A hexagon with a white circle and white text

Description automatically generatedA black rectangle with a black background

Description automatically generated



***Submitted To***: Dr. Salman Raza

***Group Members***

**1.** Muhammad Zohaib Warraich.

**2.** Abu Obaida.

**3.** Hanzala Zahid.

**4.** Nimra Javed.

***Project:*** Snake Game

**Programming Fundamentals**

**HEADER FILES**

1. **Iostream file:**

We used <iostream> file to perfume any input and output operations in program. Without an <iostream> header file, we cannot take input from the user or print any output.

1. **Ctime file:**

ctime stands for status change time. This timestamp tells you when was the last time the property and metadata of the file were changed.

1. **Unistd.h file:**

We used <unistd.h> file for using the function that controls the speed of snake.

**MAIN FUNCTION ()**

1. **Comment 1:**

Used to generate random numbers for fruit position.

1. **Comment 2:**

Used to ask the name of player.

1. **Comment 3:**

Used to clear the terminal/console.

1. **Comment 4:**

Game setup function will be called and setup the game including set height, width, snake’s position in the beginning of game, fruit’s location.

**GAME SETUP FUNCTION ()**

**DECLARATION OF VARIABLE**

const int height = 20  
 const int width = 20;   
 int snake\_x = width / 2;   
 int snake\_y = height / 2;   
 int fx = rand() % width;   
 int fy = rand() % height;   
 bool gameOver = false;   
 int score = 0;  
 int tailLength = 1;   
 int \*pTailX = new int[100];   
 int \*pTailY = new int[100];  
 int store\_snake\_x = 0;   
 int store\_snake\_y = 0;

1. **Comment 1:**

Declared a constant int variable for setting the height of box.

1. **Comment 2:**

Declared a constant int variable for setting the width of box.

1. **Comment 3:**

Declared an int variable for snake’s position on x-axis when the game begins.

1. **Comment 4:**

Declared an int variable for snake’s position on y-axis when the game begins.

1. **Comment 5:**

Declared an int variable for fruit’s position on x-axis when the game begins.

1. **Comment 6:**

Declared an int variable for fruit’s position on y-axis when the game begins.

1. **Comment 7:**

Declared a bool variable to check whether the game is over or not.

1. **Comment 8:**

Declared an int variable to count the score of players.

1. **Comment 9:**

Declared an int variable to count the tail of snake as it eats the fruit its tail length will increase.

1. **Comment 10:**

Declared a dynamical int array for snake’s tail on x-axis. As snake will eat the fruit its tail will increase, and its tail’s position will be stored in this array. We used dynamic integer because the size of tail we provided in form of array is 100 which is very large size. So, for the program’s safety that it won’t crash the program. We used dynamic array.

1. **Comment 11:**

Declared a dynamical int array for snake’s tail on y-axis. As snake will eat the fruit its tail will increase, and its tail’s position will be stored in this array.

**IMPORTANT NOTE**

We used dynamic integer because the size of tail we provided in form of array is 100 which is very large size. So, for the program’s safety that it won’t crash the program. We used dynamic array.

1. **Comment 12:**

Declared an int variable to store the current position of snake’s head on x-axis at program’s running time.

1. **Comment 13:**

Declared an int variable to store the current position of snake’s head on y-axis at program’s running time.

**STARTING THE GAME**

instructions();  
 char input;   
 cout << "\nDo you want to start the game? (Y/N): ";  
 cin >> input;  
 if (input == 'y' || input == 'Y')   
 {  
 while (!gameOver)   
 {  
 game\_starter(height, width, snake\_x, snake\_y, fx, fy, score, tailLength, pTailX, pTailY);   
 gettingInput(snake\_x, snake\_y, gameOver, store\_snake\_x, store\_snake\_y);   
 changing\_position(tailLength, pTailX, pTailY, store\_snake\_x, store\_snake\_y);   
 gameOver = checkGameOver(height, width, snake\_x, snake\_y);   
 updation(height, width, snake\_x, snake\_y, fx, fy, score, tailLength);   
 usleep(100010);   
 }  
 cout << "\nThankYou for Playing the Game";   
 }  
 else   
 {  
 cout << "GOODBYE.";   
 }  
 delete[] pTailX;   
 delete[] pTailY;

1. **Comment 14:**

Used to call the instruction function. This function will give instructions to the player that how to move snake and quit the game any time.

1. **Comment 15:**

Declared a character variable to get input from the player if he is ready to play.

1. **Comment 16:**

Used if statement to check the condition if player enter “Y”. Then the game will run.

1. **Comment 17:**

Used a while loop to check whether the game is over. If the game is over it will stop, and program will terminate.

1. **Comment 17.1:**

Game starter function () will be called to start the game and make the board of game, create snake’s position and fruit’s position.

1. **Comment 17.2:**

Getting input function () will be called to get the indication/ direction for snake’s movement.

1. **Comment 17.3:**

To check whether the game is over after getting input from the player.

1. **Comment 17.3.1:**

If game is over the loop will stop and program will be finished

1. **Comment 17.4:**

Changing position function () will be called to change the position of snake’s tail.

1. **Comment 17.5:**

Check game over function () will be called to check whether the game is over. And used a bool variable store the return value of that function.

1. **Comment 17.6:**

Updation function () will be called to update the values including score, fruit’s position, and tail’s length.

1. **Comment 18:**

To clear the terminal/console.

1. **Comment 19:**

To call the function and open the file and save the score of players.

1. **Comment 20:**

To ask the user whether he wants to see the score board.

1. **Comment 21:**

If player enter ‘Y’ then scoreboard will be displayed on the terminal/console.

1. **Comment 21.1:**

To clear the terminal/console.

1. **Comment 21.2:**

To call the function and display the scoreboard on terminal/console.

1. **Comment 22:**

Used the cout operation to greet the player for playing the game. It will only work at the end. When game will be over.

1. **Comment 23:**

Used else statement. It will work when user enter “N” instead of “Y” means if player is not ready to play then the game will not start, and program will jump to else statement instead of if statement in which game must run.

1. **Comment 23.1:**

Used a cout operation to say the user “Goodbye” if he doesn’t want to play the game.

1. **Comment 24:**

Used a delete keyword to delete the dynamic array that was declared to store the snake’s tail position on x-axis.

1. **Comment 25:**

Used a delete keyword to delete the dynamic array that was declared to store the snake’s tail position on x-axis.

**IMPORTANT NOTE**

It necessary to delete the dynamic memory allocation when not needed. It clears the heap memory of program. And make more space for other dynamic memory allocation.

**INSTRUCTION FUNCTION ()**

void **instructions**()  
{  
 cout << endl;  
 cout << "\t\t\t\t\t\t\*INSTRUCTIONS TO PLAY THE GAME\*" <<   
 cout << "-> Press 'a' to move 'LEFT' side." << endl;   
 cout << "-> Press 'd' to move 'RIGHT' side." << endl;   
 cout << "-> Press 'w' to move 'UP' side." << endl;   
 cout << "-> Press 's' to move 'DOWN' side." << endl;   
 cout << "-> NOTE: To Quit the game anytime. Press 'q'";   
}

1. **Comment 1:**

Used a cout operation for printing the heading of “instructions to play the game”.

1. **Comment 2:**

Used a cout operation for telling the player use of “a” for movement.

1. **Comment 3:**

Used a cout operation for telling the player use of “d” for movement.

1. **Comment 4:**

Used a cout operation for telling the player use of “w” for movement.

1. **Comment 5:**

Used a cout operation for telling the player use of “s” for movement.

1. **Comment 6:**

Used a cout operation for telling the player how to quit the game at any time.

**GAME STATER FUNCTION ()**

system("clear");  
 for (int i = 0; i < width + 2; i++)  
 {  
 cout << "\*";  
 }  
 cout << endl;  
 for (int i = 0; i < height; i++)   
 {  
 for (int j = 0; j < width; j++)  
 {  
 if (j == 0)   
 {  
 cout << "\*";  
 }  
 if (i == snake\_y && j == snake\_x)   
 {  
 cout << "S";  
 }  
 else if (i == fy && j == fx)   
 {  
 cout << "F";  
 }

1. **Comment 1:**

Used function to clear the terminal at every iteration when the game is not over. It will show that the snake is moving as animation. If we don’t use it the snake’s movement will be shown in different terminals but not as an animation. So, it is used only for the displaying the snake’s movement animation.

1. **Comment 2:**

Used the for loop to create the header border at first line of game box.

1. **Comment 3:**

Used for loop to create height of box row-wise.

1. **Comment 3.1:**

Used if statement to create the border on left side of box column-wise. It will first print the left border then move to then next columns.

1. **Comment 3.2:**

Used if statement to check the snake’s position on x-axis and y-axis in the start of game while creating the box int which snake will be move/placed.

If this condition becomes true it will print the snake’s head.

else  
 {  
 bool printSpace = false;   
 for (int k = 1; k < tailLength; k++)   
 {  
 if (pTailY[k] == i && pTailX[k] == j)   
 {  
 cout << "s";  
 printSpace = true;   
 }  
 }  
 if (!printSpace)   
 cout << " ";  
 }  
 if (j == width - 1)   
 {  
 cout << "\*";  
 }  
 }  
 cout << endl;   
 }  
 for (int i = 0; i < width + 2; i++)  
 {  
 cout << "\*";  
 }  
 cout << endl;  
 cout << "Score: " << score << endl;   
 cout << "Enter key: ";

1. **Comment 3.3:**

Used else if statement to check the fruit’s position on x-axis and y-axis while creating the box int which snake will be move/placed. If this condition becomes true it will print the fruit on that position.

1. **Comment 3.4:**

Int the else statement we declared a bool variable to check whether to print the empty space or snake’s tail length.

1. **Comment 3.5:**

Used a for loop to print the snake’s tail.

1. **Comment 3.6:**

Used an if statement. If the tail of snake on x-axis and y-axis become equal to i’s iteration, then it will print the tail of snake along its head.

1. **Comment 3.7:**

If snake’s tail is to be printed then bool, created to print empty space if snake’s tail is not printed, will become true and empty space will not print.

1. **Comment 3.8:**

Used an if statement to check the bool value, created to print empty space, if that bool value is not true It will print empty space instead of snake’s tail.

1. **Comment 3.9:**

Used and if statement to create the right-side border of box.

1. **Comment 3.10:**

Used endl function to jump to the next line after printing right side border of box.

1. **Comment 4:**

Used for loop to create the footer border of box.

1. **Comment 5:**

Used cout operation to display the score of player.

1. **Comment 6:**

Used cout operation to ask the user to enter the key for snake’s movement.

**GETTING INPUT FUNCTION ()**

temp\_snake\_x = snake\_x;   
 temp\_snake\_y = snake\_y;   
 char input;  
 cin >> input;   
 while (!(input == 'a' || input == 'd' || input == 's' || input == 'w' || input == 'q'))   
 {  
 cout << "You entered wrong key: ";   
 cin >> input;  
 }  
 switch (input)   
 {  
 case 'a':  
 snake\_x--;   
 break;  
 case 'd':  
 snake\_x++;   
 break;  
 case 'w':  
 snake\_y--;   
 break;  
 case 's':  
 snake\_y++;   
 break;  
 case 'q':  
 gameOver = true;   
 break;  
 }

1. **Comment 1:**

As in game setup function () on 12th comment we declare this variable to store the snake’s position on x-axis. It will be used to create snake’s tail on x-axis.

1. **Comment 2:**

As in game setup function () on 13th comment we declare this variable to store the snake’s position on y-axis. It will be used to create snake’s tail on y-axis.

1. **Comment 3:**

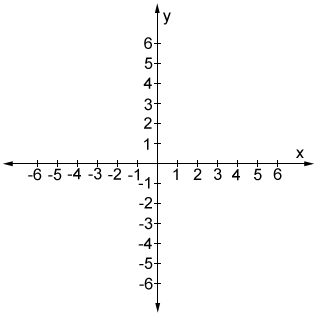
Declared a char variable to get indication/direction for snake’s movement.

1. **Comment 4:**

Used a while. It will check the user’s input. If user enter any other key than the given keys for snake’s movement this while loop will ask the user to enter the key again.

1. **Comment 4.1:**

Used a cout operation to display that the user has entered a wrong key.



**IMPORTANT NOTE**

* As in above graph we can see that positive x-axis move to right side and negative x-axis move to the left side. Similar case with the positive y and negative y axis. So, according to this we used some keys to move the snake in x-axis and y-axis. The keys are following:

1. **Comment 5:**

Used switch statement. It will check the input entered by the user and then it will change the snake’s position.

1. **Comment 5.1:**

If user enters ‘a’ key the direction of snake at x-axis will decrease and move to the left side.

1. **Comment 5.2:**

If user enters ‘d’ key the direction of snake at x-axis will increase and move to the right side.

1. **Comment 5.3:**

If user enters ‘w’ key the direction of snake at y-axis will decrease and move to the upward direction.

1. **Comment 5.4:**

If user enters ‘s’ key the direction of snake at y-axis will increase and move to the downward direction.

1. **Comment 5.6:**

If user enters ‘q’ key the bool value of game over variable will become true and the program will terminate.

**CHANGING POSITION FUNCTION ()**

pTailX[0] = temp\_snake\_x;   
 pTailY[0] = temp\_snake\_y;   
 for (int i = tailLength; i > 0; i--)   
 {  
 pTailX[i] = pTailX[i - 1];   
 pTailY[i] = pTailY[i - 1];   
 }

1. **Comment 1:**

To store the snake’s current position on x-axis for creating tail on x-axis. By doing this we can make the snake’s tail and move its tail with its head.

1. **Comment 2:**

To store the snake’s current position on y-axis for creating tail on y-axis. By doing this we can make the snake’s tail and move its tail with its head.

1. **Comment 3:**

Used for loop to store the snake’s position in array’s indexes for creating tail.

* **For example:**

If fruit is place at 4th of x-axis and 5th of y-axis and the head of snake is placed at 4th of x-axis and 6th of y-axis that is just behind the fruit. So, in the next move if snake eats the fruit, its head will be replaced by the fruit’s previous position that was 4th of x-axis and 5th of y-axis and as he ate the fruit a new tail will be formed. So, according to the this for loop we are talking about the tail’s index [1] **(**we said index [1] because the tail length will start from 1 instead of 0 that is the condition of for loop**)** will be equal to the previous location of snake’s head that was 4th of x-axis and 6th of y-axis. So, when the loop iterates it will print the snake’s head at previous position of fruit and tail will be printed at the previous position of snake’s head.

1. **Comment 3.1:**

To store the snake’s tail index on x-axis equal to the previous tail’s index on x-axis. So that the newly formed tail will be placed at the previous tail’s position.

1. **Comment 3.2:**

Similarly, to store the snake’s tail index on y-axis equal to the previous tail’s index on y-axis. So that the newly formed tail will be placed at the previous tail’s position.

**CHECK GAME OVER FUNCTION ()**

if (snake\_y < 0 || snake\_y > height - 1)   
 {  
 return true;  
 }  
 if (snake\_x < 0 || snake\_x > width - 1)   
 {  
 return true;  
 }  
 return false;

1. **Comment 1:**

Used an If statement to check whether the snake’s head touch the height of border and if yes then it will return the true the bool value and program will terminate.

1. **Comment 2:**

Used an If statement to check whether the snake’s head touch the width of border and if yes then it will return the true the bool value and program will terminate.

1. **Comment 3:**

If snake’s head doesn’t touch the height/width border it will return false bool value.

**UPDATION FUNCTION ()**

if (snake\_x == fx && snake\_y == fy)   
 {  
 fx = rand() % width;   
 fy = rand() % height;   
 score += 10;   
 tailLength++;  
 }

1. **Comment 1:**

Used an if statement to check whether the snake’s head coincide with fruit’s position.

1. **Comment 1.1:**

To generate new position of fruit on x-axis if snake eats the fruit.

1. **Comment 1.2:**

To generate new position of fruit on y-axis if snake eats the fruit.

1. **Comment 1.3:**

To update the score value when snake eats fruit.

1. **Comment 1.4:**

To increase the length of snake’s tail when snake eats the fruit.

**STORE SCORE IN FILE FUNCTION ()**

ofscore write\_score;

write\_score.open(“Score Board.txt,ios::app);

if(write\_score.is\_open())

{

write\_score<<name<<endl;

write\_Score<<score<<endl;

write\_score.close();

}

else

{  
 cout<<”File opening error”.

}

1. **Comment 1:**

To declare the object for writing in file.

1. **Comment 2:**

To open the file in append mode. So that new score will be written at end of file without removing the old data of file.

1. **Comment 3:**

To check whether the file is open.

1. **Comment 3.1:**

To write the name of users in file.

1. **Comment 3.2:**

To write the score of users in file.

1. **Comment 3.3:**

To close the file.

1. **Comment 4:**

If file is not opened.

**SCOREBOARD FUNCTION ()**

1. **Comment 1:**

To declare a dynamic array for storing the name of players from file.

1. **Comment 2:**

To declare a dynamic array for storing the score of players from file.

1. **Comment 3:**

To declare a object for reading the file.

1. **Comment 4:**

To open the file.

1. **Comment 5:**

To check whether the file is open.

1. **Comment 6:**

To get data until it reaches at the end of file.

1. **Comment 6.1:**

To store the name of players getting from file.

1. **Comment 6.2:**

To store the score of players getting data from file.

1. **Comment 6.3:**

To convert the score of players obtained from file. Because the data stored in file is in the form of string. So, we converted that string into integer datatype by this Int\_Score variable.

1. **Comment 6.4:**

To convert the string into integer datatype.

1. **Comment 6.5:**

If there is any data in line, then it will store the score and name f players in dynamic array from file. If there is not data in line, then it will store anything.

1. **Comment 7:**

To close the file.

1. **Comment 8:**

If file is not opened, then it will display the error.

1. **Comment 9:**

Used this loop to arrange the score and names of player in descending order.

1. **Comment 10:**

To arrange the score and name of players for scoreboard in descending order.

1. **Comment 10.1:**

To swap the scores.

1. **Comment 10.2:**

To swap the names.

1. **Comment 11:**

To display the scoreboard.

1. **Comment 12:**

To delete the dynamically allocated memory for score of players.

1. **Comment 13:**

To delete the dynamically allocated memory for score of players.

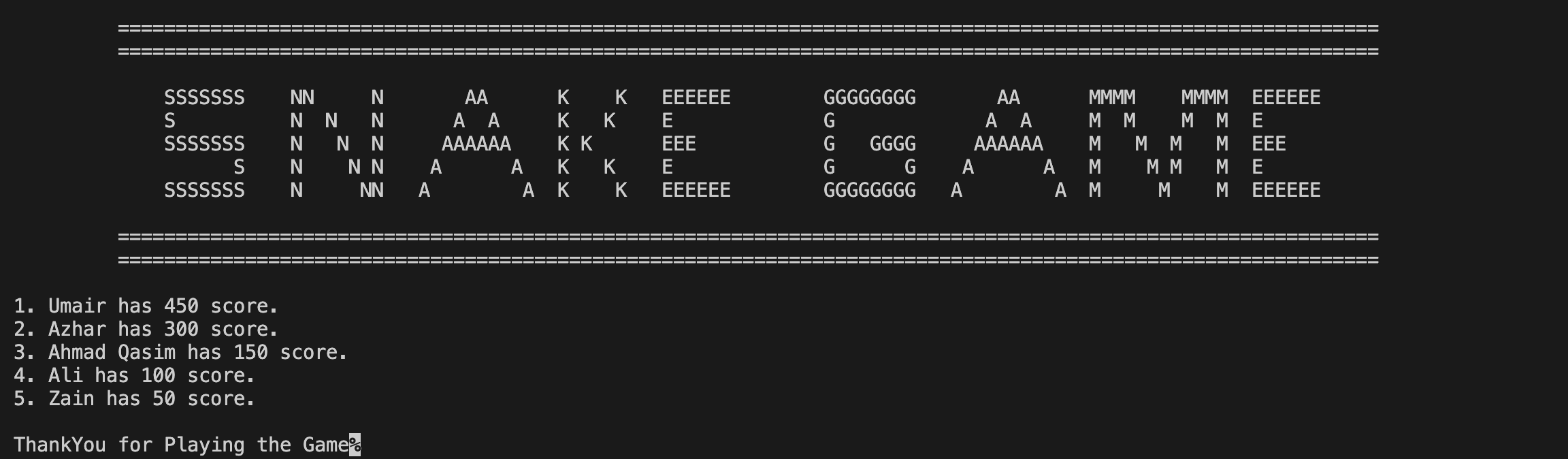
**PICTURES OUTPUT**



**Figure 1: Instructions**



**Figure 2: Game Board**



**Figure 3: Score Board**