Лабораторная работа № 6 по курсу дискретного анализа: Калькулятор

Выполнил студент группы 08-208 МАИ *Скворцов Александр.*

Условие

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 операций: сложение(+), вычитание(-), умножение(*), возведение в степень(^), деление(/), больше(>), меньше(<), равно(=).

Метод решения

Поскольку в длинной арифметике числа могут быть неограниченно большими, для их представления следует воспользоваться вектором, каждый элемент которого предствляет один разряд. Система счисления может быть произвольной, но для удобства ввода-вывода я выбрал основание, равное 10000.

Сами алгоритмы сложения, вычитания, умножения и деления полностью соответствуют школьным, которые изучаются в начальных классах. Например, сложение начинается с соответствующих младших разрядов с переносом остатка на следущую итерацию. Операции сравнения начинаются со старших чисел до первого несовпадения. Единственную сложность в работе представлют возведение в степень и деление.

Так сушествует быстрый бинарный алгоритм возведения числа в степень, основанный на представлении самой степени n в двоичном виде $(m_k m_{k-1} ... m_1)_2$. Тогда $x^n = x^{m_k \cdot 2^k} x^{m_{k-1} \cdot 2^{k-1}} ... x^{m_1}$. То есть количество уможений равно $log_2 n$, где на каждой итерации i вычисляется $x^{2^i} = x^{2^{i-1}} x^{2^{i-1}}$.

В школьном алгоритме деления присутствует элемент угадывания разрядов результата, этот процесс нужно как-то формализовать. Узнать частное при делении $(u_n ... u_1)$ на $(v_{n-1} ... v_1)$ можно с помощью $\hat{q} = \left\lfloor \frac{u_n * b + u_{n-1}}{v_{n-1}} \right\rfloor$, которое больше истинного q не больше, чем на 3. Однако при этом v_{n-1} должно быть больше или равно $\lfloor b/2 \rfloor$, но мы всегда можем этого добиться, умножив каждое из чисел на $\lfloor b/(v_{n-1} + 1) \rfloor$.

Описание программы

Для начала стоит отметить, что записывать числа удобно в вектор слева направо от младшего разряда с старшему, так как при добавении нового разряда легче его дописать, а не сдвигать другие числа.

Сама библиотека предоставляет новый класс TBigInt с перегруженными для него операциями ввода-вывода и арифметическими операторами, и представлена интерфейсным файлом bigint.h и файлом с реализацией bigint.cpp.

1. main.cpp — считывает операнды и операторы, после чего выполняет заданные действия
2. bigint.h — интерфейс класса TBigInt
3. bigint.cpp — реализация класса TBigInt
    - std::vector num — вектор, хранящий длинное число.
    - inline int& operator[](int k) — возвращает элемент, находящицся на k-ой позиции вектора num.
    - inline int operator[](int k) const — константная версия предыдущего метода.
    - inline size_t size() const — возвращает размер вектора num.
    - TBigInt& Minus(const TBigInt& i2, int l, int r) — вычитает из данного числа i2 в позициях с l по r.
    - TBigInt& Plus(const TBigInt& i2, int l, int r) — складывает i2 с данным числом в позициях с l по r.
    - bool LessThen(const TBigInt& i2, int l, int r) — проверяет, меншье ли данное число, чем i2, в позициях с l по r.
    - TBigInt() — конструктор по-умолчанию;
    - TBigInt(int i) — конструирует длинное число из короткого;
    - int ToInt() — приводит, данное длинное число к короткому, если это возможно, иначе возвращает MAX_INT.
    - friend std::istream& operator»(std::istream& is, TBigInt& i) — вводит TBigInt из потока is.
    - friend std::ostream& operator«(std::ostream& os, const TBigInt& i) — выводит число i в поток os.
    - friend bool operator==(const TBigInt& i1, const TBigInt& i2) — сравнивает два длинных числа.
    - friend bool operator>(const TBigInt& i1, const TBigInt& i2) — проверяет, больше ли i1, чем i2.
    - friend bool operator<(const TBigInt& i1, const TBigInt& i2) — проверяет, больше ли i2, чем i1.
    - friend TBigInt operator+(const TBigInt& i1, const TBigInt& i2) — возвращает результат скложения двух длинных чисел.

- friend TBigInt operator-(const TBigInt& i1, const TBigInt& i2) — возвращает результат вычитания двух длинных чисел.
- friend TBigInt operator*(const TBigInt& i1, const TBigInt& i2) — возвращает результат умножения двух длинных чисел.
- friend TBigInt operator/(TBigInt i1, TBigInt i2) — возвращает результат деления двух длинных чисел.
- friend TBigInt pow(TBigInt val, int power) — возводит число val в степень power.

Дневник отладки

№	Ответ чекера	Причина ошибки
1, 2	Wrong answer at test 2plus.t	программа корректно складывала большее число с меньшим, но не наоборот
3	Wrong answer at test 5div.t	Метод LessThen, возвращал неправильный ответ при равентсве чисел
4	Unknown file extension	Неаправильно упакованный архив
5-8	Wrong answer at test 8div.t	Некорректное деление при равном количестве разрядов или при $u/v > b$

Тест производительности

Очевидно, что операции сравнения, а также сложения и вычитания работают за $O(n)$, так как в теле каждой функции присутствует всего один цикл for с постоянным количеством действий за итерацию. Протестируем лучше умножение и деление.

1. Умножение $n * m$

кол-во разрядов n	кол-во разрядов n	Время работы в мс
5000	1000	160
10000	2000	578
300000	4000	3410

   Т.е. при увеличении входных данных в 2 раза время увеличивается в 4 раза. При увеличении первого множителя в 3 раза, а второго в 2 время возрастает в 6 раз. Таким образом сложность умножения $O(mn)$.

2. Деление n/m

   Всего выполняется $n - m$ итераций, на каждой из которых производится постоянное количество вычитаний и умножений длинного на коротокое, т.е. работа на итерации $O(n)$. Тогда общая сложность $O((n - m)m)$.

Пусть $m \sim n$, тогда сложность $O(m)$.

кол-во разрядов n	кол-во разрядов n	Время работы в мс
40004	40000	20
80008	80000	42
160016	160000	80

Пусть $n \sim m^2$, тогда сложность $O(m^3)$.

кол-во разрядов n	кол-во разрядов n	Время работы в мс
10000	100	124
40000	200	914
90000	300	7002

Выводы

Длинная арифметика позволяет работать с большими числами, длина которых ограничена только оперативной памятью компьютера, и, очевидно, находит свое применение в различных областях математики. Эффективность работы с ними зависит не только от выбранного алгоритма, но и от системы счисления. Чем она больше, тем меньше элементарных операций нужно выполнить. Обычно основание системы ограничено одним машинным словом. Для убонобства организации ввода-вывода она может быть ограничено степенью 10.

Сложность программирования данных "школьных"алгоритмов невилика, однако лично мне потребовалось некотрое время, что понять принцип и теорию деления. Стоит отметить, что существуют также и другие более эффективные алгоритмы умножения и деления, но реализвоть их не так просто.