ДЕПАРТАМЕНТ ОБРАЗОВАНИЯ И НАУКИ

ГОРОДА СЕВАСТОПОЛЯ

ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБЩЕОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ГОРОДА СЕВАСТОПОЛЯ «№54»

**Тема проекта**

**«Длинная арифметика и ее применение в программировании»**

Работа ученика 11 «Б» класса

Михеева Григория Дмитриевича

Руководитель проекта

Дыминский Игорь Анатольевич

Работа допущена к защите «\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2024 г.

Подпись руководителя проекта \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)

Севастополь

2024 г.

# СОДЕРЖАНИЕ

|  |  |  |
| --- | --- | --- |
| ВВЕДЕНИЕ | …………………………… | 3 |
| ГЛАВА 1 Определение длинной арифметики | …………………………… | 4 |
| ГЛАВА 2 Реализация длинной арифметики | …………………………… | 5 |
| ГЛАВА 3 Реализация класса | …………………………… | 6 |
| ЗАКЛЮЧЕНИЕ | …………………………… | 10 |
| СПИСОК ЛИТЕРАТУРЫ | …………………………… | 11 |
| ПРИЛОЖЕНИЕ | …………………………… | 12 |

# ВВЕДЕНИЕ

Длинная арифметика является важной областью в программировании, которая позволяет работать с числами произвольной или очень большой длины, с которыми невозможно работать стандартными средствами языков программирования. Что является препятствием при работе с реальными данными в (*криптографии,* астрономии, геодезии, климатологии и +т.п.) [1] вычислениями с повышенной точностью. Длинная арифметика решает эту проблему, предоставляя инструменты для работы с числами любой длины.

Проблема: работа с большими числами, превышающими стандартные числовые типы данных

Решение: реализация алгоритмов вычислений с использованием длинной арифметики

Что необходимо сделать: создать инструменты работы с большими числами

Средства: язык программирования C++

В данной работе будет выбрано средство, язык программирования и методы реализации программного класса, для выполнения следующих операций над целыми числами превышающими 64 разряда:

1. Сложение
2. Вычитание
3. Умножение
4. Деление
5. Возведение в степень
6. Перевод из десятичной системы счисления в двоичную

# ГЛАВА 1 Определение длинной арифметики

Длинная арифметика (иногда также называемая большой арифметикой) — это математическая область, которая занимается выполнением арифметических операций над очень большими числами, превышающими по размеру границы, установленные стандартными числовыми типами в компьютерных системах[1].

В отличие от обычной арифметики, где числа представлены ограниченным количеством битов (обычно 32 или 64 бита), длинная арифметика позволяет работать с числами, состоящими из сотен, тысяч или даже миллионов цифр. Для хранения и обработки таких чисел используются специальные алгоритмы и структуры данных[2].

Все арифметические операции, которые выполняются обычно на малых числах со стандартными типами данных, могут быть применены и к числам с неограниченным числовым представлением с использованием длинной арифметики[2]. В отличие от стандартной операции над числами, использующей фиксированный размер числа, длинная арифметика предлагает возможность работы с числами любой заданной длины.

Основная идея длинной арифметики заключается в том, что числа представляются в виде последовательностей цифр или символов и далее операции выполняются посимвольно. Для эффективной реализации этой арифметики в программном коде применяются различные методы, такие как длинное сложение, вычитание, умножение и деление чисел.

Важность длинной арифметики проявляется в следующих областях[3]:

* составление кода для процессоров (микроконтроллеров) низкой разрядности;
* криптография;
* астрономия;
* геодезия;
* обработка больших массивов данных.

# ГЛАВА 2 Реализация длинной арифметики

Средством реализации выбран язык программирования C++. Его простота, а также ориентированность на ООП позволяют в полном объёме выполнять поставленные задачи. Средой разработки был выбран Microsoft Visual Studio Enterprise 2022 из-за его удобства во время разработки.

Операции будут выполняться в столбик. Метод реализации был выбран именно такой, чтобы снизить возможное количество ошибок в ходе разработки, а также, чтобы логическая составляющая основных функций класса была наиболее понятной.

Например, найдем сумму двух чисел 397 и 894

Складываем последние цифры чисел: 7+4=11. Так 1 будет находиться в разряде единиц. 1 отвечающую за десятки мы записываем или запоминаем. 9+9+1=19. 9 записываем в разряд десятков, 1 запоминаем и переносим в разряд сотен. 3+8+1=12. 2 записываем в разряд сотен, а 1 записываем в разряд тысяч.

Получаем: 1291.

При реализации данного алгоритма с использованием языка программирования C++, для удобства и более понятной логической составляющей, строки(числа) записываются в обратном порядке. Так операции производятся слева направо, от первого элемента, до последнего, что означает последовательное выполнение операций над каждым разрядом чисел, начиная от единиц.

Одна из основных проблем, с которой сталкиваются при использовании длинной арифметики, — это то, что для представления чисел большой длины требуется значительно большее количество битов памяти. В компьютерной архитектуре количество доступной памяти ограничено, и это может стать проблемой при обработке больших чисел[4].

Для решения этой проблемы можно использовать различные методы представления чисел в длинной арифметике, такие как представление чисел в виде строки символов, массива цифр или битов, или использование специальной структуры данных, называемой «большое целое число» (big integer), которая хранит информацию о числе и операции для его обработки.

Операции в длинной арифметике выполняются методом «цифру-за-цифрой», что означает, что операции выполняются над отдельными цифрами чисел, а не над их целыми значениями. Например, при сложении двух чисел с использованием длинной арифметики каждая цифра в одном числе складывается с соответствующей цифрой во втором числе, а результат сохраняется в новом числе. Этот процесс продолжается, пока все цифры не будут сложены.

Существует несколько подходов к реализации длинной арифметики[4]. Одним из самых распространенных способов является использование массива цифр, где каждая цифра числа представлена одним элементом массива. На основе этого массива реализуются алгоритмы для выполнения арифметических операций, которые распространяются на все цифры чисел. Также нередко используется класс «string» в различных языках программирования. Благодаря распространённости и простоте этого класса, использование его для алгоритмов длинной арифметики не усложняет работу, а код аналогичен тому, который написан для массива.

# ГЛАВА 3 Реализация класса

Для удобства написания кода и его отладки был создан класс «bigint». Был создан раздел «Public» и «Private». В «Public» содержатся все функции, которые доступны пользователю, конструктор класса для основных типов данных, а также перегрузка операторов. В «Private» содержится строковая переменная, хранящая в себе число в виде строки.

Диаграмма класса показана на рисунке 1.

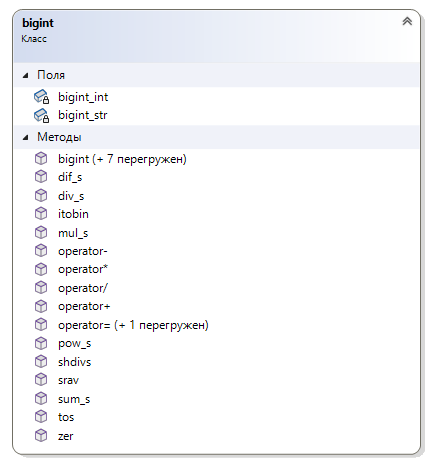


рис 1. Диаграмма класса bigint

Конструкторы класса реализованы для:

1. Строк
2. Символов
3. Чисел
4. Объектов класса

Методы класса показаны на рисунке 2.

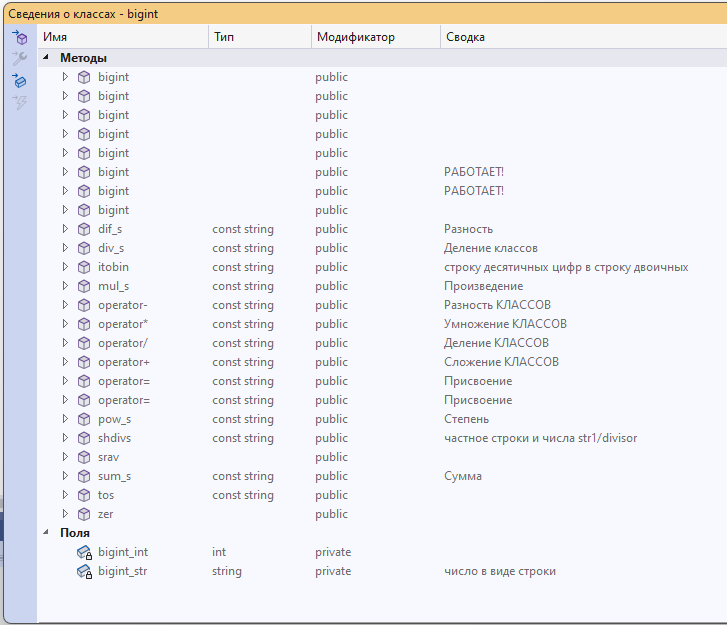


рис 2. Методы класса

Перегружены операторы:

1. «+»
2. «-»
3. «/»
4. «=»
5. « << »
6. « >> »

Для перегрузки операторов для действий над двумя числами типа bigint используется перегрузка типа: const string operator.

Для перегрузки операторов для действий над bigint и string, int используются оператора типа friend.

Перегружена функция «*pow*», для возведения чисел в степень.

Алгоритм выполнения действий с числами. Каждое число в классе представляет собой строку десятичных цифр. Действия производятся по разрядам. Для функции вычитания определяется какое из чисел больше, уменьшаемое или вычитаемое, исходя из этого, программа определяет какой знак будет иметь разность. Во всех функциях кроме вычитания используется булева функция «*srav*», которая возвращает истину, если число *a >= b*, и ложь, если и *b < a* . Для передвижения от одного разряда к другому использован цикл for, с шагом *1*. Цикл ограничен длинной числа, имеющего меньшее количество разрядов. Все функции работают по принципу выполнения действий в столбик. Для удобства реализации такого принципа действий, строки реверсируются. Благодаря этому, логика программы становится более простой, т.к., например нулевой элемент каждой из строк, задающих числа, это меньший разряд. В конце выполнения большинства операций, результатная строка реверсируется обратно.

Разберём выполнение функций более подробно на примере функции сложения «*sum\_s*»:

Проверка на попытку прибавить к числу 0 или наоборот:

if (a == "0") return b;

if (b == "0") return a;

Реверсирование строк, определение меньшей из них по длине:

sz1 = a.size(); sz2 = b.size();

reverse(a.begin(), a.end());

reverse(b.begin(), b.end());

if (sz2 > sz1)

{

swap(a, b);

sz1 = a.size(); sz2 = b.size();

}

Создаётся цикл *for* от *0* до длины наименьшей строки с шагом *1*. Таким образом, при обращении к элементу строки с индексом *i*, будем иметь цифру из разряда на *1* старше. Далее происходит преобразование элемента контейнерного класса, в данном случае приведение символа, как элемента строки, к целому числу. Операция производится для соответствующих элементов двух строк. Два полученных одноразрядных числа суммируются, третьим слагаемым является переменная carry. В конец строковой переменной *sr* записывается остаток от деления суммы двух чисел на *10*. В переменную carry записывается результат целочисленного деления суммы на *10*.

for (int i = 0; i < sz2; i++)

{

n1 = a[i] - '0';

n2 = b[i] - '0';

sum = n1 + n2 + carry;

sr += ((sum % 10) + '0');

carry = sum / 10;

}

После выполнения цикла *for* необходимо убедиться, что все цифры были успешно сложены. Если длины слагаемых различны (выполнение *if*), необходима проверка переменной *carry*, если она не равна *0*, то продолжается сложение, первым слагаемым выступает часть большего числа, которая не учувствовала в сложении, вторым слагаемым – *carry*. В случае, если переменная carry равна *0*, то к *sr* добавляется часть большего числа, которая не учувствовала в сложении. В случае, если длины строк равны, проверяется значение carry, если значение carry не равно *0*, то её значение дописывается к *sr*. После конструкции if else if проверяется чему равна последняя цифра в *sr*, если она равна 0, то дописывается carry т.к. первый значащий разряд не может быть равен *0*. Затем строка *sr* реверсируется, так как сложение происходило на строках, в которых цифры записаны в обратном порядке. Затем функция возвращает как результат строку *sr*.

if (sz1 - sz2 != 0)

{

if (carry != 0)

{

for (size\_t i = sz2; i < sz1; i++)

{

n1 = a[i] - '0';

sum = n1 + carry;

sr += ((sum % 10) + '0');

carry = sum / 10;

}

}

else

{

sr += a.substr(sz2, sz1 - sz2);

}

}

else if (carry != 0) sr += carry + '0';

if (sr[sr.size() - 1] == '0') sr += carry + '0';

reverse(sr.begin(), sr.end());

return sr;

Таким образом, мой класс предоставляет надёжные и точные инструменты для вычисления различных значений большой длинны.

# ЗАКЛЮЧЕНИЕ

В результате проделанной работы реализован класс позволяющий осуществлять вычисления с использованием длинной арифметики, над числами превышающими стандартные числовые типы данных.

В классе реализованные удобные для использования методы для следующих арифметических операций:

1. Сложение
2. Вычитание
3. Умножение
4. Деление
5. Возведение в степень
6. Перевод из десятичной системы счисления в двоичную

Поставленные в работе задачи выполнены в полном объёме, тестирование класса «bigint» показало удовлетворительную работу, как по времени выполнения кода, так и по реализованному функционалу.

# СПИСОК ЛИТЕРАТУРЫ

1. Knuth, D.E. (1998). The Art of Computer Programming, Volume 2: Seminumerical Algorithms (3rd ed.). Addison-Wesley Professional.
2. Montgomery, P.L. (1987). "Modular multiplication without trial division". Mathematics of Computation.
3. Crandall, R.; Pomerance, C. (2005). "Chapter 1 - Arithmetic Algorithms". Prime Numbers: A Computational Perspective. Springer.
4. Электронные средства и системы управления : Материалы докладов Международной научно-практической конференции (13-16 октября 2010 г.). — Томск: В-Спектр, 2011: В 2 ч. — Ч. 2. — 178 с., С.123-129
5. Электронный ресурс «Документация по интегрированной среде разработки Visual Studio». url: https://learn.microsoft.com/ru-ru/visualstudio/ide/?view=vs-2022
6. Пахомов, Борис С/С++ и MS Visual C++ 2012 для начинающих / Борис Пахомов. - М.: "БХВ-Петербург", 2013. - 502 c.

ПРИЛОЖЕНИЕ