# Module 3 – Basic C Program and its execution


BITS Pilani
Pilani Campus

Dr. Jagat Sesh Challa
Department of Computer Science & Information Systems

# Module Overview



- Basic C Program
- Compilation and Execution of a C Program
- Errors in C Programs

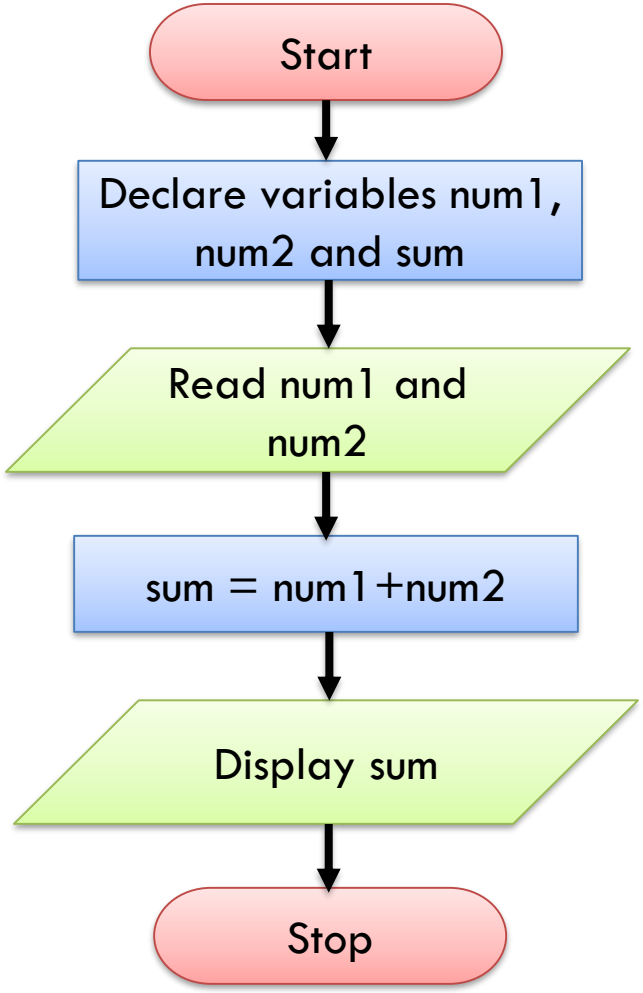

# Basic C Program

# Steps of Programming Practices

innovate achieve lead

- **Step1:** Requirements
- **Step2:** Creating a flow chart
- **Step3:** Creating algorithms
- **Step4:** Writing Code
- **Step5:** Debugging
- **Step6:** Documentation

# Example – Sum of Two Numbers




1. **START**
2. Initialize sum=0
3. **INPUT** the numbers num1, num2
4. Set sum = num1 + num2.
5. **PRINT** sum
6. **END**

# C Code for Sum of Two Numbers



```
/* myfirst.c: to compute the sum of two numbers */
#include<stdio.h>
/*Program body*/
int main()
{
   int a, b, sum;   //variable declarations
   printf("Please enter the values of a and b:\n");
   scanf("%d %d", &a, &b);
   sum = a + b;   // the sum is computed
   printf("The sum is %d\n", sum);
   return 0; //terminates the program
}
```

Preprocessor Directive

The main() function where any program's execution begins

Block of the `main()` function. Any block is enclosed in {…}

`a, b, sum` are variables or placeholders for values

A block consists of statements ending with `;`

# Header files



```
#include <stdio.h>
```

- A directive to place the contents of the header file, `stdio.h`, in the program
- A header file contains data used by multiple programs
- Processed by a preprocessor

# Functions



```
printf("Please enter the values of a and b:\n");
scanf("%d %d", &a, &b);
```

- Called by its name to execute a set of statements
- Multiple programs can use the same function
- **`printf`** is used to print some value to the screen
- **`scanf`** is used to read some value from the user
- **`main`** is a function
- A program can consist of many more *user-defined functions*.

# Adding comments



```
/* This is a comment */
// And so is this
```

Judicious use of comments helps in easy understanding of the code (for yourself and others)

innovate achieve lead

BITS Pilani

Pilani Campus

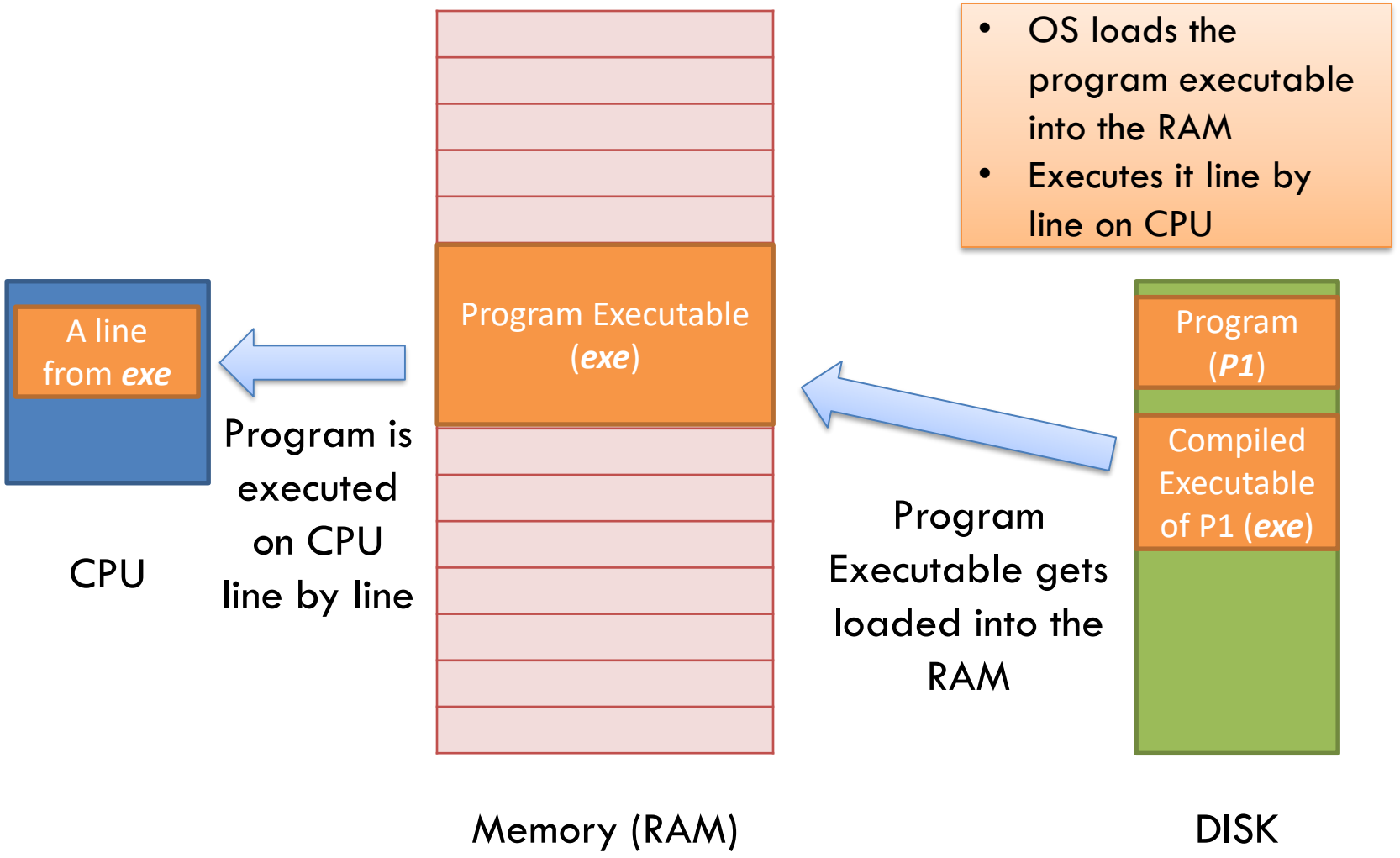
# Compilation & Execution of a C Program

# Steps to run a C Program



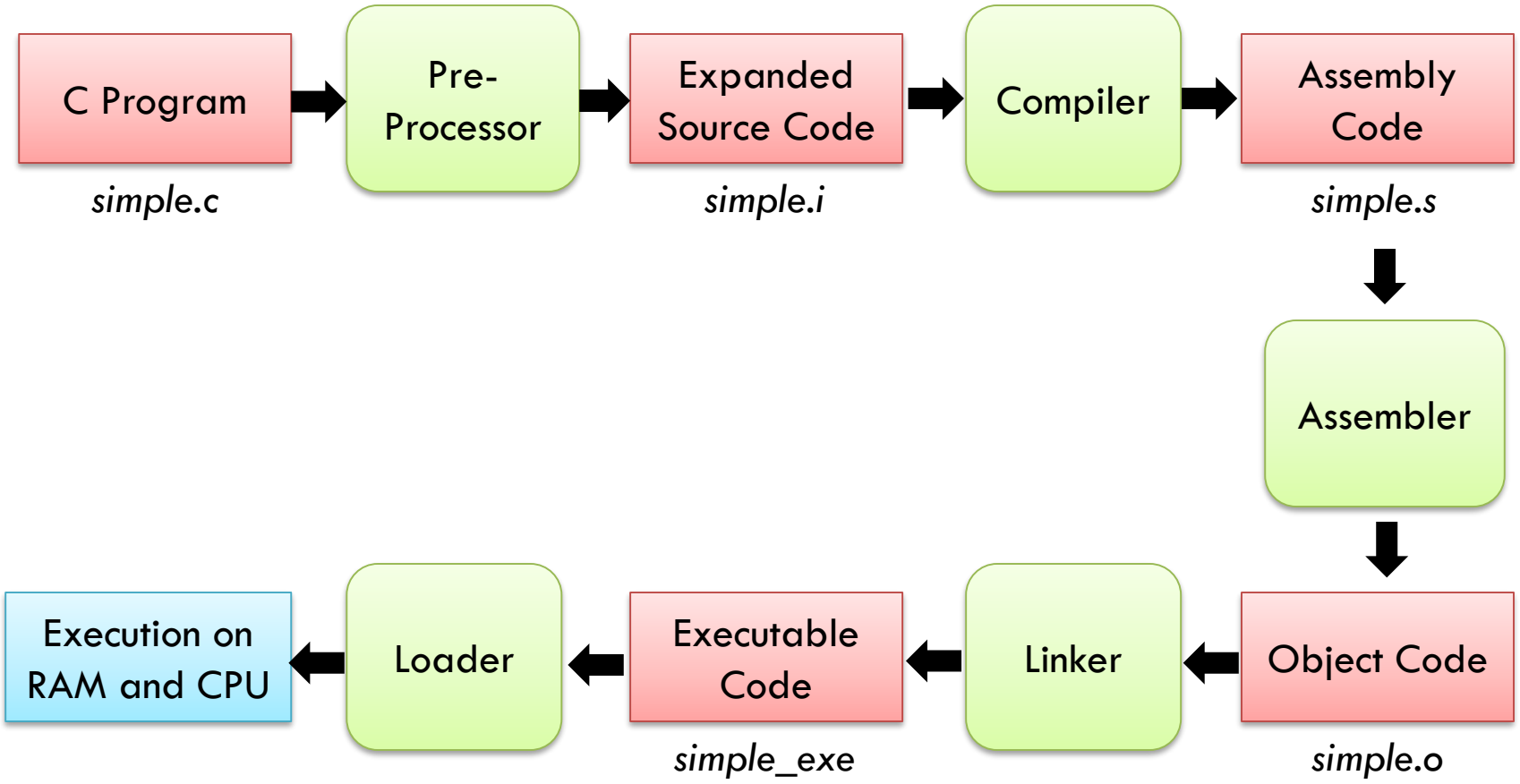
1) Write the program and save it
   - Where does it get saved? ← *Disk*
2) Compile the program to generate its executable
   - Where will the executable be saved? ← *Disk*
3) Run the executable
   - Where will the executable run? ← *Executable is loaded into RAM and executed line by line on the CPU*

# Retrospection of Program execution (in a better way)




- OS loads the program executable into the RAM
- Executes it line by line on CPU

# The compilation process (Illustrated)



# The compilation process



## The Preprocessor

- A Text manipulation phase that gives expanded source code to be fed into the compiler
- Removes comments
- Processes lines beginning with # (preprocessor directives)
  - #include… or #define PI 3.142

## The Compiler

- Checks the source code for errors
- Creates object (or machine) code (not ready to run) (why?)

# The compilation process (contd.)

innovate achieve lead

## The Linker

- Acts on unresolved references left by compiler
- `printf`, `scanf`… their machine code is added
- Combines multiple object files resultant of compiler phase
- This code is ready to run

## The Assembler (optional)

- Some compilers first translate to assembly language.
- The assembler converts it to machine code

# Compiling C Program

innovate achieve lead

Say, our C program is stored in **`myfirst.c`**

To generate executable file, navigate to the directory where **`myfirst.c`** is stored and run the following command:

```
$ gcc myfirst.c
```

Files generated (executable): **`a.out`**

The above process is known as **compilation** of the program to generate the executable **`a.out`**. To run the executable:

```
$ ./a.out
```

# Compiling C Program (contd.)



C Compiler allows you to generate intermediate temporary files during its compilation. To generate them use the following command:

```
$ gcc myfirst.c -save-temps -o myfirst.exe
```

Files generated:

**myfirst.exe**

**myfirst.o**

**myfirst.s**

**myfirst.i**

**myfirst.c** (our original C code)

Exercise: Open each of the above files in a suitable text editor and see what is inside…

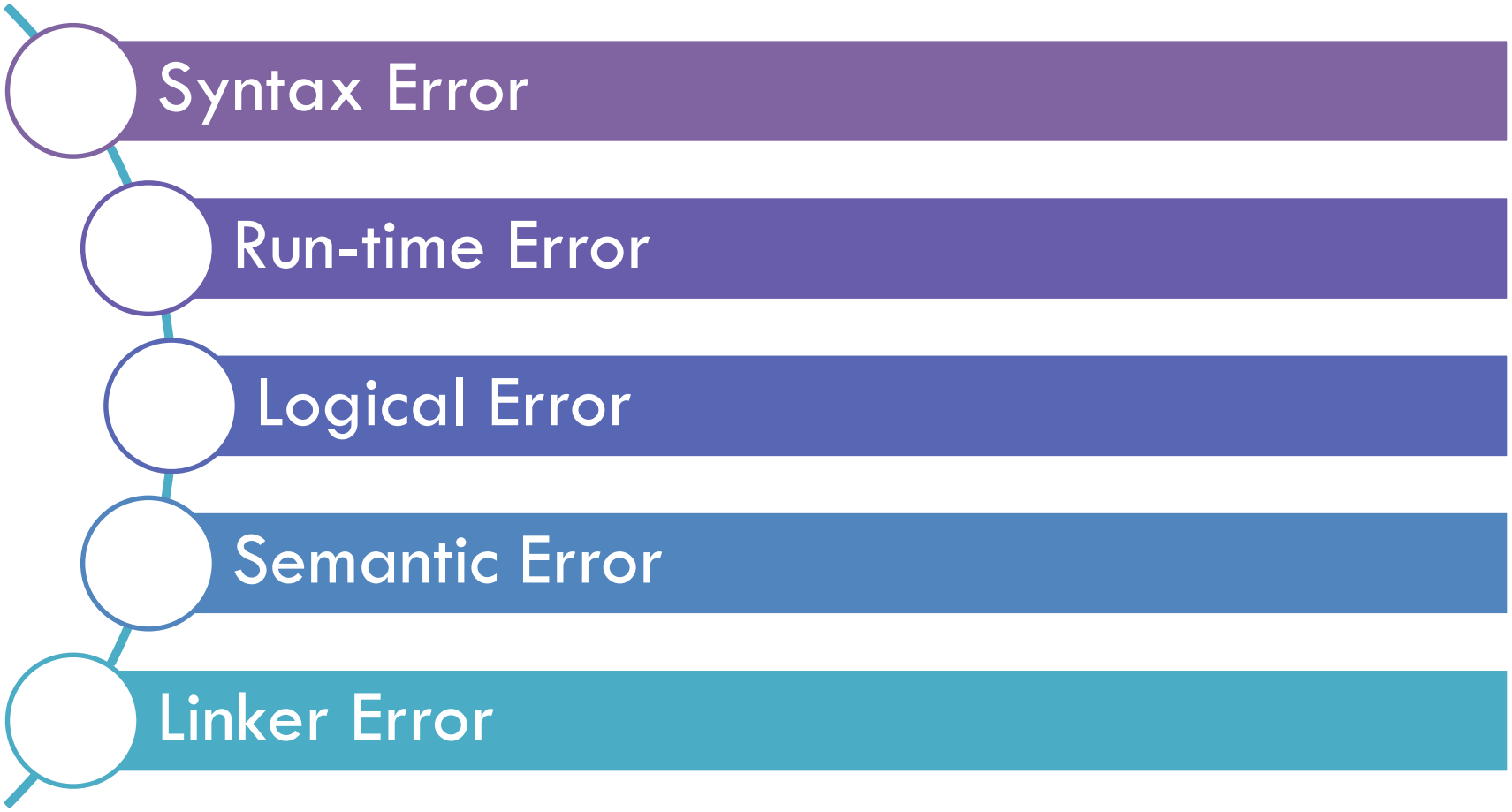
innovate achieve lead

BITS Pilani

Pilani Campus

# Errors in C Programs

# Errors in C Programs



# Syntax Errors



- Occur when programmers make mistakes in typing the code's syntax correctly
  - Rules of C syntax are not followed
- Detected in the compiler phase
- Programmers can detect and rectify them easily
- Most common syntax errors:
  - Missing semi-colon (;)
  - Missing or open parenthesis ({})
  - Assigning value to a variable without declaring it

# Exercise: Find out Syntax Errors in this Program


```
#include <stdio.h>
void main()
{
  int a = 10;
  var = 5;
  printf("The variable is: %d", var)
  for (int i = 0;) {
     printf(”BITSians on ROLL");
  }

```

undeclared variable `var`

missing semi-colon

wrong syntax

missing parenthesis

# Run-time Errors



- Errors that occur during the execution of a program
- Occur after the program has been compiled successfully
- Identified only when a program is running the program
- Common run-time errors:
  - dividing a number by zero
  - calculating square root of -1
  - array index out of bounds
  - infinite loops due to missing break statement
  - memory leaks

# Example

```
#include <stdio.h>
void main()
{
    int n = 9, div = 0;
    div = n/0;

    printf("result = %d", div);
}

Output:
warning: division by zero [-Wdiv-by-zero] div = n/0;
```

# Logical Error



- Code is successfully running without compilation and run-time errors
- But output is not as intended
  - There is a logical error

Consequence:

In 1999, NASA lost a spacecraft due to a logical error

# Example of Logical Error

innovate achieve lead

```
#include <stdio.h>
void main() {
   float a = 10;
   float b = 5;
   if (b = 0) // we wrote = instead of ==
   {
       printf("Division by zero is not possible");
   }
   else
   {
       printf("The output is: %f", a/b);
   }
}

Output:
The output is INF
```

# Semantic Error

Errors that occur because the compiler is unable to understand the written code, although the code adheres to the syntax structure.

Detected during compiler phase

# Example:

```
#include <stdio.h>
void main()
{
   int a, b, c;
   a * b = c; // This will generate a semantic error
}



Output:
error: lvalue required as left operand of assignment
```

# Linker Error

Can occur when we link multiple files to create an executable.

We will study in Lab Sheet 8.

# Don’t be errored!

innovate achieve lead

Don’t worry if you haven’t understood everything!

You will experience all these errors when you start coding.

# Write your first code



Write a program in C to compute the average of 3 numbers. Take the numbers as an input by the user.

innovate achieve lead

BITS Pilani

Pilani Campus

# Thank you

## Q & A