Стажировка (ООО ПриматЛаб)

Формулировка задания

1. Написать на языке C++ приложение-сервер (веб-сервис), которое
   - является кроссплатформенным
   - имеет REST API, которое позволяет тестировать запущенный сервер при помощи утилиты curl или веб-браузера
   - реализует требуемую функциональность и её тесты.
2. Предоставить исходники приложения и инструкции по сборке в виде ссылки на удалённый репозиторий.
3. Предоставить исчерпывающую англоязычную документацию к коду, включающую в себя описание архитектуры, всех разработанных классов и методов.

Описание требуемой функциональности и соответствующие методы API

Метод 1 - численное нахождение вещественных корней функции-полинома одной переменной

Функция-полином степени N задаётся в виде математического выражения

```
f(x) = A[0] + A[1]*x + A[2]*x^2 + A[3]*x^3 + ... + A[N]*x^N
```

Корнями функции `f(x)` считаются любые значения `x`, для которых выполняется условие `f(x) = 0`.

Требуется найти в Интернете и реализовать программно численный метод нахождения всех корней функции-полинома произвольной степени, заданной при помощи своих коэффициентов `A[0], ..., A[N]`.

Метод REST API должен иметь следующий синтаксис:

GET /api/findroots

Тело запроса должно иметь вид JSON-массива, содержащего коэффициенты, которые определяют функцию-полином в порядке возрастания индекса `[A[0], A[1], A[2], ..., A[N]]`. Таким образом, функцию-полином степени N задаёт массив из N+1 вещественного числа.

В качестве результата сервер должен возвращать список найденных корней полинома, упорядоченный по возрастанию.

Пример 1 (линейное уравнение):

Запрос вида

```
curl -X GET localhost:/api/findroots -d '[6, 2]'
```

означает, что мы просим сервер, запущенный на localhost, решить уравнение

```
f(x) = 6 + 2x = 0
```

Решением этого уравнения является x=-3. Поэтому в качестве ожидаемого ответа рассматривается следующее сообщение от сервера

```
[-3]
```

с кодом ошибки 200 OK.

Пример 2 (квадратичное уравнение)

Запрос вида

```
curl -X GET localhost:/api/findroots -d '[2, -3, 1]'
```

означает, что мы просим сервер, запущенный на localhost, решить уравнение

```
f(x) = 2 - 3*x + x^2= 0
```

Решением этого уравнения являются значения x1=1 и x2=2. Поэтому в качестве ожидаемого ответа рассматривается следующее сообщение от сервера

```
[1, 2]
```

с кодом ошибки 200 OK.

Пример 3 (квадратичное уравнение без вещественных корней)

Запрос вида

```
curl -X GET localhost:/api/findroots -d '[1, 1, 1]'
```

означает, что мы просим сервер, запущенный на localhost, решить уравнение

```
f(x) = 1 + x + x^2= 0
```

У этого уравнения нет вещественных корней. Поэтому в качестве ожидаемого ответа рассматривается следующее сообщение от сервера

```
[]
```

с кодом ошибки 200 OK.

Пример 4 (уравнение нулевой степени без корней)

Запрос вида

```
curl -X GET localhost:/api/findroots -d '[1]'
```

означает, что мы просим сервер, запущенный на localhost, решить уравнение

```
f(x) = 1 = 0
```

У этого уравнения нет вещественных корней. Поэтому в качестве ожидаемого ответа рассматривается следующее сообщение от сервера

```
[]
```

с кодом ошибки 200 OK.

Пример 5 (уравнение нулевой степени с бесконечным количеством корней)

Запрос вида

```
curl -X GET localhost:/api/findroots -d '[1]'
```

означает, что мы просим сервер, запущенный на localhost, решить уравнение

```
f(x) = 0 = 0
```

Любое вещественное число является корнем этого уравнения. В качестве ожидаемого ответа рассматривается следующее сообщение от сервера

```
[all numbers]
```

с кодом ошибки 200 OK.

Пример 6 (уравнение 4-ой степени)

Запрос вида

```
curl -X GET localhost:/api/findroots -d '[-1, 0, 0, 0, 1]'
```

означает, что мы просим сервер, запущенный на localhost, решить уравнение

```
f(x) = -1 + x^4= 0
```

Решением этого уравнения являются значения `x1=-1` и `x2=1`. Поэтому в качестве ожидаемого ответа рассматривается следующее сообщение от сервера

```
[-1, 1]
```

с кодом ошибки `200 OK`.

Комментарии

1. Корни полинома в общем случае находятся приближённо.
2. Для простоты будем считать, что все корни заданного многочлена лежат в интервале `[-100, 100]`.
3. Во избежание дополнительных сложностей будем считать, что максимальная степень полинома ограничена значением `N <= 10`. В случае, если степень полинома превышена в исходных данных, сервер должен вернуть пустое сообщение с кодом ошибки `400` и сообщением `Polynom degree is too high`.
4. В случае любой другой ошибки в формате входных данных необходимо вернуть код ошибки `400` и сообщение `Incorrect input`.

Метод 2 - вычисление значения функции-полинома одной переменной в точке

Данный метод должен осуществлять вычисление функции `f()` в точке `x` по формуле

```
f(x) = A[0] + A[1]*x + A[2]*x^2 + ... + A[N]*x^N
```

имея заданные значения коэффициентов полинома `A[0], A[1], ..., A[N]` и заданное значение `x`.

Метод REST API должен иметь следующий синтаксис:

GET /api/checkvalue

Тело запроса должно иметь вид JSON-объекта, содержащего коэффициенты, которые определяют функцию-полином и точку `x` в порядке `{"x":x, "polynomial":[A[0], A[1], A[2], ..., A[N]]}`. Таким образом, для проверки функции-полинома степени `N` в точке необходимо задать массив из `N+2` вещественного чисел.

В качестве результата сервер должен возвращать значение заданной функции-полинома в заданной точке.

Пример

Запрос вида

```
curl -X GET localhost:/api/checkvalue -d '{"x":3, "polynomial":[2,-3,1]}
```

означает, что мы просим сервер, запущенный на localhost, посчитать

```
f(x) = 2 - 3*x + x^2
```

при `x=3`.

Решением являтся значения `f(x)=2`. Поэтому в качестве ожидаемого ответа рассматривается следующее сообщение от сервера

```
2
```

с кодом ошибки `200 OK`.

Комментарий

1. В случае любой другой ошибки в формате входных данных необходимо вернуть код ошибки `400` и сообщение `Incorrect input`.

Рекомендации

1. Среда разработки - QtCreator.
2. Сборочные инструкции - qmake (идёт в комплекте с QtCreator)
3. Библиотека для разработки сервера - mongoose.
4. Библиотека для тестирования - Boost.Test или googletest.
5. Формат документирования - Doxygen.
6. Система контроля версий - Git/Mercurial.
7. Ресурс для размещения удалённого репозитория - bitbucket или github.

Режим работы

План

1. Разработка архитектуры сервиса.
2. Реализация сервиса и JSON API с заглушками вместо настоящих математических алгоритмов.
3. Разработка тестов для математических алгоритмов.
4. Реализация математических алгоритмов.
5. Разработка документации.
6. Тестирование.

Время

Старт - 18 декабря 2017 года

Окончание - 18 марта 2018 года

Интенсивность - до 10 часов в неделю

Общее затраченное время - до 130 часов

Обсуждение промежуточных результатов - 1 раз в 2 недели в офисе (дата и время согласуются по почте)