Lecture 2.1

Topics

1. Computer and Computer Software – Brief
2. Computer Program Creation & Development Process – Brief
3. Object Orientation – Introduction
4. C++ Program and Components – Introduction/Review
5. C++ Basic Data Types – Introduction/Review
6. Tools For Class – Discussed as Needed

---

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 operations and required functionalities for 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.

And as time goes on, the software may need to be updated or upgraded to fix bugs or to take advantage of new components (existing hardware and/or software).

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

Thus as demanding and demanded,

- To create a computer program, one may need to have tools and to learn proper techniques or ways when using these tools.
- 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

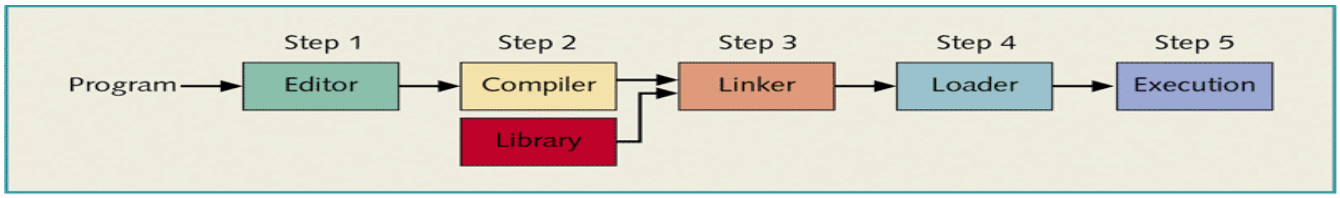

Figure 1 Creating and processing a high-level language program

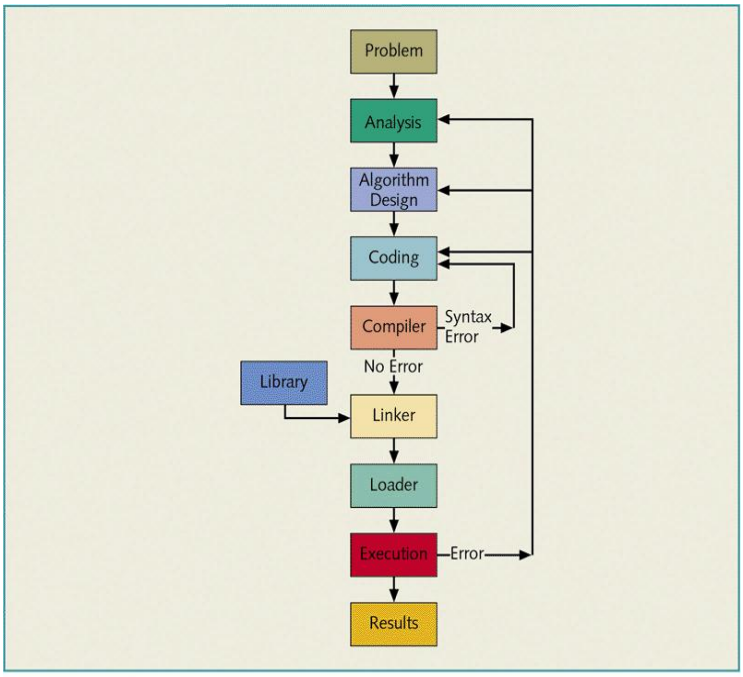

Figure 2 Development process – Problem analysis-design-coding-execution cycle

In general, it is said that computer programming is the technique that uses computer to manipulate data, which are provided by the user in order to produce the required output in the most efficient way.

The manipulation process will follow some specific sets of instructions or steps, which are then language-specific while being implemented.

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

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

Let's touch on some of these steps.

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 information.

2.2 Analyze Problem

Analyzing the problem statement would break things into: (a) input, (b) output, and (c) 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. 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 first goal is to have the program working. There may be errors that need to be corrected; they can either be syntax errors or logic errors. They must be fixed by looping back to step 4.

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: cis25L0211.c
 * Written By:   T. Nguyen
 * Discussion:   C Elements
 */
#include <stdio.h>

int main() { // Required
  printf("Welcome to C World!\n"); // Defined in stdio.h

  return 0;
}
```

OUTPUT

```
Welcome to C World!
```

Example 2

```
/**
 * Program Name: cis25L0211.cpp
 * Written By:   T. Nguyen
 * Discussion:   C++ Elements
 */
#include <iostream>
using namespace std;

int main() { // Required
  cout << "Welcome to C++ World!" << endl; // Defined and scoped
                                           // within iostream and
                                           // std

  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? Let's go through them here briefly (for now).

3. Object Orientation – Introduction

3.1 Objects – Definitions

Generally, Object-Orientation can be of any consideration, abstraction, construction, and manipulation that deliver to some object(s) as part of the solution within computer science discussions and problems.

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

So, what is an OBJECT?

> An object is a conceptually real or abstracted thing that one can use to store data and functions/methods to manipulate data.

Read with your own measure – http://en.wikipedia.org/wiki/Object_(computer_science)

Toward the programming end, an object

- Is identifiable (e.g., variables)
- Has some lifetime and space (i.e., scope and memory storage)

- Provides data & operations that can be used to manipulate & interface with its surrounding environment

Note that

- An **Object Type** is a description for the creation of some objects,
- Whereas an **object** is an **instance** or **sample** of a given Object Type.

Example 1

It is fair to say that "**`Car`**" is an object type and "**`Airplane`**" is another object type. Consider the following:

A Buick is a passenger car and a 300Z is a sport car. Thus, Buick and 300Z are instances of (general) car. A Boeing747 is a jumbo plane and an F15 is a fighter jet. Thus, Boeing747 and F15 are instances of (general) airplane.

Functions/Methods:

Each object will have some data (or attributes) and functions/methods. Methods specify the way in which data may be manipulated. Methods of a given object operate only on the data belonging to this object and no other objects.

Encapsulation:

To protect data integrity, object may hide its data from the outside world. That means an object will protect its own internal characteristics.

Request (Message):

To use an object, a request must be sent to this object. The request will invoke some methods to perform the desired operations. The request contains object's identity, specified operation, and necessary parameters.

3.2 OO Analysis (OOA), OO Design (OOD), OO Programming (OOP)

OOA:

OOA is a method (technique) of analysis that examines and produces output (description/solution) for a problem constrained by the consideration and use of **objects**.

The given input will be examined in terms of what (e.g., what do we have?), why (e.g., why do we must work with this set of values?), and how (e.g., how do we get the data?) for data and logic validations. Then the inter-relationship between the problem domain and solution domain will be analyzed using objects as the building blocks.

OOD:

OOD provides a method of designing or constructing the building blocks for solution domain based on object behaviors. The designers must then decide **how** data and methods (of course, how do we create each function and its operations) should be packaged to form objects.

OOP:

OOP is the method of implementation (coding) in which the objects were created and used to perform the required tasks using the results obtained through OOA and OOD.

3.3 Object-Oriented Programming (OOP) Paradigm

Putting together the analysis and design into the programming phase we are constrained to follow an OOP paradigm. This programming paradigm has several essential elements: (1) Abstraction, (2) Encapsulation, (3) Modularity, (4) Hierarchy, (5) Concurrency, and (6) Persistence.

Abstraction:

An abstraction is a description for a set of characteristics (or attributes) of an object to make it unique from other objects.

Encapsulation:

Encapsulation is the process of hiding information (data and functions) belonged to an abstraction (object) from the external sources. Object encapsulation preserves its data integrity and structure.

Modularity:

System modularity refers to a decomposition of system into sets of well-defined and cohesive modules. In OOP, a module can generally be thought as a group of objects so that when put all together will interact and make up the system.

Hierarchy:

Hierarchy is a ranking of abstractions or, perhaps, of object types. Hierarchy induces inheritance; And in OOP, inheritance plays an important role. Using inheritance, more objects can be created from a few with ease.

In object orientation, a base abstraction (base class) is the one that can be used to create the new abstraction. This new abstraction is called the derived abstraction (derived **`class`**). The derived abstraction will inherit some or the entire base abstraction plus some of its own.

Concurrence:

In loose terms, concurrency is the property that distinguishes scope of objects within a system. Objects can be invoked concurrently (that is, active at the same time) or by some interactive order (e.g., nested process). These concurred objects will retain their abstraction, encapsulation, and hierarchical properties.

Persistence:

Persistence is the property of an object through which its existence transcends time (i.e., the object continues to exist after its creator ceases to exist) and/or space (that means object can be relocated to different places in the system, e.g., location in memory space).

In particular, C++ is used in an OOP paradigm to allow the creation of objects through a so-called **`class`** (a mean to create an abstraction). Thus, C++ is also called the class-based object-oriented language and will have the following traits:

Encapsulation:

Encapsulation is found in classes (object types) to provide ways for binding (to work with) methods and data together and to prevent outside interference. Methods and data of an object can be kept either public or private (or a third kind: protected).

Private members are known to and accessible only by other members of the abstraction (or internally). Public members can be accessed externally of the abstraction.

Polymorphism:

Polymorphism is the property that allows one name to be used for two or more related but technically different purposes or data.

For examples, the same function name can be used for many different purposes, this is called function overloading; Or different data types can also be used with the same operator, this is called operator overloading.

Inheritance:

Inheritance is the process that allows derived classes to be created from some existing base classes. Inheritance can either be single or multiple. That means the derived class can be obtained either only from a single base class (i.e., single inheritance) or from a group of two or more bases classes (i.e., multiple inheritance).

4. C++ Program and Components – Introduction/Review

A simple C++ program may be of the following form

```
// Program Name: cis25L0212.cpp
// Written by:   T. T. Nguyen
// Discussion:   C++ program with cout
//               (An object from ostream class)

/**
 * Program Name: cis25L0212.cpp
 * Written by:   T. T. Nguyen
 * Discussion:   C++ program with cout
 *               (An object from ostream class)
 */

//#include <iostream.h>     //A preprocessor directive

#include <iostream>
using namespace std;

int main() {
  cout << "Welcome to Object-Orientation Using C++";

  return 0;
}
```

Program Discussion:

Comments — `//`

This is one of the two possible ways a comment can be inserted into a program. Everything after the double forward slash sign to the end of the line will be treated as comment.

The other commenting style is the use of the `/*` and `*/` companion pair. In this style, all characters enclosed between the pair are considered as comment and ignored by the compiler (multi-line comment).

Preprocessor Directive — `#include <iostream>`

The inclusion of `iostream` (or, `iostream.h`) will provide a set of predefined parameters and class definitions so that they can be used in the program. For example, **`cout`** is an object from a class defined in `iostream.h`; its definition can be found in that header file.

Function main() — `int main()`

Every C++ program must have a function `main()`. Since C++ is case sensitive,

`int main()` is not the same as `int Main()`.

In this particular case, `main()` comes with two additional keywords: `int` and `void`.

Keywords are explicitly reserved names that have strict meaning in C++. They cannot be redefined or used in other contexts. Some of these keywords are:

char	class	do	double	else	float
for	if	int	long	private	protected
public	return	short	signed	unsigned	void
while					

In this example, the program does not need any parameter for its execution (i.e., `void`). After completing the execution, a value of 0 will be issued to the system (specifically handled through **`return`** statement). The type of this integer (value of 0) is enforced because of the specified type `int`.

Curly Brackets— `{ }`

The body of `main()` must be enclosed by this pair of braces. Personal style in programming will reflect the use and locations of the braces. It is immaterial to the execution of the program. However, a consistency in placing these braces will help to clarify the reading and presentation of the coding.

Semicolon — ;

Every executable C++ statement must be terminated by a semicolon (`;`).

Object Interaction — `cout`

`cout` is an object of `ostream` class defined in `iostream.h`.

This `ostream` class represents a predefined stream that is automatically linked to the standard output device, which is usually the screen.

What is a stream? A stream represents a flow of individual bytes in or out of a file or device.

The insertion operator **`<<`** is used to send the data, which is a string of characters, to `cout` to operate on.

5. C++ Basic (Data) Types – Introduction/Review

The main goal of writing a computer program is to work on (or process/manipulate) data. Data can be a single number, a sequence of words (text), some combination of numbers and text, etc.

As data being used in a program, the computer must store data in memory. While the program runs and executes the statements (one statement at a time), data will be retrieved and processed – It should be emphasized that each (data) value is stored based on how the programmer specifies through the declaration statements)..

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

5.1 Data Types

C++ has many data types that can be grouped in three categories as follows,

(i) Simple Data Type

(ii) Structured Data Type

(iii) Pointers

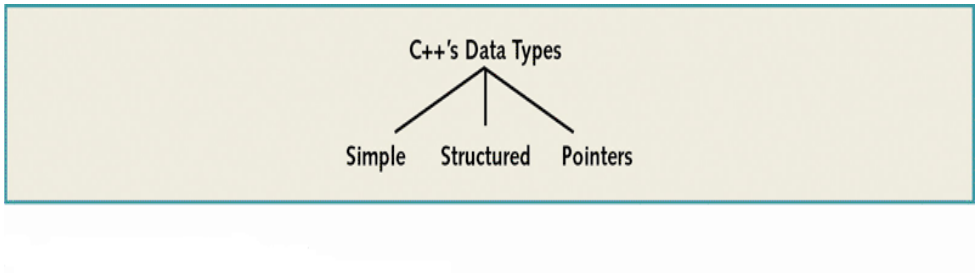

Figure 3 C++'s data types

Structured and pointer data types will be considered in later chapters. For now, we will work with **Simple Data Types** as they are depicted in the following figures.

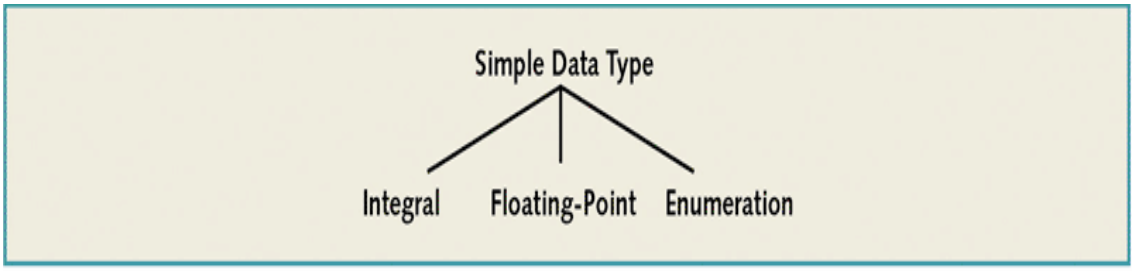

Figure 4 Simple data types

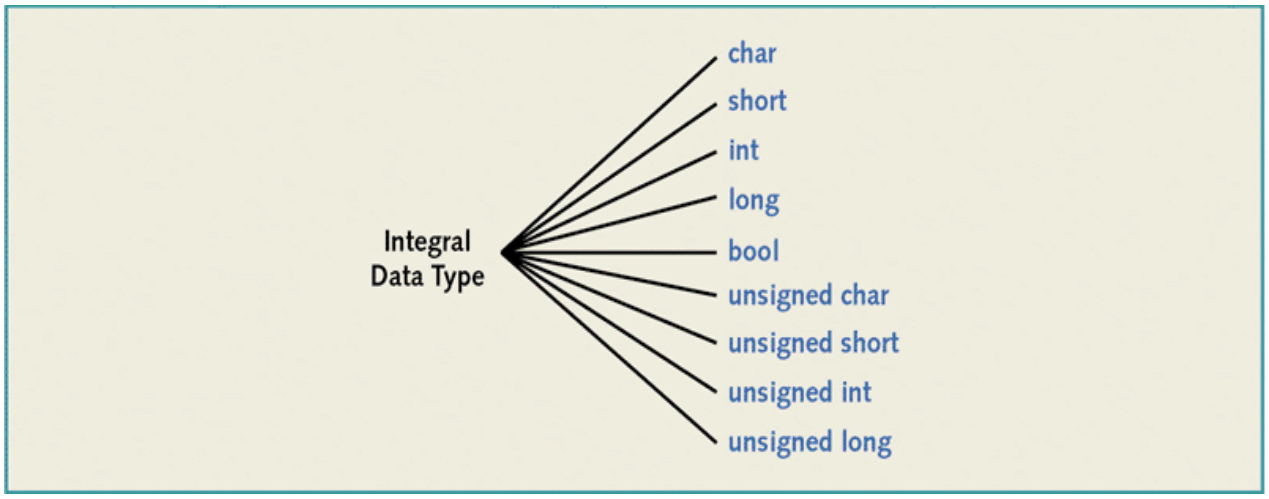

Figure 5 Integral data types

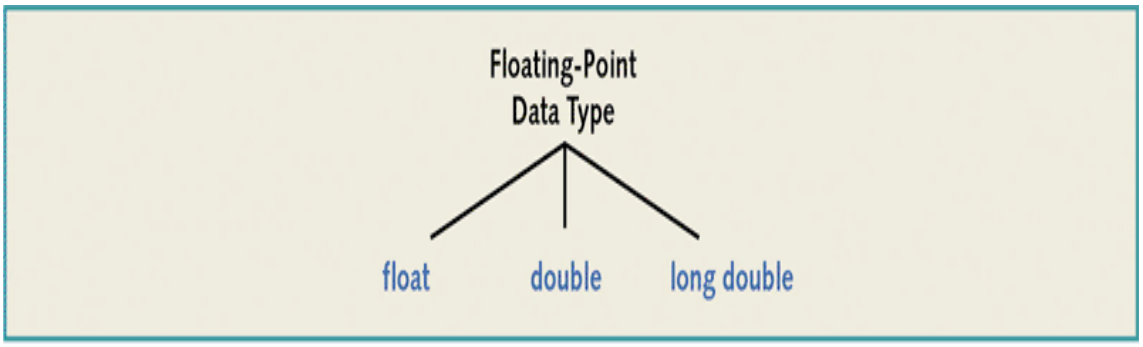

Figure 6 Floating-point data types

In C++, there are several built-in data types for integers and floating points. Some of them are given as follows:

```
char                    float
short                   double
int                     long double
long
unsigned char
unsigned short
unsigned int
unsigned long
bool
```

These types may be modified by using one of several qualifiers (keywords) of `signed`, `unsigned`, `const`, and `volatile`.

6. Tools For Class – Discussed as Needed