ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«СИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ И

ИНФОРМАТИКИ»

Кафедра вычислительных систем

КУРСОВАЯ РАБОТА

по дисциплине «Технологии разработки программного обеспечения»

на тему "Игра «Быки и Коровы»"

Выполнил:  
Мягких Сергей

Проверил:  
доц., к.ф.-м.н. Пудов С. Г.

Новосибирск, 2017

Содержание

[ВВЕДЕНИЕ И ПОСТАНОВКА ЗАДАЧИ 3](#_Toc482440476)

[ТЕХНИЧЕСКОЕ ЗАДАНИЕ 4](#_Toc482440477)

[ОПИСАНИЕ ВЫПОЛНЕННОГО ПРОЕКТА 5](#_Toc482440478)

[ЛИЧНЫЙ ВКЛАД В ПРОЕКТ](#_Toc482440479) 9

[ПРИЛОЖЕНИЕ. ТЕКСТ ПРОГРАММЫ](#_Toc482440480) 10

# ВВЕДЕНИЕ И ПОСТАНОВКА ЗАДАЧИ

Задача курсовой работы заключалась в командной поэтапной реализации игры «Быки и Коровы» с использованием системы контроля версий.

Правила игры:

Компьютер задумывает четырехзначное число. Цифры в числе не повторяются, 0 может стоять на первом месте. Игрок делает ходы, чтобы узнать это число. В ответ на каждый ход компьютер показывает число отгаданных цифр, стоящих на своих местах (число **быков**) и число отгаданных цифр, стоящих не на своих местах (число **коров**).

Для правильной работы программы требовалось реализовать:

* проверку корректности входных данных
* функцию генерации случайного числа компьютером
* функции подсчета «быков» и «коров»
* интерактивное меню

# ТЕХНИЧЕСКОЕ ЗАДАНИЕ

# Техническое задание

# "Быки и Коровы"

## 1. Функционал программы.

Приложение представляет собой логическую игру «Быки и коровы». В программе будет реализован ряд следующих функций:

* Генерация случайного четырехзначного числа(по разрядам). Цифры в числе не должны повторяться.
* Цикличный запрос четырехзначного числа, ограниченный условием «пока введенное число не равно числу сгенерированному».
* Сравнение введенного пользователя числа с тем, которое он пытается угадать.
* Вывод количества «Быков» и «Коров» после каждого введенного пользователем числа.
* Интерактивное меню

## 2. Формат входных данных.

На вход приложения через аргументы командной строки поступает целое четырехзначное число.

## 3. Интерфейс приложения

Приложение работает в интерактивном режиме. Взаимодействие пользователя с функционалом программы осуществляется посредством интерактивного меню. На данный момент в запланировано реализовать следующие пункты меню:

* Начать игру. Генерируется четырехзначное число, игра начинается.
* Правила игры. Выводит на экран правила игры.
* Выход. Заканчивает работу приложения.

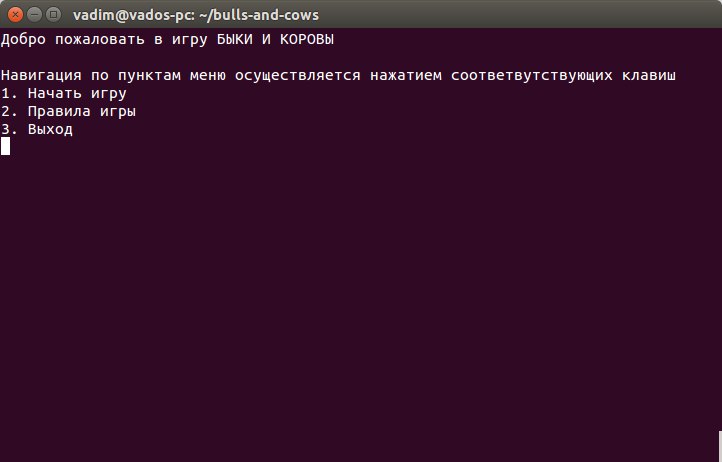
Также будет реализован функционал запуска игры заново. После окончания игры(победы или проигрыша) пользователю будет предложено вернутся в меню.. При вводе пользователем некорректного числа приложение оповестит об этом на экране монитора ошибкой: «Incorrect Input». Весь интерфейс приложения будет осуществляться через консоль. Данный пункт будет дорабатываться по мере разработки игры.

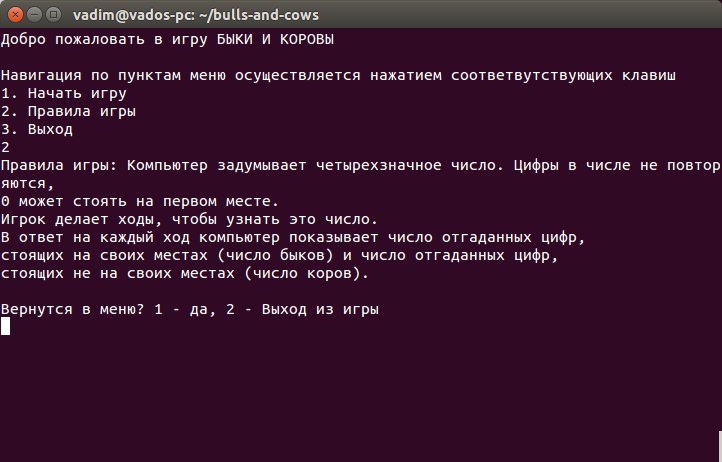
# ОПИСАНИЕ ВЫПОЛНЕННОГО ПРОЕКТА

Общекомандная часть, с примерами работы ПО

Выполненный проект соответствует ТЗ. В общекомандную часть проекта вошли совместные усилия по устранению возникших во время разработки проблем (ошибок, багов и т.д.),

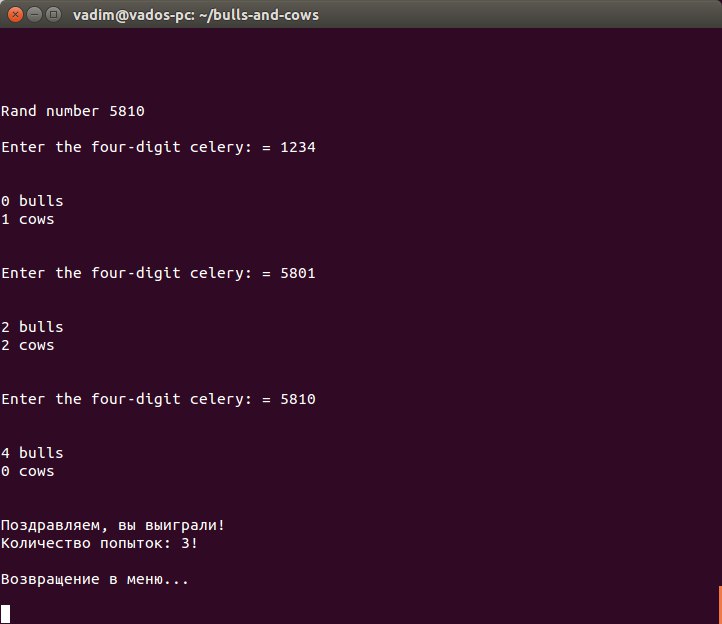
**Примеры работы ПО:**

Приложение начинает работу с приветствия и вывода на экран меню игры:

Если вы не знаете правила игры, можно ознакомиться с ними нажав на клавиатуре цифру 2 и Enter:

Далее есть два варианта: либо вернутся назад в меню(1), либо – выйти из игры(2).

Пункт меню «Начать игру» запускает игру, компьютер генерирует число, и вы начинаете отгадывать:

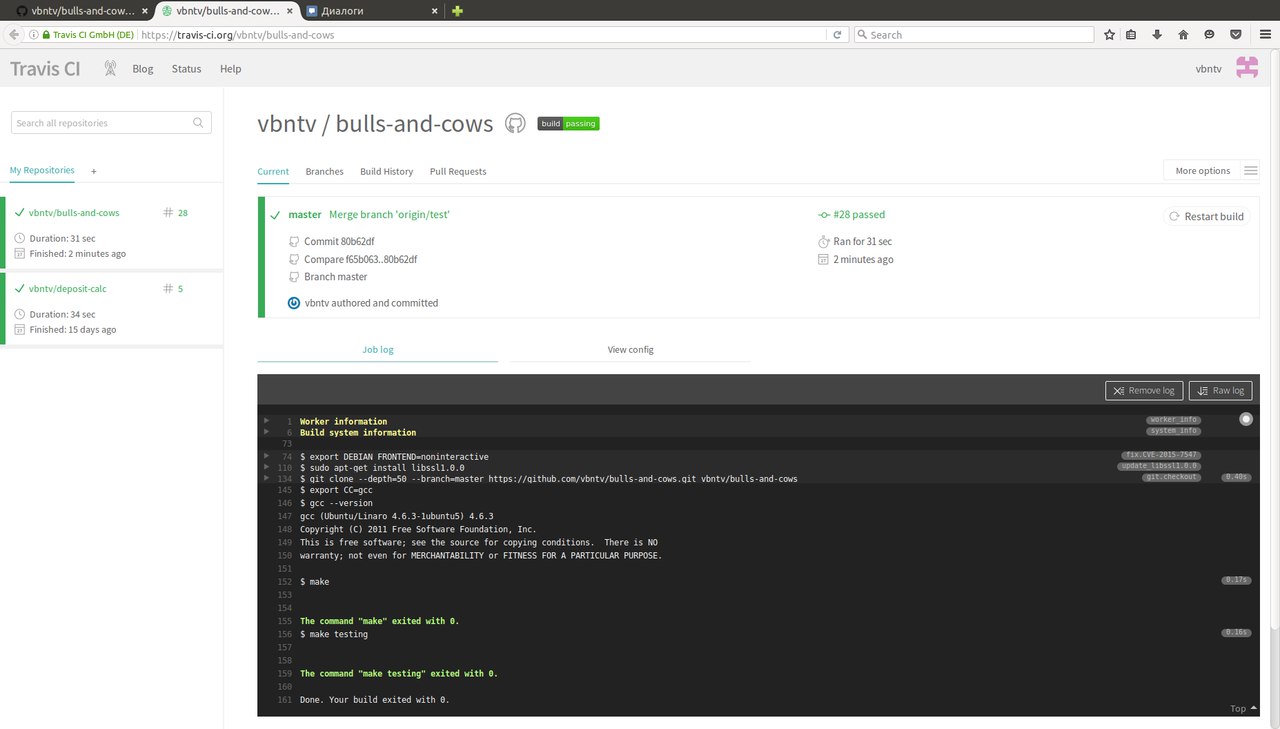


После того, как вы отгадали число, игра выводит сообщение об этом и сообщает количество попыток, за которое вы это число угадали, что полезно для отслеживания собственных результатов.

# C:\Users\Aetterno\Desktop\HOp8k9RAt8A.jpgНа следующей картинке можно увидеть примеры работы функции safe\_input:

# 

Также репозиторий разработки был подключен к сервису непрерывной интеграции Travis CI:



# ЛИЧНЫЙ ВКЛАД В ПРОЕКТ

# 

* Разработка страницы Issues(цели)
* Участие в совместной разработке функции проверки корректности ввода – safe\_input(char \*str )
* Разработка тестов Validation-test и Function-test
* Подключение к сервису Travis.ci

# 

# ПРИЛОЖЕНИЕ. ТЕКСТ ПРОГРАММЫ

|  |
| --- |
|  |
|  | #include <stdio.h>  #include <stdlib.h> |
|  | #include <locale.h> |
|  | #include <ctype.h> |
|  | #include <time.h> |
|  | #include "function.h" |
|  | #include <unistd.h> |
|  |  |
|  |  |
|  | void menu() |
|  | { |
|  | system("clear"); |
|  | printf("Добро пожаловать в игру БЫКИ И КОРОВЫ\n\n"); |
|  | printf("Навигация по пунктам меню осуществляется нажатием соответвутствующих клавиш\n"); |
|  | printf("1. Начать игру\n"); |
|  | printf("2. Правила игры\n"); |
|  | printf("3. Выход\n"); |
|  | int punkt2; |
|  | int punkt; |
|  | scanf("%d", &punkt); |
|  | int a, b, c, d; |
|  | switch(punkt){ |
|  | case 1: |
|  | system("clear"); |
|  | bcrandom(&a, &b, &c, &d); |
|  | bulls\_cows(&a, &b, &c, &d); |
|  | break; |
|  | case 2: |
|  | printf("Правила игры: "); |
|  | printf("Компьютер задумывает четырехзначное число. Цифры в числе не повторяются, \n"); |
|  | printf("0 может стоять на первом месте.\n"); |
|  | printf("Игрок делает ходы, чтобы узнать это число. \n"); |
|  | printf("В ответ на каждый ход компьютер показывает число отгаданных цифр,\n"); |
|  | printf("стоящих на своих местах (число быков) и число отгаданных цифр, \n"); |
|  | printf("стоящих не на своих местах (число коров).\n\n"); |
|  | printf("Вернутся в меню? 1 - да, 2 - Выход из игры\n"); |
|  | scanf("%d", &punkt2); |
|  |  |
|  | switch(punkt2) |
|  | { |
|  | case 1: |
|  | menu(); |
|  | break; |
|  | case 2: |
|  | \_Exit (EXIT\_SUCCESS); |
|  | } |
|  | case 3: |
|  | \_Exit (EXIT\_SUCCESS); |
|  | break; |
|  | } |
|  | } |
|  |  |
|  |  |
|  | int safe\_input(char \*str) |
|  | { |
|  | int test, number; |
|  |  |
|  | test = atoi(str); |
|  |  |
|  | printf("\n"); |
|  |  |
|  | if(str[2] == 0 || str[3] == 0) |
|  | { |
|  | if(str[0] == 0 || str[1] == 0) |
|  | { |
|  | printf("Enter the four-digit celery: = "); |
|  | return 0; |
|  | } |
|  | printf("Incorrect input! Please check rules\n"); |
|  | printf("Enter the four-digit celery: = "); |
|  | return 0; |
|  | } |
|  |  |
|  |  |
|  | if(test > 10000 || test < 100) |
|  | { |
|  | printf("Incorrect input! Please check rules\n"); |
|  | printf("Enter the four-digit celery: = "); |
|  | return 0; |
|  | } |
|  |  |
|  | if(str[0] == str[1] || str[0] == str[2] || str[0] == str[3] || str[1] == str[2] || str[1] == str[3] || str[2] == str[3]) |
|  | { |
|  | printf("Incorrect input! Please check rules\n"); |
|  | printf("Enter the four-digit celery: = "); |
|  | return 0; |
|  | } |
|  |  |
|  | if(isdigit(str[0]) && isdigit(str[1]) && isdigit(str[2]) && isdigit(str[3])) |
|  | { |
|  | number = atoi(str); |
|  | return number; |
|  | } else |
|  | { |
|  | printf("Incorrect input! Please check rules\n"); |
|  | printf("Enter the four-digit celery: = "); |
|  | return 0; |
|  | } |
|  | } |
|  |  |
|  |  |
|  | int bcrandom(int \*a, int \*b, int \*c, int \*d) |
|  | { |
|  | srand(time(NULL)); |
|  | do |
|  | { |
|  | \*a = 0 + rand() % 9; |
|  | \*b = 0 + rand() % 9; |
|  | \*c = 0 + rand() % 9; |
|  | \*d = 0 + rand() % 9; |
|  | } while (\*d == \*a || \*d == \*b || \*d == \*c || \*c == \*a || \*c == \*b || \*b == \*a); |
|  |  |
|  | printf("Rand number %d%d%d%d\n", \*a, \*b, \*c, \*d); |
|  |  |
|  | return 0; |
|  | } |
|  |  |
|  |  |
|  | int bulls\_func(int \*a, int \*b, int \*c, int \*d, int \*fst, int \*scnd, int \*thrd, int \*frth) |
|  | { |
|  | int bulls = 0; |
|  |  |
|  | if (\*fst == \*a) |
|  | bulls++; |
|  | if (\*scnd == \*b) |
|  | bulls++; |
|  | if (\*thrd == \*c) |
|  | bulls++; |
|  | if (\*frth == \*d) |
|  | bulls++; |
|  |  |
|  | return bulls; |
|  | } |
|  |  |
|  | int cows\_func(int \*a, int \*b, int \*c, int \*d, int \*fst, int \*scnd, int \*thrd, int \*frth) |
|  | { |
|  | int cows = 0; |
|  |  |
|  | if (\*fst == \*b || \*fst == \*c || \*fst == \*d) |
|  | cows++; |
|  | if (\*scnd == \*a || \*scnd == \*c || \*scnd == \*d) |
|  | cows++; |
|  | if (\*thrd == \*b || \*thrd == \*a || \*thrd == \*d) |
|  | cows++; |
|  | if (\*frth == \*b || \*frth == \*c || \*frth == \*a) |
|  | cows++; |
|  |  |
|  | return cows; |
|  | } |
|  |  |
|  |  |
|  | int bulls\_cows(int \*a, int \*b, int \*c, int \*d) |
|  | { |
|  | int sc = 0; |
|  | int number = 0, fst, scnd, thrd, frth; |
|  | int bulls, cows; |
|  | char str[100]; |
|  |  |
|  | while (number != (\*a \* 1000 + \*b \* 100 + \*c \* 10 + \*d)) |
|  | { |
|  | fgets(str, sizeof(str), stdin); |
|  | number = safe\_input(str); |
|  | if(number > 0) |
|  | { |
|  | printf("\n"); |
|  | fst = number / 1000; |
|  | scnd = number / 100 % 10; |
|  | thrd = number / 10 % 10; |
|  | frth = number % 10; |
|  |  |
|  | bulls = bulls\_func(a, b, c, d, &fst, &scnd, &thrd, &frth); |
|  | cows = cows\_func(a, b, c, d, &fst, &scnd, &thrd, &frth); |
|  | sc++; |
|  |  |
|  | printf("%d bulls\n", bulls); |
|  | printf("%d cows\n", cows); |
|  | printf("\n"); |
|  | printf("\n"); |
|  |  |
|  | if(bulls == 4) |
|  | break; |
|  | printf("Enter the four-digit celery: = "); |
|  | } |
|  | } |
|  |  |
|  | printf("Поздравляем, вы выиграли!\nКоличество попыток: %d!\n\nВозвращение в меню...\n\n", sc); |
|  | sleep(2); |
|  | menu(); |
|  |  |
|  | return 0; |
|  |  |
|  | } |
|  |  |