Семинар №1

Тема семинара: Обзор общих подходов к тестированию, изучение архитектуры программного комплекса «Калькулятор».

Цель семинара: Постановка и обсуждение задачи, используемой в качестве сквозного примера в ходе семинарских занятий. Обсуждение общей архитектуры системы, разбиение на модули.

План семинара:

1. Введение

Наши занятия будут направлены на приобретение практических навыков в области тестирования и верификации программного обеспечения (ПО). На протяжении всего семестра мы будем изучать данный курс на одном сквозном примере – на программном продукте “Калькулятор”.

Предположим, что мы являемся частью коллектива разработчиков, которому поступил заказ на разработку программной системы «Калькулятор» (в дальнейшем просто “Система”). Предположим также, что другая часть коллектива уже сформировала функциональные требования, архитектуру и написала программный код системы. Таким образом, на нас ложится участок жизненного цикла системы по тестированию и проверке требований.

2. Система “Калькулятор”

2.1. Общее описание

Основная цель Системы – вычисление математических выражений с корректной структурой. Формально данное предложение раскрыто в приложении к семинару, а здесь дадим некоторые комментарии. В самом простом случае, будем считать корректными следующие выражения:

1
1+1
(1+1)
(1+1)*2

и т.д., то есть, выражения, корректные в математическом смысле.

Однако, множество вычисляемых калькулятором выражений все же несколько “меньше” чем просто корректные математические выражения. Это связано с некоторыми математическими операциями и дробными числами, корректность обработки которых сложно протестировать: не будем забывать, что Калькулятор - прежде всего учебный пример.

2.2. Требования к Системе

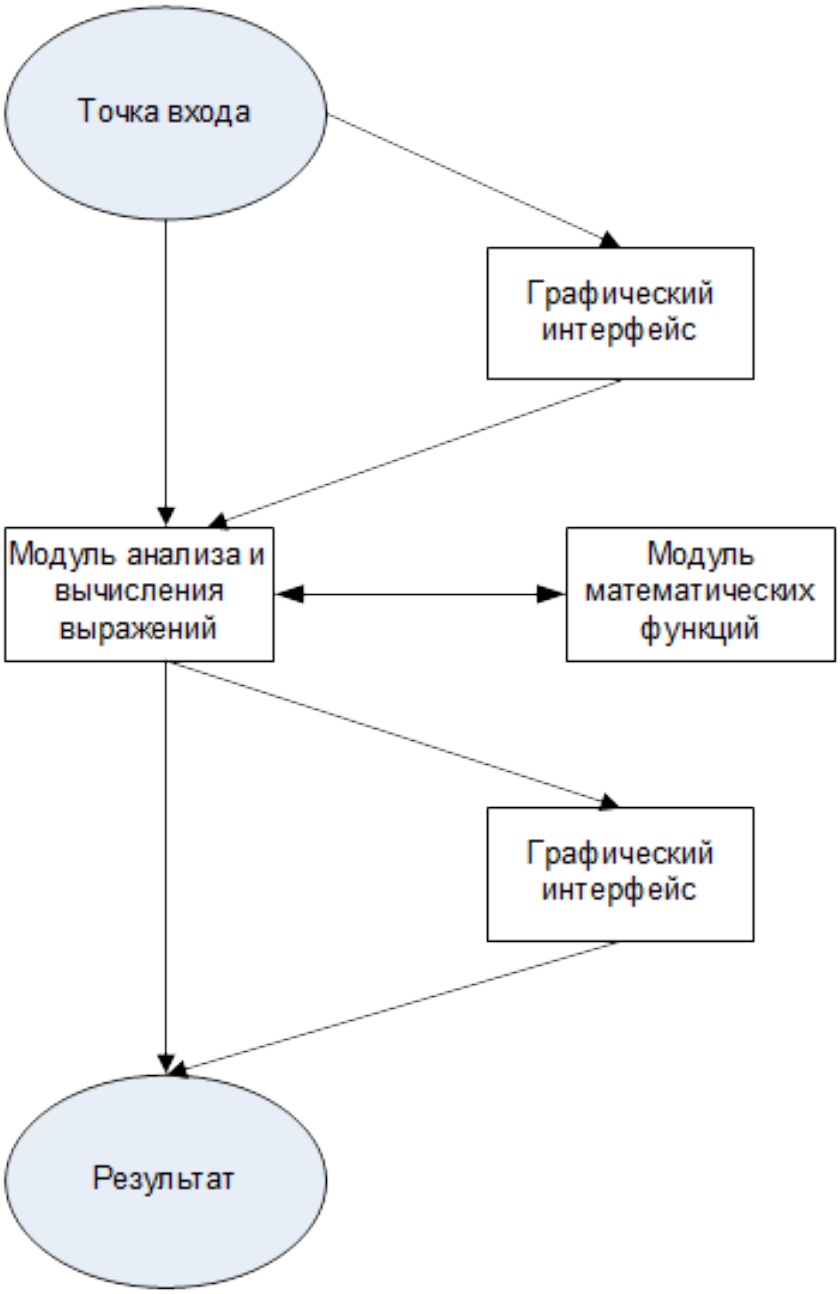
Система должна выполнять свою основную цель двумя способами: с помощью графического интерфейса и с помощью командной строки.

2.3. Архитектура

В архитектуре системы выделено 3 модуля. Каждый из модулей занимается определенной задачей. Соответственно, Система – это взаимодействие этих 3-х модулей. Разбиение Системы на модули вытекает из различной функциональности этих модулей. Рассмотрим их:

1) Модуль математических функций – так как Система будет иметь дело с математикой, нам потребуется подобный модуль. В него включены такие функции как сложение, умножение и др.
2) Модуль анализа и вычисления выражений – это модуль, который занимается главной задачей Системы. Разбор и компиляция выражений – вот основные функции этого модуля. Непосредственные вычисления этот модуль не проводит, а лишь вызывает функции из математического модуля.
3) Модуль графического интерфейса – обеспечивает управление системы в графической форме. Основные функции этого модуля – ввод и вывод данных.

Взаимодействие модулей показано на рисунке:



Как видно из рисунка, передать данные в Систему можно двумя способами: либо через графический модуль, либо через командную строку (последнее неявно прослеживается по рисунку). В любом случае, после передачи выражения Системе, начинает работу модуль анализа и вычислений, который по мере необходимости использует модуль математики, для вычисления арифметических функций. После окончания работы модуля анализа и вычислений, на выход передается результат.

2.4. Программный код

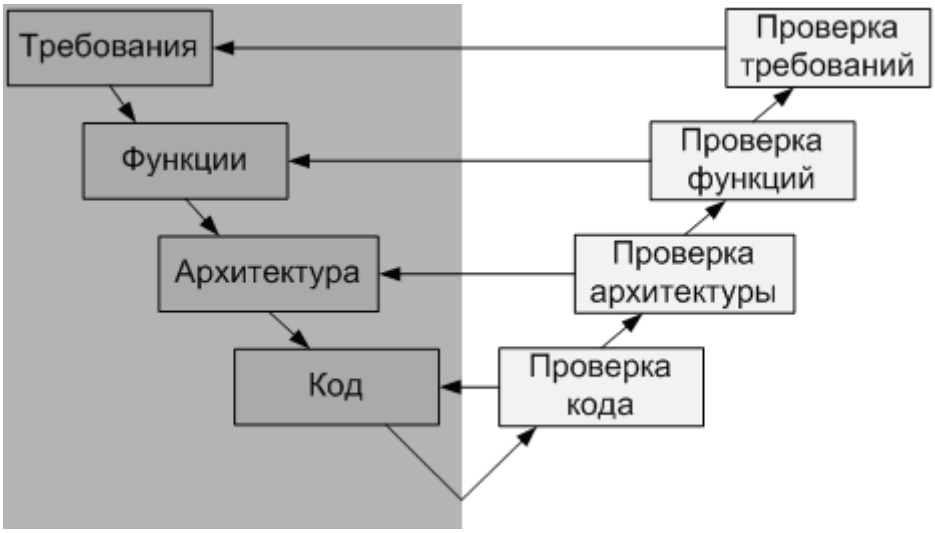
Весь программный код Системы разбит на модули соответственно архитектуре. Это позволит нам тестировать каждый модуль отдельно, о чем мы и будем говорить далее.

3. Тестирование Системы

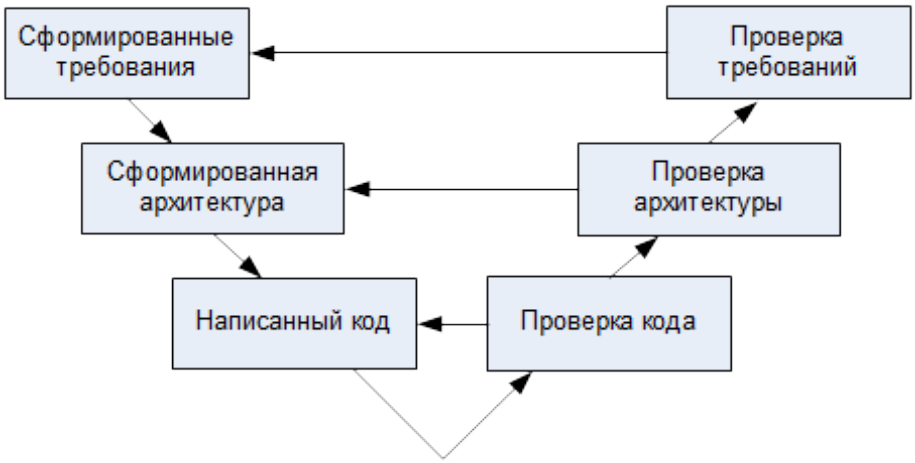
3.1. Общее описание

Как было сказано, наша основная задача – протестировать Систему.

Если в общем случае рассматривать жизненный цикл системы (например, V-образный), то наша задача лежит где-то справа.



Применительно к нашей системе, жизненный цикл можно представить так:



Сначала необходимо проверить код, затем архитектуру и требования.

3.2. Проверка кода

На этом этапе необходимо проверить корректность работы написанного кода. Для этого предлагается проводить тестирование каждого модуля отдельно. То есть, мы будем тестировать модули по отдельности, подменяя используемые методы других модулей «заглушками».

Например, при тестировании модуля анализа и вычислений выражений модуль, отвечающий за вычисления простых математический функций, можно заменить на модуль, содержащий стандартные методы области Math. Так мы будем точно знать, что все ошибки, выявленные при тестировании, не имеют отношения к нашей заглушке. Таким образом, заменив все модули, кроме тестируемого, заглушками, мы сможем утверждать, что все ошибки, обнаруженные при тестировании, будут относиться к “настоящему” (тестируемому) модулю.

Более того, заглушки дают нам дополнительное преимущество в тестировании. Мы можем написать заглушки, возвращающие пользователю дополнительную информацию во время тестирования. Например, нам необходимо узнать значение определенной переменной во время выполнения программы. Для этого мы можем написать заглушку, которая будет записывать значение этой переменной в какой-нибудь лог-файл или на консоль.

Немного о тестировании конкретных модулей:

GUI

На примере этого модуля можно было бы узнать, какие подходы существуют для тестирования современного графического интерфейса. В рамках данного курса этот вид тестирования рассматриваться не будет.

Математические функции

Этот модуль мы будем исследовать как "Черный ящик" и выяснять, действительно ли реализованные в нем математические функции работают корректно.

Вычисление выражений

При тестировании этого модуля нам предстоит проверить корректность алгоритмов разбора и компиляции математических выражений.

Последовательность тестирования модулей следующая:

Сначала мы познакомимся с методами ручного тестирования в среде разработки при ручном тестировании модуля анализа и вычисления выражений. Затем мы перейдем к модульному тестированию. В завершении тестирования компонент мы проведем формальные инспекции кода. После этого мы узнаем, что такое покрытия и как они используются в процессе тестирования.

3.3. Проверка архитектуры

После проверки каждого модуля по отдельности мы проведем интеграционное тестирование. На этом этапе проверяется, как модули взаимодействуют друг с другом. При условии, что все модули протестированы и ошибок в них не выявлено, все ошибки на этом этапе будут относиться именно к взаимодействию модулей между собой.

3.4. Проверка требований

После прохождения всех этапов тестирования необходимо провести проверку требований Системы в целом, то есть провести системное тестирование. Но в рамках данного курса этот вид тестирования рассматриваться не будет.

4. Приложение 1

4.1. Раздаточный материал

Спецификация на программу «Калькулятор. Базовая версия».

1. **Общее описание.**
   Часть общего описания – см. 2.1.
   Калькулятор состоит из трех модулей – «Графический интерфейс», «Модуль, анализирующий и вычисляющий введенное выражение» (AnalaizerClass.dll) и «Модуль, реализующий математические функции» (CalcClass.dll). После того, как пользователь введет вычисляемое выражение одним из двух вышеописанных способов, управление передается анализирующему модулю, который форматирует выражение, выделяя числа и операторы, проверяет корректность скобочной структуры, а также выявляет неверные с точки зрения математики конструкции (например, 3+*+3), переводит выражение в обратную польскую запись, после чего вычисляет выражения, используя математические функции из модуля CalcClass.

2. **Описание интерфейса.**

   2.1. Входные данные

   2.1.1. Параметры вызова (формат командной строки)
   calc.exe [expression]
   expression –математическое выражение удовлетворяющее требованию 3.2

   2.1.2. Состояние информационного окружения .
   В папке с программой также находятся файлы CalcClass.dll, AnalaizerClass.dll

   2.2. Выходные данные.

   2.2.1. Коды возврата программы.
   Число и 0 на новой строке – результат вычислений выражения.
   Error: <сообщение об ошибке> и код ошибки на новой строке - сообщение об ошибке в случае несоответствия входного выражения требованиям 3.2

   2.2.2. Состояние информационного окружения после завершения программы.
   В папке с программой также находятся файлы CalcClass.dll, AnalaizerClass.dll

   2.2.3. Сообщения об ошибках, выдаваемые программой (коды ошибок).
   Error 01 at <i> - Неправильная скобочная структура, ошибка на <i> символе
   Error 02 at <i> - Неизвестный оператор на <i> символе.
   Error 03 - Неверная синтаксическая конструкция входного выражения
   Error 04 at <i> - Два подряд оператора на <i> символе.
   Error 05 - Незаконченное выражение.
   Error 06 - Слишком малое или слишком большое значение числа для int
   Числа должны быть в пределах от -2147483648 до 2147483647
   Error 07 - Слишком длинное выражение. Максмальная длина - 65536 символов.
   Error 08 - Суммарное количество чисел и операторов превышает 30
   Error 09 – Ошибка деления на 0.

2.3. Описание файлов, входящих в пакете калькулятора.

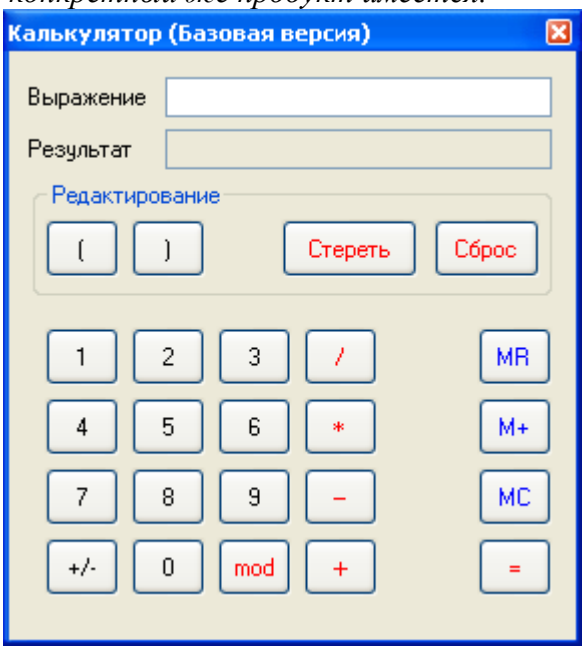
CalcClass.dll – библиотека, в которой реализованы все необходимые математические функции.

AnalaizerClass.dll – модуль, в котором реализован синтаксический разбор выражения, а также его вычисление.

calc.exe – графическая оболочка, главный модуль.

2.4. Интерфейс пользователя.

С одной стороны – это спецификация и в ней не указывают конкретный вид программы, с другой – конкретный же продукт имеется.



Клавиши «1» «2» «3» «4» «5» «6» «7» «8» «9» «0» «/» «*» «-» «+» «mod» «(» «)»– вводят соответствующий символ в поле выражение. Клавиша «Сброс» очищает поле «Выражение», клавиша «Стереть» удаляет последний введенный символ. Клавиша «=» начинает выполнение вычислений. «MR», «M+» и «MC» управляют памятью калькулятора, «+/-» - триггер унарного плюса унарного минуса.

3. Описание архитектуры.

Как уже отмечалось выше, в архитектуре системы выделено 3 модуля. Каждый из модулей занимается определенной задачей. Соответственно, Система – это взаимодействие этих 3-х модулей. Рассмотрим их подробнее.

1. Модуль математических операций (CalcClass.dll)

Модуль содержит все математические функции, используемые в программе.

```
/// <summary>
/// Функция сложения числа a и b
/// </summary>
/// <param name="a">слагаемое</param>
/// <param name="b">слагаемое</param>
/// <returns>сумма</returns>
public static int Add(long a, long b)

/// <summary>
/// функция вычитания чисел a и b
/// </summary>
/// <param name="a">уменьшаемое</param>
/// <param name="b">вычитаемое</param>
/// <returns>разность</returns>
public static int Sub(long a, long b)
```

```
/// <summary>
/// функция умножения чисел a и b
/// </summary>
/// <param name="a">множитель</param>
/// <param name="b">множитель</param>
/// <returns>произведение</returns>
public static int Mult(long a, long b)

/// <summary>
/// функция нахождения частного
/// </summary>
/// <param name="a">делимое</param>
/// <param name="b">делитель</param>
/// <returns>частное</returns>
public static int Div(long a, long b)

/// <summary>
/// функция деление по модулю
/// </summary>
/// <param name="a">делимое</param>
/// <param name="b">делитель</param>
/// <returns>остаток</returns>
public static int Mod(long a, long b)

/// <summary>
/// унарный плюс
/// </summary>
/// <param name="a"></param>
/// <returns></returns>
public static int ABS(long a)

/// <summary>
/// унарный минус
/// </summary>
/// <param name="a"></param>
/// <returns></returns>
public static int IABS(long a)
```

Используется также глобальная переменная:

```
/// <summary>
/// Последнне сообщение об ошибке.
/// Поле и свойство для него
/// </summary>
private static string _lastError = "";

public static string lastError
```

2. Модуль анализа и вычисления выражений

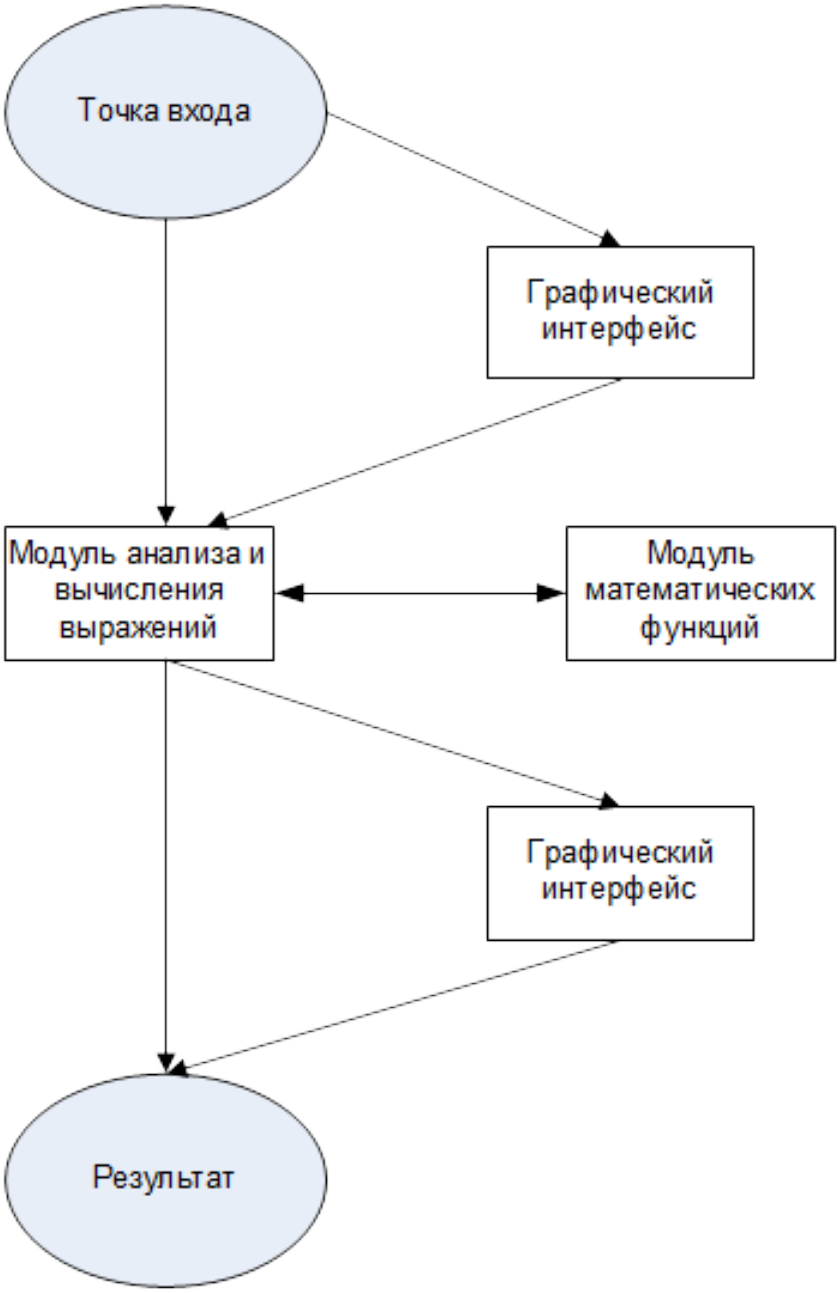
Состоит из следующих методов и свойств:

```
/// <summary>
/// позиция выражения на которой отловлена синтаксическая ошибка (в
    случае ловли на уровне выполнения - не определяется)
/// </summary>
private static int erposition = 0;
/// <summary>
/// Входное выражение
/// </summary>
public static string expression = "";
```

```
/// <summary>
/// Показывает, есть ли необходимость в выводе сообщений об ошибках.
    В случае консольного запуска программы это значение - false.
/// </summary>
public static bool ShowMessage = true;
/// <summary>
/// Проверка корректности скобочной структуры входного выражения
/// </summary>
/// <returns>true - если все нормально, false- если есть
    ошибка</returns>
/// метод бежит по входному выражению символ за символом анализирую
    его и считая количество скобочек. В случае возникновения
/// ошибки возвращает false а в erposition записывает позицию на
    которой возникла ошибка.
public static bool CheckCurrency()

/// <summary>
/// Форматирует входное выражение, выставляя между операторами
    проьелы и удаляя лишние, а также отлавливает неопознанные
    операторы, следит за концом строки
/// а также отлавливает ошибки на конце строки
/// </summary>
/// <returns>конечную строку или сообщение об ошибки, начинающиесь со
    спец символа &</returns>
public static string Format()

/// <summary>
/// Создает  массив, в котором располагаются операторы и символы
    представленные в обратной польской записи (безскобочной)
/// На этом же этапе отлавливаются почти все остальные ошибки (см
    код). По сути - это компиляция.
/// </summary>
/// <returns>массив обратной польской записи</returns>
public static System.Collections.ArrayList CreateStack()

/// <summary>
/// Вычисление обратной польской записи
/// </summary>
/// <returns>результат вычислений или сообщение об ошибке</returns>
public static string RunEstimate()

/// <summary>
/// Метод, организующий вычисления. По очереди запускает
    CheckCorrncy, Format, CreateStack и RunEstimate
/// </summary>
/// <returns></returns>
public static string Estimate()
```

3. Модуль графического интерфейса – обеспечивает управление системы в графической форме. Основные функции этого модуля – ввод и вывод данных.

Взаимодействие модулей показано на рисунке:



4. Функциональные требования.

4.1. Требования к программе.

4.1.1. Калькулятор должен выполнять следующие арифметические операции: сложение, вычитание, умножение, нахождение частного, нахождение остатка. Спецификацию на них см. 3.2.

4.1.2. Калькулятор должен поддерживать работу с целыми числами в пределах от -2147483648 до 2147483647 (в дальнейшем MININT и MAXINT). В случае выхода за эти пределы должно выдаваться сообщение об ошибке Error 06.

4.1.3. Калькулятор должен иметь память на одно целое число, а также возможность выводить это число на экран, сбрасывать его значение на 0 и прибавлять к нему любое другое число, введенное в поле ввода.

4.1.3.1. При нажатии на клавишу M+, к числу записанному в память прибавляется число записанное в поле «Результат». При этом на сложения накладываются ограничения из 3.2.1.

4.1.3.2. Если в поле “Результат” записан код ошибки, то при нажатии на клавишу M+ должно выдаваться сообщение «Невозможно преобразовать к числу».

4.1.3.3. При нажатии на кнопку MC число в памяти обнуляется.

4.1.3.4. При нажатии на кнопку MR число из памяти приписывается в конец выражения в строке «Выражение».

4.1.4. Калькулятор должен предоставлять возможность пользователю работать с операциями унарного плюса и унарного минуса.

4.1.4.1. Если между нажатиями на кнопку <+/-> проходит менее 3 секунд то введенный оператор меняется на противоположный.

4.1.4.2. Если между нажатиями на кнопку <+/-> проходит более 3 секунд то к выражению дописывается знак «-».

4.1.5. Калькулятор должен иметь графический интерфейс, содержащий кнопки с цифрами и арифметическими операциями, кнопкой равенства, кнопками работы с памятью, кнопками редактирования скобочек и кнопками сброса, переключателем унарного минуса/унарного плюса, текстовыми полями для ввода выражения и вывода результата.

4.1.6. При нажатии на клавишу <Enter> калькулятор должен проводить вычисления выражения.

4.1.7. При нажатии на клавишу <ESC> программа должна прекращать свою работу.

4.1.8. В случае неверно построенного вычисляемого выражения или несоответствия его требованиям 3.2 в текстовое окно результат должно выводиться соответствующие сообщение (см 2.2.3)

4.2. Арифметические операции.

4.2.1. Сложение.

4.2.1.1. Для чисел, каждое из которых меньше либо равна MAXINT и больше либо равна MININT, функция суммирования должна возвращать правильную сумму с точки зрения математики.

4.2.1.2. Для чисел, сумма которых больше чем MAXINT и меньше чем MININT, а также, в случае если любое из слагаемых больше чем MAXINT или меньше чем MININT, программа должна выдавать ошибку Error 06(см 2.2.3)

4.2.2. Вычитание.

4.2.2.1. Для чисел, каждое из которых меньше либо равна MAXINT и больше либо равна MININT, функция вычитания должна возвращать правильную разность с точки зрения математики.

4.2.2.2. Для чисел, разность которых больше чем MAXINT и меньше чем MININT, а также, в случае если любое из чисел больше чем MAXINT или меньше чем MININT, программа должна выдавать ошибку Error 06(см 2.2.3)

4.2.3. Умножение.

4.2.3.1. Для чисел, произведение которых меньше либо равна MAXINT и больше либо равна MININT, функция умножения должна возвращать правильное произведение с точки зрения математики.

4.2.3.2. Для чисел, произведение которых больше чем MAXINT и меньше чем MININT, а также, в случае если любой из множителей больше чем MAXINT или меньше чем MININT, программа должна выдавать ошибку Error 06(см 2.2.3)

4.2.4. Нахождение частного.

4.2.4.1. Для чисел, меньших либо равных MAXINT и больших либо равных MININT , частное которых меньше либо равна MAXINT и больше либо равна MININT и делитель не равен 0, функция деления должна возвращать правильное частное с точки зрения математики.

4.2.4.2. Для чисел, частное которых больше чем MAXINT и меньше чем MININT, а также, в случае если любое из чисел больше чем MAXINT или меньше чем MININT и для делителя не равного 0, программа должна выдавать ошибку Error 06(см 2.2.3)

4.2.4.3. Если делитель равен 0, программа должна выдавать ошибку Error 09

4.2.5. Деление с остатком.

4.2.5.1. Для чисел, меньших либо равных MAXINT и больших либо равных MININT , остаток которых меньше либо равна MAXINT и больше либо равна MININT и делитель не равен 0, функция деления должна возвращать правильный остаток с точки зрения математики.

4.2.5.2. Для чисел, остаток которых больше чем MAXINT и меньше чем MININT, а также, в случае если любое из чисел больше чем MAXINT или меньше чем MININT и для делителя не равного 0, программа должна выдавать ошибку Error 06(см 2.2.3)

4.2.5.3. Если делитель равен 0, программа должна выдавать ошибку Error 09

4.2.6. Унарный плюс \ минус.

4.2.6.1. Для чисел, меньших либо равных MAXINT и больших либо равных MININT операция унарного плюса / минуса должна возвращать число соответствующего знака.

4.2.6.2. Для чисел больших MAXINT или меньших MININT функция должна выдавать ошибку Error 06(см 2.2.3)

4.3. Дополнительные требования к входному выражению.

4.3.1. Максимальное суммарное число операторов и чисел – 30.

4.3.2. Максимальная глубина вложенности скобочной структуры – 3.

4.3.3. В качестве унарного минуса используется символ «m» , в качестве унарного плюса - «p».

4.3.4. Для операции нахождения частного – «/», для нахождения остатка «mod».

4.3.5. Между операторами скобками и числами может быть любое количество пробелов.

4.3.6. Разрешается использовать лишь скобки вида «(» и «)»

4.3.7. Максимальная длина выражения – 65535 символов.

5. Домашнее задание.

Изучать спецификацию и выявить имеющиеся изъяны. Обосновать их.