**Министерство цифрового развития, связи и**

**массовых коммуникаций Российской Федерации**

**ФГБОУ ВО «СибГУТИ»**

**Лабораторная работа №4**

**«CWT-анализ игры Змейка»**

(курсовой работы по объектно-ориентированному программированию)

Выполнил: студент гр. ИП-114

Яворский Д. И.

Проверила: преподаватель Мерзлякова Е.Ю.

Новосибирск 2023 г.

***1. Цель работы***

Научиться проводить CWT-анализ.

***2. Описание анализируемой программы***

Данная программа является реализацией знаменитой игры. «Змейка» состоит из последовательности символов. Управление головой «змейки» должно осуществляться нажатием клавиш со стрелками. Движение прекращается, если голова упирается в хвост или границы игрового поля.

***2.1. Главное окно***

Главное окно приложения - меню, в котором содержатся два пункта: 1) New game (начало игры) и 2) Exit (выход из игры).

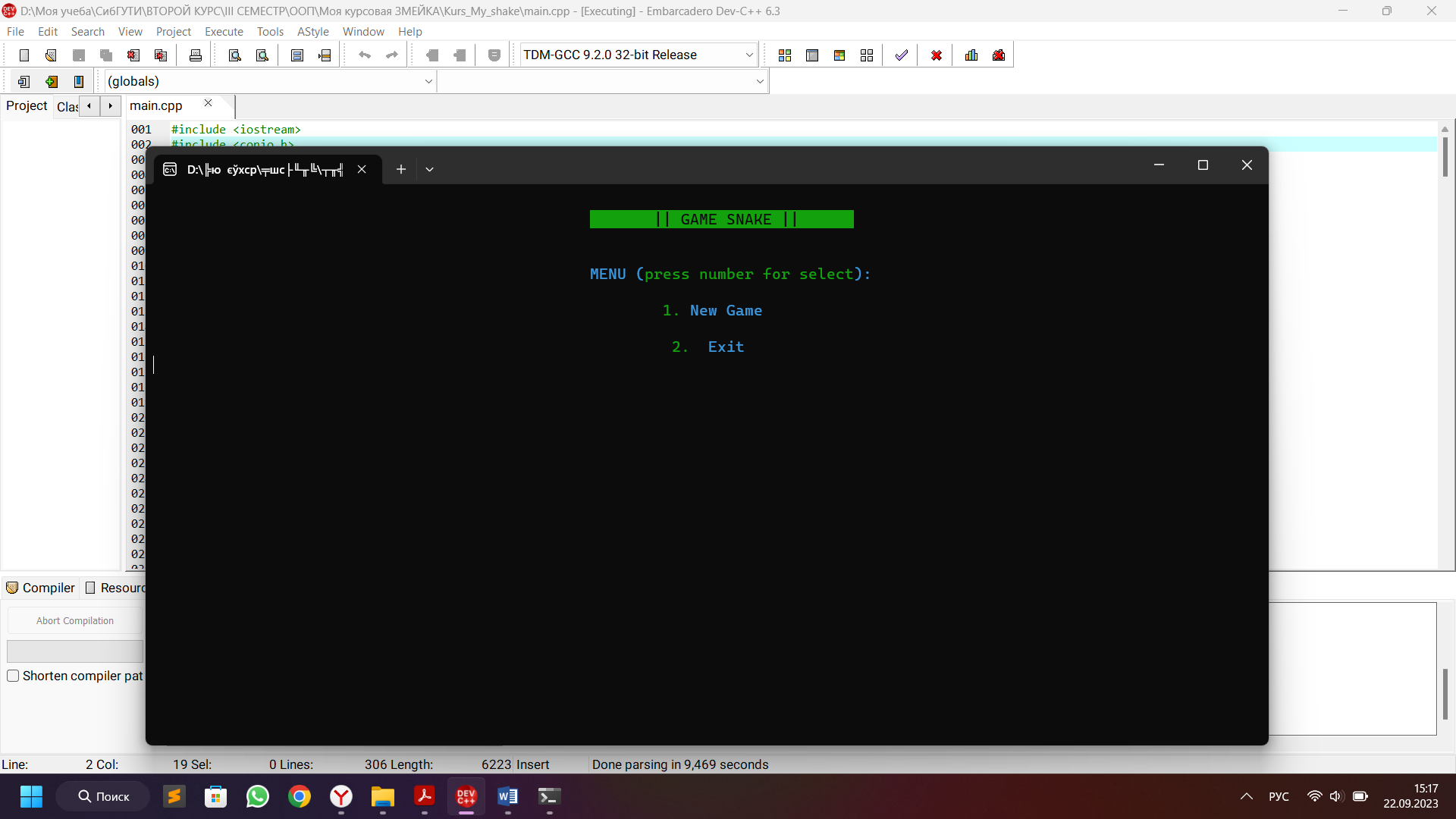


Рис. 1 – стартовое окно

***2.2. Геймплей***

Суть игры заключается в съедании фруктов (в моём случае – звёздочек), и тем самым увеличивать длину хвоста змеи. Благодаря этому у игрока появляется ещё одна задача – не столкнуться с растущим хвостом змеи, а также избегать «стен» игрового поля.

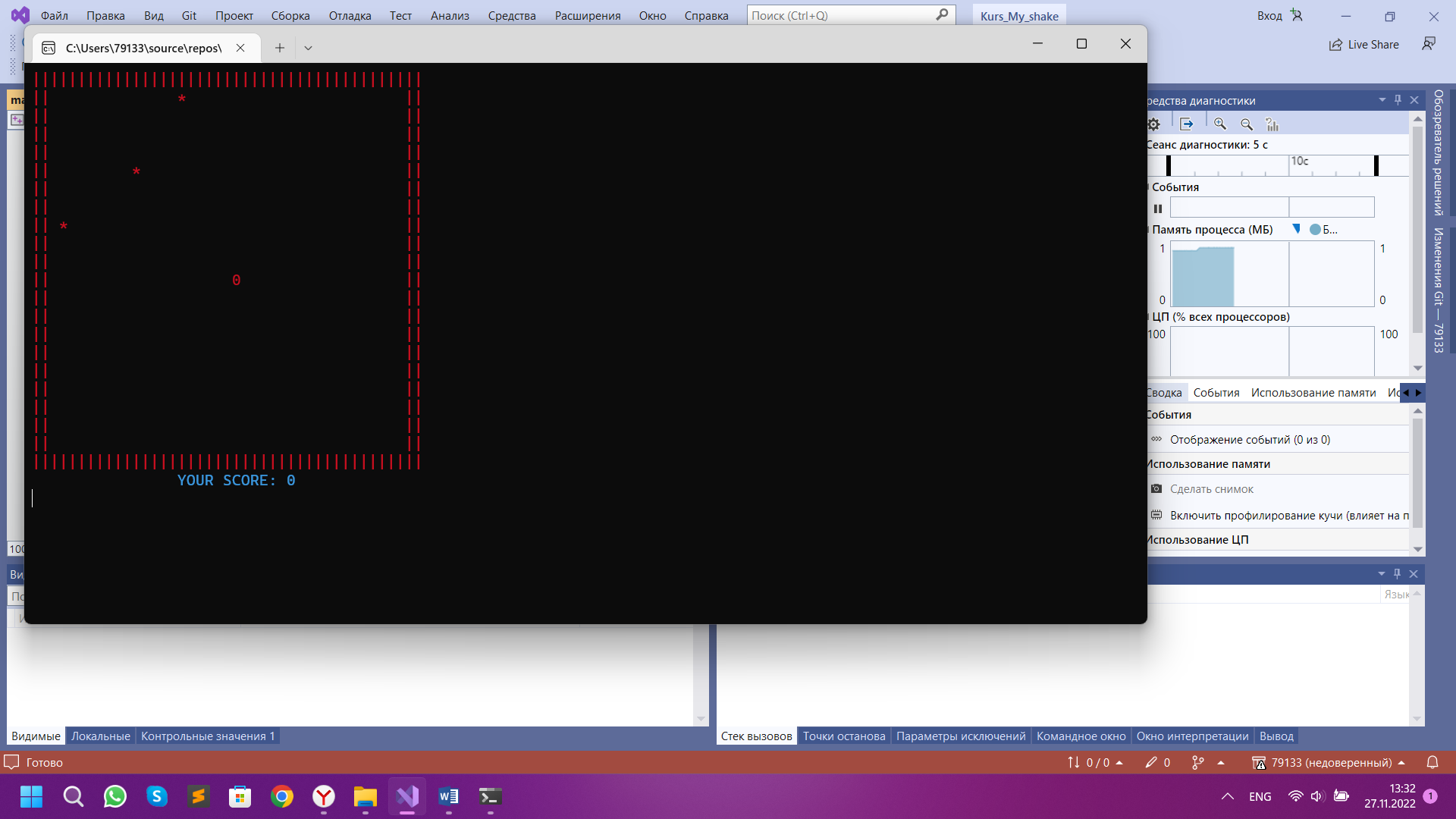
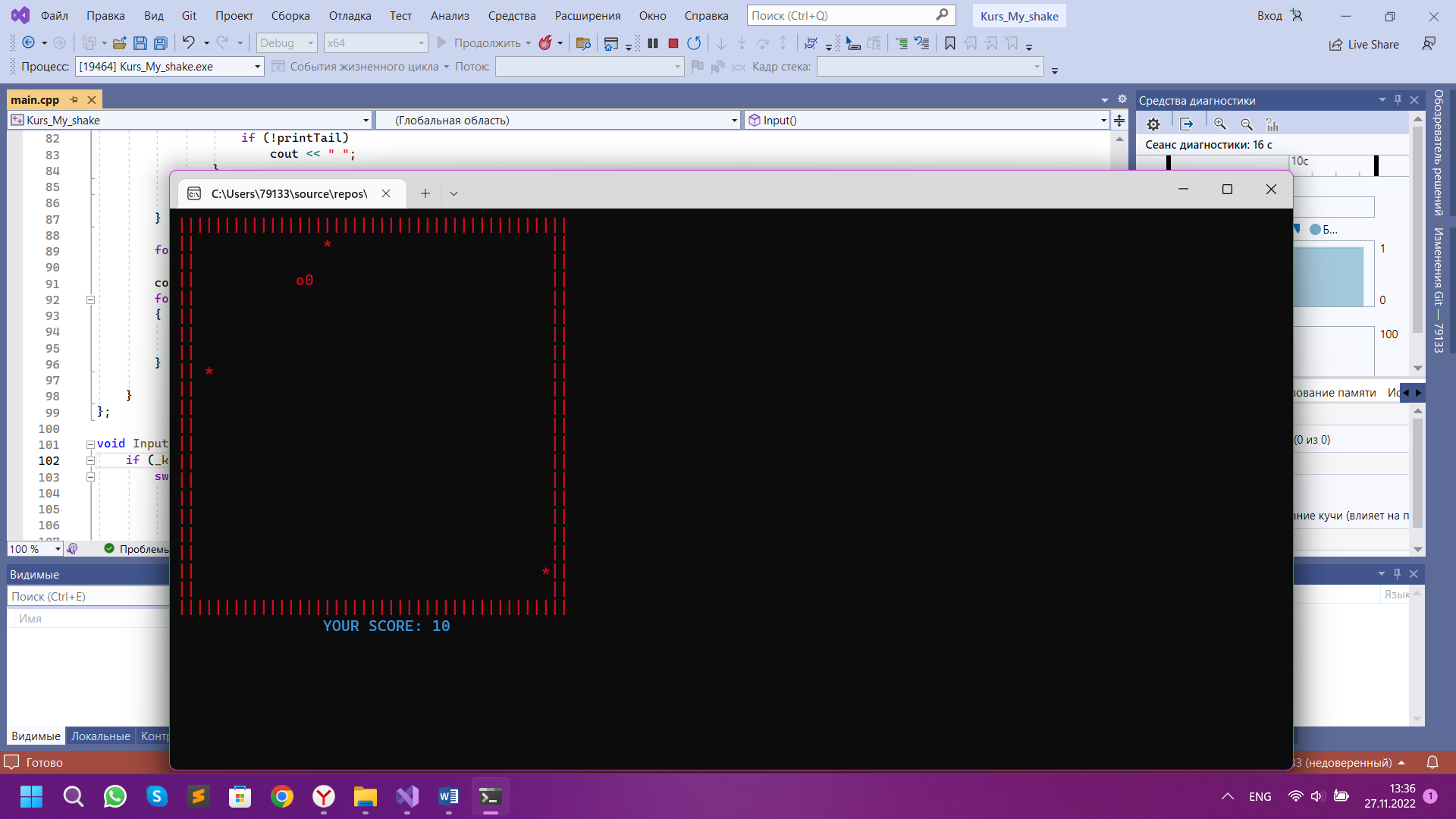
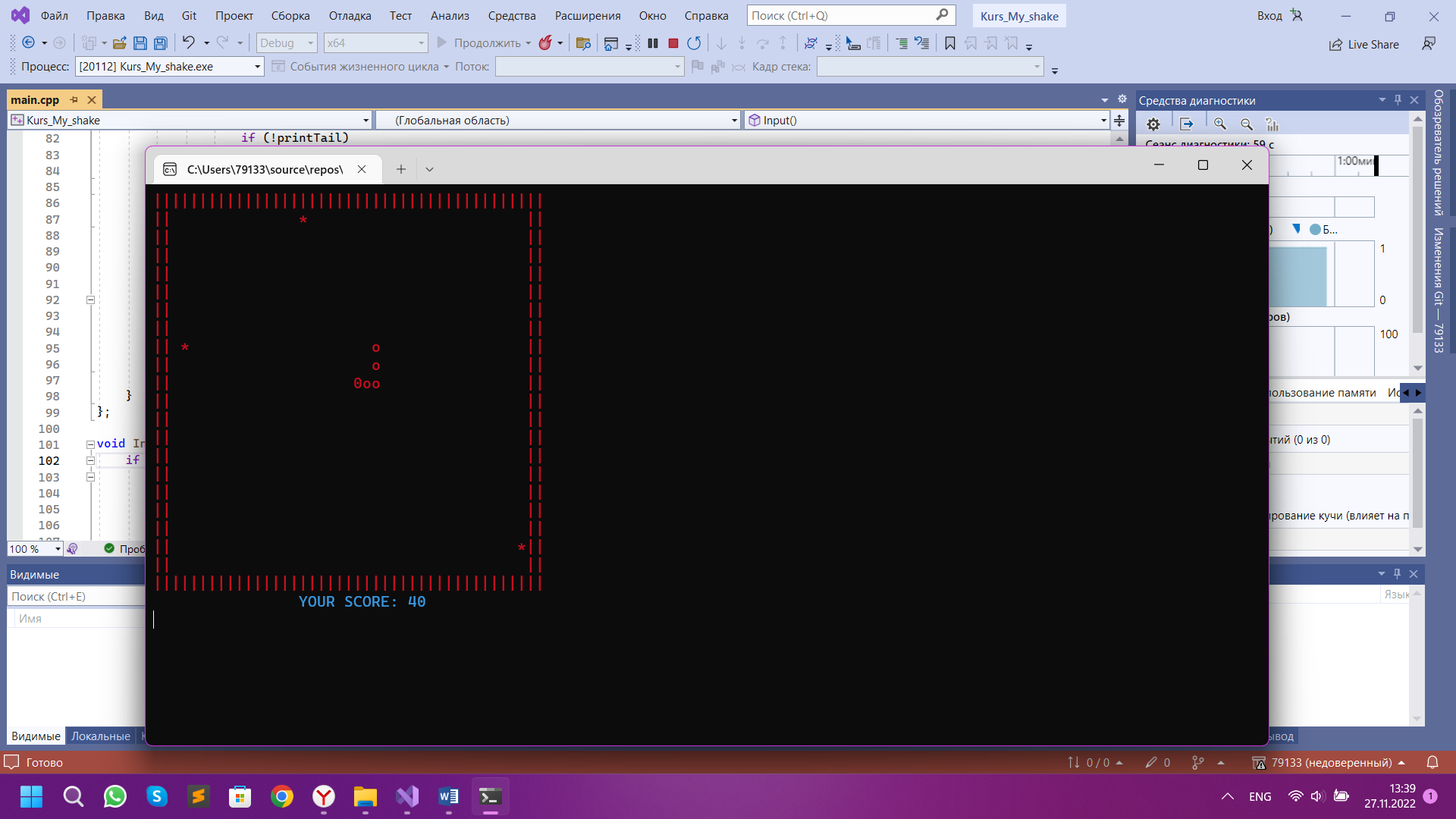


Рисунок 2. Начальная позиция головы «Змейки»

Рисунки 3, 4. Демонстрация движения объекта

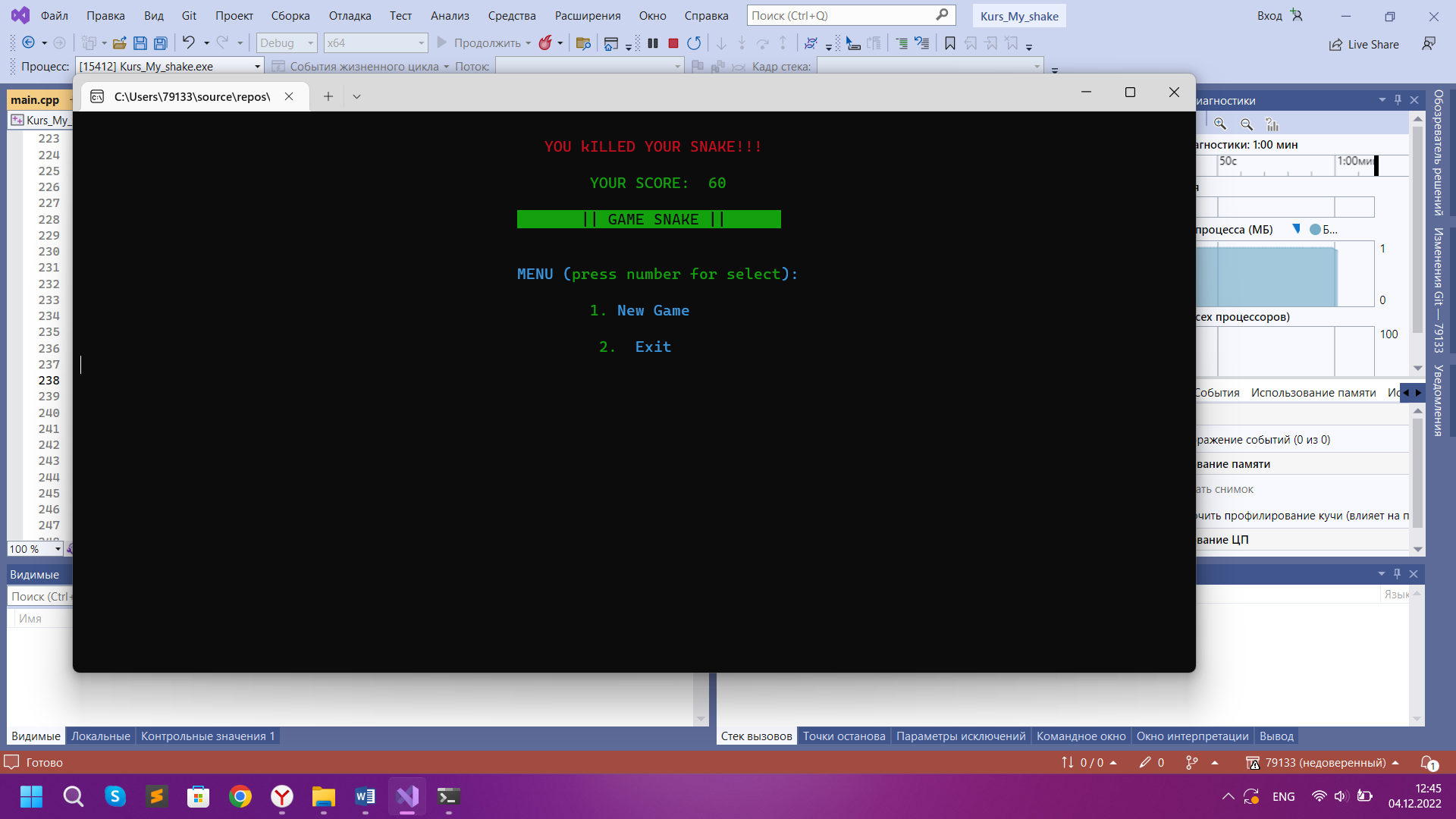


Рисунок 5. Окончание игры

***2.3. Использованные технологии ООП***

*Инкапсуляция*

Вся информация, которая нужна для работы конкретного объекта, должна храниться внутри этого объекта. Если нужно вносить изменения, методы для этого тоже должны лежать в самом объекте — посторонние объекты и классы этого делать не могут. Для внешних объектов доступны только публичные атрибуты и методы. Например, метод для внесения данных в карточку товара должен обязательно быть прописан в классе «Карточка товара». А не в классе «Корзина» или «Каталог товаров».

Такой принцип обеспечивает безопасность и не даёт повредить данные внутри какого-то класса со стороны. Ещё он помогает избежать случайных зависимостей, когда из-за изменения одного объекта что-то ломается в другом.

*Наследование*

В этом принципе — вся суть объектно-ориентированного программирования. Разработчик создаёт:

● Класс с определёнными свойствами;

● Подкласс на его основе, который берёт свойства класса и добавляет свои;

● Объект подкласса, который также копирует его свойства и добавляет свои.

Каждый дочерний элемент наследует методы и атрибуты, прописанные в родительском. Он может использовать их все, отбросить часть или добавить новые. При этом заново прописывать эти атрибуты и методы не нужно.

*Полиморфизм*

Один и тот же метод может работать по-разному в зависимости от объекта, где он вызван, и данных, которые ему передали. Например, метод «Удалить» при вызове в корзине удалит товар только из корзины, а при вызове в карточке товара — удалит саму карточку из каталога.

То же самое с объектами. Можно использовать их публичные методы и атрибуты в других функциях и быть уверенным, что всё сработает нормально.

Этот принцип ООП, как и другие, обеспечивает отсутствие ошибок при использовании объектов.

Реализация иерархии объектов достаточно проста. По большей части мне нужно было реализовать механизм наследования трансформаций от одного объекта к другому (или нескольким). Для этого каждый объект должен иметь собственную матрицу трансформации и, если для него задан “родительский” объект, он должен так же наследовать трансформацию согласно иерархии от этого родительского объекта. Абстрактный класс - это класс, имеющий хотя бы одну чисто виртуальную функцию.

***2.4. Структура классов***

В данной курсовой работе всего 7 классов: класс ***set*** (абстрактный класс); класс ***field*** (отвечает за игровое поле); класс ***set\_snake*** (отвечает за спаун (появление) головы змейки и «фруктов» на поле и содержит начальное значение очков игрока (0); класс ***body*** (отвечает за отрисовку змейки, «фруктов», и границ поля, скорость игры, а также вывод очков на текущий момент игры); класс ***CONTROL*** (отвечает за управление змейкой, а также реализует возможность выхода в меню по клавише Х); класс ***LOGIKA*** (отвечает за логику игры: движение змейки, подсчёт очков за съедание «фруктов», конец игры при достижении границ поля и случайное расположение «фруктов»); класс ***MENU*** (отвечает за меню, его дизайн: расположение пунктов меню, их функционал, цвет, вывод итогового результата очков, полученного за игру, и надписи «GAME OVER: YOU KILLED YOUR SNAKE!!!»).

*Наследование классов:*

1. Класс *set\_snake* наследуется от класса *field*;
2. Класс *body* наследуется от класса *set\_snake*;
3. Класс *LOGIKA* наследуется от класса *field*;
4. Класс *MENU* наследуется от класса *CONTROL*.

***2.5 Программная реализация***

Первым делом было создано несколько глобальных переменных, задано 3 основных цвета в игре, поле для игровой области шириной 40 и высотой 20, создан класс set\_snake. Данный класс содержит в себе только информацию о различных установках игры: спаун головы змеи в середине поля, случайном появлении фруктов на поле, начальном значении очков игрока («0»).

Класс body отвечает за скорость игры, оформление стенок поля, появления головы змейки, спаун «фруктов», увеличения тела змейки, а также вывод очков игрока на экран.

Затем в классе CONTROL было создано два варианта управления: с помощью стрелок и символов в английской раскладке. И добавлен быстрый переход в меню игры нажатием на клавишу «х» в английской раскладке.

В классе LOGIKA написана логика игры: движение змейки (влево, вправо, вверх, вниз), при движении в одну сторону нельзя начать двигаться в противоположную, условие для окончания игры: при достижении одной из 4-х стенок или если голова упирается в хвост игра заканчивается. За «съедание» каждого фрукта даётся 10 очков.

Класс MENU отвечает за меню игры, состоящее из двух пунктов: 1) новая игра; 2) выход. Для перехода в меню из самой игры нужно нажать «x» в английской раскладке. Также при проигрыше игроку выводится сообщение: «YOU KILLED YOUR SNAKE!!!».

После этого было добавлено ещё несколько небольших классов, добавлены необходимые элементы, требуемые заданием, такие как наследуемые классы, инициализации, перегрузки, конструкторы и т.д. В конце была доделана визуальная часть игры: размер поля, цвет символов и оформление меню.

***3. CWT анализ двух выбранных задач***

***3.1. Задача №1, список доступных действий***

Попробовать сыграть в игру.

***Список действий:***

1. Запустить игру;
2. Начать игру;
3. Попробовать набрать 100 очков;

***3.2. Задача №1, анализ действий: проблемы и их решение***

Пройдемся по всем шагам:

* ***Первые 2 действия под силу даже самому неопытному пользователю***. Так что проблем с удобством запуска игры возникнуть не должно. Возможно стоило бы добавить выбор пунктов в меню при помощи стрелочек или мыши (изначально это реализовано при помощи нажатия цифр, 1 или 2) и несколько слов о разработчиках.
* ***С управлением «змейкой» в первый раз смогла помочь инструкция при первом запуске в текущей сессии новой игры***, появлялись бы используемые клавиши в игре и их пояснение, например помимо клавиш управления, есть быстрый выход в меню при помощи нажатия «Х».
* ***В игре нет возможности поставить игру на паузу.***

Возможные решения:

1) Как было указано выше, одно из решений данной проблемы, может быть добавление подсказок для неопытного пользователя;

2) Сделать возможность ставить игру на паузу и продолжить играть, когда захочет пользователь.

***3.3. Задача №2, список доступных действий***

Геймплей игры устарел, поэтому чтобы завлечь пользователя, нужно сделать игру интереснее.

***Список действий:***

1. Разработать интересные механики;
2. Добавить возможность сохранения рекордов.

***3.4. Задача №2, анализ действий: проблемы и их решение***

Пройдемся по всем шагам:

* ***Применение интересных механик в геймплее игры.***
* ***Сохранение рекордов.***

Возможные решения:

1. Интересные механики:
2. При получении определенного количества очков, повышать уровень сложности игры (скорость перемещения змеи), оповещая пользователя о изменении уровня;
3. Добавление несколько жизней змее, чтобы игра не казалась такой сложной.
4. Добавление бонусов на поле, которые может подобрать пользователь как фрукты (**магнит** – притягивание фруктов к змее, **slow motion** – замедление движения змеи на 10 секунд, тем самым давая игроку отдохнуть и заработать как можно больше очков, **перепрыжка** – перепрыгивание тело змеи головой, давая неуязвимость игроку не некоторое время, **жизнь** – дополнительная жизнь и т.д.);
5. Создать третий пункт меню под названием «Рекорды», при входе в который показывалась бы таблица с рекордами пользователя, возможно даже с датами установления рекордов, либо добавив горячую клавишу «R» для возможности перехода к рекордам прямо из игры или при паузе.