Звіт з лабораторної роботи №2
за дисципліною "Програмування II"
студента групи ПА-17-1
Панасенка Єгора Сергійовича
Кафедра комп'ютерних технологій
ФПМ, ДНУ, 2017-2018 навч.р.

Постановка задачі

Розробити об’єктно-орієнтовану бібліотеку для роботи зі структурами даних за однією з нижченаведених тем у відповідності з нижченаведеними 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.

Тема: Довга арифметика: реалізація основних арифметичних операцій.

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 зберігається 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.

Клас SuperInt

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()`
    i. Конструктор у якому створюється пусте число, тобто нуль.
    ii. Не має аргументів.
    iii. Нічого не виводить.
2. `SuperInt(const SuperInt& x)`
    i. Конструктор копіювання.
    ii. Має один аргумент-посилання на інше число x, яке потрібно скопіювати.
    iii. Нічого не виводить.
3. `SuperInt(const string& str)`
    i. Конструктор неявного перетворення із текстової строки.

ii. Має один аргумент-посилання на строку str, у якій зберігається число.
iii. Нічого не виводить.

4. `SuperInt(const char * str)`
   i. Конструктор неявного перетворення із текстової строки.
   ii. Має один аргумент-вказівник на строку str, у якій зберігається число.
   iii. Нічого не виводить.
5. `SuperInt(const int& right)`
   i. Конструктор неявного перетворення із числа.
   ii. Має один аргумент-посилання на число right.
   iii. Нічого не виводить.
6. `SuperInt& operator=(const SuperInt& right)`
   i. Оператор присвоювання іншого числа.
   ii. Має один аргумент-посилання на інше число right, яке потрібно скопіювати.
   iii. Виводить посилання на себе.
7. `SuperInt& operator=(const string& right)`
   i. Оператор присвоювання іншого числа перетворенням із строки.
   ii. Має один аргумент-посилання на строку str, у якій зберігається число.
   iii. Виводить посилання на себе.
8. `SuperInt& operator=(const char * str)`
   i. Оператор присвоювання іншого числа перетворенням із строки.
   ii. Має один аргумент-посилання на строку str, у якій зберігається число.
   iii. Виводить посилання на себе.
9. `SuperInt& operator=(const int& right)`
   i. Оператор присвоювання іншого числа.
   ii. Має один аргумент-посилання на число right.
   iii. Виводить посилання на себе.
10. `position operator[](const size_t& digit)`
    i. Оператор індексації, виводить розряд числа у вигляді класу position.
    ii. Має один аргумент-посилання на номер розряду right.
    iii. Виводить клас position.
11. `const_position operator[](const size_t& digit) const`
    i. Оператор індексації константного класу, виводить розряд числа у вигляді класу const_position.
    ii. Має один аргумент-посилання на номер розряду right.
    iii. Виводить клас const_position.
12. `SuperInt& operator()(const SuperInt& right)`
    i. Оператор псевдозмінної для копіювання числа right.
    ii. Має один аргумент-посилання на інше число right, яке потрібно скопіювати.
    iii. Виводить посилання на себе.
13. `interval operator()(const size_t& begin, const size_t& length)`
    i. Оператор псевдозмінної для визначення інтервалу подальшої обробки числа.
    ii. Аргументи:
        a. begin - номер розряду з якого починати інтервал.
        b. length - кількість розрядів які входять у інтервал після begin.

iii. Виводить клас interval.

14. `SuperInt& operator()(const SuperInt& right, const size_t& begin, const size_t& length)`
    i. Оператор псевдозмінної для копіювання числа на інтервалі.
    ii. Аргументи:
        a. right - число яке потрібно скопіювати
        b. begin - номер розряду з якого починати інтервал.
        c. length - кількість розрядів які входять у інтервал після begin.
    iii. Виводить посилання на себе.
15. `const string get_string()`
    i. Функція яка перетворює число у строку.
    ii. Не має аргументів.
    iii. Нічого не виводить.
16. `int get_digit(const size_t& digit)`
    i. Функція яка знаходить розряд числа.
    ii. Має один аргумент-посилання на номер розряду digit.
    iii. Виводить цифру у розряді digit.
17. `int get_digit(const size_t& digit) const`
    i. Константна функція яка знаходить розряд числа.
    ii. Має один аргумент-посилання на номер розряду digit.
    iii. Виводить цифру у розряді digit.
18. `SuperInt& set_digit(const size_t& digit, const int& number)`
    i. Функція яка знаходить розряд числа та змінює цифру у цьому розряді.
    ii. Має один аргумент-посилання на номер розряду digit.
    iii. Виводить посилання на себе.

Дружелюбні функції

1. `SuperInt operator+(const SuperInt& left, const SuperInt& right);`
    i. Оператор додавання двох чисел.
    ii. Аргументи:
        a. left - посилання на перше число.
        b. right - посилання на друге число.
    iii. Виводить нове число.
2. `SuperInt operator-(const SuperInt& left, const SuperInt& right);`
    i. Оператор віднімання двох числел.
    ii. Аргументи:
        a. left - посилання на перше число.
        b. right - посилання на друге число.
    iii. Виводить нове число.
3. `SuperInt operator*(const SuperInt& left, const SuperInt& right);`
    i. Оператор множення двох числел.
    ii. Аргументи:
        a. left - посилання на перше число.
        b. right - посилання на друге число.
    iii. Виводить нове число.

4. `SuperInt operator/(const SuperInt& left, const int& right);`
    i. Оператор ділення великого числа на маленьке.
    ii. Аргументи:
        a. left - посилання на велике число.
        b. right - посилання на маленьке число.
    iii. Виводить нове число.
5. `SuperInt operator/(const SuperInt& left, const SuperInt& right);`
    i. Оператор ділення двох числел.
    ii. Аргументи:
        a. left - посилання на перше число.
        b. right - посилання на друге число.
    iii. 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);`
    i. Оператори порівняння двох чисел
    ii. Аргументи:
        a. left - посилання на перше число.
        b. right - посилання на друге число.
    iii. Виводить відповідь на питання чи виконується ця нерівність.
7. `ostream& operator<<(ostream& output,const SuperInt& right);`
   `istream& operator>>(istream& input, SuperInt& right);`
    i. Оператори вводу та виводу з потоку.
    ii. Аргументи:
        a. input/output - посилання на потоки вводу/виводу.
        b. right - посилання на число, яке потрібно вивести.
    iii. Виводить посилання на поток з першого аргументу.

Приватні функції

1. `char compare(const SuperInt& right) const;`
    i. Константна функція порівняння поточного числа з числом right.
    ii. Має єдиний аргумент-посилання на число.
    iii. Виводить -1 - якщо поточне число менше за right, 0 - якщо дорівнює, 1 - якщо більше.
2. `char compare_module(const SuperInt& right) const;`
    i. Константна функція порівняння поточного числа по модулю з числом right по модулю.
    ii. Має єдиний аргумент-посилання на число.
    iii. Виводить так само як і compare;
3. `bool parse_string(const string& str);`
    i. Функція конвертує строку у число.
    ii. Має єдиний аргумент-посилання на строку.
    iii. Виводить успішність виконання.
4. `bool validate(const string& str);`

    i. Перевіряє чи можна зчитати число.
    ii. Має єдиний аргумент-посилання на строку.
    iii. Виводить валідність строки.
5. `SuperInt operator_sum_sub(char s, const SuperInt& right) const;`
    i. Функція визначає яку операцію потрібно зробити та у якій послідовності виконувати, так як алгоритм не може від меншого відняти більше.
    ii. Аргументи:
        a. s - поточна операція -1 для віднімання, 1 для додавання.
        b. right - посилання на друге число.
    iii. Виводить результат у вигляді нового числа
6. `void summation(const vector<int>& b);`
    i. Функція сумує числа по модулю, ігноруючи та не змінюючи знаки.
    ii. Має єдиний аргумент-посилання на масив з великим числом.
    iii. Нічого не виводить.
7. `void subtraction(const vector<int>& b);`
    i. Функція віднімає друг число по модулю від поточного по модулю, ігноруючи та не змінюючи знаки.
    ii. Має єдиний аргумент-посилання на масив з великим числом, але меншим за поточне.
    iii. Нічого не виводить.
8. `void clear_leading_zeros();`
    i. Функція видаляє непотрібні нулі перед числом.
    ii. Не має аргументів.
    iii. 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)`
    i. Конструктор з параметрами, потрібними для обробки числа.
    ii. Аргументи:
        a. number - посилання на число.
        b. digit - номер розряду.
    iii. Нічого не виводить.
2. `SuperInt& operator=(const SuperInt& right)`
    i. Оператор присвоювання, який копіює цифру на розряді digit з числа right на тому ж розряді.
    ii. Має єдиний аргумент-посилання на число.
    iii. Виводить посилання на поле number.
3. `SuperInt& operator=(const position& right)`
   `SuperInt& operator=(const const_position& right)`
    i. Оператор присвоювання, який замінює цифру на розряді digit на цифру на яку вказує right.

   ii. Має єдиний аргумент-посилання на клас position.
   iii. Виводить посилання на поле number.
4. `SuperInt& operator=(const int& right)`
   i. Оператор присвоювання, який замінює цифру на розряді digit на цифру right.
   ii. Має єдиний аргумент-посилання на число.
   iii. Виводить посилання на поле number.
5. `int operator*()` `int operator*() const`
   i. Оператори доступу до даних.
   ii. Не мають аргументів.
   iii. Виводять цифру з числа number на розряді digit.

Дружелюбні функції

1. `ostream& operator<<(ostream& output, const position& right)`
   `istream& operator>>(istream& input, position& right)`
   i. Оператори вводу та виводу з потоку.
   ii. Аргументи:
      a. input/output - посилання на потоки вводу/виводу.
      b. right - посилання на клас, який потрібно вивести.
   iii. 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)`
   i. Конструктор з параметрами, потрібними для обробки числа.
   ii. Аргументи:
      a. number - посилання на число.
      b. digit - номер розряду.
   iii. Нічого не виводить.
2. `const_position(const position& right)`
   i. Конструктор неявного перетворення з position;
   ii. Має єдиний аргумент-посилання на клас position.
   iii. Нічого не виводить.
3. `int operator*()`
   `int operator*() const`
   i. Оператори доступу до даних.
   ii. Не мають аргументів.
   iii. Виводять цифру з числа number на розряді digit.

Дружелюбні функції

1. `ostream& operator<<(ostream& output, const const_position& right)`
   i. Оператор виводу з потоку.

ii. Аргументи:
    a. output - посилання на потоки виводу.
    b. right - посилання на клас, який потрібно вивести.
iii. 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)`
    i. Конструктор з параметрами, потрібними для обробки числа.
    ii. Аргументи:
        a. number - посилання на число.
        b. begin - номер розряду з якого починається інтервал.
        c. length - кількість розрядів в інтервалі.
    iii. Нічого не виводить.
2. `SuperInt& operator=(const SuperInt& right)`
    i. Оператор присвоювання який копіює цифри на заданому інтервалі з числа right.
    ii. Має єдиний аргумент-посилання на число.
    iii. Виводить посилання на поле number.
3. `SuperInt operator*()`
   `SuperInt operator*() const`
    i. Оператори доступу до даних.
    ii. Не мають аргументів.
    iii. 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)`
    i. Оператори вводу та виводу з потоку.
    ii. Аргументи:
        a. input/output - посилання на потоки вводу/виводу.
        b. right - посилання на клас, який потрібно вивести.
    iii. 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

i. Виводиться меню.

ii. Запитується вибір.

iii. Якщо потрібно виводиться масив

iv. Згідно з відповіддю запитуються додаткові необхідні данні та виконується дія.

Вихідний текст програми розв’язку задачі

Файл 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)
```

Файл src/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";
}
```

Файл src/run_tests.cpp

```
#include <iostream>
#include "superint.h"
#include "tests.h"

using namespace std;

int main() {
        return run_tests();
}
```

Файл src/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
```

Файл 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 = "12345678913579246800";
                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 = "263394643483111100259";
                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;
}
```

Файл 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
```

Файл 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;
}

}
```

Опис іінтерфейсу

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.

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 "12345678913579246 80" в SuperInt::SuperInt и
  сравниваем вывод объекта со стокой "12345678913579246 80"
  Получили: 12345678913579246 80
  Статус: OK
Тест 5: Ввод отриательных больших чисел.
  Вводим "-12345678913579246 80" в SuperInt::SuperInt и
  сравниваем вывод объекта со стокой "-12345678913579246 80"
  Получили: -12345678913579246 80
  Статус: OK
Тест 6: Конструктор копирования.
  Вводим "-12345678913579246 80" в SuperInt::SuperInt и копируем в другой
  SuperInt::SuperInt сравниваем вывод объекта со стокой "-12345678913579246 80"
  Получили: -12345678913579246 80
  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] => -3634009911183720290366O5
        [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] => -3634009911183720290366O5
        [2] => -239795
        [3] => -974885203778827055850654
        [4] => -312933884
        [5] => -8715695
        [6] => 8566264836769174464030B2
        [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
85662648367691710095879 3
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
7337999940138309002205732007804 95
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
8566264836769174484030 82
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
```