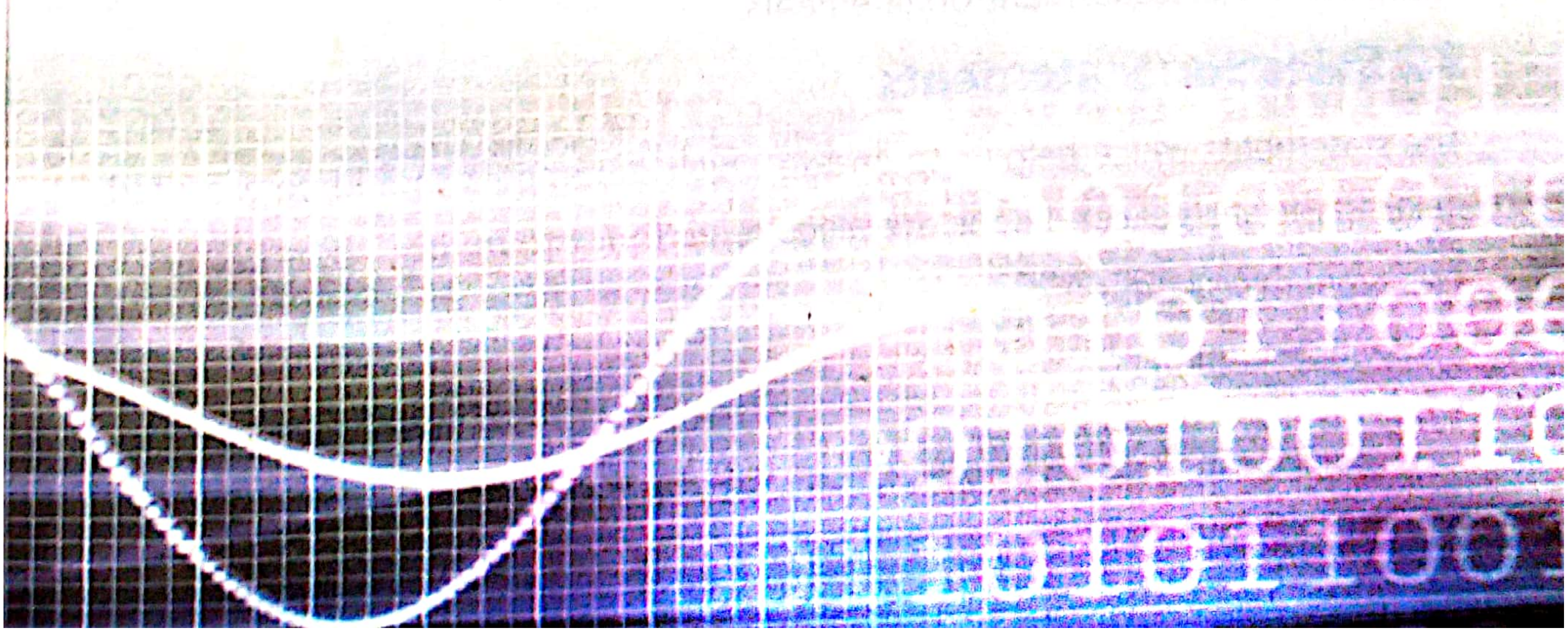
Unit # 3

CONDITIONAL LOGIC

Students Learning Outcomes

After completing this unit students will be able to

- Define a control statement
- Define a selection statement
- Know the structure of *if statement*
- Use *if statement*
- Know the structure of *if-else statement*
- Use nested selection structures



Unit Introduction

In our daily life, we often do certain tasks depending upon the situation e.g. I will go for a walk if I wake up at 6 am. If the weather turns cloudy, I will take umbrella with me. If Sara passes the exam, I will gift her a watch. All these decisions are taken on the basis of condition. If the condition is true, we perform the specified task, otherwise we do not. Sometimes, if the condition is not true then we perform some other task. This is called ***conditional logic***. In this chapter, we discuss how to implement conditional logic in C programming language.

3.1 Control Statements

In order to solve a problem, there is a need to control the flow of execution of a program. Sometimes we need to execute one set of instructions if a particular condition is true and another set of instructions if the condition is false. Moreover, sometimes we need to repeat a set of statements for a number of times. We can control the flow of program execution through control statements. There are three types of control statements in C language.

1- Sequential Control Statements
2- Selection Control Statements
3- Repetition Control Statements

Sequential control is the default control structure in C language. According to the sequential control, all the statements are executed in the given sequence. Till now, we have just worked according to the sequential control. In this chapter, our focus is on the selection control statements.

3.2 Selection Statements

The statements which help us to decide which statements should be executed next, on the basis of conditions, are called ***selection statements***.

Two types of selection statements are:

1. If statement
2. If-else statement

3.2.1 If statement

C language provides ***if statement*** in which we specify a condition, and associate a code to it. The code gets executed if the specified condition turns out to be *true*, otherwise the code does not get executed.

Structure of if statement

If statement has the following structure in C language:

```
if (condition)
    Associated Code
```

Here is a brief description of different components involved in the general structure of *if statement*.

1- In the given structure, ***if*** is a keyword that is followed by a *condition* inside parentheses **( )**.

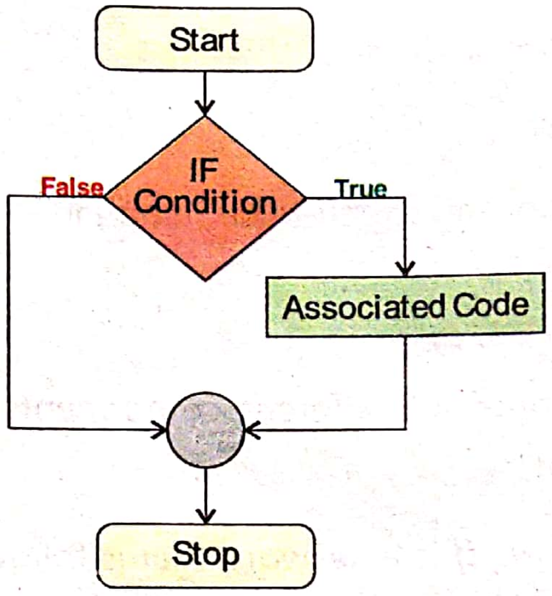
2- A ***condition*** could be any valid expression including arithmetic expressions, relational expressions, logical expressions, or a combination of these. Here are a few examples of valid expressions that can be used as condition.

a-	5	(true)
b-	5 + 4	(true)
c-	5 – 5	(false)
d-	5 > 4	(true)
e-	5 == 4	(false)
f-	! (4 > 5)	(true)
g-	(5 > 4) && (10 < 9)	(false)
h-	(5 > 4) || (9 < 10)	(true)

Any expression that has a non-zero value calculates to *true*, e.g. expressions *a* and *b* above produce a *true* value, but the expression *c* produces a *false* value. The expression can also include variables, in that case values inside the variables are used to calculate the true/ false value of the expression.

3- The ***associated code*** is any valid C language set of statements. It may contain one or more statements.

The following flow chart shows the basic flow of an *if statement*.



If we want to associate more than one statements to an *if statement*, then they need to be enclosed inside a { } block, but if we want to associate only one statement, then although it may be enclosed inside { } block, but it is not mandatory. It is demonstrated through the following examples.

</> EXAMPLE CODE 3.1

```
#include<stdio.h>
void main()
{
        int a = 12;
        if (a % 2 == 0)
        {
                printf("The variable a contains an even value.");
                printf("\nYou are doing a great job.");
        }
}
```

Output:

```
The variable a contains an even value.
You are doing a great job.
```

Because, when value 12 is divided by 2, it gives a remainder equal to 0, so the condition inside if parentheses is true. As both the *printf* statements are inside { } block, so both the statements get executed.

Now look at the following example:

EXAMPLE CODE 3.2

```
#include<stdio.h>
void main()
{
        int a = 4;
        int b = 5;
        if (a > b)
                printf("The value of a is greater than b.");
        printf("\nYou are doing a great job.");
}
```

Output:

```
You are doing a great job.
```

As the condition inside the *if parentheses* is false, and the statements following the *if statement* are not inside { } block, so only the 2nd statement is executed because without a { } block, only 1st statement is considered to be associated with the *if statement*.

Using if statement in C

Let's understand the concept of if statement using different examples.

PROGRAMMING TIME 3.1

Problem:

Write a program in C language that takes the percentage of student as an input and displays "PASS" if the percentage is above 50.

```
#include <stdio.h>
void main()
{
        float percentage;
        printf ("Enter the percentage: ");
```

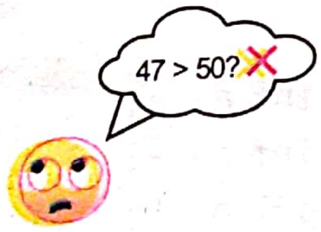
Continued

```
        scanf ("%f", &percentage);
        if (percentage > 50)
                printf ("PASS\n");
}
```

Output:

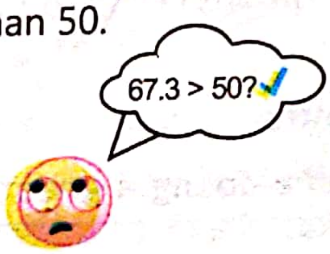
On the input 47, program simply ends because 47 is less than 50 and the condition turns *false*.

```
Enter the percentage : 47
```



When 67.3 is entered as an input, "PASS" gets printed on console because condition is *true*, as 67.3 is greater than 50.

```
Enter the percentage : 67.3
PASS
```



PROGRAMMING TIME 3.2

Problem:

A marketing firm calculates the salary of its employees according to the following formula.

Gross Salary = Basic Salary + (Number of Items Sold X 8) + Bonus

If the number of sold items are more than 100 and the number of broken items are 0, then bonus is Rs. 10000, otherwise bonus is 0.

Write a program that takes basic salary, the number of sold and broken items as input from user, then calculates and displays the gross salary of the employee.

Continued

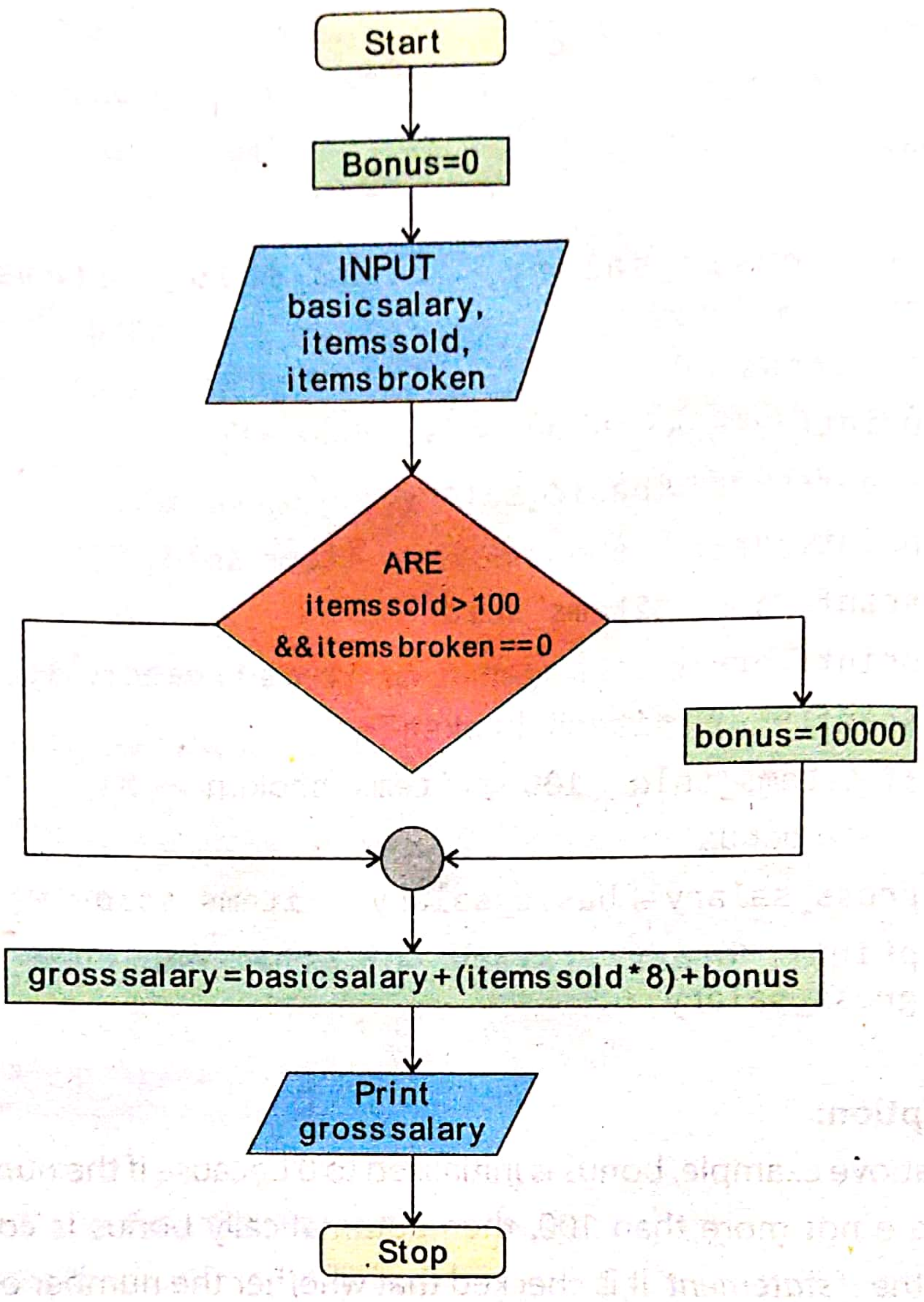
Program:

```
#include<stdio.h>
void main()
{
        int  basic_salary,  items_sold,  items_broken,
        gross_salary;
        int bonus = 0;
        printf("Enter the basic salary: ");
        scanf("%d", &basic_salary);
        printf("Enter the number of items sold: ");
        scanf("%d", &items_sold);
        printf("Enter the number of items broken: ");
        scanf("%d", &items_broken);
        if (items_sold > 100 && items_broken == 0)
                bonus = 10000;
        gross_salary = basic_salary + (items_sold * 8) + bonus;
        printf("Gross  salary  of  the  employee  is  %d",
        gross_salary);
}
```

Description:

In the above example, bonus is initialized to 0 because if the number of sold items are not more than 100, then automatically bonus is considered 0. Inside the *if statement*, it is checked that whether the number of sold items are greater than 100. If so, the bonus is assigned 10000. It is to be noted that gross salary is calculated outside the if block, because whether the number of sold items are more than 100 or not, the gross salary must be calculated.

Following figure explains the flow of program with the help of a flow chart.


Start
Bonus=0
INPUT basic salary, items sold, items broken
ARE items sold > 100 && items broken == 0
bonus=10000
gross salary = basic salary + (items sold * 8) + bonus
Print gross salary
Stop

ACTIVITY 3.1

Write a program that takes the age of a person as an input and displays "Teenager" if the age lies between 13 and 19.

ACTIVITY 3.2

Write a program that takes year as input and displays "Leap Year" if the input year is leap year. Leap years are divisible by 4.

IMPORTANT TIP

Properly indent the instructions under *if statement* using tab. It improves the readability of the program.

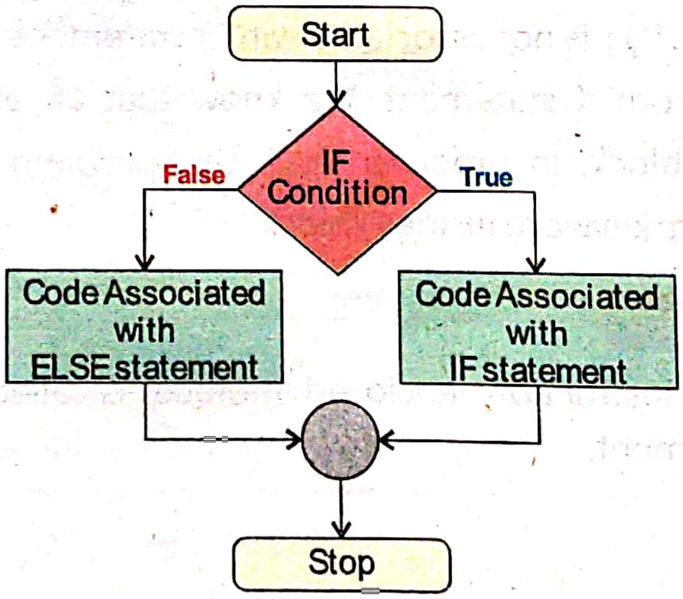
3.2.2 If-else Statement

Till now, we have demonstrated how to execute a set of instructions if a particular condition is *true*, but if the condition turns out to be *false* then we are not doing anything. What if we want to execute one set of instructions if a particular condition is *true* and another set of instructions if the condition is *false*. In such situations we use *if-else statement*. It executes the set of statements under if statement if the condition is true, otherwise executes the set of statements under else statement.

General structure of the *if-else statement* is as follows:

```
if (condition)
        Associated Code
else
        Associated Code
```

Associated code of *if statement* is executed if the condition is *true*, otherwise the code associated with *else* statement is executed. Following flow chart shows the structure of *if-else statement*.


Start
IF Condition
False
True
Code Associated with ELSE statement
Code Associated with IF statement
Stop

Important Note

An *if statement* may not have an associated *else* statement, but an *else* statement must have an *if statement* to which it is associated.

Before *else* keyword, if there are multiple statements under *if*, then they must be enclosed inside the { } block, otherwise compiler issues an error. In order to understand this concept, let's look at the following example.

</> EXAMPLE CODE 3.3

ERROR else without an associated if

```
#include<stdio.h>
void main()
{
        int a = 15;

        if (a % 2 == 0)
                printf("The variable a contains an even value.");
                printf("\nYou are doing a great job.");
        else
                printf("The variable a contains an odd value.");
}
```

The above code cannot be compiled, because as discussed earlier, without a { } block only one statement is associated with *if* statement. In this case, the 1st statement i.e. `printf("The variable a contains an even value.");` is associated with *if* statement, but the 2nd statement i.e. `printf("\nYou are doing a great job.");` is not associated with *if* statement. So, the *else* part is also disconnected from if statement. We know that an else block must be associated to an *if* block. In order to solve this problem, we can put both statements before *else* keyword inside { } block.

DID YOU KNOW?

A set of multiple instructions enclosed in braces is called a block or a compound statement.

Following code demonstrates this.

</> EXAMPLE CODE 3.4

```
#include<stdio.h>
void main()
{
        int a = 15;
        if (a % 2 == 0)
        {
                printf("The variable a contains an even value.");
                printf("\nYou are doing a great job.");
        }
        else
                printf("The variable a contains an odd value.");
}
```

Output:

```
The variable a contains an odd value.
```

Use of if-else statement

The following program takes a character as input and displays "DIGIT" if the character entered by user is a digit between 0 to 9, otherwise displays "NOT DIGIT".

</> EXAMPLE CODE 3.5

```
#include <stdio.h>
void main()
{
        char input;
        printf ("Please enter a character: ");
        scanf ("%c", &input);
        if (input >= '0' && input <= '9')
                printf ("DIGIT\n");
        else
                printf ("NOT DIGIT\n");
}
```

Output: Continued

If user enters a digit, e.g. 5, then "DIGIT" is displayed on the screen.

```
Please enter a character: 5
DIGIT
```

If user enters another character, e.g. *k*, then "NOT DIGIT" is displayed as the condition turns *false*.

```
Please enter a character: k
NOT DIGIT
```

ACTIVITY 3.3

Write a program that takes the value of body temperature of a person as an input and displays "You have fever." if body temperature is more than 98.6 otherwise displays "You don't have fever."

IMPORTANT TIP

If there are more than one instructions under *if statement* or *else statement*, enclose them in the form of a block. Otherwise, the compiler considers only one instruction under it and further instructions are considered independent.

Important Note

C language also provides an *if-else-if* statement that has the following structure.

```
if (condition 1)
        Code to execute if condition 1 is true;
else if (condition 2)
        Code to execute if condition 1 is false but condition 2
is true;
................................
else if (condition N)
        Code to execute if all previous conditions are false but
condition N is true;
else
        Code to execute if all the conditions are false;
```

PROGRAMMING TIME 3.3

Problem:

Write a program that takes percentage marks of student as input and displays his grade. Following table shows grades distribution criteria.

Percentage	Grade
80% and above	A
70% - 80%	B
60% - 70%	C
50% - 60%	D
Below 50%	F

Program:

```
#include<stdio.h>
void main()
{
        float percentage;
        printf ("Enter the percentage: ");
        scanf ("%f", &percentage);
        if (percentage >= 80)
                printf ("A\n");
        else if (percentage >= 70)
                printf ("B\n");
        else if (percentage >= 60)
                printf ("C\n");
        else if (percentage >= 50)
                printf ("D\n");
        else
                printf ("F\n");
}
```

3.2.3 Nested Selection Structures

Let's closely observe the general structure of an if-else statement given below:

```
if (condition)
        Associated Code
else
        Associated Code
```

The code associated with an *if statement* or with an *else statement* can be any valid C language set of statements. It means that inside an *if* block or inside an *else* block, we can have other *if statements* or *if-else statements*. It also means that inside those inner *if statements* or *if-else statements* we can have even more *if statements* or *if-else statements* and so on. Conditional statements within conditional statements are called *nested selection structures*.

All the following structures are valid nested selection structures.

```
if (condition1 is true)
        if (condition2 is true)
                Associated code
        else
                Associated code
```

```
if (condition1 is true)
        if (condition2 is true)
                Associated code
else
        if (condition3 is true)
                Associated code
```

```
if (condition1 is true)
        if (condition2 is true)
                Associated code
        else
                Associated code
else
        if (condition3 is true)
                Associated code
```

```
if (condition1 is true)
        if (condition2 is true)
                Associated code
        else
                Associated code
else
        if (condition3 is true)
                Associated code
        else
                Associated code
```

Use of Nested Selection Structures

In order to understand the usage of nested selection structures, let's have a look at the following example problem.

PROGRAMMING TIME 3.4

Problem:

An electricity billing company calculates the electricity bill according to the following formula.

Bill Amount = Number of Units Consumed X Unit Price

There are two types of electricity users i.e. Commercial and Home Users. For home users the unit price varies according to the following:

Units Consumed	Unit Price
Units < = 200	Rs 12
Units > 200 but Units < = 400	Rs 15
Units > 400	Rs 20

For commercial users, the unit price varies according to the following:

Units Consumed	Unit Price
Units < = 200	Rs 15
Units > 200 but Units < = 400	Rs 20
Units > 400	Rs 24

Write a program that takes the type of consumer and number of units consumed as input. The program then displays the electricity bill of the user.

Program:

```
#include<stdio.h>
void main()
{
        int units, unit_price, bill;
        char user_type;
        printf("Please enter h for home user and c for commercial
        user: ");
        scanf("%c", &user_type);
        printf("Please enter the number of units consumed: ");
        scanf("%d", &units);
```

Continued

```
    if(units <= 200)
            if(user_type == 'h')
                    unit_price = 12;
            else if(user_type == 'c')
                    unit_price = 15;
    else if(units > 200 && units <= 400)
            if(user_type == 'h')
                    unit_price = 15;
            else if(user_type == 'c')
                    unit_price = 20;
    else
            if(user_type == 'h')
                    unit_price = 15;
            else if(user_type == 'c')
                    unit_price = 24;
    bill = units * unit_price;
    printf("Your electricity bill is %d", bill);
}
```

IMPORTANT TIP

In compound statements, it is a common mistake to omit one or two braces while typing. To avoid this error, it is better to type the opening and closing braces first and then type the statements in the block.

ACTIVITY 3.4

The eligibility criteria of a university for its different undergraduate programs is as follows:

BSSE Program: 80% or more marks in Intermediate
BSCS Program: 70% or more marks in Intermediate
BSIT Program: 60% or more marks in Intermediate

Otherwise the university do not enroll a student in any of its programs. Write a program that takes the percentage of Intermediate marks and tells for which programs the student is eligible to apply.

DID YOU KNOW?

C programming language also provides Switch-Case structure to deal with conditions, but Switch-Case structure is applicable only in limited scenarios. The if, if-else structures cover all the possible decision making scenarios.

ACTIVITY 3.5

Write a program that takes two integers as input and asks the user to enter a choice from 1 to 4. The program should perform the operation according to the given table and display the result.

Choice	Operation
1	Addition
2	Subtraction
3	Multiplication
4	Division

3.2.4 Solved Example Problems

PROGRAMMING TIME 3.5

Problem:

Write a program that displays larger one out of the three given numbers.

Program:

```
include <stdio.h>
void main()
{
        int n1, n2, n3;
        printf ("Enter three numbers");
        scanf ("%d%d%d", &n1, &n2, &n3);
        if (n1 > n2 && n1 > n3)
                printf ("The largest number is %d", n1);
        else if (n2 > n3 && n2 > n1)
                printf ("The largest number is %d", n2);
        else
                printf ("The largest number is %d", n3);
}
```

PROGRAMMING TIME 3.6

Problem:
Write a program that calculates the volume of cube, cylinder or sphere, according to the choice of user.

Program:

```
#include<stdio.h>
void main()
{
        int choice;
        float volume;
        printf ("Find Volume\n");
        printf ("1.Cube\n2.Cylinder\n3.Sphere\nEnter your choice:
        ");
        scanf ("%d", &choice);
        if (choice == 1)
        {
                float length;
                printf ("Enter Length: ");
                scanf ("%f", &length);
                volume = length * length * length;
                printf ("Volume is %f", volume);
        }
        else if (choice == 2)
        {
                float length1, radius1;
                printf ("Enter Length: ");
                scanf ("%f", &length1);
                printf ("Enter Radius: ");
                scanf ("%f", &radius1);
                volume = 3.142 * radius1 * radius1 * length1;
                printf ("Volume is %f", volume);
        }
```

Continued

```
    else if (choice == 3)
    {
        float radius;
        printf ("Enter Radius: ");
        scanf ("%f", &radius);
        volume = 3.142 * radius * radius * radius;
        printf ("Volume is %f", volume);
    }
    else
        printf ("Invalid Choice");
}
```

ACTIVITY 3.6

Write a program that finds and displays area of a triangle, parallelogram, rhombus or trapezium according to the choice of user.

SUMMARY

- The flow of program execution is controlled through ***control statements.***
- ***Sequential control*** is the default control structure in C language. According to the sequential control, all the statements are executed in the given sequence.
- The statements which help us to decide which statements should be executed next, on the basis of conditions, are called ***selection statements***.
- In ***if statement*** we specify a condition, and associate a code to it. The code gets executed if the specified condition turns out to be true, otherwise the code does not get executed.
- A ***condition*** could be any valid expression including arithmetic expressions, relational expressions, logical expressions, or a combination of these.
- The ***associated code*** in if statement is any valid C language set of statements.
- ***If-else statement*** executes the set of statements under *if statement* if the condition is *true* and executes the set of statements under *else* otherwise.
- An *if statement* may not have an associated *else* statement, but an *else* statement must have an *if statement* to which it is associated.
- Selection statements within selection statements are called ***nested selection structures.***

Exercise

Q1 Multiple Choice Questions

1) Conditional logic helps in _____________.
   a) decisions b) iterations c) traversing d) all
2) ________________ statements describe the sequence in which statements of the program should be executed.
   a) Loop b) Conditional c) Control d) All
3) In *if statement*, what happens if *condition* is *false*?
   a) Program crashes
   b) Index out of bound error
   c) Further code executes
   d) Compiler asks to change condition
4)
```
int a = 5;
if (a < 10)
    a++;
else
    if (a > 4)
        a--;
```
   Which of the following statements will execute?
   a) a++; b) a--; c) both (a) and (b) d) None
5) Which of the following is the condition to check *a* is a factor of *c*?
   a) *a % c == 0* b) *c % a == 0* c) a*c==0 d) a+c==0
6) A *condition* can be any ________________ expression.
   a) *arithematic* b) *relational*
   c) logical d) arithematic, relational or logical
7) An *if* statement inside another *if* statement is called ________ structure.
   a) *nested* b) *boxed* c) repeated d) decomposed
8) A set of multiple instructions enclosed in braces is called a ___________.
   a) box b) *list* c) block d) job

Q2 Define the following.

1) Control Statements 2) Selection Statements 3) Sequential Control
4) Condition 5) Nested Selection Structures

Q3 Briefly answer the following questions.

1) Why do we need selection statements?
2) Differentiate between sequential and selection statements.
3) Differentiate between *if statement* and *if else* statement with an example.
4) What is the use of nested selection structures?
5) Write the structure of *if statement* with brief description.

Q4 Identify errors in the following code segments. Assume that variables have already been declared.

a)
```
if (x ≥ 10)
        printf (“Good”);
```

b)
```
if (a < b && b < c);
        sum = a + b + c;
else
        multiply = a * b * c;
```

c)
```
if (a < 7 < b)
        printf (“7”);
```

d)
```
if (a == b &| x == y)
        flag = true;
else
        flag = false;
```

e)
```
if (sum == 60 || product == 175)
        printf (“Accepted %c), sum);
else
        if (sum >= 45 || product > 100)
                printf (“Considered %d” + sum);
        else
                printf (“Rejected”);
```

Q5 Write down output of the following code segments.

a)
```
int a = 7, b = 10;
a = a + b;
if ( a > 20 && b < 20 )
        b = a + b;
printf ( " a = %d, b = %d", a, b);
```

b)
```
int x = 45;
if (x + 20 * 7 == 455)
        printf ("Look's Good");
else
        printf ("Hope for the Best");
```

c)
```
char c1 = 'Y', c2 = 'N';
int n1 = 5, n2 = 9;
n1 = n1 + 1;
c1 = c2;
if (n1 == n2 && c1 == c2)
        printf ("%d = %d and %c = %c", n1, n2, c1, c1);
else
        if (n1 < n2 && c1 == c2)
                printf ("%d < %d and %c = %c", n1, n2, c1, c2);
        else
                printf ("Better Luck Next Time!");
```

d)
```
int a = 34, b = 32, c = 7, d = 15;
 a = b + c + d;
 if ( a < 100 )
         a = a * 2;
 b = b * c;
 c = c + d;
 if ( a > b && c == d )
 {
         c = d;
         b = c;
         a = b;
 }
 else
         if (a > b && c > d || b >= d + c)
         {
                 d = c * c;
                 a = b * b;
         }
printf (“a=%d, b=%d, c=%d, d=%d”, a, b, c, d);
```

e)
```
int x = 5, y = 7, z = 9;
if ( x % 2 == 0)
        x++;
else
        x = y + z;
printf (“ x = %d\n”, x);
if ( x % 2 == 1 && y % 2 == 1 && z % 2 == 1)
        printf (“All are Odd”);
if ( x > y || x < z )
{
        if ( x > y )
                y++;
        else
                if ( x < z )
                        x++;
}
printf (“x = %d, y = %d, z = %d”, x, y, z);
```

Programming Exercises

Exercise 1

Write a program that takes two integers as input and tells whether first one is a factor of the second one?

Exercise 2

Write a program that takes a number as input and displays "YES" if the input number is a multiple of 3, and has 5 in unit's place e.g. 15, 75.

Exercise 3

Following is the list of discounts available in "Grocery Mart".

Total Bill	Discount
1000	10%
2500	20%
5000	35%
10000	50%

Write a program that takes *total bill* as input and tells how much discount the user has got and what is the discounted price.

Exercise 4

Write a program that takes as input, the *original price* and *sale price* of a product and tells whether the product is sold on *profit* or *loss*. The program should also tell the profit/loss percentage.

Exercise 5

Write a program that takes as input, the lengths of 3 sides of a triangle and tells whether it is a right angle triangle or not. For a right angled triangle,

$$\text{hypotenuse}^2 = \text{base}^2 + \text{height}^2.$$

Exercise 6

Following is the eligibility criteria for admission in an IT University.

- At least 60% marks in Matric.
- At least 65% marks in Intermediate (Pre-Engineering or ICS).
- At least 65% marks in entrance test.

Write a program that takes as input, the obtained and total marks of Matric, Intermediate and Entrance Test. The program should tell whether the student is eligible or not.

Exercise 7

Write a program that calculates the bonus an employee can get on the following basis:

Salary	Experience with Company	Bonus Tasks	Bonus
10000	2 Years	5	1500
10000	3 Years	10	2500
25000	3 Years	4	2000
75000	4 Years	7	3500
100000	5 Years	10	5000

The program should take as input, the salary, experience and number of bonus tasks of the employee. The program should display the bonus on the screen.