# Fundamentals of C Programming (Part 3)

By Noor Hazlini binti Borhan

UNIMASMOOC
UNIVERSITI MALAYSIA SARAWAK

# Overview

- ✓Programming Styles.
- ✓Programming Errors.

# Learning Outcomes

On completion of this unit, you should be able to:

- ✓Apply the programming styles.
- ✓Identify the types of programming errors.

# Programming Styles

- It is a collection of conventions, rules, and techniques that enable us to write programs in an elegant yet simple and efficient manner.
- Techniques of style are highly personal and subjective and are mostly a matter of common sense.
- In general, it is easier to understand and maintain such a program, even for someone other than the original developer.

- We are going to discuss the following styles:

| | | |
|---|---|---|
| Using Comments | Variable Name | Naming Style |
| Indentation and Code Format | Clarity | Whitespace |

## Comments

Ideally, a program serves two purposes:

1. Presents the computer with a set of instructions; and
2. Provides the programmer with a clear, easy-to-read description of what the programmer does.

```
/*Program Name:Student Details Program
Author Name:Noor Hazlini
Descriptions:This is a simple C program. In this program, some basic
components of C such as
constants,variables,scanf(),format specifiers and data types will be
applied.*/
```

- Heading
  - The first comment should contain the name of the program. Also include a short description of what it does. You may have the most amazing program but it is useless if no one knows what it does.
- Author
  - You've gone to a lot of trouble to create this program. Take credit for it. Also, if someone else must later modify the program, he or she can come to you for information and help.
- Purpose
  - Why did you write this program? What does it do?

- Usage
  In this section give a short explanation of how to run the program. In an ideal world, every program comes with a set of documents describing how to use it.
- References
  Creative copying is a legitimate form of programming (if you don't break the copyright laws in the process). In the real world, it doesn't matter how you get a working program, as long as you get it; but, give credit where credit is due. In this section you should reference the original author of any work you copied.
- Restrictions
  List any limits or restrictions that apply to the program, such as: *The data file must be correctly formatted; the program does not check for input errors.*
- Revision history
  This section contains a list indicating who modified the program and when and what changes have been made

## Variable Name

A variable is a place in the computer's memory for storing a value. Names can be any length and should be chosen so their meaning is clear.

The following declaration tells C that you are going to use three integer (int) variables named h, i, j:

Avoid using abbreviations

```
int h, i, j;
```

Consider another declaration:

```
int id;
int age;
char name;
```

## Indentation and Code Format

- To make programs easier to understand, most programmers indent their programs.
- The general rule for a C program is to indent one level for each new block or conditional.
- There are two styles of indentation.
- Use any of the two, but make sure you are consistent in using the style that you choose.

- **1st Style: braces are put on the same line as the statements**

```
/*Program Name: Student Details Program
Author Name:Noor Hazlini
Descriptions: This is a simple C program. In this program we will
apply some basic components in C
such as pre-processor directives, escape characters,
comments,main(),reserved word and printf().*/

#include<stdio.h>       //pre-processor directives

int main() {                //main() function

        printf("Welcome to C programming course!!\n");
        printf("This program is to save student details");
return 0;
}
```

- **2nd Style: puts the curly braces on lines by themselves**

```
/*Program Name: Student Details Program
Author Name:Noor Hazlini
Descriptions: This is a simple C program. In this program we will apply some
basic components in C
such as pre-processor directives, escape characters, comments,main(),reserved
word and printf().*/

#include<stdio.h>       //pre-processor directives

int main()                  //main() function
{
        printf("Welcome to C programming course!!\n");
        printf("This program is to save student details");
return 0;

}
```

## Clarity

- A program should read like a technical paper.
- It should be organized into sections and paragraphs. Procedures form a natural section boundary.
- You should organize your code into paragraphs.
- It is a good idea to begin a paragraph with a topic sentence comment and separate it from other paragraphs by a blank line.

```
//poor programming practice

temporary = box_x1;
box_x1 = box_x2;
box_x2 = temporary;temporary = box_y1;
box_y1 = box_y2;
box_y2 =temporary;
```

A better programming practice:

```
//To swap the two corners x & y

//Swap x coordinate
temporary = box_x1;
box_x1 = box_x2;
box_x2 = temporary;

//Swap y coordinate
temporary = box_y1;
box_y1 = box_y2;
box_y2 =temporary;
```

# Whitespace

- The compiler ignores whitespace almost completely.
- Whitespace is defined as spaces, carriage returns, linefeeds, tabs, vertical tabs, and formfeeds.
- These characters are invisible to the compiler. You can put several statements on one line, separated by any number of spaces or tabs, or you can run a statement over two or more lines.

```
/*Program Name: Student Details Program
Author Name:Noor Hazlini
Descriptions: This is a simple C program. In this program we will apply
some basic components in C
such as pre-processor directives, escape characters,
comments,main(),reserved word and printf().*/

#include<stdio.h>       //pre-processor directives

int main()                  //main() function
{
        printf("Welcome    to C      programming course!!\n");
        printf("This program is to save student details");
return 0;
}
```

Bad example (whitespace)

UNIMASMOOC
UNIVERSITI MALAYSIA SARAWAK

# Programming Errors

• Another challenge that awaits is program debugging.

**Debugging** is defined as the process of finding and correcting errors in computer programs.

• No matter how careful you are as a programmer, most programs you write will contain errors. Either they won't compile or they won't execute properly.

•This situation is something that happens very frequently to every programmer. You should take program debugging as a challenge, develop your debugging skills, and enjoy the process.

There are **three** types of programming errors:

- Syntax errors
- Logic errors
- Run-time errors

# Syntax Errors

- Is violation of syntax rule, which define how the elements of a programming language must be written.
- They occur during the implementation phase and are detected by the compiler during the compilation process.
- Another name for syntax error is **compilation error**.
- Ex: Missing semicolon, writing keyword in upper case, writing variable declaration etc.

```c
/*Program Name:Student Details Program
Author Name:Noor Hazlini
Descriptions:This is a simple C program. In this program, some basic components of C such as
constants,variables,scanf(),format specifiers and data types will be applied.*/
#include <stdio.h>                                          //pre-processor directives
#define YEAR 2016                                           // define constant declarations
int main()                                                  //main()
{                                                           //open curly brace
    int student_id,year_born,student_age;                  //variable declarations
    const char letter = 'A';                               //char constant

    printf("Welcome to C programming Course!!\n");                  //printf() statement
    printf("The purpose of this program is to save the student details\n")

    printf("Enter your student ID:\n");
    scanf("%d",&student_id);                               //scanf() with format specifiers

    printf("Enter your year of born:");
    scanf("%d",&year_born);

    student_age = YEAR - year_born;

    printf("Your student ID is %d and your age is %d\n",student_id,student_age);  //printf() with %d format specifiers
    printf("value of letter : %c \n", letter );
    return 0;
}                                       //close curly brace
```

Syntax error- missing semicolon (;)

# Logic Errors

- Occur during the analysis, design, and implementation phases.
- We may choose an incorrect method of solution for the problem to be solved, mistakes in translating an algorithm into a program or design erroneous data for the program.
- The compiler cannot detect logical errors.
- A program with logical errors is compiled (translated) and run successfully but it does not give correct result.

Example:

- The sequence of instructions used in a program may be incorrect
- The mathematical formulas used in program instructions may be incorrect, etc.

# Run time Errors

- Are detected by the computer while the program is being executed.

Example:

- They are caused by program instructions that require the computer to do something illegal, such as attempting to store inappropriate data or divide a number by zero
- Trying to open a file which is not created.
- Lack of free memory space.

# Debug

- To fix the program error, programmer will do debugging task by using debugger
- Debugging process can be very frustrating especially involving huge amount of coding
- Good to have good programming style and documentation in coding!

# Debugger

- is a computer program that is used to test and debug other programs.
- Usually comes together with compilers
- Debugger will help programmer by provide list of error in compiling the program
- Still depends on programmers to solve the problem!!

- Debugging technique
  - Probe
    - Put an external variable inside the program and try to force the program to output result in the middle.
    - Good to trace modular logic error or data error
  - Trace
    - Back-tracking or front-tracking
    - Go step-by-step (or line-by-line) to detect error.
    - Good in tracking logic error.