

**Tribhuvan University**

**Institute of Engineering**

PULCHOWK CAMPUS

**Report on C-Quence: A C-Programming learning platform**

**Submitted By:**

PRAJJWAL BC (079BEL057)

PRATIK PAUDEL (079BEL060)

PUKAR SHIWAKOTI (079BEL063)

RAJU UBARKOTI (079BEL067)

**Submitted To:**

DR SURENDRA SHRESTHA

ASSOSIATE PROFESSOR

DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING

AUG 2023

# **Introduction**:

In today's world of cool technology changes, computer programming is like a magic key that opens the door to new ideas. And guess what? The C language is right in the middle of it all. It's like the strong foundation for lots of the cool things we use on our computers. If you're excited to start this amazing journey, think of C-quence as your friendly helper. It's here to guide people who want to learn computer programming with C. It's like exploring the tech world and becoming really awesome at the C language.

C-quence is a special program that helps you learn how to do computer programming using the C language. It's like a big learning adventure! C-quence teaches beginners all the important stuff about programming and how to write code that works. It's a super cool way to get started in the world of computer programming!

**Problem Statement:**

In the fast-paced world of technology, learning computer programming opens doors to endless creativity and innovation. At the heart of it all is the C language, a crucial building block for modern software. Imagine C-quence as your supportive guide, helping you dive into this exciting realm. If you're curious about coding and ready to learn, C-quence is here for you.

C-quence is a learning program designed to introduce people to computer programming using the C language. It's like a friendly mentor for beginners teaching essential. With C-quence, you'll gain confidence in programming and discover the possibilities that come with mastering C.

Created as a comprehensive educational initiative, C-quence is perfect for those starting their programming journey. It breaks down complex ideas into simple explanations, making programming accessible to everyone. You'll understand the basics and learn how to turn them into real code.

Think of C-quence as your partner in this adventure. It's an opportunity to grasp the fundamentals of coding and unleash your creativity. With C-quence, you'll build a strong foundation in programming, setting the stage for your tech exploration. Get ready to conquer coding challenges and explore the potential of the C language with C-quence by your side. Your journey to becoming a skilled programmer begins here.

**Theory**

**Input/Output Statements in C:**

Input/Output (I/O) statements in C facilitate communication between a program and the user or external devices. `printf` is used for output, allowing formatted data display. `scanf` is for input, accepting data from users or files. These statements make programs interactive and versatile.

**Control Statements in C:**

Control statements in C direct program flow based on conditions. Conditional statements like `if`, `else if`, and `switch` enable branching. Looping statements (`for`, `while`, `do-while`) iterate code. These constructs manage execution paths, enabling dynamic behavior and decision-making.

**User-Defined Functions:**

User-defined functions enhance code modularity and reusability in C. Developers create custom functions beyond built-in ones. Parameters pass data to functions, while return values provide results. Organizing code into functions simplifies debugging and maintenance.

**Arrays, Strings, and Functions:**

Arrays store multiple values of the same data type. Strings are character arrays used for text. Functions process arrays and strings, operating on elements or characters. Passing arrays to functions allows manipulation and analysis, enhancing program capabilities.

**Structures and Data Files:**

Structures group different data types under one object, aiding complex data organization. Data files store information persistently. File I/O functions (`fopen`, `fclose`, `fread`, `fwrite`) interact with data files. Combining structures and files stores structured data externally, crucial for data management and sharing.

**Some of the new functions that were used are:**

**1. strtok() Function:**

**Purpose: Tokenizes a string into smaller parts based on a specified delimiter.**

**Syntax:**

char \*strtok(char \*str, const char \*delim);

char sentence[] = "Hello, this is a sentence.";

char \*token = strtok(sentence, " ");

while (token != NULL) {

printf("%s\n", token);

token = strtok(NULL, " ");

}

**2. fseek() Function:**

**Purpose: Moves the file pointer to a specified position in a file.**

**Syntax:**

int fseek(FILE \*stream, long offset, int origin);

FILE \*file = fopen("example.txt", "r");

fseek(file, 10, SEEK\_SET); // Moves the pointer 10 bytes from the beginning

**3. sprintf() Function:**

**Purpose: Formats a string and stores the result in a character array.**

**Syntax:**

int sprintf(char \*str, const char \*format, ...);

char buffer[100];

int number = 42;

sprintf(buffer, "The number is: %d", number);

**4. kbhit() Function:**

**Purpose: Checks if a key has been pressed in the console.**

**Syntax:**

int kbhit(void);

if (kbhit()) {

char key = getch(); // Get the pressed key

printf("Key pressed: %c\n", key);

}

**Algorithm:**

**Algorithm for the opening() function:**

1. Initialize constants and variables:

* Initialize newWidth to 1370
* Initialize newHeight to 800
* Initialize ydiff to 100
* Initialize an array title[] with characters 'c', '-', 'q', 'u', 'e', 'n', 'c', 'e', '\0'

2. Draw Outer Shapes:

* Draw four sets of lines to create the outer quadrilateral shapes.

3. Draw Inner Shapes:

* Draw four sets of lines to create the inner quadrilateral shapes.

4. Color Filling:

* Create arrays to define coordinates for different quadrilateral regions (topleft1, topleft2, bottomleft1, bottomleft2, topright1, topright2, bottomright1, bottomright2, shape1, shape2).
* Fill polygons with white color using the setfillstyle() and fillpoly() functions for each region.

5. Display Text Animation:

* Create a 2D array seq[8][20] to store text sequences.
* Loop through each text sequence:
* Inside the loop, iterate through the characters of the current sequence and display it using outtextxy().
* Add a delay of 500 milliseconds after displaying each character.

6. Fill Inner Shapes with Yellow:

* Change the fillstyle to LINE\_FILL and fill polygons shape1 and shape2 with yellow color.
* Add a delay of 900 milliseconds.

7. Display Final Text:

* Change the text style to display larger text.
* Display the "Press Enter!" message using outtextxy().

8. End of the opening() function.

**Algorithm for theorybackground() function:**

1. Initialize constants and variables:

* Initialize newWidth to 1370
* Initialize newHeight to 800
* Initialize ydiff to 50
* Initialize shift to 55
* Initialize xshift to 35
* Initialize yshift to 35
* Initialize nshift to 26
* Initialize i, j, k

2. Draw Main Shape:

* Draw a complex shape using a series of line() function calls to create a specific pattern.

3. Draw Extra Lines and Squares:

* Draw additional lines and squares using line() function calls to extend the shape.

4. Color Filling:

* Define arrays for various shapes (outertriangle, sq1, sq2, sq3, botshap1, botshap2, botshap3, lineright, linetop, lineleft, toprect1, toprect2, toprect3).
* Fill polygons with white color using setfillstyle() and fillpoly() functions for each shape.

5. End of the theorybackground() function.

**Algorithm for the loading() function:**

1. Initialize constants and variables:

* Initialize newWidth to 1370
* Initialize newHeight to 800
* Initialize ydiff, shift, nshift, yshift, i, j, ndiff, xshift, and trans

2. Draw Circles:

* Loop over i from 100 to 103:
* Draw a circle at coordinates (450, 350) with radius i using the circle() function.
* Loop over i from 79 to 81:
* Draw a smaller circle at coordinates (450, 350) with radius i using the circle() function.

3. Draw Arcs:

* Loop over i from 85 to 95:
* Draw arcs at coordinates (450, 350) with varying angles and radius i using the arc() function.
* Loop over i from 65 to 75:
* Draw additional arcs at coordinates (450, 350) with varying angles and radius i using the arc() function.

4. Draw Outer Shape:

* Use line() function calls to draw a complex outer shape.

5. Draw Inner Lines and Polygons:

* Draw inner lines and fill polygons using the line() and fillpoly() functions for different shapes.

6. Draw Outer Shape Extensions:

* Draw extensions to the outer shape using line() function calls.

7. Fill Outer Shape Extension with Color:

* Define an array outer1[] for the outer shape extension.
* Fill the outer shape extension with the YELLOW color using setfillstyle() and fillpoly() functions.

8. Draw Loading Text:

* Set the text color to YELLOW.
* Set the text style to BOLD\_FONT with size 5.
* Draw the "LOADING..." text at coordinates (580, 300) using outtextxy() function.

9. Draw Loading Bar and Numbers:

* Loop over i from 0 to 25:
* Draw rectangles to represent the loading bar at different positions using the rectangle() function.
* Fill the rectangles with white color.
* Draw the loading percentage numbers at appropriate positions using outtextxy() function.
* Introduce a delay to simulate the loading animation.
* Introduce a longer delay after the loading completes to display the result.

10. End of the loading() function.

**Algorithm for the topic() function**

1. Clear the graphics screen or window.
2. Call the function theorybackground() to draw the background shapes and elements.
3. Set the text style to BOLD\_FONT, with a horizontal direction (HORIZ\_DIR), and a font size of 3.
4. Set the text color to YELLOW.
5. Display the given sentence on the graphics screen or window at coordinates (300, 100) using the function outtextxy().
6. End of topic() function

**Algorithm of Function read\_file():**

1. Open the file with the given filename for reading (fopen).
2. If the file is not opened successfully:

* Display an error message.
* Return.

1. Call the function theorybackground()
2. Initialize an array of Question structs named questions[MAX\_QUESTIONS].

* Initialize question\_count = 0.

1. Read questions and options from the file until end of file or maximum questions reached:

* Read a question line and options lines into questions[question\_count].
* Read the correct option index for the question.
* Remove newline characters from question and options strings.
* Increment question\_count.

1. Close the file (fclose).
2. Initialize score = 0.
3. For each question in the questions array:

* Display the question and options on the screen.
  + Set a timer of 30 seconds:
    - Display the countdown timer on the screen.
    - If a key is pressed:
      * Read the pressed key.
      * Convert the key to chosen\_option.
      * If chosen\_option is valid (1 to 4):
        + Break the loop.
    - If the timer reaches 0:
      * Call the function newscreen().
      * Jump to finish.

1. If chosen\_option is correct:
   * Display "Correct!" and increment score.
   * Else:
     + Display "Incorrect!"
2. Display "Press any key to continue..."

* Wait for a key press.
* Call the function newscreen().

1. Finish:
2. Display the final score on the screen.
3. Call the function feedback(score).
4. Display "Press any key to exit..."
5. Wait for a key press.

End of the function read\_file()

**Algorithm For function newscreen():**

1. Set the color of the graphics to BLACK.
2. Draw a rectangle on the screen with coordinates (127, 200) to (1248, 500).
3. Set the fill style to SOLID\_FILL and the color to BLACK.
4. Fill the drawn rectangle with the specified color
5. End of the function newscreen()

**Algorithm for functions chapter\_1\_theory() to chapter\_7\_theory()**

1. Initialize character arrays sentence, para1, para2………para"n" with corresponding content.
2. Call the function topic(sentence).
3. Call the function displaypara(para1 to para"n").
4. Wait for a key press using getch().
5. Call the function newscreen().
6. End of the function chapter-n-theory()

**Algorithm for function displaypara()**

1. Clear the screen or set a background.
2. Set the text style to bold with size 2.
3. Set the color to white.
4. Initialize x-coordinate (x) to 150 and y-coordinate (y) to 200.
5. Split the paragraph into lines using "\n" as the delimiter:

* Create an empty list called "lines".
* Create a variable called "current\_line" and set it to an empty string.
* For each character in the paragraph:

1. If the character is not "\n", add it to "current\_line".
2. If the character is "\n", append "current\_line" to "lines" and reset "current\_line" to an empty string.

* If "current\_line" is not empty, append it to "lines".

1. For each line in the list "lines":

* Display the line of text at coordinates (x, y).
* Increment y-coordinate (y) to move to the next line.

9. End of Function displaypara()

**Algorithm for function question()**

1. Set the background using theorybackground().
2. Set the color to white.
3. Set the text style to bold with size 8.
4. Display "Questions" at coordinates (450, 200) using outtextxy().
5. Set the text style to bold with size 4.
6. Display "You will have 30 seconds to answer each question" at coordinates (200, 300) using outtextxy().
7. Display "Press Enter to start" at coordinates (450, 350) using outtextxy().
8. Wait for the user to press Enter using getch().
9. End of Function question()

**Algorithm for function feedback()**

1. Set the color to yellow.
2. Set the text style to bold with size 4.
3. Initialize x-coordinate (x) to 200 and y-coordinate (y) to 300.
4. Use a switch statement based on the provided score:

* For each case, display the corresponding feedback message at coordinates (x, y) using outtextxy().

1. Set the color to white.
2. Set the text style to bold with size 2.
3. End of Function feedback()

**Source Code**

#include <stdio.h>

#include <conio.h>

#include <graphics.h>

#include <string.h>

#define MAX\_QUESTIONS 10

#define MAX\_QUESTION\_LENGTH 200

#define MAX\_OPTIONS 4

void opening();

void theorybackground();

void loading();

void chapter\_1\_theory();

void chapter\_2\_theory();

void chapter\_3\_theory();

void chapter\_4\_theory();

void chapter\_5\_theory();

void chapter\_6\_theory();

void chapter\_7\_theory();

void topic(char \*sentence);

void displaypara(char \*paragraph);

void read\_file(char \*filename);

void newscreen();

void questions();

void feedback(int score);

struct Question

{

char question[MAX\_QUESTION\_LENGTH];

char options[MAX\_OPTIONS][MAX\_QUESTION\_LENGTH];

int correct\_option;

};

int main()

{

int gd = DETECT, gm;

initgraph(&gd, &gm, "");

int newWidth = 1370;

int newHeight = 800;

initwindow(newWidth, newHeight, "c-quence");

opening();

getch();

cleardevice();

loading();

cleardevice();

chapter\_1\_theory();

cleardevice();

questions();

cleardevice();

loading();

cleardevice();

read\_file("Variables.txt");

cleardevice();

chapter\_2\_theory();

cleardevice();

questions();

cleardevice();

loading();

cleardevice();

read\_file("Input\_Output.txt");

cleardevice();

chapter\_3\_theory();

cleardevice();

questions();

cleardevice();

loading();

cleardevice();

read\_file("Control\_statements.txt");

cleardevice();

chapter\_4\_theory();

cleardevice();

questions();

cleardevice();

loading();

cleardevice();

read\_file("Function.txt");

cleardevice();

chapter\_5\_theory();

cleardevice();

questions();

cleardevice();

loading();

cleardevice();

read\_file("Array.txt");

cleardevice();

chapter\_6\_theory();

cleardevice();

questions();

cleardevice();

loading();

cleardevice();

read\_file("Structure.txt");

cleardevice();

chapter\_7\_theory();

cleardevice();

questions();

cleardevice();

loading();

cleardevice();

read\_file("Data\_Files.txt");

cleardevice();

getch();

closegraph();

return 0;

}

void opening()

{

int newWidth = 1370;

int newHeight = 800;

int ydiff = 100;

int i, j;

int title[] = {'c', '-', 'q', 'u', 'e', 'n', 'c', 'e', '\0'};

// OUTER

// TOP LEFT

line(100, 300, 250, 150);

line(250, 150, 500, 150);

line(100, 300, 115, 300);

line(115, 300, 215, 200);

line(215, 200, 200, 200);

line(300, 150, 310, 160);

line(310, 160, 450, 160);

line(450, 160, 460, 150);

// BOTTOM LEFT

line(250, newHeight - ydiff - 150, 500, newHeight - ydiff - 150);

line(100, newHeight - ydiff - 300, 250, newHeight - ydiff - 150);

line(100, newHeight - ydiff - 300, 115, newHeight - ydiff - 300);

line(115, newHeight - ydiff - 300, 215, newHeight - ydiff - 200);

line(215, newHeight - ydiff - 200, 200, newHeight - ydiff - 200);

line(300, newHeight - ydiff - 150, 310, newHeight - ydiff - 160);

line(310, newHeight - ydiff - 160, 450, newHeight - ydiff - 160);

line(450, newHeight - ydiff - 160, 460, newHeight - ydiff - 150);

// TOP RIGHT

line(newWidth - 100, 300, newWidth - 250, 150);

line(newWidth - 250, 150, newWidth - 500, 150);

line(newWidth - 115, 300, newWidth - 215, 200);

line(newWidth - 215, 200, newWidth - 200, 200);

line(newWidth - 115, 300, newWidth - 100, 300);

line(newWidth - 300, 150, newWidth - 310, 160);

line(newWidth - 310, 160, newWidth - 450, 160);

line(newWidth - 450, 160, newWidth - 460, 150);

// BOTTOM RIGHT

line(newWidth - 100, newHeight - ydiff - 300, newWidth - 250, newHeight - ydiff - 150);

line(newWidth - 250, newHeight - ydiff - 150, newWidth - 500, newHeight - ydiff - 150);

line(newWidth - 115, newHeight - ydiff - 300, newWidth - 215, newHeight - ydiff - 200);

line(newWidth - 215, newHeight - ydiff - 200, newWidth - 200, newHeight - ydiff - 200);

line(newWidth - 115, newHeight - ydiff - 300, newWidth - 100, newHeight - ydiff - 300);

line(newWidth - 300, newHeight - ydiff - 150, newWidth - 310, newHeight - ydiff - 160);

line(newWidth - 310, newHeight - ydiff - 160, newWidth - 450, newHeight - ydiff - 160);

line(newWidth - 450, newHeight - ydiff - 160, newWidth - 460, newHeight - ydiff - 150);

// INNER

// INNER TOP RIGHT

line(newWidth - 175, 300, newWidth - 275, 200);

line(newWidth - 275, 200, newWidth - 325, 200);

line(newWidth - 350, 160, newWidth - 350, 215);

line(newWidth - 375, 200, newWidth - 425, 200);

line(newWidth - 375, 215, newWidth - 500, 215);

line(newWidth - 500, 215, newWidth - 530, 245);

line(newWidth - 530, 245, newWidth - 675, 245);

line(newWidth - 275, 215, newWidth - 325, 215);

line(newWidth - 325, 215, newWidth - 325, 275);

line(newWidth - 325, 275, newWidth - 215, 275);

line(newWidth - 215, 275, newWidth - 275, 215);

// INNER TOP LEFT

line(175, 300, 275, 200);

line(275, 200, 325, 200);

line(350, 160, 350, 215);

line(375, 200, 425, 200);

line(375, 215, 500, 215);

line(500, 215, 530, 245);

line(530, 245, 675, 245);

line(275, 215, 325, 215);

line(325, 215, 325, 275);

line(325, 275, 215, 275);

line(215, 275, 275, 215);

// INNER BOTTOM LEFT

line(175, newHeight - ydiff - 300, 275, newHeight - ydiff - 200);

line(275, newHeight - ydiff - 200, 325, newHeight - ydiff - 200);

line(350, newHeight - ydiff - 160, 350, newHeight - ydiff - 215);

line(375, newHeight - ydiff - 200, 425, newHeight - ydiff - 200);

line(375, newHeight - ydiff - 215, 500, newHeight - ydiff - 215);

line(500, newHeight - ydiff - 215, 530, newHeight - ydiff - 245);

line(530, newHeight - ydiff - 245, 675, newHeight - ydiff - 245);

// INNER BOTTOM RIGHT

line(newWidth - 175, newHeight - ydiff - 300, newWidth - 275, newHeight - ydiff - 200);

line(newWidth - 275, newHeight - ydiff - 200, newWidth - 325, newHeight - ydiff - 200);

line(newWidth - 350, newHeight - ydiff - 160, newWidth - 350, newHeight - ydiff - 215);

line(newWidth - 375, newHeight - ydiff - 200, newWidth - 425, newHeight - ydiff - 200);

line(newWidth - 375, newHeight - ydiff - 215, newWidth - 500, newHeight - ydiff - 215);

line(newWidth - 500, newHeight - ydiff - 215, newWidth - 530, newHeight - ydiff - 245);

line(newWidth - 530, newHeight - ydiff - 245, newWidth - 675, newHeight - ydiff - 245);

// COLOURING

// TOP RIGHT

int topleft1[] = {100, 300, 115, 300, 215, 200, 200, 200};

int topleft2[] = {300, 150, 310, 160, 450, 160, 460, 150};

int bottomleft1[] = {100, newHeight - ydiff - 300, 115, newHeight - ydiff - 300, 215, newHeight - ydiff - 200, 200, newHeight - ydiff - 200};

int bottomleft2[] = {300, newHeight - ydiff - 150, 310, newHeight - ydiff - 160, 450, newHeight - ydiff - 160, 460, newHeight - ydiff - 150};

int topright1[] = {newWidth - 115, 300, newWidth - 215, 200, newWidth - 200, 200, newWidth - 100, 300};

int topright2[] = {newWidth - 300, 150, newWidth - 310, 160, newWidth - 450, 160, newWidth - 460, 150};

int bottomright1[] = {newWidth - 115, newHeight - ydiff - 300, newWidth - 215, newHeight - ydiff - 200, newWidth - 200, newHeight - ydiff - 200, newWidth - 100, newHeight - ydiff - 300};

int bottomright2[] = {newWidth - 300, newHeight - ydiff - 150, newWidth - 310, newHeight - ydiff - 160, newWidth - 450, newHeight - ydiff - 160, newWidth - 460, newHeight - ydiff - 150};

int shape1[] = {275, 215, 325, 215, 325, 275, 215, 275, 275, 215};

int shape2[] = {newWidth - 275, 215, newWidth - 325, 215, newWidth - 325, 275, newWidth - 215, 275, newWidth - 275, 215};

setfillstyle(SOLID\_FILL, WHITE);

fillpoly(4, topleft2);

fillpoly(4, topleft1);

fillpoly(4, bottomleft1);

fillpoly(4, bottomleft2);

fillpoly(4, topright1);

fillpoly(4, topright2);

fillpoly(4, bottomright1);

fillpoly(4, bottomright2);

settextstyle(BOLD\_FONT, HORIZ\_DIR, 9);

char seq[8][20] = {"c", "c-", "c-q", "c-qu", "c-que", "c-quen", "c-quenc", "c-quence"};

for (i = 0; i < 8; i++)

{

j = 0;

while (seq[i][j] != '\0')

{

outtextxy(450, 300, seq[i]);

j++;

}

delay(500);

}

delay(500);

setfillstyle(LINE\_FILL, YELLOW);

fillpoly(4, shape1);

fillpoly(4, shape2);

delay(900);

settextstyle(BOLD\_FONT, HORIZ\_DIR, 5);

outtextxy(540, 500, "Press Enter!");

}

void theorybackground()

{

int newWidth = 1370;

int newHeight = 800;

int ydiff = 50;

int shift = 55;

int xshift = 35;

int yshift = 35;

int nshift = 26;

int i;

int j = 70;

int k = 0;

setcolor(YELLOW);

// MAIN LOOP

line(300, 50, 1200, 50);

line(1200, 50, 1250, 100);

line(1250, 100, 1250, 500);

line(1250, 500, 1200, 550);

line(1200, 550, 1200, 700);

line(1200, 700, 800, 700);

line(800, 700, 750, 650);

line(750, 650, 550, 650);

line(550, 650, 500, 700);

line(500, 700, 400, 700);

line(400, 700, 385, 715);

line(385, 715, 100, 715);

line(100, 715, 100, 200);

line(100, 200, 300, 50);

setcolor(WHITE);

line(100, 190, 290, 50);

line(290, 50, 100, 50);

line(100, 50, 100, 190);

// EXTRA LINES TOP

line(300, 60, 350, 60);

line(350, 60, 350, 65);

line(350, 65, 300, 65);

line(300, 65, 300, 60);

line(300 + shift, 60, 350 + shift, 60);

line(350 + shift, 60, 350 + shift, 65);

line(350 + shift, 65, 300 + shift, 65);

line(300 + shift, 65, 300 + shift, 60);

line(300 + 2 \* shift, 60, 350 + 2 \* shift, 60);

line(350 + 2 \* shift, 60, 350 + 2 \* shift, 65);

line(350 + 2 \* shift, 65, 300 + 2 \* shift, 65);

line(300 + 2 \* shift, 65, 300 + 2 \* shift, 60);

// EXTRA SQUARES BOTTOM

line(115, 700, 140, 700);

line(140, 700, 140, 675);

line(140, 675, 115, 675);

line(115, 675, 115, 700);

line(xshift + 115, 700, xshift + 140, 700);

line(xshift + 140, 700, xshift + 140, 675);

line(xshift + 140, 675, xshift + 115, 675);

line(xshift + 115, 675, xshift + 115, 700);

line(115, 700 - yshift, 140, 700 - yshift);

line(140, 700 - yshift, 140, 675 - yshift);

line(140, 675 - yshift, 115, 675 - yshift);

line(115, 675 - yshift, 115, 700 - yshift);

// BOTTOM SHAPE

line(500, 650, 540, 650);

line(540, 650, 500, 690);

line(500, 690, 410, 690);

line(410, 690, 435, 665);

line(435, 665, 485, 665);

line(485, 665, 500, 650);

// BOTTOM LINES

line(420, 690, 175, 690);

line(125, 640, 125, 400);

line(125, 400, 100, 355);

// BOTTOM SHAPES

line(750, 655, 795, 700);

line(795, 700, 725, 700);

line(725, 700, 725, 655);

line(725, 655, 750, 655);

for (i = 0; i < 6; i++)

{

line(720 - i \* nshift, 700, 720 - i \* nshift, 655);

line(720 - i \* nshift, 655, 710 - i \* nshift, 655);

line(710 - i \* nshift, 655, 710 - i \* nshift, 700);

line(710 - i \* nshift, 700, 720 - i \* nshift, 700);

}

line(newWidth - j - 750, 655, newWidth - j - 795, 700);

line(newWidth - j - 795, 700, newWidth - j - 725, 700);

line(newWidth - j - 725, 700, newWidth - j - 725, 655);

line(newWidth - j - 725, 655, newWidth - j - 750, 655);

// LINE EXTENSIONS RIGHT

line(1250, 300, 1275, 325);

line(1275, 325, 1275, 450);

line(1275, 450, 1250, 475);

setcolor(WHITE);

line(1260, 465, 1260, 600);

line(1260, 600, 1200, 660);

// LINE EXTENSIONS TOP

line(1125, 50, 1100, 25);

line(1100, 25, 900, 25);

line(900, 25, 875, 50);

setcolor(WHITE);

line(890, 35, 400, 35);

line(400, 35, 385, 50);

// LINE EXTENSIONS LEFT

line(100, 700, 75, 675);

line(75, 675, 75, 500);

line(75, 500, 100, 475);

// COlOURING

// OUTER TRIANGLE

int outertriangle[] = {100, 50, 290, 50, 100, 190};

setfillstyle(SOLID\_FILL, WHITE);

fillpoly(3, outertriangle);

// EXTRA SQUARES BOTTOM

int sq1[] = {115, 700, 140, 700, 140, 675, 115, 675, 115, 700};

int sq2[] = {xshift + 115, 700, xshift + 140, 700, xshift + 140, 675, xshift + 115, 675, xshift + 115, 700};

int sq3[] = {115, 700 - yshift, 140, 700 - yshift, 140, 675 - yshift, 115, 675 - yshift, 115, 700 - yshift};

setfillstyle(SOLID\_FILL, WHITE);

fillpoly(4, sq1);

fillpoly(4, sq2);

fillpoly(4, sq3);

// BOTTOM SHAPE

int botshap1[] = {500, 650, 540, 650, 500, 690, 410, 690, 435, 665, 485, 665, 500, 650};

setfillstyle(SOLID\_FILL, WHITE);

fillpoly(6, botshap1);

// BOTTOM SHAPES

int botshap2[] = {750, 655, 795, 700, 725, 700, 725, 655, 750, 655};

setfillstyle(SOLID\_FILL, WHITE);

fillpoly(4, botshap2);

int botshap3[] = {newWidth - j - 750, 655, newWidth - j - 795, 700, newWidth - j - 725, 700, newWidth - j - 725, 655, newWidth - j - 750, 655};

setfillstyle(SOLID\_FILL, WHITE);

fillpoly(4, botshap3);

// LINE EXTENSIONS RIGHT

int lineright[] = {1250, 300, 1275, 325, 1275, 450, 1250, 475};

setfillstyle(SOLID\_FILL, WHITE);

fillpoly(4, lineright);

// LINE EXTENSIONS TOP

int linetop[] = {1125, 50, 1100, 25, 900, 25, 875, 50};

setfillstyle(SOLID\_FILL, WHITE);

fillpoly(4, linetop);

// LINE EXTENSIONS LEFT

int lineleft[] = {100, 700, 75, 675, 75, 500, 100, 475};

setfillstyle(SOLID\_FILL, WHITE);

fillpoly(4, lineleft);

// INNER RECTANGLES TOP RIGHT

int toprect1[] = {300, 60, 350, 60, 350, 65, 300, 65, 300, 60};

int toprect2[] = {300 + shift, 60, 350 + shift, 60, 350 + shift, 65, 300 + shift, 65, 300 + shift, 60};

int toprect3[] = {300 + 2 \* shift, 60, 350 + 2 \* shift, 60, 350 + 2 \* shift, 65, 300 + 2 \* shift, 65, 300 + 2 \* shift, 60};

setfillstyle(SOLID\_FILL, WHITE);

fillpoly(4, toprect1);

fillpoly(4, toprect2);

fillpoly(4, toprect3);

}

void loading()

{

int newWidth = 1370;

int newHeight = 800;

int ydiff = 100;

int shift = 50;

int nshift = 15;

int yshift = 10;

int i, j = 195;

int ndiff = 60;

int xshift = 60;

int trans = 185;

// CIRCLE

setcolor(WHITE);

for (i = 100; i < 104; i++)

{

circle(450, 350, i);

}

setcolor(WHITE);

for (i = 79; i <= 81; i++)

{

circle(450, 350, i);

}

// ARCS

setcolor(YELLOW);

for (i = 85; i <= 95; i++)

{

arc(450, 350, 10, 110, i);

arc(450, 350, 130, 230, i);

arc(450, 350, 250, 350, i);

}

setcolor(YELLOW);

for (i = 65; i <= 75; i++)

{

arc(450, 350, 66, 176, i);

arc(450, 350, 186, 296, i);

arc(450, 350, 306, 56, i);

}

// OUTER SHAPE

setcolor(WHITE);

line(shift + 465, 265, shift + 515, 265);

line(shift + 515, 265, shift + 530, 280);

line(shift + 530, 280, shift + 600, 280);

line(shift + 600, 280, shift + 615, 265);

line(shift + 615, 265, shift + 900, 265);

line(shift + 900, 265, shift + 950, 315);

line(shift + 950, 315, shift + 950, 425);

line(shift + 950, 425, shift + 925, 450);

line(shift + 925, 450, shift + 875, 450);

line(shift + 875, 450, shift + 850, 475);

line(shift + 850, 475, shift + 550, 475);

line(shift + 550, 475, shift + 505, 430);

line(shift + 505, 430, shift + 475, 430);

// INNER LINE

setcolor(WHITE);

line(850, 275, 950, 275);

line(950, 275, 975, 300);

line(975, 300, 875, 300);

line(875, 300, 850, 275);

line(850, newHeight - j - 275, 950, newHeight - j - 275);

line(950, newHeight - j - 275, 975, newHeight - j - 300);

line(975, newHeight - j - 300, 875, newHeight - j - 300);

line(875, newHeight - j - 300, 850, newHeight - j - 275);

line(850 - xshift, newHeight - ndiff - 275, 950 - xshift, newHeight - ndiff - 275);

line(950 - xshift, newHeight - ndiff - 275, 975 - xshift, newHeight - ndiff - 300);

line(975 - xshift, newHeight - ndiff - 300, 875 - xshift, newHeight - ndiff - 300);

line(875 - xshift, newHeight - ndiff - 300, 850 - xshift, newHeight - ndiff - 275);

line(newWidth - 850 + trans, newHeight - ndiff - 275, newWidth - 950 + trans, newHeight - ndiff - 275);

line(newWidth - 950 + trans, newHeight - ndiff - 275, newWidth - 975 + trans, newHeight - ndiff - 300);

line(newWidth - 975 + trans, newHeight - ndiff - 300, newWidth - 875 + trans, newHeight - ndiff - 300);

line(newWidth - 875 + trans, newHeight - ndiff - 300, newWidth - 850 + trans, newHeight - ndiff - 275);

int inner1[] = {850, 275, 950, 275, 975, 300, 875, 300, 850, 275};

int inner2[] = {850, newHeight - j - 275, 950, newHeight - j - 275, 975, newHeight - j - 300, 875, newHeight - j - 300, newHeight - j - 275};

int inner3[] =

{

850 - xshift, newHeight - ndiff - 275, 950 - xshift, newHeight - ndiff - 275,

975 - xshift, newHeight - ndiff - 300, 875 - xshift, newHeight - ndiff - 300,

875 - xshift, newHeight - ndiff - 300, 850 - xshift, newHeight - ndiff - 275};

int inner4[] =

{

newWidth - 850 + trans, newHeight - ndiff - 275, newWidth - 950 + trans, newHeight - ndiff - 275,

newWidth - 975 + trans, newHeight - ndiff - 300, newWidth - 875 + trans, newHeight - ndiff - 300,

newWidth - 850 + trans, newHeight - ndiff - 275};

setfillstyle(SOLID\_FILL, WHITE);

fillpoly(4, inner1);

fillpoly(4, inner2);

fillpoly(4, inner3);

fillpoly(4, inner4);

// OUTER SHAPE EXTENSIONS

setcolor(WHITE);

line(800, 265, 815, 250);

line(815, 250, 925, 250);

line(925, 250, 940, 265);

int outer1[] = {800, 265, 815, 250, 925, 250, 940, 265};

setfillstyle(SOLID\_FILL, YELLOW);

fillpoly(4, outer1);

// LOADING TEXT

setcolor(YELLOW);

settextstyle(BOLD\_FONT, HORIZ\_DIR, 5);

outtextxy(580, 300, "LOADING...");

// LOADING BAR + NUMBER

setcolor(YELLOW);

for (i = 0; i < 26; i++)

{

rectangle(580 + i \* nshift, 350, 590 + i \* nshift, 425);

}

char num[][26] = {"01", "05", "09", "13", "17", "21", "25", "29", "33", "37", "41", "45", "49", "53", "57", "61", "65", "69", "73", "77", "81", "85", "89", "93", "97", "100"};

setfillstyle(SOLID\_FILL, WHITE);

for (i = 0; i < 26; i++)

{

int rect[] = {

580 + i \* nshift, 350,

590 + i \* nshift, 350,

590 + i \* nshift, 425,

580 + i \* nshift, 425};

fillpoly(4, rect);

setcolor(WHITE);

settextstyle(BOLD\_FONT, HORIZ\_DIR, 6);

if (i == 25)

{

outtextxy(405, 325, num[i]);

}

else

{

outtextxy(420, 325, num[i]);

}

delay(35);

}

delay(700);

}

void topic(char \*sentence)

{

cleardevice();

theorybackground();

settextstyle(BOLD\_FONT, HORIZ\_DIR, 3);

setcolor(YELLOW);

outtextxy(300, 100, sentence);

}

void read\_file(char \*filename)

{

FILE \*file = fopen(filename, "r");

if (file == NULL)

{

printf("Error opening the file!\n");

return;

}

theorybackground();

struct Question questions[MAX\_QUESTIONS];

int question\_count = 0;

while (fgets(questions[question\_count].question, MAX\_QUESTION\_LENGTH, file) != NULL &&

question\_count < MAX\_QUESTIONS)

{

for (int i = 0; i < MAX\_OPTIONS; i++)

{

if (fgets(questions[question\_count].options[i], MAX\_QUESTION\_LENGTH, file) == NULL)

{

printf("Error reading options for question %d\n", question\_count + 1);

return;

}

}

if (fscanf(file, "Correct Option: %d\n", &questions[question\_count].correct\_option) != 1)

{

printf("Error reading correct option for question %d\n", question\_count + 1);

return;

}

questions[question\_count].question[strcspn(questions[question\_count].question, "\n")] = '\0';

for (int i = 0; i < MAX\_OPTIONS; i++)

{

questions[question\_count].options[i][strcspn(questions[question\_count].options[i], "\n")] = '\0';

}

question\_count++;

}

fclose(file);

int score = 0;

for (int i = 0; i < question\_count; i++)

{

setcolor(WHITE);

settextstyle(BOLD\_FONT, HORIZ\_DIR, 2);

outtextxy(150, 200, questions[i].question);

int y = 250;

for (int j = 0; j < MAX\_OPTIONS; j++)

{

outtextxy(150, y, questions[i].options[j]);

y = y + 30;

}

int chosen\_option = 0;

char answer;

for (int remainingTime = 30; remainingTime >= 0; remainingTime--)

{

char countdown[3];

countdown[0] = (remainingTime / 10) + '0';

countdown[1] = (remainingTime % 10) + '0';

countdown[2] = '\0';

setcolor(YELLOW);

outtextxy(1125, 110, countdown);

delay(1000);

if (kbhit())

{

answer = getch();

chosen\_option = answer - '0';

if (chosen\_option >= 1 && chosen\_option <= 4)

{

break;

}

}

if (remainingTime == 0)

{

newscreen();

goto finish;

}

}

if (chosen\_option == questions[i].correct\_option)

{

outtextxy(150, 400, "Correct!");

score++;

}

else

{

outtextxy(150, 400, "Incorrect!");

}

outtextxy(150, 450, "Press any key to continue...");

getch();

newscreen();

}

finish:

setcolor(WHITE);

settextstyle(BOLD\_FONT, HORIZ\_DIR, 2);

char score\_str[50];

sprintf(score\_str, "Final Score: %d", score);

outtextxy(150, 200, score\_str);

feedback(score);

outtextxy(150, 250, "Press any key to exit...");

getch();

}

void newscreen()

{

setcolor(BLACK);

rectangle(127, 200, 1248, 500);

setfillstyle(SOLID\_FILL, BLACK);

bar(127, 200, 1248, 500);

}

void chapter\_1\_theory()

{

char sentence[] = {"Constant, Variables and Data types"};

char para1[] =

{

"Keywords:\nWe must know some Keywords in order to go in Side This. So here we go:\nKeywords are predefined words for C programming language. All the key words have meaning\nand these meanings cannot be changed. Some keywords are: int, void, else, float, const, \nif, goto, while, return, struct, switch, char, case, and so on. We will understand those\nkeywords' meaning as per need in the whole process of learning with interaction."};

char para2[] =

{

"Data Types: \nDifferent types of Data types are available which can be used according to our needs.\nData types have three classes and its inner sub-divisions. We will just learn at the\nsurface level and useful data types.\n1. Integer Type (int)\na) Integer are whole numbers. These types require 16 bits of Storage.\nb) This type has signed and unsigned type as per need; we use what is required.\nc) For example: It represents values from -2,147,483,648 to +2,147,483,647 and from 0 to \n4,294,967,295.\nd) Its conversion characters are d, u, ld, lu according to the requirement of condition."};

char para3[] =

{

"2. Floating Point Types (float)\na) It reserves 4 bytes in memory.\nb) It represents fractional numbers of the range -1.2\*10^-38 to +3.4\*10^+38.\nc) Its conversion character is f.\nd) Variable is defined as: float a;"};

char para4[] =

{

"3. Double Precision Floating Point Type (double)\na) To provide more accuracy than the float type, it is used.\nb) A double data type number uses 64 bits (8 bytes).\nc) Variable is defined as: double a;\nd) Its conversion constant is lf."};

char para5[] =

{

"Constant: \nIt is a quantity that doesn't change during the execution. There are numerics, character \nconstants, and string constants.\nCharacter Constants:\na) It is a single character alphabet, digit, or a special symbol enclosed in single quota \nmarks.\nb) Maximum length is only one character.\nc) Examples: '2', 'a', 'D', '?' etc. \*Case Alert! (4 and '4' are not the same)"};

char para6[] =

{

"String Constant:\na) It is a sequence of characters enclosed in double quotes.\nb) It may contain letters, numbers, special characters, or blank spaces.\nc) Examples: \"Hello\", \"Hi\", \"A\", \"2008\" etc. \*Case Alert! (The character constant 'A' \nand String constant \"A\" are not similar."};

char para7[] =

{

"Variables:\na) It is a symbolic name used to store data items like numeric or character constants.\nb) It represents items of data input by the user and is used to identify them.\nc) It may have letters, digits, or underscore characters.\nd) Example: Nepal, first\_name, email\_addr, x2 etc."};

char para8[] =

{

"Variable Declaration:\n a) In programming, variables must be defined before using them.\n b) Variables are declared or defined using their own syntax.\n c) Syntax of Variable Declaration: data-type variable\_name1, variable\_name\_2,......, \n variable\_name\_n;\n d) It is needed to associate the variable with specific data. This means to declare or \n define data as what type it is, which helps the compiler to allocate memory space.\n e) For example, in the case of int, it will allocate 16 bits of Memory."};

topic(sentence);

displaypara(para1);

getch();

newscreen();

displaypara(para2);

getch();

newscreen();

displaypara(para3);

getch();

newscreen();

displaypara(para4);

getch();

newscreen();

displaypara(para5);

getch();

newscreen();

displaypara(para6);

getch();

newscreen();

displaypara(para7);

getch();

newscreen();

displaypara(para8);

getch();

newscreen();

}

void chapter\_2\_theory()

{

char sentence[] = {"Operators & Expressions and Input/Output Operations"};

char para1[] =

{

"In C programming, operators are special symbols that help us do different things with\nnumbers and values. operators play a fundamental role in performing various operations\non variables and constants. These operators are symbols that allow us to carry out basic\narithmetic calculations, such as addition (+), subtraction (-), multiplication (\*),\ndivision (/), and finding the remainder (%). Additionally, we have relational operators\nlike equality (==), inequality (!=), greater than (>), less than (<), greater than or\nequal to (>=), and less than or equal to (<=), which help us compare values and determine\ntrue or false outcomes. Logical operators, namely AND (&&), OR (||), and NOT (!), allow us\nto combine multiple conditions and evaluate expressions to check if certain conditions hold\ntrue or false."};

char para2[] =

{

"Another category of operators is the bitwise operators, which perform operations at the\nbit-level of data. These operators include bitwise AND (&), bitwise OR (|), bitwise \nXOR (^),left shift (<<), and right shift (>>). They are used to manipulate individual \nbits within a number, often utilized in low-level programming and working with hardware \ninterfaces.Furthermore, we have assignment operators like the basic assignment operator\n(=), which allows us to assign values to variables. Additionally, there are compound\n assignment operators like +=, -=, \*=, /=, and %=, which provide a shorthand way of \ncombining arithmetic operations with assignment."};

char para3[] =

{

"Expressions in C programming are combinations of operators, variables, and constants that\nyield a single value. They are an essential part of writing complex calculations and\ndecision-making structures within a program. For instance, we can use expressions to\ncompute mathematical formulas or evaluate conditions based on user input or other data.\nExpressions can be as simple as a single variable or as complex as a combination of \nmultiple operators and variables."};

char para4[] =

{

"On the other hand, input and output (I/O) operations are crucial for interacting with\nexternal devices like keyboards, screens, files, and other peripherals. In C programming,\nI/O operations are handled using the \"stdio.h\" header file, which provides functions and\nmacros to read and write data. The formatted input function, \"scanf()\", allows us to read\nformatted data from the standard input stream (stdin), while formatted output function,\n\"printf()\", displays formatted data on the standard output stream (stdout). Control strings\nare used with these functions to specify the data format, like %f for floating-point \nvalues,%c for characters, and %d for integers."};

char para5[] =

{

"In summary, C programming operators and expressions are vital tools that enable us to\nperform various calculations, comparisons, and logical evaluations. Understanding how to\nuse these operators effectively is crucial for developing efficient and powerful C\nprograms. Moreover, input and output operations facilitate communication between the\nprogram and external devices, allowing for data input and output in a formatted manner.\nBy mastering these concepts, programmers can harness the full potential of the C\nprogramming language to build a wide range of applications and software solutions."};

topic(sentence);

displaypara(para1);

getch();

newscreen();

displaypara(para2);

getch();

newscreen();

displaypara(para3);

getch();

newscreen();

displaypara(para4);

getch();

newscreen();

displaypara(para5);

getch();

}

void chapter\_3\_theory()

{

char sentence[] = {"Control Statements in C"};

char para1[] =

{

"Control statements in C programming are crucial for managing the flow of execution based \non specific conditions.The three main types of control statements are decision-making \nstatements, loop statements, and jump statements."};

char para2[] =

{

"A) Decision-making statements: \n1 )if statement: \n It allows the program to execute a block of code if a specified condition is true.\n Otherwise, the block is skipped. \n It's syntax is: \n if(test\_expression)\n { statement -block; }"};

char para3[] =

{

"2)if-else statement:\n It extends the if statement by providing an alternative block of code to\n execute if the condition is false. \n It's syntax is \n if(test\_expression) \n { true-block statement ; } \n else { false-block statement ; }"};

char para4[] =

{

"3) nested if...else statement: It allows multiple levels of conditions to be evaluated.\n It's syntax is\n if(condition1) { \n if(condition2) { \n statement; } \n else{statement;}\nelse{statement;}"};

char para5[] =

{

"B)Loop statements:\n\n1) while loop: It repeatedly executes a block of code as long as the specified condition \n is true. It's syntax is\n\n while(test\_condition)\n {\n body of loop;\n }"};

char para6[] =

{

"2)do-while loop: It is similar to the while loop, but it ensures the block of code \n executes at least once before checking the condition. It's syntax is\n\n do\n {\n body of loop;\n }\n while(test condition);"};

char para7[] =

{

"3)for loop: It provides a more compact way to define and control the loop variables.\n It's syntax is\n\n for(counter\_initialization; text\_condition; increment or decrement)\n {\n body of loop ;\n }"};

char para8[] =

{

"C) Jump Statements\n1) break statement: It is used to exit from a loop or switch statement prematurely.\n It's syntax is\n break;\n\n2) continue statement: It is used to skip the rest of the loop and start the next \n iteration. \n It's syntax is\n continue ;\n\n3) goto statement: It allows jumping to a labeled statement within the same function,\n but its use is generally discouraged due to its potential to create spaghetti code.\n It's syntax is\n goto label ;"};

topic(sentence);

displaypara(para1);

getch();

newscreen();

displaypara(para2);

getch();

newscreen();

displaypara(para3);

getch();

newscreen();

displaypara(para4);

getch();

newscreen();

displaypara(para5);

getch();

newscreen();

displaypara(para6);

getch();

newscreen();

displaypara(para7);

getch();

newscreen();

displaypara(para8);

getch();

}

void chapter\_4\_theory()

{

char sentence[] = {"Functions in C"};

char para1[] =

{

"In C programming, functions are blocks of code that perform a specific task and can be \ncalled from other parts of the program. Here's a basic theory of functions in C:\n a) Function Declaration: Before using a function in C, it needs to be declared. \n The declaration includes the function's return type, name, and parameters (if any).\n For examples:\n int add(int a, int b);"};

char para2[] =

{

"b)Function Definition: \n The function definition provides the actual implementation of the function. \n It consists of a function header and a function body enclosed in curly \n braces.\n For example: int add(int a, int b) { int sum = a + b; return sum; }"};

char para3[] =

{

"c) Function Call:\n To use a function, you call it by its name followed by parentheses containing the \n required arguments (if any). The function returns the specified data type.\n For example: int result = add(5, 3);"};

char para4[] =

{

"d) Return Statement:\n The return statement is used to return a value from a function. It can be used only \n in functions with a non-void return type.\n For example: \n return sum;"};

char para5[] =

{

"e) Void Functions:\n Functions with a return type of void do not return any value. They are used\n to perform actions without returning anything.\n For example:\n void greet() { printf(\"Hello, World!\"); }"};

char para6[] =

{

"User-defined Functions\n A user-defined function is a block of code created by the programmer to perform a\n specific task or set of tasks. These functions provide a way to \n modularize the program and enhance code reusability, making it easier to manage and \n maintain complex projects.To create a user-defined function, one must first declare\n its prototype, which includes the function's name, return type, \n and any parameters it accepts.The function's implementation \n is then defined separately, containing the actual code to be executed \n when the function is called."};

char para7[] =

{

"Function Prototypes\n In C, you can declare function prototypes at the beginning of the code \n to inform the compiler about the functions used in the program.\n It helps avoid errors related to undefined functions.\n For example: \n int add(int a, int b);\n void greet();"};

topic(sentence);

displaypara(para1);

getch();

newscreen();

displaypara(para2);

getch();

newscreen();

displaypara(para3);

getch();

newscreen();

displaypara(para4);

getch();

newscreen();

displaypara(para5);

getch();

newscreen();

displaypara(para6);

getch();

newscreen();

displaypara(para7);

getch();

}

void chapter\_5\_theory()

{

char sentence[] = {"Array, Strings and Pointers"};

char para1[] = {"In C programming, arrays, strings, and pointers are fundamental concepts used to \nhandle data efficiently. Let's towards each of them:"};

char para2[] = {

"Arrays\nAn array is a collection of elements of the same data type stored in contiguous\nmemory locations.The elements in an array can be accessed using their index, which starts\nfrom 0 to (array\_size - 1).Arrays are declared with a fixed size at compile time. \nSyntax is :\ndata\_type array\_name[size];"};

char para3[] = {

"Example: \nint numbers[5]; // Declaration of an integer array with size 5. \nnumbers[0] = 10; // Assigning value 10 to the first element of the array. \nint value = numbers[2]; // Accessing the third element of the array and storing it in \nthe 'value' variable."};

char para4[] = {

"Strings \nIn C, strings are represented as arrays of characters, terminated by a null character '\O'. \nStrings are used to store sequences of characters, such as words or sentences. \nStrings can be initialized using double quotes, and individual characters can be \naccessed using array indexing. \nSyntax is: \nchar string\_name[size]; or char string\_name[ ] = \"text\";"};

char para5[] = {

"Example: \nchar name[10] = \"John\"; // Declaration and initialization of a character array (string) \nwith size 10. \nprintf(\"Hello, %s!\\n\", name); // Printing the string using the 'printf' function."};

char para6[] = {

"Pointers \nPointers are variables that store memory addresses of other variables. They are used for \ndynamic memory allocation and efficient memory access.The '&' operator is used to get the \nmemory address of a variable, and the '\*' operator is used to declare a pointer and \ndereference it. \nSyntax is: \ndata\_type \*pointer\_name;"};

char para7[] = {

"Example: \nint num = 42; int \*ptr = # // Declaration of an integer pointer and assigning the memory \naddress of 'num' to it printf(\"Value of num: %d\", \*ptr); // Dereferencing the pointer to \nget the value stored in 'num'. \nConversion to Function Arrays and strings can be passed to functions as pointers, \nenabling functions to operate on the data without copying it. Pointers can be used to pass \nvariables by reference to functions, allowing functions to modify the original values."};

topic(sentence);

displaypara(para1);

getch();

newscreen();

displaypara(para2);

getch();

newscreen();

displaypara(para3);

getch();

newscreen();

displaypara(para4);

getch();

newscreen();

displaypara(para5);

getch();

newscreen();

displaypara(para6);

getch();

newscreen();

displaypara(para7);

getch();

}

void chapter\_6\_theory()

{

char sentence[] = {"Structure and Union"};

char para1[] = {

"Structure \nIn C, a structure is a composite data type that groups together \nvariables of different data types under a single name. Each variable within \nthe structure is called a member. The members can have different data types, \nand they are stored in contiguous memory locations. Structures are defined \nusing the struct keyword. The syntax of a structure is as follows: \nstruct structure\_name { data\_type member\_variables1; data\_type member\_variable2; \n............. ............. data\_type member\_variablen; };"};

char para2[] = {

"A structure in C programming can be initialized using the following syntax: \nstruct structure\_name structure\_variable = {value1, value2,........,valuen}; \nHere's a simple example: \nstruct Student { char name[50]; int age; float GPA; }; \nIn this example, we've defined a structure named Student with three members: \nname, age, and GPA. \nSelf-referential structures can be used as follows: \nstruct List { int data; struct List \*next; };"};

char para3[] = {

"Here, List represents a set of items organized in sequence. \nAs we know, all portions of C are related to each other. Therefore, they can be \ninterconnected through various processes. For instance, a pointer can be converted to \na structure. This conversion can be achieved using the syntax: \nptr\_variable.member\_name;"};

char para4[] = {

"Unions\nUnions are similar to structures, but they share the same memory location for all\ntheir members. Unlike structures, only one member of a union can have a value at a given\ntime, and accessing one member will overwrite the values of other members. Unions are also\ndefined using the union keyword. Here's a basic example as syntax: \nunion Data { int num;\nfloat value; char symbol; }; \nIn this case, the union Data has three members: num, value, \nand symbol. However, only one of them can contain a value at any given moment."};

topic(sentence);

displaypara(para1);

getch();

newscreen();

displaypara(para2);

getch();

newscreen();

displaypara(para3);

getch();

newscreen();

displaypara(para4);

getch();

}

void chapter\_7\_theory()

{

char sentence[] = {"File Handling Functions"};

char para1[] = {

"Data files in C programming are used to store and retrieve data from external sources. They\nare essential for saving data between program executions and for exchanging data between\nprograms. In C, the Standard I/O library provides functions to work with data files.\nFile Modes\nC provides different file modes that determine how the file can be accessed. The main\nmodes are:\n\"r\": Read mode. Open file for reading.\n\"w\": Write mode. Create or truncate a file for writing.\n\"a\": Append mode. Open file for writing at the end, or create a new file if it \ndoesn't exist.\"rb\", \"wb\", \"ab\": Binary mode. Used with respective read, write, \nand append modes for binary files."};

char para2[] = {

"File Handling Functions:\n1. fopen(): Opens a file and returns a file pointer. Syntax: FILE \*fopen(const char \n\*filename, const char \*mode);\n2. fgetc(): Reads a character from a file. Syntax: int fgetc(FILE \*stream);\n3. fputc(): Writes a character to a file. Syntax: int fputc(int char, FILE \*stream);\n4. fgets(): Reads a line from a file. Syntax: char \*fgets(char \*str, int n, FILE \*stream);\n5. fputs(): Writes a string to a file. Syntax: int fputs(const char \*str, FILE \*stream);"};

char para3[] = {

"6. fprintf(): Writes formatted output to a file. Syntax: int fprintf(FILE \*stream,\n const char \*format, ...);\n7. fscanf(): Reads formatted input from a file. Syntax: int fscanf(FILE \*stream,\n const char \*format, ...);\n8. fread(): Reads binary data from a file. Syntax: size\_t fread(void \*ptr, size\_t size,\n size\_t count, FILE \*stream);\n9. fwrite(): Writes binary data to a file. Syntax: size\_t fwrite(const void \*ptr,\n size\_t size, size\_t count, FILE \*stream);"};

char para4[] = {

"Opening a file in read mode:\nFILE \*file = fopen(\"data.txt\", \"r\");\n \nWriting to a file in append mode:\nFILE \*file = fopen(\"data.txt\", \"a\");\nfprintf(file, \"New data\n\");\n\nReading binary data from a file:\nFILE \*file = fopen(\"binary\_data.bin\", \"rb\");\nfread(buffer, sizeof(buffer), 1, file);"};

char para5[] = {

"Finally, data files are closed using fclose() which is the function for closing files.\nClosing Files\nFiles should always be closed after using them with fclose(). \nThe syntax of this function is:\nfclose(file);"};

topic(sentence);

displaypara(para1);

getch();

newscreen();

displaypara(para2);

getch();

newscreen();

displaypara(para3);

getch();

newscreen();

displaypara(para4);

getch();

newscreen();

displaypara(para5);

getch();

}

void displaypara(char \*paragraph)

{

theorybackground();

settextstyle(BOLD\_FONT, HORIZ\_DIR, 2);

setcolor(WHITE);

int x = 150;

int y = 200;

char \*line = strtok(paragraph, "\n");

while (line != NULL)

{

outtextxy(x, y, line);

y += textheight(line) + 5;

line = strtok(NULL, "\n");

}

}

void questions()

{

theorybackground();

setcolor(WHITE);

settextstyle(BOLD\_FONT, HORIZ\_DIR, 8);

outtextxy(450, 200, "Questions");

settextstyle(BOLD\_FONT, HORIZ\_DIR, 4);

outtextxy(200, 300, "You will have 30 seconds to answer each question");

setcolor(WHITE);

outtextxy(450, 350, "Press Enter to start");

getch();

}

void feedback(int score)

{

setcolor(YELLOW);

settextstyle(BOLD\_FONT, HORIZ\_DIR, 4);

int x = 200;

int y = 300;

switch (score)

{

case 0:

outtextxy(x, y, "Theory seems to have gone on vacation.");

break;

case 1:

outtextxy(x, y, "Review chapter's theory for stronger understanding.");

break;

case 2:

outtextxy(x, y, "Study chapter material more thoroughly.");

break;

case 3:

outtextxy(x, y, "Focus on chapter concepts to improve score.");

break;

case 4:

outtextxy(x, y, "Solidify understanding with chapter review.");

break;

case 5:

outtextxy(x, y, "Continue revisiting chapter for better grasp.");

break;

case 6:

outtextxy(x, y, "Dive deeper into chapter's theory for improvement.");

break;

case 7:

outtextxy(x, y, "Well done, refine knowledge for stronger grasp.");

break;

case 8:

outtextxy(x, y, "Impressive! Explore related concepts for growth.");

break;

case 9:

outtextxy(x, y, "Near mastery; explore advanced resources.");

break;

case 10:

outtextxy(x, y, "Outstanding! Consider sharing insights with others.");

break;

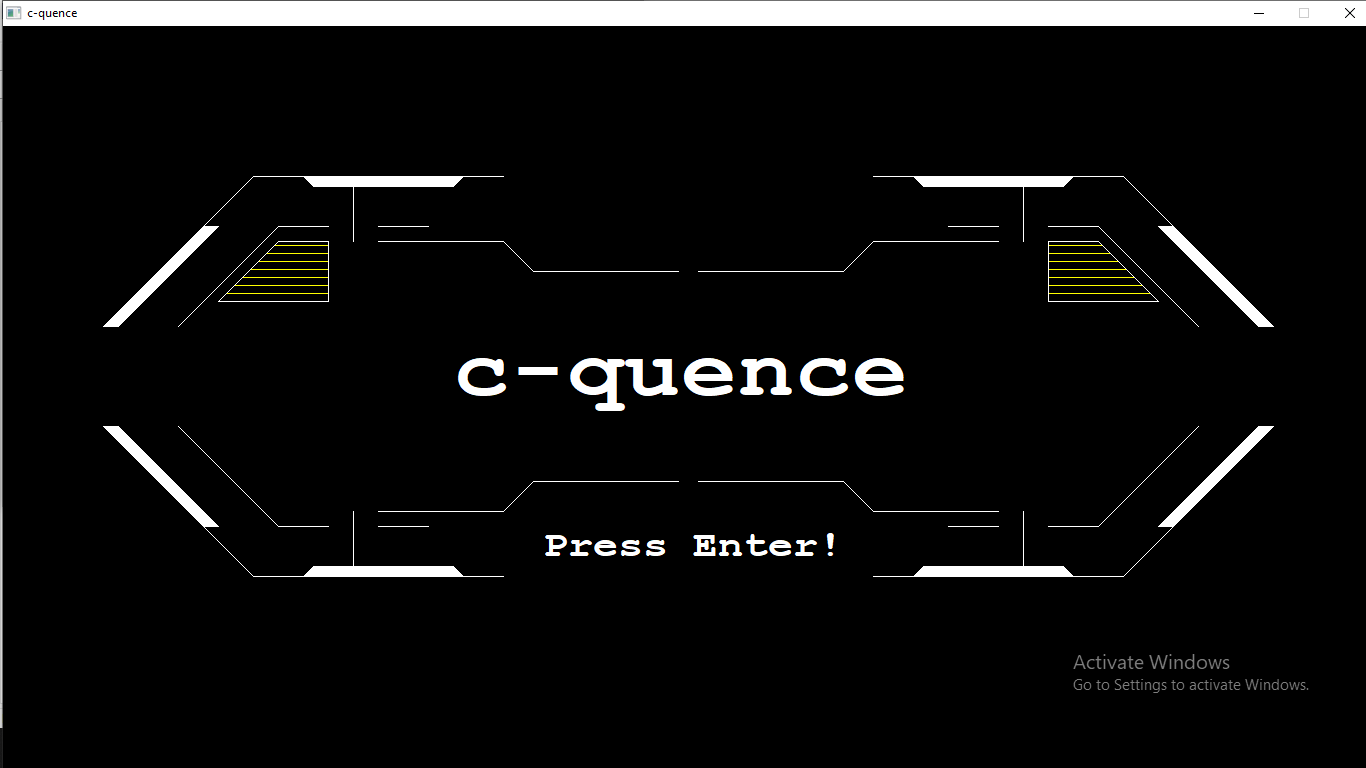
}

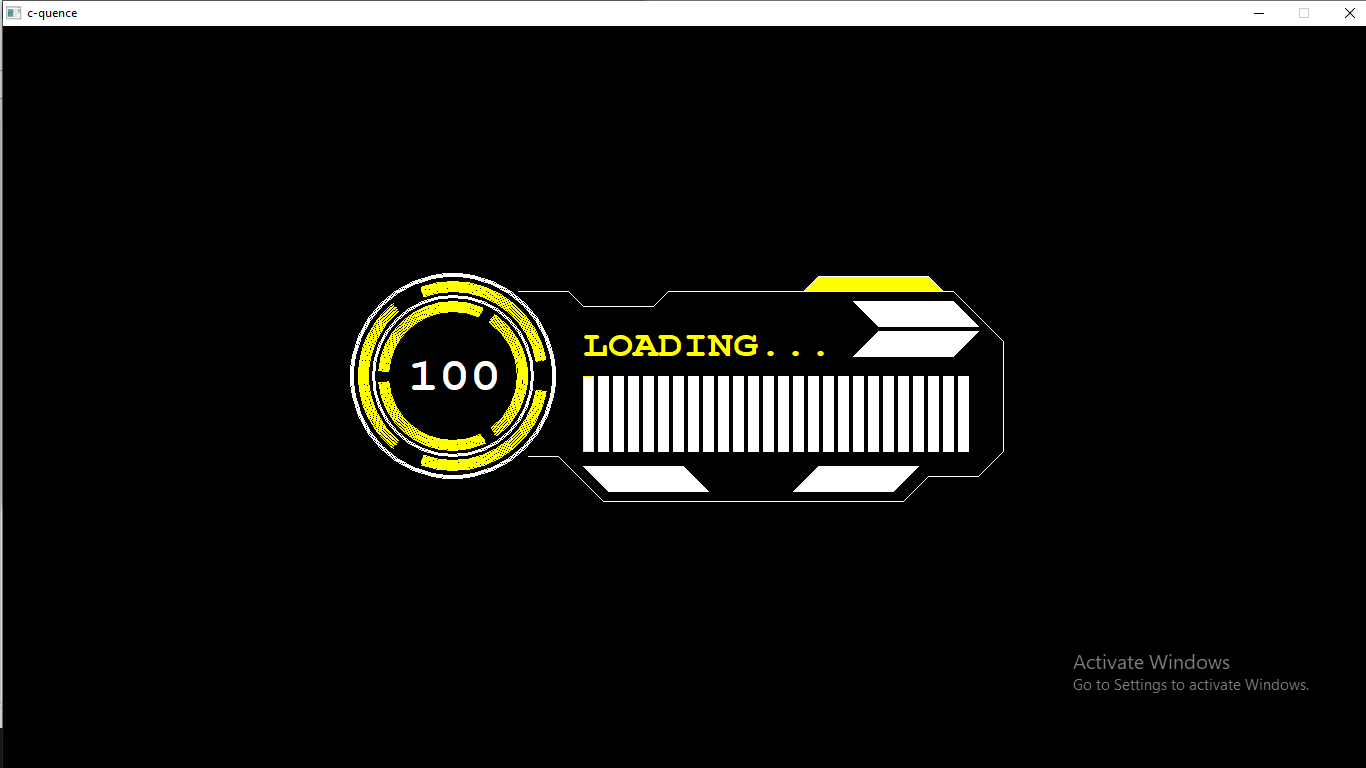
setcolor(WHITE);

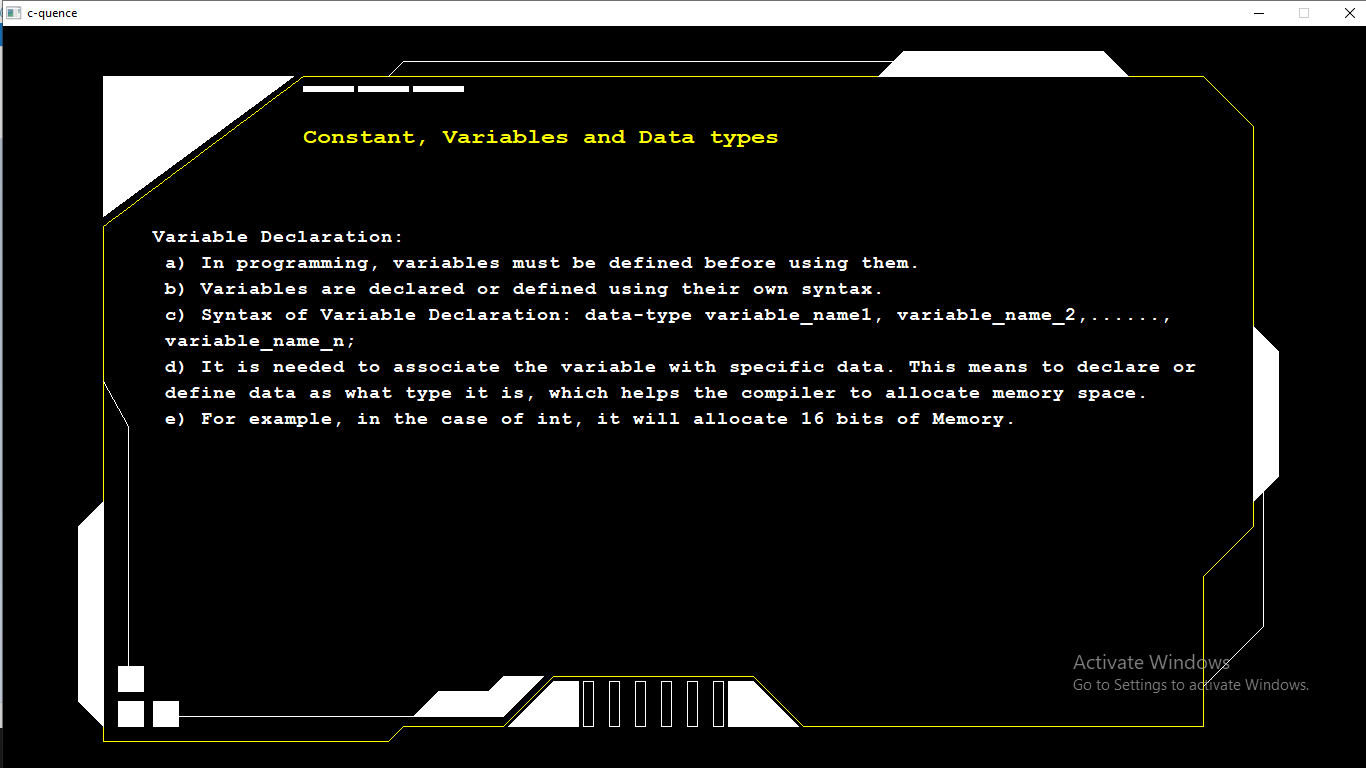
settextstyle(BOLD\_FONT, HORIZ\_DIR, 2);

}

**Output**





****

****

**Discussion**

During the creation of the C-quence project, we encountered various difficulties and challenges that tested our problem-solving skills and determination. One significant challenge we faced was related to graphics. Integrating graphics into the project proved to be trickier than expected, as we had to work with different libraries and ensure compatibility across various systems. This required careful research and troubleshooting to achieve the desired visual elements.

Compiler problems were another hurdle we had to overcome. Different compilers sometimes interpreted our code differently, leading to unexpected errors and inconsistencies. This forced us to refine our code and ensure its compatibility with multiple compilers, which was time-consuming but essential for a smooth user experience.

Perhaps one of the most frustrating challenges was when our code didn't work as intended, and we encountered multiple errors. Debugging became a crucial part of the development process, involving thorough testing, line-by-line analysis, and continuous revisions. It was a true test of patience and perseverance, as we had to identify and rectify numerous issues to make the project functional and reliable.

Despite these challenges, each obstacle became an opportunity for growth. We learned the importance of adaptability, research, and teamwork in resolving complex problems. Our dedication to overcoming these difficulties ultimately contributed to the development of C-quence, a project that not only offers valuable programming knowledge but also reflects our commitment to learning and improvement.

**Conclusion**

In conclusion, C-quence shows how important easy-to-understand education can be. By making learning fun and engaging, C-quence has made programming easy to understand and has given students a strong base in C programming. This proves the value of accessible education in helping people learn and succeed.