постусловий с помощью утверждений выполнены в единице трансляции main.cpp с функцией main, при этом, заголовочный файл содержит защиту от повторного включения. Обмен информации между тестировщиком и программой осуществляется через стандартный поток вывода.

Реализация класса включает в себя закрытые поля div_size типа static const int64_t, nums типа vector<int64_t>, num_str типа string и neg типа bool, обозначающее знак числа. Конструктор по умолчанию задает значение neg равным false. Конструкторы общего вида принимают на вход либо поток ввода, либо строку, либо число типа int64_t.

Реализована базовая арифметика с помощью перегрузки операторов +, -, *, /,++,--, а так же акцессоры для доступа к полям класса. По мимо этого операторы сравнения, операторы ввода/вывода в поток реализованы с помощью дружественных функций.

2.3 Задача №3

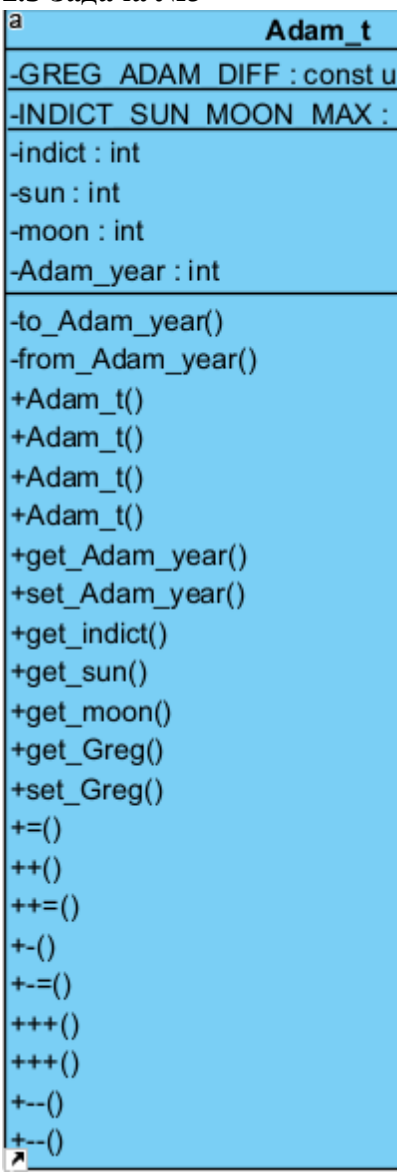

Рис. 3. UML diagram for Adam_t class

Класс написан на языке C++. Код класса размещается в одной единицах трансляции – код в файле Adam_time.cpp, прототип класса в заголовочном файле Adam_time.h. Тесты в виде проверки результатов и постусловий с помощью утверждений выполнены в единице трансляции main.cpp с функцией main, при этом, заголовочный файл содержит защиту от повторного включения. Обмен информации между тестировщиком и программой осуществляется через стандартный поток вывода.

Реализация класса включает в себя закрытые поля indict, sun, moon, Adam_year типа int и GREG_ADAM_DIFF – количество лет между началом отсчёта по Грегорианскому летоотсчислению и году от Адама, INDICT_SN_MOON_MAX – произведение максимальных значений диапазонов индикта, круга от солнца и круга от луны типа int. Конструктор по умолчанию не задаёт значения. Конструктор класса принимает либо индикт, круг от солнца и круг от луны, либо год от Адама.

Реализована базовая арифметика и акцессоры, а так же операторы ввода/вывода в поток с помощью дружественных функций.

2.4 Задача №4

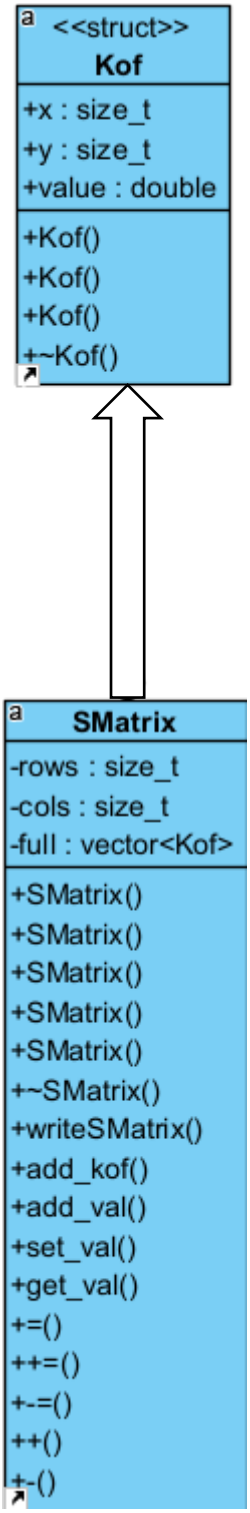

Рис. 4. UML diagram for Smatrix class

Класс написан на языке C++. Код класса размещается в одной единицах трансляции – код в файле spare_matrix.cpp, прототип класса в заголовочном файле spare_matrix.h. Тесты в виде проверки результатов и постусловий с помощью утверждений выполнены в единице трансляции main.cpp с функцией main, при этом, заголовочный файл содержит защиту от повторного включения. Обмен информации между тестировщиком и программой осуществляется через стандартный поток вывода.

Реализация класса включает в себя закрытые поля rows, cols типа int и full типа vector<Kof>, где Kof – структура, открыто реализованная в классе и хранящая информацию о координатах элемента в матрице типа int и его значении типа double. Конструктор по умолчанию не задаёт значения. Конструкторы общего вида принимают на вход либо поток ввода, либо указатель на массив типа Kof с заданным размером типа size_t, либо ссылку на vector<Kof>.

Реализована базовая арифметика согласно математическим правилам работы с матрицами и акцессоры, а так же операторы ввода/вывода в поток с помощью дружественных функций.

3 Получение исполняемых модулей

Получение исполняемых модулей происходит с помощью системы сборки cmake. Задан стандарт языка C++14 и ключи компиляции -Wall, для отображения максимального количество ошибок и предупреждений и -03, для максимальной оптимизации исполняемого файла, минимальная версия cmake 3.12.

Листинг 1 – Файл CmakeList.txt

```
cmake_minimum_required(VERSION 3.12 FATAL_ERROR)
project(lab_01)
add_definitions(-Wall -03)
// XXX – название одной из частей(data_time/new_integer/Adam_time/spare_matrix)
add_executable(${PROJECT_NAME} "main.cpp" "XXX.cpp" "XXX.h")
target_compile_features(${PROJECT_NAME} PRIVATE cxx_std_14)
```

4 Тестирование

4.1 Задача №1

Проведён базовый тест на функциональность в файле main.cpp в функции main.

4.2 Задача №2

Проведён базовый тест на функциональность в файле main.cpp в функции main.

4.3 Задача №3

Проведён базовый тест на функциональность в файле main.cpp в функции main.

4.4 Задача №4

Проведён базовый тест на функциональность в файле main.cpp в функции main.

Приложение А

А Файл data_time.h

```
#ifndef data_time_h // если не объявлен
#define data_time_h // объявление

// подключение необходимых заголовочных файлов
#include <cstdint>
#include <ctime>
#include <iostream>

// объявление макросов подстановки
#define WINDOWS_TICK 10000000       // 100 ms
#define EPOCH_DIFFER 11644473600LL  // unix time - win time



class Unix_t
{
    private:
        std::time_t utime; // для работы с методами класса time_t
        uint64_t unix_t; // секунд от 00:00 1 января 1970 года
        struct tm *info; // для работы с годом, месяцем, днём, часом, минутой и
//секундой

    public:
        int year; // хранение года
        int month;// хранение месяца
        int day;  // хранение дня
        int hour; // хранение часа
        int min;  // хранение минуты
        int sec;  // хранение секунды

    public:
        Unix_t(); // конструктор класса
        Unix_t(uint64_t unix_t); // конструктор класса
        Unix_t(int year, int month, int day=1, int hour=0, int min=0, int sec=0);
// конструктор класса
        Unix_t(const Unix_t &right); // конструктор копирования
        ~Unix_t() {}  // деструктор класса
```

```
        char* get_time() const { return std::ctime(&utime); }  // метод,
//возвращающий время в человеко-читаемом формате
        uint64_t get_utime() const { return unix_t; } // метод, возвращающий время
//в Unix формате


        // операторы приведения типов
        operator std::time_t() const { return unix_t; }
        operator uint64_t() const { return utime; }


        uint64_t winT_to_unixT() // преобразование из формат представления
//filetime
        {
            return ( (this->unix_t / WINDOWS_TICK) - EPOCH_DIFFER);
        }
        uint64_t unixT_to_winY() // преобразование в формат представления filetime
        {
            return (WINDOWS_TICK * (EPOCH_DIFFER + this->unix_t) );
        }


        // методы устанавливающие и возвращающие защищённые поля класса
        void set_utime(uint64_t unix_t);
        void set_year(int year) { this->year = year; }
        void set_month(int month) { this->month = month; }
        void set_day(int day) { this->day = day; }
        void set_hour(int hour) { this->hour = hour; }
        void set_min(int min) { this->min = min; }
        void set_sec(int sec) { this->sec = sec; }
        int get_year() { return this->year; }
        int get_month() { return this->month; }
        int get_day() { return this->day; }
        int get_hour() { return this->hour; }
        int get_min() { return this->min; }
        int get_sec() { return this->sec; }


        // перегрузка арифметических операторов
        Unix_t operator=(const Unix_t &right);
        Unix_t operator+(const Unix_t &right) const;
        Unix_t operator-(const Unix_t &right) const;
        Unix_t operator+=(const Unix_t &right);
        Unix_t operator-=(const Unix_t &right);
```

```
        // перегрузка дружественной функцией операторов ввввода/вывода в поток
        friend std::ostream & operator<<(std::ostream &out, const Unix_t &right);
        friend std::istream & operator>>(std::istream &in, Unix_t &right);

        // перегрузка дружественной функцией операторов сравнения
        friend bool operator< (const Unix_t &left, const Unix_t &right);
        friend bool operator<=(const Unix_t &left, const Unix_t &right);
        friend bool operator> (const Unix_t &left, const Unix_t &right);
        friend bool operator>=(const Unix_t &left, const Unix_t &right);
        friend bool operator==(const Unix_t &left, const Unix_t &right);
        friend bool operator!=(const Unix_t &left, const Unix_t &right);
};

#endif // data_time_h
```

А Файл data_time.cpp

```
#include <ctime>
#include "data_time.h"

Unix_t::Unix_t() :
utime(0), unix_t(0), info(std::localtime(&utime)), year(0),
month(0), day(0), min(0), sec(0) {}

Unix_t::Unix_t(uint64_t unix_t) : utime(unix_t), unix_t(unix_t)
{
    info = std::localtime(&utime);
    this->year  = info->tm_year + 1900;
    this->month = info->tm_mon + 1;
    this->day   = info->tm_mday;
    this->hour  = info->tm_hour;
    this->min   = info->tm_min;
    this->sec   = info->tm_sec;
}
Unix_t::Unix_t(int year, int month, int day, int hour, int min, int sec)
{
    this->year  = year;
```

```
    this->month = month;
    this->day   = day;
    this->hour  = hour;
    this->min   = min;
    this->sec   = sec;
    std::time(&utime);
    info = localtime(&utime);
    info->tm_year = year - 1900;
    info->tm_mon  = month - 1;
    info->tm_mday = day;
    info->tm_hour = hour;
    info->tm_min  = min;
    info->tm_sec  = sec;
    utime = mktime(info);
    unix_t = utime;
}

Unix_t::Unix_t(const Unix_t &right) :
utime(right.utime), unix_t(right.unix_t),
year(right.year), month(right.month), day(right.day),
hour(right.hour), min(right.min), sec(right.sec) {}

Unix_t Unix_t::operator=(const Unix_t &right)
{
    this->utime  = right.utime;
    this->unix_t = right.unix_t;
    this->year = right.year;
    this->month= right.month;
    this->day  = right.day;
    this->hour = right.hour;
    this->min  = right.min;
    this->sec  = right.sec;
    return *this;
}
Unix_t Unix_t::operator+(const Unix_t &right) const
{
    Unix_t var(*this);
    var.utime  += right.utime;
    var.unix_t += right.unix_t;
    var.year += right.year;
```

```
    var.month+= right.month;
    var.day  += right.day;
    var.hour += right.hour;
    var.min  += right.min;
    var.sec  += right.sec;
    return var;
}
Unix_t Unix_t::operator-(const Unix_t &right) const
{
    Unix_t var(*this);
    var.utime  -= right.utime;
    var.unix_t -= right.unix_t;
    var.year -= right.year;
    var.month-= right.month;
    var.day  -= right.day;
    var.hour -= right.hour;
    var.min  -= right.min;
    var.sec  -= right.sec;
    return var;
}
Unix_t Unix_t::operator+=(const Unix_t &right)
{
    *this = this->operator+(right);
    return *this;
}
Unix_t Unix_t::operator-=(const Unix_t &right)
{
    *this = this->operator-(right);
    return *this;
}

std::ostream &operator<<(std::ostream &out, const Unix_t &right)
{
    return out << right.get_time();
}
std::istream &operator>>(std::istream &in, Unix_t &right)
{
    uint64_t var;
    in >> var;
    Unix_t qwa(var);
```

```
    right = qwa;
    return in;
}

bool operator< (const Unix_t &left, const Unix_t &right)
{
    return left.unix_t < right.unix_t ? true : false;
}
bool operator<=(const Unix_t &left, const Unix_t &right)
{
    return left.unix_t <= right.unix_t ? true : false;
}
bool operator> (const Unix_t &left, const Unix_t &right)
{
    return left.unix_t > right.unix_t ? true : false;
}
bool operator>=(const Unix_t &left, const Unix_t &right)
{
    return left.unix_t >= right.unix_t ? true : false;
}
bool operator==(const Unix_t &left, const Unix_t &right)
{
    return left.unix_t == right.unix_t ? true : false;
}
bool operator!=(const Unix_t &left, const Unix_t &right)
{
    return left.unix_t != right.unix_t ? true : false;
}

void Unix_t::set_utime(uint64_t unix_t)
{
    this->unix_t = unix_t;
    utime = unix_t;
}

char* Unix_t::get_actuall_time() const
{
    Unix_t var(std::time(nullptr));
    return std::ctime(&utime);
}
```

А Файл main.cpp – data_tme

```
#include <iostream>
#include <ctime>
#include "data_time.cpp"
//#include <string> // for test commit

int main() {

    // тест конструктора по умолчанию
    Unix_t now;
    std::cout << now.get_time();

    // тест конструктора со входом unix time
    std::cout << "################################\n\n";

    uint64_t heh = 1618000095;
    Unix_t tmp(heh);
    std:: cout << tmp.get_time();
    std::cout << "Year" << tmp.get_year()<<std::endl;
    std::cout << "Month: "<< tmp.get_month()<< std::endl;
    std::cout << "Day: "<<  tmp.get_day() << std::endl;
    std::cout << "Time: "<<  tmp.get_hour() << ":";
    std::cout <<  tmp.get_min() << ":";
    std::cout <<  tmp.get_sec() << std::endl;
    std::cout << "unix time: "<< tmp.get_utime() << std::endl;
    std::cout << "################################\n\n";

    // тест конструктора со входом
    //года, месяца, дня, часа, минуты, секунды
    Unix_t mp(now);
    std:: cout << mp;
    mp -= now;
    std:: cout << mp.get_time();

    // сверка времени
    std::cout << "################################\n\n";
    std::time_t til = std::time(nullptr);
```

```
    std::cout << std::asctime(std::localtime(&til)) << "tmp = " << til << '\n';

}
```

Приложение Б

Б Файл new_integer.h

```
#ifndef new_integer_h
#define new_integer_h

#include <vector>
#include <string>
#include <iostream>
#include <cstdint>


class Integer
{
    private:

        // разбиение числа
        static const int64_t div_size = 1000000000;
        // хранение числа
        std::vector<int64_t> nums;
        std::string num_str;

        // знак числа
        bool neg;

        // служебные методы:
        std::string pull_str() const; // наполнение строки
        void rm_zero(); // удаляение первых нулей
        void mul_div_size(); // сдвиг разрядов

    public:

        Integer():neg(false){} // контруктор по умолчанию
```

```
        Integer(std::istream &in); // конструктор, принимающий объект istream типа
        Integer(std::string str); // конструктор, принимающий объект string типа
        Integer(int64_t var); // конструктор, принимающий объект int64_t типа
        Integer(const Integer &copy); // конструктор копирования

        std::string get_num_str() const { return this->num_str; } // getter
        void set_num_str(std::string num_str); // setter

        Integer &operator=(const Integer &right); // оператор присваивания
        Integer &operator=(std::string right); // оператор присваивания
        Integer &operator=(int64_t right); // оператор присваивания



        const Integer operator+() const; // обеспечение арифметики
        const Integer operator-() const; // обеспечение арифметики

        friend const Integer operator+(Integer left, const Integer &right); //
//оператор сложения
        Integer &operator+=(const Integer &right); // оператор увеличения числа

        friend const Integer operator-(Integer left, const Integer &right); //
//оператор вычитания
        Integer &operator-=(const Integer &right); // оператор уменьения числа

        const Integer operator++();   // префиксный инкремент
        const Integer operator++(int);// постфиксный инкремент
        const Integer operator--();   // префиксный декремент
        const Integer operator--(int);// постфиксный декремент

        friend const Integer operator*(const Integer &left, const Integer &right);
// оператор умножения
        Integer &operator*=(const Integer &right); // оператор умножения числа

        friend const Integer operator/(const Integer &left, const Integer &right);
// оператор деления
        Integer &operator /=(const Integer &right); // оператор деления числа

        friend std::istream &operator>>(std::istream &in, Integer &right); //
//вывод в поток
        friend std::ostream &operator<<(std::ostream& out, const Integer& right);
// ввод из потока
```

```
        friend bool operator==(const Integer &left, const Integer &right); //
//оператор сравнения на равенство

        friend bool operator<(const Integer &left, const Integer &right);  //
//оператор меньше

        friend bool operator!=(const Integer &left, const Integer &right); //
//оператор не равно

        friend bool operator<=(const Integer &left, const Integer &right); //
//оператор меньше либо равно

        friend bool operator>(const Integer &left, const Integer &right);  //
//оператор больше

        friend bool operator>=(const Integer &left, const Integer &right); //
//оператор больше либо равно


        operator std::string(); // оператор приведения к string

};


#endif //new_integer_h
```

Б Файл new_integer.cpp

```
#include <iostream>

#include <algorithm>

#include <cmath>

#include <sstream>


#include "new_integer.h"


// конструктор, принимающий объект istream типа

Integer::Integer(std::istream &in)

{

    std::string num_str;

    in >> num_str;

    this->num_str = num_str;

    if (num_str.length() == 0) this->neg = false;

    else

    {

        if (num_str[0] == '-')

        {

                  num_str = num_str.substr(1);

                this->neg = true;

            }

        else this->neg = false;
```

```
        for (int64_t i = num_str.length(); i > 0; i -= 9)
        {
            if (i < 9)
                this->nums.push_back(std::atoi(num_str.substr(0, i).c_str()));
            else
                this->nums.push_back(std::atoi(num_str.substr(i - 9, 9).c_str()));
        }
        this->rm_zero();
    }
}

// конструктор, принимающий объект string типа
Integer::Integer(std::string num_str) : num_str(num_str)
{
    if (num_str.length() == 0) this->neg = false;
    else
    {
        if (num_str[0] == '-')
        {
            this->neg = true;
                  num_str = num_str.substr(1);
            }
        else this->neg = false;
        for (int64_t i = num_str.length(); i > 0; i -= 9)
        {
            if (i < 9)
                this->nums.push_back(std::atoi(num_str.substr(0, i).c_str()));
            else
                this->nums.push_back(std::atoi(num_str.substr(i - 9, 9).c_str()));
        }
        this->rm_zero();
    }
}

// конструктор, принимающий объект int64_t типа
Integer::Integer(int64_t var)
{
    this->num_str = std::to_string(var);
    if (var < 0)
    {
```

```
        this->neg = true;
        var -= 2 * var;
    }
    else this->neg = false;
    do
    {
        this->nums.push_back(var % Integer::div_size);
        var /= Integer::div_size;
    } while (var != 0);
}

// конструктор копирования
Integer::Integer(const Integer &copy) :
    nums(copy.nums), num_str(copy.num_str), neg(copy.neg){}

// setter
void Integer::set_num_str(std::string num_str)
{
    this->num_str = num_str;
    Integer var(num_str);
    *this = var;
}

// оператор присваивания
Integer &Integer::operator=(const Integer &right)
{
    this->nums = right.nums;
    this->neg = right.neg;
    this->num_str = this->pull_str();
    return *this;
}

// оператор присваивания
Integer &Integer::operator=(std::string right)
{
    Integer var(right);
    return *this = var;
}

// оператор присваивания
```

```
Integer &Integer::operator=(int64_t right)
{
    Integer var(right);
    return *this = var;
}

// обеспечение арифметики
const Integer Integer::operator+() const { return Integer(*this); }

// обеспечение арифметики
const Integer Integer::operator-() const
{
      Integer copy(*this);
      copy.neg = !copy.neg;
      return copy;
}

// оператор сложения
const Integer operator+(Integer left, const Integer &right)
{
    if (left.neg)
    {
            if (right.neg) return -(-left + (-right));
            else return right - (-left);
      }
    else if (right.neg) return left - (-right);
    int var = 0; // переполнение
    for (int i = 0; i < std::max(left.nums.size(), right.nums.size()) || var != 0;
++i)
    {
        if (i == left.nums.size()) left.nums.push_back(0);

        if (i < right.nums.size()) left.nums[i] += var + right.nums[i];
        else left.nums[i] += var;

        if (left.nums[i] >= Integer::div_size) {left.nums[i] -= Integer::div_size;
var = 1;}
        else var = 0;
    }
    return left;
}
```

```
// оператор увеличения числа
Integer &Integer::operator+=(const Integer &right) { return *this = (*this +
right);}


// оператор вычитания
const Integer operator-(Integer left, const Integer &right)
{

    if (right.neg) return left + (-right);
      else if (left.neg) return -(-left + right);
      else if (left < right) return -(right - left);

    int var = 0;
    for (int i = 0; i < right.nums.size() || var != 0; ++i)
    {
        if (i < right.nums.size()) left.nums[i] -= var + right.nums[i];
        else left.nums[i] -= var;
        var = left.nums[i] < 0;
        if (var != 0) left.nums[i] += Integer::div_size;
    }

    left.rm_zero();
    return left;
}

// оператор уменьения числа
Integer &Integer::operator -=(const Integer &right) {
    return *this = (*this - right);
}

// префиксный инкремент
const Integer Integer::operator++()
{
    *this += 1;
    return *this;
}

// постфиксный инкремент
const Integer Integer::operator++(int)
```

```
{
    *this += 1;
    return (*this - 1);
}

// префиксный декремент
const Integer Integer::operator--()
{
    *this -= 1;
    return *this;
}

// постфиксный декремент
const Integer Integer::operator--(int)
{
    *this -= 1;
    return (*this + 1);
}

// оператор умножения
const Integer operator*(const Integer &left, const Integer &right)
{
      Integer result;
      result.nums.resize(left.nums.size() + right.nums.size());
    int64_t qwa = 0;
    int var = 0;
      for (int i = 0; i < left.nums.size(); ++i)
    {
            for (int j = 0; j < right.nums.size() || var != 0; ++j)
        {
            if (j < right.nums.size())
            {
                qwa =  result.nums[i + j] +
                        left.nums[i] * right.nums[j] + var;
            }
            else qwa = result.nums[i + j] +var;
                  result.nums[i + j] = static_cast<int>(qwa % Integer::div_size);
                  var = static_cast<int>(qwa / Integer::div_size);
            qwa = 0;
            }
```

```
        var = 0;
      }
    if (left.neg == right.neg) result.neg = false;
    else result.neg = true;
      result.rm_zero();
      return result;
}


// оператор умножения числа
Integer &Integer::operator*=(const Integer &right) { return *this = (*this *
right); }


// сдвиг разрядов
void Integer::mul_div_size()
{
      if (this->nums.size() == 0)
    {
            this->nums.push_back(0);
            return;
      }
      this->nums.push_back( this->nums[this->nums.size() - 1] );
      for (int i = this->nums.size() - 2; i > 0; --i) this->nums[i] = this->nums[i
- 1];
      this->nums[0] = 0;
}


// оператор деления
const Integer operator/(const Integer &left, const Integer &right)
{
      if (right == 0) throw "Error: division by zero!";
      Integer qwa(right);
      qwa.neg = false;
      Integer var, result;
      result.nums.resize(left.nums.size());
      for (int64_t i = left.nums.size() - 1; i >= 0; --i)
    {
            var.mul_div_size();
            var.nums[0] = left.nums[i];
            var.rm_zero();
            int x = 0, y = 0, z = Integer::div_size;
            while (y <= z)
```

```
        {
                int s = (y + z) / 2;
                Integer a = qwa * s;
                if (a <= var)
            {
                    x = s;
                    y = s + 1;
                }
                else z = s - 1;
            }
            result.nums[i] = x;
            var = var - qwa * x;
      }

      result.neg = left.neg != right.neg;
      result.rm_zero();
      return result;
}

// оператор деления числа
Integer &Integer::operator/=(const Integer &right) { return *this = (*this /
right); }

// вывод в поток
std::ostream& operator<<(std::ostream& out, const Integer& right)
{
    return out << right.num_str;
}

// ввод из потока
std::istream& operator>>(std::istream &in, Integer &right)
{
    Integer var(in);
    right = var;
    return in;
}

// оператор сравнения на равенство
bool operator==(const Integer &left, const Integer &right)
{
```

```
    if (left.neg != right.neg) return false;
    if (left.nums.empty())
    {
        if (right.nums.empty() || (right.nums.size() == 1 && right.nums[0] == 0))
return true;
        else return false;
    }

    if (right.nums.empty())
    {
        if ( left.nums[0] == 0 && left.nums.size() == 1) return true;
        else return false;
    }

    if (left.nums.size() != right.nums.size()) return false;
    for (int i = 0; i < left.nums.size(); ++i) if (left.nums[i] != right.nums[i])
return false;

    return true;
}

// оператор меньше
bool operator<(const Integer &left, const Integer &right)
{
    if (left == right) return false;
    if (left.neg)
    {
        if (right.neg) return ((-right) < (-left));
        else return true;
    }
    else if (right.neg) return false;
    else
    {
        if (left.nums.size() != right.nums.size()) return left.nums.size() <
right.nums.size();
        else
        {
            for (int i = left.nums.size() - 1; i >= 0; --i)
            {
                if (left.nums[i] != right.nums[i]) return left.nums[i] <
right.nums[i];
            }
```

```
            return false;
        }
    }
}

// оператор не равно
bool operator!=(const Integer &left, const Integer &right)
{
    return !(left == right);
}

// оператор меньше либо равно
bool operator<=(const Integer &left, const Integer &right)
{
    return (left < right || left == right);
}

// оператор больше
bool operator>(const Integer &left, const Integer &right)
{
    return !(left <= right);
}

// оператор больше либо равно
bool operator>=(const Integer &left, const Integer &right)
{
    return !(left < right);
}

// оператор приведения к string
Integer::operator std::string()
{
    this->num_str = this->pull_str();
    return this->num_str;
}


// служебные функции:

// наполнение строки
```

```
std::string Integer::pull_str() const
{
    if (this->nums.empty()) return "0";
    else
    {
        std::stringstream var;
        if (this->neg) var << '-';
        var << this->nums.back();
        for (long long i = static_cast<long long>(this->nums.size()) - 2; i >= 0;
--i)
            {  var << this->nums[i]; }
        std::string qwa;
        var >> qwa;
        return qwa;
    }

}

// удаляение первых нулей
void Integer::rm_zero()
{
    while (this->nums.size() > 1 && this->nums.back() == 0) this->nums.pop_back();
    if (this->nums.size() == 1 && this->nums[0] == 0) this->neg = false;
}
```

Б main.cpp – new_integer

```
#include <stdexcept>
#include <iostream>
#include <string>
#include <cstdint>


//#include "integer.h"
#include "new_integer.cpp"


int main()
{
```

```
Integer a;
std::cin >> a;
std::cout << "################################\n";
Integer b("10000000");
std::cout << "b = " << b << '\n';
Integer c;
c = a + b;
std::cout << "a + b = " << c << '\n';

c = a - b;
std::cout << "a - b = " << c << '\n';

a += b;
std::cout << "a += b\t a = " << a << '\n';

a -= b;
std::cout << "a -= b\t a = " << a << '\n';

a *= b;
std::cout << "a *= b\t a = " << a << '\n';

a = 100;
b /= a;
std::cout << "b /= a\t a = 100, b = " << b << '\n';

std::cout << "################################\n";

std::cout << "a == b : " << std::boolalpha << (a == b) << '\n';
std::cout << "a != b : " << std::boolalpha << (a != b) << '\n';
std::cout << "a >= b : " << std::boolalpha << (a >= b) << '\n';
std::cout << "a <= b : " << std::boolalpha << (a <= b) << '\n';
std::cout << "a > b  : " << std::boolalpha << (a > b) << '\n';
std::cout << "a < b  : " << std::boolalpha << (a < b) << '\n';


std::cout << "################################\n";
Integer test("123456");
std::cout << "test = "<< test << '\n';
std::cout << "\n################################\n\n";
test.set_num_str("1111111111");
```

```
    std::cout << " set_num_str : "<< test << '\n';
    ++test;
    std::cout << "++test = "<< test << '\n';
    --test;
    std::cout << "--test = "<< test << '\n';
    test++;
    std::cout << "test++ = "<< test << '\n';
    test--;
    std::cout << "test-- = "<< test << '\n';

    //std::cout << "################################\n";
    //std::cout << "test = " << (number) << '\n';
    /*
    int64_t a = 120, b = 7, c = 1;
    c = a + b;
    std::cout << "a + b = " << c << '\n';
    c = a - b;
    std::cout << "a - b = " << c << '\n';
    c = a * b;
    std::cout << "a * b = " << c << '\n';
    c = a % b;
    std::cout << "a % b = " << c << '\n';
    c = a / b;
    std::cout << "a / b = " << c << '\n';
    */

}
```

Приложение В

В Файл Adam_time.h

```
#ifndef time_Adam_h
#define time_Adam_h

#include <iostream>


class Adam_t
```

```
{
    private:

        static const unsigned int GREG_ADAM_DIFF = 5508;
        static const unsigned int INDICT_SUN_MOON_MAX = 7980;

        int indict;
        int sun;
        int moon;

        int Adam_year;

        void to_Adam_year();
        void from_Adam_year();

    public:

        Adam_t(){} // пустой конструктор
        Adam_t(int Adam_year); // конструктор из года от Адама
        Adam_t(int indict, int sun, int moon); // конструктор в год от Адама
        Adam_t(const Adam_t &copy); // констуктор копирования


        int get_Adam_year() const; // Adam_year getter
        void set_Adam_year(int Adam_year); // Adam_year setter

        int get_indict() const; // indict getter

        int get_sun() const; // sun getter

        int get_moon() const; // moon getter

        int get_Greg() const; // Greg_year getter
        void set_Greg(int Greg_year); // Greg_year setter


        Adam_t operator=(const Adam_t &right); // оператор присваивания

        Adam_t operator+(const Adam_t &right) const; // оператор сложения
        Adam_t operator+=(const Adam_t &right); // оператор увеличения числа
```

```
        Adam_t operator-(const Adam_t &right) const; // оператор вычитания
        Adam_t operator-=(const Adam_t &right); // оператор уменьения числа

        const Adam_t operator++();   // префиксный инкремент
        const Adam_t operator++(int);// постфиксный инкремент
        const Adam_t operator--();   // префиксный декремент
        const Adam_t operator--(int);// постфиксный декремент

        friend std::istream &operator>>(std::istream &in, Adam_t &right); // вывод
//в поток
        friend std::ostream& operator<<(std::ostream& out, const Adam_t &right);
// ввод из потока
};

#endif //time_Adam_h
```

В Файл Adam_time.cpp

```
#include "time_Adam.h"

//в год от Адама
void Adam_t::to_Adam_year()
{
    for(size_t year = 1; year < INDICT_SUN_MOON_MAX; ++year)
    {
        Adam_t var(year);
        if (var.indict % 15 == this->indict && var.sun % 28 == this->sun &&
var.moon % 19 == this->moon)
        {
            this->Adam_year = var.Adam_year;
            break;
        }
    }
}
// из года от Адама
void Adam_t::from_Adam_year()
{
    if (!(this->Adam_year % 15)) this->indict = 15;
    else this->indict = this->Adam_year % 15;
```

```
    if (!(this->Adam_year % 28)) this->sun = 28;
    else this->sun = this->Adam_year % 28;

    if (!(this->Adam_year % 19)) this->moon = 19;
    else this->moon = this->Adam_year % 19;
}

// конструктор из года от Адама
Adam_t::Adam_t(int Adam_year) : Adam_year(Adam_year)
{
    from_Adam_year();
}

// конструктор в год от Адама
Adam_t::Adam_t(int indict, int sun, int moon) : indict(indict), sun(sun),
moon(moon)
{
    to_Adam_year();
}

// констуктор копирования
Adam_t::Adam_t(const Adam_t &copy)
{
    this->Adam_year = copy.Adam_year;
    this->indict = copy.indict;
    this->sun = copy.sun;
    this->moon = copy.moon;
}

// Adam_year getter
int Adam_t::get_Adam_year() const { return this->Adam_year; }

// Adam_year setter
void Adam_t::set_Adam_year(int Adam_year)
{
    Adam_t var(Adam_year);
    *this = var;
}
```

```
// indict getter
int Adam_t::get_indict() const { return this->indict; }

// sun getter
int Adam_t::get_sun() const { return this->sun; }

// moon getter
int Adam_t::get_moon() const { return this->moon; }

// Greg_year getter
int Adam_t::get_Greg() const
{
    return this->Adam_year - GREG_ADAM_DIFF;
}

// Greg_year setter
void Adam_t::set_Greg(int Greg_year)
{
    this->Adam_year = Greg_year + GREG_ADAM_DIFF;
    from_Adam_year();
}

// оператор присваивания
Adam_t Adam_t::operator=(const Adam_t &right)
{
    this->indict = right.indict;
    this->sun = right.sun;
    this->moon = right.moon;
    this->Adam_year = right.Adam_year;
    return *this;
}

// оператор сложения
Adam_t Adam_t::operator+(const Adam_t &right) const
{
    Adam_t var(*this);
    var.Adam_year += right.Adam_year;
    if (this->Adam_year > INDICT_SUN_MOON_MAX) throw "Error: overflow Adams
time!";
    var.from_Adam_year();
```

```
    return var;
}

// оператор увеличения числа
Adam_t Adam_t::operator+=(const Adam_t &right)
{
    *this = *this + right;
    return *this;
}

// оператор вычитания
Adam_t Adam_t::operator-(const Adam_t &right) const
{
    Adam_t var(*this);
    var.Adam_year -= right.Adam_year;
    if (this->Adam_year < 0) throw "Error: minus Adams time!";
    var.from_Adam_year();
    return var;
}

// оператор уменьения числа
Adam_t Adam_t::operator-=(const Adam_t &right)
{
    *this = *this - right;
    return *this;
}

// префиксный инкремент
const Adam_t Adam_t::operator++()
{
    *this += 1;
    return *this;
}

// постфиксный инкремент
const Adam_t Adam_t::operator++(int)
{
    *this += 1;
    return (*this - 1);
}
```

```
// префиксный декремент
const Adam_t Adam_t::operator--()
{
    *this -= 1;
    return *this;
}

// постфиксный декремент
const Adam_t Adam_t::operator--(int)
{
    *this -= 1;
    return (*this + 1);
}

// вывод из потока
std::istream &operator>>(std::istream &in, Adam_t &right)
{
    in >> right.Adam_year;
    right.from_Adam_year();
    return in;
}

// ввод в поток
std::ostream& operator<<(std::ostream& out, const Adam_t &right)
{
    return out << right.Adam_year;
}
```

В Файл main.cpp – Adam_time

```
#include <iostream>

#include "time_Adam.cpp"

int main()
{
    Adam_t test1;
    std::cout << "###############################################\n";
```

```
    test1.set_Greg(2021);
    std::cout << "set_Greg(2021)\ttest1 = "<< test1 << '\n';
    std::cout << "###############################################\n";

    Adam_t test2(7528);
    std::cout << "test2(7528)\ttest2 = "<< test2 << '\n';
    std::cout << "get_Greg()\ttest2 = "<< test2.get_Greg() << '\n';
    std::cout << "###############################################\n";

    Adam_t test3(1);
    test2 = test2 + test3;
    std::cout << "test2 = test2 + test3\t:test2 = " << test2 << '\n';
    test2 = test2 - test3;
    std::cout << "test2 = test2 - test3\t:test2 = " << test2 << '\n';
    test2 += test3;
    std::cout << "test2 += test3\t:test2 = " << test2 << '\n';
    test2 -= test3;
    std::cout << "test2 -= test3\t:test2 = " << test2 << '\n';
    std::cout << "###############################################\n";

    Adam_t test4(5,20,14);
    std::cout << "test4(5,20,14)\ttest4 = "<< test4 << '\n';
    std::cout << "get_Greg()\ttest4 = "<< test4.get_Greg() << '\n';
    std::cout << "###############################################\n";

    return 0;
}
```

Приложение Г

Г Файл spare_matrix.h

```
#ifndef matrix_h
#define matrix_h

#include <iostream>
#include <vector>
```

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



class SMatrix
{
    public:
        struct Kof
        {
            size_t x = 0;
            size_t y = 0;
            double value = 0.0;
            Kof(){}
            Kof(int x, int y, double value) :
            x(x), y(y), value(value){}
            Kof(const Kof &right) :
            x(right.x), y(right.y), value(right.value){}
            ~Kof(){}
        };
    private:
        size_t rows = 1;
        size_t cols = 1;
        std::vector<Kof> full;

    public:
        SMatrix(){}
        SMatrix(std::istream &in);
        SMatrix(Kof *a, size_t size);
        SMatrix(std::vector<Kof> &a);
        SMatrix(const SMatrix &right); // copy constructor
        ~SMatrix(){}

        std::ostream & writeSMatrix(std::ostream &out = std::cout) const;
        void add_kof(Kof a);
        void add_val(int x, int y, double value);
        void set_val(int x, int y, double value);
        double get_val(int x, int y);
```

```
        SMatrix operator=(const SMatrix & right) ;
        SMatrix operator+=(const SMatrix & right);
        SMatrix operator-=(const SMatrix & right);
        SMatrix operator+(const SMatrix & right) const;
        SMatrix operator-(const SMatrix & right) const;

        friend std::ostream & operator<< (std::ostream &left, const SMatrix
&right);
        friend std::istream & operator>> (std::istream &left, SMatrix &right);
};

#endif // matrix_h
```

Γ spare_matrix.cpp

```
#include <iostream>
#include <algorithm>

#include "spare_matrix.h"

SMatrix::SMatrix(std::istream &in ) {
    full.reserve(4);
    int x = 0;
    int y = 0;
    double var = 0;
    do
    {
        in >> x;
        if (!x) break;
        if (x > this->rows) this->rows = x;

        in >> y;
        if (!y) break;
        if (y > this->cols) this->cols = y;

        in >> var;
        if ( var <= 0 + 0.001 && var >= 0 - 0.001 ) continue;

        Kof a(x,y,var);
        full.push_back(a);
```

```
    } while ( x && y && full.size() < 10000);
}

SMatrix::SMatrix(const SMatrix &right)
{
    this->rows = right.rows;
    this->cols = right.cols;
    this->full = right.full;
}

SMatrix::SMatrix(Kof *a, size_t size)
{
    full.reserve(size);
    for(int i = 0; i < size; ++i) this->add_kof(a[i]);
}

SMatrix::SMatrix(std::vector<Kof> &a)
{
    full = a;
    for(int i = 0; i < full.size(); ++i)
    {
        if(full[i].x > this->rows) this->rows = full[i].x;
        if(full[i].y > this->cols) this->cols = full[i].y;
    }
}

std::ostream & SMatrix::writeSMatrix(std::ostream &out ) const
{
    bool flg = false;
    for(int i = 1; i < this->rows+1; ++i)
    {
        for(int j = 1; j < this->cols+1; ++j)
        {
            for(int k = 0; k < full.size(); ++k)
            {
                if (full[k].x == i && full[k].y == j)
                {
                    out << full[k].value << ' ';
                    flg = true;
```

```
                    break;
                }
            }
            if (!flg)
            {
                out << 0.0 << ' ';
            }
            flg = false;
        }
        out << '\n';
    }
    return out;
}

void SMatrix::add_kof(Kof a)
{
    this->full.push_back(a);
    if(full[full.size()-1].x > this->rows) this->rows = full[full.size()-1].x;
    if(full[full.size()-1].y > this->cols) this->cols = full[full.size()-1].y;
}

void SMatrix::add_val(int x, int y, double value)
{
    Kof a(x,y,value);
    this->full.push_back(a);
    if(full[full.size()-1].x > this->rows) this->rows = full[full.size()-1].x;
    if(full[full.size()-1].y > this->cols) this->cols = full[full.size()-1].y;
}

void SMatrix::set_val(int x, int y, double value)
{
    for(int i = 0; i < full.size(); ++i)
    {
        if(full[i].x == x && full[i].y == y)
        {
            full[i].value = value;
            return;
        }
    }
    this->add_val(x,y,value);
```

```
}

SMatrix SMatrix::operator=(const SMatrix & right)
{
    this->rows = right.rows;
    this->cols = right.cols;
    this->full = right.full;
    return *this;
}

SMatrix SMatrix::operator+(const SMatrix & right) const
{
    SMatrix var(*this);
    var.rows = std::max(this->rows, right.rows);
    var.cols = std::max(this->cols, right.cols);
    bool flg = true;
    for(int i = 0; i < right.full.size(); ++i)
    {
        for(int j = 0; j < this->full.size(); ++j)
        {
            if(right.full[i].x == var.full[j].x &&
               right.full[i].y == var.full[j].y)
               {
                   var.full[j].value += right.full[i].value;
                   flg = false;
                   break;
               }
        }
        if (flg) var.full.push_back(right.full[i]);
        flg = true;
    }
    return var;
}

SMatrix SMatrix::operator+=(const SMatrix & right)
{
    *this = this->operator+(right);
    return *this;
}
```

```
SMatrix SMatrix::operator-(const SMatrix & right) const
{
    SMatrix var(*this);
    var.rows = std::max(this->rows, right.rows);
    var.cols = std::max(this->cols, right.cols);
    bool flg = true;
    for(int i = 0; i < right.full.size(); ++i)
    {
        for(int j = 0; j < this->full.size(); ++j)
        {
            if(right.full[i].x == var.full[j].x &&
               right.full[i].y == var.full[j].y)
               {
                   var.full[j].value -= right.full[i].value;
                   flg = false;
                   break;
               }
        }
        if (flg)
        {
            var.full.push_back(right.full[i]);
            var.full[var.full.size()-1].value -= 2 * var.full[var.full.size()-
1].value;
        }
        flg = true;
    }
    return var;
}

SMatrix SMatrix::operator-=(const SMatrix & right)
{
    *this = this->operator-(right);
    return *this;
}

std::ostream & operator<< (std::ostream &out, const SMatrix &right)
{
    return right.writeSMatrix(out);
}
```

```
std::istream & operator>> (std::istream &left, SMatrix &right)
{
    SMatrix a(left);
    right = a;
    return left;
}
```

Г – Файл main.cpp – spare_matrix

```
#include <iostream>
#include "spare_matrix.cpp"
int main()
{
    SMatrix a;
    std::cin >> a;
    std::cout << "a = \n" << a;
    std::vector<SMatrix::Kof> var = {{1,1,11},{2,2,22},{3,3,33}};
    SMatrix b(var);
    std::cout << "\n##############################\n";
    std::cout << "b = \n" << b;
    std::cout << "\n##############################\n";
    a.add_val(1,1,33);
    std::cout << "now a = \n" << a;
    std::cout << "\n##############################\n";
    a -= b;
    std::cout << "a - b =  \n" << a;
    a += b;
    a = a + b;
    std::cout << "a + b =  \n" << a;
    return 0;
}
```