CSC10001

SƠ LƯỢC VỀ MÔI TRƯỜNG LẬP TRÌNH VỚI C/C++

FIT-HCMUS

Mục lục

1 Các trình biên dịch C/C++ thông dụng

1.1 GCC (GNU Compiler Collection)

- Trình dịch chính thức của hệ thống GNU
- Thường dùng trên các hệ điều hành thuộc Unix/Linux
- Thường dùng làm môi trường phát triển cho các phần mềm thương mại chạy trên đa nền tảng

1.2 Clang

- Dùng trên hệ điều hành Unix (vd: MacOS)
- Thiết kế hướng tới tốc độ xử lý với bộ thông báo lỗi rõ ràng chi tiết

1.3 Microsoft Visual C++

- Được viết bởi Microsoft, và là thiết kế riêng cho bộ IDE Microsoft Visual C++
- Chạy trên Windows

2 ... và các IDE thường dùng

2.1 Dev C++ *(Compiler: gcc)*

1. Ưu điểm:
   - Nhẹ
   - Đơn giản dễ dùng
2. Khuyết điểm:
   - Không hỗ trợ nhiều chức năng
   - Không quản lý project tốt
   - Chỉ chạy được trên Windows

2.2 Microsoft Visual C++ *(Compiler: Microsoft Visual C++)*

1. Ưu điểm:
   - Quản lý project tốt
   - Nhiều tính năng (debugger, incremental link, ...)
   - Tích hợp cho nhiều ngôn ngữ (C, C++, C#), hỗ trợ công cụ cho nhiều dạng sản phẩm (app, web, ...)
2. Khuyết điểm:
   - Nặng
   - Nhiều thư viện hướng hệ điều hành
   - Chỉ chạy được trên Windows

2.3 Eclipse *(Compiler: gcc/clang/MVC)*

1. Ưu điểm:
    - Chạy trên nhiều nền tảng Windows, Linux, Mac OS
    - Quản lý project tốt
    - Nhiều tính năng
2. Khuyết điểm:
    - Nặng

2.4 Code::Blocks *(Compiler: gcc)*

1. Ưu điểm:
    - Chạy trên nhiều nền tảng Windows, Linux, Mac OS
    - Quản lý project tốt
    - Nhiều tính năng
2. Khuyết điểm:
    - Nặng

2.5 Xcode *(Compiler: gcc, clang)*

1. Ưu điểm:
    - Nhiều tính năng
    - Hỗ trợ lập trình trên di động cho các máy có hệ điều hành thuộc Mac và iOS
2. Khuyết điểm:
    - Chỉ chạy trên Mac OS
    - Nặng

3 Thêm: một số code editor

3.1 Sublime Text

3.2 Visual Studio Code

3.3 Atoms

3.4 VI/VIM

4 Phong cách lập trình (Programming style)

4.1 Tài liệu tham khảo

1. Một số chú ý trong viết code

2. Các câu hỏi thường gặp liên quan đến phong cách lập trình
3. C++ Programming Style Guidelines
4. Google C++ Style Guide

4.2 Cơ bản

Phong cách code tùy theo sở thích và thói quen mỗi người. Tuy nhiên cũng có những quy chuẩn chung trong phong cách lập trình để phần code lập trình trong sáng, dễ đọc dễ theo dõi và mang tính kế thừa.

1. Đặt tên biến
   - Các cách thường dùng:
     - Phong cách lạc đà (camel case style) viết thường chữ cái đầu tiên: `totalMoney`
     - Các từ cách nhau bởi dấu gạch dưới (_): `total_money`
   - Đặt tên biến có chức năng gợi ý ý nghĩa của biến.
     - Ví dụ: đặt tên biến để lưu trữ tổng số tiền
     - Nên đặt: `totalMoney`
     - Không nên đặt: ~~`temp1`~~, ~~`abc`~~,...
2. Đặt tên hàm
   - Các cách thường dùng:
     - Phong cách lạc đà (camel case style) viết thường chữ cái đầu tiên của động từ: `calculateTotalMoney(...)` Lưu ý: nếu đã đặt tên biến theo phong cách lạc đà thì nên chọn cách đặt tên hàm khác để tránh nhập nhằng.
     - Phong cách lạc đà (camel case style) viết hoa toàn bộ chữ cái bắt đầu của từ: `CalculateTotalMoney(...)`
   - Nên đặt tên hàm có động từ đầu tiên
     - Ví dụ: hàm thực hiện tính tổng số tiền
     - Nên đặt: `CalculateTotalMoney(...)`
     - Không nên đặt: ~~`TotalMoney(...)`~~, ~~`Money(...)`~~,...
3. Comment
   - Comment theo dòng:

```
// This is comment line
```

   - Comment theo đoạn:

```
/*
   This is comment block
   Line 1
   Line 2
   ...
*/
```

4. Đóng/mở đoạn code

- Phong cách 1:

```
if (is_student == true) {
    // Do something
} else {
    // Do something else
}
```

- Phong cách 2:

```
if (is_student == true)
{
    // Do something
}
else
{
    // Do something else
}
```

5. Thụt dòng và khoảng trằng

- Quy tắc 1: Đoạn code con của một đoạn code phải thụt vào 1 dấu tab hoặc 4 khoảng trắng so với đoạn code cha.
  - Nên:

```
int i;
int sum = 0;

for (i = 0; i <= 100; i++)
{
    cout << i << "\n";
    sum += i;
}
```

  - Không nên:

```
int i;
int sum = 0;

for (i = 0; i <= 100; i++)
{
cout << i << "\n";
        sum += i;
}
```

- Quy tắc 2: Các toán tử và toán hạng phải cách nhau bởi một dấu cách.
  - Nên:

```
int first_number;
int second_number;
int sum_of_two_numbers;

first_number = 5;
sencond_number = 10;

sum_of_two_numbers = first_number + second_number;
```

  - Không nên:

```
int first_number;
int second_number;
int sum_of_two_numbers;

first_number=5;
sencond_number=10;

sum_of_two_numbers=first_number+second_number;
```

- Quy tắc 3: Các thành phần trong câu lệnh phải được ngắn cách với nhau. Các dấu chấm phẩy (;), dấu phẩy (,), dấu hai chấm (:),... nằm sát thành phần đầu tiên và cách thành phần sau bởi 1 khoảng trắng.
  - Nên:

```
int i;

for (i = 5; i <= 50; i++)
{
    // Do something
}
```

  - Không nên:

```
int i;

for (i=5;i<=50;i++)
{
    // Do something
}
```

- Quy tắc 4: Các dòng code "có liên quan" nên đặt gần nhau và cách các dòng code "không liên quan" từ 1-2 dòng trống. Nên có chú thích cho mỗi đoạn code.
  - Nên:

```
// Declare variable
int first_number;
int second_number;
int sum_of_two_numbers;

// Assign data to the variable
first_number = 5;
sencond_number = 10;

// Calculate the sum of two numbers and print it
sum_of_two_numbers = first_number + second_number;
cout << first_number << " + " << second_number << " = " sum_of_two_numbers;
```

  - Không nên:

```
int first_number;
int second_number;
int sum_of_two_numbers;
first_number = 5;
sencond_number = 10;
sum_of_two_numbers = first_number + second_number;
cout << first_number << " + " << second_number << " = " sum_of_two_numbers;
```

5 Kết luận

1. Không có IDE, compiler hay editor tốt nhất, tùy vào mục đích sử dụng mà lựa chọn cho phù hợp
2. Không có phong cách lập trình tốt nhất, tùy vào sở thích và tư duy mỗi người mà lựa chọn cho phù hợp
3. “Aim for best readability. Primary importance is that code is easy to read and logically follow”, Monash University’s Programming styles