**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ  
федеральное государственное автономное образовательное   
учреждение высшего образования   
«Санкт-Петербургский политехнический университет Петра Великого»  
уНИВЕРСИТЕТСКИЙ ПОЛИТЕХНИЧЕСКИЙ КОЛЛЕДЖ**

|  |  |
| --- | --- |
|  | **Допустить к защите** Заместитель директора  по УМР \_\_\_\_\_\_\_\_\_\_\_\_ Е.Г. Конакина\_  (Подпись) (ФИО)  «\_\_\_»\_\_\_\_\_\_\_\_\_\_ 2017 г. |
|  |

**курсовой проект**Тема «Игра с графическим интерфейсом BANG»

специальность 09.02.03 группа 32928/2

Студент (ка) \_\_\_\_\_\_\_\_\_\_\_\_  
 (подпись) (ФИО)  
Преподаватель Девятко Н.С.

(подпись) (ФИО)

Санкт-Петербург  
2018

Содержание

[ВВЕДЕНИЕ 3](#_Toc532064176)

[1 ТЕОРЕТИЧЕСКИЕ ОСНОВЫ РАЗРАБОТКИ 4](#_Toc532064177)

[1.1 Описание предметной области 4](#_Toc532064178)

[1.2 Анализ методов решения 6](#_Toc532064179)

[1.3 Обзор средств программирования 6](#_Toc532064180)

[1.4 Описание языка C# 7](#_Toc532064181)

[2 ПРАКТИЧЕСКАЯ ЧАСТЬ 11](#_Toc532064182)

[2.1 Постановка задачи 11](#_Toc532064183)

[2.1.1 Основания для разработки 11](#_Toc532064184)

[2.1.2 Назначение программы 11](#_Toc532064185)

[2.2 Проектирование приложения 11](#_Toc532064186)

[2.2.1 Диаграмма прецедентов 11](#_Toc532064187)

[2.2.2 Диаграмма последовательностей 12](#_Toc532064188)

[2.2.3 Диаграмма классов 14](#_Toc532064189)

[2.3 Текст программы 14](#_Toc532064190)

[2.4 Описание программы 15](#_Toc532064191)

[2.4.1 Общие сведения 15](#_Toc532064192)

[2.4.2 Функциональное назначение 15](#_Toc532064193)

[2.4.3 Описание логической структуры 15](#_Toc532064194)

[2.4.4 Используемые технические и программные средства 17](#_Toc532064195)

[2.4.5 Вызов и загрузка 17](#_Toc532064196)

[2.5 Руководство оператора 18](#_Toc532064197)

[2.5.1 Назначение программы 18](#_Toc532064198)

[2.5.2 Выполнение программы и сообщения оператору 18](#_Toc532064199)

[2.6 Программа и методика испытаний 19](#_Toc532064200)

[2.6.1 Объект испытаний 19](#_Toc532064201)

[2.6.2 Цель испытаний 19](#_Toc532064202)

[2.6.3 Требования к программе 19](#_Toc532064203)

[2.6.4 Требования к программной документации 20](#_Toc532064204)

[2.6.5 Средства и порядок испытаний 20](#_Toc532064205)

[2.6.6 Методы испытаний 21](#_Toc532064206)

[2.7 Протокол испытаний 24](#_Toc532064207)

[ЗАКЛЮЧЕНИЕ 26](#_Toc532064208)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 27](#_Toc532064209)

[ПРИЛОЖЕНИЕ А 28](#_Toc532064210)

[ПРИЛОЖЕНИЕ Б 34](#_Toc532064211)

[ПРИЛОЖЕНИЕ В 38](#_Toc532064212)

[ПРИЛОЖЕНИЕ Г 42](#_Toc532064213)

# ВВЕДЕНИЕ

В настоящее время компьютеры широко применяются в самых разных областях: науке, промышленности, медицине, образовании. Применение компьютеров в системе образования значительно облегчает труд преподавателя, так как позволяет быстрее и легче проверить знания учащихся, подготовить для них обучающие и тестовые задания. Применение компьютеров способствует также повышению уровня знаний студентов. Большое количество игровых программ способствует повышению интереса пользователей к компьютерным технологиям. Многие игры носят не только развлекательный характер: они повышают скорость реакции, развивают память, внимание, логическое мышление, скорость принятия решения в нестандартных ситуациях.

Разработанная в данном курсовом проекте программа может быть использована в развлекательных целях.

# 1 ТЕОРЕТИЧЕСКИЕ ОСНОВЫ РАЗРАБОТКИ

## 1.1 Описание предметной области

Компью́терная игра́ (также — видеоигра) — компьютерная программа, служащая для организации игрового процесса (геймплея), связи с партнёрами по игре, или сама выступающая в качестве партнёра.

Компьютерные игры могут создаваться на основе фильмов и книг; есть и обратные случаи. С 2011 года компьютерные игры официально признаны в США отдельным видом искусства.

Компьютерные игры могут быть классифицированы по нескольким признакам:

**Жанр:** игра может принадлежать как к одному, так и к нескольким жанрам, а в уникальных случаях — открывать новый или быть вне всяких жанров;

**Количество игроков и способ их взаимодействия:** игра может быть однопользовательской  — рассчитанной на игру одного человека, или многопользовательской — рассчитанной на одновременную игру нескольких человек; а также вестись на одном компьютере, через интернет, электронную почту, или массово;

**Визуальное представление:** игра может как использовать графические средства оформления, так и напротив, быть текстовой. Игра также может быть двухмерной или трехмерной. Есть и звуковые игры — в них вместо визуального представления используются звуки.

**Платформа:** игра может принадлежать как к одной платформе, так и быть мультиплатформенной.

Жанр определяется целью игры. Выделяют следующие жанры:

* *Приключенческая игра* (Adventure) — игра, обладающая полноценным литературным сюжетом, и игрок в процессе игры сам раскрывает все перипетии этого сюжета.
* *Боевик* (Action) — игра, состоящая, в основном, из боевых сцен, драк и перестрелок. Боевики подразделяются на:
  + «шутеры» (от англ. *shoot — стрелять*) (пример: Counter-Strike, серии игр Call of Duty и Battlefield);
  + «файтинг» (от англ. *fight — драться*) (пример: Mortal Kombat, Street Fighter);
  + «ужасы» (примеры: серия игр F.E.A.R., Silent Hill, Resident Evil);
  + «стелс» (от англ. *stealth — скрытный*) (пример: серии игр Hitman, Metal Gear Solid и Splinter Cell).
* *Ролевая игра* (RPG — англ. *Role Playing Game*) — игра, отличительной особенностью которой является наличие у персонажей определённых навыков и характеристик, которые можно обрести, а впоследствии развивать, выполняя какие-либо действия. К этому жанру относятся и многопользовательские ролевые игры, которые, в отличие от однопользовательских, не имеют ни конечной цели, ни законченного сюжета. Примеры: The Elder Scrolls (серия), Gothic (серия)
* *Стратегическая игра* (Strategy) — игра, представляющая собой управление масштабными процессами, как, например, строительство городов, ведение бизнеса, командование армией и т. д. Игровой процесс может идти как в реальном времени (RTS — real time strategy), так и в пошаговом режиме (TBS — turn based strategy).
* *Компьютерный симулятор* (Simulator) — игра, полностью имитирующая какую-либо область реальной жизни, например, имитация управления гоночным автомобилем или самолётом.
* *Головоломка* (Puzzle) — игра, полностью или более чем наполовину состоящая из решения различных логических задач и головоломок.
* *Образовательная игра* — игра, включающая в себя элементы обучающих программ, которые подаются через сам игровой процесс и, благодаря повышению интереса к ним в связи с необычным антуражем, впоследствии хорошо запоминаются.
* *Забавы* — игры, рассчитанные, в основном, на детей, где психологическое впечатление от происходящего на экране гораздо важнее самого процесса игры — например, вид лопающихся пузырьков.

Жанровая классификация компьютерных игр предполагает введение нескольких дополнительных оснований:

* динамика — игровой процесс может происходить в условиях «реального времени» или пошагово;
* перспектива — игра может вестись как от первого, так и от третьего лица.

По некоторым компьютерным играм проводятся любительские и профессиональные [соревнования](https://ru.wikipedia.org/wiki/%D0%A1%D0%BE%D1%80%D0%B5%D0%B2%D0%BD%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) — такого рода соревнования называются [киберспортом](https://ru.wikipedia.org/wiki/%D0%9A%D0%B8%D0%B1%D0%B5%D1%80%D1%81%D0%BF%D0%BE%D1%80%D1%82).

## 1.2 Анализ методов решения

Жанр моей игры можно определить как симулятор игрового автомата. Игровой автомат «BANG» может быть реализован следующими способами:

1. Ручной способ: в виде настольной игры. Этот способ требует времени на подготовку материалов игры, требует подсчёта очков вручную и является не столь зрелищным и интересным.
2. Автоматизированный способ: исходные данные вводятся пользователем вручную, а результат игры (очки, рекорды) рассчитывается программой на компьютере.
3. Автоматический способ: исходные данные автоматически считываются с внешнего устройства (например, с файла на диске) и обрабатываются программой.

Автоматический способ не может быть использован, так как игра предполагает участие человека.

В нашем случае мы рассмотрим автоматизированный способ.

## 1.3 Обзор средств программирования

Требования к курсовому проекту включали удобный пользовательский интерфейс, поэтому при написании данного курсового проекта моя задача состояла в выборе между Java, C#, Delphi.

Java я не стал выбирать, так как из этих трех доступных мне языков программирования, я его знаю меньше всего.

Сравнивать C# и Delphi можно по разным характеристикам:

1) Удобство среды программирования. Это, безусловно, важный пункт, так как от удобства среды программирования зависит производительность программиста. В этом пункте я отдаю предпочтение C#. Удобный, понятный и приятный интерфейс его среды программирования, всплывающие подсказки, подсвечиваемый код очень помогают в работе.

2) Синтаксис, семантика языка программирования. Принципы, правила написания кода, используемые ключевые слова опять же делают C# более привлекательным вариантом.

Есть еще разные характеристики для сравнения такие как:  
 Типизация (например, поддерживается ли неявная типизация, неявное приведение типов без потери данных и др.), вопросы Компиляции (возможность компиляции, условная компиляция и др.), вопросы управления памятью (возможность создавать объекты на стеке, сборка мусора и т.д.) и т.д.

Но их я рассматривать подробно не буду так при решении этой задачи они либо не используются, либо поддерживаются обеими средами.

Тем самым я сравнивал эти два языка не столько по функциональным характеристикам, сколько по удобству работы с ним.

Таким образом, средством программирования данного курсового проекта был выбран язык С#, среда разработки Visual Studio 2017.

## 1.4 Описание языка C#

На сегодняшний момент язык программирования C# одни из самых мощных, быстро развивающихся и востребованных языков в ИТ-отрасли. В настоящий момент на нем пишутся самые различные приложения: от небольших десктопных программок до крупных веб-порталов и веб-сервисов, обслуживающих ежедневно миллионы пользователей.

По сравнению с другими языками C# достаточно молодой, но в то же время он уже прошел большой путь. Первая версия языка вышла вместе с релизом Microsoft Visual Studio .NET в феврале 2002 года. Текущей версией языка является версия C# 6.0, которая вышла в 20 июля 2015 года вместе с Visual Studio 2015.

C# является языком с Си-подобным синтаксисом и близок в этом отношении к C++ и Java. Поэтому, если вы знакомы с одним из этих языков, то овладеть C# будет легче.

C# является объектно-ориентированным и в этом плане много перенял у Java и С++. Например, C# поддерживает полиморфизм, наследование, перегрузку операторов, статическую типизацию. Объектно-ориентированный подход позволяет решить задачи по построению крупных, но в тоже время гибких, масштабируемых и расширяемых приложений. C# продолжает активно развиваться, и с каждой новой версией появляется все больше интересных функциональностей, как, например, лямбды, динамическое связывание, асинхронные методы и т.д.

**Ключевые слова C#:**

|  |  |  |  |
| --- | --- | --- | --- |
| abstract | extern | null | struct |
| as | false | object | switch |
| base | finally | operator | this |
| bool | fixed | out | throw |
| break | float | override | true |
| byte | for | params | try |
| case | foreach | private | typeof |
| catch | goto | protected | uint |
| const | if | public | ulong |
| continue | implicit | readonly | unchecked |
| decimal | in | ref | unsafe |
| default | int | return | ushort |
| delegate | interface | sbyte | using |
| do | internal | sealed | virtual |
| double | is | short | void |
| else | lock | sizeof | while |
| enum | long | stackalloc |  |
| event | namespace | static |  |
| explicit | new | string |  |

**Контекстные ключевые слова C#**

|  |  |  |  |
| --- | --- | --- | --- |
| add | equals | join | set |
| ascending | from | let | value |
| async | get | on | var |
| await | global | orderby | where |
| by | group | partial | yield |
| descending | in | remove |  |
| dynamic | into | select |  |

**Основные алгоритмические конструкции C#**

|  |  |
| --- | --- |
| Оператор присваивания | string hello = "hello " + "world"; //результат равен "hello world"  int x1 = 2 + 4; // результат равен 6  int x2 = 10 - 6; //результат равен 4  int x3 = 10 \* 6; //результат равен 60  double x4 = 10.0 / 4.0; //результат равен 2.5  double x5 = 10.0 % 4.0; //результат равен 2  int y1 = 5;  int z1 = ++y1; // z1=6; y1=6  int y2 = 5;  int z2 = y2++; // z2=5; y2=6  int y3 = 5;  int z3 = --y3; // z3=4; y3=4  int y4 = 5;  int z4 = y4--; // z4=5; y4=4 |
| Условный оператор | **if**  (*условие)* **{**(*действие)* } **else** {(*альтернатива)}* ;  int num1 = 8;  int num2 = 6;  if(num1 > num2)  {      Console.WriteLine("Число {0} больше числа {1}", num1, num2);  }  else  {      Console.WriteLine("Число {0} меньше числа {1}", num1, num2);  } |
|  | *Конструкция switch/case аналогична конструкции if/else, так как позволяет обработать сразу несколько условий:*  Console.WriteLine("Нажмите Y или N");  string selection = Console.ReadLine();  switch (selection)  {      case "Y":          Console.WriteLine("Вы нажали букву Y");          break;      case "N":          Console.WriteLine("Вы нажали букву N");          break;      default:          Console.WriteLine("Вы нажали неизвестную букву");          break;  }  *После ключевого слова switch в скобках идет сравниваемое выражение. Значение этого выражения последовательно сравнивается со значениями, помещенными после оператора сase. И если совпадение будет найдено, то будет выполняться определенный блок сase.*  *В конце блока сase ставится оператор break, чтобы избежать выполнения других блоков.*  *Если мы хотим также обработать ситуацию, когда совпадения не будет найдено, то можно добавить блок default, как в примере выше.* |
| Арифметический цикл  **(**применяется, когда известно количество повторений цикла) | *for ([инициализация счетчика]; [условие]; [изменение счетчика])*  for (int i = 0; i < 9; i++)  {      Console.WriteLine("Квадрат числа {0} равен {1}", i, i \* i);  } |
| Цикл с предусловием  **(**применяется, когда неизвестно количество повторений цикла) | Этот цикл будет выполняться до тех пор, пока истинно *условие* (логическое выражение, возвращающее значение типа **Boolean**). При этом если это выражение сразу равно **false**, *тело цикла* не будет выполнено ни разу. Нужно очень внимательно следить за написанием *условия* и контролем завершения цикла, так как в результате ошибки цикл **while** будет повторяться бесконечное количество раз, что приведёт к "зацикливанию" и "зависанию" программы. |
| Цикл с постусловием  **(**применяется, когда неизвестно количество повторений цикла) | *do { тело цикла } while условие;*  Повторения сначала выполняет *тело цикла*, а затем уже проверяет выполнение *условия*: Таким образом, этот вариант цикла гарантирует, что *тело цикла* будет выполнен по крайней мере один раз. И будет выполняться до тех пор, пока *условие* не станет истинным (т.е. **true**). |

# 2 ПРАКТИЧЕСКАЯ ЧАСТЬ

## 2.1 Постановка задачи

### 2.1.1 Основания для разработки

Разработка ведётся на основании задания к курсовому проекту по профессиональному модулю ПМ.01 «Разработка программных модулей программного обеспечения для компьютерных систем» МДК 01.02 «Прикладное программирование» и утверждена Университетским политехническим колледжем.

### 2.1.2 Назначение программы

Игровая программа с графическим интерфейсом «Bang»: человек играет против машины; машина прокручивает картинки в случайном порядке по типу работы игровых автоматов, если три картинки из трёх совпадают, происходит повышение ставки.

Полный текст технического задания приведен в приложении А.

## 2.2 Проектирование приложения

На этапе проектирования были разработаны диаграмма прецедентов, диаграмма последовательностей и диаграмма классов.

### 2.2.1 Диаграмма прецедентов

Диаграммы прецедентов частично описывает use case – прецедент использования проектируемой системы, давая частичное описание частичного применения системы с точки зрения условного внешнего обозревателя (за которым – в идеале - стоит согласованная точка зрения участников работ). При этом описание фокусируется на том, что должна делать система по отношению к своему внешнему окружению (периферии), а не то на том, как она эта делает, то есть диаграмма прецедентов есть частичная спецификация (рисунок 1).



Рисунок 1 Диаграмма прецедентов

### 2.2.2 Диаграмма последовательностей

Диаграмма последовательности отражает поток событий, происходящих в рамках варианта использования.

Все действующие лица показаны в верхней части диаграммы. Стрелки соответствуют сообщениям, передаваемым между действующим лицом и объектом или между объектами для выполнения требуемых функций. На диаграмме последовательности объект изображается в виде прямоугольника, от которого вниз проведена пунктирная вертикальная линия. Эта линия называется линией жизни (lifeline) объекта. Она представляет собой фрагмент жизненного цикла объекта в процессе взаимодействия.



Рисунок 2 Диаграмма последовательностей

### 2.2.3 Диаграмма классов

На рисунке 3 показана диаграмма классов проекта. Программа содержит 2 класса: Form1-класс формы игры, класс Login.cs содержит методы работы с бинарным файлом логинов и результатов. Этот класс использует модуль Form1 во время авторизации и при отображении таблицы рекордов.



Рисунок 3 Диаграмма классов

## 2.3 Текст программы

Текст программы в соответствии с ГОСТ 19.101-77 (СТ СЭВ 1626-79) и ГОСТ 19.401-79 (СТ СЭВ 3746-82) представляет собой запись программы на исходном языке программирования с необходимыми комментариями. Текст программы представляет собой документ, выполненный машинным способом, и приведен в приложении Г.

## 2.4 Описание программы

### 2.4.1 Общие сведения

Игровая программа с графическим интерфейсом «Bang»: человек играет против машины; машина прокручивает картинки в случайном порядке по типу работы игровых автоматов, если три картинки из трёх совпадают, происходит повышение ставки.

### 2.4.2 Функциональное назначение

Основное назначение программного продукта заключается в организации игры с графическим интерфейсом «Bang»: человек играет против машины; машина прокручивает картинки в случайном порядке по типу работы игровых автоматов, если три картинки из трёх совпадают, происходит повышение ставки.

### 2.4.3 Описание логической структуры

Программа содержит 2 модуля:

Form1-модуль формы игры, класс Login.cs содержит методы работы с бинарным файлом логинов и результатов. Этот класс использует модуль Form1 во время авторизации и при отображении таблицы рекордов.

Схемы отдельных методов приведены в приложении В.

#### 2.4.3.1 Описание методов класса Login.cs

Класс Login предназначен для организации работы с бинарным файлом. Связность модуля функциональная (СС=10), так как он направлен на решение одной проблемы.

Модуль содержит 3 поля password , record и login, и соответствующие им свойства для чтения и установки значений полей, и 4 метода:

public void LoginToFile(string filename) метод для записи нового логина в файл;

public void RecordToFile(string filename) метод для записи рекорда в файл (Рис. 8);

public void RecordToTable(string filename, DataGridView dg) метод для вывода рекордов в таблицу (Рис. 9);

public int FindLogin(string filename) метод для поиска логина в файле при авторизации (Рис. 10).

#### 2.4.3.2 Описание методов класса Form1.cs

Класс формы Form1.cs предназначен для организации игры, данная форма является главной. Связность модуля информационная (последовательная) (СС=9), так как действия внутри модуля связаны с данными, а также важна последовательность действий.

Модуль содержит 10 методов:

public void Music() предназначен для проигрывания звукового файла;

private void Exit\_Click(object sender, EventArgs e) предназначен для завершения игры и содержит команду закрытия главной формы;

private void Exit \_MouseMove(object sender, MouseEventArgs e) — метод обработки события перемещения мыши над кнопкой и private void Exit\_MouseLeave(object sender, EventArgs e) — метод обработки события мышь покидает область компонента (кнопку) служат для изменения изображения на кнопке, что позволяет менять внешний вид кнопки;

private void Start\_Click(object sender, EventArgs e) метод нажатия на кнопку Start запускает таймер 1 (для прокрутки картинок);

private void timer1\_Tick(object sender, EventArgs e) метод работы таймера 1 (Рис. 11) в течении 30 секунд меняет картинки случайным образом и затем проверяет совпадают ли картинки при остановке «барабана» (таймера) и изменяет игровую сумму очков: при совпадении 3 картинок к игровой сумме добавляется 500 очков, при выпадении трёх семёрок добавляется 1000 очков, в противном случае игровая сумма уменьшается на 100 очков;

private void Form1\_Load(object sender, EventArgs e) метод обработки события загрузки формы служит для загрузки изображений в список, скрытия картинок и отображения панели для авторизации;

private void Vabank\_Click(object sender, EventArgs e) метод нажатия на кнопку Vabank запускает таймер 2 (для прокрутки картинок);

private void timer2\_Tick(object sender, EventArgs e) метод работы таймера 2 аналогичен методу для таймера 1, а отличается другим способом изменения очков: при совпадении 3 картинок сумма увеличивается в 5 раз, при выпадении трёх семёрок сумма увеличивается в 10 раз, а в остальных случаях сумма аннулируется;

private void Continue\_Click(object sender, EventArgs e) метод нажатия на кнопку Continue служит для проверки логина и пароля при авторизации.

Программа использует функции следующих библиотек среды С#:

System, System.Collections.Generic,System.ComponentModel, System.Data, System.Drawing,System.Linq, System.Text, System.Windows.Forms, System.IO, System.Runtime.InteropServices.

Исполняемый файл программы создан средствами среды C#, имеет имя   
Bang.exe и размер 16 751 байт.

### 2.4.4 Используемые технические и программные средства

Для нормального функционирования данной информационной системы необходим компьютер, клавиатура, мышь и следующие технические средства:

- процессор Intel или другой совместимый;

- объем свободной оперативной памяти ~500 Кб;

- объем необходимой памяти на жестком диске ~3Мб;

- стандартный VGA-монитор или совместимый;

- стандартная клавиатура;

- манипулятор «мышь».

### 2.4.5 Вызов и загрузка

Программа может быть загружена как с диска, так и с жесткого диска. В последнем случае требуется предварительно переписать программу с диска на жесткий диск.

Исполняемым файлом программы является файл Bang.exe. Для его запуска необходимо дважды щелкнуть по исполняемому файлу левой кнопкой мышки.

## 2.5 Руководство оператора

### 2.5.1 Назначение программы

Основное назначение программного продукта заключается в организации игры с графическим интерфейсом «Bang»: человек играет против машины; машина прокручивает картинки в случайном порядке по типу работы игровых автоматов, если три картинки из трёх совпадают, происходит повышение ставки.

### 2.5.2 Выполнение программы и сообщения оператору

Для запуска программы дважды щелкните левой кнопкой мыши по исполняемому файлу Bang.exe.

Процесс игры, этапы работы приложения показаны в виде иллюстраций в приложении Б.

При запуске работы приложения загружается главная форма, на которой появляется панель для ввода логина и пароля. Игровые картинки скрываются до окончания авторизации. При вводе действующего логина и пароля происходит авторизация, панель авторизации скрывается, и становятся доступными кнопки Start и Vabank. Пользователь может начать игру, нажав на одну из этих кнопок. Перед началом игры пользователю дается начальная игровая сумма 1000 очков. При каждом нажатии на кнопку Start происходит «прокручивание» картинок, при их остановке программой проверяется совпадение картинок и изменение игровой суммы. При совпадении 3 картинок к игровой сумме добавляется 500 очков, при выпадении трёх семёрок добавляется 1000 очков, в противном случае игровая сумма уменьшается на 100 очков.

Иначе работает кнопка Vabank: при совпадении 3 картинок сумма увеличивается в 5 раз, при выпадении трёх семёрок сумма увеличивается в 10 раз, а в остальных случаях сумма аннулируется. Игра заканчивается, если игровая сумма становится равной нулю. При этом сразу выводится таблица рекордов предыдущих сеансов игры разных пользователей. Рекордом считается максимальное количество очков, которое смог набрать пользователь во время сеанса игры. Новый рекорд пользователя записывается вместо предыдущего, если во время сеанса игры удалось превысить предыдущий рекорд.

## 2.6 Программа и методика испытаний

### 2.6.1 Объект испытаний

Объектом испытаний является игровая программа Bang.exe. Человек играет против машины; машина прокручивает картинки в случайном порядке по типу работы игровых автоматов, если три картинки из трёх совпадают, происходит повышение ставки.

### 2.6.2 Цель испытаний

Целью испытаний является проверка соответствия программы требованиям Технического Задания.

### 2.6.3 Требования к программе

В процессе испытаний подлежат проверке следующие требования к программе:

#### 2.6.3.1 Требования к функциональным характеристикам

- программа должна обеспечивать авторизацию пользователя (ввод логина и пароля), создание новых аккаунтов для игры;

- программа должна прокручивать несколько картинок в случайном порядке, в течении 30 секунд, и демонстрировать это на экране;

- программа должна обеспечивать проверку на совпадение картинок;

- программа должна вести подсчёт ставки игры, при совпадении картинок ставка увеличивается на 500, при выпадении трёх семёрок ставка увеличивается на 1000, при несовпадении картинок, ставка уменьшается на 100;

- программа должна иметь функцию “va bank” с возможностью поставить всю ставку, в случае совпадения всех картинок ставка увеличивается в несколько раз;

- игра завершает свою работу в случае закрытия игрового окна вручную, или проигрыше всей ставки;

- программа должна сохранять рекорд пользователя;

- программа должна обеспечивать просмотр таблицы рекордов различных пользователей.

#### 2.6.3.2 Требования к информационной и программной совместимости

Для полноценного функционирования данной системы необходимо наличие операционной системы выше Microsoft Windows 7 или совместимой. Язык интерфейса – русский.

#### 2.6.3.3 Требования к маркировке и упаковке

Программа должна поставляться на диске в виде исполняемого (еxе) файла, документации и проекта. На диске должна быть наклейка с надписью "Игра с графическим интерфейсом «Bang»". Диск должна быть упакован в пластиковую коробку.

### 2.6.4 Требования к программной документации

На испытания должны быть представлены следующие программные документы:

* техническое задание
* текст программы
* описание программы
* руководство оператора
* описание языка

### 2.6.5 Средства и порядок испытаний

Для проведения испытаний необходимы следующие технические средства:

* процессор Intel или другой совместимый;
* объем свободной оперативной памяти ~500 Кб;
* объем необходимой памяти на жестком диске ~3Мб;
* стандартный VGA-монитор или совместимый;
* стандартная клавиатура;
* манипулятор «мышь».

Для проведения испытаний необходимы следующие программные средства:

Операционная система Windows 7.

Испытания проводятся в следующем порядке:

1) проверяется наличие и комплектность программной документации (п.2.6.4)

2) проверяется соответствие требованиям к маркировке и упаковке (п.2.6.3.3)

3) проверяется соответствие требованиям к функциональным характеристикам (п.2.6.3.1)

4) проверяется соответствие требованиям к информационной и программной совместимости (п.2.6.3.2)

### 2.6.6 Методы испытаний

#### 2.6.6.1 Для проверки способности программы осуществлять авторизацию пользователя по введённому логину и паролю, необходимо:

* запустить программу
* ввести действующий логин и пароль в поля авторизации
* нажать кнопку «continue»
* убедиться, что скрылись поля для авторизации и стала доступной кнопка «Start» (авторизация произошла).

#### 2.6.6.2 Для проверки способности программы обеспечивать добавление новых пользователей (регистрацию), необходимо:

* запустить программу
* ввести новый логин и пароль в поля авторизации
* убедиться, что появилось сообщение «Save a new login?»
* в диалоговом окне нажать кнопку «Да»
* убедиться, что скрылись поля для авторизации и стала доступной кнопка «Start» (авторизация произошла).

#### 2.6.6.3 Для проверки способности программы прокручивать несколько картинок в случайном порядке, в течении 30 секунд, и демонстрировать это на экране, необходимо:

* запустить программу
* ввести действующий логин и пароль в поля авторизации
* нажать кнопку «continue»
* нажать кнопку «Start»
* убедиться, что картинки прокручиваются в случайном порядке в течение 10 секунд, а затем останавливаются.

#### 2.6.6.4 Для проверки способности программы обеспечивать проверку на совпадение картинок и вести подсчёт ставки игры, необходимо:

* запустить программу
* ввести действующий логин и пароль в поля авторизации
* нажать кнопку «continue»
* нажать кнопку «Start»
* убедиться, что при совпадении картинок очки увеличиваются на 500, а при выпадении 3-х семёрок на 1000 очков; при несовпадении картинок очки должны уменьшаться на 100.

#### 2.6.6.5 Для проверки способности программы обеспечивать функцию “Vabank” с возможностью поставить всю ставку, необходимо:

* запустить программу
* ввести действующий логин и пароль в поля авторизации
* нажать кнопку «Continue»
* нажать кнопку «Vabank»
* убедиться, что при совпадении картинок очки увеличиваются в 5 раз, а при выпадении 3-х семёрок в 10 раз; при несовпадении картинок очки должны обнуляются и игра завершается.

#### 2.6.6.6 Для проверки способности программы завершать свою работу в случае закрытия игрового окна вручную, или проигрыше всей ставки, необходимо:

* запустить программу
* ввести действующий логин и пароль в поля авторизации
* нажать кнопку «Continue»
* нажать кнопку «Start» или «Vabank»
* убедиться, что при уменьшении очков до нуля программа завершает сеанс игры и выводит таблицу рекордов;
* убедиться, что при закрытии окна с игрой, работа приложения завершается.

2.6.6.7 Для проверки способности программы сохранять рекорд пользователя и обеспечивать просмотр таблицы рекордов различных пользователей, необходимо:

* запустить программу
* ввести действующий логин и пароль в поля авторизации
* нажать кнопку «Continue»
* нажать кнопку «Start» или «Vabank»
* убедиться, что при уменьшении очков до нуля программа завершает сеанс игры и выводит таблицу рекордов;
* убедиться, что в бинарном файле login.bin сохраняется логин и максимальное количество очков, которое набрал текущий пользователь за последний сеанс игры, если это максимальное количество очков превысило предыдущий рекорд пользователя, и рекорды всех пользователей отображаются в таблице рекордов.

## 2.7 Протокол испытаний

Результаты испытаний программы представлены в таблице 1, рисунки приведены в приложении Б.

**Результаты испытаний программы**

*Таблица 1*

|  |  |  |
| --- | --- | --- |
| **Проверяемые требования** | **Сообщения программы и вводимые значения** | **Результаты** |
| Способность программы осуществлять авторизацию пользователя по введённому логину и паролю | * запустить программу * ввести действующий логин и пароль в поля авторизации * нажать кнопку «continue» | Скрылись поля для авторизации и стала доступной кнопка «Start» (авторизация произошла) –  смотри Рис. 1 |
| Способность программы обеспечивать добавление новых пользователей (регистрацию) | * запустить программу * ввести новый логин и пароль в поля авторизации * убедиться, что появилось сообщение «Save a new login?» * в диалоговом окне нажать кнопку «Да» | Скрылись поля для авторизации и стала доступной кнопка «Start» (авторизация произошла) –  смотри Рис. 2 |
| Способность программы прокручивать несколько картинок в случайном порядке, в течении 10 секунд, и демонстрировать это на экране | * запустить программу * ввести действующий логин и пароль в поля авторизации * нажать кнопку «Continue» * нажать кнопку «Start» | Картинки прокручиваются в случайном порядке в течение 10 секунд, а затем останавливаются –  смотри Рис. 3 |
| Способность программы обеспечивать проверку на совпадение картинок и вести подсчёт ставки игры | * запустить программу * ввести действующий логин и пароль в поля авторизации * нажать кнопку «Continue» * нажать кнопку «Start» | При совпадении картинок очки увеличиваются на 500, а при выпадении 3-х семёрок на 1000 очков; при несовпадении картинок очки должны уменьшаться на 100 – смотри Рис. 4 - 5 |

|  |  |  |
| --- | --- | --- |
| **Проверяемые требования** | **Сообщения программы и вводимые значения** | **Результаты** |
| Способность программы обеспечивать функцию “va bank” с возможностью поставить всю ставку | * запустить программу * ввести действующий логин и пароль в поля авторизации * нажать кнопку «Continue» * нажать кнопку «Vabank» | При совпадении картинок очки увеличиваются в 5 раз, а при выпадении 3-х семёрок в 10 раз; при несовпадении картинок очки должны обнуляются и игра завершается – смотри Рис. 6 |
| Способность программы завершать свою работу в случае закрытия игрового окна вручную, или проигрыше всей ставки | * запустить программу * ввести действующий логин и пароль в поля авторизации * нажать кнопку «Continue» * нажать кнопку «Start» или «Vabank» | При уменьшении очков до нуля программа завершает сеанс игры и выводит таблицу рекордов; при закрытии окна с игрой работа приложения завершается – смотри Рис. 7 |
| Способность программы сохранять рекорд пользователя и обеспечивать просмотр таблицы рекордов различных пользователей | * запустить программу * ввести действующий логин и пароль в поля авторизации * нажать кнопку «Continue» * нажать кнопку «Start» или «Vabank» * убедиться, что при уменьшении очков до нуля программа завершает сеанс игры и выводит таблицу рекордов | В бинарном файле login.bin сохраняется логин и максимальное количество очков, которое набрал текущий пользователь за последний сеанс игры, если это максимальное количество очков превысило предыдущий рекорд пользователя, и рекорды всех пользователей отображаются в таблице рекордов – смотри Рис. 7 |

# ЗАКЛЮЧЕНИЕ

Разработанная в ходе выполнения курсового проекта программа удовлетворяет всем требованиям технического задания, что подтверждается протоколом испытаний.

Разработанная программа может быть использована в развлекательных целях, для отдыха и релаксации.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. А. Хейлсберг, М. Торгерсен, С. Вилтамут, П. Голд ., Язык программирования C#, 4-е издание, СПб:-ПИТЕР, 2011
2. Фленов М., Библия C#, 3-е издание, СПб:-БХВ, 2016
3. Культин Н., Microsoft Visual C# в задачах и примерах, 2-е издание, СПб:-БХВ, 2015
4. Стандарты Единой Системы Программной Документации:

ГОСТ 19.105-78 Общие требования к программным документам

ГОСТ 19.106-78 Требования к программным документам, выполненным печатным способом

ГОСТ 19.201-78 Техническое задание. Требования к содержанию и оформлению

ГОСТ 19.301-78 Программа и методика испытаний. Требования к содержанию и оформлению

ГОСТ 19.401-78 Текст программы. Требования к содержанию и оформлению

ГОСТ 19.402-78 Описание программы. Требования к содержанию и оформлению

ГОСТ 19.505-79 Руководство оператора. Требования к содержанию и оформлению

ГОСТ 19.701-90 Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения

ПРИЛОЖЕНИЕ А

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ  
федеральное государственное автономное образовательное учреждение высшего образования  
«Санкт-Петербургский политехнический университет Петра Великого»  
(ФГАОУ ВО «СПбПУ»)  
**Университетский политехнический колледж**

УТВЕРЖДАЮ  
Председатель ПЦК   
\_\_\_\_\_\_\_\_\_\_\_ Андреев  
\_\_.\_\_. 2018

Игра с графическим интерфейсом «Bang»  
**Техническое задание**  
Листов 6

ПРОВЕРИЛ  
Преподаватель   
\_\_\_\_\_\_\_\_\_\_\_ Девятко Н.С.  
\_\_.\_\_. 2018

ВЫПОЛНИЛ  
Студент группы 32928/2   
\_\_\_\_\_\_\_\_\_\_\_ ФИО  
\_\_.\_\_. 2018

2018

1 ВВЕДЕНИЕ

* 1. Полное наименование программной разработки: «ИГРА С ГРАФИЧЕСКИМ ИНТЕРФЕЙСОМ  «Bang» (как на титульном листе).
  2. Игровая программа с графическим интерфейсом «Bang»: человек играет против машины; машина прокручивает картинки в случайном порядке по типу работы игровых автоматов, если три картинки из трёх совпадают, происходит повышение ставки.
  3. В соответствии с заданием программный продукт состоит из теоретической и практической частей. Теоретическая часть включает подробное описание работы с приложениями, показывающими схему работы системы и алгоритмы отдельных модулей. Практическая часть включает разработку и реализацию с использованием среды программирования C# программных модулей программного продукта.
  4. Программа предназначена для развлекательных целей, для отдыха и релаксации.

2 ОСНОВАНИЕ ДЛЯ РАЗРАБОТКИ

2.1. Разработка ведётся на основании задания к курсовому проекту по профессиональному модулю ПМ.01 «Разработка программных модулей программного обеспечения для компьютерных систем» МДК 01.02 «Прикладное программирование» и утверждена Университетским политехническим колледжем.

3 НАЗНАЧЕНИЕ РАЗРАБОТКИ

* 1. Основное назначение программного продукта заключается в организации компьютерной игры с графическим интерфейсом «Bang»: человек играет против машины, машина прокручивает картинки в случайном порядке по типу работы игровых автоматов, если три картинки из трёх совпадают, происходит повышение ставки.

4 ТРЕБОВАНИЯ К РАЗРАБОТКЕ

4.1 Требования к функциональным характеристикам:

**1 версия:**

* программа должна прокручивать несколько картинок в случайном порядке, в течение 10 секунд, и демонстрировать это на экране;
* программа должна обеспечивать проверку на совпадение картинок;
* программа должна вести подсчёт ставки игры, при совпадении картинок ставка увеличивается, при выпадении трёх семёрок ставка удваивается, при несовпадении картинок ставка уменьшается.

**2 версия:**

* программа должна обеспечивать авторизацию пользователя (ввод логина и пароля), создание новых аккаунтов для игры;
* программа должна иметь функцию “Vabank” с возможностью поставить всю ставку, в случае совпадения всех картинок ставка увеличивается в несколько раз;
* игра завершает свою работу в случае закрытия игрового окна вручную, или проигрыше всей ставки;
* программа должна сохранять рекорд пользователя;
* программа должна обеспечивать просмотр таблицы рекордов различных пользователей (по выбору: всех пользователей или текущего пользователя).

4.2 Требования к надежности:

* использование лицензированного программного обеспечения;
* проверка программы на наличие вирусов;
* организация бесперебойного питания.

4.3 Требования к составу и параметрам технических средств

Для нормального функционирования данной информационной системы необходим компьютер, клавиатура, мышь и следующие технические средства:

- процессор Intel или другой совместимый;

- объем свободной оперативной памяти ~500 Кб;

- объем необходимой памяти на жестком диске ~20Мб;

- стандартный VGA-монитор или совместимый;

- стандартная клавиатура;

- манипулятор «мышь».

4.4 Требования к информационной и программной совместимости

Для полноценного функционирования данной системы необходимо наличие операционной системы выше Microsoft Windows XP или совместимой. Язык интерфейса – русский.

4.5 Требования к маркировке и упаковке

Программа должна поставляться на диске в виде исполняемого (еxе) файла, документации и проекта. На диске должна быть наклейка с надписью "Bang". Диск должна быть упакован в пластиковую коробку.

4.6 Требования к транспортировке и хранению

Диск с программой должен храниться вдали от электромагнитных полей и не подвергаться механической деформации. Место и условия хранения должны соответствовать санитарным требованиям отрасли. Сроки хранения устанавливаются в соответствии с гарантийными сроками поставщика магнитных носителей.

Основные требования к транспортировке – создание условий, исключающих механические повреждения магнитного носителя.

5 ТРЕБОВАНИЯ К ПРОГРАММНОЙ ДОКУМЕНТАЦИИ

5.1 Предварительный состав программной документации:

- «Техническое задание»;

- разрабатываемые программные модули должны быть самодокументированы, т.е. тексты программ должны содержать все необходимые комментарии;

- разрабатываемое программное обеспечение должно включать справочную систему.

5.2 Перечень материалов пояснительной записки

Введение

1. Теоретические основы разработки
   1. Описание предметной области
   2. Анализ методов решения
   3. Обзор средств программирования
   4. Описание выбранного языка программирования
2. Практическая часть
   1. Постановка задачи
   2. Проектирование приложения
   3. Текст программы
   4. Описание программы
   5. Руководство оператора
   6. Программа и методика испытаний
   7. Протокол испытаний

Заключение

Список использованных источников

Приложения

6 ТЕХНИКО-ЭКОНОМИЧЕСКИЕ ПОКАЗАТЕЛИ

Технико-экономические показатели не рассчитываются.

7 СТАДИИ И ЭТАПЫ РАЗРАБОТКИ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Содержание стадии** | **Содержание этапа** | **Срок 2018 г.** | | **Форма  отчетности** |
| **начало** | **конец** |
| Техническое задание | Составление технического задания | 15.10 | 21.10 | Техническое задание |
| Эскизный проект | Разработка спецификаций | 22.10 | 28.10 | Спецификации программного обеспечения |
| Рабочий проект | Проектирование программы | 29.10 | 11.11 | Схема работы системы и спецификации компонентов |
| Составление программы | 12.11 | 25.11 | Программная документация |
| Приёмо-сдаточные испытания | 26.11 | 09.12 | Протокол испытаний (п. 2.7 пояснительной записки) |
| Приёмка | Защита курсового проекта | 10.12 | 15.12 | Оценка за курсовой проект |

8 ПОРЯДОК КОНТРОЛЯ И ПРИЕМКИ

8.1 Порядок контроля

Контроль выполнения должен осуществляться руководителем курсового проекта (преподавателем) в соответствие с п.7.

8.2 Порядок приемки

Приемка должна осуществляться с участием руководителя после проведения приемо-сдаточных испытаний. В результате защиты курсового проекта должна быть выставлена оценка за курсовой проект.

ПРИЛОЖЕНИЕ Б



Рисунок 1 Авторизация

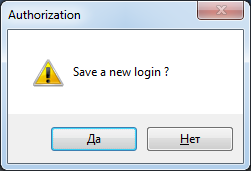


Рисунок 2 Регистрация нового пользователя

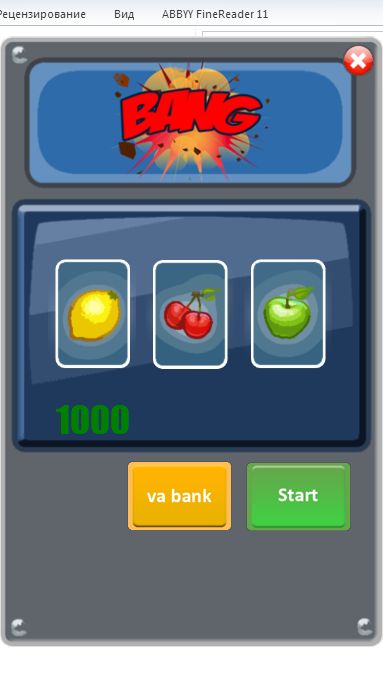


Рисунок 3 Начало игры

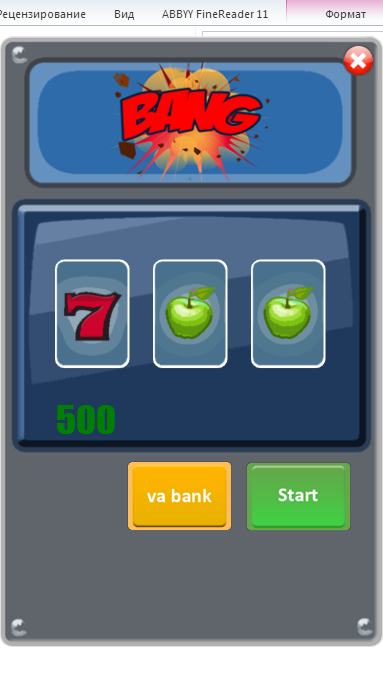


Рисунок 4 Уменьшение очков при несовпадении картинок

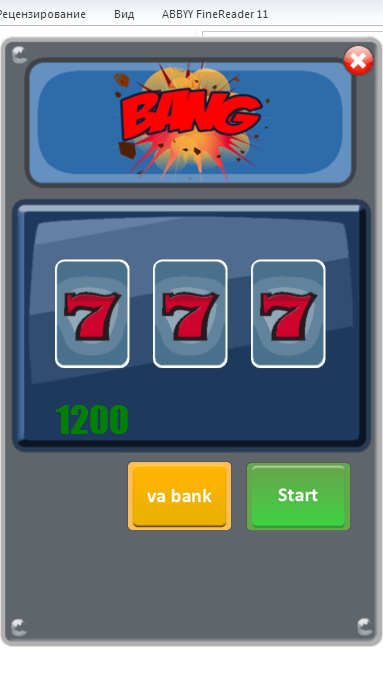


Рисунок 5 Увеличение очков при совпадении картинок

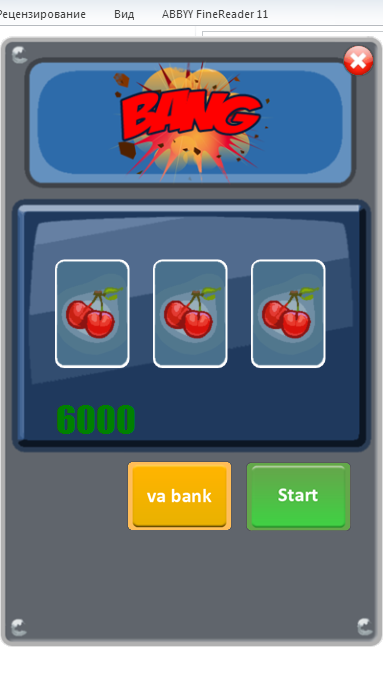


Рисунок 6 Умножение очков при выборе кнопки «Vabank»   
и совпадении картинок

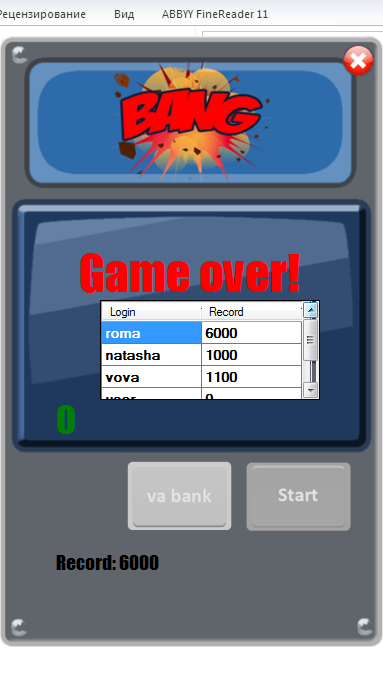


Рисунок 7 Окончание игры, просмотр таблицы рекордов

ПРИЛОЖЕНИЕ В



Рисунок 8 Схема алгоритма метода void RecordToFile(string filename)



Рисунок 9 Схема алгоритма метода void RecordToTable(string filename, DataGridView dg)



Рисунок 10 Схема алгоритма метода int FindLogin(string filename)



Рисунок 11 Схема алгоритма метода void timer1\_Tick(object sender, EventArgs e)

ПРИЛОЖЕНИЕ Г

**Текст программы**

**Модуль Form1.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

using System.IO;

using System.Runtime.InteropServices;

namespace Bang

{

public partial class Form1 : Form

{

[DllImport("winmm.dll")]

private static extern long mciSendString(string strCommand, StringBuilder strReturn,

int iReturnLength, IntPtr hwndCallback);

private string sCommand = "";

Random rnd = new Random();

int sec1 = 0;

int sec2 = 0;

List<string> imageList = new List<string>();

int s = 1000;

int max = 1000;

Login N;

public Form1()

{

InitializeComponent();

}

public void Music()

{

sCommand = "open \"" + Application.StartupPath +

@"\4.mp3" + "\" type mpegvideo alias MediaFile";

mciSendString(sCommand, null, 0, IntPtr.Zero);

//команда для воспроизведения файла

sCommand = "play MediaFile";

//посылаем команду

mciSendString(sCommand, null, 0, IntPtr.Zero);

}

private void Exit\_Click(object sender, EventArgs e)

{

this.Close();

}

private void Exit\_MouseMove(object sender, MouseEventArgs e)

{

button1.ImageIndex = 1;

}

private void button1\_MouseLeave(object sender, EventArgs e)

{

button1.ImageIndex = 0;

}

private void Start\_Click(object sender, EventArgs e)

// кнопка "Start"

{

button2.ImageIndex = 1;

timer1.Start();

sec1 = 0;

label1.Text = "";

}

private void timer1\_Tick(object sender, EventArgs e)

// таймер для кнопки "Start"

{

sec1++;

if (sec1 > 0) button2.ImageIndex = 0;

int x = rnd.Next(4);

int y = rnd.Next(4);

int z = rnd.Next(4);

pictureBox1.Load(imageList[x]);

pictureBox2.Load(imageList[y]);

pictureBox3.Load(imageList[z]);

if (sec1 > 30)

{

timer1.Stop();

if (x == y && y == z)

{

if (x == 0)

{

label1.Text = "Jack Pot!";

s += 1000;

}

else

if (x == 1 || x == 2 || x == 3)

{

label1.Text = "Win";

s += 500;

}

}

else

{

s -= 100;

if (s == 0)

{

pictureBox1.Visible = false;

pictureBox2.Visible = false;

pictureBox3.Visible = false;

button2.Enabled = false;

button3.Enabled = false;

label1.Visible = true;

label1.Text = "Game over!";

label5.Text = "Record: " + max;

N.RecordToTable("login.bin", dataGridView1);

dataGridView1.Visible = true;

}

}

label2.Text = s.ToString();

if (s > max) max = s;

}

}

private void Form1\_Load(object sender, EventArgs e)

{

DirectoryInfo di = new DirectoryInfo(Application.StartupPath);

FileInfo[] fi = di.GetFiles("\*.png");

imageList.Clear();

label2.Text = s.ToString();

foreach (FileInfo fk in fi)

{

imageList.Add(fk.Name);

}

dataGridView1.Visible = false;

pictureBox1.Visible = false;

pictureBox2.Visible = false;

pictureBox3.Visible = false;

label2.Visible = false;

button2.Enabled = false;

button3.Enabled = false;

panel1.Visible = true;

}

private void Vabank\_Click(object sender, EventArgs e) //кнопка Va bank

{

sec2 = 0;

button3.ImageIndex = 1;

timer2.Start();

}

private void timer2\_Tick(object sender, EventArgs e)

// таймер для кнопки "Va bank"

{

sec2++;

if (sec2 > 0) button3.ImageIndex = 0;

int x = rnd.Next(4);

int y = rnd.Next(4);

int z = rnd.Next(4);

pictureBox1.Load(imageList[x]);

pictureBox2.Load(imageList[y]);

pictureBox3.Load(imageList[z]);

if (sec2 > 30)

{

timer2.Stop();

if (x == y && y == z)

{

if (x == 0)

{

s \*= 10;

}

else

if (x == 1 || x == 2 || x == 3)

{

s \*= 5;

}

}

else

{

s = 0;

}

if (s == 0)

{

pictureBox1.Visible = false;

pictureBox2.Visible = false;

pictureBox3.Visible = false;

button2.Enabled = false;

button3.Enabled = false;

label1.Visible = true;

label1.Text = "Game over!";

label5.Text = "Record: " + max;

N.Record = max;

N.RecordToFile("login.bin");

N.RecordToTable("login.bin", dataGridView1);

dataGridView1.Visible = true;

}

}

label2.Text = s.ToString();

if (s > max) max = s; // запомнить рекорд

}

private void Continue\_Click(object sender, EventArgs e)

// метод нажатия на кнопку "Continue"

// происходит проверка логина и пароля, авторизация

{

N = new Login(textBox1.Text, textBox2.Text, 0);

bool flag = false;

int x = N.FindLogin("login.bin"); // поиск логина

if (x == 0) // если нет в файле такого логина

{

DialogResult dr;

dr = MessageBox.Show("Save a new login ?", "Authorization",

MessageBoxButtons.YesNo, MessageBoxIcon.Warning);

if (dr == DialogResult.Yes)

{

N.LoginToFile("login.bin");

flag = true;

}

else

return;

}

else

if (x == 1) // если логин совпал, а пароль нет

{

MessageBox.Show("Incorrect password");

textBox2.Clear();

return;

}

else

if (x == 2) // если совпал и логин и пароль

{

flag = true;

}

if (flag) // если вход разрешен

{

pictureBox1.Visible = true; // показать картинки

pictureBox2.Visible = true;

pictureBox3.Visible = true;

label2.Visible = true;

button2.Enabled = true;

button3.Enabled = true;

panel1.Visible = false;

label1.Visible = false;

Music();

return;

}

}

}

}

**Модуль Login.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Windows.Forms;

using System.IO;

using System.Runtime.Serialization.Formatters.Binary;

namespace Bang

{

[Serializable]

class Login

{

string login;

string password;

int record;

public Login()

{

this.login = "";

this.password = "";

this.record = 0;

}

public Login(string login, string password, int record)

{

this.login = login;

this.password = password;

this.record = record;

}

public string Log

{

get

{

return login;

}

set

{

login = value;

}

}

public string Password

{

get

{

return password;

}

set

{

password = value;

}

}

public int Record

{

get

{

return record;

}

set

{

record = value;

}

}

public void LoginToFile(string filename)

{

FileStream f = new FileStream(filename, FileMode.Append);

BinaryFormatter bf = new BinaryFormatter();

bf.Serialize(f, this);

f.Close();

}

public void RecordToFile(string filename)

{

int i = 0, num = 0;

Login buf = null;

FileStream f;

f = new FileStream(filename, FileMode.Open);

BinaryFormatter bf = new BinaryFormatter();

try

{

Login l;

i = 0;

while ((l = (Login)bf.Deserialize(f)) != null)

{ // ищем запись с нужным логином

if (l.login == this.login)

{

buf = new Login(l.login, l.password, l.record);

num = i; // запомним номер этой записи

break;

}

i++;

}

}

catch

{

}

f.Close();

f = new FileStream(filename, FileMode.Open);

try

{

Login l;

i = 0; // текущий счётчик записей

for (i = 0; i < num; i++ )

{

l = (Login)bf.Deserialize(f);

// передвинем указатель в файле на нужную позицию

}

if (this.record > buf.record)

{

bf.Serialize(f, this);

}

}

catch

{

}

f.Close();

}

public void RecordToTable(string filename, DataGridView dg)

{

FileStream f = new FileStream(filename, FileMode.Open);

BinaryFormatter bf = new BinaryFormatter();

int i = 0;

try

{

Login l;

dg.SetBounds(100, 300, 220, 100);

while ((l = (Login)bf.Deserialize(f)) != null)

{

dg.Rows.Add();

dg.Rows[i].Cells[0].Value = l.login;

dg.Rows[i].Cells[1].Value = l.record;

i++;

}

}

catch

{

}

//dg.RowCount = i;

f.Close();

}

public int FindLogin(string filename)

{

int r = 0;

FileStream f = new FileStream(filename, FileMode.OpenOrCreate);

BinaryFormatter bf = new BinaryFormatter();

try

{

if (f.Length != 0)

{

Login l;

while ((l = (Login)bf.Deserialize(f)) != null)

{

if (l.login == this.login)

{

if (l.password == this.password)

r = 2;

else

r = 1;

break;

}

}

}

}

catch

{

}

f.Close();

return r;

}

}

}