Problem solving skill develops with time. Once you start solving smaller and simpler problem, you become more aware about the steps required for writing solutions of the problems. In previous unit of this block, you learned about algorithms and flowcharts. An algorithm can not run on a computer, you have to write the algorithm in a language that a computer can understand. The languages that a computer understands are known as computer programming language.

The C is one of the very popular programming language. The C was written by Kerningham and Richie in the 1970s. C is used for developing computer applications in the area of system software, utility software such as device driver, network programming, database systems, and many other areas of applications software. You may take this unit as beginning of learning a programming language. It is easy to learn other computer programming languages after learning C.

2.1 OBJECTIVES

After going through this unit you should be:

• explain the need of C programming language;
• explain C program structure;
• explain basic components of C program;
• write simple C program;
• explain sequential, conditional, iterative statements, and
• use array variables in C programs.
2.2 EXECUTION OF A C PROGRAM

C is a high level programming language. It is having a rich set of library. A library is a collection of specific programs/functions used in different programs. When we write our programs in C, there are libraries of functions to help us so that we do not have to write the same code over and over. Using the library functions will also make it easier to write and learn to program. You can take a function as a module often consisting of a set of statements to perform specific task. Process of execution of a C program is given in Figure 1.

![Figure 1: Process of Execution of a C Program](image)

2.3 STRUCTURE OF A C PROGRAM

Before we discuss about structure of a C program let us see a simple C program which does not perform any operation and only prints “Hello World” message:

First C Programme

1. /* This is My First C Program */
2. #include <stdio.h>
3. /* The First C Program to print Hello World */
4. int main(
5. {
6.   printf("Hello World\n");
7. return 0;
8. }

Comments.
The main() function is always the point from where your program starts running.
Print out a message and ‘\n’ means “new line character”.
Return ‘0’ from main function

A blocks of code are marked by { … }

After you compile and run this program it will print out the message: Hello World. This is the first C program we are exposed. Though the program is very simple, but you may learn a few points by going through eight lines of this program. #include< inserts another
file. “.h” is used for calling “header” files. These header file contain methods for performing needed tasks. All header files have interface to libraries and their cod for performing task is kept in other “.c” files.

Every C program contains a function called main. This is the starting point of the program. #include <stdio.h> is used to allow a program to interact with the screen as output on your computer. You will find it at the beginning of almost every C program. The details of need of #include <stdio.h> we will discuss later. The main() function declares the start of the program and the two curly brackets( { and } ) show the start and end of the main function. Curly brackets in C are used to group statements in a block together to mark the beginning and end of a function or in the body of a loop( loop will be discussed later in this unit). This type of grouping is known as block in a program.

printf(“Hello World
”); prints the words on your computer screen. The text to be printed is enclosed in double quotes. The ‘\n’ at the end of the text tells the program to print a new line as part of the output.\n is used for the purpose of more clarity in the output. By putting \n at the end of the message to be printed it is ensured that when ever in a program some thing more will be printed on screen it will be printed in the next line.

One more point you may note here is that C is case sensitive therefore it differentiate between a lower case letter and a upper case letter. For example main is different than Main.

return 0 - is written in this program for completing it. Because in int main( ) int means function main() returns an integer value, so there must be a statement that indicates what this int value is. The statement return 0 ; indicates that main() returns a value of 0 (zero) to the operating system(OS). If this program has completed successfully and terminated execution it will return 0 to OS. Here you do not have to know in detail about this concept. Without getting worried about this concept let us move on to know more about structure of a C program. But you just remember to use the return statement in your program when you write because main function always returns an integer.

In this program there are two functions: main and printf.

On the basis of our fist C program now let us see the basic structure of a C program which generally has following components:

1. program header comment
2. preprocessor directives (if any)
3. int main ( void )
4. {
5. statement(s);
6. return 0 ;
7. }

Now let us see the meaning of each line of this structure:

i) Program Header Comment:

A comment is description used for clarification. It helps a reader of the program understand the purpose or functionality of statements in the program. A C program may have several comments in whole program. It is good to have comment in the beginning of a program which state about the objective of the program for which it is used.

How to write a Comment?

In C all comments must begin with the characters /* and end with the characters */. For example if you write
/* This is my First C Program */
then statement “This is my First C Program” will be treated as comment. In any program
the program header comment always comes first.

ii) Preprocessor Directives:

Preprocessor directives are special statements that is included in the program code as you
have seen in section 2.3.1 of this unit. These directives are not program statements but
special instructions (directives) for the preprocessor and on the basis of these directives
some activities are performed before actual program code compilation. For example
#include <stdio.h> is called preprocessor directive (command). It directs the preprocessor
to include all the functions defined in stdio.h in the program before it compiles the code
written in the program. For more clarity please see the following printf() statement:

printf("Hello World\n"); statement in program given in previous page function printf( )
is defined in stdio.h and has been included in that program.

You remember that a preprocessor directive is written in only a single line of code. As
soon as a newline character ‘\n’ is found, the preprocessor directive is considered to end.
No semicolon (;) is used for the end of a preprocessor directive as in the case of other
statement like: printf("Hello World\n"); or return 0;
In general “a preprocessor is a program which processes the input data and produce
output that is used as input to another program”.

stdio.h – The #include <stdio.h> directive indicates the preprocessor to include a copy of
the standard input/output header file stdio.h in the program code.

iii) int main ( void ): Generally every program has a function called main. This is the
point from where program execution begins. The main () is placed in the source code file
as the first function for the purpose of readability. In your program there must be a
function with this name otherwise c compiler can not successfully compile and link your
program.

Prototype for writing a function is:
return type function_name ( arguments)

A function may take some value for performing operations. The value(s) given to a
function is known as arguments to the function. For example suppose you have written a
function:

int sum( int a, int b)

for finding sum of two integers. This function return an integer and take two integers as
argument it means you can pass two values of integers to the function sum. In side
parenthesis ( ) after main, void is written, it means nothing is there inside ( ).

iv) The Function Body of main( ): The main function starts with a left brace (curly
brace)- { - begins the body of every function. A corresponding right brace - } - ends
the function body. In side { and } are the statements to be performing the operation. For
example if you are writing a program to find the area of a circle in C, then operations need
to be performed for this purpose will be written inside { and }.
v) **Statement(s):** These are the main portion of any program. Statements written in a program decides about the activity to be performed such as input operations, output operations, logical operations etc.

vi) **return 0; :** Because function main() returns an integer value, there must be a statement that indicates what this value is. The last statement is return 0; . It ends with a semicolon. Also remember that all statements in C end with a semicolon.

After learning about structure of a C program, now let us see the basic components of C programming. These components are used for writing C programs for solving problems.

Next section is going to cover almost all the basic components of C programming. This section will give you idea for using basic components in the program you will write.

---

**2.4 BASIC COMPONENTS OF PROGRAMMING**

The basic component of C includes token, keywords, variables, datatypes, operators etc. Let us see some details of these basic components of C programming language.

**2.4.1 Tokens and Keywords**

**Token**

In C language tokens are the smallest element. A token may be a single character or a sequence of characters to form a single item for c program. Tokens can be numeric constants, character constants, keywords, names (lable or identifiers), punctuation or operators used for different kind of operations including arithmetic and numeric operations.

**Keywords**

In any programming language, some words are reserved for some special use. These words are not allowed to be used by programmer for their own purpose. These words are called keywords and sometimes called reserved words also. C language like any other programming language has defined some keywords. Every keyword have a fixed meaning to the compiler and can not change the meaning. For example int is used for representing integer. C keywords are given below:

<table>
<thead>
<tr>
<th>auto</th>
<th>default</th>
<th>float</th>
<th>register</th>
<th>struct</th>
</tr>
</thead>
<tbody>
<tr>
<td>break</td>
<td>do</td>
<td>for</td>
<td>return</td>
<td>switch</td>
</tr>
<tr>
<td>case</td>
<td>double</td>
<td>goto</td>
<td>short</td>
<td>typedef</td>
</tr>
<tr>
<td>char</td>
<td>else</td>
<td>if</td>
<td>signed</td>
<td>union</td>
</tr>
<tr>
<td>const</td>
<td>enum</td>
<td>int</td>
<td>sizeof</td>
<td>unsigned</td>
</tr>
<tr>
<td>continue</td>
<td>extern</td>
<td>volatile</td>
<td>long</td>
<td>while</td>
</tr>
<tr>
<td>static</td>
<td>void</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Now let us learn about data types and variables in C.

**2.4.2 Data Type and Variables in C**

Data Types in C
Most of the time in calculation, data are used for example in finding the sum of two numbers first of all these two numbers need to be stored in computer memory then result obtained after adding these two numbers is again stored in the memory. Now a question arises what size of memory should be used for storing these numbers? To answer this question one need to know what the type of number is. A number might be an integer (whole number such as 20) or real number (floating point number such as 10.25). An integer needs 2 bytes for storage. One byte equals eight bits. There are three basic data type which are:

1) Character

2) Integer

3) Real numbers

In addition to these three two other basic data types *enum* and *void* which are out of the scope of discussion here. For real numbers *float* and *double* types are used. The size of three basic data type in C is given in table given below.

### Table 1: Data Types in C

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Size (in Bytes)</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Char</td>
<td>1</td>
<td>Character</td>
</tr>
<tr>
<td>Int</td>
<td>2</td>
<td>Integer</td>
</tr>
<tr>
<td>Float</td>
<td>4</td>
<td>Floating point</td>
</tr>
<tr>
<td>double</td>
<td>8</td>
<td>Long floating point</td>
</tr>
</tbody>
</table>

**Variables**

Once you have data to store, you need to associate with a specific memory location. These memory locations are given some names and these names are called variable names. The value stored in these locations can change or vary throughout the program’s lifetime. These data are referred with variable name. Each Variable has a specific type, which is known as variable data type. This data type tells the computer how much memory is taken by the variable to store the data.

Name of a variable must start with letters and contain letters, digits, and underscores. For example x, y, z, sum, number_of_points, mars12 are valid variable names. A variable must be declared in a C program before being used. Variable declaration tells the compiler about variable name and their type.

For example: `int i;` means a variable named i of integer type is declared. Now i is available as a variable to store integer value. In C you can declare more than one variable in single statement (line). A declaration begins with the type of the variable(s), followed by the name of one or more variables.

`float i,j,k,sum;` means, four variables named i,j,k, and sum of float type are declared. When you declare your variable it is better to have some meaningful name of a variable.
While you declare variables you may give explanatory comment for more clarification. You can initialize a variable at the time of declaration. This is done by adding an equal’s sign and the required value after the declaration. For example:

```c
int high = 25; /* Maximum Length */
int low = 10; /* Minimum Length */
```

two variables high and low are declared and initialized. Remember that, you cannot use any of C's keywords like main, while, switch etc as variable names.

**Names or Identifiers**

Sometimes called identifiers or labels. These are the names of functions, variables etc. It can be of anything length, but it should not be too long or too short. Names are case sensitive xyz is different from XYZ. Any name must begin with a letter and the rest can be letters, digits, and underscores.

Different operators perform different kind of operations such as arithmetical operations, logical operations etc. In next section we will learn about different operators in C.

### 2.4.3 Operators and Punctuations in C

**Operators**

An operator is a kind of function which is applied to values to give a result. For example for finding sum of two numbers ‘Addition’ operator is required. C is having a wide range of useful operators. You are familiar with operators such as +, -, /, which are used for performing addition, subtraction, and division.

The common type of operators in C are:

1. **Arithmetic Operator**
2. **Assignment Operator**
3. **Comparison Operator**
4. **Logical Operator**
5. **Binary Operator**

Arithmetic operators are the most commonly used operator. Comparison operators are used for comparison of values, logical operators are used for deciding combination of logical states, assignment operator is used assigning value to a variable, and binary operators are used for manipulation of individual binary digits of data. Here we will not cover binary. Now let us see in detail about each operator type mentioned above.

#### i) Arithmetic Operators

Commonly used arithmetic operators in C are

+ Addition

**Example**

4+6 will give 10

- Subtraction

**Example**

10 - 6 will give 4
Problem Solving Techniques

* Multiplication

**Example**
2*5 will give 10

/ Division

**Example**
8/2 will give 4

% Modulos (give remainder from integer division)

**Example**
5%2 will give 1

Assignment and Expressions

If you combine operators and values expressions are formed. The values produced by these expressions can be stored in variables. For example: \(2 + 4 \times 3\) is a expression and after evaluating this expression you will get 14.

A simple expression is an assignment statement. An expression is evaluated and the result is saved in a variable. For example expression: \(a = (i \times j) + k\), the result will be saved after evaluating \((i\times)k\) in variable \(a\). The result will be saved after evaluation.

In any expression, operators *, / and % will be performed before + or -. If you want to have different order of evaluation the you have to use brackets. Any expression in brackets will evaluate first. For example, if you write expression: \(2 + 4 \times 3\) as: \((2+4)\times3\) then you will get 18.

One important point to note about / operator is that when division is performed between two integers, the result will be an integer, and remainder is discarded. For example if expression: \(15/4\) is evaluated it will give 3 as result. You can use modulus operator(%) between integers only. Never divide a number by zero in your program, this will cause an error, and usually due to this your program will crash (will terminate abnormally).

Some examples of operators, according to precedence.

\[
\begin{align*}
1 + 5 \times 2 & \rightarrow 1 + 10 & \rightarrow 11 \\
(1 + 4) \times 2 & \rightarrow 5 \times 2 & \rightarrow 10
\end{align*}
\]

You can use symbols for storing value. In this case, symbols are evaluated to their values before being combined

```cpp
int x=2;
int y=3;
y + x*x \rightarrow y + 2 \times 2 \rightarrow y + 4 \rightarrow 3 + 4 \rightarrow 7
```

ii) Assignment Operator

Assignment operator (=) is used for assigning value to a variable. Now, let us take a problem in which speed of a car is given to 50 km/h, distance to be covered is 200km and you have to find the time it will take to cover this distance. Here, you can use assignment operator for assigning values:

```cpp
int speed, distance, time; /*are three variables */
speed = 50;
distance = 200;
time = distance/time; /* time will have value 4 , that is 200/50 */
```
Problem: Write a program in C to find a+b, a-b, a*b, a/b, and a%b. The value of a is 10 and the value of b is 5.

### C Program for showing use of arithmetic operators

```c
/* C program for arithmetic operations */
#include <stdio.h>
int main( void )
{
    int a= 10; /* a is initialized with value 10 */
    int b = 5 ; /* a is initialized with value 10 */
    printf("a+b = %d\n",a+b) ;
    printf("a-b = %d\n",a-b) ;
    printf("a*b = %d\n",a*b) ;
    printf("a/b = %d\n",a/b) ;
    printf("a%b = %d\n",a%b) ;
    return 0 ;
}
```

When you will run this program following output will come:

```
a+b = 15
a-b = 5
a*b = 50
a/b = 2
a%b = 0
```

You must have noticed that printf function is having following format:

```c
printf("message = %d\n", value to print) ;
```

Here message is a string of character, %d for indicating that value for printing is of integer type. Similarly for character value printing %c and for float value printing %f are used.

In the above program fixed value to a and b are assigned but if you need to take value as input from your computer for that in C `scanf()` function is used. As `printf()` used for printing output on your computer screen, `scanf()` is used for reading input from your computer screen. The syntax for using `scanf()` is:

```c
scanf("%d",&variable);
```

Here, %d indicate that the variable in for which data to be read is of integer and & sign is added (prefixed) to the variable name.

The above program can be modified as follows for giving input from screen instead of assigning fixed value.

```c
/* C program for arithmetic operations */
#include <stdio.h>
int main( void )
{
    int a, b /* a and b are declared */
    printf("Give vale of a: ");
    scanf("%d",&a); 
    printf("Give vale of b: ");
    scanf("%d",&b); 
    printf("a+b = %d\n",a+b) ;
    printf("a-b = %d\n",a-b) ;
    printf("a*b = %d\n",a*b) ;
```
Problem Solving Techniques

```c
#include <stdio.h>

int main(void)
{
    int a,c;
    int b=10;
    a=5;
    c = a+b; /* C is assigned 10+5 */
    printf("Sum: %d + %d -> %d\n",a,b,c);
    a++;
    b--;
    printf("Now a=%d \n",a);
    printf("Now  b=%d\n",b);
    c = a+b; /* C is assigned 9+4 */
    printf("Now Sum is: %d + %d -> %d\n",a,b,c);
    return 0;
}
```

When you will run this program following output will come:
Give vale of a: 20
Give vale of b: 4
a+b = 24
a-b = 16
a*b = 80
a/b = 5
a%b = 0

**Special Operators ++ and --**

In C some operators allow abbreviation of certain types of arithmetic assignment statements. These operators are ++ and --.
- `i++` or `++i` is equivalent to `i = i+1;`
- `i--` or `--i` is equivalent to `i = i-1;`

Now let us see an example program which shows how to use `++` and `--`:

C program to show the use of `++` and `--`:

```c
int main(void)
{
    int a,c;
    int b=10;
    a=5;
    c = a+b; /* C is assigned 10+5 */
    printf("Sum: %d + %d -> %d\n",a,b,c);
    a++;
    b--;
    printf("Now a=%d \n",a);
    printf("Now  b=%d\n",b);
    c = a+b; /* C is assigned 9+4 */
    printf("Now Sum is: %d + %d -> %d\n",a,b,c);
    return 0;
}
```

**iii) Comparison Operator**

Many time in programming requirement arise to compare two values. C is not having special data type to represent logical values (0 or 1). In C by using char and int, with a value of 0 means false and any other value represent true. Comparison operators are used to compare values. When two numeric values are compared using comparison operators, they compare values and produce a logical result.

There are six comparison operators:

<table>
<thead>
<tr>
<th>Operator</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>==</td>
<td>Equal to</td>
</tr>
<tr>
<td>&gt;</td>
<td>Greater than</td>
</tr>
<tr>
<td>&lt;</td>
<td>Less than</td>
</tr>
<tr>
<td>&gt;=</td>
<td>Greater than equal to</td>
</tr>
</tbody>
</table>
Problems Solving using
C Programming Language

These comparison operators are used in expression like:

I == J

Marks > 10

Marks != 100

Example of use of comparison operator
int x = 5;
(x < 10) → (4 < 10) → true
(x = 5) → (5 = 5) → false

More detailed discussion about using comparison operators will be taken up when we will discuss about control statements such as if statement or a for or a while statement in subsequent section of this unit.

iv) Logical Operators

AND, OR and NOT are the commonly used logical operators in C.

Operator | Meaning
--- | ---
|| OR
&& | AND
! | NOT

a || b will be true if either of a or b is having value true. It will be false only if both a and b are having false value, a && b will be true if both a and b are having true value, otherwise false. If a is true then !a will be false.

Logical operators are frequently used to combine comparison operators, for example if a person is eligible for a job if his age is between 18 and 30 years then it can be written as following expression:

age >= 18 && age <= 30

A person will be eligible only if this expression is evaluated to true. Let us check this expression for a person having age 25.

25 >= 18 && 25 <= 30 → true && true → true

We can come to conclusion that false && anything is always false, true || anything is always true.

Punctuation

In C programming language Semicolons (;), colons (:), commas (,), apostrophes (’), quotation marks (“”), braces ([ ]), brackets ({ }), and parentheses ( () ) ; , ‘ “ [ ] { } ( ) are punctuation symbol.

2.4.4 Constants

A constant can be seen as a type of variable which do not change its value in program. There are many types of constants in c such as numeric constant, character constant, string constant etc.
Character constants are typically just the character enclosed in single quotes; ‘x’, ‘y’, ‘z’. Some characters can’t be represented in single quotes. For them a two character sequence is used. For example:

new line - ‘n’
single quote – ‘’
back slash – ‘\\’

Numeric constants are an continuous sequence of digits (and may contain a period). A numeric constant never contains a comma.

Examples of numeric constants: 123, 55.45, 100

Check Your Progress 1

1) State True/False for the following statements:

i) A program in C is compiled before running it.
ii) C is a case sensitive language.
iii) For storing an character value variable of integer type is used.
iv) In C register is a valid variable name.
v) In C ++ is an operator.

2) Fill in the Blank:

i) C is a ------ language
ii) For storing integers ------- data type is used.
iii) #include is a -------
iv) The symbol of assignment operator is ----
v) ‘*’ is -------- operator

3) Write a C program which read two integers and find the product of them.

4) Write a C program which read two float numbers and apply arithmetic operators on them.

A program can be seen as a collection of statements in a logical order. Every statement in a program is assigned some specific task to perform. In a program statements may execute in a sequential order or it may execute based on satisfaction of certain condition. Some statements may execute more than once. In the next section we will discuss about various types of statements in C programming.

2.5 STATEMENTS IN C PROGRAMMING

A program consists of statements. These statements may be a variable declaration, arithmetical expression, logical expression, call of a function etc. According to the nature of statements belong to different category. These categories are:
2.5.1 Sequential Statement

Sequential statements are those statements in a program that executes one by one in a given sequence. For example following statements:

1. float a,b;
2. printf("Give the first number:");
3. scanf("%f", &a);
4. printf("Give the second number:");
5. scanf("%f", &b);

Above five statements are part of sequential statements of a C program. These statements will be executed one by one in a C program.

2.5.2 Conditional Statements

Many time while writing program you will need to take some decisions. These decisions are to be taken based on certain conditions. For example suppose you have to find whether a person can vote or not. For this you need to compare the age of the person with the minimum age required to qualify for voting right.

Age >= Age required for voting

You can say that a program can be much more powerful if you control the order in which statements are run. Some time you may need to select between optional sections of a program. Suppose you are required to prepare two lists of candidates as given below:

1. Who will be appearing in examination
2. Who will not be appearing in examination

On the basis of percentage of attendance of student. The condition is that - if a student is having 75 % or more attendance will be appearing in the examination otherwise not. To decide whether a student will be appearing in examination or not, you need to compare his percentage of attendance with 75 (the percentage of attendance required to appear in exam), and then make a decision about what to do next.

In C there are some conditional statements:

- if
- if-else
- if-else if
- switch-case

are there for performing conditional task.  We will discuss about if, if-else, and if-else if, and switch-case.
structure of if statement:

if( condition)
{
 statement(s)
}

Here if condition is true then only statement(s) in { } will be executed. In other words you can say that control will reach in { } only after checking condition and finding it true.

/* if evaluated expression is not 0 */
if (expression)
{
 /* then execute this block */
}
/* Continue other operations in sequence*/

/*Example program to test whether a given number is greater than 10 */
#include <stdio.h>
void main(void)
{
 int num = 20; /* initialize num by 20*/
 if( num > 10)
 {
  printf(“The number is greater than 10\n”);
 }
 printf(“The number is:%d”,num);
 return 0;
}

The output of this program will be:
The number is greater than 10
The number is:20

Now let us see if –else statement. This statement is used if you have to choose one from two given set of statement(s).The structure of if-else is

if(condition)
{
 /* if condition is true this block will execute*/
if-else also work similar to if statement. The only difference between if and if-else is that in if after finding condition true a set of statement(s) is executed and in if-else for true value of condition one set of statement(s) is executed and for false value of condition another set of statement(s) is executed. Below is the flowchart for if-else statement.

![Flowchart for if-else statement](image)

You can say that to decide between two courses of action if-else is used. For example to decide whether a student has passed an exam with a pass mark of 45 you can write c code:

```c
/*
  if (Marks >= 45)
  {
    printf("The Student is Pass\n");
  } else
  {
    printf("The Student is Fail\n");
  }
*/
```

Another conditional statement is if-elseif-else. This statement is also known as ladder if statement. Flowchart for if-else if-else is given below.
This conditional statement is used in situations where you wish to make a multi-way decision based on certain conditions. The way of doing such type of testing is by using several `else-if` statements. This works by cascading more than one comparison. As soon as a comparison gives a true result, the following statement(s) or block is executed, and after that no further comparisons are performed. You have already seen one example of comparison where you found whether a student is pass or not. Suppose you have find the grade of a student on the basis of marks of the student then you need to do many comparison. In the following example we will find the grades depending on the exam result.

```c
if (Marks >= 75) {
    printf("Passed and Grade is A\n");
}
else if (Marks >= 60)
```

![Flowchart for if-else if](image-url)
Problems Solving using C Programming Language

In this example, all comparisons are testing a single variable called Marks. There may be cases where you have to do multiple comparisons on more than one variable. While doing programming the same pattern can be used with more or fewer else if's. Note that finally only else may be left out as last block.

Now, let us see one more very useful statement used for multi way decision making. This statement is switch-case. This statement work similar to if-else-if statement.

The switch-case Statement

This statement is for the multi way decision. It is well structured, but it is having one limitation that it can be used only in certain cases where only one variable is tested for condition checking. All branches of the decision must depend on the value of that variable only. Also the variable used in condition checking must be an integral type (int, long, short or char).

In switch-case statement each possible value of the variable can control a single branch and a final, catch all leftover cases. You should use default branch (final) for catching all unspecified cases.

Now, let us see the example below. This will clarify you the concept of switch-case. This example takes an objective which converts an integer into a text description you can say at this program is working to estimate the number given as input.

```c
int number;
/* Estimate a number as none, one, two, more, many */
switch(number)
{/n    case 0 :
        printf("The number is : None\n");
        break;
    case 1 :
        printf("The number is :One\n");
        break;
    case 2 :
        printf("The number is : Two\n");
        break;
    case 3 :
    case 4 :
    case 5 :
        printf("The number is : More than Two\n");
        break;
    default :
        break;
    }
```
In the above example, each case is listed with some corresponding action. For example, case 0 is associated with the following action:

```c
printf("The number is : None\n");
break;
```

Here, the `break statement` is a new thing for you to notice. The `break` statement prevents any further statements from being executed by leaving the switch statement (block), in other words, you can say that break statement is used for transferring control to the next block from the current block. The break statement is generally used to exit from a loop or a switch, control passing to the first statement beyond the loop or a switch block. You can see that the case 3 and case 4 have no following break, those continue on allowing the same action for several values (3, 4, 5) of number.

The similarity between if and switch statement is that both allow a programmer to make a selection from a number of possible actions.

### 2.5.3 Iterative Statements

You may need to write the same code more than one in your program. For example if you have to print numbers from 1 to 10 and one number in a line, then in general you will need 10 `printf()` statements in your program— one `printf` for printing one number.

There is a concept of loop in C programming. Conceptually a loop is like a circle. If you walk on the border of a circle after completing a round you will reach the same point. There may be a set of statements in a program which are executed several times. Such statements are called iterative statements. This concept helps in writing such programs where the same code is repeated several times. You can only write once that code in a loop and run the loop as many times as required. In the next section we will discuss about `for` and `while` loop of C.

#### Loops in C

In C you have a choice of three types of loop: `while`, `do-while` and `for`. Before you use any looping statement it is necessary to decide the condition which take control of the program out of the loop. Once, control of program enters into a loop it repeats the statement(s) inside the loop. The control comes out of the loop after test condition becomes false. You can say that a loop runs till a test condition is true and stops when test condition becomes false. Here, you may ask that how a test condition becomes false from a true value? While a loop runs, value of variables involve in test condition also may change and that leads to change the test condition from true to false.

#### The for Loop

The `for` loop is frequently used. It is very flexible in terms of condition check and number of iterations. You can say that the `for` loop works well where the number of iterations of the loop is known before the loop is entered. Structure of `for` loop is:

```c
for( initial condition; test condition; increment or decrement)
{
  statement(s);
}
```
The beginning of the for loop consists of three parts and they are separated by semicolons. The first part is run before the loop is entered and is usually the initialization of the loop variable. The second is a test condition. The for loop will exit when this condition returns false. The third and final statement runs every time the loop body is completed. This is usually an increment or decrement of the loop counter and on the basis of this counter value test condition retain true or become false.

Now let us see a program which uses for loop for printing numbers 1 to 10.

```c
#include <stdio.h>

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

When you will run this program following output will come:
Number: 1
Number: 2
Number: 3
Number: 4
Number: 5
Number: 6
Number: 7
Number: 8
Number: 8
Number: 10

You are getting this output because:
printf("Number: %d",i);
statement in this program is inside a for loop which is running 10 times.
You may write a program using for loop for finding the sum of a series: 1+2+3+4+……………..+n. This program will be like:

```
//Program for finding sum of series: 1+2+3+4+……………..+n
#include <stdio.h>

void main(void)
{
    int i,n,sum=0;
    printf("Give the Number of n: ");
    scanf("%d",&n);
    for (i=1; i<= n; i++)
    {
        sum = sum+i;
    }
    printf("The sum of the series is: %d",sum);
    return 0;
}
```

If you run this program and give 5 as value of n then it will print:
The sum of the series is:15

The while Loop
Similar to **for loop** in the while loop also you can keep repeating an action until an associated test condition returns false. The while loop is useful where the programmer (the person writes/design the program) does not know in advance how many times the loop will be repeated (iterated). For example in a situation where a loop terminates after specific input from the user which make test condition false. For example in while loop if a test condition is asking for the acceptance (Yes or No) from the users to decide about continuity of the loop, then for terminating (coming out of the loop) the loop user has to give value No. The syntax of while loop is:

```c
while (condition)  
{  
  statements(s)  
}
```

By seeing the flowchart of **while-loop** you will get clearer picture of working of this loop. Below is the **Flowchart for while-loop**:

![while loop flowchart](image)

Now, you may write a program using while loop for finding the sum of a series: 1+2+3+4+………..+n. This program will be like:

```c
#include <stdio.h>  
void main(void)  
{  
  int i=1;  
  int n,sum=0;  
  printf("Give the Number of n:");  
  scanf("%d",&n);  
  while ( i<= n)  
  {  
    sum = sum+i;  
    i=i+1;  
  }  
  printf("The sum of the series is: %d",sum);  
  return 0;  
}  
```

If you run this program and give 7 as value of n then it will print:

The sum of the series is: 28
The do-while Loop

The *do-while loops* is similar, but the test occurs after the loop body is executed. This ensures that the loop body is run at least once. The while loop repeats a statement until the test at the top proves false. The do while loop is frequently used where data is to be read. In such process read the data, then test and verifies the data, and loops back to read again if it was undesirable.

The syntax of do-while loop is:

```c
do{
    statement(s)
} while(condition)
```

By seeing the flowchart of *do-while loop* you will get clearer picture of working of this loop. Below is the Flow Chart for do-while-loop:

![Flowchart for do-while loop](image)

Syntax of *do-while loop:*  
```c
do{
    statement(s)
} while(condition)
```

For example the statements:

```c
printf("Enter 1 for exit from loop");
scanf("%d", &input_value);
```

will repeatedly run inside the do-while loop given below until you give/read 1 as input_value.

```c
do{
    printf("Enter 1 for exit from loop");
    scanf("%d", &input_value);
} while(input_value != 1)
```
Problem Solving Techniques

Now let us see a program which uses do while loop for printing numbers 1 to 10.
```
#include <stdio.h>
void main(void)
{
    int i=1;
    do
    {
        printf("Number is: %d",i);
        i=i+1;
    }
    while( i<=10)
    return 0;
}
```

When you will run this program following output will come:
Number is: 1
Number is: 2
Number is: 3
Number is: 4
Number is: 5
Number is: 6
Number is: 7
Number is: 8
Number is: 8
Number is: 10

Sometimes a need arises to have many variables of the same type. For example to store marks of different subjects of a student one need to have as many variable as the number of subjects. In such situations array variable is very useful. In next section you will learn arrays.

### 2.6 ARRAYS IN C

An array is a collection of variables of the same type. In a collection variable of the same type, an individual variable is known as an array element of the array. Total number of variables in the collection is known as size of the array. Each element in the array is identified by an integer index. In C the index begins at zero and is always written inside square brackets. For example if we have an array of type integer for storing marks of all the papers in a semester of students in a class. In C this can be declared as:

```
int Marks[size];
```
here size is the total number of subject in that particular semester.

Here Marks is a single dimensional array. If you declare Marks like this
```
int Marks[10];
```

Then you can store marks of 10 subjects. You can identify marks of each subject separately as Marks[0], Marks[1], Marks[2], Marks[3], ..........,Marks[9].

Arrays can have more dimensions, in which case they might be declared as
```
int results_2d[20][5];
```
int results_3d[20][5][3];
Here each of the index has its own set of square brackets.

Now let us see an example to find average percentage marks of a student in a semester. Here we assume that student is having 6 subjects is a semester and each subject is having maximum marks 100.

/* Finding average marks using array variable*/
#include <stdio.h>
int main(void)
{
    int Marks[6], sum=0;
    for (int i=0; i<6; i++)
    {
        printf("Give the Marks of Subject %d:",i+1);
        scanf("%d",&Marks[i]);
        sum = sum+Marks[i];
    }
    printf("The average percentage Marks of the Semester is:%d",sum/6);
    return(0);
}

output:
Give the Marks of Subject 1: 50
Give the Marks of Subject 2:70
Give the Marks of Subject 3:75
Give the Marks of Subject 4:80
Give the Marks of Subject 5:40
Give the Marks of Subject 6:85
The average percentage Marks of the Semester is:66

Now in the last section of this unit let us see some areas of applications of C programming language.

2.7 USES OF C PROGRAMMING LANGUAGE

C is extensively used for developing applications in various area of computer applications. Initially, it was used for system development work, for example, in developing the operating system. You may ask why C is used in these areas? It is mainly because it produces code that runs very fast. You can say nearly as fast as code written in assembly language.

Here is some examples of the use of C:
Operating Systems
Language Compilers
Assemblers
Text Editors
Print Spoolers
Network Drivers
Data Bases Program
Language Interpreters
Utility Programs

Check Your Progress 2
1) State True/False for the following statements:
   i) switch-case statement is used for the multi way decisions.
   ii) C programming language can not be used for writing device drivers.
   iii) if statement is conditional statement.
   iv) while loop and do-while loop are same.
   v) An array is a collection of variables of different data type.

2) Write a C program to find largest among three numbers.
   ……………………………………………………………………………………………
   ……………………………………………………………………………………………
   …………………………………………………………………………………………….

3) Write a C program to find sum of series: \(1^2+2^2+3^2+\ldots+10^2\).
   ……………………………………………………………………………………………
   ……………………………………………………………………………………………
   …………………………………………………………………………………………….

4) What is an array? Write a C program using array to find the largest number from a list of 10 given numbers.
   ……………………………………………………………………………………………
   ……………………………………………………………………………………………
   …………………………………………………………………………………………….

2.8 SUMMARY

In this unit, you have learned basics of C programming. This unit explained about structure of C program, need of preprocessors, and main method in a C program. You learned in this unit about basic components such as data type, variables and operators in C. There are three basic types of statements in a C program sequential, conditional and iterative. You have learned how to incorporate these statements in your program. Also this unit covers basics of conditional and iterative constructs of C such as if, else if, switch-case, for loop, and while loop. In the last of this unit you were introduced to the concept of array variable.

2.9 SOLUTIONS/ANSWERS

Check Your Progress 1

1) i) True
   ii) True
   iii) False
   iv) False
   v) True

2) i) programming
   ii) int
   iii) directive
problems solving using c programming language

3) /* program for multiplication */
   #include <stdio.h>
   int main( void )
   {
      int num1, num2, product; /* three variable of integer type declared */
      printf("Enter First Integer Value: ");
      scanf("%d", &num1); /* read num1 */
      printf("Enter Second Integer Value: ");
      scanf("%d", &num2); /* read num2 */
      product = num1 * num2; /* product of num1 and num2 */
      printf("The Product = %d\n", product);
      return 0;
   }

4) /* application of arithmetic operators on floating point numbers */
   #include <stdio.h>
   int main()
   {
      float a,b;
      printf("Give the first number:");
      scanf("%f",&a);
      printf("Give the second number:");
      scanf("%f",&b);
      printf("a+b=%f\n",a+b);
      printf("a-b=%f\n",a-b);
      printf("a*b=%f\n",a*b);
      printf("a/b=%f\n",a/b);
      return 0;
   }

check your progress 2

1) i) true
   ii) false
   iii) true
   iv) false
   v) false

2) /* program to find largest among three numbers */
   #include <stdio.h>
   #include<conio.h>
   main()
   {
      int num1,num2,num3;
      clrscr();
Problem Solving Techniques

printf("Enter three integer values\n");
scanf("%d%d%d",&num1,&num2,&num3);
if ((num1>num2)&&(num1>num3))
{
    printf("%d is greatest\n",num1);
}
else if((num1<num2)&&(num2>num3))
{
    printf("%d is greatest\n",num2);
}
else
{
    printf("%d is greatest\n",num3);
}

3) //Program for finding sum of series: $1^2+2^2+3^2+\ldots+10^2$
#include<stdio.h>
void main(void)
{
    int i=1;
    int n = 10,sum=0;
    printf("Give the Number of n: ");
    scanf("%d",&n);
    while(i<= n)
    {
        sum = sum+i*i;
        i = i+1;
    }
    printf("The sum of the series is: %d",sum);
    return 0;
}

4) An array is structure which holds multiple variables of the same data type. Consider the case where a teacher needs to keep track of a number of students in a class. For writing program for this application basic initial effort will be to create a specific variable for each student. But if array is used for creating these variables then a single variable name will serve this purpose because an array is a collection of variables of the same type.

Program
/* Program for finding largest number from a list of 10 given numbers */
#include<stdio.h>
int main(void)
{
    int Number[10], Max =0;
    for ( int i=0; i <10; i++)/* Loop for reading the numbers in the list*/
    {
        printf("Give the Number%d in the list: ",i+1);
        scanf("%d",&Number[i]);
    }
    for ( int i=0; i <10; i++)/* Loop for finding largest number in the list*/
    {
        if( Max < Number[i])
            Max = Number[i];
    }
```c
)
printf("The Largest Number in the List is %d:\n",Max);
return(0);
}

2.10  FURTHER READINGS AND REFERENCES

2) Complete Knowledge in C by Sukhendu Dey and Debabrata Dutta,
Narosa Publishing House, Delhi, Year 2009.

Reference Web Links:
   i)  http://www.le.ac.uk/cc/tutorials/c/cccover.html
   ii) http://www.lysator.liu.se/c/bwk-tutor.html
   iii) http://en.wikipedia.org