**CÁC CHỦ ĐỂ SEMINAR CỦA LỚP CNTN**

* (A) Thiết đặt môi trường lập trình trên nền khác Windows (đề nghị Linux), viết code demo, debugging.
  + Sinh viên trình bày tiến trình triển khai một hệ thống mã nguồn mở để có thể viết code được.
  + Trình bày các công cụ dùng để phát triển và debugging.
* (B) Đệ qui, khử đệ qui và bài toán tháp Hà Nội.
  + Sinh viên trình bày về các dạng bài toán giải quyết bằng kỹ thuật đệ qui.
  + Trình bày về điểm mạnh và điểm yếu của kỹ thuật này.
  + Sinh viên cài đặt bài toán tháp Hà Nội làm demo cho topic này (thực hiện đệ qui và khử đệ qui).
* (C) Tổ chức bộ nhớ Stack, Heap và quản lý con trỏ trong C
  + Sinh viên trình bày về vùng nhớ của biến local và biến được cấp phát bởi các hàm malloc, alloc, realloc.
  + Độ lớn của biến được cấp phát động, demo một chương trình cấp phát vùng nhở RẤT lớn.
* (D) Unicode trong C
  + Sinh viên nghiên cứu cách thức lưu trữ các character Unicode ở mức binary.
  + Cho biết có bao nhiêu kiểu Unicode (UTF8, UTF16...) cùng các kiểu biểu diễn của nó ở mức nhị phân.
  + Tham khảo:
    - <http://www.i18nguy.com/unicode/c-unicode.html>
    - <http://linuxgazette.net/147/pfeiffer.html>
* (E) So sánh giữa ANSI C và Microsoft C/C++ (Tham khảo (A))
  + Sinh viên so sánh về mặt syntax của C trên các môi trường khác đối với Microsoft C
  + Viết chương trình trên 2 platform và chỉ ra các điểm khác biệt giữa chúng.
* (F) Biểu diễn vật lý của các kiểu dữ liệu cơ sở trong bộ nhớ
  + Tìm hiểu lý thuyết về bit pattern của các kiểu số nguyên, số thực, bool, pointer, và khái niệm “chuỗi c” (char \*).
  + Dùng kỹ thuật ép kiểu con trỏ để thực nghiệm in chuỗi bit ra, sửa bit để xem ảnh hưởng đến giá trị.
  + Tìm hiểu vấn đề tràn số và tràn bộ đệm (với char \*).
* (G) Quy trình dịch chuơng trình: preproc, compile, link (⇓tham khảo (A))
  + Tìm hiểu lý thuyết và dùng lệnh trên command line để dịch từng bước: preprocessing, compile, link để xem kết quả trung gian.
* (H) Tổ chức chương trình (mã nguồn & mã đích) (⇓tham khảo (G))
  + Tìm hiểu & phân tích 2 phong cách: *monolithic* và *modular*,
  + Về mã nguồn modular: chia ra thành nhiều file .c/.h, lợi ích & vấn đề “include hell”, giải pháp include guards (#ifndef, #pragma)
  + Về mã đích modular: chia thành nhiều thư viện (.lib/.dll), lợi ích & khái niệm “black box implementation”.
* (I) Component based development (⇓tham khảo (H))
  + Bài toán: Một nhóm lập trình viên cần phát triển một ứng dụng. Họ cần phải chia ứng dụng ra làm nhiều module có thể phát triển song song. Làm thế nào để các thành viên có thể cùng lập trình cùng 1 lúc mà không bị chờ đợi lẫn nhau.
  + Giải pháp: Phải thiết kế thống nhất bộ giao diện (header file) cho các  module trước. Sau đó mỗi thành viên có thể cài đặt độc lập với nhau. → Nảy sinh “giao diện yêu cầu”.
  + Ứng dụng: viết 1 game đơn giản (tìm đường trong mê cung) cho các bạn khác plugin vào chơi (người chơi viết một module điều khiển nhân vật đi trong mê cung, dịch ra DLL và bỏ vào là chạy được.).

Các nội dung cần cover:

Biến, kiểu, bộ nhớ, dịch, (rẽ nhánh, vòng lặp,) hàm, module