← SmartCalc_v1.0

Отзы

К сожалению, вы не можете участвовать в этом проекте

✓ Подписаться

Задача

СмартКалькулятор v1.0

> При начале работы над проектом просим вас постараться отследить время работы над проектом. По завершении работы над проектом просим вас ответить на два вопроса в этом опросе

Реализация SmartCalc v1.0.

Русскоязычную версию задания можно найти в репозитории.

Содержание

Преамбула

Планета Земля, США, Калифорния, Купертино, 10260 Бэндли Драйв, 5 августа 1983 года.

Приглушенные шаги быстро приблизились к комнате. Было странно, как будто человек шел босиком. Внезапная остановка у двери, какой-то невнятный диалог, видимо, с прохожим... дверь открывается, и он входит. На ходу осматривая комнату, он быстро выхватывает из толпы, которая что-то обсуждала, нескольких человек и жестом приглашает их следовать за собой.

Он говорит на ходу:

-- Энди, как дела с приветствием?

-- Ну, презентация еще не скоро, так что я уверен, что она будет готова вовремя.

-- Хорошо, Билл, мне нужна альфа-версия MacPaint на следующей неделе, я хочу показать ее Джону Скалли.

-- Но его разработка началась всего две недели назад.

-- Вот почему я даю вам еще одну неделю! Где Крис? Кто-нибудь видел его?

Крис Эспиноса был за дверью в своем кабинете и услышал, что Джобс ищет его. Пятая версия калькулятора для Macintosh уже была написана, и у него было ощущение, что это только начало. Вот только его боевой дух вот-вот иссякнет! Внезапно он услышал, как кто-то сказал *: «Кажется, он в своем кабинете»* , а затем быстрые шаги толпы людей устремились в сторону Криса. Страх овладел им на мгновение, но он быстро взял себя в руки и пришел в себя. Ведь это был всего лишь калькулятор! Дверь открылась, и Джобс вошел в комнату, а остальные остались за кабинетом.

-- Крис, как поживает наш калькулятор?

-- Вот смотрите, я все исправил по вашим комментариям и реализовал некоторые свои идеи.

-- Хорошо, так может ли он считать выражения с учетом порядка операций? И если вам нужно сделать график, он тоже может это сделать? А как насчет кредитов или депозитов, может ли пользователь получить от вашего калькулятора какой-то функционал, который поможет ему сэкономить доллары? Да и интерфейс, прямо скажем, не очень.

Крис понял, что калькулятор не может делать ничего из этого и полностью вышел из строя. Джобс понял это без слов, немного развеселил и ушел. Ты наблюдал за всем со стороны, и у тебя было ощущение, что ты должен ему помочь.
Вы подождали, пока все уйдут, а затем подошли к нему, чтобы предложить свою помощь:

- Слушай, Крис, у меня есть свободное время по вечерам. Я могу помочь вам с калькулятором, потому что я вижу, что вы очень расстроены. Переделывать одно и то же тысячу раз не очень приятно, хотя это большая часть нашей работы как программистов.

Он невероятно обрадовался и сразу же ввел вас в курс дела. Что ж, вы спасли его работу, осталось реализовать калькулятор!

Глава I

Введение

В этом проекте вы будете реализовывать расширенную версию обычного калькулятора, который можно найти в стандартных приложениях каждой операционной системы на языке программирования Си с использованием структурного программирования. Помимо основных арифметических операций, таких как сложение/вычитание и умножение/деление, необходимо дополнить калькулятор возможностью вычисления арифметических выражений в порядке следования, а также некоторыми математическими функциями (синус, косинус, логарифм и т. .

Помимо вычисления выражений, он также должен поддерживать использование переменной *x* и построение графика соответствующей функции.

Что касается других улучшений, вы можете рассмотреть кредитный и депозитный калькулятор.

Глава II

Информация

Историческая справка

История развития калькулятора начинается в 17 веке, а первые прототипы этого устройства относятся к 6 веку до нашей эры. Само слово «калькулятор» происходит от латинского «calculo», что означает «считать», «вычислять».

Но если мы углубимся в изучение этимологии этого слова, то увидим, что изначально следует говорить о слове «исчисление», которое переводится как «камешек». Ведь - изначально именно камешки использовались как атрибут для счета.

История вычислительных машин, в том числе калькуляторов, традиционно начинается с арифмометра Паскаля, изобретенного в 1643 г. Блезом Паскалем, и ступенчатого счётчика Лейбница, изобретенного в 1673 г. немецким математиком Готфридом Вильгельмом Лейбницем.

Счетная машина представляла собой коробку со связанными шестернями, которые вращались специальными колесами, каждое из которых соответствовало одной десятичной цифре.

Когда одно из колес делало десятый оборот, происходило смещение следующей передачи на одну позицию, что увеличивало цифру числа. После выполнения математических операций результат отображался в окнах над колесами.

Двадцать лет спустя Лейбниц создал свой вариант калькулятора, принцип работы которого был тот же, что и у арифмометра Паскаля — шестеренки и колеса. Однако калькулятор Лейбница имел подвижную часть, ставшую прообразом подвижных кареток будущих настольных калькуляторов, и ручку, вращавшую ступенчатое колесо, которое позже заменили цилиндром. Эти дополнения сделали повторяющиеся операции, такие как умножение и деление, намного быстрее.

Хотя калькулятор Лейбница несколько упростил процесс счета, он дал толчок другим изобретателям - подвижная часть и цилиндр калькулятора Лейбница использовались в счетных машинах вплоть до середины XX века.

Затем в 1957 году Casio выпустила один из первых массовых калькуляторов — 14-A. Он выполнял четыре арифметические операции над 14-значными десятичными числами. Машина использовала реле, весила 140 кг и выглядела как стол с вычислительным блоком, клавиатурой и дисплеем, потребляя 300 Вт.

Однако калькуляторы быстро развивались и становились более сложными, как и другие вычислительные машины. В 1965 году Wang Laboratories выпустила калькулятор Wang LOCI-2, который мог вычислять логарифмы. Casio выпустила первый калькулятор с функцией памяти — «Casio 001» (37x48x25 см, вес 17 кг), а Olivetti выпустила «Programma 101», первый калькулятор, который мог сохранять программу и многократно выполнять по ней вычисления.

Малогабаритные настольные и карманные калькуляторы начали выпускать с 1970 года, после появления интегральных схем, резко уменьшивших габариты, вес и энергопотребление электронных устройств. В 1970 году Sharp и Canon начали продавать калькуляторы, которые можно было держать в руке (весом около 800 граммов). В 1971 году появился первый по-настоящему карманный (131x77x37 мм) калькулятор Bomwar 901B. Он выполнял 4 арифметических операции, имел светодиодный дисплей и стоил 240 долларов. В 1973 году на рынке впервые появился калькулятор Sharp EL-805 с ЖК-дисплеем. В 1979 году компания Hewlett Packard выпустила первый программируемый калькулятор с буквенно-цифровым индикатором и возможностью подключения дополнительных модулей — ОЗУ, ПЗУ, считывателя штрих-кодов, кассет, дискет, принтеров и т. д.

Польская запись и обратная польская запись

Несмотря на абсолютное удобство работы с классической формой записи выражений, когда дело доходит до написания программ, запросы чаще всего формируются в формате `what to do -> what data to perform operations with` . Итак, примерно в 1920 году польский логик Ян Лукасевич изобрел префиксную запись (позднее также называемую польской записью или нормальной польской записью), чтобы упростить логику высказываний.

Рассмотрим способы написания выражений:

Выражение, записанное в общепринятой инфиксной записи:

> 2/(3+2)*5

Выражение, записанное в польской префиксной нотации:

> * (/ 2 (+ 3 2)) 5

Обычно скобки между операциями с одинаковым приоритетом не нужны, и окончательное выражение выглядит так:

> * / 2 (+ 3 2) 5

Польская нотация широко используется в области вычислений, в частности она используется во многих стек-ориентированных языках программирования, таких как PostScript, и долгое время была основой для многих вычислительных машин (калькуляторов).

В середине 1950-х годов австралийский философ и ученый-компьютерщик Чарльз Хэмблин разработал обратную польскую нотацию (RPN). Работа Хэмблина была представлена на конференции в июне 1957 года и опубликована в 1957 и 1962 годах.

Первыми компьютерами, поддерживающими обратную польскую нотацию, были KDF9 от English Electric Company, анонсированный в 1960 году и выпущенный в 1963 году, и американский Burroughs B5000, анонсированный в 1961 году и выпущенный в 1963 году.

Фриден использовал RPN в настольных калькуляторах и представил EC-130 в июне 1964 года. В 1968 году инженеры Hewlett-Packard разработали настольный калькулятор 9100A с поддержкой RPN. Этот калькулятор сделал обратную польскую запись популярной среди ученых и инженеров, хотя в ранней рекламе 9100A RPN не упоминался. В 1972 году HP-35 с поддержкой RPN стал первым научным карманным калькулятором.

РПН применялся в советском инженерном вычислителе Б3-19М (совместная разработка с ГДР) выпуска 1976 года. Все программируемые микрокалькуляторы, выпускавшиеся в СССР до конца 1980-х годов, кроме "Электроника МК-85" и "Электроника МК-85". 90", использовалась RPN - она была проще в реализации и позволяла использовать меньшее количество команд при программировании по сравнению с обычной алгебраической записью (объем памяти программ всегда был критическим ресурсом в моделях того времени). РПН использовался в отечественных программируемых калькуляторах "Электроника МК-152" и "Электроника МК-161", обеспечивая их совместимость с программами, написанными для советских калькуляторов.

Обратная польская нотация обладала всеми преимуществами прародителя, поскольку устраняла необходимость в круглых скобках, что позволяло уменьшить объем выражений. Это привело к уменьшению количества команд при написании компьютерных программ. Вот почему иногда обратную польскую запись называют обратной записью без скобок.

Выражение, записанное в обратной польской (без скобок) нотации:

> 2 3 2 + / 5 *

Алгоритм Дейкстры

Эдсгер Дейкстра изобрел алгоритм преобразования выражений (включая функции) из

> Прежде чем мы рассмотрим сам алгоритм, сделаем небольшое замечание и представим токен. Маркер — это простейшая единица морфологического

> разбора выражения. Итак, выражение 2/(3+2)*5 (пробелы добавлены для удобства восприятия и не несут никакого смыслового смысла), разбитое на лексемы, будет выглядеть так [2], [/], [(] , [3], [+], [2], [)], [*], [5], где [2, 3, 5] — числовые токены, [/, (, +,), *] — токены-операции.

Алгоритм Shunting-yard основан на стеке. В преобразовании участвуют две текстовые переменные: входная и выходная строка. Процесс преобразования использует стек для хранения операций, еще не добавленных в выходную строку. Программа преобразования последовательно считывает каждый токен из входной строки и на каждом шаге выполняет какие-то действия в зависимости от того, какие токены были прочитаны.

Реализация алгоритма

Пока во входной строке есть необработанные токены, читаем следующий токен:

Если токен:

- Число – поместить его в очередь вывода
- Функция или левая скобка — поместите ее в стек
- Разделитель аргументов функции (например, запятая):
  - Перемещайте операторы из стека в очередь вывода до тех пор, пока маркер наверху стека не станет левой скобкой. Если в стеке нет левой скобки, в выражении есть ошибка.
- Оператор (O1):
  - Хотя наверху стека находится токен-оператор O2, который имеет больший приоритет, чем O1, или они имеют одинаковый приоритет, а O1 является левоассоциативным:
  - Вытолкнуть O2 из стека в выходную очередь
  - Поместите O1 в стек
- Правая скобка:
  - Пока токен наверху стека не является левой скобкой, поместите операторы-токены из стека в выходную очередь.
  - Извлеките левую скобку из стека и выбросьте ее.
  - Если в верхней части стека есть токен функции, то вытолкните функцию из стека в очередь вывода.
  - Если стек закончился до того, как была прочитана левая скобка, в выражении есть ошибка.

Если во входной строке больше не осталось токенов:

- Пока в стеке есть операторы:
  - Если наверху стека стоит скобка - в выражении ошибка.
  - Вставьте оператор из стека в выходную очередь

Конец.

Глава III

Часть 1. Реализация SmartCalc v1.0

Программу SmartCalc v1.0 необходимо внедрить:

- Программа должна быть написана на языке C стандарта C11 с использованием компилятора gcc. Вы можете использовать любые дополнительные библиотеки и модули QT
- Код программы должен находиться в папке src
- Программа должна быть собрана с помощью Makefile, который содержит стандартный набор целей для GNU-программ: all, install, uninstall, clean, dvi, dist, test, gcov_report. Каталог установки может быть произвольным, кроме каталога сборки
- Программа должна быть разработана в соответствии с принципами структурного программирования.
- При написании кода необходимо следовать стилю Google
- Подготовьте полное покрытие модулей, связанных с вычислением выражений с помощью юнит-тестов, используя библиотеку Check.
- Реализация графического интерфейса на основе любой библиотеки графического интерфейса с API для C89/C99/C11
  Для Linux: GTK+, CEF, Qt
  Для Mac: GTK+, Nuklear, raygui, microui, libagar, libui, IUP, LCUI, CEF, Qt
- В программу можно вводить как целые, так и действительные числа с точкой. При желании вы можете предоставить ввод чисел в экспоненциальной записи
- Вычисление должно быть выполнено после того, как вы завершите ввод расчетного выражения и нажмете = символ .
- Вычисление произвольных арифметических выражений в квадратных скобках в инфиксной записи
- Вычислять произвольные скобочные арифметические выражения в инфиксной записи с подстановкой значения переменной *x* в виде числа
- Построение графика функции, заданной выражением в инфиксной записи с переменной *x* (с осями координат, отметкой используемого масштаба и адаптивной сеткой)
  - Необязательно предоставлять пользователю возможность изменения масштаба
- Домен и кодовый домен функции ограничены как минимум числами от -1000000 до 1000000
  - Для построения графика функции необходимо дополнительно указать отображаемый домен и кодовый домен
- Проверяемая точность дробной части не менее 7 знаков после запятой
- Пользователи должны иметь возможность вводить до 255 символов.
- Арифметические выражения в скобках в инфиксной нотации должны поддерживать следующие арифметические операции и математические функции:
  - **Арифметические операторы** :

Имя оператора	Инфиксная нотация (классическая)	Префиксная запись (польская запись)	Постфиксная запись (обратная польская запись)
Кронштейны	(a + b)	(+ab)	ab +

Имя оператора	Инфиксная нотация (классическая)	Префиксная запись (польская запись)	Постфиксная запись (обратная польская запись)
Добавление	a + b	+ ab	ab +
вычитание	a - b	- ab	ab -
Умножение	a * b	* ab	ab *
Разделение	a/b	/ ab	ab \
Власть	a ^ b	^ ab	ab ^
Модуль	mod b	mod ab	ab mod
Унарный плюс	+a	+a	+
Унарный минус	-a	-a	a-

> Обратите внимание, что оператор умножения содержит обязательный знак `*`. Обработка выражения с пропущенным `*` знаком не является обязательной и остается на усмотрение разработчика.

- **Функции** :

Описание функции	Функция
Вычисляет косинус	потому что (x)
Вычисляет синус	грех (x)
Вычисляет тангенс	загар(x)
Вычисляет арккосинус	акос(x)
Вычисляет арксинус	асин(x)
Вычисляет арктангенс	атан (x)
Вычисляет квадратный корень	квадрат (x)
Вычисляет натуральный логарифм	пер(x)
Вычисляет десятичный логарифм	журнал (x)

Часть 2. Бонус. Кредитный калькулятор

Предусмотреть специальный режим «кредитный калькулятор» (можно взять для примера banki.ru и calcus.ru):

- Ввод: общая сумма кредита, срок, процентная ставка, тип (аннуитетный, дифференцированный)
- Вывод: ежемесячный платеж, переплата по кредиту, общий платеж

Часть 3. Бонус. Депозитный калькулятор

Предусмотреть специальный режим "калькулятор доходности вклада" (можно взять для примера banki.ru и calcus.ru):

- Ввод: сумма вклада, срок вклада, процентная ставка, налоговая ставка, периодичность выплат, капитализация процентов, список пополнений, список частичных изъятий
- Вывод: начисленные проценты, сумма налога, сумма депозита к концу срока

💡 Нажмите здесь **, чтобы оставить отзыв о проекте** . Команда Pedago действительно старается сделать ваш образовательный опыт лучше.