ЗВІТ З ЛАБОРАТОРНОЇ РОБОТИ №2

за курсом «Обєктно-орієнтоване прорамування»

студентка групи ПА-20-1з

Мовсісян Лаури Ростомівни

кафедра комп’ютерних технологій, ДНУ

2021/2022

1. Постановка задачі

Розробити об’єктно-орієнтовану бібліотеку для роботи зі структурами даних за однією з нижче наведених тем у відповідності з нижче наведеними вимогами. Властивості та методи для класів розробити у відповідності з відомими визначеннями відповідних структур даних. Скласти тести для перевірки працездатності бібліотеки. Скласти програму, що демонструє можливості розробленої бібліотеки.

Загальні вимоги

1. Реалізація методів ініціалізації (конструктор по замовчуванню та конструктор з параметрами), копіювання (конструктор копіювання), індексації (перевантаження []),

присвоювання (перевантаження =), візуалізації, збереження (на диск) та відновлення,

діалогового керування, "розумного доступу" (перевантаження ->), а також псевдозмінних (забезпечення можливості виду: f(x)=const).

2. Перевантаження (спільне використання) потокового введення/виведення. (введення зфайл, виведення в файл)

3. Створення та використання файла бібліотеки (\*.LIB).

4. Повторне використання класів без їх перекомпіляції (ReUse).

5. Застосування вказаної структури даних для розв’язання типової задачі.

В ході демонстрації роботи, програма дозволяє користувачеві в діалоговому режимі (за допомогою меню) виконувати наступні операції:

1. Створювати об’єкти класу (у довільній кількості).

2. Вводити/виводити об’єкти на диск.

3. Візуалізувати об’єкти.

4. Виконувати операції над об’єктами.

Тема 14:Довга арифметика: реалізація основних арифметичних операцій.

2. Опис розв’язку

### Опис бібліотеки superint

Бібліотека складається з декількох класів SuperInt, const\_position, position, interval. Клас SuperInt - це структура даних яка зберігає та обробляє велике число, причому число зберігаеться у массиві з чисел по 9 разрядів і більш старші розряди знаходяться на початку масиву, тобто перші 9 розрядів у кінці масиву, наступні 9 ближче до початку і так далі. Класи const\_position і position призначені для переходу до деякого розряду та зміні цифри у цьому розряді. Клас interval призначений для обробки частини великого числа і реалізовано це через ітерації по розрядам.

##### Поля

1. vector<int> a; - массив із чисел по 9 розрядів
2. char sign; - знак числа: -1 - від'ємне, 1 - додатне число

##### Публічні функції

1. SuperInt()
   1. Конструктор у якому створюється пусте число, тобто нуль.
   2. Не має аргументів.
   3. Нічого не виводить.
2. SuperInt(const SuperInt& x)
   1. Конструктор копіювання.
   2. Має один аргумент-посилання на інше число x, яке потрібно скопіювати.
   3. Нічого не виводить.
3. SuperInt(const string& str)
   1. Конструктор неявного перетворення із текстової строки.
   2. Має один аргумент-посилання на строку str, у якій зберігається число.
   3. Нічого не виводить.
4. SuperInt(const char \* str)
   1. Конструктор неявного перетворення із текстової строки.
   2. Має один аргумент-вказівник на строку str, у якій зберігається число.
   3. Нічого не виводить.
5. SuperInt(const int& right)
   1. Конструктор неявного перетворення із числа.
   2. Має один аргумент-посилання на число right.
   3. Нічого не виводить.
6. SuperInt& operator=(const SuperInt& right)
   1. Оператор присвоювання іншого числа.
   2. Має один аргумент-посилання на інше число right, яке потрібно скопіювати.
   3. Виводить посилання на себе.
7. SuperInt& operator=(const string& right)
   1. Оператор присвоювання іншого числа перетворенням із строки.
   2. Має один аргумент-посилання на строку str, у якій зберігається число.
   3. Виводить посилання на себе.
8. SuperInt& operator=(const char \* str)
   1. Оператор присвоювання іншого числа перетворенням із строки.
   2. Має один аргумент-посилання на строку str, у якій зберігається число.
   3. Виводить посилання на себе.
9. SuperInt& operator=(const int& right)
   1. Оператор присвоювання іншого числа.
   2. Має один аргумент-посилання на число right.
   3. Виводить посилання на себе.
10. position operator[](const size\_t& digit)
    1. Оператор індексації, виводить розряд числа у вигляді класу position.
    2. Має один аргумент-посилання на номер розряду right.
    3. Виводить клас position.
11. const\_position operator[](const size\_t& digit) const
    1. Оператор індексації константного класу, виводить розряд числа у вигляді класу const\_position.
    2. Має один аргумент-посилання на номер розряду right.
    3. Виводить клас const\_position.
12. SuperInt& operator()(const SuperInt& right)
    1. Оператор псевдозмінної для копіювання числа right.
    2. Має один аргумент-посилання на інше число right, яке потрібно скопіювати.
    3. Виводить посилання на себе.
13. interval operator()(const size\_t& begin, const size\_t& length)
    1. Оператор псевдозмінної для визначення інтервалу подальшої обробки числа.
    2. Аргументи:
       1. begin - номер розряду з якого починати інтервал.
       2. length - кількість розрядів які входять у інтервал після begin.
    3. Виводить клас interval.
14. SuperInt& operator()(const SuperInt& right, const size\_t& begin, const size\_t& length)
    1. Оператор псевдозмінної для копіювання числа на інтервалі.
    2. Аргументи:
       1. right - число яке потрібно скопіювати
       2. begin - номер розряду з якого починати інтервал.
       3. length - кількість розрядів які входять у інтервал після begin.
    3. Виводить посилання на себе.
15. const string get\_string()
    1. Функція яка перетворює число у строку.
    2. Не має аргументів.
    3. Нічого не виводить.
16. int get\_digit(const size\_t& digit)
    1. Функція яка знаходить розряд числа.
    2. Має один аргумент-посилання на номер розряду digit.
    3. Виводить цифру у розряді digit.
17. int get\_digit(const size\_t& digit) const
    1. Константна функція яка знаходить розряд числа.
    2. Має один аргумент-посилання на номер розряду digit.
    3. Виводить цифру у розряді digit.
18. SuperInt& set\_digit(const size\_t& digit, const int& number)
    1. Функція яка знаходить розряд числа та змінює цифру у цьому розряді.
    2. Має один аргумент-посилання на номер розряду digit.
    3. Виводить посилання на себе.

##### Дружелюбні функції

1. SuperInt operator+(const SuperInt& left, const SuperInt& right);
   1. Оператор додавання двох чисел.
   2. Аргументи:
      1. left - посилання на перше число.
      2. right - посилання на друге число.
   3. Виводить нове число.
2. SuperInt operator-(const SuperInt& left, const SuperInt& right);
   1. Оператор віднімання двох числел.
   2. Аргументи:
      1. left - посилання на перше число.
      2. right - посилання на друге число.
   3. Виводить нове число.
3. SuperInt operator\*(const SuperInt& left, const SuperInt& right);
   1. Оператор множення двох числел.
   2. Аргументи:
      1. left - посилання на перше число.
      2. right - посилання на друге число.
   3. Виводить нове число.
4. SuperInt operator/(const SuperInt& left, const int& right);
   1. Оператор ділення великого числа на маленьке.
   2. Аргументи:
      1. left - посилання на велике число.
      2. right - посилання на маленьке число.
   3. Виводить нове число.
5. SuperInt operator/(const SuperInt& left, const SuperInt& right);
   1. Оператор ділення двох числел.
   2. Аргументи:
      1. left - посилання на перше число.
      2. right - посилання на друге число.
   3. Виводить нове число.
6. bool operator <(const SuperInt& left, const SuperInt& right);  
   bool operator >(const SuperInt& left, const SuperInt& right);  
   bool operator ==(const SuperInt& left, const SuperInt& right);  
   bool operator !=(const SuperInt& left, const SuperInt& right);  
   bool operator <=(const SuperInt& left, const SuperInt& right);  
   bool operator >=(const SuperInt& left, const SuperInt& right);
   1. Оператори порівняння двох чисел
   2. Аргументи:
      1. left - посилання на перше число.
      2. right - посилання на друге число.
   3. Виводить відповідь на питання чи виконується ця нерівність.
7. ostream& operator<<(ostream& output,const SuperInt& right);  
   istream& operator>>(istream& input, SuperInt& right);
   1. Оператори вводу та виводу з потоку.
   2. Аргументи:
      1. input/output - посилання на потоки вводу/виводу.
      2. right - посилання на число, яке потрібно вивести.
   3. Виводить посилання на поток з першого аргументу.

##### Приватні функції

1. char compare(const SuperInt& right) const;
   1. Константна функція порівняння поточного числа з числом right.
   2. Має єдиний аргумент-посилання на число.
   3. Виводить -1 - якщо поточне число менше за right, 0 - якщо дорівнює, 1 - якщо більше.
2. char compare\_module(const SuperInt& right) const;
   1. Константна функція порівняння поточного числа по модулю з числом right по модулю.
   2. Має єдиний аргумент-посилання на число.
   3. Виводить так само як і compare;
3. bool parse\_string(const string& str);
   1. Функція конвертує строку у число.
   2. Має єдиний аргумент-посилання на строку.
   3. Виводить успішність виконання.
4. bool validate(const string& str);
   1. Перевіряє чи можна зчитати число.
   2. Має єдиний аргумент-посилання на строку.
   3. Виводить валідність строки.
5. SuperInt operator\_sum\_sub(char s, const SuperInt& right) const;
   1. Функція визначає яку операцію потрібно зробити та у якій послідовності виконувати, так як алгоритм не може від меншого відняти більше.
   2. Аргументи:
      1. s - поточна операція -1 для віднімання, 1 для додавання.
      2. right - посилання на друге число.
   3. Виводить результат у вигляді нового числа
6. void summation(const vector<int>& b);
   1. Функція сумує числа по модулю, ігноруючи та не змінюючи знаки.
   2. Має єдиний аргумент-посилання на масив з великим числом.
   3. Нічого не виводить.
7. void subtraction(const vector<int>& b);
   1. Функція віднімає друг число по модулю від поточного по модулю, ігноруючи та не змінюючи знаки.
   2. Має єдиний аргумент-посилання на масив з великим числом, але меншим за поточне.
   3. Нічого не виводить.
8. void clear\_leading\_zeros();
   1. Функція видаляє непотрібні нулі перед числом.
   2. Не має аргументів.
   3. Нічого не виводить.

#### Клас position

Клас position призначений для зберігання поточного розряду для подальшої обробки цього розряду. Дружить з класом const\_position.

##### Поля

1. SuperInt& number - посилання на число
2. size\_t digit - номер розряду

##### Функції

1. position(SuperInt& number, const size\_t& digit)
   1. Конструктор з параметрами, потрібними для обробки числа.
   2. Аргументи:
      1. number - посилання на число.
      2. digit - номер розряду.
   3. Нічого не виводить.
2. SuperInt& operator=(const SuperInt& right)
   1. Оператор присвоювання, який копіює цифру на розряді digit з числа right на тому ж розряді.
   2. Має єдиний аргумент-посилання на число.
   3. Виводить посилання на поле number.
3. SuperInt& operator=(const position& right)  
   SuperInt& operator=(const const\_position& right)
   1. Оператор присвоювання, який замінює цифру на розряді digit на цифру на яку вказує right.
   2. Має єдиний аргумент-посилання на клас position.
   3. Виводить посилання на поле number.
4. SuperInt& operator=(const int& right)
   1. Оператор присвоювання, який замінює цифру на розряді digit на цифру right.
   2. Має єдиний аргумент-посилання на число.
   3. Виводить посилання на поле number.
5. int operator\*() int operator\*() const
   1. Оператори доступу до даних.
   2. Не мають аргументів.
   3. Виводять цифру з числа number на розряді digit.

##### Дружелюбні функції

1. ostream& operator<<(ostream& output, const position& right)  
   istream& operator>>(istream& input, position& right)
   1. Оператори вводу та виводу з потоку.
   2. Аргументи:
      1. input/output - посилання на потоки вводу/виводу.
      2. right - посилання на клас, який потрібно вивести.
   3. Виводить посилання на поток з першого аргументу.

#### Клас const\_position

##### Поля

1. SuperInt& number - посилання на число
2. size\_t digit - номер розряду

##### Функції

1. const\_position(const SuperInt& number, const size\_t& digit)
   1. Конструктор з параметрами, потрібними для обробки числа.
   2. Аргументи:
      1. number - посилання на число.
      2. digit - номер розряду.
   3. Нічого не виводить.
2. const\_position(const position& right)
   1. Конструктор неявного перетворення з position;
   2. Має єдиний аргумент-посилання на клас position.
   3. Нічого не виводить.
3. int operator\*()  
   int operator\*() const
   1. Оператори доступу до даних.
   2. Не мають аргументів.
   3. Виводять цифру з числа number на розряді digit.

##### Дружелюбні функції

1. ostream& operator<<(ostream& output, const const\_position& right)
   1. Оператор виводу з потоку.
   2. Аргументи:
      1. output - посилання на потоки виводу.
      2. right - посилання на клас, який потрібно вивести.
   3. Виводить посилання на поток з першого аргументу.

#### Клас interval

##### Поля

1. SuperInt& number - посилання на число.
2. size\_t begin - номер розряду з якого починається інтервал.
3. size\_t length - кількість розрядів в інтервалі.

##### Функції

1. interval(SuperInt& number, const size\_t& begin, const size\_t& length)
   1. Конструктор з параметрами, потрібними для обробки числа.
   2. Аргументи:
      1. number - посилання на число.
      2. begin - номер розряду з якого починається інтервал.
      3. length - кількість розрядів в інтервалі.
   3. Нічого не виводить.
2. SuperInt& operator=(const SuperInt& right)
   1. Оператор присвоювання який копіює цифри на заданому інтервалі з числа right.
   2. Має єдиний аргумент-посилання на число.
   3. Виводить посилання на поле number.
3. SuperInt operator\*()  
   SuperInt operator\*() const
   1. Оператори доступу до даних.
   2. Не мають аргументів.
   3. Виводять нове число починаючи з розряду begin на кількість розрядів length.

##### Дружелюбні функції

1. ostream &operator<<(ostream& output, const interval& right)  
   istream &operator>>(istream& input, interval& right)
   1. Оператори вводу та виводу з потоку.
   2. Аргументи:
      1. input/output - посилання на потоки вводу/виводу.
      2. right - посилання на клас, який потрібно вивести.
   3. Виводить посилання на поток з першого аргументу.

### Опис головної програми

Головна програма має дві функції для виводу масивом великих чисел та перевірки чи знаходиться строчка у звичайному масиві.

У головній функції виконуються такі дії:

1. Ініціалізуються необхідні змінні.
2. Запускається цикл доки не вийдуть з програми, у циклі виконуються такі дії
   1. Виводиться меню.
   2. Запитується вибір.
   3. Якщо потрібно виводиться масив
   4. Згідно з відповіддю запитуються додаткові необхідні данні та виконується дія.

3. Вихідний текст програми розв’язку задачі

CMakeLists.txt

cmake\_minimum\_required(VERSION 2.8)

project(superint)

add\_definitions ("-Wall")

#link\_directories(${GL\_LIBRARY\_)

include\_directories(${CMAKE\_CURRENT\_SOURCE\_DIR}/libsuperint ${CMAKE\_CURRENT\_SOURCE\_DIR}/src)

add\_library(superint SHARED libsuperint/superint.cpp)

add\_executable(main src/main.cpp src/tests.cpp)

target\_link\_libraries(main superint)

add\_executable(run\_tests src/run\_tests.cpp src/tests.cpp)

target\_link\_libraries(run\_tests superint)

main.cpp

#include "superint.h"

#include "tests.h"

#include <iostream>

#include <sstream>

using namespace std;

std::ostream &operator<<(std::ostream &output,

const vector<SuperInt::SuperInt>& right) {

output << "array(" << right.size() << ") {\n";

for (size\_t i = 0; i < right.size(); i++) {

output << "\t[" << i << "] => " << right[i] << endl;

}

output << "}\n";

return output;

}

bool is\_in(const string& str, const string arr[], const int& size) {

for (int i = 0; i < size; i++)

if (str == arr[i]) return 1;

return 0;

}

int main() {

srand(time(NULL));

string answer = "", buf;

vector<SuperInt::SuperInt> arr;

static const string show\_arr[] =

{"3", "5", "7", "8", "9", "10", "11", "12", "13", "14"};

static const int show\_arr\_size = 10;

size\_t tmp0 = 0, tmp1 = 0, tmp2 = 0, tmp3 = 0;

while (answer != "0") {

cout << " 1. Запустить тесты\n"

<< " 2. Добавить число в конец массива\n"

<< " 3. Удалить число из массива по индексу\n"

<< " 4. Сгенегировать 10 случайных чисел\n"

<< " 5. Вывести массив на экран\n"

<< " 6. Очистить массив\n"

<< " 7. Сложить два числа из массива\n"

<< " 8. Отнять два числа из массива\n"

<< " 9. Умножить два числа из массива\n"

<< "10. Поделить два числа из массива\n"

<< "11. Поделить число из массива на маленькое число\n"

<< "12. Переместить часть числа в другое число на той же позиции\n"

<< "13. Забрать разряд числа\n"

<< "14. Изменить цифру на разряде числа\n"

<< " 0. Выйти\n>>> ";

getline(cin, answer);

if (is\_in(answer, show\_arr, show\_arr\_size)) cout << arr;

if (answer == "0") exit(0);

else if (answer == "1") run\_tests();

else if (answer == "2") {

getline(cin, buf);

arr.push\_back(buf);

}

else if (answer == "3") {

getline(cin, buf);

tmp0 = atoll(buf.c\_str());

if (tmp0 < arr.size())

arr.erase(arr.begin() + tmp0);

else cout << "Неправильный индекс\n";

}

else if (answer == "4") {

for (int i = 0; i < 10; i++) {

stringstream ss;

if (rand() % 2) ss << "-";

tmp0 = rand() % 32;

for (size\_t j = 0; j < tmp0; j++)

ss << abs(rand() % 10);

arr.push\_back(ss.str());

}

cout << arr;

}

else if (answer == "5") {}

else if (answer == "6") arr.clear();

else if (answer == "7" || answer == "8"

|| answer == "9" || answer == "10") {

cout << "Введите индексы:\n>>> ";

cin >> tmp0 >> tmp1;

getline(cin, buf);

if (tmp0 >= arr.size() || tmp1 >= arr.size())

cout << "Неправильные индексы\n";

else if (answer == "7")

cout << arr[tmp0] + arr[tmp1] << endl;

else if (answer == "8")

cout << arr[tmp0] - arr[tmp1] << endl;

else if (answer == "9")

cout << arr[tmp0] \* arr[tmp1] << endl;

else if (answer == "10")

cout << arr[tmp0] / arr[tmp1] << endl;

} else if (answer == "11") {

cout << "Введите индекс:\n>>> ";

cin >> tmp0;

getline(cin, buf);

cout << "Введите число:\n>>> ";

cin >> tmp1;

getline(cin, buf);

cout << arr[tmp0] / tmp1 << endl;

} else if (answer == "12") {

cout << "Введите индексы:\n>>> ";

cin >> tmp0 >> tmp1;

getline(cin, buf);

if (tmp0 >= arr.size() || tmp1 >= arr.size())

cout << "Неправильные индексы\n";

else {

cout << "Введите начальный разряд и количество цифр:\n>>> ";

cin >> tmp2 >> tmp3;

getline(cin, buf);

cout << (arr[tmp0](tmp2, tmp3) = arr[tmp1]) << endl;

}

} else if (answer == "13") {

cout << "Введите индекс:\n>>> ";

cin >> tmp0;

getline(cin, buf);

cout << "Введите разряд:\n>>> ";

cin >> tmp1;

getline(cin, buf);

cout << arr[tmp0][tmp1] << endl;

} else if (answer == "14") {

cout << "Введите индекс:\n>>> ";

cin >> tmp0;

getline(cin, buf);

cout << "Введите разряд:\n>>> ";

cin >> tmp1;

getline(cin, buf);

cout << "Введите цифру:\n>>> ";

cin >> tmp2;

getline(cin, buf);

arr[tmp0][tmp1] = tmp2;

cout << arr[tmp0] << endl;

}

else cout << "Нет такой опции\n";

cout << "Нажмите ENTER для продолжения... ";

getline(cin, buf);

}

cout << "\n\n\n";

}

run\_tests.cpp

#include <iostream>

#include "superint.h"

#include "tests.h"

using namespace std;

int main() {

return run\_tests();

}

**tests.h**

#ifndef TESTS\_H

#define TESTS\_H

#include <iostream>

#include "superint.h"

using namespace std;

// Запустить тест с номером id

int test(int id);

// Вывести цветной статус

int test\_status(bool status);

// Запустить все тесты

int run\_tests();

#endif

### [tests.cpp](https://github.com/gaurapanasenko/unilab/blob/master/03/ProgrammingII_Lab02/src/tests.cpp)

#include "tests.h"

using namespace std;

int test(int id) {

int x = 1;

if (id == x++) {

string str = "1234";

cout << "Тест " << id << ": Инициализация.\n" << " Вводим \""

<< str << "\" в SuperInt::SuperInt и сравниваем вывод\n"

<< " объекта со стокой \"" << str << "\"\n";

SuperInt::SuperInt num(str);

cout << " Получили: " << num << endl;

return test\_status(num.get\_string() == str);

} else if (id == x++) {

string str = "1231231312312313-2312312313123";

string str2 = "0";

cout << "Тест " << id << ": Инициализация с неверной строкой.\n"

<< " Вводим \"" << str

<< "\" в SuperInt::SuperInt и сравниваем вывод\n"

<< " объекта со стокой \"" << str2 << "\"\n";

SuperInt::SuperInt num(str);

cout << " Получили: " << num << endl;

return test\_status(num.get\_string() == str2);

} else if (id == x++) {

string str = "1234";

cout << "Тест " << id << ": Присваивание.\n"

<< " Вводим \"" << str

<< "\" в SuperInt::SuperInt через присваивание и\n"

<< " сравниваем вывод объекта со стокой \"" << str << "\"\n";

SuperInt::SuperInt num;

num = str;

cout << " Получили: " << num << endl;

return test\_status(num.get\_string() == str);

} else if (id == x++) {

string str = "1234567891357924680";

cout << "Тест " << id << ": Ввод больших чисел.\n"

<< " Вводим \"" << str << "\" в SuperInt::SuperInt и\n"

<< " сравниваем вывод объекта со стокой \"" << str << "\"\n";

SuperInt::SuperInt num(str);

cout << " Получили: " << num << endl;

return test\_status(num.get\_string() == str);

} else if (id == x++) {

string str = "-1234567891357924680";

cout << "Тест " << id << ": Ввод отриательных больших чисел.\n"

<< " Вводим \"" << str << "\" в SuperInt::SuperInt и\n"

<< " сравниваем вывод объекта со стокой \"" << str << "\"\n";

SuperInt::SuperInt num(str);

cout << " Получили: " << num << endl;

return test\_status(num.get\_string() == str);

} else if (id == x++) {

string str = "-1234567891357924680";

cout << "Тест " << id << ": Конструктор копирования.\n"

<< " Вводим \"" << str

<< "\" в SuperInt::SuperInt и копируем в другой\n"

<< " SuperInt::SuperInt сравниваем вывод объекта со стокой \""

<< str << "\"\n";

SuperInt::SuperInt num(str);

SuperInt::SuperInt num2(num);

cout << " Получили: " << num2 << endl;

return test\_status(num.get\_string() == str);

} else if (id == x++) {

string str1 = "547509411375863014";

string str2 = "547509411375863014";

cout << "Тест " << id << ": Сравнение одинаковых чисел.\n"

<< " " << str1 << " == " << str2 << endl;

SuperInt::SuperInt num1(str1), num2(str2);

return test\_status(num1 == num2);

} else if (id == x++) {

string str1 = "547509411375863014";

string str2 = "-547509411375863014";

cout << "Тест " << id << ": Сравнение одинаковых по модулю чисел.\n"

<< " " << str1 << " > " << str2 << endl;

SuperInt::SuperInt num1(str1), num2(str2);

return test\_status(num1 > num2);

} else if (id == x++) {

string str1 = "547509411375863014";

string str2 = "47509411375863014";

cout << "Тест " << id << ": Сравнение разных чисел.\n"

<< " " << str1 << " > " << str2 << endl;

SuperInt::SuperInt num1(str1), num2(str2);

return test\_status(num1 > num2);

} else if (id == x++) {

string str1 = "547509411375863014";

string str2 = "54750941137586301";

cout << "Тест " << id << ": Сравнение разных чисел.\n"

<< " " << str1 << " > " << str2 << endl;

SuperInt::SuperInt num1(str1), num2(str2);

return test\_status(num1 > num2);

} else if (id == x++) {

string str1 = "-547509411375863014";

string str2 = "54750941137586301";

cout << "Тест " << id << ": Сравнение разных чисел.\n"

<< " " << str1 << " < " << str2 << endl;

SuperInt::SuperInt num1(str1), num2(str2);

return test\_status(num1 < num2);

} else if (id == x++) {

string str1 = "547509411375863014";

string str2 = "24440630276";

cout << "Тест " << id << ": Сравнение разных чисел.\n"

<< " " << str1 << " > " << str2 << endl;

SuperInt::SuperInt num1(str1), num2(str2);

return test\_status(num1 > num2);

} else if (id == x++) {

string str1 = "547509411375863014";

string str2 = "1";

cout << "Тест " << id << ": Сравнение разных чисел.\n"

<< " " << str1 << " > " << str2 << endl;

SuperInt::SuperInt num1(str1), num2(str2);

return test\_status(num1 > num2);

} else if (id == x++) {

string str1 = "12345";

string str2 = "23456";

string str3 = "35801";

cout << "Тест " << id << ": Сложение.\n"

<< " " << str1 << " + " << str2 << " = " << str3 << endl;

SuperInt::SuperInt num1(str1), num2(str2), num3;

num3 = num1 + num2;

cout << " Получили: " << num3 << endl;

return test\_status(num3.get\_string() == str3);

} else if (id == x++) {

string str1 = "547509411375863014";

string str2 = "24440630276";

string str3 = "547509435816493290";

cout << "Тест " << id << ": Сложение огромных чисел.\n"

<< " " << str1 << " + " << str2 << " = " << str3 << endl;

SuperInt::SuperInt num1(str1), num2(str2), num3;

num3 = num1 + num2;

cout << " Получили: " << num3 << endl;

return test\_status(num3.get\_string() == str3);

} else if (id == x++) {

string str1 = "-547509411375863014";

string str2 = "24440630276";

string str3 = "-547509386935232738";

cout << "Тест " << id << ": Сложение отрицательного числа и "

<< "положительного.\n" << " " << str1 << " + " << str2

<< " = " << str3 << endl;

SuperInt::SuperInt num1(str1), num2(str2), num3;

num3 = num1 + num2;

cout << " Получили: " << num3 << endl;

return test\_status(num3.get\_string() == str3);

} else if (id == x++) {

string str1 = "547509411375863014";

string str2 = "-24440630276";

string str3 = "547509386935232738";

cout << "Тест " << id << ": Сложение отрицательного числа и "

<< "положительного.\n" << " " << str1 << " + " << str2

<< " = " << str3 << endl;

SuperInt::SuperInt num1(str1), num2(str2), num3;

num3 = num1 + num2;

cout << " Получили: " << num3 << endl;

return test\_status(num3.get\_string() == str3);

} else if (id == x++) {

string str1 = "-12345";

string str2 = "-23456";

string str3 = "-35801";

cout << "Тест " << id << ": Сложение отрицательных чисел.\n"

<< " " << str1 << " + " << str2 << " = " << str3 << endl;

SuperInt::SuperInt num1(str1), num2(str2), num3;

num3 = num1 + num2;

cout << " Получили: " << num3 << endl;

return test\_status(num3.get\_string() == str3);

} else if (id == x++) {

string str1 = "547509411375863014";

string str2 = "24440630276";

string str3 = "547509386935232738";

cout << "Тест " << id << ": вычитание огромных чисел.\n"

<< " " << str1 << " - " << str2 << " = " << str3 << endl;

SuperInt::SuperInt num1(str1), num2(str2), num3;

num3 = num1 - num2;

cout << " Получили: " << num3 << endl;

return test\_status(num3.get\_string() == str3);

} else if (id == x++) {

string str1 = "-547509411375863014";

string str2 = "24440630276";

string str3 = "-547509435816493290";

cout << "Тест " << id << ": вычитание отрицательного числа и "

<< "положительного.\n" << " " << str1 << " - " << str2

<< " = " << str3 << endl;

SuperInt::SuperInt num1(str1), num2(str2), num3;

num3 = num1 - num2;

cout << " Получили: " << num3 << endl;

return test\_status(num3.get\_string() == str3);

} else if (id == x++) {

string str1 = "547509411375863014";

string str2 = "-24440630276";

string str3 = "547509435816493290";

cout << "Тест " << id << ": вычитание отрицательного числа и "

<< "положительного.\n" << " " << str1 << " - " << str2

<< " = " << str3 << endl;

SuperInt::SuperInt num1(str1), num2(str2), num3;

num3 = num1 - num2;

cout << " Получили: " << num3 << endl;

return test\_status(num3.get\_string() == str3);

} else if (id == x++) {

string str1 = "-12345";

string str2 = "-23456";

string str3 = "-11111";

cout << "Тест " << id << ": вычитание отрицательных чисел.\n"

<< " " << str1 << " - " << str2 << " = " << str3 << endl;

SuperInt::SuperInt num1(str1), num2(str2), num3;

num3 = num1 - num2;

cout << " Получили: " << num3 << endl;

return test\_status(num3.get\_string() == str3);

} else if (id == x++) {

string str1 = "547509411375863014";

string str2 = "24440630276";

string str3 = "13381475096067856395597011864";

cout << "Тест " << id << ": Умножение огромных чисел.\n"

<< " " << str1 << " \* " << str2 << " = " << str3 << endl;

SuperInt::SuperInt num1(str1), num2(str2), num3;

num3 = num1 \* num2;

cout << " Получили: " << num3 << endl;

return test\_status(num3.get\_string() == str3);

} else if (id == x++) {

string str1 = "-547509411375863014";

string str2 = "24440630276";

string str3 = "-13381475096067856395597011864";

cout << "Тест " << id << ": Умножение отрицательного числа и "

<< "положительного.\n" << " " << str1 << " \* " << str2

<< " = " << str3 << endl;

SuperInt::SuperInt num1(str1), num2(str2), num3;

num3 = num1 \* num2;

cout << " Получили: " << num3 << endl;

return test\_status(num3.get\_string() == str3);

} else if (id == x++) {

string str1 = "547509411375863014";

string str2 = "-24440630276";

string str3 = "-13381475096067856395597011864";

cout << "Тест " << id << ": Умножение отрицательного числа и "

<< "положительного.\n" << " " << str1 << " \* " << str2

<< " = " << str3 << endl;

SuperInt::SuperInt num1(str1), num2(str2), num3;

num3 = num1 \* num2;

cout << " Получили: " << num3 << endl;

return test\_status(num3.get\_string() == str3);

} else if (id == x++) {

string str1 = "-12345";

string str2 = "-23456";

string str3 = "289564320";

cout << "Тест " << id << ": Умножение отрицательных чисел.\n"

<< " " << str1 << " \* " << str2 << " = " << str3 << endl;

SuperInt::SuperInt num1(str1), num2(str2), num3;

num3 = num1 \* num2;

cout << " Получили: " << num3 << endl;

return test\_status(num3.get\_string() == str3);

} else if (id == x++) {

string str1 = "1000";

string str2 = "10";

string str3 = "100";

cout << "Тест " << id << ": Деление чисел.\n"

<< " " << str1 << " / " << str2 << " = " << str3 << endl;

SuperInt::SuperInt num1(str1), num2(str2), num3;

num3 = num1 / num2;

cout << " Получили: " << num3 << endl;

return test\_status(num3.get\_string() == str3);

} else if (id == x++) {

string str1 = "594881688386039612131121336848";

string str2 = "-48872148152877800551";

string str3 = "-12172202591";

cout << "Тест " << id << ": Деление огромных чисел.\n"

<< " " << str1 << " / " << str2 << " = " << str3 << endl;

SuperInt::SuperInt num1(str1), num2(str2), num3;

num3 = num1 / num2;

cout << " Получили: " << num3 << endl;

return test\_status(num3.get\_string() == str3);

} else if (id == x++) {

string str1 = "26339464348311100259";

string str2 = "21430594771958";

string str3 = "1229058";

cout << "Тест " << id << ": Деление огромных чисел.\n"

<< " " << str1 << " / " << str2 << " = " << str3 << endl;

SuperInt::SuperInt num1(str1), num2(str2), num3;

num3 = num1 / num2;

cout << " Получили: " << num3 << endl;

return test\_status(num3.get\_string() == str3);

}

return 0;

}

int run\_tests() {

int x = 0, y = 0, z = 0;

for (int i = 1; (x = test(i)); i++)

if (x == 1) y++; else z++;

cout << "Всего пройденных тестов: " << y + z << endl;

cout << "Количество тестов с ошибками: \033[0;31m" << y << "\033[0m\n";

cout << "Количество успешных тестов: \033[0;32m" << z << "\033[0m\n";

return y;

}

int test\_status(bool status) {

cout << " Статус: ";

if (status) cout << "\033[0;32mOK\033[0m\n";

else cout << "\033[0;31mERROR\033[0m\n";

return ((int)status) + 1;

}

### [superint.h](https://github.com/gaurapanasenko/unilab/blob/master/03/ProgrammingII_Lab02/libsuperint/superint.h)

#ifndef SUPERINT\_H

#define SUPERINT\_H

#include <ostream>

#include <string>

#include <vector>

#define SUPERINT\_CELL\_MAX 1000000000

using namespace std;

namespace SuperInt {

class interval;

class position;

class const\_position;

class SuperInt {

private:

vector<int> a;

char sign;

public:

SuperInt();

SuperInt(const SuperInt& x);

SuperInt(const string& str);

SuperInt(const char \* str);

SuperInt(const int& right);

SuperInt& operator=(const SuperInt& right);

SuperInt& operator=(const string& right);

SuperInt& operator=(const char \* str);

SuperInt& operator=(const int& right);

position operator[](const size\_t& digit);

const\_position operator[](const size\_t& digit) const;

SuperInt& operator()(const SuperInt& right);

interval operator()(const size\_t& begin, const size\_t& length);

SuperInt& operator()(const SuperInt& right, const size\_t& begin,

const size\_t& length);

const string get\_string();

int get\_digit(const size\_t& digit);

int get\_digit(const size\_t& digit) const;

SuperInt& set\_digit(const size\_t& digit, const int& number);

friend SuperInt operator+(const SuperInt& left, const SuperInt& right);

friend SuperInt operator-(const SuperInt& left, const SuperInt& right);

friend SuperInt operator\*(const SuperInt& left, const SuperInt& right);

friend SuperInt operator/(const SuperInt& left, const int& right);

friend SuperInt operator/(const SuperInt& left, const SuperInt& right);

friend bool operator <(const SuperInt& left, const SuperInt& right);

friend bool operator >(const SuperInt& left, const SuperInt& right);

friend bool operator ==(const SuperInt& left, const SuperInt& right);

friend bool operator !=(const SuperInt& left, const SuperInt& right);

friend bool operator <=(const SuperInt& left, const SuperInt& right);

friend bool operator >=(const SuperInt& left, const SuperInt& right);

friend ostream& operator<<(ostream& output,const SuperInt& right);

friend istream& operator>>(istream& input, SuperInt& right);

private:

char compare(const SuperInt& right) const;

char compare\_module(const SuperInt& right) const;

bool parse\_string(const string& str);

bool validate(const string& str);

SuperInt operator\_sum\_sub(char s, const SuperInt& right) const;

void summation(const vector<int>& b);

void subtraction(const vector<int>& b);

void clear\_leading\_zeros();

};

class position {

private:

SuperInt& number;

size\_t digit;

public:

position(SuperInt& number, const size\_t& digit)

: number(number), digit(digit) {}

SuperInt& operator=(const SuperInt& right);

SuperInt& operator=(const position& right);

SuperInt& operator=(const const\_position& right);

SuperInt& operator=(const int& right)

{return number.set\_digit(digit, right);}

int operator\*() {return number.get\_digit(digit);}

int operator\*() const {return number.get\_digit(digit);}

friend const\_position;

friend ostream& operator<<(ostream& output,

const position& right);

friend istream& operator>>(istream& input, position& right);

};

class const\_position {

private:

const SuperInt& number;

size\_t digit;

public:

const\_position(const SuperInt& number, const size\_t& digit)

: number(number), digit(digit) {}

const\_position(const position& right)

: number(right.number), digit(right.digit) {}

int operator\*() {return number.get\_digit(digit);}

int operator\*() const {return number.get\_digit(digit);}

friend ostream& operator<<(ostream& output,

const const\_position& right);

};

class interval {

private:

SuperInt& number;

size\_t begin, length;

public:

interval(SuperInt& number, const size\_t& begin, const size\_t& length)

: number(number), begin(begin), length(length) {};

SuperInt& operator=(const SuperInt& right);

SuperInt operator\*();

SuperInt operator\*() const;

friend ostream &operator<<(ostream& output,

const interval& right);

friend istream &operator>>(istream& input, interval& right);

};

}

#endif

### [superint.cpp](https://github.com/gaurapanasenko/unilab/blob/master/03/ProgrammingII_Lab02/libsuperint/superint.cpp)

#include "superint.h"

#include <iomanip>

#include <sstream>

#include <math.h>

using namespace std;

namespace SuperInt {

SuperInt::SuperInt() {

a.resize(1,0); sign = 1;

}

SuperInt::SuperInt(const SuperInt& right) { \*this = right; }

SuperInt::SuperInt(const std::string& right) { \*this = right; }

SuperInt::SuperInt(const char \* right) { \*this = right; }

SuperInt::SuperInt(const int& right) { \*this = right; }

SuperInt& SuperInt::operator=(const SuperInt& right) {

a = right.a; sign = right.sign;

return \*this;

}

SuperInt& SuperInt::operator=(const std::string& right) {

parse\_string(right);

return \*this;

}

SuperInt& SuperInt::operator=(const char \* right) {

parse\_string(string(right));

return \*this;

}

SuperInt& SuperInt::operator=(const int& right) {

sign = (right < 0) ? -1 : 1;

a.clear();

a.push\_back(right);

return \*this;

}

position SuperInt::operator[](const size\_t& digit) {

return position(\*this, digit);

}

const\_position SuperInt::operator[](const size\_t& digit) const {

return const\_position(\*this, digit);

}

SuperInt& SuperInt::operator()(const SuperInt& right) {

\*this = right;

return \*this;

}

interval SuperInt::operator()(const size\_t& begin,

const size\_t& length) {

//limit = 1; this->begin = begin; this->length = length;

return interval(\*this, begin, length);

}

SuperInt& SuperInt::operator()(const SuperInt& right, const size\_t& begin,

const size\_t& length) {

(\*this)(begin,length) = right;

return \*this;

}

const string SuperInt::get\_string() {

stringstream ss;

ss << \*this;

return ss.str();

}

int SuperInt::get\_digit(const size\_t& digit) {

return static\_cast<const SuperInt&>(\*this).get\_digit(digit);

}

int SuperInt::get\_digit(const size\_t& digit) const {

size\_t d = digit, b = d / 9, c = d % 9;

if (b >= a.size()) return 0;

stringstream ss;

ss << setfill('0') << setw(9) << a[b];

return ss.str()[9 - c] - '0';

}

SuperInt& SuperInt::set\_digit(const size\_t& digit, const int& number) {

if (number < 0 || number > 9) return \*this;

size\_t d = digit, b = d / 9, c = d % 9;

if (b >= a.size())

a.resize(b + 1);

stringstream ss;

ss << setfill('0') << setw(9) << a[b];

string str = ss.str();

str[9 - c] = number + '0';

istringstream iss(str);

iss >> a[b];

clear\_leading\_zeros();

return \*this;

}

SuperInt operator+(const SuperInt& left, const SuperInt& right) {

return left.operator\_sum\_sub(1, right);

}

SuperInt operator-(const SuperInt& left, const SuperInt& right) {

return left.operator\_sum\_sub(-1, right);

}

SuperInt operator\*(const SuperInt& left, const SuperInt& right) {

SuperInt num;

num.sign = left.sign \* right.sign;

const std::vector<int> \* a = &left.a, \* b = &right.a;

std::vector<int> \* c = &num.a;

long long base = SUPERINT\_CELL\_MAX;

c->resize(a->size() + b->size(), 0);

long long cur;

for (size\_t i = 0; i < a->size(); i++)

for (int j = 0, carry = 0; j < (int)b->size() || carry; j++) {

cur = c->at(i+j) + a->at(i) \* 1ll

\* (j < (int)b->size() ? b->at(j) : 0) + carry;

c->at(i+j) = int (cur % base);

carry = int (cur / base);

}

num.clear\_leading\_zeros();

return num;

}

SuperInt operator/(const SuperInt& left, const int& right) {

SuperInt num = left;

int carry = 0;

long long base = SUPERINT\_CELL\_MAX;

long long cur;

for (int i = int(num.a.size()) - 1; i >= 0; i--) {

cur = num.a[i] + carry \* 1ll \* base;

num.a[i] = int (cur / right);

carry = int (cur % right);

}

num.clear\_leading\_zeros();

return num;

}

SuperInt operator/(const SuperInt& left, const SuperInt& right) {

SuperInt num, l, r, m;

num.sign = left.sign \* right.sign;

if (left.a.size() < right.a.size()) return 0;

r.a.clear();

r.a.resize(left.a.size() - right.a.size() + 2, 1);

while (r != l + 1) {

m = (r + l) / 2;

if (left.compare\_module(m \* right) == 1)

l = m;

else if (left.compare\_module(m \* right) == -1)

r = m;

else {

r = m; l = m; break;

}

}

num.a = l.a;

num.clear\_leading\_zeros();

return num;

}

bool operator <(const SuperInt& left, const SuperInt& right) {

if (left.compare(right) == -1) return true;

else return false;

}

bool operator >(const SuperInt& left, const SuperInt& right) {

if (left.compare(right) == 1) return true;

else return false;

}

bool operator ==(const SuperInt& left, const SuperInt& right) {

if (left.compare(right) == 0) return true;

else return false;

}

bool operator !=(const SuperInt& left, const SuperInt& right) {

if (left.compare(right) != 0) return true;

else return false;

}

bool operator <=(const SuperInt& left, const SuperInt& right) {

char x = left.compare(right);

if (x == -1 || x == 0) return true;

else return false;

}

bool operator >=(const SuperInt& left, const SuperInt& right) {

char x = left.compare(right);

if (x == 1 || x == 0) return true;

else return false;

}

std::ostream &operator<<(std::ostream &output, const SuperInt &right) {

if (right.sign == -1) output << '-';

output << (right.a.empty() ? 0 : right.a.back());

for (int i = right.a.size() - 2; i >= 0; i--)

output << std::setfill('0') << std::setw(9) << right.a[i];

return output;

}

std::istream &operator>>(std::istream &input, SuperInt &right) {

std::string str;

input >> str;

right.parse\_string(str);

return input;

}

char SuperInt::compare(const SuperInt& right) const {

if (sign == 1 && right.sign == -1) return 1;

else if (sign == -1 && right.sign == 1) return -1;

return compare\_module(right) \* sign;

}

char SuperInt::compare\_module(const SuperInt& right) const {

if (a.size() > right.a.size()) return 1;

else if (a.size() < right.a.size()) return -1;

for (int i = (int)a.size() - 1; i >= 0; i--) {

if (a[i] > right.a[i]) return 1;

else if (a[i] < right.a[i]) return -1;

}

return 0;

}

bool SuperInt::parse\_string(const std::string& str) {

if (!validate(str)) return false;

a.clear();

if (str[0] == '-') sign = -1; else sign = 1;

int x;

if (sign == -1) x = 1; else x = 0;

for (int i = (int)str.length(); i > x; i -= 9) {

if (i - x < 9) a.push\_back(atoi(str.substr(x, i - x).c\_str()));

else a.push\_back(atoi(str.substr(i - 9, 9).c\_str()));

}

clear\_leading\_zeros();

return true;

}

bool SuperInt::validate(const std::string& str) {

for (size\_t i = 0; i < str.size(); i++)

if ((i != 0 || str[i] != '-') && (str[i] < '0' || str[i] > '9'))

return false;

return true;

}

SuperInt SuperInt::operator\_sum\_sub(char s, const SuperInt& right) const {

SuperInt num;

if (sign \* right.sign \* s == 1)

num(\*this).summation(right.a);

else if (compare\_module(right) == -1)

num(right).subtraction(a);

else num(\*this).subtraction(right.a);

return num;

}

void SuperInt::summation(const std::vector<int>& b) {

int carry = 0;

for (size\_t i = 0; i < std::max(a.size(), b.size()) || carry; i++) {

if (i == a.size())

a.push\_back(0);

a[i] += carry + (i < b.size() ? b[i] : 0);

carry = a[i] >= SUPERINT\_CELL\_MAX;

if (carry) a[i] -= SUPERINT\_CELL\_MAX;

}

}

void SuperInt::subtraction(const std::vector<int>& b) {

int carry = 0;

for (size\_t i = 0; i < b.size() || carry; i++) {

a[i] -= carry + (i < b.size() ? b[i] : 0);

carry = a[i] < 0;

if (carry) a[i] += SUPERINT\_CELL\_MAX;

}

clear\_leading\_zeros();

}

void SuperInt::clear\_leading\_zeros() {

while (a.size() > 1 && a.back() == 0) a.pop\_back();

}

SuperInt& position::operator=(const SuperInt& right)

{return (\*this) = right[digit];}

SuperInt& position::operator=(const position& right)

{return (\*this) = \*right;}

SuperInt& position::operator=(const const\_position& right)

{return (\*this) = \*right;}

std::ostream& operator<<(std::ostream& output, const position& right)

{return output << right.number.get\_digit(right.digit);}

std::istream &operator>>(std::istream& input, position& right) {

int n;

input >> n;

right = n;

return input;

}

std::ostream& operator<<(std::ostream& output, const const\_position& right)

{return output << right.number.get\_digit(right.digit);}

SuperInt& interval::operator=(const SuperInt& right) {

size\_t a = begin, b = begin + length;

for (size\_t i = a; i <= b; i++) number[i] = right;

return number;

}

SuperInt interval::operator\*() {

return \*(static\_cast<const interval&>(\*this));

}

SuperInt interval::operator\*() const {

SuperInt out;

size\_t a = begin, b = begin + length;

for (size\_t i = a; i <= b; i++) out[i - a] = number[i];

return out;

}

std::ostream &operator<<(std::ostream& output, const interval& right) {

return output << \*right;

}

std::istream &operator>>(std::istream& input, interval& right) {

SuperInt in;

input >> in;

size\_t a = right.begin, b = right.begin + right.length;

for (size\_t i = a; i <= b; i++) right.number[i] = in[i - a];

return input;

}

}

4. Опис інтерфейсу програми

Програма має одну бібліотеку libsuperint.so та два виконуваних файлів run\_tests і main. За допомогою run\_tests можна перевірити чи правильно виконуються арифметичні операції, при запуску цього файлу автоматично запускаються стандартні тести, які друкуються на екрані ти автоматично перевіряються у кінці запуску друкується резюме. За допомогою main можна тестувати бібліотеку своїми тестами за допомогою меню.

5. Опис тестових прикладів

### Програма run\_tests

Тест 1: Инициализация.

Вводим "1234" в SuperInt::SuperInt и сравниваем вывод объекта со стокой "1234"

Получили: 1234

Статус: OK

Тест 2: Инициализация с неверной строкой.

Вводим "1231231312312313-2312312313123" в SuperInt::SuperInt и сравниваем

вывод объекта со стокой "0"

Получили: 0

Статус: OK

Тест 3: Присваивание.

Вводим "1234" в SuperInt::SuperInt через присваивание и

сравниваем вывод объекта со стокой "1234"

Получили: 1234

Статус: OK

Тест 4: Ввод больших чисел.

Вводим "1234567891357924680" в SuperInt::SuperInt и

сравниваем вывод объекта со стокой "1234567891357924680"

Получили: 1234567891357924680

Статус: OK

Тест 5: Ввод отриательных больших чисел.

Вводим "-1234567891357924680" в SuperInt::SuperInt и

сравниваем вывод объекта со стокой "-1234567891357924680"

Получили: -1234567891357924680

Статус: OK

Тест 6: Конструктор копирования.

Вводим "-1234567891357924680" в SuperInt::SuperInt и копируем в другой

SuperInt::SuperInt сравниваем вывод объекта со стокой "-1234567891357924680"

Получили: -1234567891357924680

Статус: OK

Тест 7: Сравнение одинаковых чисел.

547509411375863014 == 547509411375863014

Статус: OK

Тест 8: Сравнение одинаковых по модулю чисел.

547509411375863014 > -547509411375863014

Статус: OK

Тест 9: Сравнение разных чисел.

547509411375863014 > 47509411375863014

Статус: OK

Тест 10: Сравнение разных чисел.

547509411375863014 > 54750941137586301

Статус: OK

Тест 11: Сравнение разных чисел.

-547509411375863014 < 54750941137586301

Статус: OK

Тест 12: Сравнение разных чисел.

547509411375863014 > 24440630276

Статус: OK

Тест 13: Сравнение разных чисел.

547509411375863014 > 1

Статус: OK

Тест 14: Сложение.

12345 + 23456 = 35801

Получили: 35801

Статус: OK

Тест 15: Сложение огромных чисел.

547509411375863014 + 24440630276 = 547509435816493290

Получили: 547509435816493290

Статус: OK

Тест 16: Сложение отрицательного числа и положительного.

-547509411375863014 + 24440630276 = -547509386935232738

Получили: -547509386935232738

Статус: OK

Тест 17: Сложение отрицательного числа и положительного.

547509411375863014 + -24440630276 = 547509386935232738

Получили: 547509386935232738

Статус: OK

Тест 18: Сложение отрицательных чисел.

-12345 + -23456 = -35801

Получили: -35801

Статус: OK

Тест 19: вычитание огромных чисел.

547509411375863014 - 24440630276 = 547509386935232738

Получили: 547509386935232738

Статус: OK

Тест 20: вычитание отрицательного числа и положительного.

-547509411375863014 - 24440630276 = -547509435816493290

Получили: -547509435816493290

Статус: OK

Тест 21: вычитание отрицательного числа и положительного.

547509411375863014 - -24440630276 = 547509435816493290

Получили: 547509435816493290

Статус: OK

Тест 22: вычитание отрицательных чисел.

-12345 - -23456 = -11111

Получили: -11111

Статус: OK

Тест 23: Умножение огромных чисел.

547509411375863014 \* 24440630276 = 13381475096067856395597011864

Получили: 13381475096067856395597011864

Статус: OK

Тест 24: Умножение отрицательного числа и положительного.

-547509411375863014 \* 24440630276 = -13381475096067856395597011864

Получили: -13381475096067856395597011864

Статус: OK

Тест 25: Умножение отрицательного числа и положительного.

547509411375863014 \* -24440630276 = -13381475096067856395597011864

Получили: -13381475096067856395597011864

Статус: OK

Тест 26: Умножение отрицательных чисел.

-12345 \* -23456 = 289564320

Получили: 289564320

Статус: OK

Тест 27: Деление чисел.

1000 / 10 = 100

Получили: 100

Статус: OK

Тест 28: Деление огромных чисел.

594881688386039612131121336848 / -48872148152877800551 = -12172202591

Получили: -12172202591

Статус: OK

Тест 29: Деление огромных чисел.

26339464348311100259 / 21430594771958 = 1229058

Получили: 1229058

Статус: OK

Всего пройденных тестов: 29

Количество тестов с ошибками: 0

Количество успешных тестов: 29

### Програма main

1. Запустить тесты

2. Добавить число в конец массива

3. Удалить число из массива по индексу

4. Сгенегировать 10 случайных чисел

5. Вывести массив на экран

6. Очистить массив

7. Сложить два числа из массива

8. Отнять два числа из массива

9. Умножить два числа из массива

10. Поделить два числа из массива

11. Поделить число из массива на маленькое число

12. Переместить часть числа в другое число на той же позиции

13. Забрать разряд числа

14. Изменить цифру на разряде числа

0. Выйти

>>> 4

array(10) {

[0] => 6473508637718769

[1] => -363400991118372029036605

[2] => -239795

[3] => -974885203778827055850654

[4] => -312933884

[5] => -8715695

[6] => 856626483676917446403082

[7] => -84192940897292860778236641

[8] => 31035440310109

[9] => 345444289

}

Нажмите ENTER для продолжения...

1. Запустить тесты

2. Добавить число в конец массива

3. Удалить число из массива по индексу

4. Сгенегировать 10 случайных чисел

5. Вывести массив на экран

6. Очистить массив

7. Сложить два числа из массива

8. Отнять два числа из массива

9. Умножить два числа из массива

10. Поделить два числа из массива

11. Поделить число из массива на маленькое число

12. Переместить часть числа в другое число на той же позиции

13. Забрать разряд числа

14. Изменить цифру на разряде числа

0. Выйти

>>> 2

a

Нажмите ENTER для продолжения...

1. Запустить тесты

2. Добавить число в конец массива

3. Удалить число из массива по индексу

4. Сгенегировать 10 случайных чисел

5. Вывести массив на экран

6. Очистить массив

7. Сложить два числа из массива

8. Отнять два числа из массива

9. Умножить два числа из массива

10. Поделить два числа из массива

11. Поделить число из массива на маленькое число

12. Переместить часть числа в другое число на той же позиции

13. Забрать разряд числа

14. Изменить цифру на разряде числа

0. Выйти

>>> 2

34234

Нажмите ENTER для продолжения...

1. Запустить тесты

2. Добавить число в конец массива

3. Удалить число из массива по индексу

4. Сгенегировать 10 случайных чисел

5. Вывести массив на экран

6. Очистить массив

7. Сложить два числа из массива

8. Отнять два числа из массива

9. Умножить два числа из массива

10. Поделить два числа из массива

11. Поделить число из массива на маленькое число

12. Переместить часть числа в другое число на той же позиции

13. Забрать разряд числа

14. Изменить цифру на разряде числа

0. Выйти

>>> 3

array(12) {

[0] => 6473508637718769

[1] => -363400991118372029036605

[2] => -239795

[3] => -974885203778827055850654

[4] => -312933884

[5] => -8715695

[6] => 856626483676917446403082

[7] => -84192940897292860778236641

[8] => 31035440310109

[9] => 345444289

[10] => 0

[11] => 34234

}

8

Нажмите ENTER для продолжения...

1. Запустить тесты

2. Добавить число в конец массива

3. Удалить число из массива по индексу

4. Сгенегировать 10 случайных чисел

5. Вывести массив на экран

6. Очистить массив

7. Сложить два числа из массива

8. Отнять два числа из массива

9. Умножить два числа из массива

10. Поделить два числа из массива

11. Поделить число из массива на маленькое число

12. Переместить часть числа в другое число на той же позиции

13. Забрать разряд числа

14. Изменить цифру на разряде числа

0. Выйти

>>> 7

array(11) {

[0] => 6473508637718769

[1] => -363400991118372029036605

[2] => -239795

[3] => -974885203778827055850654

[4] => -312933884

[5] => -8715695

[6] => 856626483676917446403082

[7] => -84192940897292860778236641

[8] => 345444289

[9] => 0

[10] => 34234

}

Введите индексы:

>>> 0 1

-363400984644863391317836

Нажмите ENTER для продолжения...

1. Запустить тесты

2. Добавить число в конец массива

3. Удалить число из массива по индексу

4. Сгенегировать 10 случайных чисел

5. Вывести массив на экран

6. Очистить массив

7. Сложить два числа из массива

8. Отнять два числа из массива

9. Умножить два числа из массива

10. Поделить два числа из массива

11. Поделить число из массива на маленькое число

12. Переместить часть числа в другое число на той же позиции

13. Забрать разряд числа

14. Изменить цифру на разряде числа

0. Выйти

>>> 8

array(11) {

[0] => 6473508637718769

[1] => -363400991118372029036605

[2] => -239795

[3] => -974885203778827055850654

[4] => -312933884

[5] => -8715695

[6] => 856626483676917446403082

[7] => -84192940897292860778236641

[8] => 345444289

[9] => 0

[10] => 34234

}

Введите индексы:

>>> 6

8

856626483676917100958793

Нажмите ENTER для продолжения...

1. Запустить тесты

2. Добавить число в конец массива

3. Удалить число из массива по индексу

4. Сгенегировать 10 случайных чисел

5. Вывести массив на экран

6. Очистить массив

7. Сложить два числа из массива

8. Отнять два числа из массива

9. Умножить два числа из массива

10. Поделить два числа из массива

11. Поделить число из массива на маленькое число

12. Переместить часть числа в другое число на той же позиции

13. Забрать разряд числа

14. Изменить цифру на разряде числа

0. Выйти

>>> 10

array(11) {

[0] => 6473508637718769

[1] => -363400991118372029036605

[2] => -239795

[3] => -974885203778827055850654

[4] => -312933884

[5] => -8715695

[6] => 856626483676917446403082

[7] => -84192940897292860778236641

[8] => 345444289

[9] => 0

[10] => 34234

}

Введите индексы:

>>> 10 1

0

Нажмите ENTER для продолжения...

1. Запустить тесты

2. Добавить число в конец массива

3. Удалить число из массива по индексу

4. Сгенегировать 10 случайных чисел

5. Вывести массив на экран

6. Очистить массив

7. Сложить два числа из массива

8. Отнять два числа из массива

9. Умножить два числа из массива

10. Поделить два числа из массива

11. Поделить число из массива на маленькое число

12. Переместить часть числа в другое число на той же позиции

13. Забрать разряд числа

14. Изменить цифру на разряде числа

0. Выйти

>>> 9

array(11) {

[0] => 6473508637718769

[1] => -363400991118372029036605

[2] => -239795

[3] => -974885203778827055850654

[4] => -312933884

[5] => -8715695

[6] => 856626483676917446403082

[7] => -84192940897292860778236641

[8] => 345444289

[9] => 0

[10] => 34234

}

Введите индексы:

>>> 5 7

733799994013830900220573200780495

Нажмите ENTER для продолжения...

1. Запустить тесты

2. Добавить число в конец массива

3. Удалить число из массива по индексу

4. Сгенегировать 10 случайных чисел

5. Вывести массив на экран

6. Очистить массив

7. Сложить два числа из массива

8. Отнять два числа из массива

9. Умножить два числа из массива

10. Поделить два числа из массива

11. Поделить число из массива на маленькое число

12. Переместить часть числа в другое число на той же позиции

13. Забрать разряд числа

14. Изменить цифру на разряде числа

0. Выйти

>>> 11

array(11) {

[0] => 6473508637718769

[1] => -363400991118372029036605

[2] => -239795

[3] => -974885203778827055850654

[4] => -312933884

[5] => -8715695

[6] => 856626483676917446403082

[7] => -84192940897292860778236641

[8] => 345444289

[9] => 0

[10] => 34234

}

Введите индекс:

>>> 5

Введите число:

>>> 876

-9949

Нажмите ENTER для продолжения...

1. Запустить тесты

2. Добавить число в конец массива

3. Удалить число из массива по индексу

4. Сгенегировать 10 случайных чисел

5. Вывести массив на экран

6. Очистить массив

7. Сложить два числа из массива

8. Отнять два числа из массива

9. Умножить два числа из массива

10. Поделить два числа из массива

11. Поделить число из массива на маленькое число

12. Переместить часть числа в другое число на той же позиции

13. Забрать разряд числа

14. Изменить цифру на разряде числа

0. Выйти

>>> 12

array(11) {

[0] => 6473508637718769

[1] => -363400991118372029036605

[2] => -239795

[3] => -974885203778827055850654

[4] => -312933884

[5] => -8715695

[6] => 856626483676917446403082

[7] => -84192940897292860778236641

[8] => 345444289

[9] => 0

[10] => 34234

}

Введите индексы:

>>> 1 3

Введите начальный разряд и количество цифр:

>>> 5 10

-363400991778827055856605

Нажмите ENTER для продолжения...

1. Запустить тесты

2. Добавить число в конец массива

3. Удалить число из массива по индексу

4. Сгенегировать 10 случайных чисел

5. Вывести массив на экран

6. Очистить массив

7. Сложить два числа из массива

8. Отнять два числа из массива

9. Умножить два числа из массива

10. Поделить два числа из массива

11. Поделить число из массива на маленькое число

12. Переместить часть числа в другое число на той же позиции

13. Забрать разряд числа

14. Изменить цифру на разряде числа

0. Выйти

>>> 13

Введите индекс:

>>> 5

Введите разряд:

>>> 8

0

Нажмите ENTER для продолжения...

1. Запустить тесты

2. Добавить число в конец массива

3. Удалить число из массива по индексу

4. Сгенегировать 10 случайных чисел

5. Вывести массив на экран

6. Очистить массив

7. Сложить два числа из массива

8. Отнять два числа из массива

9. Умножить два числа из массива

10. Поделить два числа из массива

11. Поделить число из массива на маленькое число

12. Переместить часть числа в другое число на той же позиции

13. Забрать разряд числа

14. Изменить цифру на разряде числа

0. Выйти

>>> 14

Введите индекс:

>>> 6

Введите разряд:

>>> 7

Введите цифру:

>>> 8

856626483676917448403082

Нажмите ENTER для продолжения...

1. Запустить тесты

2. Добавить число в конец массива

3. Удалить число из массива по индексу

4. Сгенегировать 10 случайных чисел

5. Вывести массив на экран

6. Очистить массив

7. Сложить два числа из массива

8. Отнять два числа из массива

9. Умножить два числа из массива

10. Поделить два числа из массива

11. Поделить число из массива на маленькое число

12. Переместить часть числа в другое число на той же позиции

13. Забрать разряд числа

14. Изменить цифру на разряде числа

0. Выйти

>>> 6

Нажмите ENTER для продолжения...

1. Запустить тесты

2. Добавить число в конец массива

3. Удалить число из массива по индексу

4. Сгенегировать 10 случайных чисел

5. Вывести массив на экран

6. Очистить массив

7. Сложить два числа из массива

8. Отнять два числа из массива

9. Умножить два числа из массива

10. Поделить два числа из массива

11. Поделить число из массива на маленькое число

12. Переместить часть числа в другое число на той же позиции

13. Забрать разряд числа

14. Изменить цифру на разряде числа

0. Выйти

>>> 0