***Bon Maharaj Institute of***

***Management & Technology***

affiliated to Agra University, Agra.

Project Report

On

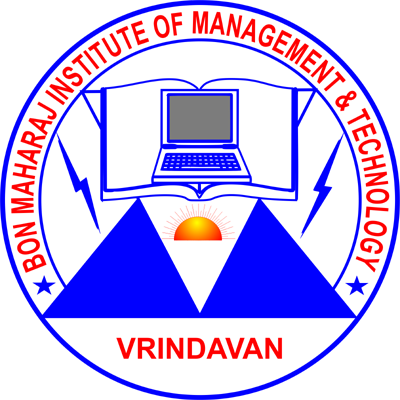
Snake Game

SUBMITTED TO

DEPARTMENT OF COMPUTER APPLICATIONS

BACHELOR OF COMPUTER APPLICATIONS (BCA)

SESSION 2021-24



By

Name : Priyanshu pal

Roll No: 2101345011021

Enrolment no: A-21001132

Submitted to:- Mr. Debashis Chakraborty

Acknowledgement

It gives me immense pleasure to present my project on Snake game. It has been a process of hard work, learning and dedication. I express my gratitude and thanks to my teachers being a source of strength, inspiration and helped me to surmount the obstacles on my way to the successful completion of this challenging work.

I express my sincere thanks and gratitude to my course coordinator Mr. Debashis Chakraborty for his throughout advice and guidance.

I express my sincere thanks and gratitude to the Director, Dr Mukesh Sharma, Head of B.C.A. Department Mr. Yogen Sharma for their problem-solving guidance and helping nature. Without their support this project could not be possible. I thank my parents who supported me, helped me for this project work and my friends for their enthusiasm and timely help during this academic year 2021-24.

Declaration

I undersigned solemnly declare that report of this project work entitled “Snake Game” is based on my own work carried out during the course of my study under the supervision of Mr. Debashis Chakraborty.

I assert that the statements made & conclusions drawn are outcome of my project work. I further declare that to the best of my knowledge and belief the report contains any entirely my original work and all ideas and references have been duly acknowledged.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Signature of the candidate)

Name: Priyanshu pal RollNo: 2101345011021

Certificate

This is to certify that the work incorporated in the project report “Snake Game” is a record of project work carried out by “Priyanshu Pal” under my guidance and supervision for the award of Degree of Bachelor of Computer Application of Bon Maharaj Institute of Management & Technology affiliated to Agra University, Agra.

To best of my knowledge and belief the project

i. Embodies the work of candidate himself.

ii.He has duly completed.

iii. Fulfills the requirement of degree.

iv. Is up to the desired standard both in respect of contents and language being referred.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Signature of Guide)

Name of Guide: Mr. Debashis Chakraborty

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Signature of the Head of Department)

Name of HOD: Mr. Yogen Sharma

Index

* INTRODUCTION---------------------------------------------------{ 6-7}
* Objectives----------------------------------------------------------{8}
* Abstract-------------------------------------------------------------{9}
* Introduction to Tool Used----------------------------------------{10-11}
* Code Overview-------------------------------------------------------{12}
* Files Included---------------------------------------------------------{13-14}
* Use Of Functions With Code----------------------------------------{15-39}
* Code Structure and Design------------------------------------------{40}
* Windows API and Sleep Function-----------------------------------{41}
* Feasibility Study-------------------------------------------------------{42-43}
* System Designing----------------------------------------------------{44}
* Testing-----------------------------------------------------------------{45}
* Outputs---------------------------------------------------------------{46-48}
* Conclusion-----------------------------------------------------------{49}
* Bibliography----------------------------------------------------------{50}

INTRODUCTION

The Project is a game written in java based on the game called 'snake' which has been around since the earliest days of home computing and has re-emerged in recent years on mobile phones.

This Project aims to bring fun and simplicity of snake game. In that project it calculate score and time. Snake is the common name for a video game concept where the player maneuvers a line which grows in length, with the line itself being a primary obstacle. The concept originated in the 1976 arcade game Blockade, and the ease of implementing Snake has led to hundreds of versions (some of which have the word snake or worm in the title) for many platforms. After a variant was preloaded on Nokia mobile phones in 1998, there was a resurgence of interest in the snake concept as it found a larger audience.

It is not the world's greatest game, but it does give you an idea of what you can achieve with a relatively simple java swing program, and perhaps the basis by which to extend the principles and create more interesting games of your own.

The snake moves to the Right, timer is set to some amount of time, and if you manage to press Up or Down arrow and then very fast Left (when the timer hasn't gone off), the snake will move backwards and finally eat himself. Most of the games doesn't allow you to change from Right to Left (or Up to Down etc.) in one step, but this is allowed. I prevent players from doing that by implementing temporary direction, that reads the current direction at the beginning of a time period (or refresh rate - it is the same here).

This game has been built using Java which is platform independent and applet based hence lightweight.

The snake game is simple game. In this game the snake is eating some foods and get its size larger in step by step and getting some score

This is small game .In this if the player has to change the direction of the snake for getting food for snake.

Objectives

This game aims to change the way people think of traditional snake game. It will offer the  experience of commercial multilayer games to the player retaining the simplicity of traditional  snake game. The major objectives of this project are:

●Create a snake game that will have all the functionality of traditional snake games.

● Introduce multilayer functionality in the game that will allow several players to play a game simultaneously. It should be able to give the experience of a real time multiplayer game to the players.

● Introduce computer controlled intelligent opponent (unique feature of this game) to make the game more challenging and interesting. The movement and action of these intelligent opponents will be controlled by computer whose aim will be to eat the food before human players capture it.

Abstract

This project aims to bring the fun and simplicity of snake game with some new features. It will include computer controlled intelligent opponents whose aim will be to challenge the human players. It will also have the multiplayer feature that will allow more than one players to play the game over a network. This project explores a new dimension in the traditional snake game to make it more interesting and challenging. The simplicity of this game makes it an ideal candidate for a minor project as we can focus on advanced topics like multiplayer functionality and implementation of computer controlled intelligent opponen

Introduction to Tool Used

Programming Language :

C is a procedural programming language initially developed by Dennis Ritchie in the year 1972 at Bell Laboratories of AT&T Labs. It was mainly developed as a system programming language to write the UNIX operating system.

The main features of the C language include:

• General Purpose and Portable

• Low-level Memory Access

• Fast Speed

• Clean Syntax

These features make the C language suitable for system programming like an operating system or compiler development.

C is a fundamental language in the field of computer science, and it is strongly associated with UNIX, as it was developed to write the UNIX operating system. If you know C, you will have no problem learning other popular programming languages such as Java, Python, C++, C#, etc, as the syntax is similar.

Here is an example of a simple C program:

A computer screen with colorful text

Description automatically generated

System Software :

Windows 11 is the brand new operating system released by Microsoft which is the successor to Windows 10. Microsoft has announced Windows 11 on June 24, 2021, and was released officially on 5th October 2021, it is considered as a turning point in the Windows franchise. A fresh new design, better performance, and new security features change the way people could interact with their devices with this new OS.

With a focus on simplicity, speed, and security, Windows 11 is designed to provide a seamless and intuitive user experience. The new operating system boasts a revamped interface, featuring a centered Start menu, improved multitasking capabilities, and a more streamlined Settings app. Additionally, Windows 11 introduces several innovative features, such as Snap Layouts, Widgets, and a revamped Microsoft Store, which are expected to enhance productivity and entertainment.

System Specification :

Processor: AMD Ryzen™ 9 6900HX Mobile Processor (8-core/16-thread, 20MB cache, up to 4.9 GHz max boost)

Graphics: The laptop comes with various graphics options, including:

• NVIDIA® GeForce RTX™ 3080 Laptop GPU with ROG Boost: 1595MHz at 150W (1545MHz Boost Clock+50MHz OC, 125W+25W Dynamic Boost) and 8GB GDDR6

Display: The laptop features a 15.6-inch display with various options, including:

• WQHD (2560 x 1440) 16:9 anti-glare display with DCI-P3:100.00%, Refresh Rate:165Hz, Response Time:3ms, IPS-level, Adaptive-Sync, MUX Switch + Optimus, and Dolby Vision

**Code Overview**

The provided code is a C program that appears to be a simple game implementation. It includes several header files from different directories, indicating a modular design. The program uses the Windows API (via windows.h) and the standard input/output library (stdio.h).

**Global Variables:**

The program defines several global variables:

• height and width: Set to 20, these variables likely represent the game board's dimensions.

• x and y: Initialized to the center of the game board, these variables seem to represent the player's position.

• fruitx and fruity: Uninitialized, these variables might be related to the game's fruit or scoring system.

• flag: Uninitialized, this variable's purpose is unclear.

• game\_end: Initialized to 0, this variable likely controls the game loop.

• score: Initialized to 0, this variable tracks the player's score.

• piece: Initialized to 0, this variable's purpose is unclear.

• tailx and taily: Arrays of size 100, these variables might be related to the player's tail or movement history.

Files Included

1. "logic/input.c"

• This file is likely responsible for handling user input or providing input-related functionality to the program.

• It is included in the "logic" directory, suggesting that input handling is a logical component of the program.

2. "intro/print.c"

• This file appears to be related to printing or outputting introductory information or messages.

• Its inclusion in the "intro" directory implies that it is responsible for initializing or setting up the program's output.

3. "logic/logic.c"

• This file is likely the core logic component of the program, containing the main algorithms, calculations, or decision-making processes.

• Its inclusion in the "logic" directory reinforces this assumption.

4. "setup/draw.c"

• This file is probably responsible for setting up or drawing graphical elements, such as windows, menus, or other visual components.

• Its inclusion in the "setup" directory suggests that it is involved in the program's initialization or setup process.

5. "setup/setup.c"

• This file appears to be a general setup or initialization file, responsible for configuring the program's environment or resources.

• Its inclusion in the "setup" directory implies that it is a crucial component of the program's startup process.

6. "exit/exit.c"

• This file is likely responsible for handling the program's exit or shutdown process, including any necessary cleanup or resource release.

• Its inclusion in the "exit" directory suggests that it is a separate component from the main program logic.

**Use Of Functions With Code**

Main Function :

The main function is the program's entry point. It:

1. Calls the print function, which is not defined in this code snippet.

2. Initializes the player's position to the center of the game board.

3. Calls the setup function, which is not defined in this code snippet.

4. Enters a game loop that continues until game\_end is set to 1.

5. Within the loop, the program:

• Calls the input function, which is not defined in this code snippet.

• Calls the draw function, which is not defined in this code snippet.

• Calls the logic function, which is not defined in this code snippet.

• Waits for 60 milliseconds using the Sleep function from the Windows API.

6. Once the game loop ends, the program calls the exit\_game function, which is not defined in this code snippet.

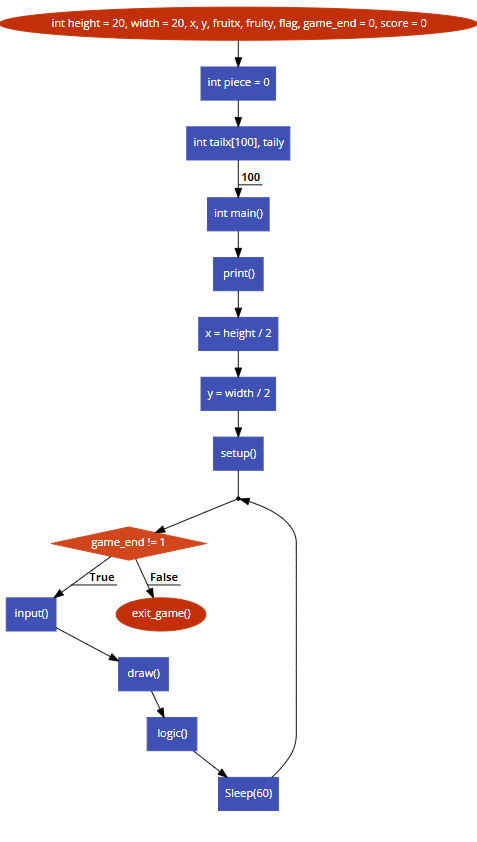
Suggestions and Improvements

• Consider using more descriptive variable names to improve code readability.

• The program's modular design is a good practice, but it would be helpful to include the contents of the included header files for a more comprehensive understanding.

• The use of global variables can make the code harder to maintain and debug. Consider encapsulating game state and logic within a struct or class.

• The program's functionality is not immediately clear due to the absence of implementation details for the print, setup, input, draw, logic, and exit\_game functions.





\*\*Note\*\*

In C language there are a simple method to link multiple file

Use header file #include”file\_name.c”, If the file is in another folder so you can Use #include”folder\_name/fime\_name”.

Another way:

1. Create your source files. Let's say you have two files, file1.c and file2.c.

2. In file1.c, you can define functions that you want to use in file2.c

**A computer screen with text

Description automatically generated**

3.In file2.c, you can include a header file that contains the function prototypes for the functions defined in file1.c. For example:

**A computer screen with text

Description automatically generated**

**A computer screen with colorful text

Description automatically generated**

4.To compile the program, you can use the gcc command with the -c option to compile each source file separately, followed by the gcc command again to link the object files together:

**A black background with white letters

Description automatically generated**

5. After compiling, you can run the program with the ./ command followed by the output file name:

****

Print():

The print.c file is a part of a simple snake game project. It contains a single function, print(), which is responsible for displaying various messages and instructions to the user.

Here's a breakdown of the print() function:

1. Welcome Message: The function starts by displaying a welcome message to the user, inviting them to press any key to continue.

2. Clearing the Screen: After the welcome message, the screen is cleared using the system("cls") command.

3. Game Instructions: The function then displays the game instructions, which include:

• How to move the snake (using W, S, A, D keys)

• How to eat food and increase the snake's length and score

• Information about the three lives provided

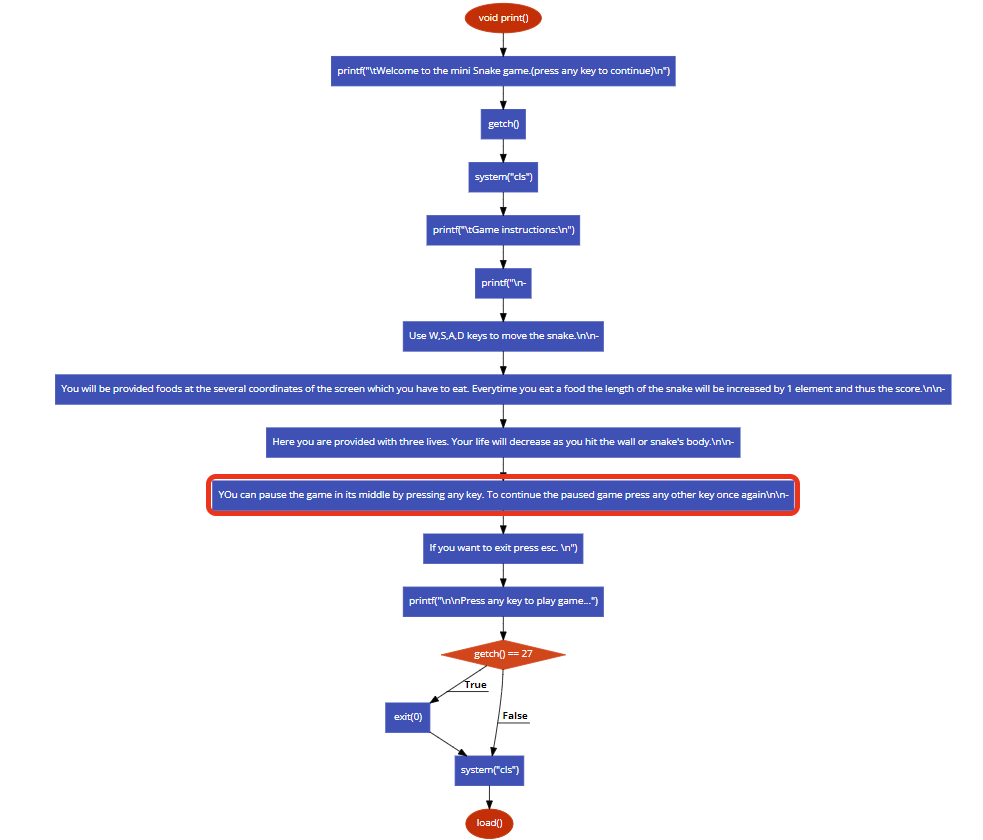
• How to pause and unpause the game

• How to exit the game (by pressing the 'esc' key)

4. Starting the Game: Finally, the function prompts the user to press any key to start the game. If the user presses the 'esc' key, the game exits.

Note: The load() function is called after the game instructions are displayed, but it is not defined in the provided context. It is likely defined in the load.c file.

In conclusion, the print.c file plays a crucial role in the snake game project by providing clear and concise instructions to the user.



A black background with text on it

Description automatically generated

Load():

The load. C file is a C program that contains two functions: gotoxy and load.

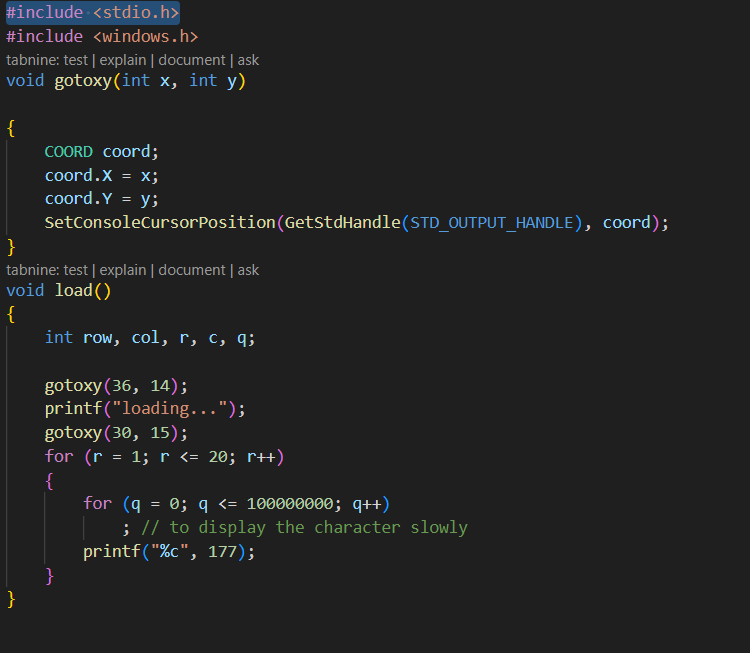
The gotoxy function is used to move the cursor to a specific position in the console window. It takes two arguments: x and y, which represent the column and row number respectively. The function sets the cursor position by using the SetConsoleCursorPosition function from the windows.h library.

The load function is used to display a loading animation on the console window. It does this by printing the block character (177 in ASCII) repeatedly in the same position, with a delay between each character to create the illusion of movement. The function uses a nested loop to print the character 20 times, with a delay of 100000000 iterations between each character.

Overall, the load. C file provides a simple way to display a loading animation in a console application. However, it is worth noting that the delay mechanism used in the load function may not be the most efficient way to create a delay, as it relies on a busy wait loop. A more efficient approach might be to use a sleep function or a timer to create the delay.

A diagram of a computer program

Description automatically generated

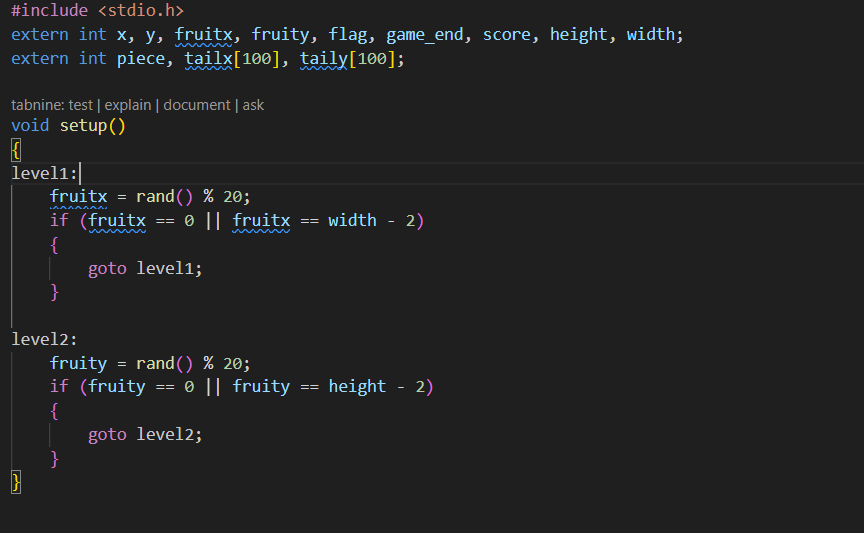


setup():

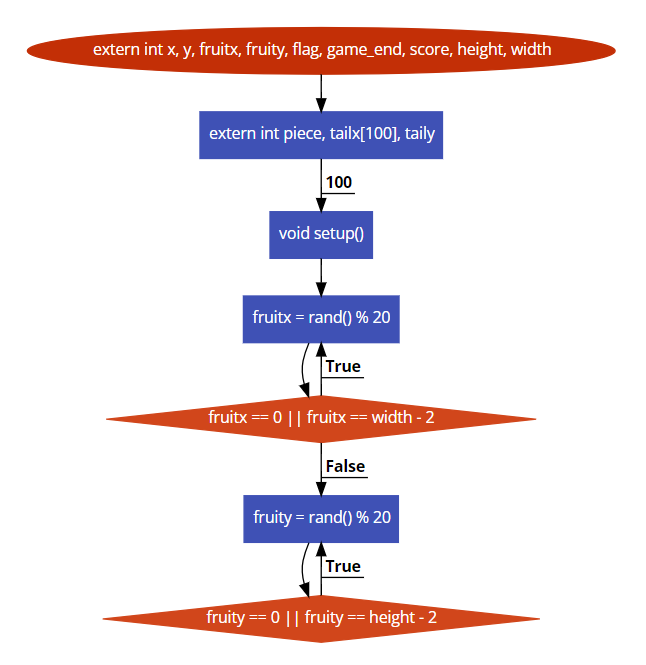
The setup. C code is a part of a larger program, possibly a game, that sets up the initial positions of the fruit and the player's piece on the game board. The code uses the rand() function to generate random positions for the fruit and the piece.

The setup() function contains two labeled sections, level1 and level2, which are used to handle cases where the randomly generated positions for the fruit and the piece are not valid. If the fruit's x-coordinate is either 0 or width - 2, the code jumps to level1 and generates a new random x-coordinate. Similarly, if the fruit's y-coordinate is either 0 or height - 2, the code jumps to level2 and generates a new random y-coordinate.

The setup() function does not return any value, and it does not take any arguments. It modifies the global variables fruitx, fruity, x, and y.



In summary, the setup.c code sets up the initial positions of the fruit and the piece on the game board. It uses the rand() function to generate random positions, and it handles cases where the generated positions are not valid. The code modifies the global variables fruitx, fruity, x, and y.



draw():

Variables Report:-

The following variables are declared as extern in draw.c:

Game Status:-

• game\_end: An integer variable indicating whether the game has ended or not.

• flag: An integer variable used as a flag for some game logic.

Game Board:-

• height: An integer variable representing the height of the game board.

• width: An integer variable representing the width of the game board.

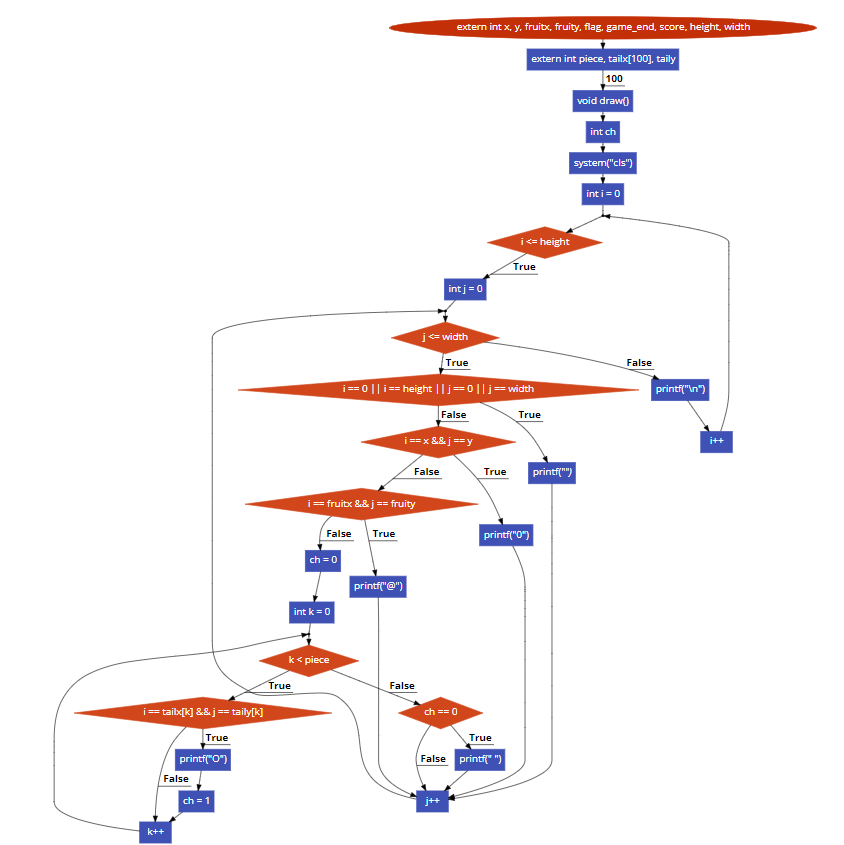
Game Pieces:-

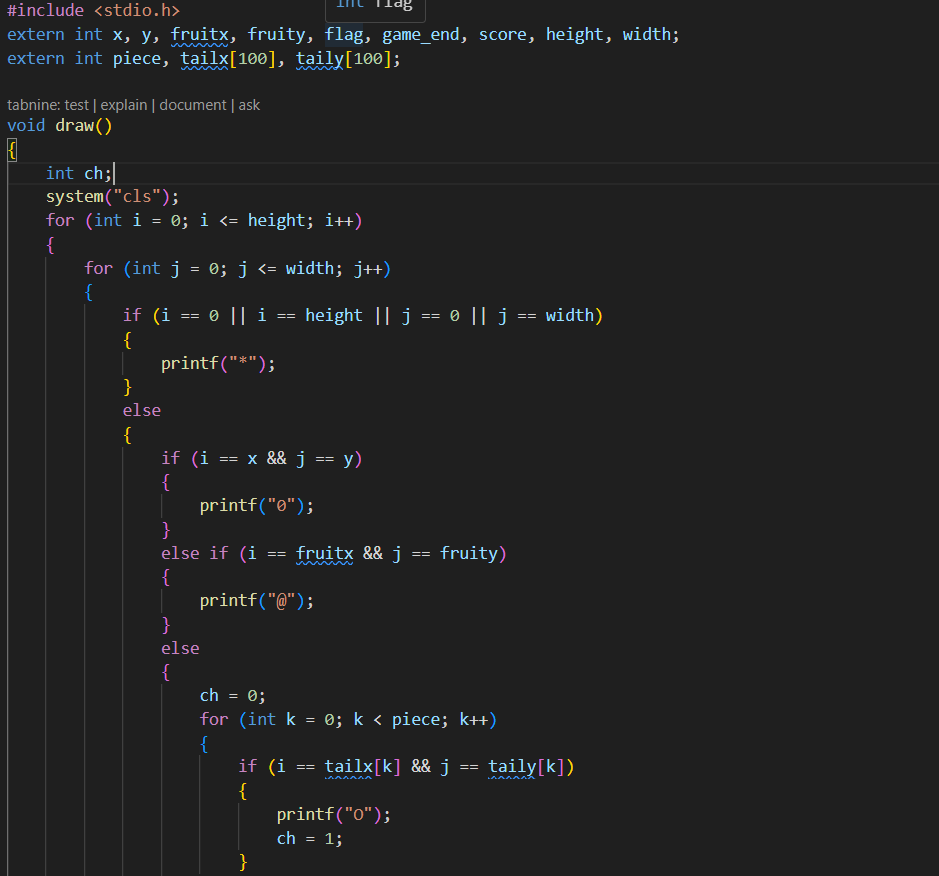
• piece: An integer variable representing the current game piece.

• x and y: Integer variables representing the coordinates of the current game piece.

• fruitx and fruity: Integer variables representing the coordinates of the fruit on the game board.

• tailx and taily: Integer arrays of size 100, representing the coordinates of the tail of the game piece.



**A computer screen with colorful text

Description automatically generated**

Input():

Game State:-

• x: The current x-coordinate of the snake's head.

• y: The current y-coordinate of the snake's head.

• fruitx: The x-coordinate of the fruit.

• fruity: The y-coordinate of the fruit.

• flag: The current direction of the snake (1: up, 2: down, 3: left, 4: right).

• game\_end: A flag indicating whether the game has ended (1: yes, 0: no).

• score: The current score of the game.

• height: The height of the game board.

• width: The width of the game board.

Tail Coordinates:-

• piece: The number of pieces in the snake's tail.

• tailx[i]: The x-coordinate of the i-th piece of the snake's tail.

• taily[i]: The y-coordinate of the i-th piece of the snake's tail.

Input Function:

The input() function is used to handle user input. It checks if a key has been pressed using kbhit() and then uses getch() to get the character of the pressed key. Based on the key pressed, it sets the flag variable to change the direction of the snake.

A diagram of a algorithm

Description automatically generated

A screen shot of a computer program

Description automatically generated

logic():

Global Variables:-

The following variables are declared as extern, indicating that they are defined in another file and are being referenced in logic.c:

1. x and y: These variables are likely used to store coordinates or positions in a 2D space.

2. fruitx and fruity: These variables might be related to the position of a fruit or an object in the game.

3. flag: This variable could be a boolean flag used to indicate a specific state or condition in the game.

4. game\_end: This variable is likely used to determine whether the game has ended or not.

5. score: This variable stores the player's score in the game.

6. height and width: These variables might represent the dimensions of the game board or screen.

Array Variables

The following array variables are declared:

1. tailx and taily: These arrays, each with a size of 100, are likely used to store the coordinates of a tail or a trail in the game. The indices of these arrays might correspond to different points in the tail.

Piece Variable:-

piece: This variable is likely used to store information about a game piece, such as its type, position, or state.

Overall, the variables declared in logic.c suggest that this file is part of a game implementation, possibly a snake game or a similar type of game.

Logic explain :-

1. x and y: These variables are likely used to store the current position of the player's character or game piece on the screen. They might be updated as the player moves around the game board or screen.

2. fruitx and fruity: These variables could represent the position of a fruit or a target object that the player's character needs to collect or interact with. When the player's character reaches the position of the fruit, the game might update the position of the fruit to a new random location.

3. flag: This variable could be used as a boolean flag to indicate various game states, such as whether the game is paused, whether the player has won or lost, or whether a specific event has occurred.

4. game\_end: This variable is likely used to determine whether the game has ended or not. When the game ends, the value of game\_end might be set to 1 or another non-zero value, indicating that the game loop should terminate.

5. score: This variable stores the player's score in the game. The score might increase when the player collects fruits or achieves certain objectives.

6. height and width: These variables might represent the dimensions of the game board or screen. They could be used to ensure that the game objects stay within the boundaries of the screen or to calculate the positions of game objects relative to the screen size.

7. tailx and taily: These arrays are likely used to store the coordinates of the tail or trail of the player's character. As the character moves around the screen, the tail might follow behind, with the positions of the tail segments stored in these arrays.

8. piece: This variable is likely used to store information about the player's character or game piece, such as its type, position, or state. The value of piece might determine the behavior of the character or the rules of the game.

In summary, the variables declared in logic.c are used to manage various aspects of the game, such as the positions of game objects, the game state, and the player's score. These variables are likely used in conjunction with other game logic and rendering code to create a complete game implementation

A diagram of a computer program

Description automatically generated

A screen shot of a computer program

Description automatically generated

**A screen shot of a computer program

Description automatically generated**

Exit\_game():

The exit. C file contains the implementation of the exit\_game() function, which is responsible for handling the game over condition in a snake game. The function checks for various conditions that result in the game ending, such as the snake colliding with the game boundaries or its own tail. If any of these conditions are met, the function sets the game\_end variable to 1, clears the screen, displays a "Game Over" message, and prompts the user to press any key to quit the game. The function then calls the record() function, which is presumably responsible for updating the game's high score record.

Here is a breakdown of the exit\_game() function:

• The function first initializes a variable check to 0, which will be used to check if the snake has collided with its own tail.

• The function then enters a loop that iterates over the tailx and taily arrays, which store the coordinates of the snake's tail segments. For each tail segment, the function checks if the snake's head (represented by the x and y variables) is colliding with the tail segment. If a collision is detected, the check variable is incremented.

• If the loop reaches the end of the tail segments or if a collision is detected, the function breaks out of the loop.

• The function then checks if the snake's head is within the game boundaries. If the head is outside the boundaries or if a collision with the tail was detected, the function sets the game\_end variable to 1, clears the screen, displays a "Game Over" message, and prompts the user to press any key to quit the game. The function then calls the record() function to update the high score record.

Overall, the exit\_game() function is an important part of the snake game, as it handles the game over condition and ensures that the game ends gracefully when the snake collides with the game boundaries or its own tail.

A screenshot of a computer

Description automatically generated

**A screen shot of a computer program

Description automatically generated**

Record():

Overview:-

The provided code is a C program that records the player's name, the date and time of the game, and the score achieved by the player. It stores this information in a file named "record.txt". The program also allows the user to view past records if they wish.

Functions:-

The code defines one function, record(), which is responsible for recording the player's information and displaying past records.

record():-

This function performs the following tasks:

1. Clears the screen using system("cls").

2. Prompts the user to enter their name and stores it in the plname character array.

3. Converts the first letter of the name and the first letter after each space to uppercase.

4. Gets the current time using time() and converts it to a string using ctime().

5. Writes the player's name, the current date and time, and the score to "record.txt" using fprintf().

6. Writes a line of underscores to "record.txt" for formatting purposes.

7. Closes the file using fclose().

8. Prompts the user to press 'y' if they want to see past records.

9. If the user presses 'y', the function opens "record.txt" in read mode and displays its contents using getc() and putchar().

10. Closes the file using fclose().

Variables:-

The code uses several external variables, which are not defined in this code snippet but are assumed to be defined elsewhere. These variables include x, y, fruitx, fruity, flag, game\_end, score, height, and width. The code also defines several local variables, including plname, nplname, cha, c, i, j, and px.

File Operations:-

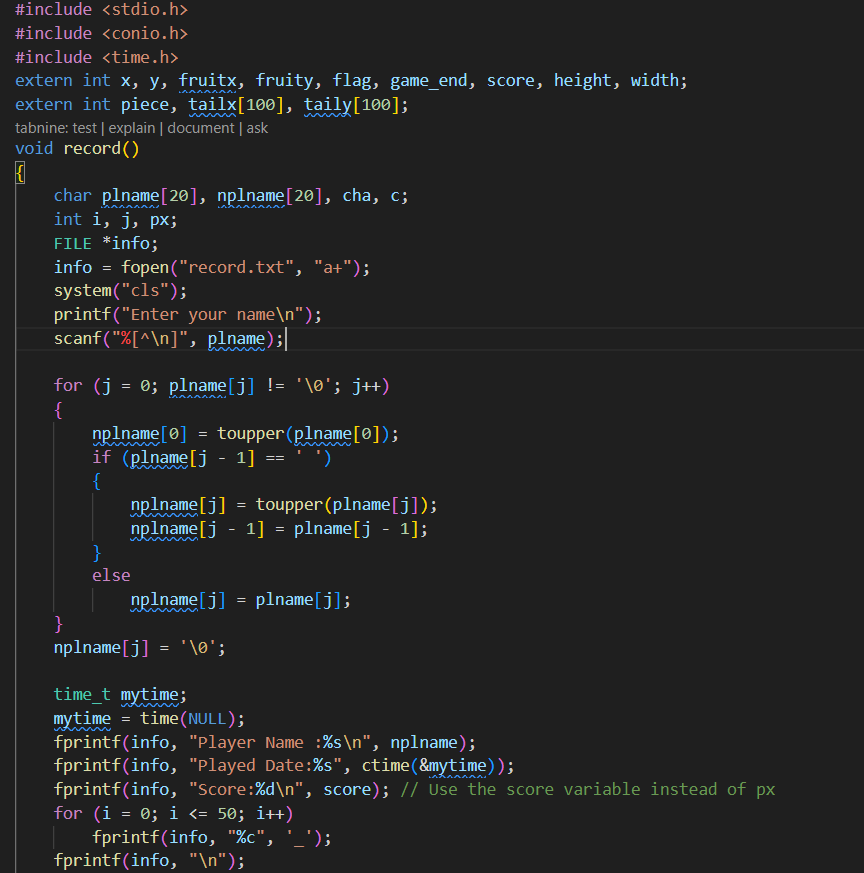
The code uses the fopen(), fprintf(), fclose(), and getc() functions to perform file operations. The fopen() function is used to open "record.txt" in append mode ("a+") or read mode ("r"). The fprintf() function is used to write formatted data to "record.txt". The fclose() function is used to close the file after it is no longer needed. The getc() function is used to read a character from "record.txt" and display it using putchar().

Potential Issues:-

The code does not check if "record.txt" exists before attempting to open it in append mode. If the file does not exist, the program may crash or produce unexpected results. It is recommended to check if the file exists using fopen() in read mode and then close it before opening it in append mode.

Additionally, the code uses system("cls") to clear the screen, which is specific to Windows. If the code is intended to be portable, it is recommended to use a platform-independent method to clear the screen, such as using the ncurses library.

Finally, the code uses a fixed-size buffer (plname and nplname) to store the player's name. If the user enters a name that is too long, it may overflow the buffer and cause a buffer overflow vulnerability. It is recommended to use a dynamically allocated buffer or a safer input function, such as fgets(), to prevent buffer overflows.

**A computer screen shot of a program

Description automatically generated**

Code Structure and Design

The code snippet provided has a modular design, with separate header files for different aspects of the game:

• logic/input.c: likely handles user input

• intro/print.c: might be responsible for printing the game's introduction or instructions

• logic/logic.c: could contain the game's core logic, such as updating the game state

• setup/draw.c: probably handles drawing the game board or graphics

• setup/setup.c: might be responsible for initializing the game state or setting up the game environment

• exit/exit.c: likely handles the game's exit or cleanup

This modular design is a good practice, as it allows for easier maintenance and modification of individual components without affecting the entire codebase. However, it would be beneficial to have a clearer understanding of how these modules interact and how the game state is shared between them.

Game Loop and Functionality

The game loop is the core of the game, and it's responsible for updating the game state and rendering the game board. The loop consists of the following steps:

1. input(): Handles user input, such as keyboard or mouse events.

2. draw(): Updates the game board or graphics based on the current game state.

3. logic(): Applies the game's rules and updates the game state accordingly.

4. Sleep(60): Introduces a delay of 60 milliseconds to control the game's speed.

The exit\_game() function is called when the game loop ends, which might be triggered by the game\_end variable being set to 1.

Windows API and Sleep Function

The Sleep function is part of the Windows API, and it's used to introduce a delay in the game loop. This delay is necessary to control the game's speed and prevent it from running too fast. The Sleep function takes a single argument, which is the duration of the delay in milliseconds.

Variable Naming and Readability

The code snippet uses relatively short and cryptic variable names, such as x, y, fruitx, and fruity. While these names might be clear to the original author, they can make the code harder to understand for others.

Using more descriptive variable names, such as playerX, playerY, fruitPositionX, and fruitPositionY, can greatly improve code readability and maintainability.

Code Improvement Suggestions

Here are some additional suggestions for improving the code:

• Consider using a struct or class to encapsulate the game state, rather than relying on global variables.

• Use const correctness to indicate which variables should not be modified.

• Add comments to explain the purpose and behavior of each function.

• Consider using a more robust input handling mechanism, such as event-driven input handling.

• Use a consistent coding style throughout the codebase.

Feasibility Study

Feasibility study determines exactly what the proposed system is to by defining its expected performance. A feasibility study is carried out to select the best system that meets performance requirements. Feasibility studies are undertaken within tight time constraint. The contents and recommendation of such a study will be used as a sound basis for deciding whether to proceeds postpone or cancel the project.

**Type of Feasibility Study:**

Organizational/Behavioural Feasibility

The organizational or behavioural feasibility as the name indicates, determines the feasibility of the system in terms of the organization and behavioural of the employees. The behavioural feasibility reflects the behaviour of the employees of on organization. Behavioural feasibility leads to the smooth functioning and implementation of the organizational plan.

Economic Feasibility

The economic feasibility study deals with the economy of the system project. The cost and returns are evaluated and therefore, it is determined whether the returns justify the investment, which was planned, in the system project or not.

Technical Feasibility

The major concern of technical feasibility is to observe whether the reliable hardware and software of the organization meets the needs of the proposed system or not. It also determines the requirements that can be developed in the required time.

Operational Feasibility

The operational feasibility is responsible for the operations of management, employees, customers and suppliers involved in a project it determines.5.

Legal Feasibility

Legal Feasibility of the system refers to the viability of the system. In other words, legal feasibility verifies whether the system abides by all the laws and regulations or not.6.

Social Feasibility

The fact that social feasibility is a determination of whether a proposed project will be acceptable to the people or not.

Schedule Feasibility

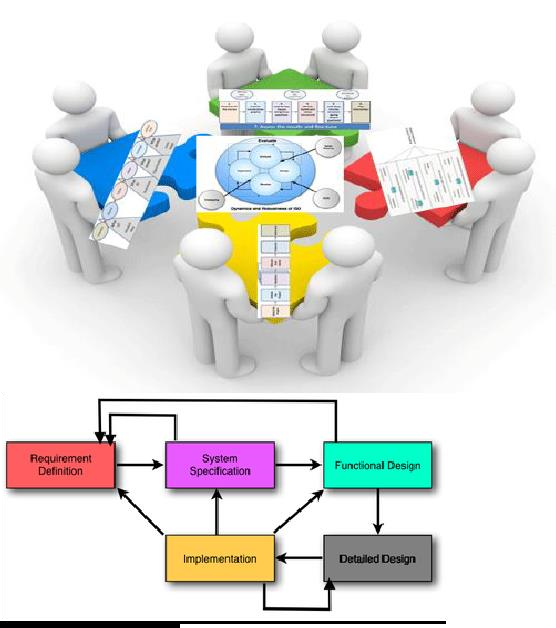
The scheduled feasibility of the system evaluates whether the system finishes its task within the provided time of develop mentor not. It is recommended for a system to complete all the tasks will before the requirements.

Management Feasibility

It is a determination of whether a proposed project will be acceptable to the management or not. If the management does not accept a project or gives a negligible support to it, the analyst will tend to view the project as a non-feasible one.

System Designing

System design is the first step in the development phase for any project or system. It may be defined as "The process of applying various techniques and principles for the purpose of defining a device, a process, or a system insufficient detail to permit its physical realization.



Testing

Testing is a critical aspect of Software Quality Assurance and represents the ultimate review of specification, design and coding. Testing is a process of executing a program with the intent of finding an error. A good test is one that has a probability of finding an as yet undiscovered error. The purpose of testing is to identify and correct bugs in the developed system. Nothing is complete without testing. Testing is the vital to the success of the system.

The testing is the process to check the program and find the errors. There are basic three type of testing: -1) Code testing2) Specification testing3) Unit testing

Code testing **:-**

The code testing strategy examines the login of the program. To follow this testing method, the analyst develops test cases that result in executing every instruction in the program. **Specification Testing**

The specification stating with the program should do and how it should performunder various condition by Examining the result the analyst can deter mine Whether the program according to its.

Unit Testing **:-**

In this testing each and every program is tested individually using record to see whether that program produced satisfactory outputs and validation also.

Outputs

A black rectangle with white text

Description automatically generatedA computer screen with text

Description automatically generatedA black background with white text

Description automatically generated

A black screen with white dots

Description automatically generated



A black background with white text

Description automatically generated

A screenshot of a computer

Description automatically generated

Conclusion

The included C files appear to be organized into distinct directories, each responsible for a specific aspect of the program's functionality. The files are likely responsible for handling input, printing introductory information, performing core logic, setting up graphical elements, initializing the program, and handling exit processes. Further analysis or context would be required to provide more detailed insights into the specific roles of each file.

Bibliography

Record():-

* <https://www.youtube.com/>
* <https://chat.openai.com/>

Logic():-

* <https://www.youtube.com/>
* <https://chat.openai.com/>
* https://github.com/