**Acknowledgement**

We extend our heartfelt gratitude to all those who have contributed directly or indirectly to the creation of this project and have supported us along the way. Your efforts have been instrumental in making this project a success, and we are truly grateful for your assistance.

This project has been a journey of learning and growth for us, and we are deeply appreciative of the valuable experiences it has provided, which we believe will have a profound impact on our future careers.

We would like to take this opportunity to introduce our project, the Pac-Man Game Application.

First and foremost, we express our sincere appreciation to Prof. Sai Charan Samudrala for providing us with the invaluable opportunity to delve into web application development. His guidance, constant inspiration, and valuable suggestions have been instrumental throughout the project.

We are also indebted to the faculty members of the Faculty of Computer Application and IT for generously sharing their time and expertise with us, providing guidance, and helping us navigate through the challenges encountered during the project.

Special thanks are also due to all the members of the All-In-One Application for their unwavering support and valuable suggestions during the analysis stage, which have significantly contributed to the project's success.

Once again, we extend our heartfelt thanks to everyone involved in making this project a reality.

**INDEX**

|  |  |  |
| --- | --- | --- |
| **Sr. No** | **Content** | **Page No.** |
| 1. | About | 1 |
| 2. | Problem Statement | 3 |
| 3. | Introduction | 6 |
| 4. | Implementation | 7 |
| 5. | Design | 9 |
| 6. | Coding | 10 |
| 7. | Output | 20 |
| 8. | Conclusion | 25 |

**About**

The Pac-Man project is an implementation of the classic arcade game Pac-Man using C++. This project aims to recreate the nostalgic gameplay experience of the original Pac-Man while introducing some additional features and customization options.

**Key Features:**

1. **Classic and Survival Modes:** The game offers two distinct modes: Classic mode, where players navigate through mazes to collect dots and avoid the Eater, and Survival mode, where players must survive as long as possible while collecting dots and avoiding the Eater with a faster pace.
2. **Difficulty Levels:** Players can choose from three difficulty levels (Easy, Normal, and Hard) to tailor the gameplay experience to their skill level. Each difficulty level adjusts the speed and behavior of the ghost, providing a varied challenge for players.
3. **Dynamic Maze Generation:** The game features dynamically generated mazes with walls, dots and the ghost. This ensures that each playthrough offers a unique maze layout, keeping the gameplay fresh and engaging.
4. **User Interface:** The game provides a simple user interface that allows players to navigate menus, select game modes, and adjust settings easily.

**Technologies/Libraries Used:**

- C++ programming language.

- Windows API for console manipulation.

**How To Play:** Players control Pac-Man using the arrow keys on the keyboard. In Classic mode, the objective is to navigate through the maze, eat all the dots, and avoid being caught by the ghost. In Survival mode, players aim to survive as long as possible while collecting dots and avoiding the ghost, with the speed of the ghosts increasing as the game progresses.

**Objective**: The objective of the game is to move Pac-Man around the maze-like screen to consume lines of 240 dots while avoiding the ghost. If Pac-Man collides with a ghost, he loses a life and the game restarts. The player has 3 Lives until the game exits and asks the player to start again.

**Problem Statement**

The development of the Pac-Man project posed several challenges and objectives that needed to be addressed to create a fully functional and enjoyable game experience. These challenges included:

**Game Logic Implementation:** One of the primary challenges was implementing the core game mechanics and logic, including Pac-Man movement, ghost behavior, collision detection, scoring, and level progression. These elements needed to faithfully replicate the gameplay of the original Pac-Man while providing a smooth and responsive experience for players.

**AI Algorithms:** Developing effective artificial intelligence (AI) algorithms for the ghost presented a significant challenge. The ghost needed to exhibit varied behaviors, such as chasing Pac-Man, and patrolling the maze. Designing AI that provided a balanced challenge for players while remaining true to the original game's behavior required careful planning and testing.

**User Interface and Control :** Designing an intuitive user interface (UI) and responsive controls was essential for providing an enjoyable gameplay experience. Players needed clear instructions on how to play the game, as well as easy-to-use controls for navigating Pac-Man through the maze.

**Objectives:**

* Implement the core game mechanics of Pac-Man, including movement, collision detection, scoring, and AI ghost behavior.
* Create visually appealing maze that provide a challenging gameplay experience.
* Balance the difficulty level to cater to players of different skill levels.
* Test the game thoroughly to identify and resolve any bugs or gameplay issues.

**Pros:**

* Gameplay Mechanics: The code implements basic Pac-Man gameplay mechanics such as moving the player character (hero), collecting dots, avoiding enemies (eater), and scoring points.
* Multiple Modes: The game offers two modes - Classic mode and Survival mode, providing players with different gameplay experiences.
* Difficulty Levels: The game allows players to select from different difficulty levels (Easy, Normal, Hard) in Classic mode, which can cater to players with varying skill levels.
* Clear Instructions: The code includes instructions on how to play the game, providing guidance to players who are unfamiliar with Pac-Man.
* Dynamic Difficulty: In Survival mode, the speed of the enemy (eater) increases as the player scores more points, adding a level of challenge and dynamism to the gameplay.

**Cons:**

* Platform Dependency: The code uses Windows-specific libraries (Windows.h) for handling console input/output and cursor manipulation, making it non-portable to other operating systems like Linux or macOS.
* Global Variables: The code relies heavily on global variables, which can make it harder to understand and maintain, especially as the codebase grows larger.
* Limited Graphics: The game relies solely on console text output, lacking graphical user interface (GUI) elements, which may limit the visual appeal and immersion of the game.
* Hardcoded Map: The game map is hardcoded directly into the source code, making it challenging for developers to modify or expand the game's level design without modifying the source code itself.

**Introduction**

**Overview:** The Pac-Man project is a recreation of the classic arcade game originally developed by Namco in 1980. In this project, we aimed to faithfully replicate the gameplay and experience of the original Pac-Man while incorporating modern programming techniques and features. The player controls Pac-Man, who must eat all the dots inside an enclosed maze while avoiding the Eater.

**Purpose:** The primary purpose of this project is to showcase our basic skills in game development, programming, and problem-solving. By creating a functional and enjoyable version of Pac-Man, we demonstrate our ability to implement complex game mechanics, design intuitive user interfaces, and optimize performance.

**Target Audience:** This documentation is intended for game enthusiasts, and anyone interested in learning about the process of recreating classic video games. Whether you're a seasoned programmer looking to delve into game development or a Pac-Man fan curious about how the game was brought to life, this documentation provides insights into the project's development process and implementation details.

Pac-Man holds a special place in gaming history as one of the most iconic and influential video games of all time. Our recreation of Pac-Man pays homage to this classic title while introducing it to a new generation of players. Additionally, the project serves as a learning experience for us to explore game development concepts and techniques in a practical setting.

**Implementation**

Implementing a Pac-Man game requires programming skills and an understanding of game development concepts. Here are some general steps involved in implementing a Pac-Man game:

**Map Display**:

Implement a function to display the game map on the console.

**Player Control**:

Allow the player to control Pacman's movement using arrow keys.

**Game Modes:**

Implement two game modes: **Classic and Survival**

In Classic mode, the player chooses a difficulty level (Hard, Normal, or Easy) and plays through a predefined maze, scoring points by eating dots and avoiding the Eater.

In Survival mode, the player navigates through a maze, scoring points by eating dots while the Eater's speed increases progressively.

**Enemy AI:**

Implement an AI for the Eater character that navigates through the maze and tries to catch Pacman.

**Collision Detection:**

Detect collisions between Pacman, dots, and the Eater.

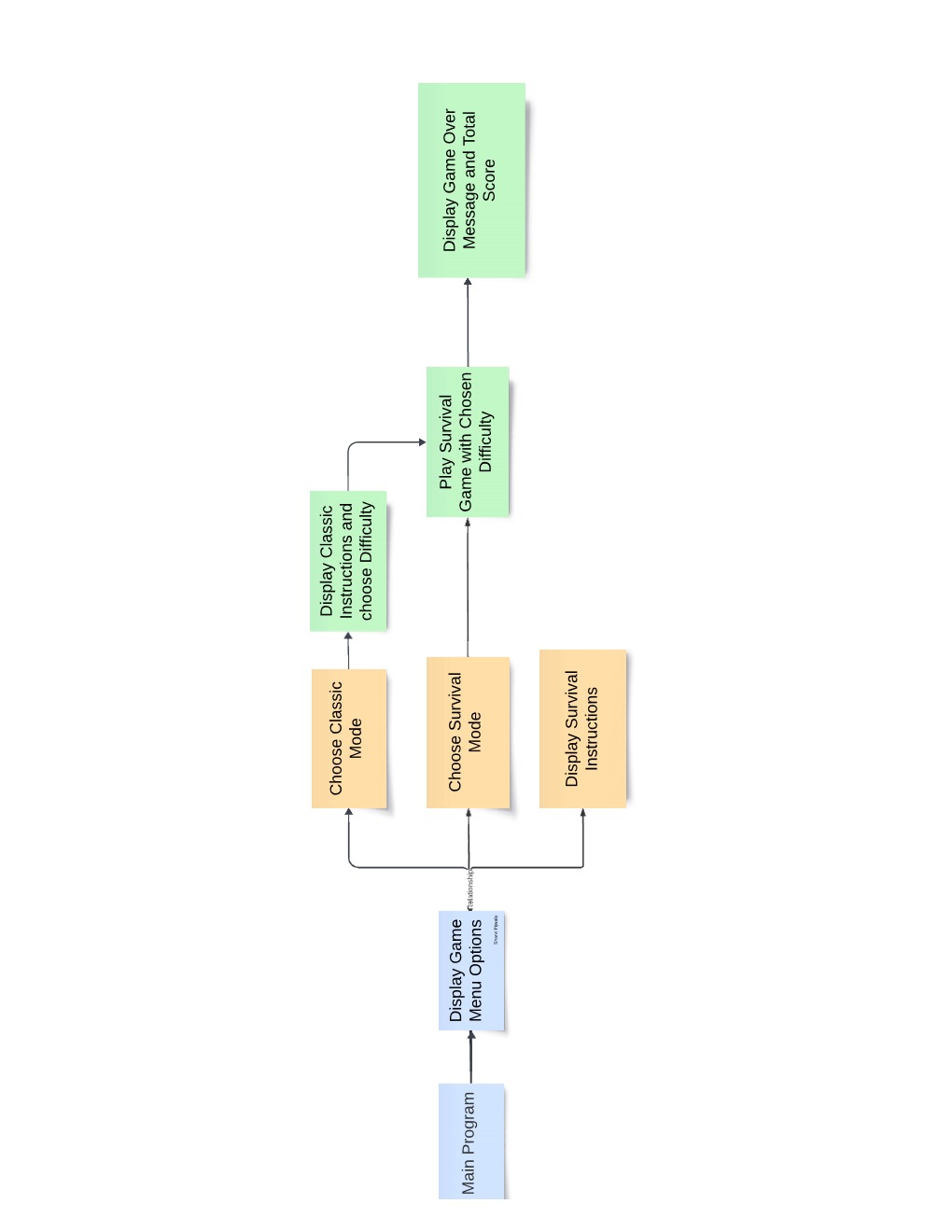
**Score Tracking:**

Track and display the player's score during gameplay.

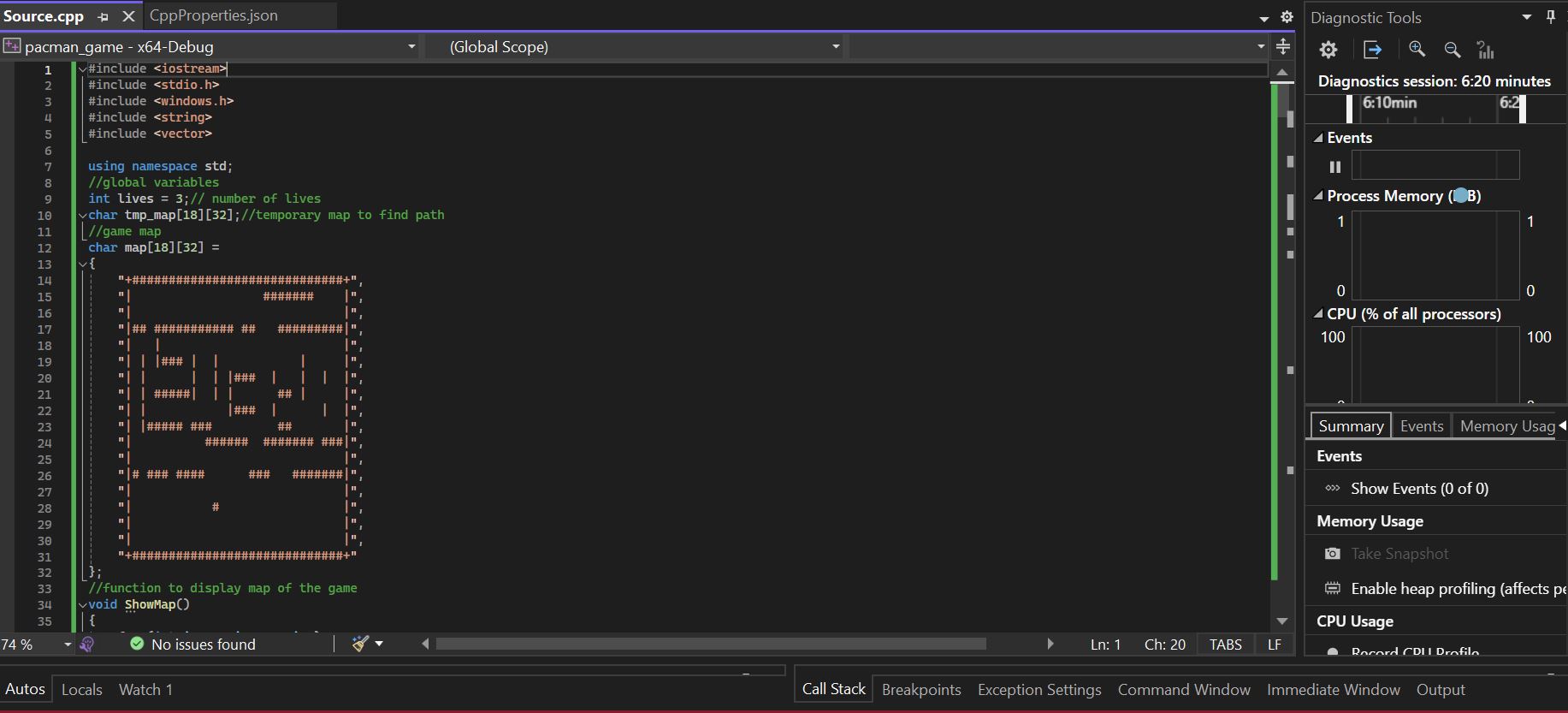
**Game Over:** End the game when Pacman collides with the Eater or when the player completes the objective (e.g., surviving for a certain duration or reaching a specific score threshold).

**User Interface:** Provide a simple menu interface for selecting game modes and difficulty levels.

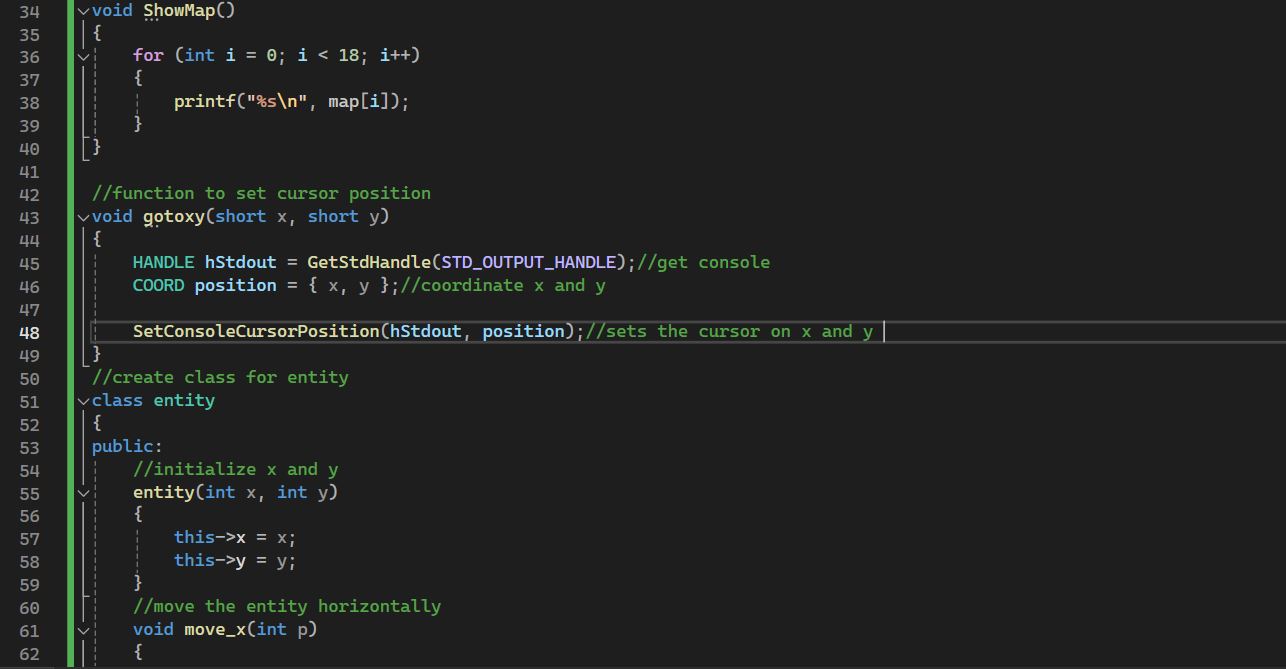
**Design**

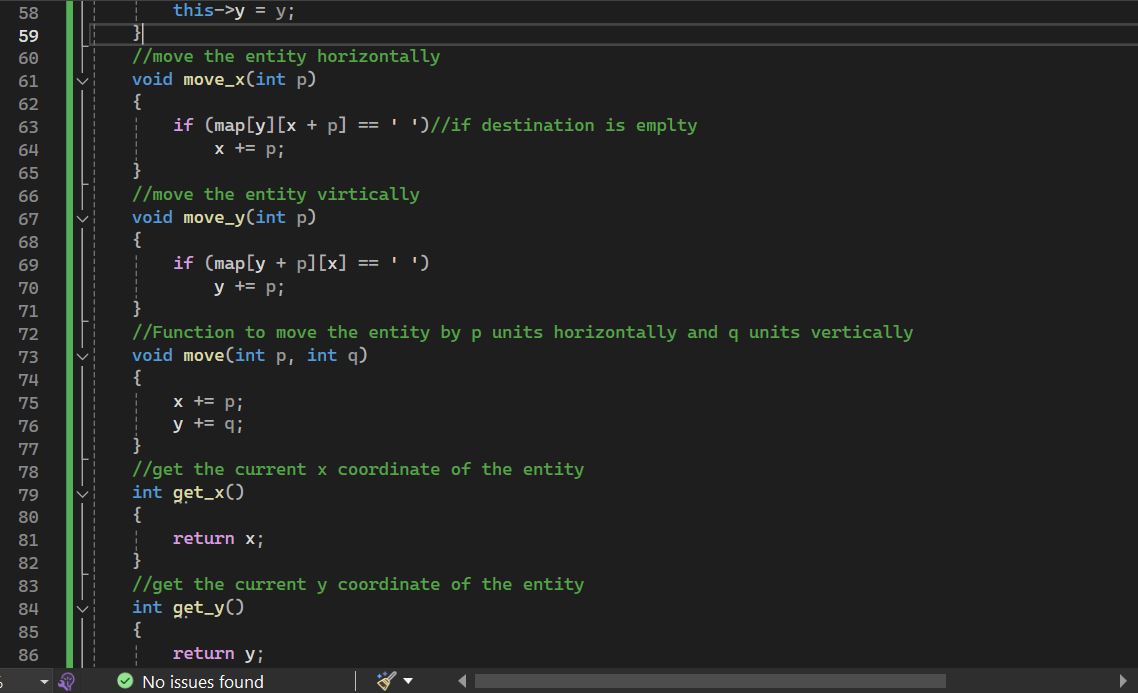
****

**Coding**

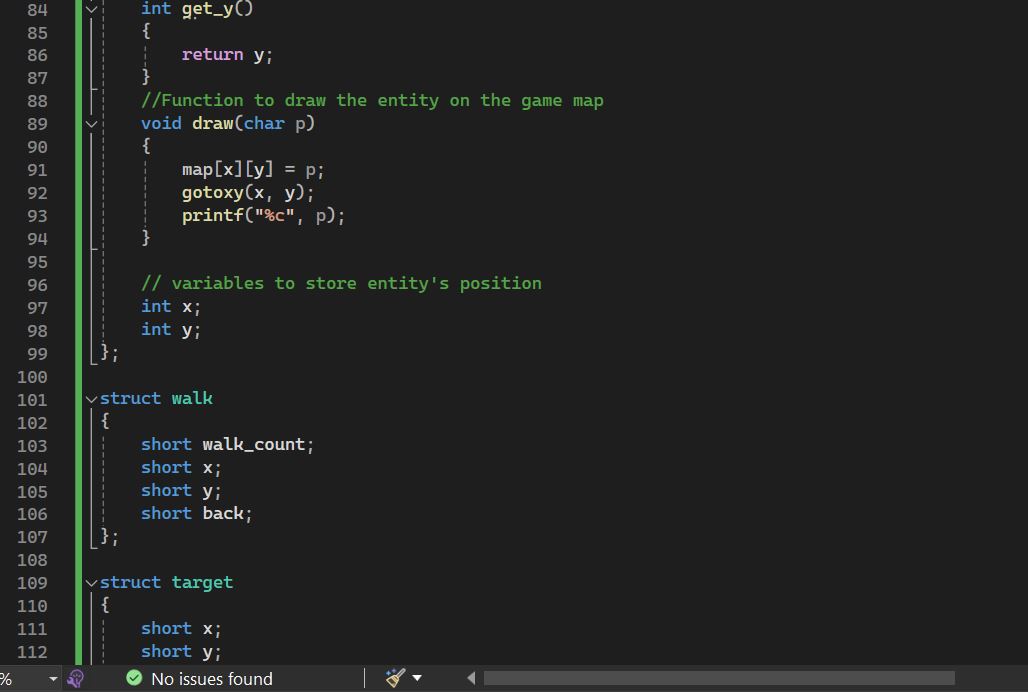


This include necessary header files for input/output, standard C library functions, Windows API functions, and data structures like strings and vectors and game map as a 2D character array.

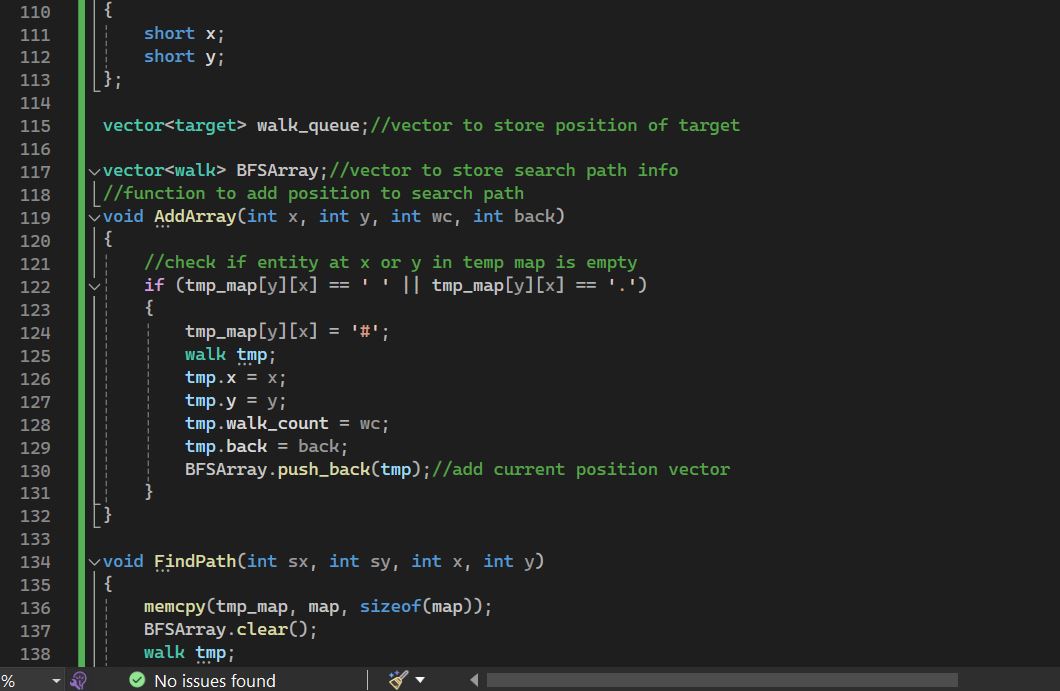
Defines a function **ShowMap()** which is responsible for displaying the game map on the console, **gotoxy()** to set the cursor position in the console window.

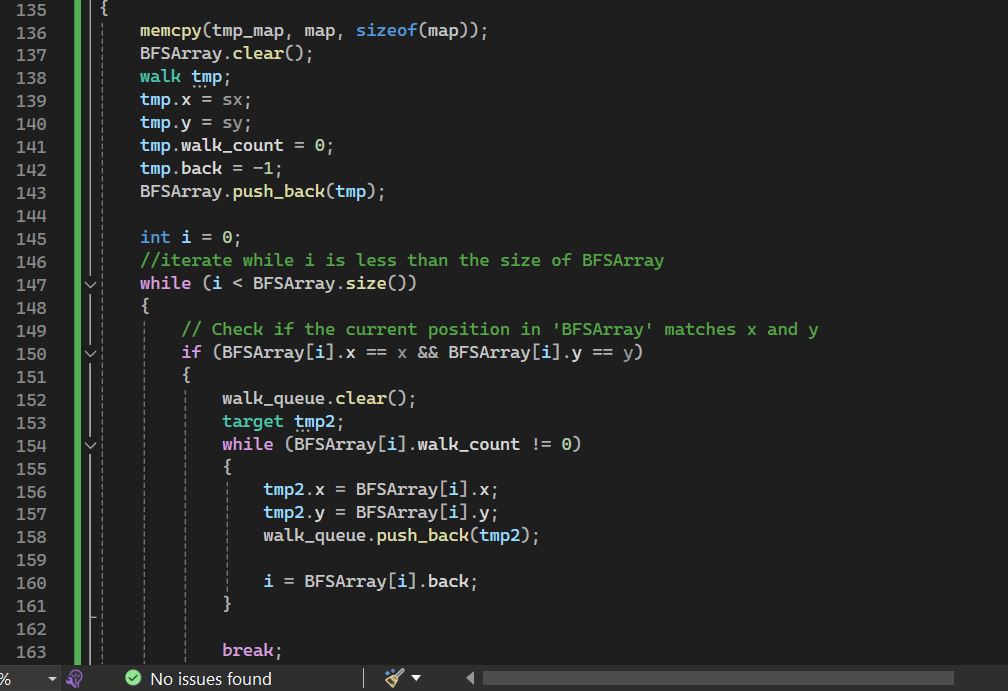


Defines a class **entity** which represents game entities such as the player character and enemies.

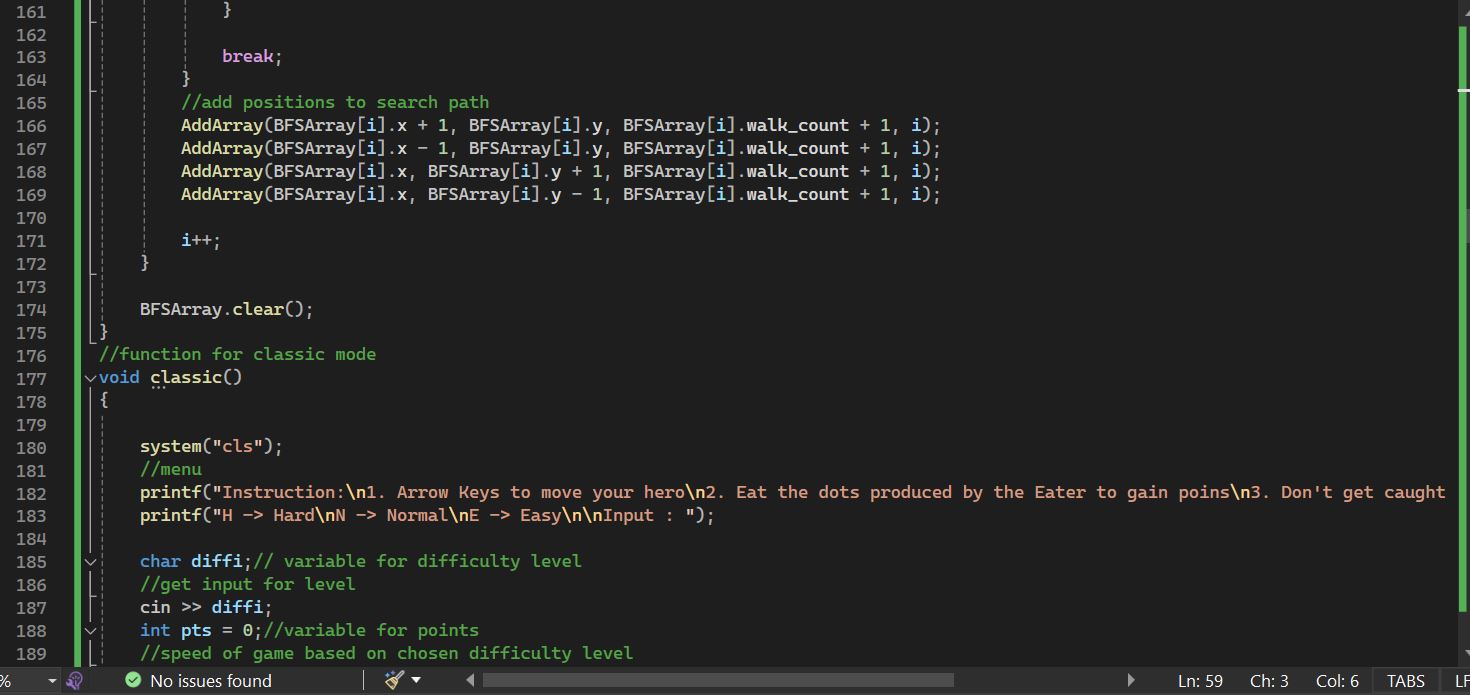


Defines structs **walk** and **target** for storing pathfinding data and target positions.

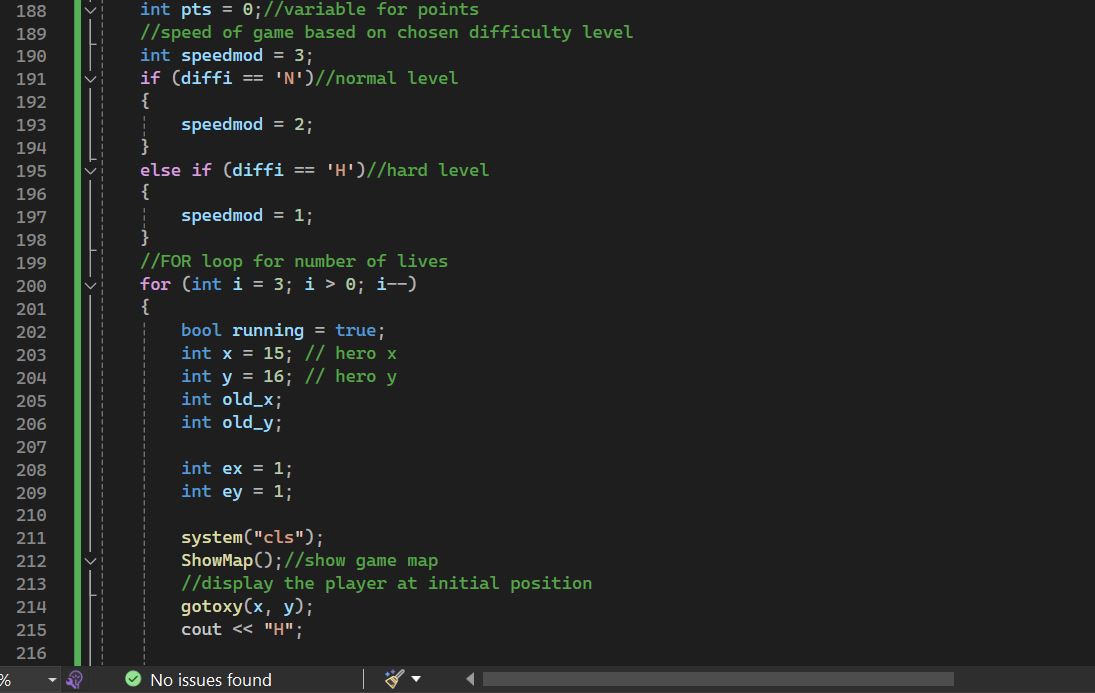


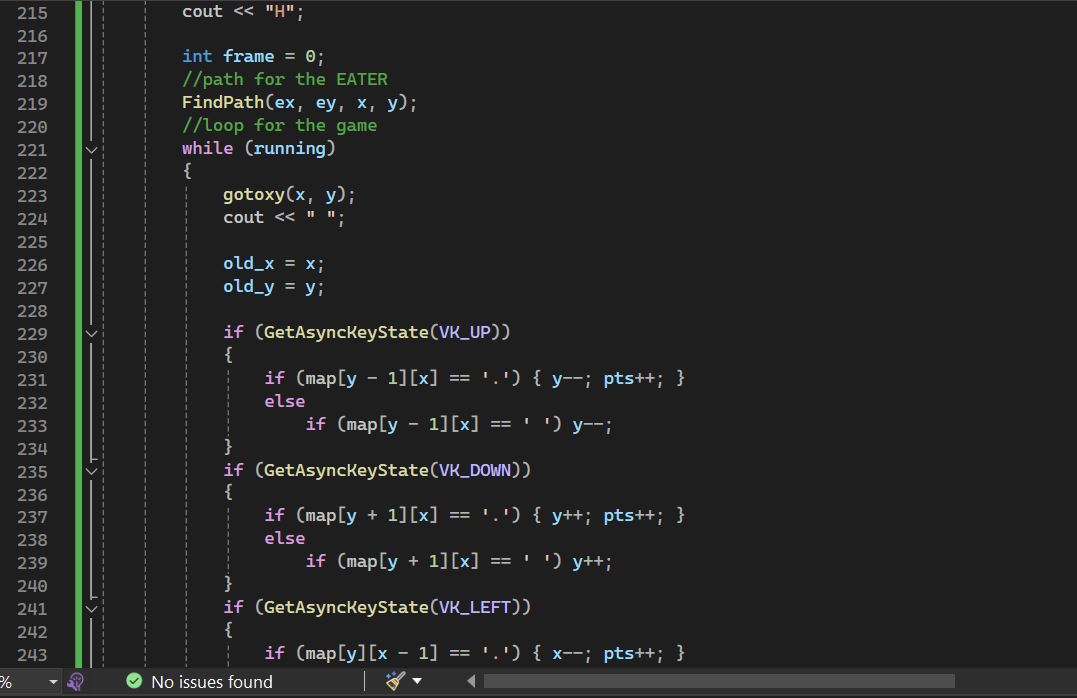


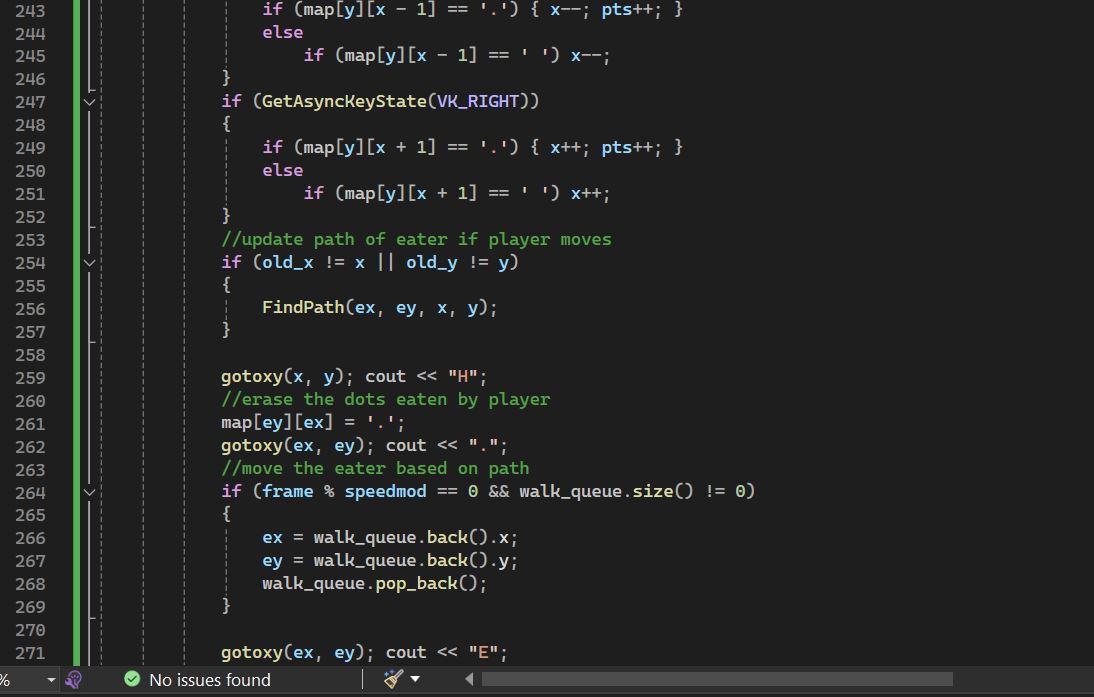
Defines utility functions **AddArray()** and **FindPath()** used for pathfinding.

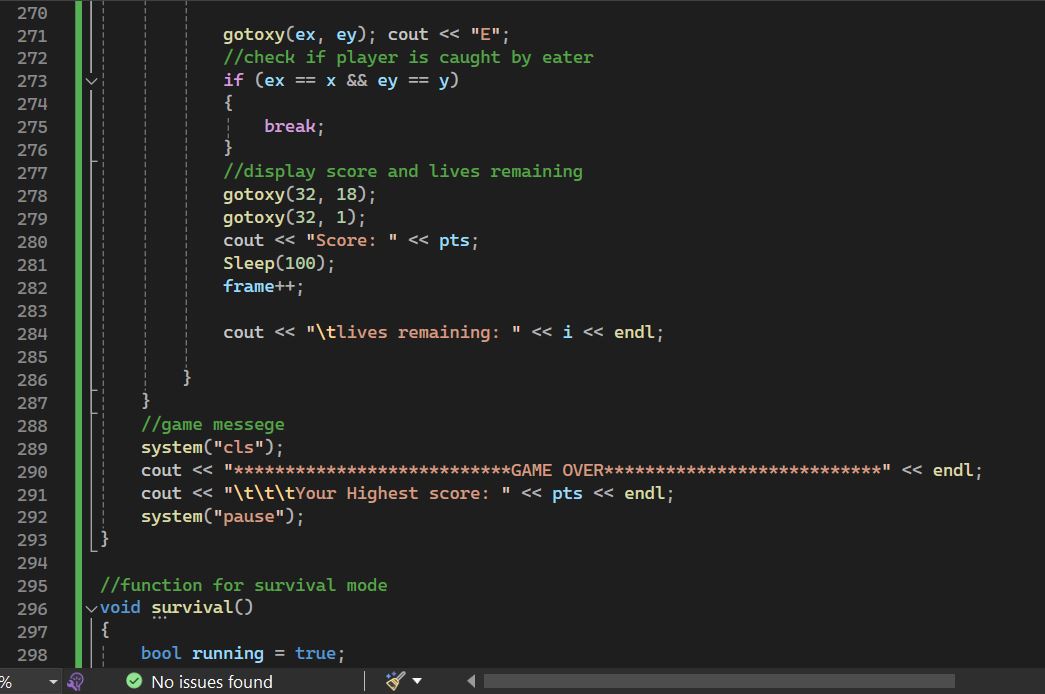


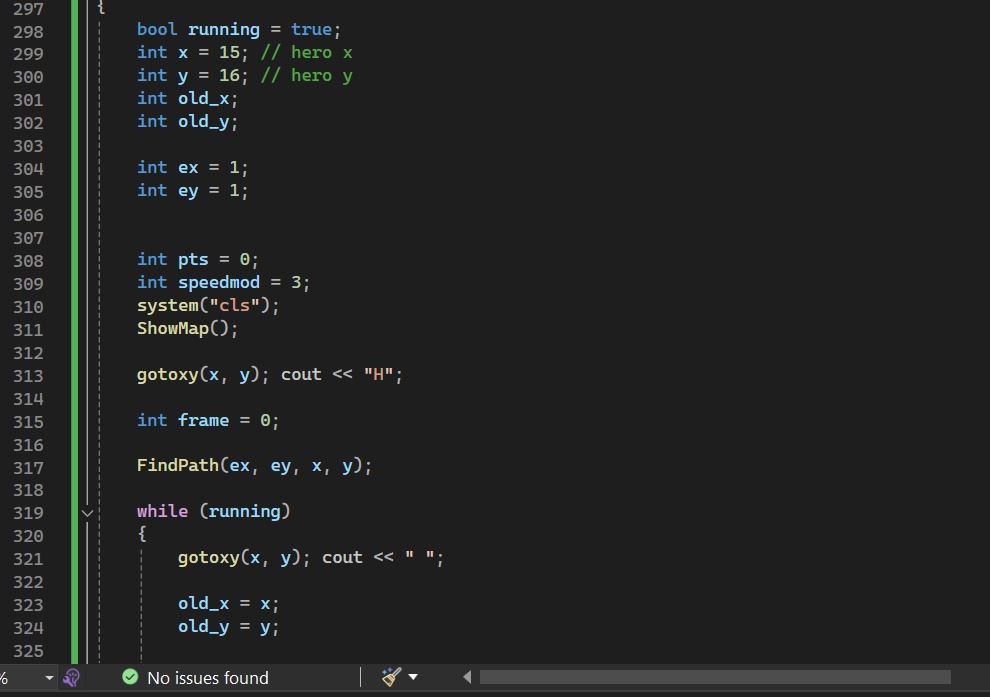
Defines the **classic()** function which implements the classic mode of the game.



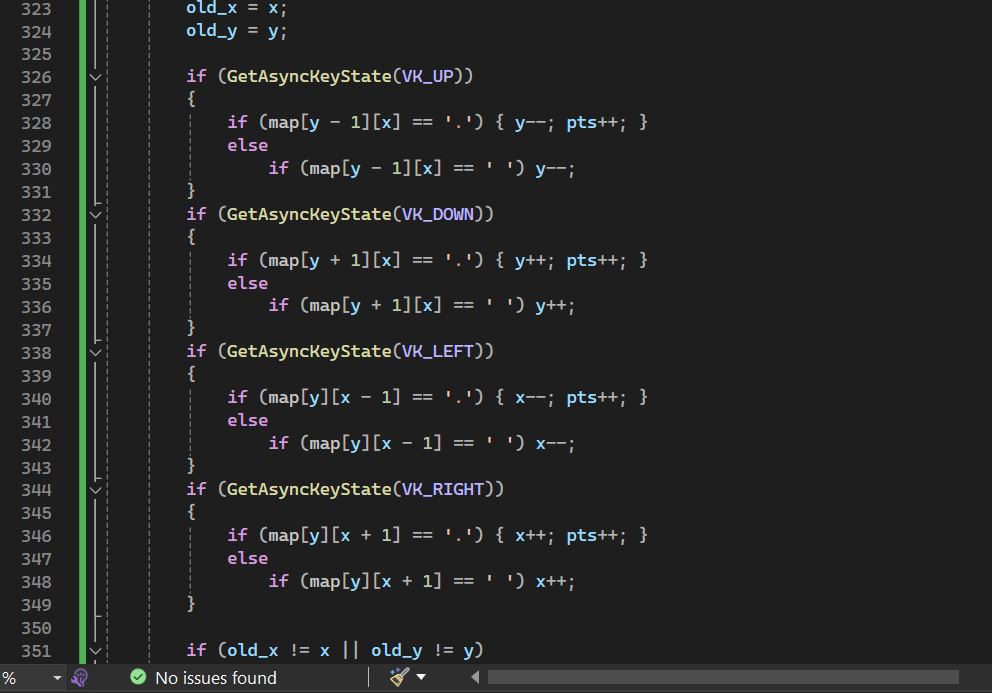


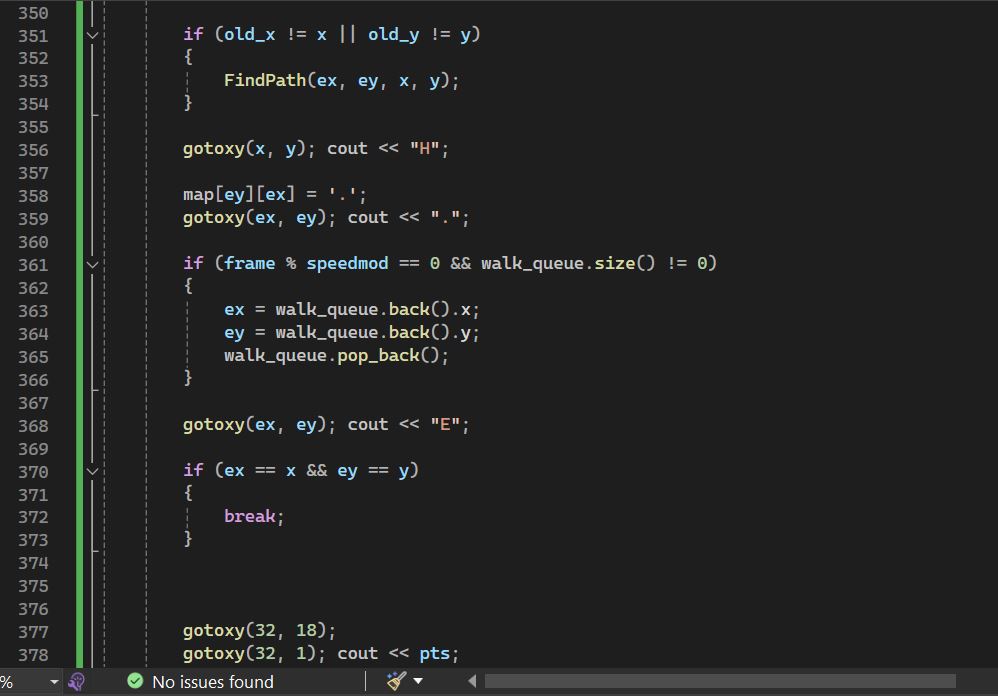


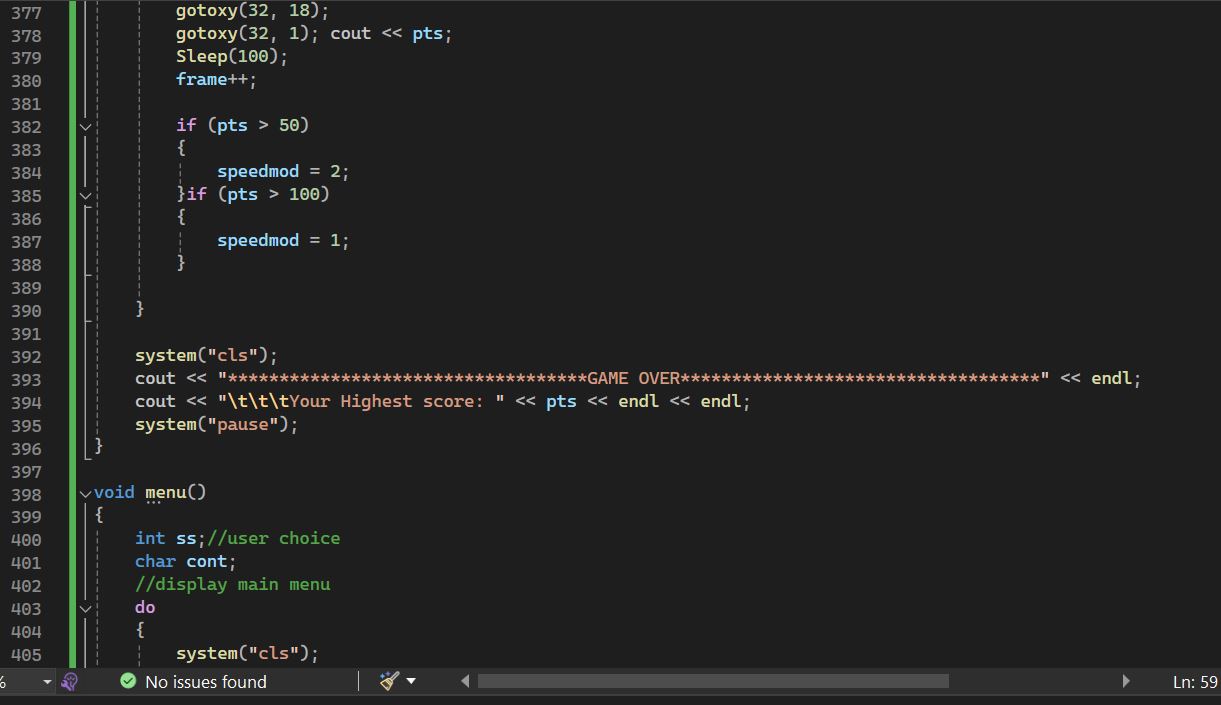


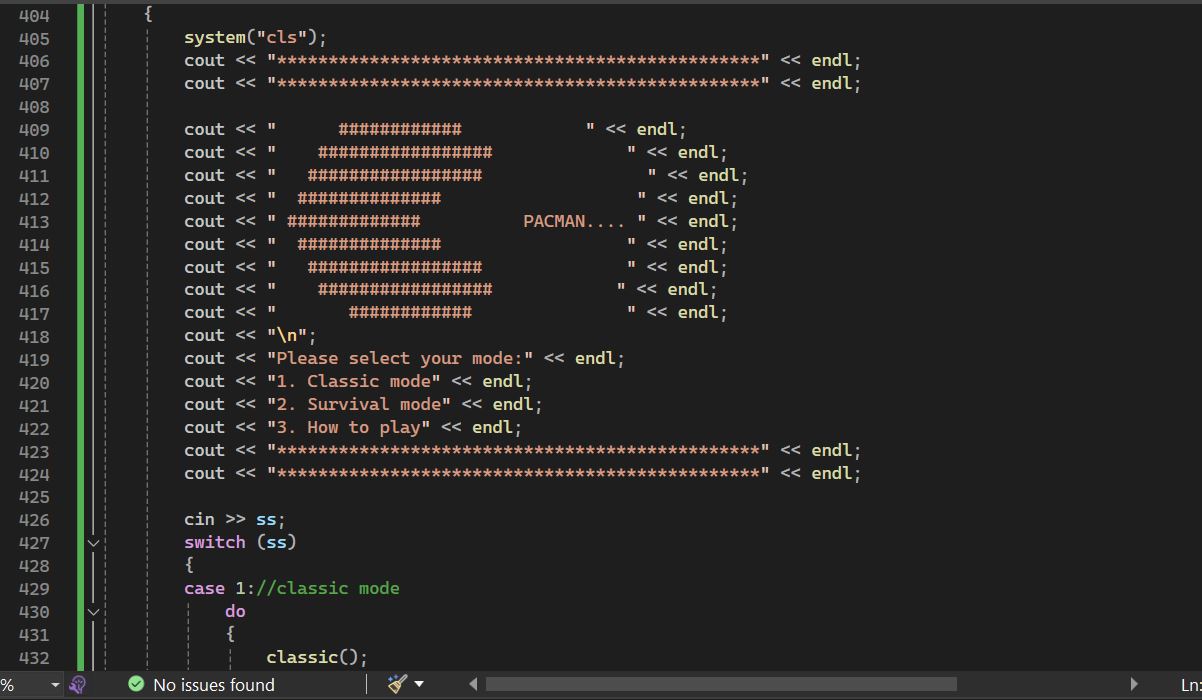


Defines the **survival()** function which implements the survival mode of the game.

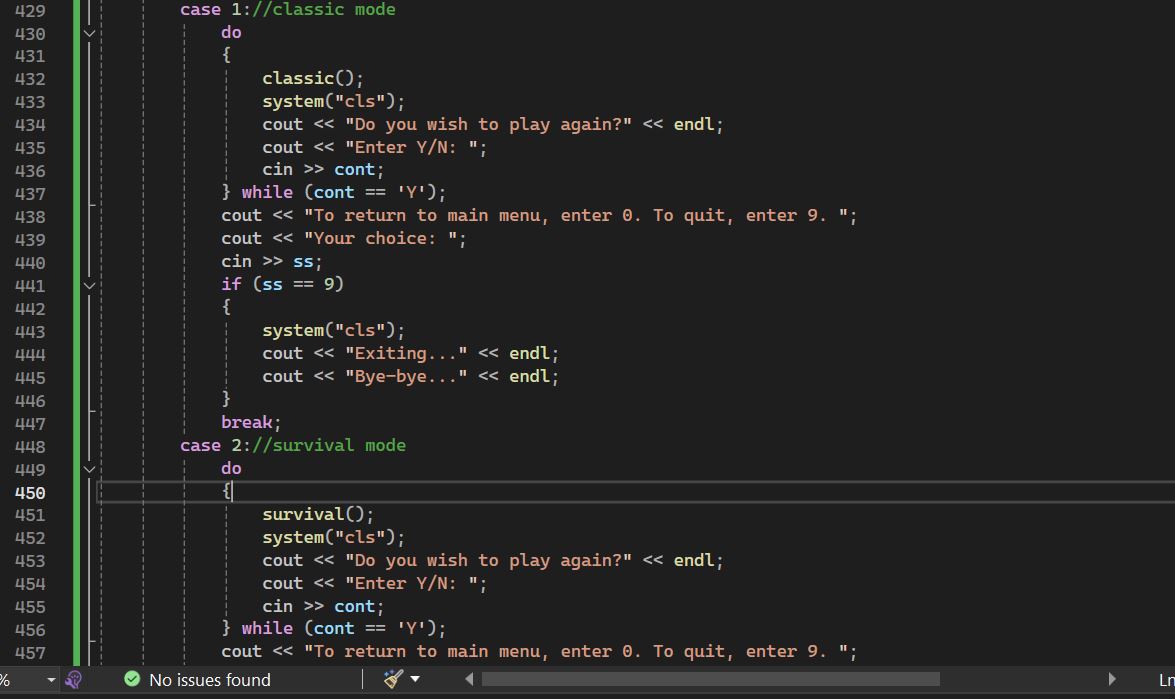


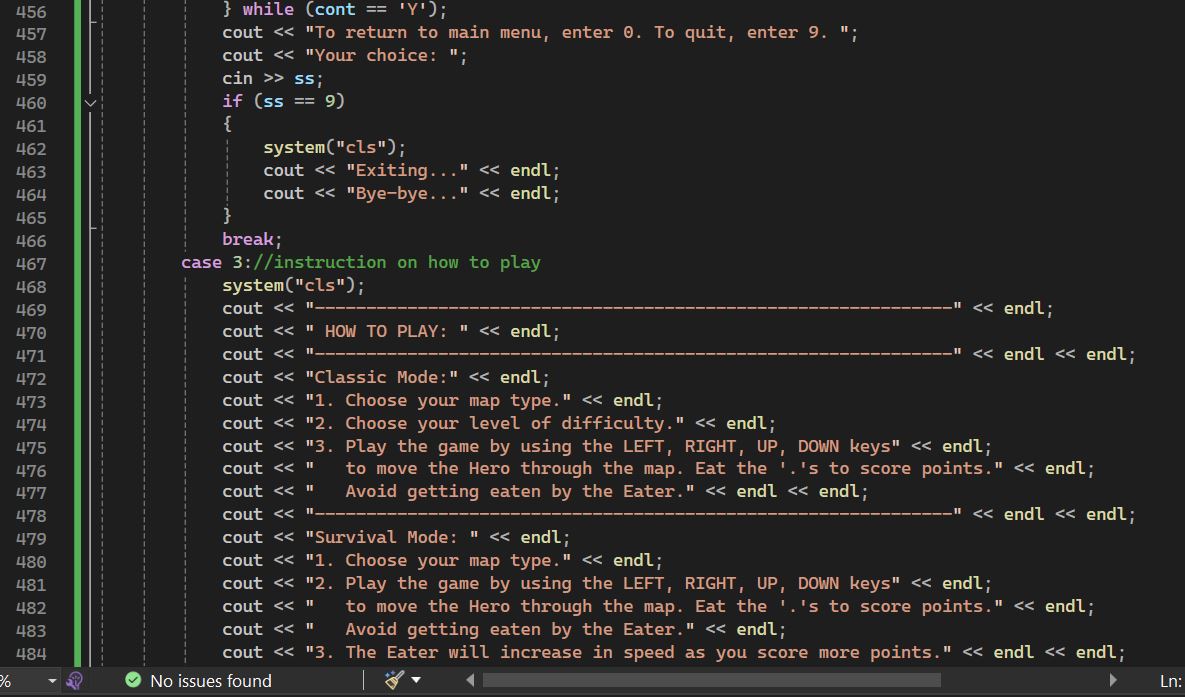


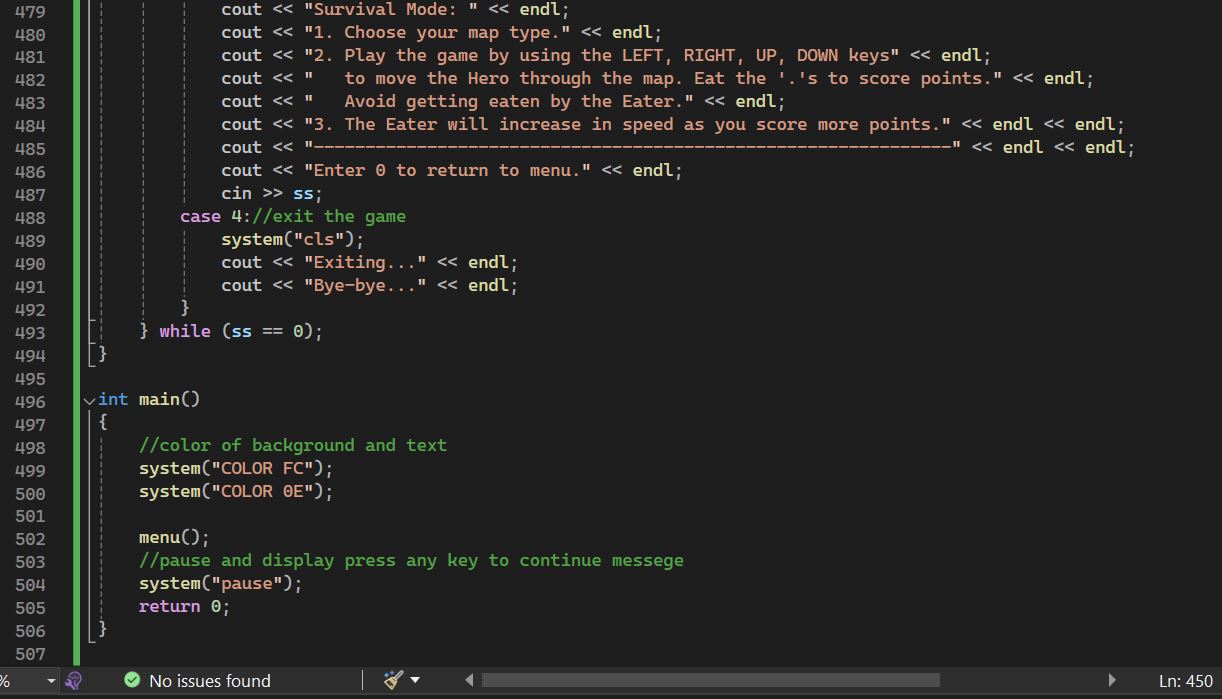




**menu()** function which displays the main menu and handles user input for game mode selection.



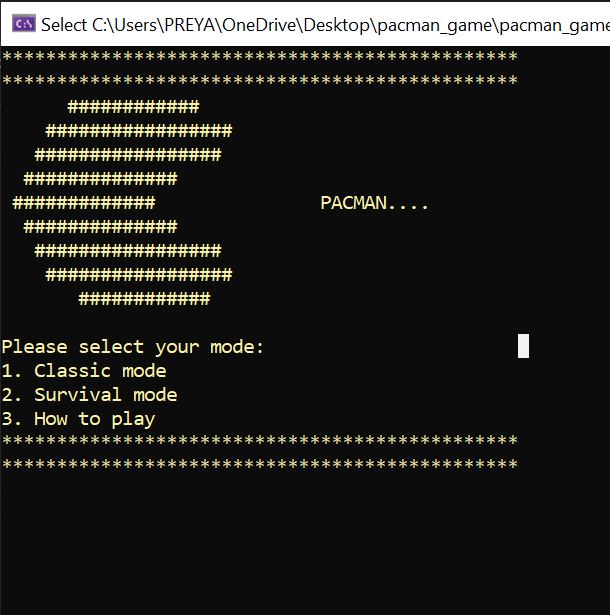




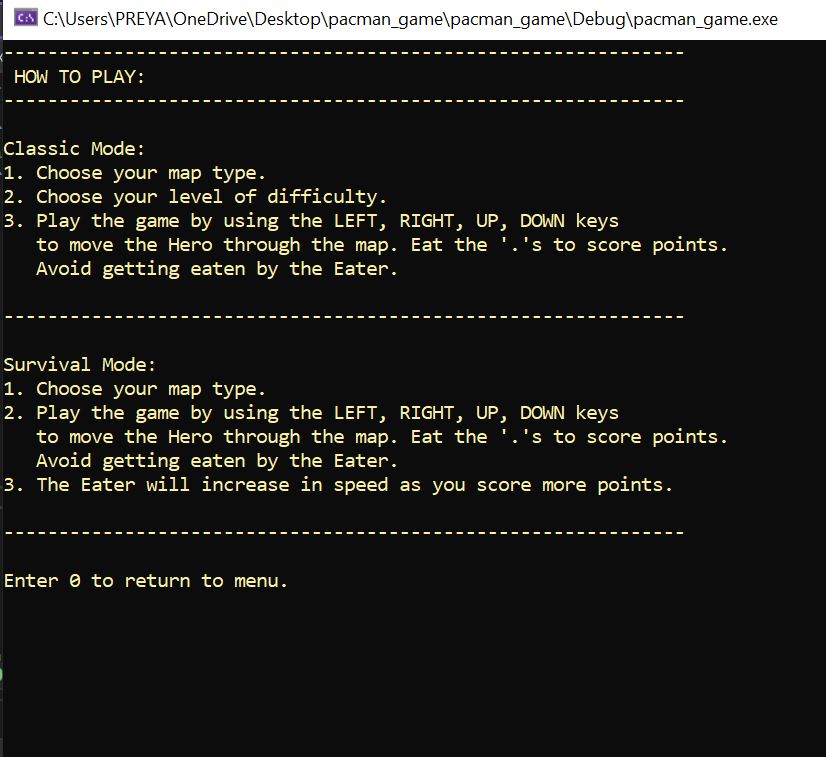
Defines the **main()** function which serves as the entry point of the program.

This function initializes the console colors, calls the **menu()** function to start the game, and pauses the program at the end.

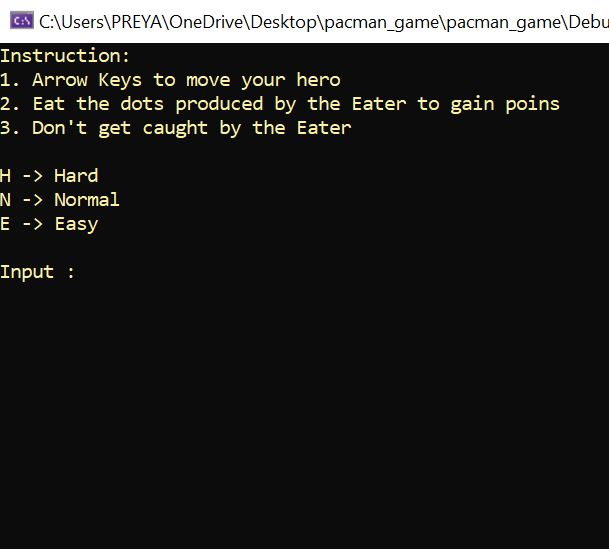
**Output**



The main Menu of PACMAN game includes modes of the game and a guide on how to play for beginners.



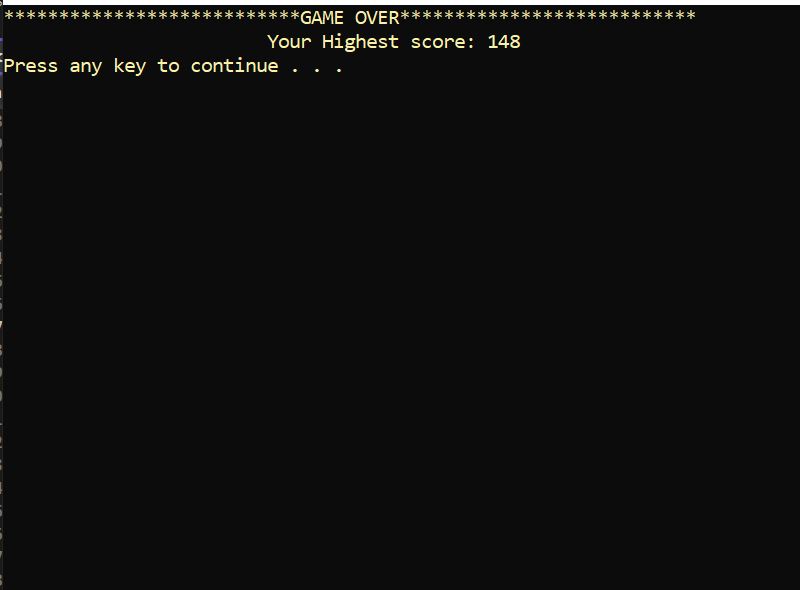
Redirects the player to instructions page when the player selects “How to play”.



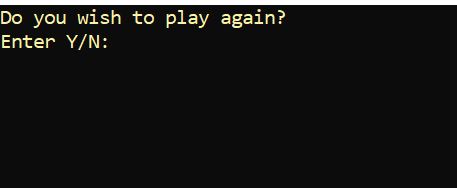
Instructions page and options to choose the difficulty level.

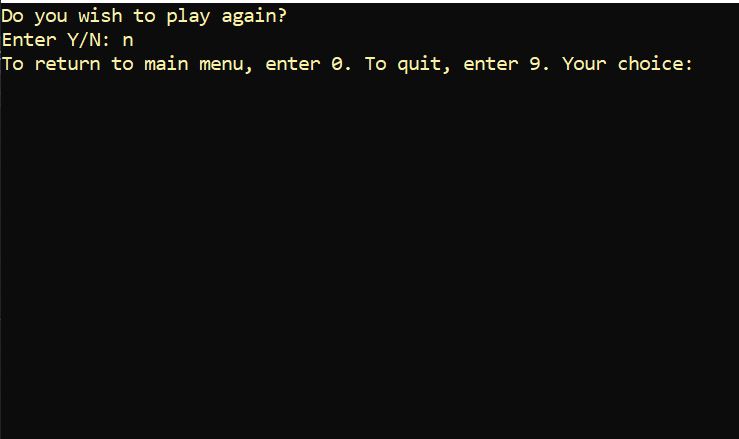


Displays map of the game including live score counting and lives remaining.

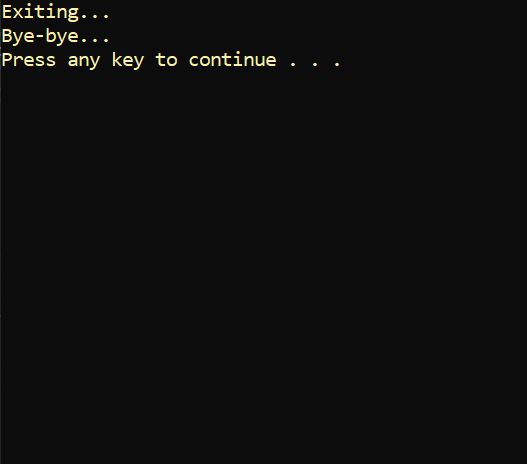


Displays Highest score after finishing the game.





Asks the player if they wish to play again and redirects them back to the main menu.



Ends the game if they choose to exit.

**Conclusion**

The project features two distinct modes: Classic Mode and Survival Mode, each offering unique challenges and gameplay mechanics. This adaptation offers simple yet engaging gameplay, ensuring hours of entertainment for players of all ages.

Throughout the project, various programming concepts and techniques were utilized. Additionally, the project demonstrates effective use of object-oriented programming principles, with entities like the Pac-Man character and the Eater represented as classes, each with their own behaviors and attributes.

Overall, the Pac-Man project not only provides an entertaining gaming experience but also serves as a practical demonstration of fundamental programming concepts and techniques. It showcases the ability to create engaging games within a console environment and lays the foundation for further exploration and expansion in game development.

With further refinement and enhancement, this project could serve as a solid foundation for aspiring game developers to build upon, exploring additional features, levels, and mechanics to create even more immersive gaming experiences.

The completion of this project marks a significant milestone in the journey of programming proficiency, demonstrating the ability to conceptualize, design, and implement complex systems within the context of a fun and engaging game.

**Citations**

* [**https://stackoverflow.com/**](https://stackoverflow.com/)
* [**https://aruneworld.com/programming-language/c/c-library/c-library-windows-h/**](https://aruneworld.com/programming-language/c/c-library/c-library-windows-h/)
* [**https://www.w3schools.com/cpp/cpp\_arrays\_multi.asp**](https://www.w3schools.com/cpp/cpp_arrays_multi.asp)
* [**https://www.delftstack.com/howto/cpp/cpp-detect-keypress/**](https://www.delftstack.com/howto/cpp/cpp-detect-keypress/)