C Programming

(Lecture-3)

Dr. Asif Uddin Khan

Programming Language

- The language understood by computer
- The language used for programming
- It is a vocabulary and set of grammatical rules for instructing a **computer** or computing device to perform specific tasks.

  Example: C, C++, Java, Python, VB, C#

Evolution of Programming Languages

- First Generation: Machine Language (01010101)
- Second Generation: Assembly Language(ADD R1, R2)
- Third Generation: High-level Language(C, C++, JAVA)
- Fourth Generation: Very High-level Language(SQL)

First Generation: Machine Language

- ***Machine language*** is the only language that the computer understands.
- In machine language, all instructions, memory locations, numbers and characters are represented in the binary form (using strings of 1s and 0s).
- The main advantage of machine language is that the code can run very fast and efficiently, since it is directly executed by the CPU.
- However, on the down side, machine language is difficult to learn and is far more difficult to edit if errors occur.

Second Generation: Assembly Language

- ***Assembly languages*** are symbolic programming languages that use symbolic notation to represent machine language instructions.
- Assembly language is also called as low-level language.
- These languages use symbolic codes also known as mnemonic codes that are easy to remember abbreviations, rather than numbers. Examples of these codes include ADD for addition, CMP for comparison, MUL for multiplication, etc.
- An assembly language program can not be executed by a computer directly as it is not in a binary form.
- **Programs written in assembly language need a translator known as the *assembler* to convert them into machine language**.

Third Generation: High-level Language

- High-level languages fall somewhere between natural languages and machine languages.
- In HLL, instructions are written using English language with symbols and digits.
- A translator is needed to translate the instructions written in high-level language into computer executable machine language. Such translators are commonly known as **interpreters** and **compilers**.
- Examples of high-level languages include FORTRAN, BASIC, COBOL, PASCAL, C, C++, JAVA etc.

Difference between Interpreter and Compiler

- Interpreters translate statement by statement where as compilers translate the entire program at a time.
- Interpreter requires more time to execute the program where as compiler requires less time to execute.
- Interpreters do not require much memory space in the computer where as compilers require large memory space in the computer.
- Interpreters are simple and easy to write where as compilers are complex and difficult.

Fourth Generation: Very High-level Language

- These languages are little different from their prior generation because they are basically non-procedural.
- When writing code using a procedural language, the programmer has to tell the computer how a task is to be done. While using a non-procedural language, the programmers define only what they want the computer to do, without supplying all the details of how it has to be done.
- A typical example of a fourth generation language is the **query language** that allows a user to request information from a database with precisely worded English-like sentences.

C Programming Language

- Developed in 1972 along with Unix at AT & T Bell Labs, USA by **Dennis Ritchie** &**Brian Kernighan**
- It is a general purpose language
- Designed for systems programming
  - Operating systems
  - Utility programs
  - Compilers
  - Evolved from B, which evolved from BCPL
- ALGOL→CPL → BCPL → B → C
- Bridge the gap between machine language and high level language.

Characteristics of C

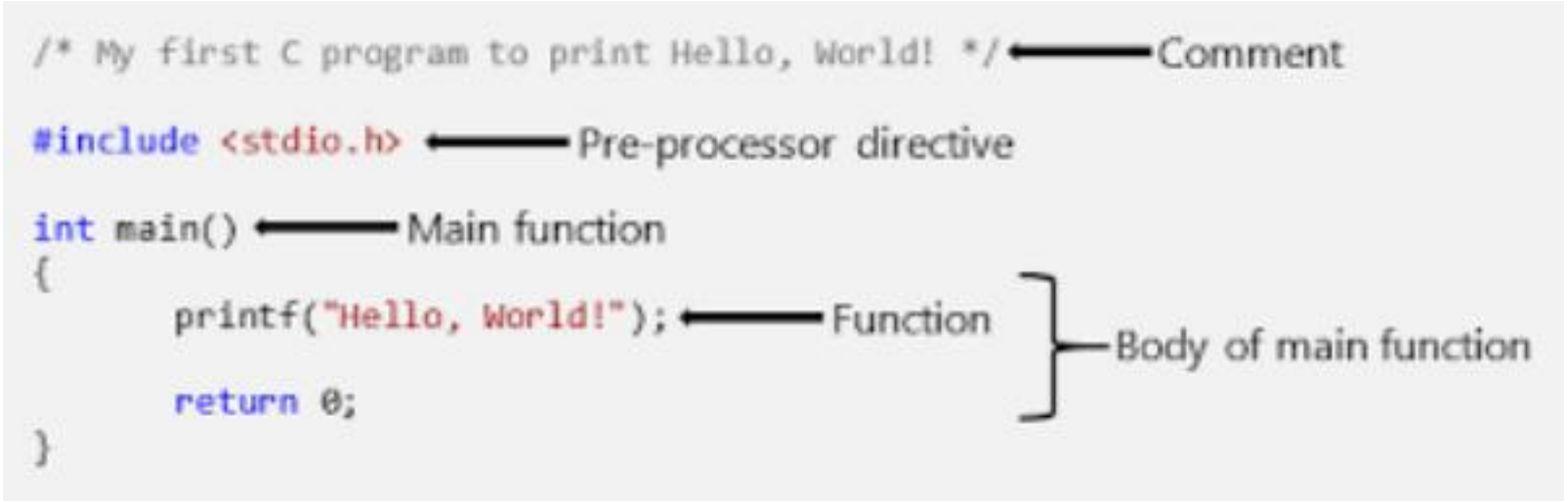
- Easy to learn
- Structured language
- It produces efficient programs
- It can handle low-level activities
- It can be compiled on a variety of computer platforms

First C Program

```
/* My first C program to print Hello World! */
#include <stdio.h>
int main(){
printf("Hello World!");
return 0;
}
```

Output: Hello World!

First C Program



- The **main function** is a **special function** because it is the entry point for program execution.
- The main function is "called" by the operating system when the user runs the program
- Returns 0 value to OS on when successfully executed

Second C Program: finding Area of a circle

```
/* C program for finding Area of a circle */
#include<stdio.h>
#define PI 3.141
int main()
{
    float radius,area;
    radius=5;
    area = PI * radius * radius;
    printf("Area of circle : %0.4f\n", area);
    return 0;
}
```

Output:

Area of circle : 78.5250

C Program Structure

- It is a function oriented structured programming language.
  - It Is a collection of functions.
  - It should contain one & only one ‘main’ function and may contain any number of other functions.
  - Beginning of main function is the entry point for program execution & end of main function is the exit point of program execution.

C Program Structure

< Documentation Section > /* First program*/

<Link Section> Example: #include<stdio.h>

<Definition section> Example: #define PI 3.141

<Global variable & function declarations section> Example: float radious;

<Main function Section>

```
main( )
{
   <variable & function declarations & initializations>   float area;
   <Input Section>
   <Program Logic>
   <Output Section>
} //end of the program
```

<Subprogram Section>

Third C Program: finding Area of a circle using gloabal variable

```
/* C program for finding Area of a circle */
#include<stdio.h>
#define PI 3.141
float radius=5;
int main()
{
    float area;
    area = PI * radius * radius;
    printf("Area of circle : %0.4f\n", area);
    return 0;
}
```

Output:

Area of circle : 78.5250

Fourth C Program(Addition of two Numbers)

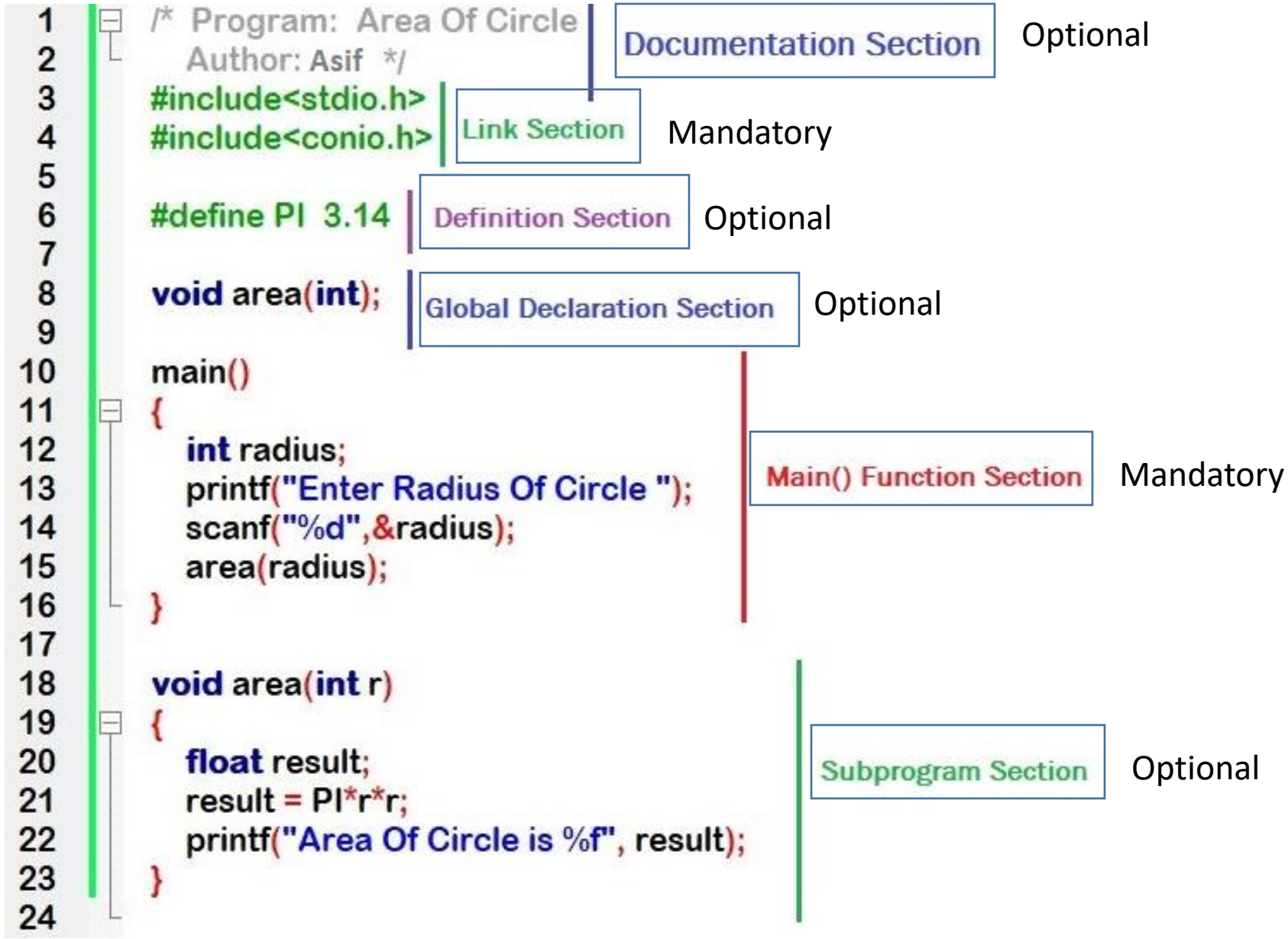
```
// Static initialization of a and b
#include<stdio.h>
int main()
{
   int a,b,sum;
   a=5;
   b=6;
   sum=a+b;
   printf("Result: %d\n", sum);
   return 0;
}
```

Output:
Result: 11

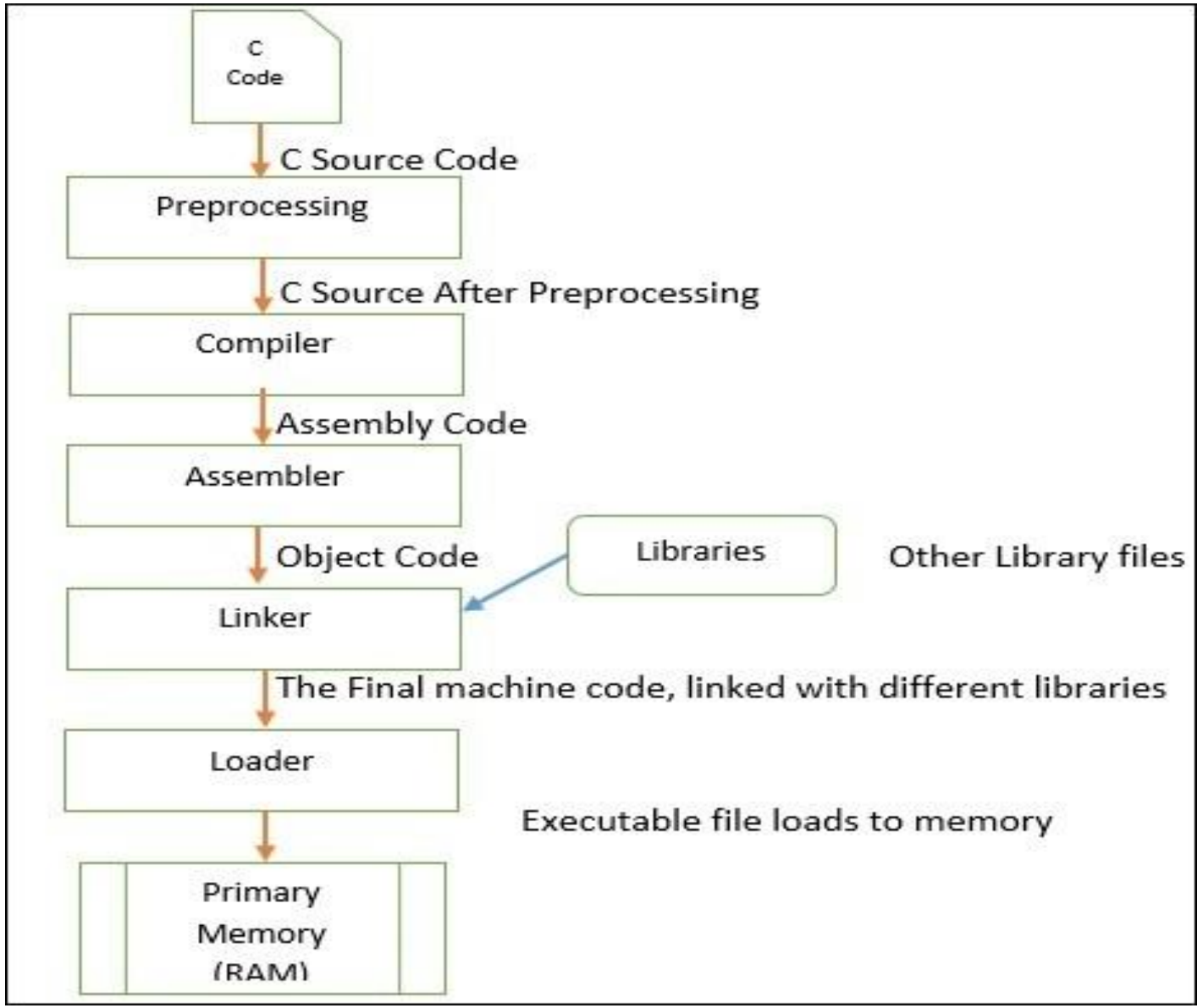
```
// Run time initialization of a and b
#include<stdio.h>
int main()
{
   int a,b,sum;
   printf("Enter value of a:");
   scanf("%d", &a);
   printf("Enter value of b:");
   scanf("%d", &b);
   sum=a+b;
   printf("Result: %d\n", sum);
   return 0;
}
```

Output:
Enter value of a:5
Enter value of b:6
Result: 11

C Program Structure



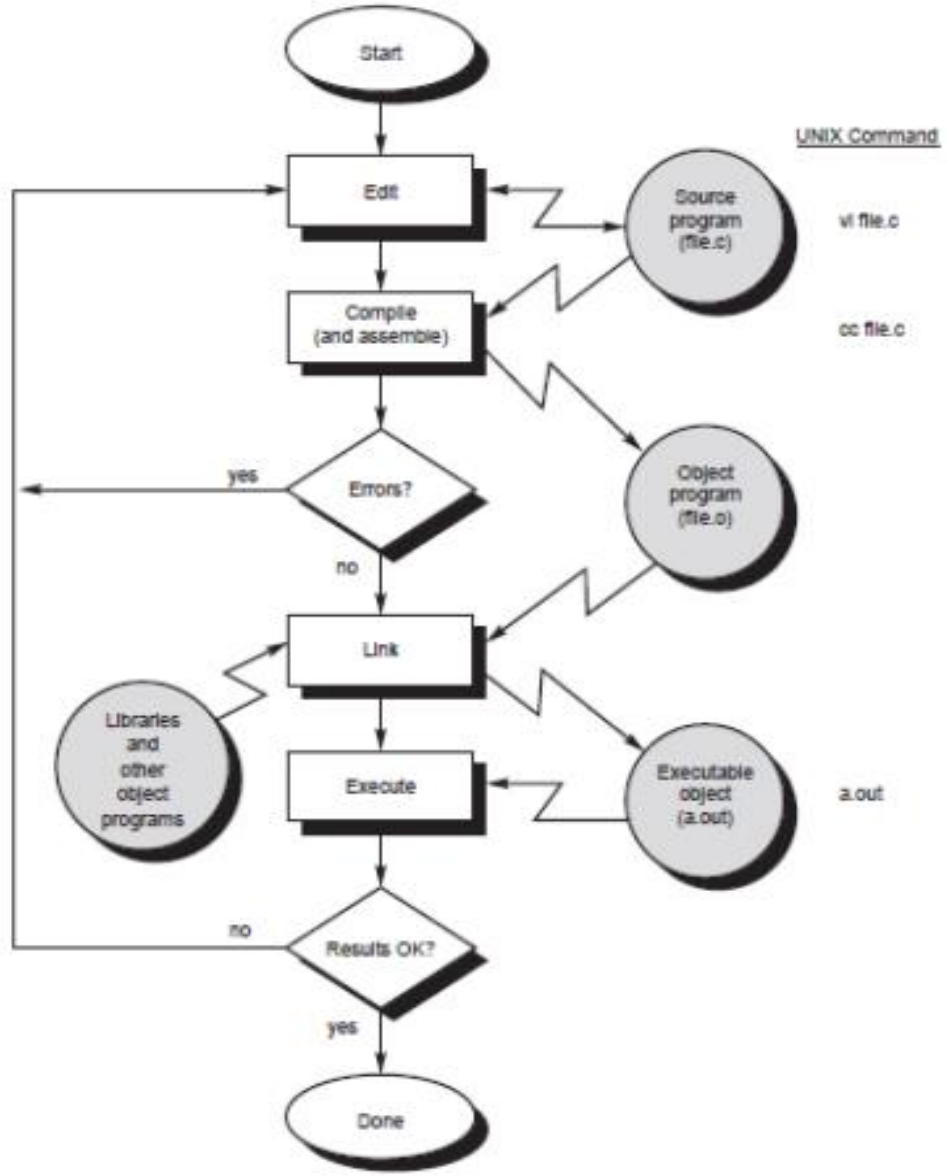
Execution Steps of a C Program



Execution Steps of a C Program

- **C Code** − This is the code that you have written. This code is sent to the Preprocessors section.
- **Preprocessing** − In this section the preprocessor files are attached with our code. We have used different header files like stdio.h, math.h etc. These files are attached with the C source code and the final C Source generated. (‘#include’, ‘#define’ These are Preprocessor Directives.)
- **Compiler** − After generating the preprocessed source code, it moves to the compiler and the compiler generates the assembly level code after compiling the whole program.
- **Assembler** − This part takes the assembly level language from compiler and generates the Object code, this code is quite similar to the machine code (set of binary digits).
- **Linker** − Linker is another important part of the compilation process. It takes the object code and link it with other library files, these library files are not the part of our code, but it helps to execute the total program. After linking the Linker generates the final Machine code which is ready to execute.
- **Loader** − A program, will not be executed until it is not loaded in to Primary memory. Loader helps to load the machine code to RAM and helps to execute it. While executing the program is named as Process. So process is (Program in execution).

Steps for Compiling and executing C Programs in a Unix based system



Example

Open file and Write Program

$vi hello.c

Compile the program

$cc hello.c

or

$cc hello.c -o hello

Executing the program

$./a.out

or

$./hello

Day 03:Lab Assignment

Q#	Experiment Details	Input	Output
1.	WAP to perform the addition of two integers and display the result. Input must be given by user.	Enter 1st number:12 Enter 2nd number:13	Sum is 25.
2.	WAP to find Fahrenheit for a given centigrade temperature.	Enter the temperature in Centigrade: 30	The Fahrenheit temperature is: 86
3.	WAP to calculate area of a circle while taking radius as user input.	Enter the radius of the circle: 11	The area is: 380.12
4	WAP to calculate area of a triangle who's base and height are user input.	Enter the height of the triangle: 12 Enter the base of the triangle: 10	The area of the triangle is: 60.
5	Write a C program to perform swapping of two integers using a third variable.	Enter num1: 10 Enter num2: 20	Before Swapping num1=10, num2=20 After Swapping num1=20, num2=10

Formulae: $°F = °C \times (9/5) + 32$

Day 03:Home Assignment

Q#	Experiment Details	Input	Output
1.	Write a C program to perform swapping of two integers without using a third variable.	Enter num1: 10 Enter num2: 20	Before Swapping num1=10 num2=20 After Swapping num1=20 num2=10
2.	WAP to find the average mark of 5 subjects of a student and find the percentage. Assume full mark of each subject is 200. All the input must be given by user.	Enter the number of 1st subject: 160 Enter the number of 2nd subject: 170 Enter the number of 3rd subject: 165 Enter the number of 4th subject: 180 Enter the number of 5th subject: 185	The average is: 172. The percentage is: 86%
3	WAP to convert given paisa into its equivalent rupee and paisa as per the following format.	Enter the amount:550 paisa.	550 paisa = 5 Rupee and 50 paisa