Лабораторная работа №6 по курсу дискретного анализа: Калькулятор

Выполнил студент МАИ группы М8О-201Б *Ефимов Александр.*

Постановка задачи

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.
Список арифметических операций:

- Сложение (+);
- Вычитание (-);
- Умножение (*);
- Возведение в степень (^);
- Деление (/).

В случае возникновения переполнения в результате вычислений, попытки вычесть из меньшего числа большее, деления на ноль или возведении 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, в противном случае — false.

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

Метод решения

Сравнение чисел производится достаточно просто: два числа равны, если равен каждый их разряд. Если они не равны, то больше то число, самый старший разряд которого больше. Если число разрядов не одинаково, то больше число, у которого больше разрядов.

Сложение происходит по разрядам, начиная с младшего разряда. После сложения всех разрядов может появится один дополнительный разряд со значением 1.

Вычитание происходит по разрядам, начиная с младшего разряда. После вычитания могут появится ведущие нули, которые необходимо ликвидировать. Учитывая условия задания, не должнен произойти вычет из несуществующего разряда в конце, иначе второе число больше первого (проверку производят до вычета).

Умножение выполняется путем простого умножения каждого разряда первого числа на каждый разряд второго (соответственно сложность $O(n*m)$). При этом максимальное количество разрядов в новом числе равно сумму количества разрядов у множетелей.

Деление выполняется простым делением в столбик: в цикле помещается единственный разряд числа, начиная с самого левого, в отдельное число (далее *current*). К этому числу подбирается такое число, которое при умножении на делитель дает наименьшую положительную разность между полученным числом и *current*. Полученное число является одним из разрядов ответа, поэтому оно в него и добавляется, и оно, после умножения на делитель, вычитается из *current*.

Описание программы

1. **bigint.h** Содержит в себе класс *TBigInt* и объявления всех используемых им конструкторов и методов.

    - *TPrimitive* – базовый тип для большого числа, который будет использоваться для хранения самих разрядов;
    - *Конструкторы* – содержит конструкторы для другого длинного целого числа, для числа типа TPrimitive (и меньше) и для строки, содержащей число;
    - *Операторы* – выполняют все соответствующие действия, как и числовые типы, за исключением "^) который возводит в степень;
    - *SumAtPos, SubAtPos* – выполняют соответственно сумму и вычитание разрядов двух чисел под указанным индексом. Если индекс указывает за пределы числа, то ничего не прибавляет и не вычетает;
    - *RemoveLeadingZeroes* – отбрасывает все ведущие нули числа;
    - *Exponentate* – рекурсивно возводит в степень число, причем делает это путем дробления степени на две, в два раза меньшие, и умножения полученных двух чисел. Если степень нечетная, то результат дополнительно домножается на основание. **Переменные:**
    - *container* – содержит в себе все разряды числа;

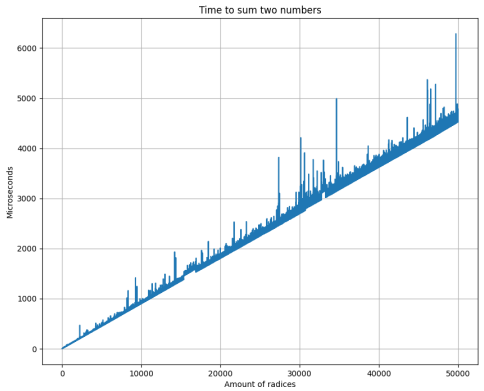
- *BASE_ VAL* – Основание и максимальный размер каждого разряда;
- *BASE_ LEN* – Количество цифр, которое может держать в себе разряд.

2. **bigint.cpp** – сама реализация объявленых методов и операторов.
3. **main.cpp** – сама программа, получающая два числа и оператор, выполняющий действие с этими числами.

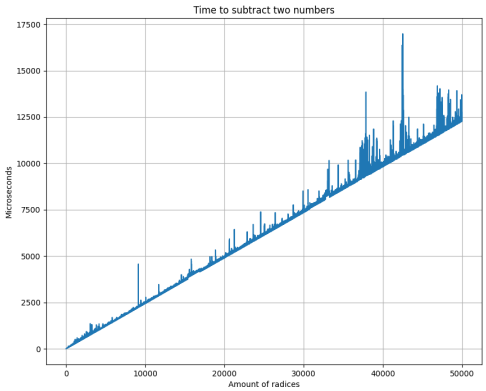
Дневник отладки

Номер	Ошибка	Обнаруженная причина	Исправление
1	Превышено реальное время работы	Попытка высчитать тривиальный случай (т.е. возведение единицы в длинное число)	Отдельно добавить возведение единицы и нуля

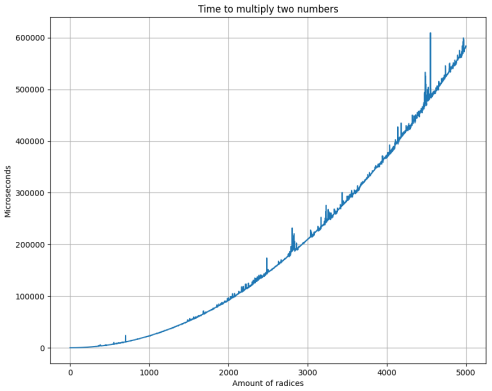
Тест производительности



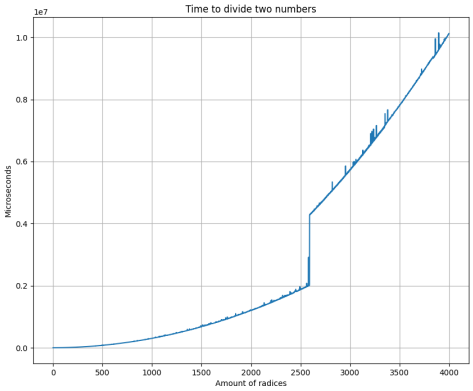
(a) Время суммы двух чисел. .



(b) Время вычитания двух чисел. .



(c) Время умножения двух чисел. .



(d) Время деления двух чисел. .

На рисунках (a)-(d) показано время (в микросекундах), которое ушло на действие, выполняемое над двумя длинными числами (операция для каждого рисунка указана под ними). По оси абсцисс указано количество разрядов у обоих чисел.

Как и ожидалось, сумма и вычитание имеют линейную сложность, а умножение – квадратичную.

Почему возникает такой скачок при делении определить не удалось: повторные тесты дают такой же скачок, но в другом положении, а ручной запуск указанного количество разрядов дает другое время.

Выводы

Чаще всего во внутренней работе программы достаточно длины чисел, которые умещаются в доступные процессору 64 бита (иногда и 128). Но в математически специализированных программах (например *MatLab*, *Sage*) или в конкретных областях (криптография, подсчет RSA ключа) уместить результаты в одно слово без переполнения невозможно. Требуется более сложный метод хранения длинных чисел.

Для этого и придумана длинная арифметика: хранение и работа с неограниченно (разве что рабочей памятью) длинными числами за счет потери скорости (все числа, умещающиеся в одно слово, высчитываются за одну инструкцию, в то время, как длинная арифметика требует дополнительные функциональные вызовы).

Самые простые операции над длинными числами могут быть реализованы за линейное время путем простой по разрядной суммы двух длинных чисел. Простое умножение можно выполнить за $O(n^2)$, но существуют и улучшенные алгоритмы, такие как алгоритм Карацубы, умножающего за $O(n^{1.5})$. Деление выполняется путем выполнения деления уголком.