CL1002 ***Programming Fundamentals Lab***	Lab 05 Nested Structures, Operators (Logical, Conditional, Bitwise, Modulus)

NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES

Fall 2024

Author: Muhammad Nouman Hanif

AIMS AND OBJECTIVES

Aim:

To understand and implement nested decision structures and operators in C programming. This lab will help students apply logical, conditional, bitwise, and modulus operators, and develop skills in handling complex decision-making scenarios using nested if-else statements.

Objectives:

- To learn how to use nested if-else statements for more complex decision-making.
- To understand and apply various operators, including logical, conditional (ternary), bitwise, and modulus.
- To write efficient and readable C programs using these concepts.
- To solve real-world problems by applying nested decision structures and operators.

Introduction

In this lab, we'll dive deeper into decision-making in C programming. While you've already learned the basics of if-else and switch statements, this session will explore how these decision structures can be nested to handle more complex conditions. We'll also look at operators that are crucial for making decisions, such as logical operators (AND, OR, NOT), conditional (ternary) operators, bitwise operators, and the modulus operator. By the end of this lab, you should be able to construct more intricate programs that involve multiple conditions and make decisions accordingly.

Section 1: Nested if-else Statements

Explanation: Nested if-else statements allow you to make a decision within another decision. This is particularly useful when multiple conditions need to be evaluated sequentially. For example, if you're building a grading system, you might want to check if a student's score is above a certain threshold, and within that, decide if it's an 'A' or a 'B'.

Example 1:

```
#include <stdio.h>

int main() {
    int score;
    printf("Enter your score: ");
    scanf("%d", &score);

    if (score >= 90) {
        if (score >= 95) {
            printf("Grade: A+\n");
        } else {
            printf("Grade: A\n");
        }
    } else if (score >= 80) {
        if (score >= 85) {
            printf("Grade: B+\n");
        } else {
            printf("Grade: B\n");
        }
    } else {
        printf("Grade: C or lower\n");
    }

    return 0;
}
```

Example 2:

This program asks the user for the temperature and categorizes it into different levels using nested if-else statements. The innermost condition checks if the temperature is extremely hot.

```
#include <stdio.h>

int main() {
  int temperature;
  printf("Enter the temperature in Celsius: ");
  scanf("%d", & temperature);
  if (temperature >= 30) {
    if (temperature >= 40) {
      printf("It's very hot outside!\n");
    } else {
      printf("It's hot outside.\n");
   }
  } else if (temperature >= 20) {
    printf("It's warm outside.\n");
  } else if (temperature >= 10) {
    printf("It's cool outside.\n");
  } else {
    printf("It's cold outside.\n");
 }
  return 0;
}
```

References:

- C Programming Absolute Beginner's Guide by Greg Perry and Dean Miller
- Programming in ANSI C by E. Balagurusamy.

Problems:

1. Write a program that categorizes a person's age into different life stages: Child, Teenager, Adult, and Senior, using nested if-else statements.
2. Create a program that determines if a number is positive, negative, or zero, and if it's positive, checks if it's an even or odd number.

Section 2: Logical Operators

Explanation: Logical operators are used to combine two or more conditions. The common logical operators in C are && (AND), || (OR), and ! (NOT). These are essential in making decisions where multiple conditions need to be true or where at least one condition needs to be true.

Example 1:

```
#include <stdio.h>

int main() {
    int age = 20;
    int hasLicense = 1;

    if (age >= 18 && hasLicense) {
        printf("You are eligible to drive.\n");
    } else {
        printf("You are not eligible to drive.\n");
    }

    return 0;
}
```

Example 2:

```
#include <stdio.h>

int main() {
    int score1, score2, score3;
    printf("Enter three test scores: ");
    scanf("%d %d %d", &score1, &score2, &score3);

    if (score1 > 50 && score2 > 50 && score3 > 50) {
        printf("You passed all the tests.\n");
    } else {
        printf("You did not pass all the tests.\n");
    }

    return 0;
}
```

This example uses logical AND (&&) to determine if a student passed all tests by checking if each score is above 50. If all conditions are met, the program outputs that the student passed; otherwise, it indicates a failure.

References:

- The C Programming Language by Brian W. Kernighan and Dennis M. Ritchie.
- Let Us C by Yashavant Kanetkar.

Problems:

1. Write a program that checks if a number is divisible by both 3 and 5 using logical operators.
2. Create a program that checks if a person is eligible to vote based on their age and citizenship status.

Section 3: Conditional (Ternary) Operators

Explanation: The conditional (ternary) operator is a compact way to make decisions. It is used as a shorthand for the if-else statement and can make the code more concise.

Example 1:

```
#include <stdio.h>

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

    (number % 2 == 0) ? printf("Even\n") : printf("Odd\n");

    return 0;
}
```

Example 2:

This example uses the ternary operator to check if the user is eligible to vote based on their age. It's a concise alternative to the if-else statement.

```
#include <stdio.h>

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

    age >= 18 ? printf("You are eligible to vote.\n") :
printf("You are not eligible to vote.\n");

    return 0;

}
```

References:

- C Primer Plus by Stephen Prata.
- C Programming: A Modern Approach by K. N. King.

Problems:

1. Write a program using a ternary operator to find the maximum of two numbers.
2. Use the ternary operator to check if a number is positive, negative, or zero.

Section 4: Bitwise Operators

Explanation: Bitwise operators allow you to manipulate individual bits of data. They are often used in low-level programming, where performance and memory efficiency are critical.

Example 1:

```
#include <stdio.h>
int main() {
    int a = 5;  // binary: 0101
    int b = 9;  // binary: 1001
    printf("a & b = %d\n", a & b);  // AND operation
    printf("a | b = %d\n", a | b);  // OR operation
    printf("a ^ b = %d\n", a ^ b);  // XOR operation
    printf("~a = %d\n", ~ );        // NOT operation

    return 0;
}
```

Example 2:

This program demonstrates the bitwise left shift (<<) operator. It shifts the bits of a one position to the left, effectively multiplying the value by 2.

```
#include <stdio.h>
int main() {
    int a = 5;  // binary: 0101
    int result = a << 1;  // Left shift operation
    printf("Result after left shift: %d\n", result);  // Outputs
10, binary: 1010

    return 0;
}
```

References:

- Expert C Programming: Deep C Secrets by Peter van der Linden.
- Understanding and Using C Pointers by Richard Reese.

Problems:

1. Write a program to swap two numbers using bitwise XOR.
2. Create a program that counts the number of 1s in the binary representation of a number.

Section 5: Modulus Operator

Explanation: The modulus operator (%) returns the remainder of a division operation. It's often used to determine if a number is even or odd, or to perform operations that require cyclical behavior.

Example 1:

```
#include <stdio.h>

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

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

    return 0;
}
```

Example 2:

This program checks whether a number is divisible by 5 using the modulus operator (%). If the remainder is zero, it outputs that the number is divisible by 5.

```
#include <stdio.h>

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

    if (num % 5 == 0) {
        printf("The number is divisible by 5.\n");
    } else {
        printf("The number is not divisible by 5.\n");
    }

    return 0;
}
```

References:

- Head First C by David Griffiths and Dawn Griffiths.
- The C Programming Tutor by Mark Burgess.

Problems:

1. Write a program that checks if a year is a leap year using the modulus operator.
2. Create a program that calculates the sum of digits of a number until the result is a single digit (e.g., 123 -> 6).

More Problems ;D

1. Write a program to find the greatest of three numbers using nested if-else statements.
2. Create a program that calculates the final grade of a student based on multiple criteria, including attendance, assignment scores, and exam results, using nested decision structures.
3. Write a program that uses bitwise operators to perform encryption and decryption of a character.
4. Develop a program that uses logical operators to determine if a person is eligible for a loan based on age, income, and credit score.