**ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ**

**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ**

**«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

Факультет компьютерных наук

Департамент программной инженерии

|  |  |  |
| --- | --- | --- |
| СОГЛАСОВАНО  Д. ф. -м. наук, профессор департамента программной инженерии факультета компьютерных наук  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ М.И. Кумсков  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2018 г. |  | УТВЕРЖДАЮ  Академический руководитель образовательной программы «Программная инженерия»  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.В. Шилов  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2018 г. |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | ***Подп. и дата*** |  | | ***Инв. № дубл.*** |  | | ***Взам. инв. №*** |  | | ***Подп. и дата*** |  | | ***Инв. № подл*** | RU.17701729.04.15-01 81 01-1 | | **пРОГРАММА СТАТИСТИЧЕСКОГО АНАЛИЗА ФИНАНСОВЫХ КОТИРОВОК ПРОВЕРКИ ПАТТЕРНОВ “Японских свечей”**  **Пояснительная записка**  **ЛИСТ УТВЕРЖДЕНИЯ**  **RU.17701729.04.15-01 81 01-1** | | |
|  |  | |
| Исполнитель:  студент группы БПИ172  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / Иванов М.В. /  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2018 г. | |
|  | | |
|  | |  |

**2018**

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  | |
| |  |  | | --- | --- | | ***Подп. и дата*** |  | | ***Инв. № дубл.*** |  | | ***Взам. инв. №*** |  | | ***Подп. и дата*** |  | | ***Инв. № подл*** | RU.17701729.04.15-01 81 011 | | **пРОГРАММА СТАТИСТИЧЕСКОГО АНАЛИЗА ФИНАНСОВЫХ КОТИРОВОК ПРОВЕРКИ ПАТТЕРНОВ “Японских свечей”**  **Пояснительная записка**  **RU.17701729.** **508830-01 81 01-1ЛУ**  **Листов 23** | | | | |
|  |  | | | |
|  | | | |
|  | | | | |
|  | | |  | |

**2018**

СОДЕРЖАНИЕ

[**1.Введение** 3](#_Toc514658215)

[**1.1.** **Наименование разработки** 3](#_Toc514658216)

[**1.2 Основание для разработки** 3](#_Toc514658217)

[2. НАЗНАЧЕНИЕ И ОБЛАСТЬ ПРИМЕНЕНИЯ 4](#_Toc514658218)

[**2.1.** **Функциональное** **назначение** 4](#_Toc514658219)

[**2.2.** **Эксплуатационное** **назначение** 4](#_Toc514658220)

[3. ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ 5](#_Toc514658221)

[**3.1.** **Постановка задачи на разработку программы** 5](#_Toc514658222)

[3.2 Описание алгоритма и функционирования программы 5](#_Toc514658223)

[3.2.1 Описание алгоритма поиска паттернов. Паттерн “Молот”. 6](#_Toc514658224)

[3.2.2 Описание алгоритма поиска паттернов. Паттерн “Падающая звезда”. 7](#_Toc514658225)

[3.2.3 Описание алгоритма поиска паттернов. Паттерн “Повешенный”. 7](#_Toc514658226)

[3.2.4 Описание алгоритма поиска паттернов. Паттерн “Дожи”. 8](#_Toc514658227)

[3.2.5 Описание алгоритма поиска паттернов. Паттерн “Бычье поглощение”. 8](#_Toc514658228)

[3.2.6 Описание алгоритма поиска паттернов. Паттерн “Медвежье поглощение” 9](#_Toc514658229)

[3.2.7 Описание алгоритма поиска паттернов. Паттерн “Бычий захват за пояс”. 10](#_Toc514658230)

[3.2.8 Описание алгоритма поиска паттернов. Паттерн “Медвежий захват за пояс”. 10](#_Toc514658231)

[3.2.9 Сбор статистики о появлении паттернов и их работе. 11](#_Toc514658232)

[3.2.10 Чтение данных из файла 13](#_Toc514658233)

[3.3 Описание и обоснование метода организации входных данных 13](#_Toc514658234)

[**3.4 Описание и обоснование выбора и состава технических и программных средств** 14](#_Toc514658235)

[3.4.1 Состав технических и программных средств 14](#_Toc514658236)

[3.4.2 Обоснование выбора состава технических и программных средств 14](#_Toc514658237)

[ОЖИДАЕМЫЕ ТЕХНИКО-ЭКОНОМИЧЕСКИЕ ПОКАЗАТЕЛИ 14](#_Toc514658238)

[4.1. Предполагаемая потребность 14](#_Toc514658239)

[4.2. Экономические преимущества разработки по сравнению с отечественными и зарубежными образцами или аналогами 14](#_Toc514658240)

[Поиск в сети Интернет приложения, аналогичного программе FindPatterns, не дал положительного результата. 14](#_Toc514658241)

[5. ИСТОЧНИКИ, ИСПОЛЬЗОВАННЫЕ ПРИ РАЗРАБОТКЕ 15](#_Toc514658242)

[ПРИЛОЖЕНИЕ 1. ОПИСАНИЕ И ФУНКЦИОНАЛЬНОЕ НАЗНАЧЕНИЕ КЛАССОВ МЕТОДОВ И ПОЛЕЙ 15](#_Toc514658243)

[Описание и функциональное назначение классов 15](#_Toc514658244)

[ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ 22](#_Toc514658245)

**1.Введение**

* 1. **Наименование разработки**

Полное наименование программы на русском языке – «Программа статистического анализа финансовых котировок проверки паттернов "японских свечей".»

Полное наименование программы на английском языке – «The program of statistical analysis of financial quotes for the verification of "Japanese candle" patterns.»

**1.2 Основание для разработки**

**Документ, на основании которого ведется разработка:**

Приказ декана факультета компьютерных наук Национального исследовательского университета «Высшая школа экономики» № 2.3-02/1212-01 от 12.12.2017 «Об утверждении тем, руководителей курсовых работ студентов образовательной программы Программная инженерия факультета компьютерных наук».

**Организация, утвердившая документ:**

Документ утвержден НИУ ВШЭ, факультетом компьютерных наук, департаментом программной инженерии.

1. **НАЗНАЧЕНИЕ И ОБЛАСТЬ ПРИМЕНЕНИЯ**
   1. **Функциональное** **назначение**

Программа представляет из себя финансовый индикатор для получения статистики появления и работы паттернов Японских свечей. Для получения результата пользователю необходимо загрузить историю валютной пары/акции в индикатор.

|  |  |  |
| --- | --- | --- |
|  |  |  |
| а) |  | б) |

Рисунок 1. Пример работы программы. а) Кнопка загрузить; б) Кнопка поиска паттернов;

* 1. **Эксплуатационное** **назначение**

Данная программа предназначена для сбора данных о поведении индексов и цен акций на основе данных о предыдущих торговых периодах.

1. **ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ**
   1. **Постановка задачи на разработку программы**

* Программа должна предоставлять пользователю возможность загрузить данные в формате .csv ;
* Программа должна предоставить пользователю увидеть загруженные данные в формате таблицы;
* Программа должна строить график по загруженным данным;
* Программа должна осуществлять поиск паттернов японских свечей:

"Молот", "Падающая звезда", "Повешенный”, "Дожи", "Бычье поглощение",

"Медвежье поглощение", "Бычий захват за пояс", "Медвежий захват за пояс"

* Программа должна предоставлять возможность выхода и свертки;
* Программа должна давать возможность просмотреть статистику найденных паттернов;

**3.2** **Описание алгоритма и функционирования программы**

|  |  |  |
| --- | --- | --- |
|  |  |  |
| а) |  |  |

Рисунок 1. Таблица с входными данными (ее мы считываем)

Далее мы создаем класс Candle

Класс Candle на языке C#:

public class Candle

{

public DateTime time;

public double open;

public double high;

public double close;

public double low;

public string volume;

public string patternName;

public bool work;

}

Считываем все свечки из файла и записываем их в массив CandleArray

массив CandleArray на языке C#:

public Candle[] CandleArray = Base.ParseCandle(MainWindow.filename,Base.ParseCsv(MainWindow.filename).Count);

Далее имея массив “ японских свечей” мы можем осуществлять поиск и проверку паттернов по такому алгоритму:

1)Находим в массиве CandleArray Свечу-Паттерн (далее пункт 2.)

2)Проверяем 4 предыдущие свечи на направление тренда, если три из четырех свечей закрыты в одну сторону засчитываем паттерн. (далее пункт 3.)

3)Проверяем 4 следующие свечи на направление тренда, если три из четырех свечей закрыты в противоположную сторону считаем паттерн рабочим, иначе считаем нерабочим.



Рисунок 1. Пример паттерна, который сработал

Для всех алгоритмов kol – 4 предыдущие свечи, kol1– 4 следующие свечи,

Open-число открытия свечи, close – число закрытия свечи, high- наивысшее значение на которое поднималась свеча, low – наименьшее значение на которое опускалась свеча, patternname – имя паттерна, kek – объект для вывода пользователю

**3.2.1** **Описание алгоритма поиска паттернов. Паттерн “Молот”.**

Алгоритм поиска паттерна "Молот" на языке C#.

if ((kol >= 3) && (CandleArray[i].open > CandleArray[i].close))

{

if(((CandleArray[i].open-CandleArray[i].close) < ((CandleArray[i].close - CandleArray[i].low) \* 0.5))&&

((CandleArray[i].high -CandleArray[i].open) < (CandleArray[i].open - CandleArray[i].close)))

{

CandleArray[i].patternName = "Молот";

if (kol1 >= 3)

CandleArray[i].work = true;

else

CandleArray[i].work = false;

PatternsToList kek = new PatternsToList();

kek.patternname = CandleArray[i].patternName;

kek.pattertime = CandleArray[i].time;

if (CandleArray[i].work == true)

kek.Work = "Сработал";

else

kek.Work = "Не Сработал";

pot.Add(kek);

}

}

Первые два “if ” проверяют отдельно взятую свечу на наличие паттерна “молот”, далее смотрим на последующие 4 свечи и если 3 из них закрылись положительно, считаем паттерн успешным.

**3.2.2** **Описание алгоритма поиска паттернов. Паттерн “Падающая звезда”.**

Алгоритм поиска паттерна "Падающая звезда" на языке C#.

if ((kol < 2) && (CandleArray[i].open < CandleArray[i].close))

{

if (((CandleArray[i].close - CandleArray[i].open) < ((CandleArray[i].high - CandleArray[i].close) \* 0.5))

&& ((CandleArray[i].open - CandleArray[i].low) < (CandleArray[i].close - CandleArray[i].open)))

{

CandleArray[i].patternName = "Падающая звезда";

if (kol1 < 2)

CandleArray[i].work = true;

else

CandleArray[i].work = false;

PatternsToList kek = new PatternsToList();

kek.patternname = CandleArray[i].patternName;

kek.pattertime = CandleArray[i].time;

if (CandleArray[i].work == true)

kek.Work = "Сработал";

else

kek.Work = "Не Сработал";

pot.Add(kek);

}

}

Первые два “if ” проверяют отдельно взятую свечу на наличие паттерна “Падающая звезда”, далее смотрим на последующие 4 свечи и если 3 из них закрылись отрицательно, считаем паттерн успешным.

**3.2.3** **Описание алгоритма поиска паттернов. Паттерн “Повешенный”.**

Алгоритм поиска паттерна "Повешанный" на языке C#.

if ((kol < 2) && (CandleArray[i].open > CandleArray[i].close))

{

if (((CandleArray[i].open - CandleArray[i].close) < ((CandleArray[i].close - CandleArray[i].low) \* 0.5))

&& ((CandleArray[i].high - CandleArray[i].open) < (CandleArray[i].open - CandleArray[i].close)))

{

CandleArray[i].patternName = "Повешанный";

if (kol1 < 2)

CandleArray[i].work = true;

else

CandleArray[i].work = false;

PatternsToList kek = new PatternsToList();

kek.patternname = CandleArray[i].patternName;

kek.pattertime = CandleArray[i].time;

if (CandleArray[i].work == true)

kek.Work = "Сработал";

else

kek.Work = "Не Сработал";

pot.Add(kek);

}

}

Первые два “if ” проверяют отдельно взятую свечу на наличие паттерна “Повешанный ”, далее смотрим на последующие 4 свечи и если 3 из них закрылись отрицательно, считаем паттерн успешным.

**3.2.4** **Описание алгоритма поиска паттернов. Паттерн “Дожи”.**

Алгоритм поиска паттерна "Дожи" на языке C#.

if (Math.Abs(CandleArray[i].close - CandleArray[i].open) < 0.02 \* (CandleArray[i].high - CandleArray[i].low))

{

if (((kol1 < 2) && (kol < 2)) || ((kol1 >= 3) && (kol >= 3)))

{

CandleArray[i].patternName = "Дожи";

CandleArray[i].work = true;

}

else

{

CandleArray[i].patternName = "Дожи";

CandleArray[i].work = false;

}

PatternsToList kek = new PatternsToList();

kek.patternname = CandleArray[i].patternName;

kek.pattertime = CandleArray[i].time;

if (CandleArray[i].work == true)

kek.Work = "Сработал";

else

kek.Work = "Не Сработал";

pot.Add(kek);

}

Первые два “if ” проверяют отдельно взятую свечу на наличие паттерна “Дожи ”, далее смотрим на последующие 4 свечи и если 3 из них закрылись в противоположную сторону предыдущим 4, считаем паттерн успешным.

**3.2.5** **Описание алгоритма поиска паттернов. Паттерн “Бычье поглощение”.**

Алгоритм поиска паттерна "Бычье поглощение" на языке C#.

if ((CandleArray[i].open > CandleArray[i].close) && (kol >= 3))

{

if (CandleArray[i + 1].open < CandleArray[i + 1].close)

{

if ((CandleArray[i + 1].close > CandleArray[i].open) && (CandleArray[i + 1].open < CandleArray[i].close))

{

if (kol1 >= 3)

{

if (CandleArray[i].patternName == null)

{

CandleArray[i].patternName = "Бычье поглощение";

CandleArray[i].work = true;

}

else

{

CandleArray[i + 1].patternName = "Бычье поглощение";

CandleArray[i + 1].work = true;

}

}

else

{

if (CandleArray[i].patternName == null)

{

CandleArray[i].patternName = "Бычье поглощение";

CandleArray[i].work = false;

}

else

{

CandleArray[i + 1].patternName = "Бычье поглощение";

CandleArray[i + 1].work = false;

}

}

PatternsToList kek = new PatternsToList();

kek.patternname = CandleArray[i].patternName;

kek.pattertime = CandleArray[i].time;

if (CandleArray[i].work == true)

kek.Work = "Сработал";

else

kek.Work = "Не Сработал";

pot.Add(kek);

}

}

}

Первые четыре “if ” проверяют отдельно взятую свечу на наличие паттерна “Бычье поглощение ”, далее смотрим на последующие 3 свечи и если 2 из них закрылись положительно, считаем паттерн успешным.

**3.2.6** **Описание алгоритма поиска паттернов. Паттерн “Медвежье поглощение”**

Алгоритм поиска паттерна "Медвежье поглощение" на языке C#.

if ((CandleArray[i].open < CandleArray[i].close) && (kol1 >= 3))

{

if (CandleArray[i + 1].open > CandleArray[i + 1].close)

{

if ((CandleArray[i + 1].close < CandleArray[i].open) && (CandleArray[i + 1].open > CandleArray[i].close))

{

if (kol1 < 2)

{

if (CandleArray[i].patternName == null)

{

CandleArray[i].patternName = "Медвежье поглощение";

CandleArray[i].work = true;

}

else

{

CandleArray[i + 1].patternName = "Медвежье поглощение";

CandleArray[i + 1].work = true;

}

}

else

{

if (CandleArray[i].patternName == null)

{

CandleArray[i].patternName = "Медвежье поглощение";

CandleArray[i].work = false;

}

else

{

CandleArray[i + 1].patternName = "Медвежье поглощение";

CandleArray[i + 1].work = false;

}

}

PatternsToList kek = new PatternsToList();

kek.patternname = CandleArray[i].patternName;

kek.pattertime = CandleArray[i].time;

if (CandleArray[i].work == true)

kek.Work = "Сработал";

else

kek.Work = "Не Сработал";

pot.Add(kek);

}

}

}

Первые четыре “if ” проверяют отдельно взятую свечу на наличие паттерна “Бычье поглощение ”, далее смотрим на последующие 3 свечи и если 2 из них закрылись отрицательно, считаем паттерн успешным.

**3.2.7** **Описание алгоритма поиска паттернов. Паттерн “Бычий захват за пояс”.**

Алгоритм поиска паттерна " Бычий захват за пояс" на языке C#.

if ((kol >= 3) && (CandleArray[i].close > CandleArray[i].open))

{

if (((CandleArray[i].close - CandleArray[i].open) > ((CandleArray[i].high - CandleArray[i].close) \* 2)) && ((CandleArray[i].open - CandleArray[i].low) == 0))

{

CandleArray[i].patternName = "Бычий захват за пояс";

if (kol1 >= 3)

CandleArray[i].work = true;

else

CandleArray[i].work = false;

PatternsToList kek = new PatternsToList();

kek.patternname = CandleArray[i].patternName;

kek.pattertime = CandleArray[i].time;

if (CandleArray[i].work == true)

kek.Work = "Сработал";

else

kek.Work = "Не Сработал";

pot.Add(kek);

}

}

Первые два “if ” проверяют отдельно взятую свечу на наличие паттерна “Бычье поглощение ”, далее смотрим на последующие 3 свечи и если 2 из них закрылись положительно, считаем паттерн успешным.

**3.2.8** **Описание алгоритма поиска паттернов. Паттерн “Медвежий захват за пояс”.**

Алгоритм поиска паттерна " Медвежий захват за пояс" на языке C#.

if ((kol < 2) && (CandleArray[i].close < CandleArray[i].open))

{

if (((CandleArray[i].open - CandleArray[i].close) > ((CandleArray[i].close - CandleArray[i].low) \* 2)) && ((CandleArray[i].high - CandleArray[i].open) == 0))

{

CandleArray[i].patternName = "Медвежий захват за пояс";

if (kol1 < 2)

CandleArray[i].work = true;

else

CandleArray[i].work = false;

PatternsToList kek = new PatternsToList();

kek.patternname = CandleArray[i].patternName;

kek.pattertime = CandleArray[i].time;

if (CandleArray[i].work == true)

kek.Work = "Сработал";

else

kek.Work = "Не Сработал";

pot.Add(kek);

}

}

Первые два “if ” проверяют отдельно взятую свечу на наличие паттерна “Бычье поглощение ”, далее смотрим на последующие 3 свечи и если 2 из них закрылись отрицательно, считаем паттерн успешным.

**3.2.9** **Сбор статистики о появлении паттернов и их работе.**

Алгоритм сбора статистики о появлении паттернов на языке C#.

for (int i = 0; i < pot.Length; i++)

{

if (pot[i].patternname == "Молот")

{

KolM++;

if (pot[i].Work == "Сработал")

KolMT++;

}

else if (pot[i].patternname == "Падающая звезда")

{

KolPA++;

if (pot[i].Work == "Сработал")

KolPAT++;

}

else if (pot[i].patternname == "Повешанный")

{

KolPO++;

if (pot[i].Work == "Сработал")

KolPOT++;

}

else if (pot[i].patternname == "Дожи")

{

KolD++;

if (pot[i].Work == "Сработал")

KolDT++;

}

else if (pot[i].patternname == "Бычье поглощение")

{

KolBP++;

if (pot[i].Work == "Сработал")

KolBPT++;

}

else if (pot[i].patternname == "Медвежье поглощение")

{

KolMP++;

if (pot[i].Work == "Сработал")

KolMPT++;

}

else if (pot[i].patternname == "Бычий захват за пояс")

{

KolBZ++;

if (pot[i].Work == "Сработал")

KolBZT++;

}

else if (pot[i].patternname == "Медвежий захват за пояс")

{

KolMZ++;

if (pot[i].Work == "Сработал")

KolMZT++;

}

}

Stat M = new Stat();

M.name = "Молот";

M.quantity = KolM;

M.Work = KolMT;

M.NotWork = KolM - KolMT;

LAK.Add(M);

Stat PA = new Stat();

PA.name = "Падающая звезда";

PA.quantity = KolPA;

PA.Work = KolPAT;

PA.NotWork = KolPA - KolPAT;

LAK.Add(PA);

Stat PO = new Stat();

PO.name = "Повешанный";

PO.quantity = KolPO;

PO.Work = KolPOT;

PO.NotWork = KolPO - KolPOT;

LAK.Add(PO);

Stat D = new Stat();

D.name = "Дожи";

D.quantity = KolD;

D.Work = KolDT;

D.NotWork = KolD - KolDT;

LAK.Add(D);

Stat BP = new Stat();

BP.name = "Бычье поглощение";

BP.quantity = KolBP;

BP.Work = KolBPT;

BP.NotWork = KolBP - KolBPT;

LAK.Add(BP);

Stat MP = new Stat();

MP.name = "Медвежье поглощение";

MP.quantity = KolMP;

MP.Work = KolMPT;

MP.NotWork = KolMP - KolMPT;

LAK.Add(MP);

Stat BZ = new Stat();

BZ.name = "Бычий захват за пояс";

BZ.quantity = KolBZ;

BZ.Work = KolBZT;

BZ.NotWork = KolBZ - KolBZT;

LAK.Add(BZ);

Stat MZ = new Stat();

MZ.name = "Медвежий захват за пояс";

MZ.quantity = KolMZ;

MZ.Work = KolMZT;

MZ.NotWork = KolMZ - KolMZT; LAK.Add(MZ);

В данном куске кода мы обрабатываем паттерны, полученные с помощью алгоритмов поиска. Kol\*\* -количество паттернов определенного вида; Kol\*\*T – количество сработавших паттернов определенного вида; LAK- лист со статистикой; M, BZ, MZ, MP, BP ,D, PO – обьекты собирающие статистику из массива с паттернами.

**3.2.10** **Чтение данных из файла**

List<Base> res = new List<Base>();

try

{

var parser = new Microsoft.VisualBasic.FileIO.TextFieldParser(file);

parser.TextFieldType = Microsoft.VisualBasic.FileIO.FieldType.Delimited;

parser.SetDelimiters(new string[] { ";" });

parser.ReadFields();

while (!parser.EndOfData)

{

string[] row = parser.ReadFields();

Base base1 = new Base();

base1.TICKER = row[0];

base1.PER = row[1];

base1.DATA = new DateTime(int.Parse(row[2].Substring(0, 4)), int.Parse(row[2].Substring(4, 2)), int.Parse(row[2].Substring(6, 2)));

base1.TIME = int.Parse(row[3]);

base1.OPEN = double.Parse(row[4], CultureInfo.InvariantCulture);

base1.HIGH = double.Parse(row[5], CultureInfo.InvariantCulture);

base1.LOW = double.Parse(row[6], CultureInfo.InvariantCulture);

base1.CLOSE = double.Parse(row[7], CultureInfo.InvariantCulture);

res.Add(base1);

kol++;

}

return res;

}

catch

{

return res;

}

Где res – это list считанных из файла данных.

**3.3** **Описание и обоснование метода организации входных данных**

Входные данные получают из стандартного файла истории биржевых котировок формата .csv . Содержащего столбцы: <TICKER>, <PER>, <TIME>,<DATE> , <OPEN> ,<HIGH> ,<LOW> ,<CLOSE>.Где : TICKER – краткое название котируемых инструментов  в биржевой информации (Например GAZP-акции Газпрома), PER - число, обозначающее периодичность баров данных (Например 60 – одна минута), DATE- дата (например 20180103 – 2018.01.03), TIME-время (Например 60 – одна минута), OPEN – число открытия свечи (например 131.0300000), HIGH –максимум на который поднималась свеча (например 137.0350000), LOW- минимум на который опускалась свеча (например 120.0370000), CLOSE – число закрытия свечи (например 134.0300000).

**3.4 Описание и обоснование выбора и состава технических и программных средств**

**3.4.1** **Состав технических и программных средств**

Для работы программы необходим следующий набор программных средств:

1. Операционная система Microsoft Windows 7 или более поздняя версия;
2. установленный Microsoft .NET Framework 4.0, требующий Windows Installer 3.1 или более поздняя версия;

Для работы программы необходим следующий набор технических средств:

1. Персональный компьютер, оснащенный 32-разрядным (x86) или 64-разрядным (x64) процессором Pentium с тактовой частотой 400 MГц и выше или аналогичный процессор (рекомендуется Pentium с тактовой частотой 1 ГГц и выше или аналогичный процессор);
2. Не менее 10 МБ свободной памяти
3. мышь Microsoft Mouse или совместимое указывающее устройство;
4. видеокарта и монитор, с разрешением не менее чем 1080x600 точек;

**3.4.2** **Обоснование выбора состава технических и программных средств**

При создании программы использовался тип List<T>, в котором T – тип элементов в списке, находящийся в пространства имен System.Collections.Generic, который применяется с платформы Microsoft .NET Framework 2.0. Также во время написания программы использовалась библиотека Material design in XAML.

**ОЖИДАЕМЫЕ ТЕХНИКО-ЭКОНОМИЧЕСКИЕ ПОКАЗАТЕЛИ**

**4.1.** **Предполагаемая потребность**

Программа может активно использоваться людьми занятыми в финансовой сфере.

Многие трейдеры перед инвестицией осуществляют анализ и расчет рисков вложения капитала, а данная программа выполняет часть этих функций. Пользователь программы получает статистику появления и работы паттернов на основе данных за предыдущие периоды.

**4.2.** **Экономические преимущества разработки по сравнению с отечественными и  
зарубежными образцами или аналогами**

Поиск в сети Интернет приложения, аналогичного программе FindPatterns, не дал положительного результата.

1. **ИСТОЧНИКИ, ИСПОЛЬЗОВАННЫЕ ПРИ РАЗРАБОТКЕ**
2. Шилдт, Герберт. C# 4.0: полное руководство.: Пер. с англ. — М.: ООО "И.Д. Вильямс", 2011.
3. Книга автора: М.Принг. “О ценовых моделях”-книга по price action и техническому анализу.
4. ГОСТ 19.101-77 Виды программ и программных документов. //Единая система
5. программной документации. – М.: ИПК Издательство стандартов, 2001.
6. ГОСТ 19.102-77 Стадии разработки. //Единая система программной документации.
7. – М.: ИПК Издательство стандартов, 2001.
8. ГОСТ 19.103-77 Обозначения программ и программных документов.
9. система программной документации. – М.: ИПК Издательство стандартов, 2001.
10. ГОСТ 19.104-78 Основные надписи. //Единая система программной документации.
11. – М.: ИПК Издательство стандартов, 2001.
12. ГОСТ 19.105-78 Общие требования к программным документам. //Единая система
13. программной документации. – М.: ИПК Издательство стандартов, 2001.
14. ГОСТ 19.106-78 Требования к программным документам, выполненным печатным
15. способом. //Единая система программной документации. – М.: ИПК Издательство
16. ГОСТ 19.404-79 Пояснительная записка. Требования к содержанию и оформлению.
17. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
18. ГОСТ 19.603-78 Общие правила внесения изменений. //Единая система
19. программной документации. – М.: ИПК Издательство стандартов, 2001.
20. ГОСТ 19.604-78 Правила внесения изменений в программные документы.

# ПРИЛОЖЕНИЕ 1. ОПИСАНИЕ И ФУНКЦИОНАЛЬНОЕ НАЗНАЧЕНИЕ КЛАССОВ МЕТОДОВ И ПОЛЕЙ

## Описание и функциональное назначение классов

Таблица 1 – Описание и функциональное назначение классов

|  |  |
| --- | --- |
| Класс | Назначение |
| Base | Класс, отвечающий за чтение вводимых данных |
| Candle | Класс, отвечающий за составление структуры свечи. |
| MainWindow | Класс, связывающий все элементы с главным окном |
| PatternsToList | Класс, отвечающий за составление структуры паттерна. |
| Stat | Класс, отвечающий за сбор статистики работы паттернов. |
| UserControlChart | Класс отвечающий за отрисовку графика |
| UserControlCreate | Класс отвечающий за просмотр загруженных данных |
| UserControlHome | Класс реализующий алгоритмы поиска паттернов японских свечей |
| UserControlStat | Класс собирающий статистику по найденным паттернам |

Описание полей, методов Base

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Base | | | | | |
| **Поля** | | | | | |
| Имя | Модификатор доступа | Тип | Описание | | |
| TICKER | public | string | Тикер | | |
| PER | string | Период торгов | | |
| DATA | DateTime | Дата торгов | | |
| TIME | int | Время торгов | | |
| OPEN | double | Число открытия свечи | | |
| HIGH | double | Наибольшая цена за этот период | | |
| LOW | double | Наименьшая цена за этот период | | |
| CLOSE | double | Число закрытия свечи | | |
| **Методы** | | | | | |
| Имя | Модификатор доступа | Тип | | Аргументы | Назначение |
| ParseCsv | public | List<Base> | | string file | Чтение файла |
| ParseCandle |  | Candle[] | | String file , int k | Создание массива свечей |

Описание полей, методов Candle

|  |  |  |  |
| --- | --- | --- | --- |
| Candle | | | |
| **Поля** | | | |
| Имя | Модификатор доступа | Тип | Описание |
| time | public | DateTime | Время свечи |
| open | double | Открытие свечи |
| high | double | Высшая точка свечи |
| close | double | Закрытие свечи |
| low | double | Минимум |
| volume | string | Имя для распознования |
| patternName | string | Название паттерна |
| work | bool | Сработал ли паттерн |

Описание полей, методов MainWindow

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| MainWindow | | | | |
| **Поля** | | | | |
| Имя | Модификатор доступа | Тип | Описание | |
| filename | public | string | Имя считываемого файла | |
| Bases | List<Base> | Период торгов | |
| **Методы** | | | | |
| Имя | Модификатор доступа | Тип | Аргументы | Назначение |
| ButtonOpenMenu\_Click | public | void | object sender, RoutedEventArgs e | Элемент графической визуализации |
| ButtonCloseMenu\_Click | public | void | object sender, RoutedEventArgs e | Элемент графической визуализации |
| ListViewMenu\_SelectionChanged | public | void | object sender, RoutedEventArgs e | Элемент графической визуализации |
| Button\_Click | public | void | object sender, RoutedEventArgs e | Графический элемент- закрытие программы |
| Button\_Click\_1 | public | void | object sender, RoutedEventArgs e | Графический элемент-свертка программы |

Описание полей, методов PatternsToList

|  |  |  |  |
| --- | --- | --- | --- |
| Base | | | |
| **Поля** | | | |
| Имя | Модификатор доступа | Тип | Описание |
| patternname | public | string | Имя паттерна |
| pattertime | DateTime | Период и дататоргов |
| Work | string | Сработал ли паттерн |

Описание полей, методов Stat

|  |  |  |  |
| --- | --- | --- | --- |
| Base | | | |
| **Поля** | | | |
| Имя | Модификатор доступа | Тип | Описание |
| name | public | string | Имя паттерна |
| quantity | int | Количество паттернов |
| Work | int | Количество рабочих |
| NotWork | int | Количество нерабочих |

Описание полей, методов UserControlChart

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| UserControlChart | | | | |
| **Поля** | | | | |
| Имя | Модификатор доступа | Тип | Описание | |
| Bases | public | List<Base> | List с обьектами считанными из файла | |
| **Методы** | | | | |
| Имя | Модификатор доступа | Тип | Аргументы | Назначение |
| CreateChart | public | void |  | метод создающий чарт |
| buttonRew\_Click | public | void | object sender, RoutedEventArgs e | кнопка delete |
| StartPaintChart | private | void |  | метод вызывающийся в новом потоке, для прорисовки графика |
| LoadCandleOnChart | private | void |  | прогрузить загруженные свечки на график |
| LoadNewCandle | private | void | Candle newCandle,  int numberInArray | добавить одну свечу на график |
| ChartResize | private | void |  | устанавливает границы представления по оси У |
| GetMinValueOnChart | private | double | Candle[] Book, int start, int end | берёт минимальное значение из массива свечек |
| GetMaxValueOnChart | public | double | Candle[] Book, int start, int end | Графический элемент-свертка программы |
| Rewind | private | void |  | Перемотка, чтобы свечи успевали отрисовываться |

Описание полей, методов UserControlCreate

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| UserControlCreate | | | | |
| **Поля** | | | | |
| Имя | Модификатор доступа | Тип | Описание | |
| Bases | public | List<Base> | List с обьектами считанными из файла | |
| **Методы** | | | | |
| Имя | Модификатор доступа | Тип | Аргументы | Назначение |
| Data\_Show | public | void |  | Метод показывающий считанные данные в виде таблицы |

Описание полей, методов UserControlStat

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| UserControlStat | | | | |
| **Поля** | | | | |
| Имя | Модификатор доступа | Тип | Описание | |
| LAK | private | List<Stat> | Лист со статистикой появления паттернов | |
| CandleArray1 | public | Candle[] | Массив свечек | |
| **Методы** | | | | |
| Имя | Модификатор доступа | Тип | Аргументы | Назначение |
| FindStat | public | List<Stat> | Candle[] CandleArray | Метод возвращающий list заполненный статистикой появления паттернов |

Описание полей, методов UserControlHome

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| UserControlHome | | | | |
| **Поля** | | | | |
| Имя | Модификатор доступа | Тип | Описание | |
| CandleArray | public | Candle[] | Массив свечек | |
| **Методы** | | | | |
| Имя | Модификатор доступа | Тип | Аргументы | Назначение |
| FindPatterns | public | List<PatternsToList> | Candle[] CandleArray | Метод возвращающий list заполненный паттернами |
| UserControlHome | public | конструктор |  | Выводит метод findpatterns() |

# 

# ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Лист регистрации изменений | | | | | | | | | |
| Номера листов (страниц) | | | | | Всего листов (страниц в докум.) | № документа | Входящий № сопроводительного докум. и дата | Подп. | Дата |
| Изм. | Измененных | Замененных | Новых | Аннулированх |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |