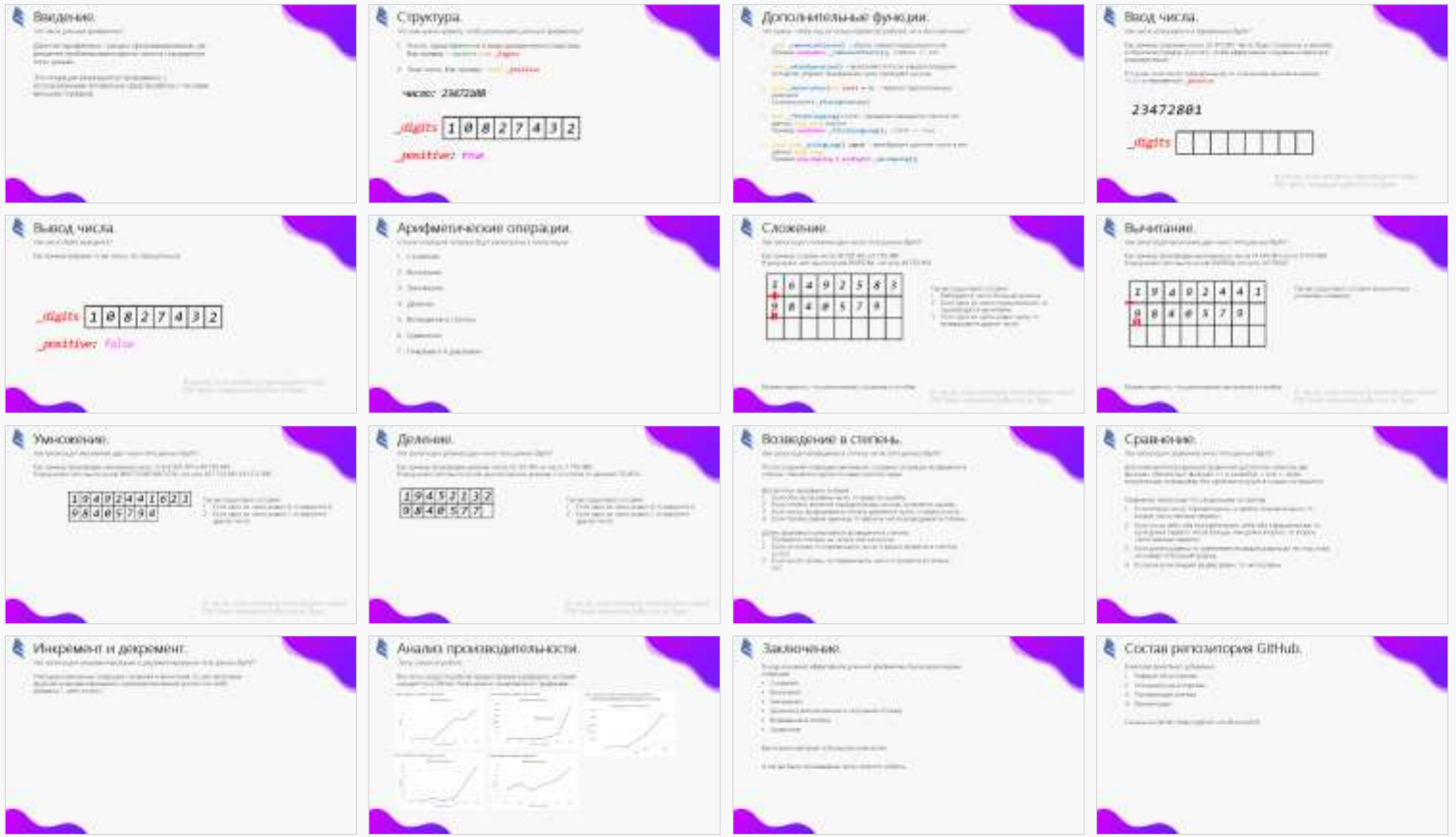
Эффективная длинная арифметика

Студент Ли Дмитрий Сенунович
Б9121-09.03.03пикд
Руководитель доцент ИМКТ Кленин Александ Сергеевич

Навигация.


Введение.
Структура.
Дополнительные функции.
Ввод числа.
Вывод числа.
Арифметические операции.
Сложение.
Вычитание.
Умножение.
Деление.
Возведение в степень.
Сравнение.
Инкремент и декремент.
Анализ производительности.
Заключение.
Состав репозитория GitHub.

Введение.

Что такое длинная арифметика?

Длинная арифметика – раздел программирования, где решается проблема вместимости чисел в стандартные типы данных.

Эти операции реализуются программно, с использованием аппаратных средств работы с числами меньших порядков.

Введение.

Для чего она применяется?

Длинная арифметика применяется в следующих областях:

- Криптография.
- Математическое и финансовое ПО.
- Тема в спортивном программировании.

Структура.

Что нам нужно хранить, чтобы реализовать длинную арифметику?

1. Число, представленное в виде динамического массива.
   Как пример – `vector<int> _digits`

2. Знак числа. Как пример – `bool _positive`

число: 23472108

_positive: true

Дополнительные функции.

Что нужно, чтобы код не только корректно работал, но и был лаконичен?

1. `void _removeLeftZeros()` – убрать (левые) лидирующие нули.
   Пример: `anyNumber._removeLeftZeros(); //00243 >> 243`

2. `void _afterOperation()` – выполняется после каждой операции.
   Алгоритм: убирает лидирующие нули; проверяет на нуль.

3. `void _doCarryOver(int start = 0)` – перенос переполненных разрядов.
   Используется в `_afterOperation()`

4. `bool _fitsInLongLong()` const – проверяет вмещается число в тип данных `long long` или нет.
   Пример: `anyNumber._fitsInLongLong(); //5345 >> true`

5. `long long _asLongLong() const` – преобразует длинное число в тип данных `long long`.
   Пример: `anyLongLong = anyBigInt._asLongLong();`

Дополнительные функции.

Что нужно, чтобы код не только корректно работал, но и был лаконичен?

6. `size_t _lenght() const` – возвращает длина числа.
   Пример: `anyBigInt._lenght()`

7. `bool _isOdd() const` – проверка на нечетное.
   Пример: `anyBigInt._isOdd() //23 >> true`

8. `bool _isEven() const` – проверка на четное.
   Пример: `anyBigInt._isEven() //23 >> false`

9. `bool _isZero() const` – проверка на нуль.
   Пример: `anyBigInt._isZero() //124 >> false`

10. `bool _isOne() const` – проверка на единицу.
    Пример: `anyBigInt._isOne() //1 >> true`

11. `bool _isPositive() const` – проверка на знак.
    Пример: `anyBigInt._isPositive() //-14253 >> false`

Дополнительные функции.

Что нужно, чтобы код не только корректно работал, но и был лаконичен?

12. `BigInt _times10(int times = 1) const` – добавление нулей (для деления).
    Пример: `anyBigInt._times10(3) //15._times10(3) >> 15000`

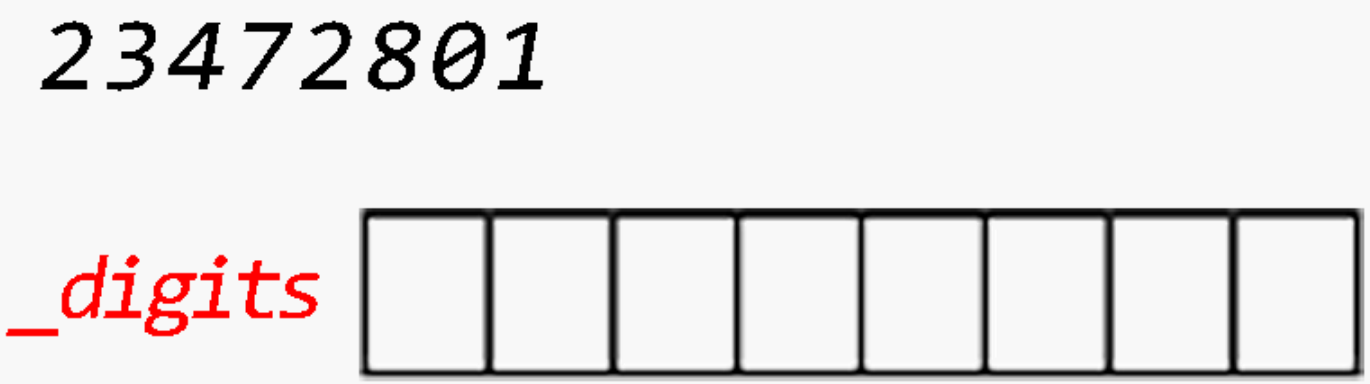
13. `BigInt _absoluteValue() const` – возвращает модуль числа.
    Пример: `anyBigInt._absoluteValue() //-124 >> 124`

Ввод числа.

Как число вписывается в переменную BigInt?

Как пример сохраним число 23 472 801. Число будет сохранено в массиве в обратном порядке. Для того, чтобы эффективнее создавать ячейки для разрядов выше.

В случае, если число отрицательное, то сначала мы внесем значение `false` в переменную `_positive`

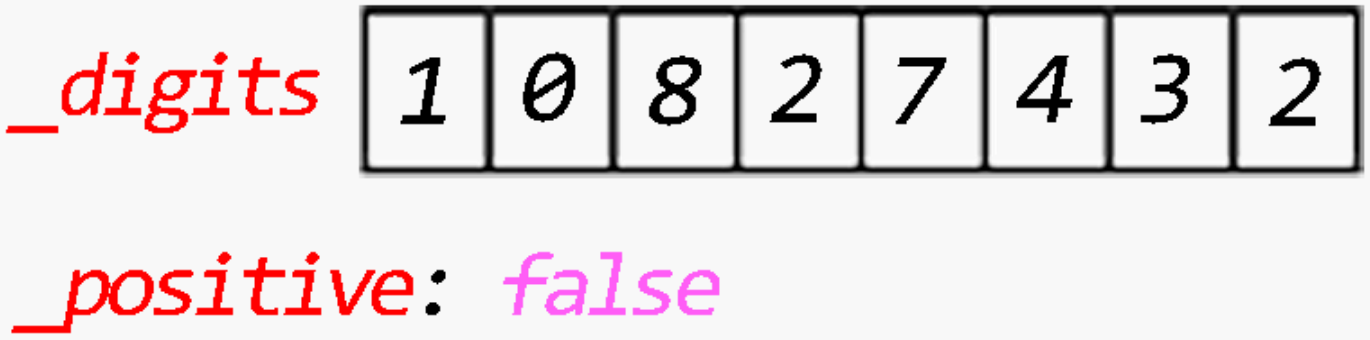


В случае, если просмотр производится через PDF файл, анимация работать не будет.

Вывод числа.

Как число BigInt выводится?

Как пример выведем то же число, но отрицательное.



В случае, если просмотр производится через PDF файл, анимация работать не будет.

Арифметические операции.

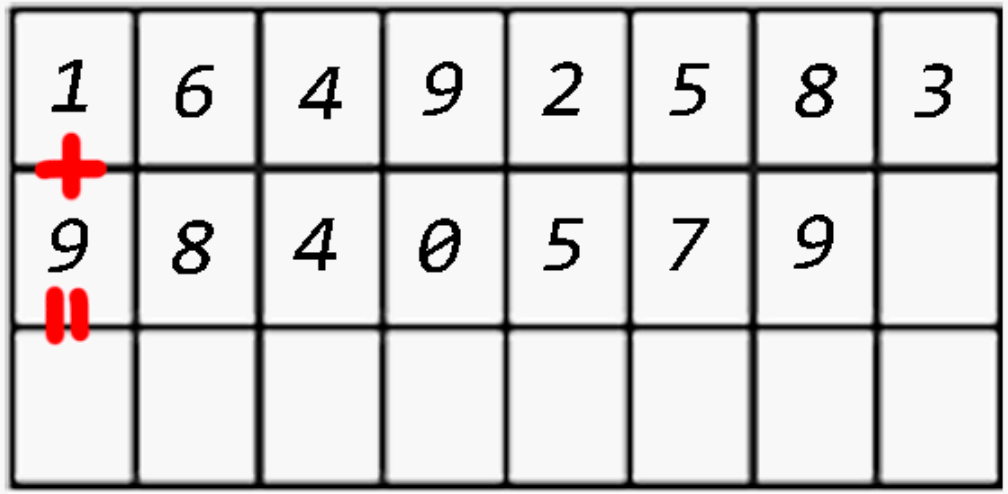
Список операций, которые будут расмотрены в презентации

1. Сложение
2. Вычитание
3. Умножение
4. Деление
5. Возведение в степень
6. Сравнение
7. Инкремент и декремент

Сложение.

Как происходит сложение двух чисел типа данных BigInt?

Как пример сложим числа 38 529 461 и 9 750 489.
В результате чего мы получим 05997284, что есть 48 279 950.



Так же существуют условия:

1. Выбирается число большей длинны.
2. Если одно из чисел отрицательное, то производится вычитание.
3. Если одно из чисел равно нулю, то возвращается другое число

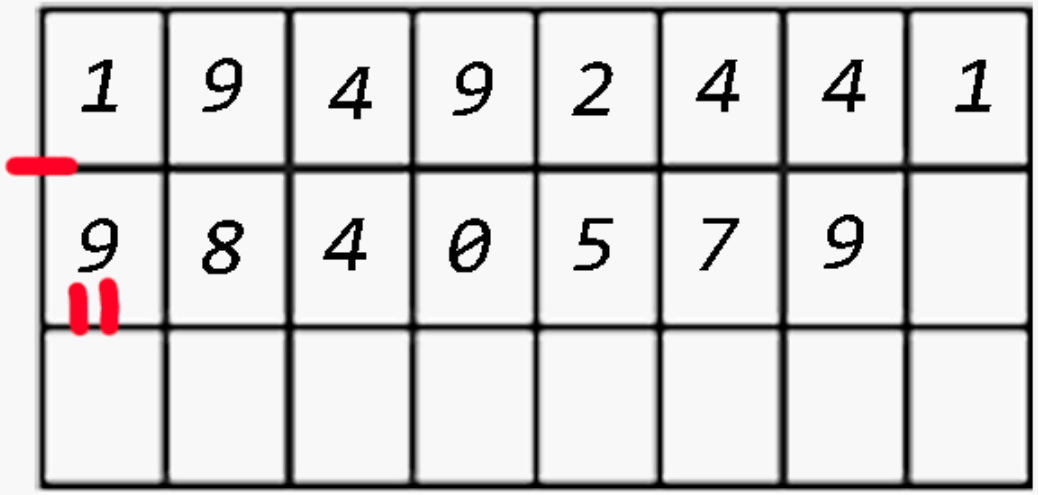
Можем заметить, что реализовано сложение в столбик.

В случае, если просмотр производится через PDF файл, анимация работать не будет.

Вычитание.

Как происходит вычитание двух чисел типа данных BigInt?

Как пример произведем вычитание из числа 14 429 491 число 9 750 489.
В результате чего мы получим 2009764, что есть 4 679 002.



Так же существуют условия аналогичные условиям сложения.

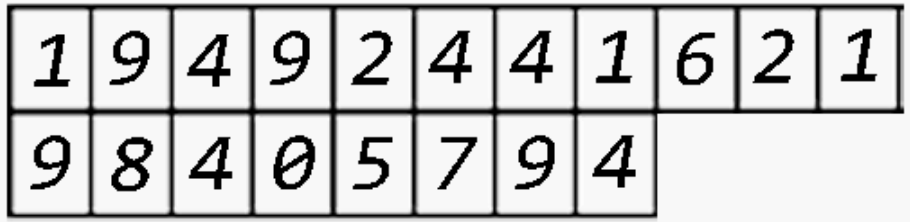
Можем заметить, что реализовано вычитание в столбик.

В случае, если просмотр производится через PDF файл, анимация работать не будет.

Умножение.

Как происходит умножение двух чисел типа данных BigInt?

Как пример произведем умножение чисел 12 614 429 491 и 49 750 489.
В результате чего мы получим 990172336530475726, что есть 627 574 035 633 271 099



Так же существуют условия:

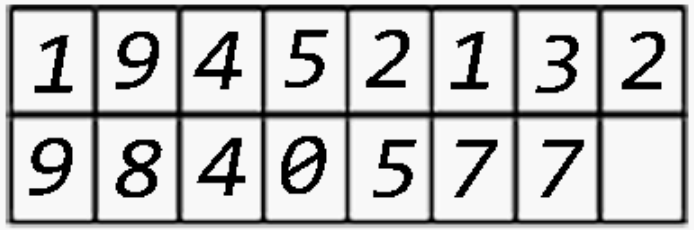
1. Если одно из чисел равно 0, то вернется 0
2. Если одно из чисел равно 1, то вернется другое число

В случае, если просмотр производится через PDF файл, анимация работать не будет.

Деление.

Как происходит деление двух чисел типа данных BigInt?

Как пример произведем деление числа 23 125 491 на число 7 750 489.
В результате чего мы получим целочисленное деление 2 и остаток от деления 7624513



Так же существуют условия:

1. Если одно из чисел равно 0, то вернется 0
2. Если одно из чисел равно 1, то вернется другое число

В случае, если просмотр производится через PDF файл, анимация работать не будет.

Возведение в степень.

Как происходит возведение в степень числа типа данных BigInt?

После создания операции умножения, создание операции возведения в степень становится одной из самых простых задач.

Достаточно проверить условия:

1. Если оба числа равны нулю, то вывести ошибку.
2. Если степень является отрицательным числом, то вывести ошибку.
3. Если число, возводимое в степень равняется нулю, то вернуть нуль.
4. Если степень равна единице, то вернуть число возводимое в степень.

Далее произвести рекусивное возведение в степень:

1. Проверить степень на четное или нечетное.
2. Если нечетная, то перемножить число 3 раза и возвести в степень (n-1)/2.
3. Если число четное, то перемножить число и возвести в степень n/2.

Сравнение.

Как происходит сравнение чисел типа данных BigInt?

Для написания всех функций сравнения достаточно написать две функции: обязательно функцию == и на выбор: > или <, затем встроенными операциями без проблем получится создать оставшиеся.

Сравнение происходит по следующему алгоритму:

1. Если второе число отрицательное, а первое положительное, то второе число меньше первого.
2. Если числа либо оба положительные, либо оба отрицательные, то если длина первого числа больше чем длина второго, то второе число меньше первого.
3. Если длинны равны, то сравнивается каждый разряд до тех пор, пока не найдется больший разряд.
4. В случае если каждый разряд равен, то числа равны.

Инкремент и декремент.

Как происходит инкрементирование и декрементирование типа данных BigInt?

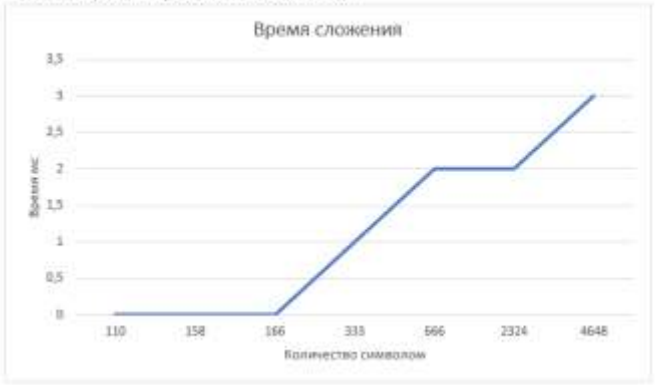
Учитывая написанные операции сложения и вычитания, то для написания функций инкрементирования и декрементирования достаточно либо добавить 1, либо отнять 1.

Анализ производительности.

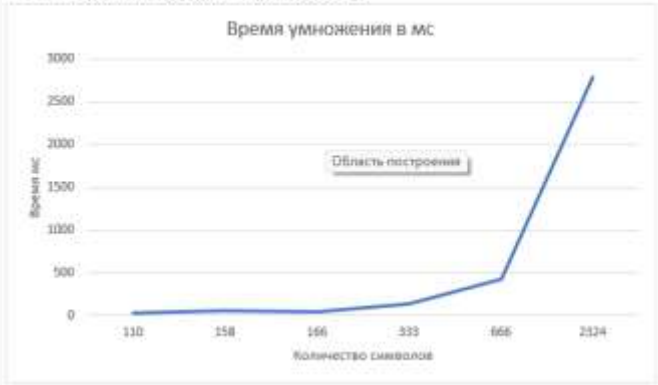
Тесты скорости работы.

Все тесты скорости работы предоставлены в реферате, который находится на GitHub. Ниже можно ознакомиться с графиками:

Тест скорости работы сложения

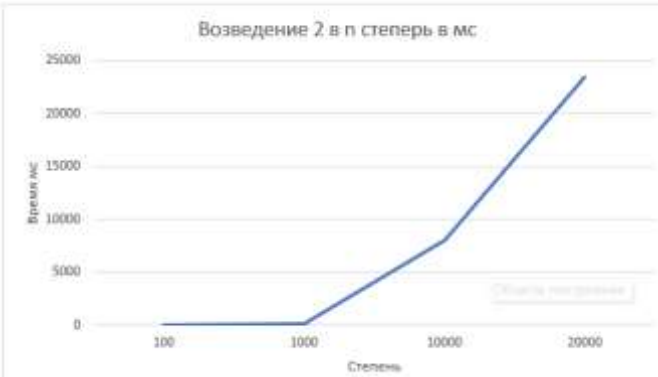


Тест скорости работы умножения

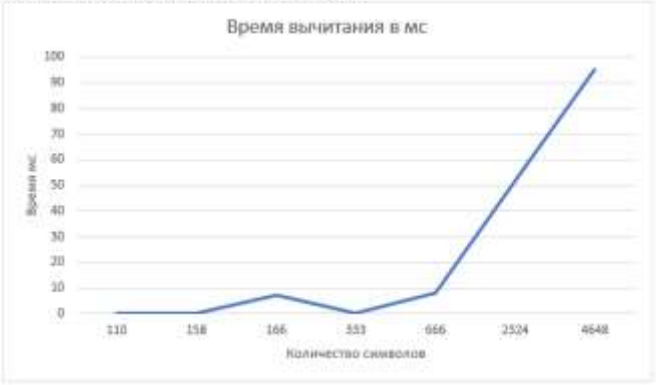


Тест скорости работы возведения в степень

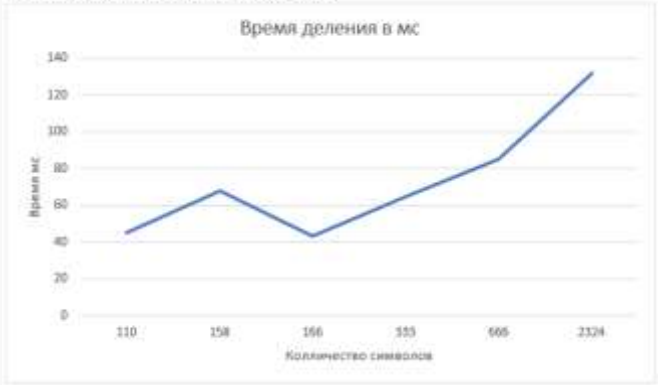
Изначально рассматривается возведение 2 в n степень.



Тест скорости работы вычитания



Тест скорости работы деления



Заключение.

В ходе изучения эффективной длинной арифметики были реализованы операции:

- Сложения
- Вычитания
- Умножения
- Деления (Целочисленное и получение остатка)
- Возведение в степень
- Сравнения

Был изучен материал в большом количестве.

А так же были произведены тесты скорости работы.

Состав репозитория GitHub.

В репозиторий было добавлено:

1. Реферат об алгоритме
2. Исходный код алгоритма
3. Тестирующая система
4. Презентация

Ссылка на GitHub: https://github.com/Elsium/ASD