|  |  |
| --- | --- |
| Изображение выглядит как коллекция картинок  Автоматически созданное описание  МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ Федеральное государственное автономное учреждение высшего образования **«Дальневосточный федеральный университет» (ДВФУ)** | |
| **ИНСТИТУТ МАТЕМАТИКИ И КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ**  **Департамент компьютерного и математического моделирования** | |
| **ДОКЛАД**  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Эффективная\_длинная\_арифметика\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  по образовательной программе подготовки бакалавров  по направлению 09.03.03 «Прикладная информатика» | |
|  | Студент группы № Б9121-09.03.03пикд-5 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Ли Д. С.  (подпись)  «\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2022г. |
| г. Владивосток  2022 | |

Оглавление

[Введение 3](#_Toc126432996)

[Теоретическая часть 4](#_Toc126432997)

[Стандартные типы данных 4](#_Toc126432998)

[Определение 4](#_Toc126432999)

[Общее 4](#_Toc126433000)

[Класс 5](#_Toc126433001)

[Сложение 6](#_Toc126433002)

[Вычитание 7](#_Toc126433003)

[Умножение 9](#_Toc126433004)

[Деление и остаток 9](#_Toc126433005)

[Возведение в степень 10](#_Toc126433006)

[Сравнение 10](#_Toc126433007)

[Инкрементирование 10](#_Toc126433008)

[Декрементирование 10](#_Toc126433009)

[Дополнительные функции 10](#_Toc126433010)

[Тесты 12](#_Toc126433011)

[Предисловие 12](#_Toc126433012)

[Тест скорости работы сложения 12](#_Toc126433013)

[Тест скорости работы вычитания 13](#_Toc126433014)

[Тест скорости работы умножения 13](#_Toc126433015)

[Тест скорости работы деления 14](#_Toc126433016)

[Тест скорости работы возведения в степень 14](#_Toc126433017)

[Заключение 15](#_Toc126433018)

[Источники 16](#_Toc126433019)

# Введение

Известно, что арифметические действия, выполняемые компьютером в ограниченном числе разрядов, не всегда позволяют получить точный результат. Более того, существуют ограничения – размер чисел, с которыми возможно работать.

Если необходимо выполнить арифметические действия над очень большими числами, например 30! = 265252859812191058636308480000000. То в таких случаях необходимо позаботиться о представлении больших чисел в машине и о точном выполнении арифметических операций над ними.

Числа, для представления которых в стандартных компьютерных типах данных не хватает количества двоичных разрядов, называются «длинными». Реализация арифметических операций над такими «длинными» числами получила название «Длинной арифметики».

«Длинная арифметика» - в вычислительной технике операции (сложение, умножение, вычитание, деление, возведение в степень и т.д.) на числами, разрядность которых превышает длину машинного слова данной вычислительной машины. Эти операции реализуются не аппаратно, а программно, используя базовые аппаратные средства работы с числами меньших порядков.

**Проблема**: существует класс задач, которые нельзя решить с помощью стандартных типов данных.

**Цель**: изучение метода «Длинная арифметика».

Данная исследовательская работа посвящена «Длинной арифметике» и её реализации на языке С++».

# Теоретическая часть

## Стандартные типы данных

Рассмотрим основные целочисленные типы данных языка C++ и диапазон их значений (табл.1)

Таблица

|  |  |
| --- | --- |
| short | От -32 768 до 32 767 |
| unsigned short | От 0 до 65 535 |
| int | От -2 147 483 648 до 2 147 483 647 |
| unsigned int | От 0 до 4 294 967 295 |
| long | От -2 147 483 648 до 2 147 483 647 |
| unsigned long | От 0 до 4 294 967 295 |
| long long | От -9 223 372 036 854 775 808 до 9 223 372 036 854 775 807 |
| unsigned long long | От 0 до 18 446 744 073 709 551 615 |

Из таблицы 1 следует, что наибольшее число, которым мы можем оперировать это 18 446 744 073 709 551 615 или просто 264-1.

Но число 264 уже не помещается ни в один из представленных типов данных. Для расчёта «длинных» чисел» потребуется другой метод.

## Определение

### Общее

Длинная арифметика – набор алгоритмов для поразрядной работы с числами произвольной длины. Она применяется как с относительно небольшими числами, превышающими ограничения типа long long в несколько раз, так и с по-настоящему большими числами (чаще всего до 10100000).

Для работы с «длинными числами их разбивают на разряды. Размер разряда может быть произвольным, но чаще всего используются следующие:

* 10 – по аналогии с цифрами числа в десятичной системе, для простоты понимания и отладки.
* 104 – наибольшая степень десятки, квадрат которой не превышает ограничения типа int. Используется для максимальной эффективности при хранении разрядов как чисел типа int.
* 109 – аналогично предыдущему пункту, но для типа long long. Позволяет достичь максимально возможной эффективности.

*(Ограничения на квадрат размера разряда связанны с необходимостью перемножать между собой разряды. Если квадрат разряда превышает ограничение своего типа, при умножении возможны переполнения.)*

В большинстве реализаций разряды хранятся в порядке, обратным привычному для упрощения работы с ними. Например, число 578002300 при размере разряда 104 представляется следующим массивом:

Количество разрядов числа может быть как ограничено, так и не ограничено, в зависимости от типа используемого контейнера: массива константной длины или вектора.

### Класс

Для начала следует определиться, какие переменные в классе нам необходимы.

Из пункта «Общее» следует, что необходимо создать массив, в котором будут храниться разряды «длинного» числа. Для максимальной эффективности следует использовать \_int64 для хранение в одном разряде числа с ограничением 109. Но для упрощения написания деления и умножения в проекте будем использовать 1 цифру в 1 разряде. Так же поставим цель упросить код, для понимания

Далее требуется определять знак (положительный\отрицательный) у «длинного» числа. Для этого следует добавить переменную типа данных bool. Что в дальнейшем будет означать: при true – положительное, а при false – отрицательное.

Так же следует добавить константу, которая будет отмечать ограничение разряда числа. В данном случае эта константа будет равна 1 000 000 000, что равно 109. Но так как цель упростить код, то использовать константу для ограничения числа не нужно. (Так как в 1 разряде 1 цифра)

Все переменные следует отнести к доступу private, для ограничения доступа вне класса, так же в доступ private попадают функции, которые пользователь не должен вызывать. Так же отдельно они будут помечены: название функций будет начинаться с «\_».

### Сложение

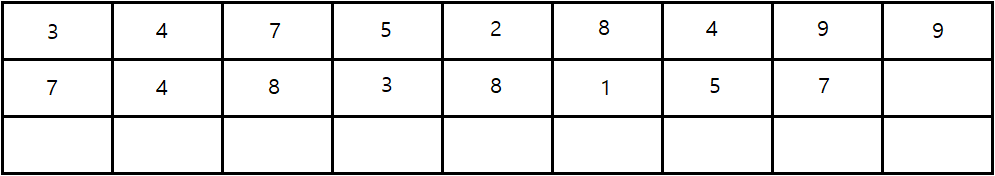
«Длинную арифметику» часто сравниваю с детским вычислением «в столбик». Это достаточно справедливо, так как оба метода основаны на поразрядных операциях.

Перед сложением необходимо проверить следующие условия:

* Первое число отрицательное, второе положительное: в этом случае достаточно отнять из второго числа первое, поменяв знак первого.
* Первое число положительно, второе отрицательное: в этом случае достаточно отнять из первого числа второе, поменяв знак второго.
* Оба числа отрицательные: нужно сложить модули чисел, а затем поменять знак.

Учитывая вышеописанные условия в алгоритм вычитание могут попасть числа только положительные. Для примера будем использовать массив с ограничением размера разряда 104 (int), обратный порядок хранения разрядов. Сложим числа:  
994 825 743  
   75 183 847

Рисунок



На рисунке 1 показан способ хранения чисел в массиве. Далее сложим первый разряд.

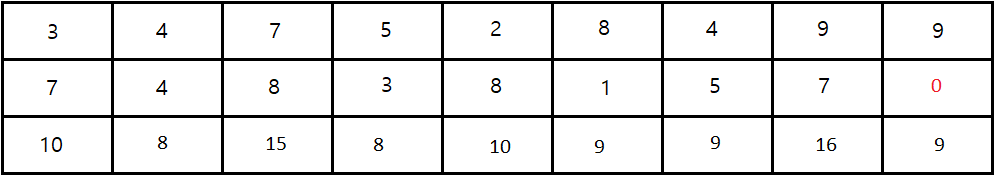
Рисунок

Изображение выглядит как седзи, яркий

Автоматически созданное описание

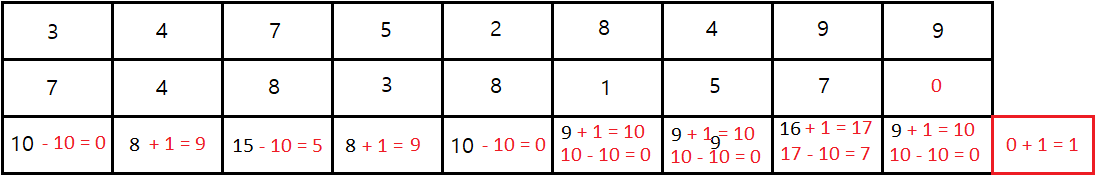
На рисунке 2 показан результат сложения первого разряда. Сложим остальные разряды.

Рисунок



На рисунке 3 показан результат сложения остальных разрядом. Можем заметить, что есть разряды, в котором количество цифр превышает задуманное. Проведем операцию переноса лишних разрядов.

Рисунок



На рисунке 4 представлен полный результат сложение двух вышеописанных чисел. Он равен 1 070 009 590. Нужно заметить, что лишнее число в разряде переносится в следующий разряд, если в сложении получается еще один лишний разряд, то действие повторяется. В конце создаем дополнительную ячейку для разряда, так как в предыдущем количество цифр превысило допустимое.

Таким образом в дальнейшем будет реализовано сложение.

### Вычитание

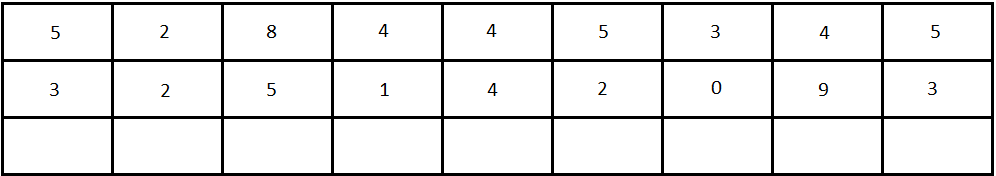
Вычитание реализуется симметрично сложению. Так, как и сложение, происходит в столбик.

Перед вычитанием необходимо проверить следующие условия:

* Второе число отрицательное: достаточно сложить модули двух чисел.
* Первое число отрицательное: сложить модули двух чисел, затем поменять знак результата.
* Первое число меньше второго: отнять из второго числа первое, затем поменять знак результата.

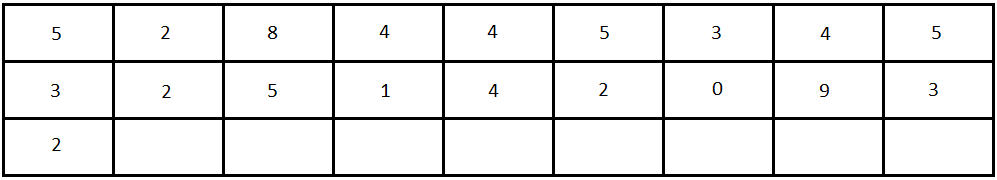
Учитывая вышеописанные условия в алгоритм вычитание могут попасть числа только положительные и гарантированно первое число будет больше второго. Условия для примера вычитания будут аналогичны сложению. Произведем вычитание чисел:  
543 544 825  
390 241 523

Рисунок



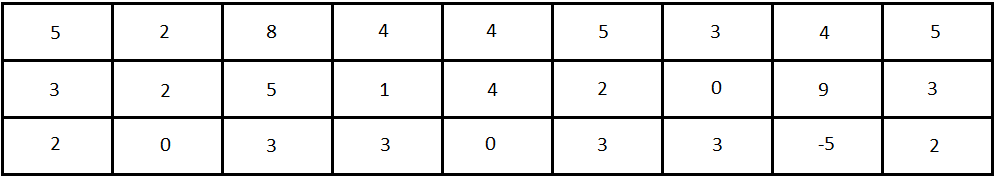
На рисунке 5 показано расположение. Произведем вычитание первого разряда.

Рисунок



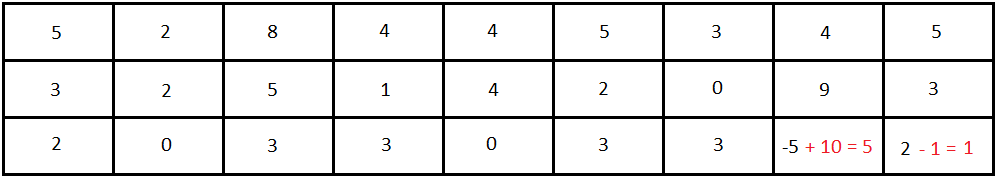
На рисунке 6 результат вычитания первых разрядов. Произведем вычитание остальных разрядов.

Рисунок



На рисунке 7 результат вычитания всех разрядов. Можно заметить, что один из разрядов получился отрицательным. Следует занять единицу у следующего разряда.

Рисунок



На рисунке 8 показано занятие 1 у разряда выше. Завершим вычитание. Вычитание завершено. Разница равна 153 303 302

Таким образом в дальнейшем будет реализовано вычитание.

### Умножение

Реализаций умножения существует множество. Но самая эффективная – алгоритм Карацубы. Его сложность O(N1.58). Что превосходит обычное умножение в стобик (O(N2)). Алгоритм Карацубы основан на парадигме «разделяй и властвуй».

Вначале нужно проверить длину входящих чисел, если они меньше 10 (оптимальный вариант длинны для разделения метода карацубы и обычного умножени), то преобразуем числа к типу данных longlong и производим обычное умножение. В ином случае – используем алгоритм Карацубы.

Дальше разделяем числа на половину длины наибольшего. Рекурсивно перемножаем соответствующие части чисел в отдельные переменные (X, Y). Также перемножаем суммы соответствующих частей в отдельную переменную (S).

Далее создаем 3 переменные, в которые будут добавляться нули в зависимости от их разделения по разрядам. (Максимум нулей, половина нулей, ноль нулей).

Складываем 3 созданные переменные. (Алгоритм действует рекурсивно, до длинны числа меньше 10).

### Деление и остаток

Производится в отдельной функции, которая возвращает pair<BigInt,BigInt>, где first – результат целочисленного деления, а second – остаток от деления.

Сразу производит деление чисел, которые возможно преобразовать в longlong с помощью встроенного деления в ЯП.

Если не получилось преобразовать в longlong, то создает 2 переменные типа BigInt – mod и div, остаток от деления и результат деления соответственно, где mod изначально равен делимому. Так же создает абсолютное значение делителя (без знака).

Далее в цикле подставляет делитель под высшие разряды делимого с помощью функции добавления нулей. Производит вычитание из переменной mod, до тех пор, пока mod больше делителя. При каждом вычитании из mod прибавляет к переменной div значение, к которому добавили такое же количество нулей, что и в делитель. И повторяет цикл.

На выходе получается 2 числа упомянутых выше. Которые соответственно возвращаются.

### Возведение в степень

Рекурсивно умножает число само на себя, до получения нужной степени.

### Сравнение

Для описания функций, всех операторов сравнения, достаточно описать оператор равенства и один из операторов больше или меньше.

Рассмотрим равенство: для начала проверим различие знаков. Далее проверим числа на нули. После этого проверим числа по длине, если они не равны, то числа не равны. Дальше сравним каждый разряд, если какой-то из разрядов не равен другому, то числа не равны. В ином случае числа равны.

Рассмотрим операцию меньше: для начала проверим равенство, благодаря описанному выше алгоритму. Далее проверим различие знаков «длинных» чисел. Далее сравним длину чисел и, наконец, сравним поразрядно.

Все остальные операции сравнения (!=, >, <=, >=) реализуются с помощью уже написанных операций.

### Инкрементирование

Достаточно с помощью сложения вернуть значение +1. (При условии ++x).

Реализация x++: увеличивание на 1, затем возвращение значения -1.

### Декрементирование

Аналогично инкрементированию, описанного выше.

### Дополнительные функции

* Функция изменения знака: копирует текущее «длинное» число, меняет переменную, которая хранит в себе знак, на противоположную себе.
* Функция преобразования в строку: использует библиотеку stringstream для создания строки в потоке.
* Абсолютное значение: избавляется от отрицательного знака.
* Добавление 0: принимает значение общего количество разрядов, которое нужно получить, затем добавляет нули.
* Разделение числа: разделяет число на 2 части.
* Проверка на вместимость в longlong: проверяет поместится ли число в тип данных longlong.
* Преобразования в longlong: преобразует длинное число в longlong
* Длина числа: возвращает длину длинного числа
* Является четным или нечетным: возвращает true или false в зависимости от числа и выбранной функции.
* Проверка числа на 0 и на 1: возвращает true или false.
* Проверка на положительное значение: возвращает true соответственно.
* Удаление нулей слева: удаляет все первые нули числа.
* Перенос разряда: вспомогательная функция для сложения и вычитания.
* Функция, которая используется после любой операции: удаляет лидирующие нули, затем проверяет число на нуль.

# Тесты

### Предисловие

Тесты проведены с характеристиками пк:

* Процессор: AMD Ryzen 3 3250U with Radeon Graphics 2.60 GHz
* Оперативная память: 12,0 ГБ (доступно: 9,92 ГБ)
* Тип система: 64-разрядная операционная система, процессор x64
* Видеокарта: встроенная

### Тест скорости работы сложения

Можно заметить, что сложность повышается резко на 1000 символов до 1мс, дальше сложность растет медленно 2000 символов 2мс, 4000 символов 3мс. Предположительно дальше: 6000 символов 4мс, 8000 символов 5мс.

### Тест скорости работы вычитания

Можно заметить, что сложность растет слишком быстро после 1000 символов. На 2300 символов уже выполняется за 52мс, а на 4500 за 95мс. Предполагая дальнейший рост: 6000 символов 130мс, 8000 символов 160мс.

### Тест скорости работы умножения

Можно заметить, что сложность растет на 600 символов уже до 500мс, а на 2300 уже до 2400мс. Предполагая дальнейший рост: 5000 символов 5000мс, 7000 символов 7000мс.

### Тест скорости работы деления

Можно заметить, что сложность растет плавно. 666 символов 80мс, 2000 символов 120мс. Предполагая дальнейший рост: 3000 символов 160мс, 4000 символов 200мс.

### Тест скорости работы возведения в степень

Изначально рассматривается возведение 2 в n степень.

Можно заметить слишком высокий рост. 10000 степень занимает 8000мс, 20000 степень занимает 23000мс. Предполагая дальнейший рост: 30000 степень 100000мс, 40000 степень 400000мс.

# Заключение

В процессе работы над данной темой был создан класс работы с «длинными» числами, написаны функции для основных арифметических операций (сложение, вычитание, умножение, деление, взятие остатка). А так же различные функции для удобства работы с написанным классом.

# Источники

1. https://brestprog.by/topics/longarithmetics/
2. https://ru.wikipedia.org/wiki/Длинная\_арифметика
3. https://e-maxx.ru/algo/big\_integer
4. https://habr.com/ru/post/207754/
5. https://inf2086.ru/crypto\_basics/book/algo\_long\_arithmetic.html
6. https://megaobuchalka.ru/6/33526.html
7. https://topref.ru/referat/50385.html
8. https://intellect.icu/dlinnaya-arifmetika-s-primerami-na-si-8291
9. http://cppstudio.com/post/5036/
10. https://rg-gaming.ru/kompjutery/dlinnaja-arifmetika-c-delenie
11. http://comp-science.narod.ru/DL-AR/okulov.htm
12. https://habr.com/ru/post/124258/
13. https://studfile.net/preview/7014549/page:6/
14. https://forkettle.ru/vidioteka/programmirovanie-i-set/algoritmy-i-struktury-dannykh/73-lektsii-ot-nou-intuit/bazovye-algoritmy-dlya-shkolnikov-lektsii-ot-nou-intuit/572-lektsiya-9-dlinnaya-arifmetika
15. https://moluch.ru/archive/180/46418/
16. https://dic.academic.ru/dic.nsf/ruwiki/1299482
17. https://studassistent.ru/c/dlinnaya-arifmetika-na-si-c-si
18. https://www.pvsm.ru/algoritmy/29587
19. https://itnan.ru/post.php?c=1&p=451860
20. https://habr.com/ru/post/172285/
21. https://pro-prof.com/forums/topic/разность-чисел-длинная-арифметика-си
22. https://lisiynos.github.io/s1/long\_ar.html
23. https://habr.com/ru/post/135590/
24. https://inf.1sept.ru/2000/1/art/okul1.htm
25. https://habr.com/ru/post/578718/
26. https://ru.stackoverflow.com/questions/1320123/Оптимизация-длинной-арифметики-c
27. https://www.stud24.ru/programming-computer/dlinnaya-arifmetika/22260-63531-page1.html