|  |  |  |
| --- | --- | --- |
| МИНОБРНАУКИ РОССИИ | | |
| Федеральное государственное автономное  образовательное учреждение высшего образования  «Пермский государственный национальный  исследовательский университет» | | |
|  | Институт компьютерных наук и технологий | |
| **ОТЧЁТ**  по индивидуальной работе №1  по дисциплине «Языки программирования»  “Быки и коровы” | | |
|  | | Работу выполнил  студент группы ИТ-12-2024 1 курса  Мастыко М.В.  «20» Января 2024 г. |
| Работу проверил  Кнутова Н.С.  «20» Января 2024 г. |
| Пермь 2024 | | |

Оглавление

[Постановка задачи 3](#_Toc187955197)

[Алгоритм решения 3](#_Toc187955198)

[Структуры: 3](#_Toc187955199)

[Функции: 3](#_Toc187955200)

[Тестирование 6](#_Toc187955201)

[Код программы 8](#_Toc187955202)

# Постановка задачи

Требуется написать программный код для игры под названием “Быки и коровы”. В саму программу добавить проверки на правильность ввода и дружественный интерфейс.

Суть игры:

Загадывается 4-значное число, состоящее из уникальных чисел. Игрок, который отгадывает это число, делает свои предположения. Быки – кол-во цифр, стоящих на своих местах. Коровы – кол-во цифр, присутствующих в загаданном числе, но стоящих не на своих местах. С помощью исходя из быков и коров, игрок делает следующее предположение с целью отгадать заданное число.

# Алгоритм решения

## Структуры:

Перечисление (enum) game\_state – включает в себя состояния игры: exit\_game, repeated\_game, continue\_game.

Используется для определения состояния игры, чтобы в циклах вида do{}while можно было определить из какого цикла выходить и когда. Данная структура является оптимальным решением, когда программа включает в себя не только саму игру, но и выход из игры, продолжение игры, и повтор игры при желании игрока.

## Функции:

* check\_input- тип bool, проверяет правильность ввода числа.

На вход подается строка s, чтобы полностью отследить что введет игрок. Для проверки числа на уникальность цифр в нем используется множество number. Цикл проверяет чтобы элемент s[i] строки имел в таблице символов ASCII номер, лежащий в диапазоне от 48 до 57 (цифры от 0-9). И в этом же цикле s[i] добавляется во множество number. После цикла делаются проверки на размер вводимого числа (четырехзначное), на ведущие нули и на уникальность числа путем проверки размера мн-ва – оно должно равняться 4. Если при какой-либо проверки число не соответствует нужному формату, то выводится соответствующая ошибка и функция возвращает 0. В ином случае функция возвращает 1.

* Random – тип string, генерирует и возвращает 4-х значное число с уникальными цифрами.

Объявляется пустая строка digit в которую будут добавляться цифры, пока её размер не будет равен 4. Каждая новая цифра генерируется с помощью rand() и таблицы ASCII. Делается проверка на ведущий ноль: если размер digit равняется ноль, и сгенерированная цифра равняется нулю, то цифра генерируется заново, пока она не будет равняться любой другой цифре, но не ноль. Затем проверяется есть ли такое число с помощью find, если новая цифра не найдена в digit, то есть она уникальна, то цифра добавляется в digit.

* countBullsAndCows – тип void, считает кол-во быков и коров.

В функцию через ссылки подаются переменные bulls, cows, которые соответственно будут считать быков и коров. В первом цикле считаются быки. Во втором считаются коровы с помощью вложенных циклов, сравнивая каждую цифру из предполагаемого числа с каждой цифрой из изначально загаданного числа. Функция типа void ничего не возвращает, лишь по ссылке переназначает bulls и cows, которые описаны в main.

* play\_again – тип void, предлагает сыграть в игру еще раз.

Игроку предлагается вариант 1-сдаться, либо 2-продолжить играть. Ввод от пользователя производиться с помощью строковой переменной choice. В цикле while проверяется корректность ввода с помощью проверки размеры choice и проверки символов с помощью таблицы ASCII. Если игрок выбрал 2-выйти, то по ссылке передаем в game значение exit\_game, если выбрал 1- сыграть еще раз, то в game передается значение repeat\_game.

* rules – тип void, выводит правила
* repeated\_input – тип void, для повторного вода чтобы сократить код в main.
* Main – главная функция, содержащая основной код программы.

Описывается переменная game, типа game\_state (описание структуры описано в части Структуры). Выводится приветствие и правила игры.

Первый основной цикл типа do{}while(условие) определяет конец игры и повтор игры. Цикл повторяется до тех пор, пока переменной game не будет назначено значение game\_exit -то есть игрок захотел выйти. С начала цикла назначается game состояние continue\_game, объявляются нужные переменные. Игроку предлагается выбрать противника: компьютер или другой игрок. Выбор запоминается в строковую переменную choice. Затем проверяется правильность ввода с помощью размера choice и таблицы символов ASCII. В зависимости от выбора, число загадывается с помощью функции Random, либо дает ввести число самому и проверяет его правильность с помощью функции check\_input. С помощью system("cls") очищаем строку, чтобы не видеть загаданного числа. Вводятся переменные guess (предположение игрока), bulls, cows, attempts (счетчик попыток).

Начинается второй (вложенный в первый) основной цикл, который отвечает за отгадывание числа. Длиться до тех пор пока либо число не будет отгадано (bulls !=4), либо пока game\_state остается в состоянии continue\_game. В начале цикла вводится предположение и проверяется на правильность с помощью функции check\_input и while (пока ввод не будет правильным – продолжать вводить). Даже если игрок произвел некорректный ввод, то попытка засчитывается. Затем в функции countBullsAndCows cчитаются быки и коровы. Быки и коровы обязательно выводятся на экран.

Также добавлена возможность сдаться каждые 7 попыток. Если кол-во попыток кратно 7 – то выполняем следующий алгоритм.

Выводится само предложение сдаться, игрок вводит свой выбор – сдаться, продолжить, или посмотреть правила. Ввод проверяется на правильность. С помощью switch распределяем случаи в зависимости от choice.

1. Сдаться (1): выводит правильный ответ и запускает функцию play\_again, которая предлагает сыграть в игру еще раз. В ходе выполнения play\_again назначается дальнейшее состояние игры.
2. Продолжить игру (2): продолжает игру
3. Показать правила (3): выводит правила, предлагает сдаться или продолжить.

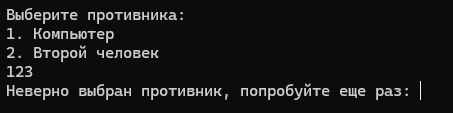
После окончания части алгоритма со сдачей, идет проверка на выигрыш игрока, если кол-во быков равняется 4 (все цифры стоят на своих местах), то выводится изначальное загаданное число, кол-во попыток и запускается функция play\_again.

Смотрятся условия двух основных циклов, и в зависимости от значения в переменной game (состояние игры) и количества быков. Игра либо повторяется с момента выбора игрока, либо выходит.

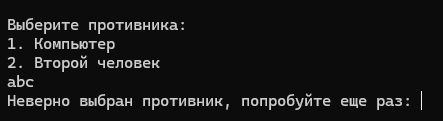
# Тестирование

* Выбор противника: надо выбрать либо 1 либо 2

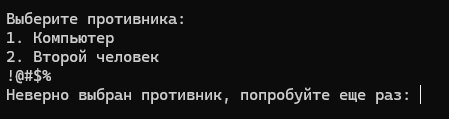
123 - числа (не равняющиеся 1 и 2)



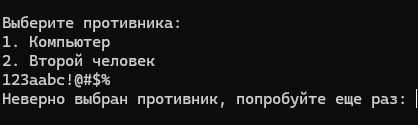
abc – буквы



!@#$% -спец. Символы



123abc!@#$% - все вместе



– Выводит “Неверно выбран противник, попробуйте еще раз: “

* Предположение числа и его загадывание вторым игроком: надо ввести четырехзначное число, с уникальными цифрами и без ведущих нулей:

123 или 12345 – “Число должно быть 4-х значным, то есть состоять из 4 цифр!”







(abc) , (!@#$%) , (123abc!@#$%) - “Вводимое значение должно содержать только цифры от 0 до 9”









0123 – “Первой цифрой не может быть 0!”





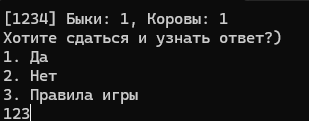
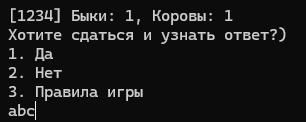
1232 – “Все цифры в числе должны быть уникальны!”

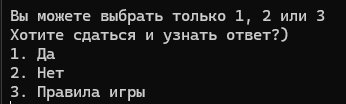
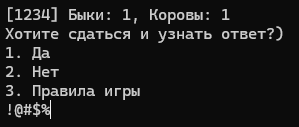




* Предложение сдаться:

Такой же набор входных данных, как и при тесте проверке выбора противника (числа, не равняющиеся 1,2,3, буквы, спец. символы)

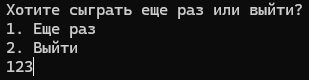
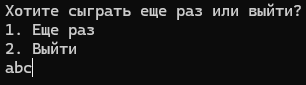
 

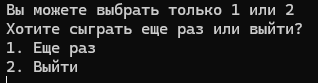
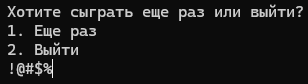


- Выводит “Вы можете выбрать только 1, 2 или 3 Хотите сдаться и узнать ответ?)”

* Предложение повторить игру или выйти из игры:

Такой же набор входных данных, как и при тесте проверке выбора противника (числа, не равняющиеся 1,2, буквы, спец. символы)



-Выводит “Вы можете выбрать только 1 или 2” и опять предлагает тот же выбор.

# Код программы

<https://github.com/MineMoon/BullsAndCows>