

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ  
МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ  
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

## ЛАБОРАТОРНАЯ РАБОТА №1 по курсу объектно-ориентированное программирование I семестр, 2021/22 уч. год

Студент Молчанов Владислав Дмитриевич, группа М80-208Б-20  
Преподаватель Дорохов Евгений Павлович

### **Цель:**

- Изучение системы сборки на языке C++, изучение систем контроля версии.
- Изучение основ работы с классами в C++;

### **Порядок выполнения работы**

1. Ознакомиться с теоретическим материалом.
2. Получить у преподавателя вариант задания.

3. Реализовать задание своего варианта в соответствии с поставленными требованиями.
4. Подготовить тестовые наборы данных.
5. Создать репозиторий на GitHub.
6. Отправить файлы лабораторной работы в репозиторий.
7. Отчитаться по выполненной работе путём демонстрации работающей программы на тестовых наборах данных (как подготовленных самостоятельно, так и предложенных преподавателем) и ответов на вопросы преподавателя (как из числа контрольных, так и по реализации программы).

### Требования к программе

Разработать программу на языке C++ согласно варианту задания. Программа на C++ должна собираться с помощью системы сборки CMake. Программа должна получать данные из стандартного ввода и выводить данные в стандартный вывод.

Необходимо настроить сборку лабораторной работы с помощью CMake. Собранная программа должна называться **oop\_exercise\_01** (в случае использования Windows **oop\_exercise\_01.exe**)

Необходимо зарегистрироваться на GitHub (если студент уже имеет регистрацию на GitHub то можно использовать ее) и создать репозиторий для задания лабораторной работы.

Преподавателю необходимо предъявить ссылку на публичный репозиторий на Github. Имя репозитория должно быть [https://github.com/login/oop\\_exercise\\_01](https://github.com/login/oop_exercise_01)

Где login – логин, выбранный студентом для своего репозитория на Github.

Репозиторий должен содержать файлы:

- main.cpp // файл с заданием работы
- CMakeLists.txt // файл с конфигурацией CMake
- test\_xx.txt // файл с тестовыми данными. Где xx – номер тестового набора 01, 02, ... Тестовых наборов должно быть больше 1.
- report.doc // отчет о лабораторной работе

## Описание программы

Исходный код лежит в 3 файлах:

1. main.cpp - исполняемый код.
2. Long.h - специальный файл .h, содержащий прототипы используемых мною функций.
3. Long.cpp - реализация функций для моего задания.
4. CMakeLists.txt - специальный дополнительный файл типа CMakeLists.

## Дневник отладки

Во время выполнения лабораторной работы программа не нуждалась в

отладке, все ошибки компиляции были исправлены с первой попытки. После их исправления программа работала так, как было задумано изначально.

## **Недочёты**

Недочётов не было обнаружено.

## **Выводы**

Данная лабораторная работа помогла мне использовать полученные на лекциях теоретические знания на практике, и я написал простенький полностью работающий класс.

## **Исходный код**

Long.h

```
#ifndef LONG_H
#define LONG_H
#include <iostream>
```

```

using namespace std;
class Long {
public:
    Long();
    Long(unsigned long long x, unsigned long long y);
    Long(istream &is);
    void Display();
    string toString();
    friend bool operator>(Long& x, Long& y);
    friend bool operator<(Long& x, Long& y);
    friend bool operator==(Long& x, Long& y);
    friend bool operator!=(Long& x, Long& y);
    friend istream& operator>>(istream& is, Long& a);

    void add(Long a, Long b);
    void diff(Long a, Long b);
    void mov(Long a, Long b);
    void sub(Long a, Long b);
    // ~Long();
private:
    unsigned int first;
    unsigned int second;
};
#endif

```

## Long.cpp

```

#include "Long.h"
#include <bits/stdc++.h>
using namespace std;
Long::Long(){
    first = 0;
    second = 0;
    cout << "Default has been created" << endl;
}

Long::Long(unsigned long long x, unsigned long long y){

```

```

    first = x;
    second = y;
}

Long::Long(istream &is){
    cout << "Enter all data" << endl;
    cin >> first;
    cin >> second;
}

istream& operator>>(istream& is, Long& x) {
    is >> x.first >> x.second;
    return is;
}

void Long::add(Long x, Long y){
    first = x.first + y.first;
    second = x.second + y.second;
    if(second > 4294967295){
        first += 1;
        second = second - 999999999 + 1;
    }
}

void Long::diff(Long x, Long y){
    if(x.first > y.first && x.second > y.second){
        first = x.first - y.first;
        second = x.second - y.second;
    }
    else if(x.first > y.first && x.second < y.second){
        string str1 = to_string(y.second);
        int cnt = 0;
        for(int i = 0; i < str1.size(); i++){
            cnt += 1;
        }
        int sec = y.second - x.second;
        int ans = 10 * cnt - sec;
        first = x.first - y.first - 1;
        second = ans;
    }
    else if(x.first < y.first && x.second > y.second){
        diff(y, x);
    }
    else if(x.first < y.first && x.second < y.second){
        first = y.first - x.first;
        second = y.second - x.second;
    }
}

void Long::mov(Long x, Long y){
    string strx1 = to_string(x.first);
    string strx2 = to_string(x.second);
    string x_ = strx1 + strx2;
}

```

```

string stry1 = to_string(y.first);
string stry2 = to_string(y.second);
string y_ = stry1 + stry2;

long long p = stoll(x_);
long long q = stoll(y_);
long long r = p * q;

string ans = to_string(r);
int h = ans.size() / 2 + 1;
char f[h - 1];
char s[h];
for(int i = 0; i < h; i++){
    if(h > 1){
        f[i] = ans[i];
        s[i] = ans[h + i];
    }
    else{
        f[i] = 0;
        s[i] = ans[i];
    }
}
first = atoi(f);
second = atoi(s);
}

```

```

void Long::sub(Long x, Long y){
    string strx1 = to_string(x.first);
    string strx2 = to_string(x.second);
    string x_ = strx1 + strx2;

    string stry1 = to_string(y.first);
    string stry2 = to_string(y.second);
    string y_ = stry1 + stry2;

    long long p = stoll(x_);
    long long q = stoll(y_);
    long long r = p / q;

    string ans = to_string(r);
    int h = ans.size() / 2;
    char f[h];
    char s[h+1];
    for(int i = 0; i <= h; i++){
        if(h > 1){
            f[i] = ans[i];
            s[i] = ans[h - i];
        }
        else{
            f[i] = 0;
            s[i] = ans[i];
        }
    }
}

```

```

    }
    first = atoi(f);
    second = atoi(s);

}

void Long::Display(){
    cout << toString() << endl;
}

string Long::toString(){ // Первод в строку
    string str1 = to_string(first);
    string str2 = to_string(second);
    string str = str1 + str2;
    return str;
}

bool operator> (Long& x, Long& y){
    if(x.first > y.first){
        return true;
    }
    else{
        return false;
    }
}

bool operator< (Long& x, Long& y){
    if(x.first > y.first){
        return false;
    }
    else{
        return true;
    }
}

bool operator== (Long& x, Long& y){
    if(x.first == y.first && x.second == y.second){
        return true ;
    }
    else{
        return false;
    }
}

bool operator!= (Long& x, Long& y){
    if(x.first == y.first && x.second == y.second){
        return false;
    }
    else{
        return true;
    }
}

```

main.cpp



```
#include "Long.h"
```

```
int main(){  
    Long a(cin);  
    Long b(cin);  
    Long c(0,0);  
    if(a != b){  
        cout << "Success";  
    }  
    else{  
        cout << "Fail";  
    }  
}
```