ОЛИМПИАДА ШКОЛЬНИКОВ «ШАГ В БУДУЩЕЕ»

НАУЧНО-ОБРАЗОВАТЕЛЬНОЕ СОРЕВНОВАНИЕ «ШАГ В БУДУЩЕЕ, МОСКВА»

1798

регистрационный номер

«Программное обеспечение ЭВМ и информационные технологии»
название факультета

ИУ7
название кафедры

Калькулятор с длинной арифметикой
название работы

Автор: **Мачильский Даниил Дмитриевич**
фамилия, имя, отчество
ГБОУ города Москвы «Бауманская инженерная школа № 1580», 10 класс
наименование учебного заведения, класс

Научный руководитель: **Партанский Михаил Сергеевич**
фамилия, имя, отчество
ГБОУ города Москвы «Бауманская инженерная школа № 1580»
место работы
учитель информатики
звание, должность

_____
подпись научного руководителя

Москва - 2020

Содержание

Аннотация

Цель работы

Целью работы было создать калькулятор, работающий с длинными вещественными числами и в различных системах счисления, а также исследовать, какие способы подходят для этого больше всего:

1) Реализация алгоритма для быстрой обработки длинных вещественных чисел;
2) Реализация вычисления тригонометрических функций, арифметического квадратного корня и других операций;
3) Реализация вычислений в различных системах счисления и перевода вещественных чисел между ними;
4) Написание приложения под Android, работающего на основе этого алгоритма, публикация в Google Play.

Методы

Чтобы выполнить какую-либо операцию над большими числами, нужно хранить их как строки. Сложение, вычитание и умножение должны выполняться столбиком. Деление работает на основе вычитания.

Синус и косинус вычисляются с помощью ряда Тейлора.

Арифметический квадратный корень можно вычислить с помощью бинарного поиска.

Написано на языке Java, так как он лучше всего подходит для создания приложений под Android.

Результаты

Получился уникальный калькулятор, аналогов которому по возможным системам счисления и точности под Android нет. Подобраны максимально эффективные алгоритмы для вычисления. Пользователи, которым нужна длинная арифметика, смогут получить из Google Play калькулятор, не имеющий ограничений на ввод.

Введение

Актуальность

В калькуляторах нет возможности обрабатывать длинные вещественные числа. Данное приложение даёт возможность вычислять значения выражений, которые обычный калькулятор не обрабатывает, также с его помощью можно переводить абсолютно любые числа между системами счисления.

Основная часть

Числа хранятся как строки, это наиболее удобно для реализации алгоритмов. Принцип работы заключается в разбиении выражения на действия. Скобки обрабатываются рекурсивно. Сложение и умножение выполняются столбиком.

Деление работает на основе вычитания: чтобы разделить a на b, число b вычитается из a N раз. N – это и есть ответ. Если ответ не является целым числом, то умножаем делимое на 10 и продолжаем делить, пока не получим частное с заданной точностью. Поскольку получение частного реализовано с помощью вычитания, деление может очень долго работать. Поэтому алгоритм сначала делит 1 на делитель, затем умножает результат на частное, но из-за этого понижается точность, так как значением частного может быть бесконечная десятичная дробь. Для решения данной проблемы необходимо писать функцию для округления или пользоваться обычным делением при работе с маленькими числами.

Все тригонометрические функции можно вычислить, зная значение косинуса и синуса. Синус находится с помощью ряда Тейлора:

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} + \ldots + \frac{(-1)^n \cdot (x^{2n+1})}{(2n+1)!} .$$

Поскольку функция $\sin x$ периодична с периодом 2π, то можно взять остаток от деления x на это число и вычислить $\sin x$. Для этого необходимо реализовать функцию *mod(a, b, base)*, возвращающую остаток от деления a на b в системе счисления с основанием *base*. Рассматривались разные способы реализации:

— с помощью переменных типа *double*: полученная строка переводится в число (числа хранятся как строки), затем выполняются все нужные вычисления, а результат переводится в строку при помощи функции *String.valueOf()*. Таким образом, алгоритм будет работать гораздо быстрее, но потеряется точность.

Второй способ отличается отсутствием типа double, то есть все вычисления

выполняются с использованием длинной арифметики. Его преимущества в точности, а к недостаткам можно отнести маленькую скорость работы. В итоге был выбран второй способ, так как он соответствует цели проекта.

Основное тригонометрическое тождество ($\cos x = \sqrt{1 - \sin^2 x}$) позволяет вычислить только модуль косинуса (если известен только синус), поэтому необходимо использовать другой ряд Тейлора:

$$\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \ ... = \sum_{n=0}^{\infty}(-1)^n \cdot \frac{x^{2n}}{(2n)!} \ .$$

(Реализовано аналогично синусу.)

Вычисление тангенса: $tgx = \frac{\sin x}{\cos x}$.

Вычисление котангенса: $ctgx = \frac{\cos x}{\sin x}$.

Использован алгоритм для быстрого возведения в степень.

Реализована функция *pow*(a, b) = a^b, причём b – целое, так как этот калькулятор не возводит числа в дробную степень.

Алгоритм:

1) Значение выражения a^b не будет определено, если a = 0 и b = 0, так как 0 нельзя возвести в 0 степень.
2) a^b = 1, если и b = 0 и при этом $a \neq 0$.
3) a^b = ($a^{b/2}$)^2, если b чётно.
4) a^b = $a^{b-1} \cdot a$, если b нечётно.

То есть для того, чтобы было проще найти a^b, сначала нужно решить более простую задачу, получив $a^{b/2}$ или a^{b-1}.

Арифметический квадратный корень вычисляется подбором, поэтому самый эффективный алгоритм - двоичный поиск. Если число меньше 1, то нужное значение лежит на отрезке [0; 1], иначе берём промежуток [1; n], где n – число, из которого извлекается корень. r – правая граница этого промежутка, а l – левая. Затем вычисляем среднее арифметическое из r и l: $a = \frac{r+l}{2}$. Если $a^2 > n$, то r принимает значение a, иначе левая граница двигается в точку a. И так до тех пор, пока $|a^2 - n| > E$, где E – минимальный модуль разности полученного числа и

реального значения $\sqrt{x}$, то есть оно вводится в зависимости от необходимой точности.

Реализован алгоритм для перевода вещественных чисел в другие системы счисления. Пусть число *a* в системе счисления с основанием *base1* переводится в систему счисления с основанием *base2*. *base1* и *base2* хранятся в десятичной системе счисления. Чтобы перевести целую часть числа в *base2*, необходимо сначала представить *base2* в системе счисления *base1*, затем записывать в конец строки остатки с результатом от последовательного деления *a* на *base2*. Перевернём полученную строку, чтобы получить целую часть числа в нужной системе счисления. Для перевода дробной части в систему счисления *base2*, нужно взять *base2* в системе счисления *base1*. Затем дробная часть числа умножается на *base2*, целая часть полученного числа – это и есть следующая цифра, которая вычитается из данного числа для следующего умножения. Цикл повторяется, пока нужная точность не будет достигнута (100 цифр после десятичной точки).

Число π можно вычислить различными способами, например:

— Формула Валлиса: $\frac{2}{1}\cdot\frac{2}{3}\cdot\frac{4}{3}\cdot\frac{4}{5}\cdot\frac{6}{5}\cdot\frac{6}{7}\cdot\frac{8}{7}\cdot\frac{8}{9}\ldots=\frac{\pi}{2}$

— Ряд Лейбница: $1-\frac{1}{3}+\frac{1}{5}-\frac{1}{7}+\frac{1}{9}-\ldots=\frac{\pi}{4}$

Опытным путём проверено, что все алгоритмы, работающие с учётом требуемой точности, основанные на этих способах, не эффективны, поэтому выгоднее создать константу, равную π с точностью 100 цифр после запятой.

Для удобства использования написан алгоритм, который при нажатии на кнопку определяет, какую скобку поставить ‘(’ или ‘)’:

- если предыдущий символ ‘(’, то вводится ‘(’ ;
- иначе, если скобочная последовательность правильная, печатается ‘(’, а если нет, то ‘)’.

Аналогично реализована расстановка других видов скобок.

Также разработан алгоритм для корректного подсчёта модуля. Все ‘|’ заменяются на ‘<’ или ‘>’:

- если предыдущий символ - цифра или закрывающаяся скобка, то ‘|’ заменяется на ‘>’;
- иначе на ‘<’. Если его нет, то вводится ‘<’.

Использование

Ввод осуществляется при помощи экранной клавиатуры. Для расстановки любых скобок используется только одна кнопка (алгоритм сам определит, какую скобку напечатать).

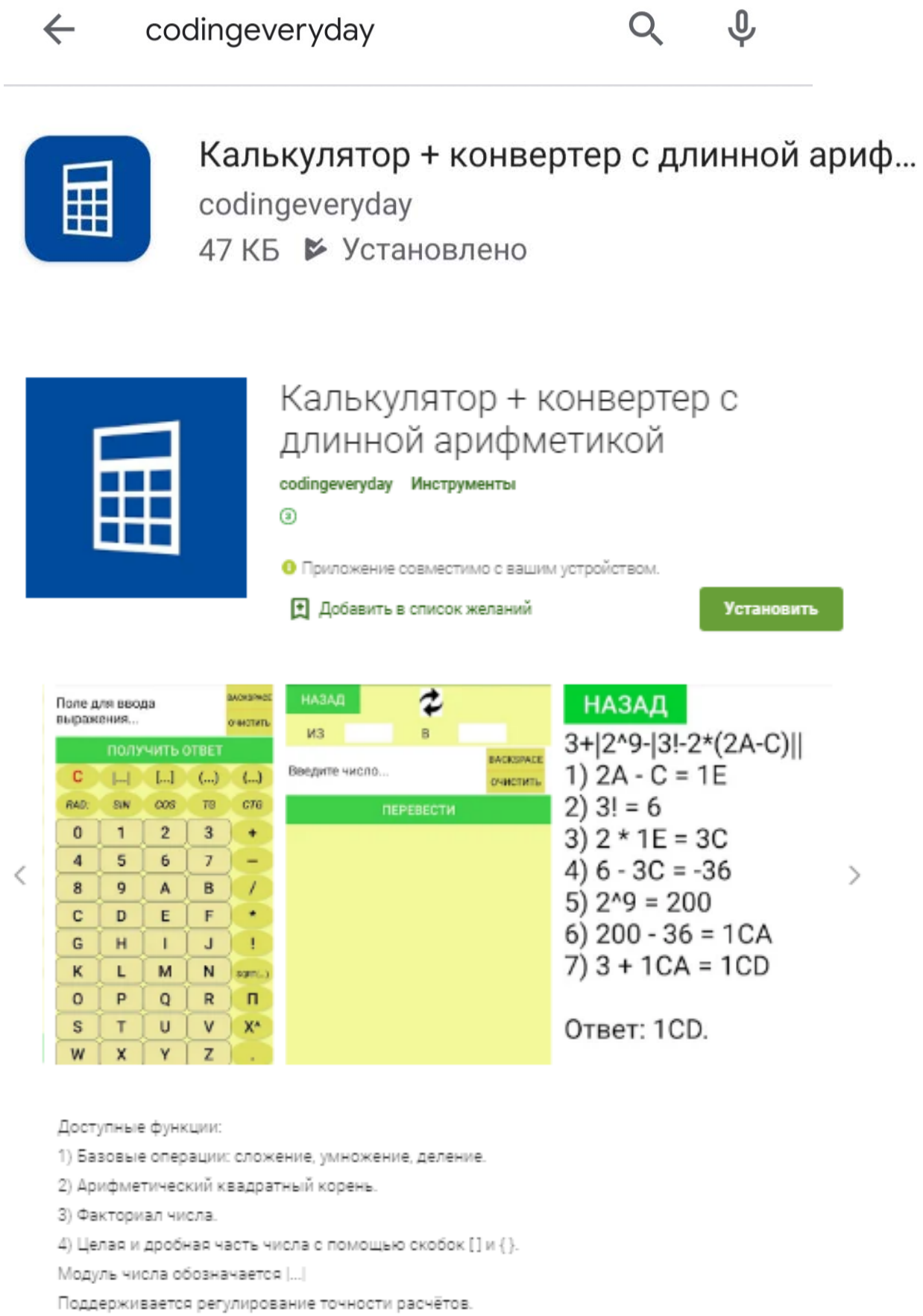
Обозначения:

- Модуль: ‘|’
- Целая часть числа: [ ].
- Дробная часть числа: { }.
- Возведение в степень: ‘^’.
- Получение факториала числа: ‘!’.

Есть защита от некорректного ввода, деления на 0, отрицательного числа под корнем и неправильных скобочных последовательностей.

Заключение

Получился калькулятор (Приложение 1), предназначенный для вычисления значений выражений, с которыми обычный калькулятор не справляется. Для этого подобраны наиболее удобные и рациональные алгоритмы. Пользователи могут скачать данное приложение в Google Play по ссылке: https://play.google.com/store/apps/details?id=com.calculator.calcapp



Доступные функции:

1) Базовые операции: сложение, умножение, деление.

2) Арифметический квадратный корень.

3) Факториал числа.

4) Целая и дробная часть числа с помощью скобок [ ] и { }.

Модуль числа обозначается |...|

Поддерживается регулирование точности расчётов.