содержание

ВВЕДЕНИЕ ………………………………………………………………………...2

1. ПОСТАНОВКА ЗАДАЧИ ………………………………………………………3

2. РАЗАРАБОТКА АЛГОРИТМОВ……………………………………………….4

3. РАЗРАБОТКА ПРОГРАММЫ………………………………………………….7

3.1. Выбор средств программирования …………………………………………..7

3.2. Разработка функций программы……………………………………………..7

4. ТЕСТИРОВАНИЕ ……………………………………………………………...10

4.1. Описание входных и выходных данных………………………………….....10

4.2. Результаты тестирования……………………………………………………..11

ЗАКЛЮЧЕНИЕ…………………………………………………………………….21

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ………………………………22

ПРИЛОЖЕНИЕ А ТЕКСТ ПРОГРАММЫ

ПРИЛОЖЕНИЕ Б ГРАФИЧЕСКИЙ МАТЕРИАЛ

# ВВЕДЕНИЕ

Целью данной курсовой работы является разработка игры «Крестики-нолики». Предметом работы является написание программы, содержащей в себе все необходимые элементы для правильной и интересной игры. Программа должна выводить игровое поле, спрашивать у пользователя о его ходе, делать свой ход, анализировать результат игры и выводить сообщение о том, кто победил.

Для создания данной программы необходимо пройти ряд этапов: анализ постановки задачи, разработка структуры ПО, разработка алгоритмов функционирования, тестирование и анализ результата проектирования, построение и тестирование программы.

Проектирование программы является одной из наиболее сложных и ответственных задач, связанных с созданием правильного алгоритма работы. В результате его решения программа должна анализировать каждый следующий ход игрока и рационально отвечать пользователю.

1. ПОСТАНОВКА ЗАДАЧИ

Необходимо разработать программный комплекс по обработке заданного множества данных. Множество данных представляет собой массив записей. Программа должна быть реализована в виде нескольких модулей. Требуется обеспечить реализацию следующих запросов к заданному множеству структурированной информации:

1. Ввод информации из текстового файла в массив указателей на записи.

2. Добавление новых элементов в конец массива.

3. Просмотр всех элементов массива.

4. Вывод информации из массива в файл.

5. Сортировка записей по выбранному полю с созданием индексных файлов.

6. Корректировка полей выбранного элемента.

7. удаление выбранного элемента.

8. Поиск записи по значению ключевого поля.

9. Вставка новой записи перед выбранной.

10. Выборка записей и вычисление среднего на множестве тех элементов, которые попадают в заданный диапазон по заданному полю (поле числового типа).

11. Фильтр записей по условию (значение выбранного поля больше или меньше заданного).

2. РАЗРАБОТКА АЛГОРИТМОВ

Этот этап один из самых важных. После того, как мы осуществили постановку задачи, для дальнейшей работы нам будет необходимо разработать алгоритмы. Разработка алгоритма – метод для создания математического способа решения проблем, т.е. на основе выбранного метода записывается последовательность действий, приводящих к решению задачи. Успешная разработка алгоритма позволяет избежать многих ошибок, поскольку именно на этом этапе определяется логика будущей программы.

Алгоритм ввода информации из файла:

1. Открываем файл для чтения.

2. Читаем строки.

3. Каждые пять строк из файла заносим в массив.

4. Преобразовываем прочитанные строки в поля структуры.

5. Добавляем элемент в конец массива.

6. Увеличиваем счетчик элементов массива.

7. Когда будет достигнут конец файла, файл закрывается, и работа завершается.

Алгоритм добавления нового элемента:

1. Запрашивается информация о школьнике.
2. Открывается файл.
3. Структура записывается в файл.
4. Файл закрывается.

Алгоритм вывода элементов массива:

1. Запускаем цикл для всех элементов массива.

2. Производим вывод полей структуры на экран в одну строку.

Алгоритм корректировки полей элементов массива:

1. Выбираем номер элемента, для корректировки полей.

2. Выбираем поле для корректировки.

3. На основании выбора вносим новое значение выбранного поля.

Алгоритм поиска записи по значению ключевого поля:

1. Выбираем поле для поиска.

2. Вводим значение для поиска.

3. Запускаем цикл для всех элементов массива.

4. Функция возвращает номер структуры.

Алгоритм вывода информации в текстовый файл:

1. Открываем файл для записи.

2. Запускаем цикл для всех элементов массива.

3. Каждое поле элемента массива преобразуется в символьное значение и записывается в строку файла.

4. Закрываем файл.

Алгоритм удаления выбранного элемента:

1. Запускаем цикл для нахождения элемента в массиве.

2. Удаляем элемент из массива.

3. Сдвигаем все элементы массива вверх.

4. Уменьшаем счетчик элементов массива.

Алгоритм сортировки записей по выбранному полю:

1. Выбор поля для сортировки

2. Последовательно сравниваем элементы попарно.

3. Если порядок в паре неверный, выполняется обмен элементов.

4. Проходы по массиву повторяются n-1 раз, пока на очередном проходе не окажется, что обмены больше не нужны.

Алгоритм вставки новой записи перед выбранной:

1. Открываем файл.

2. Выбираем элемент массива.

3. Записываем элементы массива до выбранного.

4. Вводим новую структуру.

5. Записываем ее в файл.

6. Записываем оставшиеся элементы массива в файл.

7. Закрываем файл.

Алгоритм выборки записей и вычисления среднего на множестве тех элементов, которые попадают в заданный диапазон по заданному полю (поле числового типа):

1. Вводим границы диапазона.

2. Выбираем поле, по которому будем считать среднее значение.

3. Организовываем цикл для считывания суммы значений выбранного поля всех элементов.

4. Делим полученное значение на количество элементов массива в заданном диапазоне.

5. Выводим полученное значение.

Алгоритм фильтрации записей по условию (значение выбранного поля больше или меньше заданного):

1. Выбираем поле.

2. Выбираем первый параметр для фильтра.

3. Вводим второй параметр для фильтра.

4. Запускаем цикл для всех элементов массива.

3. Выводим элементы массива, удовлетворяющие условию, заданному ранее.

3. РАЗРАБОТКА ПРОГРАММЫ

3.1. Выбор средств программирования

Используемая среда разработки VisualStudio, OC Windows 10, язык программирования – С++.

Используемые библиотеки: iostream — [заголовочный файл](https://ru.wikipedia.org/wiki/%D0%97%D0%B0%D0%B3%D0%BE%D0%BB%D0%BE%D0%B2%D0%BE%D1%87%D0%BD%D1%8B%D0%B9_%D1%84%D0%B0%D0%B9%D0%BB) с классами, функциями и переменными для организации ввода-вывода в языке программирования [C++](https://ru.wikipedia.org/wiki/C%2B%2B); string — класс с методами и переменными для организации работы с символами в языке программирования [C++](https://ru.wikipedia.org/wiki/C%2B%2B); iomanip — этот файл содержит манипуляторы для работы с потоковыми операциями; windows.h — этот файл содержит множество определений типов данных, макросов, прототипов функций, констант и т. д..

3.2. Разработка функций программы

**Функция void play\_area(char data[3][3])**

Входные параметры: массив символов.

Назначение: вывод игрового поля в консоль.

Возвращаемое значение: -.

**Функция int get\_key()**

Входные параметры: -.

Назначение: .

Возвращаемое значение: .

**Функция bool matrix\_writing(char matrix[3][3], int index, char temp)**

Входные параметры: массив символов, индекс элемента, символьная переменная.

Назначение: .

Возвращаемое значение: переменная типа bool.

**Функция int y\_or\_n(char data[3][3], int index)**

Входные параметры: массив символов, индекс элемента.

Назначение: .

Возвращаемое значение: -.

**Функция bool person\_action(char data[3][3])**

Входные параметры: массив символов.

Назначение: запись действий пользователя.

Возвращаемое значение: переменная типа bool.

**Функция bool check\_gorizontal(char data[3][3], char flag1, char flag2)**

Входные параметры: массив символов, символьная переменная, символьная переменная.

Назначение: проверка горизонталей игрового поля.

Возвращаемое значение: переменная типа bool.

**Функция bool check\_vertikal(char data[3][3], char flag1, char flag2)**

Входные параметры: массив символов, символьная переменная, символьная переменная.

Назначение: проверка вертикалей игрового поля.

Возвращаемое значение: переменная типа bool.

**Функция bool check\_diagonals(char data[3][3], char flag1, char flag2)**

Входные параметры: массив символов, символьная переменная, символьная переменная.

Назначение: проверка диагоналей игрового поля.

Возвращаемое значение: переменная типа bool.

**Функция bool other\_fields(char data[3][3])**

Входные параметры: массив символов.

Назначение: .

Возвращаемое значение: переменная типа bool.

**Функция bool person\_win(char data[3][3])**

Входные параметры: массив символов.

Назначение: .

Возвращаемое значение: переменная типа bool.

**Функция bool komputer\_action(char data[3][3])**

Входные параметры: массив символов.

Назначение: анализ игрового поля для следующего хода компьютера.

Возвращаемое значение: переменная типа bool.

**Функция void game()**

Входные параметры: -.

Назначение: запуск игры «Крестики-нолики».

Возвращаемое значение: -.

**Функция void menu()**

Входные параметры: -.

Назначение: вывод меню в консоль.

Возвращаемое значение: -.

4. ТЕСТИРОВАНИЕ

4.1. Описание входных и выходных данных

При запуске программы на экране появляется меню для игры. В ходе выполнения программы проходят игры между пользователем и компьютером, в конце каждой партии выводится результат игры.

4.2. Результаты тестирования

При запуске программа должна выводить на экран меню для игры (см. рисунок 4.1).

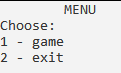


Рисунок 4.1

В пункте 1 запускается игра и пользователю предоставляется право первого хода (см. рисунок 4.2).

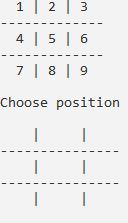


Рисунок 4.2

Пользователь выбирает позицию, куда записать «крестик» (см. рисунок 4.3).

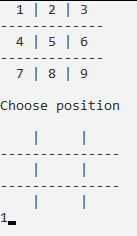


Рисунок 4.3

Программа записывает ход пользователя и сразу выводит на экран. Также выполняется проверка ответа пользователя (см. рисунок 4.4).

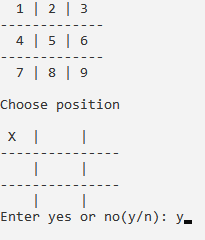


Рисунок 4.4

Программа делает свой ход и предоставляет пользователю выбор для следующей позиции (см. рисунок 4.5).

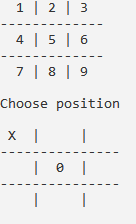
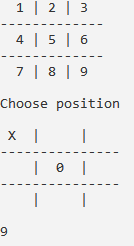
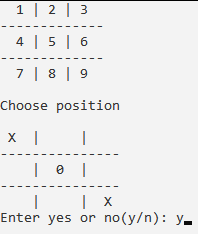
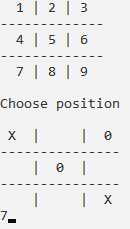
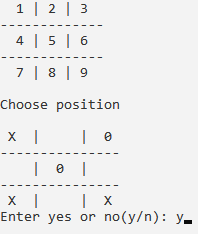


Рисунок 4.5

Далее игра продолжается аналогично предыдущим ходам, пока кто-либо не выиграет игру (см. рисунок 4.6)

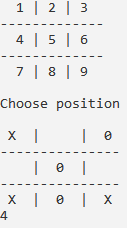
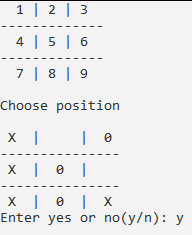
 

Рисунок 4.6

Когда игра закончена, программа выводит сообщение о том, кто выиграл (см. рисунок 4.7)

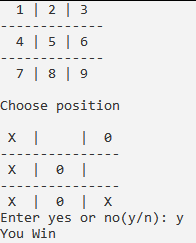


Рисунок 4.7

После окончания игры на экран выводится меню для выбора действий (см. рисунок 4.8).

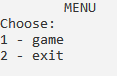
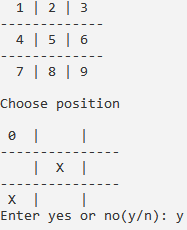
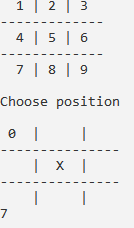
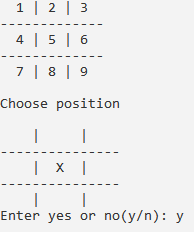
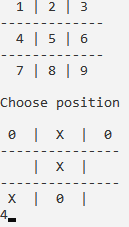
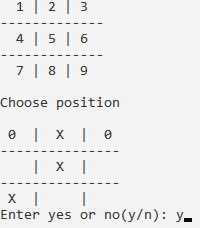
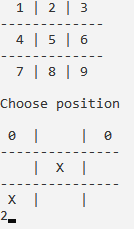


Рисунок 4.8

Игра может также закончится «вничью» (см. рисунок 4.9).





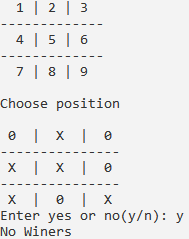
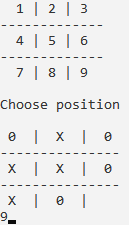
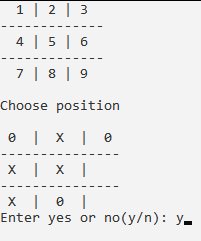
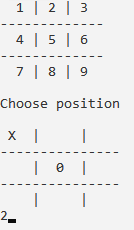
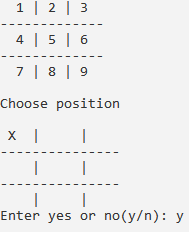
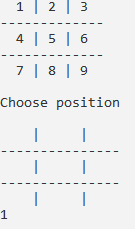
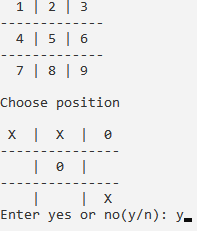
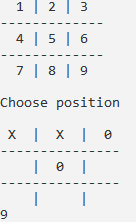
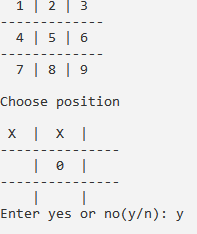


Рисунок 4.9

Игра может также закончится победой компьютера (см. рисунок 4.10).





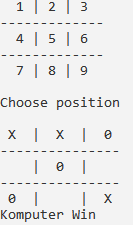
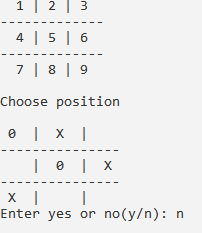
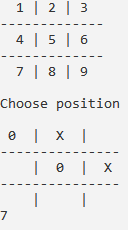


Рисунок 4.10

В программе реализована проверка хода пользователя и есть возможность переписать свой ход (см. рисунок 4.11).



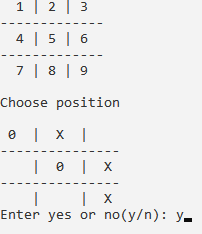
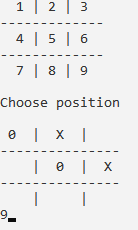


Рисунок 4.11

В пункте 2 реализован выход из программы (см. рисунок 4.12).

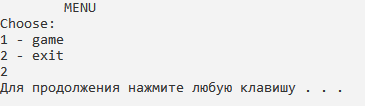


Рисунок 4.12

ЗАКЛЮЧЕНИЕ

В результате выполнения курсовой работы была получена работающая программа, в которой были реализованы все функции в соответствии с предоставленным заданием. Программа работает исправно, вероятность возникновения ошибок и исключений сведена к минимуму. Все условия и ограничения были соблюдены. Программа предоставляет удобную среду для игры в «Крестики-нолики». Играть становится проще благодаря встроенному функционалу.

В ходе выполнения работы были практически закреплены знания о работе с инструментами файлового ввода и вывода, основными языковыми средствами языка C++.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. ГОСТ 19.504 – 79. ЕСПД. Руководство программиста. Требования к содержанию и оформлению.

2. ГОСТ 2.105-95. ЕСКД. Общие требования к текстовым документам.

3. ГОСТ 19.701-90 ЕСПД. Схемы алгоритмов, программ, данных и систем.