

# LỜI CẢM ƠN

Sau quá trình học tập và rèn luyện bộ môn Công nghệ thông tin trường Đại học Giao thông Vận tải – Phân hiệu tại thành phố Hồ Chí Minh nhóm em đã được trang bị các kiến thức cơ bản, các kỹ năng thực tế để có thể hoàn thành đề tài bài tập lớn của nhóm.

Nhóm em gửi lời cảm ơn chân thành đến quý thầy, cô bộ môn Công nghệ thông tin trường Đại học Giao thông Vận tải – Phân hiệu tại thành phố Hồ Chí Minh đã quan tâm, hướng dẫn, truyền đạt những kiến thức và kinh nghiệm cho chúng em trong suốt thời gian học tập, và thực hiện bài tập lớn này một cách tận tình. Em xin chúc quý thầy, cô thật nhiều sức khỏe và luôn đạt được thành công trong cuộc sống. Đặc biệt em xin cảm ơn cô Trần Thị Dung người đã trực tiếp hướng dẫn và chỉ bảo chúng em trong quá trình thực hiện đề tài “Quản lý sinh viên”.

Sau một thời gian nỗ lực thực hiện thì đề tài cũng đã hoàn thành. Nhưng không thể tránh khỏi những sai sót do chúng em còn chưa có nhiều kinh nghiệm thực tế. Chúng em kính mong nhận được sự góp ý và nhận xét từ quý thầy, cô để chúng em có thể hoàn thiện và hoàn thành tốt hơn cho đề tài của mình.

Lời sau cùng chúng em một lần nữa kính chúc quý thầy, cô bộ môn Công nghệ thông tin Trường Đại học Giao thông Vận tải – Phân hiệu tại thành phố Hồ Chí Minh thật nhiều sức khỏe và thành công.

# Mục lục

|  |    |
|--|----|
| Mục lục.....                                     | 2  |
| Lời mở đầu.....                                  | 4  |
| <b>CHƯƠNG 1: CƠ SỞ LÝ THUYẾT</b>                 |    |
| 1.1 Các thuật toán sắp xếp.....                  | 5  |
| 1.1.1 Sắp xếp đổi chỗ.....                       | 5  |
| 1.1.2 Sắp xếp chọn.....                          | 6  |
| 1.1.3 Sắp xếp chèn.....                          | 6  |
| 1.1.4 Sắp xếp nổi bọt.....                       | 7  |
| 1.1.5 Sắp xếp hỗn hợp.....                       | 7  |
| 1.2 Các thuật toán tìm kiếm.....                 | 8  |
| 1.2.1 Tìm kiếm tuyến tính.....                   | 8  |
| 1.2.2 Tìm kiếm nhị phân.....                     | 8  |
| 1.2.3 Tìm kiếm nội suy.....                      | 9  |
| 1.3 Danh sách liên kết đơn.....                  | 10 |
| 1.3.1 Cài đặt danh sách liên kết đơn.....        | 10 |
| 1.3.2 Thêm Node vào danh sách liên kết đơn.....  | 11 |
| 1.3.3 Xoá Node khỏi danh sách liên kết đơn.....  | 14 |
| 1.3.4 Lấy giá trị ở vị trí bất kỳ.....           | 16 |
| 1.3.5 Tìm kiếm trong danh sách liên kết đơn..... | 17 |
| 1.3.6 Duyệt danh sách liên kết đơn.....          | 17 |
| 1.3.7 Một số hàm hỗ trợ khác.....                | 18 |
| 1.3.8 Hàm main của danh sách liên kết đơn.....   | 19 |
| 1.4 Làm việc với tệp.....                        | 20 |

## **Chương II: Chương trình ứng dụng**

|   |    |
|---|----|
| 2.1 Lí do chọn đề tài.....              | 23 |
| 2.2 Ý tưởng làm đề tài.....             | 23 |
| 2.3 Hướng dẫn sử dụng chương trình..... | 24 |

## **Chương III: Kết Luận**

|                             |    |
|-----------------------------|----|
| 3.1 Kết quả đạt được.....   | 29 |
| 3.2 Nhược điểm.....         | 29 |
| 3.3 Hướng phát triển.....   | 29 |
| 3.4 Tài liệu tham khảo..... | 29 |

## LỜI MỞ ĐẦU

Trong xã hội ngày càng phát triển hiện nay, khoa học công nghệ là thứ không thể thiếu với mỗi quốc gia, doanh nghiệp, trường học hay mỗi cá nhân đặc biệt là công nghệ thông tin. Với sự phát triển nhanh chóng không ngừng nghỉ như vậy của công nghệ thông tin đã giúp giải quyết các công việc học tập, nghiên cứu, quản lý thông tin... Một cách dễ dàng và tiện lợi. Thấy được tiềm năng đó các quốc gia, doanh nghiệp, trường học, cá nhân... Đã ứng dụng nó vào thực tiễn cuộc sống để giải quyết công việc, học tập giải trí chỉ với một chiếc điện thoại thông minh. Trong những năm gần đây, số lượng học sinh, sinh viên của cả nước tăng dần đều, chứng tỏ người dân dần cải thiện được những thế hệ càng về sau. Bên cạnh đó, các thiết bị thông minh đã và đang được sử dụng rộng rãi trên toàn thế giới từ trẻ em cho đến người già. Nhờ vào sự phát triển của những ứng dụng công nghệ, ngày càng nhiều ứng dụng được xuất hiện, trong đó có các ứng dụng hỗ trợ cho các nhà trường, giáo viên kiểm soát được số lượng học viên của mình. Đồng thời nắm bắt cụ thể quá trình học tập của từng học viên, quan tâm, hỗ trợ cùng đồng thời bồi dưỡng các học sinh cá biệt. Do đó phần mềm quản lý sinh viên là một phần không thể thiếu trong quá trình giảng dạy.

# CHƯƠNG I: LÝ THUYẾT

## 1.1 Các thuật toán sắp xếp

*Khái niệm thuật toán sắp xếp:*

- Trong khoa học máy tính và trong toán học, **thuật toán sắp xếp** là một thuật toán sắp xếp các phần tử của một danh sách (hoặc một mảng) theo thứ tự (tăng hoặc giảm). Người ta thường xét trường hợp các phần tử cần sắp xếp là các số.

### 1.1.1 Sắp xếp đổi chỗ

- Xuất phát từ đầu dãy, lần lượt so sánh phần tử đầu dãy với các phần tử còn lại, nếu thấy lớn hơn thì đổi chỗ cho nhau, mục đích là để sau khi quét một lượt, phần tử bé nhất sẽ về đầu dãy.
- Code:

```
1  void interchangeSort (int a[], int n) {
2      for (int i = 0; i < n - 1; i++)
3          for (j = i + 1; j < n; j++)
4              if (a[i] > a[j]) swap(a, i, j);
5  }
6  void swap (int a[], int i, int j) {
7      int tmp = a[i];
8      a[i] = a[j];
9      a[j] = tmp;
10 }
```

Hình 1.1

### 1.1.2 Sắp xếp chọn

- Tìm phần tử nhỏ nhất trước, rồi mới đổi chỗ nó cho phần tử đầu tiên, xong lặp lại với phần còn lại.
- Code:

```
1 void selectionSort (int a[], int n) {
2     int min;
3     int iMin;
4     for (i = 0; i < n - 1; i++) {
5         min = a[i];
6         iMin = i;
7         for (j = i + 1; j < n; j++)
8             if (a[j] < a[iMin]) { min = a[j]; iMin = j; }
9         swap(a, iMin, i);
10    }
11 }
```

Hình 1.2

### 1.1.3 Sắp xếp chèn

- Chúng ta coi phần bên trái của biến chạy i là phần đã được sắp xếp theo đúng thứ tự (sẽ chuyển dần các phần tử từ mới hỗn độn ở phần bên phải sang). Tìm vị trí thích hợp trong nửa trái đó để chèn phần tử a[i] vào cho đúng thứ tự, sau đó dịch chuyển dần các phần tử bên phải kể từ vị trí vừa tìm ra để nhường lấy một chỗ trống, rồi mới chèn x (mang giá trị của a[i]) vào.
- Code:

```
1 void insertionSort (int a[], int n) {
2     int pos, x;
3     for (int i = 1; i < n; i++) {
4         x = a[i]; pos = i - 1;
5         while (pos >= 0 && a[pos] > x) {
6             a[pos + 1] = a[pos];
7             pos--;
8         }
9         a[pos + 1] = x;
10    }
11 }
```

Hình 1.3

### 1.1.4 Sắp xếp nổi bọt

- Lần lượt đổi chỗ các cặp đôi phần tử cạnh nhau nếu chúng ngược thứ tự, mục đích cũng là sau một lượt, phần tử bé nhất sẽ về đầu dãy.
- Code:

```
1 void bubbleSort (int a[], int n) {  
2     for (i = 0; i < n - 1; i++)  
3         for (j = n - 1; j > i; j--)  
4             if (a[j] < a[j - 1]) swap(a, j, j - 1);  
5 }
```

Hình 1.4

### 1.1.5 Sắp xếp hỗn hợp

- Lấy hình ảnh xóc bình cocktail, biến j có thể chạy 2 chiều, chiều xuôi thì đưa phần tử lớn nhất về cuối dãy, sau đó đảo lại để đưa phần tử bé nhất về đầu dãy.
- Code:

```
1 void shakerSort (int a[], int n) {  
2     int left = 0, right = n - 1, k = left;  
3     while (left < right) {  
4         for (int i = left; i < right; i++)  
5             if (a[i] > a[i + 1]) { swap(a, i, i + 1); k = i; }  
6         right = k;  
7         for (int i = right; i > left; i--)  
8             if (a[i - 1] > a[i]) { swap(a, i - 1, i); k = i; }  
9         left = k;  
10    }  
11 }
```

Hình 1.5

## 1.2 Thuật toán tìm kiếm

### *Khái niệm thuật toán tìm kiếm*

- Trong ngành khoa học máy tính một **giải thuật tìm kiếm** là một thuật toán lấy đầu vào là một bài toán và trả về kết quả là một lời giải cho bài toán đó, thường là sau khi cân nhắc giữa một loạt các lời giải có thể. Hầu hết các thuật toán được nghiên cứu bởi các nhà khoa học máy tính để giải quyết các bài toán đều là các thuật toán tìm kiếm. Tập hợp tất cả các lời giải có thể đối với một bài toán được gọi là không gian tìm kiếm. Thuật toán thử sai (*brute-force search*) hay các thuật toán tìm kiếm "sơ đẳng" không có thông tin sử dụng phương pháp đơn giản nhất và trực quan nhất. Trong khi đó, các thuật toán tìm kiếm có thông tin sử dụng Heuristics để áp dụng các tri thức về cấu trúc của không gian tìm kiếm nhằm giảm thời gian cần thiết cho việc tìm kiếm.

### 1.2.1 Tìm kiếm tuyến tính

- Là kiểm tra tuần tự từng phần tử của mảng, đến khi nào giống thì thôi.

- Code:

```
1  int linearSearch (int a[], int n, int x) {  
2      for (int i = 0; i < n; i++) {  
3          if (a[i] == x) return i; //trả về vị trí tìm thấy  
4          else return -1; //trả về -1 nếu ko tìm thấy  
5      }  
6  }
```

Hình 1.6

### 1.2.2 Tìm kiếm nhị phân

- Điều kiện của thuật toán này là mảng đã được sắp xếp tăng dần. So sánh x với giá trị của phần tử nằm ở giữa mảng ( $mid = (left + right) / 2$ ). Nếu x nhỏ hơn  $a[mid]$  thì nó chỉ có thể nằm ở nửa bên trái, ngược lại x lớn hơn  $a[mid]$  thì x nằm ở nửa bên phải. Xác định x nằm ở nửa nào thì ta lặp lại thuật toán với nửa đó. Như vậy số lần kiểm tra giảm đi nhiều do ko phải mất công kiểm tra những phần tử thuộc nửa còn lại.



- Code:

```
1  int binarySearch (int a[], int n, int x) {
2      int left = 0, right = n - 1, mid;
3      do {
4          mid = (left + right) / 2;
5          if (a[mid] == x) return mid;
6          else if (a[mid] <= x) left = mid + 1;
7          else right = mid - 1;
8      } while (left <= right);
9      return -1;
10 }
```

Hình 1.7

### 1.2.3 Tìm kiếm nội suy

- Là cải tiến của thuật toán tìm kiếm nhị phân. Thay vì chia đôi, thuật toán này chia theo phép tính  $mid = left + \frac{right - left}{a[right] - a[left]}(x - a[left])$  giúp thu gọn khoảng tìm kiếm hơn. Chỉ cần thay biểu thức tính mid này vào code mẫu của thuật toán tìm kiếm nhị phân là được.

- Code:

```
1  int interpolationSearch (int a[], int n, int x) {
2      int left = 0, right = n - 1, mid;
3      do {
4          mid = left + (right - left) / (a[right] - a[left]) * (x - a[left]);
5          if (a[mid] == x) return mid;
6          else if (a[mid] <= x) left = mid + 1;
7          else right = mid - 1;
8      } while (left <= right);
9      return -1;
10 }
```

Hình 1.8

## 1.3 Danh sách liên kết đơn

### *Khái niệm danh sách liên kết đơn*

- Danh sách liên kết đơn là một tập hợp các **Node** được phân bố động, được sắp xếp theo cách sao cho mỗi **Node** chứa “một giá trị” (**Data**) và “một con trỏ” (**Next**). Con trỏ sẽ trỏ đến phần tử kế tiếp của danh sách liên kết đó. Nếu con trỏ mà trỏ tới **NULL**, nghĩa là đó là phần tử cuối cùng của **linked list**.

### 1.3.1 Cài đặt danh sách liên kết đơn

#### \* Khai báo Linked List

- Đầu tiên data của chúng ta sẽ là số nguyên(int).
- Code:

```
struct LinkedList{  
    int data;  
    struct LinkedList *next;  
};
```

Hình 1.9

- Khai báo trên sẽ được sử dụng cho mọi **Node** trong **linked list**. Trường **data** sẽ lưu giữ giá trị và **next** sẽ là con trỏ để trỏ đến phần tử kế tiếp của nó. Và **next** là kiểu **LinkedList** của chính nó vì nó là con trỏ của chính bản thân nó, và nó trỏ tới 1 phần tử **Node** kế tiếp cũng có kiểu là **LinkedList**.

#### \* Tạo mới 1 Node

- Tạo 1 kiểu dữ liệu của **Struct LinkedList** để code ngắn hơn.

- Code:

```
typedef struct LinkedList *node;

node CreatNode(int value){
    node temp;
    temp = (node)malloc(sizeof(struct LinkedList));
    temp->next = NULL;
    temp->data = value;
    return temp;
}
```

Hình 1.10

- Mỗi 1 **Node** được khởi tạo, chúng ta cần cấp phát bộ nhớ cho nó, và mặc định cho con trỏ next trỏ tới **NULL**. Giá trị của Node sẽ được cung cấp khi thêm **Node** vào **linked list**.
- **typedef** được dùng để định nghĩa một kiểu dữ liệu trong C.
- **malloc** là hàm cấp phát bộ nhớ của C.
- **sizeof** là hàm trả về kích thước của kiểu dữ liệu, dùng làm tham số cho hàm **malloc**.

### 1.3.2 Thêm Node vào danh sách liên kết đơn

#### \* Thêm vào đầu

- Việc thêm vào đầu chính là việc cập nhật lại **head**. Ta gọi Node mới (temp), ta có:
  - Nếu **head** đang trỏ tới NULL, nghĩa là linked list đang trống, Node mới thêm vào sẽ làm head luôn
  - Ngược lại, ta phải thay thế phần tử **head** cũ bằng **head** mới. Việc này phải làm theo thứ tự như sau:
    - Cho **next** của temp trỏ tới **head** hiện hành
    - Đặt temp làm **head** mới

- Code:

```
node AddHead(node head, int value){  
    node temp = CreateNode(value);  
    if(head == NULL){  
        head = temp;  
    } else {  
        temp->next = head;  
        head = temp;  
    }  
    return head;  
}
```

Hình 1.11

### \* Thêm vào cuối

- Chúng ta sẽ cần Node đầu tiên, và giá trị muốn thêm. Khi đó, ta sẽ:

1. Tạo một **Node** mới với giá trị **value**
2. Nếu **head** = **NULL**, tức là danh sách liên kết đang trống. Khi đó **Node** mới (temp) sẽ là **head** luôn.
3. Ngược lại, ta sẽ duyệt tới **Node** cuối cùng (**Node** có **next** = **NULL**), và trở **next** của phần tử cuối tới **Node** mới (temp).

- Code:

```
node AddTail(node head, int value){
    node temp, p;
    temp = CreateNode(value);
    if(head == NULL){
        head = temp;
    }
    else{
        p = head;
        while(p->next != NULL){
            p = p->next;
        }
        p->next = temp;
    }
    return head;
}
```

Hình 1.12

### \* Thêm vào vị trí bất kỳ

- Để làm được việc này, ta phải duyệt từ đầu để tìm tới vị trí của **Node** cần chèn, giả sử là **Node Q**, khi đó ta cần làm theo thứ tự sau:

- Cho **next** của Node mới trở tới **Node** mà **Q** đang trở tới
- Cho **Node Q** trở tới **Node** mới

**Lưu ý:** Chỉ số chèn bắt đầu từ chỉ số 0 nhé.

- Code:

```
node AddAt(node head, int value, int position){
    if(position == 0 || head == NULL){
        head = AddHead(head, value);
    }else{
        int k = 1;
        node p = head;
        while(p != NULL && k != position){
            p = p->next;
            ++k;
        }

        if(k != position){
            head = AddTail(head, value);
        }else{
            node temp = CreateNode(value);
            temp->next = p->next;
            p->next = temp;
        }
    }
    return head;
}
```

Hình 1.13

- **Lưu ý:** Bạn phải làm theo thứ tự trên, nếu bạn cho  $p \rightarrow \text{next} = \text{temp}$  trước. Khi đó, bạn sẽ không thể lấy lại phần sau của danh sách liên kết nữa (Vì **next** chỉ được được lưu trong  $p \rightarrow \text{next}$  mà thay đổi  $p \rightarrow \text{next}$  rồi thì còn đâu giá trị cũ).

### 1.3.3 Xóa Node khỏi danh sách liên kết đơn

#### \* Xóa đầu

- Xóa đầu chỉ cần cho phần tử kế tiếp của **head** làm **head** là được thôi. Mà phần tử kế tiếp của **head** chính là **head->next**.

- Code:

```
node DelHead(node head){
    if(head == NULL){
        printf("\nChà có gì để xóa hết!");
    }else{
        head = head->next;
    }
    return head;
}
```

Hình 1.14

### \* Xóa cuối

- Xóa cuối phải duyệt từ đầu đến phần tử cuối -1, cho **next** của cuối -1 đó bằng **NULL**.

- Code:

```
node DelTail(node head){
    if (head == NULL || head->next == NULL){
        return DelHead(head);
    }
    node p = head;
    while(p->next->next != NULL){
        p = p->next;
    }
    p->next = p->next->next;
    return head;
}
```

Hình 1.15

- Hàm **Node** cuối - 1 là hàm có  $p \rightarrow \text{next} \rightarrow \text{next} = \text{NULL}$ . Bạn cho **next** của nó bằng **NULL** là xong.

### \* Xóa ở vị trí bất kỳ

- Việc xóa ở vị trí bất kỳ cũng khá giống xóa ở cuối kia. Đơn giản là chúng ta bỏ qua một phần tử.

**Lưu ý:** Chỉ số xóa bắt đầu từ 0 nhé. Việc tìm vị trí cần xóa chỉ duyệt tới **Node** gần cuối thôi (cuối - 1).

- Code:

```
node DelAt(node head, int position){
    if(position == 0 || head == NULL || head->next == NULL){
        head = DelHead(head);
    }else{
        int k = 1;
        node p = head;
        while(p->next->next != NULL && k != position){
            p = p->next;
            ++k;
        }

        if(k != position){
            head = DelTail(head);
        }else{
            p->next = p->next->next;
        }
    }
    return head;
}
```

Hình 1.16

### 1.3.4 Lấy giá trị ở vị trí bất kỳ

- Viết một hàm để truy xuất giá trị ở chỉ số bất kỳ. Trong trường hợp chỉ số vượt quá chiều dài của **linked list** - 1, hàm này trả về vị trí cuối cùng. Do hạn chế là ta không thể **raise error** khi chỉ số không hợp lệ. - Ta mặc định chỉ số truyền vào phải là số nguyên không âm. Nếu bạn muốn kiểm tra chỉ số hợp lệ thì nên kiểm tra trước khi gọi hàm này.



- Code:

```
int Get(node head, int index){
    int k = 0;
    node p = head;
    while(p->next != NULL && k != index){
        ++k;
        p = p->next;
    }
    return p->data;
}
```

Hình 1.17

- Lý do dùng `p->next != NULL` là vì chúng ta chỉ muốn đi qua các phần tử có **value**.

### 1.3.5 Tìm kiếm trong danh sách liên kết đơn

- Hàm tìm kiếm này sẽ trả về chỉ số của **Node** đầu tiên có giá trị bằng với giá trị cần tìm. Nếu không tìm thấy, chúng ta trả về -1.

- Code:

```
int Search(node head, int value){
    int position = 0;
    for(node p = head; p != NULL; p = p->next){
        if(p->data == value){
            return position;
        }
        ++position;
    }
    return -1;
}
```

Hình 1.18

### 1.3.6 Duyệt danh sách liên kết đơn

- Việc duyệt danh sách liên kết cực đơn giản. Khởi tạo từ **Node head**, bạn cứ thế đi theo con trỏ **next** cho tới trước khi **Node** đó **NULL**.

- Code:

```
void Traverser(node head){
    printf("\n");
    for(node p = head; p != NULL; p = p->next){
        printf("%5d", p->data);
    }
}

void Traverser(node head){
    printf("\n");
    for(node p = head; p != NULL; p = p->next){
        printf("%5d", p->data);
    }
}
```

Hình 1.19

### 1.3.7 Một số hàm hỗ trợ khác

#### \* Hàm khởi tạo Node head

- Đơn giản là cho con trỏ **head** = **NULL** thôi. Nếu bạn để ý, chúng ta vẫn check **head** = **NULL** để biết rằng danh sách liên kết chưa có phần tử nào ở các hàm phía trên

- Code:

```
node InitHead(){
    node head;
    head = NULL;
    return head;
}
```

Hình 1.20

#### \* Hàm nhập danh sách liên kết đơn

- Code:

```
node Input(){
    node head = InitHead();
    int n, value;
    do{
        printf("\nNhap so luong phan tu n = ");
        scanf("%d", &n);
    } while(n <= 0);

    for(int i = 0; i < n; ++i){
        printf("\nNhap gia tri can them: ");
        scanf("%d", &value);
        head = AddTail(head, value);
    }
    return head;
}
```

Hình 1.21

### 1.3.8 Hàm main của danh sách liên kết đơn

- Code:

```
int main(){
    printf("\n==Tao 1 danh sach lien ket==");
    node head = Input();
    Traverser(head);

    printf("\n==Thu them 1 phan tu vao linked list==");
    head = AddAt(head, 100, 2);
    Traverser(head);

    printf("\n==Thu xoa 1 phan tu khoi linked list==");
    head = DelAt(head, 1);
    Traverser(head);

    printf("\n==Thu tim kiem 1 phan tu trong linked list==");
    int index = Search(head, 5);
    printf("\nTim thay tai chi so %d", index);
}
```

Hình 1.22

- Kết quả chạy:

```
==Tao 1 danh sach lien ket==  
Nhap so luong phan tu n = 5  
Nhap gia tri can them: 1  
Nhap gia tri can them: 2  
Nhap gia tri can them: 3  
Nhap gia tri can them: 4  
Nhap gia tri can them: 5  
1 2 3 4 5  
==Thu them 1 phan tu vao linked list==  
1 2 100 3 4 5  
==Thu xoa 1 phan tu khoi linked list==  
1 100 3 4 5  
==Thu timkiem 1 phan tu trong linked list==  
Tim thay tai chi so 4
```

Hình 1.23

## 1.4 Một số khái niệm làm việc với tệp

- Dữ liệu trong chương trình được lưu trữ ở RAM máy tính, vì vậy khi kết thúc chương trình, tất cả dữ liệu sẽ bị giải phóng (mất dữ liệu). Để tránh vấn đề đó dữ liệu cần phải lưu trữ trên bộ nhớ ngoài (đĩa cứng, USB, ...) dưới dạng file. File có các đặc trưng sau:
  - Là một đơn vị lưu trữ logic.
  - Hiện thị bằng một tên.
  - Bao gồm một tập hợp dữ liệu do người tạo xác định.
- Hầu hết các chương trình đều cần phải lưu trữ sau khi xử lý. Vì vậy C cung cấp cho chúng ta kỹ thuật xử lý lưu trữ trên file.

### Các bước khi làm việc với File trong ngôn ngữ C

#### 1. Khai báo biến file **FILE \*<tên biến file> ;**

- Kiểu dữ liệu của file là kiểu FILE \*
- **<tên biến file>** : cách đặt tên biến file theo yêu cầu đặt tên của ngôn ngữ C
- Code :

```
FILE *f1; //khai báo một biến file
```

```
FILE *f2, *f3; //khai báo nhiều biến file
```

Hình 1.24

## 2. Gán biến file (đã khai báo) với một file vật lý

**<tên biến file>=fopen(<duong dan file>, <cách mở file>);**

- **fopen** là hàm mở file
- **<duong dan file>** : là chuỗi chứa đường dẫn tới file bạn muốn mở.
- **<cách mở file>** : là một chuỗi quy định mục đích cho việc sử dụng file, có thể sử dụng một số trong các giá trị sau :
  - ◆ **“w”** : mở file ghi. Nếu file muốn mở chưa có thì chương trình sẽ tự động tạo ra file đó.
  - ◆ **“r”** : mở file để đọc, nếu file chưa có thì chương trình sẽ báo lỗi.
  - ◆ **“a”** : mở file để ghi thêm, cách hoạt động giống “w”, tuy nhiên nếu file đã có dữ liệu thì chương trình sẽ không xóa dữ liệu có trước mà sẽ ghi tiếp vào phía sau.
  - ◆ **“r+”, “w+”** : mở file, cho phép đọc và ghi file.

- Code

```
f1 = fopen(dulieu.txt, "r");  
f2 = fopen(dlieu.txt, "w");
```

Hình 1.25

## 3. Đọc và ghi file

- Thao tác cơ bản khi làm việc với file trong ngôn ngữ C là đọc và ghi file. Cú pháp đọc và ghi file trong C tương tự cú pháp đọc và ghi dữ liệu từ màn hình.
  - **Hàm xử lý trên file sẽ có thêm ký tự **f** so với hàm xử lý trên bàn phím và màn hình:**
    - ◆ `printf(“ định dạng ”,<biến>); => fprintf(<biến file>,” định dạng ”,<biến>);`
    - ◆ `scanf(“ định dạng ”,&<biến>); => fscanf(<biến file>,” định dạng ”,&<biến>);`
    - ◆ `gets(<tenchuoi>); => fgets(<tenchuoi>, <kích thước>,<biến file>);`
  - Các tham số trong câu lệnh xử lý trên file sẽ nhiều tham số hơn câu lệnh xử lý trên bàn phím và màn hình <biến file> để xác định mình đang làm việc với file nào.

- Code :

```
int a;  
fscanf(f1, "%d", &a);  
fprintf(f2, "so ban vua nhap la %d ", a);
```

Hình 1.26

**Lưu ý:** Để kiểm tra ta đã đọc dữ liệu đến hết file hay chưa ta dùng một hằng số là EOF (End Of File)

#### 4. Đóng file

**fclose(<tên biến file>);**

- Sau khi thao tác với file xong, một điều quan trọng là cần đóng file đã mở kết thúc quá trình làm việc với file.
- Code

```
fclose(f1);
```

Hình 1.27

# CHƯƠNG II: CHƯƠNG TRÌNH ỨNG DỤNG

## 2.1 Lí do chọn đề tài

Trong xã hội ngày càng phát triển hiện nay, ứng dụng công nghệ vào đời sống không còn gì là quá xa lạ đối với mỗi chúng ta. Với sự phát triển nhanh một cách không ngừng của công nghệ thông tin đã giúp giải quyết các công việc học tập, nguyên cứu, quản lý thông tin,... Một cách dễ dàng và tiện lợi. Thấy được tiềm năng đó các quốc gia, doanh nghiệp, trường học, các cá nhân, ... đã ứng dụng nó vào thực tiễn cuộc sống để giải quyết công việc, học tập, giải trí với những chiếc điện thoại thông minh nhỏ gọn. Trong những năm gần đây nhu cầu về các phần mềm, ứng dụng học tập, giải trí càng nhiều hơn do nhu cầu sử dụng điện thoại các thiết bị gọn nhẹ ngày càng cao và thuận tiện cho người sử dụng. Nhằm giúp đỡ người sử dụng cụ thể là các thầy cô tiện lợi trong việc rà soát, nắm thông tin của từng sinh viên mọi lúc mọi nơi mà không phụ thuộc quá nhiều vào sổ sách công kênh. Ứng dụng giúp mọi người sử dụng một cách dễ dàng, giao diện thân thiện với người sử dụng.

## 2.2 Ý tưởng làm đề tài

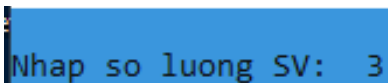
Muốn làm phần mềm quản lý sinh viên trước tiên chúng ta phải sử dụng cấu trúc struct để lưu trữ một đối tượng có nhiều thuộc tính.

Sau đó chúng ta sử dụng cấu trúc lặp: for, do while, while, để thực hiện các câu lệnh tiếp theo. Ngoài ra chúng ta phải sử dụng cấu trúc điều khiển và rẽ nhánh: if else, switch case.

Cuối cùng chúng ta nhập xuất file để lưu trữ dữ liệu, dễ dàng sao chép, di chuyển dữ liệu giữa các thiết bị với nhau.

## 2.3 Hướng dẫn sử dụng

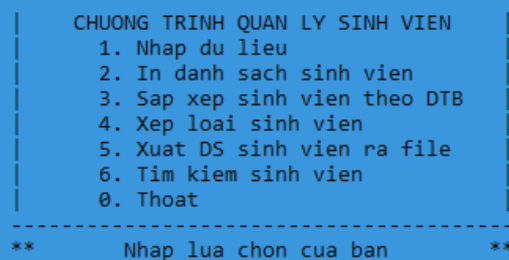
- Chào tất cả mọi người, sau đây mình xin hướng dẫn mọi người cách sử dụng chương trình của nhóm chúng mình.
- Đầu tiên ta sẽ nhập số lượng sinh viên cần nhập vào danh sách.



```
Nhap so luong SV: 3
```

Hình 2.1

- Sau khi nhập số lượng sinh viên xong màn hình sẽ chuyển sang các sự lựa chọn sau.



```
CHUONG TRINH QUAN LY SINH VIEN
1. Nhap du lieu
2. In danh sach sinh vien
3. Sap xep sinh vien theo DTB
4. Xep loai sinh vien
5. Xuat DS sinh vien ra file
6. Tim kiem sinh vien
0. Thoat

**      Nhap lua chon cua ban      **
```

Hình 2.2

- Trước khi chọn các sự lựa chọn phía dưới thì việc đầu tiên ta cần làm là nhập dữ liệu trước, nếu không có dữ liệu các lựa chọn phía dưới sẽ không chạy và nó sẽ yêu cầu bạn nhập danh sách sinh viên trước.
- Vì vậy chúng ta sẽ bắt buộc phải chọn lựa chọn thứ nhất để thực hiện việc quản lý sinh viên.
- Đầu tiên ta sẽ nhập họ tên sinh viên, tiếp theo là giới tính và tuổi, cuối cùng là điểm 3 môn toán, lý, hóa, như hình dưới.



```
1
CHUONG TRINH QUAN LY SINH VIEN
1. Nhap du lieu
2. In danh sach sinh vien
3. Sap xep sinh vien theo DTB
4. Xep loai sinh vien
5. Xuat DS sinh vien ra file
6. Tim kiem sinh vien
0. Thoat
**      Nhap lua chon cua ban      **

Ban da chon nhap DS sinh vien!

Nhap SV thu 1:
Nhap ten: Long
Nhap gioi tinh: Nam
Nhap tuoi: 19
Nhap diem 3 mon: 10 10 10

Nhap SV thu 2:
Nhap ten: Nhat
Nhap gioi tinh: Nam
Nhap tuoi: 19
Nhap diem 3 mon: 1 2 3

Nhap SV thu 3:
Nhap ten: Nhi
Nhap gioi tinh: Nu
Nhap tuoi: 19
Nhap diem 3 mon: 10 10 10

Ban da nhap thanh cong!
Bam phim bat ky de tiep tuc!
```

Hình 2.3

- Sau khi nhập hoàn tất, nếu bạn chọn lựa chọn thứ hai thì chương trình sẽ xuất danh sách bạn vừa nhập ra màn hình, như hình dưới:

```
Ban da chon xuat DS sinh vien!

Thong tin SV thu 1:
Ho ten SV: Long
Gioi tinh: Nam
Tuoi SV : 19
Diem Toan - Ly - Hoa: 10.00 - 10.00 - 10.00
Diem TB: 10.00
Thong tin SV thu 2:
Ho ten SV: Nhat
Gioi tinh: Nam
Tuoi SV : 19
Diem Toan - Ly - Hoa: 1.00 - 2.00 - 3.00
Diem TB: 2.00
Thong tin SV thu 3:
Ho ten SV: Nhi
Gioi tinh: Nu
Tuoi SV : 19
Diem Toan - Ly - Hoa: 10.00 - 10.00 - 10.00
Diem TB: 10.00

Bam phim bat ky de tiep tuc!
```

Hình 2.4

- Tiếp theo nếu bạn chọn lựa chọn thứ ba thì chương trình sẽ sắp xếp điểm trung bình 3 môn toán, lý, hóa theo thứ tự tăng dần, như hình dưới.

```
Ban da chon sap xep SV theo DTB!

Thong tin SV thu 1:
Ho ten SV: Nhat
Gioi tinh: Nam
Tuoi SV : 19
Diem Toan - Ly - Hoa: 1.00 - 2.00 - 3.00
Diem TB: 2.00
Thong tin SV thu 2:
Ho ten SV: Long
Gioi tinh: Nam
Tuoi SV : 19
Diem Toan - Ly - Hoa: 10.00 - 10.00 - 10.00
Diem TB: 10.00
Thong tin SV thu 3:
Ho ten SV: Nhi
Gioi tinh: Nu
Tuoi SV : 19
Diem Toan - Ly - Hoa: 10.00 - 10.00 - 10.00
Diem TB: 10.00

Bam phim bat ky de tiep tuc!
```

Hình 2.5

- Lựa chọn thứ tư là xếp loại sinh viên theo điểm trung bình nếu điểm trung bình sinh viên nào lớn hơn hoặc bằng 8 sẽ được xếp loại giỏi, lớn hơn hoặc bằng 6.5 sẽ được khá còn lớn hơn hoặc bằng 4 sẽ là trung bình. Ngược lại nếu bé hơn 4 sẽ xếp loại yếu, như hình dưới:

```
Ban da chon thoat xep loai SV!  
  
Xep loai cua SV thu 1 la: Yeu  
Xep loai cua SV thu 2 la: Gioi  
Xep loai cua SV thu 3 la: Gioi  
  
Bam phim bat ky de tiep tuc!
```

Hình 2.6

- Lựa chọn tiếp theo là lựa chọn thứ năm là xuất danh sách sinh viên ra file dạng txt ở bên ngoài, khi bạn chọn lựa chọn thứ 5 chương trình sẽ ra như hình dưới:

```
Ban da chon xuat DS SV!  
Xuat DSSV thanh cong vao file DSSV.txt!  
Bam phim bat ky de tiep tuc!
```

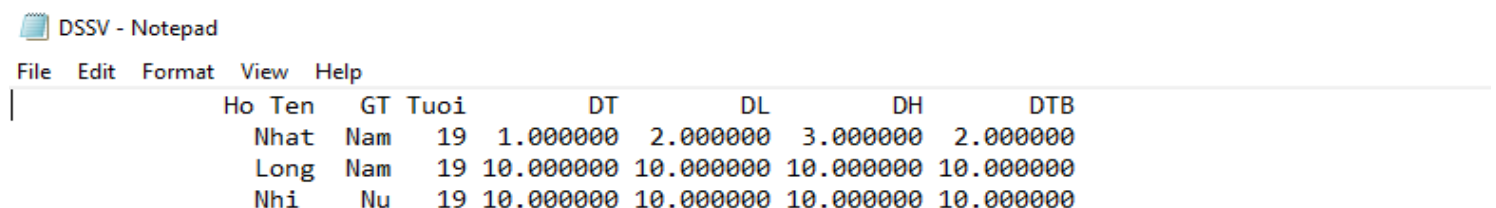
Hình 2.7

- Sau khi xuất file thành công nó sẽ hiển thị file cùng với nơi ta đã lưu chương trình.



Hình 2.8

- Mở file lên ta sẽ thấy nó đã xuất danh sách sinh viên ra thành dạng txt.



Hình 2.9

- Lựa chọn tiếp theo là tìm kiếm theo tên sinh viên, đầu tiên bạn sẽ nhập tên sinh viên cần tìm.

```
Ban da chon tim kiem thong tin sinh vien
_____
Nhap ten sinh vien can tim: Long_
```

Hình 2.10

- Sau đó chương trình sẽ xuất ra thông tin của sinh viên đó.

```
Ban da chon tim kiem thong tin sinh vien
_____
Nhap ten sinh vien can tim: Long
Ket qua tim kiem sinh vien Long:
Ho ten SV: Long
Gioi tinh: Nam
Tuoi SV : 19
Diem Toan - Ly - Hoa: 10.00 - 10.00 - 10.00
Diem TB: 10.00
_____
Bam phim bat ky de tiep tuc!_
```

Hình 2.11

- Nếu tên sinh viên bạn tìm không có trong danh sách thì chương trình sẽ không hiển thị tên sinh viên nào cả.
- Và lựa chọn cuối cùng đó là thoát chương trình, khi bạn chọn thì bạn hãy bấm phím bất kỳ 2 lần để thoát chương trình nhé.

# Chương III: Kết Luận

## 3.1 Kết quả đạt được

- Nhóm chúng em đã hoàn thành xong chương trình quản lý sinh viên với các chức năng cơ bản khác nhau gồm: In danh sách sinh viên ra file “txt”, sắp xếp sinh viên theo điểm trung bình, xếp loại sinh viên, tìm kiếm sinh viên theo tên, ...

## 3.2 Nhược điểm

- Chưa hiểu sâu về lý thuyết của danh sách liên kết đơn.
- Không có nhiều ý tưởng về các chức năng mới mẻ cho chương trình quản lý sinh viên.

## 3.3 Hướng phát triển

- Nhóm chúng em sẽ cố gắng học hỏi và phát triển chương trình quản lý sinh viên với nhiều chức năng hơn, ứng dụng vào thực tế nhiều hơn, thoả mãn nhu cầu của người sử dụng trong xã hội.

## 3.4 Tài liệu tham khảo

- Codelearn.io, daynhayhoc.com, laptrinhkhongkho.com, vietjack.com, topdev.vn...