

NỘI DUNG PRE-AFTER-C24

1. Tổng quan

1.1. Mục đích

Nội dung Pre-After-C24 được giao cho các bạn khóa C24 trong giai đoạn trước khi khóa học After-C24 diễn ra nhằm hướng tới các mục đích chính sau:

- Giúp thành viên ôn tập lại kiến thức đã học trong khóa học C24.
- củng cố kiến thức cơ bản nhằm tạo nền tảng cho thành viên mới trước khi tham gia khóa After-C24 sắp tới.
- Giới thiệu và hướng dẫn thành viên mới sử dụng các nền tảng cho mục đích lưu trữ, báo cáo.

1.2. Nội dung chính

Phần 1. Tổng quan: trình bày mục đích của nội dung, tóm tắt nội dung chính từng phần, đối tượng và thời gian thực hiện bài tập.

Phần 2. Nội dung: trình bày kiến thức cần nắm trong giai đoạn chuẩn bị cho khóa After-C24. *(Lưu ý: nội dung được trình bày ở phần này đã được tóm tắt, nên sử dụng các từ khóa có sẵn để tìm kiếm nội dung đầy đủ).*

Phần 3. Bài tập: bài tập liên quan đến các nội dung được trình bày trong Phần 2. **Nội dung** và các yêu cầu cần thực hiện.

Phần 4. Hình thức nộp bài: yêu cầu và hướng dẫn khi nộp bài.

1.3. Đối tượng và thời gian thực hiện bài tập

Đối tượng: thành viên khóa C24.

Thời gian thực hiện: tuần 5, tuần 6, tuần 7, tuần 8.

Hạn cuối nộp trong form: 23h59 20/02/2024 (Thứ 3 Tuần 8).

Bài sửa sẽ được đăng vào ngày 21/02/2024 (Thứ 4 Tuần 8).

2. Nội dung

2.1. Các hệ thống số và phép toán logic

2.1.1. Hệ thống số

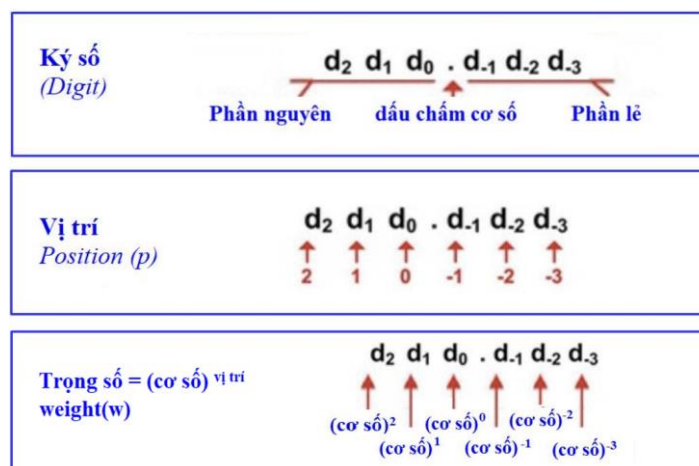
Hệ đếm hay còn gọi hệ cơ số là hệ thống dùng để biểu diễn các chữ số. Nó bao gồm các ký hiệu toán học dùng để thể hiện các số của một tập hợp số bằng cách sử dụng nhất quán các chữ số hay các ký hiệu. Ví dụ: Số có hai ký số "11" có thể được hiểu là biểu diễn hệ thập phân của số 11 hoặc hệ nhị phân của số 3.

Các hệ thống số thường dùng: nhị phân (binary), bát phân (octal), thập phân (decimal), thập lục phân (hexadecimal).

Cấu trúc cơ bản của một hệ thống số bao gồm: **phần nguyên** và **phần lẻ** được hình thành bởi các **ký số** và cách nhau bởi **dấu chấm cơ số**.

Các khái niệm cơ bản về hệ thống số:

- Cơ số (Radix)
- Ký số (Digit)
- Vị trí của ký số (Position)
- Trọng số (Weight)

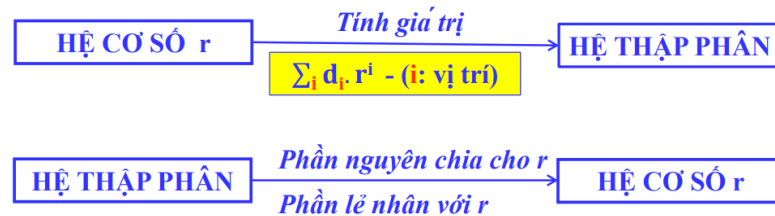


| HỆ THỐNG | THẬP PHÂN (Decimal) | NHỊ PHÂN (Binary) | BÁT PHÂN (Octal) | THẬP LỤC PHÂN (HexaDecimal) |
|------------------|---|--------------------------------|--------------------------------|---------------------------------|
| KÝ SỐ (Digit) | 0 ÷ 9 | 0, 1 | 0 ÷ 7 | 0 ÷ 9, A ÷ F |
| CƠ SỐ (Radix) | $r = 10$ | $r = 2$ | $r = 8$ | $r = 16$ |
| KÝ HIỆU (Symbol) | { D, ₁₀ } | { B, ₂ } | { O, ₈ } | { H, ₁₆ } |
| VÍ DỤ | 125.37D 125.37 ₁₀ | 101.11B 101.11 ₂ | 623.14O 623.14 ₈ | F8E.0CH F8E.0C ₁₆ |
| GIÁ TRỊ (Value) | $\sum_i d_i \cdot r^i$ Giá trị của một số chính là chuyển đổi số đó về hệ thống số thập phân (i: vị trí - r: cơ số) | | | |

Ví dụ: Các khái niệm trong hệ thập phân (cơ số 10)

| Position | 3 | 2 | 1 | 0 | -1 | -2 | ... |
|------------------------|-------|-------|-------|-------|----------|----------|-----|
| Weight | b^3 | b^2 | b^1 | b^0 | b^{-1} | b^{-2} | ... |
| Digit | a_3 | a_2 | a_1 | a_0 | c_1 | c_2 | ... |
| Decimal example weight | 1000 | 100 | 10 | 1 | 0.1 | 0.01 | ... |
| Decimal example digit | 4 | 3 | 2 | 7 | 0 | 0 | ... |

Cách chuyển đổi giữa các hệ thống số: để chuyển số từ hệ thống cơ số m sang hệ thống cơ số n bất kỳ, ta tính giá trị của số đó trong hệ thập phân (chuyển từ hệ cơ số m sang hệ thập phân) và chuyển giá trị đó sang hệ thống cơ số n (chuyển từ hệ thập phân sang hệ thống cơ số n).



Ví dụ: Chuyển đổi số nhị phân (cơ số 2) sang hệ thập phân (cơ số 10):

$$(0110\ 1001)_2 \xrightarrow{\sum_i d_i \cdot r^i = 1 \cdot 2^0 + 0 \cdot 2^1 + 0 \cdot 2^2 + 1 \cdot 2^3 + 0 \cdot 2^4 + 1 \cdot 2^5 + 0 \cdot 2^6 + 0 \cdot 2^7} (105)_{10}$$

Cách chuyển đổi nhanh áp dụng giữa hệ nhị phân, bát phân, thập lục phân:



Ví dụ: Chuyển đổi số thập lục phân (cơ số 16) sang hệ nhị phân (cơ số 2):

$$(F6B)_{16(HEX)} \xrightarrow{(F)_{16} \rightarrow (1111)_2; (6)_{16} \rightarrow (0110)_2; (B)_{16} \rightarrow (1011)_2} (1111\ 0110\ 1011)_2$$

2.1.2. Hệ nhị phân (BIN)

Hệ nhị phân (hay hệ đếm cơ số hai hoặc mã nhị phân) là một hệ đếm dùng hai ký số 0 và 1 để biểu đạt một số có giá trị là tổng số các lũy thừa của 2. Hai ký số 0 và 1 còn được dùng để biểu đạt hai giá trị hiệu điện thế tương ứng, vì đó hệ nhị phân được dùng rộng rãi trong lĩnh vực điện tử.

Một số định nghĩa của hệ nhị phân:

- Mỗi ký số trong hệ nhị phân được gọi là **bit** (binary digit).
- MSB (**M**ost **S**ignificant **B**it): bit có trọng số lớn nhất.
- LSB (**L**east **S**ignificant **B**it): bit có trọng số nhỏ nhất.

Tính chất của hệ nhị phân:

- Số nhị phân n bit không dấu có tầm giá trị từ $0 \rightarrow (2^n - 1)$.
Ví dụ: số nhị phân có 7 bit có tầm giá trị từ $0 \rightarrow 127$.
- Số nhị phân có giá trị lẻ có LSB = 1.
- Số nhị phân có giá trị chẵn có LSB = 0.
- Lý do: vì bit LSB có trọng số là 1.

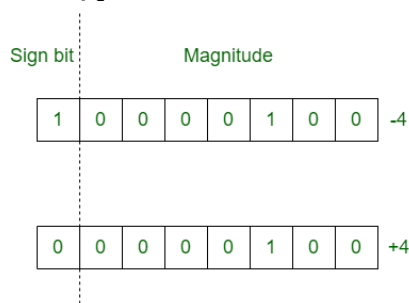
Các bội số của bit:

- 1 nibble = 4 bit
- 1 byte = 8 bit = 2 nibble
- 1KB = 2^{10} byte = 1024 byte
- 1MB = 2^{10} KB = 2^{20} byte
- 1GB = 2^{10} MB = 2^{30} byte
- 1 word = n bit, $\{n = 16, 32, \dots\}$
- 1 half-word = $\frac{1}{2}$ word
- 1 kb = 1000 bit

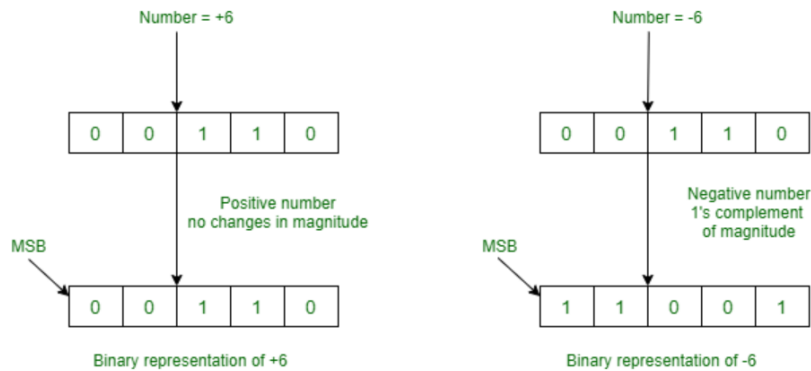
Số nhị phân có dấu:

| Số có dấu theo biên độ (Signed Magnitude) | Số có dấu theo số bù 1 (1's complement) | Số có dấu theo số bù 2 (2's complement) |
|---|--|--|
| QUI ƯỚC | | |
| - MSB = bit dấu - Số (+) = 0 Số (-) = 1 - Phần còn lại là độ lớn. | - Số (+): MSB = 0, phần còn lại là độ lớn. - Số (-): lấy bù 1 của số dương tương ứng. | - Số (+): MSB = 0, phần còn lại là độ lớn. - Số (-): lấy bù 2 của số dương tương ứng. |
| - Số 0 có 2 cách biểu diễn | - Số 0 có 2 cách biểu diễn | - Số 0 có 1 cách biểu diễn |
| PHẠM VI BIỂU DIỄN | | |
| $-(2^{n-1} - 1) \div + (2^{n-1} - 1)$ | $-(2^{n-1} - 1) \div + (2^{n-1} - 1)$ | $-(2^{n-1}) \div + (2^{n-1} - 1)$ |

- Số nhị phân có dấu theo biên độ:



- Số nhị phân có dấu theo dạng số bù 1: (đảo các bit của số nhị phân dương)



- Số nhị phân có dấu theo dạng số bù 2:

0 0 0 1 0 1 0 0 → Binary number

1 1 1 0 1 0 1 1 → One's complement

| |
|-----------------|
| 1 1 1 0 1 0 1 1 |
| + 1 |
| 1 1 1 0 1 1 0 0 |

→ 2s complement

Lưu ý: Trong lập trình thường sử dụng số nhị phân có dấu theo dạng số bù 2.

Cách chuyển đổi từ số thập phân có dấu sang số nhị phân có dấu:

- Trường hợp số thập phân có giá trị **âm**:

$$(\text{SỐ ÂM})_{10} \xrightarrow{\text{GIÁ TRỊ TUYỆT ĐỐI}} (\text{SỐ DƯƠNG})_{10} \xrightarrow{\text{HỆ NHỊ PHÂN}} (\text{NHỊ PHÂN})_2 \xrightarrow{\text{BĐ/BÙ 1/BÙ 2}} (\text{NHỊ PHÂN CÓ DẤU})_2$$

- Trường hợp số thập phân có giá trị **dương**: (chuyển đổi như số nhị phân không dấu với MSB = 0)

$$(\text{SỐ DƯƠNG})_{10} \xrightarrow{\text{HỆ NHỊ PHÂN}} (\text{NHỊ PHÂN})_2$$

Cách chuyển đổi nhanh số nhị phân có dấu dạng số bù 2:

- Từ LSB đến MSB, giữ nguyên các bit cho đến khi gặp bit 1 đầu tiên, thực hiện đảo bit các bit sau bit 1 đó cho đến MSB.

Ví dụ: chuyển đổi số -360 sang số nhị phân có dấu bù 2:

$$(-360)_{10} \xrightarrow{|-360| \text{ Hệ nhị phân}} (01\ 0110\ 1000)_2$$

Chuyển nhanh sang dạng số nhị phân bù 2:

$$\begin{array}{l} \overline{(01\ 0110\ 1000)}_2 \\ (10\ 1001\ 1000)_{2_bù2} \end{array}$$

2.1.3. Hệ thập lục phân (HEX)

Hệ thập lục phân (hay số HEX) là hệ thống số đếm có cơ số là 16, được biểu diễn thông qua 16 ký tự bao gồm: 0, 1, 2, ..., 9 và A, B, C, D, E, F (chữ hoa và chữ thường như nhau). Việc sử dụng hệ HEX giúp việc lập trình với hệ thống các bit được gọn gàng hơn vì 1 chữ số hệ thập lục bằng 4 bit hệ nhị phân.

Cách chuyển 16 kí tự HEX sang hệ nhị phân:

| Hệ thập lục | Hệ nhị phân |
|-------------|-------------|
| 0 | 0000 |
| 1 | 0001 |
| 2 | 0010 |
| 3 | 0011 |
| 4 | 0100 |
| 5 | 0101 |
| 6 | 0110 |
| 7 | 0111 |
| 8 | 1000 |
| 9 | 1001 |
| A | 1010 |
| B | 1011 |
| C | 1100 |
| D | 1101 |
| E | 1110 |
| F | 1111 |

Chuyển đổi từ hệ nhị phân sang thập lục phân: để chuyển đổi một dãy nhị phân thành số thập lục phân, ta nhóm lần lượt các bit nhị phân từ phải qua trái thành các nhóm 4 bit, sau đó chuyển các nhóm 4 bit này thành số thập lục phân. Ngược lại khi chuyển từ hệ thập lục phân sang nhị phân.

Ví dụ: $(1\ 0101\ 1011\ 1111)_2 = (15BF)_{16}$

*Mở rộng: Chuyển đổi từ hệ nhị phân sang bát phân: tương tự trên với nhóm 3 bit.

Ví dụ: $(11\ 001\ 000\ 111)_2 = (3107)_8$

2.1.4. Các phép toán logic (Xem lại kiến thức buổi 4 khóa ĐTCB2023)

Phép toán logic là các phép toán được thực hiện trên các giá trị logic (true hoặc false). Các phép toán logic cơ bản bao gồm: NOT, AND, OR, XOR được biểu diễn bởi các bảng chân trị sau:

AND Truth Table

| A | B | Y |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

OR Truth Table

| A | B | Y |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

XOR Truth Table

| A | B | Y |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

NOT Truth Table

| A | B |
|---|---|
| 0 | 1 |
| 1 | 0 |

Ví dụ: Cho hai số nhị phân 8-bit A và B:

A = 1011 1001

B = 0110 1000

(AND) A&B = 0010 1000

(OR) A|B = 1111 1001

(XOR) A^B = 1101 0001

(NOT) ~A = 0100 0110

2.2. Embedded C

2.2.1. Phần mềm

Cài đặt Visual Studio Code (và extension C/C++), compile: MinGW và cài đặt môi trường theo hướng dẫn sau: [Cài Đặt Visual Studio Code Lập Trình C++ Dễ Như Ăn Kẹo \(codelearn.io\)](https://codelearn.io/c/c++/visual-studio-code). Sau đó viết chương trình hello_world.c in ra dòng chữ "hello_world" ở terminal.

2.2.2. C cơ bản

Xem lại phần Basic C (Buổi 1 và Buổi 2) của khóa C24 và nắm được kiến thức sau:

- Biến: biến toàn cục, biến cục bộ, các kiểu dữ liệu và format specifier của từng loại, biến của stdint.h (uint8_t, uint16_t, uint32_t, uint64_t, int8_t, int16_t, int32_t, int64_t).
- Toán tử: toán tử số học, toán tử quan hệ (so sánh), toán tử logic, ép kiểu (tìm hiểu ngoài).
- Cấu trúc rẽ nhánh: if, if .. else, if ... else if.
- Cấu trúc vòng lặp: while, do ... while, for, switch case.
- Hàm (function): cách định nghĩa hàm, giá trị tham số/tham trị truyền vào hàm, giá trị trả về của hàm.
- Con trỏ (pointer): khai báo biến con trỏ, cách truy xuất địa chỉ đang được lưu trong biến con trỏ, cách truy xuất giá trị của ô nhớ mà biến con trỏ đang trỏ đến.
- Mảng (array): array là gì? cú pháp tạo mảng, cách truy xuất tới phần tử trong mảng.
- Chuỗi ký tự (string): string là gì? cách tạo chuỗi, các hàm trong string.h (strlen(), strcmp(), strcat(), strcpy(), strlwr(),strupr(), strrev(), strchr(), strstr()), sprintf().
- Mã ASCII.
- Struct: struct là gì? cú pháp tạo struct, cách tạo biến và mảng từ struct, truy xuất tới 1 trường trong struct.

2.2.3. Kiến thức C bổ sung

Enum:

- Enum trong ngôn ngữ C là kiểu dữ liệu do người dùng tự định nghĩa. Được sử dụng chủ yếu để gán tên cho các hằng số, giúp chương trình dễ đọc và dễ bảo trì.
- Cú pháp định nghĩa enum:

```
enum <tên_enum> { <DANH SÁCH CÁC GIÁ TRỊ> };
```

Ví dụ: `enum weekday {MON, TUE, WED, THU, FRI, SAT, SUN};`

Không cần trực tiếp gán giá trị cho các tên hằng số, compiler tự động khởi tạo giá trị cho danh sách các giá trị, bắt đầu với giá trị 0 và tăng dần. Ở ví dụ trên, giá trị của danh sách tương ứng: `MON=0`, `TUE=1`, `WED=2`, `THU=3`, `FRI=4`, `SAT=5`, `SUN=6`.

- Khi định nghĩa enum, nếu ta gán giá trị cho một hằng số, giá trị của các hằng số tiếp theo sẽ tăng dần từ giá trị đã gán cho hằng số đó:

Ví dụ: `enum weekday {MON, TUE=3, WED, THU, FRI, SAT, SUN};`

Giá trị của danh sách: `MON=0`, `TUE=3`, `WED=4`, `THU=5`, `FRI=6`, `SAT=7`, `SUN=8`.

- Cú pháp khai báo biến enum:

`enum <tên_enum> <tên_biến_enum>;`

- Lúc này, `<tên_biến_enum>` có giá trị là tên của một các hằng số nằm trong danh sách đã định nghĩa.

Ví dụ: `enum weekday today;`
`today = TUE;`
`today = today + 1; //today = TUE + 1 = 3 + 1 = 4`

Typedef:

- Typedef được sử dụng để tạo tên mới cho một kiểu dữ liệu đã có sẵn.

- Cú pháp:

`typedef <kiểu_cũ> <kiểu_mới>;`

Ví dụ: `typedef unsigned char uint8_t;`

Lúc này, thay vì khai báo biến: `unsigned char count;`

thì ta có thể khai báo bằng cách `uint8_t count;`

- Typedef thường dùng khi định nghĩa struct và enum:

```
typedef struct {
    uint8_t height;
    uint8_t width;
} rectangle;

rectangle rect_a;

typedef enum {mon, tue, wed, thu, fri, sat, sun} weekday;

weekday today;
```

2.3. Coding convention

Coding convention là một tập hợp các quy tắc (rules), hướng dẫn (guidelines) giúp chúng ta viết code tốt hơn, hướng tới hai mục tiêu chính: đảm bảo code đúng kỹ thuật và tính đồng bộ code giữa một cộng đồng, một nhóm làm việc chung với nhau.

Coding convention có thể được triển khai bởi một nhóm dự án (dùng trong một dự án cụ thể), một công ty (dùng trong nội bộ công ty) hoặc các tổ chức (được dùng rộng rãi trên trái đất). Một số chuẩn coding convention được sử dụng phổ biến hiện nay:



Trong file này, ta sẽ sử dụng chuẩn coding convention của **Barr Group's Embedded C Coding Standard** (xem tại sách coding convention bên dưới).

Tham khảo slide After C23 - Coding convention tutorial: [PIF Coding Standards.pdf](#)

Tham khảo sách coding convention: [barr c coding standard 2018.pdf](#)

Các mục coding convention liên quan đến kiến thức C24 đã học:

2.3.1. Line widths

Chiều dài tối đa của một dòng trong một chương trình là 80 ký tự.

2.3.2. Braces

- Các dấu ngoặc nhọn phải luôn bao quanh một khối code, theo sau ``if, else, switch, while, do, for`` và ``for staments``.
- Mỗi dấu ngoặc nhọn trái `{` phải phải xuất hiện trên một dòng bắt đầu của khối code mà nó mở ra. Dấu ngoặc nhọn phải `}` tương ứng phải xuất hiện ở dòng cuối cùng của khối code mà nó khép lại.

```

1  {
2      if (depth_in_ft > 10) dive_stage = DIVE_DEEP;    // This is legal...
3      else if (depth_in_ft > 0)
4          dive_stage = DIVE_SHALLOW;                // ... as is this.
5      else
6          {                                           // But using braces is always safer.
7              dive_stage = DIVE_SURFACE;
8          }
9      ...
10 }
```

2.3.3. Spaces

- Mỗi keywords `if, while, for, switch, return` theo sau nó phải là một khoảng trắng sau đó mới là nội dung code tiếp theo.

```

1  if (...)
2  {
3      ...
4  }
5
6  while (...)
7  {
8      ...
9  }

```

- Mỗi toán tử gán `=, +=, -=, *=, /=, %=, &=, |=, ^=, ~=, !=` luôn đứng trước và theo sau bởi một khoảng trắng.

```

1  count += 1;
2  reload = count;

```

- Mỗi toán tử nhị phân `+, -, *, /, %, <, <=, >, >=, ==, !=, <<, >>, &, |, ^, &&, ||` luôn đứng trước và theo sau bởi một khoảng trắng.

```

1  ODR = 1 << 12;
2  is_debounced = (time - time_btn) >= 50;

```

- Toán tử một ngôi `+, -, ++, --, !, ~` sẽ được viết không có dấu khoảng trắng ở phía toán hạng.

```

1  count++;
2  state = !state;

```

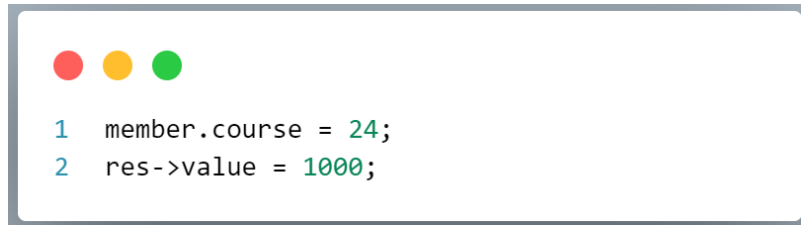
- Toán tử con trỏ `*` và `&` sẽ được viết bằng khoảng trắng ở mỗi bên trong các khai báo nhưng mặt khác không có khoảng trắng ở phía toán hạng.

```

1  /* Pointer Declaration */
2  uint32_t * reg_p;
3  /* Pointer Operator */
4  reg_p = &APB1;
5  reg_read = *reg_p;

```

- `?` và `:` sẽ luôn đứng trước và theo sau một khoảng trắng.
- Structure pointer và toán tử structure member `->` sẽ luôn không có khoảng trắng xung quanh.



```
1 member.course = 24;
2 res->value = 1000;
```

- Dấu ngoặc vuông trái và phải của một array `[]` sẽ không có khoảng trắng bao quanh.



```
1 arr[1] = ...;
```

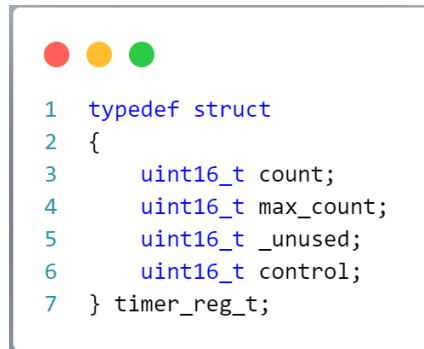
- Các biểu thức trong dấu ngoặc đơn sẽ luôn không có khoảng trắng liền kề với ký tự dấu ngoặc đơn bên trái và bên phải.
- Dấu ngoặc đơn bên trái và phải của phép gọi hàm sẽ luôn không có khoảng trắng xung quanh.
- Trừ trường hợp ở cuối dòng, mỗi tham số hàm phải được phân tách nhau bằng dấu phẩy phải theo sau một dấu cách.
- Mỗi dấu chấm phẩy ngăn cách các phần tử của câu lệnh for phải luôn được theo sau bởi một dấu cách.
- Mỗi dấu chấm phẩy sẽ theo sau câu lệnh mà nó kết thúc mà trước nó không có bất kì khoảng trắng nào.

2.3.4. Blank Lines

- Không có dòng code nào được chứa nhiều hơn một câu lệnh.
- Phải có một dòng trống trước và sau mỗi khối code. Như là các vòng lặp `if ... else`, `switch` và các câu lệnh liên tiếp.
- Mỗi source file sẽ kết thúc bằng một dấu comment đánh dấu phần cuối của file, sau đó là một dòng trống.
- Mỗi level thụt lề phải căn chỉnh theo bội số của 4 ký tự ngay từ đầu của dòng (1 Tab).
- Trong cấu trúc `switch`, các label phải được căn chỉnh, nội dung của mỗi khối phải được thụt lề từ đó.
- Bất cứ khi nào dòng code quá dài vượt quá độ dài tối đa, hãy thụt dòng thứ hai và bất kỳ dòng tiếp theo nào theo các dễ đọc nhất.

2.3.5. Data Type Rules - Naming Conventions

- Tên của tất cả các kiểu dữ liệu mới, bao gồm structure, union, enum sẽ chỉ bao gồm các ký tự chữ thường và gạch dưới và kết thúc bằng `_t`.
- Tất cả các structure, union, enum sẽ được đặt tên thông qua `typedef`.
- Tên của tất cả các loại dữ liệu sẽ có tiền tố là tên module và dấu gạch dưới.



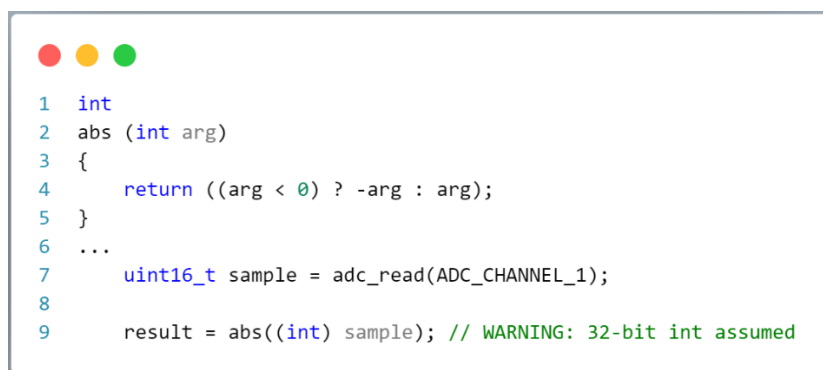
```

1  typedef struct
2  {
3      uint16_t count;
4      uint16_t max_count;
5      uint16_t _unused;
6      uint16_t control;
7  } timer_reg_t;

```

2.3.6. Casts

- Mỗi lần ép kiểu nên có phần comment liên quan đến việc đảm bảo việc ép kiểu là phù hợp trên phạm vi giá trị mà kiểu có.



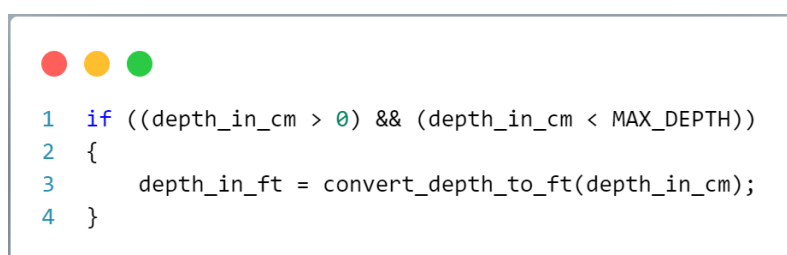
```

1  int
2  abs (int arg)
3  {
4      return ((arg < 0) ? -arg : arg);
5  }
6  ...
7      uint16_t sample = adc_read(ADC_CHANNEL_1);
8
9      result = abs((int) sample); // WARNING: 32-bit int assumed

```

2.3.7. Parentheses

- Không dựa vào các quy tắc ưu tiên của toán tử C, vì chúng có thể không rõ ràng. Để rõ ràng, sử dụng dấu ngoặc đơn (hoặc chia các câu lệnh dài thành nhiều dòng code) để đảm bảo thứ tự thực hiện đúng trong một chuỗi phép toán.
- Trừ khi đó là một toán tử riêng lẻ hoặc một hằng số duy nhất, mỗi toán hạng logic AND `&&` và OR `||` sẽ được bao quanh bởi dấu ngoặc đơn.



```

1  if ((depth_in_cm > 0) && (depth_in_cm < MAX_DEPTH))
2  {
3      depth_in_ft = convert_depth_to_ft(depth_in_cm);
4  }

```

2.3.8. Procedure Rule - Naming Conventions

Procedure là thủ tục ở trong C: đề cập đến các thành phần có tên để gọi khi lập trình (hàm, biến, macro, khai báo struct, union, enum,...)

- Không có tên của thủ tục nào có chứa keyword của bất kỳ tiêu chuẩn phiên bản nào của C hoặc C++. Các tên bị hạn chế bao gồm `'interrupt'`, `'inline'`, `'class'`, `'true'`, `'false'`, `'public'`, `'private'`, `'friend'`, `'protected'`,...
- Không có thủ tục nào có tên trùng với một hàm trong C Standard Library. Như `'strlen'`, `'atoi'`, `'memset'`, `'scanf'`, `'printf'`,...
- Không có thủ tục nào có tên bắt đầu bằng dấu gạch dưới.
- Không có thủ tục nào có tên dài quá 31 ký tự.
- Không tên hàm nào được chứa chữ viết hoa.
- Không có tên macro nào chứa chữ viết thường.
- Dấu gạch dưới sẽ dùng để phân cách các chữ riêng biệt trong tên của thủ tục.
- Tên của mỗi thủ tục phải mô tả được đúng mục đích của nó. Các thủ tục gói gọn các "hành động" của một chương trình bằng cách sử dụng động từ trong tên của thủ tục (ví dụ: `'adc_read'`) `'noun-verb'`. Ngoài ra các thủ tục cố thể được đặt tên theo câu trả lời mà thủ tục muốn trả lời (ví dụ: `'led_is_on'`).
- Tên của tất cả public function sẽ có tiền tố là tên module của chúng và phân cách bằng dấu gạch dưới (ví dụ: `'sensor_read'`).

2.4. Altium Designer

Link file Install Altium Designer Tutorial: [Altium Designer Tutorial.pdf](#)

2.5. Git – Github

Link file Git-Github Tutorial: [Git-Github Tutorial Pre AfterC24.pdf](#)

3. Bài tập

3.1. Hệ thống số

(Trình bày phần giải bài tập trong file *Pre_AfterC24_Report.pdf* ở Phần 4. **Yêu cầu bài nộp**)

Bài tập hệ thống số: Cho 2 số $a = (22122009)_{16}$ và $b = (A10420F3)_{16}$

- Biểu diễn a trong hệ thập phân, nhị phân.
- Thực hiện phép toán: $\text{not } a$; $a \text{ and } b$; $a \text{ or } b$; $a \text{ xor } b$.
- Kích thước của số a và b là bao nhiêu byte?
- Nếu b đang được biểu diễn dưới dạng số bù 1 thì giá trị trong hệ thập phân của b là bao nhiêu?
- Giải lại câu d với dạng số bù 2.
- Hãy đoán ý nghĩa của số a .

3.2. Embedded C

(Mỗi bài giải sẽ là 1 file source code .c. Yêu cầu chụp lại kết quả trên terminal của mỗi bài tập và đưa vào file *Pre_AfterC24_Report.pdf* ở Phần 4. **Yêu cầu bài nộp**).

Bài 1: Viết chương trình khởi tạo mảng gồm n phần tử kiểu `int` (với n nhập từ bàn phím), với điều kiện $0 < n \leq 16$, nếu n không thỏa điều kiện thì yêu cầu nhập lại n cho đến khi n thỏa điều kiện.

- Thực hiện nhập giá trị cho từng phần tử từ bàn phím.
- Sau đó in ra terminal giá trị của từng phần tử đã nhập
- Sau đó in ra terminal địa chỉ của từng phần tử trong mảng đó.

Ví dụ: sau khi run code, nhập thử 2 giá trị n không thỏa điều kiện, sau đó nhập n thỏa điều kiện, nhập giá trị từng phần tử, terminal sẽ ở như sau:

```
Nhap so phan tu n = -4
Nhap so phan tu n = 17
Nhap so phan tu n = 5
Khoi tao mang int arr[5]
Nhap gia tri tung phan tu:
arr[0] = 3
arr[1] = -5
arr[2] = 2
arr[3] = 16
arr[4] = 22
arr[] = { 3 -5 2 16 22 }
Dia chi cua tung phan tu:
&arr[0] = 6422220
&arr[1] = 6422224
&arr[2] = 6422228
&arr[3] = 6422232
&arr[4] = 6422236
```

Bài 2: Ở câu 1, dựa vào địa chỉ được in ra, nhận xét một phần tử của mảng có dung lượng là bao nhiêu byte? Lần lượt định nghĩa lại mảng với kiểu dữ liệu sau: `uint8_t`, `uint16_t`, `uint32_t`, `uint64_t` (phải include thêm thư viện `stdint.h`) và nhận xét dung lượng của từng phần tử sau mỗi lần định nghĩa lại.

Bài 3: Sử dụng lại câu 1 với điều kiện $n > 5$, tìm phần tử lớn nhất và bé nhất trong mảng và in ra terminal.

Bài 4: Sử dụng lại câu 1 với điều kiện $n > 5$, tìm giá trị trung bình của mảng và in ra terminal kết quả của giá trị trung bình tìm được ở dạng số thập phân có phần thập phân chứa 3 số.

Bài 5: Định nghĩa một struct có tên là `'infor'` có 3 trường sau: `'name'` (là chuỗi tối đa 32 phần tử), `'mssv'` (là chuỗi có 8 phần tử), `'course_c'` (kiểu dữ liệu `uint8_t`).

- Viết hàm `infor_input(struct infor* stu_infor)` yêu cầu nhập từ bàn phím giá trị các trường trong struct đã truyền vào hàm:
 - `stu_infor` → `name`: nhập vào [tên][họ] của người code. Ví dụ: GptTran
 - `stu_infor` → `mssv`: nhập vào mssv của người code. Ví dụ: 2312345
 - `stu_infor` → `coure_c`: nhập số thứ tự khóa C đã tham gia. Ví dụ: 24
- Viết hàm `infor_print(struct infor* stu_infor)` in ra terminal giá trị của 3 trường được chứa trong struct đã truyền vào hàm.

3.3. Coding convention

Dựa vào file tutorial ở Phần **2.3. Coding convention**, thực hiện sửa lại source code project C24 (file `main.c` và các thư viện đã add) theo đúng các chuẩn được liệt kê trong Phần **2.3. Coding convention**. (*Upload toàn bộ project theo Phần 4. Hình thức bài nộp*).

3.4. Altium Desinger

Tải phần mềm Altium Designer phiên bản 22.10.1 bằng tài khoản sinh viên theo hướng dẫn tại mục **3.3. Altium Designer**. (*Chụp lại minh chứng đã tải và đưa vào file báo cáo*).

3.5. Git – Github

Dựa vào phần tutorial ở Phần **2.4. Git - Github**, thực hiện tạo repository và up các file cần nộp được ghi trong Phần **4. Yêu cầu bài nộp**.

4. Hình thức bài nộp

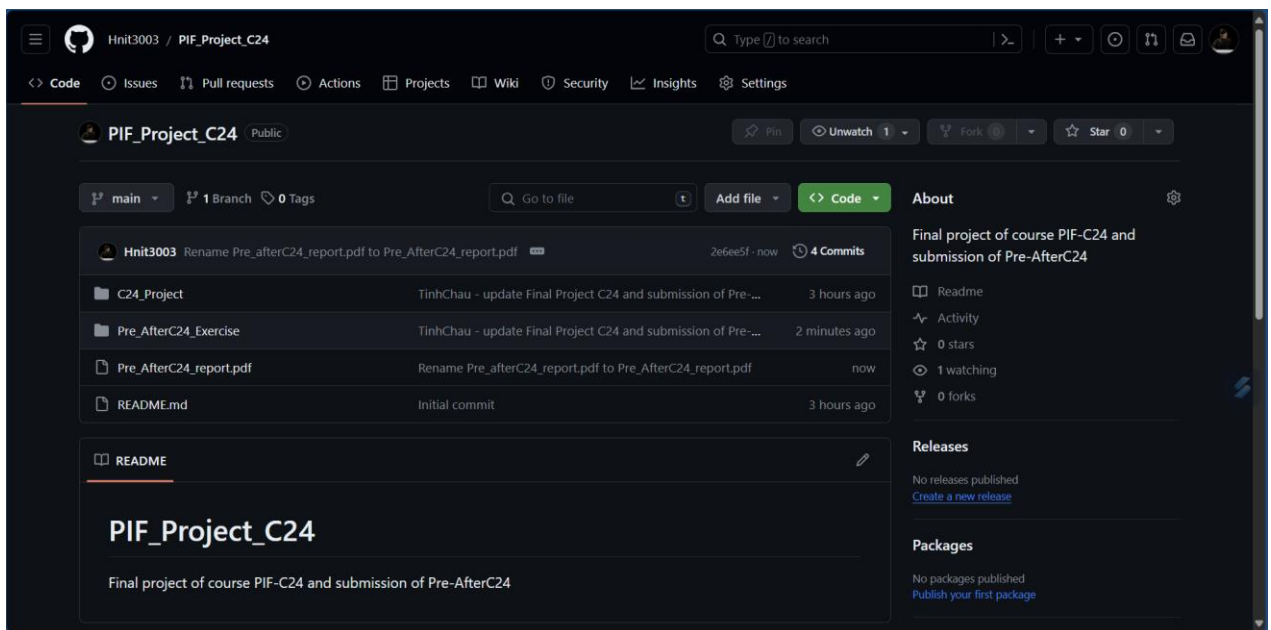
4.1. Các file và folder cần nộp

- **C24_Project**: đây là folder chứa toàn bộ project cuối khóa được generate từ STM32IDE.
- **Pre_AfterC24_Exercise**: đây là folder chứa toàn bộ file source code giải bài tập phần Embedded C.
- **Pre_AfterC24_Report.pdf**: đây là file PDF chứa phần giải bài tập của Phần 3.1. **Hệ thống số**, hình ảnh kết quả Terminal của Phần 3.2. **Embedded C**, hình ảnh yêu cầu của phần 3.4. **Altium Designer** và Phần 3.5. **Git – Github**

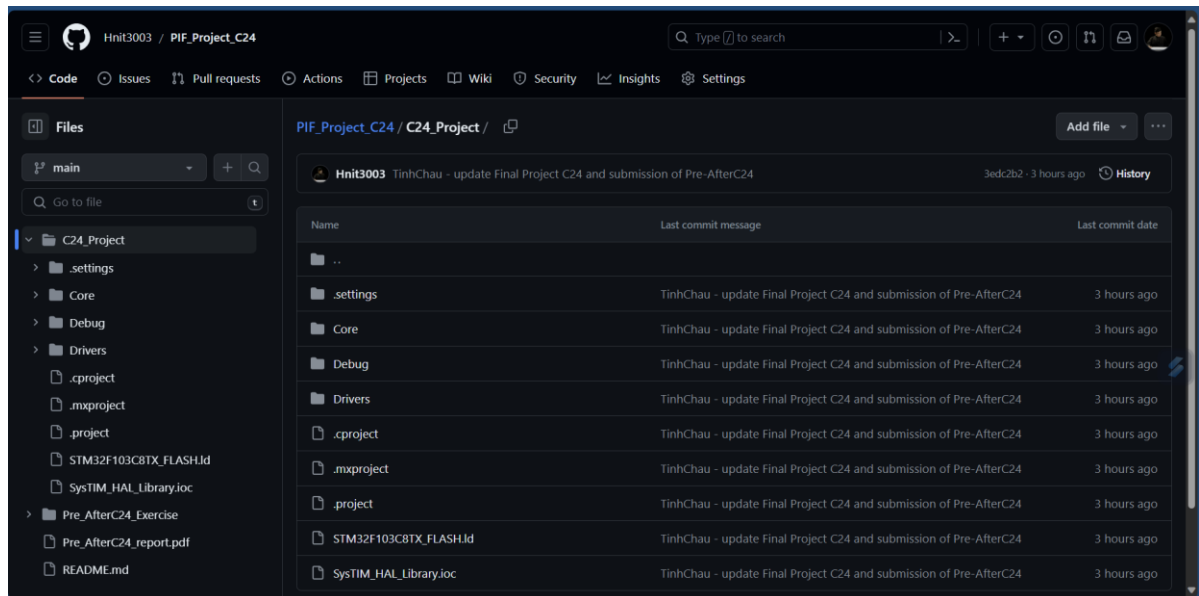
4.2. Hướng dẫn nộp bài

Toàn bộ các file và folder ở Phần 4.1. **Các file và folder cần nộp** phải được push lên repository dựa theo file tutorial trong phần 2.5. **Git – Github**.

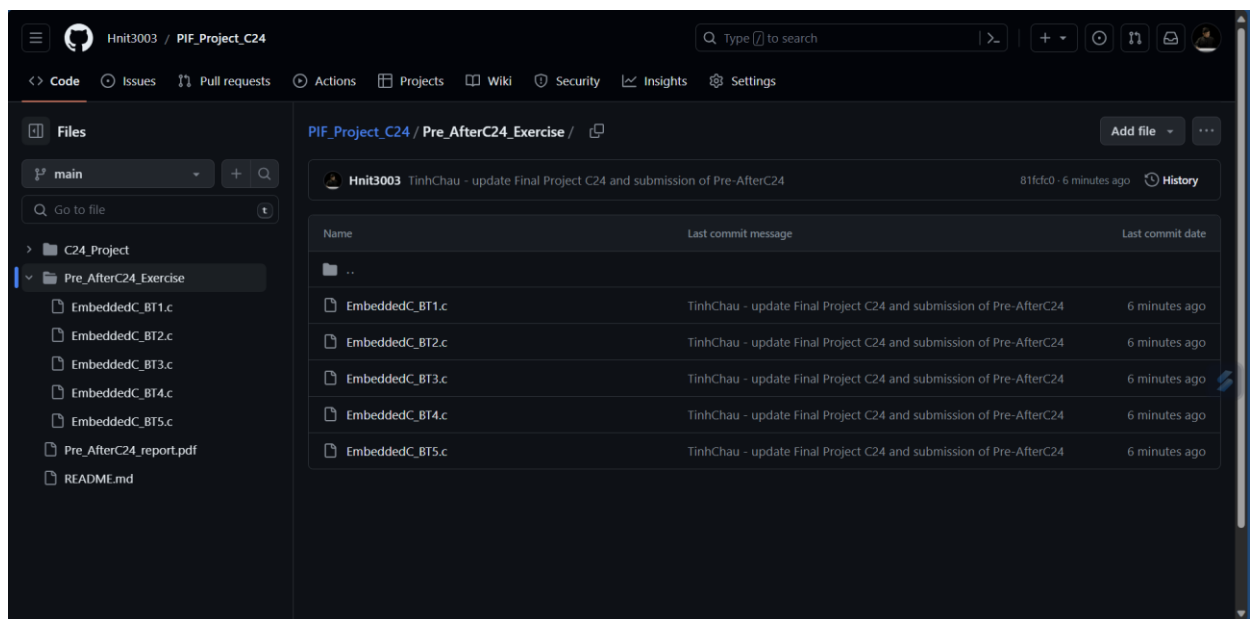
Repository sẽ có dạng như bên dưới:



*Folder **C24_Project** chứa toàn bộ project cuối khóa được generate từ STM32IDE.*



Folder Pre_AfterC24_Exercise chứa toàn bộ file source code giải bài tập phần Embedded C.



Sau khi hoàn thành tất cả các bước trên, copy đường dẫn đến repository và điền vào form sau: <https://forms.gle/ANTGPPVvmHonqhBv8>.

TÀI LIỆU THAM KHẢO

[1] Lê Thị Kim Anh, *Bài giảng môn Kỹ Thuật Số*.