**МИНИСТЕРСТВО науки и высшего ОБРАЗОВАНИЯ РОссИЙСКОЙ ФЕДЕРАЦИИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ**

**УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ**

(национальный исследовательский университет)»

Институт №3 «Системы управления, информатика и электроэнергетика»

Кафедра № 304 «Вычислительные машины, системы и сети»

Защита информации

Отчет по лабораторной работе № 1

Работа с большими числами

Выполнил студент группы М3О-409Б-20:

Мокшин И. Н.

Проверил, Ведьманов И. С.

Москва 2023 г.

Оглавление

[1. Код программы 3](#_Toc153353661)

[2. Результат работы 5](#_Toc153353662)

# Код программы

#pragma once

#include <vector>

#include <algorithm>

#include <iostream>

#include <vector>

#include <cmath>

#include <iomanip>

using namespace std;

constexpr auto dec\_mod = 10000;

class BigInt {

private:

vector<int> parts;

void increase\_digit(int, int);

public:

BigInt() : parts({ 0 }) {};

BigInt(int n) : parts({ 0 }) { increase\_digit(0, n); };

~BigInt() {};

BigInt& operator\*(int x);

BigInt& operator+(const BigInt& rhs) const;

BigInt& operator\*(const BigInt& rhs) const;

BigInt pow(long power);

friend ostream& operator<<(ostream&, const BigInt&);

};

BigInt BigInt::pow(long power) {

BigInt newVal = \*this;

BigInt initial = \*this;

for (long i = 0; i < power; i++)

{

newVal = newVal \* initial;

}

return newVal;

}

void BigInt::increase\_digit(int idx, int n) {

if (parts.size() <= idx) parts.resize(idx + 1);

parts[idx] += n;

int mod\_g = parts[idx] % dec\_mod;

int extra = parts[idx] / dec\_mod;

parts[idx] = mod\_g;

if (extra > 0) {

increase\_digit(idx + 1, extra);

}

}

BigInt& BigInt::operator\*(int x) {

int extra = 0;

for (auto& g : parts) {

g = g \* x + extra;

int mod\_g = g % dec\_mod;

extra = g / dec\_mod;

g = mod\_g;

}

while (extra > 0) {

parts.push\_back(extra % dec\_mod);

extra /= dec\_mod;

}

return \*this;

}

BigInt& BigInt::operator\*(const BigInt& rhs) const {

BigInt\* mul = new BigInt();

int size\_lhs = parts.size();

int size\_rhs = rhs.parts.size();

for (int i\_lhs = 0; i\_lhs < size\_lhs; i\_lhs++) {

for (int i\_rhs = 0; i\_rhs < size\_rhs; i\_rhs++) {

mul->increase\_digit(i\_lhs + i\_rhs, parts[i\_lhs] \* rhs.parts[i\_rhs]);

}

}

return \*mul;

}

BigInt& BigInt::operator+(const BigInt& rhs) const {

BigInt& sum = const\_cast<BigInt&>(\*this);

int size\_rhs = rhs.parts.size();

for (int i = 0; i < size\_rhs; i++) {

sum.increase\_digit(i, rhs.parts[i]);

}

return sum;

}

ostream& operator<<(ostream& os, const BigInt& bi) {

os.fill('0');

for (auto pg = bi.parts.rbegin(); pg != bi.parts.rend(); pg++) {

if (pg == bi.parts.rbegin()) {

os << \*pg;

}

else {

os << std::setw(4) << \*pg;

}

}

return os;

}

void lr1() {

BigInt a(21);

BigInt b(131);

a = a.pow(21);

b = b.pow(41);

BigInt c = a + b;

std::cout << c.pow(2) << std::endl;

}

# Результат работы

Рассчёт *(2121+13141)2*

597287261625516521814629178473908115983764072909168194916788334015618415781151645826477554174496231008586750909405991532941534347969025095861362067606840824218213643204326055883475444115304206643261554403689279173798931499977998675740084093083502363748412442444990408

Число гораздо больше любого встроенного числового типа. Была проверена верность расчёта на другой (проверенной) реализации больших чисел.