Московский авиационный институт (национальный исследовательский университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №6 по курсу «Дискретный анализ»

Студент: М. А. Бронников
Преподаватель: А. А. Кухтичев
Группа: М8О-207Б

Дата:
Оценка:
Подпись:

Москва, 2019

Лабораторная работа №6

Задача: Небходимо разработать программную бибилиотеку на языке 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.

Список арифметических операций:

- Сложение (+).
- Вычитание (-).
- Умножение (*).
- Возведение в степень ($\wedge$).
- Деление (/).

В случае возникновения переполнения в результате вычислений, попытки вычесть из меньшего большее, деления на ноль или попытки возведения нуля в нулевую степень программа должна вывести на экран строку Error.

Список условий:

- Больше (>).
- Меньше (<).
- Равно (=).

В случае выполнения условия программа должна вывести на экран строку true, а в противном случае — false.

Ограничения: Количество десятичных разрядов целых чисел не превышает 100000. Основание выбранной системы счисления для внутреннего представления «длинных» чисел должно быть не меньше 10000.

1 Описание

Как сказано в [1]:«Под длинной арифметикой мы, как и все, будем понимать математические действия над числами, которые по своему размеру превышают ограничения стандартных типов данных».
Хранить число в памяти мы будем хранить в виде массива чисел, где каждая i-ая ячейка массива — i-ый разряд числа с основанием 10000.
Помимо массива «длинное» число будет хранить количество разрядов внутреннего представления (количество занятых ячеек массива) и флаг, сигнализирующий о том, что число является ошибкой, возвращаемой арифметическими операциями.

Описание реализаций арифметических операций:

- *Сложение:* Результат — «длинное» число, в котором i-ая ячейка является суммой i-ых ячеек массивов 2-ух операндов между собой с добавлением к сумме еденицу, если результат суммы на i-1 -ом шаге превысил основание, с последующей проверкой, чтобы полученная сумма не превысила основание.
- *Вычитание:* Для получения результата последовательно вычтаем из i-ой ячейки 1-го операнда i-ый разряд 2-го числа, при этом если получился отрицательный результат, то добавляем к нему основание и вычитаем 1 из i+1 -ой ячейки 1-го операнда. Если выяснилось, что первый операнд больше второго, устанавливаем флаг ошибки и возвращаем результат.
- *Умножение:* Реализуется обыкновенное умножение в столбик, со вложенными циклами(что дает не самую оптимальную сложность $O(n^2)$), где на i-ом шаге реализуется умножение i-ой ячейки 1-го числа со 2-ым числом.
- *Возведение в степень:* После проверки особых случаев преобрзуем 2-ой операнд в базовый unsigned (так как 2 разряда «длинного» числа помещаются в базовый тип, а при большем количестве разрядов произойдет гарантированное переполнение с выставленным флагом ошибки), после чего реализуем ускоренное возведение в степень из [1].
- *Деление:* Реализуем обычное деление в столбик. [1]:«На каждой итерации имеем текущее значение, которое пытаемся уменьшить на максимально большое количество раз делимым. Итак: как будем искать это “максимально большое количество”? Ответ прост. Вместо того, чтобы линейно перебирать цифры в интервале от 0 до основания в поисках нужного ответа, мы будем использовать бинарный поиск». Итак при делении последовательно ищем максимальное число на которое можно уменьшить текущее значение. Эти числа и будут составлять разряды получаемого длинного числа.

Реализация **условий** довольно проста и однотипна, поэтому опустим ее описание.

2 Исходный код

Реализуем множество «длинных» чисел в виде класса *BigInteger* для удобства работы и хранения чисел.

Для непосредственного хранения числа каждому члену класса понадобится вектор *data*, каждая ячейка которого будет хранить один разряд длинного числа, начиная с младшего.
Помимо этого, класс будет содержать атрибут *amount*, значение которого будет соответствовать количеству разрядов «длинного» числа.
Так как возможны случаи, когда результатом работы оператора является ошибка, добавим классу последний атрибут $flag$, принимающий истинные значения в случае ошибочного результата.

```
#include <iostream>
#include <cstring>
#include <vector>
#include <string>

using namespace std;

#define BASE 10000
#define MAX_I 25000
#define NUM_NUMS 4

class BigInteger{
public:
  BigInteger(const BigInteger& obj);
  BigInteger();
  BigInteger(const unsigned int s1, const unsigned int s2);
  friend const BigInteger operator+(const BigInteger& left, const BigInteger& right);
  friend const BigInteger operator-(const BigInteger& left, const BigInteger& right);
  bool operator==(const BigInteger& right);
  friend const BigInteger operator*(const BigInteger& left, const BigInteger& right);
  friend const BigInteger operator/(const BigInteger& left, const BigInteger& right);
  friend const BigInteger operator^(const BigInteger& left, const BigInteger& right);
  bool operator>(const BigInteger& right);
  bool operator<(const BigInteger& right);
  friend istream& operator>>(istream& is, BigInteger& obj);
  friend ostream& operator<<(ostream& os, const BigInteger& obj);

private:
  std::vector<unsigned int> data;
  bool flag;
  unsigned int amount;
};
```

Основные функции и операторы	
BigInteger(const unsigned int s1, const unsigned int s2)	Конструктор «длинного числа», резервирующий память на количество разрядов равное максимальному из аргументов
friend const BigInteger operator+(const BigInteger& left, const BigInteger& right)	Оператор сложения чисел, возвращает новое число
friend const BigInteger operator-(const BigInteger& left, const BigInteger& right)	Оператор вычитания чисел, возвращает новое число
bool operator==(const BigInteger& right)	Оператор проверки на равенсвто двух чисел
friend const BigInteger operator*(const BigInteger& left, const BigInteger& right)	Опреатор умножения двух чисел столбиком, возвращает новое число
friend const BigInteger operator/(const BigInteger& left, const BigInteger& right)	Оператор деления двух чисел друг на друга
friend const BigInteger operator∧(const BigInteger& left, const BigInteger& right)	Возведение левого операнда в степень, равную второму операнду
bool operator>(const BigInteger& right)	Оператор возвращает true, если первые операнд больше второго
bool operator<(const BigInteger& right)	Оператор возвращает true, если первые операнд меньше второго
friend istream& operator»(istream& is, BigInteger& obj)	Оператор считывания числа из стандартного потока
friend ostream& operator«(ostream& os, const BigInteger& obj)	Оператор вывода числа в стандартный поток

3 Консоль

```
max@max-X550CC:~/DA/lab6$ ls
main.cpp  test
max@max-X550CC:~/DA/lab6$ g++ -Wall -pedantic main.cpp -o run
max@max-X550CC:~/DA/lab6$ cat test
100000000000000000000000
10000000000000000000
*
389434329835214353464363534
35435325432838311111114
*
904094384738493247293847 3843
234354342324524523
-
972323
2173937
>
2
3
-
10
52452352
^
4523524352435234523452352342
4523524352435234523452352342
=
max@max-X550CC:~/DA/lab6$ ./run <test
1000000000000000000000000000000000000000000000000
13799732212430114674236280417963096860071356371687 6
904094384715057813061394932 0
false
Error
Error
true
```

4 Выводы

Выполнив шестую лабораторную работу по курсу «Дискретный анализ», я научился реализации «длинной» арифметики на языке C++ и получил представлении о том как она описанна в других языках программирования, таких как Java. Приобретенный мною навык может быть полезным для меня в будущем, если мне придется столкнуться с программами, работающими со сколь угодно большими аналитическими данными.

Данная работа оставила у меня смешанные впечатления, так как реализация условий и большинства арифметических операций оказалась тривиальным делом, которое практически не создало никаких проблем. Однако операции деления и возведения в степень задержали меня в написании программы и заставили начать поиск оптимальных (иногда даже нестандартных) решений, что дало неоценимый опыт работы с «длинными» числами и расширило мой кругозор.

В итоге я не пожалел о том, что мне пришлось выполнить лабораторную работу на такую тему, но сильного интереса и удвольствия при выполнении задания я не ощутил.

Список литературы

[1] *Алгоритмы на C++ (Олимпиадный подход)*
URL: `http://cppalgo.blogspot.com/2010/05/blog-post.html` (дата обращения: 16.05.2019).