# Snake Game in C

**Rafi Saad (2422435642)**

**Rashad Rafid Alamgir (2421190642)**

**Riyadh Ahmed Desh (2423292642)**

**Yasir Obayed (2423524642)**

**Abstract-** As part of our CSE115 Programming language course, we were instructed to make a snake game using the C programming language. C is a basic mid-level programming language with low level access to the memory giving it the ability to manipulate the memory. Snake game is an arcade game that rose to popularity in the late 90s. The game is about a snake eating fruit and growing in length; the objective of the game is to eat as much fruit as possible without colliding with the boundaries or eating/colliding with itself. Building the game using the C language will provide us with a unique opportunity to understand the c language, as well as develop a sound understanding of how memory works.

**Keywords**- C, programming language, snake game

## Introduction

Programming languages are simply languages that are used to provide command or any task to the computer for it to complete the task. There are many factors by which languages are categorized. C is a programming language, born in the lab of the then Bell Lab in early 1970s. One of the key aspects of this language is its easy to learn syntaxes, its simplicity, and yet low level access making memory manipulation possible. It is also a pre-compiled language, which means pre-compiled into a binary format before executing the code, which generally results in faster execution speed at the expense of increased occupied memory. Its syntaxes are easy to learn, and using pointers, memory manipulation is possible. The best way of understanding how a programming language works is to write code using the language. The snake game provides us with an outstanding opportunity for deeply understanding the syntax and its execution.

## History

Created by Gremlin Industries, the snake game is a popular game . Initially released as Blockade , it quickly started catching attention of others. Even after Gremlin Industries went bankrupt, ceasing their operation in late 80s, the game still kept rising in popularity and soon by the late 90s, it had found its way into everyone's pockets. It was also shipped into the legendary Nokia 6110 and other phones of that time. Still, countless people around the world enjoy this game very much.

## Making the Game

### Execution-

To create a Snake game that allows users to control the movement of a snake on a screen, to get points for eating food and avoiding running

into the walls or the growing tail of the snake itself. In this problem, we want to write a game where a graphical representation of a snake moves across the screen. When it encounters a piece of food, the snake grows longer and we gain a point. If it hits the wall we die.

To write this program we are going to need:

1. A way of representing the sake of representing the food.

2. A way to display the score,

3. A  way for our instructions to reach the snake

# Snake Game in C

4. A way to indicate when we've run into the boundary or the snake's body itself and the game reached the end state.

The total code consists of 195 lines. We used 5 header files in the game- conio.h, ctype.h, stdio.h, stdlib.h, windows.h . In order to implement the logic and the mechanics of the game, 4 functions were prototyped in total-

1. **Setup-** Determines the initial state before the game begins.
2. **Draw-** Uses characters to graphically represent the snake, the fruit and its boundary as well as prints the score and instructions of playing the game.
3. **Input-** Takes input from the user.
4. **Logic-** The total mechanics for the game.

It took us almost a week to fully understand and develop and outline for the code. Coding took us around 4-5 days. In spite of the fact that the code would need far less time to be written and compiled, we took more time than needed in order to reduce the time needed to debug the final code.

## Problems faced during the coding process-

Since the code was simple, we didn't ran into any major problems. However, we faced dilemma while executing some functions.

As we was coding, we found out that the standard input output header file was not working as intended. It was a compiler problem of course, but we found no other choice from our end than to include windows.h header file into it. As a precompiled language C is, inclusion of this very library means that the code will need more time to execute and run than it is needed. However, we can choose not to use windows.h anyways, but it's there as a backup in case any of the header files malfunction.

The second problem we fell into was how to take input from the user. Using scanf() function from the standard library header files meant that we would need to keep flushing the buffer so that the program kept taking input and giving proper output. This is rather a complicated thing to implement and also would need far more lines of code. That is why we chose to use kbhit() function of the conio.h header file in order to take inputs and showing output on the screen in real time. W,A,S,D keys were selected for controlling the snakes movement. To avoid confusion, we decided to use tolower() function so that, we don't face any issue taking inputs even when the CapsLK is accidentally turned on.

The last problem we fell into was, we didn't know how to control the speed of the snake moving to and fro inside the boundaries. After much researching, we decided to use the sleep() function to control the iteration of the loops inside the functions.

## Conclusion

The completion of the project has provided us with a better understanding of how C works at a lower level, and how to effectively take input in case of events where using one function many times can cause issues with buffer overflowing and returning garbage output.