**BÁO CÁO BÀI TẬP LỚN**

**LỜI CÁM 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 cám 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, thầy Trần Phong Nhã là người đã trực tiếp hướng dẫn và chỉ bảo chúng em trong quá trình thực hiện đề tài “Quản lý trường mầm non”.

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

**CHƯƠNG 1: CƠ SỞ LÝ THUYẾT**

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

2

**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 triễn…………………………………….....................29

3.4 Tài liệu tham khảo………………………………………………...29

**LỜI MỞ ĐẦU**

Trong đời sống chúng ta hiện nay, sự phát triển của Khoa học và Công nghệ đang làm thay đổi thế giới một cách chóng mặt. Vì thế bất kì quốc gia, tập đoàn, công ty, doanh nghiệp, trường học hay bất kì cá nhân nào cũng cần phải ứng dụng được khoa học công nghệ vào lĩnh vực của mình. Đặc biệt với ngành Công nghệ thông tin đã giúp ích rất nhiều cho các công việc học tập, làm việc và nghiên cứu,… giúp cho mọi thứ trở nên đơn giản và tiện lợi hơn. Và trong những năm gần đây, lượng trẻ em được sinh ra ngày càng tăng lên dẫn đến việc các trường mầm non công và tư ngày càng mọc lên nhiều và cũng có sức chưa ngày càng nhiều. Cho nên dẫn đến việc quản lí trẻ trong một trường mầm non ngày càng tốn nhiều công sức hơn. Từ đó dẫn đến nhu cầu đơn giản hóa các thông tin, học phí,… của các bé ngày càng cao. Vì vậy chúng em xin mang đến cho mọi người ứng dụng quản lý thông tin và học phí trong tháng đầu tiên của một trường mầm non

**CHƯƠNG I: LÝ THUYẾT**

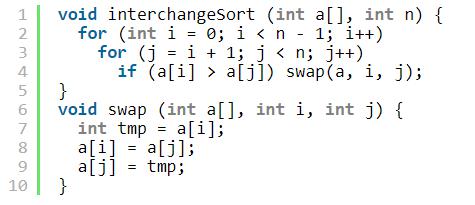
**1.1 Các thuật toán sắp xếp**

*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:

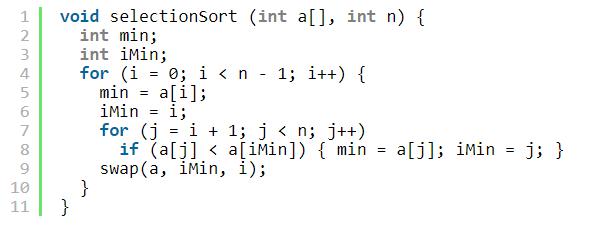


Hình 1.1

5

**1.1.2 Sắp xếp chọn**

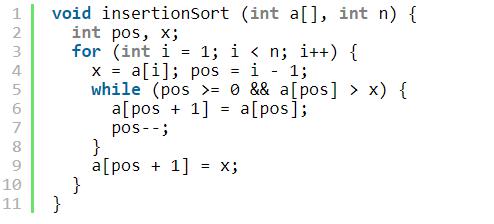
* Tìm phần tử nhỏ nhất trước, rồi mới đổi chỗ nó cho phần tử đầu tiên, xong lặp lại với phần còn lại.
* Code:



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 mớ 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 nong lấy một lỗ trống, rồi mới chèn x (mang giá trị của a[i]) vào.
* Code:

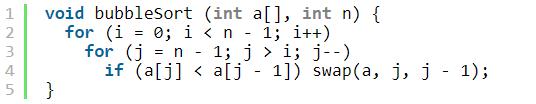


Hình 1.3

6

**1.1.4 Sắp xếp nổi bọt**

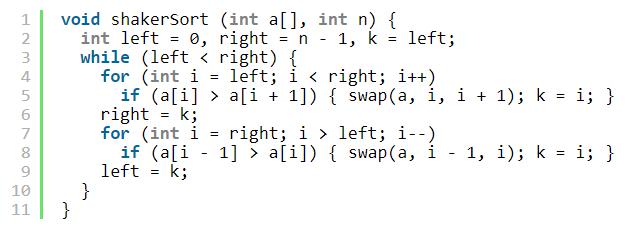
* Lần lượt đổi chỗ các cặp đôi phần tử cạnh nhau nếu chúng ngược thứ tự, mục đích cũng là sau một lượt, phần tử bé nhất sẽ về đầu dãy.
* Code:



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:



Hình 1.5

7

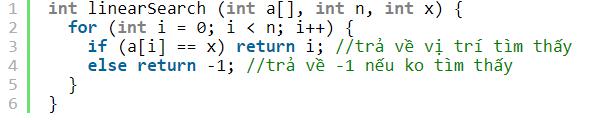
**1.2 Thuật toán tìm kiếm**

*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:



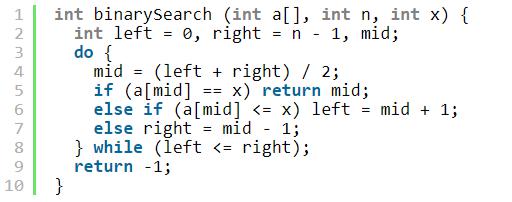
Hình 1.6

**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.

8

- Code:



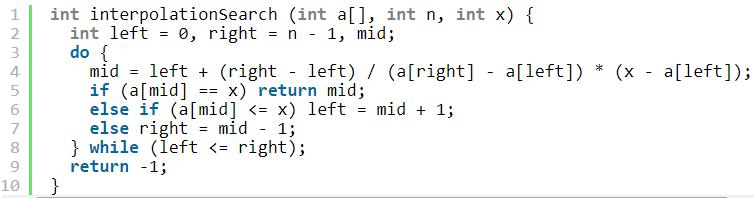
Hình 1.7

**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 + | | !"#$%& ()\*% | (x – a[left]) giúp thu gọn khoảng tìm kiếm hơn. Chỉ |
| cần thay biểu thức |  | +[!"#$%]&+[()\*%] |  |
|  | tính mid này vào code mẫu của thuật toán tìm kiếm nhị phân là được. | | |

- Code:



Hình 1.8

9

**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 giữa 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.

10

- 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**.



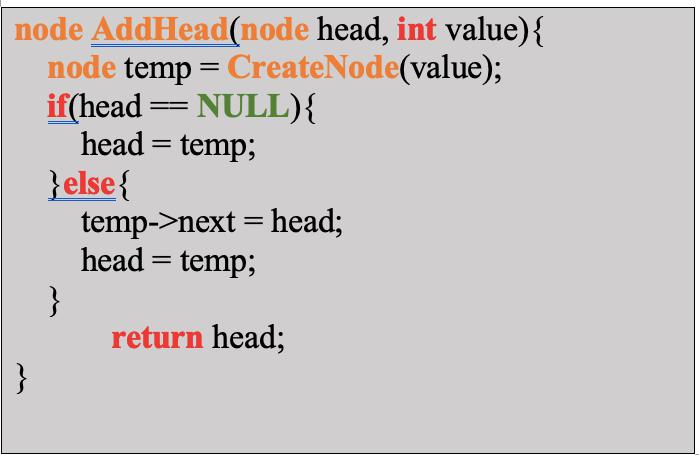
**1.3.2 Thêm Node vào danh sách liên kết đơn**

* + - **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:

1. Cho **next** của temp trỏ tới **head** hiện hành o Đặt temp làm **head** mới

11

* Code:



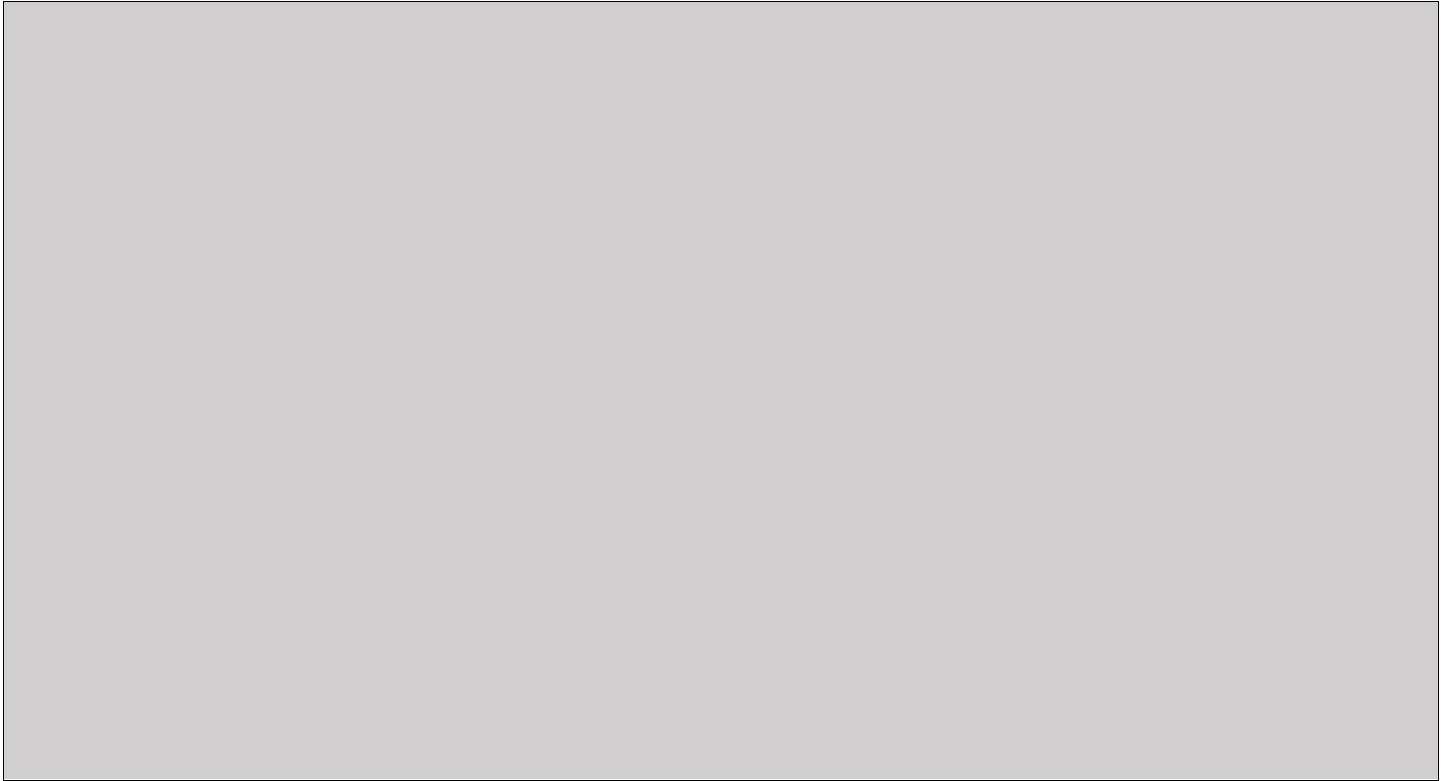
Hình 1.11

* + **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 trỏ **next** của phần tử cuối tới **Node** mới (temp).

12

- 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 trỏ tới **Node** mà **Q** đang trỏ tới
  + Cho **Node Q** trỏ tới **Node** mới

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

13

- 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->**next** = temp trước. Khi đó, bạn sẽkhông thể lấy lại phần sau của danh sách liên kết nữa (Vì **next** chỉ được được lưu trong p->**next** mà thay đổi p->**next** rồi thì còn đâu giá trị cũ).

**1.3.3 Xoá 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****.*

14

- Code:

**node DelHead**(**node** head){



**if**(head == **NULL**){

printf("\nCha co gi de xoa het!");

}**else**{

head = head->next;

}

**return** head;

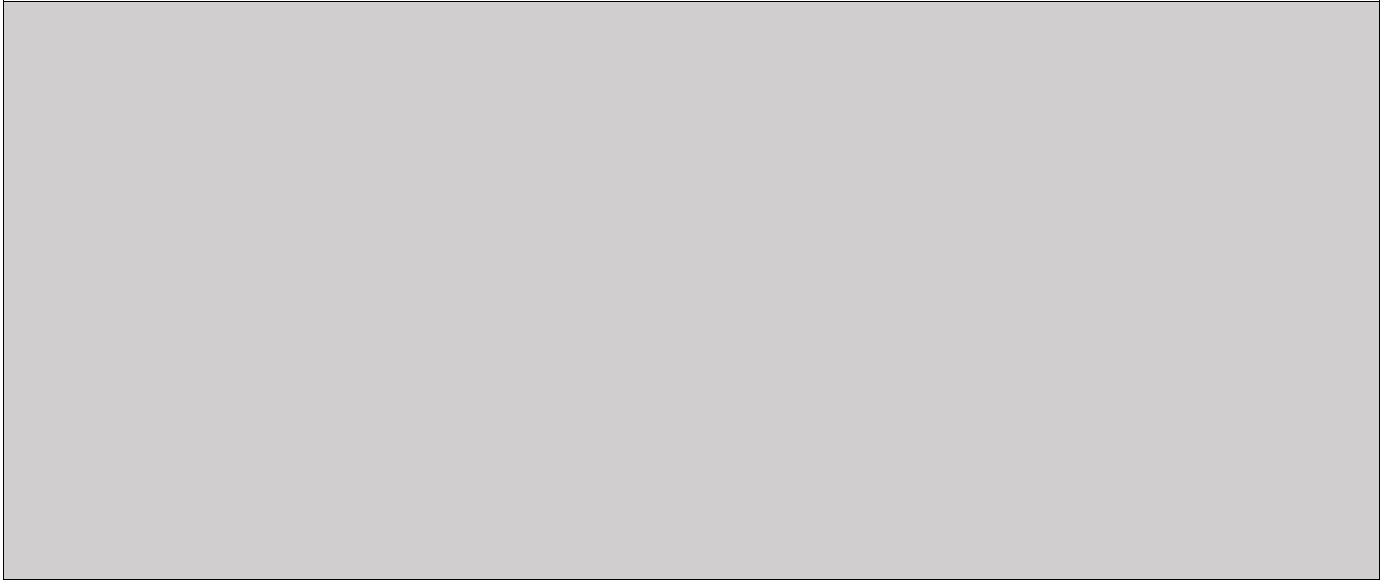
}

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->****next****->****next*** *=* ***NULL****.* Bạn cho **next** của nó bằng **NULL** là xong.

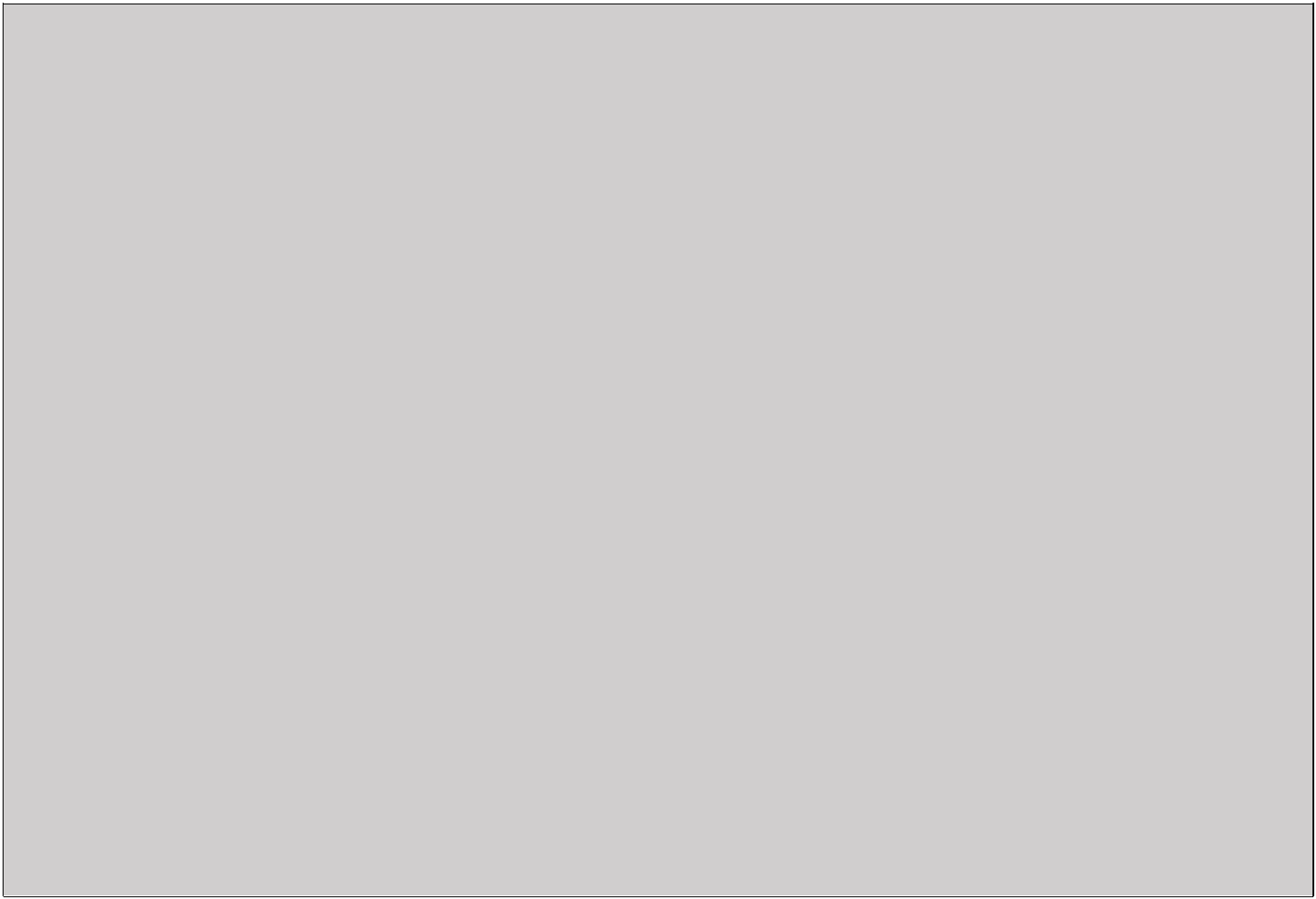
**\* Xóa ở vị trí bất kỳ**

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

15

**Lưu ý**: Chỉ số xóa bắt đầu từ 0 nhé. Việc tìm vị trí càn xóa chỉ duyệt tới **Node** gần cuốithôi (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 nguyênkhông âm. Nếu bạn muốn kiểm tra chỉ số hợp lệ thì nên kiểm tra trước khi gọi hàm này.

16

- 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**.

17

- 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;

}

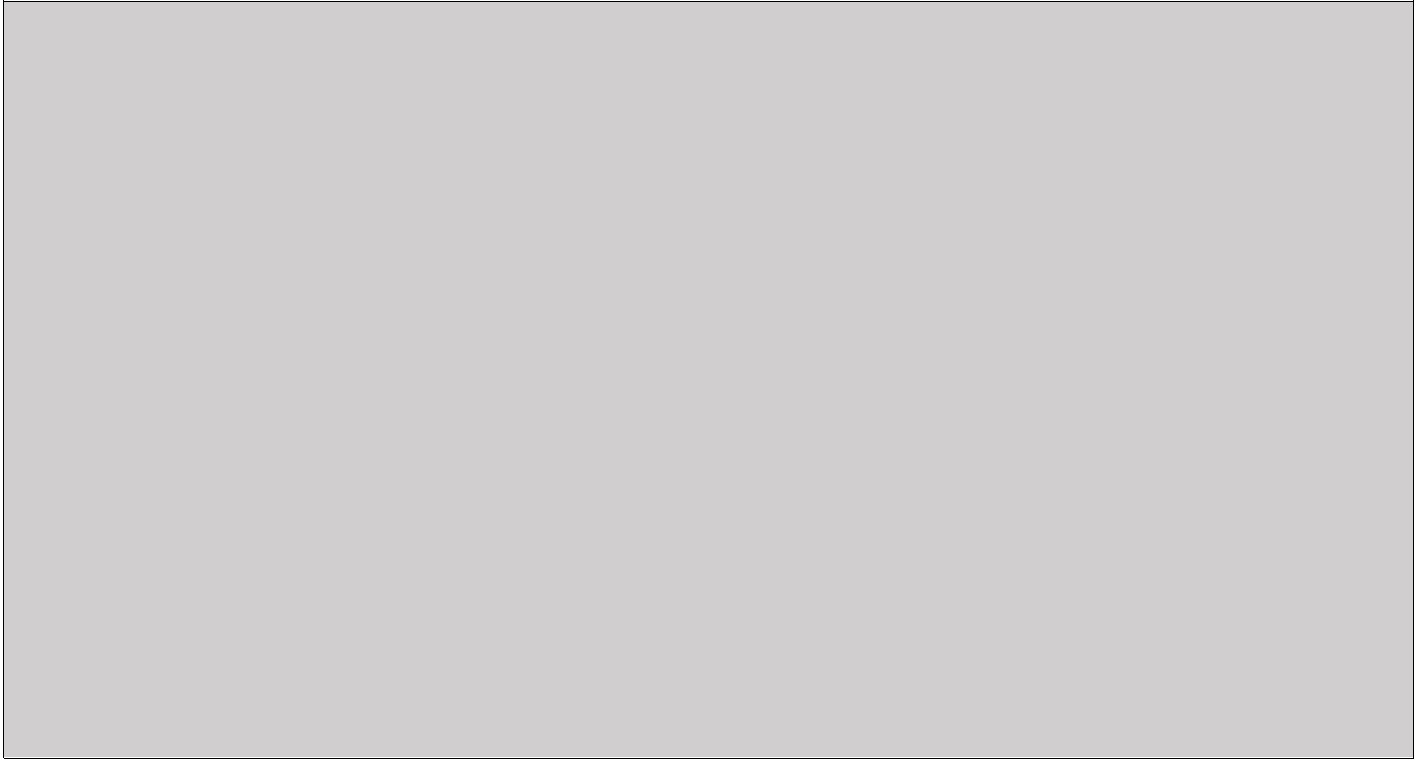
Hình 1.20

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

18

- 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;

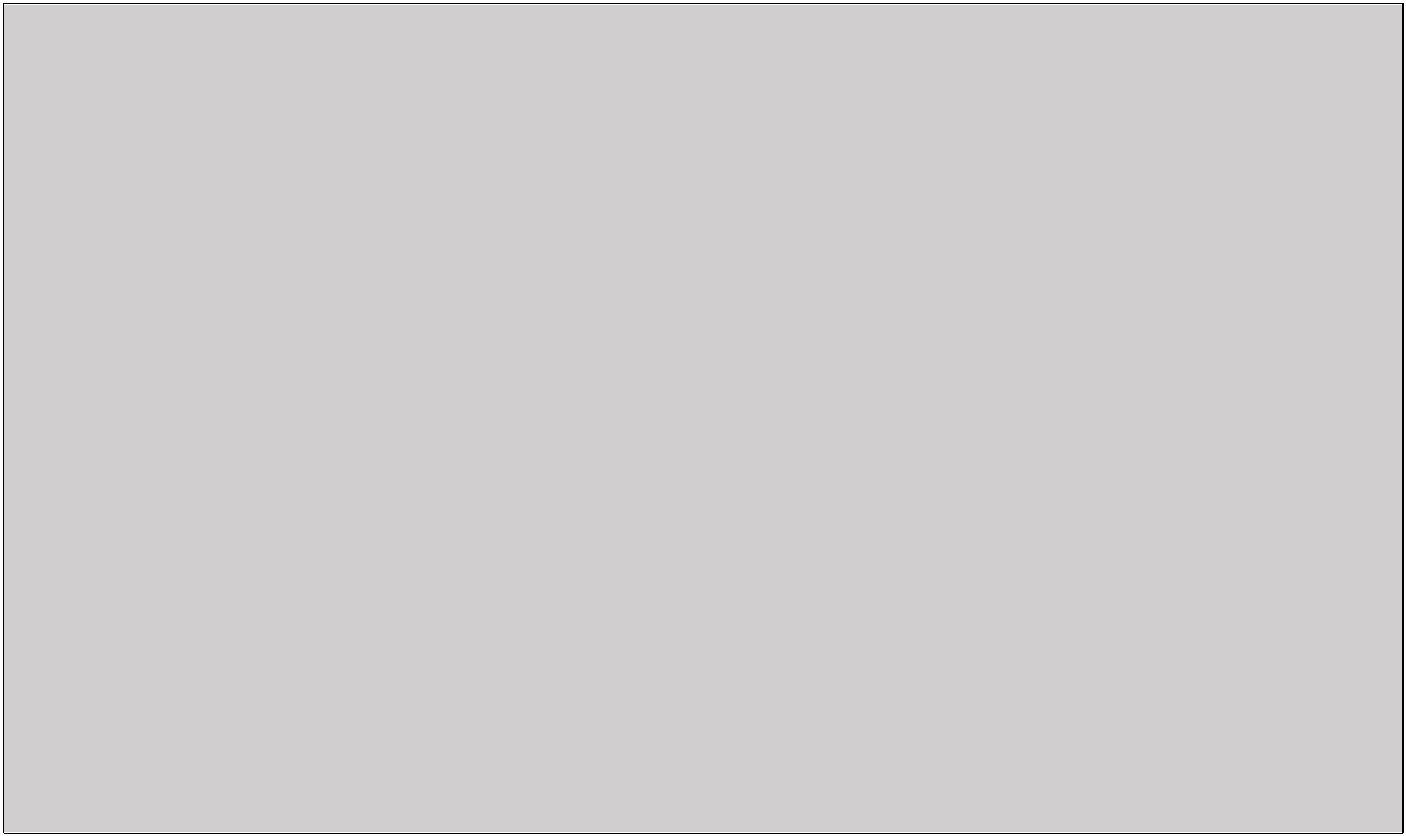
}

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 = **Inpu**t(); **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

19

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

**1. Khai báo biến file**

o **<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

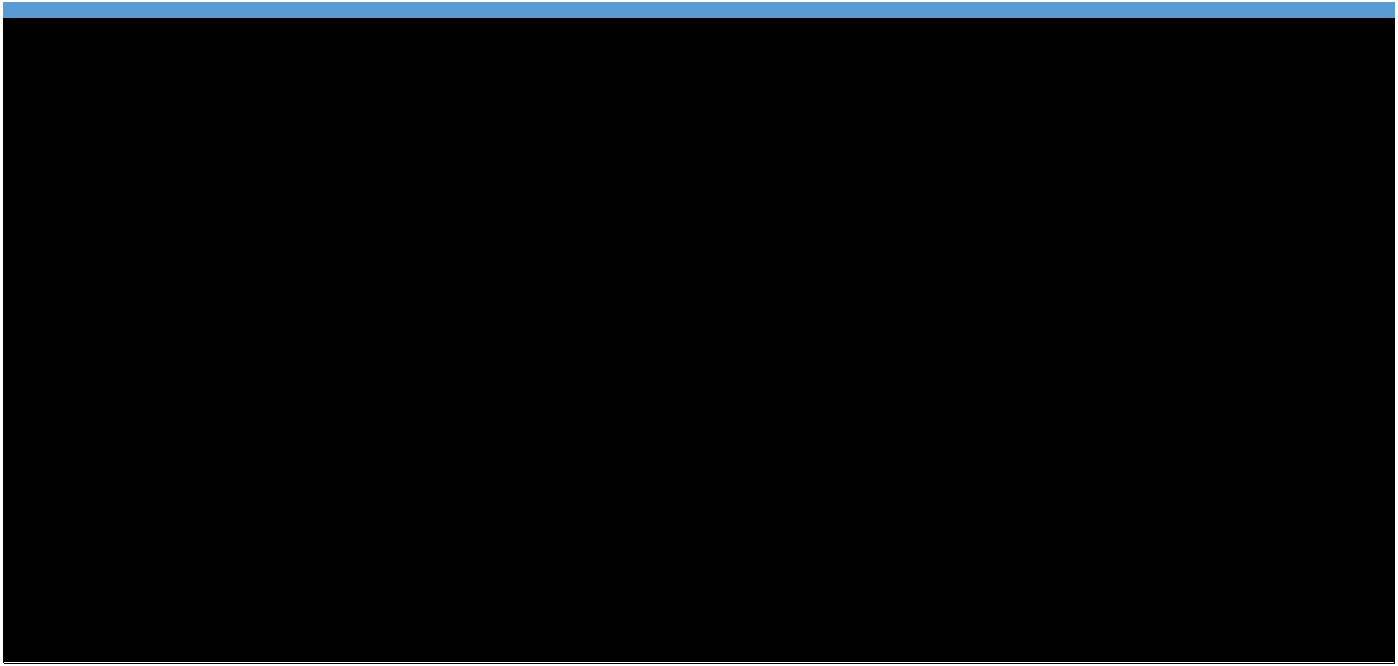
FILE \*f2, \*f3; //khai báo nhiều biến file

Hình 1.24

20

* 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 2100 3 4 5

==Thu xoa 1 phan tu khoi linked list==

1100 3 4 5

==Thu tim kiem 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, tắt máy 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.
  + Hiển 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.

**FILE \*<tên biến file> ;** o Kiểu dữ liệu của file là kiểu FILE \*



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

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

1. **fopen** là hàm mở file
2. **<duong dan file> :** là chuỗi chứa đường dẫn tới file bạn muốn mở.
3. **<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 :**

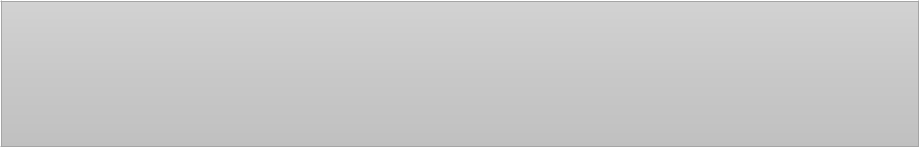
**“w”**: mở file ghi. Nếu file muốn mở chưa có thì chương trình sẽ tự động tạora file đó.

**“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.

1. **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 thuoc>,<biến file>);

1. 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.

21

* 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 làEOF (End Of File)

* 1. **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**

Vì việc phải quản lý các giấy tờ về lí lịch (tên, tuôi, ngày sinh,..) có thể gặp sai sót, hoặc mất thông tin hay muốn quản lý tiền học phí dễ dàng hơn và tiện lợi hơn. Nên chúng em muốn tạo một ứng dụng để quản lí tất cả những thông tin về trường mầm non, về từng bé trong trường để những khó khan đó được giải quyết. Cũng như là giúp phụ huynh kiểm soát việc đóng học phí của con mình dễ dàng hơn, và giúp người hiệu trưởng quản lý thông tin đó dễ dàng hơn trong một ứng dụng dễ hiểu dễ sử dụng.

**2.2 Ý tưởng làm đề tài**

Muốn làm phần mềm quản lý trường mầm non 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**

* Khi chạy chương trình, sẽ xuất hiện một bảng menu với các lựa chọn bằng số, nhập một số tương ứng với một chức năng trong Menu.

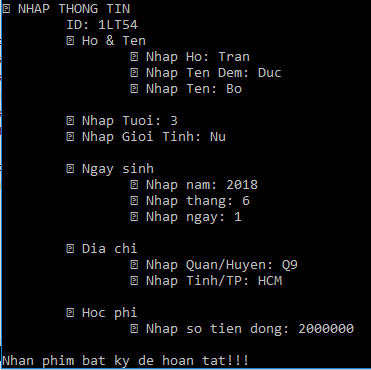


* Nếu nhập số khác thì chương trình sẽ báo sai và nhập lại:



* Chức năng 1: Nhập thông tin bé. Ở chức năng này, chúng ta sẽ nhập thông tin một bé cần nhập, trong thực tế thì chức năng này nhằm mục đích xác nhận nhập học.

+ Bằng cách ấn phím 1: Người quản lí (Hiệu trưởng) sẽ là người trực tiếp nhập các thông tin bé nhập học vào đây. Nhập lần lượt các thông tin theo những gì chương trình đưa ra. Tên -> Tuổi -> Giới tính -> Ngày tháng năm sinh -> Địa chỉ -> Học phí.

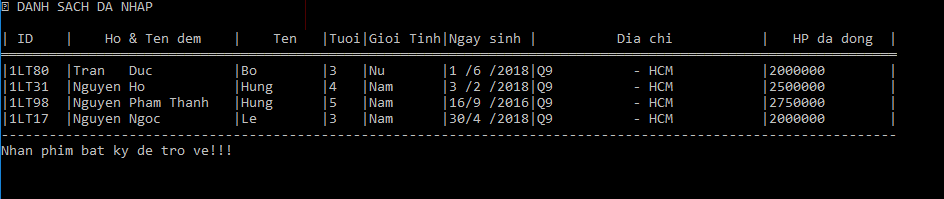


+ Bạn có thể nhập thông tin các bé mới bằng cách ấn vào phím 1 và làm như cũ nhé .

+ Sau khi nhập xong thì thông tin đã được lưu rồi nè. Chuyển sang chức năng 2 thui nèo !!! <3

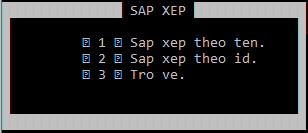
* Chức năng 2: Xuất thông tin bé: Sau khi thực hiện chức năng 1 thì chức năng 2 sử dụng được rồi nè.

+ Bằng cách ấn phím 2. Các thông tin bạn đã nhập ở chức năng 1 sẽ được xuất ra dưới dạng thống kê theo 1 bảng. Trong thực tế chức năng này giúp người quản lí dễ quan sát các thông tin hơn.

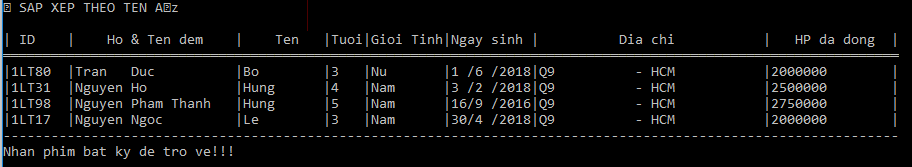


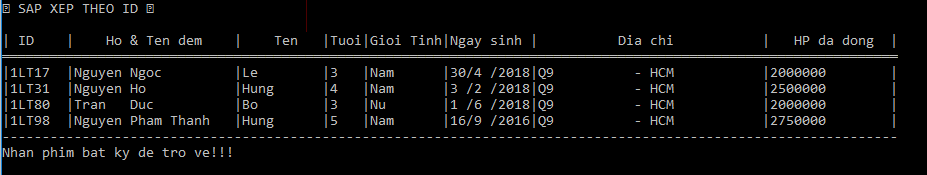
* Chức năng 3: Sắp xếp thông tin.

+ Bằng cách ấn phím 3. Trong chức năng này bạn sẽ có 2 sự lựa chọn cách sắp xếp. 1 là xếp theo tên (tức xếp theo bảng chữ cái). 2 là xếp theo ID (tức xếp theo ID lúc nhập thông tin theo thứ tự từ bé đến lớn). Và lỡ ấn nhầm thì có thể trở về Menu bằng cách ấn phím 3 nè.



+ Sau khi chọn 1 trong 2 lựa chọn trên thì máy tính sẽ cho ra một thống kê thông tin bằng bảng. Những thông tin nãy đã được sắp xếp theo yêu cầu của bạn.

 Sắp xếp theo tên

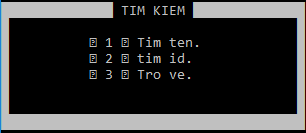


Sắp xếp theo id

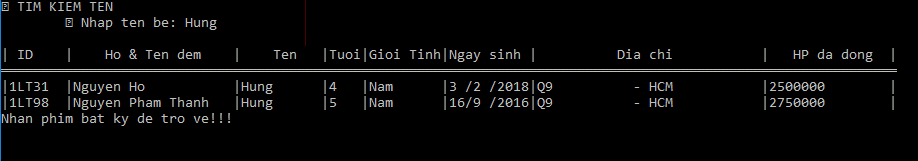
+ Trong thực tế: chức năng này cũng giúp tối ưu hơn việc quản lí các thông tin bé hơn.

* Chức năng 4: Tim kiếm thông tin:

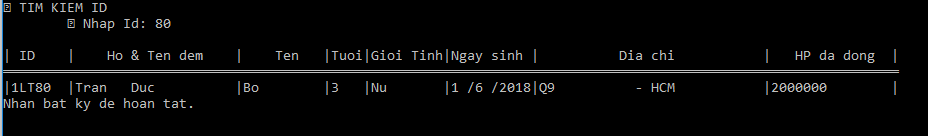
+ Bằng cách ấn phím 4: Sẽ cho phép bạn lựa chọn tìm kiếm thông tin bé theo 2 cách.



\*1 là tìm theo tên: Bạn sẽ nhập phần tên của bé và hệ thống sẽ in ra các bé có tên như vậy thành một bảng.



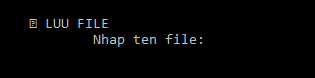
\*2 là tìm theo ID: tương tự như 1 nhưng sẽ chỉ có 1 thông tin in ra vì ID là duy nhất.

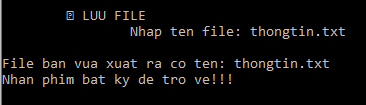


+ Trên thực tế: chức năng này dùng để phụ huynh kiểm tra thông tin của con mình đã đúng chưa hoặc người quản lí muốn rà soát thông tin từng bé để quản lí. Và bản thân người phụ huynh sẽ phải ghi nhớ ID này.

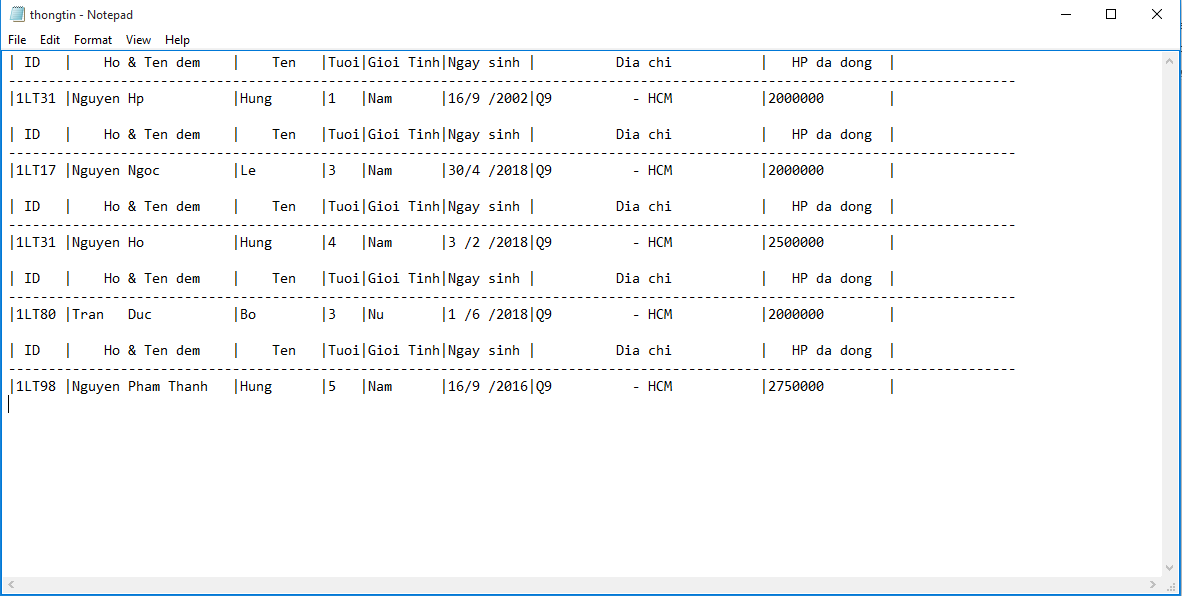
* Chức năng 5: Lưu file:

+ Bằng cách ấn phím 5: chương trình sẽ yêu cầu bạn nhập tên file muốn đặt. (yêu cầu đuôi phải có .txt)





+ Sau đó mọi thông tin đã nhập ở chức năng 1 sẽ được lưu lại vào file dưới dạng văn bản và khi tắt chương trình. Những thông tin được lưu vào file sẽ không thể mất trừ khi bạn tìm đến file đó và delete.



+ Trên thực tế: chức năng này nhằm đảm bảo thông tin sẽ được lưu sau khi tắt chương trình và cũng là một chứng năng thiết yêu cho bất kì một ứng dụng quản lí thông tin nào.

* Chức năng 6: Tắt chương trình:

+ Bằng cách ấn phím 0: chương trình sẽ ngừng lại.

**3.1 Kết quả đạt được**

* Nhóm chúng em đã hoàn thành xong chương trình quản lý thông tin các bé trong tháng đầu tiên với các chức năng cơ bản khác nhau gồm: In danh sách bé ra file “txt”, sắp xếp thông tin bé theo ID hoặc tên, tìm kiếm thông tin theo tên hoặc ID, …

**3.2 Nhược điểm**

* Chưa hiểu sâu về lý thuyết của danh sách liên kết đơn.
* Chưa có nhiều chức năng mới lạ.

**3.3 Hướng phát triển trong tương lai**

* Nếu có cơ hội, chúng em có thể thực hiện một công trình nghiên cứu ứng dụng này sao cho thêm nhiều chức năng mới lạ hơn và áp dụng vào cuộc sống nhiều hơn nhằm mục đích giúp cho việc quản lý thông tin không chỉ trong một trường mầm non mà có thể là tường cấp 1,2,3, đại học, các công ty, doing nghiệp,… trở nên dễ dàng hơn.