К сожалению, вы не можете участвовать в этом проекте

Подписаться

Задача

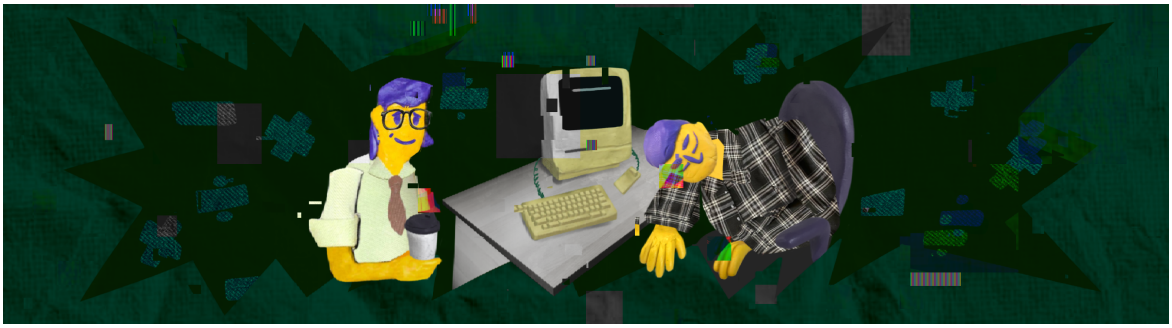
СмартКалькулятор v2.0

Реализация SmartCalc v2.0.

Русскоязычную версию задания можно найти в репозитории.

Содержание

Глава I



Планета Земля, США, Калифорния, где-то в Купертино, 20 августа 1983 года.

Крис поставил рядом с тобой на стол бумажный стаканчик с кофе, чтобы ты проснулась.

-- Чистый черный американо, я слышал, он тебе нравится. Это поможет тебе проснуться.

- О, да, мне это очень нужно, спасибо. Где были мы? - спросила ты, попивая свой горячий кофе.

-- Вы закончили базовую логику калькулятора?

- Ах, это все еще в процессе. Встреча со Стивом будет завтра?

-- Это верно. Если мы сможем показать ему некоторые основные арифметические операции с реорганизованной структурой программы, я думаю, ему это понравится.

Уже больше недели вы по вечерам помогаете Крису Эспинозе переписать объектно-ориентированный калькулятор. Его последняя версия в стандартном структурированном подходе оказалась недостаточно гибкой и расширяемой для бурлящего идеями Стива Джобса. Итак, вы и Крис решили попробовать новую парадигму объектно-ориентированного программирования на молодом языке C++, чтобы решить эти проблемы. Конечно, изучение новых технологий вызывало некоторые... трудности, но есть надежда, что полностью переписывать калькулятор в восьмой раз не придется.

- Ну, тогда давай побыстрее.

-- Кстати, на днях пересекся с разработчиком из Норвегии: Торкве или Трюкве, не помню.
Так или иначе, он рассказал мне об одной схеме организации данных и логики приложения, которая позволяет очень гибко и быстро менять некоторые компоненты. Например, мы можем полностью отделить интерфейс от остального кода, понимаете? А если Стиву что-то в интерфейсе снова не понравится, это можно будет быстро и безопасно изменить. Нам даже не нужно было бы переписывать тесты для остальной логики.

- Похоже, это то, что нам нужно! Я весь во внимании.

Введение

В этом проекте вам потребуется реализовать расширенную версию стандартного калькулятора на C++ в парадигме объектно-ориентированного программирования, реализующую те же функции, что и ранее разработанное приложение в проекте SmartCalc v1.0. 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

Обратите внимание, что для реализации калькулятора следует использовать *алгоритм Дейкстры* для перевода выражений в *обратную польскую запись* . Вы можете найти всю необходимую информацию в описании проекта SmartCalc v1.0, чтобы освежить свои знания.

Шаблон MVC

Шаблон Model-View-Controller (MVC) представляет собой схему разделения модулей приложения на три макрокомпонента: модель, содержащую бизнес-логику, представление, представляющее собой форму пользовательского интерфейса для взаимодействия с программой, и контроллер, изменяющий модель по действию пользователя.

Концепция MVC была представлена Трюгве Реенскаугом в 1978 году, который работал над языком программирования Smalltalk в Xerox PARC. Позже Стив Бербек реализовал шаблон в Smalltalk-80. Окончательная версия концепции MVC была опубликована в журнале Technology Object в 1988 году. Шаблон MVC впоследствии эволюционировал, породив такие варианты, как HMVC, MVA, MVVM.

Основная потребность в этом шаблоне проистекает из желания разработчиков отделить бизнес-логику программы от представлений, что упрощает замену представлений и повторное использование логики, реализованной однажды в других средах. Модель, отделенная от представления, и контроллер для взаимодействия с ним позволяют более эффективно повторно использовать или модифицировать уже написанный код.

Модель хранит и получает доступ к основным данным, выполняет операции по запросам, определенным бизнес-логикой программы, то есть отвечает за часть программы, отвечающую за все алгоритмы и обработку информации. Эти модели, модифицированные контроллером, влияют на отображение информации в пользовательском интерфейсе. Моделью в этой программе должна быть библиотека классов, выполняющая вычисления. Эта библиотека должна предоставлять все необходимые классы и методы для их выполнения. И это бизнес-логика программы, потому что она предоставляет средства для решения проблемы.

Контроллер — это тонкий макрокомпонент, выполняющий модификации модели. Он используется для генерации запросов к нему. В коде это выглядит неким «фасадом» модели, то есть набором методов, которые уже работают непосредственно с моделью. Он называется тонким, потому что идеальный контроллер не содержит никакой дополнительной логики, кроме вызова одного или нескольких методов модели. Контроллер служит связующим звеном между интерфейсом и моделью. Это позволяет полностью отделить модель от дисплея. Это разделение полезно тем, что оно позволяет коду представления ничего не знать о коде модели и обращаться только к контроллеру, чей интерфейс предоставляемых функций, вероятно, не будет сильно изменен. Модель же может претерпевать существенные изменения, и если «перейти» на другие алгоритмы, технологии, или даже языки программирования в модели, нужно будет изменить только небольшой участок кода в контроллере, непосредственно связанный с моделью. В противном случае, вероятно, пришлось бы переписывать значительную часть кода интерфейса, так как это сильно зависело бы от реализации модели. Итак, при взаимодействии с интерфейсом пользователь вызывает методы контроллера, модифицирующие модель.

Представление включает в себя весь код, связанный с интерфейсом программы. Код идеального интерфейса не должен содержать никакой бизнес-логики. Он лишь

представляет собой форму для взаимодействия с пользователем.

Глава III

Часть 1. Реализация SmartCalc v2.0

Вам необходимо реализовать SmartCalc v2.0:

- Программа должна быть разработана на языке C++ стандарта C++17.
- Код программы должен находиться в папке src
- При написании кода необходимо следовать стилю Google
- Классы должны быть реализованы в `s21` пространстве имен
- Подготовьте полное покрытие модулей вычисления выражений модульными тестами с использованием библиотеки GTest.
- Программа должна быть собрана с помощью Makefile, который содержит стандартный набор целей для GNU-программ: все, установить, удалить, очистить, dvi, dist, тесты. Каталог установки может быть произвольным
- Реализация графического интерфейса, основанная на любой библиотеке графического интерфейса с API для C++17:
  - Для Linux: GTK+, CEF, Qt, JUCE
  - Для Mac: GTK+, CEF, Qt, JUCE, SFML, Nanogui, Nngui
- Программа должна быть реализована с использованием паттерна MVC, а также:
  - в коде представления не должно быть кода бизнес-логики
  - в контроллере и модели не должно быть кода интерфейса
  - контроллеры должны быть тонкими
- В программу можно вводить как целые, так и действительные числа с точкой. Вы должны обеспечить 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 + б)	(+аб)	аб +
Добавление	a + б	+ аб	аб +
вычитание	a - б	- аб	аб -
Умножение	a * б	* аб	аб *
Разделение	a/б	/ аб	аб \
Власть	a ^ б	^ аб	аб ^
Модуль	мод б	мод аб	аб мод
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.