**TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI**

**PHÂN HIỆU TẠI TP. HỒ CHÍ MINH**

**BỘ MÔN CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO**

**MÔN: KỸ THUẬT LẬP TRÌNH**

Giảng viên hướng dẫn: TRẦN PHONG NHÃ

Sinh viên thực hiện: NGUYỄN CÔNG MINH QUÂN

Lớp : CQ.65.CNTT

Khoá :65

Tp. Hồ Chí Minh, năm 2025

**TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI**

**PHÂN HIỆU TẠI TP. HỒ CHÍ MINH**

**BỘ MÔN CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO**

**MÔN: KỸ THUẬT LẬP TRÌNH**

Giảng viên hướng dẫn: TRẦN PHONG NHÃ

Sinh viên thực hiện: NGUYỄN CÔNG MINH QUÂN

Lớp : CQ.65.CNTT

Khoá :65

Tp. Hồ Chí Minh, năm 2025

**LỜI CẢM ƠN**

Lời nói đầu tiên, em xin gửi tới Quý Thầy Cô Bộ môn Công nghệ Thông tin

Trường Đại học Giao thông vận tải phân hiệu tại thành phố Hồ Chí Minh lời chúc sức

khỏe và lòng biết ơn sâu sắc.

Em xin chân thành cảm ơn quý thầy cô đã giúp đỡ tạo điều kiện để em hoàn

thành báo cáo. Đặc biệt em xin cảm ơn và thầy Trần Phong Nhã đã nhiệt tình giúp đỡ, hướng dẫn cho em kiến thức, định hướng và kỹ năng để có thể hoàn thành bài báo cáo này này.

Tuy đã cố gắng trong quá trình nghiên cứu tìm hiểu tuy nhiên do kiến thức còn

hạn chế nên vẫn còn tồn tại nhiều thiếu sót. Vì vậy em rất mong nhận được sự đóng góp

ý kiến của Quý thầy cô bộ môn để đề tài của em có thể hoàn thiện hơn.

Lời sau cùng, em xin gửi lời chúc tới Quý Thầy Cô Bộ môn Công nghệ thông tin

và hơn hết là thầy Trần Phong Nhã có thật nhiều sức khỏe, có nhiều thành công trong công việc. Em xin chân thành cảm ơn!

**NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN**

|  |
| --- |
| ***Tp. Hồ Chí Minh, ngày ….… tháng ….… năm ….…***  **Giáo viên hướng dẫn**  **TRẦN PHONG NHÃ** |

[LỜI CẢM ƠN 3](#_Toc20697)

[NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN 4](#_Toc25899)

[HÀM 7](#_Toc2315)

[1.1 Khái niệm 7](#_Toc25240)

[1.2 Quy tắc xây dựng hàm 8](#_Toc3928)

[1.3 Ví dụ 9](#_Toc530)

[CON TRỎ 10](#_Toc3944)

[2.1 Khái niệm 10](#_Toc1417)

[2.2 Quy tắc sử dụng 10](#_Toc7337)

[2.3 Quy tắc về kiểu giá trị 10](#_Toc12015)

[2.4 Hàm đối con trỏ 10](#_Toc25755)

[2.5 Khi nào sử dụng đối con trỏ 10](#_Toc25549)

[2.6 Ví dụ 11](#_Toc5250)

[CON TRỎ MẢNG 12](#_Toc2739)

[3.1 Con trỏ và mảng một chiều 12](#_Toc5540)

[3.1.1 Phép toán lấy địa chỉ 12](#_Toc22775)

[3.1.2 Tên mảng là một hằng địa chỉ 12](#_Toc7293)

[3.1.3 Nếu tham số thực là tên mảng a (một chiều) kiểu int (float double,…) thì đối pa tương ứng phải là con trỏ kiểu int (float, double,..) 12](#_Toc15262)

[3.2 Con trỏ và mảng nhiều chiều 12](#_Toc10896)

[3.2.1 Phép toán lấy địa chỉ 12](#_Toc7273)

[3.2.2 Phép cộng địa chỉ 12](#_Toc24510)

[3.3 Ví dụ 13](#_Toc3156)

[MẢNG CON TRỎ 14](#_Toc8003)

[4.1 Khái niệm 14](#_Toc26383)

[4.2 Ví dụ 14](#_Toc28117)

[CON TRỎ HÀM 15](#_Toc24372)

[5.1 Cách khai báo con trỏ hàm và mảng con trỏ 15](#_Toc22900)

[5.2 Tác dụng của con trỏ hàm 15](#_Toc8525)

[5.3 Đối con trỏ hàm 15](#_Toc229)

[5.4 Ví dụ 16](#_Toc27621)

[CẤP PHÁT ĐỘNG 17](#_Toc16377)

[6.1 Khái niệm 17](#_Toc11573)

[6.2 Ví dụ 18](#_Toc28422)

[XỬ LÝ TỆP 19](#_Toc31537)

[7.1 Khái niệm 19](#_Toc22341)

[7.2 Hàm dùng chung cho 2 loại tệp 19](#_Toc8446)

[7.2.1 Hàm fopen 19](#_Toc19177)

[7.2.2 Hàm fclose 20](#_Toc5399)

[7.2.3 Hàm fcloseall 20](#_Toc1839)

[7.2.4 Hàm fflush 20](#_Toc1457)

[7.2.5 Hàm feof 20](#_Toc30651)

[7.3 Ví dụ 21](#_Toc21787)

[KIỂU CẤU TRÚC 22](#_Toc28149)

[8.1 Khái niệm 22](#_Toc28198)

[8.2 Ví dụ 22](#_Toc13387)

[DANH SÁCH LIÊN KẾT 24](#_Toc22753)

[9.1 Khái niệm 24](#_Toc25728)

[9.2 Ví dụ 24](#_Toc31279)

[TÀI LIỆU THAM KHẢO 28](#_Toc24942)

**HÀM**

**1.1 Khái niệm**

Một chương trình viết theo ngôn ngữ C là một dãy các hàm trong đó có một hàm chính (hàm main). Thứ tự là bất kì nhưng luôn được thực hiện từ hàm main. Mỗi hàm làm phần khác nhau và chương trình sẽ giải quyết cả bài toán trọn vẹn. Một trong những ưu điểm của C là cho phép tổ chức và sử dụng các hàm đơn giản và hiệu quả.

Lưu ý:

* Mỗi hàm cần có tên, có thể đặt tên bất kì nhưng phải tuân theo quy tắc, nhưng cũng nên đặt tên hàm phù hợp theo đúng chức năng.
* Không cần khai báo nguyên mẫu của hàm, nhưng nên có vì nó cho phép chương trình biên dịch phát hiện lỗi khi gọi hàm.

**1.2 Quy tắc xây dựng hàm**

Hàm có thể xem là đơn vị độc lập của chương trình. Các hàm có vai trò ngang nhau, vì vậy không được phép xây dựng hàm khác bên trong hàm khác. Hàm được viết theo như sau:

* Dòng tiêu đề  
   Trong dòng đầu tiên chứa thông tin: kiểu hàm, tên, kiểu và tên mỗi đối
* Thân hàm  
  -Thân hàm là nội dung chính của hàm bắt đầu bằng dấu { và kết thúc bởi dấu }.  
  -Trong thân hàm có thể sử dụng biến khác nhưng đấy chỉ là biến cục bộ, chỉ có tác dụng trong thân hàm và không có liên hệ gì đến biến khác trong hàm khác của chương trình.  
  -Trong thân hàm có thể sử dụng lệnh return, có thể dùng ở nhiều chỗ khác nhau và cũng có thể không sử dụng. Dạng tổng quát là: return [ biểu thức ]. Giá trị của biểu thức trong lệnh return được gắn trong hàm.

**1.3 Ví dụ**

#include <stdio.h>

void greet() {

printf("Xin chao!\n");

}

int main() {

greet();

return 0;

}

**CON TRỎ**

**2.1 Khái niệm**

Con trỏ là một biến dùng để chứa địa chỉ, có nhiều loại địa chỉ nên có nhiều con trỏ tương ứng. Con trỏ kiểu int dùng để chứa kiểu int, tương tự như kiểu float và double,… Cũng như với bất kì biến nào ta cần phải khai báo con trỏ trước khi sử dụng, khai báo như sau: type \*tên\_con\_trỏ;

**2.2 Quy tắc sử dụng**

Có thể sử dụng tên con trỏ hoặc dạng khai báo của nó trong biểu thức. VD với con trỏ px, ta có thể dùng cách viết: px ( tên con trỏ), và \*px (dạng khai báo của con trỏ).

* Sử dụng tên con trỏ:   
  Con trỏ cũng là một biến nên khi nó xuất hiện trong biểu thức thì giá trị của nó cũng chỉ được sử dụng trong biểu thức này. Cần lưu ý: Giá trị của con trỏ cũng là địa chỉ của biến dó, khui con trỏ đứng bên trái của một toán tử thì giá trị của biểu thức bên phải được gắn cho con trỏ.
* Sử dụng dạng khai báo của con trỏ:  
  Khi biết được địa chỉ của một biến thì ta vừa có thể sử dụng giá trị của nó và còn có thể gán cho nó một giá trị mới làm thay đổi nội dung.

**2.3 Quy tắc về kiểu giá trị**

Mọi thành phần của cùng một khai báo (biến, phần tử mảng, hàm, con trỏ) khi xuất hiện trong biểu thức đều cho cùng một kiểu giá trị.

**2.4 Hàm đối con trỏ**

Nếu đối của hàm là con trỏ kiểu int (float, double,…) thì tham số tương ứng phải là địa chỉ của biến hoặc địa chỉ của phần tử mảng kiểu int (float, double,…). Khi đó địa chỉ của biến được truyền cho con trỏ tương ứng. Do đã biết được địa chỉ của biến, nên ta có thể gắn cho nó các giá trị mới bằng cách sử dụng các câu lệnh thích hợp trong thân hàm.

**2.5 Khi nào sử dụng đối con trỏ**

Trong số các đối của hàm có thể chia ra hai loại. Loại thứ nhất gồm các đối để chứa các giá trị đã biết, gọi là đối vào. Loại thứ hai gồm các đối dùng để chứa các kết quả mới nhận được, gọi là đối ra. Vậy sử dụng hàm đối con trỏ thì các đối ra phải là con trỏ.

**2.6 Ví dụ**

#include <stdio.h>;

#include <math.h>;

int ptb2(float a, float b, float c, float \*x1, float \*x2);

main()

{

int s,ch;

float a,b,c,x1,x2;

printf(“\n Vao a,b,c”);

scanf(“%f%f%f”,&a,&b,&c);

s = ptb2(a,b,c,&x1,&x2);

if (s == 0)

printf(&quot;\n a = 0 &quot;);

else

if (s == -1)

printf(&quot;\n delta &lt; 0 &quot;);

else

printf(“\n x1 = %0.2f x2 = %0.2”,x1,x2 );

}

/\* Hàm giải phương trình bậc hai \*/

int ptb2(float a, float b, float c, float \*x1, float \*x2)

{

float delta;

if (a == 0) return 0;

delta = b\*b-4\*a\*c;

if (delta<0) return -1;

\*x1 = (-b - sqrt(delta))/(2\*a);

\*x2 = (-b + sqrt(delta))/(2\*a);

return (1);

}

**CON TRỎ MẢNG**

**3.1 Con trỏ và mảng một chiều**

**3.1.1 Phép toán lấy địa chỉ**

Ta khai báo double b[4] thì phép toán: &b[i] với i từ khoảng [0,3] cho địa chỉ của phần từ b[i].

**3.1.2 Tên mảng là một hằng địa chỉ**

**Khi ta có mảng float a[10] và khi biết được địa chỉ của một phần tử nào đó của mảng** a, thì có thể suy ra địa chỉ của các phần từ khác. Tên mảng là một hằng địa chỉ, cũng chính là địa chỉ của phần tử đầu tiên của mảng. Trong mọi ngữ cảnh:

a tương đương với &a[0]

a+i tương đương với &a[i]

\*(a+i) tương đương với a[i]

**3.1.3 Nếu tham số thực là tên mảng a** (một chiều) kiểu int (float double,…) thì đối pa tương ứng phải là con trỏ kiểu int (float, double,..)

**3.2 Con trỏ và mảng nhiều chiều**

Xử lí mảng nhiều chiều phức tạp hơn so với mảng một chiều, không phải quy tắc nào của mảng một chiều cũng đúng với mảng nhiều chiều

**3.2.1 Phép toán lấy địa chỉ**

Không dùng được với các phần tử của mảng nhiều chiều, trong nhiều trường hợp câu lệnh: &a[i][j] là không hợp lệ và gây lỗi

**3.2.2 Phép cộng địa chỉ**

Tên mảng a biểu thị địa chỉ đầu tiên của mảng. Nhưng phép cộng phải hiểu như sau, với khai báo float a[2][3] thì a là mảng mỗi phàn tử của nó là một dẫy 3 số thực. Vì vậy:

a trỏ tới đầu hàng thứ nhất ( phần tử a[0][0])

a+1 trỏ tới đầu hàng thứ hai ( phần tử a[1][0])

...

**3.3 Ví dụ**

#include <stdio.h>

#include <stdlib.h>

int main(){

int n = 5;

int a[5] = {3, 8, 4, 2, 9};

printf("Gia tri cua a : %d\n", a);

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

printf("Dia chi cua a[%d] : %d\n", i, a + i);

}

printf("Mang a : ");

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

printf("%d ", \*(a + i));

}

return 0;

}

**MẢNG CON TRỎ**

**4.1 Khái niệm**

Mảng con trỏ là sự mở rộng khái niệm con trỏ. Mảng con trỏ là một mảng mà mỗi phần tử của nó có thể chứa địa chỉ nào đó. Mảng con trỏ có nhiều kiểu: Mỗi phần tử mảng con trỏ kiểu int sẽ chứa được địa chỉ kiểu int. Tương tự các con trỏ kiểu float, char,… Mảng con trỏ được khai báo theo mẫu: type \*namearr[N]; với type là int, float, double, char,… namearr là tên mảng, N là hằng số nguyên xác định độ lớn của mảng.

Lưu ý

* Mảng con trỏ không thể dùng để lưu trữ số liệu, tuy nhiên mảng con trỏ cho phép sử dụng mảng khác để lưu trữ số liệu một cách có hiểu quả hơn bằng cách: Chia mảng thành các phần và ghi nhớ địa chỉ đầu của mỗi phần vào một phần tử của mảng con trỏ.
* Trước khi sử dụng mảng con trỏ cần gán cho mỗi phần tử của nó một giá trị, giá trị này phải là địa chỉ của một biến hoặc của một phần tử mảng. Các phần tử của mảng con trỏ kiểu char có thể được khởi đầu bằng các xâu kí tự

**4.2 Ví dụ**

#include <stdio.h>

int main() {

const char \*fruit[] = {"Apple", "Banana", "Cherry", "Mango"};

int n = sizeof(fruit) / sizeof(fruit[0]);

printf("Danh sách trái cây:\n");

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

printf("%s\n", fruit[i]);

}

return 0;

}

**CON TRỎ HÀM**

**5.1 Cách khai báo con trỏ hàm và mảng con trỏ**

float (\*f)(float), (\*mf[50])(int);

Với f là con trỏ hàm kiểu float có đối float, mf là mảng con trỏ hàm kiểu float có đối int (mảng có 50 phần tử).

**5.2 Tác dụng của con trỏ hàm**

Dùng để chứa địa chỉ của hàm, cần thực hiện phép gán tên hàm cho con trỏ hàm, để phép gán có nghĩa thì kiểu hàm và kiểu con trỏ phải tương thích, sau phép gán, có thể dùng tên con trỏ hàm thay cho tên hàm.

**5.3 Đối con trỏ hàm**

C cho thiết kế các hàm mà tham số thực trong lời gọi tới nó lại là tên của hàm khác, khi đó tham số hình thức tương ứng phải là con trỏ hàm.

Cách dùng con trỏ hàm trong thân hàm: Với đối được khai báo double (\*f)(double, int) thì trong thân hàm có thể dùng cách viết sau để xác định giá trị của hàm: f(x,m) (f)(x,m) (\*f)(x,m) với x là biến double, m là biến int.

**5.4 Ví dụ**

#include <stdio.h>

//Khai báo hàm

int add(int a, int b)

{

return a + b;

}

int main(void)

{

int m = 2, n = 5, sum;

int (\*fp)(int, int);

fp = add;

sum = (\*fp)(m, n);

printf("sum = %d", sum);

return 0;

}

**CẤP PHÁT ĐỘNG**

**6.1 Khái niệm**

Việc khai báo biến trước chính là cấp phát tĩnh bộ nhớ. Nhưng không phải lúc nào số lượng và kích thước của các biến cũng xác định được ngay khi biên dịch, nó có thể phụ thuộc vào các thông tin người dùng cung cấp khi thực hiện chương trình. Trong trường hợp này, bộ nhớ cần được phát động. Để cấp phát động, sử dụng hàm malloc trong thư viện stdlib.h: void \*malloc(size\_t size) trong đó:

-Size là kích thước vùng nhớ cần cấp.

-Size\_t là kiểu dữ liệu định sẵn trong thư viện stdlib.h.

-Hàm trả về con trỏ kiểu void chỉ đến ô nhớ đầu của vùng nhớ được cấp. Nếu bộ nhớ không đủ, giá trị con trỏ là NULL.

Để giải phóng vùng nhớ được trỏ bởi con trỏ p, sử dụng hàm free trong stdlib.h: void free(void\*p).

Lưu ý:

* Giá trị p = NULL => Hàm free sẽ không làm gì.

**6.2 Ví dụ**

#include <stdio.h>

#include <stdlib.h>

int main()

{

int n, i, \*ptr, sum = 0;

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

scanf("%d", &n);

ptr = (int \*)malloc(n \* sizeof(int));

if (ptr == NULL)

{

printf("Khong cap phat duoc bo nho.");

exit(0);

}

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

{

printf("Nhap gia tri %d: ", i+1);

scanf("%d", ptr + i);

printf("\n");

sum += \*(ptr + i);

}

printf("Tong = %d", sum);

free(ptr);

return 0;

}

**XỬ LÝ TỆP**

**7.1 Khái niệm**

Các thao tác têp được thực hiện nhờ hàm thư viện, chia thành hai nhóm: cấp 1 và cấp 2

Các hàm cấp 1 có đặc trưng cơ bản:

* Không có xuất nhập riêng cho từng dữ liệu mà chỉ có đọc ghi một dẫy các byte.
* Mỗi tệp có một thẻ (handle) và các hàm cấp 1 làm việc với tệp thông qua số hiệu.

Các hàm cấp 2 dược xây dựng từ hàm cấp 1 nên dễ sử dụng và có nhiều khả năng:

* Có dịch vụ truy xuất cho từng kiểu dữ liệu.
* C tự động cung cấp một vùng đệm, mỗi lần đọc ghi thì thường tiến hành trên vùng đệm. Khi ghi số nguyên thì đưa vào vùng đệm và khi nào đầy thì vùng đệm mới được đẩy lên đĩa. Khi đọc, thông tin được lấy từ vùng đệm, và khi vùng đệm đã trống thì máy lấy dữ liệu từ đĩa chứa vào vùng đệm. Việc sử dụng vùng đệm sẽ giảm số lần nhập xuất trên đĩa và nâng cao tốc độ làm việc. Tuy vậy trong một số trường hợp ta phải nhờ tới tác nghiệp vét vùng đệm để đề phòng mất thông tin.
* Làm việc với tệp thông qua biến con trỏ tệp.

**7.2 Hàm dùng chung cho 2 loại tệp**

**7.2.1 Hàm fopen**

Dạng hàm: FILE \*fopen(const chả \*tên\_tệp, const char \*kiểu);

Công dụng: Dùng mở tệp, nếu thành công hàm cho con trỏ kiểu FILE tương ứng với tệp vừa mở. Hàm cấp 2 làm việc với tệp thông qua con trỏ này, nếu có lỗi hàm trả về giá trị NULL.

Các đối:

* Tên tệp.
* Kiểu truy nhập.  
  Kiểu có các giá trị sau:  
  ”r” “rt” - mở tệp để đọc theo kiểu văn bản, tệp cần tồn tại nếu không sẽ có lỗi.  
  ”w” “wt” - mở tệp mới để ghi theo kiểu văn bản, tệp đã tồn tại thì bị xoá.  
  …

Lưu ý: Trong các kiểu đọc/ghi, cần làm sạch vùng đệm trước khi từ đọc sang ghi và ngược lại. Có thể dùng hàm fflush và hàm di chuyển dể giải quyết.

**7.2.2 Hàm fclose**

Dạng hàm: int fclose(FILE \*fp);

Công dụng: Dùng đóng tệp gồm: Đẩy dữ liệu còn trong vùng đệm lên đĩa, xoá vùng đệm và giải phóng biến fp để dùng cho tệp khác, nếu thành công hàm cho giá trị 0, trái lại hàm cho EOF.

Đối: fp là con trỏ tương ứng với tệp cần đóng.

**7.2.3 Hàm fcloseall**

Dạng hàm: int fclose(void);

Công dụng: Đóng tất cả hàm đang mở.

**7.2.4 Hàm fflush**

Dạng hàm: int fflush(void);

Công dụng: Làm sạch vùng đệm các tệp đang mở.

**7.2.5 Hàm feof**

Dạng hàm: int feof(FILE\*fp);

Công dụng: Hàm dể kiểm tra cuối tệp, cho giá trị khác 0 nếu gặp cuối tệp khi đọc, ngược lại bằng 0.

Đối: fp là con trỏ tệp.

**7.3 Ví dụ**

#include <stdio.h>

#include <math.h>

int prime(int n){

if(n < 2) return 0;

for(int i = 2; i <= sqrt(n); i++){

if(n % i == 0){

return 0;

}

}

return 1;

}

int main(){

FILE \*f;

f = fopen("prime.txt", "w");

int dem = 0;

for(int i = 1; i <= 200; i++){

if(prime(i)){

fprintf(f, "%d ", i);

++dem;

if(dem % 10 == 0){

fprintf(f, "\n");

}

}

}

fclose(f);

return 0;

}

**KIỂU CẤU TRÚC**

**8.1 Khái niệm**

Cấu trúc có thể xem như sự mở rộng của các khái niệm biến và mảng, cho phép lưu trữ và xử lý các dạng thông tin phức tạp hơn. Cấu trúc là một tập hợp các biến, các mảng và được biểu thị bởi một tên duy nhất.

Để định nghĩa một cấu trúc, sử dụng từ khóa struct theo mẫu sau:

struct tên\_cấu\_trúc

{

Khai báo các thành phần

};

Lưu ý: thành phần của một cấu trúc có thể là một biến kiểu cơ bản hoặc có thể

là một cấu trúc khác.

Để định nghĩa một kiểu cấu trúc, sử dụng từ khóa typedef theo mẫu sau:

typedef struct

{

Khai báo các thành phần

} kiểu\_cấu\_trúc;

Khi xây dựng được kiểu\_cấu\_trúc, nó có thể được dùng để khai báo các biến cấu trúc.

Thành phần cơ bản của một cấu trúc là biến và mảng, một quy tắc cần ghi nhớ là việc xử lý một cấu trúc bao giờ cũng phải được thực hiện thông qua các thành phần cơ bản của nó.

Để truy nhập tới các thành phần của cấu trúc cần thông qua tên của biến cấu

trúc theo mẫu sau:

biến\_cấu\_trúc.tên\_thành\_phần

**8.2 Ví dụ**

#include <stdio.h>

#include<stdlib.h>

typedef struct

{

char ht[20];

float diem;

} kieu\_sv;

int main()

{

int i, n;

float t;

kieu\_sv \*sv;

printf(“So sinh vien= “);

scanf(“%d”,&n);

sv= (kieu\_sv\*)malloc(n\*sizeof(kieu\_sv));

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

{

fflush(stdin);

printf(“SV thu %d= \n”, i+1);

printf(“Ho ten: “);

gets(sv[i].ht);

printf(“Diem: “);

scanf(“%f”, &t);

sv[i].diem= t;

}

puts(“\nDanh sach sinh vien dat\n”);

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

{

if(sv[i].diem>=5)

printf(“%s%6.2f\n”,sv[i].ht, sv[i].diem);

}

free(sv);

}

**DANH SÁCH LIÊN KẾT**

**9.1 Khái niệm**

Cấu trúc tự trỏ được dùng để xây dựng danh sách liên kết (móc nối), là một nhóm các cấu trúc có tính chất sau:

--Biết địa chỉ cấu trúc đầu đang được lưu trữ trong một con trỏ nào đó

Trong mỗi cấu trúc (trừ cấu trúc cuối) chứa địa chỉ của cấu trúc tiếp theo của danh sách.

-Cấu trúc cuối chứa hằng NULL.

-Danh sách có 3 tính chất trên gọi là danh sách móc nối theo chiều thuận. Với danh sách này, có thể lần lượt truy nhập từ cấu trúc đầu cuối theo chiềutừ trên xuống dưới.

Tương tự, danh sách liên kết theo chiều ngược cũng có 3 tính chất trên nhưng theo chiều ngược lai:

-Biết địa chỉ cấu trúc cuối.

-Trong mỗi cấu trúc (trừ cấu trúc đầu) chứa địa chỉ của cấu trúc trước.

-Cấu trúc đầu chứa hằng NULL.

-Với danh sách này, có thể lần lượt truy nhập từ cấu trúc cuối đầu theo chiều từ dưới lên trên.

Ngoài ra có thể xây dựng các danh sách mà mỗi phần tử chứa hai địa chỉ: địa chỉ cấu trúc trước sau. Với loại danh sách này, có thể truy nhập từ trên xuống dưới theo chiều thuận hoặc từ dưới lên trên theo chiều ngược.

**9.2 Ví dụ**

#include <stdio.h>

#include <stdlib.h>

#include <conio.h>

#include <string.h>

typedef struct sv

{

char ht[25];

float diem;

struct sv \*tiep;

} ksv;

main()

{

float t;

char ht[25];

ksv \*pdau,\*p,\*p1;

pdau=NULL;

// nhap

while(1)

{

fflush(stdin);

printf("\nHo ten:");

gets(ht);

if (ht[0]==0) break;

if (pdau==NULL)

{

pdau=(ksv\*)malloc(sizeof(ksv));

p=pdau;

}

else

{

p->tiep=(ksv\*)malloc(sizeof(ksv));

p=p->tiep;

}

strcpy(p->ht,ht);

printf("\nDiem: ");

scanf("%f",&t);

p->diem= t;

p->tiep = NULL;

}

// in danh sach lien ket

p=pdau;

while(p!=NULL)

{

printf("\nHo ten:%s", (\*p).ht);

printf("\nDiem:%6.2f";, (\*p).diem);

p = p->tiep;

}

// bsung cuoi ds

printf("\nBo sung vao cuoi ");

if(pdau==NULL) // ds rong

{

pdau=(sv\*)malloc(sizeof(ksv));

p=pdau;

}

else

{

p=pdau;

while(p->tiep!=NULL)

p=p->tiep;

// cap phat dong, nhap

p->tiep=(ksv\*)malloc(sizeof(ksv));

p=p->tiep;

}

// nhap bsung

printf("\nHo ten: ");

gets(p->ht);

printf("\nDiem: ");

scanf("%f",&t);

p->diem=t;

p->tiep=NULL;

// in dslk

p=pdau;

while(p!=NULL)

printf("\nHo ten:%s", ",(\*p).ht);

printf("\nDiem:%6.2f", ",(\*p).diem);

p = p->tiep;

}

}

**TÀI LIỆU THAM KHẢO**

1. Giáo trình môn Kỹ thuật lập trình trong classroom