Pathwise
https://www.pathwise.io.vn/roadmaps/c-basics

Quét để xem sơ đồ trực tuyến

PROGRAMMING

Lập trình C cơ bản: Từ con số 0 đến dự án thực chiến

Lộ trình nhập môn C theo C17: cài compiler, cú pháp, điều kiện, vòng lặp, hàm, mảng, chuỗi, con trỏ, bộ nhớ động, struct và đọc ghi file; có lab, kiểm thử, bảo mật và dự án CLI.

Cơ bản · 8 giai đoạn · 40 chủ đề · Cập nhật 04/11/2026

Tài liệu nguồn chính: https://cvw.cac.cornell.edu/Cintro

MỤC TIÊU

- Tự cài đặt môi trường biên dịch và phân biệt lỗi cú pháp, liên kết, logic, chạy
- Tự xây chương trình C với điều kiện, vòng lặp, hàm và dữ liệu đầu vào hợp lệ
- Hiểu mảng, chuỗi và con trỏ; nhận diện tràn bộ đệm, con trỏ treo, cấp phát và giải phóng bộ nhớ
- Chia mã nguồn thành nhiều tệp, đọc ghi file có xử lý lỗi và kiểm tra chương trình bằng test/sanitizer
- Hoàn thiện chương trình quản lý sinh viên CLI nhiều chức năng với README và ca kiểm thử

CÁC GIAI ĐOẠN

Giai đoạn 1: Nhập môn C và công cụ

Giai đoạn 1: xây dựng năng lực qua 5 bài về nhập môn c và công cụ, kết hợp kiến thức, code và kiểm thử. (5 chủ đề)

Giai đoạn 2: Cú pháp, dữ liệu và biểu thức

Giai đoạn 2: xây dựng năng lực qua 5 bài về cú pháp, dữ liệu và biểu thức, kết hợp kiến thức, code và kiểm thử. (5 chủ đề)

Giai đoạn 3: Rẽ nhánh và vòng lặp

Giai đoạn 3: xây dựng năng lực qua 5 bài về rẽ nhánh và vòng lặp, kết hợp kiến thức, code và kiểm thử. (5 chủ đề)

Giai đoạn 4: Hàm và tổ chức mã nguồn

Giai đoạn 4: xây dựng năng lực qua 5 bài về hàm và tổ chức mã nguồn, kết hợp kiến thức, code và kiểm thử. (5 chủ đề)

Giai đoạn 5: Mảng và chuỗi trong C

Giai đoạn 5: xây dựng năng lực qua 5 bài về mảng và chuỗi trong c, kết hợp kiến thức, code và kiểm thử. (5 chủ đề)

Giai đoạn 6: Con trỏ và bộ nhớ

Giai đoạn 6: xây dựng năng lực qua 5 bài về con trỏ và bộ nhớ, kết hợp kiến thức, code và kiểm thử. (5 chủ đề)

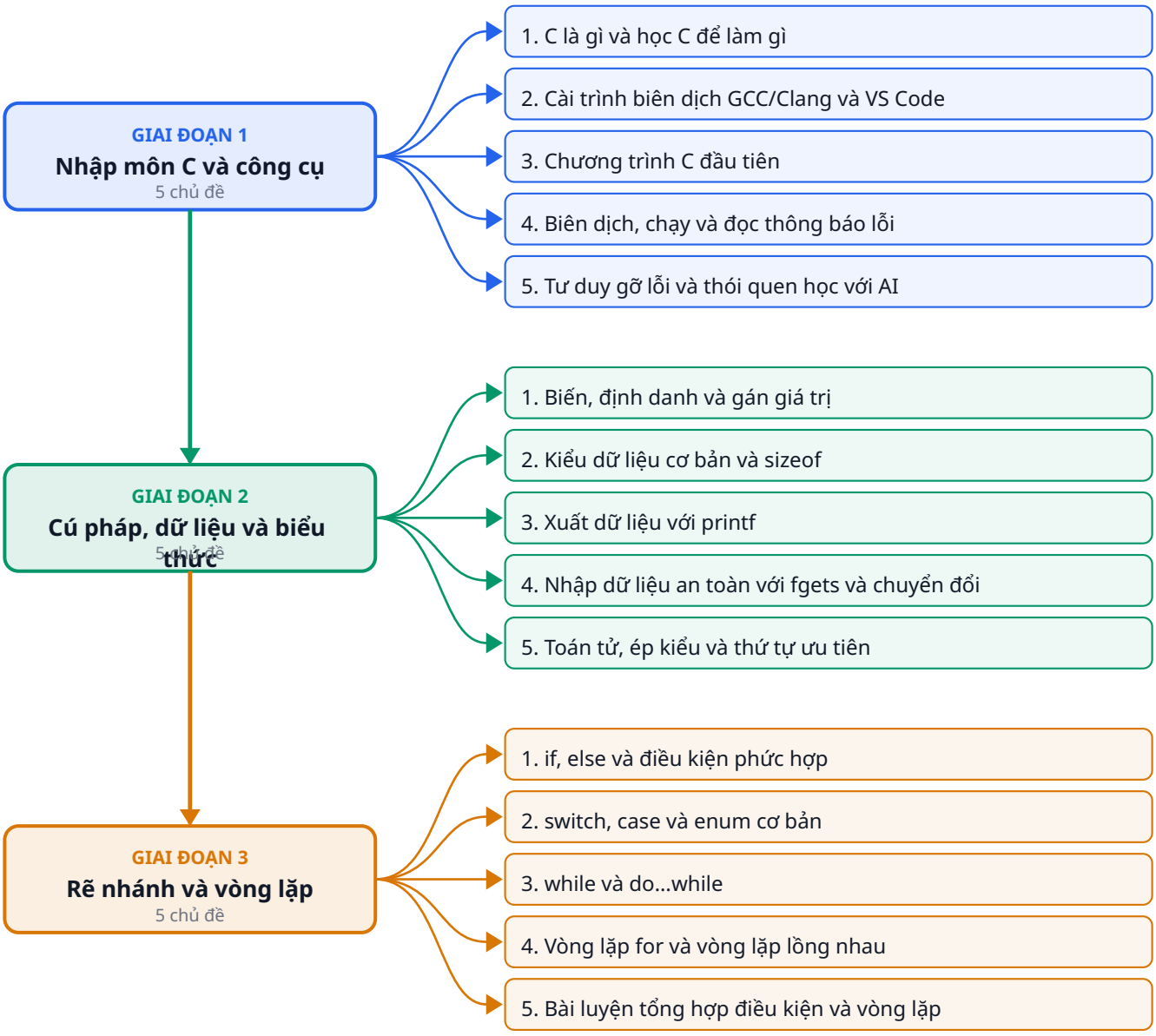
Giai đoạn 7: struct, tệp và tiền xử lý

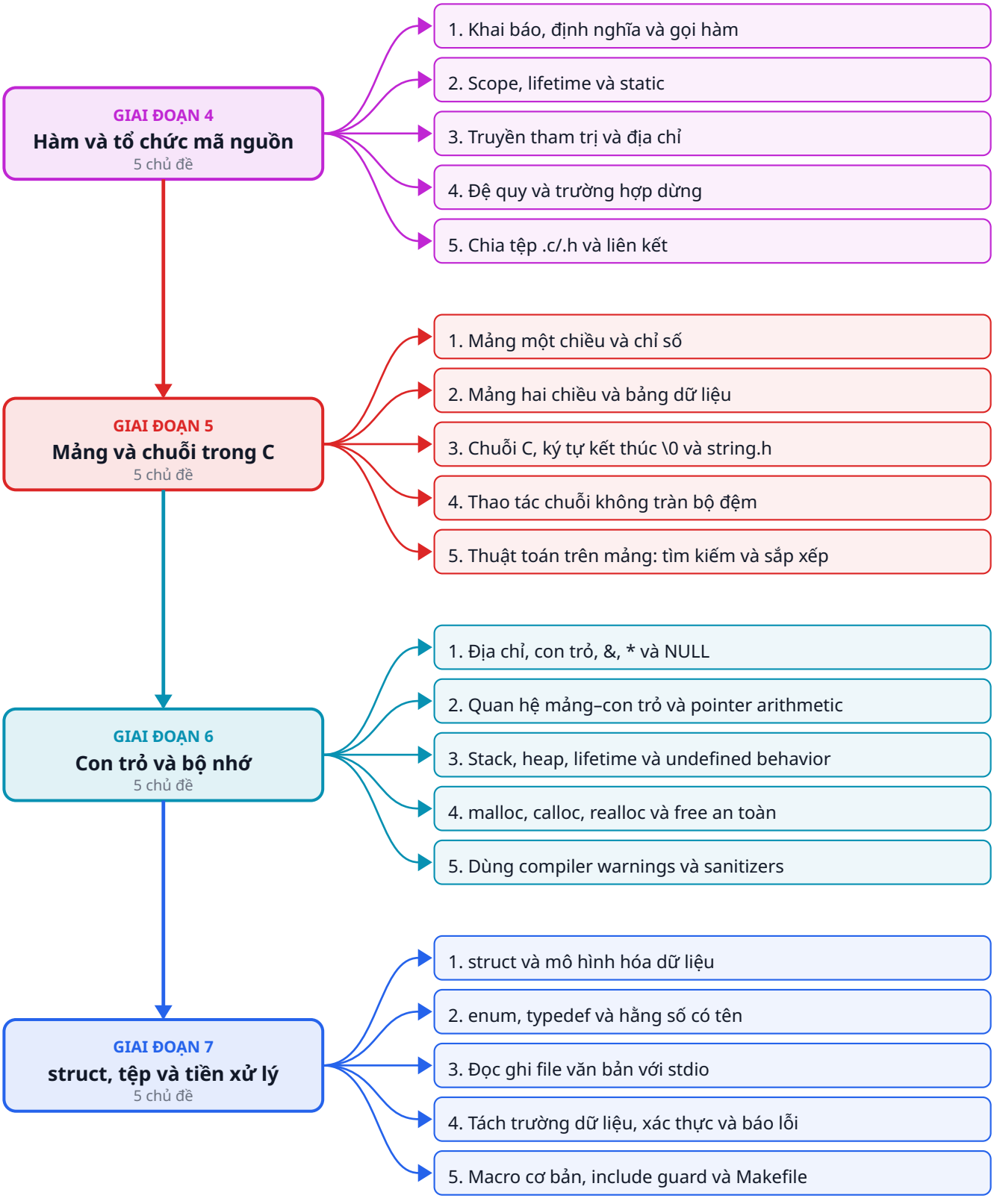
Giai đoạn 7: xây dựng năng lực qua 5 bài về struct, tệp và tiền xử lý, kết hợp kiến thức, code và kiểm thử. (5 chủ đề)

Giai đoạn 8: Thực chiến, an toàn và định hướng

Giai đoạn 8: xây dựng năng lực qua 5 bài về thực chiến, an toàn và định hướng, kết hợp kiến thức, code và kiểm thử. (5 chủ đề)

SƠ ĐỒ LỘ TRÌNH




GIAI ĐOẠN 4
Hàm và tổ chức mã nguồn
5 chủ đề
1. Khai báo, định nghĩa và gọi hàm
2. Scope, lifetime và static
3. Truyền tham trị và địa chỉ
4. Đệ quy và trường hợp dừng
5. Chia tệp .c/.h và liên kết
GIAI ĐOẠN 5
Mảng và chuỗi trong C
5 chủ đề
1. Mảng một chiều và chỉ số
2. Mảng hai chiều và bảng dữ liệu
3. Chuỗi C, ký tự kết thúc \0 và string.h
4. Thao tác chuỗi không tràn bộ đệm
5. Thuật toán trên mảng: tìm kiếm và sắp xếp
GIAI ĐOẠN 6
Con trỏ và bộ nhớ
5 chủ đề
1. Địa chỉ, con trỏ, &, * và NULL
2. Quan hệ mảng–con trỏ và pointer arithmetic
3. Stack, heap, lifetime và undefined behavior
4. malloc, calloc, realloc và free an toàn
5. Dùng compiler warnings và sanitizers
GIAI ĐOẠN 7
struct, tệp và tiền xử lý
5 chủ đề
1. struct và mô hình hóa dữ liệu
2. enum, typedef và hằng số có tên
3. Đọc ghi file văn bản với stdio
4. Tách trường dữ liệu, xác thực và báo lỗi
5. Macro cơ bản, include guard và Makefile


GIAI ĐOẠN 8
Thực chiến, an toàn và định hướng
5 chủ đề
1. Viết test và kiểm tra ca biên
2. Tư duy thuật toán và Big O nền tảng
3. Lỗi an toàn bộ nhớ và lập trình phòng thủ
4. Dự án cuối khóa: quản lý sinh viên CLI
5. Tổng kết, Git, AI và hướng học tiếp

GIAI ĐOẠN 1

Nhập môn C và công cụ

Giai đoạn 1: xây dựng năng lực qua 5 bài về nhập môn c và công cụ, kết hợp kiến thức, code và kiểm thử.

1. C là gì và học C để làm gì

C là ngôn ngữ biên dịch, gần hệ thống, phù hợp học cách chương trình dùng bộ nhớ; phân biệt C với C++ và Python.

Mục tiêu học tập

- Giải thích được: C là ngôn ngữ biên dịch, gần hệ thống, phù hợp học cách chương trình dùng bộ nhớ; phân biệt C với C++ và Python.
- Thực hiện độc lập: Viết sơ đồ đường đi từ mã .c tới chương trình chạy và liệt kê ba ứng dụng phù hợp với C.
- Dự đoán, chạy và giải thích kết quả của ví dụ; nhận diện lỗi biên dịch và lỗi logic.
- Phòng tránh lỗi điển hình: Đừng nghĩ C có quản lý bộ nhớ tự động hoặc C và C++ là một ngôn ngữ.

Kiến thức cần học

compiler · source code · executable · portability

Giải thích chi tiết

Mục tiêu của bài

C là ngôn ngữ biên dịch, gần hệ thống, phù hợp học cách chương trình dùng bộ nhớ; phân biệt C với C++ và Python.

Kiến thức cốt lõi

- Bản chất: C là ngôn ngữ biên dịch, gần hệ thống, phù hợp học cách chương trình dùng bộ nhớ; phân biệt C với C++ và Python.
- Cần nắm: compiler, source code, executable, portability. Giải thích được từng thuật ngữ bằng ví dụ nhỏ, thay vì chỉ học thuộc cú pháp.
- Cách tư duy: Xác định dữ liệu đầu vào, phép biến đổi, đầu ra và những điều kiện có thể làm chương trình sai.
- Chất lượng: Ưu tiên code rõ ràng, tên dễ hiểu, cảnh báo biên dịch sạch và kiểm tra các trường hợp biên.

Ví dụ C / lệnh minh họa

```
// C source file: hello.c
```

Chú ý: Một số ví dụ là đoạn minh họa để đặt trong hàm hoặc tệp phù hợp, không phải mọi đoạn đều là chương trình hoàn chỉnh tự chạy.

Thực hành từng bước

1. Đọc ví dụ và dự đoán đầu ra hoặc trạng thái trước khi chạy.
2. Tạo một file bài tập riêng, tự gõ code và biên dịch với -std=c17 -Wall -Wextra -Wpedantic khi thích hợp.
3. Bài tập chính: Viết sơ đồ đường đi từ mã .c tới chương trình chạy và liệt kê ba ứng dụng phù hợp với C.

4. Thử ít nhất ba tình huống: đầu vào điển hình, ca biên và đầu vào sai nếu chương trình có nhập liệu.
5. Ghi lại warning/lỗi gặp phải, giải thích nguyên nhân và cách sửa bằng lời của mình.

Làm việc với AI đúng cách

- Có thể nhờ AI: yêu cầu giải thích khái niệm, đặt câu hỏi gợi mở, đề xuất test case và đọc warning đã loại bỏ thông tin nhạy cảm.
- Phải tự kiểm chứng: tự chạy code, đối chiếu với tài liệu, đọc từng dòng thay đổi, xác định giới hạn bộ nhớ và đầu vào trước khi dùng.
- Prompt mẫu: "Tôi đang học bài C là gì và học C để làm gì. Hãy đóng vai gia sư C17, đưa ra 3 gợi ý tăng dần cho bài tập: Viết sơ đồ đường đi từ mã .c tới chương trình chạy và liệt kê ba ứng dụng phù hợp với C. Không đưa lời giải hoàn chỉnh trước khi tôi thử; sau đó hãy review code của tôi về tính đúng, cảnh báo, ca biên và an toàn bộ nhớ."

Lỗi thường gặp

Đừng nghĩ C có quản lý bộ nhớ tự động hoặc C và C++ là một ngôn ngữ.

Điều kiện hoàn thành

Tự giải thích được nội dung bài; hoàn tất bài tập chính; có code minh họa chạy được trong bối cảnh phù hợp; chứng minh đã thử trường hợp biên và sửa các warning quan trọng.

Hiểu đúng về C hiện đại

Trong khóa này, các bài được viết theo C17 để môi trường học nhất quán và phổ biến; C23 được giới thiệu như hướng cập nhật, không yêu cầu người học có compiler hỗ trợ đầy đủ. Khi ví dụ phụ thuộc hệ điều hành, trình biên dịch hoặc phần mở rộng, phải nhận biết điều đó trước khi dùng trong dự án khác.

Ví dụ minh họa

C là gì và học C để làm gì

```
// C source file: hello.c
```

C là ngôn ngữ biên dịch, gần hệ thống, phù hợp học cách chương trình dùng bộ nhớ; phân biệt C với C++ và Python.

Checklist

- [ ] Tôi có thể giải thích ý nghĩa của bài bằng ngôn ngữ của mình.
- [ ] Đã thực hành: Viết sơ đồ đường đi từ mã .c tới chương trình chạy và liệt kê ba ứng dụng phù hợp với C.
- [ ] Đã thử ca thông thường và ca biên; ghi rõ kết quả.
- [ ] Đã đọc warning compiler và xử lý lỗi phù hợp.
- [ ] Đã kiểm tra lỗi: Đừng nghĩ C có quản lý bộ nhớ tự động hoặc C và C++ là một ngôn ngữ.
- [ ] Đã tự xem và hiểu code, không chỉ sao chép lời giải AI.

Tài liệu tham khảo

C language reference — cppreference Tài liệu
https://www.cppreference.com/c/language

Cornell — Introduction to C Programming Khóa học
https://cvw.cac.cornell.edu/Cintro

GCC Warning Options Tài liệu
https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html

GCC Instrumentation Options Tài liệu
https://gcc.gnu.org/onlinedocs/gcc/Instrumentation-Options.html

2. Cài trình biên dịch GCC/Clang và VS Code

Phân biệt editor, compiler, linker, debugger; thiết lập công cụ và xác minh phiên bản.

Mục tiêu học tập

- Giải thích được: Phân biệt editor, compiler, linker, debugger; thiết lập công cụ và xác minh phiên bản.
- Thực hiện độc lập: Cài GCC hoặc Clang, chạy --version, tạo thư mục c-lab và biên dịch file đầu tiên.
- Dự đoán, chạy và giải thích kết quả của ví dụ; nhận diện lỗi biên dịch và lỗi logic.
- Phòng tránh lỗi điển hình: Không tải compiler không rõ nguồn; không nhầm extension VS Code là compiler.

Kiến thức cần học

gcc · clang · IDE · terminal · linker

Giải thích chi tiết

Mục tiêu của bài

Phân biệt editor, compiler, linker, debugger; thiết lập công cụ và xác minh phiên bản.

Kiến thức cốt lõi

- Bản chất: Phân biệt editor, compiler, linker, debugger; thiết lập công cụ và xác minh phiên bản.
- Cần nắm: gcc, clang, IDE, terminal, linker. Giải thích được từng thuật ngữ bằng ví dụ nhỏ, thay vì chỉ học thuộc cú pháp.
- Cách tư duy: Xác định dữ liệu đầu vào, phép biến đổi, đầu ra và những điều kiện có thể làm chương trình sai.
- Chất lượng: Ưu tiên code rõ ràng, tên dễ hiểu, cảnh báo biên dịch sạch và kiểm tra các trường hợp biên.

Ví dụ C / lệnh minh họa

```
gcc --version
clang --version
```

Chú ý: Một số ví dụ là đoạn minh họa để đặt trong hàm hoặc tệp phù hợp, không phải mọi đoạn đều là chương trình hoàn chỉnh tự chạy.

Thực hành từng bước

1. Đọc ví dụ và dự đoán đầu ra hoặc trạng thái trước khi chạy.
2. Tạo một file bài tập riêng, tự gõ code và biên dịch với -std=c17 -Wall -Wextra -Wpedantic khi thích hợp.
3. Bài tập chính: Cài GCC hoặc Clang, chạy --version, tạo thư mục c-lab và biên dịch file đầu tiên.
4. Thử ít nhất ba tình huống: đầu vào điển hình, ca biên và đầu vào sai nếu chương trình có nhập liệu.
5. Ghi lại warning/lỗi gặp phải, giải thích nguyên nhân và cách sửa bằng lời của mình.

Làm việc với AI đúng cách

- Có thể nhờ AI: yêu cầu giải thích khái niệm, đặt câu hỏi gợi mở, đề xuất test case và đọc warning đã loại bỏ thông tin nhạy cảm.
- Phải tự kiểm chứng: tự chạy code, đối chiếu với tài liệu, đọc từng dòng thay đổi, xác định giới hạn bộ nhớ và đầu vào trước khi dùng.

- Prompt mẫu: "Tôi đang học bài Cài trình biên dịch GCC/Clang và VS Code. Hãy đóng vai gia sư C17, đưa ra 3 gợi ý tăng dần cho bài tập: Cài GCC hoặc Clang, chạy --version, tạo thư mục c-lab và biên dịch file đầu tiên. Không đưa lời giải hoàn chỉnh trước khi tôi thử; sau đó hãy review code của tôi về tính đúng, cảnh báo, ca biên và an toàn bộ nhớ."

Lỗi thường gặp

Không tải compiler không rõ nguồn; không nhầm extension VS Code là compiler.

Điều kiện hoàn thành

Tự giải thích được nội dung bài; hoàn tất bài tập chính; có code minh họa chạy được trong bối cảnh phù hợp; chứng minh đã thử trường hợp biên và sửa các warning quan trọng.

Hiểu đúng về C hiện đại

Trong khóa này, các bài được viết theo C17 để môi trường học nhất quán và phổ biến; C23 được giới thiệu như hướng cập nhật, không yêu cầu người học có compiler hỗ trợ đầy đủ. Khi ví dụ phụ thuộc hệ điều hành, trình biên dịch hoặc phần mở rộng, phải nhận biết điều đó trước khi dùng trong dự án khác.

Ví dụ minh họa

Cài trình biên dịch GCC/Clang và VS Code

```
gcc --version
clang --version
```

Phân biệt editor, compiler, linker, debugger; thiết lập công cụ và xác minh phiên bản.

Checklist

[ ] Tôi có thể giải thích ý nghĩa của bài bằng ngôn ngữ của mình.

[ ] Đã thực hành: Cài GCC hoặc Clang, chạy --version, tạo thư mục c-lab và biên dịch file đầu tiên.

[ ] Đã thử ca thông thường và ca biên; ghi rõ kết quả.

[ ] Đã đọc warning compiler và xử lý lỗi phù hợp.

[ ] Đã kiểm tra lỗi: Không tải compiler không rõ nguồn; không nhầm extension VS Code là compiler.

[ ] Đã tự xem và hiểu code, không chỉ sao chép lời giải AI.

Tài liệu tham khảo

C language reference — cppreference Tài liệu
https://www.cppreference.com/c/language

Cornell — Introduction to C Programming Khóa học
https://cvw.cac.cornell.edu/Cintro

GCC Warning Options Tài liệu
https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html

GCC Instrumentation Options Tài liệu
https://gcc.gnu.org/onlinedocs/gcc/Instrumentation-Options.html

Nên học trước

C là gì và học C để làm gì

---

3. Chương trình C đầu tiên

Hiểu #include, hàm main, khối lệnh, dấu chấm phẩy, printf và return.

Mục tiêu học tập

- Giải thích được: Hiểu #include, hàm main, khối lệnh, dấu chấm phẩy, printf và return.
- Thực hiện độc lập: Tạo hello.c, biên dịch với -Wall -Wextra -Wpedantic và chỉnh nội dung in ra.

- Dự đoán, chạy và giải thích kết quả của ví dụ; nhận diện lỗi biên dịch và lỗi logic.
- Phòng tránh lỗi điển hình: Không dùng void main; chú ý dấu chấm phẩy và dấu ngoặc kép.

Kiến thức cần học

stdio.h · main · printf · return · header

Giải thích chi tiết

Mục tiêu của bài

Hiểu #include, hàm main, khối lệnh, dấu chấm phẩy, printf và return.

Kiến thức cốt lõi

- Bản chất: Hiểu #include, hàm main, khối lệnh, dấu chấm phẩy, printf và return.
- Cần nắm: stdio.h, main, printf, return, header. Giải thích được từng thuật ngữ bằng ví dụ nhỏ, thay vì chỉ học thuộc cú pháp.
- Cách tư duy: Xác định dữ liệu đầu vào, phép biến đổi, đầu ra và những điều kiện có thể làm chương trình sai.
- Chất lượng: Ưu tiên code rõ ràng, tên dễ hiểu, cảnh báo biên dịch sạch và kiểm tra các trường hợp biên.

Ví dụ C / lệnh minh họa

```
#include <stdio.h>
int main(void) {
    printf("Xin chao C!\n");
    return 0;
}
```

Chú ý: Một số ví dụ là đoạn minh họa để đặt trong hàm hoặc tệp phù hợp, không phải mọi đoạn đều là chương trình hoàn chỉnh tự chạy.

Thực hành từng bước

1. Đọc ví dụ và dự đoán đầu ra hoặc trạng thái trước khi chạy.
2. Tạo một file bài tập riêng, tự gõ code và biên dịch với -std=c17 -Wall -Wextra -Wpedantic khi thích hợp.
3. Bài tập chính: Tạo hello.c, biên dịch với -Wall -Wextra -Wpedantic và chỉnh nội dung in ra.
4. Thử ít nhất ba tình huống: đầu vào điển hình, ca biên và đầu vào sai nếu chương trình có nhập liệu.
5. Ghi lại warning/lỗi gặp phải, giải thích nguyên nhân và cách sửa bằng lời của mình.

Làm việc với AI đúng cách

- Có thể nhờ AI: yêu cầu giải thích khái niệm, đặt câu hỏi gợi mở, đề xuất test case và đọc warning đã loại bỏ thông tin nhạy cảm.
- Phải tự kiểm chứng: tự chạy code, đối chiếu với tài liệu, đọc từng dòng thay đổi, xác định giới hạn bộ nhớ và đầu vào trước khi dùng.
- Prompt mẫu: "Tôi đang học bài Chương trình C đầu tiên. Hãy đóng vai gia sư C17, đưa ra 3 gợi ý tăng dần cho bài tập: Tạo hello.c, biên dịch với -Wall -Wextra -Wpedantic và chỉnh nội dung in ra. Không đưa lời giải hoàn chỉnh trước khi tôi thử; sau đó hãy review code của tôi về tính đúng, cảnh báo, ca biên và an toàn bộ nhớ."

Lỗi thường gặp

Không dùng void main; chú ý dấu chấm phẩy và dấu ngoặc kép.

Điều kiện hoàn thành

Tự giải thích được nội dung bài; hoàn tất bài tập chính; có code minh họa chạy được trong bối cảnh phù hợp; chứng minh đã thử trường hợp biên và sửa các warning quan trọng.

Ví dụ minh họa

Chương trình C đầu tiên

```
#include <stdio.h>
int main(void) {
    printf("Xin chao C!\n");
    return 0;
}
```

Hiểu #include, hàm main, khối lệnh, dấu chấm phẩy, printf và return.

Checklist

[ ] Tôi có thể giải thích ý nghĩa của bài bằng ngôn ngữ của mình.

[ ] Đã thực hành: Tạo hello.c, biên dịch với -Wall -Wextra -Wpedantic và chỉnh nội dung in ra.

[ ] Đã thử ca thông thường và ca biên; ghi rõ kết quả.

[ ] Đã đọc warning compiler và xử lý lỗi phù hợp.

[ ] Đã kiểm tra lỗi: Không dùng void main; chú ý dấu chấm phẩy và dấu ngoặc kép.

[ ] Đã tự xem và hiểu code, không chỉ sao chép lời giải AI.

Tài liệu tham khảo

C language reference — cppreference Tài liệu
https://www.cppreference.com/c/language

Cornell — Introduction to C Programming Khóa học
https://cvw.cac.cornell.edu/Cintro

GCC Warning Options Tài liệu
https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html

GCC Instrumentation Options Tài liệu
https://gcc.gnu.org/onlinedocs/gcc/Instrumentation-Options.html

Nên học trước

Cài trình biên dịch GCC/Clang và VS Code

---

4. Biên dịch, chạy và đọc thông báo lỗi

Nắm các giai đoạn tiền xử lý, biên dịch, liên kết và lỗi xảy ra ở mỗi giai đoạn.

Mục tiêu học tập

- Giải thích được: Nắm các giai đoạn tiền xử lý, biên dịch, liên kết và lỗi xảy ra ở mỗi giai đoạn.
- Thực hiện độc lập: Cố ý tạo lỗi thiếu ;, tên hàm sai, thiếu thư viện; sửa dựa trên thông báo compiler.
- Dự đoán, chạy và giải thích kết quả của ví dụ; nhận diện lỗi biên dịch và lỗi logic.
- Phòng tránh lỗi điển hình: Không bỏ qua warning; đừng chạy binary cũ sau lần biên dịch thất bại.

Kiến thức cần học

warnings · errors · linking · exit-code

Giải thích chi tiết

Mục tiêu của bài

Nắm các giai đoạn tiền xử lý, biên dịch, liên kết và lỗi xảy ra ở mỗi giai đoạn.

Kiến thức cốt lõi

- Bản chất: Nắm các giai đoạn tiền xử lý, biên dịch, liên kết và lỗi xảy ra ở mỗi giai đoạn.
- Cần nắm: warnings, errors, linking, exit-code. Giải thích được từng thuật ngữ bằng ví dụ nhỏ, thay vì chỉ học thuộc cú pháp.
- Cách tư duy: Xác định dữ liệu đầu vào, phép biến đổi, đầu ra và những điều kiện có thể làm chương trình sai.
- Chất lượng: Ưu tiên code rõ ràng, tên dễ hiểu, cảnh báo biên dịch sạch và kiểm tra các trường hợp biên.

Ví dụ C / lệnh minh họa

```
gcc -std=c17 -Wall -Wextra -Wpedantic hello.c -o hello
./hello
```

Chú ý: Một số ví dụ là đoạn minh họa để đặt trong hàm hoặc tệp phù hợp, không phải mọi đoạn đều là chương trình hoàn chỉnh tự chạy.

Thực hành từng bước

1. Đọc ví dụ và dự đoán đầu ra hoặc trạng thái trước khi chạy.
2. Tạo một file bài tập riêng, tự gõ code và biên dịch với -std=c17 -Wall -Wextra -Wpedantic khi thích hợp.
3. Bài tập chính: Cố ý tạo lỗi thiếu ;, tên hàm sai, thiếu thư viện; sửa dựa trên thông báo compiler.
4. Thử ít nhất ba tình huống: đầu vào điển hình, ca biên và đầu vào sai nếu chương trình có nhập liệu.
5. Ghi lại warning/lỗi gặp phải, giải thích nguyên nhân và cách sửa bằng lời của mình.

Làm việc với AI đúng cách

- Có thể nhờ AI: yêu cầu giải thích khái niệm, đặt câu hỏi gợi mở, đề xuất test case và đọc warning đã loại bỏ thông tin nhạy cảm.
- Phải tự kiểm chứng: tự chạy code, đối chiếu với tài liệu, đọc từng dòng thay đổi, xác định giới hạn bộ nhớ và đầu vào trước khi dùng.
- Prompt mẫu: "Tôi đang học bài Biên dịch, chạy và đọc thông báo lỗi. Hãy đóng vai gia sư C17, đưa ra 3 gợi ý tăng dần cho bài tập: Cố ý tạo lỗi thiếu ;, tên hàm sai, thiếu thư viện; sửa dựa trên thông báo compiler. Không đưa lời giải hoàn chỉnh trước khi tôi thử; sau đó hãy review code của tôi về tính đúng, cảnh báo, ca biên và an toàn bộ nhớ."

Lỗi thường gặp

Không bỏ qua warning; đừng chạy binary cũ sau lần biên dịch thất bại.

Điều kiện hoàn thành

Tự giải thích được nội dung bài; hoàn tất bài tập chính; có code minh họa chạy được trong bối cảnh phù hợp; chứng minh đã thử trường hợp biên và sửa các warning quan trọng.

Ví dụ minh họa

Biên dịch, chạy và đọc thông báo lỗi

```
gcc -std=c17 -Wall -Wextra -Wpedantic hello.c -o hello
./hello
```

Nắm các giai đoạn tiền xử lý, biên dịch, liên kết và lỗi xảy ra ở mỗi giai đoạn.

Checklist

[ ] Tôi có thể giải thích ý nghĩa của bài bằng ngôn ngữ của mình.

[ ] Đã thực hành: Cố ý tạo lỗi thiếu ;, tên hàm sai, thiếu thư viện; sửa dựa trên thông báo compiler.

[ ] Đã thử ca thông thường và ca biên; ghi rõ kết quả.

[ ] Đã đọc warning compiler và xử lý lỗi phù hợp.

[ ] Đã kiểm tra lỗi: Không bỏ qua warning; đừng chạy binary cũ sau lần biên dịch thất bại.

[ ] Đã tự xem và hiểu code, không chỉ sao chép lời giải AI.

Tài liệu tham khảo

C language reference — cppreference Tài liệu
https://www.cppreference.com/c/language

Cornell — Introduction to C Programming Khóa học
https://cvw.cac.cornell.edu/Cintro

GCC Warning Options Tài liệu
https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html

GCC Instrumentation Options Tài liệu
https://gcc.gnu.org/onlinedocs/gcc/Instrumentation-Options.html

Nên học trước

Chương trình C đầu tiên

---

5. Tư duy gỡ lỗi và thói quen học với AI

Phân biệt syntax, runtime và logic error; dùng print/debugger, tái hiện lỗi, đặt giả thuyết.

Mục tiêu học tập

- Giải thích được: Phân biệt syntax, runtime và logic error; dùng print/debugger, tái hiện lỗi, đặt giả thuyết.
- Thực hiện độc lập: Ghi một bug report: đầu vào, kỳ vọng, kết quả thực tế, lệnh tái hiện và cách sửa.
- Dự đoán, chạy và giải thích kết quả của ví dụ; nhận diện lỗi biên dịch và lỗi logic.
- Phòng tránh lỗi điển hình: Không chấp nhận code AI chưa tự chạy; không nhập secrets, dữ liệu riêng vào chatbot.

Kiến thức cần học

debugging · breakpoint · reproducibility · AI-verification

Giải thích chi tiết

Mục tiêu của bài

Phân biệt syntax, runtime và logic error; dùng print/debugger, tái hiện lỗi, đặt giả thuyết.

Kiến thức cốt lõi

- Bản chất: Phân biệt syntax, runtime và logic error; dùng print/debugger, tái hiện lỗi, đặt giả thuyết.
- Cần nắm: debugging, breakpoint, reproducibility, AI-verification. Giải thích được từng thuật ngữ bằng ví dụ nhỏ, thay vì chỉ học thuộc cú pháp.
- Cách tư duy: Xác định dữ liệu đầu vào, phép biến đổi, đầu ra và những điều kiện có thể làm chương trình sai.
- Chất lượng: Ưu tiên code rõ ràng, tên dễ hiểu, cảnh báo biên dịch sạch và kiểm tra các trường hợp biên.

Ví dụ C / lệnh minh họa

```
gcc -std=c17 -g -O0 -Wall -Wextra hello.c -o hello
```

Chú ý: Một số ví dụ là đoạn minh họa để đặt trong hàm hoặc tệp phù hợp, không phải mọi đoạn đều là chương trình hoàn chỉnh tự chạy.

Thực hành từng bước

1. Đọc ví dụ và dự đoán đầu ra hoặc trạng thái trước khi chạy.
2. Tạo một file bài tập riêng, tự gõ code và biên dịch với -std=c17 -Wall -Wextra -Wpedantic khi thích hợp.
3. Bài tập chính: Ghi một bug report: đầu vào, kỳ vọng, kết quả thực tế, lệnh tái hiện và cách sửa.
4. Thử ít nhất ba tình huống: đầu vào điển hình, ca biên và đầu vào sai nếu chương trình có nhập liệu.
5. Ghi lại warning/lỗi gặp phải, giải thích nguyên nhân và cách sửa bằng lời của mình.

Làm việc với AI đúng cách

- Có thể nhờ AI: yêu cầu giải thích khái niệm, đặt câu hỏi gợi mở, đề xuất test case và đọc warning đã loại bỏ thông tin nhạy cảm.
- Phải tự kiểm chứng: tự chạy code, đối chiếu với tài liệu, đọc từng dòng thay đổi, xác định giới hạn bộ nhớ và đầu vào trước khi dùng.
- Prompt mẫu: "Tôi đang học bài Tư duy gỡ lỗi và thói quen học với AI. Hãy đóng vai gia sư C17, đưa ra 3 gợi ý tăng dần cho bài tập: Ghi một bug report: đầu vào, kỳ vọng, kết quả thực tế, lệnh tái hiện và cách sửa. Không đưa lời giải hoàn chỉnh trước khi tôi thử; sau đó hãy review code của tôi về tính đúng, cảnh báo, ca biên và an toàn bộ nhớ."

Lỗi thường gặp

Không chấp nhận code AI chưa tự chạy; không nhập secrets, dữ liệu riêng vào chatbot.

Điều kiện hoàn thành

Tự giải thích được nội dung bài; hoàn tất bài tập chính; có code minh họa chạy được trong bối cảnh phù hợp; chứng minh đã thử trường hợp biên và sửa các warning quan trọng.

Ví dụ minh họa

Tư duy gỡ lỗi và thói quen học với AI

```
gcc -std=c17 -g -O0 -Wall -Wextra hello.c -o hello
```

Phân biệt syntax, runtime và logic error; dùng print/debugger, tái hiện lỗi, đặt giả thuyết.

Checklist

- [ ] Tôi có thể giải thích ý nghĩa của bài bằng ngôn ngữ của mình.
- [ ] Đã thực hành: Ghi một bug report: đầu vào, kỳ vọng, kết quả thực tế, lệnh tái hiện và cách sửa.
- [ ] Đã thử ca thông thường và ca biên; ghi rõ kết quả.
- [ ] Đã đọc warning compiler và xử lý lỗi phù hợp.
- [ ] Đã kiểm tra lỗi: Không chấp nhận code AI chưa tự chạy; không nhập secrets, dữ liệu riêng vào chatbot.
- [ ] Đã tự xem và hiểu code, không chỉ sao chép lời giải AI.

Tài liệu tham khảo

C language reference — cppreference Tài liệu
https://www.cppreference.com/c/language

Cornell — Introduction to C Programming Khóa học
https://cvw.cac.cornell.edu/Cintro

GCC Warning Options Tài liệu
https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html

GCC Instrumentation Options Tài liệu
https://gcc.gnu.org/onlinedocs/gcc/Instrumentation-Options.html

Nên học trước

Biên dịch, chạy và đọc thông báo lỗi

GIAI ĐOẠN 2

Cú pháp, dữ liệu và biểu thức

Giai đoạn 2: xây dựng năng lực qua 5 bài về cú pháp, dữ liệu và biểu thức, kết hợp kiến thức, code và kiểm thử.

1. Biến, định danh và gán giá trị

Khai báo trước khi dùng, khởi tạo, phép gán, tên biến dễ hiểu và phạm vi khối.

Mục tiêu học tập

- Giải thích được: Khai báo trước khi dùng, khởi tạo, phép gán, tên biến dễ hiểu và phạm vi khối.
- Thực hiện độc lập: Viết chương trình lưu họ tên bằng mảng char ở phần sau; hiện tại tính tuổi qua năm sinh và năm hiện tại.
- Dự đoán, chạy và giải thích kết quả của ví dụ; nhận diện lỗi biên dịch và lỗi logic.
- Phòng tránh lỗi điển hình: Đừng dùng biến chưa khởi tạo; đừng nhầm = với ==.

Kiến thức cần học

variable · initialization · assignment · identifier

Giải thích chi tiết

Mục tiêu của bài

Khai báo trước khi dùng, khởi tạo, phép gán, tên biến dễ hiểu và phạm vi khối.

Kiến thức cốt lõi

- Bản chất: Khai báo trước khi dùng, khởi tạo, phép gán, tên biến dễ hiểu và phạm vi khối.
- Cần nắm: variable, initialization, assignment, identifier. Giải thích được từng thuật ngữ bằng ví dụ nhỏ, thay vì chỉ học thuộc cú pháp.
- Cách tư duy: Xác định dữ liệu đầu vào, phép biến đổi, đầu ra và những điều kiện có thể làm chương trình sai.
- Chất lượng: Ưu tiên code rõ ràng, tên dễ hiểu, cảnh báo biên dịch sạch và kiểm tra các trường hợp biên.

Ví dụ C / lệnh minh họa

```
int age = 18;
age = age + 1;
printf("%d\n", age);
```

Chú ý: Một số ví dụ là đoạn minh họa để đặt trong hàm hoặc tệp phù hợp, không phải mọi đoạn đều là chương trình hoàn chỉnh tự chạy.

Thực hành từng bước

1. Đọc ví dụ và dự đoán đầu ra hoặc trạng thái trước khi chạy.
2. Tạo một file bài tập riêng, tự gõ code và biên dịch với -std=c17 -Wall -Wextra -Wpedantic khi thích hợp.
3. Bài tập chính: Viết chương trình lưu họ tên bằng mảng char ở phần sau; hiện tại tính tuổi qua năm sinh và năm hiện tại.
4. Thử ít nhất ba tình huống: đầu vào điển hình, ca biên và đầu vào sai nếu chương trình có nhập liệu.

5. Ghi lại warning/lỗi gặp phải, giải thích nguyên nhân và cách sửa bằng lời của mình.

Làm việc với AI đúng cách

- Có thể nhờ AI: yêu cầu giải thích khái niệm, đặt câu hỏi gợi mở, đề xuất test case và đọc warning đã loại bỏ thông tin nhạy cảm.
- Phải tự kiểm chứng: tự chạy code, đối chiếu với tài liệu, đọc từng dòng thay đổi, xác định giới hạn bộ nhớ và đầu vào trước khi dùng.
- Prompt mẫu: "Tôi đang học bài Biến, định danh và gán giá trị. Hãy đóng vai gia sư C17, đưa ra 3 gợi ý tăng dần cho bài tập: Viết chương trình lưu họ tên bằng mảng char ở phần sau; hiện tại tính tuổi qua năm sinh và năm hiện tại. Không đưa lời giải hoàn chỉnh trước khi tôi thử; sau đó hãy review code của tôi về tính đúng, cảnh báo, ca biên và an toàn bộ nhớ."

Lỗi thường gặp

Đừng dùng biến chưa khởi tạo; đừng nhầm = với ==.

Điều kiện hoàn thành

Tự giải thích được nội dung bài; hoàn tất bài tập chính; có code minh họa chạy được trong bối cảnh phù hợp; chứng minh đã thử trường hợp biên và sửa các warning quan trọng.

Ví dụ minh họa

Biến, định danh và gán giá trị

```
int age = 18;
age = age + 1;
printf("%d\n", age);
```

Khai báo trước khi dùng, khởi tạo, phép gán, tên biến dễ hiểu và phạm vi khối.

Checklist

- [ ] Tôi có thể giải thích ý nghĩa của bài bằng ngôn ngữ của mình.
- [ ] Đã thực hành: Viết chương trình lưu họ tên bằng mảng char ở phần sau; hiện tại tính tuổi qua năm sinh và năm hiện tại.
- [ ] Đã thử ca thông thường và ca biên; ghi rõ kết quả.
- [ ] Đã đọc warning compiler và xử lý lỗi phù hợp.
- [ ] Đã kiểm tra lỗi: Đừng dùng biến chưa khởi tạo; đừng nhầm = với ==.
- [ ] Đã tự xem và hiểu code, không chỉ sao chép lời giải AI.

Tài liệu tham khảo

C language reference — cppreference Tài liệu
https://www.cppreference.com/c/language

Cornell — Introduction to C Programming Khóa học
https://cvw.cac.cornell.edu/Cintro

Nên học trước

Tư duy gỡ lỗi và thói quen học với AI

---

2. Kiểu dữ liệu cơ bản và sizeof

Làm quen char, int, long, float, double, _Bool/bool qua stdbool.h trong C17; kích thước phụ thuộc nền tảng.

Mục tiêu học tập

- Giải thích được: Làm quen char, int, long, float, double, _Bool/bool qua stdbool.h trong C17; kích thước phụ thuộc nền tảng.
- Thực hiện độc lập: In sizeof của các kiểu và kiểm tra giới hạn INT_MAX từ limits.h.
- Dự đoán, chạy và giải thích kết quả của ví dụ; nhận diện lỗi biên dịch và lỗi logic.
- Phòng tránh lỗi điển hình: Không giả định int luôn 4 byte hoặc float biểu diễn chính xác mọi số thập phân.

Kiến thức cần học

int · char · float · double · sizeof · limits.h

Giải thích chi tiết

Mục tiêu của bài

Làm quen char, int, long, float, double, _Bool/bool qua stdbool.h trong C17; kích thước phụ thuộc nền tảng.

Kiến thức cốt lõi

- Bản chất: Làm quen char, int, long, float, double, _Bool/bool qua stdbool.h trong C17; kích thước phụ thuộc nền tảng.
- Cần nắm: int, char, float, double, sizeof, limits.h. Giải thích được từng thuật ngữ bằng ví dụ nhỏ, thay vì chỉ học thuộc cú pháp.
- Cách tư duy: Xác định dữ liệu đầu vào, phép biến đổi, đầu ra và những điều kiện có thể làm chương trình sai.
- Chất lượng: Ưu tiên code rõ ràng, tên dễ hiểu, cảnh báo biên dịch sạch và kiểm tra các trường hợp biên.

Ví dụ C / lệnh minh họa

```
#include <limits.h>
printf("int bytes: %zu, max: %d\n", sizeof(int), INT_MAX);
```

Chú ý: Một số ví dụ là đoạn minh họa để đặt trong hàm hoặc tệp phù hợp, không phải mọi đoạn đều là chương trình hoàn chỉnh tự chạy.

Thực hành từng bước

1. Đọc ví dụ và dự đoán đầu ra hoặc trạng thái trước khi chạy.
2. Tạo một file bài tập riêng, tự gõ code và biên dịch với -std=c17 -Wall -Wextra -Wpedantic khi thích hợp.
3. Bài tập chính: In sizeof của các kiểu và kiểm tra giới hạn INT_MAX từ limits.h.
4. Thử ít nhất ba tình huống: đầu vào điển hình, ca biên và đầu vào sai nếu chương trình có nhập liệu.
5. Ghi lại warning/lỗi gặp phải, giải thích nguyên nhân và cách sửa bằng lời của mình.

Làm việc với AI đúng cách

- Có thể nhờ AI: yêu cầu giải thích khái niệm, đặt câu hỏi gợi mở, đề xuất test case và đọc warning đã loại bỏ thông tin nhạy cảm.
- Phải tự kiểm chứng: tự chạy code, đối chiếu với tài liệu, đọc từng dòng thay đổi, xác định giới hạn bộ nhớ và đầu vào trước khi dùng.

- Prompt mẫu: "Tôi đang học bài Kiểu dữ liệu cơ bản và sizeof. Hãy đóng vai gia sư C17, đưa ra 3 gợi ý tăng dần cho bài tập: In sizeof của các kiểu và kiểm tra giới hạn INT_MAX từ limits.h. Không đưa lời giải hoàn chỉnh trước khi tôi thử; sau đó hãy review code của tôi về tính đúng, cảnh báo, ca biên và an toàn bộ nhớ."

Lỗi thường gặp

Không giả định int luôn 4 byte hoặc float biểu diễn chính xác mọi số thập phân.

Điều kiện hoàn thành

Tự giải thích được nội dung bài; hoàn tất bài tập chính; có code minh họa chạy được trong bối cảnh phù hợp; chứng minh đã thử trường hợp biên và sửa các warning quan trọng.

Ví dụ minh họa

Kiểu dữ liệu cơ bản và sizeof

```
#include <limits.h>
printf("int bytes: %zu, max: %d\n", sizeof(int), INT_MAX);
```

Làm quen char, int, long, float, double, _Bool/bool qua stdbool.h trong C17; kích thước phụ thuộc nền tảng.

Checklist

[ ] Tôi có thể giải thích ý nghĩa của bài bằng ngôn ngữ của mình.
[ ] Đã thực hành: In sizeof của các kiểu và kiểm tra giới hạn INT_MAX từ limits.h.
[ ] Đã thử ca thông thường và ca biên; ghi rõ kết quả.
[ ] Đã đọc warning compiler và xử lý lỗi phù hợp.
[ ] Đã kiểm tra lỗi: Không giả định int luôn 4 byte hoặc float biểu diễn chính xác mọi số thập phân.
[ ] Đã tự xem và hiểu code, không chỉ sao chép lời giải AI.

Tài liệu tham khảo

C language reference — cppreference Tài liệu
https://www.cppreference.com/c/language

Cornell — Introduction to C Programming Khóa học
https://cvw.cac.cornell.edu/Cintro

Nên học trước

Biến, định danh và gán giá trị

---

3. Xuất dữ liệu với printf

Dùng định dạng %d, %u, %zu, %f, %c, %s đúng loại đối số; escape sequences.

Mục tiêu học tập

- Giải thích được: Dùng định dạng %d, %u, %zu, %f, %c, %s đúng loại đối số; escape sequences.
- Thực hiện độc lập: In bảng điểm với căn cột và hiển thị hai chữ số thập phân.
- Dự đoán, chạy và giải thích kết quả của ví dụ; nhận diện lỗi biên dịch và lỗi logic.
- Phòng tránh lỗi điển hình: Sai format có thể gây undefined behavior; không đưa chuỗi người dùng làm format.

Kiến thức cần học

format-specifiers · printf · escape-sequence

Giải thích chi tiết

Mục tiêu của bài

Dùng định dạng %d, %u, %zu, %f, %c, %s đúng loại đối số; escape sequences.

Kiến thức cốt lõi

- Bản chất: Dùng định dạng %d, %u, %zu, %f, %c, %s đúng loại đối số; escape sequences.
- Cần nắm: format-specifiers, printf, escape-sequence. Giải thích được từng thuật ngữ bằng ví dụ nhỏ, thay vì chỉ học thuộc cú pháp.
- Cách tư duy: Xác định dữ liệu đầu vào, phép biến đổi, đầu ra và những điều kiện có thể làm chương trình sai.
- Chất lượng: Ưu tiên code rõ ràng, tên dễ hiểu, cảnh báo biên dịch sạch và kiểm tra các trường hợp biên.

Ví dụ C / lệnh minh họa

```
double gpa = 3.45;
printf("GPA: %.2f\n", gpa);
```

Chú ý: Một số ví dụ là đoạn minh họa để đặt trong hàm hoặc tệp phù hợp, không phải mọi đoạn đều là chương trình hoàn chỉnh tự chạy.

Thực hành từng bước

1. Đọc ví dụ và dự đoán đầu ra hoặc trạng thái trước khi chạy.
2. Tạo một file bài tập riêng, tự gõ code và biên dịch với -std=c17 -Wall -Wextra -Wpedantic khi thích hợp.
3. Bài tập chính: In bảng điểm với căn cột và hiển thị hai chữ số thập phân.
4. Thử ít nhất ba tình huống: đầu vào điển hình, ca biên và đầu vào sai nếu chương trình có nhập liệu.
5. Ghi lại warning/lỗi gặp phải, giải thích nguyên nhân và cách sửa bằng lời của mình.

Làm việc với AI đúng cách

- Có thể nhờ AI: yêu cầu giải thích khái niệm, đặt câu hỏi gợi mở, đề xuất test case và đọc warning đã loại bỏ thông tin nhạy cảm.
- Phải tự kiểm chứng: tự chạy code, đối chiếu với tài liệu, đọc từng dòng thay đổi, xác định giới hạn bộ nhớ và đầu vào trước khi dùng.
- Prompt mẫu: "Tôi đang học bài Xuất dữ liệu với printf. Hãy đóng vai gia sư C17, đưa ra 3 gợi ý tăng dần cho bài tập: In bảng điểm với căn cột và hiển thị hai chữ số thập phân. Không đưa lời giải hoàn chỉnh trước khi tôi thử; sau đó hãy review code của tôi về tính đúng, cảnh báo, ca biên và an toàn bộ nhớ."

Lỗi thường gặp

Sai format có thể gây undefined behavior; không đưa chuỗi người dùng làm format.

Điều kiện hoàn thành

Tự giải thích được nội dung bài; hoàn tất bài tập chính; có code minh họa chạy được trong bối cảnh phù hợp; chứng minh đã thử trường hợp biên và sửa các warning quan trọng.

Ví dụ minh họa

Xuất dữ liệu với printf

```
double gpa = 3.45;
printf("GPA: %.2f\n", gpa);
```

Dùng định dạng %d, %u, %zu, %f, %c, %s đúng loại đối số; escape sequences.

Checklist

[ ] Tôi có thể giải thích ý nghĩa của bài bằng ngôn ngữ của mình.

[ ] Đã thực hành: In bảng điểm với căn cột và hiển thị hai chữ số thập phân.

[ ] Đã thử ca thông thường và ca biên; ghi rõ kết quả.

[ ] Đã đọc warning compiler và xử lý lỗi phù hợp.

[ ] Đã kiểm tra lỗi: Sai format có thể gây undefined behavior; không đưa chuỗi người dùng làm format.

[ ] Đã tự xem và hiểu code, không chỉ sao chép lời giải AI.

Tài liệu tham khảo

C language reference — cppreference Tài liệu
https://www.cppreference.com/c/language

Cornell — Introduction to C Programming Khóa học
https://cvw.cac.cornell.edu/Cintro

Nên học trước

Kiểu dữ liệu cơ bản và sizeof

---

4. Nhập dữ liệu an toàn với fgets và chuyển đổi

Nhận biết scanf có giới hạn; dùng fgets, strtol để đọc số, kiểm tra lỗi đầu vào.

Mục tiêu học tập

- Giải thích được: Nhận biết scanf có giới hạn; dùng fgets, strtol để đọc số, kiểm tra lỗi đầu vào.
- Thực hiện độc lập: Viết chương trình nhập tuổi: từ chối chuỗi rỗng, chữ và số âm.
- Dự đoán, chạy và giải thích kết quả của ví dụ; nhận diện lỗi biên dịch và lỗi logic.
- Phòng tránh lỗi điển hình: Không dùng gets; không dùng scanf("%s", buf) không giới hạn độ rộng.

Kiến thức cần học

fgets · strtol · stdin · validation

Giải thích chi tiết

Mục tiêu của bài

Nhận biết scanf có giới hạn; dùng fgets, strtol để đọc số, kiểm tra lỗi đầu vào.

Kiến thức cốt lõi

- Bản chất: Nhận biết scanf có giới hạn; dùng fgets, strtol để đọc số, kiểm tra lỗi đầu vào.
- Cần nắm: fgets, strtol, stdin, validation. Giải thích được từng thuật ngữ bằng ví dụ nhỏ, thay vì chỉ học thuộc cú pháp.
- Cách tư duy: Xác định dữ liệu đầu vào, phép biến đổi, đầu ra và những điều kiện có thể làm chương trình sai.
- Chất lượng: Ưu tiên code rõ ràng, tên dễ hiểu, cảnh báo biên dịch sạch và kiểm tra các trường hợp biên.

Ví dụ C / lệnh minh họa

```
char line[64];
if (fgets(line, sizeof line, stdin)) {
    char *end;
    long age = strtol(line, &end, 10);
    printf("Age: %ld\n", age);
}
```

Chú ý: Một số ví dụ là đoạn minh họa để đặt trong hàm hoặc tệp phù hợp, không phải mọi đoạn đều là chương trình hoàn chỉnh tự chạy.

Thực hành từng bước

1. Đọc ví dụ và dự đoán đầu ra hoặc trạng thái trước khi chạy.
2. Tạo một file bài tập riêng, tự gõ code và biên dịch với -std=c17 -Wall -Wextra -Wpedantic khi thích hợp.
3. Bài tập chính: Viết chương trình nhập tuổi: từ chối chuỗi rỗng, chữ và số âm.
4. Thử ít nhất ba tình huống: đầu vào điển hình, ca biên và đầu vào sai nếu chương trình có nhập liệu.
5. Ghi lại warning/lỗi gặp phải, giải thích nguyên nhân và cách sửa bằng lời của mình.

Làm việc với AI đúng cách

- Có thể nhờ AI: yêu cầu giải thích khái niệm, đặt câu hỏi gợi mở, đề xuất test case và đọc warning đã loại bỏ thông tin nhạy cảm.
- Phải tự kiểm chứng: tự chạy code, đối chiếu với tài liệu, đọc từng dòng thay đổi, xác định giới hạn bộ nhớ và đầu vào trước khi dùng.
- Prompt mẫu: "Tôi đang học bài Nhập dữ liệu an toàn với fgets và chuyển đổi. Hãy đóng vai gia sư C17, đưa ra 3 gợi ý tăng dần cho bài tập: Viết chương trình nhập tuổi: từ chối chuỗi rỗng, chữ và số âm. Không đưa lời giải hoàn chỉnh trước khi tôi thử; sau đó hãy review code của tôi về tính đúng, cảnh báo, ca biên và an toàn bộ nhớ."

Lỗi thường gặp

Không dùng gets; không dùng scanf("%s", buf) không giới hạn độ rộng.

Điều kiện hoàn thành

Tự giải thích được nội dung bài; hoàn tất bài tập chính; có code minh họa chạy được trong bối cảnh phù hợp; chứng minh đã thử trường hợp biên và sửa các warning quan trọng.

Ví dụ minh họa

Nhập dữ liệu an toàn với fgets và chuyển đổi

```
char line[64];
if (fgets(line, sizeof line, stdin)) {
    char *end;
    long age = strtol(line, &end, 10);
    printf("Age: %ld\n", age);
}
```

Nhận biết scanf có giới hạn; dùng fgets, strtol để đọc số, kiểm tra lỗi đầu vào.

Checklist

[ ] Tôi có thể giải thích ý nghĩa của bài bằng ngôn ngữ của mình.
[ ] Đã thực hành: Viết chương trình nhập tuổi: từ chối chuỗi rỗng, chữ và số âm.
[ ] Đã thử ca thông thường và ca biên; ghi rõ kết quả.
[ ] Đã đọc warning compiler và xử lý lỗi phù hợp.
[ ] Đã kiểm tra lỗi: Không dùng gets; không dùng scanf("%s", buf) không giới hạn độ rộng.
[ ] Đã tự xem và hiểu code, không chỉ sao chép lời giải AI.

Tài liệu tham khảo

C language reference — cppreference Tài liệu
https://www.cppreference.com/c/language

Cornell — Introduction to C Programming Khóa học
https://cvw.cac.cornell.edu/Cintro

Nên học trước

Xuất dữ liệu với printf

---

5. Toán tử, ép kiểu và thứ tự ưu tiên

Số nguyên chia lấy phần nguyên, %, so sánh, && || !, ++ -- và chuyển đổi kiểu.

Mục tiêu học tập

- Giải thích được: Số nguyên chia lấy phần nguyên, %, so sánh, && || !, ++ -- và chuyển đổi kiểu.
- Thực hiện độc lập: Tính BMI từ cân nặng/chiều cao; thử 5 / 2 và 5.0 / 2.
- Dự đoán, chạy và giải thích kết quả của ví dụ; nhận diện lỗi biên dịch và lỗi logic.
- Phòng tránh lỗi điển hình: Tránh biểu thức có nhiều phép ++ gây hành vi không xác định; đừng dùng == cho so sánh float tùy tiện.

Kiến thức cần học

arithmetic · logical · precedence · casting

Giải thích chi tiết

Mục tiêu của bài

Số nguyên chia lấy phần nguyên, %, so sánh, && || !, ++ -- và chuyển đổi kiểu.

Kiến thức cốt lõi

- Bản chất: Số nguyên chia lấy phần nguyên, %, so sánh, && || !, ++ -- và chuyển đổi kiểu.
- Cần nắm: arithmetic, logical, precedence, casting. Giải thích được từng thuật ngữ bằng ví dụ nhỏ, thay vì chỉ học thuộc cú pháp.
- Cách tư duy: Xác định dữ liệu đầu vào, phép biến đổi, đầu ra và những điều kiện có thể làm chương trình sai.
- Chất lượng: Ưu tiên code rõ ràng, tên dễ hiểu, cảnh báo biên dịch sạch và kiểm tra các trường hợp biên.

Ví dụ C / lệnh minh họa

```
int a = 5, b = 2;
printf("%d %.1f\n", a / b, (double)a / b);
```

Chú ý: Một số ví dụ là đoạn minh họa để đặt trong hàm hoặc tệp phù hợp, không phải mọi đoạn đều là chương trình hoàn chỉnh tự chạy.

Thực hành từng bước

1. Đọc ví dụ và dự đoán đầu ra hoặc trạng thái trước khi chạy.
2. Tạo một file bài tập riêng, tự gõ code và biên dịch với -std=c17 -Wall -Wextra -Wpedantic khi thích hợp.
3. Bài tập chính: Tính BMI từ cân nặng/chiều cao; thử 5 / 2 và 5.0 / 2.
4. Thử ít nhất ba tình huống: đầu vào điển hình, ca biên và đầu vào sai nếu chương trình có nhập liệu.
5. Ghi lại warning/lỗi gặp phải, giải thích nguyên nhân và cách sửa bằng lời của mình.

Làm việc với AI đúng cách

- Có thể nhờ AI: yêu cầu giải thích khái niệm, đặt câu hỏi gợi mở, đề xuất test case và đọc warning đã loại bỏ thông tin nhạy cảm.
- Phải tự kiểm chứng: tự chạy code, đối chiếu với tài liệu, đọc từng dòng thay đổi, xác định giới hạn bộ nhớ và đầu vào trước khi dùng.
- Prompt mẫu: "Tôi đang học bài Toán tử, ép kiểu và thứ tự ưu tiên. Hãy đóng vai gia sư C17, đưa ra 3 gợi ý tăng dần cho bài tập: Tính BMI từ cân nặng/chiều cao; thử 5 / 2 và 5.0 / 2. Không đưa lời giải hoàn chỉnh trước khi tôi thử; sau đó hãy review code của tôi về tính đúng, cảnh báo, ca biên và an toàn bộ nhớ."

Lỗi thường gặp

Tránh biểu thức có nhiều phép ++ gây hành vi không xác định; đừng dùng == cho so sánh float tùy tiện.

Điều kiện hoàn thành

Tự giải thích được nội dung bài; hoàn tất bài tập chính; có code minh họa chạy được trong bối cảnh phù hợp; chứng minh đã thử trường hợp biên và sửa các warning quan trọng.

Ví dụ minh họa

Toán tử, ép kiểu và thứ tự ưu tiên

```
int a = 5, b = 2;
printf("%d %.1f\n", a / b, (double)a / b);
```

Số nguyên chia lấy phần nguyên, %, so sánh, && || !, ++ -- và chuyển đổi kiểu.

Checklist

- [ ] Tôi có thể giải thích ý nghĩa của bài bằng ngôn ngữ của mình.
- [ ] Đã thực hành: Tính BMI từ cân nặng/chiều cao; thử 5 / 2 và 5.0 / 2.
- [ ] Đã thử ca thông thường và ca biên; ghi rõ kết quả.
- [ ] Đã đọc warning compiler và xử lý lỗi phù hợp.
- [ ] Đã kiểm tra lỗi: Tránh biểu thức có nhiều phép ++ gây hành vi không xác định; đừng dùng == cho so sánh float tùy tiện.
- [ ] Đã tự xem và hiểu code, không chỉ sao chép lời giải AI.

Tài liệu tham khảo

C language reference — cppreference Tài liệu
https://www.cppreference.com/c/language

Cornell — Introduction to C Programming Khóa học
https://cvw.cac.cornell.edu/Cintro

Nên học trước

Nhập dữ liệu an toàn với fgets và chuyển đổi

GIAI ĐOẠN 3

Rẽ nhánh và vòng lặp

Giai đoạn 3: xây dựng năng lực qua 5 bài về rẽ nhánh và vòng lặp, kết hợp kiến thức, code và kiểm thử.

1. if, else và điều kiện phức hợp

Biết viết điều kiện biên, lồng if vừa đủ, nhóm biểu thức bằng ngoặc.

Mục tiêu học tập

- Giải thích được: Biết viết điều kiện biên, lồng if vừa đủ, nhóm biểu thức bằng ngoặc.
- Thực hiện độc lập: Phân loại điểm học phần theo ngưỡng và kiểm tra các giá trị sát biên.
- Dự đoán, chạy và giải thích kết quả của ví dụ; nhận diện lỗi biên dịch và lỗi logic.
- Phòng tránh lỗi điển hình: Đừng dùng dấu = thay cho == trong điều kiện.

Kiến thức cần học

if · else · comparison · boundary

Giải thích chi tiết

Mục tiêu của bài

Biết viết điều kiện biên, lồng if vừa đủ, nhóm biểu thức bằng ngoặc.

Kiến thức cốt lõi

- Bản chất: Biết viết điều kiện biên, lồng if vừa đủ, nhóm biểu thức bằng ngoặc.
- Cần nắm: if, else, comparison, boundary. Giải thích được từng thuật ngữ bằng ví dụ nhỏ, thay vì chỉ học thuộc cú pháp.
- Cách tư duy: Xác định dữ liệu đầu vào, phép biến đổi, đầu ra và những điều kiện có thể làm chương trình sai.
- Chất lượng: Ưu tiên code rõ ràng, tên dễ hiểu, cảnh báo biên dịch sạch và kiểm tra các trường hợp biên.

Ví dụ C / lệnh minh họa

```
if (score >= 8.5) puts("A");
else if (score >= 7.0) puts("B");
else puts("Can co gang");
```

Chú ý: Một số ví dụ là đoạn minh họa để đặt trong hàm hoặc tệp phù hợp, không phải mọi đoạn đều là chương trình hoàn chỉnh tự chạy.

Thực hành từng bước

1. Đọc ví dụ và dự đoán đầu ra hoặc trạng thái trước khi chạy.
2. Tạo một file bài tập riêng, tự gõ code và biên dịch với -std=c17 -Wall -Wextra -Wpedantic khi thích hợp.
3. Bài tập chính: Phân loại điểm học phần theo ngưỡng và kiểm tra các giá trị sát biên.
4. Thử ít nhất ba tình huống: đầu vào điển hình, ca biên và đầu vào sai nếu chương trình có nhập liệu.
5. Ghi lại warning/lỗi gặp phải, giải thích nguyên nhân và cách sửa bằng lời của mình.

Làm việc với AI đúng cách

- Có thể nhờ AI: yêu cầu giải thích khái niệm, đặt câu hỏi gợi mở, đề xuất test case và đọc warning đã loại bỏ thông tin nhạy cảm.
- Phải tự kiểm chứng: tự chạy code, đối chiếu với tài liệu, đọc từng dòng thay đổi, xác định giới hạn bộ nhớ và đầu vào trước khi dùng.
- Prompt mẫu: "Tôi đang học bài if, else và điều kiện phức hợp. Hãy đóng vai gia sư C17, đưa ra 3 gợi ý tăng dần cho bài tập: Phân loại điểm học phần theo ngưỡng và kiểm tra các giá trị sát biên. Không đưa lời giải hoàn chỉnh trước khi tôi thử; sau đó hãy review code của tôi về tính đúng, cảnh báo, ca biên và an toàn bộ nhớ."

Lỗi thường gặp

Đừng dùng dấu = thay cho == trong điều kiện.

Điều kiện hoàn thành

Tự giải thích được nội dung bài; hoàn tất bài tập chính; có code minh họa chạy được trong bối cảnh phù hợp; chứng minh đã thử trường hợp biên và sửa các warning quan trọng.

Ví dụ minh họa

if, else và điều kiện phức hợp

```
if (score >= 8.5) puts("A");
else if (score >= 7.0) puts("B");
else puts("Can co gang");
```

Biết viết điều kiện biên, lồng if vừa đủ, nhóm biểu thức bằng ngoặc.

Checklist

[ ] Tôi có thể giải thích ý nghĩa của bài bằng ngôn ngữ của mình.
[ ] Đã thực hành: Phân loại điểm học phần theo ngưỡng và kiểm tra các giá trị sát biên.
[ ] Đã thử ca thông thường và ca biên; ghi rõ kết quả.
[ ] Đã đọc warning compiler và xử lý lỗi phù hợp.
[ ] Đã kiểm tra lỗi: Đừng dùng dấu = thay cho == trong điều kiện.
[ ] Đã tự xem và hiểu code, không chỉ sao chép lời giải AI.

Tài liệu tham khảo

C language reference — cppreference Tài liệu
https://www.cppreference.com/c/language

Cornell — Introduction to C Programming Khóa học
https://cvw.cac.cornell.edu/Cintro

Nên học trước

Toán tử, ép kiểu và thứ tự ưu tiên

---

2. switch, case và enum cơ bản

Chọn nhánh khi điều kiện là giá trị nguyên/rời rạc; break, default, fallthrough.

Mục tiêu học tập

- Giải thích được: Chọn nhánh khi điều kiện là giá trị nguyên/rời rạc; break, default, fallthrough.
- Thực hiện độc lập: Tạo menu máy tính bỏ túi và xử lý lựa chọn không hợp lệ.
- Dự đoán, chạy và giải thích kết quả của ví dụ; nhận diện lỗi biên dịch và lỗi logic.
- Phòng tránh lỗi điển hình: Thiếu break dẫn đến chạy xuyên case; switch không thay thế mọi if.

Kiến thức cần học

switch · case · break · default

Giải thích chi tiết

Mục tiêu của bài

Chọn nhánh khi điều kiện là giá trị nguyên/rời rạc; break, default, fallthrough.

Kiến thức cốt lõi

- Bản chất: Chọn nhánh khi điều kiện là giá trị nguyên/rời rạc; break, default, fallthrough.
- Cần nắm: switch, case, break, default. Giải thích được từng thuật ngữ bằng ví dụ nhỏ, thay vì chỉ học thuộc cú pháp.
- Cách tư duy: Xác định dữ liệu đầu vào, phép biến đổi, đầu ra và những điều kiện có thể làm chương trình sai.
- Chất lượng: Ưu tiên code rõ ràng, tên dễ hiểu, cảnh báo biên dịch sạch và kiểm tra các trường hợp biên.

Ví dụ C / lệnh minh họa

```
switch (choice) {
case 1: puts("Them"); break;
case 2: puts("Xoa"); break;
default: puts("Khong hop le");
}
```

Chú ý: Một số ví dụ là đoạn minh họa để đặt trong hàm hoặc tệp phù hợp, không phải mọi đoạn đều là chương trình hoàn chỉnh tự chạy.

Thực hành từng bước

1. Đọc ví dụ và dự đoán đầu ra hoặc trạng thái trước khi chạy.
2. Tạo một file bài tập riêng, tự gõ code và biên dịch với -std=c17 -Wall -Wextra -Wpedantic khi thích hợp.
3. Bài tập chính: Tạo menu máy tính bỏ túi và xử lý lựa chọn không hợp lệ.
4. Thử ít nhất ba tình huống: đầu vào điển hình, ca biên và đầu vào sai nếu chương trình có nhập liệu.
5. Ghi lại warning/lỗi gặp phải, giải thích nguyên nhân và cách sửa bằng lời của mình.

Làm việc với AI đúng cách

- Có thể nhờ AI: yêu cầu giải thích khái niệm, đặt câu hỏi gợi mở, đề xuất test case và đọc warning đã loại bỏ thông tin nhạy cảm.
- Phải tự kiểm chứng: tự chạy code, đối chiếu với tài liệu, đọc từng dòng thay đổi, xác định giới hạn bộ nhớ và đầu vào trước khi dùng.
- Prompt mẫu: "Tôi đang học bài switch, case và enum cơ bản. Hãy đóng vai gia sư C17, đưa ra 3 gợi ý tăng dần cho bài tập: Tạo menu máy tính bỏ túi và xử lý lựa chọn không hợp lệ. Không đưa lời giải hoàn chỉnh trước khi tôi thử; sau đó hãy review code của tôi về tính đúng, cảnh báo, ca biên và an toàn bộ nhớ."

Lỗi thường gặp

Thiếu break dẫn đến chạy xuyên case; switch không thay thế mọi if.

Điều kiện hoàn thành

Tự giải thích được nội dung bài; hoàn tất bài tập chính; có code minh họa chạy được trong bối cảnh phù hợp; chứng minh đã thử trường hợp biên và sửa các warning quan trọng.

Ví dụ minh họa

switch, case và enum cơ bản

```
switch (choice) {
case 1: puts("Them"); break;
case 2: puts("Xoa"); break;
default: puts("Khong hop le");
}
```

Chọn nhánh khi điều kiện là giá trị nguyên/rời rạc; break, default, fallthrough.

Checklist

[ ] Tôi có thể giải thích ý nghĩa của bài bằng ngôn ngữ của mình.

[ ] Đã thực hành: Tạo menu máy tính bỏ túi và xử lý lựa chọn không hợp lệ.

[ ] Đã thử ca thông thường và ca biên; ghi rõ kết quả.

[ ] Đã đọc warning compiler và xử lý lỗi phù hợp.

[ ] Đã kiểm tra lỗi: Thiếu break dẫn đến chạy xuyên case; switch không thay thế mọi if.

[ ] Đã tự xem và hiểu code, không chỉ sao chép lời giải AI.

Tài liệu tham khảo

C language reference — cppreference Tài liệu
https://www.cppreference.com/c/language

Cornell — Introduction to C Programming Khóa học
https://cvw.cac.cornell.edu/Cintro

Nên học trước

if, else và điều kiện phức hợp

---

3. while và do...while

Hiểu điều kiện trước/sau, bất biến vòng lặp, điểm dừng, tránh vòng lặp vô hạn.

Mục tiêu học tập

- Giải thích được: Hiểu điều kiện trước/sau, bất biến vòng lặp, điểm dừng, tránh vòng lặp vô hạn.
- Thực hiện độc lập: Viết chương trình yêu cầu nhập lại đến khi số nằm trong khoảng 1–100.
- Dự đoán, chạy và giải thích kết quả của ví dụ; nhận diện lỗi biên dịch và lỗi logic.
- Phòng tránh lỗi điển hình: Không quên cập nhật biến điều khiển; luôn có đường thoát.

Kiến thức cần học

while · do-while · termination

Giải thích chi tiết

Mục tiêu của bài

Hiểu điều kiện trước/sau, bất biến vòng lặp, điểm dừng, tránh vòng lặp vô hạn.

Kiến thức cốt lõi

- Bản chất: Hiểu điều kiện trước/sau, bất biến vòng lặp, điểm dừng, tránh vòng lặp vô hạn.
- Cần nắm: while, do-while, termination. Giải thích được từng thuật ngữ bằng ví dụ nhỏ, thay vì chỉ học thuộc cú pháp.

- Cách tư duy: Xác định dữ liệu đầu vào, phép biến đổi, đầu ra và những điều kiện có thể làm chương trình sai.
- Chất lượng: Ưu tiên code rõ ràng, tên dễ hiểu, cảnh báo biên dịch sạch và kiểm tra các trường hợp biên.

Ví dụ C / lệnh minh họa

```
int n = 0;
do {
    n++;
} while (n < 5);
```

Chú ý: Một số ví dụ là đoạn minh họa để đặt trong hàm hoặc tệp phù hợp, không phải mọi đoạn đều là chương trình hoàn chỉnh tự chạy.

Thực hành từng bước

1. Đọc ví dụ và dự đoán đầu ra hoặc trạng thái trước khi chạy.
2. Tạo một file bài tập riêng, tự gõ code và biên dịch với -std=c17 -Wall -Wextra -Wpedantic khi thích hợp.
3. Bài tập chính: Viết chương trình yêu cầu nhập lại đến khi số nằm trong khoảng 1–100.
4. Thử ít nhất ba tình huống: đầu vào điển hình, ca biên và đầu vào sai nếu chương trình có nhập liệu.
5. Ghi lại warning/lỗi gặp phải, giải thích nguyên nhân và cách sửa bằng lời của mình.

Làm việc với AI đúng cách

- Có thể nhờ AI: yêu cầu giải thích khái niệm, đặt câu hỏi gợi mở, đề xuất test case và đọc warning đã loại bỏ thông tin nhạy cảm.
- Phải tự kiểm chứng: tự chạy code, đối chiếu với tài liệu, đọc từng dòng thay đổi, xác định giới hạn bộ nhớ và đầu vào trước khi dùng.
- Prompt mẫu: "Tôi đang học bài while và do...while. Hãy đóng vai gia sư C17, đưa ra 3 gợi ý tăng dần cho bài tập: Viết chương trình yêu cầu nhập lại đến khi số nằm trong khoảng 1–100. Không đưa lời giải hoàn chỉnh trước khi tôi thử; sau đó hãy review code của tôi về tính đúng, cảnh báo, ca biên và an toàn bộ nhớ."

Lỗi thường gặp

Không quên cập nhật biến điều khiển; luôn có đường thoát.

Điều kiện hoàn thành

Tự giải thích được nội dung bài; hoàn tất bài tập chính; có code minh họa chạy được trong bối cảnh phù hợp; chứng minh đã thử trường hợp biên và sửa các warning quan trọng.

Ví dụ minh họa

while và do...while

```
int n = 0;
do {
    n++;
} while (n < 5);
```

Hiểu điều kiện trước/sau, bất biến vòng lặp, điểm dừng, tránh vòng lặp vô hạn.

Checklist

[ ] Tôi có thể giải thích ý nghĩa của bài bằng ngôn ngữ của mình.

[ ] Đã thực hành: Viết chương trình yêu cầu nhập lại đến khi số nằm trong khoảng 1–100.

[ ] Đã thử ca thông thường và ca biên; ghi rõ kết quả.

[ ] Đã đọc warning compiler và xử lý lỗi phù hợp.

[ ] Đã kiểm tra lỗi: Không quên cập nhật biến điều khiển; luôn có đường thoát.

[ ] Đã tự xem và hiểu code, không chỉ sao chép lời giải AI.

Tài liệu tham khảo

C language reference — cppreference Tài liệu
https://www.cppreference.com/c/language

Cornell — Introduction to C Programming Khóa học
https://cvw.cac.cornell.edu/Cintro

Nên học trước

switch, case và enum cơ bản

---

4. Vòng lặp for và vòng lặp lồng nhau

Mô tả khởi tạo, điều kiện, cập nhật; xử lý dãy số và bảng hai chiều.

Mục tiêu học tập

- Giải thích được: Mô tả khởi tạo, điều kiện, cập nhật; xử lý dãy số và bảng hai chiều.
- Thực hiện độc lập: In bảng cửu chương, tổng n số đầu tiên, tam giác ký tự.
- Dự đoán, chạy và giải thích kết quả của ví dụ; nhận diện lỗi biên dịch và lỗi logic.
- Phòng tránh lỗi điển hình: Phân biệt < n với <= n; kiểm tra tràn số khi cộng dồn.

Kiến thức cần học

for · nested-loops · off-by-one

Giải thích chi tiết

Mục tiêu của bài

Mô tả khởi tạo, điều kiện, cập nhật; xử lý dãy số và bảng hai chiều.

Kiến thức cốt lõi

- Bản chất: Mô tả khởi tạo, điều kiện, cập nhật; xử lý dãy số và bảng hai chiều.
- Cần nắm: for, nested-loops, off-by-one. Giải thích được từng thuật ngữ bằng ví dụ nhỏ, thay vì chỉ học thuộc cú pháp.
- Cách tư duy: Xác định dữ liệu đầu vào, phép biến đổi, đầu ra và những điều kiện có thể làm chương trình sai.
- Chất lượng: Ưu tiên code rõ ràng, tên dễ hiểu, cảnh báo biên dịch sạch và kiểm tra các trường hợp biên.

Ví dụ C / lệnh minh họa

```
for (int i = 1; i <= 10; i++) {
    printf("%d ", i * 2);
}
```

Chú ý: Một số ví dụ là đoạn minh họa để đặt trong hàm hoặc tệp phù hợp, không phải mọi đoạn đều là chương trình hoàn chỉnh tự chạy.

Thực hành từng bước

1. Đọc ví dụ và dự đoán đầu ra hoặc trạng thái trước khi chạy.
2. Tạo một file bài tập riêng, tự gõ code và biên dịch với -std=c17 -Wall -Wextra -Wpedantic khi thích hợp.
3. Bài tập chính: In bảng cửu chương, tổng n số đầu tiên, tam giác ký tự.
4. Thử ít nhất ba tình huống: đầu vào điển hình, ca biên và đầu vào sai nếu chương trình có nhập liệu.
5. Ghi lại warning/lỗi gặp phải, giải thích nguyên nhân và cách sửa bằng lời của mình.

Làm việc với AI đúng cách

- Có thể nhờ AI: yêu cầu giải thích khái niệm, đặt câu hỏi gợi mở, đề xuất test case và đọc warning đã loại bỏ thông tin nhạy cảm.
- Phải tự kiểm chứng: tự chạy code, đối chiếu với tài liệu, đọc từng dòng thay đổi, xác định giới hạn bộ nhớ và đầu vào trước khi dùng.
- Prompt mẫu: "Tôi đang học bài Vòng lặp for và vòng lặp lồng nhau. Hãy đóng vai gia sư C17, đưa ra 3 gợi ý tăng dần cho bài tập: In bảng cửu chương, tổng n số đầu tiên, tam giác ký tự. Không đưa lời giải hoàn chỉnh trước khi tôi thử; sau đó hãy review code của tôi về tính đúng, cảnh báo, ca biên và an toàn bộ nhớ."

Lỗi thường gặp

Phân biệt < n với <= n; kiểm tra tràn số khi cộng dồn.

Điều kiện hoàn thành

Tự giải thích được nội dung bài; hoàn tất bài tập chính; có code minh họa chạy được trong bối cảnh phù hợp; chứng minh đã thử trường hợp biên và sửa các warning quan trọng.

Ví dụ minh họa

Vòng lặp for và vòng lặp lồng nhau

```
for (int i = 1; i <= 10; i++) {
    printf("%d ", i * 2);
}
```

Mô tả khởi tạo, điều kiện, cập nhật; xử lý dãy số và bảng hai chiều.

Checklist

[ ] Tôi có thể giải thích ý nghĩa của bài bằng ngôn ngữ của mình.
[ ] Đã thực hành: In bảng cửu chương, tổng n số đầu tiên, tam giác ký tự.
[ ] Đã thử ca thông thường và ca biên; ghi rõ kết quả.
[ ] Đã đọc warning compiler và xử lý lỗi phù hợp.
[ ] Đã kiểm tra lỗi: Phân biệt < n với <= n; kiểm tra tràn số khi cộng dồn.
[ ] Đã tự xem và hiểu code, không chỉ sao chép lời giải AI.

Tài liệu tham khảo

C language reference — cppreference Tài liệu
https://www.cppreference.com/c/language

Cornell — Introduction to C Programming Khóa học
https://cvw.cac.cornell.edu/Cintro

Nên học trước

while và do...while

5. Bài luyện tổng hợp điều kiện và vòng lặp

Kết hợp if, switch, for, while để giải bài có nhiều điều kiện và ca biên.

Mục tiêu học tập

- Giải thích được: Kết hợp if, switch, for, while để giải bài có nhiều điều kiện và ca biên.
- Thực hiện độc lập: Tạo máy tính console có menu lặp, cộng/trừ/nhân/chia, kiểm tra chia 0.
- Dự đoán, chạy và giải thích kết quả của ví dụ; nhận diện lỗi biên dịch và lỗi logic.
- Phòng tránh lỗi điển hình: Tránh code trùng lặp và chỉ thử một đầu vào đẹp.

Kiến thức cần học

flowchart · test-cases · input-validation

Giải thích chi tiết

Mục tiêu của bài

Kết hợp if, switch, for, while để giải bài có nhiều điều kiện và ca biên.

Kiến thức cốt lõi

- Bản chất: Kết hợp if, switch, for, while để giải bài có nhiều điều kiện và ca biên.
- Cần nắm: flowchart, test-cases, input-validation. Giải thích được từng thuật ngữ bằng ví dụ nhỏ, thay vì chỉ học thuộc cú pháp.
- Cách tư duy: Xác định dữ liệu đầu vào, phép biến đổi, đầu ra và những điều kiện có thể làm chương trình sai.
- Chất lượng: Ưu tiên code rõ ràng, tên dễ hiểu, cảnh báo biên dịch sạch và kiểm tra các trường hợp biên.

Ví dụ C / lệnh minh họa

```
while (running) {
    /* hien menu, doc lenh, xu ly, thoat */
}
```

Chú ý: Một số ví dụ là đoạn minh họa để đặt trong hàm hoặc tệp phù hợp, không phải mọi đoạn đều là chương trình hoàn chỉnh tự chạy.

Thực hành từng bước

1. Đọc ví dụ và dự đoán đầu ra hoặc trạng thái trước khi chạy.
2. Tạo một file bài tập riêng, tự gõ code và biên dịch với -std=c17 -Wall -Wextra -Wpedantic khi thích hợp.
3. Bài tập chính: Tạo máy tính console có menu lặp, cộng/trừ/nhân/chia, kiểm tra chia 0.
4. Thử ít nhất ba tình huống: đầu vào điển hình, ca biên và đầu vào sai nếu chương trình có nhập liệu.
5. Ghi lại warning/lỗi gặp phải, giải thích nguyên nhân và cách sửa bằng lời của mình.

Làm việc với AI đúng cách

- Có thể nhờ AI: yêu cầu giải thích khái niệm, đặt câu hỏi gợi mở, đề xuất test case và đọc warning đã loại bỏ thông tin nhạy cảm.
- Phải tự kiểm chứng: tự chạy code, đối chiếu với tài liệu, đọc từng dòng thay đổi, xác định giới hạn bộ nhớ và đầu vào trước khi dùng.

- Prompt mẫu: "Tôi đang học bài Bài luyện tổng hợp điều kiện và vòng lặp. Hãy đóng vai gia sư C17, đưa ra 3 gợi ý tăng dần cho bài tập: Tạo máy tính console có menu lặp, cộng/trừ/nhân/chia, kiểm tra chia 0. Không đưa lời giải hoàn chỉnh trước khi tôi thử; sau đó hãy review code của tôi về tính đúng, cảnh báo, ca biên và an toàn bộ nhớ."

Lỗi thường gặp

Tránh code trùng lặp và chỉ thử một đầu vào đẹp.

Điều kiện hoàn thành

Tự giải thích được nội dung bài; hoàn tất bài tập chính; có code minh họa chạy được trong bối cảnh phù hợp; chứng minh đã thử trường hợp biên và sửa các warning quan trọng.

Ví dụ minh họa

Bài luyện tổng hợp điều kiện và vòng lặp

```
while (running) {
    /* hien menu, doc lenh, xu ly, thoat */
}
```

Kết hợp if, switch, for, while để giải bài có nhiều điều kiện và ca biên.

Checklist

[ ] Tôi có thể giải thích ý nghĩa của bài bằng ngôn ngữ của mình.

[ ] Đã thực hành: Tạo máy tính console có menu lặp, cộng/trừ/nhân/chia, kiểm tra chia 0.

[ ] Đã thử ca thông thường và ca biên; ghi rõ kết quả.

[ ] Đã đọc warning compiler và xử lý lỗi phù hợp.

[ ] Đã kiểm tra lỗi: Tránh code trùng lặp và chỉ thử một đầu vào đẹp.

[ ] Đã tự xem và hiểu code, không chỉ sao chép lời giải AI.

Tài liệu tham khảo

C language reference — cppreference Tài liệu
https://www.cppreference.com/c/language

Cornell — Introduction to C Programming Khóa học
https://cvw.cac.cornell.edu/Cintro

Nên học trước

Vòng lặp for và vòng lặp lồng nhau

GIAI ĐOẠN 4

Hàm và tổ chức mã nguồn

Giai đoạn 4: xây dựng năng lực qua 5 bài về hàm và tổ chức mã nguồn, kết hợp kiến thức, code và kiểm thử.

---

1. Khai báo, định nghĩa và gọi hàm

Đọc prototype, tham số, đối số, kiểu trả về; tách hàm một trách nhiệm.

Mục tiêu học tập

- Giải thích được: Đọc prototype, tham số, đối số, kiểu trả về; tách hàm một trách nhiệm.
- Thực hiện độc lập: Tách máy tính console thành add, subtract, multiply, divide_checked.
- Dự đoán, chạy và giải thích kết quả của ví dụ; nhận diện lỗi biên dịch và lỗi logic.
- Phòng tránh lỗi điển hình: Không khai báo thiếu kiểu; không quên trả về đúng kiểu.

Kiến thức cần học

function · prototype · parameter · return

Giải thích chi tiết

Mục tiêu của bài

Đọc prototype, tham số, đối số, kiểu trả về; tách hàm một trách nhiệm.

Kiến thức cốt lõi

- Bản chất: Đọc prototype, tham số, đối số, kiểu trả về; tách hàm một trách nhiệm.
- Cần nắm: function, prototype, parameter, return. Giải thích được từng thuật ngữ bằng ví dụ nhỏ, thay vì chỉ học thuộc cú pháp.
- Cách tư duy: Xác định dữ liệu đầu vào, phép biến đổi, đầu ra và những điều kiện có thể làm chương trình sai.
- Chất lượng: Ưu tiên code rõ ràng, tên dễ hiểu, cảnh báo biên dịch sạch và kiểm tra các trường hợp biên.

Ví dụ C / lệnh minh họa

```
int square(int x);
int square(int x) { return x * x; }
```

Chú ý: Một số ví dụ là đoạn minh họa để đặt trong hàm hoặc tệp phù hợp, không phải mọi đoạn đều là chương trình hoàn chỉnh tự chạy.

Thực hành từng bước

1. Đọc ví dụ và dự đoán đầu ra hoặc trạng thái trước khi chạy.
2. Tạo một file bài tập riêng, tự gõ code và biên dịch với -std=c17 -Wall -Wextra -Wpedantic khi thích hợp.
3. Bài tập chính: Tách máy tính console thành add, subtract, multiply, divide_checked.
4. Thử ít nhất ba tình huống: đầu vào điển hình, ca biên và đầu vào sai nếu chương trình có nhập liệu.
5. Ghi lại warning/lỗi gặp phải, giải thích nguyên nhân và cách sửa bằng lời của mình.

Làm việc với AI đúng cách

- Có thể nhờ AI: yêu cầu giải thích khái niệm, đặt câu hỏi gợi mở, đề xuất test case và đọc warning đã loại bỏ thông tin nhạy cảm.
- Phải tự kiểm chứng: tự chạy code, đối chiếu với tài liệu, đọc từng dòng thay đổi, xác định giới hạn bộ nhớ và đầu vào trước khi dùng.
- Prompt mẫu: "Tôi đang học bài Khai báo, định nghĩa và gọi hàm. Hãy đóng vai gia sư C17, đưa ra 3 gợi ý tăng dần cho bài tập: Tách máy tính console thành add, subtract, multiply, divide_checked. Không đưa lời giải hoàn chỉnh trước khi tôi thử; sau đó hãy review code của tôi về tính đúng, cảnh báo, ca biên và an toàn bộ nhớ."

Lỗi thường gặp

Không khai báo thiếu kiểu; không quên trả về đúng kiểu.

Điều kiện hoàn thành

Tự giải thích được nội dung bài; hoàn tất bài tập chính; có code minh họa chạy được trong bối cảnh phù hợp; chứng minh đã thử trường hợp biên và sửa các warning quan trọng.

Ví dụ minh họa

Khai báo, định nghĩa và gọi hàm

```
int square(int x);
int square(int x) { return x * x; }
```

Đọc prototype, tham số, đối số, kiểu trả về; tách hàm một trách nhiệm.

Checklist

[ ] Tôi có thể giải thích ý nghĩa của bài bằng ngôn ngữ của mình.
[ ] Đã thực hành: Tách máy tính console thành add, subtract, multiply, divide_checked.
[ ] Đã thử ca thông thường và ca biên; ghi rõ kết quả.
[ ] Đã đọc warning compiler và xử lý lỗi phù hợp.
[ ] Đã kiểm tra lỗi: Không khai báo thiếu kiểu; không quên trả về đúng kiểu.
[ ] Đã tự xem và hiểu code, không chỉ sao chép lời giải AI.

Tài liệu tham khảo

C language reference — cppreference Tài liệu
https://www.cppreference.com/c/language

Nên học trước

Bài luyện tổng hợp điều kiện và vòng lặp

---

2. Scope, lifetime và static

Hiểu biến cục bộ/toàn cục, thời gian sống, static local và tác dụng phụ.

Mục tiêu học tập

- Giải thích được: Hiểu biến cục bộ/toàn cục, thời gian sống, static local và tác dụng phụ.
- Thực hiện độc lập: Viết hàm đếm số lần được gọi và so sánh local thông thường với static local.
- Dự đoán, chạy và giải thích kết quả của ví dụ; nhận diện lỗi biên dịch và lỗi logic.
- Phòng tránh lỗi điển hình: Hạn chế global state; không trả con trỏ trỏ tới biến cục bộ.

Kiến thức cần học

scope · lifetime · static · global

Giải thích chi tiết

Mục tiêu của bài

Hiểu biến cục bộ/toàn cục, thời gian sống, static local và tác dụng phụ.

Kiến thức cốt lõi

- Bản chất: Hiểu biến cục bộ/toàn cục, thời gian sống, static local và tác dụng phụ.
- Cần nắm: scope, lifetime, static, global. Giải thích được từng thuật ngữ bằng ví dụ nhỏ, thay vì chỉ học thuộc cú pháp.
- Cách tư duy: Xác định dữ liệu đầu vào, phép biến đổi, đầu ra và những điều kiện có thể làm chương trình sai.
- Chất lượng: Ưu tiên code rõ ràng, tên dễ hiểu, cảnh báo biên dịch sạch và kiểm tra các trường hợp biên.

Ví dụ C / lệnh minh họa

```
int next_id(void) {
    static int id = 0;
    return ++id;
}
```

Chú ý: Một số ví dụ là đoạn minh họa để đặt trong hàm hoặc tệp phù hợp, không phải mọi đoạn đều là chương trình hoàn chỉnh tự chạy.

Thực hành từng bước

1. Đọc ví dụ và dự đoán đầu ra hoặc trạng thái trước khi chạy.
2. Tạo một file bài tập riêng, tự gõ code và biên dịch với -std=c17 -Wall -Wextra -Wpedantic khi thích hợp.
3. Bài tập chính: Viết hàm đếm số lần được gọi và so sánh local thông thường với static local.
4. Thử ít nhất ba tình huống: đầu vào điển hình, ca biên và đầu vào sai nếu chương trình có nhập liệu.
5. Ghi lại warning/lỗi gặp phải, giải thích nguyên nhân và cách sửa bằng lời của mình.

Làm việc với AI đúng cách

- Có thể nhờ AI: yêu cầu giải thích khái niệm, đặt câu hỏi gợi mở, đề xuất test case và đọc warning đã loại bỏ thông tin nhạy cảm.
- Phải tự kiểm chứng: tự chạy code, đối chiếu với tài liệu, đọc từng dòng thay đổi, xác định giới hạn bộ nhớ và đầu vào trước khi dùng.
- Prompt mẫu: "Tôi đang học bài Scope, lifetime và static. Hãy đóng vai gia sư C17, đưa ra 3 gợi ý tăng dần cho bài tập: Viết hàm đếm số lần được gọi và so sánh local thông thường với static local. Không đưa lời giải hoàn chỉnh trước khi tôi thử; sau đó hãy review code của tôi về tính đúng, cảnh báo, ca biên và an toàn bộ nhớ."

Lỗi thường gặp

Hạn chế global state; không trả con trỏ trỏ tới biến cục bộ.

Điều kiện hoàn thành

Tự giải thích được nội dung bài; hoàn tất bài tập chính; có code minh họa chạy được trong bối cảnh phù hợp; chứng minh đã thử trường hợp biên và sửa các warning quan trọng.

Ví dụ minh họa

Scope, lifetime và static

```c
int next_id(void) {
    static int id = 0;
    return ++id;
}
```

Hiểu biến cục bộ/toàn cục, thời gian sống, static local và tác dụng phụ.

Checklist

[ ] Tôi có thể giải thích ý nghĩa của bài bằng ngôn ngữ của mình.
[ ] Đã thực hành: Viết hàm đếm số lần được gọi và so sánh local thông thường với static local.
[ ] Đã thử ca thông thường và ca biên; ghi rõ kết quả.
[ ] Đã đọc warning compiler và xử lý lỗi phù hợp.
[ ] Đã kiểm tra lỗi: Hạn chế global state; không trả con trỏ trỏ tới biến cục bộ.
[ ] Đã tự xem và hiểu code, không chỉ sao chép lời giải AI.

Tài liệu tham khảo

C language reference — cppreference Tài liệu
https://www.cppreference.com/c/language

Nên học trước

Khai báo, định nghĩa và gọi hàm

---

3. Truyền tham trị và địa chỉ

C truyền tham trị; muốn đổi biến bên ngoài phải truyền địa chỉ bằng con trỏ.

Mục tiêu học tập

- Giải thích được: C truyền tham trị; muốn đổi biến bên ngoài phải truyền địa chỉ bằng con trỏ.
- Thực hiện độc lập: Cài đặt swap(int a, int b) và giải thích vì sao swap(int a, int b) không đổi được hai biến gốc.
- Dự đoán, chạy và giải thích kết quả của ví dụ; nhận diện lỗi biên dịch và lỗi logic.
- Phòng tránh lỗi điển hình: Không truyền con trỏ NULL rồi dereference; phân biệt giá trị với địa chỉ.

Kiến thức cần học

pass-by-value · address-of · pointer

Giải thích chi tiết

Mục tiêu của bài

C truyền tham trị; muốn đổi biến bên ngoài phải truyền địa chỉ bằng con trỏ.

Kiến thức cốt lõi

- Bản chất: C truyền tham trị; muốn đổi biến bên ngoài phải truyền địa chỉ bằng con trỏ.
- Cần nắm: pass-by-value, address-of, pointer. Giải thích được từng thuật ngữ bằng ví dụ nhỏ, thay vì chỉ học thuộc cú pháp.
- Cách tư duy: Xác định dữ liệu đầu vào, phép biến đổi, đầu ra và những điều kiện có thể làm chương trình sai.
- Chất lượng: Ưu tiên code rõ ràng, tên dễ hiểu, cảnh báo biên dịch sạch và kiểm tra các trường hợp biên.

Ví dụ C / lệnh minh họa

```
void swap(int *a, int *b) {
    int t = *a; *a = *b; *b = t;
}
```

Chú ý: Một số ví dụ là đoạn minh họa để đặt trong hàm hoặc tệp phù hợp, không phải mọi đoạn đều là chương trình hoàn chỉnh tự chạy.

Thực hành từng bước

1. Đọc ví dụ và dự đoán đầu ra hoặc trạng thái trước khi chạy.
2. Tạo một file bài tập riêng, tự gõ code và biên dịch với -std=c17 -Wall -Wextra -Wpedantic khi thích hợp.
3. Bài tập chính: Cài đặt swap(int a, int b) và giải thích vì sao swap(int a, int b) không đổi được hai biến gốc.
4. Thử ít nhất ba tình huống: đầu vào điển hình, ca biên và đầu vào sai nếu chương trình có nhập liệu.
5. Ghi lại warning/lỗi gặp phải, giải thích nguyên nhân và cách sửa bằng lời của mình.

Làm việc với AI đúng cách

- Có thể nhờ AI: yêu cầu giải thích khái niệm, đặt câu hỏi gợi mở, đề xuất test case và đọc warning đã loại bỏ thông tin nhạy cảm.
- Phải tự kiểm chứng: tự chạy code, đối chiếu với tài liệu, đọc từng dòng thay đổi, xác định giới hạn bộ nhớ và đầu vào trước khi dùng.
- Prompt mẫu: "Tôi đang học bài Truyền tham trị và địa chỉ. Hãy đóng vai gia sư C17, đưa ra 3 gợi ý tăng dần cho bài tập: Cài đặt swap(int a, int b) và giải thích vì sao swap(int a, int b) không đổi được hai biến gốc. Không đưa lời giải hoàn chỉnh trước khi tôi thử; sau đó hãy review code của tôi về tính đúng, cảnh báo, ca biên và an toàn bộ nhớ."

Lỗi thường gặp

Không truyền con trỏ NULL rồi dereference; phân biệt giá trị với địa chỉ.

Điều kiện hoàn thành

Tự giải thích được nội dung bài; hoàn tất bài tập chính; có code minh họa chạy được trong bối cảnh phù hợp; chứng minh đã thử trường hợp biên và sửa các warning quan trọng.

Ví dụ minh họa

Truyền tham trị và địa chỉ

```
void swap(int *a, int *b) {
    int t = *a; *a = *b; *b = t;
}
```

C truyền tham trị; muốn đổi biến bên ngoài phải truyền địa chỉ bằng con trỏ.

Checklist

- [ ] Tôi có thể giải thích ý nghĩa của bài bằng ngôn ngữ của mình.
- [ ] Đã thực hành: Cài đặt swap(int a, int b) và giải thích vì sao swap(int a, int b) không đổi được hai biến gốc.
- [ ] Đã thử ca thông thường và ca biên; ghi rõ kết quả.
- [ ] Đã đọc warning compiler và xử lý lỗi phù hợp.
- [ ] Đã kiểm tra lỗi: Không truyền con trỏ NULL rồi dereference; phân biệt giá trị với địa chỉ.
- [ ] Đã tự xem và hiểu code, không chỉ sao chép lời giải AI.

Tài liệu tham khảo

C language reference — cppreference Tài liệu
https://www.cppreference.com/c/language

Nên học trước

Scope, lifetime và static

---

4. Đệ quy và trường hợp dừng

Học base case, recursive case, call stack; biết khi nào nên đổi sang vòng lặp.

Mục tiêu học tập

- Giải thích được: Học base case, recursive case, call stack; biết khi nào nên đổi sang vòng lặp.
- Thực hiện độc lập: Viết factorial đệ quy và lặp; đo/so sánh trên n nhỏ, kiểm tra tràn.
- Dự đoán, chạy và giải thích kết quả của ví dụ; nhận diện lỗi biên dịch và lỗi logic.
- Phòng tránh lỗi điển hình: Thiếu base case gây tràn stack; đệ quy không phải lúc nào cũng tốt hơn.

Kiến thức cần học

recursion · base-case · stack

Giải thích chi tiết

Mục tiêu của bài

Học base case, recursive case, call stack; biết khi nào nên đổi sang vòng lặp.

Kiến thức cốt lõi

- Bản chất: Học base case, recursive case, call stack; biết khi nào nên đổi sang vòng lặp.
- Cần nắm: recursion, base-case, stack. Giải thích được từng thuật ngữ bằng ví dụ nhỏ, thay vì chỉ học thuộc cú pháp.
- Cách tư duy: Xác định dữ liệu đầu vào, phép biến đổi, đầu ra và những điều kiện có thể làm chương trình sai.
- Chất lượng: Ưu tiên code rõ ràng, tên dễ hiểu, cảnh báo biên dịch sạch và kiểm tra các trường hợp biên.

Ví dụ C / lệnh minh họa

```
unsigned long long fact(unsigned n) {
    return n <= 1 ? 1ULL : n * fact(n - 1);
}
```

Chú ý: Một số ví dụ là đoạn minh họa để đặt trong hàm hoặc tệp phù hợp, không phải mọi đoạn đều là chương trình hoàn chỉnh tự chạy.

Thực hành từng bước

1. Đọc ví dụ và dự đoán đầu ra hoặc trạng thái trước khi chạy.
2. Tạo một file bài tập riêng, tự gõ code và biên dịch với -std=c17 -Wall -Wextra -Wpedantic khi thích hợp.
3. Bài tập chính: Viết factorial đệ quy và lặp; đo/so sánh trên n nhỏ, kiểm tra tràn.
4. Thử ít nhất ba tình huống: đầu vào điển hình, ca biên và đầu vào sai nếu chương trình có nhập liệu.
5. Ghi lại warning/lỗi gặp phải, giải thích nguyên nhân và cách sửa bằng lời của mình.

Làm việc với AI đúng cách

- Có thể nhờ AI: yêu cầu giải thích khái niệm, đặt câu hỏi gợi mở, đề xuất test case và đọc warning đã loại bỏ thông tin nhạy cảm.
- Phải tự kiểm chứng: tự chạy code, đối chiếu với tài liệu, đọc từng dòng thay đổi, xác định giới hạn bộ nhớ và đầu vào trước khi dùng.
- Prompt mẫu: "Tôi đang học bài Đệ quy và trường hợp dừng. Hãy đóng vai gia sư C17, đưa ra 3 gợi ý tăng dần cho bài tập: Viết factorial đệ quy và lặp; đo/so sánh trên n nhỏ, kiểm tra tràn. Không đưa lời giải hoàn chỉnh trước khi tôi thử; sau đó hãy review code của tôi về tính đúng, cảnh báo, ca biên và an toàn bộ nhớ."

Lỗi thường gặp

Thiếu base case gây tràn stack; đệ quy không phải lúc nào cũng tốt hơn.

Điều kiện hoàn thành

Tự giải thích được nội dung bài; hoàn tất bài tập chính; có code minh họa chạy được trong bối cảnh phù hợp; chứng minh đã thử trường hợp biên và sửa các warning quan trọng.

Ví dụ minh họa

Đệ quy và trường hợp dừng

```
unsigned long long fact(unsigned n) {
    return n <= 1 ? 1ULL : n * fact(n - 1);
}
```

Học base case, recursive case, call stack; biết khi nào nên đổi sang vòng lặp.

Checklist

[ ] Tôi có thể giải thích ý nghĩa của bài bằng ngôn ngữ của mình.
[ ] Đã thực hành: Viết factorial đệ quy và lặp; đo/so sánh trên n nhỏ, kiểm tra tràn.
[ ] Đã thử ca thông thường và ca biên; ghi rõ kết quả.
[ ] Đã đọc warning compiler và xử lý lỗi phù hợp.
[ ] Đã kiểm tra lỗi: Thiếu base case gây tràn stack; đệ quy không phải lúc nào cũng tốt hơn.
[ ] Đã tự xem và hiểu code, không chỉ sao chép lời giải AI.

Tài liệu tham khảo

C language reference — cppreference Tài liệu
https://www.cppreference.com/c/language

Nên học trước

Truyền tham trị và địa chỉ

---

5. Chia tệp .c/.h và liên kết

Khai báo API trong header, định nghĩa trong .c, include guards, compile nhiều tệp.

Mục tiêu học tập

- Giải thích được: Khai báo API trong header, định nghĩa trong .c, include guards, compile nhiều tệp.
- Thực hiện độc lập: Tách mathutils.h, mathutils.c, main.c; build bằng 1 lệnh GCC.
- Dự đoán, chạy và giải thích kết quả của ví dụ; nhận diện lỗi biên dịch và lỗi logic.
- Phòng tránh lỗi điển hình: Không #include file .c; tránh định nghĩa biến/hàm thường trong header.

Kiến thức cần học

header · include-guard · translation-unit · linker

Giải thích chi tiết

Mục tiêu của bài

Khai báo API trong header, định nghĩa trong .c, include guards, compile nhiều tệp.

Kiến thức cốt lõi

- Bản chất: Khai báo API trong header, định nghĩa trong .c, include guards, compile nhiều tệp.
- Cần nắm: header, include-guard, translation-unit, linker. Giải thích được từng thuật ngữ bằng ví dụ nhỏ, thay vì chỉ học thuộc cú pháp.
- Cách tư duy: Xác định dữ liệu đầu vào, phép biến đổi, đầu ra và những điều kiện có thể làm chương trình sai.
- Chất lượng: Ưu tiên code rõ ràng, tên dễ hiểu, cảnh báo biên dịch sạch và kiểm tra các trường hợp biên.

Ví dụ C / lệnh minh họa

```
/* math_utils.h */
#ifndef MATH_UTILS_H
#define MATH_UTILS_H
int square(int x);
#endif
```

Chú ý: Một số ví dụ là đoạn minh họa để đặt trong hàm hoặc tệp phù hợp, không phải mọi đoạn đều là chương trình hoàn chỉnh tự chạy.

Thực hành từng bước

1. Đọc ví dụ và dự đoán đầu ra hoặc trạng thái trước khi chạy.
2. Tạo một file bài tập riêng, tự gõ code và biên dịch với -std=c17 -Wall -Wextra -Wpedantic khi thích hợp.
3. Bài tập chính: Tách mathutils.h, mathutils.c, main.c; build bằng 1 lệnh GCC.
4. Thử ít nhất ba tình huống: đầu vào điển hình, ca biên và đầu vào sai nếu chương trình có nhập liệu.
5. Ghi lại warning/lỗi gặp phải, giải thích nguyên nhân và cách sửa bằng lời của mình.

Làm việc với AI đúng cách

- Có thể nhờ AI: yêu cầu giải thích khái niệm, đặt câu hỏi gợi mở, đề xuất test case và đọc warning đã loại bỏ thông tin nhạy cảm.
- Phải tự kiểm chứng: tự chạy code, đối chiếu với tài liệu, đọc từng dòng thay đổi, xác định giới hạn bộ nhớ và đầu vào trước khi dùng.
- Prompt mẫu: "Tôi đang học bài Chia tệp .c/.h và liên kết. Hãy đóng vai gia sư C17, đưa ra 3 gợi ý tăng dần cho bài tập: Tách mathutils.h, mathutils.c, main.c; build bằng 1 lệnh GCC. Không đưa lời giải hoàn chỉnh trước khi tôi thử; sau đó hãy review code của tôi về tính đúng, cảnh báo, ca biên và an toàn bộ nhớ."

Lỗi thường gặp

Không #include file .c; tránh định nghĩa biến/hàm thường trong header.

Điều kiện hoàn thành

Tự giải thích được nội dung bài; hoàn tất bài tập chính; có code minh họa chạy được trong bối cảnh phù hợp; chứng minh đã thử trường hợp biên và sửa các warning quan trọng.

Ví dụ minh họa

Chia tệp .c/.h và liên kết

```
/* math_utils.h */
#ifndef MATH_UTILS_H
#define MATH_UTILS_H
int square(int x);
#endif
```

Khai báo API trong header, định nghĩa trong .c, include guards, compile nhiều tệp.

Checklist

- [ ] Tôi có thể giải thích ý nghĩa của bài bằng ngôn ngữ của mình.
- [ ] Đã thực hành: Tách mathutils.h, mathutils.c, main.c; build bằng 1 lệnh GCC.
- [ ] Đã thử ca thông thường và ca biên; ghi rõ kết quả.
- [ ] Đã đọc warning compiler và xử lý lỗi phù hợp.
- [ ] Đã kiểm tra lỗi: Không #include file .c; tránh định nghĩa biến/hàm thường trong header.
- [ ] Đã tự xem và hiểu code, không chỉ sao chép lời giải AI.

Tài liệu tham khảo

C language reference — cppreference Tài liệu
https://www.cppreference.com/c/language

Nên học trước

Đệ quy và trường hợp dừng

GIAI ĐOẠN 5

Mảng và chuỗi trong C

Giai đoạn 5: xây dựng năng lực qua 5 bài về mảng và chuỗi trong c, kết hợp kiến thức, code và kiểm thử.

1. Mảng một chiều và chỉ số

Mảng chứa các phần tử cùng kiểu, index 0..n-1, sizeof trong phạm vi phù hợp.

Mục tiêu học tập

- Giải thích được: Mảng chứa các phần tử cùng kiểu, index 0..n-1, sizeof trong phạm vi phù hợp.
- Thực hiện độc lập: Nhập 10 điểm, tính trung bình, min/max, đếm điểm đạt.
- Dự đoán, chạy và giải thích kết quả của ví dụ; nhận diện lỗi biên dịch và lỗi logic.
- Phòng tránh lỗi điển hình: Truy cập ngoài mảng là undefined behavior; kiểm tra chỉ số.

Kiến thức cần học

array · index · bounds · iteration

Giải thích chi tiết

Mục tiêu của bài

Mảng chứa các phần tử cùng kiểu, index 0..n-1, sizeof trong phạm vi phù hợp.

Kiến thức cốt lõi

- Bản chất: Mảng chứa các phần tử cùng kiểu, index 0..n-1, sizeof trong phạm vi phù hợp.
- Cần nắm: array, index, bounds, iteration. Giải thích được từng thuật ngữ bằng ví dụ nhỏ, thay vì chỉ học thuộc cú pháp.
- Cách tư duy: Xác định dữ liệu đầu vào, phép biến đổi, đầu ra và những điều kiện có thể làm chương trình sai.
- Chất lượng: Ưu tiên code rõ ràng, tên dễ hiểu, cảnh báo biên dịch sạch và kiểm tra các trường hợp biên.

Ví dụ C / lệnh minh họa

```
int scores[3] = {8, 9, 7};
for (size_t i = 0; i < 3; ++i) printf("%d ", scores[i]);
```

Chú ý: Một số ví dụ là đoạn minh họa để đặt trong hàm hoặc tệp phù hợp, không phải mọi đoạn đều là chương trình hoàn chỉnh tự chạy.

Thực hành từng bước

1. Đọc ví dụ và dự đoán đầu ra hoặc trạng thái trước khi chạy.
2. Tạo một file bài tập riêng, tự gõ code và biên dịch với -std=c17 -Wall -Wextra -Wpedantic khi thích hợp.
3. Bài tập chính: Nhập 10 điểm, tính trung bình, min/max, đếm điểm đạt.
4. Thử ít nhất ba tình huống: đầu vào điển hình, ca biên và đầu vào sai nếu chương trình có nhập liệu.
5. Ghi lại warning/lỗi gặp phải, giải thích nguyên nhân và cách sửa bằng lời của mình.

Làm việc với AI đúng cách

- Có thể nhờ AI: yêu cầu giải thích khái niệm, đặt câu hỏi gợi mở, đề xuất test case và đọc warning đã loại bỏ thông tin nhạy cảm.
- Phải tự kiểm chứng: tự chạy code, đối chiếu với tài liệu, đọc từng dòng thay đổi, xác định giới hạn bộ nhớ và đầu vào trước khi dùng.
- Prompt mẫu: "Tôi đang học bài Mảng một chiều và chỉ số. Hãy đóng vai gia sư C17, đưa ra 3 gợi ý tăng dần cho bài tập: Nhập 10 điểm, tính trung bình, min/max, đếm điểm đạt. Không đưa lời giải hoàn chỉnh trước khi tôi thử; sau đó hãy review code của tôi về tính đúng, cảnh báo, ca biên và an toàn bộ nhớ."

Lỗi thường gặp

Truy cập ngoài mảng là undefined behavior; kiểm tra chỉ số.

Điều kiện hoàn thành

Tự giải thích được nội dung bài; hoàn tất bài tập chính; có code minh họa chạy được trong bối cảnh phù hợp; chứng minh đã thử trường hợp biên và sửa các warning quan trọng.

Ví dụ minh họa

Mảng một chiều và chỉ số

```
int scores[3] = {8, 9, 7};
for (size_t i = 0; i < 3; ++i) printf("%d ", scores[i]);
```

Mảng chứa các phần tử cùng kiểu, index 0..n-1, sizeof trong phạm vi phù hợp.

Checklist

[ ] Tôi có thể giải thích ý nghĩa của bài bằng ngôn ngữ của mình.
[ ] Đã thực hành: Nhập 10 điểm, tính trung bình, min/max, đếm điểm đạt.
[ ] Đã thử ca thông thường và ca biên; ghi rõ kết quả.
[ ] Đã đọc warning compiler và xử lý lỗi phù hợp.
[ ] Đã kiểm tra lỗi: Truy cập ngoài mảng là undefined behavior; kiểm tra chỉ số.
[ ] Đã tự xem và hiểu code, không chỉ sao chép lời giải AI.

Tài liệu tham khảo

C language reference — cppreference Tài liệu
https://www.cppreference.com/c/language

Nên học trước

Chia tệp .c/.h và liên kết

---

2. Mảng hai chiều và bảng dữ liệu

Duyệt row/column, truyền mảng 2D với kích thước chiều đúng.

Mục tiêu học tập

- Giải thích được: Duyệt row/column, truyền mảng 2D với kích thước chiều đúng.
- Thực hiện độc lập: Tạo bảng điểm 3 sinh viên x 4 môn và tính điểm trung bình từng hàng.
- Dự đoán, chạy và giải thích kết quả của ví dụ; nhận diện lỗi biên dịch và lỗi logic.
- Phòng tránh lỗi điển hình: Không nhầm thứ tự dòng/cột hoặc dùng chỉ số vượt giới hạn.

Kiến thức cần học

2d-array · row · column · matrix

Giải thích chi tiết

Mục tiêu của bài

Duyệt row/column, truyền mảng 2D với kích thước chiều đúng.

Kiến thức cốt lõi

- Bản chất: Duyệt row/column, truyền mảng 2D với kích thước chiều đúng.
- Cần nắm: 2d-array, row, column, matrix. Giải thích được từng thuật ngữ bằng ví dụ nhỏ, thay vì chỉ học thuộc cú pháp.
- Cách tư duy: Xác định dữ liệu đầu vào, phép biến đổi, đầu ra và những điều kiện có thể làm chương trình sai.
- Chất lượng: Ưu tiên code rõ ràng, tên dễ hiểu, cảnh báo biên dịch sạch và kiểm tra các trường hợp biên.

Ví dụ C / lệnh minh họa

```
int grid[2][3] = {{1, 2, 3}, {4, 5, 6}};
printf("%d\n", grid[1][2]);
```

Chú ý: Một số ví dụ là đoạn minh họa để đặt trong hàm hoặc tệp phù hợp, không phải mọi đoạn đều là chương trình hoàn chỉnh tự chạy.

Thực hành từng bước

1. Đọc ví dụ và dự đoán đầu ra hoặc trạng thái trước khi chạy.
2. Tạo một file bài tập riêng, tự gõ code và biên dịch với -std=c17 -Wall -Wextra -Wpedantic khi thích hợp.
3. Bài tập chính: Tạo bảng điểm 3 sinh viên x 4 môn và tính điểm trung bình từng hàng.
4. Thử ít nhất ba tình huống: đầu vào điển hình, ca biên và đầu vào sai nếu chương trình có nhập liệu.
5. Ghi lại warning/lỗi gặp phải, giải thích nguyên nhân và cách sửa bằng lời của mình.

Làm việc với AI đúng cách

- Có thể nhờ AI: yêu cầu giải thích khái niệm, đặt câu hỏi gợi mở, đề xuất test case và đọc warning đã loại bỏ thông tin nhạy cảm.
- Phải tự kiểm chứng: tự chạy code, đối chiếu với tài liệu, đọc từng dòng thay đổi, xác định giới hạn bộ nhớ và đầu vào trước khi dùng.
- Prompt mẫu: "Tôi đang học bài Mảng hai chiều và bảng dữ liệu. Hãy đóng vai gia sư C17, đưa ra 3 gợi ý tăng dần cho bài tập: Tạo bảng điểm 3 sinh viên x 4 môn và tính điểm trung bình từng hàng. Không đưa lời giải hoàn chỉnh trước khi tôi thử; sau đó hãy review code của tôi về tính đúng, cảnh báo, ca biên và an toàn bộ nhớ."

Lỗi thường gặp

Không nhầm thứ tự dòng/cột hoặc dùng chỉ số vượt giới hạn.

Điều kiện hoàn thành

Tự giải thích được nội dung bài; hoàn tất bài tập chính; có code minh họa chạy được trong bối cảnh phù hợp; chứng minh đã thử trường hợp biên và sửa các warning quan trọng.

Ví dụ minh họa

Mảng hai chiều và bảng dữ liệu

```
int grid[2][3] = {{1, 2, 3}, {4, 5, 6}};
printf("%d\n", grid[1][2]);
```

Duyệt row/column, truyền mảng 2D với kích thước chiều đúng.

Checklist

[ ] Tôi có thể giải thích ý nghĩa của bài bằng ngôn ngữ của mình.
[ ] Đã thực hành: Tạo bảng điểm 3 sinh viên x 4 môn và tính điểm trung bình từng hàng.
[ ] Đã thử ca thông thường và ca biên; ghi rõ kết quả.
[ ] Đã đọc warning compiler và xử lý lỗi phù hợp.
[ ] Đã kiểm tra lỗi: Không nhầm thứ tự dòng/cột hoặc dùng chỉ số vượt giới hạn.
[ ] Đã tự xem và hiểu code, không chỉ sao chép lời giải AI.

Tài liệu tham khảo

C language reference — cppreference Tài liệu
https://www.cppreference.com/c/language

Nên học trước

Mảng một chiều và chỉ số

---

3. Chuỗi C, ký tự kết thúc \0 và string.h

Chuỗi là mảng char kết thúc bởi nul; strlen đo độ dài, strcmp so sánh nội dung.

Mục tiêu học tập

- Giải thích được: Chuỗi là mảng char kết thúc bởi nul; strlen đo độ dài, strcmp so sánh nội dung.
- Thực hiện độc lập: Đọc tên bằng fgets, loại ký tự xuống dòng, so sánh hai tên.
- Dự đoán, chạy và giải thích kết quả của ví dụ; nhận diện lỗi biên dịch và lỗi logic.
- Phòng tránh lỗi điển hình: Không dùng == để so sánh chuỗi; cấp đủ chỗ cho \0.

Kiến thức cần học

null-terminator · strlen · strcmp · string.h

Giải thích chi tiết

Mục tiêu của bài

Chuỗi là mảng char kết thúc bởi nul; strlen đo độ dài, strcmp so sánh nội dung.

Kiến thức cốt lõi

- Bản chất: Chuỗi là mảng char kết thúc bởi nul; strlen đo độ dài, strcmp so sánh nội dung.
- Cần nắm: null-terminator, strlen, strcmp, string.h. Giải thích được từng thuật ngữ bằng ví dụ nhỏ, thay vì chỉ học thuộc cú pháp.
- Cách tư duy: Xác định dữ liệu đầu vào, phép biến đổi, đầu ra và những điều kiện có thể làm chương trình sai.
- Chất lượng: Ưu tiên code rõ ràng, tên dễ hiểu, cảnh báo biên dịch sạch và kiểm tra các trường hợp biên.

Ví dụ C / lệnh minh họa

```
char name[16] = "An";
printf("%zu\n", strlen(name));
```

Chú ý: Một số ví dụ là đoạn minh họa để đặt trong hàm hoặc tệp phù hợp, không phải mọi đoạn đều là chương trình hoàn chỉnh tự chạy.

Thực hành từng bước

1. Đọc ví dụ và dự đoán đầu ra hoặc trạng thái trước khi chạy.
2. Tạo một file bài tập riêng, tự gõ code và biên dịch với -std=c17 -Wall -Wextra -Wpedantic khi thích hợp.
3. Bài tập chính: Đọc tên bằng fgets, loại ký tự xuống dòng, so sánh hai tên.
4. Thử ít nhất ba tình huống: đầu vào điển hình, ca biên và đầu vào sai nếu chương trình có nhập liệu.
5. Ghi lại warning/lỗi gặp phải, giải thích nguyên nhân và cách sửa bằng lời của mình.

Làm việc với AI đúng cách

- Có thể nhờ AI: yêu cầu giải thích khái niệm, đặt câu hỏi gợi mở, đề xuất test case và đọc warning đã loại bỏ thông tin nhạy cảm.
- Phải tự kiểm chứng: tự chạy code, đối chiếu với tài liệu, đọc từng dòng thay đổi, xác định giới hạn bộ nhớ và đầu vào trước khi dùng.
- Prompt mẫu: "Tôi đang học bài Chuỗi C, ký tự kết thúc \0 và string.h. Hãy đóng vai gia sư C17, đưa ra 3 gợi ý tăng dần cho bài tập: Đọc tên bằng fgets, loại ký tự xuống dòng, so sánh hai tên. Không đưa lời giải hoàn chỉnh trước khi tôi thử; sau đó hãy review code của tôi về tính đúng, cảnh báo, ca biên và an toàn bộ nhớ."

Lỗi thường gặp

Không dùng == để so sánh chuỗi; cấp đủ chỗ cho \0.

Điều kiện hoàn thành

Tự giải thích được nội dung bài; hoàn tất bài tập chính; có code minh họa chạy được trong bối cảnh phù hợp; chứng minh đã thử trường hợp biên và sửa các warning quan trọng.

Ví dụ minh họa

Chuỗi C, ký tự kết thúc \0 và string.h

```
char name[16] = "An";
printf("%zu\n", strlen(name));
```

Chuỗi là mảng char kết thúc bởi nul; strlen đo độ dài, strcmp so sánh nội dung.

Checklist

[ ] Tôi có thể giải thích ý nghĩa của bài bằng ngôn ngữ của mình.
[ ] Đã thực hành: Đọc tên bằng fgets, loại ký tự xuống dòng, so sánh hai tên.
[ ] Đã thử ca thông thường và ca biên; ghi rõ kết quả.
[ ] Đã đọc warning compiler và xử lý lỗi phù hợp.
[ ] Đã kiểm tra lỗi: Không dùng == để so sánh chuỗi; cấp đủ chỗ cho \0.
[ ] Đã tự xem và hiểu code, không chỉ sao chép lời giải AI.

Tài liệu tham khảo

C language reference — cppreference Tài liệu
https://www.cppreference.com/c/language

Nên học trước

Mảng hai chiều và bảng dữ liệu

4. Thao tác chuỗi không tràn bộ đệm

Kiểm soát kích thước đích khi nối/copy, ưu tiên snprintf với nhu cầu định dạng.

Mục tiêu học tập

- Giải thích được: Kiểm soát kích thước đích khi nối/copy, ưu tiên snprintf với nhu cầu định dạng.
- Thực hiện độc lập: Ghép họ tên vào buffer có giới hạn; xử lý input quá dài.
- Dự đoán, chạy và giải thích kết quả của ví dụ; nhận diện lỗi biên dịch và lỗi logic.
- Phòng tránh lỗi điển hình: strcpy/strcat dễ tràn nếu không kiểm tra sức chứa; strncpy có thể không kết thúc nul.

Kiến thức cần học

buffer-overflow · snprintf · input-length

Giải thích chi tiết

Mục tiêu của bài

Kiểm soát kích thước đích khi nối/copy, ưu tiên snprintf với nhu cầu định dạng.

Kiến thức cốt lõi

- Bản chất: Kiểm soát kích thước đích khi nối/copy, ưu tiên snprintf với nhu cầu định dạng.
- Cần nắm: buffer-overflow, snprintf, input-length. Giải thích được từng thuật ngữ bằng ví dụ nhỏ, thay vì chỉ học thuộc cú pháp.
- Cách tư duy: Xác định dữ liệu đầu vào, phép biến đổi, đầu ra và những điều kiện có thể làm chương trình sai.
- Chất lượng: Ưu tiên code rõ ràng, tên dễ hiểu, cảnh báo biên dịch sạch và kiểm tra các trường hợp biên.

Ví dụ C / lệnh minh họa

```
char label[32];
int n = snprintf(label, sizeof label, "ID-%d", 42);
if (n < 0 || (size_t)n >= sizeof label) puts("Truncated");
```

Chú ý: Một số ví dụ là đoạn minh họa để đặt trong hàm hoặc tệp phù hợp, không phải mọi đoạn đều là chương trình hoàn chỉnh tự chạy.

Thực hành từng bước

1. Đọc ví dụ và dự đoán đầu ra hoặc trạng thái trước khi chạy.
2. Tạo một file bài tập riêng, tự gõ code và biên dịch với -std=c17 -Wall -Wextra -Wpedantic khi thích hợp.
3. Bài tập chính: Ghép họ tên vào buffer có giới hạn; xử lý input quá dài.
4. Thử ít nhất ba tình huống: đầu vào điển hình, ca biên và đầu vào sai nếu chương trình có nhập liệu.
5. Ghi lại warning/lỗi gặp phải, giải thích nguyên nhân và cách sửa bằng lời của mình.

Làm việc với AI đúng cách

- Có thể nhờ AI: yêu cầu giải thích khái niệm, đặt câu hỏi gợi mở, đề xuất test case và đọc warning đã loại bỏ thông tin nhạy cảm.
- Phải tự kiểm chứng: tự chạy code, đối chiếu với tài liệu, đọc từng dòng thay đổi, xác định giới hạn bộ nhớ và đầu vào trước khi dùng.

- Prompt mẫu: "Tôi đang học bài Thao tác chuỗi không tràn bộ đệm. Hãy đóng vai gia sư C17, đưa ra 3 gợi ý tăng dần cho bài tập: Ghép họ tên vào buffer có giới hạn; xử lý input quá dài. Không đưa lời giải hoàn chỉnh trước khi tôi thử; sau đó hãy review code của tôi về tính đúng, cảnh báo, ca biên và an toàn bộ nhớ."

Lỗi thường gặp

strcpy/strcat dễ tràn nếu không kiểm tra sức chứa; strncpy có thể không kết thúc nul.

Điều kiện hoàn thành

Tự giải thích được nội dung bài; hoàn tất bài tập chính; có code minh họa chạy được trong bối cảnh phù hợp; chứng minh đã thử trường hợp biên và sửa các warning quan trọng.

Ví dụ minh họa

Thao tác chuỗi không tràn bộ đệm

```
char label[32];
int n = snprintf(label, sizeof label, "ID-%d", 42);
if (n < 0 || (size_t)n >= sizeof label) puts("Truncated");
```

Kiểm soát kích thước đích khi nối/copy, ưu tiên snprintf với nhu cầu định dạng.

Checklist

[ ] Tôi có thể giải thích ý nghĩa của bài bằng ngôn ngữ của mình.

[ ] Đã thực hành: Ghép họ tên vào buffer có giới hạn; xử lý input quá dài.

[ ] Đã thử ca thông thường và ca biên; ghi rõ kết quả.

[ ] Đã đọc warning compiler và xử lý lỗi phù hợp.

[ ] Đã kiểm tra lỗi: strcpy/strcat dễ tràn nếu không kiểm tra sức chứa; strncpy có thể không kết thúc nul.

[ ] Đã tự xem và hiểu code, không chỉ sao chép lời giải AI.

Tài liệu tham khảo

C language reference — cppreference Tài liệu
https://www.cppreference.com/c/language

Nên học trước

Chuỗi C, ký tự kết thúc \0 và string.h

---

5. Thuật toán trên mảng: tìm kiếm và sắp xếp

Thực hành linear search, selection/bubble sort, độ phức tạp O(n), O(n^2); kiểm thử mảng rỗng.

Mục tiêu học tập

- Giải thích được: Thực hành linear search, selection/bubble sort, độ phức tạp O(n), O(n^2); kiểm thử mảng rỗng.
- Thực hiện độc lập: Cài linear search và selection sort, kiểm tra mảng tăng/giảm/trùng.
- Dự đoán, chạy và giải thích kết quả của ví dụ; nhận diện lỗi biên dịch và lỗi logic.
- Phòng tránh lỗi điển hình: Không dùng binary search trên mảng chưa sắp xếp; tránh index âm.

Kiến thức cần học

linear-search · sort · big-O

Giải thích chi tiết

Mục tiêu của bài

Thực hành linear search, selection/bubble sort, độ phức tạp O(n), O(n^2); kiểm thử mảng rỗng.

Kiến thức cốt lõi

- Bản chất: Thực hành linear search, selection/bubble sort, độ phức tạp O(n), $O(n^2)$; kiểm thử mảng rỗng.
- Cần nắm: linear-search, sort, big-O. Giải thích được từng thuật ngữ bằng ví dụ nhỏ, thay vì chỉ học thuộc cú pháp.
- Cách tư duy: Xác định dữ liệu đầu vào, phép biến đổi, đầu ra và những điều kiện có thể làm chương trình sai.
- Chất lượng: Ưu tiên code rõ ràng, tên dễ hiểu, cảnh báo biên dịch sạch và kiểm tra các trường hợp biên.

Ví dụ C / lệnh minh họa

```
int find(const int a[], size_t n, int target) {
    for (size_t i = 0; i < n; ++i) if (a[i] == target) return (int)i;
    return -1;
}
```

Chú ý: Một số ví dụ là đoạn minh họa để đặt trong hàm hoặc tệp phù hợp, không phải mọi đoạn đều là chương trình hoàn chỉnh tự chạy.

Thực hành từng bước

1. Đọc ví dụ và dự đoán đầu ra hoặc trạng thái trước khi chạy.
2. Tạo một file bài tập riêng, tự gõ code và biên dịch với -std=c17 -Wall -Wextra -Wpedantic khi thích hợp.
3. Bài tập chính: Cài linear search và selection sort, kiểm tra mảng tăng/giảm/trùng.
4. Thử ít nhất ba tình huống: đầu vào điển hình, ca biên và đầu vào sai nếu chương trình có nhập liệu.
5. Ghi lại warning/lỗi gặp phải, giải thích nguyên nhân và cách sửa bằng lời của mình.

Làm việc với AI đúng cách

- Có thể nhờ AI: yêu cầu giải thích khái niệm, đặt câu hỏi gợi mở, đề xuất test case và đọc warning đã loại bỏ thông tin nhạy cảm.
- Phải tự kiểm chứng: tự chạy code, đối chiếu với tài liệu, đọc từng dòng thay đổi, xác định giới hạn bộ nhớ và đầu vào trước khi dùng.
- Prompt mẫu: "Tôi đang học bài Thuật toán trên mảng: tìm kiếm và sắp xếp. Hãy đóng vai gia sư C17, đưa ra 3 gợi ý tăng dần cho bài tập: Cài linear search và selection sort, kiểm tra mảng tăng/giảm/trùng. Không đưa lời giải hoàn chỉnh trước khi tôi thử; sau đó hãy review code của tôi về tính đúng, cảnh báo, ca biên và an toàn bộ nhớ."

Lỗi thường gặp

Không dùng binary search trên mảng chưa sắp xếp; tránh index âm.

Điều kiện hoàn thành

Tự giải thích được nội dung bài; hoàn tất bài tập chính; có code minh họa chạy được trong bối cảnh phù hợp; chứng minh đã thử trường hợp biên và sửa các warning quan trọng.

Ví dụ minh họa

Thuật toán trên mảng: tìm kiếm và sắp xếp

```
int find(const int a[], size_t n, int target) {
    for (size_t i = 0; i < n; ++i) if (a[i] == target) return (int)i;
    return -1;
}
```

Thực hành linear search, selection/bubble sort, độ phức tạp O(n), O(n^2); kiểm thử mảng rỗng.

Checklist

- [ ] Tôi có thể giải thích ý nghĩa của bài bằng ngôn ngữ của mình.
- [ ] Đã thực hành: Cài linear search và selection sort, kiểm tra mảng tăng/giảm/trùng.
- [ ] Đã thử ca thông thường và ca biên; ghi rõ kết quả.
- [ ] Đã đọc warning compiler và xử lý lỗi phù hợp.
- [ ] Đã kiểm tra lỗi: Không dùng binary search trên mảng chưa sắp xếp; tránh index âm.
- [ ] Đã tự xem và hiểu code, không chỉ sao chép lời giải AI.

Tài liệu tham khảo

C language reference — cppreference Tài liệu
https://www.cppreference.com/c/language

Nên học trước

Thao tác chuỗi không tràn bộ đệm

GIAI ĐOẠN 6

Con trỏ và bộ nhớ

Giai đoạn 6: xây dựng năng lực qua 5 bài về con trỏ và bộ nhớ, kết hợp kiến thức, code và kiểm thử.

1. Địa chỉ, con trỏ, &, * và NULL

Phân biệt địa chỉ, giá trị tại địa chỉ, kiểu con trỏ và NULL; vẽ mô hình stack.

Mục tiêu học tập

- Giải thích được: Phân biệt địa chỉ, giá trị tại địa chỉ, kiểu con trỏ và NULL; vẽ mô hình stack.
- Thực hiện độc lập: Viết chương trình thay đổi biến qua int; in địa chỉ bằng %p (ép void).
- Dự đoán, chạy và giải thích kết quả của ví dụ; nhận diện lỗi biên dịch và lỗi logic.
- Phòng tránh lỗi điển hình: Không dereference NULL, con trỏ chưa khởi tạo hoặc đã hết hiệu lực.

Kiến thức cần học

pointer · address · dereference · NULL

Giải thích chi tiết

Mục tiêu của bài

Phân biệt địa chỉ, giá trị tại địa chỉ, kiểu con trỏ và NULL; vẽ mô hình stack.

Kiến thức cốt lõi

- Bản chất: Phân biệt địa chỉ, giá trị tại địa chỉ, kiểu con trỏ và NULL; vẽ mô hình stack.
- Cần nắm: pointer, address, dereference, NULL. Giải thích được từng thuật ngữ bằng ví dụ nhỏ, thay vì chỉ học thuộc cú pháp.
- Cách tư duy: Xác định dữ liệu đầu vào, phép biến đổi, đầu ra và những điều kiện có thể làm chương trình sai.
- Chất lượng: Ưu tiên code rõ ràng, tên dễ hiểu, cảnh báo biên dịch sạch và kiểm tra các trường hợp biên.

Ví dụ C / lệnh minh họa

```
int value = 10;
int *p = &value;
*p = 20;
printf("%d\n", value);
```

Chú ý: Một số ví dụ là đoạn minh họa để đặt trong hàm hoặc tệp phù hợp, không phải mọi đoạn đều là chương trình hoàn chỉnh tự chạy.

Thực hành từng bước

1. Đọc ví dụ và dự đoán đầu ra hoặc trạng thái trước khi chạy.
2. Tạo một file bài tập riêng, tự gõ code và biên dịch với -std=c17 -Wall -Wextra -Wpedantic khi thích hợp.
3. Bài tập chính: Viết chương trình thay đổi biến qua int; in địa chỉ bằng %p (ép void).
4. Thử ít nhất ba tình huống: đầu vào điển hình, ca biên và đầu vào sai nếu chương trình có nhập liệu.
5. Ghi lại warning/lỗi gặp phải, giải thích nguyên nhân và cách sửa bằng lời của mình.

Làm việc với AI đúng cách

- Có thể nhờ AI: yêu cầu giải thích khái niệm, đặt câu hỏi gợi mở, đề xuất test case và đọc warning đã loại bỏ thông tin nhạy cảm.
- Phải tự kiểm chứng: tự chạy code, đối chiếu với tài liệu, đọc từng dòng thay đổi, xác định giới hạn bộ nhớ và đầu vào trước khi dùng.
- Prompt mẫu: "Tôi đang học bài Địa chỉ, con trỏ, &, và NULL. Hãy đóng vai gia sư C17, đưa ra 3 gợi ý tăng dần cho bài tập: Viết chương trình thay đổi biến qua int; in địa chỉ bằng %p (ép void*). Không đưa lời giải hoàn chỉnh trước khi tôi thử; sau đó hãy review code của tôi về tính đúng, cảnh báo, ca biên và an toàn bộ nhớ."

Lỗi thường gặp

Không dereference NULL, con trỏ chưa khởi tạo hoặc đã hết hiệu lực.

Điều kiện hoàn thành

Tự giải thích được nội dung bài; hoàn tất bài tập chính; có code minh họa chạy được trong bối cảnh phù hợp; chứng minh đã thử trường hợp biên và sửa các warning quan trọng.

Ví dụ minh họa

Địa chỉ, con trỏ, &, * và NULL

```
int value = 10;
int *p = &value;
*p = 20;
printf("%d\n", value);
```

Phân biệt địa chỉ, giá trị tại địa chỉ, kiểu con trỏ và NULL; vẽ mô hình stack.

Checklist

[ ] Tôi có thể giải thích ý nghĩa của bài bằng ngôn ngữ của mình.

[ ] Đã thực hành: Viết chương trình thay đổi biến qua int; in địa chỉ bằng %p (ép void).

[ ] Đã thử ca thông thường và ca biên; ghi rõ kết quả.

[ ] Đã đọc warning compiler và xử lý lỗi phù hợp.

[ ] Đã kiểm tra lỗi: Không dereference NULL, con trỏ chưa khởi tạo hoặc đã hết hiệu lực.

[ ] Đã tự xem và hiểu code, không chỉ sao chép lời giải AI.

Tài liệu tham khảo

C language reference — cppreference Tài liệu
https://www.cppreference.com/c/language

C dynamic memory management Tài liệu
https://en.cppreference.com/w/c/memory

Nên học trước

Thuật toán trên mảng: tìm kiếm và sắp xếp

---

2. Quan hệ mảng–con trỏ và pointer arithmetic

Mảng trong nhiều biểu thức chuyển thành con trỏ phần tử đầu, nhưng không đồng nhất với con trỏ; phép toán con trỏ có giới hạn.

Mục tiêu học tập

- Giải thích được: Mảng trong nhiều biểu thức chuyển thành con trỏ phần tử đầu, nhưng không đồng nhất với con trỏ; phép toán con trỏ có giới hạn.

- Thực hiện độc lập: Duyệt mảng bằng index và con trỏ, so sánh sizeof(array) tại nơi khai báo với tham số hàm.
- Dự đoán, chạy và giải thích kết quả của ví dụ; nhận diện lỗi biên dịch và lỗi logic.
- Phòng tránh lỗi điển hình: Không truy cập ngoài một-past-end; không dùng sizeof(pointer) để tính độ dài mảng.

Kiến thức cần học

array-decay · pointer-arithmetic · one-past-end

Giải thích chi tiết

Mục tiêu của bài

Mảng trong nhiều biểu thức chuyển thành con trỏ phần tử đầu, nhưng không đồng nhất với con trỏ; phép toán con trỏ có giới hạn.

Kiến thức cốt lõi

- Bản chất: Mảng trong nhiều biểu thức chuyển thành con trỏ phần tử đầu, nhưng không đồng nhất với con trỏ; phép toán con trỏ có giới hạn.
- Cần nắm: array-decay, pointer-arithmetic, one-past-end. Giải thích được từng thuật ngữ bằng ví dụ nhỏ, thay vì chỉ học thuộc cú pháp.
- Cách tư duy: Xác định dữ liệu đầu vào, phép biến đổi, đầu ra và những điều kiện có thể làm chương trình sai.
- Chất lượng: Ưu tiên code rõ ràng, tên dễ hiểu, cảnh báo biên dịch sạch và kiểm tra các trường hợp biên.

Ví dụ C / lệnh minh họa

```
int a[] = {1, 2, 3};
int *p = a;
printf("%d\n", *(p + 1));
```

Chú ý: Một số ví dụ là đoạn minh họa để đặt trong hàm hoặc tệp phù hợp, không phải mọi đoạn đều là chương trình hoàn chỉnh tự chạy.

Thực hành từng bước

1. Đọc ví dụ và dự đoán đầu ra hoặc trạng thái trước khi chạy.
2. Tạo một file bài tập riêng, tự gõ code và biên dịch với -std=c17 -Wall -Wextra -Wpedantic khi thích hợp.
3. Bài tập chính: Duyệt mảng bằng index và con trỏ, so sánh sizeof(array) tại nơi khai báo với tham số hàm.
4. Thử ít nhất ba tình huống: đầu vào điển hình, ca biên và đầu vào sai nếu chương trình có nhập liệu.
5. Ghi lại warning/lỗi gặp phải, giải thích nguyên nhân và cách sửa bằng lời của mình.

Làm việc với AI đúng cách

- Có thể nhờ AI: yêu cầu giải thích khái niệm, đặt câu hỏi gợi mở, đề xuất test case và đọc warning đã loại bỏ thông tin nhạy cảm.
- Phải tự kiểm chứng: tự chạy code, đối chiếu với tài liệu, đọc từng dòng thay đổi, xác định giới hạn bộ nhớ và đầu vào trước khi dùng.

- Prompt mẫu: "Tôi đang học bài Quan hệ mảng–con trỏ và pointer arithmetic. Hãy đóng vai gia sư C17, đưa ra 3 gợi ý tăng dần cho bài tập: Duyệt mảng bằng index và con trỏ, so sánh sizeof(array) tại nơi khai báo với tham số hàm. Không đưa lời giải hoàn chỉnh trước khi tôi thử; sau đó hãy review code của tôi về tính đúng, cảnh báo, ca biên và an toàn bộ nhớ."

Lỗi thường gặp

Không truy cập ngoài một-past-end; không dùng sizeof(pointer) để tính độ dài mảng.

Điều kiện hoàn thành

Tự giải thích được nội dung bài; hoàn tất bài tập chính; có code minh họa chạy được trong bối cảnh phù hợp; chứng minh đã thử trường hợp biên và sửa các warning quan trọng.

Ví dụ minh họa

Quan hệ mảng–con trỏ và pointer arithmetic

```
int a[] = {1, 2, 3};
int *p = a;
printf("%d\n", *(p + 1));
```

Mảng trong nhiều biểu thức chuyển thành con trỏ phần tử đầu, nhưng không đồng nhất với con trỏ; phép toán con trỏ có giới hạn.

Checklist

- [ ] Tôi có thể giải thích ý nghĩa của bài bằng ngôn ngữ của mình.
- [ ] Đã thực hành: Duyệt mảng bằng index và con trỏ, so sánh sizeof(array) tại nơi khai báo với tham số hàm.
- [ ] Đã thử ca thông thường và ca biên; ghi rõ kết quả.
- [ ] Đã đọc warning compiler và xử lý lỗi phù hợp.
- [ ] Đã kiểm tra lỗi: Không truy cập ngoài một-past-end; không dùng sizeof(pointer) để tính độ dài mảng.
- [ ] Đã tự xem và hiểu code, không chỉ sao chép lời giải AI.

Tài liệu tham khảo

C language reference — cppreference Tài liệu
https://www.cppreference.com/c/language

C dynamic memory management Tài liệu
https://en.cppreference.com/w/c/memory

Nên học trước

Địa chỉ, con trỏ, &, * và NULL

---

3. Stack, heap, lifetime và undefined behavior

Mô hình khái niệm automatic/static/dynamic storage, phân biệt leak, dangling pointer, use-after-free, UB.

Mục tiêu học tập

- Giải thích được: Mô hình khái niệm automatic/static/dynamic storage, phân biệt leak, dangling pointer, use-after-free, UB.
- Thực hiện độc lập: Phân tích 5 đoạn code và đánh dấu con trỏ nào còn hợp lệ.
- Dự đoán, chạy và giải thích kết quả của ví dụ; nhận diện lỗi biên dịch và lỗi logic.
- Phòng tránh lỗi điển hình: Không nghĩ mọi biến địa phương chắc chắn nằm vật lý trên stack; đây là mô hình khái niệm.

Kiến thức cần học

stack · heap · lifetime · undefined-behavior

Giải thích chi tiết

Mục tiêu của bài

Mô hình khái niệm automatic/static/dynamic storage, phân biệt leak, dangling pointer, use-after-free, UB.

Kiến thức cốt lõi

- Bản chất: Mô hình khái niệm automatic/static/dynamic storage, phân biệt leak, dangling pointer, use-after-free, UB.
- Cần nắm: stack, heap, lifetime, undefined-behavior. Giải thích được từng thuật ngữ bằng ví dụ nhỏ, thay vì chỉ học thuộc cú pháp.
- Cách tư duy: Xác định dữ liệu đầu vào, phép biến đổi, đầu ra và những điều kiện có thể làm chương trình sai.
- Chất lượng: Ưu tiên code rõ ràng, tên dễ hiểu, cảnh báo biên dịch sạch và kiểm tra các trường hợp biên.

Ví dụ C / lệnh minh họa

```
/* Khong tra ve dia chi bien dia phuong */
/* int *bad(void) { int x = 1; return &x; } */
```

Chú ý: Một số ví dụ là đoạn minh họa để đặt trong hàm hoặc tệp phù hợp, không phải mọi đoạn đều là chương trình hoàn chỉnh tự chạy.

Thực hành từng bước

1. Đọc ví dụ và dự đoán đầu ra hoặc trạng thái trước khi chạy.
2. Tạo một file bài tập riêng, tự gõ code và biên dịch với -std=c17 -Wall -Wextra -Wpedantic khi thích hợp.
3. Bài tập chính: Phân tích 5 đoạn code và đánh dấu con trỏ nào còn hợp lệ.
4. Thử ít nhất ba tình huống: đầu vào điển hình, ca biên và đầu vào sai nếu chương trình có nhập liệu.
5. Ghi lại warning/lỗi gặp phải, giải thích nguyên nhân và cách sửa bằng lời của mình.

Làm việc với AI đúng cách

- Có thể nhờ AI: yêu cầu giải thích khái niệm, đặt câu hỏi gợi mở, đề xuất test case và đọc warning đã loại bỏ thông tin nhạy cảm.
- Phải tự kiểm chứng: tự chạy code, đối chiếu với tài liệu, đọc từng dòng thay đổi, xác định giới hạn bộ nhớ và đầu vào trước khi dùng.
- Prompt mẫu: "Tôi đang học bài Stack, heap, lifetime và undefined behavior. Hãy đóng vai gia sư C17, đưa ra 3 gợi ý tăng dần cho bài tập: Phân tích 5 đoạn code và đánh dấu con trỏ nào còn hợp lệ. Không đưa lời giải hoàn chỉnh trước khi tôi thử; sau đó hãy review code của tôi về tính đúng, cảnh báo, ca biên và an toàn bộ nhớ."

Lỗi thường gặp

Không nghĩ mọi biến địa phương chắc chắn nằm vật lý trên stack; đây là mô hình khái niệm.

Điều kiện hoàn thành

Tự giải thích được nội dung bài; hoàn tất bài tập chính; có code minh họa chạy được trong bối cảnh phù hợp; chứng minh đã thử trường hợp biên và sửa các warning quan trọng.

Ví dụ minh họa

Stack, heap, lifetime và undefined behavior

```
/* Khong tra ve dia chi bien dia phuong */
/* int *bad(void) { int x = 1; return &x; } */
```

Mô hình khái niệm automatic/static/dynamic storage, phân biệt leak, dangling pointer, use-after-free, UB.

Checklist

- [ ] Tôi có thể giải thích ý nghĩa của bài bằng ngôn ngữ của mình.
- [ ] Đã thực hành: Phân tích 5 đoạn code và đánh dấu con trỏ nào còn hợp lệ.
- [ ] Đã thử ca thông thường và ca biên; ghi rõ kết quả.
- [ ] Đã đọc warning compiler và xử lý lỗi phù hợp.
- [ ] Đã kiểm tra lỗi: Không nghĩ mọi biến địa phương chắc chắn nằm vật lý trên stack; đây là mô hình khái niệm.
- [ ] Đã tự xem và hiểu code, không chỉ sao chép lời giải AI.

Tài liệu tham khảo

C language reference — cppreference Tài liệu
https://www.cppreference.com/c/language

C dynamic memory management Tài liệu
https://en.cppreference.com/w/c/memory

Nên học trước

Quan hệ mảng–con trỏ và pointer arithmetic

---

4. malloc, calloc, realloc và free an toàn

Kiểm tra NULL, dùng sizeof *ptr, tránh double free; với realloc dùng con trỏ tạm.

Mục tiêu học tập

- Giải thích được: Kiểm tra NULL, dùng sizeof *ptr, tránh double free; với realloc dùng con trỏ tạm.
- Thực hiện độc lập: Tạo mảng n số nhập từ người dùng, tính tổng, giải phóng đúng một lần.
- Dự đoán, chạy và giải thích kết quả của ví dụ; nhận diện lỗi biên dịch và lỗi logic.
- Phòng tránh lỗi điển hình: Không quên kiểm tra overflow kích thước cấp phát; free đúng vùng đã malloc.

Kiến thức cần học

malloc · calloc · realloc · free

Giải thích chi tiết

Mục tiêu của bài

Kiểm tra NULL, dùng sizeof *ptr, tránh double free; với realloc dùng con trỏ tạm.

Kiến thức cốt lõi

- Bản chất: Kiểm tra NULL, dùng sizeof *ptr, tránh double free; với realloc dùng con trỏ tạm.
- Cần nắm: malloc, calloc, realloc, free. Giải thích được từng thuật ngữ bằng ví dụ nhỏ, thay vì chỉ học thuộc cú pháp.
- Cách tư duy: Xác định dữ liệu đầu vào, phép biến đổi, đầu ra và những điều kiện có thể làm chương trình sai.

- Chất lượng: Ưu tiên code rõ ràng, tên dễ hiểu, cảnh báo biên dịch sạch và kiểm tra các trường hợp biên.

Ví dụ C / lệnh minh họa

```
size_t n = 5;
int *a = malloc(n * sizeof *a);
if (a == NULL) return 1;
free(a);
a = NULL;
```

Chú ý: Một số ví dụ là đoạn minh họa để đặt trong hàm hoặc tệp phù hợp, không phải mọi đoạn đều là chương trình hoàn chỉnh tự chạy.

Thực hành từng bước

1. Đọc ví dụ và dự đoán đầu ra hoặc trạng thái trước khi chạy.
2. Tạo một file bài tập riêng, tự gõ code và biên dịch với -std=c17 -Wall -Wextra -Wpedantic khi thích hợp.
3. Bài tập chính: Tạo mảng n số nhập từ người dùng, tính tổng, giải phóng đúng một lần.
4. Thử ít nhất ba tình huống: đầu vào điển hình, ca biên và đầu vào sai nếu chương trình có nhập liệu.
5. Ghi lại warning/lỗi gặp phải, giải thích nguyên nhân và cách sửa bằng lời của mình.

Làm việc với AI đúng cách

- Có thể nhờ AI: yêu cầu giải thích khái niệm, đặt câu hỏi gợi mở, đề xuất test case và đọc warning đã loại bỏ thông tin nhạy cảm.
- Phải tự kiểm chứng: tự chạy code, đối chiếu với tài liệu, đọc từng dòng thay đổi, xác định giới hạn bộ nhớ và đầu vào trước khi dùng.
- Prompt mẫu: "Tôi đang học bài malloc, calloc, realloc và free an toàn. Hãy đóng vai gia sư C17, đưa ra 3 gợi ý tăng dần cho bài tập: Tạo mảng n số nhập từ người dùng, tính tổng, giải phóng đúng một lần. Không đưa lời giải hoàn chỉnh trước khi tôi thử; sau đó hãy review code của tôi về tính đúng, cảnh báo, ca biên và an toàn bộ nhớ."

Lỗi thường gặp

Không quên kiểm tra overflow kích thước cấp phát; free đúng vùng đã malloc.

Điều kiện hoàn thành

Tự giải thích được nội dung bài; hoàn tất bài tập chính; có code minh họa chạy được trong bối cảnh phù hợp; chứng minh đã thử trường hợp biên và sửa các warning quan trọng.

Ví dụ minh họa

malloc, calloc, realloc và free an toàn

```
size_t n = 5;
int *a = malloc(n * sizeof *a);
if (a == NULL) return 1;
free(a);
a = NULL;
```

*Kiểm tra NULL, dùng sizeof *ptr, tránh double free; với realloc dùng con trỏ tạm.*

Checklist

[ ] Tôi có thể giải thích ý nghĩa của bài bằng ngôn ngữ của mình.

[ ] Đã thực hành: Tạo mảng n số nhập từ người dùng, tính tổng, giải phóng đúng một lần.

[ ] Đã thử ca thông thường và ca biên; ghi rõ kết quả.

[ ] Đã đọc warning compiler và xử lý lỗi phù hợp.

[ ] Đã kiểm tra lỗi: Không quên kiểm tra overflow kích thước cấp phát; free đúng vùng đã malloc.

[ ] Đã tự xem và hiểu code, không chỉ sao chép lời giải AI.

Tài liệu tham khảo

C language reference — cppreference Tài liệu
https://www.cppreference.com/c/language

C dynamic memory management Tài liệu
https://en.cppreference.com/w/c/memory

Nên học trước

Stack, heap, lifetime và undefined behavior

---

5. Dùng compiler warnings và sanitizers

Bật -Wall -Wextra -Wpedantic và AddressSanitizer/UBSan khi môi trường hỗ trợ.

Mục tiêu học tập

- Giải thích được: Bật -Wall -Wextra -Wpedantic và AddressSanitizer/UBSan khi môi trường hỗ trợ.
- Thực hiện độc lập: Cố ý tạo lỗi truy cập a[3] khi a có 3 phần tử trong dự án test, quan sát sanitizer và sửa.
- Dự đoán, chạy và giải thích kết quả của ví dụ; nhận diện lỗi biên dịch và lỗi logic.
- Phòng tránh lỗi điển hình: Sanitizer không chứng minh chương trình không có bug; không chạy ví dụ lỗi với dữ liệu thật.

Kiến thức cần học

asan · ubsan · compiler-warnings · debugging

Giải thích chi tiết

Mục tiêu của bài

Bật -Wall -Wextra -Wpedantic và AddressSanitizer/UBSan khi môi trường hỗ trợ.

Kiến thức cốt lõi

- Bản chất: Bật -Wall -Wextra -Wpedantic và AddressSanitizer/UBSan khi môi trường hỗ trợ.
- Cần nắm: asan, ubsan, compiler-warnings, debugging. Giải thích được từng thuật ngữ bằng ví dụ nhỏ, thay vì chỉ học thuộc cú pháp.
- Cách tư duy: Xác định dữ liệu đầu vào, phép biến đổi, đầu ra và những điều kiện có thể làm chương trình sai.
- Chất lượng: Ưu tiên code rõ ràng, tên dễ hiểu, cảnh báo biên dịch sạch và kiểm tra các trường hợp biên.

Ví dụ C / lệnh minh họa

```
gcc -std=c17 -g -O1 -fsanitize=address,undefined -fno-omit-frame-pointer main.c -o app
```

Chú ý: Một số ví dụ là đoạn minh họa để đặt trong hàm hoặc tệp phù hợp, không phải mọi đoạn đều là chương trình hoàn chỉnh tự chạy.

Thực hành từng bước

1. Đọc ví dụ và dự đoán đầu ra hoặc trạng thái trước khi chạy.

2. Tạo một file bài tập riêng, tự gõ code và biên dịch với -std=c17 -Wall -Wextra -Wpedantic khi thích hợp.
3. Bài tập chính: Cố ý tạo lỗi truy cập a[3] khi a có 3 phần tử trong dự án test, quan sát sanitizer và sửa.
4. Thử ít nhất ba tình huống: đầu vào điển hình, ca biên và đầu vào sai nếu chương trình có nhập liệu.
5. Ghi lại warning/lỗi gặp phải, giải thích nguyên nhân và cách sửa bằng lời của mình.

Làm việc với AI đúng cách

- Có thể nhờ AI: yêu cầu giải thích khái niệm, đặt câu hỏi gợi mở, đề xuất test case và đọc warning đã loại bỏ thông tin nhạy cảm.
- Phải tự kiểm chứng: tự chạy code, đối chiếu với tài liệu, đọc từng dòng thay đổi, xác định giới hạn bộ nhớ và đầu vào trước khi dùng.
- Prompt mẫu: "Tôi đang học bài Dùng compiler warnings và sanitizers. Hãy đóng vai gia sư C17, đưa ra 3 gợi ý tăng dần cho bài tập: Cố ý tạo lỗi truy cập a[3] khi a có 3 phần tử trong dự án test, quan sát sanitizer và sửa. Không đưa lời giải hoàn chỉnh trước khi tôi thử; sau đó hãy review code của tôi về tính đúng, cảnh báo, ca biên và an toàn bộ nhớ."

Lỗi thường gặp

Sanitizer không chứng minh chương trình không có bug; không chạy ví dụ lỗi với dữ liệu thật.

Điều kiện hoàn thành

Tự giải thích được nội dung bài; hoàn tất bài tập chính; có code minh họa chạy được trong bối cảnh phù hợp; chứng minh đã thử trường hợp biên và sửa các warning quan trọng.

Ví dụ minh họa

Dùng compiler warnings và sanitizers

```
gcc -std=c17 -g -O1 -fsanitize=address,undefined -fno-omit-frame-pointer main.c -o app
```

Bật -Wall -Wextra -Wpedantic và AddressSanitizer/UBSan khi môi trường hỗ trợ.

Checklist

- [ ] Tôi có thể giải thích ý nghĩa của bài bằng ngôn ngữ của mình.
- [ ] Đã thực hành: Cố ý tạo lỗi truy cập a[3] khi a có 3 phần tử trong dự án test, quan sát sanitizer và sửa.
- [ ] Đã thử ca thông thường và ca biên; ghi rõ kết quả.
- [ ] Đã đọc warning compiler và xử lý lỗi phù hợp.
- [ ] Đã kiểm tra lỗi: Sanitizer không chứng minh chương trình không có bug; không chạy ví dụ lỗi với dữ liệu thật.
- [ ] Đã tự xem và hiểu code, không chỉ sao chép lời giải AI.

Tài liệu tham khảo

C language reference — cppreference Tài liệu
https://www.cppreference.com/c/language

C dynamic memory management Tài liệu
https://en.cppreference.com/w/c/memory

Nên học trước

malloc, calloc, realloc và free an toàn

GIAI ĐOẠN 7

struct, tệp và tiền xử lý

Giai đoạn 7: xây dựng năng lực qua 5 bài về struct, tệp và tiền xử lý, kết hợp kiến thức, code và kiểm thử.

1. struct và mô hình hóa dữ liệu

Định nghĩa cấu trúc gồm nhiều trường, dùng ., -> và khởi tạo rõ ràng.

Mục tiêu học tập

- Giải thích được: Định nghĩa cấu trúc gồm nhiều trường, dùng ., -> và khởi tạo rõ ràng.
- Thực hiện độc lập: Tạo struct Student { id, name, score }, nhập và in thông tin.
- Dự đoán, chạy và giải thích kết quả của ví dụ; nhận diện lỗi biên dịch và lỗi logic.
- Phòng tránh lỗi điển hình: Không so sánh hai struct bằng ==; chú ý kích thước mảng char.

Kiến thức cần học

struct · member · dot · arrow

Giải thích chi tiết

Mục tiêu của bài

Định nghĩa cấu trúc gồm nhiều trường, dùng ., -> và khởi tạo rõ ràng.

Kiến thức cốt lõi

- Bản chất: Định nghĩa cấu trúc gồm nhiều trường, dùng ., -> và khởi tạo rõ ràng.
- Cần nắm: struct, member, dot, arrow. Giải thích được từng thuật ngữ bằng ví dụ nhỏ, thay vì chỉ học thuộc cú pháp.
- Cách tư duy: Xác định dữ liệu đầu vào, phép biến đổi, đầu ra và những điều kiện có thể làm chương trình sai.
- Chất lượng: Ưu tiên code rõ ràng, tên dễ hiểu, cảnh báo biên dịch sạch và kiểm tra các trường hợp biên.

Ví dụ C / lệnh minh họa

```
typedef struct {
    int id;
    char name[40];
    double score;
} Student;
```

Chú ý: Một số ví dụ là đoạn minh họa để đặt trong hàm hoặc tệp phù hợp, không phải mọi đoạn đều là chương trình hoàn chỉnh tự chạy.

Thực hành từng bước

1. Đọc ví dụ và dự đoán đầu ra hoặc trạng thái trước khi chạy.
2. Tạo một file bài tập riêng, tự gõ code và biên dịch với -std=c17 -Wall -Wextra -Wpedantic khi thích hợp.
3. Bài tập chính: Tạo struct Student { id, name, score }, nhập và in thông tin.
4. Thử ít nhất ba tình huống: đầu vào điển hình, ca biên và đầu vào sai nếu chương trình có nhập liệu.
5. Ghi lại warning/lỗi gặp phải, giải thích nguyên nhân và cách sửa bằng lời của mình.

Làm việc với AI đúng cách

- Có thể nhờ AI: yêu cầu giải thích khái niệm, đặt câu hỏi gợi mở, đề xuất test case và đọc warning đã loại bỏ thông tin nhạy cảm.
- Phải tự kiểm chứng: tự chạy code, đối chiếu với tài liệu, đọc từng dòng thay đổi, xác định giới hạn bộ nhớ và đầu vào trước khi dùng.
- Prompt mẫu: "Tôi đang học bài struct và mô hình hóa dữ liệu. Hãy đóng vai gia sư C17, đưa ra 3 gợi ý tăng dần cho bài tập: Tạo struct Student { id, name, score }, nhập và in thông tin. Không đưa lời giải hoàn chỉnh trước khi tôi thử; sau đó hãy review code của tôi về tính đúng, cảnh báo, ca biên và an toàn bộ nhớ."

Lỗi thường gặp

Không so sánh hai struct bằng ==; chú ý kích thước mảng char.

Điều kiện hoàn thành

Tự giải thích được nội dung bài; hoàn tất bài tập chính; có code minh họa chạy được trong bối cảnh phù hợp; chứng minh đã thử trường hợp biên và sửa các warning quan trọng.

Ví dụ minh họa

struct và mô hình hóa dữ liệu

```
typedef struct {
    int id;
    char name[40];
    double score;
} Student;
```

Định nghĩa cấu trúc gồm nhiều trường, dùng ., -> và khởi tạo rõ ràng.

Checklist

[ ] Tôi có thể giải thích ý nghĩa của bài bằng ngôn ngữ của mình.

[ ] Đã thực hành: Tạo struct Student { id, name, score }, nhập và in thông tin.

[ ] Đã thử ca thông thường và ca biên; ghi rõ kết quả.

[ ] Đã đọc warning compiler và xử lý lỗi phù hợp.

[ ] Đã kiểm tra lỗi: Không so sánh hai struct bằng ==; chú ý kích thước mảng char.

[ ] Đã tự xem và hiểu code, không chỉ sao chép lời giải AI.

Tài liệu tham khảo

C language reference — cppreference Tài liệu
https://www.cppreference.com/c/language

C input/output library Tài liệu
https://en.cppreference.com/w/c/io

Nên học trước

Dùng compiler warnings và sanitizers

---

2. enum, typedef và hằng số có tên

Dùng enum cho trạng thái, const cho dữ liệu bất biến, tránh magic numbers.

Mục tiêu học tập

- Giải thích được: Dùng enum cho trạng thái, const cho dữ liệu bất biến, tránh magic numbers.

- Thực hiện độc lập: Định nghĩa trạng thái bài tập TODO/DOING/DONE và chuyển trạng thái bằng switch.
- Dự đoán, chạy và giải thích kết quả của ví dụ; nhận diện lỗi biên dịch và lỗi logic.
- Phòng tránh lỗi điển hình: Không giả định enum tự động ngăn mọi giá trị không hợp lệ.

Kiến thức cần học

enum · typedef · const · magic-number

Giải thích chi tiết

Mục tiêu của bài

Dùng enum cho trạng thái, const cho dữ liệu bất biến, tránh magic numbers.

Kiến thức cốt lõi

- Bản chất: Dùng enum cho trạng thái, const cho dữ liệu bất biến, tránh magic numbers.
- Cần nắm: enum, typedef, const, magic-number. Giải thích được từng thuật ngữ bằng ví dụ nhỏ, thay vì chỉ học thuộc cú pháp.
- Cách tư duy: Xác định dữ liệu đầu vào, phép biến đổi, đầu ra và những điều kiện có thể làm chương trình sai.
- Chất lượng: Ưu tiên code rõ ràng, tên dễ hiểu, cảnh báo biên dịch sạch và kiểm tra các trường hợp biên.

Ví dụ C / lệnh minh họa

```
typedef enum { TODO, DOING, DONE } Status;
Status s = TODO;
```

Chú ý: Một số ví dụ là đoạn minh họa để đặt trong hàm hoặc tệp phù hợp, không phải mọi đoạn đều là chương trình hoàn chỉnh tự chạy.

Thực hành từng bước

1. Đọc ví dụ và dự đoán đầu ra hoặc trạng thái trước khi chạy.
2. Tạo một file bài tập riêng, tự gõ code và biên dịch với -std=c17 -Wall -Wextra -Wpedantic khi thích hợp.
3. Bài tập chính: Định nghĩa trạng thái bài tập TODO/DOING/DONE và chuyển trạng thái bằng switch.
4. Thử ít nhất ba tình huống: đầu vào điển hình, ca biên và đầu vào sai nếu chương trình có nhập liệu.
5. Ghi lại warning/lỗi gặp phải, giải thích nguyên nhân và cách sửa bằng lời của mình.

Làm việc với AI đúng cách

- Có thể nhờ AI: yêu cầu giải thích khái niệm, đặt câu hỏi gợi mở, đề xuất test case và đọc warning đã loại bỏ thông tin nhạy cảm.
- Phải tự kiểm chứng: tự chạy code, đối chiếu với tài liệu, đọc từng dòng thay đổi, xác định giới hạn bộ nhớ và đầu vào trước khi dùng.
- Prompt mẫu: "Tôi đang học bài enum, typedef và hằng số có tên. Hãy đóng vai gia sư C17, đưa ra 3 gợi ý tăng dần cho bài tập: Định nghĩa trạng thái bài tập TODO/DOING/DONE và chuyển trạng thái bằng switch. Không đưa lời giải hoàn chỉnh trước khi tôi thử; sau đó hãy review code của tôi về tính đúng, cảnh báo, ca biên và an toàn bộ nhớ."

Lỗi thường gặp

Không giả định enum tự động ngăn mọi giá trị không hợp lệ.

Điều kiện hoàn thành

Tự giải thích được nội dung bài; hoàn tất bài tập chính; có code minh họa chạy được trong bối cảnh phù hợp; chứng minh đã thử trường hợp biên và sửa các warning quan trọng.

Ví dụ minh họa

enum, typedef và hằng số có tên

```
typedef enum { TODO, DOING, DONE } Status;
Status s = TODO;
```

Dùng enum cho trạng thái, const cho dữ liệu bất biến, tránh magic numbers.

Checklist

[ ] Tôi có thể giải thích ý nghĩa của bài bằng ngôn ngữ của mình.
[ ] Đã thực hành: Định nghĩa trạng thái bài tập TODO/DOING/DONE và chuyển trạng thái bằng switch.
[ ] Đã thử ca thông thường và ca biên; ghi rõ kết quả.
[ ] Đã đọc warning compiler và xử lý lỗi phù hợp.
[ ] Đã kiểm tra lỗi: Không giả định enum tự động ngăn mọi giá trị không hợp lệ.
[ ] Đã tự xem và hiểu code, không chỉ sao chép lời giải AI.

Tài liệu tham khảo

C language reference — cppreference Tài liệu
https://www.cppreference.com/c/language

C input/output library Tài liệu
https://en.cppreference.com/w/c/io

Nên học trước

struct và mô hình hóa dữ liệu

---

3. Đọc ghi file văn bản với stdio

fopen, fgets/fputs, fprintf, fclose, xử lý NULL và trạng thái lỗi.

Mục tiêu học tập

- Giải thích được: fopen, fgets/fputs, fprintf, fclose, xử lý NULL và trạng thái lỗi.
- Thực hiện độc lập: Lưu danh sách sinh viên ra students.txt và nạp lại từ file.
- Dự đoán, chạy và giải thích kết quả của ví dụ; nhận diện lỗi biên dịch và lỗi logic.
- Phòng tránh lỗi điển hình: Không giả định file luôn mở được hoặc mỗi fgets đọc cả dòng rất dài.

Kiến thức cần học

fopen · fclose · fgets · fprintf · FILE

Giải thích chi tiết

Mục tiêu của bài

fopen, fgets/fputs, fprintf, fclose, xử lý NULL và trạng thái lỗi.

Kiến thức cốt lõi

- Bản chất: fopen, fgets/fputs, fprintf, fclose, xử lý NULL và trạng thái lỗi.
- Cần nắm: fopen, fclose, fgets, fprintf, FILE. Giải thích được từng thuật ngữ bằng ví dụ nhỏ, thay vì chỉ học thuộc cú pháp.

- Cách tư duy: Xác định dữ liệu đầu vào, phép biến đổi, đầu ra và những điều kiện có thể làm chương trình sai.
- Chất lượng: Ưu tiên code rõ ràng, tên dễ hiểu, cảnh báo biên dịch sạch và kiểm tra các trường hợp biên.

Ví dụ C / lệnh minh họa

```
FILE *f = fopen("notes.txt", "w");
if (f == NULL) return 1;
fputs("Hello\n", f);
if (fclose(f) != 0) return 1;
```

Chú ý: Một số ví dụ là đoạn minh họa để đặt trong hàm hoặc tệp phù hợp, không phải mọi đoạn đều là chương trình hoàn chỉnh tự chạy.

Thực hành từng bước

1. Đọc ví dụ và dự đoán đầu ra hoặc trạng thái trước khi chạy.
2. Tạo một file bài tập riêng, tự gõ code và biên dịch với -std=c17 -Wall -Wextra -Wpedantic khi thích hợp.
3. Bài tập chính: Lưu danh sách sinh viên ra students.txt và nạp lại từ file.
4. Thử ít nhất ba tình huống: đầu vào điển hình, ca biên và đầu vào sai nếu chương trình có nhập liệu.
5. Ghi lại warning/lỗi gặp phải, giải thích nguyên nhân và cách sửa bằng lời của mình.

Làm việc với AI đúng cách

- Có thể nhờ AI: yêu cầu giải thích khái niệm, đặt câu hỏi gợi mở, đề xuất test case và đọc warning đã loại bỏ thông tin nhạy cảm.
- Phải tự kiểm chứng: tự chạy code, đối chiếu với tài liệu, đọc từng dòng thay đổi, xác định giới hạn bộ nhớ và đầu vào trước khi dùng.
- Prompt mẫu: "Tôi đang học bài Đọc ghi file văn bản với stdio. Hãy đóng vai gia sư C17, đưa ra 3 gợi ý tăng dần cho bài tập: Lưu danh sách sinh viên ra students.txt và nạp lại từ file. Không đưa lời giải hoàn chỉnh trước khi tôi thử; sau đó hãy review code của tôi về tính đúng, cảnh báo, ca biên và an toàn bộ nhớ."

Lỗi thường gặp

Không giả định file luôn mở được hoặc mỗi fgets đọc cả dòng rất dài.

Điều kiện hoàn thành

Tự giải thích được nội dung bài; hoàn tất bài tập chính; có code minh họa chạy được trong bối cảnh phù hợp; chứng minh đã thử trường hợp biên và sửa các warning quan trọng.

Ví dụ minh họa

Đọc ghi file văn bản với stdio

```
FILE *f = fopen("notes.txt", "w");
if (f == NULL) return 1;
fputs("Hello\n", f);
if (fclose(f) != 0) return 1;
```

fopen, fgets/fputs, fprintf, fclose, xử lý NULL và trạng thái lỗi.

Checklist

[ ] Tôi có thể giải thích ý nghĩa của bài bằng ngôn ngữ của mình.

[ ] Đã thực hành: Lưu danh sách sinh viên ra students.txt và nạp lại từ file.

[ ] Đã thử ca thông thường và ca biên; ghi rõ kết quả.

[ ] Đã đọc warning compiler và xử lý lỗi phù hợp.

[ ] Đã kiểm tra lỗi: Không giả định file luôn mở được hoặc mỗi fgets đọc cả dòng rất dài.

[ ] Đã tự xem và hiểu code, không chỉ sao chép lời giải AI.

Tài liệu tham khảo

C language reference — cppreference Tài liệu
https://www.cppreference.com/c/language

C input/output library Tài liệu
https://en.cppreference.com/w/c/io

Nên học trước

enum, typedef và hằng số có tên

---

4. Tách trường dữ liệu, xác thực và báo lỗi

Nhập dữ liệu dòng theo dòng, tách thành trường bằng sscanf/strtok có kiểm soát, phát hiện file sai định dạng.

Mục tiêu học tập

- Giải thích được: Nhập dữ liệu dòng theo dòng, tách thành trường bằng sscanf/strtok có kiểm soát, phát hiện file sai định dạng.
- Thực hiện độc lập: Đọc file CSV đơn giản không có trường được quote; báo dòng lỗi và bỏ qua/thoát có chủ đích.
- Dự đoán, chạy và giải thích kết quả của ví dụ; nhận diện lỗi biên dịch và lỗi logic.
- Phòng tránh lỗi điển hình: CSV thực tế có dấu phẩy trong dấu nháy; đừng tự coi parser đơn giản là chuẩn CSV đầy đủ.

Kiến thức cần học

parsing · validation · EOF · error-handling

Giải thích chi tiết

Mục tiêu của bài

Nhập dữ liệu dòng theo dòng, tách thành trường bằng sscanf/strtok có kiểm soát, phát hiện file sai định dạng.

Kiến thức cốt lõi

- Bản chất: Nhập dữ liệu dòng theo dòng, tách thành trường bằng sscanf/strtok có kiểm soát, phát hiện file sai định dạng.
- Cần nắm: parsing, validation, EOF, error-handling. Giải thích được từng thuật ngữ bằng ví dụ nhỏ, thay vì chỉ học thuộc cú pháp.
- Cách tư duy: Xác định dữ liệu đầu vào, phép biến đổi, đầu ra và những điều kiện có thể làm chương trình sai.
- Chất lượng: Ưu tiên code rõ ràng, tên dễ hiểu, cảnh báo biên dịch sạch và kiểm tra các trường hợp biên.

Ví dụ C / lệnh minh họa

```
char line[128];
while (fgets(line, sizeof line, f)) {
    /* kiem tra va phan tich tung dong */
}
```

Chú ý: Một số ví dụ là đoạn minh họa để đặt trong hàm hoặc tệp phù hợp, không phải mọi đoạn đều là chương trình hoàn chỉnh tự chạy.

Thực hành từng bước

1. Đọc ví dụ và dự đoán đầu ra hoặc trạng thái trước khi chạy.
2. Tạo một file bài tập riêng, tự gõ code và biên dịch với -std=c17 -Wall -Wextra -Wpedantic khi thích hợp.
3. Bài tập chính: Đọc file CSV đơn giản không có trường được quote; báo dòng lỗi và bỏ qua/thoát có chủ đích.
4. Thử ít nhất ba tình huống: đầu vào điển hình, ca biên và đầu vào sai nếu chương trình có nhập liệu.
5. Ghi lại warning/lỗi gặp phải, giải thích nguyên nhân và cách sửa bằng lời của mình.

Làm việc với AI đúng cách

- Có thể nhờ AI: yêu cầu giải thích khái niệm, đặt câu hỏi gợi mở, đề xuất test case và đọc warning đã loại bỏ thông tin nhạy cảm.
- Phải tự kiểm chứng: tự chạy code, đối chiếu với tài liệu, đọc từng dòng thay đổi, xác định giới hạn bộ nhớ và đầu vào trước khi dùng.
- Prompt mẫu: "Tôi đang học bài Tách trường dữ liệu, xác thực và báo lỗi. Hãy đóng vai gia sư C17, đưa ra 3 gợi ý tăng dần cho bài tập: Đọc file CSV đơn giản không có trường được quote; báo dòng lỗi và bỏ qua/thoát có chủ đích. Không đưa lời giải hoàn chỉnh trước khi tôi thử; sau đó hãy review code của tôi về tính đúng, cảnh báo, ca biên và an toàn bộ nhớ."

Lỗi thường gặp

CSV thực tế có dấu phẩy trong dấu nháy; đừng tự coi parser đơn giản là chuẩn CSV đầy đủ.

Điều kiện hoàn thành

Tự giải thích được nội dung bài; hoàn tất bài tập chính; có code minh họa chạy được trong bối cảnh phù hợp; chứng minh đã thử trường hợp biên và sửa các warning quan trọng.

Ví dụ minh họa

Tách trường dữ liệu, xác thực và báo lỗi

```
char line[128];
while (fgets(line, sizeof line, f)) {
    /* kiem tra va phan tich tung dong */
}
```

Nhập dữ liệu dòng theo dòng, tách thành trường bằng sscanf/strtok có kiểm soát, phát hiện file sai định dạng.

Checklist

- [ ] Tôi có thể giải thích ý nghĩa của bài bằng ngôn ngữ của mình.
- [ ] Đã thực hành: Đọc file CSV đơn giản không có trường được quote; báo dòng lỗi và bỏ qua/thoát có chủ đích.
- [ ] Đã thử ca thông thường và ca biên; ghi rõ kết quả.
- [ ] Đã đọc warning compiler và xử lý lỗi phù hợp.

- [ ] Đã kiểm tra lỗi: CSV thực tế có dấu phẩy trong dấu nháy; đừng tự coi parser đơn giản là chuẩn CSV đầy đủ.
- [ ] Đã tự xem và hiểu code, không chỉ sao chép lời giải AI.

Tài liệu tham khảo

C language reference — cppreference Tài liệu
https://www.cppreference.com/c/language

C input/output library Tài liệu
https://en.cppreference.com/w/c/io

Nên học trước

Đọc ghi file văn bản với stdio

---

5. Macro cơ bản, include guard và Makefile

Hiểu #include, #define, điều kiện tiền xử lý; make để tự động biên dịch tệp phụ thuộc.

Mục tiêu học tập

- Giải thích được: Hiểu #include, #define, điều kiện tiền xử lý; make để tự động biên dịch tệp phụ thuộc.
- Thực hiện độc lập: Viết Makefile target all, run, clean cho dự án nhiều file; kiểm thử build sạch.
- Dự đoán, chạy và giải thích kết quả của ví dụ; nhận diện lỗi biên dịch và lỗi logic.
- Phòng tránh lỗi điển hình: Macro không có type safety; không viết macro có hiệu ứng phụ lặp lại.

Kiến thức cần học

preprocessor · macro · make · dependencies

Giải thích chi tiết

Mục tiêu của bài

Hiểu #include, #define, điều kiện tiền xử lý; make để tự động biên dịch tệp phụ thuộc.

Kiến thức cốt lõi

- Bản chất: Hiểu #include, #define, điều kiện tiền xử lý; make để tự động biên dịch tệp phụ thuộc.
- Cần nắm: preprocessor, macro, make, dependencies. Giải thích được từng thuật ngữ bằng ví dụ nhỏ, thay vì chỉ học thuộc cú pháp.
- Cách tư duy: Xác định dữ liệu đầu vào, phép biến đổi, đầu ra và những điều kiện có thể làm chương trình sai.
- Chất lượng: Ưu tiên code rõ ràng, tên dễ hiểu, cảnh báo biên dịch sạch và kiểm tra các trường hợp biên.

Ví dụ C / lệnh minh họa

```
#define MAX_STUDENTS 100
#ifndef STUDENT_H
#define STUDENT_H
/* declarations */
#endif
```

Chú ý: Một số ví dụ là đoạn minh họa để đặt trong hàm hoặc tệp phù hợp, không phải mọi đoạn đều là chương trình hoàn chỉnh tự chạy.

Thực hành từng bước

1. Đọc ví dụ và dự đoán đầu ra hoặc trạng thái trước khi chạy.
2. Tạo một file bài tập riêng, tự gõ code và biên dịch với -std=c17 -Wall -Wextra -Wpedantic khi thích hợp.
3. Bài tập chính: Viết Makefile target all, run, clean cho dự án nhiều file; kiểm thử build sạch.
4. Thử ít nhất ba tình huống: đầu vào điển hình, ca biên và đầu vào sai nếu chương trình có nhập liệu.
5. Ghi lại warning/lỗi gặp phải, giải thích nguyên nhân và cách sửa bằng lời của mình.

Làm việc với AI đúng cách

- Có thể nhờ AI: yêu cầu giải thích khái niệm, đặt câu hỏi gợi mở, đề xuất test case và đọc warning đã loại bỏ thông tin nhạy cảm.
- Phải tự kiểm chứng: tự chạy code, đối chiếu với tài liệu, đọc từng dòng thay đổi, xác định giới hạn bộ nhớ và đầu vào trước khi dùng.
- Prompt mẫu: "Tôi đang học bài Macro cơ bản, include guard và Makefile. Hãy đóng vai gia sư C17, đưa ra 3 gợi ý tăng dần cho bài tập: Viết Makefile target all, run, clean cho dự án nhiều file; kiểm thử build sạch. Không đưa lời giải hoàn chỉnh trước khi tôi thử; sau đó hãy review code của tôi về tính đúng, cảnh báo, ca biên và an toàn bộ nhớ."

Lỗi thường gặp

Macro không có type safety; không viết macro có hiệu ứng phụ lặp lại.

Điều kiện hoàn thành

Tự giải thích được nội dung bài; hoàn tất bài tập chính; có code minh họa chạy được trong bối cảnh phù hợp; chứng minh đã thử trường hợp biên và sửa các warning quan trọng.

Ví dụ minh họa

Macro cơ bản, include guard và Makefile

```
#define MAX_STUDENTS 100
#ifndef STUDENT_H
#define STUDENT_H
/* declarations */
#endif
```

Hiểu #include, #define, điều kiện tiền xử lý; make để tự động biên dịch tệp phụ thuộc.

Checklist

[ ] Tôi có thể giải thích ý nghĩa của bài bằng ngôn ngữ của mình.
[ ] Đã thực hành: Viết Makefile target all, run, clean cho dự án nhiều file; kiểm thử build sạch.
[ ] Đã thử ca thông thường và ca biên; ghi rõ kết quả.
[ ] Đã đọc warning compiler và xử lý lỗi phù hợp.
[ ] Đã kiểm tra lỗi: Macro không có type safety; không viết macro có hiệu ứng phụ lặp lại.
[ ] Đã tự xem và hiểu code, không chỉ sao chép lời giải AI.

Tài liệu tham khảo

C language reference — cppreference Tài liệu
https://www.cppreference.com/c/language

C input/output library Tài liệu
https://en.cppreference.com/w/c/io

Nên học trước

Tách trường dữ liệu, xác thực và báo lỗi

GIAI ĐOẠN 8

Thực chiến, an toàn và định hướng

Giai đoạn 8: xây dựng năng lực qua 5 bài về thực chiến, an toàn và định hướng, kết hợp kiến thức, code và kiểm thử.

1. Viết test và kiểm tra ca biên

Thiết kế test danh sách rỗng, dữ liệu trùng, số âm, độ dài tối đa, chia 0; dùng assert cho kiểm thử nội bộ.

Mục tiêu học tập

- Giải thích được: Thiết kế test danh sách rỗng, dữ liệu trùng, số âm, độ dài tối đa, chia 0; dùng assert cho kiểm thử nội bộ.
- Thực hiện độc lập: Tạo test riêng cho tìm kiếm, sắp xếp và xử lý dữ liệu không hợp lệ.
- Dự đoán, chạy và giải thích kết quả của ví dụ; nhận diện lỗi biên dịch và lỗi logic.
- Phòng tránh lỗi điển hình: assert có thể bị tắt bằng NDEBUG; không dùng thay cho kiểm tra đầu vào người dùng.

Kiến thức cần học

unit-test · assert · boundary-cases

Giải thích chi tiết

Mục tiêu của bài

Thiết kế test danh sách rỗng, dữ liệu trùng, số âm, độ dài tối đa, chia 0; dùng assert cho kiểm thử nội bộ.

Kiến thức cốt lõi

- Bản chất: Thiết kế test danh sách rỗng, dữ liệu trùng, số âm, độ dài tối đa, chia 0; dùng assert cho kiểm thử nội bộ.
- Cần nắm: unit-test, assert, boundary-cases. Giải thích được từng thuật ngữ bằng ví dụ nhỏ, thay vì chỉ học thuộc cú pháp.
- Cách tư duy: Xác định dữ liệu đầu vào, phép biến đổi, đầu ra và những điều kiện có thể làm chương trình sai.
- Chất lượng: Ưu tiên code rõ ràng, tên dễ hiểu, cảnh báo biên dịch sạch và kiểm tra các trường hợp biên.

Ví dụ C / lệnh minh họa

```
assert(square(3) == 9);
assert(find((int[]){1, 2, 3}, 3, 2) == 1);
```

Chú ý: Một số ví dụ là đoạn minh họa để đặt trong hàm hoặc tệp phù hợp, không phải mọi đoạn đều là chương trình hoàn chỉnh tự chạy.

Thực hành từng bước

1. Đọc ví dụ và dự đoán đầu ra hoặc trạng thái trước khi chạy.
2. Tạo một file bài tập riêng, tự gõ code và biên dịch với -std=c17 -Wall -Wextra -Wpedantic khi thích hợp.
3. Bài tập chính: Tạo test riêng cho tìm kiếm, sắp xếp và xử lý dữ liệu không hợp lệ.
4. Thử ít nhất ba tình huống: đầu vào điển hình, ca biên và đầu vào sai nếu chương trình có nhập liệu.

5. Ghi lại warning/lỗi gặp phải, giải thích nguyên nhân và cách sửa bằng lời của mình.

Làm việc với AI đúng cách

- Có thể nhờ AI: yêu cầu giải thích khái niệm, đặt câu hỏi gợi mở, đề xuất test case và đọc warning đã loại bỏ thông tin nhạy cảm.
- Phải tự kiểm chứng: tự chạy code, đối chiếu với tài liệu, đọc từng dòng thay đổi, xác định giới hạn bộ nhớ và đầu vào trước khi dùng.
- Prompt mẫu: "Tôi đang học bài Viết test và kiểm tra ca biên. Hãy đóng vai gia sư C17, đưa ra 3 gợi ý tăng dần cho bài tập: Tạo test riêng cho tìm kiếm, sắp xếp và xử lý dữ liệu không hợp lệ. Không đưa lời giải hoàn chỉnh trước khi tôi thử; sau đó hãy review code của tôi về tính đúng, cảnh báo, ca biên và an toàn bộ nhớ."

Lỗi thường gặp

assert có thể bị tắt bằng NDEBUG; không dùng thay cho kiểm tra đầu vào người dùng.

Điều kiện hoàn thành

Tự giải thích được nội dung bài; hoàn tất bài tập chính; có code minh họa chạy được trong bối cảnh phù hợp; chứng minh đã thử trường hợp biên và sửa các warning quan trọng.

Ví dụ minh họa

Viết test và kiểm tra ca biên

```
assert(square(3) == 9);
assert(find((int[]){1, 2, 3}, 3, 2) == 1);
```

Thiết kế test danh sách rỗng, dữ liệu trùng, số âm, độ dài tối đa, chia 0; dùng assert cho kiểm thử nội bộ.

Checklist

[ ] Tôi có thể giải thích ý nghĩa của bài bằng ngôn ngữ của mình.
[ ] Đã thực hành: Tạo test riêng cho tìm kiếm, sắp xếp và xử lý dữ liệu không hợp lệ.
[ ] Đã thử ca thông thường và ca biên; ghi rõ kết quả.
[ ] Đã đọc warning compiler và xử lý lỗi phù hợp.
[ ] Đã kiểm tra lỗi: assert có thể bị tắt bằng NDEBUG; không dùng thay cho kiểm tra đầu vào người dùng.
[ ] Đã tự xem và hiểu code, không chỉ sao chép lời giải AI.

Tài liệu tham khảo

C language reference — cppreference Tài liệu
https://www.cppreference.com/c/language

GCC Warning Options Tài liệu
https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html

GCC Instrumentation Options Tài liệu
https://gcc.gnu.org/onlinedocs/gcc/Instrumentation-Options.html

Nên học trước

Macro cơ bản, include guard và Makefile

---

2. Tư duy thuật toán và Big O nền tảng

Phân biệt O(1), O(n), O(n^2), phân rã vấn đề, cân bằng dễ hiểu và hiệu năng.

Mục tiêu học tập

- Giải thích được: Phân biệt O(1), O(n), O(n^2), phân rã vấn đề, cân bằng dễ hiểu và hiệu năng.

- Thực hiện độc lập: So sánh tuyến tính với vòng lặp lồng và đếm thao tác cho n=10,100,1000.
- Dự đoán, chạy và giải thích kết quả của ví dụ; nhận diện lỗi biên dịch và lỗi logic.
- Phòng tránh lỗi điển hình: Đừng tối ưu vi mô trước khi chương trình đúng và test được.

Kiến thức cần học

big-O · algorithm · tradeoff

Giải thích chi tiết

Mục tiêu của bài

Phân biệt O(1), O(n), O(n^2), phân rã vấn đề, cân bằng dễ hiểu và hiệu năng.

Kiến thức cốt lõi

- Bản chất: Phân biệt O(1), O(n), O(n^2), phân rã vấn đề, cân bằng dễ hiểu và hiệu năng.
- Cần nắm: big-O, algorithm, tradeoff. Giải thích được từng thuật ngữ bằng ví dụ nhỏ, thay vì chỉ học thuộc cú pháp.
- Cách tư duy: Xác định dữ liệu đầu vào, phép biến đổi, đầu ra và những điều kiện có thể làm chương trình sai.
- Chất lượng: Ưu tiên code rõ ràng, tên dễ hiểu, cảnh báo biên dịch sạch và kiểm tra các trường hợp biên.

Ví dụ C / lệnh minh họa

```
/* linear: n lan */
for (size_t i = 0; i < n; ++i) { /* ... */ }
```

Chú ý: Một số ví dụ là đoạn minh họa để đặt trong hàm hoặc tệp phù hợp, không phải mọi đoạn đều là chương trình hoàn chỉnh tự chạy.

Thực hành từng bước

1. Đọc ví dụ và dự đoán đầu ra hoặc trạng thái trước khi chạy.
2. Tạo một file bài tập riêng, tự gõ code và biên dịch với -std=c17 -Wall -Wextra -Wpedantic khi thích hợp.
3. Bài tập chính: So sánh tuyến tính với vòng lặp lồng và đếm thao tác cho n=10,100,1000.
4. Thử ít nhất ba tình huống: đầu vào điển hình, ca biên và đầu vào sai nếu chương trình có nhập liệu.
5. Ghi lại warning/lỗi gặp phải, giải thích nguyên nhân và cách sửa bằng lời của mình.

Làm việc với AI đúng cách

- Có thể nhờ AI: yêu cầu giải thích khái niệm, đặt câu hỏi gợi mở, đề xuất test case và đọc warning đã loại bỏ thông tin nhạy cảm.
- Phải tự kiểm chứng: tự chạy code, đối chiếu với tài liệu, đọc từng dòng thay đổi, xác định giới hạn bộ nhớ và đầu vào trước khi dùng.
- Prompt mẫu: "Tôi đang học bài Tư duy thuật toán và Big O nền tảng. Hãy đóng vai gia sư C17, đưa ra 3 gợi ý tăng dần cho bài tập: So sánh tuyến tính với vòng lặp lồng và đếm thao tác cho n=10,100,1000. Không đưa lời giải hoàn chỉnh trước khi tôi thử; sau đó hãy review code của tôi về tính đúng, cảnh báo, ca biên và an toàn bộ nhớ."

Lỗi thường gặp

Đừng tối ưu vi mô trước khi chương trình đúng và test được.

Điều kiện hoàn thành

Tự giải thích được nội dung bài; hoàn tất bài tập chính; có code minh họa chạy được trong bối cảnh phù hợp; chứng minh đã thử trường hợp biên và sửa các warning quan trọng.

Ví dụ minh họa

Tư duy thuật toán và Big O nền tảng

```
/* linear: n lan */
for (size_t i = 0; i < n; ++i) { /* ... */ }
```

Phân biệt O(1), O(n), O(n^2), phân rã vấn đề, cân bằng dễ hiểu và hiệu năng.

Checklist

- [ ] Tôi có thể giải thích ý nghĩa của bài bằng ngôn ngữ của mình.
- [ ] Đã thực hành: So sánh tuyến tính với vòng lặp lồng và đếm thao tác cho n=10,100,1000.
- [ ] Đã thử ca thông thường và ca biên; ghi rõ kết quả.
- [ ] Đã đọc warning compiler và xử lý lỗi phù hợp.
- [ ] Đã kiểm tra lỗi: Đừng tối ưu vi mô trước khi chương trình đúng và test được.
- [ ] Đã tự xem và hiểu code, không chỉ sao chép lời giải AI.

Tài liệu tham khảo

C language reference — cppreference Tài liệu
https://www.cppreference.com/c/language

GCC Warning Options Tài liệu
https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html

GCC Instrumentation Options Tài liệu
https://gcc.gnu.org/onlinedocs/gcc/Instrumentation-Options.html

Nên học trước

Viết test và kiểm tra ca biên

---

3. Lỗi an toàn bộ nhớ và lập trình phòng thủ

Nhận diện buffer overflow, integer overflow, format-string bug, use-after-free; giảm quyền và kiểm tra đầu vào.

Mục tiêu học tập

- Giải thích được: Nhận diện buffer overflow, integer overflow, format-string bug, use-after-free; giảm quyền và kiểm tra đầu vào.
- Thực hiện độc lập: Audit mini project: liệt kê 10 điểm có thể lỗi, sửa và test lại bằng sanitizer.
- Dự đoán, chạy và giải thích kết quả của ví dụ; nhận diện lỗi biên dịch và lỗi logic.
- Phòng tránh lỗi điển hình: Không chạy ví dụ khai thác trên hệ thống không có quyền; AI không thay thế review.

Kiến thức cần học

secure-c · overflow · format-string · defensive-programming

Giải thích chi tiết

Mục tiêu của bài

Nhận diện buffer overflow, integer overflow, format-string bug, use-after-free; giảm quyền và kiểm tra đầu vào.

Kiến thức cốt lõi

- Bản chất: Nhận diện buffer overflow, integer overflow, format-string bug, use-after-free; giảm quyền và kiểm tra đầu vào.
- Cần nắm: secure-c, overflow, format-string, defensive-programming. Giải thích được từng thuật ngữ bằng ví dụ nhỏ, thay vì chỉ học thuộc cú pháp.
- Cách tư duy: Xác định dữ liệu đầu vào, phép biến đổi, đầu ra và những điều kiện có thể làm chương trình sai.
- Chất lượng: Ưu tiên code rõ ràng, tên dễ hiểu, cảnh báo biên dịch sạch và kiểm tra các trường hợp biên.

Ví dụ C / lệnh minh họa

```
/* format khong do nguoi dung kiem soat */
printf("%s", user_text);
```

Chú ý: Một số ví dụ là đoạn minh họa để đặt trong hàm hoặc tệp phù hợp, không phải mọi đoạn đều là chương trình hoàn chỉnh tự chạy.

Thực hành từng bước

1. Đọc ví dụ và dự đoán đầu ra hoặc trạng thái trước khi chạy.
2. Tạo một file bài tập riêng, tự gõ code và biên dịch với -std=c17 -Wall -Wextra -Wpedantic khi thích hợp.
3. Bài tập chính: Audit mini project: liệt kê 10 điểm có thể lỗi, sửa và test lại bằng sanitizer.
4. Thử ít nhất ba tình huống: đầu vào điển hình, ca biên và đầu vào sai nếu chương trình có nhập liệu.
5. Ghi lại warning/lỗi gặp phải, giải thích nguyên nhân và cách sửa bằng lời của mình.

Làm việc với AI đúng cách

- Có thể nhờ AI: yêu cầu giải thích khái niệm, đặt câu hỏi gợi mở, đề xuất test case và đọc warning đã loại bỏ thông tin nhạy cảm.
- Phải tự kiểm chứng: tự chạy code, đối chiếu với tài liệu, đọc từng dòng thay đổi, xác định giới hạn bộ nhớ và đầu vào trước khi dùng.
- Prompt mẫu: "Tôi đang học bài Lỗi an toàn bộ nhớ và lập trình phòng thủ. Hãy đóng vai gia sư C17, đưa ra 3 gợi ý tăng dần cho bài tập: Audit mini project: liệt kê 10 điểm có thể lỗi, sửa và test lại bằng sanitizer. Không đưa lời giải hoàn chỉnh trước khi tôi thử; sau đó hãy review code của tôi về tính đúng, cảnh báo, ca biên và an toàn bộ nhớ."

Lỗi thường gặp

Không chạy ví dụ khai thác trên hệ thống không có quyền; AI không thay thế review.

Điều kiện hoàn thành

Tự giải thích được nội dung bài; hoàn tất bài tập chính; có code minh họa chạy được trong bối cảnh phù hợp; chứng minh đã thử trường hợp biên và sửa các warning quan trọng.

Ví dụ minh họa

Lỗi an toàn bộ nhớ và lập trình phòng thủ


```
/* format khong do nguoi dung kiem soat */
printf("%s", user_text);
```


Nhận diện buffer overflow, integer overflow, format-string bug, use-after-free; giảm quyền và kiểm tra đầu vào.

Checklist

[ ] Tôi có thể giải thích ý nghĩa của bài bằng ngôn ngữ của mình.

[ ] Đã thực hành: Audit mini project: liệt kê 10 điểm có thể lỗi, sửa và test lại bằng sanitizer.

[ ] Đã thử ca thông thường và ca biên; ghi rõ kết quả.

[ ] Đã đọc warning compiler và xử lý lỗi phù hợp.

[ ] Đã kiểm tra lỗi: Không chạy ví dụ khai thác trên hệ thống không có quyền; AI không thay thế review.

[ ] Đã tự xem và hiểu code, không chỉ sao chép lời giải AI.

Tài liệu tham khảo

C language reference — cppreference Tài liệu
https://www.cppreference.com/c/language

GCC Warning Options Tài liệu
https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html

GCC Instrumentation Options Tài liệu
https://gcc.gnu.org/onlinedocs/gcc/Instrumentation-Options.html

Nên học trước

Tư duy thuật toán và Big O nền tảng

---

4. Dự án cuối khóa: quản lý sinh viên CLI

Kết hợp struct, mảng cấp phát động, hàm, file, menu, tìm kiếm, sắp xếp, kiểm thử.

Mục tiêu học tập

- Giải thích được: Kết hợp struct, mảng cấp phát động, hàm, file, menu, tìm kiếm, sắp xếp, kiểm thử.
- Thực hiện độc lập: Xây chương trình thêm/sửa/xóa/tìm kiếm/lưu/tải; dữ liệu mẫu 30 bản ghi, README và 15 ca test.
- Dự đoán, chạy và giải thích kết quả của ví dụ; nhận diện lỗi biên dịch và lỗi logic.
- Phòng tránh lỗi điển hình: Không dùng dữ liệu cá nhân thật; phải có kiểm tra lỗi file và cấp phát.

Kiến thức cần học

capstone · CLI · crud · file-storage

Giải thích chi tiết

Mục tiêu của bài

Kết hợp struct, mảng cấp phát động, hàm, file, menu, tìm kiếm, sắp xếp, kiểm thử.

Kiến thức cốt lõi

- Bản chất: Kết hợp struct, mảng cấp phát động, hàm, file, menu, tìm kiếm, sắp xếp, kiểm thử.
- Cần nắm: capstone, CLI, crud, file-storage. Giải thích được từng thuật ngữ bằng ví dụ nhỏ, thay vì chỉ học thuộc cú pháp.
- Cách tư duy: Xác định dữ liệu đầu vào, phép biến đổi, đầu ra và những điều kiện có thể làm chương trình sai.
- Chất lượng: Ưu tiên code rõ ràng, tên dễ hiểu, cảnh báo biên dịch sạch và kiểm tra các trường hợp biên.

Ví dụ C / lệnh minh họa

```
/* modules: student.h, student.c, storage.c, main.c */
int main(void) { /* menu loop */ return 0; }
```

Chú ý: Một số ví dụ là đoạn minh họa để đặt trong hàm hoặc tệp phù hợp, không phải mọi đoạn đều là chương trình hoàn chỉnh tự chạy.

Thực hành từng bước

1. Đọc ví dụ và dự đoán đầu ra hoặc trạng thái trước khi chạy.
2. Tạo một file bài tập riêng, tự gõ code và biên dịch với -std=c17 -Wall -Wextra -Wpedantic khi thích hợp.
3. Bài tập chính: Xây chương trình thêm/sửa/xóa/tìm kiếm/lưu/tải; dữ liệu mẫu 30 bản ghi, README và 15 ca test.
4. Thử ít nhất ba tình huống: đầu vào điển hình, ca biên và đầu vào sai nếu chương trình có nhập liệu.
5. Ghi lại warning/lỗi gặp phải, giải thích nguyên nhân và cách sửa bằng lời của mình.

Làm việc với AI đúng cách

- Có thể nhờ AI: yêu cầu giải thích khái niệm, đặt câu hỏi gợi mở, đề xuất test case và đọc warning đã loại bỏ thông tin nhạy cảm.
- Phải tự kiểm chứng: tự chạy code, đối chiếu với tài liệu, đọc từng dòng thay đổi, xác định giới hạn bộ nhớ và đầu vào trước khi dùng.
- Prompt mẫu: "Tôi đang học bài Dự án cuối khóa: quản lý sinh viên CLI. Hãy đóng vai gia sư C17, đưa ra 3 gợi ý tăng dần cho bài tập: Xây chương trình thêm/sửa/xóa/tìm kiếm/lưu/tải; dữ liệu mẫu 30 bản ghi, README và 15 ca test. Không đưa lời giải hoàn chỉnh trước khi tôi thử; sau đó hãy review code của tôi về tính đúng, cảnh báo, ca biên và an toàn bộ nhớ."

Lỗi thường gặp

Không dùng dữ liệu cá nhân thật; phải có kiểm tra lỗi file và cấp phát.

Điều kiện hoàn thành

Tự giải thích được nội dung bài; hoàn tất bài tập chính; có code minh họa chạy được trong bối cảnh phù hợp; chứng minh đã thử trường hợp biên và sửa các warning quan trọng.

Ví dụ minh họa

Dự án cuối khóa: quản lý sinh viên CLI

```
/* modules: student.h, student.c, storage.c, main.c */
int main(void) { /* menu loop */ return 0; }
```

Kết hợp struct, mảng cấp phát động, hàm, file, menu, tìm kiếm, sắp xếp, kiểm thử.

Checklist

- [ ] Tôi có thể giải thích ý nghĩa của bài bằng ngôn ngữ của mình.
- [ ] Đã thực hành: Xây chương trình thêm/sửa/xóa/tìm kiếm/lưu/tải; dữ liệu mẫu 30 bản ghi, README và 15 ca test.
- [ ] Đã thử ca thông thường và ca biên; ghi rõ kết quả.
- [ ] Đã đọc warning compiler và xử lý lỗi phù hợp.
- [ ] Đã kiểm tra lỗi: Không dùng dữ liệu cá nhân thật; phải có kiểm tra lỗi file và cấp phát.
- [ ] Đã tự xem và hiểu code, không chỉ sao chép lời giải AI.

Tài liệu tham khảo

C language reference — cppreference Tài liệu
https://www.cppreference.com/c/language

GCC Warning Options Tài liệu
https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html

GCC Instrumentation Options Tài liệu
https://gcc.gnu.org/onlinedocs/gcc/Instrumentation-Options.html

Nên học trước

Lỗi an toàn bộ nhớ và lập trình phòng thủ

---

5. Tổng kết, Git, AI và hướng học tiếp

Tự kiểm chứng năng lực, dùng AI như trợ giảng/reviewer; chọn nhánh DSA, Linux, embedded hoặc hệ thống.

Mục tiêu học tập

- Giải thích được: Tự kiểm chứng năng lực, dùng AI như trợ giảng/reviewer; chọn nhánh DSA, Linux, embedded hoặc hệ thống.
- Thực hiện độc lập: Công bố repo có README, hướng dẫn build, test, lưu bài học rút ra và tự chấm rubric.
- Dự đoán, chạy và giải thích kết quả của ví dụ; nhận diện lỗi biên dịch và lỗi logic.
- Phòng tránh lỗi điển hình: Không ghi rằng đã hiểu chỉ vì AI chạy hộ; phải giải thích code và sửa bug độc lập.

Kiến thức cần học

portfolio · git · learning-plan · ai-literacy

Giải thích chi tiết

Mục tiêu của bài

Tự kiểm chứng năng lực, dùng AI như trợ giảng/reviewer; chọn nhánh DSA, Linux, embedded hoặc hệ thống.

Kiến thức cốt lõi

- Bản chất: Tự kiểm chứng năng lực, dùng AI như trợ giảng/reviewer; chọn nhánh DSA, Linux, embedded hoặc hệ thống.
- Cần nắm: portfolio, git, learning-plan, ai-literacy. Giải thích được từng thuật ngữ bằng ví dụ nhỏ, thay vì chỉ học thuộc cú pháp.
- Cách tư duy: Xác định dữ liệu đầu vào, phép biến đổi, đầu ra và những điều kiện có thể làm chương trình sai.
- Chất lượng: Ưu tiên code rõ ràng, tên dễ hiểu, cảnh báo biên dịch sạch và kiểm tra các trường hợp biên.

Ví dụ C / lệnh minh họa

```
git init
git add .
git commit -m "feat: complete C fundamentals capstone"
```

Chú ý: Một số ví dụ là đoạn minh họa để đặt trong hàm hoặc tệp phù hợp, không phải mọi đoạn đều là chương trình hoàn chỉnh tự chạy.

Thực hành từng bước

1. Đọc ví dụ và dự đoán đầu ra hoặc trạng thái trước khi chạy.

2. Tạo một file bài tập riêng, tự gõ code và biên dịch với -std=c17 -Wall -Wextra -Wpedantic khi thích hợp.
3. Bài tập chính: Công bố repo có README, hướng dẫn build, test, lưu bài học rút ra và tự chấm rubric.
4. Thử ít nhất ba tình huống: đầu vào điển hình, ca biên và đầu vào sai nếu chương trình có nhập liệu.
5. Ghi lại warning/lỗi gặp phải, giải thích nguyên nhân và cách sửa bằng lời của mình.

Làm việc với AI đúng cách

- Có thể nhờ AI: yêu cầu giải thích khái niệm, đặt câu hỏi gợi mở, đề xuất test case và đọc warning đã loại bỏ thông tin nhạy cảm.
- Phải tự kiểm chứng: tự chạy code, đối chiếu với tài liệu, đọc từng dòng thay đổi, xác định giới hạn bộ nhớ và đầu vào trước khi dùng.
- Prompt mẫu: "Tôi đang học bài Tổng kết, Git, AI và hướng học tiếp. Hãy đóng vai gia sư C17, đưa ra 3 gợi ý tăng dần cho bài tập: Công bố repo có README, hướng dẫn build, test, lưu bài học rút ra và tự chấm rubric. Không đưa lời giải hoàn chỉnh trước khi tôi thử; sau đó hãy review code của tôi về tính đúng, cảnh báo, ca biên và an toàn bộ nhớ."

Lỗi thường gặp

Không ghi rằng đã hiểu chỉ vì AI chạy hộ; phải giải thích code và sửa bug độc lập.

Điều kiện hoàn thành

Tự giải thích được nội dung bài; hoàn tất bài tập chính; có code minh họa chạy được trong bối cảnh phù hợp; chứng minh đã thử trường hợp biên và sửa các warning quan trọng.

Hiểu đúng về C hiện đại

Trong khóa này, các bài được viết theo C17 để môi trường học nhất quán và phổ biến; C23 được giới thiệu như hướng cập nhật, không yêu cầu người học có compiler hỗ trợ đầy đủ. Khi ví dụ phụ thuộc hệ điều hành, trình biên dịch hoặc phần mở rộng, phải nhận biết điều đó trước khi dùng trong dự án khác.

Ví dụ minh họa

Tổng kết, Git, AI và hướng học tiếp

```
git init
git add .
git commit -m "feat: complete C fundamentals capstone"
```

Tự kiểm chứng năng lực, dùng AI như trợ giảng/reviewer; chọn nhánh DSA, Linux, embedded hoặc hệ thống.

Checklist

[ ] Tôi có thể giải thích ý nghĩa của bài bằng ngôn ngữ của mình.
[ ] Đã thực hành: Công bố repo có README, hướng dẫn build, test, lưu bài học rút ra và tự chấm rubric.
[ ] Đã thử ca thông thường và ca biên; ghi rõ kết quả.
[ ] Đã đọc warning compiler và xử lý lỗi phù hợp.
[ ] Đã kiểm tra lỗi: Không ghi rằng đã hiểu chỉ vì AI chạy hộ; phải giải thích code và sửa bug độc lập.
[ ] Đã tự xem và hiểu code, không chỉ sao chép lời giải AI.

Tài liệu tham khảo

C language reference — cppreference Tài liệu
https://www.cppreference.com/c/language

GCC Warning Options Tài liệu
https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html

GCC Instrumentation Options Tài liệu
https://gcc.gnu.org/onlinedocs/gcc/Instrumentation-Options.html

Nên học trước

Dự án cuối khóa: quản lý sinh viên CLI