Government Polytechnic Kanpur

Department of [DIPLIMA IN ELECTRONICS]

LAB MANUAL

PROGRAMING IN C

Lab Title: [**Programing in C Lab Manual**]
Subject: [**Programing in C -2268**]
Course: [Diploma in Electronics]
Semester: [5th Semester)]
Session: [2024-2025]
Lab Instructor: [**Mr. Himanshu Singh**]

Submitted by:
Name:_____
Roll Number:_____
EnRollment Number:_____
Class: [ELECTRONICS 5TH Semester]

INDEX

SR NO	DATE	OBJECTIVE	PAGE NO	REMARKS
1.		**Objective 1:**		
2.		**Objective 2:**		
3.		**Objective 3:**		
4.		**Objective 4:**		
5.		**Objective 5:**		
6.		**Objective 6:**		
7.		**Objective 7:**		
8.		**Objective 8:**		
9.		**Objective 9:**		
10.		**Objective 10:**		
11.		**Objective 11:**		
12.		**Objective 12:**		
13.		**Objective 13:**		
14.		**Objective 14:**		
15.		**Objective 15:**		
16.		**Objective 16:**		

the students will be able to:

- **understand the concepts of C programming language**
- **install C software on the system and debug the programme**
- **identify a problem and formulate an algorithm for it.**
- **identify various control structures and implement them.**
- **identify various types of variables. • use pointer in an array and structure.**
- **implement the language control structure**
- **understand and execute member functions of C in the programme**
- **implement array concept in C programme**
- **execute pointers**

DETAILED CONTENTS

THEORY SYLLABUS:

UNIT 1:

ALGORITHM AND PROGRAMMING DEVELOPMENT

1. Steps in development of a program.
2. Flow charts, Algorithm development.
3. Programm debugging.
4. Basic of C programming.

UNIT 2:

PROGRAM STRUCTURE

1. I/O statements assign statements.
2. Constants, variable and data types.
3. Operators and expressions.
4. Standards and formatted IOS.
5. Data type Casting.

UNIT 3:

CONTROL STRUCTURES

1. Introduction.
2. Decision making with IF- statement.
3. If else and nested IF.
4. While and do while, for loop.
5. Break, Continue, goto and switch statements.

UNIT4:

POINTERS

1. Introduction to Pointers.
2. Address operator and pointers.
3. Declaring and initializing pointers.
4. Single pointer.

UNIT 5:

FUNCTIONS

1. **Introduction to functions.**
2. **Global and local variables.**
3. **Function declaration.**
4. **Standard functions.**
5. **Parameters and parameter passing.**
6. **Call by value/reference.**
7. **Recursion.**

UNIT 6:

ARRAYS

1. **Introduction to arrays.**
2. **Array declaration , length of array.**
3. **Single and multidimensional array.**
4. **Array of characters.**
5. **Passing an array to function.**
6. **Pointers to an array.**

LIST OF PRACTICALS

1. **Programming exercises on executing and editing a C program.**
2. **Programming exercises on defining variables and assigning values to variables.**
3. **Programming exercises on arithmetic and relational operators.**
4. **Programming exercises on arithmetic expressions and their evaluation.**
5. **Programming exercises on formatting input/output using printf and scanf and their Return type values.**
6. **Programming exercises using if statement.**
7. **Programming exercises using if – Else.**
8. **Programming exercises on switch statement.**
9. **Programming exercises on do – while, statement.**
10. **Programming exercises on for – statement.**
11. **Programs on one-dimensional array.**
12. **Programs on two-dimensional array**
13. **(i) Programs for putting two strings together.**
    (ii) Programs for comparing two strings.
14. **Simple programs using structures.**
15. **Simple programs using pointers.**
16. **Simple programs using union.**

RECOMMENDED BOOKS

1. **Let us C by Yashwant Kanetkar**
2. **Programming in ANSI C by E Balaguruswami,**

LINKS:

Learn C Programming Language Tutorial - javatpoint
C Programming Language Tutorial - GeeksforGeeks
C Tutorial (tutorialspoint.com)
C Tutorial (w3schools.com)

PROGRAMMING IN C SYLLABUS:

Microsoft Word - 330 NSQF NITTR (bteup.ac.in)

LIST OF PRACTICALS

1. Programming exercises on executing and editing a C program.
2. Programming exercises on defining variables and assigning values to variables.
3. Programming exercises on arithmetic and relational operators.
4. Programming exercises on arithmetic expressions and their evaluation.
5. Programming exercises on formatting input/output using printf and scanf and their return type values.
6. Programming exercises using if statement.
7. Programming exercises using if – Else.
8. Programming exercises on switch statement.
9. Programming exercises on do – while, statement.
10. Programming exercises on for – statement.
11. Programs on one-dimensional array.

12. Programs on two-dimensional array.
13. (i) Programs for putting two strings together.
    (ii) Programs for comparing two strings.
14. Simple programs using structures.
15. Simple programs using pointers.
16. Simple programs using union.

Lab Manual: Programming Exercises on Executing and Editing a C Program

Objective:

To gain hands-on experience with executing and editing C programs.

By the end of these exercises, students will have learned the basic structure of a C program,

how to compile and execute it, and how to modify existing C code.

Exercise 1: Writing, Compiling, and Running a Simple C Program

Task:

Write a simple C program that prints "Hello, World!" to the console,

compile it using the GCC compiler, and execute the compiled program.

Instructions:

1. Open a terminal on your system or IDE with a C compiler (GCC).
2. Create a new file named hello.c.
3. Write the following code in the file:

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

4. Save the file.
5. Compile the program using the GCC compiler by typing:
   gcc hello.c -o hello

6. Run the executable by typing:
   ./hello

Expected Output:
Copy code
Hello, World!

Exercise 2: Modifying the Program

Task:

Modify the hello.c program to print the following information:

1. Your name.
2. Your age.
3. Your favorite programming language.

Instructions:

1. Edit hello.c to print the required information. For example:
   ```
   #include <stdio.h>
   int main() {
      printf("Hello, my name is [Your Name].\n");
      printf("I am [Your Age] years old.\n");
      printf("My favorite programming language is C.\n");
      return 0;
   }
   ```
2. Save the file.
3. Compile and run the program using the commands:
   ```
   gcc hello.c -o hello
   ./hello
   ```


Expected Output:

Hello, my name is [Your Name].
I am [Your Age] years old.
My favorite programming language is C.

Exercise 3: Adding User Input

Task:

Modify the program to take input from the user for their name and age, and then print the information back.

Instructions:

1. Update the code to use scanf for taking input:

```
#include <stdio.h>
int main() {
    char name[50];
    int age;

    printf("Enter your name: ");
    scanf("%s", name);
    printf("Enter your age: ");
    scanf("%d", &age);
    printf("Hello, %s. You are %d years old.\n", name, age);
    return 0;
}
```


2. Save the file.
3. Compile and run the program again.

Expected Output:
Enter your name: [User Input]
Enter your age: [User Input]
Hello, [User Input]. You are [User Input] years old.

Exercise 4: Adding Conditional Statements

Task:

Modify the program to check if the user's age is 18 or above. If so, print "You are an adult." Otherwise, print "You are a minor."

Instructions:

1. Add an if condition in the main function:

```
#include <stdio.h>
int main() {
    char name[50];
    int age;

    printf("Enter your name: ");
    scanf("%s", name);

    printf("Enter your age: ");
    scanf("%d", &age);

    if (age >= 18) {
        printf("Hello, %s. You are an adult.\n", name);
    } else {
        printf("Hello, %s. You are a minor.\n", name);
    }

    return 0;
}
```

2. Save, compile, and run the program.

Expected Output:
Enter your name: [User Input]
Enter your age: [User Input]
Hello, [User Input]. You are an adult.
Hello, [User Input]. You are a minor.

Exercise 5: Using Loops

Task:

Write a program that prints numbers from 1 to 10 using a loop.

Instructions:

1. Create a new file named loop.c.
2. Write the following code:

```
#include <stdio.h>
int main() {
    int i;

    for (i = 1; i <= 10; i++) {
        printf("%d\n", i);
    }

    return 0;
}
```

3. Save, compile, and run the program.

Expected Output:

```
1
2
3
4
5
6
7
8
9
10
```

Lab Manual: Programming Exercises in C on Defining Variables and Assigning Values

Objective:

The goal of this lab is to understand how to define and assign values to variables in C. You will learn about different data types, how to declare variables, and how to perform basic operations on variables.

Exercise 1: Defining Variables and Printing Their Values

Task:

Define different types of variables (integer, float, and char), assign values to them, and print their values.

Instructions:

1. Create a new file called `variables.c`.
2. Write the following code:

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



```
int main() {
    int myInt = 25;          // Integer variable
    float myFloat = 12.34;   // Float variable
    char myChar = 'A';       // Character variable

    printf("Integer: %d\n", myInt);
    printf("Float: %.2f\n", myFloat);
    printf("Character: %c\n", myChar);

    return 0;
}
```

3. Save the file.
4. Compile the program using GCC:

```
bash
Copy code
gcc variables.c -o variables
```

5. Run the program:

```
./variables
```

Expected Output:

```
makefile
```

```
Integer: 25
Float: 12.34
Character: A
```

Exercise 2: Changing Variable Values

Task:

Change the values of the variables after they have been initialized and print the updated values.

Instructions:

1. Modify the previous program to change the values of `myInt`, `myFloat`, and `myChar` after they have been declared:

```
#include <stdio.h>

int main() {
    int myInt = 25;
    float myFloat = 12.34;
    char myChar = 'A';

    // Print initial values
    printf("Initial Integer: %d\n", myInt);
    printf("Initial Float: %.2f\n", myFloat);
    printf("Initial Character: %c\n", myChar);

    // Change variable values
    myInt = 50;
    myFloat = 24.68;
    myChar = 'B';

    // Print updated values
    printf("Updated Integer: %d\n", myInt);
    printf("Updated Float: %.2f\n", myFloat);
    printf("Updated Character: %c\n", myChar);

    return 0;
}
```


2. Save, compile, and run the program again.

Expected Output:

```
Initial Integer: 25
```

```
Initial Float: 12.34
Initial Character: A
Updated Integer: 50
Updated Float: 24.68
Updated Character: B
```

Exercise 3: Arithmetic Operations on Variables

Task:

Perform basic arithmetic operations (addition, subtraction, multiplication, and division) on integer and float variables.

Instructions:

1. Create a new file named `arithmetic.c`.
2. Write the following code to perform arithmetic operations:

```
#include <stdio.h>

int main() {
    int a = 10;
    int b = 5;
    float x = 5.5;
    float y = 2.0;

    // Perform arithmetic on integers
    printf("Addition of a + b = %d\n", a + b);
    printf("Subtraction of a - b = %d\n", a - b);
    printf("Multiplication of a * b = %d\n", a * b);
    printf("Division of a / b = %d\n", a / b);

    // Perform arithmetic on floats
    printf("Addition of x + y = %.2f\n", x + y);
    printf("Subtraction of x - y = %.2f\n", x - y);
    printf("Multiplication of x * y = %.2f\n", x * y);
    printf("Division of x / y = %.2f\n", x / y);

    return 0;
}
```


3. Save, compile, and run the program.

Expected Output:

```
Addition of a + b = 15
Subtraction of a - b = 5
Multiplication of a * b = 50
Division of a / b = 2
Addition of x + y = 7.50
Subtraction of x - y = 3.50
Multiplication of x * y = 11.00
Division of x / y = 2.75
```

Exercise 4: Declaring Multiple Variables

Task:

Declare and initialize multiple variables of the same type in a single statement and perform arithmetic operations on them.

Instructions:

1. Modify the previous code to declare multiple variables in one line:

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



```
int main() {
    int a = 10, b = 20, c = 30;
    float x = 1.1, y = 2.2, z = 3.3;

    // Perform arithmetic on integers
    printf("Sum of a, b, and c = %d\n", a + b + c);

    // Perform arithmetic on floats
    printf("Product of x, y, and z = %.2f\n", x * y * z);

    return 0;
}
```

2. Save, compile, and run the program.

Expected Output:

```
Sum of a, b, and c = 60
Product of x, y, and z = 7.99
```

Exercise 5: Working with Constants

Task:

Declare and use constant variables using the `const` keyword. Attempt to modify the value of a constant and observe the result.

Instructions:

1. Create a new file called `constants.c`.
2. Write the following code to declare constants:

```
#include <stdio.h>

int main() {
    const int MY_CONST_INT = 100;
    const float MY_CONST_FLOAT = 3.14;

    printf("Constant Integer: %d\n", MY_CONST_INT);
    printf("Constant Float: %.2f\n", MY_CONST_FLOAT);

    // Uncomment the following lines and observe the error
    // MY_CONST_INT = 200;
    // MY_CONST_FLOAT = 6.28;

    return 0;
}
```


3. Save, compile, and run the program.

Expected Output:

```
mathematica
Copy code
Constant Integer: 100
Constant Float: 3.14
```

Lab Manual: Programming Exercises in C on Arithmetic and Relational Operators

Objective:

The goal of this lab is to understand how to use arithmetic and relational operators in C. You will perform mathematical calculations using arithmetic operators and compare values using relational operators.

Exercise 1: Using Arithmetic Operators

Task:

Write a C program to perform basic arithmetic operations: addition, subtraction, multiplication, division, and modulus.

Instructions:

1. Create a new file called `arithmetic_operations.c`.
2. Write the following code:

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



```
int main() {
    int a = 20;
    int b = 7;

    printf("Addition: %d + %d = %d\n", a, b, a + b);
    printf("Subtraction: %d - %d = %d\n", a, b, a - b);
    printf("Multiplication: %d * %d = %d\n", a, b, a * b);
    printf("Division: %d / %d = %d\n", a, b, a / b);
    printf("Modulus: %d %% %d = %d\n", a, b, a % b);

    return 0;
}
```

3. Save the file.
4. Compile the program using the following command:

```
gcc arithmetic_operations.c -o arithmetic_operations
```

5. Run the program:

```
bash
Copy code
./arithmetic_operations
```

Expected Output:

```
makefile
Addition: 20 + 7 = 27
Subtraction: 20 - 7 = 13
Multiplication: 20 * 7 = 140
Division: 20 / 7 = 2
Modulus: 20 % 7 = 6
```

Exercise 2: Performing Operations on Float Variables

Task:

Modify the program to use floating-point numbers and observe the results of division and modulus operations.

Instructions:

1. Modify the previous code to use `float` instead of `int` for `a` and `b`:

```
#include <stdio.h>

int main() {
    float a = 20.5;
    float b = 7.3;

    printf("Addition: %.2f + %.2f = %.2f\n", a, b, a + b);
    printf("Subtraction: %.2f - %.2f = %.2f\n", a, b, a - b);
    printf("Multiplication: %.2f * %.2f = %.2f\n", a, b, a *
b);
    printf("Division: %.2f / %.2f = %.2f\n", a, b, a / b);

    return 0;
}
```


2. Save, compile, and run the program.

Expected Output:

```
Addition: 20.50 + 7.30 = 27.80
Subtraction: 20.50 - 7.30 = 13.20
Multiplication: 20.50 * 7.30 = 149.65
Division: 20.50 / 7.30 = 2.81
```

Exercise 3: Using Relational Operators

Task:

Write a C program to compare two integer variables using relational operators and print the results.

Instructions:

1. Create a new file named `relational_operators.c`.
2. Write the following code:

```
#include <stdio.h>

int main() {
    int x = 10;
    int y = 20;

    printf("x == y: %d\n", x == y);   // Equal to
    printf("x != y: %d\n", x != y);   // Not equal to
    printf("x > y: %d\n", x > y);     // Greater than
    printf("x < y: %d\n", x < y);     // Less than
    printf("x >= y: %d\n", x >= y);   // Greater than or equal to
    printf("x <= y: %d\n", x <= y);   // Less than or equal to

    return 0;
}
```


3. Save the file.
4. Compile the program using the following command:

```
gcc relational_operators.c -o relational_operators
```

5. Run the program:

```
./relational_operators
```

Expected Output:

```
x == y: 0
x != y: 1
x > y: 0
x < y: 1
x >= y: 0
x <= y: 1
```

Exercise 4: Combining Arithmetic and Relational Operators

Task:

Write a program that uses arithmetic operators to perform calculations and relational operators to compare the results.

Instructions:

1. Create a new file named `arithmetic_relational.c`.
2. Write the following code:

```
#include <stdio.h>

int main() {
    int a = 15, b = 10, c = 5;

    // Perform arithmetic operations
    int sum = a + b;
    int product = a * c;

    // Use relational operators to compare results
    printf("Sum of a and b > product of a and c: %d\n", sum > product);
    printf("Sum of a and b == product of a and c: %d\n", sum == product);
    printf("Sum of a and b < product of a and c: %d\n", sum < product);

    return 0;
}
```


3. Save, compile, and run the program.

Expected Output:

```
Sum of a and b > product of a and c: 0
Sum of a and b == product of a and c: 1
Sum of a and b < product of a and c: 0
```

Exercise 5: Implementing a Simple Calculator

Task:

Write a program that asks the user for two numbers and a mathematical operation (addition, subtraction, multiplication, or division), performs the operation, and displays the result.

Instructions:

1. Create a new file called `simple_calculator.c`.
2. Write the following code:

```
#include <stdio.h>

int main() {
    float num1, num2;
    char operator;

    // Ask user for input
    printf("Enter first number: ");
    scanf("%f", &num1);

    printf("Enter an operator (+, -, *, /): ");
    scanf(" %c", &operator); // Note the space before %c to
handle newline character

    printf("Enter second number: ");
    scanf("%f", &num2);

    // Perform the chosen operation
    switch (operator) {
        case '+':
            printf("%.2f + %.2f = %.2f\n", num1, num2, num1 +
num2);
            break;
        case '-':
            printf("%.2f - %.2f = %.2f\n", num1, num2, num1 -
num2);
            break;
        case '*':
            printf("%.2f * %.2f = %.2f\n", num1, num2, num1 *
num2);
            break;
        case '/':
            if (num2 != 0)
```

```
                    printf("%.2f / %.2f = %.2f\n", num1, num2,
    num1 / num2);
                else
                    printf("Error: Division by zero is not
    allowed.\n");
                break;
            default:
                printf("Error: Invalid operator\n");
        }

        return 0;
    }
```

3. Save, compile, and run the program.

Expected Output (Example for user inputs: 5, +, and 3):

```
Enter first number: 5
Enter an operator (+, -, *, /): +
Enter second number: 3
5.00 + 3.00 = 8.00
```

Lab Manual: Programming Exercises in C on Arithmetic Expressions and Their Evaluation

Objective:

The aim of this lab is to learn how to write and evaluate arithmetic expressions in C. You will use different operators to create complex expressions and understand the precedence and associativity of these operators.

Exercise 1: Basic Arithmetic Expressions

Task:

Write a program that evaluates simple arithmetic expressions involving addition, subtraction, multiplication, division, and modulus.

Instructions:

1. Create a new file called `basic_arithmetic.c`.
2. Write the following code:


```
#include <stdio.h>
int main() {
    int a = 10, b = 5, c = 2;
    int result;

    // Simple arithmetic operations
    result = a + b - c;
    printf("Result of a + b - c = %d\n", result);

    result = a * b / c;
    printf("Result of a * b / c = %d\n", result);

    result = a % b + c;
    printf("Result of a %% b + c = %d\n", result);

    return 0;
}
```

3. Save the file.
4. Compile the program:
   bash
   Copy code

```
gcc basic_arithmetic.c -o basic_arithmetic
```

5. Run the program:
   ```
   ./basic_arithmetic
   ```

Expected Output:

```
Result of a + b - c = 13
Result of a * b / c = 25
Result of a % b + c = 2
```

Exercise 2: Operator Precedence and Associativity

Task:

Write a program to evaluate arithmetic expressions that include multiple operators, and observe how operator precedence and associativity affect the result.

Instructions:

1. Create a new file named `operator_precedence.c`.
2. Write the following code:


```
#include <stdio.h>

int main() {
    int a = 10, b = 5, c = 2;
    int result;

    // Expression with mixed operators
    result = a + b * c;
    printf("Result of a + b * c = %d\n", result);

    // Expression with parentheses to change precedence
    result = (a + b) * c;
    printf("Result of (a + b) * c = %d\n", result);

    result = a + b / c;
    printf("Result of a + b / c = %d\n", result);

    result = (a + b) / c;
    printf("Result of (a + b) / c = %d\n", result);

    return 0;
```

```
}
```

3. Save, compile, and run the program.

Expected Output:

```
css
Copy code
Result of a + b * c = 20
Result of (a + b) * c = 30
Result of a + b / c = 12
Result of (a + b) / c = 7
```

Exercise 3: Working with Floating-Point Arithmetic

Task:

Write a program that evaluates arithmetic expressions using floating-point numbers and observe the differences compared to integer arithmetic.

Instructions:

1. Create a new file called `float_arithmetic.c`.
2. Write the following code:

```
#include <stdio.h>

int main() {
    float x = 10.5, y = 5.2, z = 2.0;
    float result;

    result = x + y - z;
    printf("Result of x + y - z = %.2f\n", result);

    result = x * y / z;
    printf("Result of x * y / z = %.2f\n", result);

    result = x / y + z;
    printf("Result of x / y + z = %.2f\n", result);

    return 0;
}
```

3. Save, compile, and run the program.

Expected Output:

```
Result of x + y - z = 13.70
Result of x * y / z = 27.30
Result of x / y + z = 3.92
```

Exercise 4: Compound Arithmetic Expressions

Task:

Write a program that evaluates more complex arithmetic expressions involving multiple operators and parentheses.

Instructions:

1. Create a new file called `complex_expression.c`.
2. Write the following code:

```
#include <stdio.h>

int main() {
    int a = 8, b = 3, c = 6, d = 2;
    int result;

    // Evaluate a complex arithmetic expression
    result = (a + b) * (c - d) / b;
    printf("Result of (a + b) * (c - d) / b = %d\n", result);

    result = a + b - c * d / a;
    printf("Result of a + b - c * d / a = %d\n", result);

    return 0;
}
```


3. Save, compile, and run the program.

Expected Output:

```
Result of (a + b) * (c - d) / b = 10
Result of a + b - c * d / a = 7
```

Exercise 5: Evaluating Expressions from User Input

Task:

Write a program that asks the user to input values for variables and then evaluates an arithmetic expression using those values.

Instructions:

1. Create a new file named `user_input_expression.c`.
2. Write the following code:

```
#include <stdio.h>

int main() {
    int a, b, c;
    int result;

    // Ask user for input
    printf("Enter value for a: ");
    scanf("%d", &a);
    printf("Enter value for b: ");
    scanf("%d", &b);
    printf("Enter value for c: ");
    scanf("%d", &c);

    // Evaluate an arithmetic expression
    result = a * b + c;
    printf("Result of a * b + c = %d\n", result);

    result = (a + b) * c;
    printf("Result of (a + b) * c = %d\n", result);

    return 0;
}
```


3. Save, compile, and run the program.

Expected Output (Example for user inputs: 3, 4, 5):

```
rust
Copy code
Enter value for a: 3
Enter value for b: 4
Enter value for c: 5
Result of a * b + c = 17
```

```
Result of (a + b) * c = 35
```

Exercise 6: Implementing a Formula

Task:

Write a program to evaluate a formula using predefined values for variables.

Instructions:

1. Create a new file called `formula_evaluation.c`.
2. Write the following code to evaluate the formula:

   result=a+bc−d\text{result} = \frac{a + b}{c - d}result=c−da+b

```
#include <stdio.h>

int main() {
    float a = 10.0, b = 5.0, c = 8.0, d = 4.0;
    float result;

    // Evaluate the formula
    result = (a + b) / (c - d);
    printf("Result of (a + b) / (c - d) = %.2f\n", result);

    return 0;
}
```


3. Save, compile, and run the program.

Expected Output:

```
Result of (a + b) / (c - d) = 3.75
```

Lab Manual: Programming Exercises in C on Formatting Input/Output using `printf` and `scanf` and Their Return Type Values

Objective:

The goal of this lab is to practice formatting input and output in C using the `printf` and `scanf` functions. You will learn how to use format specifiers for different data types and understand the return type values of these functions.

Exercise 1: Basic Input and Output with `printf` and `scanf`

Task:

Write a C program that takes user input and displays the output using `printf` and `scanf`.

Instructions:

1. Create a new file called `basic_io.c`.
2. Write the following code:


```
#include <stdio.h>

int main() {
    int age;
    float height;
    char name[20];

    // Input values
    printf("Enter your name: ");
    scanf("%s", name);
    printf("Enter your age: ");
    scanf("%d", &age);
    printf("Enter your height (in meters): ");
    scanf("%f", &height);

    // Output values
    printf("\nName: %s\n", name);
    printf("Age: %d\n", age);
    printf("Height: %.2f meters\n", height);

    return 0;
}
```

3. Save, compile, and run the program.

Expected Output (Example Input: `John`, `25`, `1.75`):

```
Enter your name: John
Enter your age: 25
Enter your height (in meters): 1.75

Name: John
Age: 25
Height: 1.75 meters
```

Exercise 2: Using Format Specifiers in `printf`

Task:

Write a program that demonstrates the use of various format specifiers with `printf`.

Instructions:

1. Create a new file called `format_specifiers.c`.
2. Write the following code:


```
#include <stdio.h>

int main() {
    int a = 123;
    float b = 45.678;
    char ch = 'A';
    char str[] = "Hello, World!";

    // Different format specifiers
    printf("Integer: %d\n", a);
    printf("Float: %.2f\n", b);
    printf("Character: %c\n", ch);
    printf("String: %s\n", str);

    return 0;
}
```

3. Save, compile, and run the program.

Expected Output:

```
Integer: 123
Float: 45.68
Character: A
String: Hello, World!
```

Exercise 3: Handling Multiple Inputs and Outputs

Task:

Write a program that takes multiple inputs from the user and prints them using proper formatting.

Instructions:

1. Create a new file named `multiple_io.c`.
2. Write the following code:

```
#include <stdio.h>

int main() {
    int id;
    char name[30];
    float salary;

    // Take multiple inputs
    printf("Enter Employee ID: ");
    scanf("%d", &id);
    printf("Enter Employee Name: ");
    scanf("%s", name);
    printf("Enter Employee Salary: ");
    scanf("%f", &salary);

    // Print formatted output
    printf("\nEmployee Details:\n");
    printf("ID: %d\n", id);
    printf("Name: %s\n", name);
    printf("Salary: %.2f\n", salary);

    return 0;
}
```


3. Save, compile, and run the program.

Expected Output (Example Input: `101`, `Alice`, `55000.75`):

```
Enter Employee ID: 101
Enter Employee Name: Alice
Enter Employee Salary: 55000.75

Employee Details:
```

```
ID: 101
Name: Alice
Salary: 55000.75
```

Exercise 4: Return Values of `scanf` and `printf`

Task:

Write a program that demonstrates how `scanf` and `printf` return values can be used in C.

Instructions:

1. Create a new file called `return_values.c`.
2. Write the following code:

```
#include <stdio.h>

int main() {
    int x;
    int result_scanf;
    int result_printf;

    // Input with scanf and check return value
    printf("Enter an integer: ");
    result_scanf = scanf("%d", &x);

    // Output with printf and check return value
    result_printf = printf("You entered: %d\n", x);

    // Display return values of scanf and printf
    printf("Return value of scanf: %d\n", result_scanf);
    printf("Return value of printf: %d\n", result_printf);

    return 0;
}
```


3. Save, compile, and run the program.

Expected Output (Example Input: `42`):

```
Enter an integer: 42
You entered: 42
Return value of scanf: 1
Return value of printf: 14
```

Explanation:

- `scanf` returns the number of inputs successfully read. In this case, 1 input (`%d`) was read, so the return value is 1.
- `printf` returns the number of characters printed. In this case, it prints "You entered: 42\n", which is 14 characters.

Exercise 5: Formatting Output with Width and Precision

Task:

Write a program to format output with specified width and precision using `printf`.

Instructions:

1. Create a new file called `formatting_output.c`.
2. Write the following code:

```
#include <stdio.h>

int main() {
    int num = 123;
    float pi = 3.14159;

    // Format output with width and precision
    printf("Integer with width 5: %5d\n", num);
    printf("Float with 2 decimal places: %.2f\n", pi);
    printf("Float with width 8 and 3 decimal places: %8.3f\n",
pi);

    return 0;
}
```


3. Save, compile, and run the program.

Expected Output:

```
Integer with width 5:   123
Float with 2 decimal places: 3.14
Float with width 8 and 3 decimal places:    3.142
```

Questions:

- What happens if the width specified is smaller than the number of digits in the integer?
- How can you format a floating-point number in scientific notation using `printf`?

Exercise 6: Handling Character Input/Output

Task:

Write a program that demonstrates how to input and output characters using `scanf` and `printf`.

Instructions:

1. Create a new file called `char_io.c`.
2. Write the following code:
   c
   Copy code
   ```
   #include <stdio.h>

   int main() {
       char ch;

       // Input a character
       printf("Enter a character: ");
       scanf(" %c", &ch);  // Space before %c to handle the newline character

       // Output the character
       printf("You entered: %c\n", ch);

       return 0;
   }
   ```
3. Save, compile, and run the program.

Expected Output (Example Input: `A`):

```
Enter a character: A
You entered: A
```

Lab Manual: Programming Exercises Using `if` Statement in C

Objective:

The objective of this lab is to practice using the `if` statement in C to control the flow of a program based on conditions. You will write programs that use the `if` statement to make decisions based on user input and variable values.

Exercise 1: Simple `if` Statement

Task:

Write a program that checks if a number entered by the user is positive.

Instructions:

1. Create a new file called `positive_check.c`.
2. Write the following code:


```
#include <stdio.h>
int main() {
    int num;

    // Input from user
    printf("Enter a number: ");
    scanf("%d", &num);

    // Check if the number is positive
    if (num > 0) {
        printf("The number is positive.\n");
    }

    return 0;
}
```

3. Save, compile, and run the program.

Expected Output:

```
csharp
Copy code
Enter a number: 5
```

```
The number is positive.
```

Exercise 2: `if-else` Statement

Task:

Write a program that checks if a number is even or odd.

Instructions:

1. Create a new file named `even_odd.c`.
2. Write the following code:

```
#include <stdio.h>

int main() {
    int num;

    // Input from user
    printf("Enter a number: ");
    scanf("%d", &num);

    // Check if the number is even or odd
    if (num % 2 == 0) {
        printf("The number is even.\n");
    } else {
        printf("The number is odd.\n");
    }

    return 0;
}
```

3. Save, compile, and run the program.

Expected Output (Example Input: `4`):

```
csharp
Copy code
Enter a number: 4
The number is even.
```

Exercise 3: Using `if` with Logical Operators

Task:

Write a program to check if a number is within the range of 1 to 100.

Instructions:

1. Create a new file called `range_check.c`.
2. Write the following code:

```
#include <stdio.h>

int main() {
    int num;

    // Input from user
    printf("Enter a number: ");
    scanf("%d", &num);

    // Check if the number is in the range of 1 to 100
    if (num >= 1 && num <= 100) {
        printf("The number is within the range of 1 to
100.\n");
    } else {
        printf("The number is outside the range of 1 to
100.\n");
    }

    return 0;
}
```

3. Save, compile, and run the program.

Expected Output (Example Input: `50`):

```
python
Copy code
Enter a number: 50
The number is within the range of 1 to 100.
```

Exercise 4: Nested `if` Statements

Task:

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

Instructions:

1. Create a new file called `number_check.c`.
2. Write the following code:

```
#include <stdio.h>

int main() {
    int num;

    // Input from user
    printf("Enter a number: ");
    scanf("%d", &num);

    // Check if the number is positive, negative, or zero
    if (num > 0) {
        printf("The number is positive.\n");
    } else {
        if (num < 0) {
            printf("The number is negative.\n");
        } else {
            printf("The number is zero.\n");
        }
    }

    return 0;
}
```


3. Save, compile, and run the program.

Expected Output (Example Input: `-10`):

```
Enter a number: -10
The number is negative.
```

Exercise 5: Using `if-else-if` Ladder

Task:

Write a program to grade a student based on marks entered by the user using the following rules:

- Marks >= 90: Grade A
- Marks >= 75: Grade B
- Marks >= 50: Grade C
- Marks < 50: Grade F

Instructions:

1. Create a new file called `grade.c`.
2. Write the following code:

```
#include <stdio.h>

int main() {
    int marks;

    // Input marks from user
    printf("Enter marks: ");
    scanf("%d", &marks);

    // Grading using if-else-if ladder
    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;
}
```


3. Save, compile, and run the program.

Expected Output (Example Input: 85):

```
Enter marks: 85
Grade: B
```

Exercise 6: Checking Leap Year

Task:

Write a program to check if a given year is a leap year.

Instructions:

1. Create a new file called `leap_year.c`.
2. Write the following code:

```
#include <stdio.h>

int main() {
    int year;

    // Input year from user
    printf("Enter a year: ");
    scanf("%d", &year);

    // Check if the year is a leap year
    if ((year % 4 == 0 && year % 100 != 0) || (year % 400 ==
0)) {
        printf("%d is a leap year.\n", year);
    } else {
        printf("%d is not a leap year.\n", year);
    }

    return 0;
}
```


3. Save, compile, and run the program.

Expected Output (Example Input: 2024):

```
Enter a year: 2024
2024 is a leap year.
```

Exercise 7: Checking Largest of Three Numbers

Task:

Write a program to find the largest of three numbers entered by the user.

Instructions:

1. Create a new file called `largest_number.c`.
2. Write the following code:

```
#include <stdio.h>

int main() {
    int a, b, c;

    // Input three numbers from user
    printf("Enter three numbers: ");
    scanf("%d %d %d", &a, &b, &c);

    // Find the largest number
    if (a > b && a > c) {
        printf("The largest number is %d\n", a);
    } else if (b > a && b > c) {
        printf("The largest number is %d\n", b);
    } else {
        printf("The largest number is %d\n", c);
    }

    return 0;
}
```


3. Save, compile, and run the program.

Expected Output (Example Input: 3, 5, 2):

```
Enter three numbers: 3 5 2
The largest number is 5
```

Exercise 8: Validating User Input

Task:

Write a program to check if a number entered by the user is within a valid range (e.g., 0 to 100). If not, display an error message.

Instructions:

1. Create a new file called `input_validation.c`.
2. Write the following code:

```
#include <stdio.h>

int main() {
    int num;

    // Input from user
    printf("Enter a number between 0 and 100: ");
    scanf("%d", &num);

    // Validate input
    if (num >= 0 && num <= 100) {
        printf("Valid input: %d\n", num);
    } else {
        printf("Error: Input out of range!\n");
    }

    return 0;
}
```


3. Save, compile, and run the program.

Expected Output (Example Input: `150`):

```
Enter a number between 0 and 100: 150
Error: Input out of range!
```

Lab Manual: Programming Exercises in C on Switch Statements

Objective

The goal of this lab is to practice using the `switch` statement in C to solve decision-making problems. By the end of the lab, students will be able to use the `switch` statement effectively for various programming challenges.

Exercise 1: Simple Calculator Using Switch

Problem Statement: Write a program that simulates a simple calculator performing addition, subtraction, multiplication, or division based on user input.

Instructions:

1. Prompt the user to enter two numbers.
2. Ask the user to choose an operation (`+`, `-`, `*`, `/`).
3. Use a `switch` statement to perform the selected operation.
4. Handle division by zero with an error message.

Sample Input:


```
Enter first number: 12
Enter second number: 4
Enter operator (+, -, *, /): *
```

Expected Output:

```
Result: 48
```

Code Outline:

```
c
Copy code
#include <stdio.h>

int main() {
    float num1, num2;
    char operator;

    printf("Enter first number: ");
    scanf("%f", &num1);
    printf("Enter second number: ");
    scanf("%f", &num2);
    printf("Enter operator (+, -, *, /): ");
    scanf(" %c", &operator);
```

```
    switch(operator) {
        case '+':
            printf("Result: %.2f\n", num1 + num2);
            break;
        case '-':
            printf("Result: %.2f\n", num1 - num2);
            break;
        case '*':
            printf("Result: %.2f\n", num1 * num2);
            break;
        case '/':
            if (num2 != 0)
                printf("Result: %.2f\n", num1 / num2);
            else
                printf("Error: Division by zero\n");
            break;
        default:
            printf("Invalid operator\n");
    }

    return 0;
}
```

Exercise 2: Day of the Week

Problem Statement: Write a program that takes a number (1-7) as input and prints the corresponding day of the week.

Instructions:

1. Use a `switch` statement to map numbers to days (1 = Monday, 2 = Tuesday, ..., 7 = Sunday).
2. If the input is outside the range of 1 to 7, display an error message.

Sample Input:
```
Enter a number (1-7): 5
```

Expected Output:
```
Friday
```

Code Outline:
```
#include <stdio.h>
```

```
int main() {
    int day;

    printf("Enter a number (1-7): ");
    scanf("%d", &day);

    switch(day) {
        case 1:
            printf("Monday\n");
            break;
        case 2:
            printf("Tuesday\n");
            break;
        case 3:
            printf("Wednesday\n");
            break;
        case 4:
            printf("Thursday\n");
            break;
        case 5:
            printf("Friday\n");
            break;
        case 6:
            printf("Saturday\n");
            break;
        case 7:
            printf("Sunday\n");
            break;
        default:
            printf("Invalid day\n");
    }

    return 0;
}
```

Exercise 3: Grade Evaluation

Problem Statement: Write a program that takes a letter grade as input (A, B, C, D, F) and displays the corresponding description.

Instructions:

1. Use a `switch` statement to display:
    - A = Excellent
    - B = Good
    - C = Fair
    - D = Poor
    - F = Fail
2. If an invalid grade is entered, display an error message.

Sample Input:

```
Enter your grade (A-F): B
```

Expected Output:

```
bash
Copy code
Good
```


Code Outline:

```
#include <stdio.h>

int main() {
    char grade;

    printf("Enter your grade (A-F): ");
    scanf(" %c", &grade);

    switch(grade) {
        case 'A':
            printf("Excellent\n");
            break;
        case 'B':
            printf("Good\n");
            break;
        case 'C':
            printf("Fair\n");
            break;
        case 'D':
            printf("Poor\n");
```

```
            break;
        case 'F':
            printf("Fail\n");
            break;
        default:
            printf("Invalid grade\n");
    }

    return 0;
}
```

Exercise 4: Simple Unit Converter

Problem Statement: Write a program that converts kilometers to meters, centimeters, and millimeters based on user input.

Instructions:

1. Prompt the user to enter a distance in kilometers.
2. Use a `switch` statement to select from the following options:
    - Convert to meters
    - Convert to centimeters
    - Convert to millimeters
3. Display the converted result.

Sample Input:

```
Enter distance in kilometers: 2.5
1. Convert to meters
2. Convert to centimeters
3. Convert to millimeters
Enter your choice: 2
```

Expected Output:

```
Distance in centimeters: 250000.00 cm
```

Code Outline:

```
c
Copy code
#include <stdio.h>

int main() {
    float km;
    int choice;
```

```
    printf("Enter distance in kilometers: ");
    scanf("%f", &km);

    printf("1. Convert to meters\n");
    printf("2. Convert to centimeters\n");
    printf("3. Convert to millimeters\n");
    printf("Enter your choice: ");
    scanf("%d", &choice);

    switch(choice) {
        case 1:
            printf("Distance in meters: %.2f m\n", km * 1000);
            break;
        case 2:
            printf("Distance in centimeters: %.2f cm\n", km *
100000);
            break;
        case 3:
            printf("Distance in millimeters: %.2f mm\n", km *
1000000);
            break;
        default:
            printf("Invalid choice\n");
    }

    return 0;
}
```

Exercise 5: Traffic Light Simulation

Problem Statement: Write a program that simulates a traffic light system. The user will input a color (red, yellow, green), and the program will display the corresponding action (stop, get ready, go).

Instructions:

1. Use a `switch` statement to map the traffic light colors to actions:
   - red = stop
   - yellow = get ready
   - green = go
2. Handle invalid color input with an error message.

Sample Input:
```
Enter traffic light color (red, yellow, green): green
```

Expected Output:
```
Go
```

Code Outline:


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

int main() {
    char color[10];

    printf("Enter traffic light color (red, yellow, green): ");
    scanf("%s", color);

    switch(color[0]) {
        case 'r':
            printf("Stop\n");
            break;
        case 'y':
            printf("Get ready\n");
            break;
        case 'g':
            printf("Go\n");
            break;
        default:
            printf("Invalid color\n");
    }
```

```
    return 0;
}
```

Lab Manual: Programming Exercises in C on `do-while` Statement

Objective

The objective of this lab is to gain hands-on experience using the `do-while` loop in C. The `do-while` loop executes the block of code at least once before checking the condition, making it useful for scenarios where the body of the loop must run at least once. By the end of this lab, students will be able to implement and utilize `do-while` loops effectively.

Exercise 1: Sum of Positive Numbers

Problem Statement: Write a program that repeatedly prompts the user to enter a positive number and calculates the sum of all entered numbers. The loop should terminate when the user enters a negative number.

Instructions:

1. Use a `do-while` loop to prompt the user to input numbers.
2. Sum the positive numbers.
3. Stop the loop if a negative number is entered.

Sample Input:

```
Enter a number: 5
Enter a number: 10
Enter a number: -1
```

Expected Output:

```
Sum of positive numbers: 15
```

Code Outline:

```
#include <stdio.h>

int main() {
    int num, sum = 0;

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

```
        if (num > 0) {
            sum += num;
        }
    } while (num > 0);

    printf("Sum of positive numbers: %d\n", sum);

    return 0;
}
```

Exercise 2: Menu-Driven Program

Problem Statement: Write a menu-driven program that allows the user to choose between options (1. Print Hello, 2. Print Goodbye, 3. Exit). The program should keep displaying the menu until the user selects option 3.

Instructions:

1. Display the menu inside a `do-while` loop.
2. The loop should terminate only when the user selects the "Exit" option.

Sample Input:

```
1. Print Hello
2. Print Goodbye
3. Exit
Enter your choice: 1
```

Expected Output:

```
Hello
```

Code Outline:

```
#include <stdio.h>

int main() {
    int choice;

    do {
        printf("1. Print Hello\n");
        printf("2. Print Goodbye\n");
        printf("3. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
```

```
        switch (choice) {
            case 1:
                printf("Hello\n");
                break;
            case 2:
                printf("Goodbye\n");
                break;
            case 3:
                printf("Exiting program...\n");
                break;
            default:
                printf("Invalid choice\n");
        }
    } while (choice != 3);

    return 0;
}
```

Exercise 3: Number Guessing Game

Problem Statement: Create a simple number guessing game where the user must guess a predefined number. The program should give feedback on whether the guess is too high or too low and keep prompting the user until they guess the correct number.

Instructions:

1. Use a `do-while` loop to keep asking the user for guesses.
2. Display whether the guess is too high, too low, or correct.
3. End the loop when the user guesses the correct number.

Sample Input:

```
Enter your guess: 50
Too high!
Enter your guess: 25
Correct! You guessed the number.
```

Code Outline:

```
#include <stdio.h>

int main() {
    int guess, number = 25;

    do {
```

```
        printf("Enter your guess: ");
        scanf("%d", &guess);

        if (guess > number) {
            printf("Too high!\n");
        } else if (guess < number) {
            printf("Too low!\n");
        } else {
            printf("Correct! You guessed the number.\n");
        }
    } while (guess != number);

    return 0;
}
```

Exercise 4: Factorial Calculation

Problem Statement: Write a program that calculates the factorial of a given number using a `do-while` loop.

Instructions:

1. Prompt the user to input a positive integer.
2. Use a `do-while` loop to calculate the factorial.
3. Output the result.

Sample Input:

```
Enter a number: 5
```

Expected Output:

```
Factorial of 5 is 120
```

Code Outline:

```
#include <stdio.h>

int main() {
    int num, i = 1;
    unsigned long long factorial = 1;

    printf("Enter a number: ");
    scanf("%d", &num);

    if (num < 0) {
```

```
        printf("Factorial is not defined for negative
numbers.\n");
    } else {
        do {
            factorial *= i;
            i++;
        } while (i <= num);

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

    return 0;
}
```

Exercise 5: Reverse a Number

Problem Statement: Write a program that takes a positive integer as input and prints the reverse of that number.

Instructions:

1. Prompt the user to enter a positive integer.
2. Use a `do-while` loop to reverse the digits of the number.
3. Display the reversed number.

Sample Input:
```
Enter a number: 1234
```

Expected Output:
```
Reversed number: 4321
```

Code Outline:
```
#include <stdio.h>

int main() {
    int num, reversed = 0, remainder;

    printf("Enter a number: ");
    scanf("%d", &num);

    do {
        remainder = num % 10;
        reversed = reversed * 10 + remainder;
```

```
        num /= 10;
    } while (num != 0);

    printf("Reversed number: %d\n", reversed);

    return 0;
}
```

Exercise 6: Check if a Number is Prime

Problem Statement: Write a program that checks if a given number is prime using a `do-while` loop.

Instructions:

1. Prompt the user to enter a positive integer.
2. Use a `do-while` loop to check if the number has any divisors other than 1 and itself.
3. Print whether the number is prime or not.

Sample Input:
```
Enter a number: 7
```


Expected Output:
```
7 is a prime number.
```

Code Outline:
```
#include <stdio.h>

int main() {
    int num, i = 2, is_prime = 1;

    printf("Enter a number: ");
    scanf("%d", &num);

    if (num <= 1) {
        is_prime = 0;
    } else {
        do {
            if (num % i == 0) {
                is_prime = 0;
                break;
            }
            i++;
```

```
        } while (i <= num / 2);
    }

    if (is_prime) {
        printf("%d is a prime number.\n", num);
    } else {
        printf("%d is not a prime number.\n", num);
    }

    return 0;
}
```

Lab Manual: Programming Exercises in C on `for` Statement

Objective

The objective of this lab is to understand and apply the `for` loop in C programming. The `for` loop is used for iterating over a range of values and is commonly used for counting iterations. By the end of this lab, students should be able to use `for` loops effectively to solve a variety of problems.

Exercise 1: Print Numbers 1 to 10

Problem Statement: Write a program that prints the numbers from 1 to 10 using a `for` loop.

Instructions:

1. Use a `for` loop to iterate from 1 to 10.
2. Print each number in the iteration.

Expected Output:

```
bash
Copy code
1
2
3
4
5
6
7
8
9
10
```


Code Outline:

```
c
Copy code
#include <stdio.h>

int main() {
    for (int i = 1; i <= 10; i++) {
        printf("%d\n", i);
    }
    return 0;
}
```

Exercise 2: Sum of First N Natural Numbers

Problem Statement: Write a program that calculates the sum of the first `N` natural numbers, where `N` is provided by the user.

Instructions:

1. Prompt the user to enter a positive integer `N`.
2. Use a `for` loop to compute the sum of numbers from 1 to `N`.
3. Display the result.

Sample Input:

```
bash
Copy code
Enter a number: 5
```

Expected Output:

```
bash
Copy code
Sum of the first 5 natural numbers is 15
```

Code Outline:


```
c
Copy code
#include <stdio.h>

int main() {
    int N, sum = 0;

    printf("Enter a number: ");
    scanf("%d", &N);

    for (int i = 1; i <= N; i++) {
        sum += i;
    }

    printf("Sum of the first %d natural numbers is %d\n", N, sum);

    return 0;
}
```

Exercise 3: Factorial Calculation

Problem Statement: Write a program that calculates the factorial of a given number using a `for` loop.

Instructions:

1. Prompt the user to enter a positive integer.
2. Use a `for` loop to compute the factorial of the number.
3. Display the result.

Sample Input:

```
bash
Copy code
Enter a number: 4
```

Expected Output:

```
bash
Copy code
Factorial of 4 is 24
```

Code Outline:

```
c
Copy code
#include <stdio.h>
```



```
int main() {
    int num;
    unsigned long long factorial = 1;

    printf("Enter a number: ");
    scanf("%d", &num);

    if (num < 0) {
        printf("Factorial is not defined for negative
numbers.\n");
    } else {
        for (int i = 1; i <= num; i++) {
            factorial *= i;
        }
        printf("Factorial of %d is %llu\n", num, factorial);
    }

    return 0;
}
```

Exercise 4: Print Multiplication Table

Problem Statement: Write a program that prints the multiplication table for a given number `N` using a `for` loop.

Instructions:

1. Prompt the user to enter a number `N`.
2. Use a `for` loop to print the multiplication table from 1 to 10.

Sample Input:

```
bash
Copy code
Enter a number: 3
```

Expected Output:

```
bash
Copy code
3 x 1 = 3
3 x 2 = 6
3 x 3 = 9
3 x 4 = 12
3 x 5 = 15
3 x 6 = 18
3 x 7 = 21
3 x 8 = 24
3 x 9 = 27
3 x 10 = 30
```


Code Outline:

```
c
Copy code
#include <stdio.h>

int main() {
    int N;

    printf("Enter a number: ");
    scanf("%d", &N);

    for (int i = 1; i <= 10; i++) {
        printf("%d x %d = %d\n", N, i, N * i);
    }
```

```
    return 0;
}
```

Exercise 5: Print Fibonacci Series

Problem Statement: Write a program that prints the first `N` terms of the Fibonacci series using a `for` loop.

Instructions:

1. Prompt the user to enter the number of terms `N`.
2. Use a `for` loop to compute and print the Fibonacci series up to `N` terms.

Sample Input:

```
Enter the number of terms: 5
```

Expected Output:

```
0
1
1
2
3
```


Code Outline:

```
c
Copy code
#include <stdio.h>

int main() {
    int N, a = 0, b = 1, c;

    printf("Enter the number of terms: ");
    scanf("%d", &N);

    if (N <= 0) {
        printf("Number of terms must be positive.\n");
    } else {
        printf("%d\n", a);
        if (N > 1) {
            printf("%d\n", b);
            for (int i = 3; i <= N; i++) {
                c = a + b;
                printf("%d\n", c);
```

```
                a = b;
                b = c;
            }
        }
    }

    return 0;
}
```

Exercise 6: Find Prime Numbers in a Range

Problem Statement: Write a program that finds and prints all prime numbers between 1 and a given number `N` using a `for` loop.

Instructions:

1. Prompt the user to enter a number `N`.
2. Use nested `for` loops to determine and print all prime numbers between 1 and `N`.

Sample Input:

```
Enter a number: 10
```


Expected Output:

```
2
3
5
7
```

Code Outline:

```
c
Copy code
#include <stdio.h>

int main() {
    int N, i, j, is_prime;

    printf("Enter a number: ");
    scanf("%d", &N);

    for (i = 2; i <= N; i++) {
        is_prime = 1;
        for (j = 2; j <= i / 2; j++) {
            if (i % j == 0) {
```

```
                is_prime = 0;
                break;
            }
        }
        if (is_prime) {
            printf("%d\n", i);
        }
    }

    return 0;
}
```

Exercise 7: Reverse a Number

Problem Statement: Write a program that reverses the digits of a given number using a `for` loop.

Instructions:

1. Prompt the user to enter a positive integer.
2. Use a `for` loop to reverse the digits of the number and display the result.

Sample Input:
```
Enter a number: 1234
```

Expected Output:
```
Reversed number: 4321
```

Code Outline:
```
#include <stdio.h>
#include <math.h>

int main() {
    int num, reversed = 0, digit, temp;

    printf("Enter a number: ");
    scanf("%d", &num);

    temp = num;
    for (; temp > 0; temp /= 10) {
        digit = temp % 10;
        reversed = reversed * 10 + digit;
    }
```

```
    printf("Reversed number: %d\n", reversed);

    return 0;
}
```

Lab Manual: Programming Exercises in C on Arrays

Objective

The goal of this lab is to practice working with one-dimensional and two-dimensional arrays in C. By the end of this lab, students will be able to effectively use arrays for storing and manipulating data in a structured manner.

Section 1: One-Dimensional Arrays

Exercise 1: Find the Maximum and Minimum Elements

Problem Statement: Write a program to find the maximum and minimum values in an array of integers.

Instructions:

1. Prompt the user to enter the size of the array and then the array elements.
2. Use a loop to find the maximum and minimum values.

Sample Input:


```
Enter the number of elements: 5
Enter the elements: 12 45 23 78 56
```

Expected Output:

```
Maximum value: 78
Minimum value: 12
```

Code Outline:

```
#include <stdio.h>

int main() {
    int n, i;
    int max, min;

    printf("Enter the number of elements: ");
    scanf("%d", &n);

    int arr[n];

    printf("Enter the elements:\n");
    for (i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
```

```
    }

    max = min = arr[0];

    for (i = 1; i < n; i++) {
        if (arr[i] > max) max = arr[i];
        if (arr[i] < min) min = arr[i];
    }

    printf("Maximum value: %d\n", max);
    printf("Minimum value: %d\n", min);

    return 0;
}
```

Exercise 2: Calculate the Average of Array Elements

Problem Statement: Write a program to calculate the average of elements in an array.

Instructions:

1. Prompt the user to enter the size of the array and then the array elements.
2. Use a loop to calculate the sum and then compute the average.

Sample Input:

```
Enter the number of elements: 4
Enter the elements: 10 20 30 40
```

Expected Output:

```
Average value: 25.00
```

Code Outline:

```
#include <stdio.h>

int main() {
    int n, i;
    float sum = 0.0, average;

    printf("Enter the number of elements: ");
    scanf("%d", &n);

    int arr[n];
```

```
    printf("Enter the elements:\n");
    for (i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
        sum += arr[i];
    }

    average = sum / n;

    printf("Average value: %.2f\n", average);

    return 0;
}
```

Exercise 3: Reverse an Array

Problem Statement: Write a program to reverse the elements of an array.

Instructions:

1. Prompt the user to enter the size of the array and then the array elements.
2. Use a loop to reverse the elements.

Sample Input:

```
Enter the number of elements: 4
Enter the elements: 1 2 3 4
```

Expected Output:

```
Reversed array: 4 3 2 1
```

Code Outline:

```
#include <stdio.h>

int main() {
    int n, i;

    printf("Enter the number of elements: ");
    scanf("%d", &n);

    int arr[n];

    printf("Enter the elements:\n");
    for (i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
```

```
    }

    printf("Reversed array:\n");
    for (i = n - 1; i >= 0; i--) {
        printf("%d ", arr[i]);
    }
    printf("\n");

    return 0;
}
```

Exercise 4: Find the Frequency of Each Element

Problem Statement: Write a program to find and display the frequency of each element in an array.

Instructions:

1. Prompt the user to enter the size of the array and then the array elements.
2. Use nested loops to calculate and display the frequency of each element.

Sample Input:

```
bash
Copy code
Enter the number of elements: 6
Enter the elements: 1 2 2 3 3 3
```

Expected Output:

```
Element 1 occurs 1 time(s)
Element 2 occurs 2 time(s)
Element 3 occurs 3 time(s)
```

Code Outline:

```
#include <stdio.h>

int main() {
    int n, i, j, freq;

    printf("Enter the number of elements: ");
    scanf("%d", &n);

    int arr[n];
```

```
    printf("Enter the elements:\n");
    for (i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }

    printf("Frequencies of elements:\n");
    for (i = 0; i < n; i++) {
        freq = 1;
        if (arr[i] != -1) {
            for (j = i + 1; j < n; j++) {
                if (arr[i] == arr[j]) {
                    freq++;
                    arr[j] = -1;
                }
            }
            printf("Element %d occurs %d time(s)\n", arr[i],
freq);
        }
    }

    return 0;
}
```


Section 2: Two-Dimensional Arrays

Exercise 1: Matrix Addition

Problem Statement: Write a program to add two matrices of size `m x n`.

Instructions:

1. Prompt the user to enter the dimensions of the matrices and then the elements of both matrices.
2. Use nested loops to add the matrices and display the result.

Sample Input:

```
Enter the number of rows and columns: 2 2
Enter elements of matrix A:
1 2
3 4
Enter elements of matrix B:
5 6
7 8
```

Expected Output:

```
Resultant matrix after addition:
6 8
10 12
```

Code Outline:

```
#include <stdio.h>

int main() {
    int m, n;

    printf("Enter the number of rows and columns: ");
    scanf("%d %d", &m, &n);

    int A[m][n], B[m][n], C[m][n];

    printf("Enter elements of matrix A:\n");
    for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++) {
            scanf("%d", &A[i][j]);
        }
    }
```



```
    printf("Enter elements of matrix B:\n");
    for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++) {
            scanf("%d", &B[i][j]);
        }
    }

    printf("Resultant matrix after addition:\n");
    for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++) {
            C[i][j] = A[i][j] + B[i][j];
            printf("%d ", C[i][j]);
        }
        printf("\n");
    }

    return 0;
}
```

Exercise 2: Matrix Multiplication

Problem Statement: Write a program to multiply two matrices.

Instructions:

1. Prompt the user to enter the dimensions of the matrices and then the elements of both matrices.
2. Use nested loops to multiply the matrices and display the result.

Sample Input:

```
Enter the number of rows and columns for matrix A: 2 3
Enter the elements of matrix A:
1 2 3
4 5 6
Enter the number of rows and columns for matrix B: 3 2
Enter the elements of matrix B:
7 8
9 10
11 12
```

Expected Output:

```
Resultant matrix after multiplication:
58 64
139 154
```


Code Outline:

```
#include <stdio.h>

int main() {
    int m, n, p, q;

    printf("Enter the number of rows and columns for matrix A: ");
    scanf("%d %d", &m, &n);

    int A[m][n];

    printf("Enter the elements of matrix A:\n");
    for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++) {
            scanf("%d", &A[i][j]);
        }
    }

    printf("Enter the number of rows and columns for matrix B: ");
    scanf("%d %d", &p, &q);
```

```
    int B[p][q], C[m][q];

    if (n != p) {
        printf("Matrix multiplication is not possible.\n");
        return 0;
    }

    printf("Enter the elements of matrix B:\n");
    for (int i = 0; i < p; i++) {
        for (int j = 0; j < q; j++) {
            scanf("%d", &B[i][j]);
        }
    }

    // Initialize result matrix
    for (int i = 0; i < m; i++) {
        for (int j = 0; j < q; j++) {
            C[i][j] = 0;
        }
    }
```



```
    // Multiply matrices
    for (int i = 0; i < m; i++) {
        for (int j = 0; j < q; j++) {
            for (int k = 0; k < n; k++) {
                C[i][j] += A[i][k] * B[k][j];
            }
        }
    }

    printf("Resultant matrix after multiplication:\n");
    for (int i = 0; i < m; i++) {
        for (int j = 0; j < q; j++) {
            printf("%d ", C[i][j]);
        }
        printf("\n");
    }

    return 0;
}
```

Exercise 3: Transpose of a Matrix

Problem Statement: Write a program to find the transpose of a matrix.

Instructions:

1. Prompt the user to enter the dimensions of the matrix and then the elements of the matrix.
2. Use nested loops to compute and display the transpose.

Sample Input:

```
bash
Copy code
Enter the number of rows and columns: 2 3
Enter the elements of the matrix:
1 2 3
4 5 6
```

Expected Output:

```
Transpose of the matrix:
1 4
2 5
3 6
```


Code Outline:

```
#include <stdio.h>

int main() {
    int m, n;

    printf("Enter the number of rows and columns: ");
    scanf("%d %d", &m, &n);

    int A[m][n], T[n][m];

    printf("Enter the elements of the matrix:\n");
    for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++) {
            scanf("%d", &A[i][j]);
        }
    }

    printf("Transpose of the matrix:\n");
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            T[i][j] = A[j][i];
```

```
            printf("%d ", T[i][j]);
        }
        printf("\n");
    }

    return 0;
}
```

Exercise 4: Check if Two Matrices are Equal

Problem Statement: Write a program to check if two matrices are equal.

Instructions:

1. Prompt the user to enter the dimensions and elements of both matrices.
2. Use nested loops to compare the matrices.

Sample Input:

```
bash
Copy code
Enter the number of rows and columns: 2 2
Enter elements of matrix A:
1 2
3 4
Enter elements of matrix B:
1 2
3 4
```


Expected Output:

```
Matrices are equal.
```

Code Outline:

```
c
Copy code
#include <stdio.h>

int main() {
    int m, n, i, j;
    int equal = 1;

    printf("Enter the number of rows and columns: ");
    scanf("%d %d", &m, &n);
```

```
    int A[m][n], B[m][n];

    printf("Enter elements of matrix A:\n");
    for (i = 0; i < m; i++) {
        for (j = 0; j < n; j++) {
            scanf("%d", &A[i][j]);
        }
    }

    printf("Enter elements of matrix B:\n");
    for (i = 0; i < m; i++) {
        for (j = 0; j < n; j++) {
            scanf("%d", &B[i][j]);
        }
    }

    for (i = 0; i < m; i++) {
        for (j = 0; j < n; j++) {
            if (A[i][j] != B[i][j]) {
                equal = 0;
                break;
            }
        }
        if (!equal) break;
    }

    if (equal) {
        printf("Matrices are equal.\n");
    } else {
        printf("Matrices are not equal.\n");
    }

    return 0;
}
```

Lab Manual: String Manipulation Programs in C

Objective

The goal of this lab is to practice basic string manipulation techniques in C. By the end of this lab, students will be able to concatenate and compare strings effectively.

Section 1: Programs for Putting Two Strings Together

Exercise 1: String Concatenation

Problem Statement: Write a program to concatenate two strings and display the resulting string.

Instructions:

1. Prompt the user to enter two strings.
2. Use string concatenation functions to combine them.
3. Display the concatenated string.

Sample Input:

```
Enter the first string: Hello
Enter the second string: World
```


Expected Output:

```
Concatenated string: HelloWorld
```

Code Outline:

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

int main() {
    char str1[100], str2[100];

    printf("Enter the first string: ");
    fgets(str1, sizeof(str1), stdin);
    str1[strcspn(str1, "\n")] = 0; // Remove trailing newline
character

    printf("Enter the second string: ");
    fgets(str2, sizeof(str2), stdin);
    str2[strcspn(str2, "\n")] = 0; // Remove trailing newline
character
```

```
    strcat(str1, str2); // Concatenate str2 to str1

    printf("Concatenated string: %s\n", str1);

    return 0;
}
```

Section 2: Programs for Comparing Two Strings

Exercise 2: String Comparison

Problem Statement: Write a program to compare two strings and determine if they are equal or not.

Instructions:

1. Prompt the user to enter two strings.
2. Use string comparison functions to compare the two strings.
3. Display whether the strings are equal or not.

Sample Input:


```
Enter the first string: Hello
Enter the second string: Hello
```

Expected Output:

```
The strings are equal.
```

Code Outline:

```
c
Copy code
#include <stdio.h>
#include <string.h>

int main() {
    char str1[100], str2[100];

    printf("Enter the first string: ");
    fgets(str1, sizeof(str1), stdin);
    str1[strcspn(str1, "\n")] = 0; // Remove trailing newline
character

    printf("Enter the second string: ");
    fgets(str2, sizeof(str2), stdin);
```

```
    str2[strcspn(str2, "\n")] = 0; // Remove trailing newline
character

    if (strcmp(str1, str2) == 0) {
        printf("The strings are equal.\n");
    } else {
        printf("The strings are not equal.\n");
    }

    return 0;
}
```

Additional Notes

- **`fgets()` Function**: Used to read strings with spaces. Make sure to remove the trailing newline character added by `fgets()`.
- **`strcat()` Function**: Concatenates two strings.
- **`strcmp()` Function**: Compares two strings and returns 0 if they are equal.

Lab Manual: Simple Program Using Structures in C

Objective

The objective of this lab is to introduce the concept of structures in C programming. By the end of this lab, students will be able to define, initialize, and manipulate structures to store and process grouped data.

Exercise 1: Define and Use a Structure

Problem Statement: Write a program to define a structure for a `Student` that contains the following information: `name`, `roll number`, and `marks`. The program should allow the user to input the details of a student and then display them.

Instructions:

1. Define a structure named `Student` with members for `name`, `roll number`, and `marks`.
2. Prompt the user to enter the student's name, roll number, and marks.
3. Display the entered information.

Sample Input:


```
Enter student name: John Doe
Enter roll number: 123
Enter marks: 85.5
```

Expected Output:

```
Student Details:
Name: John Doe
Roll Number: 123
Marks: 85.5
```

Code Outline:

```
#include <stdio.h>

struct Student {
    char name[100];
    int roll_number;
    float marks;
};

int main() {
    struct Student student;
```

```
    printf("Enter student name: ");
    fgets(student.name, sizeof(student.name), stdin);
    student.name[strcspn(student.name, "\n")] = 0; // Remove
trailing newline character

    printf("Enter roll number: ");
    scanf("%d", &student.roll_number);

    printf("Enter marks: ");
    scanf("%f", &student.marks);

    printf("\nStudent Details:\n");
    printf("Name: %s\n", student.name);
    printf("Roll Number: %d\n", student.roll_number);
    printf("Marks: %.2f\n", student.marks);

    return 0;
}
```

Additional Notes

- **Structure Definition**: The `struct` keyword is used to define a structure, followed by a name for the structure and a list of members enclosed in braces.
- **Accessing Members**: Use the dot (`.`) operator to access the members of a structure.
- **Input Handling**: For strings, `fgets()` is used to read input with spaces. Ensure to remove the newline character that `fgets()` adds.

Lab Manual: Simple Programs Using Pointers in C

Objective

The objective of this lab is to introduce the concept of pointers in C programming. By the end of this lab, students will be able to understand and utilize pointers for various tasks such as accessing variables, arrays, and functions.

Exercise 1: Basic Pointer Usage

Problem Statement: Write a program to demonstrate basic pointer usage. The program should include:

1. Declaring and initializing a pointer.
2. Using the pointer to modify the value of a variable.
3. Displaying the original and modified values.

Instructions:

1. Declare an integer variable and a pointer to an integer.
2. Initialize the pointer to point to the variable.
3. Use the pointer to modify the variable's value.
4. Display the value of the variable before and after modification.

Sample Input/Output:
```
Original value: 10
Modified value: 20
```

Code Outline:
```
#include <stdio.h>

int main() {
    int num = 10;         // Declare an integer variable
    int *ptr = &num;      // Declare a pointer and initialize it
with the address of num

    printf("Original value: %d\n", num);

    *ptr = 20;            // Use the pointer to modify the value of
num

    printf("Modified value: %d\n", num);
```

```
    return 0;
}
```

Exercise 2: Pointer and Array

Problem Statement: Write a program to demonstrate the relationship between pointers and arrays. The program should:

1. Initialize an array of integers.
2. Use a pointer to traverse and display the elements of the array.

Instructions:

1. Declare an array of integers and a pointer to an integer.
2. Initialize the pointer to point to the first element of the array.
3. Use a loop to traverse the array using the pointer and display each element.

Sample Output:

```
Array elements:
1 2 3 4 5
```


Code Outline:

```
#include <stdio.h>

int main() {
    int arr[] = {1, 2, 3, 4, 5};  // Declare and initialize an
array
    int *ptr = arr;               // Initialize the pointer to
point to the first element of the array

    printf("Array elements:\n");
    for (int i = 0; i < 5; i++) {
        printf("%d ", *(ptr + i)); // Traverse the array using the
pointer
    }
    printf("\n");

    return 0;
}
```

Exercise 3: Pointer to a Function

Problem Statement: Write a program to demonstrate the use of a pointer to a function. The program should:

1. Define a function that performs a simple operation (e.g., addition).
2. Declare a pointer to the function.
3. Use the pointer to call the function.

Instructions:

1. Define a function that takes two integers as arguments and returns their sum.
2. Declare a function pointer and initialize it to point to the defined function.
3. Use the function pointer to call the function and display the result.

Sample Output:

```
Sum: 30
```

Code Outline:

```
#include <stdio.h>

// Define a function that adds two integers
int add(int a, int b) {
    return a + b;
}

int main() {
    int (*func_ptr)(int, int) = add; // Declare a function pointer and initialize it

    int result = func_ptr(10, 20);   // Use the function pointer to call the function

    printf("Sum: %d\n", result);

    return 0;
}
```

Exercise 4: Pointer to a Structure

Problem Statement: Write a program to demonstrate the use of pointers with structures. The program should:

1. Define a structure to store student information (e.g., name, roll number).
2. Use a pointer to access and modify the structure members.

Instructions:

1. Define a structure named `Student` with members for `name` and `roll_number`.
2. Declare a structure variable and a pointer to the structure.
3. Initialize the structure and use the pointer to modify and display its members.

Sample Output:

```
Student Details:
Name: Alice
Roll Number: 123
```

Code Outline:

```
#include <stdio.h>

// Define the structure
struct Student {
    char name[100];
    int roll_number;
};
```



```
int main() {
    struct Student student;       // Declare a structure variable
    struct Student *ptr = &student; // Declare a pointer to the
structure

    // Initialize the structure members using the pointer
    snprintf(ptr->name, sizeof(ptr->name), "Alice");
    ptr->roll_number = 123;

    // Display the structure members using the pointer
    printf("Student Details:\n");
    printf("Name: %s\n", ptr->name);
    printf("Roll Number: %d\n", ptr->roll_number);

    return 0;
}
```

Additional Notes

- **Pointer Basics**: Pointers store memory addresses and can be used to access or modify the values stored in those addresses.
- **Pointer Arithmetic**: You can perform arithmetic operations on pointers to navigate through arrays.
- **Function Pointers**: Allow functions to be passed as arguments or returned from other functions.

PROGRAMING IN C LAB MANUAL BY HIMANSHU SINGH

Lab Manual: Simple Programs Using Unions in C

Objective

The objective of this lab is to introduce the concept of unions in C programming. By the end of this lab, students will be able to define, initialize, and use unions to manage different types of data in a single memory location.

Exercise 1: Basic Union Usage

Problem Statement: Write a program to demonstrate the basic usage of a union. The program should:

1. Define a union to store different types of data (e.g., an integer, a float, and a character).
2. Initialize the union with different data types and display the value of each member.

Instructions:

1. Define a union named `Data` with members for an integer, a float, and a character.
2. Declare a variable of type `Data` and initialize it with different values.
3. Display the values of the union members.

Sample Output:

```
bash
Copy code
Integer value: 10
Float value: 3.14
Character value: A
```

Code Outline:

```
c
Copy code
#include <stdio.h>

// Define the union
union Data {
    int i;
    float f;
    char c;
};

int main() {
    union Data data;
```

```
    // Initialize and display the integer value
    data.i = 10;
    printf("Integer value: %d\n", data.i);

    // Initialize and display the float value
    data.f = 3.14;
    printf("Float value: %.2f\n", data.f);

    // Initialize and display the character value
    data.c = 'A';
    printf("Character value: %c\n", data.c);

    return 0;
}
```

Exercise 2: Union for Multiple Data Types

Problem Statement: Write a program to demonstrate the use of a union for handling multiple data types in a single memory location. The program should:

1. Define a union to handle a `name` (string), an `age` (integer), and a `height` (float).
2. Use the union to store and display different types of data.

Instructions:

1. Define a union named `Person` with members for a string, an integer, and a float.
2. Declare a variable of type `Person` and initialize it with different types of data.
3. Display the value of each union member.

Sample Output:

```
Name: John
Age: 25
Height: 5.9
```

Code Outline:

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

// Define the union
union Person {
    char name[50];
    int age;
```

```
    float height;
};

int main() {
    union Person person;

    // Store and display the name
    strcpy(person.name, "John");
    printf("Name: %s\n", person.name);

    // Store and display the age
    person.age = 25;
    printf("Age: %d\n", person.age);

    // Store and display the height
    person.height = 5.9;
    printf("Height: %.1f\n", person.height);

    return 0;
}
```


Exercise 3: Union with Structs

Problem Statement: Write a program to demonstrate how unions can be used within structures. The program should:

1. Define a structure with a union member.
2. Initialize and display the union member within the structure.

Instructions:

1. Define a structure named `Employee` that includes a union to store employee details (e.g., ID, salary, and department).
2. Declare a variable of type `Employee`, initialize the union members, and display their values.

Sample Output:

```
Employee ID: 1001
Salary: 75000.50
Department: HR
```

Code Outline:

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

```
// Define the union
union EmployeeDetails {
    int id;
    float salary;
    char department[50];
};

// Define the structure with a union member
struct Employee {
    char name[50];
    union EmployeeDetails details;
};

int main() {
    struct Employee emp;

    // Store and display the employee ID
    strcpy(emp.name, "Alice");
    emp.details.id = 1001;
    printf("Employee ID: %d\n", emp.details.id);

    // Store and display the salary
    emp.details.salary = 75000.50;
    printf("Salary: %.2f\n", emp.details.salary);

    // Store and display the department
    strcpy(emp.details.department, "HR");
    printf("Department: %s\n", emp.details.department);

    return 0;
}
```

Additional Notes

- **Union Basics**: A union allows storing different data types in the same memory location, but only one member can hold a value at a time.
- **Memory Usage**: The size of a union is determined by the size of its largest member.
- **Accessing Members**: Accessing different members of a union will affect the value of the previously stored member, as all members share the same memory location.

PROGRAMING IN C LAB MANUAL BY HIMANSHU SINGH

BANK MANAGEMENT SYSTEM IN C

```
// bank management system using c
// password is codewithc
#include<stdio.h>
#include<stdlib.h>
#include<windows.h>

int i,j;
int main_exit;
void menu();
struct date
        {
        int month,day,year;
        };
struct
        {
        char name[60];
        int acc_no,age;
        char address[60];
        char citizenship[15];
        double phone;
        char acc_type[10];
        float amt;
        struct date dob;
        struct date deposit;
        struct date withdraw;
        }add,upd,check,rem,transaction;

float interest(float t,float amount,int rate)
        {
        float SI;
        SI=(rate*t*amount)/100.0;
        return (SI);
```

```
        }
void fordelay(int j)
                {   int i,k;
                    for(i=0;i<j;i++)
                k=i;
                }

void new_acc()

{
   int choice;
   FILE *ptr;

   ptr=fopen("record.dat","a+");
   account_no:
   system("cls");
   printf("\t\t\t\xB2\xB2\xB2\ ADD RECORD  \xB2\xB2\xB2\xB2");
   printf("\n\n\nEnter today's date(mm/dd/yyyy):");
   scanf("%d/%d/%d",&add.deposit.month,&add.deposit.day,&add.deposit.year);

        printf("\nEnter the account number:");
        scanf("%d",&check.acc_no);
        while(fscanf(ptr,"%d %s %d/%d/%d %d %s %s %lf %s %f
%d/%d/%d\n",&add.acc_no,add.name,&add.dob.month,&add.dob.day,&add.dob.year,
&add.age,add.address,add.citizenship,&add.phone,add.acc_type,&add.amt,&add.deposit
.month,&add.deposit.day,&add.deposit.year)!=EOF)
                {
        if (check.acc_no==add.acc_no)
                {
                        printf("Account no. already in use!");
                fordelay(1000000000);
                goto account_no;
                }
                }
   add.acc_no=check.acc_no;
      printf("\nEnter the name:");
```

```
        scanf("%s",add.name);
        printf("\nEnter the date of birth(mm/dd/yyyy):");
        scanf("%d/%d/%d",&add.dob.month,&add.dob.day,&add.dob.year);
        printf("\nEnter the age:");
        scanf("%d",&add.age);
        printf("\nEnter the address:");
        scanf("%s",add.address);
        printf("\nEnter the citizenship number:");
        scanf("%s",add.citizenship);
        printf("\nEnter the phone number: ");
        scanf("%lf",&add.phone);
        printf("\nEnter the amount to deposit:$");
        scanf("%f",&add.amt);
        printf("\nType of account:\n\t#Saving\n\t#Current\n\t#Fixed1(for 1
year)\n\t#Fixed2(for 2 years)\n\t#Fixed3(for 3 years)\n\n\tEnter your choice:");
        scanf("%s",add.acc_type);

      fprintf(ptr,"%d %s %d/%d/%d %d %s %s %lf %s %f
%d/%d/%d\n",add.acc_no,add.name,add.dob.month,add.dob.day,add.dob.year,add.age,
add.address,add.citizenship,add.phone,add.acc_type,add.amt,add.deposit.month,add.dep
osit.day,add.deposit.year);


        fclose(ptr);
        printf("\nAccount created successfully!");
        add_invalid:
        printf("\n\n\n\t\tEnter 1 to go to the main menu and 0 to exit:");
        scanf("%d",&main_exit);
        system("cls");

                if (main_exit==1)
        menu();
                else if(main_exit==0)
                close();
                else
                {
```

```
                    printf("\nInvalid!\a");
                    goto add_invalid;
                }
}
void view_list()
{
   FILE *view;
   view=fopen("record.dat","r");
   int test=0;
   system("cls");
   printf("\nACC. NO.\tNAME\t\t\tADDRESS\t\t\tPHONE\n");

   while(fscanf(view,"%d %s %d/%d/%d %d %s %s %lf %s %f
%d/%d/%d",&add.acc_no,add.name,&add.dob.month,&add.dob.day,&add.dob.year,&a
dd.age,add.address,add.citizenship,&add.phone,add.acc_type,&add.amt,&add.deposit.m
onth,&add.deposit.day,&add.deposit.year)!=EOF)
     {
        printf("\n%6d\t
%10s\t\t\t%10s\t\t%.0lf",add.acc_no,add.name,add.address,add.phone);
        test++;
     }

   fclose(view);
   if (test==0)
      {   system("cls");
        printf("\nNO RECORDS!!\n");}

   view_list_invalid:
    printf("\n\nEnter 1 to go to the main menu and 0 to exit:");
      scanf("%d",&main_exit);
      system("cls");
      if (main_exit==1)
        menu();
      else if(main_exit==0)
        close();
      else
```

```
        {
            printf("\nInvalid!\a");
            goto view_list_invalid;
        }
}
void edit(void)
{
    int choice,test=0;
    FILE *old,*newrec;
    old=fopen("record.dat","r");
    newrec=fopen("new.dat","w");

    printf("\nEnter the account no. of the customer whose info you want to change:");
    scanf("%d",&upd.acc_no);
    while(fscanf(old,"%d %s %d/%d/%d %d %s %s %lf %s %f
%d/%d/%d",&add.acc_no,add.name,&add.dob.month,&add.dob.day,&add.dob.year,&a
dd.age,add.address,add.citizenship,&add.phone,add.acc_type,&add.amt,&add.deposit.m
onth,&add.deposit.day,&add.deposit.year)!=EOF)
    {
        if (add.acc_no==upd.acc_no)
        {   test=1;
            printf("\nWhich information do you want to
change?\n1.Address\n2.Phone\n\nEnter your choice(1 for address and 2 for phone):");
            scanf("%d",&choice);
            system("cls");
            if(choice==1)
                {printf("Enter the new address:");
                scanf("%s",upd.address);
                fprintf(newrec,"%d %s %d/%d/%d %d %s %s %lf %s %f
%d/%d/%d\n",add.acc_no,add.name,add.dob.month,add.dob.day,add.dob.year,add.age,
upd.address,add.citizenship,add.phone,add.acc_type,add.amt,add.deposit.month,add.dep
osit.day,add.deposit.year);
                system("cls");
                printf("Changes saved!");
                }
            else if(choice==2)
```

```
            {
                printf("Enter the new phone number:");
            scanf("%lf",&upd.phone);
            fprintf(newrec,"%d %s %d/%d/%d %d %s %s %lf %s %f
%d/%d/%d\n",add.acc_no,add.name,add.dob.month,add.dob.day,add.dob.year,add.age,
add.address,add.citizenship,upd.phone,add.acc_type,add.amt,add.deposit.month,add.dep
osit.day,add.deposit.year);
            system("cls");
            printf("Changes saved!");
            }

      }
      else
          fprintf(newrec,"%d %s %d/%d/%d %d %s %s %lf %s %f
%d/%d/%d\n",add.acc_no,add.name,add.dob.month,add.dob.day,add.dob.year,add.age,
add.address,add.citizenship,add.phone,add.acc_type,add.amt,add.deposit.month,add.dep
osit.day,add.deposit.year);
   }
   fclose(old);
   fclose(newrec);
   remove("record.dat");
   rename("new.dat","record.dat");

if(test!=1)
      {   system("cls");
          printf("\nRecord not found!!\a\a\a");
          edit_invalid:
            printf("\nEnter 0 to try again,1 to return to main menu and 2 to exit:");
            scanf("%d",&main_exit);
            system("cls");
              if (main_exit==1)

                menu();
            else if (main_exit==2)
                close();
            else if(main_exit==0)
```

```
                edit();
            else
                {printf("\nInvalid!\a");
                goto edit_invalid;}
        }
    else
        {printf("\n\n\nEnter 1 to go to the main menu and 0 to exit:");
        scanf("%d",&main_exit);
        system("cls");
        if (main_exit==1)
            menu();
        else
            close();
        }
}

void transact(void)
{   int choice,test=0;
    FILE *old,*newrec;
    old=fopen("record.dat","r");
    newrec=fopen("new.dat","w");

        printf("Enter the account no. of the customer:");
    scanf("%d",&transaction.acc_no);
    while (fscanf(old,"%d %s %d/%d/%d %d %s %s %lf %s %f
%d/%d/%d",&add.acc_no,add.name,&add.dob.month,&add.dob.day,&add.dob.year,&a
dd.age,add.address,add.citizenship,&add.phone,add.acc_type,&add.amt,&add.deposit.m
onth,&add.deposit.day,&add.deposit.year)!=EOF)
    {

            if(add.acc_no==transaction.acc_no)
            {   test=1;

if(strcmpi(add.acc_type,"fixed1")==0||strcmpi(add.acc_type,"fixed2")==0||strcmpi(add.a
cc_type,"fixed3")==0)
                {
```

```
                printf("\a\a\a\n\nYOU CANNOT DEPOSIT OR WITHDRAW CASH IN FIXED ACCOUNTS!!!!!");
                fordelay(1000000000);
                system("cls");
                menu();

            }
            printf("\n\nDo you want to\n1.Deposit\n2.Withdraw?\n\nEnter your choice(1 for deposit and 2 for withdraw):");
            scanf("%d",&choice);
            if (choice==1)
            {
                printf("Enter the amount you want to deposit:$ ");
                scanf("%f",&transaction.amt);
                add.amt+=transaction.amt;
                fprintf(newrec,"%d %s %d/%d/%d %d %s %s %lf %s %f %d/%d/%d\n",add.acc_no,add.name,add.dob.month,add.dob.day,add.dob.year,add.age,add.address,add.citizenship,add.phone,add.acc_type,add.amt,add.deposit.month,add.deposit.day,add.deposit.year);
                printf("\n\nDeposited successfully!");
            }
            else
            {
                printf("Enter the amount you want to withdraw:$ ");
                scanf("%f",&transaction.amt);
                add.amt-=transaction.amt;
                fprintf(newrec,"%d %s %d/%d/%d %d %s %s %lf %s %f %d/%d/%d\n",add.acc_no,add.name,add.dob.month,add.dob.day,add.dob.year,add.age,add.address,add.citizenship,add.phone,add.acc_type,add.amt,add.deposit.month,add.deposit.day,add.deposit.year);
                printf("\n\nWithdrawn successfully!");
            }

        }
        else
        {
```

```
            fprintf(newrec,"%d %s %d/%d/%d %d %s %s %lf %s %f
%d/%d/%d\n",add.acc_no,add.name,add.dob.month,add.dob.day,add.dob.year,add.age,
add.address,add.citizenship,add.phone,add.acc_type,add.amt,add.deposit.month,add.dep
osit.day,add.deposit.year);

          }
   }
   fclose(old);
   fclose(newrec);
   remove("record.dat");
   rename("new.dat","record.dat");
   if(test!=1)
   {
      printf("\n\nRecord not found!!");
      transact_invalid:
     printf("\n\n\nEnter 0 to try again,1 to return to main menu and 2 to exit:");
     scanf("%d",&main_exit);
     system("cls");
     if (main_exit==0)
        transact();
    else if (main_exit==1)
        menu();
    else if (main_exit==2)
        close();
    else
    {
        printf("\nInvalid!");
        goto transact_invalid;
    }

   }
   else
   {
      printf("\nEnter 1 to go to the main menu and 0 to exit:");
        scanf("%d",&main_exit);
        system("cls");
```

```
        if (main_exit==1)
            menu();
        else
            close();
    }

}
void erase(void)
{
    FILE *old,*newrec;
    int test=0;
    old=fopen("record.dat","r");
    newrec=fopen("new.dat","w");
    printf("Enter the account no. of the customer you want to delete:");
    scanf("%d",&rem.acc_no);
    while (fscanf(old,"%d %s %d/%d/%d %d %s %s %lf %s %f
%d/%d/%d",&add.acc_no,add.name,&add.dob.month,&add.dob.day,&add.dob.year,&a
dd.age,add.address,add.citizenship,&add.phone,add.acc_type,&add.amt,&add.deposit.m
onth,&add.deposit.day,&add.deposit.year)!=EOF)
    {
        if(add.acc_no!=rem.acc_no)
            fprintf(newrec,"%d %s %d/%d/%d %d %s %s %lf %s %f
%d/%d/%d\n",add.acc_no,add.name,add.dob.month,add.dob.day,add.dob.year,add.age,
add.address,add.citizenship,add.phone,add.acc_type,add.amt,add.deposit.month,add.dep
osit.day,add.deposit.year);

        else
            {test++;
            printf("\nRecord deleted successfully!\n");
            }
    }
    fclose(old);
    fclose(newrec);
    remove("record.dat");
    rename("new.dat","record.dat");
    if(test==0)
```

```
        {
            printf("\nRecord not found!!\a\a\a");
            erase_invalid:
              printf("\nEnter 0 to try again,1 to return to main menu and 2 to exit:");
              scanf("%d",&main_exit);

                 if (main_exit==1)
                    menu();
                 else if (main_exit==2)
                    close();
                 else if(main_exit==0)
                    erase();
                 else
                    {printf("\nInvalid!\a");
                    goto erase_invalid;}
        }
    else
        {printf("\nEnter 1 to go to the main menu and 0 to exit:");
        scanf("%d",&main_exit);
        system("cls");
        if (main_exit==1)
            menu();
        else
            close();
        }

}

void see(void)
{
    FILE *ptr;
    int test=0,rate;
    int choice;
    float time;
    float intrst;
    ptr=fopen("record.dat","r");
```

```
   printf("Do you want to check by\n1.Account no\n2.Name\nEnter your choice:");
   scanf("%d",&choice);
   if (choice==1)
   {   printf("Enter the account number:");
       scanf("%d",&check.acc_no);

       while (fscanf(ptr,"%d %s %d/%d/%d %d %s %s %lf %s %f
%d/%d/%d",&add.acc_no,add.name,&add.dob.month,&add.dob.day,&add.dob.year,&a
dd.age,add.address,add.citizenship,&add.phone,add.acc_type,&add.amt,&add.deposit.m
onth,&add.deposit.day,&add.deposit.year)!=EOF)
       {
           if(add.acc_no==check.acc_no)
           {   system("cls");
               test=1;

               printf("\nAccount NO.:%d\nName:%s \nDOB:%d/%d/%d \nAge:%d
\nAddress:%s \nCitizenship No:%s \nPhone number:%.0lf \nType Of Account:%s
\nAmount deposited:$ %.2f \nDate Of
Deposit:%d/%d/%d\n\n",add.acc_no,add.name,add.dob.month,add.dob.day,add.dob.yea
r,add.age,add.address,add.citizenship,add.phone,
               add.acc_type,add.amt,add.deposit.month,add.deposit.day,add.deposit.year);
               if(strcmpi(add.acc_type,"fixed1")==0)
                   {
                       time=1.0;
                       rate=9;
                       intrst=interest(time,add.amt,rate);
                       printf("\n\nYou will get $%.2f as interest on
%d/%d/%d",intrst,add.deposit.month,add.deposit.day,add.deposit.year+1);
                   }
               else if(strcmpi(add.acc_type,"fixed2")==0)
                   {
                       time=2.0;
                       rate=11;
                       intrst=interest(time,add.amt,rate);
                       printf("\n\nYou will get $.%.2f as interest on
%d/%d/%d",intrst,add.deposit.month,add.deposit.day,add.deposit.year+2);
```

```
                }
            else if(strcmpi(add.acc_type,"fixed3")==0)
                {
                    time=3.0;
                    rate=13;
                    intrst=interest(time,add.amt,rate);
                    printf("\n\nYou will get $.%.2f as interest on
%d/%d/%d",intrst,add.deposit.month,add.deposit.day,add.deposit.year+3);

                }
            else if(strcmpi(add.acc_type,"saving")==0)
                {
                    time=(1.0/12.0);
                    rate=8;
                    intrst=interest(time,add.amt,rate);
                    printf("\n\nYou will get $.%.2f as interest on %d of every
month",intrst,add.deposit.day);
```



```
                }
            else if(strcmpi(add.acc_type,"current")==0)
                {

                    printf("\n\nYou will get no interest\a\a");

                }

        }
    }
  }
  else if (choice==2)
  {   printf("Enter the name:");
      scanf("%s",&check.name);
      while (fscanf(ptr,"%d %s %d/%d/%d %d %s %s %lf %s %f
%d/%d/%d",&add.acc_no,add.name,&add.dob.month,&add.dob.day,&add.dob.year,&a
```

```
dd.age,add.address,add.citizenship,&add.phone,add.acc_type,&add.amt,&add.deposit.month,&add.deposit.day,&add.deposit.year)!=EOF)
      {
         if(strcmpi(add.name,check.name)==0)
         {   system("cls");
             test=1;
             printf("\nAccount No.:%d\nName:%s \nDOB:%d/%d/%d \nAge:%d \nAddress:%s \nCitizenship No:%s \nPhone number:%.0lf \nType Of Account:%s \nAmount deposited:$%.2f \nDate Of Deposit:%d/%d/%d\n\n",add.acc_no,add.name,add.dob.month,add.dob.day,add.dob.year,add.age,add.address,add.citizenship,add.phone,
             add.acc_type,add.amt,add.deposit.month,add.deposit.day,add.deposit.year);
             if(strcmpi(add.acc_type,"fixed1")==0)
                {
                   time=1.0;
                   rate=9;
                   intrst=interest(time,add.amt,rate);
                   printf("\n\nYou will get $.%.2f as interest on %d/%d/%d",intrst,add.deposit.month,add.deposit.day,add.deposit.year+1);
                }
             else if(strcmpi(add.acc_type,"fixed2")==0)
                {
                   time=2.0;
                   rate=11;
                   intrst=interest(time,add.amt,rate);
                   printf("\n\nYou will get $.%.2f as interest on %d/%d/%d",intrst,add.deposit.month,add.deposit.day,add.deposit.year+2);

                }
             else if(strcmpi(add.acc_type,"fixed3")==0)
                {
                   time=3.0;
                   rate=13;
                   intrst=interest(time,add.amt,rate);
                   printf("\n\nYou will get $.%.2f as interest on %d/%d/%d",intrst,add.deposit.month,add.deposit.day,add.deposit.year+3);
```

```
                }
            else if(strcmpi(add.acc_type,"saving")==0)
                {
                    time=(1.0/12.0);
                    rate=8;
                    intrst=interest(time,add.amt,rate);
                    printf("\n\nYou will get $.%.2f as interest on %d of every
month",intrst,add.deposit.day);

                }
            else if(strcmpi(add.acc_type,"current")==0)
                {

                    printf("\n\nYou will get no interest\a\a");

                }
        }
    }
  }


   fclose(ptr);
    if(test!=1)
      {   system("cls");
          printf("\nRecord not found!!\a\a\a");
          see_invalid:
            printf("\nEnter 0 to try again,1 to return to main menu and 2 to exit:");
            scanf("%d",&main_exit);
            system("cls");
              if (main_exit==1)
                  menu();
             else if (main_exit==2)
                  close();
             else if(main_exit==0)
```

```
                see();
            else
                {
                system("cls");
                printf("\nInvalid!\a");
                goto see_invalid;}
      }
   else
      {printf("\nEnter 1 to go to the main menu and 0 to exit:");
      scanf("%d",&main_exit);}
      if (main_exit==1)
      {
         system("cls");
         menu();
      }

      else
         {

          system("cls");
         close();
         }

}


void close(void)
{
   printf("\n\n\n\nban_management_system_sample_by_himanshu!");
   }


void menu(void)
{   int choice;
   system("cls");
   system("color 9");
```

```
   printf("\n\n\t\t\tCUSTOMER ACCOUNT BANKING MANAGEMENT SYSTEM");
   printf("\n\n\n\t\t\t\xB2\xB2\xB2\xB2\xB2\xB2\xB2 WELCOME TO THE MAIN MENU
\xB2\xB2\xB2\xB2\xB2\xB2\xB2");
   printf("\n\n\t\t1.Create new account\n\t\t2.Update information of existing
account\n\t\t3.For transactions\n\t\t4.Check the details of existing
account\n\t\t5.Removing existing account\n\t\t6.View customer's
list\n\t\t7.Exit\n\n\n\n\n\t\t Enter your choice:");
   scanf("%d",&choice);

   system("cls");
   switch(choice)
   {
      case 1:new_acc();
      break;
      case 2:edit();
      break;
      case 3:transact();
      break;
      case 4:see();
      break;
      case 5:erase();
      break;
      case 6:view_list();
      break;
      case 7:close();
      break;

   }



}
int main()
{
   char pass[10],password[10]="codewithc";
   int i=0;
```

```
   printf("\n\n\t\tEnter the password to login:");
   scanf("%s",pass);
   /*do
   {
   //if (pass[i]!=13&&pass[i]!=8)
       {
           printf("*");
           pass[i]=getch();
           i++;
       }
   }while (pass[i]!=13);
   pass[10]='\0';*/
   if (strcmp(pass,password)==0)
       {printf("\n\nPassword Match!\nLOADING");
       for(i=0;i<=6;i++)
       {
           fordelay(100000000);
           printf(".");
       }
              system("cls");
           menu();
       }
   else
       {   printf("\n\nWrong password!!\a\a\a");
           login_try:
           printf("\nEnter 1 to try again and 0 to exit:");
           scanf("%d",&main_exit);
           if (main_exit==1)
                   {

                       system("cls");
                       main();
                   }

           else if (main_exit==0)
                   {
```

```
                system("cls");
                close();}
          else
                {printf("\nInvalid!");
                fordelay(1000000000);
                system("cls");
                goto login_try;}

       }
      return 0;
}
```

DEPARTMENTAL STORE IN C

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<string.h>
#include<ctype.h>
#include<windows.h>



#define ANS 15
#define ACS 4


COORD coord= {0,0}; // this is global variable
void gotoxy(int x,int y)
        {
        coord.X=x;
```

```
        coord.Y=y;
        SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE),coord);
        }
/*declaration of checking functions*/
void c_code(char[]);
int check(char[]);

/*structure declaration*/
typedef struct
{
   char name[ANS],code[ACS];
   float rate;
   int quantity;
} rec;
rec item;

/*declaration of display functions*/
void curser(int);
void dbill();
void d_mainmenu();
void display(rec *,int,int);
void window(int,int,int,int);
void dis_con();
void d_search();
```

```
void highlight(int,int);


/*declaration of main menu functions*/

void bill() ;

void edit();

void add();

void del();

void exit();


/*declaration of display submenu functions*/

void d_code();

void d_rate();
```



```
void d_quan();

void d_all();


/*start of main*/

int main()

{

   d_mainmenu();

   return 0;

}


void d_mainmenu()

{
```

```
    int i;
    char ch;
    const char *menu[]= {"   Calculate Bill","   Add Goods","   Edit Goods","   Display All
","   Search Goods", "   Delete Goods","   Exit"};
    system("cls");
//textbackground(11);
//textcolor(0);
//_setcursortype(_NOCURSOR);
    window(25,50,20,32);
    gotoxy(33,18);
    printf("MAIN MENU");
    for (i=0; i<=6; i++)
    {
        gotoxy(30,22+i+1);
        printf("%s\n\n\n",menu[i]);
    }
    curser(7);
}


void d_search()
{
    char ch;
    int i;
    const char *menu[]= {"   By Code","   By Rate","   By Quantity","   Back to main
menu"};
```

```
    system("cls");
//textbackground(11);
//textcolor(0);
    window(25,50,20,32);
    gotoxy(33,18);
    printf("SEARCH MENU");
    for (i=0; i<=3; i++)
    {
        gotoxy(30,22+i+1);
        printf("%s\n\n\n",menu[i]);
    }
    curser(4);
}
```



```
/*function for cursor movement*/
void curser(int no)
{
    int count=1;
    char ch='0';
    gotoxy(30,23);
    while(1)
    {
        switch(ch)
        {
```

```
        case 80:
            count++;
            if (count==no+1) count=1;
            break;
        case 72:
            count--;
            if(count==0) count=no;
            break;
        }
        highlight(no,count);
        ch=getch();
        if(ch=='\r')
        {
            if(no==7)
            {
                if (count==1) bill() ;
                else if(count==2) add();
                else if(count==3) edit();
                else if (count==4) d_all();
                else if (count==5) d_search();
                else if (count==6) del();
                else   exit(0);
            }
            if(no==4)
```

```
        {
            if (count==1) d_code();
            else if (count==2)d_rate();
            else if (count==3) d_quan();
            else d_mainmenu();
        }
    }
  }
}

void highlight(int no,int count)
{
```



```
  if (no==4)
  {
    //textbackground(11);
    //textcolor(0);
    gotoxy(30,23);
    printf("   By Code          ");
    gotoxy(30,24);
    printf("   By Rate          ");
    gotoxy(30,25);
    printf("   By Quantity      ");
    gotoxy(30,26);
    printf("   Back to main menu");
```

```
        //textcolor(0);
        //textbackground(2);
        switch (count)
        {
        case 1:
            gotoxy(30,23);
            printf(" - By Code          ");
            break;
        case 2:
            gotoxy(30,24);
            printf(" - By Rate          ");
            break;
        case 3:
            gotoxy(30,25);
            printf(" - By Quantity      ");
            break;
        case 4:
            gotoxy(30,26);
            printf(" - Back to main menu");
            break;
        }
    }

    if(no==7)
```

```
  {
      //textbackground(11);
      //textcolor(0);
      gotoxy (30,23);
      printf("   Calculate Bill ");
      gotoxy (30,24);
      printf("   Add Goods      ");
      gotoxy (30,25);
      printf("   Edit Goods     ");
      gotoxy (30,26);
      printf("   Display All    ");
      gotoxy (30,27);
      printf("   Search         ");
      gotoxy (30,28);
      printf("   Delete Goods   ");
      gotoxy (30,29);
      printf("   Exit           ");
      //textcolor(0);
      //textbackground(2);
      switch(count)
      {
      case 1:
         gotoxy (30,23);
         printf(" - Calculate Bill ");
```

```
        break;
    case 2:
        gotoxy (30,24);
        printf(" - Add Goods      ");
        break;
    case 3:
        gotoxy (30,25);
        printf(" - Edit Goods     ");
        break;
    case 4:
        gotoxy (30,26);
        printf(" - Display All    ");
        break;
    case 5:
        gotoxy (30,27);
        printf(" - Search         ");
        break;
    case 6:
        gotoxy (30,28);
        printf(" - Delete Goods   ");
        break;
    case 7:
        gotoxy (30,29);
        printf(" - Exit           ");
```

```
            break;
        }
    }
}

void bill()
{
    char x[4]= {0};
    int j=29,q=0,size=0,i=1;
    float total=0,gtotal=0;
    FILE *file;
    file=fopen("record.txt","r+b");
    rewind(file);
    system("cls");
    dbill();
    gotoxy(26,15);
    printf("Enter  \"end\" to finish input");
    while(1)
    {
        gotoxy(25,18);
        printf("                    ");
        gotoxy(25,19);
        printf("                    ");
        gotoxy(25,18);
```

```
printf("Enter Item Code:");
scanf("%s",x);
if(strcmp(x,"end")==0)
   break;
gotoxy(25,19);
printf("Enter Quantity:");
scanf("%d",&q);
rewind(file);
while(fread(&item,sizeof(item),1,file))
{
   if((strcmp(item.code,x)==0))
   {
      total=item.rate*q;
      gotoxy(11,j);
      printf("%4d",i);
      printf("%9s",item.name);
      printf("%13d",q);
      printf("%15.2f",item.rate);
      printf("%13.2f",total);
      gtotal=gtotal+total;
      size=sizeof(item);
      item.quantity=item.quantity-q;
      j+=2;
      i++;
```

```
            fseek(file,-size,SEEK_CUR);
            fwrite(&item,sizeof(item),1,file);
            break;
          }
      }
   }
   if(gtotal!=0)
   {
      gotoxy(30,j+5);
      printf("TOTAL AMOUNT = NRs. %6.2f",gtotal);
   }
   fclose(file);
   getch();
   d_mainmenu();
}
/*function to display bill window*/
void dbill()
{
   int i;
   gotoxy(20,10);
//;
   for (i=1; i<=10; i++)
      printf("-");
   printf(" OLD TOWN ");
```

```
    for (i=1; i<=10; i++)
        printf("-");
    printf("\n\n");
    gotoxy(30,11);
    printf("DEPARTMENTAL STORE");
//textcolor(1);
    gotoxy(32,25);
    printf("CUSTOMER'S BILL") ;
//textcolor(8);
    gotoxy(13,27);
    printf("SN.   Item Name     Quantity     Rate          Total");
}
/*function to add records*/
void add ()
{
    FILE *file;
    char y[ACS],x[12];
    system("cls");
//textbackground(11);
//textcolor(0);
    gotoxy(25,25);
    printf("Enter New Record(Y/N)?");
    while(toupper(getche())=='Y')
```

```
    {
        system("cls");
        file=fopen("record.txt","ab");
        c_code(y);
        strcpy(item.code,y);
        gotoxy(22,28);
        printf("Enter Rate Of The Item:");
        scanf("%f",&item.rate);
        gotoxy(22,30);
        printf("Enter Quantity Of The Item:");
        scanf("%d",&item.quantity);
        gotoxy(22,32);
        printf("Enter Name Of The Item:");
        scanf("%s",item.name);
        fseek(file,0,SEEK_END);
        fwrite(&item,sizeof(item),1,file);
        fclose(file);
        gotoxy(22,34);
        printf("Enter New Record(Y/N)?");


    }
    d_mainmenu();
}
```

```
/*function to check availability of code*/
void c_code(char y[])
{
   int flag;
   FILE *file;
   file=fopen("record.txt","rb");
   while(1)
   {
      system("cls");
      window(20,58,23,36);
      gotoxy(32,18);
      printf(" ADD ARTICLES ") ;
      flag=1;
      rewind(file);
      gotoxy(22,25);
      printf("Enter New Code Of The Article:");
      scanf(" %[^\n]",y);
      while(fread(&item,sizeof(item),1,file)==1)
      {
         if (strcmp(y,item.code)==0)
         {
            flag=0;
            gotoxy(26,30);
            printf("Code Already Exists");
```

```
            gotoxy(29,32);
            printf("Enter Again");
            getch();
            break;
          }
       }
       if (flag==1)
          break;
    }
}

/*function for editing*/
void edit()
{
    int flag=0,choice;
    char x[ACS],y[ACS];
    FILE *file;
    int size;
    system("cls");
//textcolor(0);
//textbackground(11);
    window(20,63,20,46);
    gotoxy(35,18);
    printf("EDIT RECORDS");
```

```
   ;
   gotoxy(25,23);
   printf("Enter Item Code: ");
   scanf("%s",x);
   flag=check(x);
   if(flag==0)
   {
      file=fopen("record.txt","r+b");
      rewind(file);
      while (fread(&item,sizeof (item),1,file))
      {
         if(strcmp(item.code,x)==0)
         {
            //textcolor(0);
            gotoxy(25,27);
            printf("name      = %s",item.name);
            gotoxy(25,28);
            printf("code      = %s",item.code);
            gotoxy(25,29);
            printf("rate      = %g",item.rate);
            gotoxy(25,30);
            printf("quantity  = %d",item.quantity);
            gotoxy(25,32);;
            printf("Do You Want To Edit This Record?(y/n):");
```

```
            fflush(file);
            if(toupper(getche())=='Y')
            {
                //textcolor(0);
                gotoxy(25,34);
                printf("1- Edit Name ");
                gotoxy(25,35);
                printf("2- Edit Code ");
                gotoxy(25,36);
                printf("3- Edit Rate ");
                gotoxy(25,37);
                printf("4- Edit Quantity ");
                gotoxy(25,39);  ;
                printf(" Enter Your Choice(1, 2, 3, 4) ");
                scanf("%d",&choice);
                switch(choice)
                {
                case 1:
                   system("cls");
                   window(23,48,20,40);
                   gotoxy(35,18);
                   printf("EDIT RECORDS");
                   gotoxy(25,24);
                   printf(" Enter New Name: ");
```

```
                scanf("%s",item.name);
                size=sizeof(item);
                fseek(file,-size,SEEK_CUR);
                fwrite(&item,sizeof(item),1,file);
                break;
            case 2:
                system("cls");
                window(23,65,20,40);
                gotoxy(35,18);
                printf("EDIT RECORDS");
                gotoxy(25,24);
                c_code(y);
                strcpy(item.code,y);
                size=sizeof(item);
                fseek(file,-size,SEEK_CUR);
                fwrite(&item,sizeof(item),1,file);
                break;
            case 3:
                system("cls");
                window(23,65,20,40);
                gotoxy(35,18);
                printf("EDIT RECORDS");
                gotoxy(25,24);
                printf(" Enter New Rate: ");
```

```
                scanf("%f",&item.rate);
                size=sizeof(item);
                fseek(file,-size,SEEK_CUR);
                fwrite(&item,sizeof(item),1,file);
                break;
            case 4:
                system("cls");
                window(23,65,20,40);
                gotoxy(35,18);
                printf("EDIT RECORDS");
                gotoxy(25,24);
                printf(" Enter New Quantity: ");
                scanf("%d",&item.quantity);
                size=sizeof(item);
                fseek(file,-size,1);
                fwrite(&item,sizeof(item),1,file);
                break;
            }
            gotoxy(27,30);
            printf("--- Item Edited---");
            break;
        }
    }
}
```

```
    }
    if (flag==1)
    {
        gotoxy(32,30);
        printf("Item Does Not Exist.");
        gotoxy(36,32);
        printf("TRY AGAIN");
    }
    getch();
    fclose(file);
    d_mainmenu();
}
```



```
/*function to display all records*/
void d_all()
{
    int i,j=1;
    FILE *file;
    dis_con();
    file=fopen("record.txt","rb");
    rewind(file);
    i=26;
    fflush(file);
    while(fread(&item,sizeof(item),1,file))
```

```
    {
        display(&item,i,j);
        i++;
        j++;
        if ((j%20)==0)
        {
            gotoxy(27,47);/*textcolor(0)*/;
            printf("Press Any Key To See More.....");
            getch();
            system("cls");
            dis_con();
            i=26;
            continue;
        }
    }
    getch();
    if (i==26)
    {
        gotoxy(24,30);
        printf("-- No Articles Found --");
    }
    getch();
    fclose(file);
    d_mainmenu();
```

```
}


/*function to display by quantity*/
void d_quan()
{
   int i,j=1;
   int a,b;
   FILE *file;
   dis_con();
   file=fopen("record.txt","rb");
   rewind(file);
   i=26;
   gotoxy(16,20);;
   printf("Enter Lower Range: ");
   scanf("%d",&a);
   gotoxy(16,21);
   printf("Enter Upper Range:");
   scanf("%d",&b);
   fflush(file);
   while(fread(&item,sizeof(item),1,file))
   {
      if((item.quantity>=a)&&(item.quantity<=b))
      {
         display(&item,i,j);
```

```
            i++;
            j++;
            if ((j%20)==0)
            {
                gotoxy(27,47);
                printf("Press Any Key To See More.....");
                getch();
                system("cls");
                dis_con();
                i=26;
                continue;
            }
        }
    }
    getch();
    if (i==26)
    {
        gotoxy(28,30);
        printf(" No Items Found.");
    }
    getch();
    d_search();
    fclose(file);
}
```

```
/*function to display by rate*/
void d_rate()
{
   int i,j=1;
   float a,b;
   FILE *file;
   dis_con();
   file=fopen("record.txt","rb");
   rewind(file);
   i=26;
   gotoxy(16,20);;
   printf("Enter Lower Range: ");
   scanf("%f",&a);
   gotoxy(16,21);
   printf("Enter Upper Range: ");
   scanf("%f",&b);
   fflush(file);
   while(fread(&item,sizeof(item),1,file))
   {
      if((item.rate>=a)&&(item.rate<=b))
      {
         display(&item,i,j);
         i++;
```

```
            j++;
            if ((j%20)==0)
            {
                gotoxy(27,47);
                printf("Press Any Key To See More.....");
                getch();
                system("cls");
                dis_con();
                i=26;
                continue;
            }
        }
    }
    getch();
    if (i==26)
    {
        gotoxy(28,30);
        printf(" No Item Found ");
    }
    getch();
    fclose(file);
    d_search();
}
```

```
/*function to display by code*/
void d_code()
{
   int i,j=1;
   char x[4]= {0};
   FILE *file;
   dis_con();
   file=fopen("record.txt","rb");
   rewind(file);
   i=26;
   gotoxy(16,20);;
   printf("Enter Item Code: ");
   scanf("%s",x);
   fflush(file);
   while(fread(&item,sizeof(item),1,file))
   {
      if((strcmp(item.code,x)==0))
      {
         display(&item,i,j);
         i++;
         j++;
         break;
      }
   }
```

```
    if (i==26)
    {
        gotoxy(28,30);
        printf("No Item Found");
    }
    getch();
    fclose(file);
    d_search();
}

/*function to display window for item display*/
void dis_con()
{
    int i;
    system("cls");
    gotoxy(20,10);
    ;
    for (i=1; i<=10; i++)
        printf("-");
    printf(" OLD TOWN ");
    for (i=1; i<=10; i++)
        printf("-");
    printf("\n\n");
    gotoxy(30,11);
```

```
    printf("DEPARTMENTAL STORE");
//textcolor(1);
    gotoxy(32,17);
    printf("RECORDS") ;
//textcolor(8);
    gotoxy(18,23);
    printf ("SN.   Item Name   Item Code      Rate     Quantity");
}

/*function to display in screen*/
void display(rec *item,int i,int j)
{
    gotoxy(16,i);//textcolor(13);
    printf("%4d",j);
    printf("%9s",item->name);
    printf("%12s",item->code);
    printf("%14.2f",item->rate);
    printf("%11d",item->quantity);
}

/*function to delete records*/
void del()
{
    int flag;
```

```
    char x[ANS];
    FILE *file,*file1;
    system("cls");
//textbackground(11);
//textcolor(0);
    window(23,51,25,34);
    gotoxy(29,18);
    printf("DELETE ARTICLES");
    gotoxy(27,27);
    printf("Enter Item Code: ");
    scanf("%s",x);
    flag=check(x);
    if(flag==0)
    {
        file1=fopen("record1.txt","ab");
        file=fopen("record.txt","rb");
        rewind(file);
        while (fread(&item,sizeof (item),1,file))
        {
            if(strcmp(item.code,x)!=0)
                fwrite(&item,sizeof(item),1,file1);
        }
        gotoxy(27,29);
        printf("---Item Deleted---");
```

```
        remove("record.txt");
        rename("record1.txt","record.txt");
    }
    if (flag==1)
    {
        gotoxy(25,29);
        printf("---Item Does Not Exist---");
        gotoxy(30,31);
        printf("TRY AGAIN");
    }
    fclose(file1);
    fclose(file);
    getch();
    d_mainmenu();
}

/*function to check validity of code while editing and deleting*/
int check(char x[ANS])
{
    FILE *file;
    int flag=1;
    file=fopen("record.txt","rb");
    rewind(file);
    while (fread(&item,sizeof (item),1,file))
```

```
    {
        if(strcmp(item.code,x)==0)
        {
            flag=0;
            break;
        }
    }
    fclose(file);
    return flag;
}

/*function to display box*/
void window(int a,int b,int c,int d)
{
    int i;
    system("cls");
    gotoxy(20,10);
//textcolor(1);
    for (i=1; i<=10; i++)
        printf("-");
    printf(" OLD TOWN ");
    for (i=1; i<=10; i++)
        printf("-");
    printf("\n\n");
```

```
   gotoxy(30,11);
   printf("DEPARTMENTAL STORE");
//textcolor(4);
   for (i=a; i<=b; i++)
   {
      gotoxy(i,17);
      printf("\xcd");
      gotoxy(i,19);
      printf("\xcd");
      gotoxy(i,c);
      printf("\xcd");
      gotoxy(i,d);
      printf("\xcd");
   }

   gotoxy(a,17);
   printf("\xc9");
   gotoxy(a,18);
   printf("\xba");
   gotoxy(a,19);
   printf("\xc8");
   gotoxy(b,17);
   printf("\xbb");
   gotoxy(b,18);
```

```
   printf("\xba");
   gotoxy(b,19);
   printf("\xbc");
//textcolor(4);
   for(i=c; i<=d; i++)
   {
      gotoxy(a,i);
      printf("\xba");
      gotoxy(b,i);
      printf("\xba");
   }
   gotoxy(a,c);
   printf("\xc9");
   gotoxy(a,d);
   printf("\xc8");
   gotoxy(b,c);
   printf("\xbb");
   gotoxy(b,d);
   printf("\xbc");
//textbackground(11);
//textcolor(0);
}
```

c programs

1.fibonacci series

```
#include<stdio.h>
int main()
{
 int n1=0,n2=1,n3,i,number;
 printf("Enter the number of elements:");
 scanf("%d",&number);
 printf("\n%d %d",n1,n2);//printing 0 and 1
 for(i=2;i<number;++i)//loop starts from 2 because 0 and 1 are already printed
 {
  n3=n1+n2;
  printf(" %d",n3);
  n1=n2;
  n2=n3;
 }
  return 0;
 }
```

****

2.prime no.

```
#include<stdio.h>
int main(){
int n,i,m=0,flag=0;
printf("Enter the number to check prime:");
scanf("%d",&n);
m=n/2;
for(i=2;i<=m;i++)
{
if(n%i==0)
{
printf("Number is not prime");
flag=1;
break;
}
}
if(flag==0)
printf("Number is prime");
return 0;
 }
```


****

3.pallindrome no.

```
#include<stdio.h>
int main()
{
int n,r,sum=0,temp;
printf("enter the number=");
scanf("%d",&n);
temp=n;
while(n>0)
{
r=n%10;
sum=(sum*10)+r;
n=n/10;
}
if(temp==sum)
printf("palindrome number ");
else
printf("not palindrome");
return 0;
```

****

5. factorial no

```
#include<stdio.h>
int main()
{
 int i,fact=1,number;
 printf("Enter a number: ");
  scanf("%d",&number);
    for(i=1;i<=number;i++){
      fact=fact*i;
  }
  printf("Factorial of %d is: %d",number,fact);
return 0;
}
```


****

6.factorial using recursion

```
#include<stdio.h>

long factorial(int n)
{
  if (n == 0)
    return 1;
  else
    return(n * factorial(n-1));
}

void main()
{
  int number;
  long fact;
  printf("Enter a number: ");
  scanf("%d", &number);

  fact = factorial(number);
  printf("Factorial of %d is %ld\n", number, fact);
  return 0;
}
```


****

7. armstrong no.

```
#include<stdio.h>
 int main()
{
int n,r,sum=0,temp;
printf("enter the number=");
scanf("%d",&n);
temp=n;
while(n>0)
{
r=n%10;
sum=sum+(r*r*r);
n=n/10;
}
if(temp==sum)
printf("armstrong  number ");
else
printf("not armstrong number");
return 0;
}
```

8. sum of digits program

```
#include<stdio.h>
 int main()
{
int n,sum=0,m;
printf("Enter a number:");
scanf("%d",&n);
while(n>0)
{
m=n%10;
sum=sum+m;
n=n/10;
}
printf("Sum is=%d",sum);
return 0;
}
```


****

9.reverse no program

```
#include<stdio.h>
 int main()
{
int n, reverse=0, rem;
printf("Enter a number: ");
  scanf("%d", &n);
  while(n!=0)
  {
     rem=n%10;
     reverse=reverse*10+rem;
     n/=10;
  }
  printf("Reversed Number: %d",reverse);
return 0;
}
```


****

10. swap two no without third variable program

```
#include<stdio.h>
 int main()
{
int a=10, b=20;
printf("Before swap a=%d b=%d",a,b);
a=a+b;//a=30 (10+20)
b=a-b;//b=10 (30-20)
a=a-b;//a=20 (30-10)
printf("\nAfter swap a=%d b=%d",a,b);
return 0;
}
```

****

*****

```
#include<stdio.h>
#include<stdlib.h>
 int main()
{
int a=10, b=20;
printf("Before swap a=%d b=%d",a,b);
a=a*b;//a=200 (10*20)
b=a/b;//b=10 (200/20)
a=a/b;//a=20 (200/10)
 system("cls");
printf("\nAfter swap a=%d b=%d",a,b);
return 0;
}
```

****

11. assembly program

```
#include<stdio.h>
void main() {
  int a = 10, b = 20, c;

  asm {
     mov ax,a
     mov bx,b
     add ax,bx
     mov c,ax
  }
  printf("c= %d",c);
}
```


****

13. matrix multiplication

```
#include<stdio.h>
#include<stdlib.h>
int main(){
int a[10][10],b[10][10],mul[10][10],r,c,i,j,k;
system("cls");
printf("enter the number of row=");
```

```
scanf("%d",&r);
printf("enter the number of column=");
scanf("%d",&c);
printf("enter the first matrix element=\n");
for(i=0;i<r;i++)
{
for(j=0;j<c;j++)
{
scanf("%d",&a[i][j]);
}
}
printf("enter the second matrix element=\n");
for(i=0;i<r;i++)
{
for(j=0;j<c;j++)
{
scanf("%d",&b[i][j]);
}
}

printf("multiply of the matrix=\n");
for(i=0;i<r;i++)
{
for(j=0;j<c;j++)
{
mul[i][j]=0;
for(k=0;k<c;k++)
{
mul[i][j]+=a[i][k]*b[k][j];
}
}
}
```

```
//for printing result
for(i=0;i<r;i++)
{
for(j=0;j<c;j++)
{
printf("%d\t",mul[i][j]);
}
printf("\n");
}
return 0;
}
```

****

***

C PROGRAM COMPILATION PROCESS

get to current path

cd .. ---------> one path hierarchy backword

reach to drive

to change drive

type drive_name: press enter

now dir command to show directory

cd directory_name to go to particular directory

how to compile c programs

in command prompt

type

```
gcc filename.c -ofilename
```

press enter

to run file

```
type filename
```

press enter

DATE MODIFIED September 2024

THANK YOU !!!

If You Read All Programs Then Congratulations

You Actually Learned Something About C.