Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Сибирский государственный университет телекоммуникаций и информатики»

(ФГБОУ ВО «СибГУТИ»)

*Факультет информатики и вычислительной техники*

*Кафедра вычислительных систем*

Курсовая работа по дисциплине

«Технологии разработки программного обеспечения»

«Быки и коровы»

Выполнил:

студент гр. ИП-511

Устинова А.Ю.

Проверил:

ассистент кафедры ВС

Фульман В.О.

Новосибирск

2016

Содержание:

[ТЕХНИЧЕСКОЕ ЗАДАНИЕ 3](#_Toc452724115)

[ПЛАН РАБОТ 4](#_Toc452724116)

[ОСНОВНЫЕ ИДЕИ 5](#_Toc452724117)

[ЛИЧНЫЙ ВКЛАД 6](#_Toc452724118)

[ТЕКСТ ПРОГРАММЫ 7](#_Toc452724119)

[РЕЗУЛЬТАТЫ 1](#_Toc452724120)9

**Техническое задание**

**Функционал проекта**

Программа представляет собой классический вариант игры “Быки и коровы”.

Состоит из трех уровней сложности : Первый уровень самый легкий, когда игрок видит число, которое загадал компьютер.

Второй уровень – средний, когда игрок логически угадывает число, которое загадал компьютер.

Третий уровень – сложный, когда игроку на угадывание числа предоставляется всего 5 попыток.

**Программный продукт должен решать следующие задачи:**

1. Генерация рандомного четырехзначного числа с разными цифрами.
2. Ввод пользователем числа
3. Проверка на что, число было четырехзначное .
4. Число введенное пользователем сверяется с числом, сгенерированным в начале на наличие быков и коров.
5. Вывод результатов.

**Формат входных данных:**

В качестве входных данных, программа принимает цифры, введенные с клавиатуры.

**Интерфейс приложения:**

Программа “Быки и коровы” представляет собой окно , в котором есть меню, состоящее из пунктов : “1 – New Game”, “2 – New Game (medium)”, “3 – New Game (hard)”, “Test Game” “Exit”. Позволяет выбрать уровень сложности игры или выход из игры.

**Составление плана работ:**

Чтобы правильно обрабатывать информацию, необходимо определить задачи, которые будут реализовывать программный продукт. Основными функциями программного продукта являются такие функции, как:

- ввод-вывод данных;

- обработка данных;

-вывод результатов;

**Разобьем программу на этапы:**

1. Создание интерфейса.
2. Разработка алгоритма игры легкого уровня.
3. Разработка алгоритма игры среднего уровня.
4. Разработка алгоритма игры сложного уровня.

**Основные идеи.**

Внутренне программа представляет из себя проект, написанный на языке С++. Архитектура и алгоритмы программы рассчитаны на работу с объектами таких классов как меню, логика, кучки и кнопка. Каждый класс представляет собой абстрактную идею, частные проявления которой и представлены в объектах.

Меню представляет собой типичное меню с не интерактивным заголовком и интерактивными пунктами. Содержит в себе пункты игры на разных уровнях (легкий, средний, сложный), тесты и выход из игры.

Логика является совокупностью всех механизмов функционирования программы и включает в себя такие понятия как «сделать ход», «проверить число, введенное пользователем» и пр. В общем случае объект логики содержит в себе все те данные, которые нужны, чтобы понять механику игры, а также все переменные, из которых состоит игра в полевых условиях.

Более детальное описание программы см. в технологической документации, приложенной к проекту.

**Личный вклад.**

Над данным проектом работало два участника, каждый из которых внес свой вклад в данную работу. В ходе работы все участники обсуждали между собой возникшие трудности, давали советы, принимали решения сообща. Каждый участник подошел выполнению своей работы ответственно, прислушиваясь к мнению остальных участников. Далее представлено описание моего вклада в общее дело.

В данном проекте мною было сделано Техническое Задание, где описан функционал проекта, его описание и составление плана работ. Так же составлена документация для пользователя, где подробно и доступно описано как пользоваться программным продуктом.

Так же был разработан алгоритм прохождения легкого уровня в игре “быки и коровы”. Суть алгоритма заключается в том, что он соответствует классическому варианту игры, но его особенность , что в отличии от других уровней сложности, во время игры пользователь видит последовательность цифр, которое загадал компьютер.

Мною был разработан алгоритм вычисления количества коров в числе, которое ввел пользователь. Каждая цифра, введенная пользователем с клавиатуры сравнивается с каждой цифрой числа , который загадал компьютер.

**Текст программы.**

**#include <stdio.h>**

**#include <conio.h>**

**#include <stdlib.h>**

**#include <time.h>**

**#include <string.h>**

**#include <fstream>**

**#include <iostream>**

**#include <cassert>**

**using namespace std;**

**int generate\_start\_number() // проверяет, чтобы числа были разными**

**{**

**int correct\_number = rand() % 10000;**

**bool b = false;**

**while (! b)**

**{**

**if ((correct\_number / 1000) != ((correct\_number / 100) % 10) &&**

**(correct\_number / 1000) != ((correct\_number % 100) / 10) &&**

**(correct\_number / 1000) != (correct\_number % 10) &&**

**((correct\_number / 100) % 10) != ((correct\_number % 100) / 10) &&**

**((correct\_number / 100) % 10) != (correct\_number % 10) &&**

**((correct\_number % 100) / 10) != (correct\_number % 10) &&**

**correct\_number >= 1000)**

**{**

**b = true;**

**}**

**else**

**{**

**correct\_number = rand() % 10000;**

**}**

**}**

**return correct\_number;**

**}**

**int cows(int correct\_number, int my\_number)**

**{**

**int cows\_counter;**

**cows\_counter = 0;**

**if ((correct\_number / 1000) == (my\_number / 1000) ||**

**(correct\_number / 1000) == ((my\_number / 100) % 10) ||**

**(correct\_number / 1000) == ((my\_number % 100) / 10) ||**

**(correct\_number / 1000) == (my\_number % 10))**

**{**

**cows\_counter++;**

**}**

**if (((correct\_number / 100) % 10) == (my\_number / 1000) ||**

**((correct\_number / 100) % 10) == ((my\_number / 100) % 10) ||**

**((correct\_number / 100) % 10) == ((my\_number % 100) / 10) ||**

**((correct\_number / 100) % 10) == (my\_number % 10))**

**{**

**cows\_counter++;**

**}**

**if (((correct\_number % 100) / 10) == (my\_number / 1000) ||**

**((correct\_number % 100) / 10) == ((my\_number / 100) % 10) ||**

**((correct\_number % 100) / 10) == ((my\_number % 100) / 10) ||**

**((correct\_number % 100) / 10) == (my\_number % 10))**

**{**

**cows\_counter++;**

**}**

**if ((correct\_number % 10) == (my\_number / 1000) ||**

**(correct\_number % 10) == ((my\_number / 100) % 10) ||**

**(correct\_number % 10) == ((my\_number % 100) / 10) ||**

**(correct\_number % 10) == (my\_number % 10))**

**{**

**cows\_counter++;**

**}**

**return cows\_counter;**

**}**

**int bulls(int correct\_number, int my\_number)**

**{**

**int bulls\_counter;**

**bulls\_counter = 0;**

**if ((correct\_number / 1000) == (my\_number / 1000))**

**{**

**bulls\_counter++;**

**}**

**if (((correct\_number / 100) % 10) == ((my\_number / 100) % 10))**

**{**

**bulls\_counter++;**

**}**

**if (((correct\_number % 100) / 10) == ((my\_number % 100) / 10))**

**{**

**bulls\_counter++;**

**}**

**if ((correct\_number % 10) == (my\_number % 10))**

**{**

**bulls\_counter++;**

**}**

**return bulls\_counter;**

**}**

**void Game\_new(int correct\_number)**

**{**

**int my\_number;**

**bool win, correct;**

**win = false;**

**correct = false;**

**system("CLS");**

**cout << "The specified number, good luck!" << endl;**

**//printf("The specified number:%d\n",x);**

**do**

**{**

**do**

**{**

**cout << "Enter number: ";**

**cin >> my\_number;**

**if ((my\_number >= 1000) && (my\_number <= 9999))**

**correct = true;**

**else**

**cout << "Error!Incorrect number!" << endl;**

**} while (correct == false);**

**cout << "++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++" << endl;**

**cout << "Bulls:" << bulls(correct\_number, my\_number);**

**cout << " Cows:" << cows(correct\_number, my\_number) << endl;**

**if (my\_number == correct\_number)**

**win = true;**

**} while (win == false);**

**cout << "++++++++++++++++++++++++++++++Achievement+++++++++++++++++++++++++++++++++++++" << endl;**

**system("PAUSE");**

**}**

**void New\_game\_easy(int correct\_number)**

**{**

**int my\_number;**

**bool win, correct;**

**win = false;**

**correct = false;**

**system("CLS");**

**cout << "The specified number:" << correct\_number << endl;**

**do {**

**do {**

**cout << "Enter number:";**

**cin >> my\_number;**

**if ((my\_number >= 1000) && (my\_number <= 9999))**

**correct = true;**

**else**

**cout << "Error!Incorrect number!" << endl;**

**} while (correct == false);**

**cout << "++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++" << endl;**

**cout << "Bulls:" << bulls(correct\_number, my\_number);**

**cout << " Cows:" << cows(correct\_number, my\_number) << endl;**

**if (my\_number == correct\_number)**

**win = true;**

**} while (win == false);**

**cout << "+++++++++++++++++++++++++++++++Achievement++++++++++++++++++++++++++++++++++++++" << endl;**

**system("PAUSE");**

**}**

**void New\_game\_hard(int correct\_number)**

**{**

**int my\_number, attempt;**

**bool win, correct;**

**win = false;**

**correct = false;**

**attempt = 1;**

**system("CLS");**

**cout << "The number set you have 5 attempts, good luck!" << endl;**

**//printf("The specified number:%d\n",x);**

**do**

**{**

**cout << "Attempt:" << endl;**

**cout << attempt;**

**do**

**{**

**cout << "Enter the number:" << endl;**

**cin >> my\_number;**

**if ((my\_number >= 1000) && (my\_number <= 9999))**

**correct = true;**

**else**

**cout << "Error!Incorrect number!" << endl;**

**} while (correct == false);**

**cout << "++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++" << endl;**

**cout << "Bulls:" << bulls(correct\_number, my\_number);**

**cout << " Cows:" << cows(correct\_number, my\_number) << endl;**

**if (my\_number == correct\_number)**

**win = true;**

**if (attempt == 5)break;**

**attempt++;**

**} while (win == false);**

**if (my\_number == correct\_number)**

**cout << "+++++++++++++++++++++++++++++++Achievement+++++++++++++++++++++++++++++++" << endl;**

**if (attempt == 5)**

**cout << "+++++++++++++++++++++++++++++++YUO LOSE++++++++++++++++++++++++++++++++++++" << endl;**

**system("PAUSE");**

**}**

**void runTest ()**

**{**

**assert(cows(1234, 4321) == 4);**

**puts("Test 1 passed");**

**assert(cows(4568, 2931) == 0);**

**puts("Test 2 passed");**

**assert(cows(1234, 8921) == 2);**

**puts("Test 3 passed");**

**assert(cows(1987, 1997) == 3);**

**puts("Test 4 passed");**

**assert(cows(1234, 4987) == 1);**

**puts("Test 5 passed");**

**assert(bulls(1234, 4321) == 0);**

**puts("Test 6 passed");**

**assert(bulls(1234, 1658) == 1);**

**puts("Test 7 passed");**

**assert(bulls(4895, 4823) == 2);**

**puts("Test 8 passed");**

**assert(bulls(1234, 1235) == 3);**

**puts("Test 9 passed");**

**assert(bulls(1234, 1234) == 4);**

**puts("Test 10 passed");**

**getch();**

**}**

**int main()**

**{**

**int l, j;**

**char c;**

**srand(time(0));**

**j = generate\_start\_number();**

**while (1)**

**{**

**system("CLS");**

**puts(" 1 - New Game");**

**puts(" 2 - New Game (easy)");**

**puts(" 3 - New Game (hard)");**

**puts(" 4 - Test Game");**

**puts(" 0 - Exit");**

**c = getch();**

**switch (c)**

**{**

**case '1':Game\_new(j);break;**

**case '2':New\_game\_easy(j);break;**

**case '3':New\_game\_hard(j);break;**

**case '4':runTest();break;**

**case '0':return 0;**

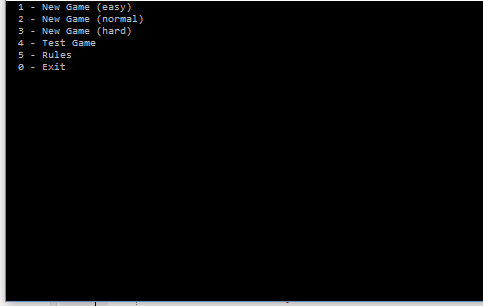
**default: puts("Error!");**

**}**

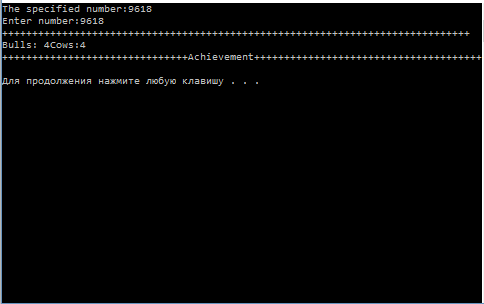
**}**

**}**

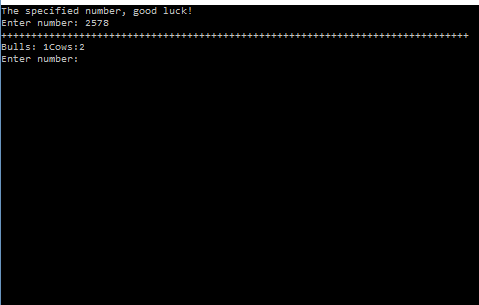
**Результаты:**

****

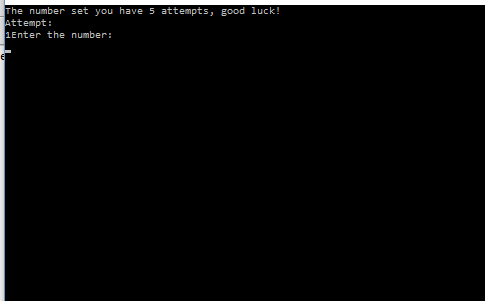
Главное меню игры.



Процесс игры на легком уровне.



Процесс игры на среднем уровне.



Процесс игры на сложном уровне.