Московский авиационный институт (национальный исследовательский университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №1 по курсу «Дискретный анализ»

Студент: А. П. Уваров
Преподаватель: А. А. Кухтичев
Группа: М8О-206Б-19

Дата:
Оценка:
Подпись:

Москва, 2021

Лабораторная работа №1

Задача: Необходимо разработать программную библиотеку на языке C или C++, реализующую простейшие арифметические действия и проверку условий над целыми неотрицательными числами. На основании этой библиотеки нужно составить программу, выполняющую вычисления над парами десятичных чисел и выводящую результат на стандартный файл вывода.

1 Описание

Требуется написать длинную арифметику, основные проблемы возникали при операциях умножения, возведения в степень и деления. Причем каждая из них могли использовать другую. Осноная идея реализации программы состояла в том, чтобы сделать свой тип длинной арифметики, который смог бы легко применяться так же, как с типом int или любым другим встроенным типом. То есть поддерживал операторы ввода, вывода и остальные.

2 Исходный код

Переменная BASE - обозначает максимальный элемент с нашим основанием RADIX - кол-во цифр в одном разряде.

Все разряды хранятся в векторе nums, т.к. у нас нет отрицательных чисел, то и тип значения в векторе беззнаковый int.

Также я сделал создание числа по вектору беззнаковых int-ов и строке, в них сразу же нормализуется число, убирая ведущие нули.

```
1  namespace NSupalg{
2    class TSuperLong{
3    private:
4        static const int BASE = 100000;
5        static const int RADIX = 5;
6
7        std::vector<uint32_t> nums;
8    public:
9        TSuperLong() = default;
10       TSuperLong(const std::string& get);
11       TSuperLong(const std::vector<uint32_t>& v);
12
13       TSuperLong operator+(const TSuperLong& rhs) const;
14       TSuperLong& operator+=(const TSuperLong& rhs);
15       TSuperLong operator-(const TSuperLong& rhs) const; // throw
16       TSuperLong& operator-=(const TSuperLong& rhs); // throw
17       TSuperLong operator*(const TSuperLong& rhs) const;
18       TSuperLong& operator*=(const TSuperLong& rhs);
19       TSuperLong operator^(TSuperLong rhs) const; // throw
20       TSuperLong operator^(const uint32_t& degree) const; // throw
21       TSuperLong operator/(const TSuperLong& rhs) const; // throw
22
23       // TODO: code all operators
24       bool operator<(const TSuperLong& rhs) const;
25       bool operator>(const TSuperLong& rhs) const;
26       bool operator>=(const TSuperLong& rhs) const;
27       bool operator==(const TSuperLong& rhs) const;
28
29       TSuperLong& expand(const size_t& count);
30       TSuperLong powten(const size_t& count) const;
31
32       void push_back(const uint32_t& input);
33       bool isEven() const;
34
35       friend std::ostream& operator<<(std::ostream& out, const TSuperLong& rhs);
36       friend std::istream& operator>>(std::istream& in, TSuperLong& rhs);
37   };
38 }
39
```

```
40 ||using suplong = NSupalg::TSuperLong;
```

ll.cpp	
TSuperLong() = default	Конструктор по-умолчанию
TSuperLong(const std::string& get); TSuperLong(const std::vector<uint32_t>& v)	Прочие конструкторы с параметрами.
TSuperLong operator+(const TSuperLong& rhs) const; TSuperLong& operator+=(const TSuperLong& rhs); TSuperLong operator-(const TSuperLong& rhs) const; TSuperLong& operator-=(const TSuperLong& rhs); TSuperLong operator*(const TSuperLong& rhs) const; TSuperLong& operator*=(const TSuperLong& rhs); TSuperLong operator∧(*TSuperLongrhs*)*const*; *TSuperLongoperator* ∧ (*constuint*32_*t*&*degree*)*const*; *TSuperLongoperator* \ (*constTSuperLong*&*rhs*)*const*	Все операторы для работы с переменными длинной арифметики.
bool operator<(const TSuperLong& rhs) const; bool operator>(const TSuperLong& rhs) const; bool operator>=(const TSuperLong& rhs) const; bool operator==(const TSuperLong& rhs) const;	Все операторы сравнения.
void push_back(const uint32_t& input); bool isEven() const;	Методы упрощения, например push_back добавляет в конец числа 0 и увеличивает на параметр, а isEven проверяет на четность.
friend std::ostream& operator«(std::ostream& out, const TSuperLong& rhs); friend std::istream& operator»(std::istream& in, TSuperLong& rhs);	Перегрузка операторов ввода и вывода для моих длинный чисел.
using suplong = NSupalg::TSuperLong;	Для упрощения добавляю using-объявление.

Вот так выглядит мой main.cpp

```cpp
  #include <iostream>
#include "ll.hpp"

int main(){
  std::cin.tie(NULL);
  std::cout.tie(NULL);
  std::ios::sync_with_stdio(false);

  suplong f, s;
  while(std::cin >> f >> s){
    char act; std::cin >> act;
    switch (act)
    {
      case '+':
        std::cout << f + s << '\n';
        break;

      case '-':
        try{
          std::cout << f - s << '\n';
        }
        catch(const std::runtime_error& e){
          std::cout << e.what() << '\n';
        }
        break;

      case '*':
        std::cout << f * s << '\n';
        break;

      case '/':
        try{
          std::cout << f / s << '\n';
        }
        catch(const std::runtime_error& e){
          std::cout << e.what() << '\n';
        }
        break;

      case '^':
        try{
          std::cout << (f ^ s) << '\n';
        }
        catch(const std::runtime_error& e){
          std::cout << e.what() << '\n';
        }
        break;

```

```
49        case '<':
50          std::cout << std::boolalpha << (f < s) << '\n';
51          break;
52
53        case '>':
54          std::cout << std::boolalpha << (f > s) << '\n';
55          break;
56
57        case '=':
58          std::cout << std::boolalpha << (f == s) << '\n';
59          break;
60
61        default:
62          break;
63      }
64    }
65    return 0;
66  }
```

3 Консоль

```
andrey@MoronWithAsusVivoBook:~/DA4$ make
g++ -g -pedantic -Wall -std=c++17 -Werror -Wno-sign-compare -O2 -lm -c ./src/main.cpp
-o ./src/main.o
g++ -g -pedantic -Wall -std=c++17 -Werror -Wno-sign-compare -O2 -lm -c ./src/ll.cpp
-o ./src/ll.o
g++ -g -pedantic -Wall -std=c++17 -Werror -Wno-sign-compare -O2 -lm ./src/main.o
./src/ll.o -o solution
andrey@MoronWithAsusVivoBook:~/DA4$ ./solution
38943432983521435346436
354353254328383
+
904094384738493247293847 3843
2343543
-
972323
2173937
>
2
3
-
3894343333787468967 4819
904094384738493247293613 0300
false
Error
```

4 Тест производительности

```
andrey@MoronWithAsusVivoBook:~/DA6$ ./benchmark <tests/01.t
my_num time = 648s
gmp time = 61s
andrey@MoronWithAsusVivoBook:~/DA6$ ./benchmark <tests/02.t
my_num time = 403s
gmp time = 30s
andrey@MoronWithAsusVivoBook:~/DA6$ ./benchmark <tests/03.t
my_num time = 1001054s
gmp time = 4265s
andrey@MoronWithAsusVivoBook:~/DA6$ ./benchmark <tests/04.t
my_num time = 4799878s
gmp time = 12690s
```

Моя реализация оказалась не очень быстрой, но и использовал я не самые быстрые алгоритмы, к тому же gmp имеет ассемблерные вставки, что также ускоряет работу программы.

5 Выводы

Выполнив шестую лабораторную работу по курсу «Дискретный анализ», я научился работать с длинной арифметикой, реализовал свой калькулятор для длинных чисел и узнал новые алгоритмы. Укрепил свои навыки в RAII.

Список литературы

[1] Томас Х. Кормен, Чарльз И. Лейзерсон, Рональд Л. Ривест, Клиффорд Штайн. *Алгоритмы: построение и анализ, 2-е издание.* — Издательский дом «Вильямс», 2007. Перевод с английского: И. В. Красиков, Н. А. Орехова, В. Н. Романов. — 1296 с. (ISBN 5-8459-0857-4 (рус.))