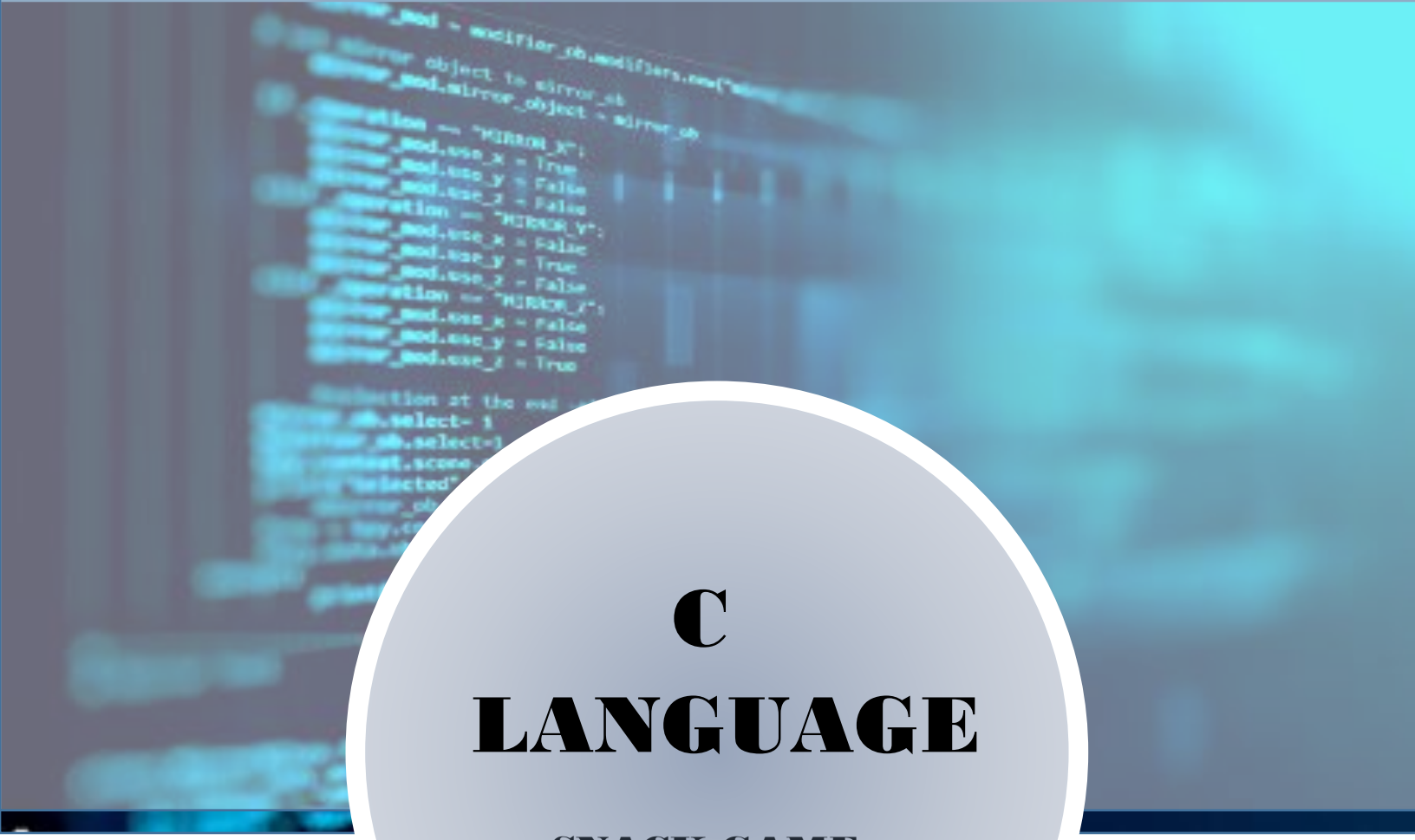

C LANGUAGE

SNACK GAME

CREATED BY: RIDA
ASSIGNED BY: SIR RIAZ ALI

DATE: 21-12-2023 **BATCH: 07-09**

Table of Contents:

C LANGUAGE

SNAKE GAME:

INTRO:

The Snake game in C is often a replication of the classic Snake game that gained popularity on early mobile phones, particularly Nokia devices. In this game, the player controls a snake that moves around the screen, eating food to grow longer while avoiding collisions with itself and the boundaries of the playing area.

C PROGRAM:

The task is to implement a basic Snake Game**.** Below given some functionalities of this game:

- The head of snake is represented with an **'▶ '** (arrow) and body with an **'o'** (lowercase 'O') symbol.
- The fruit is represented with an **'¤'** (Scarab '¤') symbol.
- The snake can move in any direction according to the user with the help of the keyboard arrow keys.
- When the snake eats a fruit the score will increase by 1 points.
- The fruit will generate automatically within the boundaries.
- Whenever the snake will touch the boundary or its body the game is over.

1. **Steps to create this game:**
    - There are five user-defined functions.
    - Set boundary for playing games.
    - Fruits are randomly generated.
    - The score increases every time the snake eats a fruit.
2. **Libraries Used to create this game:**
    - ***<stdio.h>:*** Standard Input/Output functions like printf.
    - ***<conio.h>:*** Console Input/Output functions, mainly used for getch() and kbhit().
    - ***<time.h>:*** Provides functions to work with time, used for the delay function.
    - ***<windows.h>:*** Provides functions for interacting with Windows OS, used for console manipulation.
    - ***<stdlib.h>:*** Declares various utility functions for type conversions, memory allocation, algorithms, and other similar use cases.

3. User Defined Functions Used to create this game:

The user-defined functions created in this program are given below:

- ***void gotoxy(int x, int y):*** To declare coordinates x&y. COORD is a structure to hold screen COORDinates X and Y.
- ***void boundary():*** To create a boundary for a game.
- ***void move (int *x, int *y, char *l, char c, int len):*** This function will set the movement of snake.
- ***int check (int px, int py, int *x, int *y, int len):*** We create this function to check the status of snake.
- ***void over (int x, int y, int len*)**: To declare the game is over.

4. Data types Used to create this program:

The data types used in this program are:

- ***Integer:*** (int) used in a type declaration to give a variable an integer type.
- ***Character:*** (char) It stores a single character and requires a single byte of memory in almost all compilers.
- ***Coordinates:*** (COORD) is a structure to hold screen COORDinates X and Y.

5. Function Used to create this program:

- ***SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE),p):***
  SetConsoleCursor set cursor to specified place in console that depends on cord.x and cord.y.
  GetstdHandle() gets handle for handle that handle printing characters into console. The second parameters is coordinates of the new position of the cursor.
  If you have declared this function, you can normally use gotoxy(x,y) and It would be work fine.
- ***Printf():*** C's printf function prints values to the terminal.
- ***User Defined Function:*** A user-defined function is a type of function in C language that is defined by the user himself to perform some specific task.

6. Loops Used to create this program:

- ***For loop:*** The for loop in C language is used to iterate the statements or a part of the program several times.
- ***While loop:*** The while loop in C is used to evaluate a test condition and iterate over the loop body until the condition returns True.
- ***Do while loop:*** Using the do-while loop, we can repeat the execution of several parts of the statements. The do-while loop is mainly used in the case where we need to execute the loop at least once

7. **Conditional statements Used to create this program:**

   - ***If else:*** The if-else statement in C is used to perform the operations based on some specific condition.
   - ***Switch case:*** The switch case statement is used to match the value of choice with different cases (1, 2, 3) and execute the corresponding code block.

8. **Main function:** The main function serves as the starting point for program execution.

9. **Srand():** The srand() function sets the starting point for producing a series of pseudo-random integers.

10. **clock():** The clock() function return processor runtime in millisecond.

11. **Kbhit():** kbhit() is present in conio. h and used to determine if a key has been pressed or not.

12. **Getch():** The getch function is a way to get a single character from the user. It can be used in C and C++ programs to get input from the keyboard.

13. **System("cls"):** Cls() function is used to clear the console screen like clrscr(). Where system() is a library function available inside stdlib.h [standard library library] header file.

14. **Return function:** A return statement ends the execution of a function, and returns control to the calling function.

15. **Malloc function:** Means you are allocating space off the heap to store an int. You are reserving as many bytes as an int requires.

16. **Realloc function:** 'realloc' is a function in C that changes the size of a previously allocated block of memory.

17. **Srand function:** In this function we give time(null) as a seed to get random seed or numbers. Because of this game will be different every time.

18. **SMALL DESCRIPTION OF SOURCE CODE:**
    For snake game in C Language.

    - ***WELCOME SCREEN:*** First, we must use **"U.D.F"** to draw a **"boundary"**. We created the boundary by utilizing **"horizontal and vertical bars"** and various **"angular corners using ASCII code"** in **"for loop"**. We must have to declare the **"gotoxy"** (U.D.F) function prior to the left hand boundary using the **"COORD structure".** Next, we just call the boundary function in the **"main function"** and use **"printf"** to output the game rules. However, in order to print the game rules inside the boundary, we need to declare gotoxy(1,1). We were able to get a clear screen for the game by employing **"system ("cls")".**

- ***Game screen:*** The border function is first called in **"int main()".** Next, we invoke the **"srand function"** for randomness. Next, we declare variables in **"int, char, and clock_t**". The purpose of **"clock_t"** variable is to store processor time as the number of CPU cycles that have elapsed since the process began. Then we declare the value to the variable by using the **"malloc function"**, which allocates heap space for an int's storage. As many bytes as an int requires are being reserved by you. Next, we announce first place of fruit and snake. The **"nested while loop"** for the snake game is then started. Next, we build a do while loop inside a while loop to go in the same way continuously until the key is pushed and to define a condition in the event that the back key is pressed. A U.D.F called **"move()"** is defined for the snake's movement. Next, we employ **"switch case"**, where we establish various conditions for directions and **"game over"** (we define a U.D.F named **"over"** for game over). Subsequently, we need to establish a while loop so that, as the snake moves, we can check to see if it has reached its destination if it's add another "o" to its body. For increment in body we need to know its last location in order to add another "o" to it, therefore we utilized the **"realloc function"** to find that information.

  We once more establish a random position for the target or fruit in the do while loop for the new position, shut the nested while loop, and **"return 0;"** at the conclusion of the program.

19. CONCLUSION:

To conclude, developing a Snake game in C is a valuable exercise that reinforces core programming principles such as control flow, data structures, and user input handling. This project not only offers a hands-on experience in coding but also lays the groundwork for more advanced game development concepts. It provides a tangible way to apply theoretical knowledge and encourages problem-solving skills. Overall, creating a Snake game in C is a gratifying journey for individuals looking to strengthen their programming abilities.

REFERENCE:

I copied this snake game source code from a youtube channel named **"Real Programming" [1].** Full source code is also available in video description as a google drive link [2].