SHRIDEVI INSTITUTE OF ENGINEERING & TECHNOLOGY
TUMAKURU

Six days Skill Training

On

PROGRAMMING USING C

NAME : CHANDAN S B

BRANCH : ECE

SEM : 1ST Sem

TRAINER NAME :

Er. M. RAJESH KANNAN. ME.

Date : 25/11/24 – 30/11/24

Organized by

Skill and Career Development Cell

In association

ShriTEK Innovations

CONTENTS

Day 1

Introduction to Programming and C Language Basics

1. Program to Display "Hello, World!"

```
#include <stdio.h>
int main() {
printf("Hello, World!\n");
return 0;
}
```

Output:

Hello, World!

2. Program to Add Two Numbers:

```
#include <stdio.h>
int main() {
int a, b, sum;
printf("Enter two numbers: ");
scanf("%d %d", &a, &b);
sum = a + b;
printf("Sum = %d\n", sum);
return 0;
}
```

Input:

5 10

Sum = 15

3.Program to Calculate Simple Interest:

```
#include <stdio.h>
int main() {
float principal, rate, time, interest;
printf("Enter principal, rate, and time: ");
scanf("%f %f %f", &principal, &rate, &time);
interest = (principal * rate * time) / 100;
printf("Simple Interest = %.2f\n", interest);
return 0;
}
```

Input:

1000 5 2

Output:

Simple Interest = 100.00

Day 2

Control Structures

Decision-Making Statements
#The if Statement

Syntax:

```
if (condition) {
// code to execute if condition is true
}
```

Example:

```
#include <stdio.h>
int main() {
int num = 10;
if (num > 0) {
printf("The number is positive.\n");
}
return 0;
}
```

The if-else Statement

Syntax:

```
if (condition) {
} else {
// code to execute if condition is false
}
```

Example:

```
#include <stdio.h>
int main() {
int num;
printf("Enter a number: ");
scanf("%d", &num);
if (num % 2 == 0) {
printf("The number is even.\n");
} else {
printf("The number is odd.\n");
}
return 0;
}
```

Nested if Statements

Syntax:

```
if (condition1) {
if (condition2) {}
```

Example:

```
#include <stdio.h>
int main() {
int age = 20;
if (age > 18) {
if (age < 30) {
printf("You are a young adult.\n");
}
}
return 0;
}
```

The else if Ladder

Syntax:

```
if (condition1) {
// code if condition1 is true
} else if (condition2) {
// code if condition2 is true
} else {
// code if no condition is true
}
```

Example:

```
#include <stdio.h>
int main() {
int marks;
printf("Enter your marks: ");
scanf("%d", &marks);
if (marks >= 90) {
printf("Grade: A\n");
} else if (marks >= 75) {
printf("Grade: B\n");
} else if (marks >= 50) {
printf("Grade: C\n");
} else {
printf("Grade: F\n");
}
return 0;
}
```

Write a program to check if a number is positive, negative, or zero.

```
#include <stdio.h>

int main() {
```

```
    int num;
    printf("Enter a number: ");
    scanf("%d", &num);
    if (num > 0) {
        printf("%d is positive.\n", num);
    } else if (num < 0) {
        printf("%d is negative.\n", num);
    } else {
        printf("The number is zero.\n");
    }
    return 0;
}
```

```
#include <stdio.h>
int factorial(int n) {
    if (n == 0) {
        return 1;
    }
    return n * factorial(n - 1);
}
```

Factorial Number

```
int main() {
    int num;
    printf("Enter a number: ");
    scanf("%d", &num);
```

```
    printf("Factorial of %d is %d\n", num, factorial(num));
    return 0;
}
```

Fibonacci Series

```
#include <stdio.h>
void generateFibonacci(int n) {
    int a = 0, b = 1, next;
    printf("Fibonacci Series: ");
    for (int i = 0; i < n; i++) {
        printf("%d ", a);
        next = a + b;
        a = b;
        b = next;
    }
    printf("\n");
}
int main() {
    int n;
    printf("Enter the number of terms: ");
    scanf("%d", &n);
    if (n <= 0) {
        printf("Please enter a positive number.\n");
    } else {
        generateFibonacci(n);
    }
```

```
   return 0;
}
```

#Day 3

Functions and Arrays

Ex-1

```
#include <stdio.h>
int main(){
int number [5] = {10,20,30,40,50};
printf("%d", number[2]);
return 0;
}
```

Ex-2

```
#include <stdio.h>
int add(int a, int b) {
return a + b;
}
int main() {
int result = add(5, 10);
printf("Sum = %d\n", result);
return 0;
}
```

Function declaration

```
int multiply(int x, int y);
int main() {
```

```
int result = multiply(4, 5);

printf("Product = %d\n", result);

return 0;
}
```

EX-2

```
int multiply(int x, int y) {

return x * y;
}

#include <stdlib.h>

#include <stdio.h>

// chandansb

int main() {
    int n, *arr;


    printf("Enter the number of elements: ");

    scanf("%d", &n);


    arr = (int *)malloc(n * sizeof(int));

    if (arr == NULL) {

        printf("Memory allocation failed");

        return 1;

    }


    printf("Enter %d elements:\n", n);

    for (int i = 0; i < n; i++) {

        scanf("%d", &arr[i]);
```

```
}
    printf("The elements are:\n");
    for (int i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }

    free(arr);
    return 0;
}
```

Memory allocation 2D

```
#include <stdio.h>
#include <stdlib.h>
int main() {
   int rows = 2, cols = 3;
   int **matrix = (int **)malloc(rows * sizeof(int *));
   if (matrix == NULL) {
      printf("Memory allocation failed.\n");
      return 1;
   }
   for (int i = 0; i < rows; i++) {
      matrix[i] = (int *)malloc(cols * sizeof(int));
      if (matrix[i] == NULL) {
         printf("Memory allocation failed for row %d.\n", i);
         return 1;
      }
   }
```

```
  printf("Enter values for a 2x3 matrix:\n");
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            scanf("%d", &matrix[i][j]);
        }
    }
    printf("Matrix:\n");
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            printf("%d ", matrix[i][j]);
        }
        printf("\n");
    }
    for (int i = 0; i < rows; i++) {
        free(matrix[i]);
    }
    free(matrix);

    return 0;
}
```

Memory allocation for integer

```
#include <stdio.h>
#include <stdlib.h>

int main() {
    int *num = (int *)malloc(sizeof(int));
```

```
if (num == NULL) {
        printf("Memory allocation failed.\n");
        return 1;
    }
    *num = 42;
    printf("Value: %d\n", *num);
    free(num);
    return 0;
}
```

#Day 4

Strings, Pointers, and Dynamic Memory Allocation

Basic Pointer

```
#include <stdio.h>
int main() {
    int num = 10;
    int *ptr = &num;

    printf("Value of num: %d\n", num);
    printf("Address of num: %p\n", &num);
    printf("Pointer pointing to: %p\n", ptr);
    printf("Value at pointer: %d\n", *ptr);

    return 0;
}
```

Pointer and array

```
#include <stdio.h>

int main() {
   int arr[] = {10, 20, 30, 40, 50};
   int *ptr = arr;


   printf("Array elements using pointer:\n");
   for (int i = 0; i < 5; i++) {
      printf("arr[%d] = %d\n", i, *(ptr + i));
   }


   return 0;
}
```

Pointer arthamatic

```
#include <stdio.h>
// CHANDAN S B
int main() {
   int arr[] = {10, 20, 30, 40, 50};
   int *ptr = arr;


   printf("%d\n", *ptr);
   ptr++;
   printf("%d\n", *ptr);
   return 0;
}
```

Ex-1

```
#include <stdio.h>
int main() {
   int x = 10;
   int *ptr = &x;
   printf("Address of x: %p\n", (void *)ptr);
   printf("Value of x: %d\n", *ptr);
   return 0;
}
```

Palindrome number

```
#include <stdio.h>
int main() {
   int num, reversedNum = 0, remainder, originalNum;
   // chandan sb
   printf("Enter an integer: ");
   scanf("%d", &num);
   originalNum = num;
   while (num != 0) {
      remainder = num % 10;
      reversedNum = reversedNum * 10 + remainder;
      num /= 10;
   }
   if (originalNum == reversedNum) {
      printf("%d is a palindrome number.\n", originalNum);
   } else {
      printf("%d is not a palindrome number.\n", originalNum);
```

```
}
   return 0;
}
```

#Day 5

Structures, File Handling, and Preprocessor Directives

Defining Structures

```
#include <stdio.h>
// defining stricture
//Chndan s B
struct student{
char name[50];
int roll;
float marks;
};
int main(){
 struct student s1;
 printf("Enter the name");
 scanf("%s",s1.name);
 printf("Enter the roll mumber");
 scanf("%d",&s1.roll);
 printf("Enter the marks");
 scanf("%f",&s1.marks);
 printf("\nstudent Details");
 printf("Name:%s\nRoll:%d\nMarks:%.2f",s1.name,s1.roll,s1.marks);
 return 0; }
```

Emplyment salary

```
#include <stdio.h>
struct Employee {
   char name[50];
   int id;
   float salary;
};
int main() {
   struct Employee employees[2];
   for (int i = 0; i < 2; i++) {
      printf("Enter the details of employee %d:\n", i + 1);
      printf("Name: ");
      scanf("%s", employees[i].name);
      printf("ID: ");
      scanf("%d", &employees[i].id);
      printf("Salary: ");
      scanf("%f", &employees[i].salary);
   }
   printf("\nEmployee Details:\n");
   for (int i = 0; i < 2; i++) {
      printf("Name: %s, ID: %d, Salary: %.2f\n", employees[i].name, employees[i].id,
employees[i].salary);
   }
   return 0;
}
```

Age,Name,Gender

```
#include <stdio.h>
struct User {
   char name[50];
   int age;
   char gender[10];
   char branch[50];
   char college[100];
};
int main() {
   struct User user1;
   printf("Enter Name: ");
   fgets(user1.name, 50, stdin);
   printf("Enter Age: ");
   scanf("%d", &user1.age);
   getchar();
   printf("Enter Gender: ");
   fgets(user1.gender, 10, stdin);
   printf("Enter Branch: ");
   fgets(user1.branch, 50, stdin);
   printf("Enter College: ");
   fgets(user1.college, 100, stdin);
   printf("\nUser Details:\n");
   printf("Name: %s", user1.name);
   printf("Age: %d\n", user1.age);
   printf("Gender: %s", user1.gender);
```

```
    printf("Branch: %s", user1.branch);
    printf("College: %s", user1.college);
    return 0;
}
```

Opening a File

```
FILE *file_pointer = fopen("filename", "mode");
```

Modes: "r" (read), "w" (write), "a" (append).

Writing to a File

```
#include <stdio.h>
int main() {
FILE *fp = fopen("data.txt", "w");
if (fp == NULL) {
printf("Error opening file!\n");
return 1;
}
fprintf(fp, "Hello, File Handling!\n");
fclose(fp);
printf("Data written to file successfully.\n");
return 0;
}
```

Checking Prime Number

```
#include <stdio.h>
int main() {
    int n, i, isPrime = 1;
    printf("Enter a number: ");
```

```
    scanf("%d", &n);
    for (i = 2; i <= n / 2; i++) {
        if (n % i == 0) {
            isPrime = 0;
            break;
        }
    }
    if (n < 2) isPrime = 0;
    if (isPrime)
        printf("Prime\n");
    else
        printf("Not Prime\n");
    return 0;
}
```

#Day 6

Advanced Concepts, Debugging, and Final Project

Final Project

Implement a simple Snake game using arrays and logic for movement.

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <conio.h> // For getch()
#define BOARD_WIDTH 20
```

```
#define BOARD_HEIGHT 10

// Structure to represent a snake segment

typedef struct {

   int x;

   int y;

} Segment;

// Function prototypes

void initializeBoard(char board[BOARD_HEIGHT][BOARD_WIDTH]);

void printBoard(char board[BOARD_HEIGHT][BOARD_WIDTH]);

void placeFood(char board[BOARD_HEIGHT][BOARD_WIDTH]);

void moveSnake(char board[BOARD_HEIGHT][BOARD_WIDTH], Segment *snake, int
*snakeLength, int direction, int *gameRunning);

int main() {

   char board[BOARD_HEIGHT][BOARD_WIDTH];

   Segment snake[BOARD_WIDTH * BOARD_HEIGHT]; // Allocate enough space for the
snake

   int snakeLength = 1;

   int gameRunning = 1;

   int direction = 0; // 0: Right, 1: Up, 2: Left, 3: Down

   // Initialize the board and place the snake

   initializeBoard(board);

   snake[0].x = BOARD_WIDTH / 2;

   snake[0].y = BOARD_HEIGHT / 2;

   board[snake[0].y][snake[0].x] = '*'; // Place the snake on the board

   placeFood(board);

   // Game loop
```

```
    while (gameRunning) {
        printBoard(board);
        // Get user input
        if (kbhit()) {
            char input = getch();
            switch (input) {
                case 'w': direction = 1; break;
                case 's': direction = 3; break;
                case 'a': direction = 2; break;
                case 'd': direction = 0; break;
            }
        }
        // Move the snake
        moveSnake(board, snake, &snakeLength, direction, &gameRunning);
        // Add a delay (adjust for difficulty)
        Sleep(100);
    }
    printf("Game Over!\n");
    return 0;
}
void initializeBoard(char board[BOARD_HEIGHT][BOARD_WIDTH]) {
    for (int i = 0; i < BOARD_HEIGHT; i++) {
        for (int j = 0; j < BOARD_WIDTH; j++) {
            board[i][j] = ' '; // Initialize with empty space
        }
```

```
    }
}
void printBoard(char board[BOARD_HEIGHT][BOARD_WIDTH]) {
    system("cls"); // Clear the console (Windows)
    for (int i = 0; i < BOARD_HEIGHT; i++) {
        for (int j = 0; j < BOARD_WIDTH; j++) {
            printf("%c", board[i][j]);
        }
        printf("\n");
    }
}
void placeFood(char board[BOARD_HEIGHT][BOARD_WIDTH]) {
    int x, y;
    do {
        x = rand() % BOARD_WIDTH;
        y = rand() % BOARD_HEIGHT;
    } while (board[y][x] != ' '); // Ensure food is placed on an empty cell
    board[y][x] = 'O';
}
void moveSnake(char board[BOARD_HEIGHT][BOARD_WIDTH], Segment *snake, int
*snakeLength, int direction, int *gameRunning) {
    // Calculate new head position
    int newX = snake[0].x;
    int newY = snake[0].y;
    switch (direction) {
```

```
        case 0: newX++; break; // Right
        case 1: newY--; break; // Up
        case 2: newX--; break; // Left
        case 3: newY++; break; // Down
    }
    // Check for collisions with walls or self
    if (newX < 0 || newX >= BOARD_WIDTH || newY < 0 || newY >= BOARD_HEIGHT ||
board[newY][newX] == '*') {
        *gameRunning = 0; // End the game
        return;
    }
    // Check if food is eaten
    int foodEaten = (board[newY][newX] == 'O');
    // Move the snake (shift segments)
    for (int i = *snakeLength - 1; i > 0; i--) {
        snake[i] = snake[i - 1];
    }
    snake[0].x = newX;
    snake[0].y = newY;
    // Update the board
    board[newY][newX] = '*'; // New head position
    if (!foodEaten) {
        // Clear the tail segment
        board[snake[*snakeLength - 1].y][snake[*snakeLength - 1].x] = ' ';
    } else {
```

```
        (*snakeLength)++; // Grow the snake
        placeFood(board); // Place new food
    }
Retrun 0;
}
```

Key Points

Snake Movement:
The moveSnake function calculates the new head position based on the direction and shifts all segments forward.
2. Collision Detection:
Checks if the snake collides with the walls or itself, ending the game if a collision occurs.
3. Eating Food:
If the snake's head moves onto a food cell (O), the snake grows, and new food is placed randomly.
4. Dynamic Snake Length:
The snake grows in length when food is eaten, and its tail is updated correctly.
5. Game Over:
The game ends if the snake collides with itself or the walls, displaying "Game Over!"

Example Output

#When running the program, you'll see something like this with movement.

-Thank you