**Mục tiêu môn học**  
Sau khóa học, sinh viên có thể:  
• Sử dụng các kỹ thuật lập trình C/C++ nâng cao  
• Cài đặt các cấu trúc dữ liệu thông dụng  
• Vận dụng kỹ thuật đệ quy  
• Vận dụng thuật toán sắp xếp, tìm kiếm

CON TRỎ

- Cấp phát động

C:

Student\* p = (Student \*)malloc( sizeof(Student))

free( p );  
p = nullptr;

Student\* pa = malloc( sizeof(Student) \* 5 );

free( pa );  
pa = nullptr;

c++

Student\* p = new (std:: nothrow) Student;

delete p;  
p = nullptr;

Studen\* p = new (std::nothrow) Student[n];

delete[] p;  
p = nullptr;

- cấp phát động mảng 2 chiều dùng còn trỏ cấp 2

Int\*\* arr = new (std::nothrow) int\*[m];

For (int I = 0; I < m; i++) {

Arr[i] = new int[n];

}

//delete

For (int I = 0; I < n; i++){

Delete[] arr[i];

}

Delete[] arr;

Arr = nullptr;

FILE

|  |  |  |
| --- | --- | --- |
|  | **Văn bản** | **Nhị phân** |
| **Mở** | FILE\* fp = fopen(“input.txt”, “rt”);  FILE\* fp = fopen(“output.txt”, “wt”); | FILE\* fp = fopen(“input.txt”, “rb”);  FILE\* fp = fopen(“input.txt”, “wb”); |
| **Đóng** | fclose (fp);  fp = nullptr; | |
| **Đọc 1 kí tự** | char c = fgetc(fp); |  |
| **Đọc 1 chuỗi** | Fgets(str, MAX\_SIZE, fp); |  |
| **Đọc theo định dạng** | fscanf(fp, “%d %d”, &a, &b); |  |
| **Đọc 1 khối dữ liệu** |  | size\_t fread( void \* ptr, size\_t size, size\_t count, FILE \* fp );  size\_t n\_read = fread( &b, sizeof(b), 1, fp );  if (n\_read > 0) -> success |
| **Ghi 1 khối dữ liệu** |  | size\_t fwrite( void \* ptr, size\_t size, size\_t count, FILE \* fp );  size\_t n\_write = fwrite( &a, sizeof(a), 1, fp ); if (n\_read > 0) -> success |
| **Ghi 1 kí tự** | fputc(int c, FILE \*fp); | int fputc(const char\* s, FILE \*fp); |
| **Ghi 1 chuỗi** | fputs(const char\* s, FILE \*fp); | int fputs(const char\* s, FILE \*fp); |
| **Ghi theo định dạng** | fprintf(FILE \*fp, const char\* format, ...); |  |
| **Kiểm tra cuối file** | int feof(FILE \*fp); | |
| **Lỗi thao tác** | int ferror(FILE \*fp); | |
| **Định vị con trỏ file** | long int ftell( FILE \* fp ); | |
| **Thiết lập con trỏ file** | int fseek ( FILE \* fp, long int offset, int origin );  fseek(fp, 0, SEEK\_SET / SEEK\_CUR / SEEK\_END); | |
| **Thiết lập con trỏ về đầu file** | void rewind( FILE \* fp ); | |
| **Làm sạch vùng đệm** | int fflush ( FILE \* fp ); | |
| **Tái kết nối con trỏ file** | FILE \* freopen ( const char \* filename, const char \* mode, FILE \* fp ); | |

FILE\* fp = fopen("myfile.txt","rt");  
if (! fp) {  
printf("Error opening file");  
return -1;  
}

Con trỏ FILE\* trong C++ được sử dụng để đại diện cho một tập tin trong chương trình của bạn. Con trỏ FILE\* cung cấp cho bạn khả năng mở, đóng và thao tác với các loại tập tin khác nhau, bao gồm cả tập tin văn bản và tập tin nhị phân. Dưới đây là chi tiết về các hàm được sử dụng với con trỏ FILE\* trong C++:

1. Hàm mở file (fopen) Hàm fopen được sử dụng để mở một tập tin với các tùy chọn khác nhau. Hàm này trả về một con trỏ FILE\* cho tập tin đã mở, hoặc trả về NULL nếu không thể mở tập tin.

Cú pháp: FILE \*fopen(const char \*filename, const char \*mode);

Ví dụ: FILE \*fp; fp = fopen("file.txt", "r"); // mở tập tin file.txt với chế độ đọc (read)

1. Hàm đóng file (fclose) Hàm fclose được sử dụng để đóng một tập tin đã mở.

Cú pháp: int fclose(FILE \*stream);

Ví dụ: fclose(fp); // đóng tập tin đã mở

1. Hàm nhập một kí tự từ tập tin văn bản (fgetc) Hàm fgetc được sử dụng để đọc một kí tự từ tập tin văn bản đã mở.

Cú pháp: int fgetc(FILE \*stream);

Ví dụ: char c; c = fgetc(fp); // đọc một kí tự từ tập tin đã mở

1. Hàm nhập một chuỗi từ tập tin văn bản (fgets) Hàm fgets được sử dụng để đọc một chuỗi từ tập tin văn bản đã mở.

Cú pháp: char \*fgets(char \*str, int num, FILE \*stream);

Ví dụ: char str[50]; fgets(str, 50, fp); // đọc một chuỗi từ tập tin đã mở

1. Hàm nhập theo định dạng (fscanf) Hàm fscanf được sử dụng để đọc dữ liệu từ tập tin với định dạng đã cho.

Cú pháp: int fscanf(FILE \*stream, const char \*format, ...);

Ví dụ: int x, y; fscanf(fp, "%d %d", &x, &y); // đọc hai số nguyên từ tập tin đã mở

1. Hàm ghi một kí tự vào tập tin văn bản (fputc) Hàm fputc được sử dụng để ghi một kí tự vào tập tin văn bản đã mở.

Cú pháp: int fputc(int c, FILE \*stream);

Ví dụ: fputc('a', fp); // ghi kí tự 'a' vào tập tin đã mở

1. H



tiếp tục viết

ChatGPT

1. Hàm ghi một chuỗi vào tập tin văn bản (fputs) Hàm fputs được sử dụng để ghi một chuỗi vào tập tin văn bản đã mở.

Cú pháp: int fputs(const char \*str, FILE \*stream);

Ví dụ: fputs("Hello, world!", fp); // ghi chuỗi "Hello, world!" vào tập tin đã mở

1. Hàm ghi theo định dạng (fprintf) Hàm fprintf được sử dụng để ghi dữ liệu vào tập tin với định dạng đã cho.

Cú pháp: int fprintf(FILE \*stream, const char \*format, ...);

Ví dụ: int x = 10, y = 20; fprintf(fp, "x = %d, y = %d", x, y); // ghi hai số nguyên vào tập tin đã mở với định dạng cho trước

1. Hàm đọc một khối từ tập tin nhị phân (fread) Hàm fread được sử dụng để đọc một khối dữ liệu từ tập tin nhị phân đã mở.

Cú pháp: size\_t fread(void \*ptr, size\_t size, size\_t count, FILE \*stream);

Ví dụ: int arr[10]; fread(arr, sizeof(int), 10, fp); // đọc một khối 10 số nguyên từ tập tin đã mở

1. Hàm ghi một khối vào tập tin nhị phân (fwrite) Hàm fwrite được sử dụng để ghi một khối dữ liệu vào tập tin nhị phân đã mở.

Cú pháp: size\_t fwrite(const void \*ptr, size\_t size, size\_t count, FILE \*stream);

Ví dụ: int arr[10] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; fwrite(arr, sizeof(int), 10, fp); // ghi một khối 10 số nguyên vào tập tin đã mở

1. Hàm ghi một kí tự vào tập tin nhị phân (fputc) Hàm fputc cũng có thể được sử dụng để ghi một kí tự vào tập tin nhị phân đã mở.

Cú pháp: int fputc(int c, FILE \*stream);

Ví dụ: fputc('a', fp); // ghi kí tự 'a' vào tập tin đã mở

1. Hàm ghi một chuỗi vào tập tin nhị phân (fwrite) Hàm fwrite cũng có thể được sử dụng để ghi một chuỗi vào tập tin nhị phân đã mở.

Cú pháp: size\_t fwrite(const void \*ptr, size\_t size, size\_t count, FILE \*stream);

Ví dụ: const char \*str = "Hello, world!"; fwrite(str, sizeof(char), strlen(str), fp); // ghi chuỗi "Hello, world!" vào tập tin đã mở

1. Hàm kiểm tra cuối file (feof) Hàm feof được



tiếp tục viết

ChatGPT

sử dụng để kiểm tra xem con trỏ đang ở cuối tập tin hay không.

Cú pháp: int feof(FILE \*stream);

Giá trị trả về là 0 nếu con trỏ chưa đến cuối tập tin và trả về số khác 0 nếu con trỏ đã đến cuối tập tin.

Ví dụ: while (!feof(fp)) { // lặp cho đến khi đọc hết tập tin // thực hiện đọc dữ liệu từ tập tin }

1. Hàm kiểm tra lỗi thao tác (ferror) Hàm ferror được sử dụng để kiểm tra xem có lỗi nào xảy ra trong quá trình thực hiện các thao tác với tập tin hay không.

Cú pháp: int ferror(FILE \*stream);

Giá trị trả về là 0 nếu không có lỗi và trả về số khác 0 nếu có lỗi.

Ví dụ: if (ferror(fp)) { // kiểm tra xem có lỗi nào xảy ra không // xử lý lỗi }

1. Hàm làm sạch vùng đệm (fflush) Hàm fflush được sử dụng để đẩy dữ liệu từ vùng đệm xuống tập tin.

Cú pháp: int fflush(FILE \*stream);

Ví dụ: fprintf(fp, "Hello, world!"); fflush(fp); // đẩy dữ liệu từ vùng đệm xuống tập tin

1. Hàm định vị con trỏ tập tin (ftell) Hàm ftell được sử dụng



tiếp tục viết

ChatGPT

để xác định vị trí hiện tại của con trỏ tập tin.

Cú pháp: long ftell(FILE \*stream);

Giá trị trả về là vị trí hiện tại của con trỏ tập tin tính từ đầu tập tin.

Ví dụ: fseek(fp, 0, SEEK\_END); // di chuyển con trỏ tập tin đến cuối tập tin long fileSize = ftell(fp); // lấy kích thước tập tin tính bằng byte

1. Hàm thiết lập con trỏ tập tin (fseek) Hàm fseek được sử dụng để di chuyển con trỏ tập tin đến một vị trí cụ thể trong tập tin.

Cú pháp: int fseek(FILE \*stream, long offset, int origin);

Trong đó:

* stream là con trỏ tập tin cần thiết lập.
* offset là số byte cần di chuyển con trỏ tập tin (có thể âm).
* origin là vị trí bắt đầu tính offset, có thể là:
  + SEEK\_SET: bắt đầu tính offset từ đầu tập tin.
  + SEEK\_CUR: bắt đầu tính offset từ vị trí hiện tại của con trỏ tập tin.
  + SEEK\_END: bắt đầu tính offset từ cuối tập tin.

Giá trị trả về là 0 nếu thành công và trả về số khác 0 nếu có lỗi.

Ví dụ: fseek(fp, 10, SEEK\_SET); // di chuyển con trỏ tập tin đến vị trí thứ 10 tính từ đầu tập tin

1. Hàm tái kết nối con trỏ tập tin (freopen) Hàm freopen được sử dụng để tái kết nối một tập tin đã được mở bằng một con trỏ tập tin khác.

Cú pháp: FILE \*freopen(const char \*filename, const char \*mode, FILE \*stream);

Trong đó:

* filename là tên tập tin cần mở lại.
* mode là chế độ mở tập tin (tương tự như trong hàm fopen).
* stream là con trỏ tập tin cần kết nối lại.

Giá trị trả về là con trỏ tập tin mới hoặc NULL nếu có lỗi.

Ví dụ: FILE \*fp = fopen("file.txt", "r"); // mở tập tin file.txt để đọc //... fp = freopen("file.txt", "w", fp); // tái kết nối tập tin để ghi

Ngoài các hàm trên, còn có một số hàm khác liên quan đến xử lý tập tin trong C++, nhưng những hàm này đã được thay thế bởi các lớp trong thư viện iostream của C++. Tuy nhiên, kiến thức về các hàm này vẫn có ý nghĩa trong việc hiểu và sử dụng các lớp đó



tiếp tục viết

ChatGPT

1. fseek

Hàm fseek được sử dụng để định vị con trỏ tệp đến một vị trí xác định trong tệp. Nó có thể được sử dụng để di chuyển con trỏ tệp đến vị trí bất kỳ trong tệp, cho phép thực hiện các hoạt động đọc hoặc ghi tại vị trí mới.

Cú pháp:

arduinoCopy code

int fseek(FILE \*stream, long int offset, int whence);

Trong đó:

* stream: Con trỏ tới tệp cần định vị.
* offset: Khoảng cách từ vị trí được chỉ định bằng whence.
* whence: Vị trí được định vị, có thể là một trong ba giá trị sau:
  + SEEK\_SET: Đặt con trỏ tệp đến offset byte đầu tiên của tệp.
  + SEEK\_CUR: Di chuyển con trỏ tệp offset byte theo hướng hiện tại.
  + SEEK\_END: Đặt con trỏ tệp đến offset byte cuối cùng của tệp.

Hàm này trả về 0 nếu thực hiện thành công và -1 nếu có lỗi.

Ví dụ:

scssCopy code

#include <stdio.h> int main() { FILE \*file; file = fopen("example.txt", "r"); if (file != NULL) { fseek(file, 10, SEEK\_SET); // đặt con trỏ tệp đến byte thứ 10 char c = fgetc(file); // đọc kí tự tại vị trí đó printf("Character at position 10: %c\n", c); fclose(file); } return 0; }

1. freopen

Hàm freopen được sử dụng để tái kết nối một tệp đã mở với một đường dẫn mới hoặc để thay đổi chế độ mở của một tệp. Hàm này trả về con trỏ tệp mới, hoặc NULL nếu có lỗi.

Cú pháp:

arduinoCopy code

FILE \*freopen(const char \*filename, const char \*mode, FILE \*stream);

Trong đó:

* filename: Tên tệp mới.
* mode: Chế độ mở tệp mới.
* stream: Con trỏ tệp đã mở cần được kết nối lại.

Ví dụ:

arduinoCopy code

#include <stdio.h> int main() { FILE \*file; file = fopen("example.txt", "r"); printf("File pointer before freopen: %p\n", file); file = freopen("newfile.txt", "w", file); printf("File pointer after freopen: %p\n", file); fprintf(file, "This text will be written to newfile.txt"); fclose(file); return 0; }

1. fflush

Hàm fflush được sử dụng để làm sạch vùng đệm của một tệp đã mở. Nó đảm bảo rằng các dữ liệu đã được ghi vào vùng đệ



tiếp tục viết

ChatGPT

**Hàm định vị con trỏ file: ftell**

Hàm **ftell** được sử dụng để xác định vị trí hiện tại của con trỏ file trong tệp. Hàm này có cú pháp như sau:

c++Copy code

long ftell(FILE\* stream);

Trong đó, **stream** là con trỏ tới FILE cần xác định vị trí.

Hàm **ftell** trả về vị trí hiện tại của con trỏ file trong tệp được xác định bằng số byte từ đầu của tệp.

Ví dụ:

c++Copy code

#include <iostream> #include <cstdio> int main() { FILE \*file; file = fopen("example.txt", "r"); if (file == NULL) { perror("Error opening file"); return -1; } fseek(file, 0, SEEK\_END); long size = ftell(file); printf("The size of the file is: %ld bytes\n", size); fclose(file); return 0; }

Kết quả khi chạy chương trình:

pythonCopy code

The size of the file is: 25 bytes

**Hàm thiết lập con trỏ file: fseek**

Hàm **fseek** được sử dụng để đặt vị trí của con trỏ file trong tệp. Hàm này có cú pháp như sau:

c++Copy code

int fseek(FILE\* stream, long offset, int origin);

Trong đó, **stream** là con trỏ tới FILE cần đặt vị trí, **offset** là số byte cần dịch chuyển con trỏ file và **origin** chỉ ra điểm bắt đầu để tính **offset**. Các giá trị có thể sử dụng cho **origin** bao gồm:

* SEEK\_SET: đặt vị trí từ đầu tệp
* SEEK\_CUR: đặt vị trí từ vị trí hiện tại
* SEEK\_END: đặt vị trí từ cuối tệp

Hàm **fseek** trả về 0 nếu thành công và trả về giá trị khác 0 nếu thất bại.

Ví dụ:

c++Copy code

#include <iostream> #include <cstdio> int main() { FILE \*file; file = fopen("example.txt", "r"); if (file == NULL) { perror("Error opening file"); return -1; } fseek(file, 6, SEEK\_SET); char c = fgetc(file); printf("The character at position 6 is: %c\n", c); fseek(file, -3, SEEK\_END); c = fgetc(file); printf("The third to last character is: %c\n", c); fclose(file); return 0; }

Kết quả khi chạy chương trình:

sqlCopy code

The character at position 6 is: e The third to last character is: t

**Hàm tái kết nối con trỏ file: freopen**

Hàm **freopen** được sử dụng để thay đổi hoặc tái kết nối một FILE



tiếp tục viết

ChatGPT

**Hàm fflush**

**Lí thuyết**

Hàm **fflush** dùng để đẩy nội dung của vùng đệm của tệp tin đã được mở ra để lưu trữ tới ổ đĩa. Hàm này đảm bảo rằng mọi nội dung được lưu trữ trong bộ nhớ đệm sẽ được lưu xuống ổ đĩa.

Cú pháp của hàm **fflush** như sau:

c++Copy code

int fflush(FILE\* stream);

Trong đó:

* **stream** là con trỏ tới tệp tin muốn đẩy nội dung vùng đệm xuống ổ đĩa.

Hàm **fflush** sẽ trả về 0 nếu đẩy vùng đệm thành công, và trả về EOF nếu có lỗi xảy ra.

**Ví dụ**

c++Copy code

#include <stdio.h> int main() { FILE\* fp; fp = fopen("test.txt", "w"); fputs("Hello, world!", fp); // Đẩy nội dung vùng đệm xuống ổ đĩa fflush(fp); fclose(fp); return 0; }

Trong ví dụ này, chúng ta mở một tệp tin để ghi, và sử dụng hàm **fputs** để ghi chuỗi **"Hello, world!"** vào vùng đệm của tệp tin đó. Sau đó, chúng ta gọi hàm **fflush** để đẩy nội dung của vùng đệm xuống ổ đĩa. Cuối cùng, chúng ta đóng tệp tin bằng hàm **fclose**.

Lưu ý rằng nếu chúng ta không gọi hàm **fflush** và chỉ đóng tệp tin bằng hàm **fclose**, nội dung của vùng đệm sẽ không được đẩy xuống ổ đĩa. Khi đó, nội dung sẽ được lưu lại trong bộ nhớ đệm và sẽ được đẩy xuống ổ đĩa khi bộ nhớ đệm đầy hoặc khi chương trình kết thúc.

**Hàm ftell**

**Lí thuyết**

Hàm **ftell** được sử dụng để trả về vị trí hiện tại của con trỏ tệp tin được trỏ đến bởi con trỏ **FILE\***. Hàm này trả về giá trị của con trỏ tệp tin so với đầu tệp tin.

Cú pháp của hàm **ftell** như sau:

c++Copy code

long int ftell(FILE\* stream);

Trong đó:

* **stream** là con trỏ tới tệp tin muốn lấy vị trí hiện tại của con trỏ.

Hàm **ftell** trả về vị trí hiện tại của con trỏ tệp tin so v