CL1002 P*rogramming Fundamentals Lab*	Lab 05 Nested Decision Structures

NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES

Fall 2024

Aims and Objectives

Aims:

Manage Complex Decision Making:

- To handle complex decision processes where multiple conditions must be checked in a specific order, often depending on earlier decisions.

Increase Program Flexibility:

- To allow more nuanced and flexible control flows in a program, enabling the program to respond differently based on combinations of multiple conditions.

Create Modular Logic:

- To build programs that can handle decisions in a modular way, separating different logical branches for clearer and more organized code.

Achieve Precise Outcomes:

- To ensure that the program can execute precise operations by evaluating a series of conditions and making the right decisions based on the context.

Objectives:

Efficiently Handle Multiple Scenarios:

- To allow the program to handle multiple scenarios or conditions effectively without requiring excessive repetition of code.

Simplify Complex Condition Checking:

- To break down complicated decisions into smaller, more manageable conditions, making the code easier to understand and maintain.

Support Hierarchical Decision Logic:

- To support decisions that depend on previous outcomes, where a specific condition can only be evaluated if another has already been satisfied or failed.

Optimize Code Readability and Maintenance:

- Structure code in a way that is easier to read and maintain, with clearly defined decision-making paths.

Minimize Redundancy:

- To reduce code duplication by nesting decisions, allowing for more efficient logic where common conditions can be reused in different decision branches.

Introduction

In programming, decision-making structures are fundamental for controlling the flow of execution based on certain conditions. Nested decision structures refer to a form of decision-making where one decision statement is placed inside another. This allows a program to make more complex decisions by evaluating multiple conditions in a hierarchical or dependent manner.

When a program requires more than a simple yes-or-no decision, nested decisions come into play. For instance, certain actions may only be taken if an initial condition is satisfied, and within that, further decisions may need to be made. These structures are essential when dealing with real-world problems, where multiple factors often influence the outcome.

Nested If-Else Statements

Nested decision structures are frequently used in various fields such as data processing, game development, business applications, and AI, where multiple layers of conditions need to be evaluated to arrive at the correct decision. By using these structures, programmers can write more efficient and adaptable code, which can handle complex logic seamlessly.

The use of nested decision structures introduces clarity and organization in complex decision-making processes, making the program not only functional but also easier to maintain and expand.

Placing the block of if else statement inside an existing if or else block statement is called nested If else statement. Each block of nested if else, logically perform same as simple if else statements. Whenever a user wants to check more than one condition at a time, the appropriate way is to use nested if-else statements. Following is the structure of nested if else statement in an algorithm.

```
IF (logical-expression) THEN
    statements
    IF (logical-expression) THEN
        statements
    ELSE
        statements
    END IF
    statements
ELSE
    statements
    IF (logical-expression) THEN
        statements
    END IF
    statements
END IF
This structure
```

EXAMPLE 1

Let’s consider the problem of finding the largest value if three variables are given from the user.

```
Input X, Y,Z
If(X>Y) then
   If(X>Z) then
         Max= X [X>Y, X>Z]
   Else
         Max= Z [Z>X>Y]
   Endif
Else
   If(Y>Z) then
         Max = Y [Y>X, Y>Z]]
   Else
         Max = Z [Z>Y>X]
   Endif
Endif
Print “The largest number is”, Max
```

Flowchart:

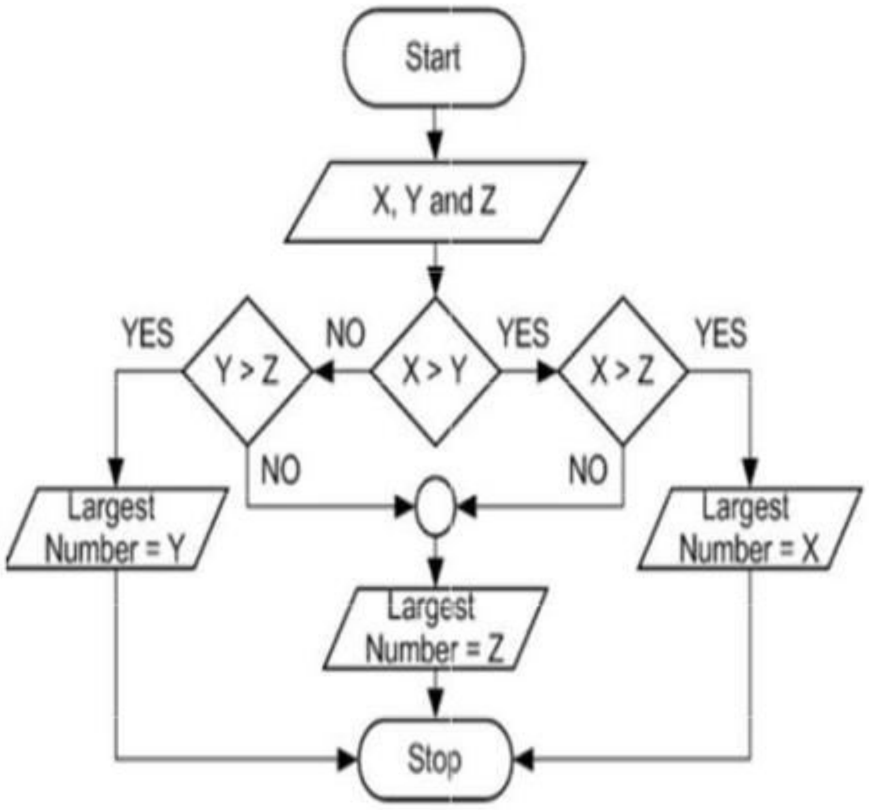


Figure 1. Flowchart of a Nested IF-ELSE Statement

C-Implementation:

```c
#include<stdio.h>

int main(){
    int x,y,z;
    printf("Enter value of X");
    scanf("%d",&x);
    printf("Enter value of Y");
    scanf("%d",&y);
    printf("Enter value of Z");
    scanf("%d",&z);
    if(x>y){
            if(x>z){
                    printf("The largest value is of x = %d",x);
            }
            else{
                    printf("The largest value is of z =%d", z);
            }
    }
    else{
            if(y>z){
                    printf("The largest value is of y= %d",y);
            }
            else{
                    printf("The largest value is of z= %d",z);
            }
    }
}
```

NESTED SWITCH-CASE STATEMENT

Placing the simple switch case statements inside an existing case statement is called nested switch-case statement. Each block of nested switch case statement logically performs the same as simple switch case statement. Following is the syntax of nested switch case statement.

```
Switch(controlling expression){
    Label set 1:
            Statement 1;
            Break;
    Label set 2:
            Statement 2;
            Switch(controlling expression){
                    Label set 1:
                            Statement 1;
                            Break;
                    Label set 2:
                            Statement 2;
                            Break;
                    Default:
                            Statement d;
            }
            Break;
    Default:
            Statement d;
}
```

EXAMPLE 2

Problem

Ayesha is interested in knowing the names of different countries. She wants a list of countries by just giving a starting and ending letter.

C-Implementation

```
#include <stdio.h>
int main()
{
   char start,e;
   printf("Please say starting letter of country");
   scanf("%c",&start);
   switch(start)
   {
        case 'A':
        case 'a':
             printf("Please say ending letter\n");
             scanf("\n%c",&e);
             switch(e)
             {
                  case 'A':
                  case 'a':
                       printf("\n Alaska \n Albania \n Algeria");
                       break;
                  default:
                       printf("\n No such country");
             }
             break;

        case 'B':
        case 'b':
             printf("Please say ending letter\n");
             scanf("\n%c",&e);
             switch(e)
             {
                  case 'A':
                  case 'a':
                       printf("\n Bulgeria \n Bolivia \n Botswana");
                       break;
                  default:
                       printf(" No such country");


             }
                  break;

        default:
             printf("Please type correct letter");
        }
}
```

TERNARY OPERATORS IN C

A ternary operator is a shorthand way of writing an if-else conditional statement in a single line of code. It is often used to simplify the code, making it more concise and readable in situations where you want to assign a value based on a condition.

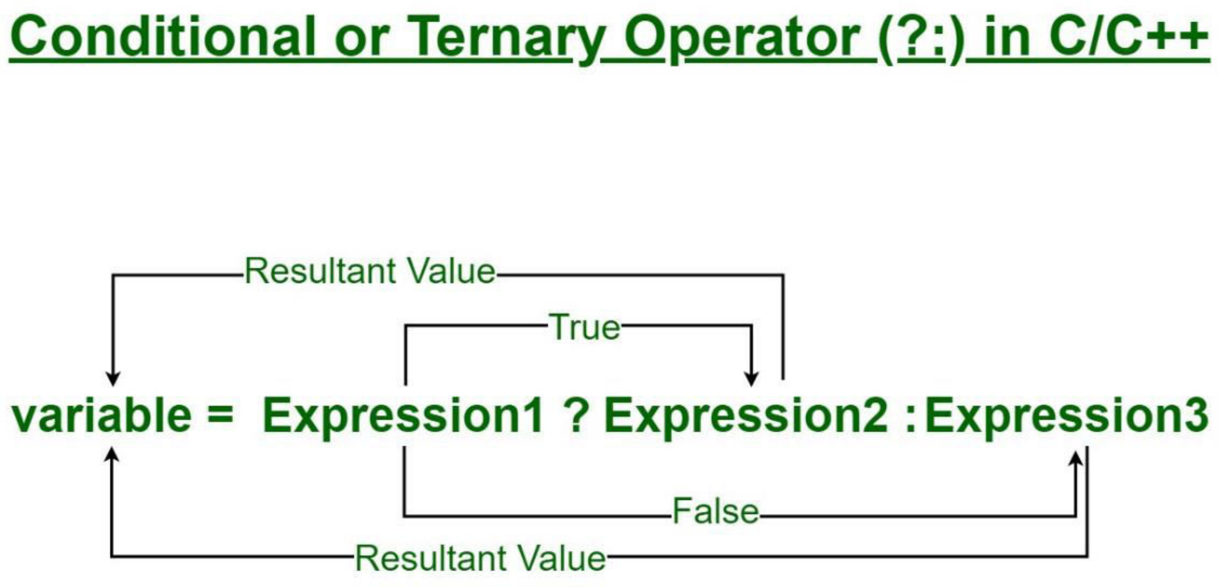

Conditional or Ternary Operator (?:) in C/C++
Resultant Value
True
variable = Expression1 ? Expression2 : Expression3
False
Resultant Value

Figure 2. Syntax of ternary operators in C

There can be variations of the ternary operators in C. The conditional operator is of the form

```
variable = Expression1 ? Expression2 : Expression3
```

Or the syntax will also be in this form

```
variable = (condition) ? Expression2 : Expression3
```

Or syntax will also be in this form

```
(condition) ? (variable = Expression2) : (variable =
Expression3)
```

The conditional operator is kind of like the if-else statement as it does follow the same algorithm as of if-else statement, but the conditional operator takes less space and helps to write the if-else statements in the shortest way possible.

Flow Chart of a ternary operator operations in C.

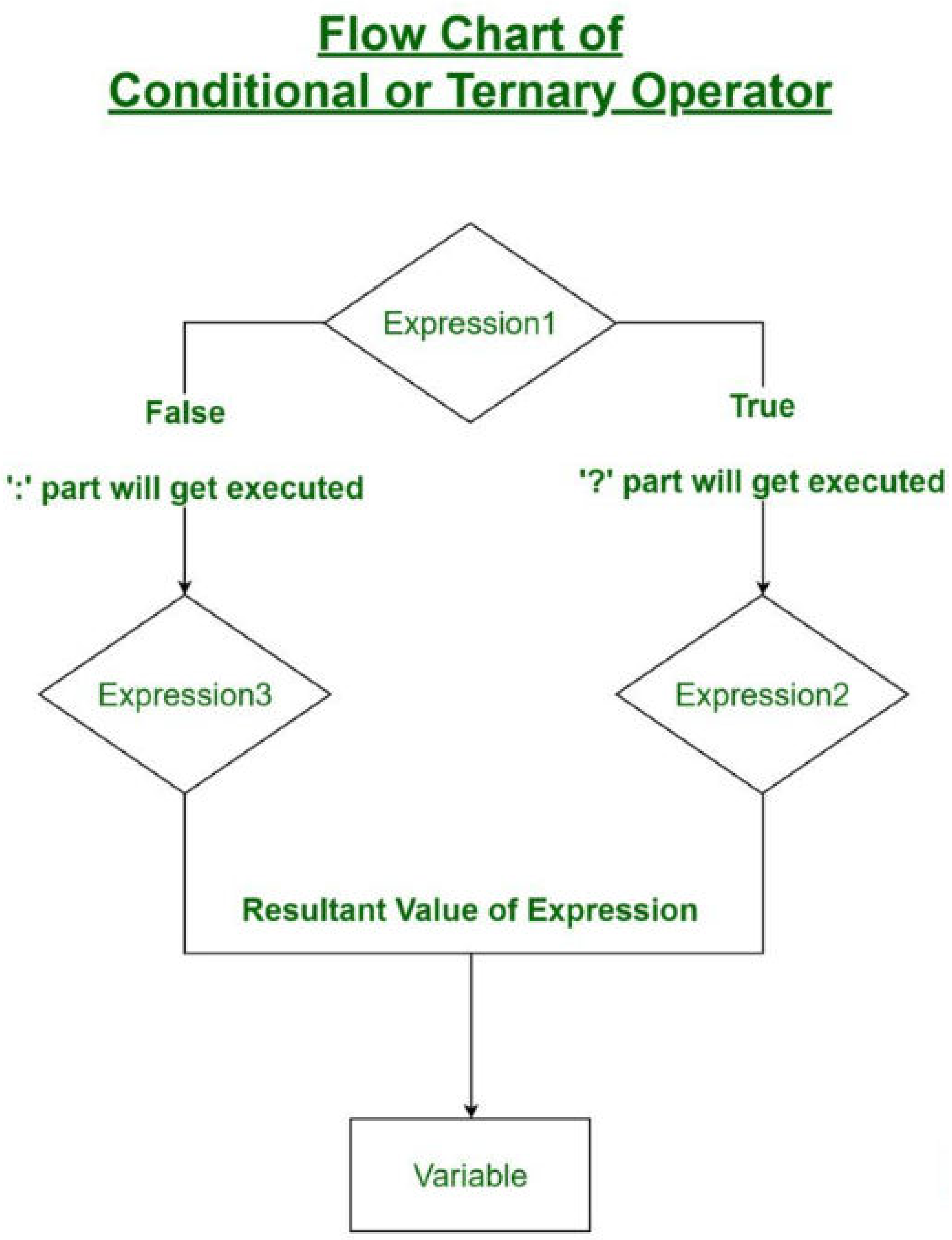

Flow Chart of
Conditional or Ternary Operator
Expression1
False
True
':' part will get executed
'?' part will get executed
Expression3
Expression2
Resultant Value of Expression
Variable

Figure 3. Flowchart of Ternary operators operations

EXAMPLE 3:

```
#include <stdio.h>

int main()
{
   int m=5, n=4;

   (m>n) ? printf("M is greater than n") : printf("n is
greater than M);
   return 0;
}
```

EXAMPLE 4

```
#include <stdio.h>

int main()
{
   int a=1, b=2, ans;
   //Nested Ternary operator
   ans=(a==1?(b==2?3:5):0);
   Printf("%d\n",ans);
   return 0;
}
```

Exercises:

Task1:

Write a program that asks for the number of calories and fat grams in a food. The program should display the percentage of calories that come from fat

One gram of fat has 9 calories, so Calories from fat = fat grams * 9

The percentage of calories from fat can be calculated as: calories from fat/total calories

Input validation: Make sure the number of calories and fat grams are not less than 0.

Also, the number of calories from fat cannot be greater than the total number of calories. If that happens, display an error message indicating that either the calories or fat grams were incorrectly entered.

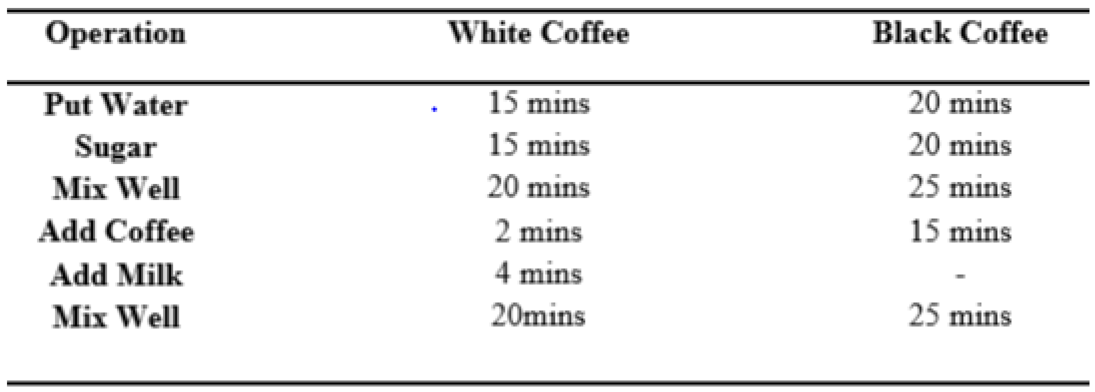
Task2:

The weekday is true if it is a weekday, and the vacation is true if we are on vacation. We sleep in if it is not a weekday or we're on vacation. Print true if we sleep in.

sleepIn(false, false) → true
sleepIn(true, false) → false
sleepIn(false, true) → true

Task3:

Write a program to control a coffee machine. Allow the user to input the type of coffee as B for Black and W for White. Ask the user if the cup size is double and if the coffee is manual. The following table details the time chart for the machine for each coffee type. Display a statement for each step. If the coffee size is double, increase the baking time by 50 percent. Use functions to display instructions to the user and to compute the coffee time.

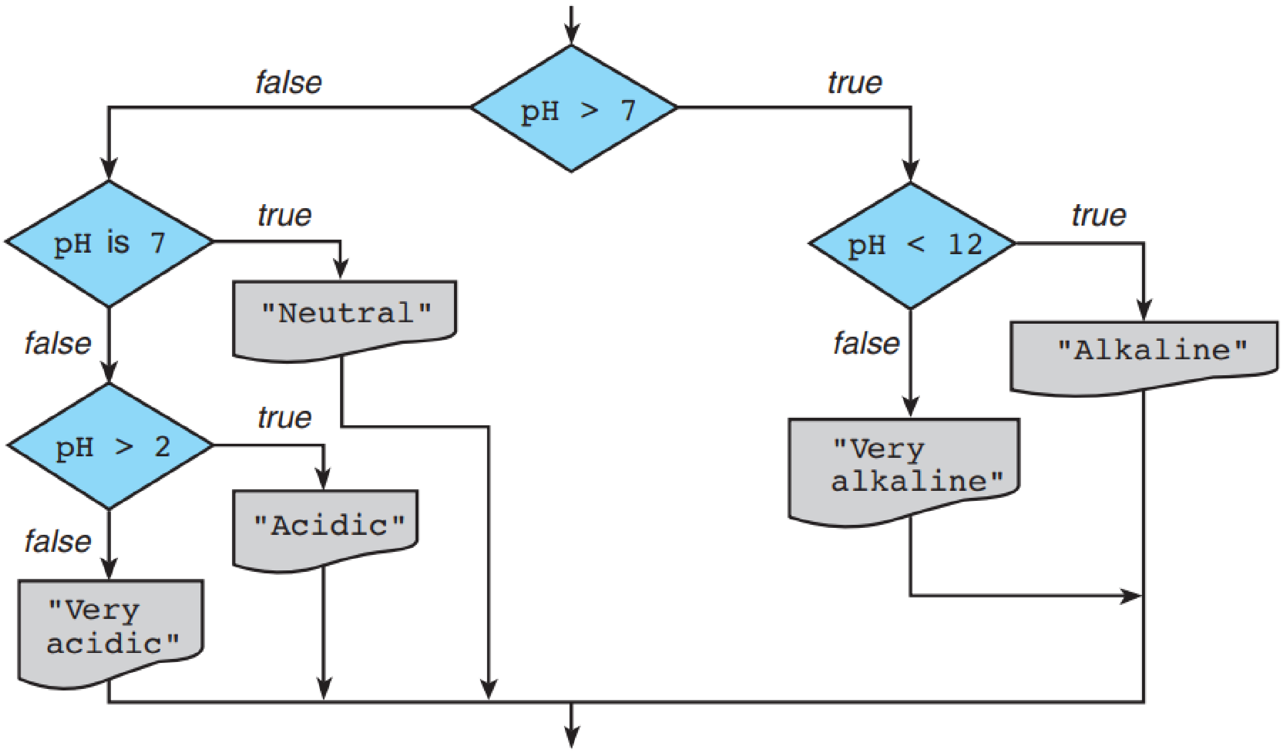


Operation	White Coffee	Black Coffee
Put Water	15 mins	20 mins
Sugar	15 mins	20 mins
Mix Well	20 mins	25 mins
Add Coffee	2 mins	15 mins
Add Milk	4 mins	-
Mix Well	20mins	25 mins

Note: Use switch structure to solve this problem.

Task4:

Write a nested if statement for the decision diagrammed in the accompanying flowchart. Use a multiple-alternative if statements for intermediate decisions where possible.



Task5:

Implement the following decision table using nested ternary operators. Assume that the grade point average is within the range 0.0 through 4.0.

Grade Point Average	Transcript Message
0.0–0.99	Failed semester—registration suspended
1.0–1.99	On probation for next semester
2.0–2.99	(no message)
3.0–3.49	Dean’s list for semester
3.5–4.00	Highest honors for semester

Task6:

```c
#include <stdio.h>
int main()
{
    int x;
    x = 5 > 8 ? 10 : 1 != 2 < 5 ? 20 : 30;
    printf("Value of x:%d", x);
    return 0;
}
```

Explain why the output is 30

Task7:

Using Ternary operators write a C-Program that can identify if the last digit of the number is zero or non-zero. If its zero, the program should print "Last Digit is zero" if it's non-zero the program should print "Last Digit non-zero"

Example:
Input 5; Output: Last Digit Non-Zero

Input 20; Output: Last Digit Zero

Task 8:
Consider yourself in a situation where you must categorize a student at Fast-NUCES Karachi campus. He has only provided you with the following details.

Roll number (He wasn't kind so that's all we have to work with).

Your job is to figure out what year is his registration. Which semester he/she is currently enrolled in and which section he/she is a part of. Note: You can use the char roll_number[7] data type to store data which I have explained in Lab. Use Ternary operators to accomplish this.

Hint: Getting the registration of the student is easy, trouble is the semester and section. To find out which semester he/she is enrolled in we can use a simple method of subtracting his registration number from the current year (i.e. 24-22 (from 22k-6412)) this gives us 2 meaning he has passed 2 years here at fast and it's the fall semester so 5th would be his semester no.

Now for class, the first digital indicates which section is allocated to the student i.e. 6412 meaning he has the sixth section allocated and comparing the sixth alphabet that would be F.