Fresher C++

18/06

* Emun: không phải là kiểu int
* Sự khác nhau giữa struct và union:
* Size của struct ít nhất bằng tổng size của các thành phần của struct, size của union bằng size của thành phần có size lớn nhất trong union.
* Tại cùng một thời điểm run-time, có thể truy cập vào tất cả các thành phần của struct. Còn chỉ có thể truy cập 1 thành phần của union.

19/06

* Lambda:
* [ capture-list ] ( params ) mutable(optional) exception attribute -> ret { body } (1)
* [ capture-list ] ( params ) -> ret { body } (2)
* [ capture-list ] ( params ) { body } (3)
* [ capture-list ] { body }
* [=] : truyền value. Biến được capture can read-only. Muốn ghi đè phải dùng mutable

Nhưng chỉ có thể ghi đè trong scope {}

[=]() mutable { a = a + 1}()

* [&]: truyền reference. Có thể ghi đè biết được capture.
* Sự khác nhau giữa reference và mutable:

**#include<iostream>**

int main() {

int a = 0, b = 0;

[a, &b]() mutable { a = 1; b = 1; } ();

std::cout << a << std::endl; *// in 0*

std::cout << b << std::endl; *// in 1*

return 0;

}

* Function Static:
* Điểm đáng lưu ý trong việc sử dụng hàm static trong struct/class là chúng không thể truy xuất đến non-static members trong struct/class đó.
* Chưa cần tạo ra đối tượng đã có thể gọi làm static.
* Mutex, semarphone
* Smart pointer

22/06

* **Move Semantic**
* &&: ký hiệu của rvalue
* Proccess and Thread

02/07

Structural design pattern

* Singleton: chỉ có duy nhất một obj

**Design pattern overview:**

* Advance:
  + Reusability
  + Expandability
  + Communicability
* Structural pattern
  + Defining relationship between objects/classes
  + Called Wrapper classes informally

**Adapter – Intent:**

**Façade – Intent:**

* Provides high level interface easy-to-use
* Hides the complexity of system to client

**Proxy – Intent:**

* Provides a representative for an object
* Virtual proxy:
  + Used when accessing objects having complex structure or contain huge data (image, voice, dataset …)
* Protection proxy
  + Applying some filter on access right for real data
* Remote proxy
  + Accesing remoted object

**Design patterns  
Template method and Visitor**

**Debugger GDB in manual:**

1. Concept of Debugger

* Definition: A **debugger** or **debugging tool** is a computer program that is used to test and debug other programs (the "target" program).
* Feature: Debuggers offer abilities to track what is going on within program executing such as forward step-by-step, breakpoint, tracking variables, reverse debugging.
* Language: Some debuggers operate on a single specific language while others can handle multiple languages transparently
* Popular Debugger: GDB, adb, Visual Studio Debugger, etc.

1. **Applications by GDB capabilities**

* The **GNU Debugger** (**GDB**) is a portable debugger that runs on many Unix-like systems and works for many programming languages, including C, C++, Go, Java, Ada, Objective-C, Free Pascal, Fortran and partially others.
* *“Origin from wiki”*
* Feature:
  + It allows you to inspect what the program is doing at a certain point during execution.
  + Support reverse debugging
  + Interact low level debug as assembly or register layer.
  + Errors like *segmentation faults, bus error* may be easier to find with the help of gdb.
  + Post-mortem core file
  + Remote debug
* IDEs use GDB: WinGDB, Qt Creator, Eclipse, NetBean, CodeLite, Code::Block, etc.

1. **Build a program under GDB**