|  |
| --- |
| Курсовая работа по программированию |
| Игра “Крестики-нолики” против компьютера |
| [Подзаголовок документа] |

|  |
| --- |
| Выполнил студент группы ИП-514  Коннов Илья  Проверил  Sample text  [Дата] |

Оглавление

Введение.................................................................................................................................2

1. Анализ требований и разработка спецификаций……......................................................3

1.1 Основные термины.………………..…….………………….............................................................3

1.2. Пользовательский интерфейс……...................................................................................3

1.3. Формат входных данных……………………...........................................................................4

1.4. Формат выходных данных..............................................................................................4

2. Проектирование……………………...........................................................................................4

2.1 Функции.............................................................................................................................4

2.2. Алгоритмы........................................................................................................................5

2.3 Баги и их решения………………………………………………………………………………………………………..6

3. Реализация..........................................................................................................................5

3.1. Программное окружение................................................................................................6

Заключение.............................................................................................................................6

Литература..............................................................................................................................6

**Введение**

**Крестики-нолики** — настольная логическая игра на 9-клеточной доске для двух соперников, сочетающая в себе элементы искусства, науки и спорта. Название берет начало из русского языка: крестики и нолики, что означает крестики и нолики.

Игра ведется на табличке 3х3, которую, при желании, можно пронумеровать латинскими буквами от a до c по горизонтали и цифрами от 1 до 3 по вертикали. У игроков вначале имеется абсолютно пустая таблица, такая, как показана на диаграмме, письменная принадлежность и надежда на победу. Один из игроков условно именуется «крестики», другой «нолики».

Партию начинает играющий крестиками. Он ставит его на любую из клеточек таблицы. Затем играющий ноликами ставит свою фигуру на любое из свободных полей. Далее крестиками и ноликами по очереди заполняются свободные клетки.

Итог игры:

Выигрыш фиксируется, если крестиками или ноликами полностью заполняется вертикаль, горизонталь или диагональ. Победителем признается обладатель фигур, которыми заполнился ряд. Победа одного из участников фиксируется, если его соперник сдался, убежал, был похищен инопланетянами, или по иным причинам отказывается продолжать игру. Победа присуждается игроку, если его соперник закрасил или порвал всю таблицу, умышленно сломал карандаш и отказался предоставить замену, попытался незаметно поставить лишний крестик или нолик либо совершил иной поступок, расцененный судьей, контроллером, секретарем, конспектирующим, судом присяжных или независимым наблюдателем, как «свинство». Также победа присуждается, если один из игроков просрочил время. (По правилам, на обдумывание одного хода дается 44 часа 32 минуты 11 секунд). Естественно присуждается другому игроку. Ничья фиксируется, если заполнены все девять клеток, и не одного ряда, заполненного только крестиками или только ноликами, не наблюдается. Также объявляется ничья, если в ходе некоторого внешнего вмешательства была утеряна позиция, и не один из участников не смог вспомнить расстановки символов. Ничья объявляется, если у игроков нет возможности сделать ход (например, сломался карандаш). Ничья фиксируется, если игроки согласились на ничью, то есть один из игроков при своём ходе предложил ничью, другой её принял. Для предложения ничьей достаточно сказать «ничья». Если противник делает ход, не ответив на предложение ничьей, оно считается отвергнутым. Так же объявляется ничья после трехкратного повторения позиции.

1. **Анализ требований и разработка спецификаций**

Требуется написать игру, в которой бы пользователь играл с компьютером на стандартном для крестиков-ноликов поле 3х3. Точные требования не указаны, поэтому, в моем случае, это будет консольное приложение, в котором пользователь, для выбора места под “Х”, будет использовать числа от 1 до 9. Игра будет иметь 3 уровня сложности: Easy, Medium и Hard. В случае Easy – компьютер ходит в случайную ячейку, Medium – компьютер уже ходит осмысленно, Hard – компьютер непобедим. Обязательное требование для завершения игры с чьей-то победой – расположение крестиков либо ноликов по горизонтали, вертикали, либо диагонали, в противном случае – ничья.

При вводе пользователем/компьютером цифры для расположения “X”/”O” также необходимо предусмотреть, чтобы не был введен какой-то другой символ, отличный от диапазона целых чисел [1;9].

**1.1. Основные термины**

Далее в работе используются следующие термины:

1. Решение ― исходный код программы, предположительно

решающей задачу по программированию.

2. Задание ― данные, которые предоставлены в теме 30 курсовой работы по программированию.

**1.2. Пользовательский интерфейс**

Так как приложение консольное, пользовательский интерфейс предоставляет обычный терминал. Главное и стартовое окно – пользовательское меню, где пользователю предоставляется, с помощью кнопок целочисленного диапазона [1; 4], возможность ознакомится с правилами игры, прочитать о разработчике, начать игру и выйти из игры. В самой игре каждая ячейка поля прономерована цифрами от 1 до 9 так, чтобы пользователю было понятно куда он может поставить свой “X”. Во время выбора уровня сложности вы так же можете вернуться в главное меню, выбрав соответствующий пункт.

* 1. **Формат входных данных**

Входные данные имеют целочисленный тип int. Пользователю предоставляется возможность выбирать пункты меню с помощью цифр, а также в самой игре вы вводите целое число для выбора клетки, в которую ставится “Х”.

* 1. **Формат выходных данных**

Среди выходных данных: литералы (строки, окруженные двойными кавычками), с помощью которых идет взаимодействие с пользователем текстом, а также целочисленный тип int, с помощью которого печатается поле со свободными ячейками.

1. **Проектирование**

В этом разделе представлена вся структура проекта: функции и алгоритмы, используемые в коде. А также возникшие при написании кода трудности.

**2.1. Функции**

int main() – основная функция, запускающая функцию gameMenu(), отвечающую за меню в цикле и берущая из неё логическую переменную isGameActive. Если переменная isGameActive имеет ложное значение – цикл в main() завершается и функция возвращает значение 0.

void gameMenu(bool \*isGameActiveP)– функция, отвечающая за вывод меню. В этой функции так же вводится значение от 1 до 4, где 1 – переход к самой игре, 2 – просмотр инструкции, 3 – вывод сведений о разработчике, 4 – завершение программы (присваивание переменной isGameActive значение false).

void mainGame(bool \*isGameActiveP) – в этой функции проходит основная логика игры.

void instruction(bool \*isGameActiveP) – в этой функции содержится вся информация о правилах игры.

void about(bool \*isGameActiveP) – здесь вы можете узнать информацию о разработчике.

void title() – содержится название игры, написанное в виде символов “X”.

В функциях hardDifficult() и mediumDifficult() описан алгоритм, по которому компьютер делает ход.

**2.2. Алгоритмы**

Для выбора следующего игрока используется следующий алгоритм: player = i%2+1, где i – номер хода, % - остаток от деления. Таким образом первый ход первого игрока будет player = 0%2+1 = 1, первый ход второго игрока player = 1%2+1 = 2, второй ход первого игрока player = 2%2+1 = 1, и т.д. Таким образом можно однозначно определить чей ход текущий (в моем случае player = 1 – игрок, player = 2 – компьютер).

Для того, чтобы пользователь (да и компьютер) не мог ввести число, не входящее в диапазон [1;9], был сделан цикл с таковым условием. Для того, чтобы пользователь не мог ввести какие-нибудь другие символы, отличные от целых чисел, ввод для пользователя был сделан в массив типа char, а затем, с помощью функции atoi() значение массива конвертировалось в целое число. В случае, если функция atoi() возвращало ненулевое значение, можно было смело говорить, что пользователь ввел целое число.

Чтобы переменная “go”, в которую пользователь и компьютер заносят числа от 1 до 9, могла преобразовать цифру в элемент в массиве, был изобретен следующий алгоритм:

row = --go/3; // строка

column = go%3; //столбец

Рассмотрим, например, случай, что пользователь ввёл “5”:

Строка тогда будет row = 4/3 =1

Столбец column = 4%3 = 1;

Тогда Х поместится:

0| 0 1 2

1| 0 X 2 - строка 1, столбец 1 как раз соответствует введенной цифре 5.

2| 0 1 2

Для того, чтобы компьютер делал осмысленные ходы, в функциях mediumDifficult() и hardDifficult были прописаны таковые условия: во-первых, идет проверка на случай, если сам компьютер поставил два “O” так, что на следующий ход может поставить еще один “O” и победить, во-вторых, идет проверка на случай, если пользователь поставил “X” так, что на следующий ход может победить. В-третьих, добавлены условия наподобие если 1 ячейка занята “O”, 4 и 7 свободна, то компьютер ставит “O” в 7. Конкретно в случае с уровнем сложности “Hard”, при условии, что пользователь поставил “X” не в центр (ячейка 5), то компьютер ставит туда, а если пользователь поставил в центр, то компьютер ставит “O” по углам.

**2.3 Баги и их решения**

Первый баг, с которым пришлось столкнуться – бесконечный цикл при вводе во время игры символов, отличающийся от типа int в целочисленную переменную go. Это возникало из-за того, что стоял цикл while с условием go<0&&go>9, таким образом, при нецелочисленном вводе в целочисленную переменную scanf() не считывал символ и цикл зацикливался. Одно из возможных решений данной проблемы – проверка сколько аргументов считал scanf(). Но мне данный способ не помог по невыясненным причинам, поэтому я использовал функцию atoi() (о которой написано на странице 5), с помощью него проверял, введено ли число или введен символ. Таким образом баг удалось решить.

Так же был ряд багов из-за невнимательного написания алгоритма, в особенности при создании функции hardDifficult.

Все баги были исправлены и на данный момент программа работает без каких-либо ошибок.

**3. Реализация**

В этом разделе описано программное окружение системы.

**3.1. Программное окружение**

Проект был написан на языке программирования “Си”, с использованием библиотек <stdio.h>, <stdlib.h>, <stdbool.h>, <time.h> в среде разработки Dev C++, использующей компилятор компилятор MinGW. Для контроля версий использовался GitHub, также проект доступен в репозитории GitHub’a по ссылке: <https://github.com/konnov-007/TerminalTic-Tac-Toe>.

**Заключение**

Была разработана игра “Крестики-нолики” против компьютера с тремя уровнями сложности, в которой были устранены возможные баги при вводе, сделано меню, добавлена основная информация и возможность выхода из игры.

**Литература**

1. atoi – Википедия [https://ru.wikipedia.org/wiki/Atoi]
2. stdbool.h – Википедия [https://ru.wikipedia.org/wiki/Stdbool.h]
3. Си-строки – CppSudio [http://cppstudio.com/post/9567]