TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI

PHÂN HIỆU THÀNH PHỐ HỒ CHÍ MINH

---🙢🕮🙠---



**BÁO CÁO BÀI TẬP LỚN**

ĐỀ TÀI:

QUẢN LÍ NHÂN VIÊN

**GVHD**: GV Trần Thị Dung

Th.S Trần Phong Nhã

**SVTH:**  Văn Công Hào (Nhóm trưởng)

Trương Quang Duy

Hồ Chí Minh, Ngày 06 tháng 05 năm 2021.

**Lời mở đầu**

Để hoàn thành bài báo cáo, chúng em xin gửi lời cảm ơn chân thành đến Cô Trần Thị Dung, Thầy Trần Phong Nhã đã tận tình giảng dạy và cung cấp những kiến thức cần thiết, bổ ích để chúng em hoàn thành tốt báo cáo bài tập.

Trong quá trình làm báo cáo bài tập, chúng em cũng xin cảm ơn sự giúp đỡ của các bạn trong lớp Công nghệ thông tin khóa 61 đã cho nhóm chúng em một vài lời khuyên hữu ích.

Với điều kiện thời gian hạn chế của một học viên, báo cáo của chúng em không thể tránh được những thiếu sót. Chúng em rất mong nhận được sự chỉ bảo, đóng góp ý kiến của quý Thầy Cô để chúng em có điều kiện bổ sung, nâng cao kiến thức của bản thân.

Chúng em xin chân thành cảm ơn!

MỤC LỤC

[**CHƯƠNG I. LÝ THUYẾT: 1**](#_Toc71146809)

[**1.Cấu trúc struct: 1**](#_Toc71146810)

[**1.1 Khái niệm: 1**](#_Toc71146811)

[**1.2 Cách định nghĩa struct: 1**](#_Toc71146812)

[**1.3 Cách khai báo biến: 1**](#_Toc71146813)

[**1.4 Truy xuất các thuộc tính của struct: 2**](#_Toc71146814)

[**1.5 Từ khóa typedef: 2**](#_Toc71146815)

[**2. Con trỏ: 2**](#_Toc71146816)

[**2.1 Định nghĩa: 2**](#_Toc71146817)

[**2.2 Cách khai báo: 2**](#_Toc71146818)

[**2.3 Cách sử dụng con trỏ: 3**](#_Toc71146819)

[**3. Mảng: 3**](#_Toc71146820)

[**3.1 Định nghĩa: 3**](#_Toc71146821)

[**3.2 Khai báo và khởi tạo mảng 1 chiều: 3**](#_Toc71146822)

[**4. Con trỏ và mảng một chiều: 4**](#_Toc71146823)

[**5. Cấp phát bộ nhớ: 5**](#_Toc71146824)

[**5.1 Khái niệm: 5**](#_Toc71146825)

[**5.2 Một số hàm cấp phát: 5**](#_Toc71146826)

[**6. Làm việc với file: 5**](#_Toc71146827)

[**6.1 Lợi ích của file: 5**](#_Toc71146828)

[**6.2 Các loại file: 5**](#_Toc71146829)

[**6.3 Các thao tác với file: 6**](#_Toc71146830)

[**7. Danh sách liên kết: 8**](#_Toc71146831)

[**7.1 khái niệm: 8**](#_Toc71146832)

[**7.2 Các thao tác trong danh sách liên kết: 8**](#_Toc71146833)

[**8. Các thuật toán tìm kiếm: 13**](#_Toc71146834)

[**8.1 Tìm kiếm tuyến tính (Linear search): 13**](#_Toc71146835)

[**8.2 Tìm kiếm nhị phân (Binary search): 14**](#_Toc71146836)

[**9.1 Sắp xếp nổi bọt (Bubble sort): 14**](#_Toc71146837)

[**9.2 Sắp xếp chèn (insrertion sort): 15**](#_Toc71146838)

[**Chương II. Nguyên nhân và mục đích làm chương trình : 15**](#_Toc71146839)

[**Chương III. Khái quát về chương trình: 16**](#_Toc71146840)

[**Chương IV.Hướng dẫn sử dụng: 20**](#_Toc71146841)

[**Tài liệu tham khảo: 25**](#_Toc71146842)

# CHƯƠNG I. LÝ THUYẾT:

## 1. Cấu trúc struct:

### 1.1 Khái niệm:

Là một kiểu dữ liệu dạng tổng hợp cho phép nhiều kiểu dữ liệu được nhóm lại với nhau. Struct thường được sử dụng để tạo và lưu giữ thông tin của đối tượng (sách, sinh viên, học viên, nhân viên,..).

### 1.2 Cách định nghĩa struct:

*struct Tên cấu trúc {*

*<Kiểu dữ liệu> < biến>;*

*};*

VD: *Struct SinhVien {*

*char Ten[30];*

*int Nam Sinh;*

*float DiemTB;*

*};*

### 1.3 Cách khai báo biến:

* Khai báo trực tiếp: khai báo biến trong phần định nghĩa struct.

VD: *struct SinhVien {*

*char Ten[30];*

*int Nam Sinh;*

*float DiemTB;*

*} Sv1,Sv2; // khai báo trực tiếp*

* Khai báo gián tiếp: khai báo biến sau khi định nghĩa struct.

VD: *struct SinhVien {*

*char Ten[30];*

*int Nam Sinh;*

*float DiemTB;*

*};*

*struct SinhVien Sv1,Sv2;*

### 1.4 Truy xuất các thuộc tính của struct:

* Dùng “ . “ khi khai báo biến thường.
* Dùng “ -> “ khi khai báo biến là con trỏ.

VD: *struct SinhVien Sv1;*

*Printf(“Ten sinh vien la:%s”,Sv1.Ten);*

### 1.5 Từ khóa typedef:

Từ khóa typedef cho phép đặt tên cho kiểu dữ liệu struct ta vừa định nghĩa để đơn giản hóa việc khai báo biến.

VD*: typedef struct SinhVien {*

*char Ten[30];*

*int Nam Sinh;*

*float DiemTB;*

*} SV;*

*int main(){*

*SV Sv1,Sv2; }*

Cũng có thể như sau :

*struct SinhVien {*

*char Ten[30];*

*int Nam Sinh;*

*float DiemTB;*

*} ;*

*typedef struct SinhVien SV ;*

## 2. Con trỏ:

### 2.1 Định nghĩa:

Con trỏ là biến dùng để chứa địa chỉ của một biến khác mà nó trỏ tới.

### 2.2 Cách khai báo:

*<kiểu dữ liệu> \* <tên con trỏ>;*

VD: *int \*a;*

*float \*b;*

### 2.3 Cách sử dụng con trỏ:

*int a, \*px, ; //khai báo con trỏ*

*px=&a; // gán giá trị cho con trỏ px là địa chỉ của a*

hoặc có thể như sau: *int \*px=&a;*

Lưu ý:

* Để lấy giá trị của biến mà con trỏ trỏ tới ta lam như sau:

*\*px=a+1;* hoặc *\*&a=a+1;* tương đương với *a=a+1;*

* Con trỏ chỉ có thể thực hiện các phép toán “ + “,” - “.

## 3. Mảng:

### 3.1 Định nghĩa:

Mảng là tập hợp các phần tử có cùng kiểu dữ liệu với độ dài cố định và gồm có : mảng một chiều, mảng hai chiều,…

### 3.2 Khai báo và khởi tạo mảng 1 chiều:

* Khai báo : <*Kiểu dữ liệu> <tên mảng> [kích thước mảng];*

VD : int Array[50]; // mảng kiểu int có 50 phần tử.

* Khởi tạo: có 2 cách:
  + Khởi tạo cùng lúc với khai báo.

VD: *int Array[4]={4,3,6,5};*

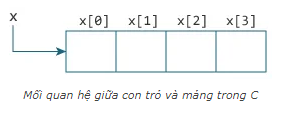
* + Khởi tạo sau khi khai báo.

VD: *int Array[4];*

*Array[1]=4;*

….

## 4. Con trỏ và mảng một chiều:

* &x[0] và x có cùng giá trị (tức là địa chỉ của x[0] = giá trị của x đang lưu trữ), x[0] =\*x ( giá trị mà x đang trỏ tới = giá trị của x[0] đang chứa).
* Tương tự: &x[1] = x+1, x[1]=\*(x+1),…

Minh họa: nhập và xuất mảng dùng con trỏ

*#include <stdio.h>*

*int main()*

*{*

*int arr[100];*

*int n;*

*do*

*{*

*printf("Nhap so luong phan tu: ");*

*scanf("%d", &n);*

*} while (n < 1);*

*for (int i = 0; i < n; i++)*

*{*

*printf("Nhap a[%d] = ", i);*

*scanf("%d", (arr + i));*

*}*

*for (int i = 0; i < n; i++)*

*{*

*printf("\nGia tri a[%d] = %d", i, \*(arr + i));*

*}*

*}*

Lưu ý: khi dùng con trỏ và mảng ta nên cấp phát bộ nhớ.

## 5. Cấp phát bộ nhớ:

### 5.1 Khái niệm:

Trước khi sử dụng mảng ta phải cung cấp rõ kích thước của mảng (tức là số lượng phần tử tối đa của mảng ). Do đó mà đôi khi mảng ta khai báo không đủ sử dụng. Để giải quyết vấn đề này, bạn có thể cấp phát thêm bộ nhớ theo cách thủ công trong thời gian chạy chương trình. Đó cũng chính là **cấp phát động trong C.**

Chú ý: trước khi cấp phát cần phải khai báo thư viện *stdlib.h .*

### 5.2 Một số hàm cấp phát:

Ta sử dụng hàm malloc() và calloc() để cấp phát bộ nhớ động, sử dụng hàm free() để giải phóng bộ nhớ. Ngoài ra, ta có thể dùng hàm realloc() để thay đổi bộ nhớ đã cấp phát.

* Hàm *malloc():* không khởi tạo giá trị ban đầu.
* Cú pháp: *<tên con trỏ> = <kiểu dữ liệu của con trỏ> malloc < kích thước >;*

VD : *ptr=(int \*) malloc( sizeof(int) );*

* Hàm *calloc():* khởi tạo các ô nhớ có giá trị bằng 0.
* Cú pháp: : *<tên con trỏ> = <kiểu dữ liệu của con trỏ> malloc < n, kích thước >;*

VD : *ptr=(int \*) calloc( 100, sizeof(int) );*

* Hàm *realloc():*
* Cú pháp: *<tên con trỏ> = realloc(tên con trỏ, n);*

VD : *ptr=realloc(ptr, sizeof(int)\*100 );*

## 6. Làm việc với file:

### 6.1 Lợi ích của file:

* Đảm bảo dữ liệu không bị mất khi chương trình ngừng chạy.
* Dễ dàng sao chép, di chuyển dữ liệu giữa các thiết bị với nhau.
* Nếu đầu vào chương trình lớn thì chỉ cần việc lưu vào file chương trình của bạn sẽ tự đọc mỗi lần khởi chạy.

### 6.2 Các loại file:

* File văn bản – text file là file thường có đuôi là *.txt*.
* File nhị phân – binary file thường có đuôi mở rộng là ***.bin*.**

### ****6.3 Các thao tác với file:****

#### ****6.3.1 Mở và đóng file:****

Trước tiền cần bạn cần khai báo 1 con trỏ kiểu FILE

Cú pháp: *FILE \*tên con trỏ;*

* Mở file: dùng hàm *fopen().*

Cú pháp : *pointer=fopen(“filename”,”mode”);*

Trong đó:

pointer: con trỏ kiểu FILE .

mode: tham số chỉ định ( VD : w,wb,ab,w+…)

VD: *FILE \*file;*

*file=fopen(“employee.dat”,”w+”);*

Một số mode:

* Đóng file: dùng hàm *fclose().*

Cú pháp : *fclose(pointer);*

VD : để đóng file employee.dat

*fclose(file);*

#### 6.3.2 Đọc/Ghi file văn bản:

Để đọc/ghi file văn bản ta dùng hàm *fprintf()* và *fscanf().*

* Ghi file:

VD: *int a ;*

*FILE \*file;*

*file = fopen(“employee.dat","w+");*

*printf(“nhap a: ");*

*scanf("%d",&a);*

*!! fprintf(file,"%d",a); // ghi giá trị a vào file employee.dat*

*fclose(file);*

* Đọc file:

VD *int a ;*

*FILE \*file;*

*file = fopen(“employee.dat","w+");*

*!! fscanf(fptr,"%d", &num); // đọc giá trị a trong file employee.dat*

*printf(“Gia tri n=%d", num);*

*fclose(fptr);*

#### 6.3.3 Đọc/Ghi file nhị phân:

Để đọc/ghi file nhị phân ta dùng hàm *fwrite()* và *fread().*

* Ghi file:

fwrite(address\_data,size\_data,numbers\_data,pointer\_to\_file);

Gồm: địa chỉ của biến lưu trữ dữ liệu; kích thước của biến; số lượng kiểu dữ liệu của biến và con trỏ tới file muốn ghi dữ liệu.

VD: *fwrite(&a , sizeof(int) , 1 , file);*

* Đọc file:

*fread(address\_data,size\_data,numbers\_data,pointer\_to\_file);*

Tương tự ghi file.

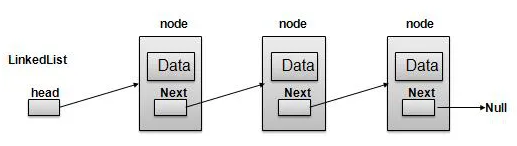
VD : *fwrite(&a , sizeof(int) , 1 , file);*

*printf(“Gia tri n=%d", num);*

## 7. Danh sách liên kết:

### 7.1 khái niệm:

Là tập hợp của nhiều Node , mỗi Node gồm Data và Next. Các Node liên kết với nhau nhờ vào Next của mỗi Node . Riêng Next của Node cuối thì trỏ NULL.



### 7.2 Các thao tác trong danh sách liên kết:

#### 7.2.1 Khai báo danh sách liên kết:

VD *: typedef struct LinkedList node {*

*int data; // data lưu giá trị*

*node \*next; // dùng next trỏ đến node kế tiếp*

*};*

Ở đây, **Next lại là kiểu LinkedList** bởi vì nó là con trỏ trỏ của chính bản thân nó, và nó trỏ tới một Node kế tiếp cũng có kiểu LinkedList.

#### 7.2.2 Tạo một Node:

*node CreateNode(int value){*

*node \*temp; // khai báo một node*

*temp = (node\*)malloc(sizeof(struct LinkedList)); // cấp phát bộ nhớ cho node*

*temp->next = NULL;// Cho next trỏ tới NULL*

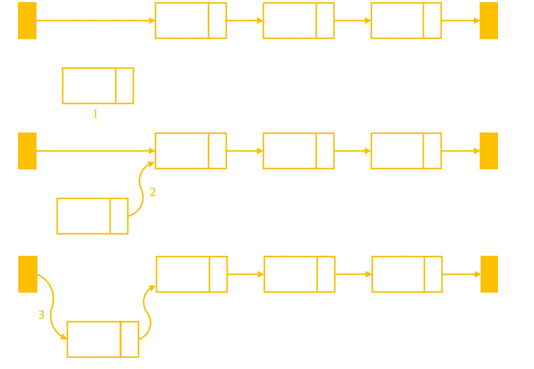
*temp->data = value; // Gán giá trị cho Node*

*return temp; //Trả về node mới đã có giá trị*

*}*

#### 7.2.3 Chèn Node:

* Chèn vào đầu: việc này chính là cập nhật lại head. Ta gọi Node mới là temp:



• Nếu head đang trỏ tới NULL, nghĩa là danh sách liên kết đang trống, Node mới thêm vào sẽ làm head luôn.

• Ngược lại, ta phải thay thế thằng head cũ bằng head mới. Việc này phải làm theo thứ tự như sau:

* Cho next của temp trỏ tới head đang có.
* Đặt temp làm head mới.

Minh họa:

*node AddHead(node head, int value){*

*node temp = CreateNode(value); // Khởi tạo node temp với data = value*

*if(head == NULL){*

*head = temp; // //Nếu linked list đang trống thì Node temp là head luôn*

*}else{*

*temp->next = head; // Trỏ next của temp = head hiện tại*

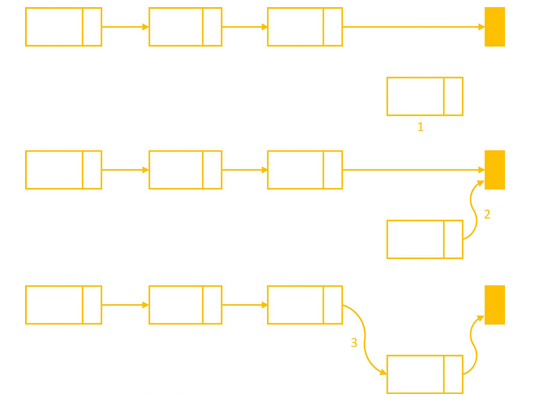
*head = temp; // Đổi head hiện tại = temp(Vì temp bây giờ là head mới mà)*

*}*

*return head;*

*}*

* Chèn vào cuối:



• Tạo một node mới (temp) cần chèn có giá trị là value.

• Nếu head=NULL thì danh sách liên kết đang trống, temp chèn vào sẽ trở thành head.

• Ngược lại, ta sẽ duyệt tới Node cuối cùng (Node có next = NULL), và trỏ next của Node cuối tới Node mới (temp).

Minh họa:

*node AddTail(node head, int value){*

*node temp,p;// Khai báo 2 node tạm temp và p*

*temp = CreateNode(value); //Gọi hàm createNode để khởi tạo node temp có next trỏ tới NULL và giá trị là value*

*if(head == NULL){*

*head = temp; //Nếu danh sách đang trống thì Node temp là head luôn*

*}*

*else{*

*p = head; // Khởi tạo p trỏ tới head*

*while(p->next != NULL){*

*p = p->next; //Duyệt danh sách liên kết đến cuối. Node cuối là Node có next = NULL*

*}*

*p->next = temp; //Gán next của Node cuối = temp. Khi đó temp sẽ là*

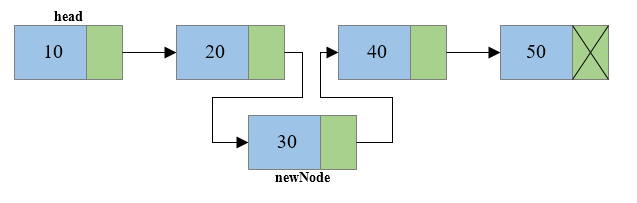
*Node cuối(temp->next = NULL mà)*

*}*

*return head;*

*}*

* Chèn vào vị trí bất kì:



Như hình trên, để chèn 30 vào giữa 20 và 40 ta làm như sau:

• Ta cho next của 30 trỏ đến 40.

• Tiếp ta cho next của 20 trỏ đến 30.

Minh họa:

*node AddAt(node head, int value, int position){*

*if (position == 0 || head == NULL){*

*head = AddHead(head, value); // Nếu vị trí chèn là 0 , ta chèn vào đầu*

*}else{*

*int k = 1; // dùng k đếm vị trí*

*node p = head;*

*while(p != NULL && k != position){*

*p = p->next;*

*++k;*

*}*

*if (k != position){*

*printf("Vi tri chen vuot qua vi tri cuoi cung!\n");*

*}else{*

*node temp = CreateNode(value);*

*temp->next = p->next;*

*p->next = temp;*

*}*

*}*

*return head;*

*}*

#### 7.2.4 Xóa Node:

* Xóa đầu: ta chỉ cần cho Node kế tiếp của head làm head là được (node kế tiếp của head: head->next).

Minh họa:

*node DelHead(node head){*

*if (head == NULL){*

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

*}else{*

*head = head->next;*

*}*

*return head;*

*}*

* Xóa cuối và xóa vị trí bất kì tương tự nhau: ta chỉ cần bỏ qua một phần tử.

• Xóa cuối:

*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; // Cho next bằng NULL*

*// Hoặc viết p->next = NULL cũng được*

*return head;*

*}*

• Xóa vị trí bất kì:

*node DelAt(node head, int position){*

*if (position == 0 || head == NULL || head->next == NULL){*

*head = DelHead(head); //Nếu vị trí xóa là 0, thì xóa đầu*

*}else{*

*int k = 1; // dùng k đếm vị trí*

*node p = head;*

*while(p->next->next != NULL && k != position){*

*p = p->next;*

*++k;*

*}*

*if (k != position){*

*printf("Vi tri xoa vuot qua vi tri cuoi cung!\n");*

*}else{*

*p->next = p->next->next;*

*}*

*}*

*return head;*

*}*

## 8. Các thuật toán tìm kiếm:

### 8.1 Tìm kiếm tuyến tính (Linear search):

Khái niệm: là một phương pháp tìm kiếm một phần tử cho trước trong một danh sách bằng cách duyệt lần lượt từng phần tử của danh sách đó cho đến lúc tìm thấy giá trị mong muốn.

*VD: tìm số lớn nhất:*

*int MaxElement(int a[], int n){*

*int maxValue = a[0];*

*for(int i = 0; i < n; i++){*

*if(a[i] > maxValue){*

*maxValue = a[i];*

*}*

*} return maxValue; }*

### 8.2 Tìm kiếm nhị phân (Binary search):

Khái niệm: là một thuật toán tìm kiếm nhanh một phần tử trên một tập đã được sắp xếp sẵn và biết trước chiều xắp xếp.Tìm kiếm nhanh ở đây nghĩa là chương trình sẽ bắt đầu tìm từ vị trí giữa danh sách. Nếu giá trị tại vị trí đó bằng giá trị cần tìm thì kết thúc chương trình. Ngược lại, giá trị tại vị trí đó lớn hơn thì chương trình sẽ tìm từ bên trái vị trí đó trở đi, còn nhỏ hơn thì sẽ tìm từ bên phải.

VD:

*int BinarySearch(int arr[], int letf, int right, int x) {*

*mid=left + (right - 1)/2; tránh left và right là số lớn*

*if (arr[mid] == x)*

*return mid;*

*if (arr[mid] > x)*

*return binarySearch(arr, left, mid - 1, x);*

*return binarySearch(arr, mid + 1, right, x);*

*}*

*}*

#### 9. Các thuật toán sắp xếp:

### 9.1 Sắp xếp nổi bọt (Bubble sort):

Khái niệm: sắp xếp nổi bọt là sắp xếp dãy số bằng cách lặp lại công việc đổi chỗ 2 số liên tiếp nhau nếu chúng đứng sai thứ tự (số sau bé hơn số trước với trường hợp sắp xếp tăng dần) cho đến khi dãy số được sắp xếp.

VD:

*void Sort(int a[], int n){*

*int temp;*

*for(int i = 0; i < n - 1; i++){*

*for(int j = i + 1; j < n; j++){*

*if(a[i] > a[j]){*

*// Hoan vi 2 so a[i] va a[j]*

*temp = a[i];*

*a[i] = a[j];*

*a[j] = temp;*

*}*

*}*

*}*

*}*

### 9.2 Sắp xếp chèn (insrertion sort):

Khái niệm: là sắp xếp dãy số theo cách duyệt từng phần tử và chèn từng phần tử đó vào đúng vị trí trong mảng (dãy số từ đầu đến phần tử phía trước nó) đã sắp xếp sao cho dãy số trong mảng sắp đã xếp đó vẫn đảm bảo tính chất của một dãy số tăng dần.

VD:

*void insertionSort (int arr[], int n){*

*int i , key , j;*

*for (i = 1; i < n; i++) {*

*key = arr[i];*

*j = i-1;*

*while (j >= 0 && arr[j] > key) {*

*arr[j+1] = arr[j];*

*j = j-1; }*

*arr[j+1] = key; }*

*}*

# Chương II. Nguyên nhân và mục đích làm chương trình:

Nguyên nhân: việc quản lý thông tin nhân viên khá khó khăn nếu như chúng ta làm điều đó bằng cách thủ công. Nếu chỉ quản lý một vài nhân viên thì đơn giản , nhưng sẽ thế nào nếu phải quản lý hàng trăm, hàng ngàn nhân viên cùng lúc. Không những thế, trong quá trình điền thông tin cho từng nhân viên nếu xảy ra sai sót thì ta lại phải làm lại từ đầu. Trong một số trường hợp, chúng ta phải thêm hoặc bỏ thông tin một số nhân viên, điều này lại buộc ta phải làm mọi thứ lại từ đầu.

=> Để khắc phục một số hạn chế này thì nhóm đã tạo ra một chương trình đáp ứng các khuyết điểm của việc quản lý nhân viên thủ công.

# Chương III. Khái quát về chương trình:

* Thư viện được khai báo sử dụng trong chương trình:

*#include<stdio.h>*

*#include<conio.h>*

*#include<string.h> // làm việc với chuỗi*

*#include<stdlib.h> // cấp phát bộ nhớ*

* Khai báo và định nghĩa cấu trúc dùng trong chương trình:

*struct employee\_st{*

*char Name[30];*

*char Province[30];*

*int Day;*

*int Month;*

*int Year;*

*};*

*typedef struct employee\_st EL; // Đặt tên cấu trúc*

* Chương trình bao gồm một số chức năng đáp ứng các yêu cầu cơ bản sau:

1. Nhập thông tin nhân viên:

*void Input(EL \*employeeList , int &n ){*

*printf("\nNhap so luong nhan vien :");*

*scanf("%d", &n);*

*for(int i=0 ; i<n ; i++){*

*printf("\nNhap thong tin nhan vien %d" , i+1);*

*Nhap1( (employeeList+i) );*

*}*

*}*

1. Xuất thông tin nhân viên:

*void Output(EL \*employeeList , int n){*

*printf("\nThong tin cua nhan vien la:");*

*printf("\nSTT\tHoten\t\t Tinh\t\tNgaysinh");*

*for(int i=0 ; i<n ; i++){*

*printf("\n%d\t%-20s%-20s%-2d/%d/%d",i+1,(employeeList+i)->Name,(employeeList+i)->Province,(employeeList+i)->Day,(employeeList+i)->Month,(employeeList+i)->Year);*

*}*

*}*

1. Sắp xếp thông tin nhân viên:

*void Sort(EL \*employeeList , int n){*

*EL temp;*

*for(int i=0 ; i<n-1 ; i++){*

*for(int j=i+1 ; j<n ; j++)*

*if(strcmp((employeeList+i)->Province,(employeeList+j)->Province)<0){*

*temp =\*(employeeList+i);*

*\*(employeeList+i)=\*(employeeList+j);*

*\*(employeeList+j)=temp;*

*}*

*}*

*}*

-> Tại điều kiện *if*  nếu chuỗi tỉnh trước mà bé hơn chuỗi tỉnh sau thì sẽ đổi vị trí nhân viên của hai tỉnh đó với nhau , còn trái lại thì sẽ thực hiện tiếp vòng lặp

1. Thống kê nhân viên:

*void Cout(EL \*employeeList , int n){*

*int dem=1;*

*for(int i=0 ; i<n ; i++){*

*if(strcmp((employeeList+i)->Province,(employeeList+i+1)->Province)==0){*

*dem++;*

*}*

*else{*

*printf("\n%s co %d nhan vien!",(employeeList+i)->Province,dem);*

*dem==1;*

*}*

*}*

*}*

->Ở hàm này, nếu tại điều kiện *if* đúng thì biến *dem* tăng thêm 1, ngược lại nó sẽ xuất dem ra sau đó sẽ dán *dem* =1 và thực hiện tiếp vòng lặp.

1. Tìm kiếm nhân viên:

*void Search(EL \*employeeList , int n){*

*char province[30];*

*int dem=0;*

*fflush(stdin);*

*printf("\nNhap ten tinh can tim:");*

*gets(province);*

*for(int i=0;i<n;i++){*

*if (strcmp((employeeList+i)->Province,province)==0) dem++;*

*}*

*if (dem != 0){*

*printf("\nThong tin cua nhan vien thuoc tinh %s :",province);*

*printf("\nHoten\t\t Tinh\t\tNamSinh");*

*for(int i=0 ; i<n ; i++){*

*if (strcmp ((employeeList+i)->Province,province)==0){*

*Output1((employeeList+i));*

*}*

*}*

*}*

*else printf("Khong co nhan vien cua tinh %s!!!!!",province);*

*}*

-> Hàm này, trong vòng lặp *for* đầu tiên nó sẽ tìm xem trong danh sách có nhân viên nào thuộc tỉnh cần tìm không , nếu có thì tăng biến *dem*. Trong điều kiện *if*  nếu biến *dem* thỏa nó sẽ thực hiện việc hiện thị thông tin của nhân viên thuộc tỉnh đó , còn ngược lại nó sẽ thông báo không có nhân viên thuộc tỉnh đó.

1. Xóa nhân viên:

*void Delete(EL \*employeeList , int &n){*

*int t;*

*printf("\nNhap STT nhan vien muon xoa:");*

*scanf("%d",&t);*

*for(int i=t-1 ; i<n-1 ; i++){*

*\*(employeeList+i)=\*(employeeList+i+1);*

*}*

*--n;*

*}*

1. Thêm nhân viên:

*void Add(EL \*employeeList , int &n){*

*for(int i=n ; i>n-1 ; i--){*

*\*(employeeList+i)=\*(employeeList+i-1);*

*}*

*printf("\nNhap thong tin nhan vien muon them:");*

*Nhap1((employeeList+n-1));*

*++n;*

*}*

-> trước khi thêm ta sẽ lùi tất cả nhân viên kể cả vị trí chèn về phía sau một ô nhớ, sau đó sẽ thêm nhân viên vào vị trí cần thêm, hàm này thì nhân viên được thêm vào vị trí kề cuối.

1. Ghi thông tin nhân viên vào tệp nhị phân:

*void Recordfile(EL \*employeeList , int n){*

*FILE \* fp;*

*fp= fopen ("employee.dat","wb");*

*fprintf(fp," %-20s\n","DANH SACH THONG TIN NHAN VIEN");*

*fprintf(fp," %-5s%-20s%-20s%-10s","STT","Hoten","Tinh","Ngaysinh");*

*for(int i=0;i<n;i++){*

*fprintf(fp,"\n%-5d%-20s%-20s%-2d/%d/%d",i+1,(employeeList+i)->Name,(employeeList+i)->Province,(employeeList+i)->Day,(employeeList+i)->Month,(employeeList+i)->Year);*

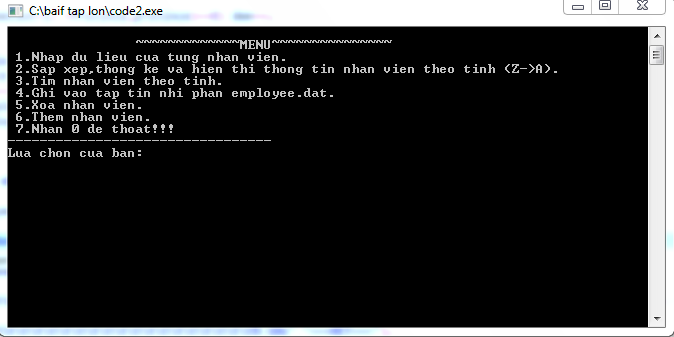
*}*

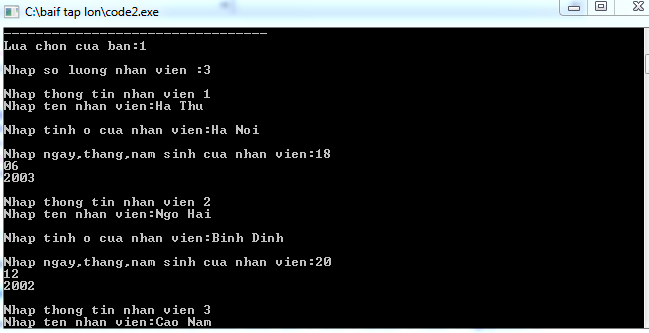
*fprintf(fp,"\n\ %-10s","~~~HET~~~");*

*fclose(fp);*

*}*

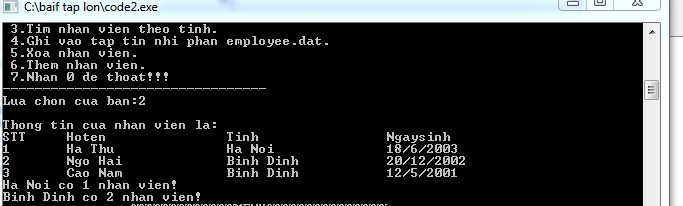
# Chương IV. Hướng dẫn sử dụng:

* Khi chạy chương trình, nó sẽ có dạng như sau:
* Để nhập thông tin nhân viên nhấn số 1 và Enter:

 • Bước này thì ta nhập số lượng nhân viên và nhập đầy đủ thông tin (tên, quê, ngày tháng, năm sinh) của từng nhân viên. Minh họa sau:

-> Ở đây, đã nhập số lượng nhân viên là 3 và thông tin nhân viên đã nhập như trên

• Nhấn Enter.

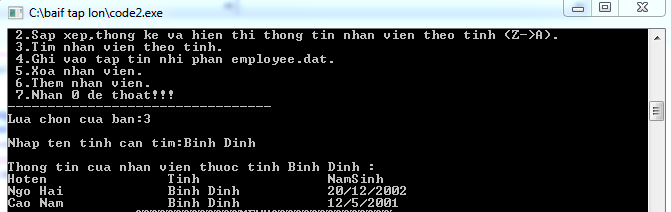
* Muốn xem thông tin của các nhân viên đã nhập thì chọn số 2:

*Lưu ý:* Các nhân viên đã được sắp xếp theo tỉnh (Z->A) rồi mới xuất ra, bên cạnh đó sẽ có thống kê xem tỉnh đó có bao nhiêu nhân viên .

-> Ví dụ trên thấy rằng Hà Nội có 1 nhân viên , Bình Định có 2 nhân viên.

* Nếu muốn tìm kiếm xem tỉnh nào đó có bao nhiêu nhân viên thì chọn số 3:

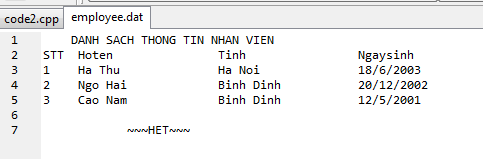
• Ta nhập tên tỉnh cần tìm.

 • Nhấn Enter.

• Sau đó thông tin các nhân viên thuộc tỉnh sẽ hiện ra đó:

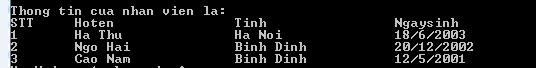
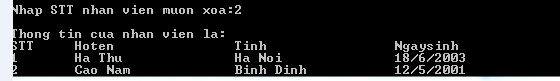
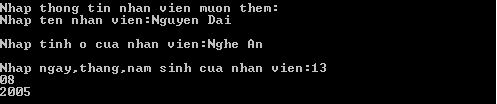
-> Tên tỉnh cần tìm ở đây là Bịnh Định và tiếp đó là thông tin của nhân viên có quê ở tỉnh Bình Định.

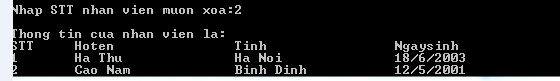
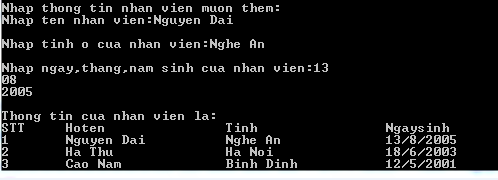
* Để dễ dàng xem danh sách nhân viên thì ta chỉ cần ghi các thông tin vào tệp nhị phân, nhấn số 4 để làm điều này:



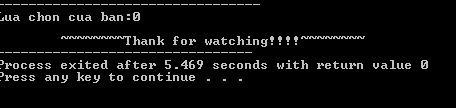
* Trong trường hợp muốn xóa hoặc thêm thông tin của nhân viên nào đó ta làm như sau:
* Nhấn số 5 để xóa nhân viên (chỉ xóa một mỗi lần 1 nhân viên):
* Nhập số thứ tự nhân viên muốn.
* Nhấn Enter.

Chẳng hạn muốn xóa nhân viên có só thứ tự: 2

* + Trước khi xóa:
  + Sau khi xóa:
* Nhấn số 6 để thêm nhân viên (chỉ có thể thêm mỗi lần 1 nhân viên):

* Nhập đầy đủ thông tin của nhân viên cần thêm.
* Nhấn Enter.
* Trước khi thêm:
* Sau khi thêm:

*Lưu ý***:** Sau khi thêm thì thông tin nhân viên sẽ tự động sắp xếp theo tên tỉnh (Z -> A).

* Để thoát khỏi chương trình:
* Nhấn số 0.
* Nhấn Enter.

# Tài liệu tham khảo:

[1] Lập trình không khó. 2021. *Thêm xóa phần tử trong mảng 1 chiều C/C++ hướng dẫn chi tiết*. [online] Available at: <https://nguyenvanhieu.vn/them-xoa-phan-tu-trong-mang-1-chieu/> [Accessed 20 April 2021].

[2] Lập trình không khó. 2021. *Đọc ghi file trong C | Hướng dẫn thao tác với file trong C*. [online] Available at: <https://nguyenvanhieu.vn/doc-ghi-file-trong-c/> [Accessed 20 April 2021].

[3] V1study.com. 2021. *Lập trình C: Hàm xử lý chuỗi (String)*. [online] Available at: <https://www.v1study.com/c-ham-xu-ly-chuoi-string-a56.html> [Accessed 20 April 2021].

**Bảng thống kê tỉ lệ làm việc của thành viên nhóm:**

|  |  |  |  |
| --- | --- | --- | --- |
| Loại công việc  Thành viên | Viết chương trình  (%) | Viết báo cáo  (%) | Ghi chú |
| Văn Công Hào | 70% | 47% | - Chương trình: hàm nhập 1 nhân viên; hàm nhập nhiều nhân viên; hàm tìm kiếm nhân viên; hàm xuất thông tin ra file; hàm thêm nhân viên; hàm xóa nhân viên  - Báo cáo: Phân tích chương trình và xây dựng đề tài |
| Trương Quang Duy | 30% | 53% | - Chương trình: hàm sắp xếp nhân viên; hàm thống kê nhân viên; hàm menu  - Báo cáo: cở lý thuyết |