HƯỚNG DẪN PHONG CÁCH LẬP TRÌNH NGÔN NGỮ

Tài liệu này trình bày về phong cách lập trình cần phải tuân thủ khi làm bài tập trong khóa học

- Cấu trúc của 1 chương trình C
- Cách đặt tên
- Layout
- if, while
- Cách khai báo
- Các hàm
- Structs
- Cách chú thích
- Các lựa chọn khi biên dịch
- Lời cảm ơn

1. Cấu trúc của 1 chương trình

a) Thứ tự trình bày trong 1 chương trình C như sau:

- Chú thích đầu tệp
- Các tệp cần include (#included files)
- #defines
- Local struct typedefs
- Khai báo nguyên mẫu hàm cục bộ (Local prototypes)
- Các biến toàn cục (chỉ dùng khi thực sự cần thiết)
- Hàm main
- Các hàm cục bộ

b) Trong một tệp header (.h), thứ tự trình bày như sau:

- Chú thích đầu tệp
- Các file cần include
- #defines
- struct typedefs
- Các nguyên mẫu hàm
- Các biến toàn cục (chỉ dùng khi thực sự cần thiết)

c) Chú thích đầu tệp nên gồm có:

- / / Tên, Mã sinh viên

- / / Ngày tháng
- / / Mục đích của file này (tóm tắt trong 1 dòng)
- / / Một số giải thích khác nếu cần thiết, bản quyền,….

2. Cách đặt tên

- Tất cả các tên cần phải có ý nghĩa
- Tên các hàm và các biến được đặt theo mẫu varName hoặc var_name
- Hằng số phải viết chữ in hoa, ví dụ LINE_LENGTH
- Các chữ cái đơn như i, j, k được sử dụng trong vòng lặp và chỉ số của mảng.

3. Layout

- Thụt đầu dòng bằng dấu cách trống chứ không dùng tab
- Thụt đầu dòng 2, 3, hoặc 4 dấu cách trống (phải thống nhất dùng hoặc 2, hoặc 3, hoặc 4 trong tất cả các file)
- Phải dùng thụt đầu dòng đề phân biệt giữa các khối lệnh và mức lệnh, không chỉ nên dựa vào { }
- Có thể thêm 1 dấu cách trống vào trước hoặc sau dấu (trong danh sách tham số của hàm.
- Đặt dấu {} như dưới đây:

```
int eatFood( int age ){
     Printf( "Yummy" );
}
```

- 1 dòng không quá 72 ký tự
- Không được có cách trống trước dấu phẩy trong danh sách tham số, ít nhất là 1 cách trống sau dấu phẩy trong danh sách tham số.
- Chỉ được viết 1 câu lệnh trên 1 dòng (trừ những trường hợp ngoại lệ)

4. If, while

- Luôn luôn dùng {}, kể cả với khối lệnh chỉ gồm 1 lệnh
- Đặt dấu { sau điều kiện (không đặt ở dòng tiếp theo), ví dụ:

```
while ( hits == 0 ){
     ...;
}
```

- Chuỗi else if và else được trình bày như minh họa dưới đây:

```
if( hits == 0 ) {
     ... ;
     ... ;
```

```
}
else if( hits < 10 ) {
     ... ;
}
else if( hits < 50 ) {
     ... ;
}
else {
     ... ;
     ... ;
}
```

– Không dùng dấu = như là điều kiện so sánh bằng (phải là ==)

5. Khai báo

– Các biến được khai báo ở trên cùng của thân hàm

– Khai báo 1 biến trên 1 dòng nếu như có gán giá trị khởi tạo khi khai báo; chỉ sử dụng khai báo nhiều biến trên cùng 1 dòng (các biến cách nhau bằng dấu phẩy) nếu như chúng có quan hệ với nhau (ví dụ `int row, col;`)

– Trong 1 khối các định nghĩa có liên quan đến nhau, align sao cho các tên, kiểu dữ liệu và khởi tạo thẳng hàng nhau.

– Các biến nên được khởi tạo ở gần chỗ mà chúng được sử dụng lần đầu tiên.

– Tất cả struct và enum được định nghĩa với typedef.

– Tất cả các nguyên mẫu hàm (prototype) và typedef cho hàm và kiểu dữ liệu mà hàm và kiểu dữ liệu này chỉ được dùng cục bộ trong file nên được đặt ở đầu của file.

– Tất cả các nguyên mẫu hàm (prototype) và typedef cho hàm và kiểu dữ liệu được sử dụng ở nhiều file nên được đặt ở 1 file .h, file .h này sẽ được include trong file mà các hàm của nó được định nghĩa và trong những file mà các hàm hoặc kiểu dữ liệu của nó được dùng.

6. Hàm

– Kiểu trả về của hàm được đặt cùng dòng với tên hàm

– Hàm nên được viết ngắn gọn, nên chia thành các chức năng nhỏ.

– Mỗi hàm nên thực hiện một mục đích cụ thể nào đó, tên hàm phải được đặt sao cho phản ánh được mục đích của hàm.

– Hàm `main` là hàm đầu tiên trong file

– Tất cả các nguyên mẫu hàm (prototype) trong file được khai báo ở phía trên hàm main.

– Hàm phải có các đối số; nếu hàm không có đối số thì phải viết `( void )` trong nguyên mẫu hàm.

7. Struct

– typedef nên được sử dụng khi dùng struct, từ khóa "`struct`" không nên xuất hiện trong thân chương trình.

– typedef nên có tên giống với tên trong struct nhưng ký tự đầu viết hoa. Ví dụ

– typedef struct node Node;

```
struct node {
   int   val;
   Node *next;
 };
```

8. Chú thích

– Sinh viên được tự do trong cách viết chú thích, sự rõ ràng trong chú thích là điều quan trọng nhất.

– Nên sử dụng kiểu chú thích `//` thay vì `/* */`

– Với chương trình có nhiều hàm, chú thích để có thể dùng các hệ thống sinh tài liệu tự động như doxygen nên được sử dụng.

– Nên sử dụng kiểu chú thích trên nhiều dòng ở đầu hàm hoặc khối lệnh, không nên sử dụng nhiều chú thích kiểu `//` một lúc.

– Chú thích đặc biệt quan trọng để giải thích các đoạn code khó hiểu.

– Tránh chú thích những thứ quá rõ ràng, ví dụ:

```
x = y + z; // gán x bằng tổng của y và z
```

9. Các lựa chọn khi biên dịch

– Trừ khi được chỉ ra một cách rõ ràng, còn lại tất cả các chương trình nên được biên dịch với `gcc -Wall`

– Nếu sau khi dịch, chương trình chạy bị chậm, biên dịch lại chương trình, sử dụng lựa chọn tối ưu mã:

```
gcc -Wall -O
```

(hoặc `-o2` hoặc `-o3` nếu muốn tối ưu hơn)

10. Lời cảm ơn

Tài liệu này được viết dựa trên hướng dẫn phong cách lập trình cho các khóa học năm thứ nhất tại CSE@UNSW (Đại học New SothWale của Úc).