Bài 12: Tìm kiếm nhị phân - Thao tác File - Tiêu chuẩn code

# Tìm kiếm nhị phân

Thuật toán tìm kiếm nhị phân là một phương pháp tìm kiếm hiệu quả được sử dụng để tìm kiếm một phần tử trong một mảng đã được sắp xếp. Thuật toán này giảm đáng kể số lượng phép so sánh cần thực hiện so với việc tìm kiếm tuyến tính.

Mô tả:

* Sắp xếp mảng: Đầu tiên, mảng cần phải được sắp xếp theo thứ tự tăng dần hoặc giảm dần.
* Khởi tạo biến: Đặt hai biến, left và right, lần lượt là vị trí bắt đầu và kết thúc của phần mảng cần xem xét. Ban đầu, left = 0 và right = n-1 (n là kích thước của mảng).
* Lặp cho đến khi tìm thấy hoặc left > right:
* Tính toán vị trí trung bình mid của mảng: mid = (left + right) / 2.
* So sánh phần tử tại vị trí mid với giá trị cần tìm kiếm:

Nếu phần tử tại mid bằng giá trị cần tìm kiếm, trả về mid.

Nếu giá trị cần tìm kiếm nhỏ hơn phần tử tại mid, cập nhật right = mid - 1.

Nếu giá trị cần tìm kiếm lớn hơn phần tử tại mid, cập nhật left = mid + 1.

* Kết thúc: Nếu left > right, có nghĩa là không tìm thấy giá trị trong mảng.

Ví dụ:

| #**include** <stdio.h> #**include** <stdlib.h>  **int** **binarySearch**(**int** \*arr, **int** l, **int** r, **int** x) {  **if** (r >= l)  {  **int** mid = l + (r - l)/2;    // Nếu giá trị giữa mảng bằng với x, trả về chỉ số của nó  **if** (arr[mid] == x) **return** mid;    // Nếu x nhỏ hơn giá trị giữa mảng, tìm kiếm trong phần trái của mảng  **if** (arr[mid] > x) **return** binarySearch(arr, l, mid-1, x);    // Nếu x lớn hơn giá trị giữa mảng, tìm kiếm trong phần phải của mảng  **return** binarySearch(arr, mid+1, r, x);  }       // Nếu không tìm thấy x  **return** -1; }  **void** **swap**(**int** \*a, **int** \*b) {  **int** temp = \*a;  \*a = \*b;  \*b = temp; }  **void** **bubbleSort**(**int** arr[], **int** n) {  **int** i, j;  **for** (i = 0; i < n-1; i++)  {  // Lặp qua tất cả các phần tử trong mảng  **for** (j = 0; j < n-i-1; j++)  {  // Nếu phần tử sau nhỏ hơn phần tử trước, hoán vị chúng  **if** (arr[j] > arr[j+1])  swap(&arr[j], &arr[j+1]);  }  } }  **int** **main**() {  **int** n, x, i;  printf("Nhap so phan tu cua mang: ");  scanf("%d", &n);  **int** \*arr = (**int**\*)malloc(n \* **sizeof**(**int**));  printf("Nhap cac phan tu cua mang: ");  **for** (i = 0; i < n; i++)  {  scanf("%d", &arr[i]);  }   bubbleSort(arr, n);  **for** (**int** i = 0; i < n; i++)  {  printf("i = %d\n", arr[i]);  }    printf("Nhap gia tri can tim: ");  scanf("%d", &x);  **int** result = binarySearch(arr, 0, n-1, x);  **if** (result == -1)  printf("Khong tim thay %d trong mang.\n", x);  **else**  printf("Tim thay %d tai vi tri %d trong mang.\n", x, result);  free(arr);  **return** 0; } |
| --- |

# Thao tác File

Ngôn ngữ lập trình C cung cấp một số thư viện và hàm tiêu biểu để thực hiện các thao tác với file.

Có 2 định dạng File là file .txt và file .bin:

* File .txt: Nội dung bên trong file gồm các ký tự tuân theo bảng mã ASCII và thường dùng để lưu trữ chuỗi ký tự
* File .bib: Nội dung bên trong file chỉ gồm “0” và “1”. Các file này chỉ được khởi tạo trong một chương trình và nội dung bên trong chỉ chương trình có thể đọc.

Dưới đây là một số thao tác cơ bản khi làm việc với file trong C:

## Mở File

Để mở một file, bạn có thể sử dụng hàm fopen(). Hàm này trả về một con trỏ FILE, và cần được kiểm tra để đảm bảo file đã mở thành công.

Cú pháp:

| FILE \*file = fopen(**const** **char** \*file\_name, **const** **char** \*access\_mode); |
| --- |

Với:

Input parameter:

file\_name**:** tên file muốn mở.

access\_mode: chế độ truy cập file.

Return value:

Nếu mở file thành công, trả về địa chỉ của file

Nếu không mở được file, trả về NULL

Các chế độ truy cập file:

| Chế độ | Mô tả |
| --- | --- |
| r | Mở file với chế độ chỉ đọc file. Nếu mở file thành công thì trả về địa chỉ của phần tử đầu tiên trong file, nếu không thì trả về NULL. |
| rb | Mở file với chế độ chỉ đọc file theo định dạng binary. Nếu mở file thành công thì trả về địa chỉ của phần tử đầu tiên trong file, nếu không thì trả về NULL. |
| w | Mở file với chế độ ghi vào file. Nếu file đã tồn tại, thì sẽ ghi đè vào nội dung bên trong file. Nếu file chưa tồn tại thì sẽ tạo một file mới. Nếu không mở được file thì trả về NULL. |
| wb | Mở file với chế độ ghi vào file theo định dạng binary. Nếu file đã tồn tại, thì sẽ ghi đè vào nội dung bên trong file. Nếu file chưa tồn tại thì sẽ tạo một file mới. Nếu không mở được file thì trả về NULL. |
| a | Mở file với chế độ nối. Nếu mở file thành công thì trả về địa chỉ của phần tử cuối cùng trong file. Nếu file chưa tồn tại thì sẽ tạo một file mới. Nếu không mở được file thì trả về NULL. |
| ab | Mở file với chế độ nối dưới định dạng binary. Nếu mở file thành công thì trả về địa chỉ của phần tử cuối cùng trong file. Nếu file chưa tồn tại thì sẽ tạo một file mới. Nếu không mở được file thì trả về NULL. |
| r+ | Mở file với chế độ đọc và ghi file. Nếu mở file thành công thì trả về địa chỉ của phần tử đầu tiên trong file, nếu không thì trả về NULL. |
| rb+ | Mở file với chế độ đọc và ghi file dưới định dạng binary. Nếu mở file thành công thì trả về địa chỉ của phần tử đầu tiên trong file, nếu không thì trả về NULL. |
| w+ | Mở file với chế độ ghi và đọc file. Nếu file đã tồn tại thì trả về địa chỉ của phần tử đầu tiên của file. Nếu file chưa tồn tại thì sẽ tạo một file mới. |
| wb+ | Mở file với chế độ ghi và đọc file dưới định dạng binary. Nếu file đã tồn tại thì trả về địa chỉ của phần tử đầu tiên của file. Nếu file chưa tồn tại thì sẽ tạo một file mới. |
| a+ | Mở file với chế độ nối và đọc file. Nếu file đã tồn tại thì trả về địa chỉ của phần tử cuối cùng của file. Nếu file chưa tồn tại thì sẽ tạo một file mới. |
| ab+ | Mở file với chế độ nối và đọc file dưới định dạng binary. Nếu file đã tồn tại thì trả về địa chỉ của phần tử cuối cùng của file. Nếu file chưa tồn tại thì sẽ tạo một file mới. |

## Đọc File

Để đọc từ File, ta sử dụng các hàm sau:

| Tên hàm | Mô tả |
| --- | --- |
| fscanf() | Sử dụng chuỗi được định dạng và danh sách đối số biến để lấy đầu vào từ một File |
| fgets() | Copy nội dung trong File vào mảng dùng để lưu trữ với tối đa số lượng phần tử của mảng hoặc tới khi gặp ký tự xuống dòng. |
| fgetc() | Lấy giá trị tại địa chỉ hiện tại của file, sau đó di chuyển tới địa chỉ tiếp theo. Kiểu trả về là char |
| fread() | Đọc một số lượng byte được chỉ định từ File .bin |

Ngoài ra, để kiểm tra xem vị trí hiện tại có phải là cuối của File chưa, ta có thể dùng hàm feof() để kiểm tra.

## Ghi File

Để ghi vào File, ta sử dụng các hàm sau:

| Tên hàm | Mô tả |
| --- | --- |
| fprintf() | Ghi chuỗi vào File, và có thể thêm danh sách các đối số |
| fputs() | Ghi chuỗi vào File |
| fputc() | Ghi một ký tự vào File |
| fwrite() | Ghi một số byte được chỉ định vào File .bin |

## Đóng File

Sau khi mở File và không dùng nữa, thì ta cần đóng File để giải phóng tài nguyên và đảm bảo dữ liệu đã được lưu trữ.

Cú pháp:

| fclose(file\_name); |
| --- |