Crypto Trading Platform

Tradin 212 Project

Akaga Pavlova

Ръководител: доц. д-р Петър Русланов Армянов

3 май 2025 г.

Съдържание

1 Overview

1.1 Project description

Web application that simulates a cryptocurrency trading platform, allowing users to view real-time prices of the top 20 cryptocurrencies, maintaining a virtual account balance for buying and selling crypto with a history of all transactions made.

1.1.1 Requirements

The following requirements are implemented:

1.1.2 Functionality

- Dynamically update the top 20 Crypto prices in real-time as they change on the exchange.
- Clearly present the cryptocurrency name, symbol, and current price in a user-friendly table or list format.
- Initialize a virtual account balance with a starting value.
- Allow users to specify the amount of cryptocurrency they want to purchase.
- Deduct the purchase cost from the account balance.
- Display a confirmation message or update the UI to reflect the updated balance and holdings.
- Allow users to specify the amount of cryptocurrency they want to sell.
- Increase the account balance by the selling amount.
- Update the UI to reflect the updated balance and holdings.
- Ensure that transactions respect account balance limitations.
- Provide clear error messages if invalid input is entered.
- A user should be able to see a history log of all his transactions.

- Include a button or link that resets the account balance to the initial value.
- Upon clicking the reset button, update the UI to show the original balance and clear any cryptocurrency holdings.
- Dynamically update the top 20 Crypto prices in real-time as they change on the exchange.
- Clearly present the cryptocurrency name, symbol, and current price in a user-friendly table or list format.
- Initialize a virtual account balance with a starting value.
- Allow users to specify the amount of cryptocurrency they want to purchase.
- Deduct the purchase cost from the account balance.
- Display a confirmation message or update the UI to reflect the updated balance and holdings.
- Allow users to specify the amount of cryptocurrency they want to sell.
- Increase the account balance by the selling amount.
- Update the UI to reflect the updated balance and holdings.
- Ensure that transactions respect account balance limitations.
- Provide clear error messages if invalid input is entered.
- A user should be able to see a history log of all his transactions.
- Include a button or link that resets the account balance to the initial value.
- Upon clicking the reset button, update the UI to show the original balance and clear any cryptocurrency holdings.уми, сортиране, принтиране и други.
- Dynamically update the top 20 Crypto prices in real-time as they change on the exchange.
- Clearly present the cryptocurrency name, symbol, and current price in a user-friendly table or list format.

- Initialize a virtual account balance with a starting value.
- Allow users to specify the amount of cryptocurrency they want to purchase.
- Deduct the purchase cost from the account balance.
- Display a confirmation message or update the UI to reflect the updated balance and holdings.
- Allow users to specify the amount of cryptocurrency they want to sell.
- Increase the account balance by the selling amount.
- Update the UI to reflect the updated balance and holdings.
- Ensure that transactions respect account balance limitations.
- Provide clear error messages if invalid input is entered.
- A user should be able to see a history log of all his transactions.
- Include a button or link that resets the account balance to the initial value.
- Upon clicking the reset button, update the UI to show the original balance and clear any cryptocurrency holdings. приложението.
- Dynamically update the top 20 Crypto prices in real-time as they change on the exchange.
- Clearly present the cryptocurrency name, symbol, and current price in a user-friendly table or list format.
- Initialize a virtual account balance with a starting value.
- Allow users to specify the amount of cryptocurrency they want to purchase.
- Deduct the purchase cost from the account balance.
- Display a confirmation message or update the UI to reflect the updated balance and holdings.
- Allow users to specify the amount of cryptocurrency they want to sell.

- Increase the account balance by the selling amount.
- Update the UI to reflect the updated balance and holdings.
- Ensure that transactions respect account balance limitations.
- Provide clear error messages if invalid input is entered.
- A user should be able to see a history log of all his transactions.
- Include a button or link that resets the account balance to the initial value.
- Upon clicking the reset button, update the UI to show the original balance and clear any cryptocurrency holdings.
- Dynamically update the top 20 Crypto prices in real-time as they change on the exchange.
- Clearly present the cryptocurrency name, symbol, and current price in a user-friendly table or list format.
- Initialize a virtual account balance with a starting value.
- Allow users to specify the amount of cryptocurrency they want to purchase.
- Deduct the purchase cost from the account balance.
- Display a confirmation message or update the UI to reflect the updated balance and holdings.
- Allow users to specify the amount of cryptocurrency they want to sell.
- Increase the account balance by the selling amount.
- Update the UI to reflect the updated balance and holdings.
- Ensure that transactions respect account balance limitations.
- Provide clear error messages if invalid input is entered.
- A user should be able to see a history log of all his transactions.
- Include a button or link that resets the account balance to the initial value.

- Upon clicking the reset button, update the UI to show the original balance and clear any cryptocurrency holdings.
- Dynamically update the top 20 Crypto prices in real-time as they change on the exchange.
- Clearly present the cryptocurrency name, symbol, and current price in a user-friendly table or list format.
- Initialize a virtual account balance with a starting value.
- Allow users to specify the amount of cryptocurrency they want to purchase.
- Deduct the purchase cost from the account balance.
- Display a confirmation message or update the UI to reflect the updated balance and holdings.
- Allow users to specify the amount of cryptocurrency they want to sell.
- Increase the account balance by the selling amount.
- Update the UI to reflect the updated balance and holdings.
- Ensure that transactions respect account balance limitations.
- Provide clear error messages if invalid input is entered.
- A user should be able to see a history log of all his transactions.
- Include a button or link that resets the account balance to the initial value.
- Upon clicking the reset button, update the UI to show the original balance and clear any cryptocurrency holdings.

1.2 Структура на документацията

Тази документация обхваща архитектурата на проекта, реализацията на основните класове и тяхната функционалност - представени са и примерни тестове, които обхващат различни сценарии за тяхното проложение. Изложени са идеи за бъдещи подобрения.

2 Преглед на предметната област

2.1 Основни дефиниции, концепции и алгоритми

2.1.1 Основни дефиниции и концепции

Проектът използва редица ключови дефиниции, които са съществени за разбирането и имплементацията на решението:

- Файл - базовата единица за съхранение на данните на всеки файл: име, съдържание и данни за всяка незаписана операция над този файл.
- Текст - базовата единица за съхранение на текстови данни.
- Команда - операция, съдържаща информация за функционалността и нейните операнди.
- Филтър - конкретна функционалност, която променя обработва в съответствие с определени критерии.
- Дисплей - грижи се за извеждането на информацията от конкретен файл на стандартния изход в подходящ вид.

2.1.2 Алгоритми

За реализацията на различните операции и филтри се използват следните алгоритми:

- Замяна на думи - алгоритъм за намиране и замяна на думи в текста(квадратична сложност).
- Сортиране - алгоритъм за сортиране на редовете в текста(полилогаритмична сложност).
- Изтриване на дубликати - алгоритъм за намиране и изтриване на повторения в текста(квадратична сложност).
- Центриране на текста - алгоритъм за подравняване на редовете при извеждане(линейна сложност).

2.2 Дефиниране на проблеми и сложност на поставената задача

2.2.1 Проблеми

Основните проблеми, които трябва да бъдат решени при разработката на избраната имплементация:

- Управление на файловете - ефективно съхранение и манипулация на данни във файловата система.
- Ефективност на операциите - бързодействие на операциите върху текста при големи обеми данни.
- Интеграция на филтри и команди - създаване на модулна система, която позволява лесно добавяне и премахване на операции.
- Потребителски интерфейс - предоставяне на удобен потребителски интерфейс за работа с текста.

2.3 Подходи, методи за решаване на проблемите

- Обектно-ориентиран модел - използване на класове и обекти за представяне на различните компоненти на текстовия процесор.
- Модулна архитектура - разделяне на функционалността на модули, което позволява лесно добавяне и промяна на нови операции и филтри.
- Оптимизация на алгоритмите - използване на ефективни алгоритми и структури от данни за подобряване на бързодействието на процесите.
- Потребителски интерфейс - създаване на прост и ясен интерфейс, който да улесни работата на потребителя с проложението.

Реализацията на гореспоменатите подходи е ключова за успешното разработване и функциониране на текстовия процесор, за неговата устойчивост и лесна разширяемост в бъдеще.

3 Проектиране

3.1 Обща архитектура – ООП дизайн

Имплементацията се свежда до дизайн и дефиниция на градивните части на програмата - файлове, текстове и команди, както и връзките между тях.

3.2 Йерархии на класовете

3.2.1 Йерархия на класовете за филтриране

4 Реализация, тестване

4.1 Реализация на класове

4.1.1 Клас File

Класът File представлява абстракцията текстов файл и осигурява методи за управление на съдържанието му.

```
class File
{
public:
    File(std::ifstream& istr, const char* name);
    void removeLastOperation();
    void addOperation(const Command& operation);
    void removeOperations();
    bool hasUnsavedChanges() const;
    void saveChanges(const MyString& path);
    void print(const Display* sink) const;
    const MyString& getName() const;
    const Text& getContent() const;
private:
```

```
    bool unsavedChanges = false;
    MyString name;
    Text content;
    MyVector<Command> operations;
};
```

4.1.2 Клас Command

Класът Command представлява команда за изпълнение на операция върху файл, като съдържа код за типа на операцията, операнди и филтър за приложение на операцията.

```
class Command
{
public:
    Command();
    Command(const Command& other);
    Command& operator=(const Command& other);
    ~Command();

    void setCode(CommandMap c);
    void setOperandsCount(int cnt);
    void setOperands(const MyString& userInput, bool whole =
        false);
    void setOperation(const BaseFilter* op);
    void setSink(const Display* sink);

    const Display* getSink() const;
    const BaseFilter* getFilter() const;
    const MyVector<MyString>& getOperands() const;
    CommandMap getCode() const;
    void free();

```

```
21 private:
22     void copyFromOther(const Command& other);
23
24     short operandsCount = 0;
25     CommandMap code = CommandMap::Invalid;
26     const BaseFilter* operation = nullptr;
27     const Display* sink = nullptr;
28     MyVector<MyString> operands;
29 };
```

4.1.3 Клас CommandManager

Класът CommandManager е синглетон, който управлява командите, идващи от потрбителя и осигурява методи за тяхната обработка и изпълнение.

```
1     class CommandManager
2     {
3     public:
4         void introduction() const;
5         void showCommandMap() const;
6         void handleInput();
7         void execute();
8
9         bool isRunning() const;
10
11        static CommandManager& getInstance()
12        {
13            static CommandManager instance("Commands.txt");
14            return instance;
15        }
16
17    private:
18        CommandManager(const char* name);
```

```
19          CommandManager(const CommandManager& other) = delete;
20          CommandManager& operator=(const CommandManager& other) =
                delete;
21          bool running = true;
22          Command operation;
23          MyString commandsFileName;
24
25      public:
26          //this method is public only for test purposes
27          void setCommand(const MyString& userInput);
28      };
29
30      typedef CommandManager TheCommandManager;
```

4.1.4 Клас FileManager

Класът FileManager е синглетон, който управлява файловете - тяхното зареждане, записване, затваряне и съхранение на информация.

```
1       class FileManager
2       {
3       public:
4           File& operator[](int index);
5           const File& operator[](int index) const;
6           File& getCurrentFile();
7
8           void addFile(const MyString& fileName);
9           void removeFile(const MyString& fileName);
10          void clearFiles();
11          void changeCurrentFile(const MyString& fileName);
12          void getInfo() const;
13          int contains(const MyString& fileName) const;
14          size_t getFilesCount() const;
```

```
15
16          static FileManager& getInstance()
17          {
18              static FileManager instance;
19              return instance;
20          }
21          ~FileManager();
22
23      private:
24          FileManager() = default;
25
26          FileManager(const FileManager& other) = delete;
27          FileManager& operator=(const FileManager& other) =
                delete;
28
29          int currentFile = 0;
30          MyVector<File*> files;
31          Display* sink = nullptr;
32      };
33      typedef FileManager TheFileManager;
```

4.2 Планиране, описание и създаване на тестови сценарии

Реализирани са различни тестове посредством Catch2 framework, обхващащи логиката и поведението на програмата, последните се намират във файла "TESTS.CPP". Подробно са тествани градивните класове - MyString, MyVector, Text, както и различните помощни фукции, декларирани във файла "helpers.hpp". Към проекта са включени готовите тестови файлове, необходими за подкарване на тестовете.

4.2.1 Примерни тестове и техния резултат

```
1       TEST_CASE("Testing Sort on lines with fixed positions")
2   {
```

```
3   Text source;
4   MyString openSourceFile = "open TextFiles\\SortTests\\
        staticLines.txt";
5   TheCommandManager::getInstance().setCommand(openSourceFile);
6   TheCommandManager::getInstance().execute();
7   source = FileManager::getInstance().getCurrentFile().getConten
        ();
8   MyString sortSourceFile = "sort";
9   TheCommandManager::getInstance().setCommand(sortSourceFile);
10  TheCommandManager::getInstance().execute();
11  MyString saveByOtherName = "saveas TextFiles\\SortTests\\
        staticLinesProgramOutput.txt";
12  TheCommandManager::getInstance().setCommand(saveByOtherName);
13  TheCommandManager::getInstance().execute();
14  MyString openOutputFile = "open TextFiles\\SortTests\\
        staticLinesProgramOutput.txt";
15  TheCommandManager::getInstance().setCommand(openOutputFile);
16  TheCommandManager::getInstance().execute();
17
18  SECTION("Comparing content")
19  {
20    CHECK(FileManager::getInstance().getCurrentFile().getConten
          ().getLinesCount() == source.getLinesCount());
21    for(size_t i = 0; i < source.getLinesCount(); ++i)
22      CHECK(FileManager::getInstance().getCurrentFile().
            getConten().getLine(i + 1) == source.getLine(i + 1));
23  }
24
25  MyString exit = "exit";
26  TheCommandManager::getInstance().setCommand(exit);
27  TheCommandManager::getInstance().execute();
28 }
```

5 Заключение

5.1 Обобщение на изпълнението на началните цели

Основната фукционалност на проекта е реализирна, направени са необходимите тестове за голяма част от логиката на програмата Постигната е абстракция и модулярност на решението, което улеснява интегрирането на бъдещи разширения.

5.2 Насоки за бъдещо развитие и усъвършенстване

По-високото ниво на абстракция на класа Command би довело до по-лесна обработка на по-широка фунцкионалност. С оглед на по-добър потребителски опит, извеждането на екрана и форматирането може да се подобри. За сигурност за запазване на данните при евентуална грешка при изпълнение на програмата може да се използва резервен буферен файл.

6 Изтоници

"THE LINUX COMMAND LINE A Complete Introduction by William E. Shotts, Jr."