GEDT019
Basis and Practice in Programming

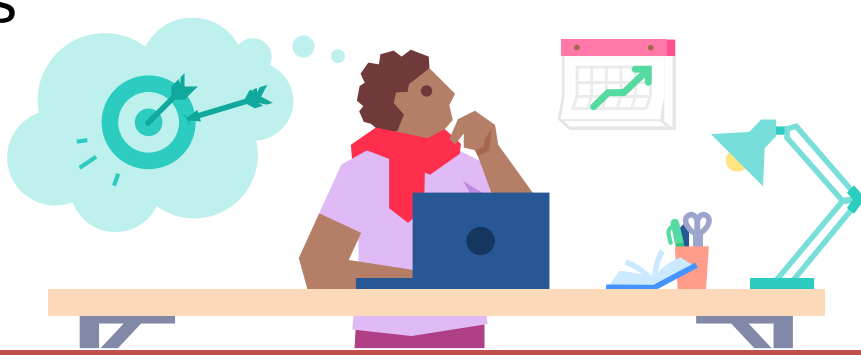
Chapter 1: Getting Ready

Prof. Tamer ABUHMED
College of Software

성균관대학교
SUNGKYUNKWAN UNIVERSITY(SKKU)

Lecture Objectives

- Course Overview
- To explain the merits of C language
- To explain where C language is used
- To explain the usage steps of C language
- To explain the operation of the compiler and linker
- To introduce the C language Standards
- Simple example



Syllabus

Course Overview

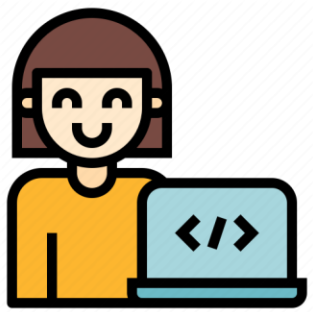
- What is this course about?
- Learn how to program a software using C programming language

- At the end of this course, you should know how to read, write, debug, analyze, and optimize C programs, that include the topics covered in the course, whether these programs are written by you or by others.

Course Overview

Hopefully frustration-free:

- We will go slowly through the essential concepts and speed through the obvious stuff
- The secret to master programming is PRACTICE. PRACTICE? PRACTICE!
- Can’t passively learn programming as a skill
- Download codes of the lecture and play with it



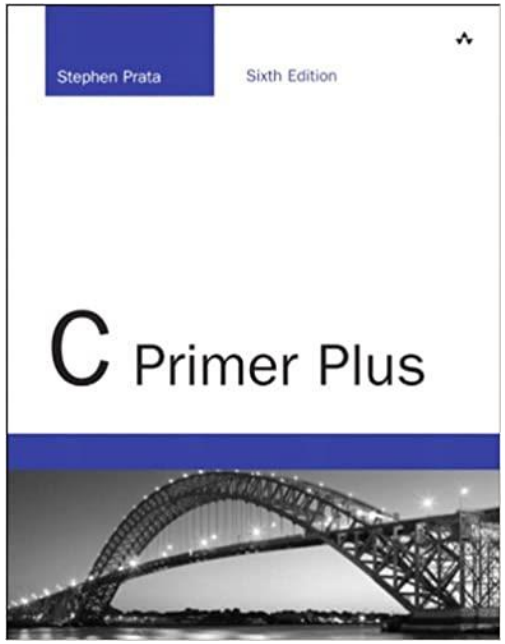
Topics

- Structure of the C program
- Data types in C programming language (Variables types, names, storage)
- Operators, expressions, and statements
- Character strings & formatted I/O
- Control statements : Looping; Branching;
- Building functions in C programming
- Introduce the arrays in C language
- Pointers in C programming language
- File Input / Output in C
- Character and Strings handling
- String Functions

Textbook

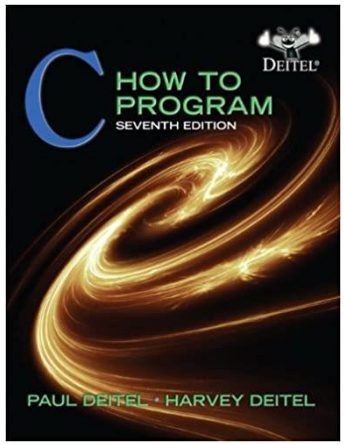
C Primer Plus by Stephen Prata (6th Edition) ★★★★☆

This book teaches principles of C programming, including structured code. Many short, practical examples illustrate one or two concepts at a time, encouraging readers to master new topics by immediately putting them to use



C: How to Program by Paul Deitel, Harvey Deitel (7th Edition) ★★★★☆

The first 7 chapters of the book teach you the basic concepts of C and the second half of the book is an intro to C++ and object-oriented programming.

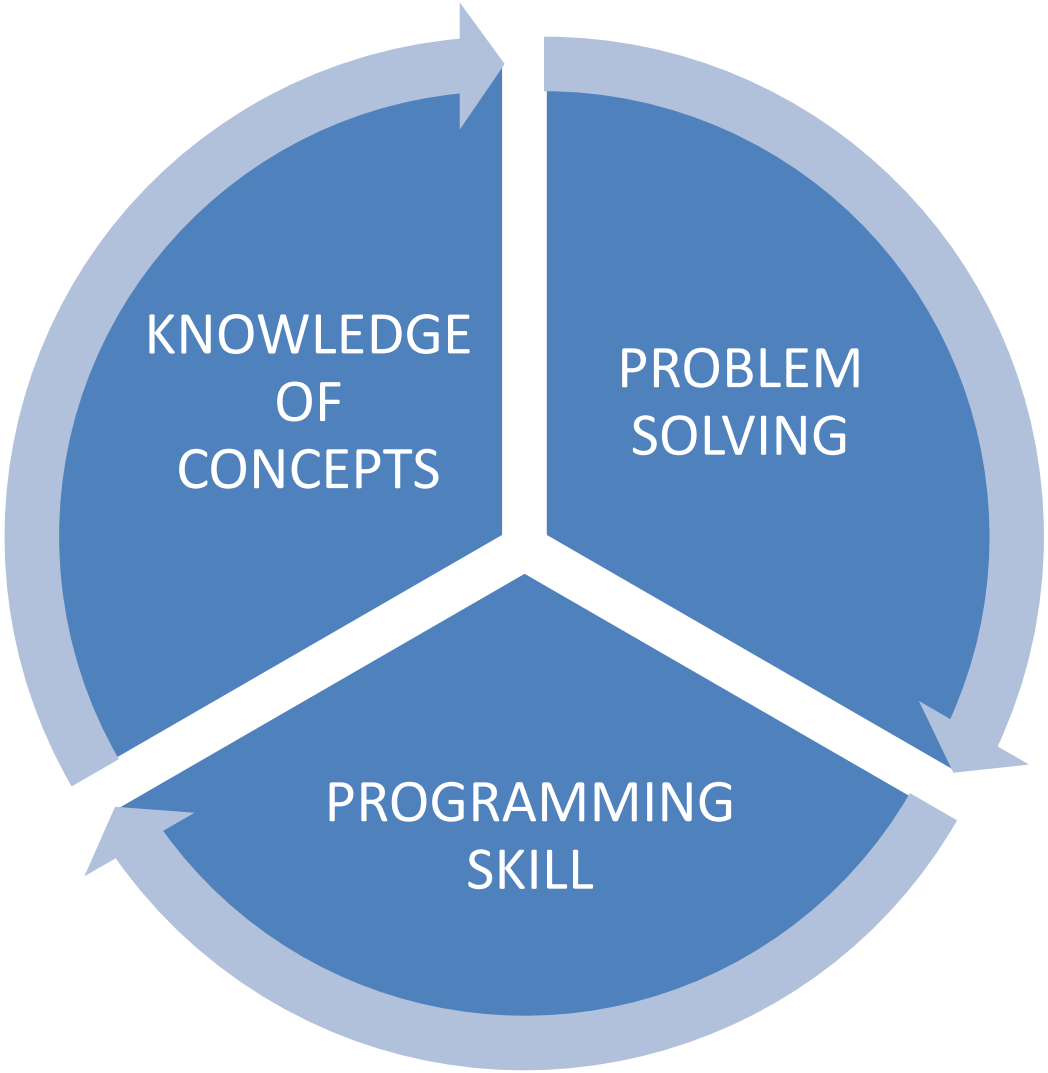


Grading Policy

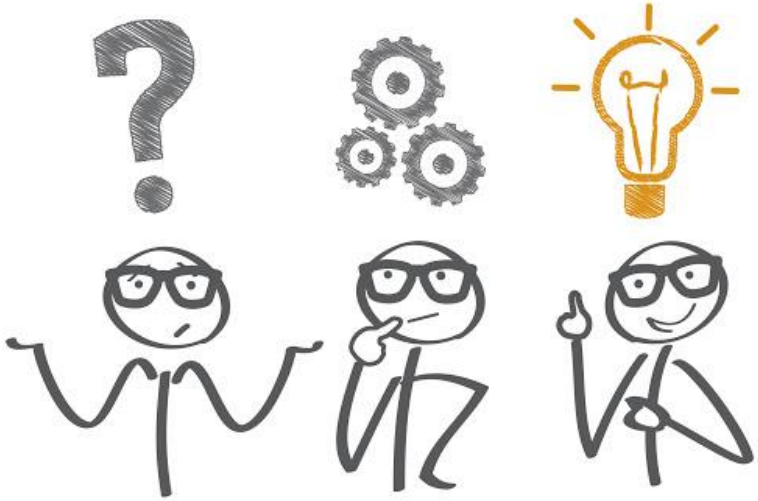
Attendance 10%
Assignments 30 %
Midterm exam 30%
Final Exam 30%

- Grading policy might change based on the course teaching method.

Skills with Programming

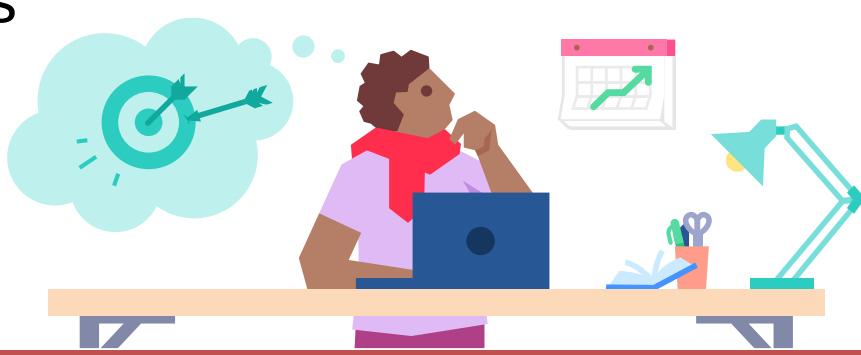


PRACTICE



Lecture Objectives

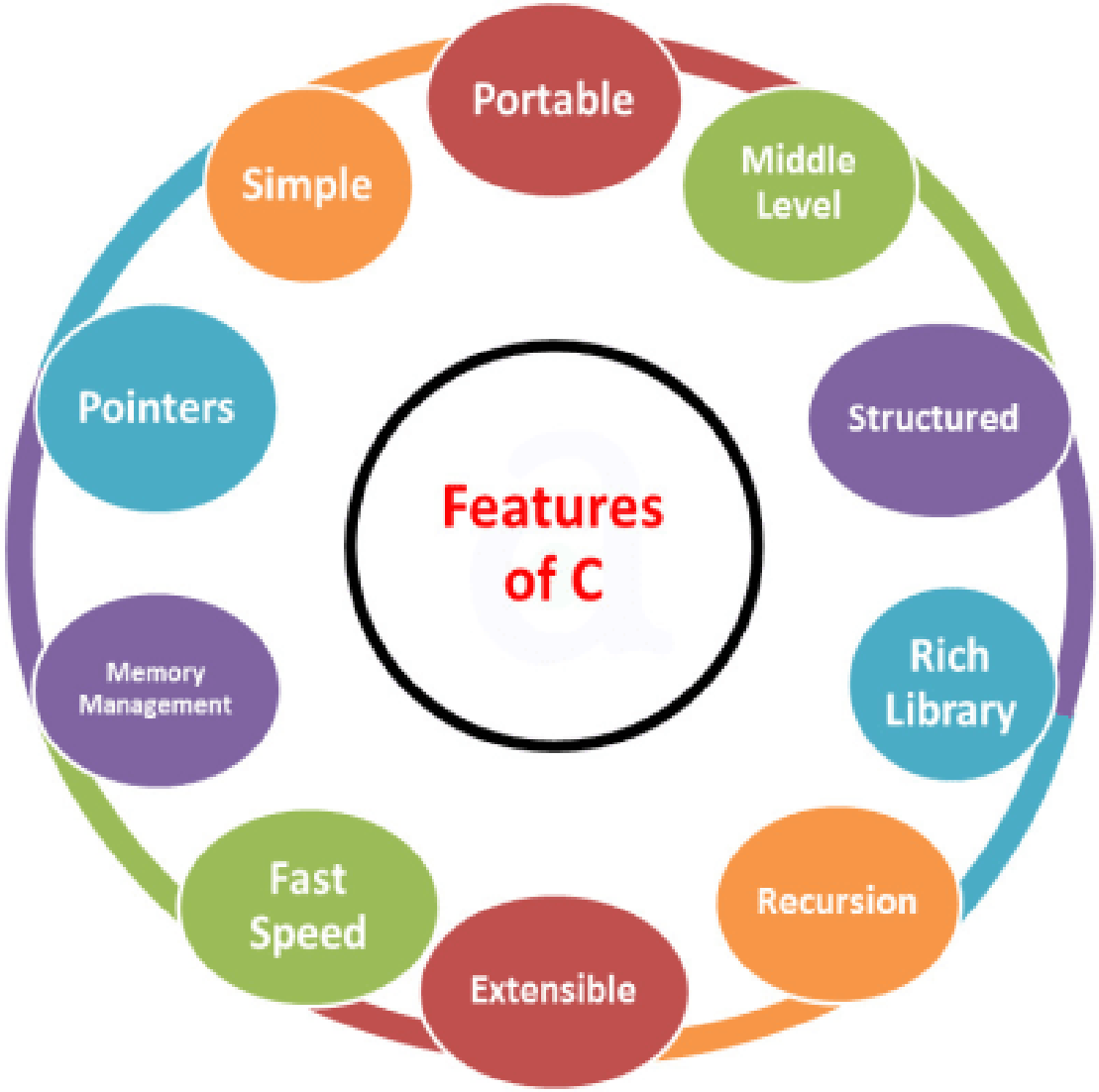
- Course Overview ✓
- To explain the merits of C language
- To explain where C language is used
- To explain the usage steps of C language
- To explain the operation of the compiler and linker
- To introduce the C language Standards
- Simple example



Why C?

- Efficiency
  - Maximum speed and efficient use of memory
- Portability
  - Programs written on one system can run on other systems
    - This might require little or no modifications
- Power & Flexibility
  - Most of the powerful, flexible Unix system are written in C
  - Many compilers for other languages – such as FORTRAN, Python – are written in C
- Programmer Oriented
  - C fulfills programmer's needs such as accessing hardware, accessing bits in memory
  - C has many built-in functions that deal with the needs that a programmer faces

Why C?

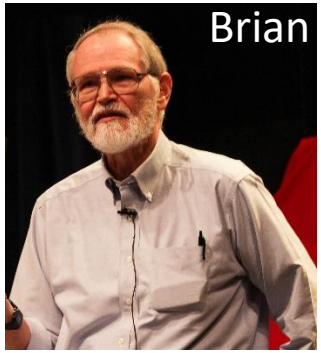

Features of C
Portable
Middle Level
Structured
Rich Library
Recursion
Extensible
Fast Speed
Memory Management
Pointers
Simple

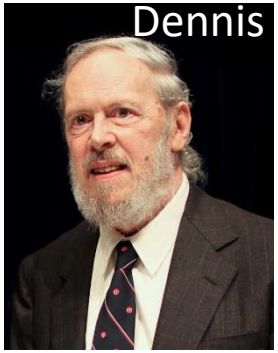
Where C is Used?

C language

- UNIX system
- Computer languages
- PC applications
- Robot factories
- Computer games
- Embedded systems

C Language Standards





- Classic C
  - 1978 by Brian Kernighan & Dennis Ritchie (K&R C)
- The First ANSI/ISO C Standard
  - 1989 by the American National Standards Institute (ANSI)
    - Referred to as “ANSI C” or “C89”
    - The standard defined both the language and a standard C library
  - 1990 by the International Standardization Organization (ISO)
    - Referred to as “C90” or even “ANSI C!”
  - "C18" is the current standard for the C programming language [2018]
- The C99 Standard
  - Built by a joint ANSI and ISO committee known as “C9X committee”
  - Main change-oriented goals: (1) Internationalization, (2) correction of deficiencies, and (3) improvement of computational usefulness

Introduction

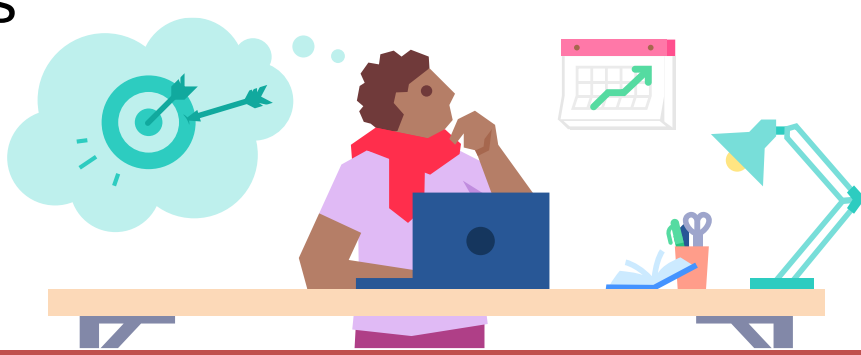
- What is a programming language?
  - It is *vocabulary* and set of *grammatical* rules for instructing a computer to perform specific *tasks*.

- Main components of a programming language
  - Syntax: The grammatical rules used to write a code
  - Semantics: The meaning of the written code
  - Statements: Instructions that indicate what to do such loops, arithmetic operations, control statements, etc.
  - Variables: named memory used to hold data including numbers and texts

- High-level vs. low-level programming languages
  - High-level: more abstract, more complicated syntax and semantics, variables and statements are richer, etc.

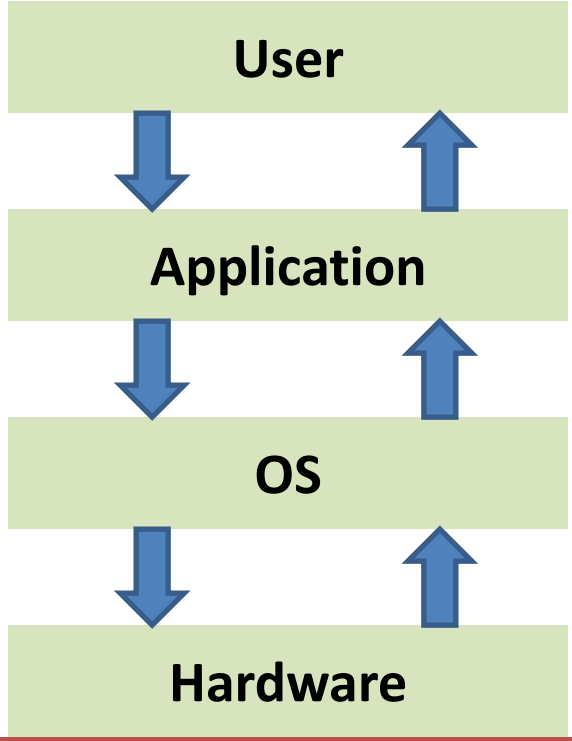
Lecture Objectives

- Course Overview ✓
- To explain the merits of C language ✓
- To explain where C language is used ✓
- To explain the usage steps of C language
- To explain the operation of the compiler and linker
- To introduce the C language Standards
- Simple example

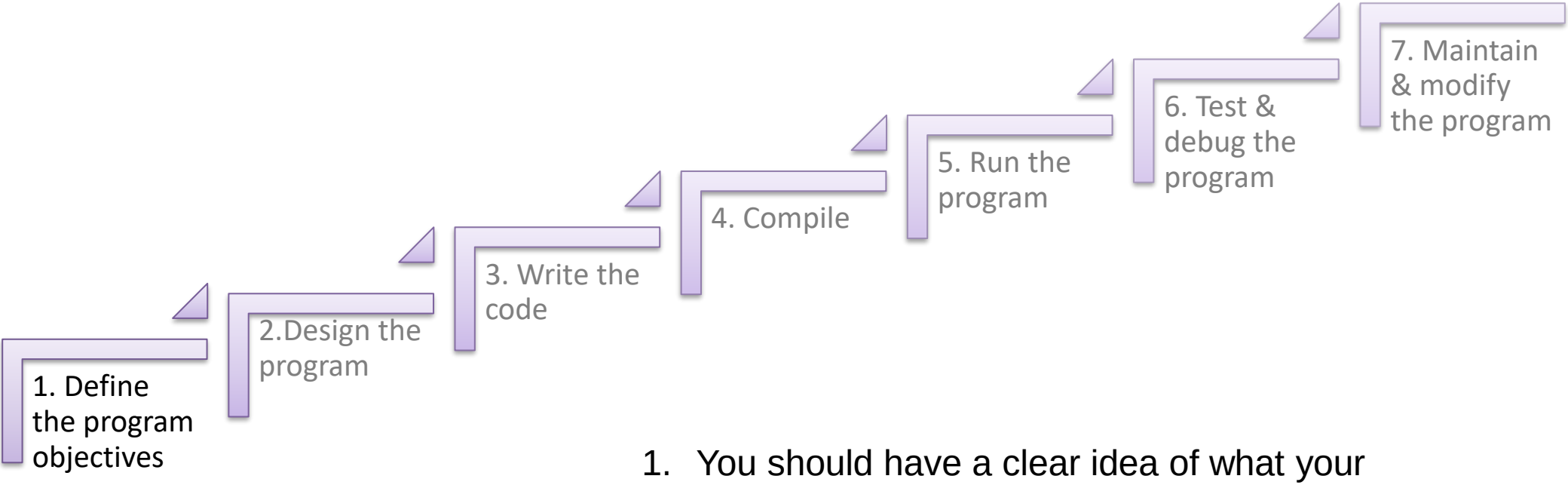


Application and OS

- What is an operating system?
  - System software that manages computer *software* and *hardware* and through which applications get access to hardware.
  - Applications frequently make *system calls* to an operating system function to access the hardware, including *memory*.
- OS
  - Windows
  - Lunix
  - macOS Sierra
  - Android



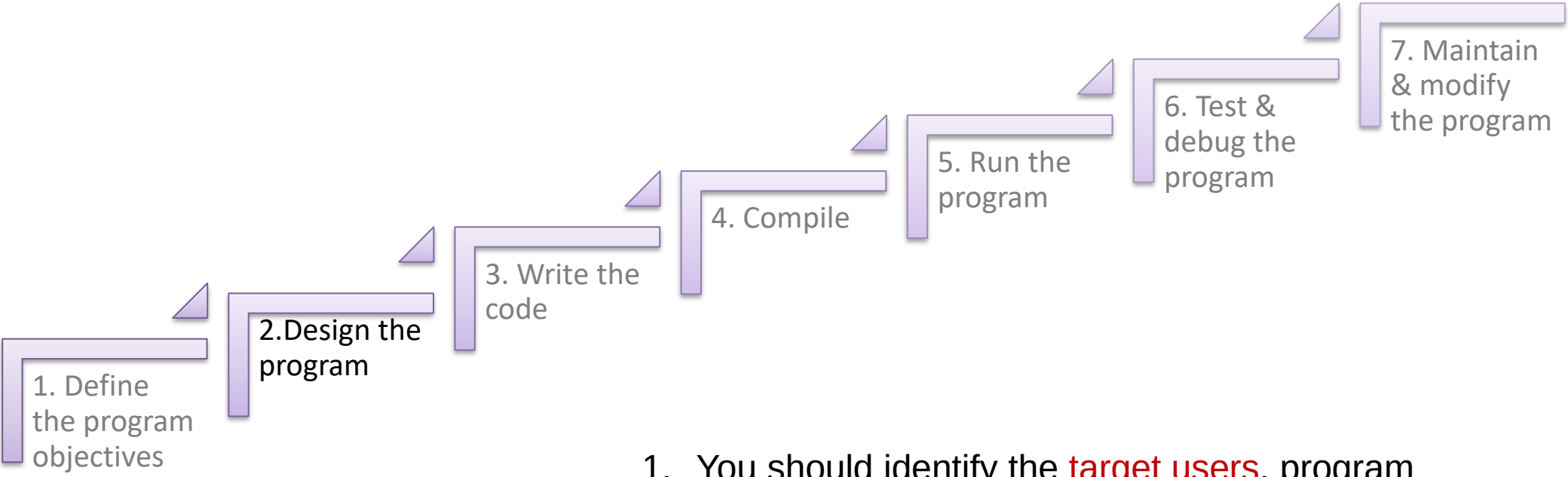
Using C: Seven Steps



1. You should have a clear idea of what your program is supposed to do

2. You should know precisely what are the inputs and output of your program

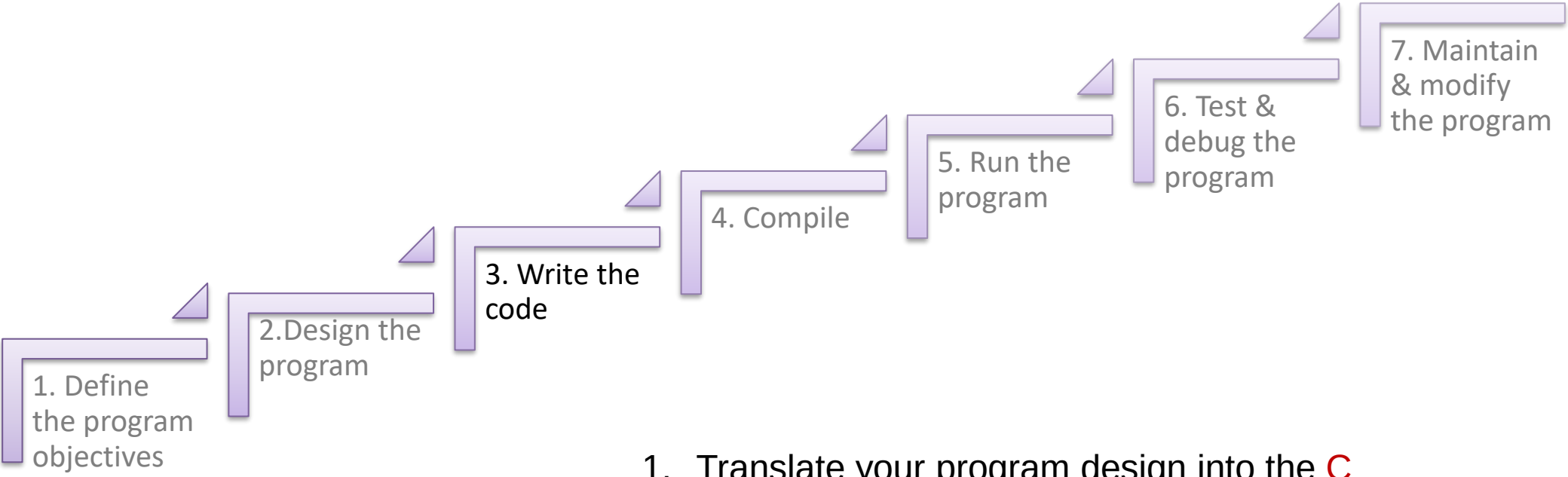
Using C: Seven Steps – contd.



1. You should identify the target users, program organization, program interface, and time required for programming

2. You should know precisely how to represent the data in the program, or in files, and which methods to process the data

Using C: Seven Steps – contd.

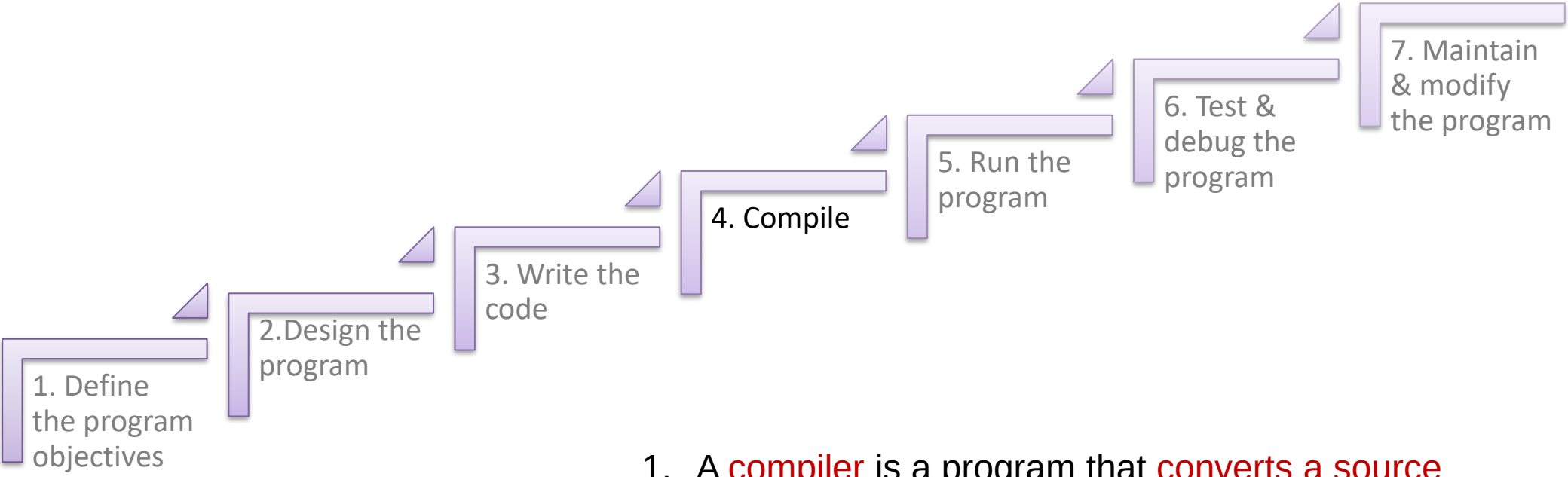


1. Translate your program design into the C language (i.e., write the source code)

```
# include <stdio.h>
void main()
{
    int dogs;

    printf("How many dogs do you have?\n");
    scanf("%d", dogs);
    printf("So you have %d dog(s)!\n", dogs);
}
```

Using C: Seven Steps – contd.

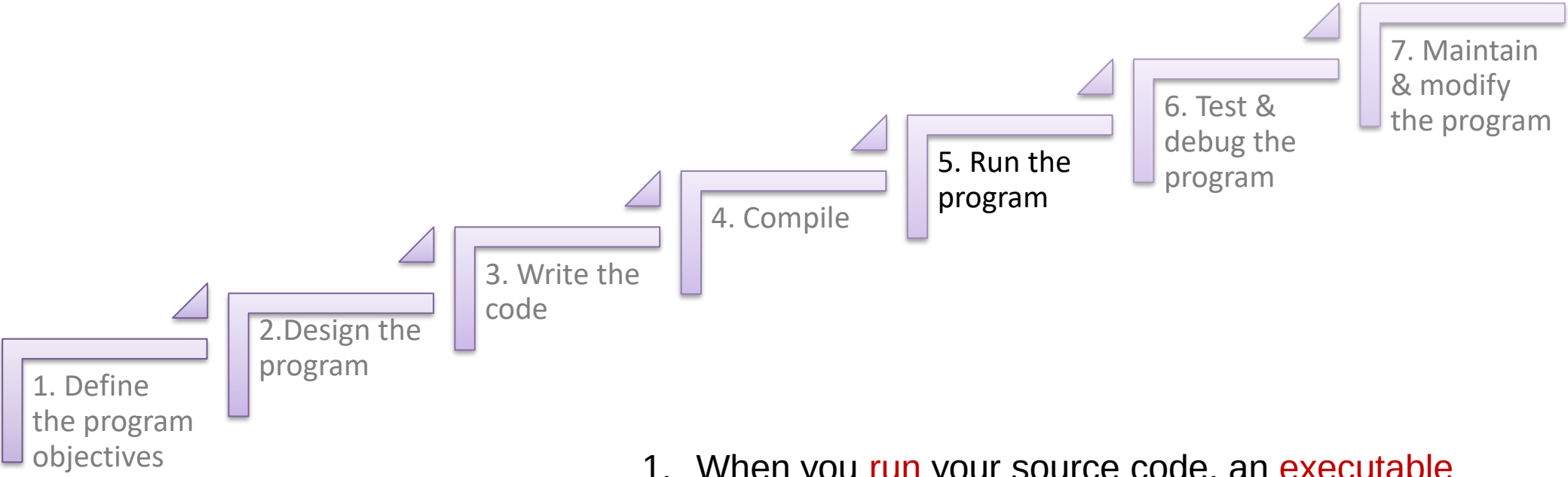


1. A compiler is a program that converts a source code into an executable code. The executable code is the native language of the computer

2. The compiler runs the linker which brings in the library routines

3. The complier checks that your program is a valid C

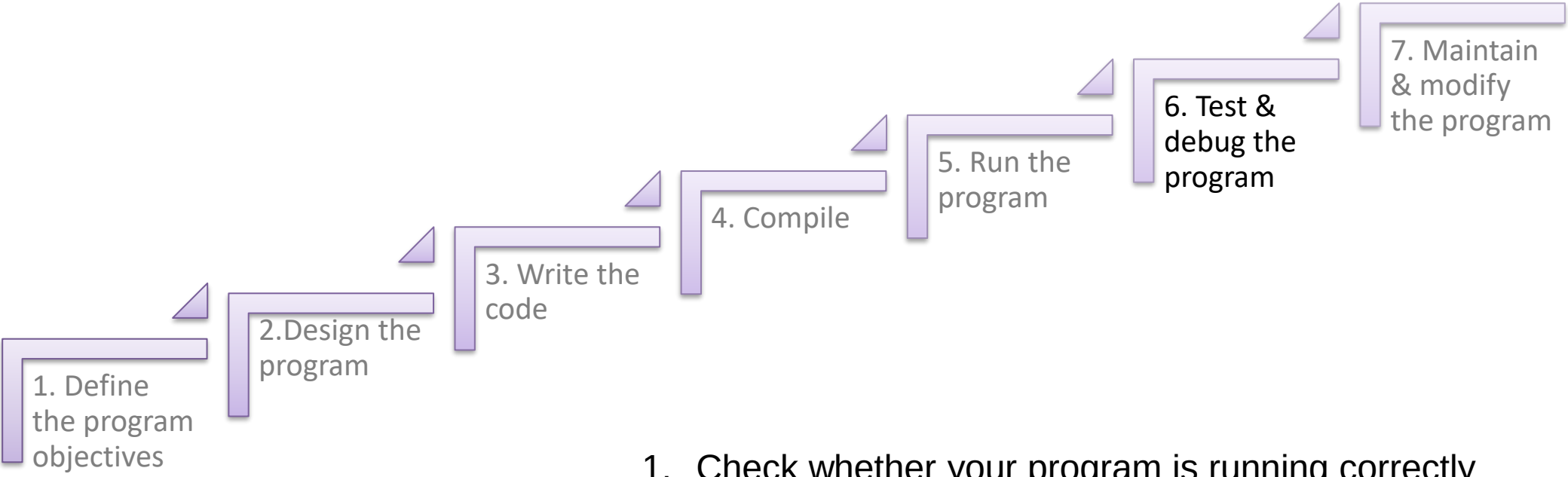
Using C: Seven Steps – contd.



1. When you run your source code, an executable code is generated.

   This executable code can be run from within the C editor - a.k.a. integrated development environment (IDE) – or in the operating system as a program

Using C: Seven Steps – contd.

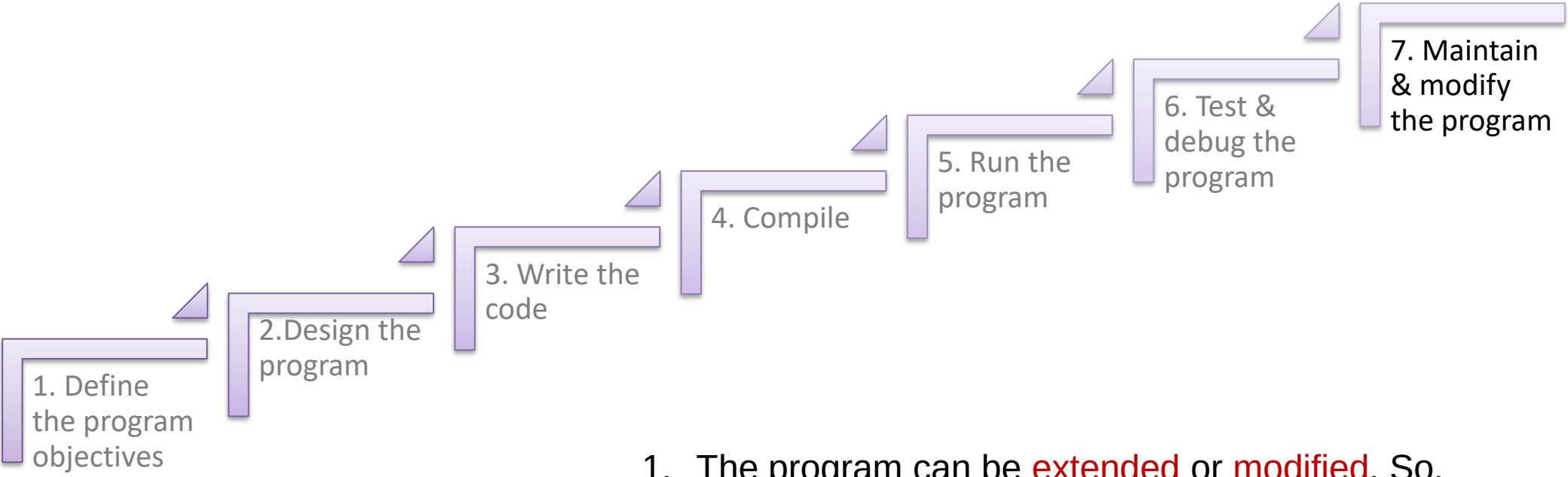


1. Check whether your program is running correctly or not, a.k.a., *debugging*

2. Programming mistakes are referred to as *bugs* and fixing the mistakes is therefore referred to as *debugging*

3. *Bugs* are the errors that the compiler does not catch

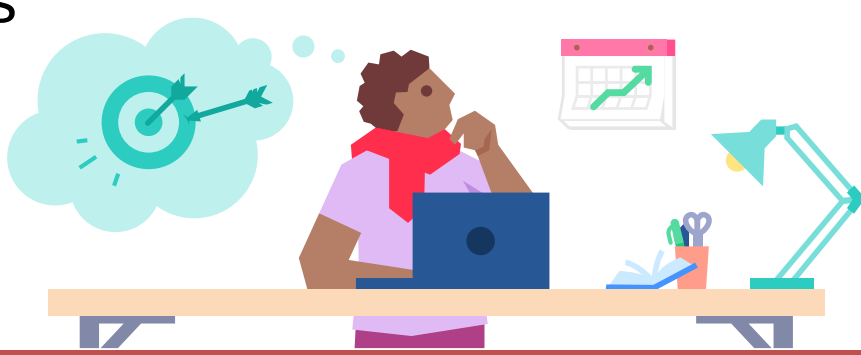
Using C: Seven Steps – contd.



1. The program can be extended or modified. So, keep your code
   a) documented,
   b) stable,
   c) and adaptive.

Lecture Objectives

- Course Overview ✓
- To explain the merits of C language ✓
- To explain where C language is used ✓
- To explain the usage steps of C language ✓
- To explain the operation of the compiler and linker
- To introduce the C language Standards
- Simple example

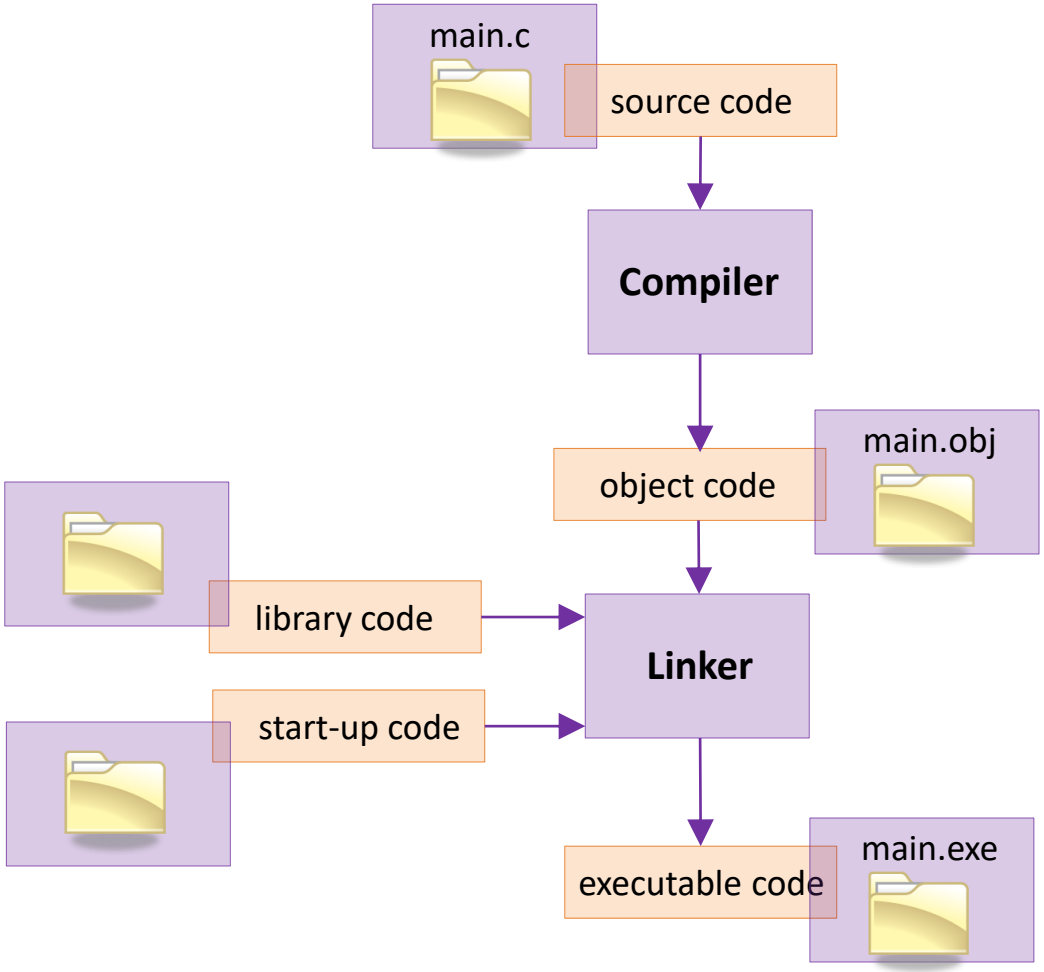


Compiled vs. Interpreted

- Compiled vs. Interpreted languages
  - A *compiler* directly translates the programming language's code into *machine language* before running the resulting program.
  - An *interpreter*, on the other hand, translates the high-level language into *intermediate instructions*, not a machine code, and execute them.
  - Compiled languages are faster than interpreted languages. If the program is long, compilers require relatively long time as they need to compile all the code. Certain compilers compile only those files that are modified, resulting in reduction in the compiling time.

Complier and Linker

- Compiler & Linker



A Simple C Program

- The example explained

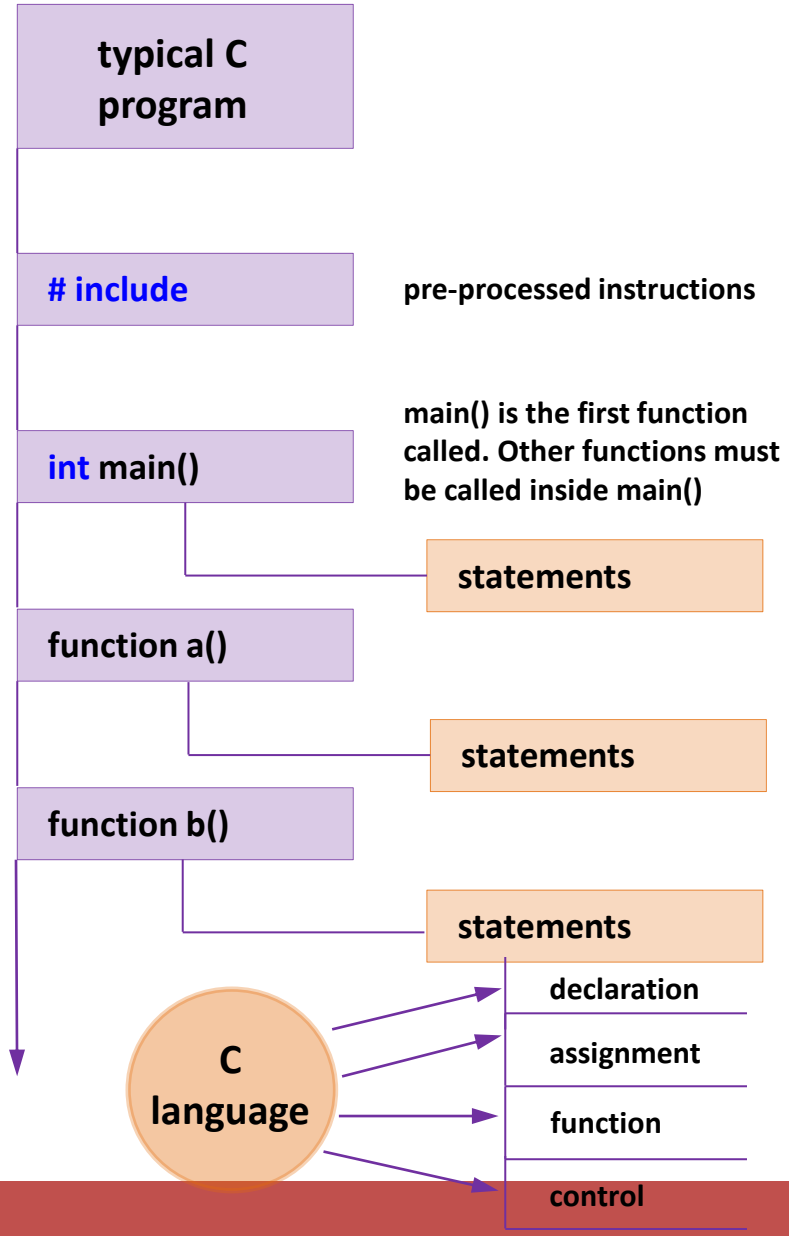
```
# include <stdio.h>
int main()          /* a simple program                */
{                   /* start of the function main()    */
    int num;        /* declare a variable called name  */
    num = 1;        /* assign a value to num           */

    /* use the printf() function  */
    printf("I am a simple computer.\n");
    printf("My favorite number is %d.\n", num);

   return 0;
}                   /* end of the function main()      */
```

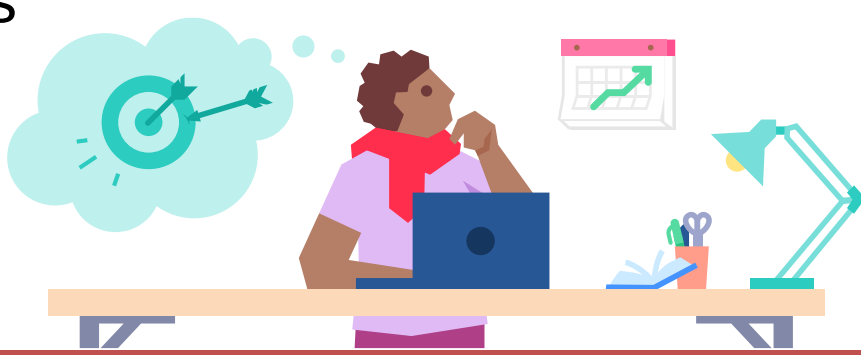
Output

I am a simple computer.
My favorite number is 1.



Lecture Objectives

- Course Overview ✓
- To explain the merits of C language ✓
- To explain where C language is used ✓
- To explain the usage steps of C language ✓
- To explain the operation of the compiler and linker ✓
- To introduce the C language Standards
- Simple example



A Simple Program – contd.

- Include directive and header files
  - # include<stdio.h>
  - stdio.h: Stands for *standard input/output header*
  - As if you typed the entire contents of the file “stdio.h” into your file
  - The file “stdio.h” includes, for instance, the function “printf()”

- The main() function
  - int main()
  - A program must include a main() function
  - The program starts execution with the function main()
  - int indicates the function main() is of type *integer*
    - The function must return an integer value
  - For example, “void main()” does not return a value

A Simple Program – contd.

- Comments
  - Using comments makes your code more readable by others
  - Multi-line comments
    - Comments should be enclosed in the /* you comment */

```
/* This is a comment     */

int num;    /* this is a comment as well */

/* This is also
a
comment  */
```

  - Single-line comments
    - A comment line should start with // your comment

```
// This is a comment

int num;    // this is another comment
```

A Simple Program – contd.

- Braces, bodies, and blocks
  - Braces mark the beginning and the end of the body of a function

```
{
    . . .
}
```

  - Braces can also gather statements within a function into a *block*
    - Examples: if, while, … statements (later, we will study them)

- Declarations
  - int num;
  - This statement indicates two things:
    - A variable called num
    - The variable num is integer, i.e., it does not have a decimal point or a fractional part
  - "int" is a *C reserved keyword* (we will discover the full list later on)

A Simple Program – contd.

- Declarations – contd.
  - Variable name:
    - Use meaningful names for variables,
      - E.g., student_count, empl_age, student_gpa, etc.
    - You can use UPPERCASE letters, lowercase letters, digits, and the underscore (_)
      - However, a variable's name must start with a letter or the underscore
    - Valid variable's names
      - Num, evaluation, error_count, _id, num2, num2power
    - Invalid variable's names
      - 2num, tax rate, $usd, Z^2, don't

A Simple Program – contd.

- Four reasons to declare variables
  - Putting all variables in the same place makes the program easily readable.
  - Thinking of which variable to declare encourages you to make a plan for your program.
  - Declaring variables helps prevent a hard-to-find bugs – that of the misspelled variable name.
  - Your C program will not compile if you don't declare your used variables.
- Assignment
  - `int num, randn;` // declaration of num, randn
  - `num = 1;` // assigning a value to num
  - `randn = num;`

A Simple Program – contd.

- The printf() function
  - printf("argument") prints the *argument* on the screen
  - printf is the function's name and what is between the parentheses is the argument
  - Examples:

```
printf("I am a simple computer.\n");
printf("My favorite food is 해물\n");
```

    - The first line prints I am a simple computer on the screen
    - The second line prints My favorite food is 해물 on the screen

  - Another example

```
...
int num;     // declaration
num = 5;     // assignment

printf("My favorite number is %d\n", num);
...
```

Output

My favorite number is 5

A Simple Program – contd.

- The return statement
  - return 0;
    - C functions which return values do so with a return statement

Lecture Summary

- Course Overview
- To explain the merits of C language
- To explain where C language is used
- To explain the usage steps of C language
- To explain the operation of the compiler and linker
- To introduce the C language Standards
- Simple example