

Министерство образования Российской Федерации
Пензенский государственный университет
Кафедра «Вычислительная техника»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе
по курсу «Программирование»
на тему «Игра в Крестики
нолики»

Выполнил:

студент группы 25BBB1

Евстратов В. С.

Принял:

к.т.н., доцент Генералова А.А.

Пенза, 2026

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Факультет Вычислительной техники
Кафедра "Вычислительная техника"

"УТВЕРЖДАЮ"

Зав. кафедрой ВТ

М.А. Митрохин

« 04 » 03 2026 г

ЗАДАНИЕ

на курсовое проектирование по курсу

Программирование

Студенту Светлову Владимиру Сергеевичу Группа 25ВВВ1

Тема проекта: *Разработка программы сложной структуры методом нисходящего программирования.*

Исходные данные (технические требования) на проектирование

Итого в "Крепости жизни" - право выбора вида просмотра изображения. Программа формирует право выбора вида и рассчитывает оптимальные значения. В программу записываются результаты.

Обязательные требования к программе:

1. Многомодульность.
2. Использование сложных типов данных – структур, списков, файлов.
3. Режим работы видеосистемы – текстовый / графический.
4. Устройство ввода информации – клавиатура, мышь.
5. Пользовательский интерфейс должен быть построен на основе меню и панели инструментов.
6. Наличие заставки.
7. Операционная система MS Windows.
8. Язык программирования – Си и Ассемблер.
9. Среда разработки ПО – Microsoft Visual Studio.

Объем работы по курсу

1. Расчетная часть

Разработка программы.

2. Графическая часть

Схема данных, схема ресурсов системы, схема работы системы, иерархическая структура программы, схема взаимодействия программы.

3. Экспериментальная часть

Отладка программы.

Срок выполнения проекта по разделам

В соответствии с графиком.

Дата выдачи задания “04” 03

Дата защиты проекта “08” июня 2026

Руководитель Г. В. Векерев

Задание получил “04” 03 2026г.

Студент Васильев В.С.

Содержание

Введение	3
Постановка задачи	4
Выбор решения	5
Описание разработки программы.....	7
Отладка и тестирование.....	9
Описание программы	9
Описание игровой логики.....	27
Приложение А Листинги программы	36
Приложение В Результаты работы программы.....	57
Приложение С Результат работы программы.....	58

Введение

На сегодняшний день компьютерные игры являются не только средством развлечения, но и важным инструментом для развития логического мышления, изучения алгоритмов и программирования. Игра «Крестики-нолики» — это классическая логическая игра, известная с детства. Её реализация на языке программирования позволяет наглядно продемонстрировать такие важные концепты, как работа с графическим интерфейсом пользователя (GUI), обработка событий мыши, реализация искусственного интеллекта (пусть и простого), работа с файловой системой для сохранения статистики, а также использование низкоуровневых возможностей языка (ассемблерные вставки) для генерации случайных чисел.

Компьютерные технологии прочно вошли в нашу жизнь, и умение создавать даже простые интерактивные приложения является важным навыком для будущего специалиста в области вычислительной техники. Данная работа направлена на закрепление теоретических знаний и получение практических навыков разработки многофункциональных приложений в среде Windows с использованием Win32 API..

В рамках курсовой работы разрабатывается программа, моделирующая игру человека против компьютера. Пользователь имеет возможность выбирать очередность хода, а программа рассчитывает свои ходы, основываясь на простом алгоритме, который пытается выиграть или заблокировать игрока. Ключевой особенностью является возможность сохранения и просмотра таблицы рекордов, что позволяет отслеживать прогресс игрока.

Постановка задачи

Необходимо разработать программу для игры в «Крестики-нолики» между человеком и компьютером. Программа должна иметь интуитивно понятный графический интерфейс (GUI), управляемый с помощью мыши. Требуется реализовать следующий функционал:

Графическое игровое поле 3×3
Отображение ходов игрока (крестики) и компьютера (нолики) с помощью графических символов.

Определение победителя и ничьей
Автоматическое завершение игры и вывод соответствующего сообщения.

Право первого хода. Право первого хода всегда имеет пользователь (игрок ставит крестик). Это упрощает правила и обеспечивает интерактивность.

Искусственный интеллект компьютера. Компьютер должен рассчитывать свои ходы, используя следующую стратегию:

- Попытка выиграть (поставить свой нолик на выигрышную линию).
- Блокировка игрока (помешать игроку собрать три крестика в линию).
- Случайный ход (если нет непосредственной угрозы или выигрыша, выбрать случайную свободную клетку).

Сохранение и загрузка таблицы рекордов. Программа должна автоматически сохранять статистику игрока (количество побед, поражений, ничьих) при завершении работы и загружать её при старте. Также должен быть предусмотрен ручной экспорт рекордов в файл.

Просмотр и очистка статистики. Пользователь должен иметь возможность в любой момент посмотреть текущую таблицу рекордов или очистить её через меню программы.

Многомодульность. Программа должна состоять из нескольких логических модулей (main.c для GUI и логики, random_utils.c для функций генерации случайных чисел с использованием ассемблера).

Интерфейс программы должен быть выполнен в виде стандартного окна Windows с системой меню для управления игрой, сохранения данных и вызова справки.

Выбор решения

В качестве среды разработки была выбрана Microsoft Visual Studio 2026, которая предоставляет мощные средства для отладки кода на языках C/C++ и ассемблере. Для реализации графического интерфейса был выбран Win32 API (Application Programming Interface). Это позволило создать нативное приложение для Windows без использования сторонних фреймворков, что дало полный контроль над отрисовкой окна, обработкой сообщений от мыши и созданием системного меню.

При запуске программы сначала отображается заставка (splash screen), содержащая название, версию и информацию о разработчике. Далее открывается главное окно. Пользователь взаимодействует с программой с помощью мыши, кликая по клеткам игрового поля, или выбирая пункты меню:

1) Игра

- 1.1 Новая игра. Запускает новый матч. Происходит очистка игрового поля, сброс флага окончания игры, ход переходит к игроку (крестики).
- 1.2 Сохранить результат. Программа сохраняет текущую статистику (победы, поражения, ничьи) в бинарный файл records.dat. Если файл уже существует, он перезаписывается.
- 1.3 Выход. Завершает работу программы с автоматическим сохранением статистики.

2) Рекорды

- 2.1 Таблица рекордов. Отображает всплывающее окно со статистикой:
 - 2.1.1 Количество побед игрока.
 - 2.1.2 Количество поражений.
 - 2.1.3 Количество ничьих.
 - 2.1.4 Общее количество сыгранных партий
 - 2.1.5 Процент побед (рассчитывается как отношение побед к общему числу игр). Процент побед (рассчитывается как отношение побед к общему числу игр). Процент побед (рассчитывается как отношение побед к общему числу игр).
- 2.2 Очистить рекорды. Программа сбрасывает все сохранённые значения статистики (победы = 0, поражения = 0, ничьи = 0) и обновляет отображение в заголовке окна.

3) Справка

- 3.1.1 О программе. Выводит диалоговое окно с информацией о версии (2.0), годе создания, авторе и перечнем ключевых особенностей программы: многомодульность, ассемблерная вставка, таблица рекордов, графический интерфейс.

Навигация в меню осуществляется с помощью мыши. Для выбора пункта достаточно однократного клика. Для хода по полю игрок кликает левой кнопкой мыши по свободной клетке.

4) Логика работы компьютера

После каждого хода игрока программа проверяет состояние игры. Если игра не окончена, ход переходит к компьютеру. Алгоритм выбора хода компьютером реализован в следующем порядке приоритета:

- 4.1 Выигрыш. Программа сканирует все пустые клетки. Если установка нолика (компьютера) в какую-либо клетку приводит к победе, компьютер делает этот ход немедленно.
- 4.2 Блокировка игрока. Если выигрышный ход не найден, программа проверяет, может ли игрок на следующем ходу победить (собрать три крестика в линию). Если такая угроза есть, компьютер ставит свой нолик в блокирующую клетку.
- 4.3 Случайный ход. Если ни выигрыш, ни блокировка не требуются, компьютер выбирает случайную свободную клетку. Для генерации случайного числа используется ассемблерная вставка с инструкцией RDTSC, которая считывает счётчик тактов процессора, обеспечивая высокое качество случайной последовательности

5) Сохранение и загрузка данных

- 5.1 Загрузка при старте. При запуске программа проверяет наличие файла records.dat в текущей директории. Если файл существует, данные о победах, поражениях и ничьих загружаются. Если файл отсутствует, статистика инициализируется нулями.
 - 5.2 Автосохранение. При завершении программы (через пункт меню «Выход» или закрытие окна) статистика автоматически сохраняется в файл records.dat.
- 6) Ручное сохранение. Пользователь может в любой момент сохранить текущую статистику через пункт меню «Сохранить результат».

Навигация в меню осуществляется с помощью клавиатуры, вводом цифр от 0 до 6.

Описание разработки программы

Разработанная программа состоит из нескольких модулей, которые реализовывались в следующем порядке:

- 1) **main.c** – основной файл программы. Содержит точку входа
- 2) **WinMain**, оконную процедуру **WndProc**, обработку сообщений от мыши и меню, а также основную игровую логику (проверка победителя, ничьей, ход компьютера).
- 3) **random_utils.c** – отдельный модуль с ассемблерной вставкой. Содержит функцию **asm_random(int max)**, которая генерирует случайное число в заданном диапазоне с использованием инструкции процессора **RDTSC**.
- 4) **Заголовочные файлы** – в программе используются стандартные заголовки Windows (**windows.h**, **commctrl.h**, **stdio.h**, **stdlib.h**, **string.h**), а также подключается библиотека **comctl32.lib** для работы с элементами управления.
- 5) Функция **ShowSplashScreen** – реализует отображение заставки-приветствия при запуске программы. Использует создание всплывающего окна, вывод текста с помощью GDI (Graphics Device Interface) и таймер для автоматического закрытия через 800 миллисекунд.
- 6) Функции работы статистикой – **SaveStatsToFile**, **LoadStatsFromFile**, **AddWinToStats**, **AddLossToStats**, **AddDrawToStats**. Обеспечивают сохранение и загрузку таблицы рекордов в бинарный файл **records.dat**.
- 7) Функции отображения рекордов. **ShowRecordsWindow**, **ClearRecords**. Реализуют вывод статистики в виде диалогового окна и очистку всех сохранённых данных.
- 8) Функции игровой логики **InitGame**, **CheckWinner**, **IsDraw**, **ComputerMove**.
Отвечают за отрисовку игрового поля (линии сетки, символы X и O) и обновление заголовка окна с отображением текущего статуса игры и статистики.
- 9) Функции графического вывода – **DrawBoard**, **UpdateTitle**
Вывод записей в определенном формате.

По порядку реализации модулей можно понять, что использовался **нисходящий** подход разработки программы. Этот метод основан на том, что сначала проектируются основные компоненты программы (главное окно, меню, обработка сообщений), а затем уже дорабатываются её мелкие детали (логика ходов, отрисовка, сохранение статистики).

Сначала была создана базовая структура Win32-приложения – зарегистрирован класс окна, создано главное окно с меню, реализована обработка основных сообщений (WM_CREATE, WM_SIZE, WM_PAINT, WM_LBUTTONDOWN, WM_COMMAND, WM_DESTROY).

Затем была реализована заставка (splash screen) для приветствия пользователя.

Далее разработана структура данных для хранения состояния игры (GameState) и базовые функции игровой логики – инициализация, проверка победителя, проверка ничьей.

После этого добавлена отрисовка игрового поля с использованием графических примитивов GDI (линии и текст).

На следующем этапе реализована обработка кликов мыши – определение координат клетки, проверка её доступности и установка символа игрока.

Затем разработан алгоритм хода компьютера, состоящий из трёх уровней приоритета: выигрыш → блокировка → случайный ход.

После завершения основной логики добавлены функции работы с файлами для сохранения и загрузки таблицы рекордов.

На заключительном этапе реализована ассемблерная вставка в отдельном модуле random_utils.c для генерации случайных чисел. Использование инструкции RDTSC (Read Time-Stamp Counter) позволяет получать более качественные случайные значения по сравнению со стандартным rand(), так как счётчик тактов процессора постоянно изменяется независимо от программы.

Такой подход позволил упростить тестирование, так как каждый модуль отлаживался сразу после его реализации, что помогло в создании более структурированной и надёжной программы.

Отладка и тестирование

В качестве среды разработки была выбрана программа Microsoft Visual Studio 2026. Данная среда обладает всеми средствами, необходимыми при разработке и отладке Windows-приложений на языках C/C++ и ассемблере.

Тестирование проводилось во время разработки и также после завершения разработки. В ходе нее было выявлено огромное количество проблем, связанных с работой с файлами, работой с памятью, работой с полями структуры.

Описание программы

Приложение TicTacToe.exe является основным модулем программы. При запуске программы происходит вывод заставки-приветствия, после чего открывается главное окно с игровым полем. Взаимодействие с программой осуществляется с помощью мыши (клики по клеткам поля и по пунктам меню). Выход из программы осуществляется после выбора пользователем пункта меню «Выход» или закрытия окна. Описание состояний программы выполнено в таблице ниже.

Таблица 1. – Главное окно и системное меню

Действие пользователя	Действие программы
Клик левой кнопкой мыши по свободной клетке игрового поля	В клетке отображается символ «X» (ход игрока). Программа проверяет наличие победителя или ничьей. Если игра не окончена, ход переходит к компьютеру. Компьютер делает ход (символ «О»), после чего снова проверяется состояние игры.
Клик левой кнопкой мыши по занятой клетке	Программа игнорирует действие, ход не меняется.

Выбран пункт меню «Игра» → «Новая игра»	Программа очищает игровое поле, сбрасывает флаги окончания игры, устанавливает очередность хода на игрока (крестики). Статистика сохраняется, заголовок окна обновляется.
Выбран пункт меню «Игра» → «Сохранить результат»	Программа сохраняет текущую статистику (победы, поражения, ничьи) в файл records.dat. На экране появляется сообщение «Результаты сохранены в файл records.dat».
Выбран пункт меню «Игра» → «Выход»	Программа автоматически сохраняет статистику и завершает работу.
Выбран пункт меню «Рекорды» → «Таблица рекордов»	Открывается диалоговое окно с отображением статистики: количество побед, поражений, ничьих, общее число игр и процент побед (см. Таблицу 2).
Выбран пункт меню «Рекорды» → «Очистить рекорды»	Программа отображает диалог подтверждения. При подтверждении все значения статистики обнуляются, файл records.dat перезаписывается, заголовок окна обновляется.
Выбран пункт меню «Справка» → «О программе»	Открывается диалоговое окно с информацией о версии программы (2.0), годе создания, авторе и перечнем ключевых особенностей.

Нажатая кнопка закрытия окна (крестик)	Программа автоматически сохраняет статистику в файл records.dat и завершает работу.
--	---

Таблица 2.

Элемент окна	Содержимое / Действие
Заголовок окна	«Таблица рекордов»
Строка «Победы:»	Отображает значение stats.wins (количество побед игрока)
Строка «Поражения:»	Отображает значение stats.losses (количество поражений)
Строка «Ничьи:»	Отображает значение stats.draws (количество ничьих)
Строка «Всего игр:»	Отображает сумму побед, поражений и ничьих
Строка «Процент побед:»	Отображает рассчитанное значение: $(wins / total * 100\%)$ или 0, если игры не проводились
Кнопка «ОК»	Закрывает диалоговое окно

Таблица 3. – Диалоговое окно подтверждения очистки рекордов

Действие пользователя	Действие программы
Нажата кнопка «Да»	Программа обнуляет статистику (wins = 0, losses = 0, draws = 0), перезаписывает файл records.dat, обновляет заголовок окна, выводит сообщение «Таблица рекордов очищена»
Нажата кнопка «Нет»	Диалоговое окно закрывается без изменений

Таблица 4 – Диалоговое окно «О программе»

Элемент окна	Содержимое
Заголовок окна	«О программе»
Название программы	«(C) 2026»
Перечень особенностей	<ul style="list-style-type: none"> • Многомодульность (main.c + random_utils.c) • Ассемблерная вставка (RDTSC) • Таблица рекордов • Сохранение статистики • Русский интерфейс • Графика, мышь, меню

Таблица 4 – Заставка (Splash Screen)

Элемент	Содержимое
Текст в верхней части	«КРЕСТИКИ-НОЛИКИ» (крупный шрифт, полужирный)
Текст в средней части	«Версия 2.0»
Текст ниже	«(C) 2026»
Текст в нижней части	«Загрузка...»
Время отображения	800 миллисекунд, после чего заставка автоматически закрывается

Далее на рисунке 1 представлена схема взаимодействия программы.

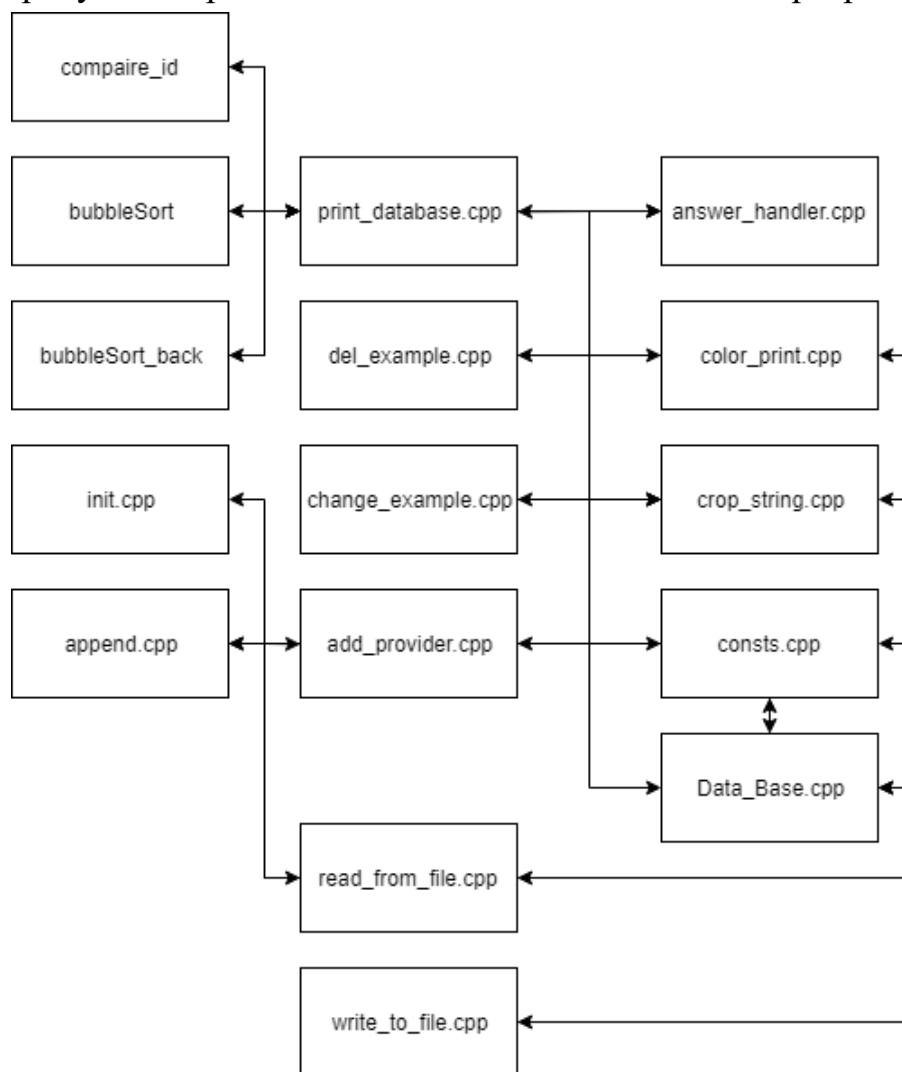


Рисунок - схема взаимодействия программы

Ниже, на рисунке 2 представлена схема данных.

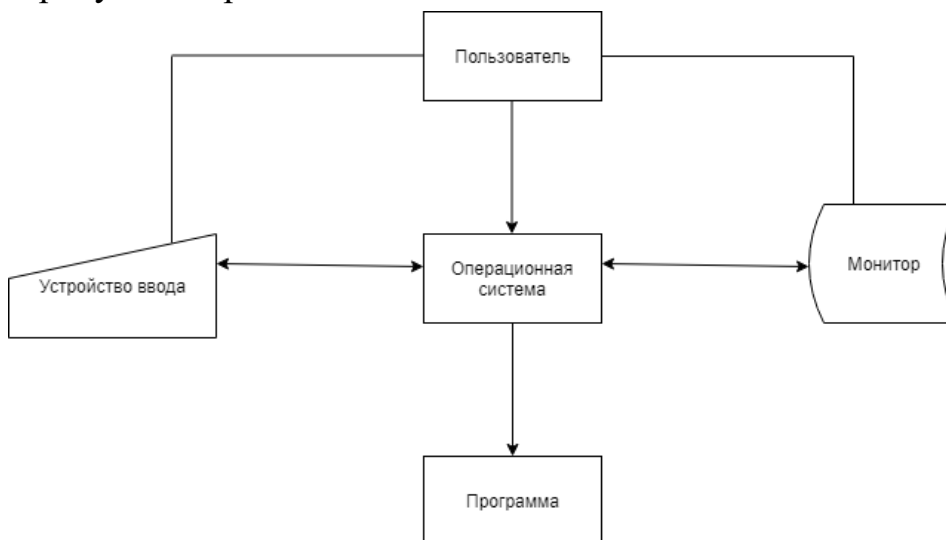


Рисунок - схема данных

Схема программы. Функция Main

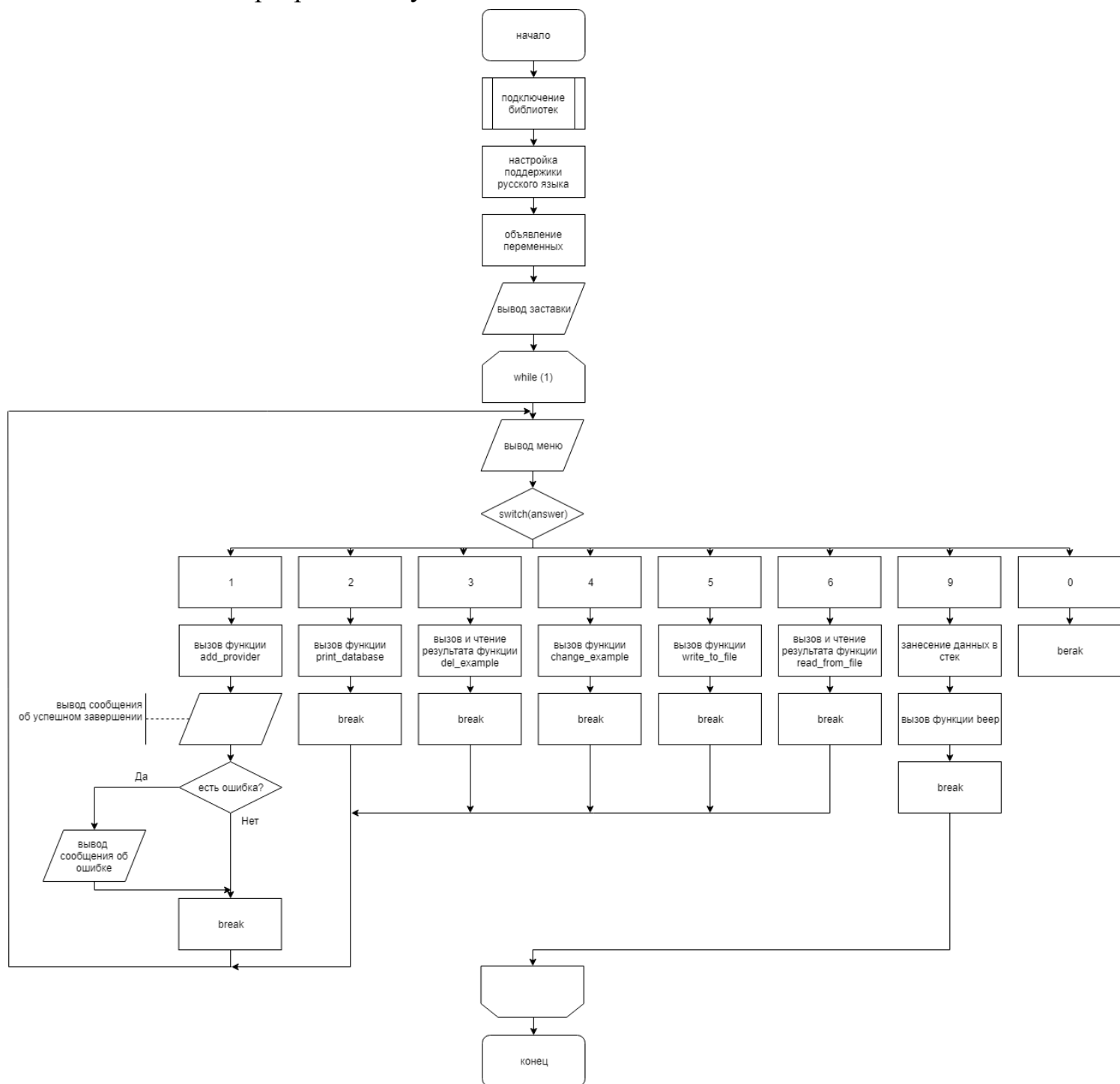
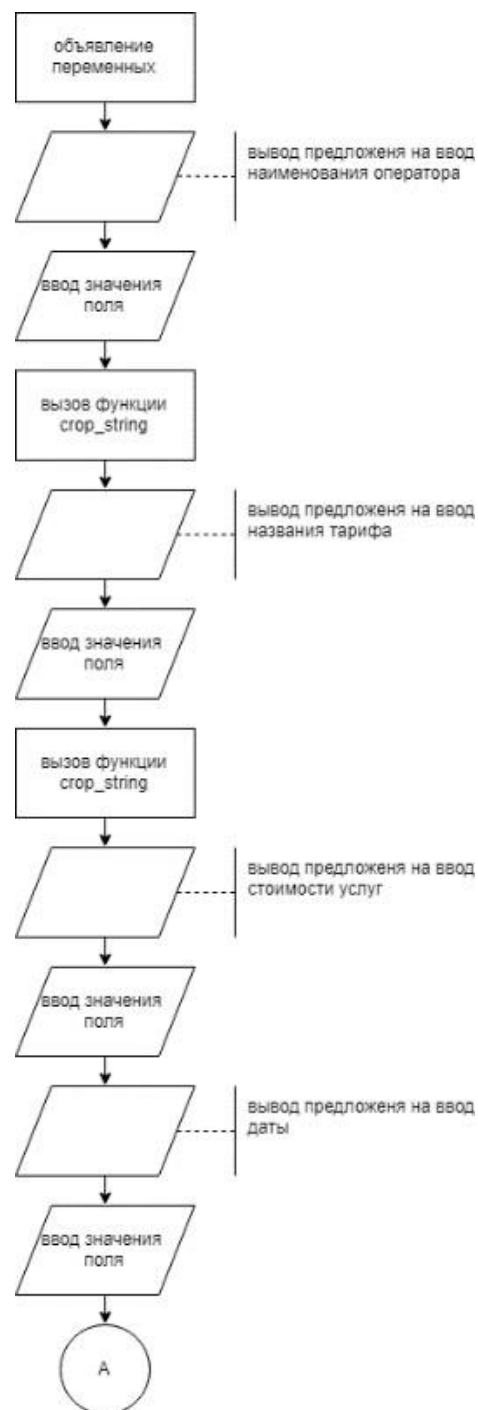
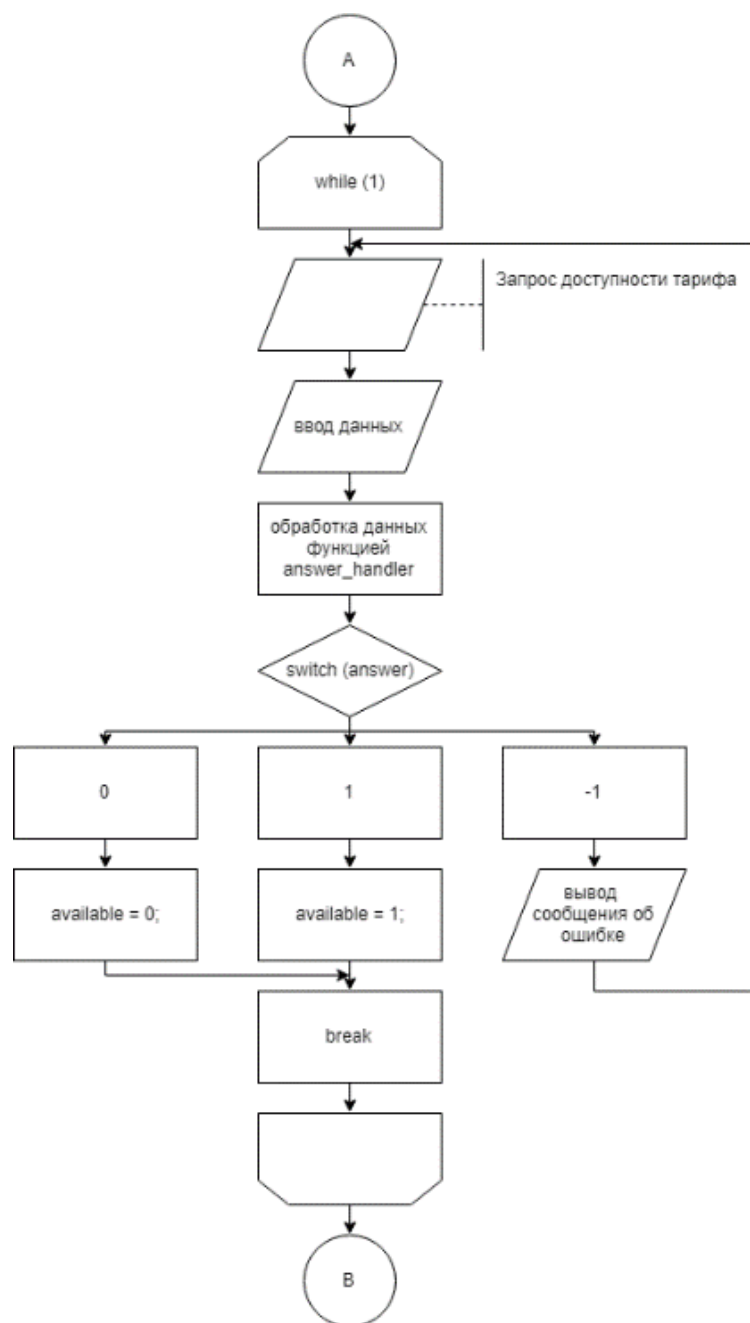


Рисунок - функция main

Схема программы. Функция add_provider





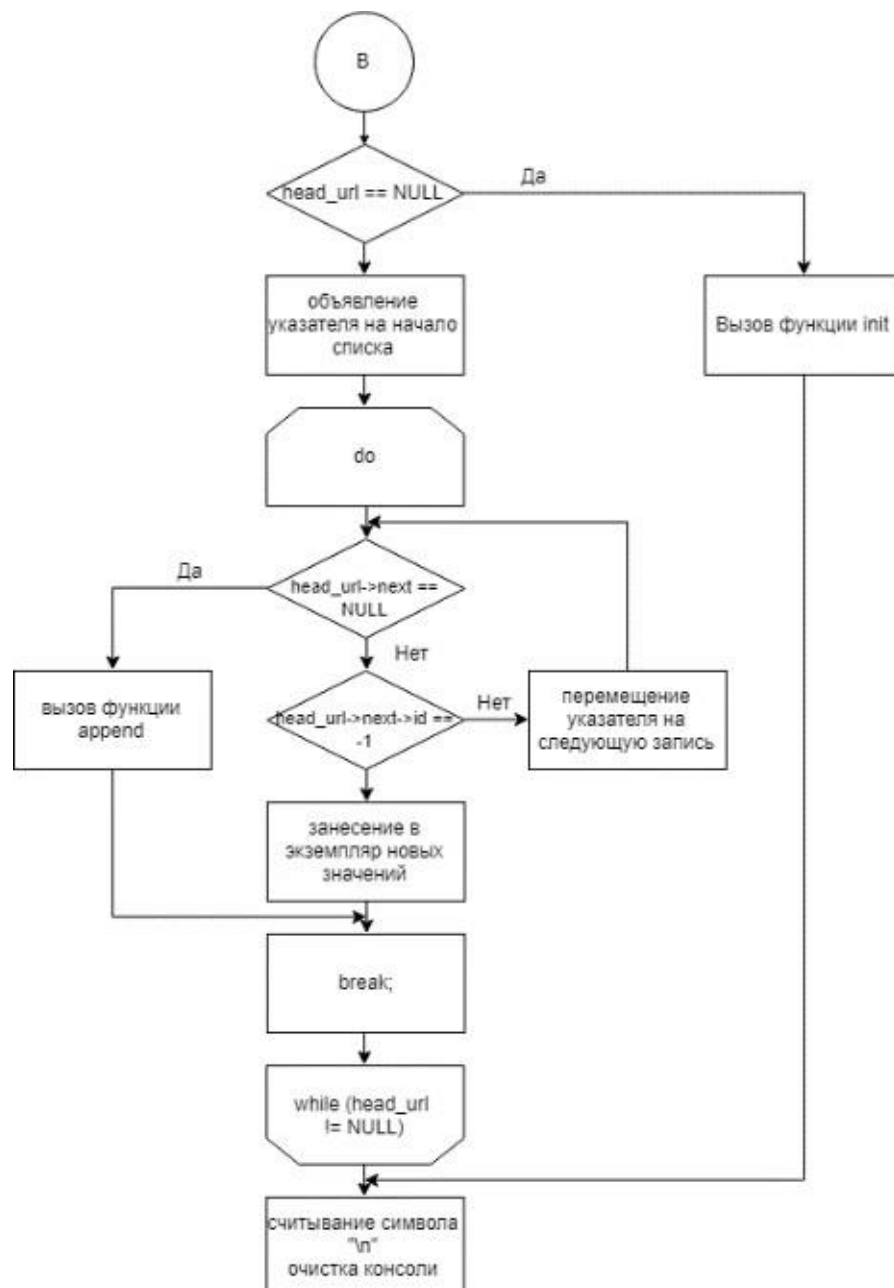
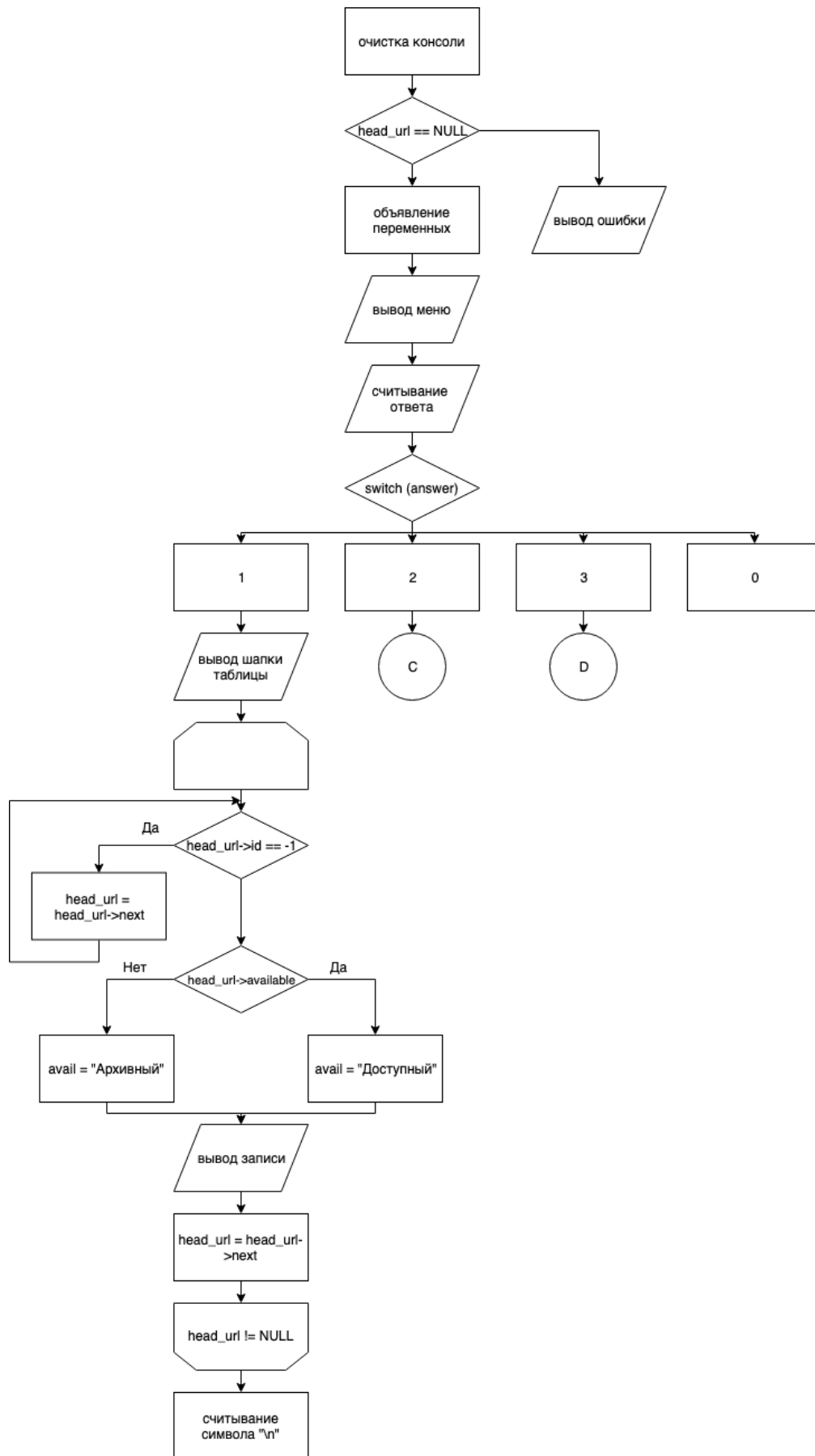
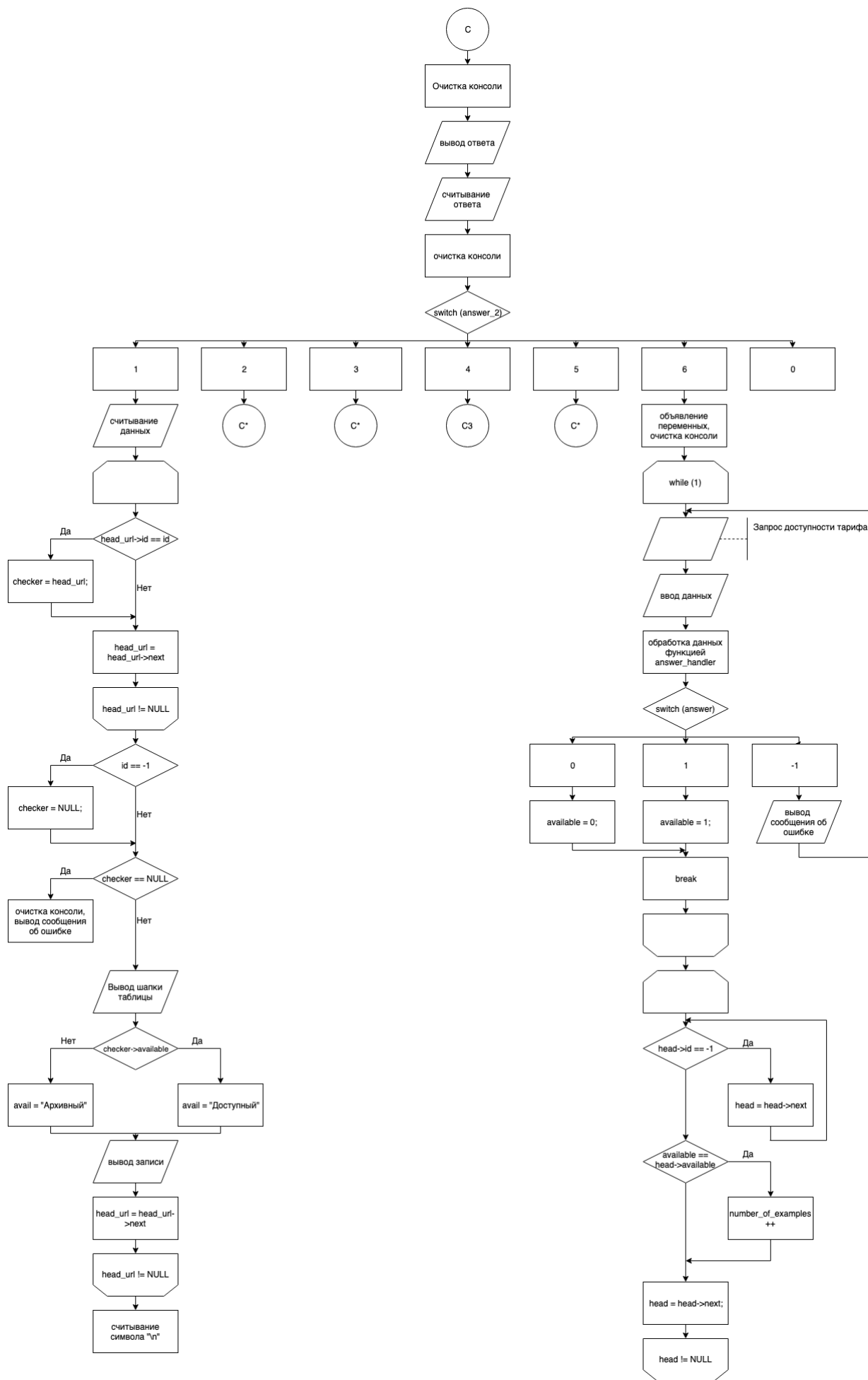
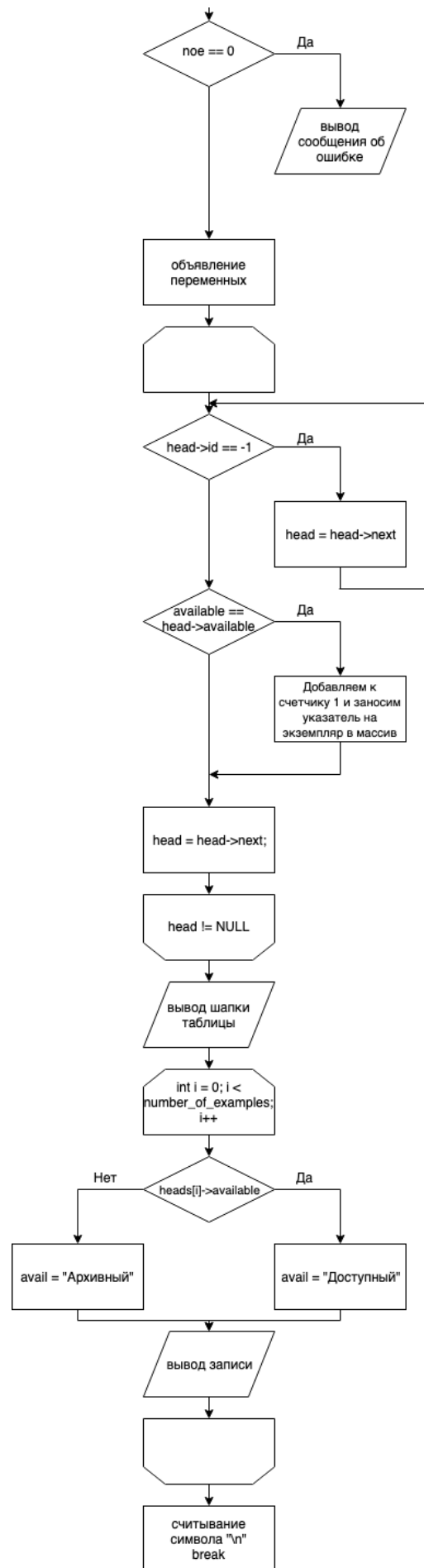


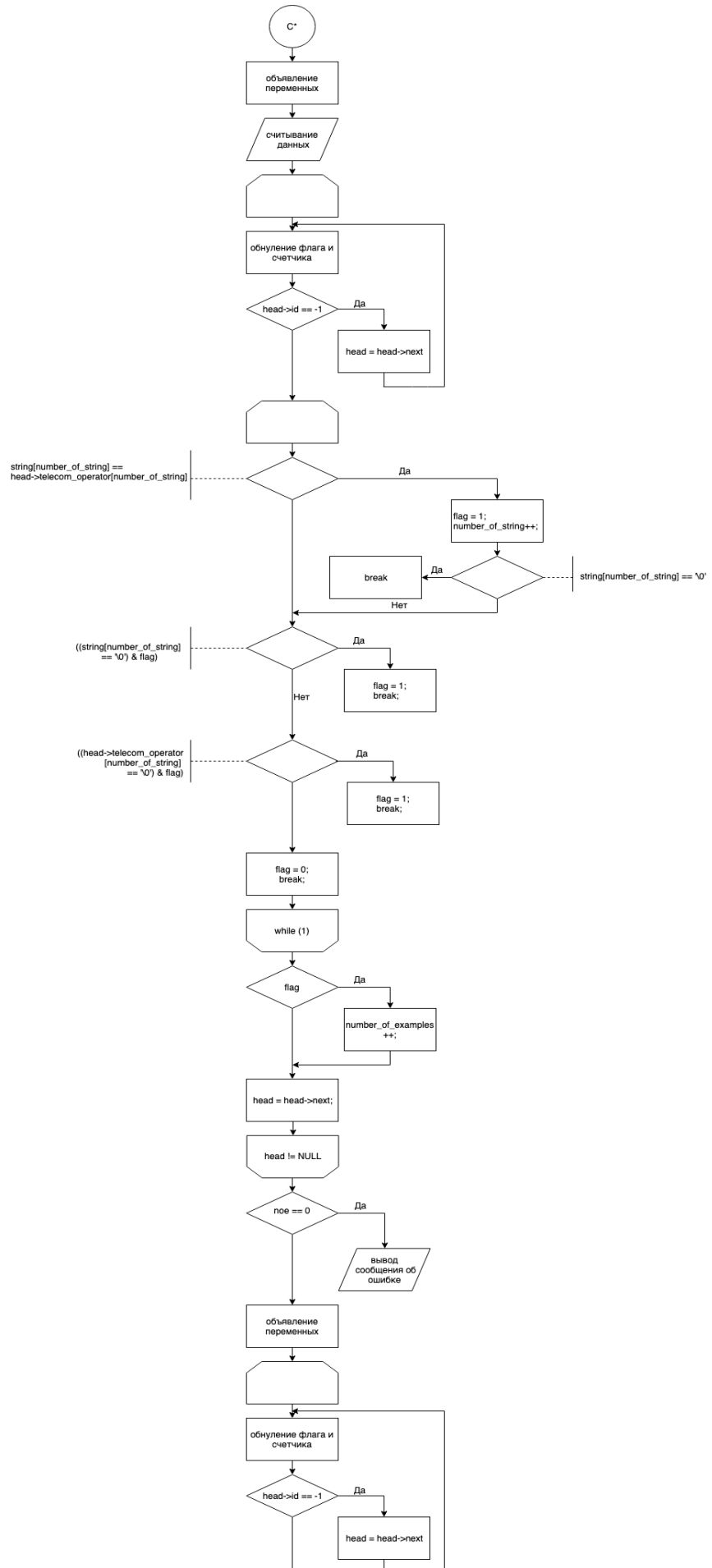
Рисунок - Функция add_provider

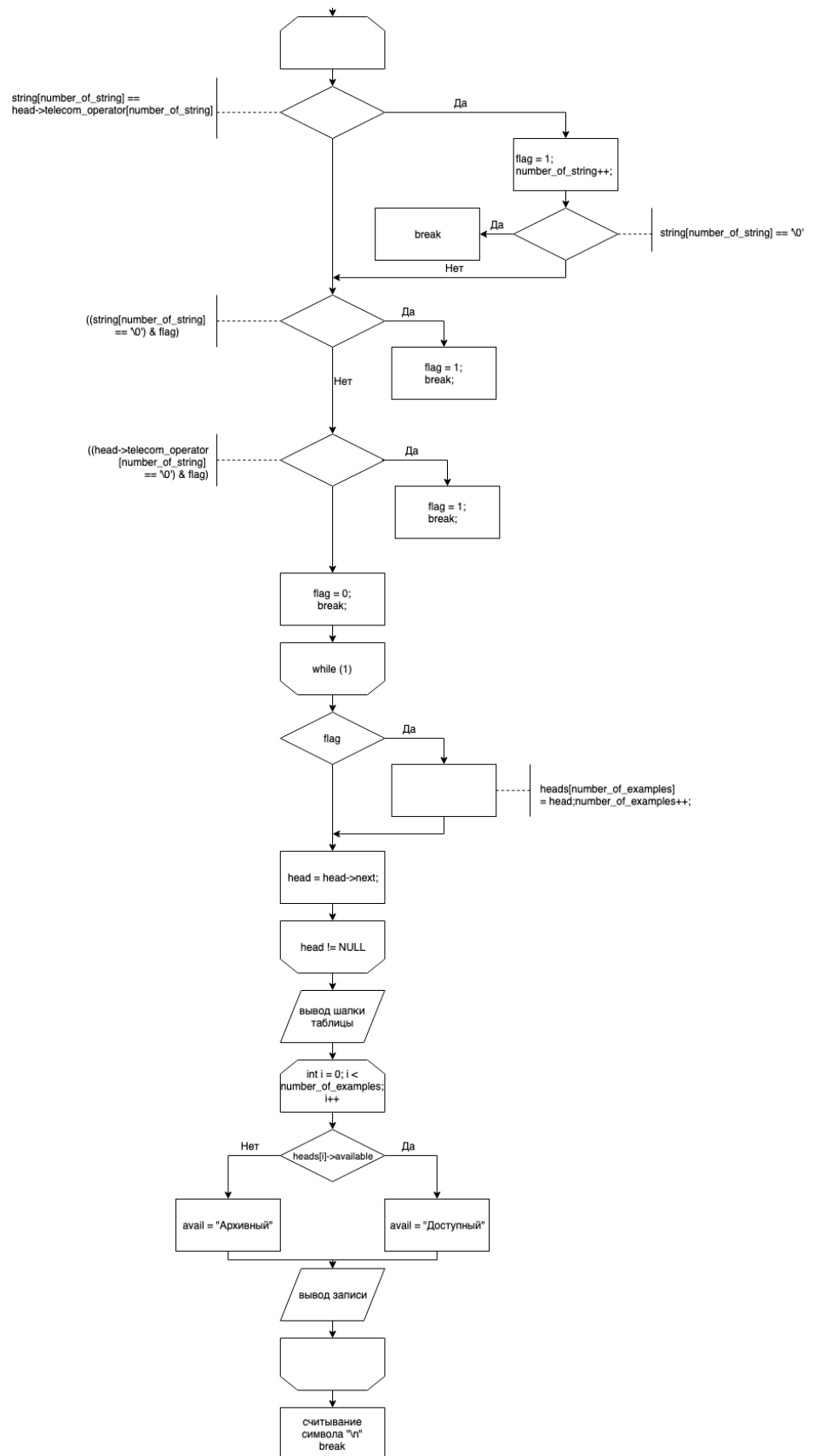
Схема программы. Функция print_database

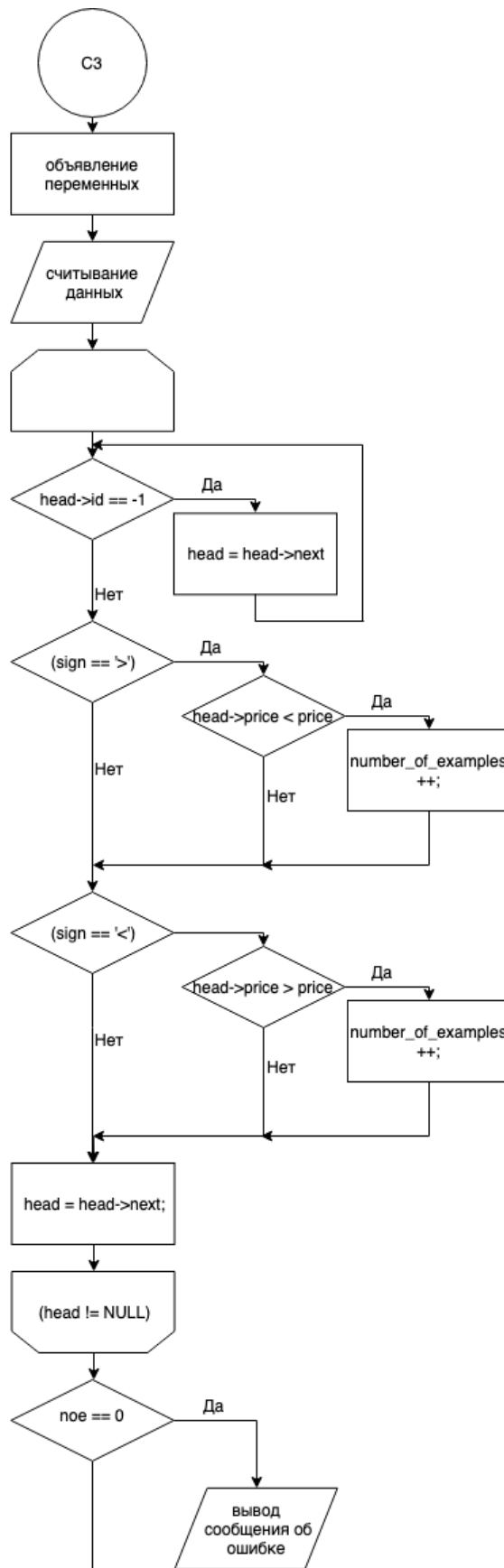


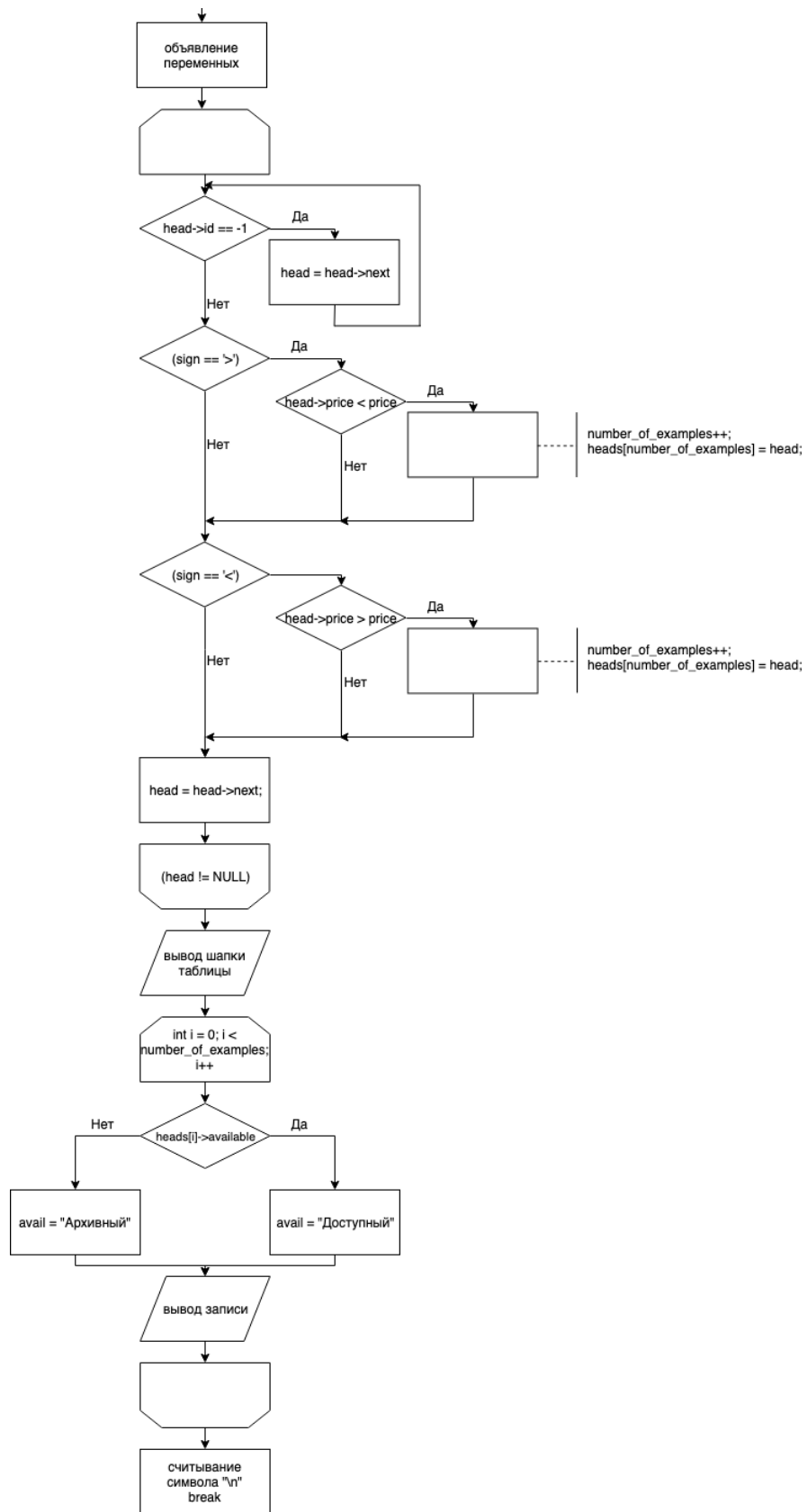












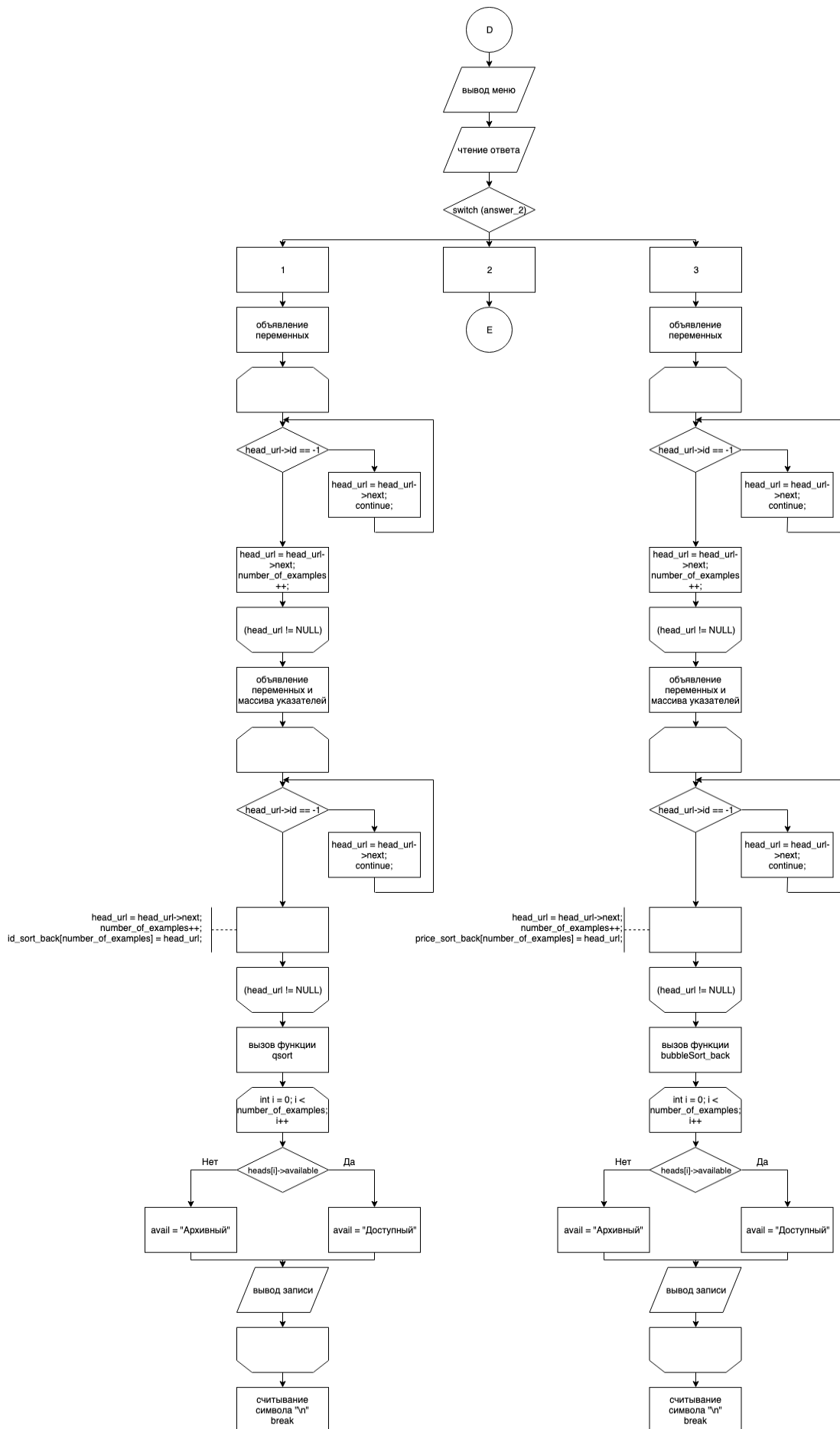


Рисунок - функция print_database

Ниже приведена иерархическая схема программы.

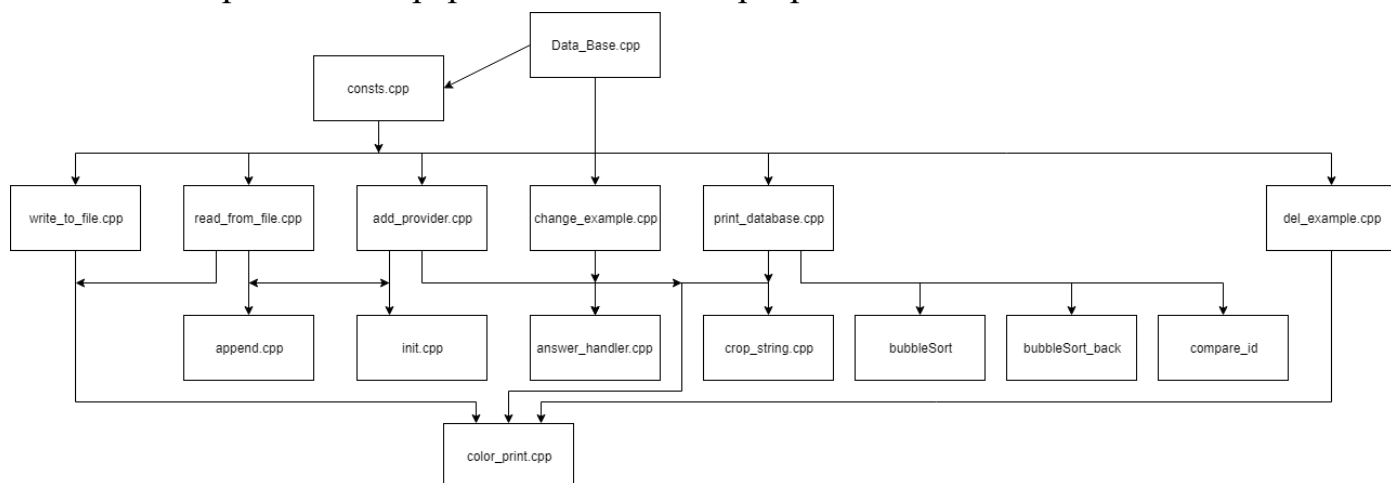


Рисунок - иерархическая схема программы

Описание игровой логики

TicTacToe.exe предназначена для игры в «Крестики-нолики» между человеком и компьютером. Программа имеет интуитивно понятный графический интерфейс и поддерживает следующие функции: новая игра, автоматический ход компьютера, сохранение статистики в файл, просмотр таблицы рекордов, очистка рекордов.

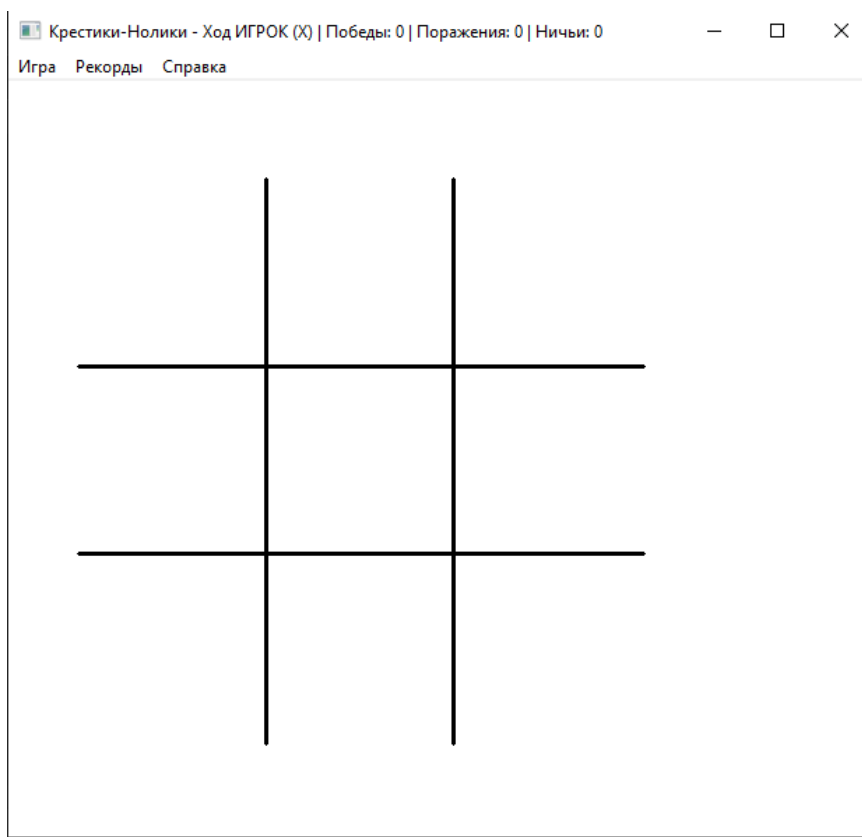


Рисунок 1 – Начальный интерфейс

Для начала новой игры необходимо выбрать пункт меню «Игра» → «Новая игра». Поле очищается, право первого хода получает игрок (крестики). Статистика при этом сохраняется.

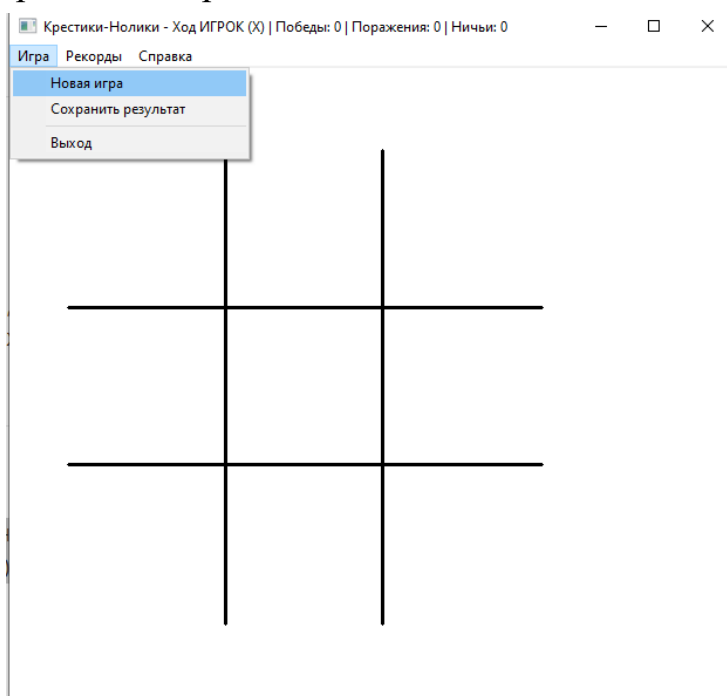


Рисунок 2 – Запуск новой игры

Игрок ходит крестиками («X»). Для совершения хода необходимо навести курсор мыши на свободную клетку игрового поля и нажать левую кнопку мыши. В выбранной клетке появляется красный символ «X».

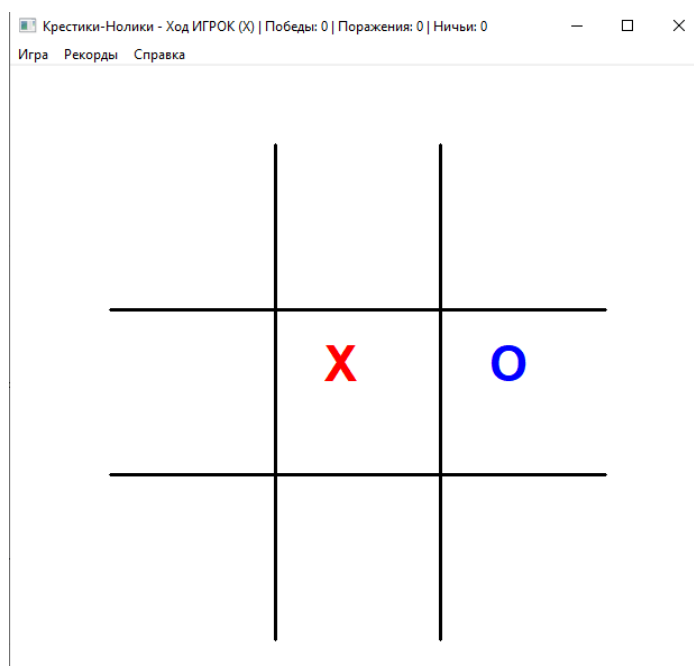


Рисунок 3 – Первый ход

После хода игрока программа проверяет, не привёл ли этот ход к победе или ничьей. Если игра не окончена, ход автоматически переходит к компьютеру.

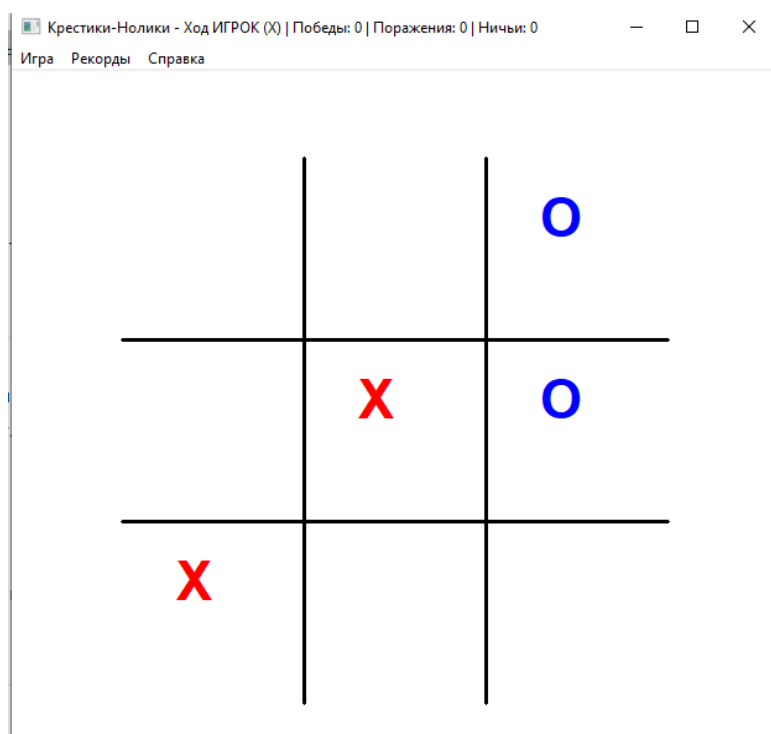


Рисунок 4 – Ход игры

При достижении игроком или компьютером выигрышной комбинации (три символа в ряд по горизонтали, вертикали или диагонали) программа отображает победителя в заголовке окна и блокирует возможность дальнейших ходов..

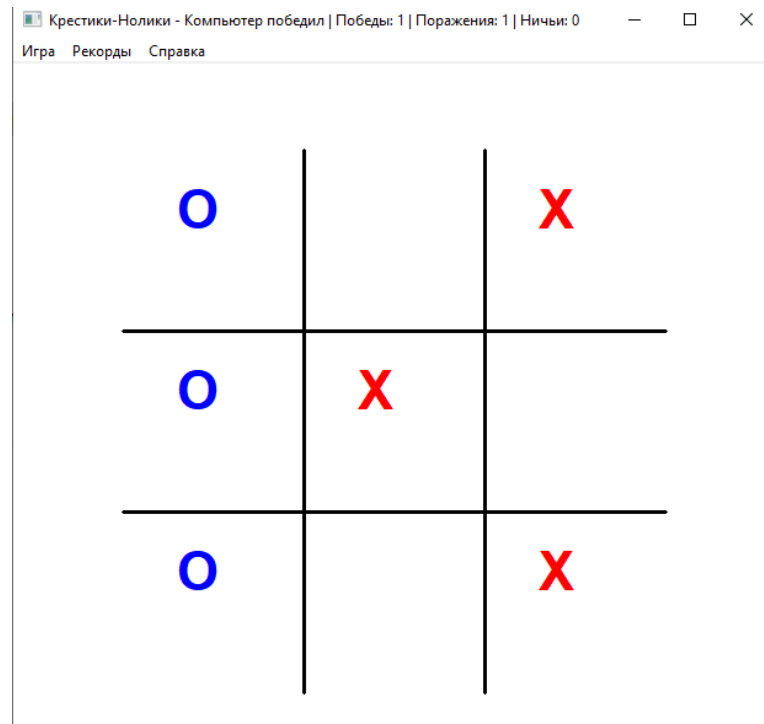


Рисунок 5 – Победа компьютера

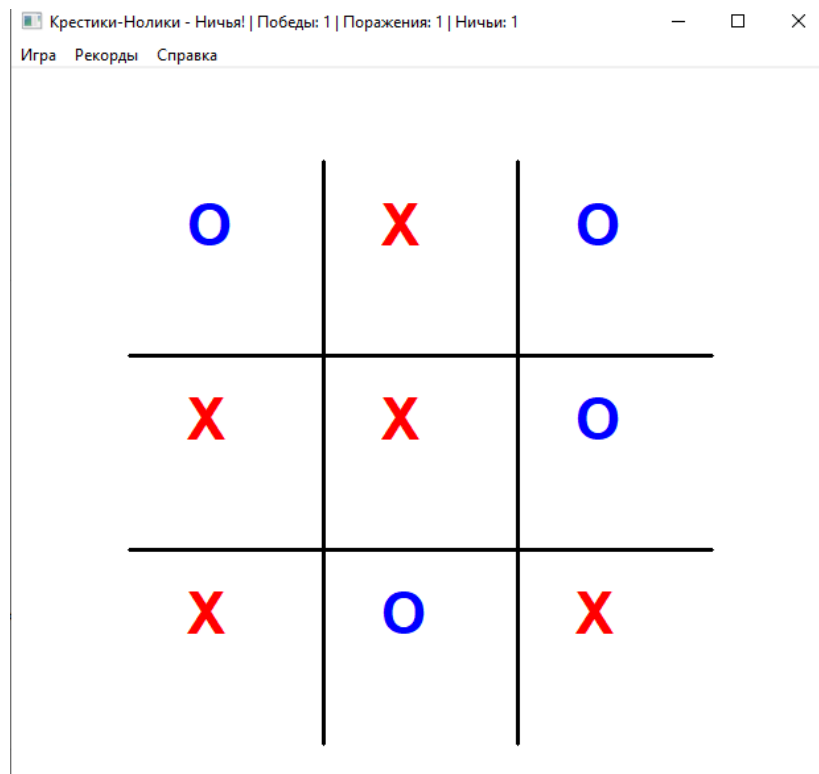


Рисунок 6 - Ничья

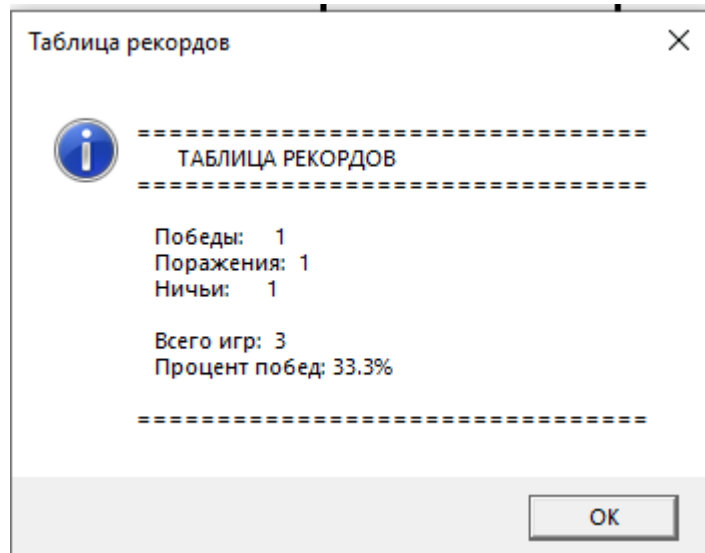


Рисунок 7 – Таблица рекордов

Для ручного сохранения текущей статистики необходимо выбрать пункт меню «Игра» → «Сохранить результат». Программа записывает данные в файл records.dat и выводит подтверждение.

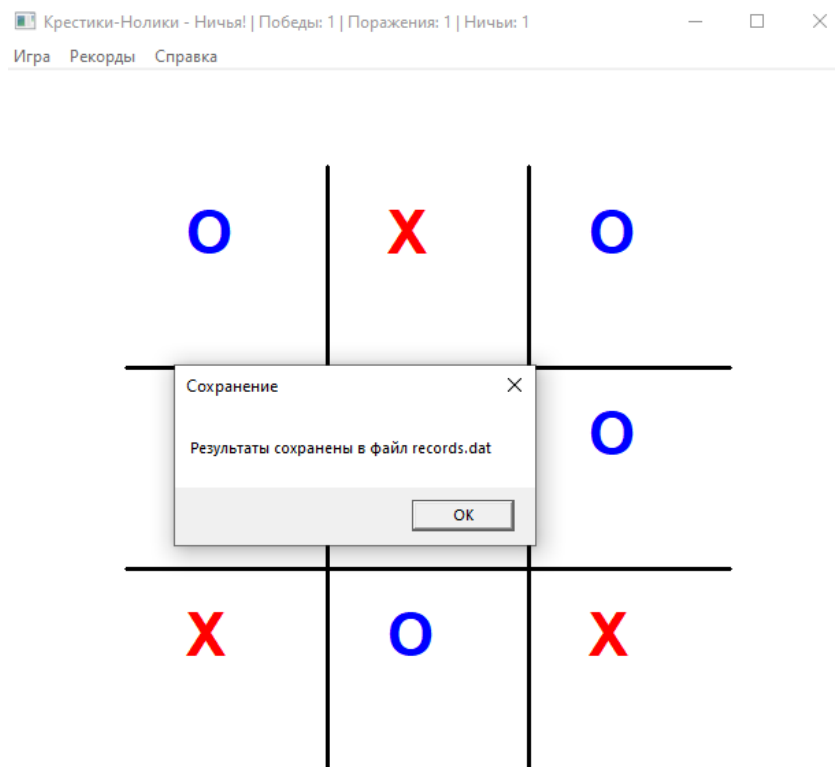


Рисунок 8 - Сохранение результата

Для очистки таблицы рекордов необходимо выбрать пункт меню «Рекорды» → «Очистить рекорды». Программа запросит подтверждение.

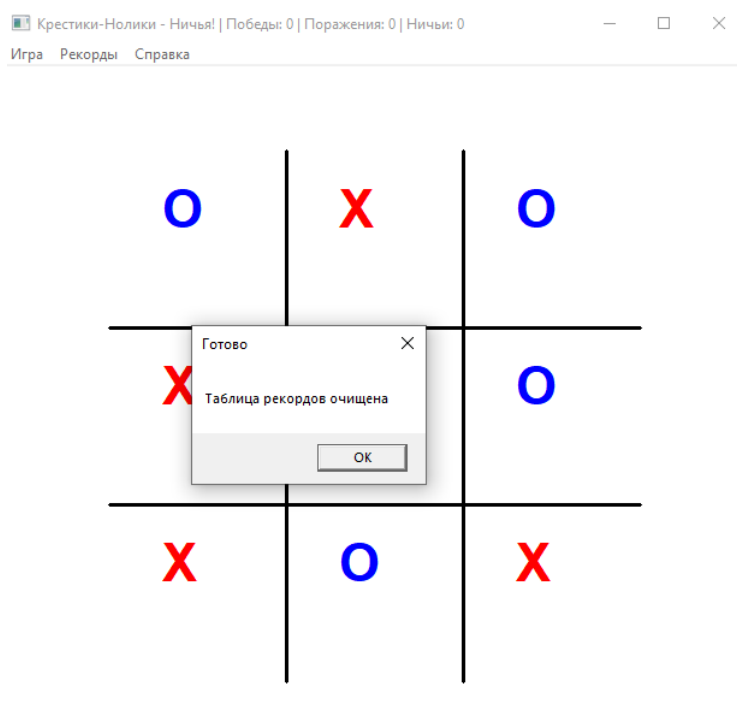


Рисунок 9 – Очистка таблицы рекордов

Для того чтобы сохранить текущую базу данных в файл необходимо выбрать пункт меню «Загрузить БД в файл». Далее ввести имя файла, в

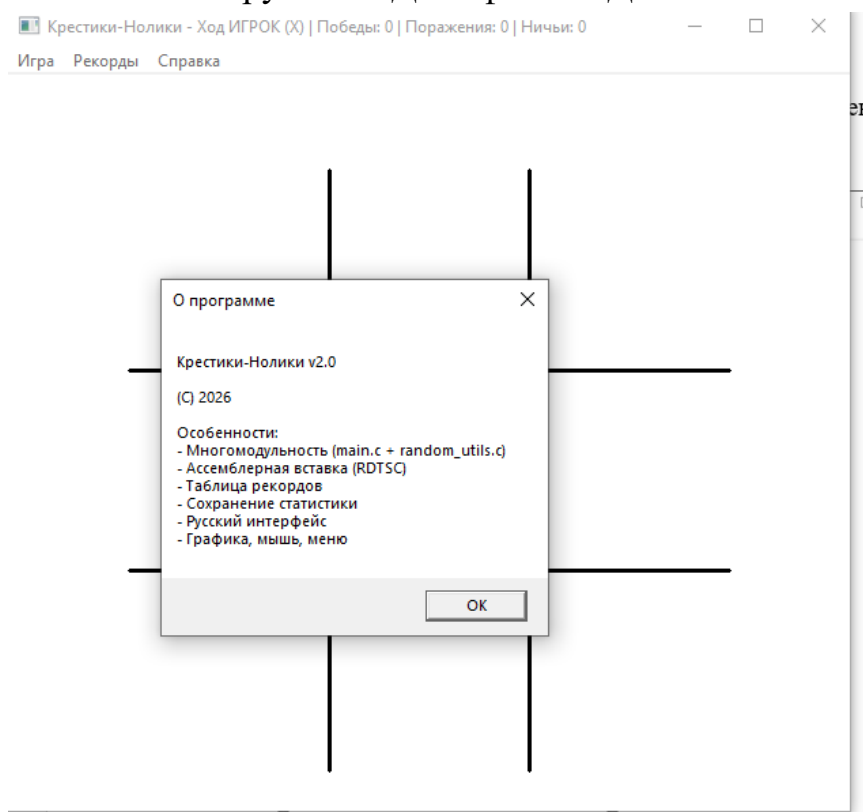


Рисунок 10 – Информация о программе

который будет записана база данных. Если файл уже существует, то программа удалит информацию из файла и запишет в него каталог. В случае если не возникли ошибки на экране появится надпись «Операция выполнена успешно!».

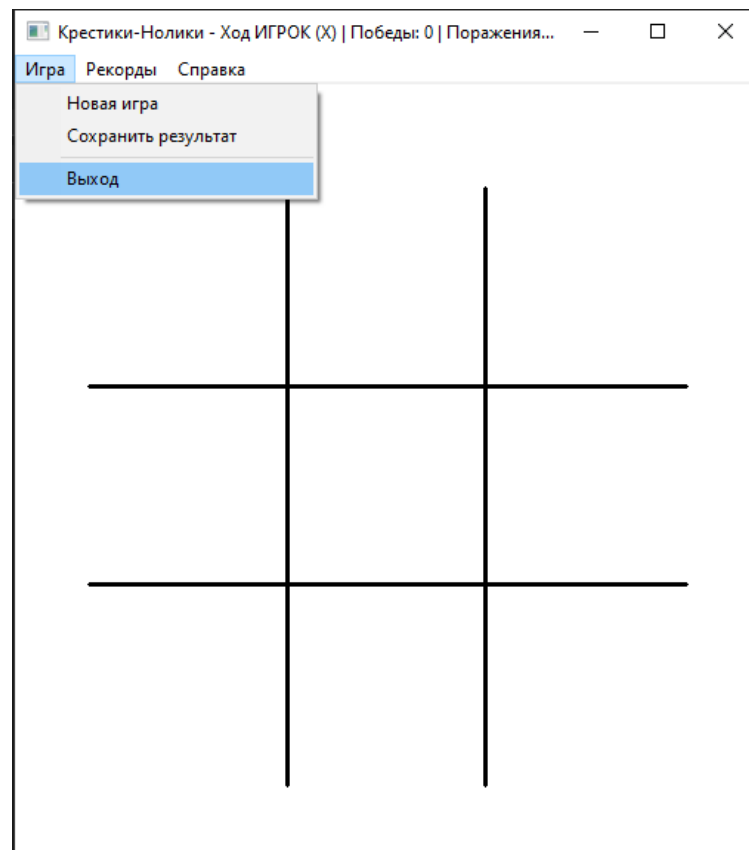


Рисунок 11 - Выход

Заключение

При выполнении данной курсовой работы были получены и закреплены навыки разработки многомодульных приложений для операционной системы Windows с использованием Win32 API. Была освоена работа с графическим интерфейсом пользователя (GUI) – создание главного окна, системного меню, обработка сообщений от мыши, отрисовка графических примитивов (линии, текст) с помощью GDI (Graphics Device Interface).

Также были изучены принципы работы с файловой системой – сериализация и десериализация структур данных в бинарный файл для сохранения и загрузки таблицы рекордов. Были получены навыки отладки и тестирования программ в среде Microsoft Visual Studio 2019 на языках C/C++ с использованием точек останова, трассировки и анализа содержимого переменных.

В рамках курсовой работы была написана программа для игры в «Крестики-нолики» между человеком и компьютером. Она имеет все необходимые функции для комфортной игры: интуитивно понятный графический интерфейс, алгоритм хода компьютера с тремя уровнями приоритета (выигрыш → блокировка → случайный ход), сохранение и просмотр таблицы рекордов, а также возможность очистки статистики.

Список используемых источников

1. Керниган Б., Ритчи Д. Язык программирования C. – М.: Вильямс, 2020. – 304 с.
2. Петзольд Ч. Программирование для Windows 95. – СПб.: BHV – Санкт-Петербург, 2021. – 752 с. (Классическое руководство по Win32 API, актуальное и для современных версий Windows).
3. MSDN (Microsoft Developer Network) – официальная документация Microsoft по Win32 API, C/C++, ассемблеру и Visual Studio. [Электронный ресурс]. – Режим доступа: <https://docs.microsoft.com/ru-ru/windows/win32/>
4. Прата С. Язык программирования C++. Лекции и упражнения. – М.: Вильямс, 2020. – 1248 с.
5. Тюгашев А.А. Языки программирования. Учебное пособие. – Саратов: Вузовское образование, 2022. – 194 с.
6. Аблязов Р. Программирование на ассемблере на платформе x86-64. – М.: ДМК Пресс, 2020. – 396 с.
7. Фленов М.Е. Библия Delphi / Михаил Фленов. – 3-е изд., перераб. и доп. – СПб.: БХВ-Петербург, 2021. – 880 с. (Использовалась для понимания принципов создания игровой логики).
8. Шилдт Г. Полный справочник по C. – 4-е издание. – М.: Вильямс, 2023. – 704 с.

Приложение А

Листинги программы

Файл main.c

```
#include <locale.h>
#define _CRT_SECURE_NO_WARNINGS
#include <windows.h>
#include <commctrl.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#pragma comment(lib,
"comctl32.lib")
#pragma comment(lib, "gdi32.lib")

// ===== КОНСТАНТЫ
=====
#define ID_NEW_GAME      1001
#define ID_SAVE_SCORE   1002
#define ID_SHOW_RECORDS 1003
#define ID_CLEAR_RECORDS 1004
#define ID_EXIT          1005
#define ID_ABOUT         1006

#define PLAYER_X 1
#define PLAYER_O 2

// ===== ГЛОБАЛЬНЫЕ
ПЕРЕМЕННЫЕ =====
HINSTANCE hInst;
HWND hMainWnd;

// Структура игры
typedef struct {
    int board[3][3];
    int turn;
    int gameOver;
    int winner;
} GameState;

GameState currentGame;
```

```

// Статистика игрока
typedef struct {
    int wins;
    int losses;
    int draws;
} PlayerStats;

PlayerStats stats;

// ===== ПРОТОТИП ФУНКЦИИ ИЗ
ДРУГОГО МОДУЛЯ =====
extern int asm_random(int max);

// ===== ПРОТОТИПЫ ФУНКЦИЙ
=====

LRESULT CALLBACK WndProc(HWND,
UINT, WPARAM, LPARAM);
void ShowSplashScreen(void);
void InitGame(void);
int CheckWinner(void);
int IsDraw(void);
void ComputerMove(void);
void DrawBoard(HDC hdc, RECT*
rcBoard);
void SaveStatsToFile(void);
void LoadStatsFromFile(void);
void ShowRecordsWindow(void);
void ClearRecords(void);
void UpdateTitle(void);
void AddWinToStats(void);
void AddLossToStats(void);
void AddDrawToStats(void);

// ===== ТОЧКА ВХОДА
=====

int WINAPI WinMain(HINSTANCE
hInstance, HINSTANCE
hPrevInstance, LPSTR lpCmdLine,
int nCmdShow) {
    hInst = hInstance;

    INITCOMMONCONTROLSEX icex;
    icex.dwSize =
sizeof(INITCOMMONCONTROLSEX);
    icex.dwICC = ICC_BAR_CLASSES;

```

```

    InitCommonControlsEx(&icex);

    ShowSplashScreen();

    LoadStatsFromFile();

    WNDCLASSEX wc = { 0 };
    wc.cbSize =
sizeof(WNDCLASSEX);
    wc.lpfnWndProc = WndProc;
    wc.hInstance = hInstance;
    wc.hCursor = LoadCursor(NULL,
IDC_ARROW);
    wc.hbrBackground =
(HBRUSH) (COLOR_WINDOW + 1);
    wc.lpszClassName =
"TicTacToeClass";
    RegisterClassExA(&wc);

    hMainWnd = CreateWindowExA(0,
"TicTacToeClass", "Крестики-
Нолики",
        WS_OVERLAPPEDWINDOW |
WS_VISIBLE,
        CW_USEDEFAULT,
CW_USEDEFAULT, 520, 580,
        NULL, NULL, hInstance,
NULL);

    ShowWindow(hMainWnd,
nCmdShow);
    UpdateWindow(hMainWnd);

    InitGame();

    MSG msg;
    while (GetMessage(&msg, NULL,
0, 0)) {
        TranslateMessage(&msg);
        DispatchMessage(&msg);
    }

    return msg.wParam;
}

// ===== ЗАДАЧА =====

```

```

void ShowSplashScreen(void) {
    HWND hSplash =
    CreateWindowExA(0, "STATIC",
    NULL, WS_POPUP | WS_BORDER,
        CW_USEDEFAULT,
    CW_USEDEFAULT, 400, 250,
        NULL, NULL, hInst, NULL);

    HDC hdc = GetDC(hSplash);

    HFONT hFont = CreateFontA(28,
    0, 0, 0, FW_BOLD, FALSE, FALSE,
    FALSE,
        DEFAULT_CHARSET,
    OUT_DEFAULT_PRECIS,
        CLIP_DEFAULT_PRECIS,
    DEFAULT_QUALITY,
        DEFAULT_PITCH, "Arial");
    HFONT hOldFont =
    (HFONT)SelectObject(hdc, hFont);

    SetBkMode(hdc, TRANSPARENT);

    TextOutA(hdc, 60, 60,
    "КРЕСТИКИ-ХОЛИКИ", 17);

    SelectObject(hdc, hOldFont);
    DeleteObject(hFont);

    hFont = CreateFontA(18, 0, 0,
    0, FW_NORMAL, FALSE, FALSE,
    FALSE,
        DEFAULT_CHARSET,
    OUT_DEFAULT_PRECIS,
        CLIP_DEFAULT_PRECIS,
    DEFAULT_QUALITY,
        DEFAULT_PITCH, "Arial");
    hOldFont =
    (HFONT)SelectObject(hdc, hFont);

    TextOutA(hdc, 140, 120,
    "Версия 2.0", 10);
    TextOutA(hdc, 150, 155, "(С)
    2026", 8);
    TextOutA(hdc, 100, 190,
    "Загрузка...", 11);

```

```

        SelectObject(hdc, hOldFont);
        DeleteObject(hFont);

        ReleaseDC(hSplash, hdc);
        ShowWindow(hSplash, SW_SHOW);

        Sleep(800);
        DestroyWindow(hSplash);
    }

    // ===== PABOTA CO
    СТАТИСТИКОЙ =====
    void SaveStatsToFile(void) {
        FILE* f =
        fopen("records.dat", "wb");
        if (f) {
            fwrite(&stats,
            sizeof(PlayerStats), 1, f);
            fclose(f);
        }
    }

    void LoadStatsFromFile(void) {
        FILE* f =
        fopen("records.dat", "rb");
        if (f) {
            fread(&stats,
            sizeof(PlayerStats), 1, f);
            fclose(f);
        }
        else {
            stats.wins = 0;
            stats.losses = 0;
            stats.draws = 0;
        }
    }

    void AddWinToStats(void) {
        stats.wins++;
        SaveStatsToFile();
    }

    void AddLossToStats(void) {
        stats.losses++;
    }

```

```

        SaveStatsToFile();
    }

void AddDrawToStats(void) {
    stats.draws++;
    SaveStatsToFile();
}

// ===== ТАБЛИЦА РЕКОРДОВ
=====

void ShowRecordsWindow(void) {
    char message[2048];
    int total = stats.wins +
stats.losses + stats.draws;
    float winRate = (total > 0) ?
(stats.wins * 100.0f / total) :
0;

    sprintf(message,

"=====
\n"
        "          ТАБЛИЦА РЕКОРДОВ
\n"

"=====
\n"
        "\n"
        "   Победы:      %d\n"
        "   Поражения:   %d\n"
        "   Ничьи:       %d\n"
        "\n"
        "   Всего игр:   %d\n"
        "   Процент побед:
%.1f%%\n"
        "\n"

"=====
\n",

        stats.wins, stats.losses,
stats.draws, total, winRate);

    MessageBoxA(hMainWnd,
message, "Таблица рекордов",
MB_OK | MB_ICONINFORMATION);
}

```

```

void ClearRecords(void) {
    if (MessageBoxA(hMainWnd,
        "Вы уверены, что хотите
очистить таблицу рекордов?\n\nВсе
данные будут потеряны!",
        "Подтверждение", MB_YESNO
| MB_ICONQUESTION) == IDYES) {
        stats.wins = 0;
        stats.losses = 0;
        stats.draws = 0;
        SaveStatsToFile();
        UpdateTitle();
        MessageBoxA(hMainWnd,
"Таблица рекордов очищена",
"Готово", MB_OK);
    }
}

// ===== ЛОГИКА ИГРЫ
=====

void InitGame(void) {
    for (int i = 0; i < 3; i++)
        for (int j = 0; j < 3;
j++)

currentGame.board[i][j] = 0;
    currentGame.turn = 0;
    currentGame.gameOver = 0;
    currentGame.winner = 0;
    UpdateTitle();
    InvalidateRect(hMainWnd,
NULL, TRUE);
}

int CheckWinner(void) {
    for (int i = 0; i < 3; i++) {
        if
(currentGame.board[i][0] &&
currentGame.board[i][0] ==
currentGame.board[i][1] &&

currentGame.board[i][1] ==
currentGame.board[i][2])
            return
currentGame.board[i][0];
    }
}

```

```

        if
        (currentGame.board[0][i] &&
currentGame.board[0][i] ==
currentGame.board[1][i] &&

currentGame.board[1][i] ==
currentGame.board[2][i])
            return
currentGame.board[0][i];
    }

    if (currentGame.board[0][0]
&& currentGame.board[0][0] ==
currentGame.board[1][1] &&
        currentGame.board[1][1]
== currentGame.board[2][2])
        return
currentGame.board[0][0];
    if (currentGame.board[0][2]
&& currentGame.board[0][2] ==
currentGame.board[1][1] &&
        currentGame.board[1][1]
== currentGame.board[2][0])
        return
currentGame.board[0][2];
    return 0;
}

int IsDraw(void) {
    for (int i = 0; i < 3; i++)
        for (int j = 0; j < 3;
j++)
            if
            (currentGame.board[i][j] == 0)
return 0;
        return 1;
}

void ComputerMove(void) {
    if (currentGame.gameOver ||
currentGame.turn != 1) return;

    // Попытка выиграть
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3;
j++) {
            if
            (currentGame.board[i][j] == 0) {

```

```

currentGame.board[i][j] =
PLAYER_O;

        if (CheckWinner()
== PLAYER_O) {

currentGame.turn = 0;

currentGame.gameOver = 1;

currentGame.winner = PLAYER_O;

AddLossToStats();

UpdateTitle();

InvalidateRect(hMainWnd, NULL,
TRUE);

                return;

        }

currentGame.board[i][j] = 0;
        }
    }

    // Блокировка игрока
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3;
j++) {
            if
(currentGame.board[i][j] == 0) {

currentGame.board[i][j] =
PLAYER_X;

                if (CheckWinner()
== PLAYER_X) {

currentGame.board[i][j] =
PLAYER_O;

currentGame.turn = 0;

UpdateTitle();

InvalidateRect(hMainWnd, NULL,
TRUE);

                    return;

```

```

        }

currentGame.board[i][j] = 0;
    }
}

// Случайный ход через
ассемблер
    int empty[9][2];
    int count = 0;
    for (int i = 0; i < 3; i++)
        for (int j = 0; j < 3;
j++)
            if
(currentGame.board[i][j] == 0) {
                empty[count][0] =
i;
                empty[count][1] =
j;
                count++;
            }

    if (count > 0) {
        int idx =
asm_random(count);

currentGame.board[empty[idx][0]][
empty[idx][1]] = PLAYER_O;

        int winner =
CheckWinner();
        if (winner) {
            currentGame.gameOver
= 1;
            currentGame.winner =
winner;
            AddLossToStats();
        }
        else if (IsDraw()) {
            currentGame.gameOver
= 1;
            AddDrawToStats();
        }
        currentGame.turn = 0;
    }
}

```

```

        UpdateTitle();
        InvalidateRect(hMainWnd,
NULL, TRUE);
    }

// ===== ПИСОБАНИЕ ПОЛЯ
=====

void DrawBoard(HDC hdc, RECT*
rcBoard) {
    int cellW = (rcBoard->right -
rcBoard->left) / 3;
    int cellH = (rcBoard->bottom
- rcBoard->top) / 3;

    HPEN hPen =
CreatePen(PS_SOLID, 3, RGB(0, 0,
0));
    HPEN hOldPen =
(HPEN)SelectObject(hdc, hPen);

    for (int i = 1; i < 3; i++) {
        MoveToEx(hdc, rcBoard-
>left + i * cellW, rcBoard->top,
NULL);
        LineTo(hdc, rcBoard->left
+ i * cellW, rcBoard->bottom);
        MoveToEx(hdc, rcBoard-
>left, rcBoard->top + i * cellH,
NULL);
        LineTo(hdc, rcBoard-
>right, rcBoard->top + i *
cellH);
    }

    HFONT hFont = CreateFontA(52,
0, 0, 0, FW_BOLD, FALSE, FALSE,
FALSE,
    DEFAULT_CHARSET,
OUT_DEFAULT_PRECIS,
    CLIP_DEFAULT_PRECIS,
DEFAULT_QUALITY,
    DEFAULT_PITCH, "Arial");
    HFONT hOldFont =
(HFONT)SelectObject(hdc, hFont);
    SetBkMode(hdc, TRANSPARENT);

```

```

        for (int row = 0; row < 3;
row++) {
            for (int col = 0; col <
3; col++) {
                char symbol[2] = { 0
};
                if
(currentGame.board[row][col] ==
PLAYER_X) {
                    symbol[0] = 'X';
                    SetTextColor(hdc,
RGB(255, 0, 0));
                }
                else if
(currentGame.board[row][col] ==
PLAYER_O) {
                    symbol[0] = 'O';
                    SetTextColor(hdc,
RGB(0, 0, 255));
                }
                if (symbol[0]) {
                    TextOutA(hdc,
rcBoard->left + col * cellW +
cellW / 3 - 5,
rcBoard->top
+ row * cellH + cellH / 6,
symbol, 1);
                }
            }
        }

        SelectObject(hdc, hOldFont);
        DeleteObject(hFont);
        SelectObject(hdc, hOldPen);
        DeleteObject(hPen);
    }

void UpdateTitle(void) {
    char title[512];
    if (currentGame.gameOver) {
        if (currentGame.winner ==
PLAYER_X) {
            sprintf(title,
"Крестики-Нолики - Вы победили! |
Победы: %d | Поражения: %d |
Ничьи: %d",
stats.wins,

```

```

stats.losses, stats.draws);
    }
    else if
(currentGame.winner == PLAYER_O)
{
        sprintf(title,
"Крестики-Нолики - Компьютер
победил | Победы: %d | Поражения:
%d | Ничьи: %d",
            stats.wins,
stats.losses, stats.draws);
    }
    else {
        sprintf(title,
"Крестики-Нолики - Ничья! |
Победы: %d | Поражения: %d |
Ничьи: %d",
            stats.wins,
stats.losses, stats.draws);
    }
}
else {
    sprintf(title, "Крестики-
Нолики - Ход %s | Победы: %d |
Поражения: %d | Ничьи: %d",
        currentGame.turn == 0
? "ИГРОК (X)" : "КОМПЬЮТЕР (O)",
        stats.wins,
stats.losses, stats.draws);
}
SetWindowTextA(hMainWnd,
title);
}

// ===== ОБРАБОТЧИК ОКНА
=====

LRESULT CALLBACK WndProc(HWND
hwnd, UINT msg, WPARAM wParam,
LPARAM lParam) {
    static RECT rcBoard;

    switch (msg) {
    case WM_CREATE: {
        HMENU hMenu =
CreateMenu();

        HMENU hGameMenu =

```

```

CreatePopupMenu();
    AppendMenuA(hGameMenu,
MF_STRING, ID_NEW_GAME, "Новая
игра");

    AppendMenuA(hGameMenu,
MF_STRING, ID_SAVE_SCORE,
"Сохранить результат");

    AppendMenuA(hGameMenu,
MF_SEPARATOR, 0, NULL);

    AppendMenuA(hGameMenu,
MF_STRING, ID_EXIT, "Выход");

    AppendMenuA(hMenu,
MF_POPUP, (UINT_PTR)hGameMenu,
"Игра");


    HMENU hRecordsMenu =
CreatePopupMenu();
    AppendMenuA(hRecordsMenu,
MF_STRING, ID_SHOW_RECORDS,
"Таблица рекордов");

    AppendMenuA(hRecordsMenu,
MF_STRING, ID_CLEAR_RECORDS,
"Очистить рекорды");

    AppendMenuA(hMenu,
MF_POPUP, (UINT_PTR)hRecordsMenu,
"Рекорды");


    HMENU hHelpMenu =
CreatePopupMenu();
    AppendMenuA(hHelpMenu,
MF_STRING, ID_ABOUT, "О
программе");

    AppendMenuA(hMenu,
MF_POPUP, (UINT_PTR)hHelpMenu,
"Справка");


    SetMenu(hwnd, hMenu);
    break;
}

case WM_SIZE: {
    int w = LOWORD(lParam);
    int h = HIWORD(lParam);
    int size = min(w, h) -
100;

    if (size < 200) size =
200;

    rcBoard.left = (w - size)

```

```

/ 2;
    rcBoard.right =
rcBoard.left + size;
    rcBoard.top = 70;
    rcBoard.bottom =
rcBoard.top + size;
    break;
}

case WM_PAINT: {
    PAINTSTRUCT ps;
    HDC hdc =
BeginPaint(hwnd, &ps);
    DrawBoard(hdc, &rcBoard);
    EndPaint(hwnd, &ps);
    break;
}

case WM_LBUTTONDOWN: {
    if (!currentGame.gameOver
&& currentGame.turn == 0) {
        POINT pt = {
LOWORD(lParam), HIWORD(lParam) };
        if (pt.x >=
rcBoard.left && pt.x <=
rcBoard.right &&
            pt.y >=
rcBoard.top && pt.y <=
rcBoard.bottom) {

            int cellW =
(rcBoard.right - rcBoard.left) /
3;

            int cellH =
(rcBoard.bottom - rcBoard.top) /
3;

            int col = (pt.x -
rcBoard.left) / cellW;
            int row = (pt.y -
rcBoard.top) / cellH;

            if (row >= 0 &&
row < 3 && col >= 0 && col < 3 &&
currentGame.board[row][col] == 0)
{

```

```

currentGame.board[row][col] =
PLAYER_X;

                                int winner =
CheckWinner();
                                if (winner) {

currentGame.gameOver = 1;

currentGame.winner = winner;

AddWinToStats();
                                }
                                else if
(IsDraw()) {

currentGame.gameOver = 1;

AddDrawToStats();
                                }
                                else {

currentGame.turn = 1;

UpdateTitle();

InvalidateRect(hwnd, NULL, TRUE);

ComputerMove();
                                }

UpdateTitle();

InvalidateRect(hwnd, NULL, TRUE);
                                }
                                }
                                }
                                break;
                                }

case WM_COMMAND:
    switch (LOWORD(wParam)) {
        case ID_NEW_GAME:
            InitGame();

```

```

        InvalidateRect (hwnd,
NULL, TRUE);
        break;
    case ID_SAVE_SCORE:
        SaveStatsToFile();
        MessageBoxA (hwnd,
"Результаты сохранены в файл
records.dat", "Сохранение",
MB_OK);
        break;
    case ID_SHOW_RECORDS:
        ShowRecordsWindow();
        break;
    case ID_CLEAR_RECORDS:
        ClearRecords();
        InitGame();
        InvalidateRect (hwnd,
NULL, TRUE);
        break;
    case ID_ABOUT:
        MessageBoxA (hwnd,
        "Крестики-Нолики
v2.0\n\n"
        "(C) 2026\n\n"
        "Особенности:\n"
        "_
Многомодульность (main.c +
random_utils.c)\n"
        "- Ассемблерная
вставка (RDTSC)\n"
        "- Таблица
рекордов\n"
        "- Сохранение
статистики\n"
        "- Русский
интерфейс\n"
        "- Графика, мышь,
меню",
        "О программе",
MB_OK);
        break;
    case ID_EXIT:
        DestroyWindow (hwnd);
        break;
}
break;

```

```
case WM_DESTROY:
    SaveStatsToFile();
    PostQuitMessage(0);
    break;

default:
    return
DefWindowProc(hwnd, msg, wParam,
lParam);
}
return 0;
}
```

```

{
    char telecom_operator[LEN_STRING];
    char tariff[LEN_STRING];
    int price;
    char data[LEN_DATA];
    char* url_data = data;
    bool available = 1;

    char answer;
    bool fbreak = 0;

    system("cls");
    printf("Введите оператора: ");
    fgets(telecom_operator, LEN_STRING, stdin);
    crop(telecom_operator);

    printf("Введите тариф: ");
    fgets(tariff, LEN_STRING, stdin);
    crop(tariff);

    printf("Введите стоимость в рублях: ");
    scanf_s("%d", &price);
    GETCHAR_HANDLER = getchar();

    printf("Введите дату в формате DD.MM.YY\n");
    fgets(data, LEN_DATA, stdin);

    while (1)
    {
        printf("Доступен ли сейчас тариф? Y/N\n");
        answer = answer_handler();
        switch (answer)
        {
            {
            case (0):
                available = 0;
                fbreak = 1;
                break;
            case (1):
                available = 1;
                fbreak = 1;
                break;
            case (-1):
                GETCHAR_HANDLER = getchar();
                printf("Повторите попытку. \nДоступен ли сейчас
тариф? Y/N\n");
            };

            if (fbreak)
                break;
        }

        if (head_url == NULL)

```

```

        head_url = init(0, telecom_operator, tariff, price,
data, available);
    else
    {
        Provider* head = head_url;
        do
        {
            if (head_url->next == NULL)
            {
                append(head_url->id + 1, telecom_operator,
tariff, price, data, available, head_url);
                break;
            }

            if (head_url->next->id == -1)
            {
                head_url->next->id = head_url->id + 1;
                strcpy_s(head_url->next->telecom_operator,
LEN_STRING, telecom_operator);
                strcpy_s(head_url->next->tariff,
LEN_STRING, tariff);
                head_url->next->price = price;
                strcpy_s(head_url->next->data, LEN_DATA,
data);
                head_url->next->available = available;
                break;
            }

            head_url = head_url->next;
        } while (head_url != NULL);
        head_url = head;
    }

    GETCHAR_HANDLER = getchar();
    system("cls");
    return head_url;
}

```

Файл random_utils.c

```
#include <stdio.h> //
Функция генерации
случайного числа через
RDTSC
int asm_random(int max)
{
    unsigned int result;

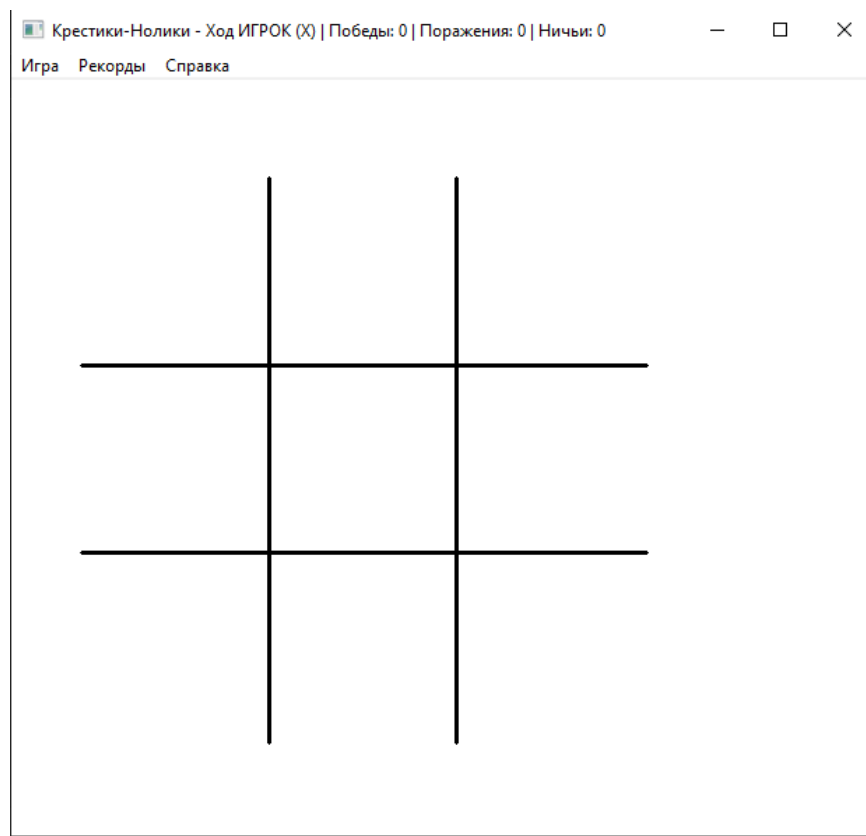
    __asm {
        rdtsc; получить
счётчик тактов
процессора
        xor edx, edx;
обнулить edx для деления
        mov ecx, max;
загрузить max в ecx
        cmp ecx, 0;
проверить, не ноль ли
max
        jle zero_case;
если ноль, вернуть 0
        div ecx; edx:eax
/ ecx, остаток в edx
        mov result, edx;
сохранить остаток в
result
        jmp done
        zero_case :
        mov result, 0
        done :
        ;
    }

    return result;
}
```

Приложение В

Результаты работы программы

Приложение В.1 – Информационная заставка



Приложение С

Результат работы программы

Крестики-Нолики - Вы победили! | Победы: 1 | Поражения: 0 | Ничьи: 0

[Игра](#) [Рекорды](#) [Справка](#)

O		O
	X	O
X	X	X