**Bài tập ứng dụng setjmp**

### **Bài 1: Cải Tiến Hàm THROW**

Mục tiêu:

* Sửa đổi hàm THROW để nó chấp nhận một thông điệp lỗi dưới dạng chuỗi ký tự, bên cạnh mã lỗi.
* Thông điệp lỗi nên được lưu trữ ở một nơi mà có thể truy cập được sau khi longjmp được gọi.

Yêu cầu:

* Sửa đổi hàm THROW trong chương trình C mẫu đã cho.
* Hàm THROW phải có khả năng nhận một thông điệp lỗi dưới dạng chuỗi ký tự và một mã lỗi.
* Thông điệp lỗi và mã lỗi phải được lưu trữ sao cho có thể được truy xuất từ khối CATCH.
* Viết hàm hoặc các hàm thử nghiệm khác, mỗi hàm ném ra một loại lỗi khác nhau, sử dụng cải tiến hàm THROW.
* Trong hàm main, xử lý các lỗi này một cách riêng biệt, in ra thông điệp lỗi tương ứng cho mỗi loại lỗi.

Ví dụ:

Thay vì bạn sử dụng đoạn mã sau:

| double divide(int a, int b) {  if (b == 0) {  THROW(1); // Sử dụng mã lỗi 1 cho lỗi chia cho 0  }  THROW(2); //Những lỗi khác  return (double)a / b; } |
| --- |

Được thay thế bằng:

| double divide(int a, int b) {  if (b == 0) {  THROW(1, "Lỗi: Phép chia cho 0!"); // Sử dụng mã lỗi 1 cho lỗi chia cho 0  }  THROW(2, "Lỗi: Những lỗi khác!");  return (double)a / b; } |
| --- |

Trong ví dụ này, mã lỗi 1 có thể tượng trưng cho lỗi chia cho 0, và chuỗi "Lỗi: Phép chia cho 0!" là thông điệp lỗi tương ứng. Sau khi gọi THROW, khối CATCH(1) trong main sẽ xử lý lỗi này và in ra thông điệp lỗi.

Thay vì bạn sử dụng đoạn mã sau:

| CATCH(1) {  printf("Lỗi: Phép chia cho 0!\n"); }CATCH(2) {  printf("Lỗi: Những lỗi khác!\n");  } |
| --- |

Được thay thế bằng:

| CATCH(1) {  printf("%s\n", error\_code) }CATCH(2) {  printf("%s\n", error\_code) } |
| --- |

### **Bài 2: Xử Lý Nhiều Loại Lỗi Trong Hệ Thống Phức Tạp Sử Dụng Macro TRY-CATCH**

Mục Tiêu:

Viết một chương trình trong ngôn ngữ C sử dụng các macro TRY, CATCH, và THROW để mô phỏng việc xử lý nhiều loại lỗi trong một hệ thống phức tạp.

Yêu Cầu:

* Định nghĩa các macro TRY, CATCH, và THROW giúp xử lý lỗi trong chương trình.
* Tạo các hàm giả lập các hoạt động khác nhau, mỗi hàm có khả năng "ném" ra một loại lỗi cụ thể sử dụng macro THROW.
* Trong hàm main, gọi các hàm này trong một khối TRY và xử lý các lỗi tương ứng trong các khối CATCH phù hợp.
* Các loại lỗi có thể bao gồm nhưng không giới hạn ở: lỗi đọc file, lỗi xử lý mạng, lỗi tính toán dữ liệu.
* In ra thông báo lỗi phù hợp khi một lỗi được bắt và xử lý.

Mô Tả Chi Tiết Hơn:

* Bạn cần viết ba hàm mô phỏng ba hoạt động khác nhau: readFile, networkOperation, và calculateData.
* Mỗi hàm này sẽ sử dụng THROW để ném ra một loại lỗi cụ thể khi gặp sự cố.
* Trong main, sử dụng TRY để bao quanh việc gọi các hàm này và sử dụng các khối CATCH để xử lý từng loại lỗi riêng biệt.
* Mỗi khối CATCH sẽ in ra một thông báo lỗi đặc trưng cho lỗi tương ứng.
* Đảm bảo chương trình kết thúc một cách an toàn, in ra thông báo kết thúc chương trình sau khi tất cả các lỗi đã được xử lý.

Cho một enum lưu các mã lỗi như sau:

| enum ErrorCodes { NO\_ERROR, FILE\_ERROR, NETWORK\_ERROR, CALCULATION\_ERROR }; |
| --- |

Thông tin các hàm:

| void readFile() {  printf("Đọc file...\n");  THROW(FILE\_ERROR, "Lỗi đọc file: File không tồn tại."); }  void networkOperation() {  // Bổ sung chương trình }  void calculateData() {  // Bổ sung chương trình } |
| --- |

Chương trình trong hàm main:

| TRY {  readFile();  networkOperation();  calculateData();  } CATCH(FILE\_ERROR) {  printf("%s\n", error\_message);} // Bổ sung thêm nhiều CATCH |
| --- |