Lecture 1.1

1. Computer and Computer Software – Brief
2. Computer Program Creation & Program Development Process – Brief
3. History of C (Computer Programming) Language
4. Components of a C Program – Simple Example
5. Data & Data Types in Computer Programming
6. Data Output with `printf()`
7. Constant Values in C

---

1. Computer and Computer Software – Brief

Computers have offered a great deal of time saving computations and applications from the 1950s to the current time. Computers have changed from a 30 tons and occupying 1500 square feet space to a very small size desktop box, laptop, tablet, or mobile/handheld devices.

The speed of today's computers is in no comparison to any of the past and it is getting to be more powerful every day.

- Different computer architectures and chipsets are being used or developed for the next generation of computers.
- All peripheral components are getting better and cheaper so that computers are making their ways to the general public more and more every day.
- In many cases, computers become the necessities and nothing would go on or even possible without the assistance from computers.

Read with your own measure – http://en.wikipedia.org/wiki/Computer

However, without software to control/operate a given computer, it would just sit and become useless. So, when referring to our computer(s), we usually meant a system with an operating system and software installed to provide the operation and required functionality of that computer.

Computer software must be developed and maintained. A piece of software may be very simple such as a utility or very sophisticated such as a complete operating system. No matter what the software was intended for, it needs to be developed and implemented.

Definitely, computer programs and computer programming are the large parts of the solutions for many modern systems and applications.

- To create a computer program, one may need to have tools and to learn proper techniques.
- A computer program can be written in a specific computer programming language to take advantage of each language in handling data and designing the solution.

Read with your own measure – http://en.wikipedia.org/wiki/Computer_software

Where and how would the development of a piece of computer program or software start?

Let's briefly look at the tools and general process of computer programming development in the next section.

2. Computer Program Creation & Program Development Process – Brief

What is computer programming?

> In general, it is said that computer programming is the technique that uses computer to manipulate some given data values in order to produce the desired results. The manipulation process will follow some specific sets of instructions or steps.

A computer program will then provide instructions for a computer to manipulate the data and produce the output.

In general, the process of developing a computer program would require one to

1. Get/specify problem statement
2. Analyze problem
3. Design/develop algorithms
4. Implement algorithms
5. Test and debug program
6. Maintain and update program

Figures 1 and 2 depict the process. Let's explain and elaborate on these steps.

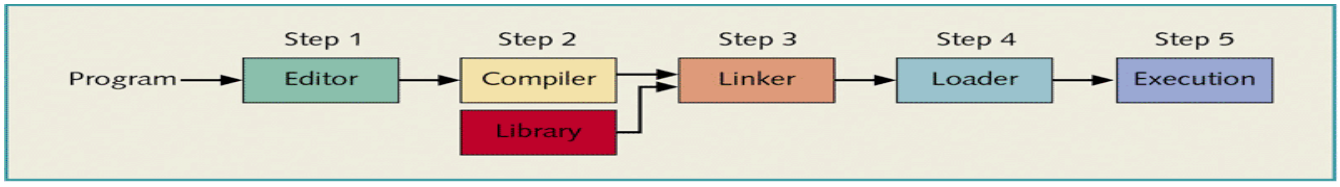

Figure 1 Creating and processing a high-level language program

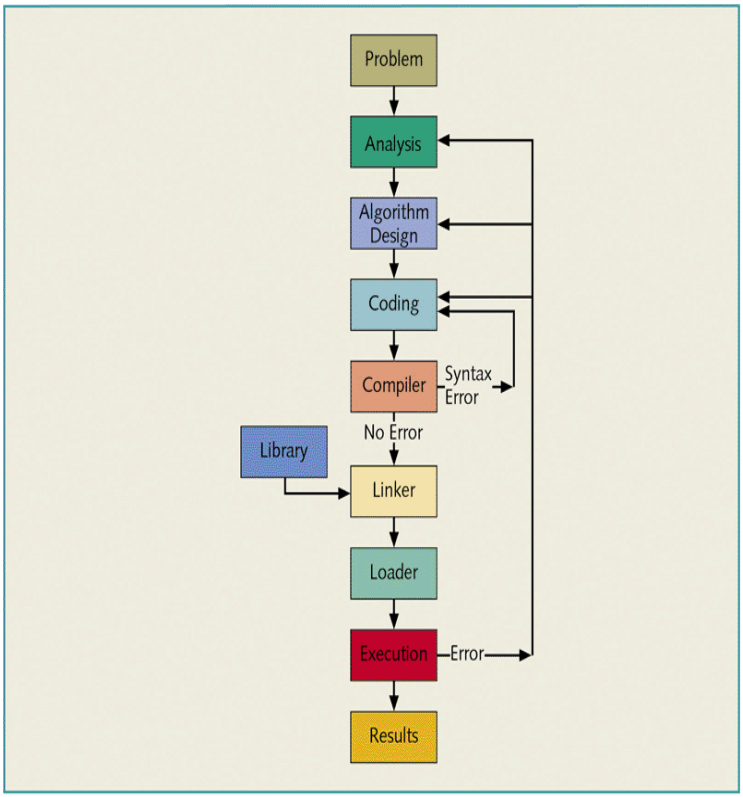

Figure 2 Development process – Problem analysis-design-coding-execution cycle

2.1 Get/Specify Problem Statement

This requires the problem to be stated clearly and formed in an unambiguous manner. Doing that would provide a definitive understanding of what is required for the solution and also to eliminate confusion.

2.2 Analyze Problem

Analyzing the problem statement would break things into:

a) What the input is/are, and

b) What the output is/are, and

c) What are, if any, additional requirements or constraints to achieving the solution?

2.3 Design/Develop Algorithms

One may need to follow a sequence or list of steps to produce the solution. This list is combined into a so-called **algorithm**.

That means HOW-TO do things or HOW-TO process things. One must follow through with each of the steps and verify that the algorithm would indeed provide the desired outcome.

2.4 Implementation

Using a specific computer language, one can just translate the steps into program statements. Group them in methods/functions and use these methods/functions as necessary.

2.5 Test & Debug Program

A major portion of effort in developing computer code is spent in testing and debugging. The goal is to have the program working. There may be errors that need to be corrected; these can either be syntax errors or logic errors. These errors must be fixed by looping back to Step 4.

Generally, the logical errors are called bugs. Debugging really means trying to get rid of these logical errors.

2.6 Maintain & Update

After getting rid of all bugs and verifying the results, the program will be either installed or used. From time to time, this program may need to be relocated or reinstalled; it may need to be rewritten (updated) to accommodate new conditions or systems.

2.7 Examples

Example 1

```
/**
 *Program Name:  cis26L01201.c
 *Written By:    T. Nguyen
 *Discussion:    C Elements
 */
#include <stdio.h>

int main( void ) {
  printf( "Welcome to C World!\n" );

  return 0;
}
```

OUTPUT

```
Welcome to C World!
```

Example 2

```
/**
 *Program Name:  cis26L01202.cpp
 *Written By:    T. Nguyen
 *Discussion:    C++ Elements
 */
#include <iostream>
using namespace std;

int main( void ) {
  cout << "Welcome to C++ World!” << endl;

  return 0;
}
```

OUTPUT

```
Welcome to C++ World!
```

Example 3

```
/**
 *Program Name:  HelloWorldApp.java
 *Written By:    T. Nguyen
 *Discussion:    Java Elements

 *               The first Java program
 */

class HelloWorldApp {
  public static void main( String[] args ) { // Required
    System.out.println( "Hello World!" );    // Using System packages

    return;
  }
}
```

OUTPUT

```
Hello World!
```

What are so different with the above programs?.

3. History of C (Computer Programming) Language – Brief

In 1972 while collaborated on designing the Unix operating system, Dennis Ritchie of Bell Labs created C as a computer programming language. By the late 1970s, C had evolved and become more useful and powerful; it was known then as “traditional C”.

In 1978, Kernighan and Ritchie wrote the first edition of the book titled “The C Programming Language” and drew wide attention to the language. This book is still being used today as required text and reference in many universities and industries.

- In 1989, the ANSI C (C89) standard was approved and the ISO (C90) was also approved later in 1990. These two standards are essentially the same.
- The ANSI C89 was later updated in 1999, thus called C99, which preserves the essential nature of C.

- Many current compilers may not fully implement all C99 features (which may not be of our concerns yet).

It is important to repeat again here that

- ✦ The goal of computer programming is to work with some (data) model so that a design and then its implementation will produce a computer program that will operate on the data model.
- ✦ A computer program may work with one or more platforms (operating systems, embedded systems, etc) to produce the desired output and results.

Why C?

- C is a structural/procedural language that is powerful and flexible.
- C is also efficient and portable. It has been one of the main tools for embedded systems and applications development (Linux).

For our purpose in this class, of course, the computer programs and exercises will be written in C language (or just C).

4. Components of a C Program – Simple Example

Let us look at a C program below.

Example

```
/**
 *Program Name:  cis26L01204.c
 *Written By:    T. Nguyen
 *Discussion:    C Elements
 */
#include <stdio.h>

int main( void ) {
  printf( "Welcome to C World Again!\n" );

  return 0;
}
```

OUTPUT

```
Welcome to C World Again!
```

The above C program has

- Two **statements**,
- Several built-in **keywords**, and
- A predefined **function** (or method). The function (method) `printf()` was defined in `stdio.h`, which is the header file to be loaded and preprocessed by the **`#include`** directive.

What are the components that one can use to form a C **statement** and eventually produce a correct program?

A C statement may be formed by using a combination of

- Keywords (C vocabularies),

- User's defined identifiers such as variables and functions/methods, and
- Constants.

4.1 Keywords

Keywords (vocabularies) are built-in terms (or words) that reserved for the language. They are defined to be used for specific purposes; thus they must be used as they were defined.

For examples, the `include` keyword is used to signal the compiler to preprocess the specified file (such as, `stdio.h`) prior to taking in the next segment of the code. The `int` keyword modifies the returned value of the method `main()` and forces it to be of type `int`.

4.2 Identifiers

Identifiers are the names created in the code by the programmer. An identifier can be used as variable name, function name, etc. To be a valid identifier, it must start with a character and may have any combination of alphanumeric characters and some special characters, such as _ or $.

It is recommended to only use alphanumeric characters for identifiers. In addition, an identifier should have some relevant and meaningful interpretation for the tasks that are being written for.

Note that C is a **case-sensitive** language. That means the two identifiers of "**`yours`**" and "**`Yours`**" are two different specifications or references in the code.

4.3 Constants

Briefly, constants are values that cannot be changed during the execution of the program.

4.4 Executable Statements

An executable statement is a statement that will get compiled and/or may get linked with other modules or functions/methods.

In C, an executable statement is terminated with a semi-colon (**`;`**).

For examples,

```
printf( "Welcome to C World Again!\n" );
```

or,

```
return 0;
```

4.5 Comments

C program may also have non-executable statements (that means the compiler will ignore them during the compilation). They are called comments and used to provide explanations and reminders along the code.

A comment may start and close with a pair of "**`/*`**" and "**`*/`**". This style of commenting can span (have) several lines as long as these lines are between the given comment pair. Note that this comment style cannot be nested.

For examples,

Valid Comment

```
/**
 *Program Name:  cis26L01204a.c
 *Written By:    T. Nguyen
 *Discussion:    C Elements
```

```
 */
```

Invalid Comment

```
/**
 *Program Name:  cis26L01204b.c
 *Written By:    T. Nguyen
 *Discussion:    C Elements

 /*Laney College*/                    ← ERROR

 */
```

Some compilers (**C99** implementation) also provide a second style of commenting. This style uses a double forward slash pair of **//** to comment out the remaining text of the current line.

For examples,

```
//Laney College
```

Or,

```
printf( "Welcome to C World again!\n" ); //Using printf()
```

4.6 Braces and Blocks

We use the pairs of curly braces (that means '{' and '}' symbols) to specify (or delimit) functions and blocks. We will have more discussion on this later.

Obviously, the above simple program produces the result, which is the output being displayed on screen. More specifically, the data (which is the phrase "`Welcome to C World again!\n`") is being processed by the function (`printf()`, which is predefined).

We may be asking a question as:

- What can a function (or program) operate or work with? Or
- What kinds of data (data types) can we have and use in a computer program?

5. Data & Data Types in Computer Programming

In the world of computers and computer architecture, data are represented by sequences of **0**'s (zeros) and **1**'s (ones). These are called bits; these bits will be arranged into some bit patterns for values. A bit is the smallest data unit used in computer.

> **A byte is a group of 8 bits and it is actually used to represent more meaningful and workable data while developing computer code.**

In the world of computer programming, a given piece of data is treated as either a number (numeric value) or some textual description (string value).

Thus, there are two general kinds of data:

(1) number, and

(2) string.

In programming, if a known value is mentioned then it is generally understood as a constant.

String data would require a bit more of explanation more than number (or numeric value). So let us look at just numeric values for now and will come back to string later.

Numeric values are further grouped into integers or floating-points. In C, there are several basic built-in numeric data types for integers and floating-points. They are indicated and coded with the following specifications:

```
char      float         void
short     double
int
long
```

5.1 Integer Data Types & Variables

Integer data are values that do not have decimal portion. For examples, `5, -5, 1000, -1000`. Their data type may be specified or assigned to a variable with one of the following keywords: `short, int, long`.

For examples,

```
short sVar;
int iCounter;
long lSum;

sVar = 10;
iCounter = 0;
lSum = 0;
```

Character may also be considered as integer type (i.e., `char`) and being implied with the integer equivalent value indicated in the ASCII Character Table. For examples,

```
char cValue1 = 'A';      /* 'A' := 65 */
char cValue2 = 'Z';      /* 'Z' := 90 */
char cValue3 = 'a';      /* 'a' := 97 */
char cValue4 = 'z';      /* 'z' := 122 */
char cValue5 = '0';      /* '0' := 48 */
char cValue6 = '9';      /* '9' := 57 */
```

5.2 Floating-Point Data & Variables

Floating-point data are numerical values that have the decimal parts. For examples, `5.5, -5.4, 1000.0, -1000.345`. Their type may be specified or assigned to a variable with one of the following keywords: `float` and `double`.

For examples,

```
float fVar1 = 5.5;
double dVar2 = -5.4;
```

6. Data Output with `printf()`

To output data on screen, one can use **`printf()`** function. There are many different ways for data (also called arguments/parameters) being passed to this function.

Let's look at some of the calls to `prinf()`.

Example

```
/**
 *Program Name:  cis26L01205.c
 *Written By:    T. T. Nguyen
 *Discussion:    printf()
 */
```

```c
#include <stdio.h>

int main( void ) {
  printf( "A character %c -- The last one.\n", 'Z' );

  printf( "This is C World!\n" );

  printf( "An integer value %d -- The first one.\n", 5 );

  printf( "A floating-point value %f -- The first one.\n", 5.0 );

  return 0;
}
```

OUTPUT

```
A character Z -- The last one.
This is C World!
An integer value 5 -- The first one.
A floating-point value 5.000000 -- The first one.
```

As shown, `printf()` can handle values of different types by using the **format specifiers**, such as **`d`** and **`f`**.

7. Constant Values in C

Constants are values that cannot be changed during the execution of the program. Constants may have types (as being implied) when these values were coded into the code. There are three types of constants: **integer**, **floating-point**, and **string**.

Let's consider them next.

7.1 Integer Constants

An integer constant, as coded with a series of digits, has the signed integer (**`int`**) type, or signed long integer (**`long`**) type if the number is large. The **default type** is **`int`**.

One can override the defaults by using the suffixes **`u`** (or **`U`**) and **`l`** (or **`L`**). **`U`** and **`L`** are preferred to avoid any misunderstanding or confusion.

The following examples show the types of the constants.

Literal	Value	Type
123	123	int
-123	-123	int
-123L	-123	long int
123LU	123	unsigned long int

7.2 Floating-Point Constants

The **default type** for any floating-point constant is **`double`**. One can use the format specifiers of **`l`** (**`L`**) and **`f`** (**`F`**) to indicate whether it is a **`long double`** or a **`float`**.

Some examples are given below.

Literal	Value	Type
123.	123.0	double
-0.123	-0.123	double
123F	-123.0	float
123.0L	123	long double

In addition to these numeric constants, C has two other constants: character and string.

7.3 Character Constants

Character constants are enclosed between two single quotes. Normally, character constants would have only one character quoted. These characters are mostly called **printable characters** that means one can print these characters on paper or see them displayed on screen.

For examples,

'a'	A character lowercase a
'A'	A character uppercase A
'o'	A character lowercase o
'1'	A character digit 1
'9'	A character digit 9
'0'	A character digit 0
''	A character of nothing; it is also called the null or NULL character

Note that the digits enclosed in the single quotes should be understood as characters and not values.

There are other singly quoted characters that have a backslash (\) added in front of characters (only one character after the backslash). These are called **nonprintable characters**; their values or effects would be either seen or heard when the system tries to display them.

For examples,

'\0'	Null character	'\a'	Bell
'\b'	Backspace	'\t'	Horizontal tab
'\n'	Newline	'\''	Single quote
'\"'	Double quote	'\\'	Backslash

7.4 String Constants

A string constant is a sequence of zero or more characters enclosed in a pair of double quotes.

For examples,

""	Null string
"a"	A string of one character of lowercase a
"CIS26"	A five character string
"CIS26\n"	A six character string