CHƯƠNG 1 : CáC KHÁI NIỆM CƠ BẢN CỦA NGÔN NGỮ C

1.1/ Tập ký tư hợp lệ dùng trong ngôn ngữ C

- Các chữ cái : A, B, C ..., 2, a,n,c,...z (26 chữ cái thường)
- Các chữ số : 0,1,..., 9.
- Ký tự gạch nối _ (chú ý phân biệt dấu -).
- Dấu cách (space) : dùng để phân biệt các từ :
  Ví dụ : lop Học(7 kí tự) - LopHoc(6 kí tự).

1.2/ Tên (định danh) : là 1 dãy kí tự bắt đầu bằng chữ hoặc ký tự gạch dưới, theo sau là chữ cái, chữ số hoặc ký tự gạch nối (-).

- Tên : dùng làm tên hằnp, tên biến , nhãn , tên hàm....
  Ví dụ : Tên đúng : _abc, Delta_1, BETA.
  Tên sai : 1xyz (vì bắt đầu là 1 chữ số)
  A#B (vì có dâu #)
  Delta (vì có khoảng trống) , X-1 (vì sử dụng dấu gạch ngang).

* Chú ý :

+ Tên : chữ hoa và chữ thường được xem là khác nhau ((# pascal)
+ Thông thường :
  . Đặt chữ hoa cho các hằng, chữ thường cho các đại lượng còn lại(biến, hàm..).
  . Nên đặt 1 cách gợi nhớ (8 kí tự đầu là có nghĩa và tuỳ thuộc chương trình).

1.3/ Từ khoá : là từ dành riêng cho ngôn ngữ. Tên biến, hằng, hàm ...không được trùng với từ khoá, luôn luôn viết bằng chữ thường. Các từ khoá trong C gồm : Break, char, continue, case, do, double, default, else, float, for, goto, int,if, long, return, struct, switch, unsigned, while, typedef, union voi, volatile,..

1.4/ Các kiểu dữ liệu cơ bản trong C : 4 kiểu : char, Int, float, double.

- Kiểu char (1 byte) : biểu diễn 1 ký tự thuộc ASCII (thực chất là số nguyên từ 0 đến 255)
  Ví dụ : Ký tự ASCII
  0 048
  A 065
  a 097
- Kiểu Int : 3 loại : Int, long Int (long) và unsigned Int (unsigned).
- Kiểu Float : biểu diễn các số thực độ chính xác định.
- Kiểu double : biểu diễn các số thực độ chính xác kép.

Stt	Kiểu Phạm vi Kích thước
1	Char 0..255 1 byte
2	Int -32768..32767 2 bytes
3	Long -2147483648..2147484647 4bytes
4	Unsigned 0..65535 2 bytes
5	Float 3.4e - 38..3.4e + 38 4 bytes
6	double 1.7e - 308 .. 1.7e + 308 8 bytes

- Kiểu void: Kiểu không giá trị, củ dùng để biểu diễn kết quả hàm cũng như nội dung củ pointer. Kiểu này sẽ nói chi tiết ở các phần liên quan.

1.5/ Biến và mảng :

a/ Biến : Biến đại lượng thay đổi; mỗi biến có 1 tên và địa chỉ vùng nhờ danh riêng cho nó.
Khai báo biến : Cú pháp < Kiểu dữ liệu > < Danh sách các biến >;
Ví dụ : Int i,j ;
long cucdai;

double tongsothue;
Int a,b = 20; float e = 35.1; x=30.5;

b/ Mảng: là tập hợp các phần tử có cùng 1 kiểu và chung 1 tên.

Khai báo :

Ví dụ : Int Mang1[10];
Float Bang [10][10];

- Mảng một chiều : là một dãy các ký tự phần tử tuần tự trong bộ nhớ, mỗi một phần tử chiếm một số byte tương ứng với kiểu của nó.
- Mảng nhiều chiều : Gồm các phần tử sắp liên tiếp từ hàng này sang hàng kia. Các chỉ số được đánh số từ 0 trở đi.

Ví dụ :

- Mãng 1[0] ..Mãng1[9]
- Bang [0][0] Bang [0][1]..Bang [0][9].
  Bang[][] Bang[2][0]..Bang[1][9]
  .....
  Bang[9][0].. Bang[9][9]

* Chú ý : &Mang1[3] đúng nhưng &Bang[2][5]sai (Đúng đối với 1 chiều và sai đối với nhiều chiều)

1.6 / Hằng : Đại lượng không thay đổi

a/ Hằng nguyên (Int): có giá trị từ -32768 đến 32767

- Có thể viết theo hệ 16 bằng cách thêm tiền tố Ox hoặc theo cơ số 8 bằng cách thêm tiền tố O (Octal = bát phân).

* Ví dụ : O306 viết theo cơ số 8 : Giá trị = 6 * 8 0 + 3* 8 * 8 = 198 trong hệ 10.
O345 = 3*8*8 + 4*8 + 5 = 229
Ox147 = 1*16*16 + 4*16 +7 =327 trong hệ 10.
OxAa= 10*16+13=173
- Lý do a hoặc A =10
b hoặc B =11
c hoặc C =12
d hoặc D =13
e hoặc E = 14
f hoặc F = 15

b/Hằng long (long Int : nguyên lớn) : giống như hằng nguyên, chỉ khác thêm L hoặc l ở đầu.

* Ví dụ : 180L, 123456789l (Hằng nguyên là giá trị vượt quá số nguyên là hằng nguyên lớn (long)

c/Hằng thực (float và double) : Có 2 cách viết

- Cách 1 : (dạng thập phân) Số gồm : phần nguyên, dấu chấm thập phân và phần phân.

* Ví dụ : 214.35 , - 234.34.

- Cách 2 : Viết theo dạng khoa học

* Ví dụ : 1.543e7 = 15430000
123.456E-4 = 0.123456 (123.456/105)

d/Hằng ký tự : Viết trong 2 dấu nháy đơn. Giá trị của hằng chính là mã ASCII của chữ.

* Ví dụ : 'A' = 65;
'd' = 100,
'9 ' - '0 ' = 57 - 48 = 9

- Hằng ký tự còn có thể viết \X1X2X3 , \x1x2x3 : trong đó x1,x2,x3 là số nguyên hệ 8.

*Ví dụ : chữ a mã hệ 10 là 97 đổi ra hệ 8 là O141 => \141='a';
\ 101='A';
\ 142 ='b'

* Một số hằng đặc biệt được viết theo qui ước như sau :

Viết	Ký tự Diễn giải
'	\ " ' dấu nháy đơn

' \" ' " dấu nháy kép
' \\ ' \ dấu chéo ngược
'\n ' \n ký tự xuống dòng
' \0 ' \0 ký tự rỗng (null)
Chú ý : - Phân biệt ký tự '0 ' và ' \0 '. Hằng ' 0 ' cùng với chữ số 0 có mã = 48
- Hằng '\0 ' cùng với ký tự \0 (null) có mã 0.
- Hằng ký tự thực sự là số nguyên => có thể dùng số nguyên hệ 10 đê biểu diễn ký tự .
* Ví dụ : printf(" %c%c", 65,66) sẽ in ra AB.
e/Hằng xâu ký tự : đặt trong dấu nháy kép (" "). Hằng được lưu trữ trong 1 mãng ký tự mà ký tự cuối cùng là rỗng
(null) ' \0 '.
Ví dụ : "Lơp Hoc"
- Hằng có thể được đinh nghĩa đối với toán tử define .
+ Cú pháp : # define < tên hằng > < giá trị>
Trong chương trình mọi biến max đều được thay đổi giá trị 100.
Ví dụ : # define MAX 100
Define pi 3.141593
1.7/ Phép toán :
+ Phép toán số học gồm : +,-,*, / (Phép chia lấy phần nguyên), % (phép chia lấy phần dư).
+ Phép toán quan hệ : <, >, <=,>=, = =, ! = (khác).
+ Phép toán logic : || (hoặc) , && (và) ! (not), #0 hay =1 : True(đúng) ; =0 : Falsse (sai)
+ Phép toán tăng giảm : ++ cộng thêm 1 vào toán hạng.
* Ví dụ : Int n=10; n++;=> n=11 <=> n=n+1;
Chú ý : - n++ : giá trị n được lấy trước khi tăng n.
- ++n : giá trị n được lấy sau khi tăng n
- tương tự n-- , --n ;
+ Toán tử thao tác bit : Không áp dụng cho kiểu float hoặc double.
& : phép hội các bít (và)
| : phép tuyển các bit (hoặc)
^ : phép tuyển các bit loại trừ
<< : phép dịch trái
>> : phép dịch phải.
: phép lấy phần bù.
Ví dụ : 105 & 7 = 1 /* 0111 1001 & 0000 0111 = 0000 0001 */
105 | 17 = 127 /* 0111 1001 | 0000 0111 = 0111.1111 */
0x60 = 0x96 /* 0110 1001 = 1001 0110 */
+ Toán tử chuyển đổi kiểu : ta có thể dùng toánt ử chuyển kiểu để chuyển 1 kiểu bất kỳ sang kiểu mong muốn bằng cách dùng toán tử sắc thái (cast) theo quy tắc sau :
ép Kiểu (type cast) : (kiểu) Biến Kiểu mong muốn
* Ví dụ : int i = 10
----ă (float) i => 10.0
- Chú ý : + Một số kiểu float khi chuyển sang kiểu Int sẽ bị chặt cụt phần thập phân.
+ Một số kiểu long khi chuyển sang kiểu Int sẽ cắt bỏ vài chữ số.
* Ví dụ : n = 2560.70 thì (int)n = 2560
+ Toán tử gán :
- Cú pháp : < biến> = < biểu thức>
* Ví dụ : c = a + b ; d= t + 3 ;
i= i+2 (Viết gọn i+=2;)
i= i*2 (i*=2;)
x = x >> 1 (x >> = 1;)
Chú ý : Các phép toán trong C có độ ưu tiên khác nhau và quy tắc kết hợp khác nhau => Bảng liệt kê các phép toán theo thứ tự ưu tiên từ trên xuống dưới, các phép toán trên dòng có thứ tự

như nhau.
Phép toán Trình tự kết hợp
(),[ ], ă trái qua phải
|, dấu ngã, &*, - -, + + , (type) size of phải qua trái
*,/, % trái qua phải
+, - trái qua phải
<< ,>> trái qua phải
<,<= ,> ,>= trái qua phải
& trái qua phải
| trái qua phải
&& trái qua phải
|| trái qua phải
? phải qua trái
= =, !=, +=, -= phải qua trái

1.8/ Biểu thức : được xây dựng bằng các toán tử , toán hạng là các hằng, biến, hàm....
- Biểu thức gán : Ví dụ : A = B =C =5 => A=5, B = 5, C = 5.
- Biểu thức điều kiện có dạng : B1?E1 : E2 : Nếu B 1 đúng giá trị biểu thức = E1. ngược lại E2.
* Ví dụ : S=x>y ? x:y cho giá trị lớn nhất của x và y.

1.9/ Cấu trúc tổng quát của chương trình viết bằng ngôn ngữ C :

```
#include < Thuvien.h> những khai báo, những chỉ thị tiền xử lý.
#define ....
/* Các khai báo kiểu dữ liệu, hằng */
Type of....
{ Các biến toàn cục, biến ngoài}
prototype { khai báo tiêu đề hàm}
main ()
    { x1,x2,.....xn}
```

* Ví dụ : Viết chương trình số lớn nhất cho trước a, b, c

```
/* Chương trình tìm số lớn nhất trong 3 số*/
# include < stdio.h>
# Include < conio.h>
void main (void)
{ int n1,n2, n3, nmax ;
 do
 {
/* đọc 3 số từ bàn phím*/
   printf(" nhập số thứ nhất : "); scanf( " %d", &n1);
   printf(" nhập số thứ hai : "); scanf( " %d", &n2);
   printf(" nhập số thứ ba : "); scanf( " %d", &n3);
/* tìm số lớn nhất */
   nmax = n1>n2 ? n1:n2;
   nmax = nmax > n3 ? nmax : n3;
/* In ra kết quả */
    printf ( " số lớn nhất trong 3 số %d%d%d là : %d \ n ", n1,n2,n3 ,nmax);
 }
   printf ( " ấn ESC để kết thúc );
   while ( getch ()! = 27 );
}
```

2/ CÁC LỆNH XUẤT NHẬP CHUẩN:

2.1/ Hàm Printf

- Printf (" formated string ", <biểu thức>);

Biểu thức có thể là : const (hằng), var (biến), function (hàm).

* Ví dụ : int Siso= 30;
Printf (" In sĩ số lớp học là %d, Siso);
a/ Các ký tự điều khiển :
\n : sang dòng mới
\b : lùi lại 1 tab.
\f : sang trang mới
\t : dấu tab
\' : In ra dấu '
\" : in ra dấu "
\ \ : in ra dấu \
b/ ký tự chuyển dạng :
Ký tự chuyển dạng Kiểu của đối Mô tả
c char đối là ký tự
d/di int đối là số nguyên
ld /li long đối là số nguyên dài
f float hoặc doubl đối là số thực
s xâu ký tự(chuỗi) đối là chuỗi
u int số nguyên hệ 10 không dấu
O int số nguyên hệ 8 không dấu
lo long số hệ 8 không dấu
x int số hệ 16 không dấu
lx long số hệ 16 không dấu
g float hay double không in ra các số không vô nghĩa
c float hoặc double đối trong dạng thập phân
Độ rộng dành cho biến , trước in ra. Lưu ý ra màn hình printf(stdpm, "\n sĩ số ..")
2.2/ Hàm scanf :
- scanf (" formated string ", các địa chỉ biến);
* Ví dụ :
int a ;
float x,y;
char cr[6], ct[6];
scanf (" %f %5f3d%35%5 ", &x , &y , &a , c r, ct);
Nhập vào 5.4 25 124 523 48ab Enter .
=> kết quả là : x=5.4 ; y=25.0; a = 124; cr= "523"; ct = "48ab"
2.3/ Dòng vào STDIN (standard in) và các hàm scanf, gets, getchar.
- StdIn dòng vào chuẩn(bàn phím).
- Lưu ý : nếu từ Stdin có đủ dữ liệu thì các hàm trên sẽ nhận 1 phần dữ liệu mà dòng yêu cầu.
Phần còn lại (chưa nhập) vẫn trên StdIn.. Nếu chưa đủ đợi đến khi Enter.
* Ví dụ : char ht[20] ;
print (" \n hoten: ") ;
gets(ht);
- Hàm getchar() nhận 1 ký tự từ stdIn và trả về ký tự nhận được.
* Ví dụ : Int ch; ch = getchar(); nếu nhập A và enter => ch='A'
'\'n vẫn còn trên stdIn và hàm getchar sau đó hàm scanf cũng như vậy.
- Làm sach stdIn : fflush(stdin);
Ví dụ : Print("\ tuoi : n"); scanf (" %d " , &tuoi);
Printf ("\n hoten :"); fflush(stdin); get(ht);
Ví dụ : scanf(%d", &a); ch =getchar(); gets(ht);
Nhập vào liên tục : 12E Trần Văn T (Enter).
=> kết quả là : a =12, ch = 'E', ht = " Trần văn T"
ã Hàm puts : đưa một chuỗi ký tự ra stdout (màn hình)
Ví dụ : puts('\n lophoc"); đưa dòng chữ lợp học lên 1 dòng mới.

* Hàm putchar : đưa 1 ký tự lên stdout .
Ví dụ : putchar('A') ; ----> in ra ký tự A.
Chú ý : Tất cả các hàm trên khai báo trong stdio.h.
2.4 các hàm vào ra màn hình , bàn phím thuộc hàm conio.h
- Hàm getch() : nhận 1 ký tự trực tiếp từ dộ đệm bàn phím và trả về ký tự nhận được
- Hàm getchc () : nhận 1 ký tự trực tiếp từ và hiển thị trên monitor
- Hàm Putch (Int ch) : hiển thị ký tự ch theo miền xác định trong hàm textcolor
#putchar () hiển thị theo màu trắng.
Int Kbhit(void) = 0 nếu bộ đệm bàn phím rỗng.
0 nếu bộ đệm bàn phím khác rỗng.
Chú ý : Nếu gõ phím khi máy dừng chờ trong các hàm scanf, gets, getchar thì ký tự vào stdin =>
if (Kbhit ()) ch = getch() ; hoặc scanf(" %d%*c, &i); (để khử ' \n ').
+ clrscr(); hàm xoá màn hình.
+ goto xy (x,y): di chuyển con trỏ đến toạ đô ỹ(x,y) : x : cột (1..80); y : dòng 1..25.
* Ví dụ : viết chương trình nhập vào tên ban và in ra lời chào :

```
# Include <stdio.h>
#Include<conio.h>
main ()
{ char name[30], ch;
printf( " nhập tên của bạn : "); scanf (" % s ", &name);
printf(" \n chào %s!\n", name);
ch = getch();
/* ( đợi nhận số 1 ký tư ỷ=> dừng màn hình*/
```

CHƯƠNG 3 : CẤU TRÚC ĐIỀU KHIỂN VÀ VÒNG LẶP

3.1/ a/Khái niệm : mọi chương trình đều có thể biểu diễn qua 3 cấu trúc :
- tuần tự : mặc định (default)
- lựa chọn (lệnh if hoặc lệnh switch)
- lặp (for, while hoặc do while)

b/ Khối lệnh : là tập hợp các câu lệnh được khai báo bởi 2 dấu { và } .
không đặt dấu chấm phẩy (;) sau một khối lệnh trừ một vài trường hợp đặc biệt.
3.2 / Các câu lệnh
3.2.1 Lệnh If :
- Cú pháp : If (biểu thức) < lệnh> ;
- Diễn giải : nếu Biểu thức đúng (khác 0) --> thực hiện <lệnh>

ngược lại nếu biểu thức sai (= 0) -ă thực hiện lệnh đứng sau câu lệnh if.
- Hoặc : If (biểu thức) <lệnhA>;
  else < lệnh B);

+ Biểu thức : # 0 (đúng) ----> < lệnh A>
  =0 (sai) ---> < lệnh B>.
* Ví dụ : tìm số lớn nhất trong 2 số a, b :
  if (a<b) max = b ;
  else max = a ;
(Viết lại hoàn chỉnh chương trình trên).
* Cách 2 : max = (a>b)? a:b; (Viết lại hoàn chỉnh chương trình).
- Chú ý : trong trường hợp có nhiều lệnh If lồng nhau thì else sẽ gắn liền với if gần nhất.
  If(bt1) <lệnh1>;
  Else

```
          If (bt2)
          If(bt3) < lệnh2>;
          else <lệnh 3>; /* bt3 = = 0 */
          else <lệnh 4>; /* bt2= = 0 */
```

*Ví dụ : Viết chương trình giải phương trình bậc nhất : Ax + B = 0 (A, B : số thực).

Giải : Xét các trường hợp xảy ra :

- Nếu A! =0 thì nghiệm x = -B/A
- Nếu A = 0 +> B=0 => Nếu B=0 : vô số nghiệm
  B != 0 (ngược lại) : vô nghiệm.

```
/* Giải phương trình bậc nhất : Ax + B = 0 */
#Include <stdio.h>
#Include < conio.h>
void main ( void)
{
  float a, b ;
/* nhập dữ liệu từ bàn phím */
   print ( "\ nhập 2 số a,b : "); scanf(" %f %f ", &a, &b);
/* giải phương trình*/
  If ( a= = 0 )
  If( b= =0 )
     Printf (" Phương trình có vô số nghiệm ! \n " );
  Else
     Printf (" phương trình vô nghiệm \n ");
  Else / * a khác 0 */
     Printf (" phương trình có nghiệm là : x= %f \n ", -b/a);
     Printf( " ấn phím bất kỳ tiếp tục ");
     Getche();
 }
```

Bài tập 1 : Tìm những lỗi cú pháp các đoạn chương trìnhh sau :

```
A/ scanf ( "d", value);
B/ printf ("tích các %d và %d là %d " \n, x,y);
C/ printf (" phần dư của %d chia cho %d là \n ", x , y , x%y );
D/ if(x=y);
      Printf (" %d bằng %d \n ", x,y);
E/ If ( age>=65);
      Printf (" gia ì! ');
Else
      Printf(' Tre! ');
```

3.2.2 Lệnh switch

- Cú pháp : Switch (biểu thức nguyên).

```
{
  Case N1 : lệnh 1;
  Case N2 : lệnh 2;
  .....
 [ default : lệnh;]
}
```

- Biểu thức nguyên là giá trị nguyên : Ni(i=1,2...) là các số nguyên.
- Với biểu thức khác với mọi Ni => thực hiện lệnh sau default.
- Chú ý : nếu nhóm câu lệnh sau nhãn case Ni không có câu lệnh break thì máy sẽ chuyển sang nhóm câu lệnh sau nhãn case Ni+1

*Ví dụ : đổi 1 số nguyên sang chuỗi ký tự là tên các môn học

```
#Include<stdio.h>
```

```
#Include<conio.h>
main( )
{
      Int ma ;
     Do
     {
       printf(" \n cho mã cần chuyển "); scanf(" %d ", &ma);
       switch(ma)
        {
          case 0 : printf(" \n lớp tin học a ");
          break;
          case 1 : printf( " \n lớp tin học b");
          break;
          case 2 : printf(" \n lớp trung cấp ");
          break;
          case 3 : printf (" \n lóp chuyên viên ");
           break;
          default : printf( " \n lợp thiế tiền học phí");
       }
    printf( " \n có tiếp tục không ?(Y/N)");
   }
while( toupper ( getch () ! = 'N '); /* Chuyển san ký tự hoa */
}
```

3.2.3 / Lệnh For :

- Cú pháp : for (bt1; bt2 ; bt3) lệnh;
- Giải thích :

+ bt1 : là toán tử gán để tạo giá trị ban đầu cho biến điều khiển.
+ bt2 : biểu thức điều kiện để thực hiện vòng lặp.
+ bt3 : biểu thức tăng giá trị của biến điều khiển của vòng lặp.

*Ví dụ : Tính Tổng S=1+2+3+..+n

```
For ( int i=1, s=0; i<=n; s+ =i, ++i );
```

* Cơ chế hoạt động :

a/Tính giá trị của biểu thức bt1 .
b/Tính giá trị của bt2
c/ + Nếu giá trị của bt2(=0) là sai máy sẽ ra khỏi lệnh For.
+ Nếu giá trị của bt2(!=0) là đúng thì máy sẽ thực hiện lệnh.
d/ Tính giá trị của bt3 và quay lại bước kiểm tra 2(b)
Chú ý : + Khi bt2 vắng mặt thì nó được coi là luôn luôn đúng

* Ví dụ : for (i=0; ; i++) lệnh ;

+ bt1 , bt3 có thể bao gồm nhiều biểu thức cách nhau bởi dấu phẩy.
+ bt2 có thể gồm nhiều biểu thức, tuy nhiên tính đúng sai của nó được xem là tính đúng sai của biểu thức cuối cùng.

* Ví dụ : tính tổng : S=1! + (1+2)! + ...+ (1+2+....i)! .....(1 + 2 + ..n)!

```
#include <stdio.h>
#include<conio.h>
#include<math>
/* int i, j, t, n ; double gt, s; */
main()
{
   int i, j, t, n ; double gt, s;
   clrscr () ;
    printf ("nhập n= "); scanf(" %d ", &n);
```

```
  Cách 1 :
  s=0 ; t=0;
  for (s= 0,t= 0,i=1; i<=n ; ++i )
   {
       t=t+i;
      for ( gt=1,j=1; j<=t ; ++j)
          gt = gt*j ; s = s+gt;
  }
   printf ( " tong s = %15.0f ", s);
  getch();
Cách 2 :
for ( s=0, t=1, i=1; i<=1; ++i , t = t + 1)
  {
     for ( gt=1,j=1;j<=t; ++j)
     gt*=j; s+= gt;
  }
Cách 3 : thân for là câu lệnh rỗng
For (s=0, t=1,i=1; i<=n; ++i, t=t+i,s+=gt)
For( gt=1,j=1; j<=t; gt* = j , ++j );
Cách 4 : không có bt 1 và bt3;
Int i=1, j=1, t=1, n ; double gt = 1, s= 0 ;
For ( ; i<=n ; ++i, t = t + i , s+ = gt) /* không có biểu thức 1*/
  {
    For ( ; j<=t ; ) /* không có bt1 , bt3*/
    Gt* = j ++ /* gt = j ; ++j */
  }
Cách 5 : không có bt1, bt2, bt3
For (; ;)
{
  for (; ;)
   {
     gt* = j++ ;
       if ( j < t ) goto tong ;
   }
    tong : s+ = gt; ++i , t = t + i ;
   if( i < n ) goto KT;
   KT : printf (" tong s= % 15.0 f " , s )
}
```

3.24/ Câu lệnh while :

- Cú pháp : while (biểu thức 1) lệnh 1 ;
- Nguyên tắc thực hiện :

+b1. Tính giá trị của biểu thức 1.
+b2. Nếu giá trị của biểu thức 1 sai (= 0) thì chương trình ra khỏi vòng while
+b3. Nếu giá trị của biểu thức đúng thì thực hiện lệnh 1 và quay lại bước 1(b1).

- Chú ý : Biểu thức 1 có thể gồm nhiều biểu thức nhưng tính đúng sai phụ thuộc vào biểu thức cuối cùng.

Ví dụ : Nhập 1 dãy số nguyên từ bàn phím

```
#include < stdio.h >
#include < conio.h >
main ()
{
  Int dayso [ 10 ] ; int i = 0 ;
```

```
While ( i < 10)
    {
       printf ( "\n Số thu %d : ", i ); scanf ( " %d", & dayso [i]);
       i ++ ;
    }
```

3.25/ Câu lệnh Do while (làm trước hỏi sau)

- Cú pháp : do lệnh 1 ;
while (biểu thức 1) ;
- Nguyên tắc thực hiện :
+b1. Máy thực hiện câu lệnh 1 ;
+b2. Sau đó tính giá trị của biểu thức 1, nếu giá trị của biểu thức 1 sai thì chương trình thoát ra khỏi vòng lặp. Nếu giá trị của biểu thức 1 đúng thì quay lại bước 1.
Chú ý : - while : Điều kiện được kiểm tra trước, nếu đúng mới thực hiện.
- do while : câu lệnh được thực hiện trước khi kiểm tra. Câu lệnh thực hiện bao giờ ít nhất là 1 lần. (do while ngược với
Repeat until của Pascal : lệnh Do while sai thì dừng, còn lệnh repeat until đúng thì dừng).
-Biểu thức 1 có thể gồm nhiều biểu thức, tuy nhiên tính đúng sai căn cứ theo biểu thức cuối cùng.
* Ví dụ : tính pi với sai số eps = 1E - 4 , pi = 4 - 4/3 + 4/5 - 4/7 + ...eps

```
#include < stdio.h >
#include < conio.h>
main ()
 {
   float pi, dau, i , eps, saiso ;
   i=1.0; dau = -1; saiso = 1e -4 ;
   pi = 4.0;
   printf ( "\n đang xử lý vui lòng đợi !");
   do
    {
      eps = 4.0 / ( 2.0 * i + 1.0 );
      pi + = dau * eps ; dau = dau * - 1.0 ; i + = 1.0;
    }
     while ( eps > saiso );
        printf ("\n số pi là : " % f ", pi ) ;
       getch ();
  }
```

3.2.6/ Câu lệnh Break :

- Cú pháp : Dùng để thoát khỏi vòng lặp. Khi gặp câu lệnh này trong vòng lặp, máy ra khỏi và chỉ đến câu lệnh sau các lệnh trên. Nếu nhiều vòng lặp ----> break sẽ thoát ra khỏi vòng lặp gần nhất.

3.2.7/ Lệnh continue :

- Cú pháp continue; : khi gặp lệnh này trong các vòng lặp, máy sẽ bỏ qua phần còn lại trong vòng lặp và tiếp tục thực hiện vòng lặp tiếp theo.
- Đối với lệnh For máy sẽ tính lại biểu thức 3 (bt3) và quay lại bước 2.
- Đối với lệnh while, do while máy sẽ tính lại giá trị của biểu thức 1 và quay lại bước 1.
* Ví dụ : Nhập 1 chuỗi ký tự kể cả ký tự trống và bỏ qua các ký tự không hợp lệ và kết thúc khi ấn ESC hoặc số ký tự vượt quá kích thước mãng.

```
char xau [MAXL], kytu ;
int i = 0 ;
while (1) /* luôn luôn đúng vòng lặp vĩnh cửu */
  {
    kytu = getch ( ) ;
```

```
    if ( kytu = = 27 ) break ;
     if ( i >= MAXL ) break ;
     if ( kytu > 122 || kytu < 65 ) continue ;
        Xau [ i ++] = kytu ;
   }
xau [ i ] = ' \ 0 ' ;
```

3.3/ Toán tử goto và nhãn (label);

- Ví dụ : tiep tuc : st = a[i]; => tiep tuc là nhãn của lệnh st = a [i];
- Lệnh goto nhãn => nhảy đến câu lệnh đứng sau nhãn.
- CHÚ Ý : PHẠM VI NHÃN TRONG CÙNG 1 HÀM.

BÀI TẬP CHƯƠNG 3

1/ Kiểm tra tìm lỗi :

```
while ( x<.= 10 )
     Total t=x;
     ++x ;
```

2/ Giải phương trình bậc 2 : ax2 + bx + c = 0 với a, b, c là số thực nhập từ bàn phím.

3/ Nhập số liệu vào bàn phím, kết thúc nhập bằng cách ấn ^Z hoặc F6 (mũ = 255). Biết : InWord = on khi ở trong 1 từ. InWord = off khi ngược lại. Đếm số dòng, số từ, số ký tự.

4/ Tìm các số nằm trong khoảng từ 150 đến 140 thoả tính chất số bằng tổng lập phương các chữ số của chúng :

Ví dụ : 153 = 13 + 53 + 33 hoặc 370 = 33 + 73 + 03

5/ Số tuyệt hảo là số bằng tổng các ước số thực sự của nó. Ví dụ : 6 = 1 + 2 + 3.Tìm các số tuyệt hảo trong khoảng từ 1 đến 3000.

6/Nhập số liệu vào mãng A gồm 10 phần tử và sắp xếp theo thứ tự tăng dần.

7/ Tìm tất cả các số nguyên tố từ 2 đến 100 bằng lệnh For.

8/ Tìm các số nguyên có 3 chữ số sao cho tổng 3 chữ bằng tích 3 chữ. Ví dụ : 123.

9/ a/ Dùng lệnh while để viết chương trình tính :

S1 = 1 x 3 x 5 x 7 x 9. . . . . x (2n - 1).

S2 = 2 x 4 x 6 x 8 x .....x (2n).

b/ làm lại bài trên bằng cách dùng do...while.

10/Giải bài toán cổ điển vừa gà vừa chó bó lại cho tròn 36 con 100 chân chẵn.

CHƯƠNG 4 : HÀM CHƯƠNG TRÌNH VÀ CẤU TRÚC CHƯƠNG TRÌNH.

Chương trình viết bằng ngôn ngữ C gồm 1 dãy các hàm trong đó có 1 hàm chính là main và chương trình bắt đầu từ main.

4.1/ Khái niệm :

- Hàm là đoạn chương trình thực hiện trọn vẹn một công việc nhất định.
- Hàm chia cắt việc lớn bằng nhiều việc nhỏ. Nó giúp cho chương trình sáng sủa, dễ sửa, nhất là đối với các chương trình lớn.

4.2/ Khai báo hàm :

```
< Tên hàm > (< danh sách các đối số>)
< Khai báo biến >
{
< Khai báo thêm các biến >
< Các câu lệnh >
}
```

- Trong đó :

+ Tên hàm : buộc phải có.

+ Danh sách các đối số : không bắt buộc. Có hay không tuỳ theo chúng ta định dùng hàm đó

làm gì.
+ Khai báo biến : Nếu Danh sách các đối số mà có thì phần này buộc phải có. Còn nếu không thì ngược lại có thể bỏ qua.
+ Phần trong { } : là thân hàm. Dấu { } là bắt buộc đối với mọi hàm.
+ < Khai báo tham biến > : ngay sau { và gọi là biến cục bộ dành riêng cho hàm sử dụng.
+ đối số luôn luôn truyền theo trị (không thay đổi giá trị).
*Ví dụ : Hàm tính giai thừa : S = x 1 /1! + x 2 /2! + ...+ x n / n!
Cách 1 :

```
#Include <stdio.h>
#Include <conio.h>
float giaithua ( int n)
{
 int i ;float KQ ;
 for ( KQ=1,i =1 ; i<=n ; i ++ )
      KQ = KQ * i ;
    return KQ ;
}
Void main ( ) /* khai báo biến toàn cục nếu có */
{
   int n ;
   printf ( " Nhập n = " ); scanf ( " %d ", &n);
   printf ( " %d giai thừa là % f ", n, giaithua (n) );
   getch ();
}
```

Cách 2 :

```
#Include <stdio.h>
# Include<conio.h>
/*Khai báo prototype*/ mục đích hàm đặt ở đâu cũng được không cần trước hàm gọi
float giaithua ( int n );
void main ()
{
}
/* Chi tiết hàm giai thừa */
float giaithua ( int n)
{ ... return KQ };
```

Chú ý : - Kiểu của hàm cùng kiểu giá trị cần trả về.
- Các hàm độc lập, không được lồng nhau.
- Kiểu void tên hàm () : không cần trả về giá trị nào, hiểu ngầm là trả về int.
- ở cách 1 : hàm ở trên không được gọi hàm dưới.
- ở cách 2 : các hàm gọi được lẫn nhau.

4.3 / Phạm vi của biến :
- Chẳng hạn trong ví dụ trên : biến n trong hàm main ( ) là cục bộ của main() chỉ có tác dụng trong hàm main() => trong hàm giai thừa có thể khai báo biến n trùng biến n của hàm main ( ) nhưng khác nhau và chỉ tồn tại trong 1 hàm.
Ví dụ : float giaithua (m);

```
  {
     int n ; float KQ = 1.0;
     for ( n = 1; n<= m ; ++n )....
```

4.4 / Đệ quy : giống như trong Pascal : hàm gọi đến chính nó.
* Ví dụ : Tính giai thừa :

```
giaithua ( n );
int n ;
```

```
{
  if ( n = 0 ) return ( i ) ;
  else return (giaithua ( n - 1 )*n );
}
```
- Chương trình sử dụng đệ quy thì dễ hiểu nhưng không tiết kiệm được bộ nhớ, không nhanh hơn.

4.5/ So sánh Lệnh trong Pascal và trong lập trình ngôn ngữ C.

- Giống nhau : + Cả Pascal và C đều có chương trình con.
- Khác nhau :

Pascal Ngôn ngữ C
Có thủ tục Chỉ có hàm
Có hàm Hàm có thể khai báo kiểu void (không trả về giá trị nào cả, giống như thủ tục của Pascal
- Khai báo hàm

function Tên hàm (<danh sách biến) < kiểu hàm>;
< Khai báo các biến cục bộ>
Begin
< Các câu lệnh>
end; < Kiểu> tên hàm (< danh sách các biến>)
{
< khai báo các biến cục bộ>
Các câu lệnh
}
Khai báo biến
<tên biến >: < kiểu biến>;
Ví dụ : Function max (a, b : integer) : integer
Begin
if a > b then max = a
Else max = b ;
End.
Trả về giá trị bằng phép gán max = giá trị (trong đó max là tên hàm). Khai báo biến
< kiểu biến> < tên biến >;
Ví dụ : int max (a, b)
{
  If (a > b) return (a);
  else return (b);
}
- Trả về giá trị bằng câu lệnh return (giá trị)

Kiểu tham số
+ Tham biến : truyền theo địa chỉ
+ Tham trị : truyền theo giá trị.
Tham biến trong Pascal
Procedure swap (var x, y : real);
Var temp : real ;
Begin
Temp : = x ; x : = y ; y : = temp;
End.
- gọi hàm : swap (a, b) Kiểu tham số
+ Chỉ có tham trị.
+ Muốn có tham biến bằng cách đưa con trỏ hình thức tham biến trong C.

Tham biến trong C
Void swap (float *x, float * y)

```
 {
   float temp ;
   temp = * x ; *x = * y ; * y = temp ;
 }
 swap ( &s, &b )
```

CHƯƠNG 5 : MẢNG VÀ BIẾN CON TRỎ

5.1/ Mảng : là tập hợp của các biến cùng kiểu được xếp liên tiếp nhau trong bộ nhớ trong.

5.1.1/ Mảng 1 chiều :

a/ Khái niệm : < kiểu phần tử > < tên mãng> [< chỉ số >]

Ví dụ : int a [5] ; => a [0] a[1] a[2] a [3] a [4] (chỉ số chạy từ 0 đến n - 1).

Char S [20] ; => 'A' 'B' ..... 'X '

S[0]S[1] S[19]

b/ Cách nhập số liệu cho mảng từ bàn phím (có thể dùng hàm Random C).

+ Mảng số nguyên :

Ví dụ : Nhập vào mảng số nguyên 5 phần tử

```
#include < stdio.h>
#include < conio.h>
#define n 5
main ()
{
   int a [ n ] ; int i ;
   for ( i = 0 ; i < n ; i ++ )
     {
       printf ( " a [ %d ] = " , i ); scanf ( " % d" , & a [ i ]);
     }
/* Xuất số liệu mảng ra màn hình */
     for ( i = 0 ; i < n ; ++ i)
     printf ( " \ n a [ % d ] = % d ", i , a [ i ]);
     getch ();
}
```

+ Mảng số thực float :

```
#include <stdio.h>
#include < conio.h>
#define n 5 ;
main ()
{
    float a [ n ] , tam ;
    .....scanf ( " % f " , &tam) ; /*nhập qua biến trung gian tạm */
    a [ i ] = tam ;
```

c/Khởi tạo mảng :

a [5] = { 1,2,3,5,4 }a[0]=1 a[2]=2 .. a[4]=4

d/ Mảng ký tự :

- là chuỗi ký tự kết thúc bằng ký tự NULL có mã ASCII là 0 .
- Ví dụ : char S [3] = { 'L', '0', 'P'] : chuỗi này không đúng do thiếu chỗ cho ký tự kết thúc là NULL.
- Ta có thể gán :

char S [4] = " Lop "; Ngôn ngữ C sẽ tự động ghi ký tự kết thúc là NULL, tức là ' \0 '.

char S[ ] = " Lop " ; Không cần khai báo số phần tử mãng.

* Ví dụ 1 : Nhập vào một mảng số nguyên sau đó sắp xếp theo thứ tự tăng dần :

```
#include < stdio.h>
```

```
#define n 5
main ( )
  {
    int a [ n ] ; int i , j, t ;
    for ( i = 0 ; i > n ; i ++ );
     {
       printf ( " nhập a [ % d] = " , i ); scanf ( " %d", & a [i ]);
     }
    /* Sắp xếp tăng dần */
     for ( i = 0 ; i < n - 1 ; i ++)
     for ( j = i + 1 ; j < n ; j ++ )
       if ( a [ i ] < a [j ] )
         {
           t = a [ i ] ; a [ i ] = a [ j ]; a [j ] = t ;
         }
/* in kết quả */
      for ( i = 0 ; i < n ; i ++ )
      printf ( " % 5d " , a [ i ] );
      getch ( );
   }
```

Ví dụ 2 : Làm lại ví dụ 1 nhưng viết riêng hàm sắp xếp và truyền tham số cho mảng 1 chiều

```
#include <stdio.h>
#include <conio.h>
#define N 5
void sapxep ( int a [ ] , int n );
void main ( )
  {
     int a [ N ] ; int i ;
   /* nhập 1 số liệu cho mãng */
    for ( i = 0 ; i < N , i ++ )
    {
       printf ( " A [ %d ] = ", i ); scanf ( " %d ", & a [ i ] ); }
  /* gọi hàm sắp xếp để sắp tăng dần */
         sapxep ( a, N );
/* in kết quả */
        for ( i = 0 ; i < N ; i ++ )
           printf ( " %5d ", a [ i ] );
           getch ( );
   }
/* hàm sắp xếp tăng dần */
void sapxep ( int a [ ], int n )
  {
     int i, j, t ;
      for ( i = 0 ; i > n - 1 ; i ++)
      for ( j = i + 1 ; j < n ; j ++ )
        if ( a [ i ] > a [ j ]
           {
               t = a [ i ] ; a [ i ] = a [ j ] ; a [j ] = t ;
           }
```

* Ví dụ 3 : chuyển đổi 1 chuỗi ký tự thường thành Hoa.

Chú ý : + Hàm tolower (ch) : đổi 1 ký tự ch thành thường.

+ Hàm toupper (ch) : đổi ký tự ch thành Hoa.

+ Cả 2 hàm trên đều năm trong thư viện : < ctyte.h>
Giải : #include < stdio.h>
include < ctyte.h>
#define n 20
main ( )
{
char s [n] ; int i ;
for (i = 0 ; i < n ; i ++)
s[i] = toupper (getchar ( )) ; /* nhập ký tự và đổi thành hoa lưu vào mãng */
/* kết xuất chuỗi s */
for (i = 0 ; i < n ; i ++)
putchar (s [i]) ; /* putchar (ch) : in ký tự ch ra màn hình */
getch ( )
}
Bài tập : 1/ viết chương trình nhập số liệu cho mảng A gồm N phần tử và mảng B gồm n phần tử , sau đó ghép 2 mãng A và B thành mãng C gồm m + n phần tử và sắp xếp tăng dần (Bài này phải dùng hàm nhập số liệu cho mảng và hàm sắp xếp).
- Tính tổng các phần tử âm, dương, số chẵn, số lẽ và tổng tất cả các phần tử của mãng C [m + n].In các số lẻ trên 1 hàng và các số chẵn trên 1 hàng.
- Nhập vào một giá trị và tìm xem giá trị đó có thuộc vào mãng C không. Nếu có in ra tất cả các phần tử tìm được.
5.2/ Mãng nhiều chiều :
a/ Khai báo : < kiểu phần tử > < tên mãng > [< chỉ số hàng >] [< chỉ số cột >]
*Ví dụ 1 : int a [3] [2] ; float b [3] [4] ; char c [5] [6] ;
=> a [0] [0] a [0] [1]
a [1] [0] a [1] [1]
a [2] [0] a [2] [1]
Ví dụ 2 : #define Hang 5
define Cot 6
int a [Hang] [Cot] ;
=> ta có các biến chạy i (chỉ số chạy từ 0 đến (Dong - 1)).
ta có các biến chạy j (chỉ số chạy từ 0 đến (Cot - 1)) .
a [0] [0] a [0][1] ..... a [0][Cot - 1]
a [1] [0] a [1][1] ..... a [a][Cot - 1]
. . . . . .
a[Dong-1][0]..... . . . . . a[Dong-1][Cot-1]
*Ví dụ : Viết chương trình tính tổng, tích các số trong mãng số thực a[3][2] ;
#include < stdio.h>
#define N 3
#define N 2
main ( )
{
int i , j ; float a [M][N] ; float tong, tich, tam ;
/* nhập số liệu */
for (i = 0 ; i < M ; i ++)
for (j = 0 ; j < N ; j ++)
{ printf (" nhập a [%d][%d] = " , i , j);
scanf (" %f " , & tam) ; a [i][j] = tam ;}
/* tính tổng */
Tong = 0 ; Tich = 1;
for (i = 0 ; i < M ; i ++)
for (j = 0); j < N ; j ++)

```
      {
          Tong = Tong + a [ i ][j] ; Tich = Tich * a [i][j] ; }
/* in kết quả */
printf ( " Tổng là tổng = %f, TONG );
printf ( " tích là TICH = %F, TICH );
getch ( ) ;
}
```

b/ Truyền tham số mãng nhiều chiều cho hàm (tham số thực là tên mãng nhiều chiều)

- giả sử a là mãng 2 chiều : float a[M][N]

+ Chương trình gọi :

```
{ float a [M][N]
Tong ( a ) ; ( truyền địa chỉ của mãng cho hàm )
}
```

+ Chương trình bị gọi (chương trình con) :

```
float tong ( float a[ ][N] ) /* khai báo đối để nhận địa chỉ của mãng */
{
}
```

Note : hàm tong chỉ dùng được đối với các mãng hai chiều có N cột và số hàng không quan trọng, không khai báo) :

* Ví dụ : Viết chương trình tính tổng của 2 ma trận cấp m x n theo công thức :

C[i][j] = a[i][j] + b [i][j]

```
#include <stdio.h>
#define m 3
#define n 4
/* các prototype ( khai báo hàm )*/
void nhap ( int a[ ][N] , int M, int N );
void TongMT ( int a[ ][N], int b[ ][N] , int c [ ][N], int M , int N );
void InMT ( int c [ ][N], int M, int N );
/* chương trình chính */
{ int a [M][N], b[M][N], c[M][N] ;
/* gọi các hàm */
Nhap ( a, M ,N ) ; nhap ( b, M,N);
TONGMT ( a, b, c , M, N );
InMT ( c, M, N );
Getch ( ) ;
}
/* Hàm nhập số liệu cho mãng 2 chiều m x n phần tử */
void Nhap ( int a [ ][N] , int M , int N )
{
  int i , j ;
  for ( i= 0 ; i < M ; i ++ )
  for ( j = 0 ; j < N ; j++ )
    {
        printf ( " a[%d][5d] = " , i , j ) ; scanf ( " %d " , &a [i][j]) ; }
    return ;
}
Void TongMT ( int a [ ][N], int b [ ][N], int c [ ][N], int M , int N )
{
  int i, j ;
  for ( i = 0 ; i < M ; i ++ )
  for ( j = 0 ; j < N ; j ++ )
      c [i][j] = a [i][j] + b [i][j] ;
```

```
    return ;
}
/* in kết quả */
void inMT ( int c[ ][N], int M, int N )
{
  int i, j ;
  for ( i = o ; i < M ; i ++ )
    { for ( j = 0 ; j < N ; j ++ )
        printf ( " % 3d", a[i][j] );
        printf ( " \n " ) ; /* xuống dòng */
    }
   return ;
}
```

Bài TậP MãNG :

1/ cho mãng 2 chiều A, là ma trận vuông cấp n x n , lập chương trình :
a/ tính tổng tất cả các phần tử dương của mãng.
b/ tính tổng các phần tử A[i][j] mà i + j chia hết cho 5 .
c/ In ra các số nguyên tố theo từng hàng.
d/ Sắp xếp theo hàng.
e/ Sắp xếp theo cột .
f/ Tính tổng các phần tử trên đường chéo (i = j) , đường biên.
g/ Tìm max ; min theo từng hàng, cột và toàn bộ ma trận.
2/ Một chuỗi gọi là palindrone nếu nó không thay đổi khi ta đảo ngược thứ tự của các ký tự trong nó (ví dụ " 12321 ")
. Lập chương trình đọc một chuỗi (xâu) ký tự và xác định xem có tính palondrone không.

5.3/ Biến con trỏ :

5.3.1/ Khái niệm con trỏ (pointer) và địa chỉ :

- Mỗi biến trong ngôn ngữ C đều có 1 tên và tương ứng với nó là một vùng nhớ dùng để chứa giá trị của nó. Tuỳ theo biến mà vùng nhớ dành cho biến có độ dài khác nhau. Địa chỉ của biến là sô thứ tự của byte đầu tiên tương ứng với biến đó. Địa chỉ của biến có kiểu khác nhau là khác nhau. Địa chỉ và biển kiểu int liên tiếp cách nhau 2 byte , biến kiểu float là 4 byte.
- Con trỏ là biến dùng để chứa địa chỉ của biến khác hoặc có thể là một hàm. Do có nhiều loại địa chỉ nên cũng có nhiều loại biến con trỏ. Con trỏ kiểu int dùng để chứa địa chỉ của kiểu int. Con trỏ kiểu float dùng để chứa địa chỉ kiểu float.
- Muốn sử dụng được pointer, trước tiên phải có được địa chỉ của biến mà ta cần quan tâm bằng phép toán lấy địa chỉ & . Kết quả của phép lấy địa chỉ & sẽ là 1 phần tử hằng.

* Ví dụ : int num ; => &num là địa chỉ của num.

```
int pnum ; /* pnum là 1 pointer chỉ đến một int */
pnum = & num ; /* pnum chứa địa chỉ biến int num*/
```

giả sử : num = 5 ; => * pnum = 5 /* do * là toán tử nội dung */
Hai câu lệnh sau đây là tương đương

```
Num = 100 ;
( * pnum ) = 100 ;
```

- Quy tắc khai báo biến con trỏ : < kiểu dữ liệu> * < tên biến con trỏ >

*Ví dụ 2 : int a, *p ;

```
a = 5 ; /* giả sử địa chỉ của a là < 106 > */
p = & a ; /* p = <106> */
p = a ; /* phép gán sai */
* p = a ; /* phép gán đúng */
scanf ( " %d " , &a ) ; tương đương scanf ( " %d , p ) ;
```

5.3.2/ tính toán trên biến con trỏ (pointer)

a/ Hai biến con trỏ cùng kiểu có thể gán cho nhau :
Ví dụ 1 : int a, * p, *a ; float * f;
a = 5 ; p = &a ; q = p ; /* đúng */
f = p ; /* sai do khác kiểu */
f = (float *)p ; /* đúng nhờ ép kiểu con trỏ nguyên về kiểu float */
Ví dụ 2 : int a ;
char *c ;
c = &a ; /* sai vì khác kiểu */
c = (char*) /* đúng */
b/ Một biến pointer có thể được cộng, trừ với một số nguyên (int , long) để cho kết quả là một pointer.
* Ví dụ : int a , *p , * p10 ;
a = 5 ;
p = &a ;
p10 = p + 10 ;
Ví dụ : int V[10] ;/* mãng 10 phần tử */
int *p ;
p = & V[0];
for (i = 0 ; i < 10 ; i ++)
{ *p = i ; /* gán giá trị i cho phần tử mà p đang trỏ đến */
p ++ /* p được tăng lên 1 để chỉ đến phần tử kế tiếp */
}
/* kết quả V[0] = 0 , V [1] = 1 ... V[9] = 9 * /
c/ Phép trừ 2 pointer cho kết quả là một số int biểu thị khoảng cách (số phần tử) giữa 2 pointer đó.
d/ Phép cộng 2 pointer là không hợp lệ, pointer không được nhân chia với 1 số nguyên hoặc nhân chia vơi nhau.
e/ p = NULL : là con trỏ p không trỏ đến đâu cả.
Chú ý : không được sử dụng biến con trỏ khi chưa được khởi gán .
Ví dụ : int a , *p ;
Scanf ("%d", p) (sai)
=> thay bằng các lệnh : p = &a và scanf ("%d" p) (đúng)
5.4/ Con trỏ mảng :
5.4.1/ Mãng 1 chiều và con trỏ :
- Trong ngôn ngữ C : giữa mãng và con trỏ có mối quan hệ chặt chẽ. Các phần tử của mãng có thể xác định nhờ chỉ số hoặc thông qua con trỏ.
- Ví dụ : int A[5] ; * p ;
P = A ;
+ mãng bố trí 5 ô nhớ liên tiếp (mỗi ô chiếm 2 byte).
+ Tên mãng là 1 hằng địa chỉ (không thay đổi được), chính là địa chỉ của phần tử đầu tiên. => A tương đương với &A[0]
(A + i) tương đương với &A[i]
*(A + i) tương đương với A[i]
p = A => p = &A[0] (p trỏ tới phần tử A[0])
*(p + i) tương đương với A[i].
=>bốn cách viết như sau là tương đương : A[i], * (a + i), * (p + i), p[i].
Ví dụ 2 : int a [5] ; *p ;
p = a ;
for (i = 0; i < 5 ; ++ i)
scanf (" %d ", &a[i]); (1)
scanf (" %d ",a + i); (2)
scanf (" %d", p + i); (3)

scanf (" % d", p ++); (4)
scanf (" %d ", a ++); sai vì địa chỉ của a là hằng.
- Các lệnh (1), (2), (3), (4) tương đương nhau.
Ví dụ 3 : Nhập 5 số nguyên vào 1 mãng gồm 5 phần tử (a[5]) sau đó sắp xếp tăng dần, in ra số lớn nhất vf nhỏ nhất và tính tổng của 5 số đó.

```
#include <stdio.h>
#define n 5
main ( )
  { int a [n], t , *p, i , j, ; int s ;
     p = a ;
     for ( i = 0; i < n ; i ++ )
      { printf ( " a[%d] = " , i ) ; scanf ( " %d ", p + i ) }
/* Sắp xếp tăng dần */
     for ( i = 0 ; i < n-1 ; i ++ )
     for ( j = i + 1 ; j<n ; j++)
         if ( *(a + i ) > * ( a + j )
             { t = * ( a + i ) ; *(a + i ) = * ( a + j) ; *(a + j ) = t ; }
      s= 0 ;
     for ( j=0 ; i < n , ++i )
        s + = a[ i];
         printf ("\n Tong = %5d ", s );
         printf ( "\n số lớn nhất là %d ", a [4] );
         printf ( " số nhỏ nhất là %d \n ", a [d] );
        getch ( );
       }
```

5.4.2 / Con trỏ và mãng nhiều chiều :
- Phép toán lấy địa chỉ & chỉ áp dụng được với mãng 2 chiều kiểu nguyên. Các kiểu khác không được.
* Ví dụ 1 : int a[2][3]
{ scanf ("%d", & a[1][1]) } (đúng)
* Ví dụ 2 : float a[2][3]
Scanf (" %f", &a[1][1]); (sai).
- Mãng 2 chiều a[2][3] => gồm 2 x 3 = 6 phần tử có 6 địa chỉ liên tiếp theo thứ tự sau :
Phần tử : a[0][0] a[0][1] a[0][2] a[1][0] a[1][1] a[1][2] (*)
Địa chỉ : 0 1 2 3 4 5
- Ngôn ngữ C quan niệm mãng 2 chiều là mãng một chiều của mãng a[2][3] tương đương không phần tử mà mỗi phần tử của nó gồm 3 số nguyên nên :
a trỏ tới hàng thứ nhất (a [0][0])
a+1 trỏ tới hàng thứ hai (a[1][0])
- Do đó để duyệt các phần tử của mãng a[2][3] ta dùng con trỏ theo cách sau :
+ (theo *) => ta có công thức a[i][j] = (int*) a + i * n + j
trong đó : int* : con trỏ a (địa chỉ a).
n : số cột.
- float a[2][3] , *p ;
p = (float*)a ; /* chú ý lệnh này */
khi đó : p trỏ tới a[0][0] /* p = & a[0][0] */
p + 1 trỏ tới a[0][1] /* *(p+1) = a[0][1] */
P + 2 trỏ tới a[0][2]
....
p + 5 trỏ tới a[1][2] /* *(p+5) = a[1][2] */
* Tổng quát : a[i][j] = * (p + i* N + 5); trong đó N : số cột)
Kết luận : Mãng 2 chiều có thể chuyển thành mãng 1 chiều nhờ con trỏ.

* Ví dụ : để nhập một số liệu vào mãng 2 chiều kiểu float a[2][3] ta có thể dùng các cách sau:
+ Cách 1 :

```
#include " stdio.h "
main ( )
  { float a[2][3] , *p ; int i ;
    p = (float*)a ; /* lưu ý lệnh này */
    for ( i = 0 ; i < 2*3 ; ++i)
      scanf ( "%f", (p+i)) ; /* (p_+ i ) là địa chỉ */ ( X )
  }
```

+ Cách 2 : Sửa lệnh (X) như sau : scanf ("%f", (float*)a + 1) ;
+ Cách 3 :

```
#include " stdio.h " #define m 2 #define n 3
main ( )
  { float a[m][n] ; int i , j ; float *p ; p = ( float* )a ;
    for ( i=0 ; i<m ; i++ )
    for ( j=0 ; j<n ; j++ )
    scanf ( "%f" , ( p +i*n + j )
    hoặc lệnh scanf ( " %f" , ( float *)a + i * N + j ));
  }
```

+ Cách 4 : sử dụng biến trung gian :

```
#include " stdio.h"
#define dong 2
#define cot 3
main ( )
  { float a[dong][cot] , tam ; int i , j ;
    for ( i = 0 ; i < dong ; i++ ) ;
    for ( j=0 ; j < cot ; ++j )
      { printf ( "\n a[%d][%d] = " , i , j );
        scanf ( " %f " , &tam ) ;
        a[i][j] = tam ;
         }
```

Bài TậP : Sắp xếp mãng 2 chiều theo hàng và toàn bộ mãng

5.4.3/ Mãng con trỏ : là mãng mà mỗi phần tử của nó có thể chứa một địa chỉ nào đó.
Khai báo : < kiểu dữ liệu > < tên mãng > [<chỉ số>].
* Ví dụ : int *a[5] ;
- trong đó : a là mãng gồm 5 ô nhớ liên tiếp, mỗi ô nhớ là 1 biến con trỏ trỏ đến kiểu int ; bản thân a không thể dùng để lưu trữ số liệu.
- Giả sử : a <100> <102> <104> <106> <108> < 110>
a[0] a[1] a[2] a[3] a[4] a[5]
Địa chỉ < 30> < 20> < 10 > < 80 > < 70 > < 100>
7 8 9 10 11
<10> <12> <14>
1 2 3 4 5
<20> <22> <24> <26> <28>
6 12 13
<30> <32> <34>
- a= &a[0] => a = <100> (địa chỉ 100).
- a[0] = < 30 > (địa chỉ bằng 30 : tại địa chỉ 30 con trỏ a[0] trỏ đến địa chỉ <30 > và giả sử tại địa chỉ < 30 > có giá trị là 6).
=> *a[0] = * (<30>> = 6 .
a[1] = < 20 > => *a[1] = 1
a [2] = < 10> => *a[2] = 7 .

Chú ý 1: Xem a là con trỏ 2 lần (con trỏ của con trỏ) :
- a = <100 > => *a = <30 > (do a = &a[0])
=> **a = 6 (do *(<30>)).
- *(*(a + 1) + 2)
*(102)
* (<20> + 2) => *<24> = 3
Chú ý 2 : - int a[5] => a là con trỏ hằng không thay dổi địa chỉ của nó được (nên a++ sai)
- int *a[5] ; => a laf con trỏ động nên thay đổi giá trị được (a++ đúng).
Ví dụ : int *a[5]
For (i = 0 ; i < 5 ; i++)
{ printf ("%d", *a[0]);
a[0]++ ;
}
* Chú ý 3 : mãng 2 chiều chẳng qua là 1 con trỏ 2 lần (con trỏ của con trỏ).
Lý do : a[i][k] ; trong đó đặt b = a[i] => b[k] = a[i][k] ;
+ Công thức : (a[i] = *(a+i)) => (b[i] = *(b+i)).
b[k] = *(b+k)).
b[k] = *(a[i] + k)
= * (*(a+i) + j).
=> a[i][k] = *(*(a+i) + k) ; trong đó *(*(a+i) là con trỏ 2 lần.
5.4.4/ Con trỏ và xâu ký tự :
- Xâu ký tự : là dãy ký tự đặt trong ngoặc kép . Ví dụ : " Lớp học ". Xâu này được chứa trong 1 mãng kiểu char.
L O P H O C \0
Địa chỉ :<100> <101> <102> NULL : kết thúc chuỗi
=> char *lop ;
lop = " Lop Hoc " ; Đúng : gán địa chỉ của chuỗi cho con trỏ lớp.
+ puts (" Lop Hoc ") ; và puts (lop) để hiển thị dòng chữ Lop Hoc.
Ví dụ : char Tenlop[10] ;
Printf ("\n Tenlop : ") ; gets(Tenlop) ; => (Nhập vào chuỗi " lớp học ")
Còn nếu chúng ta khai báo như sau là sai :
Char *lop , tenlop [10] ;
Tenlop = " lớp học " ; sai vì Tenlop và chuỗi là 2 con trỏ hằng , không được gán cho nhau . Muốn gán ta dùng hàm strcpy (Tenlop , "lớp học ");
5.4.5/ Con trỏ và việc định vị bộ nhớ động :
- Ví dụ 1 :
```
#define N=10 ;
main ( )
  { int a[N] ; int m :
     printf ( " nhập số phần tử m = "); scanf("%d", &m) ;
     for ( i= 0 ; i < m ; i++ )
     scanf ( "%d", &a[i] );
```
- Nhận xét Ví dụ 1 trên : + Nếu m <=N (N =10) : thì sẽ bị dư 1 số biến mãng là (n - m).
+ Nếu m > N (tức là m > 10) : thì chương trình sẽ chạy sai vì ta không đủ biến mãng.
=> Do đó ta phải khắc phục bằng cách : định vị bộ nhớ động. (Bằng hàm malloc và calloc).
* Ví dụ 2 :
```
#include < stdio.h>
#include<alloc.h> hoặc #include <stdio.h >
main ( )
  { int m , *a ;
     printf (" Nhập số phần tử m = " ); scanf ( "%d", &m );
/* Cấp phát và định vị bộ nhớ động */
```

```
    a = ( int*) malloc ( m* size of ( int ) ); (1)
      if ( a!= NULL ) /* cấp phát thành công */
      for ( i=0 ; i < m ; i++)
         scanf ( "%d", &a[i] );
         free (a) ; /* giải phóng vùng nhớ mãng */
 }
```

- Hàm malloc ( ) nằm trong thư viện <alloc.h> . Hàm này cung cấp số lượng byte liên tiếp từ phần bộ nhớ còn chưa sử dụng trên máy tính.

+ Ví dụ : malloc (num) = num byte và trả về con trỏ kiểu void trỏ đến địa chỉ bắt đầu của ô nhớ.

- Size of (int) : là số byte mà một biến kiểu int yêu cầu (giá trị = 2)

- (int*) : ép kiểu (type - casing) : coi địa chỉ bắt đầu là int (do malloc trỏ về con trỏ kiểu void , đặc biệt không có kiểu) , có thể nhận bất kỳ địa chỉ kiểu nào (nhờ ép kiểu).

- Muốn sử dụng hàm calloc thay cho hàm malloc => khai báo :

a = (int*) calloc (n, size of (int));

* Chú ý : Luôn gán một địa chỉ cho một con trỏ trước khi sử dụng tới nó. Nếu không biến con trỏ sẽ mang một giá trị ngẫu nhiên có thể phá huỷ chương trình.

* Cấp phát bộ nhớ động cho mãng 2 chiều m x n phần tử, m , n nhập từ bàn phím:

+ Ví dụ :

```
#include <stdio.h>
#include <alloc.h>
Void main ( )
    { int **a , m, n, OK ;
      printf ( " nhập m = " ); scanf ("%d", &m);
      printf (nhập m = n) ; scanf ( "%d", &n );
      a = ( int** ) malloc ( m*seze of (int *));
      if (a!=NULL ) /*Cấp phát thành công */
        { OK = 1 ;
          for ( i=0 ; i < m ; i++ ) } /* giá trị ban đầu cho biến con trỏ*/
          a[i] = (int*) break ;
          for ( i=0 ; i <m ; i ++ )
             { if !(OK) break ;
               a[i] = (int*) malloc ( n * size of (int));
               if ( a[i] = NULL ) OK = 0 ;
             }
             if(OK)
             { sử dụng a[0][0] , a[0][1]....., a[i][j] ...., a[m][n] }
/* giải phóng vùng nhớ cấp phát */
          if ( a!=NULL )
            { for ( i = 0 ; i < m ; i++)
               if ( a[i] ! = NULL , free ( a[i]);
                  free (a);
             }
       }
```

* Chú ý : ta xem mãng 2 chiều là mãng 1 chiều nên có thể khai báo :

a = (int*) malloc (m*n * size of (int));

VÀ A[I][J] = A[I*N + J]

Bài TậP :

1/ Làm lại các bài tập phần mãng nhưng dùng con trỏ .

2/ Dùng hàm malloc hay calloc nhập mãng n phần tử , sau đó tính tổng các phần tử và sắp xếp mãng giảm dần.

3/ Dùng hàm malloc hay calloc nhập ma trận m x n , sau đó tính tổng và sắp xếp theo tăng dần

5.4.6/ Mối liên hệ giữa con trỏ và các khái niệm quan trọng trong :

a/ Con trỏ và hàm :

- Chú ý 1 : bản thân tham số truyền cho hàm không bao giờ bị thay đổi. Nhưng nếu tham số là con trỏ thì giá trị của nó không thay đổi nhưng nội dung được chứa ở địa chỉ đó lại có thể thay đổi.
- Chú ý 2 : Truyền cho hàm một tham số hình thức được khai báo là con trỏ, và khi gọi hàm truyền cho nó một giá trị địa chỉ của biến muốn thay đổi.
- Ví dụ :giả sử tân xây dựng một hàm dùng để hoán vị biến thực, ta viết như sau :
Cách 1 :

```
#include<stdio.h>
void swap (float x , float y ) /* cách 1 sai */
{ float temp ;
  temp = x ; s<y ; y = temp;
}
main ( )
   { float a, b ; a = 10.0 ; b = 20.0 ;
       printf (" khi chưa hoán vị a = %4.0f; b = %4.0f \n" , a , b ) ;
       swap ( a , b ) ;
       printf ( " sau khi hoán vị a = %4.0f ; b = %4.0f \n" , a, b ) ;
```

- Phân tích cái sai của cách 1 của ví dụ trên :
+ Do a, b thuộc hàm main ( ). Khi khai báo sẽ dùng 2 khoảng nhớ (mỗi khoảng 3 byte) . a, b trong lời gọi hàm swap(a,b) là 2 tham số thực.
+ Các đối x, y và biến cục bộ temp được cung cấp khoảng nhớ nhưng địa chỉ khác. Do đó xx, y chỉ tồn tại ở hàm swap(_), còn a, b tồn tại suốt cả quá trình của chương trình nên hàm swap () không làm thay đổi (tức hoán vị) được giá trị của a và b => hàm viết theo cách 1 không đạt yêu cầu => yêu cầu viết lại theo cách 2.
* Cách 2 : void swap (float *x , float *y) /* viết đúng*/

```
{ float temp ;
temp = *x ; *x = *y ; * y = temp ;
}
main ( )
```

b/ Số học con trỏ (có thể thao tác số học trên nội dung con trỏ)
* Ví dụ : #include < stdio.h>

```
#include <alloc.h>
main ( )
{ #define N 3
    int *list , i ;
    list = int*) calloc ( N, size of(int));
    *list = 15 ;
    * (list + 1) = 20 ;
    *(list + 2 ) = 30 ;
      printf ( " các địa chỉ là : ");
     for ( i=o ; i < N ; i++)
     printf ("%4d",(list + i));
     printf ("\n chứa các giá trị là : ");
     for ( i=0 ; i < N ; i++)
     printf("%4d", *(list + i));
     printf("\n");
```

=> list trỏ tới một dãi bộ nhớ dài 6 byte (3*2) có các giá trị là 5,20, 30 . giá trị địa chỉ đầu là 06A
=> kết quả các địa chỉ là : 06A 06AC 06AE chứa các giá trị là : 5 20 30
c/ Con trỏ và mãng :
- Ví dụ 2 :

```
#include
main ( )
```

```
{ #define N 3
  int list [N] , i ;
  list [0] = 5 ; list [1] = 20 ; list[2]=30;
  printf ( " Các địa chỉ là : ");
  for ( i = 0 ; i < N ; i++)
  printf ( "%4p ", &list[i] );
  printf("\n chứa các giá trị là : ");
  for ( i=0; i<N ; i++)
   printf ( "%4d", list [i] ));
}
```

-Kết quả chương trình :

+ Các địa chỉ là : 163A 163C 163E

+ Chứa các giá trị là : 5 20 30

- So với ví dụ 1 thì điều khác duy nhất là giá trị địa chỉ thay đổi. Như vậy ta có thể sử dụng tên của một mãng như con trỏ và ngược lại.

=>{ list + i) = = &(list[i]) và *(list + i) = = list[i]}

d/ Con trỏ và cấu trúc :

- Ta có thể khai báo con trỏ như một biến cấu trúc, cũng như con trỏ của bấu kỳ kiểu dữ liệu nào khác. Điều này cho phép tạo một danh sách móc nối các phần tử (sẽ trình bày chương sau).

e/ Con trỏ tới hàm : dùng để chứa địa chỉ của hàm. Nên kiểu của hàm và con trỏ phải giống nhau.

Ví dụ :

```
#include <stdio.h>
Double fmax ( double x, double y ) /* hàm tính max của 2 số */
{ return ( x>y ? x:y ) ; }
/* khai báo và gán tên hàm cho con trỏ hàm */
double (*pf) (double , double ) = fmax ;
main ( )
{ printf ( " In max = % f " , pf(15.5, 20.5 ));
}
```

CHƯƠNG 6 : MỘT SỐ HÀM TRÊN CHUỖI KÝ TỰ

6.1/ Ký tự (character) :

- Ví dụ :

```
char ch , ch1 ;
ch = 'a' ; /* Đúng : ký tự chữ */
ch1 = '1' /* đúng : ký tự số */
```

- Ví dụ 2 :

```
scanf ( "%c", &ch ) ; /* gõ A và Enter */
printf ("%c", ch) ; /* In ra chữ A */
printf("%d", ch) ; /* In ra 65 là mã ASCII của A */
```

* Hàm dùng cho kiểu ký tự :

```
char ch ;
```

ch = getchar ( ) ; (Nhập 1 ký tự từ bàn phímm sau khi ấn Enter và ký tự nhập vào không hiện lên màn hinh).

putchar (ch) ; in ký tự nằm trong biến ch ra màn hình.

putch ("\n") ; đưa dấu nháy về đầu dòng.ch = getche ( ) ; Nhập 1 ksy tự từ bàn phím và ký tự nhập vào sẽ hiển thị trên màn hình.

6.2/ Chuỗi ký tự

: Ngôn ngữ C quan niệm 1 chuỗi ký tự là một mãng ký tự kết thúc bằng ký tự NULL ('\0') mã ASCII là 0.

- Ví dụ : char s[10] L E V A N A '\0'
s[0] s[1] s[3] s[4] s[5] s[7] s[8]
- Muốn nhập chuỗi ta thường dùng hàm gets(s)
- Muốn in chuỗi ta thường dùng hàm puts(s) : in xong xuống dòng.

6.3/ Một số hàm trên chuỗi : các hàm cơ bản trong thư viện string.h

a/ gets(s1) : nhập dữ liệu vào chuỗi s1.
b/ n = strlen(s1) : cho biết độ dài của chuỗi s1.
c/ n= strcmp (s1,s2) : so sánh 2 chuỗi s1,s2 (so theo mã ASCII từng ký tự).
+ nếu n>0 : s1> s2
n = 0 : s1=s2
n < 0 : s1<s2.
d/ strcpy (đích , nguồn) ; chép chuỗi nguồn vào chuỗi đích, gán chuỗi.
- Ví dụ : char [30] ;
Ten = "Nguyễn Văn Đông "; (sai).
strcpy (ten , "Nguyễn Văn Đông ");
gets (ten) : Nhập vào từ bàn phím.
e/ strcat (s1,s2) : nối s1 và s2 .
- Ví dụ : giá trị cảu s1 : " ABC" ; s2 : " ABE" => strcat(s1,s2) ; => " ABCABE";
f/ m = strncmp (s1, s2, n) ; so sánh n ký tự đầu tiên của chuỗi s1 với s2.
- Ví dụ : m = strncmp (s1, s2, 2) ; thì m = 0 do 2 ký tự đầu của chuỗi là :
+ s1 : "ABC" và s2 : " ABE" là giống nhau.
g/ strnpy (s1, s2, n) ; chép n phần tử đầu tiên của chuỗi s2 vào chuỗi s1.
- Ví dụ : strnpy (s1, "xyz", 2) ;
Puts (s1); -ă " xyC".
h/ strncat (s1,s2, n) ; nối n phần tử đầu tiên của s2 vào đuôi s1.
- Ví dụ : strncat (s1 , "xyz", 2);
Puts(s1) ; => "ABCxy".
* Chú ý : + char s1[10], s2[4]
+ strcpy (s1,"ABCDE");
+ strcpy(s2,"ABCDE"); => "ABCD" (do s[4] = "\0").
i/ Hàm strstr :
- char *p ;
p = strstr (s1,s2);
- Tìm xem chuỗi s2 có trong s1 hay không. Nếu có thì in ra cuỗi s1 tại vị trí đầu tiên mà nó thấy. Nếu không có thì in ra giá trị NULL.
- Ví dụ : s1: "abc abc ac"
s2 : "bc", s3 = "cd"
p= strstr (s1,s2);
puts (p) ; => " bc abc ac "
p = strstr (s1, s3)
Đoán thử puts(p) ; => p[(NULL)] .
k/ d= atoi (chuỗi số) ; chuyển chuỗi số thành int.
f = atof (chuỗi số) ; chuyển chuỗi số thành số thực(float).
l = atol(chuỗi số); chuyển chuỗi số thành long (nguyên 4 byte).
- Ví dụ : char s[20] ;
Gets (s) ; nhập vào s từ bàn phím chuỗi " 123.45"
d=atoi(s) ; thì d = 123.
F = atof(s); thì f = 123.45
l/ toupper (ch) ; làm thay đổi ký tự ch thành chữ Hoa.
tolower(ch); làm thay đổi ký tự ch thành chữ thường.
* Chú ý :Muốn dùng các hàm về chuỗi phải khai báo đầu chương TRÌNH #INCLUDE <STRING.H>

Bài TậP :
1/ Nhập vào chuỗi sau đó xoá các khoảng trắng xong in ra màn hình.
2/ Nhập chuỗi và xoá các khoảng trắng thừa phía trước, sau và giữa 2 từ gút lại 1 khoảng trắng.
3/ Viết hàm nhập vào một chuỗi sau đó đổi ký tự đầu mỗi từ (chữ) thành Hoa, các ký tự còn lại của 1 từ là chữ thường.
4/ Nhập chuỗi password nếu kiểm tra đúng mới cho chạy chương trình đếm số từ trong 1 chuỗi số nguyên âm, phụ âm.
5/ Đảo thức tự các từ của chuỗi. Ví dụ : s1="con mèo con cắn con chó con" đổi thành s2=" con chó con cắn con mèo con".

CHƯƠNG 7 : KIỂU CẤU TRÚC

- Khái niệm : Cấu trúc là một kiểu dữ liệu kiểu bản ghi(record) , cho phép nhiều loại dữ liệu được nhóm lại với nhau. (Khái niệm cấu trúc trong C tương tự như pascal hay Foxpro).

7.1/ Khai báo kiểu cấu trúc :

a/ struct tên _ kiểu cấu trúc
{
khai báo các thành phần của nó (các field và kiểu dữ liệu của field)
} < danh sách biến>;
- Ví dụ 1 : struct kieu HV ò-> tên kiểu cấu trúc.
{ char Ten[30] ;
int namsinh ;float diemTB ;
} HV ; (biến HV)
- Ví dụ 2 : struct kieu HV
{
các thành phần
}
struct kieu HV HV ; /* khai báo biến theo cách 2 */
b/ Dùng toán tử typedef để khai báo kiểu cấu trúc (định nghĩa kiểu mới) ;
- Ví dụ 3 : typedef struct
{ char Ten[30]
  int namsinh ;
  float diemTB ;
} kieu HV ;
kieu HV Hoc vien ;
kieu HV DSLop[20];
kieu HV Lop[ ] = { { "nguyễn văn Đông", 1980, 10.0},
{ " Trần văn Tây", 1982, 5.5},
{ " Phạm văn Nam ", 1979, 9.5}
};
- Ví dụ 4 : struct ngay{
                int ngay ;
              char Thang[10];
                int nam ;
              } ;
type struct
{ char Ten[30] ;
    ngay namsinh ; /* thành phần cấu trúc có kiểu cấu trúc*/
    float diemTB;
} kieu HV ; kieu HV HV;
* Chú ý :

- Khai báo struct phải nằm ở vị trí toàn cục của chương trình, thường sau các #include.
- Cấu trúc thường dùng để xây dựng một bảng các cấu trúc.
+ Ví dụ : kieu HV DSLop[30] ; struct kieu HV person[50];
- Có thể truyền cấu trúc như một tham số hình thức, nhưng với những cấu trúc kích thước lớn sẽ không tối ưu về thời gian lẫn độ nhớ. Khi không nên sử dụng con trỏ cấu trúc.
+ Ví dụ : struc kieu HV *HV ;

7.2/ Truy cập đến các thành phần của kiểu cấu trúc :

Tên cấu trúc. Tên thành phần
Hoặc Tên cấu trúc. Tên cấu trúc con. Tên thành phần.
- Ví dụ : + nhập vào tên, năm sinh, điểm cho biến cấu trúc học viên (ví dụ 3).

```
gets(hoc vien.ten) /* nhập " Phạm thị Bắc" và Enter */
scanf("%d ", & hoc vien.namsinh );
scanf("%f", &tam); hoc vien.diem = tam; (*)
```

+ Nhập năm sinh cho biến học viên ở ví dụ 4 :

```
scanf("%d",&hv.ngay.namsinh);
```

* Chú ý : Nếu các thành phần không phải là nguyên(int) => nhập qua trung gian như (*).

```
puts(hoc vien.ten); => " Phạm thị Bắc"
printf("%d%f", hoc vien.namsinh, hoc vien.diemTB);
```

* Lệnh gán : + Ta có thể gán 2 biến cấu trúc có cùng kiểu cho nhau :
Ví dụ : hv2=hv1;
+ Gán giá trị đầu cho biến cấu trúc và khai báo một mãng cấu TRÚC(XEM VÍ DỤ 3)

Bài TậP : viết chương trình nhập danh sách học viên gồm các trường họ tên, tuổi, điểm, và tìm kiếm trong dánhách có ai tên " Phạm Tèo " không.

Tên Tuổi điểm
HV [0] Nguyễn A 20 5.5
HV [1] Trần B 22 6.5
HV [2] Phạm Tèo 25 8.5
HV [3] Lê C 21 7.5

```
#include <stdio.h>
#define n 10
typedef struct
{ char Ten[30];
   int tuoi ;
   float diem ;
} kieu HV ;
kieu HV HV[11]
void main( )
{ int i ; float tam ; kieu HV HV;
/* nhập dữ liệu cách 1*/
for ( i = 0 ; i < n ; i++)
{ printf ("\n Nhập số liệu cho học viên thứ %d", i ) ;
   printf (" Họ và tên = " ) ; gets ( hv[i].ten);
   printf ("tuổi = "); scanf ( "%d" , &hv[i].tuoi);
   printf("điểm = "); scanf ("%f*c", &tam ); hv[i].diem = tam ;
}
/* cách 2 nhập vào biến cấu trúc và gán hv[i] = h */
for ( i = 0 ; i<n ; i++ )
{ printf("Họ và tên = "); gets(h.ten);
} hv[i] = h ;
/* tìm kiếm Phạm Tèo */
thay = 0 ; i = 0 ; /* thay = 0 : không thấy, thấy = 1 : tìm thấy */
while ((!thay)&&(i <n))
```

```
if ( strcmp(hv[i].Ten , " Phạm Tèo ") = = 0 )
{ thay = 1 ;
printf ("%s%d%f ", hv[i].ten , hv[i].tuổi, hv[i].điểm );
}
else i++ ;
if (!thay ) puts ("\n không tìm thấy Phạm Tèo !");
getch( );
}
```

Bài TậP : Viết chương trình nhập danh sách gồm na học viên gồm các thông tin như : Họ , tên, điểm pascal , điểm c, sau đó tính điểm trung bình (điemTB) = (diemC*2 + diempascal)/3 .

- Và xét kết quả đậu hay rớt theo qui ước sau :

+ nếu điểm trung bình >= 5 thì kết quả đậu.
+ Nếu điểm trung bình <5 thì kết qua rớt.
+ Nếu điểm trung bình = 4 mà phái = "Nữ" thì kết quả là đậu.

1/ in danh sách vừa nhập gồm họ tên, phái , điểm c, điểm pascal, điểm TB , kết quả .
2/ Sắp xếp giảm dần theo điểm trung bình và in ra.
3/ Nhập vào tên cần tìm và tìm trong danh sách học viên nếu không tìm thấy thì in ra học viên có tên không tìm thấy. Nếu có nhiều học viên có cùng tên cần tìm thì hãy in ra người cuối cùng được tìm thấy.
4/ Giống câu 3 nhưng in ra 2 người tìm thấy đầu tiên (nếu có nhiều người).
5/ Giống câu 3 nhưng in ra người đầu tiên và người cuối cùng (nếu có nhiều ngươi). Nên viết theo từng hàm.

7.3/ Con trỏ trỏ đến cấu trúc và địa chỉ cấu trúc :

a/ Con trỏ và địa chỉ :

- Ví dụ : typedef struct

```
{ char Ten[30] ;
int tuoi ;
float diem ;
} kieu HV ;
```

kieu HV *p , HV , lop[50] ; HS [50] (trong đó : HV là biến cấu trúc, *p : con trỏ cấu trúc dùng để lưu trữ địa chỉ cấu trúc và mãng cấu trúc) (*).

```
main ( )
/* ta có thể gán */
p = &HV ; /* Đúng do (*)*/
p = &lop[i]/*đúng do (*) */
p = lơp ; /* đúng : p = địa chỉ Lop[0] , p = &lop[0] ) do Lop = &Lop[0])
```

b/ truy cập thông qua con trỏ :

- Cách 1 : tên con trỏ -ă tên thành phần.
- Cách 2 : (*tên con trỏ).tên thành phần.
- Ví dụ : p = &HV ; p = &Lop[2] '

```
=> HV.Ten ĩ p --ă tên;
Lop[2].tuổi ĩ (p*).tuoi ĩ p -ă tuổi ;
*p = HV ;
*P = Lop[2]
```

- Giả sử cần nhập số liệu ch vùng trên thì 3 cách viết sau là tương đương :

+ (1) : gets(HV.ten)
+ (2) gets (pă ten) ĩ gets((*p).ten).
+ (3) scanf("%d",&HV.tuoi) ; ĩ scanf("%d", p -ă tuổi);

```
scanf ("%d", (*p).tuoi);
```

- Giả sử cần nhập dữ liệu cho mãng cấu trúc thì các cách viết sau đây tương đương :

+ Ví dụ : p = lop ;

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

{ gets (lop[i].tên); tương đương với :
. gets((*(lop* i)).ten);
.gets(*(p + i).ten);
.gets (p[i].ten);
.gets (p ă ten); p++ ;
.gets (*p).ten) ; p++;
- Ví dụ : làm lại bài tập mẫu nhưng sử dụng biến con trỏ :

```
#include <stdio.h>
#define n 10
typedef struct
{ char ten[30] ;
int tuoi ;
float diem ;
} kieu HV ;
main ( )
{ kieu HV hv [n], *p , h;
int i ; int thay ; float tam ; int tuổi ; p = hv;
for ( i = 0 ; i < n ; i++)
{ printf (" nhập học viên thứ %d ", i );
   printf("Họ và tên"); gets ( p ă ten);
   printf("tuổi : ") ; scanf ("%d", &tuổi); p ă tuoi = tuoi;
   printf ("diem : ") ; scanf ("%f%*c ", &tam ); p ă diem = tam;
   p++ ; printf ("%c", getchar();
}
/* nhập theo cách 2 qua biến h xong gán *p = h */
/* tìm Phạm Tèo */
thấy = 0 ; i = 0 ; p = hv ; /* để di chuyển con trỏ về đầu danh sách */
for ( i = 0 ; i < n ; i++ )
if ( strcmp(p ă ten, " Phạm Tèo " ) = = 0 )
{ thấy = 1
    printf ("%s %d%f" , p ă ten, pă tuoi, pă điểm );
    break ;
 else p++ ;
     if (!thay) puts (" không có Phạm Tèo trong danh sách ");
        getch( );
}
```

Bài TậP : làm lại bài tập trước nhưng sử dụng con trỏ.

7.4/ Cấp phát bộ nhớ động cho kiểu dữ liệu cấu trúc :

- giả sử ta cần quản lý danh sách học viên nên dùng mãng cấu trúc (cấp phát bộ nhớ tĩnh - danh sách đặc) ta phải sử dụng số học viên tối đa => thừa vùng nhớ. Để cấp phát vừa đủ sĩ số học viên như ta muốn => ta dùng phương pháp cấp phát bộ nhớ động hàm malloc hoặc calloc(.)
- Ví dụ : Nhậ danh sách n học viên gồm họ tên, điểm và sắp xếp giảm dần theo điểm.

```
#include <stdio.h> #include<conio.h> #include<alloc.h>
#include< string.h>
typedef struct
{ char ten[30] ; int diem ; char kq[5] ; } kieu HV;
kieu HV *lop , *p , tam ;
/* Hàm nhập dan sách */
void nhapDS ( int n , kieu HV lop[ ])
{ int i , diem ;
p = lop ;
for ( i = 0 ; i < n ; i++)
```

```
{ printf("nhập Họ tên người thứ %d : " , i +1 ) ; gets ( p ăten);
   printf ( " điểm = " ) ; scanf ( "%d" , &diem ) ; p ă diem = diem ;
   printf ("%c", getchar()); /* khử stdin */
   p++ ;
}
/* Hàm sắp xếp*/
void sapxep ( int n, kieu HV lop[ ])
{ int i , j ; kieu HV tam ;
    for ( i = 0 ; i < n-1 ; i++)
    for ( j=i + 1 ; j< n ; j++)
    if ( lop[i].diem < lop[j].diem )
        { tam = lop[i] ; lop[j] = lop [j] ; lop [j] = tam ; }
/* hàm in danh sách */
void inds( intn, kieu HV lop[ ] )
{ int i ;
    for ( i = 0 ; i < n ; i++ )
  { printf ("%20s%5d ", lop[i].ten,lop[i].diem );
      printf ("\n" ; /* xuống hàng */
/* chương trình chính */
void main ( )
{ int i , j, n , t, diem ;
   printf ("\n Nhập sĩ số : ") ; scanf ( "%d", &n);
  lop = (kieu HV*)malloc ( n * size of ( kieu HV) ) ; printf ("%c", getchar ());
   nhapds (n, lop ) ; sapxep ( n, lop ) ; inds ( in lop );
   getch ( );
}
```

KIỂU FILE (TậP TIN/ TệP TIN)

- Trong ngôn ngữ C , một tập tin là một khái niệm logic, được áp dụng không những đối với các tập tin trên đĩa mà cả với các terminal (bàn phím, màn hình, máy in...).
- File có 2 loại : + Text file (file văn bản).
+ Binary (nhị phân : dbf, doc, bitmap,...).
- File văn bản chỉ khác binary khi xử lý ký tự chuyển dòng (LF) (mã 10) được chuyển thành 2 ký tự CR (mã 13) và LF (mã 10) và khi đọc 2 ký tự liên tiếp CR và LF trên file cho ta một ký tự LF.
- Các thao tác trên file thực hiện thông qua con trỏ kiểu FILE. Mỗi biến FILE có 1 con trỏ lúc đầu sẽ trỏ vào phần tử đầu tiên của file. Sau mỗi thao tác đọc hay ghi dữ liệu con trỏ tự động dời xuống mẫu tin kế tiếp. Làm việc trên kiểu File thường có 3 công đoạn : mở file, nhập xuất thông trên file và đóng file.

* Một số hàm thông dụng thao tác trên file (tập tin/tệp tin) :

+ Mở file : FILE *fopen (char *filename, char *mode);

. Nếu có lỗi fp sẽ trỏ đến NULL.

+ Các mode chế độ mở file :

" r" " rt " / " rb " : mở file để đọc theo kiểu văn bản / nhị phân - file phải tồn tại trước nếu không sẽ có lỗi.

"w" "wt" / " wb " : mở (tạo) file mới để ghi theo kiểu văn bản/nhị phân - nếu file đã có nó sẽ bị xóa(ghi đè)(luôn luôn tạo mới).

"a" "at"/ "ab" : mở file để ghi bổ sung (append) thêm theo kiểu văn bản hoặc nhị phân(chưa có thì tạo mới).

+ Đóng file : int fclose (file + biến file) ;

Ví dụ :

```
Void main ( )
 { FILE *fp ;
    fp = fopen ("c:\\THUCTAP\\Data.txt", "wt" );
     if (fp = NULL )
```

```
        printf ( " không mở được file c/Thuctap\data.txt");
    else {< xử lý file > }
    fclose (fp) ; /* đóng file */
}
```
+ Làm đóng tất cả các tập đang mở : int fclose all(void) ; nếu thành công trả về số nguyên bằng tổng số các file đóng được, ngược lại trả về EOF.
+ Hàm xóa tập : remove (const + char*ten tập) ; nếu thành công cho giá trị 0, ngược lại EOF.
+ Hàm kiểm tra cuối tập : int feof(FILE*fp) : !=0 : nếu cuối tập= 0 : chưa cuối tập.
+ Hàm int putc (int ch, FILE*fp);
Hàm int fputc(int ch, FILE*fp);
Công dụng của hai hàm này :ghi một ký tự lên tập fp theo khuôn dạng được xác định trong chuỗi điều khiển dk. Chuỗi dk và danh sách đối tương tự hàm printf( ).
+ Hàm int fscanf (FILE *fp, const char *dk, ...);
Công dụng : đọc dữ liệu từ tập tin fp theo khuôn dạng (đặc tả) làm việc giống scanf( ).
*Ví dụ : giả sử có file c/data.txt lưu 10 số nguyên 1 5 7 9 8 0 4 3 15 20 . Hãy đọc các số nguyên thêm vào một mãng sau đó sắp xếp tăng dần rồi ghi vào file datasx.txt
Giải :
```
#include <stdio.h>
#include<conio.h>
#include<stdlib.h>
#define n 10
void main ( )
  { FILE *fp ; int i, j, t, a[n]
     clrscr ( ) ;
    fp = fopen (" c :\\data.txt ", "rt" ); /* mở file để đọc vào mãng */
    if (fp = NULL)
      { printf ("không mở được file ");
        exit (1);
      }
/* Sắp xếp mãng */
     for ( i=0 ; i<n-1 ; i++)
     for (j=i+1; j<n ; j++)
        if (a[i]<a[j] )
            { t = a[i] ; a[i]=a[j] ; a[j] = t ; }
     fclose (fp);
/* mở file datasx.txt để ghi sau khi sắp xếp */
    fp = fopen ("c:\\datasx.txt ", "wt");
    for ( i=0 ; i<n;i++)
    printf (fp,"%2d", a[i] );
     fclose (fp);
/* đọc dữ liệu từ file cách 2 ( tổng quát hơn ) không phụ thuộc vào n */
   i = 0 ;
while (1)
{ fscanf (fp,"%d",&a[i] ;
  i++;
  if (foef(fp) ) break ;
}
```
- Hàm int fputs (const char *s, file *fp);
Công dụng : ghi chuỗi s lên tập tin fp (dấu "\0" ghi lên tập) nếu có lỗi hàm cho eof.
- Hàm char fgets (char *s, int n , FILE *fp);
Công dụng : đọc 1 chuỗi ký tự từ tập tin fp chứa vào vùng nhớ s. Việc đọc kết thúc khi : hoặc đã đọc n-1 ký tự hoặc gặp dấu xuống DÒNG(CẮPMÃ 13 10). KHI ĐÓ MÃ 10 ĐƯỢC ĐƯA VÀO

CHUỖI KẾT QUẢ.

CáC HàM ĐọC GHI FILE KIểU CấU TRúC

- Hàm int fwrite (void *p, int size , int n , FILE*fp);

Đối : p : là con trỏ trỏ tới vùng nhớ chứa dữ liệu cần ghi.
size : là kích thước của mẫu tin theo byte.
n số mẫu tin cần ghi.
fp là con trỏ tập.

- Ví dụ : fwrite(&tam) size of(tam),1,fv); /* tam là 1 mẫu tin(record) nào đó*/

Công dụng : ghi một mẫu tin (record) kích thước sizebyte (size of (tam)) từ vùng nhớ p(&tam) lên tập fp. Hàm sẽ trả về một giá trị = số mẫu tin thực sự ghi được.

+ Hàm int fread (void*p), int size , int n, FILE *fp);

Đối : p : là con trỏ trỏ tới vùng nhớ chứa dữ liệu đọc được.
size là kích thước của mẫu tin theo byte
n : là số mẫu tin cần đọc, fp là con trỏ tập tin.
Ví dụ : fread (&tam, size of(KIEUHS) , 1, 4)>0)
Công dụng : đọc n(1) mẫu tin kích thước sizebyte (size of(tam)) từ tập tin fp chứa vào vùng nhớ p(&tam). Hàm trả về một giá trị bằng số mẫu tin thực sự đọc được.

* Ví dụ áp dụng : Nhập vào danh sách lớp gồm n học viên ("nhập vào). Thông tin về mỗi học viên gồm Họ tên, phái , điểm, kết quả. Xét kết quả theo điều kiện sau : nếu Điểm>= 5 (đậu), điểm <5 : rớt. Sau đó sắp xếp theo điểm và ghi vào tập tin c:\lop.txt. Đọc lại tập tin c:\lop.txt và xét lại kết quả nếu điểm =4 và phái là nữ sẽ đậu vf chép sang tập tin c:\ketqua.txt.

Giải :
```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<string.h>
typedef struct
{ char ten[20] ; char phai[4] ; int diem ; char kq[4] ; } KieuHV;
KieuHV *lop ,*p, tam;
/* Hàm nhập danh sách n học viên */
void nhapds ( int n, KieuHV lop[ ] )
{ int i , diem ; p = lop ;
for ( i=0; i<n ; i++ )
{ printf (" nhập họ và tên người thứ %d : " , i + 1) ; gets ( pă ten);
printf ("phái (nam/nữ ) : ") ; gets (pă phai );
printf ("nhập điểm = ") ; scanf ("%d%c*c", &diem); pă diem=diem;
if (diem>5)strcpy (p--> kq,"Đậu");
else strcpy (pă kq, "rớt " ) ; p++;
}
/* Hàm sắp xếp */
void sapxep ( int n , KieuHV lop[ ] )
{ int i , j ;
for ( i=0 ; i<n-1; i++)
for ( j=i+1 ; j<0; j++)
if (lop[i].diem< lop[j]diem )
{ tam = lop[i] ; lop[i] = lop[j] ; lop [i] = tam ;}
}
/* Hàm in danh sách */
void inds ( int n, KieuHS lop[ ] )
{ int i ;
for ( i=0 ; i<n ; i++)
printf ("\n %20s %5s%5d%5s, lop[i].ten, lop[i].phai, lop[i].diem, LOP[I].KQ );
```

/* **CHƯƠNG TRìNH CHíNH** */

```
void main ( )
{ int i , j, n, t, diem ; FILE *fp, *fr ;
   printf ("\n nhập sĩ số : ") ; scanf("%d%*c",&n);
   lop = (KieuHV*) malloc (n*size of (KieuHV));
   nhapds(n, lop) ; sapxep ( n, lop ); inds( n, lop); getch( );
   fp = fopen ( "c :\\lop.txt ", "wb");
   for ( i = 0; i<n ; i++)
   fwrite (&lop[ i], size of (KieuHV), 1, fp);
   fclose(fp);
    printf ("\n ghi dữ liệu xong ");
    printf("\n in file sau khi sắp xếp và xét kết quả lại ");
    fr = fopen ("c:\\ketqua.txt", "wb");
    while ( fread (&tam, size of ( KieuHV), 1, fp ) > 0)
        { printf ("\n %s %s%d%s", tam.ten, tam.phai, tam.diem, tam.kq);
            if (tam.diem = = 4 &&strcmp(tam.phai,"nữ")= =0 )
                strcmp(&tam.kq, "đậu");
                fwrite(&tam,size of(tam),1, fr);
       }
     fclose (fp); fclose(fr);
    printf ("\n in file ketqua.txt sau khi xét lại kết qủa ");
    fp = fopen ("c:\\ketqua.txt", "rb");
     while (fread(&tam, size of (KieuHV) , 1, fp) > 0)
        printf("\n %s%s%d%s",tam.ten,tam.phai, tam.diem,tam.kq);
         fclose (fp); getch( );
  }
```

CáC HàM XUấT NHậP NGẫU NHI?Và DI CHUYểN CON TRỏ CHỉ Vị (File position locator)

- Khi mở tệp tin để đọc hay ghi, con trỏ chỉ vị luôn luôn ở đầu tập tin (byte 0) nếu mở mode "a" (append) => con trỏ chỉ vị ở cuối tập tin.

+ Hàm void rewind (FILE*fp) : chuyển con trỏ chỉ vị của tập fp về đầu tập tin.

+ Hàm int fseek (FILE*fp, long số byte, int xp)

Đối : fp : là con trỏ tập tin; số byte : là số byte cần di chuyển.

xp " cho biết vị trí xuất phát mà việc dịch chuyển được bắt đầu từ đó.

xp = SEEK - SET hay 0 xuất phát từ đầu tập.

xp = SEEK - CUR hay 1 : xuất phát từ vị trí hiện tại của con trỏ.

xp= SEEK - END HAY 2 : xuất phát từ vị trí cuối tập của con trỏ.

+ Công dụng : hàm di chuyển con trỏ chỉ vị của tập fp từ vị trí xác định bởi xp qua một số byte bằng giá trị tuyệt đối của số byte. Nếu số byte > 0 : chuyển về hướng cuối tập ngược lại chuyển về hướng đầu tập. Nếu thành công trả về trị 0. Nếu có lỗi trả khác 0.

+ Chú ý : không nên dùng fseep trên kiểu văn bản, vì sự chuyển đổi ký tự(mã 10) sẽ làm cho việc định vị thiếu chính xác.

+ Hàm long ftell(FILE*fp) ; : cho biết vị trí hiện tại của con trỏ chỉ vị (byte thứ mấy trên tập fp) nếu không thành công trả về trị -1L.

+ Ví dụ 1: giả sử tập fp có 3 ký tự .

fseek (fp,0,SEEK-END) => ftell(fp) = 3

fseek(fp,0,2) => ftell(fp) = 3

fseek (fp,-2, SEEK-END) => ftell(fp) = 1

fseek(fp,0,SEEK -SET) => ftell(fp) = 0

fseek(fp,0, 0) =>ftell(fp) = 0

+ Ví dụ 2 : giả sử ta có tập tin c:\lop.txt chứa danh sách các học viên. Hãy đọc danh sách và sắp xếp giảm dần theo điểm sau đó ghi lại file c:\lop.txt (nối điểm)

```
#include <stdio.h>
#include<conio.h>
```

```
#include<string.h>
#define N 100
typedef struct
{ char ten[20] ; int tuoi; float diem ; } KieuHV ;
void main( )
{ KieuHV hv[N] ; t;
FILE*fp ; int i, , n ;
fp = fopen ("c:\\lop.txt ", "rat");
if (fp = =NULL)
  { printf ("không mở được file "); exit(1); }
     n = 0 ; i = 0 ;
   while (!feof (fp))
      { fread (&hv[i], size of (KieuHV), 1,fp);
         i++; n++ ;
 /* sắp xếp giảm dần theo điểm */
     for (i=0, i <n-1, i++)
     for (j=i+1; j<n, j++)
        if (hv[i].diem <hv[j].diem)
           { t =hv[i] ; hv[i] = hv[j] ; hv[j] = t }
/* ghi lên đĩa */
fseek (fp, 0, SEEK-END);
for ( i=0; i<n ; i++)
fwrite(&hv[i], size of (KieuHV), 1, fp);
}
```

CHƯƠNG 8 : TRUYỀN SỐ LIỆU CHO HÀM

1/ Truyền đối số cho hàm main( ) :

- Ví dụ : ta muốn viết một chương trình có tên là Hello.că hello.exe khi chạy trên MS-DOS ta nhập các đối số vào chương trình. Ví dụ : c:> Tom and Jerry (enter) máy sẽ in ra câu :
Chào Tom and Jerry.
- Viết chương trình trên như sau :

```
Void main ( int argc, char*argv[ ])
{
......
}
```

Trong đó :
+ argc : cho biết tổng số đối số truyền vào tính cả tên chương trình đối với ví dụ trên argc = 4. Mỗi đối số truyền vào được xem như là xâu ký tự.
+ măng argv [ ] sẽ là con trỏ , trỏ lần lượt đến các đối số.
argv[0] -->"Hello"
argv[1] --> "Tom"
argv[2] --> "and"
argv[3] --> "Jerry"

```
void main ( int argc, char *argv[ ])
{ int i ;
   printf("Chào !");
   for ( i=1 ; i < argc ; i++)
   printf("%s", argv[ i]);
}
```

2/ Truyền cấu trúc cho hàm - Hàm trên các cấu trúc :

- Chương trình gọi nhập học viên(HV).
- Chương trình bị gọi void nhap (struct Kiểu HV HV[ ])
Hoặc void nhập (Kiểu HV HV[ ])
- Ví dụ : nhập danh sách lớp :

```
#include<stdio.h>
#include<conio.h>
#include<string.h> #define N 100
typedef struct
{ char ten[20] ; int tuoi ; float diem ; } kieu HV
/* khai báo hàm nhập dữ liệu*/
void nhap ( int n , Kieu HV HV [ ] )
{int i ; float t;
    for ( i = 0; i< n ; i++)
        { printf ( " Nhập hv thử %d ", i++); scanf(....) }
/* chương trình chính */
main ( )
    { Kieu HV hv[n];
      nhap ( n, hv);
    }
```

* Hàm có thể trả về giá trị cấu trúc hoặc con trỏ cấu trúc :
+ Ví dụ : Hàm Kieu HV *ptim (char*ten, KieuHV HV[ ] , int n) : có tác dụng tìm trong danh sách n học viên trong mãng HV[ ] người có tên và hàm trả về con trỏ, trỏ tới người tìm được hoặc trả về NULL nếu không tìm thấy .
+ Hàm Kieu HV tim(char*ten, KieuHV HV[ ], int n); : cũng với mục đích như hàm trên nhưng trả về giá trị của một cấu trúc.
+ Ví dụ :

```
#include"stdio.h"
#include"conio.h"
#include"string.h"
typedef struct
{ char ten[20] ; int tuổi ; float điểm } Kieu HV ;
Kieu HV *ptim ( char*ten, KieuHV HV[ ] , int n );
Kieu HV tim ( char *ten, Kieu HV hv[ ] , int n );
main( )
{ Kieu HV *p, ds[100],h ; int i, h, n ; char ten[20] ; float diem;
clrscr ( );
printf("\n Số người n = " ) ; scanf ("%d *c ",&n);
for ( i=0 ; i<n ; ++i)
    { printf("\họ tên "); gets(h.Tên) ; /* tự viết lấy*/
      ds[i] = h ;
    }
/* tìm kiếm 1 theo ten dùng hàm ptim*/
while (1)
  { printf ("\n Họ tên người cần tìm"); gets (tên);
    if ((p =ptim ( ten, ds, n)) = = NULL)
        printf("\n không tìm thấy ");
    else
     indanhsach(*p);
  }
/* tìm kiếm theo tên dùng hàm tim*/
while(1)
  { printf("\n Họ tên cần tìm "); gets(ten);
```

```
     if ( tim ( ten, ds, n).ten[0] = = 0 )
          printf("\n Không tìm thấy");
     else indanhsach ( tim ( ten, ds, n));
  }
Kieu HV *ptim ( char*ten, Kieu HV hv[ ] , int n)
{ int i ;
for ( i= 0 ; i< n ; ++i)
   if ( strcmp ( ten, hv[i].ten= = )return (&hv[ i]);
        return (NULL);
}
Kieu HV tim (char*ten, Kieu HV hv[ ] , int n)
{ int i ; HV tam ;
  tam.ten[0]=0;
  for ( i=0 ; i<n ; ++i)
    if(strcmp(ten,hv[i].ten = = )) return (hv[i]);
       return ( tam);
}
void indanhsach (Kieu HV p)
{ printf("\n Họ tên % tuổi % điểm %f", p.ten, p.tuổi, p.diêm);
}
```

CHƯƠNG 9 : DANH SÁCH LIÊN KẾT (MÓC NỐI)

- Danh sách liên kết : Nếu sử dụng mãng để quản lý danh sách sẽ rất tốn kèm và cứng nhắc trong thao tác ă khắc phục = danh sách liên kết.
- Danh sách liên kết gồm các phần tử . Mỗi phần tử có 2 vùng chính : vùng dữ liệu và vùng liên kết. Vùng liên kết là một hay nhiều con trỏ, trỏ đến các phần tử trước hoặc sau nó tùy thuộc vào yêu cầu của công việc.
- Khai báo danh sách liên kết :

```
Typedef struct Kieu du lieu
  { <khai báo phần tử dữ liệu >;
   Kiểu dữ liệu < các con trỏ >;
 } Kiểu dữ liệu ;
```

- Dùng typedef struct kieu du lieu định nghĩa kiểu dữ liệu mới. Trong kiểu dữ liệu này có 2 phần, phần đầu tiên là phần khai báo các trường, phần thứ 2 là các con trỏ, trỏ đến chính kiểu dữ liệu đó, dòng cuối cùng là cần thiết để các con trỏ được phép khai báo chính là kiểu dữ liệu mà các con trỏ đó là thành phần.
- Ví dụ :

```
typedef struct sinhvien
{ char hoten[30] ;
   int diem ;
   struct sinhvien *tiep ;
} sinhvien ;
sinhvien *head ; / con trỏ đặc biệt luôn trỏ tới đầu danh sách*/
```

- Mỗi một phần tử có một con trỏ, trỏ đến phần tử tiếp theo. Riêng phần tử cuối cùng con trỏ sẽ trỏ đến một kiểu đặc biệt : Kiểu NULL(nghĩa là con trỏ đó không trỏ đến một phần tử nào cả). Ban đầu con trỏ danh sách (head) được gán bằng NULL.
- Để cấp phát bộ nhớ, ta cần kiểm tra xem có đủ không (tránh rối loạn chương trình)
- Ví dụ :

```
#define size of (sinhvien)
sinhvien *sv
sv=NULL ;
```

```
if ((sv = (sinhvien*)malloc (size sv) = = NULL)
  { printf (" không đủ bộ nhớ RAM \n");
    getch ( );
    return ;
  }
```
- Hàm size of (kiểu phần tử) cho kích thước của kiểu phần tử bằng byte.

sv là con trỏ phụ cần thiết cho các thao tác trong chương trình. size sv có kích thước bằng vùng nhớ một phần tử (nhờ sử dụng hàm size of( )). Cần gán sv = NULL để phòng sinhvien đang trỏ vào một phần tử của danh sách. Khi thêm vào, chương trình sẽ tự động tìm vị trí thích hợp của phần tử mới. Do trong ngôn ngữ C không định nghĩa kiểu string như trong PASCAL, nên càn dùng hàm so sánh strcmp(st1,st2). Hàm này cho kết quả kiểu int sau khi so sánh st1 và st2 như sau :

< 0 nếu st1 < st2.
= 0 nếu st1 = st2.
> 0 nếu st1 >st2.

- Các trường hợp xảy ra khi thêm một phần tử vào một danh sách :

+ Nếu phần tử mới ở đầu danh sách , cần sửa lại con trỏ head.
+ Nếu đã có phần tử đó, phải lựa chọn liệu có ghi đè lên không?
+ Các trường hợp khác cần sửa lại con trỏ như sau : Giả sử cần chèn phần tử mới vào giữa phần tử 1 và 2 ta có :

.....

- Ví dụ : Chương trình qủan lý sinh viên gồm : thêm, bớt, duyệt danh sách, tìm kiếm phần tử

```
/**********************
Chương trình qủan lý sinh viên
************************/
#include <stdio.h>
#include<conio.h>
#include< stdlib.h>
#include<type.h>
#include<string.h>
void taomenu( )
void themsv ( );
void timkiem ( );
void loaibo( );
void danhsach( );
void vitrihv (char st[ ], int d ); /* tìm vị trí hợp lý */
void lietke ( );
#define sizesv size of (sinhvien)
typedef(truct sinhvien)
   { char hoten[30] ;
     int diem ;
     struct sinhvien *tiep ;
   } sinhvien ;
 sinhvien *head;
 sinhvien *sv ;
 void main ( )
    { clrscr ( );
      gotoxy(1,12);
      printf (" chương trình quản lý danh sách sinh viên (DSLK)\n");
      getch ( ) ;
      taomenu ( );
   } /* kết thúc hàm main ( ) */
```

```
void taomenu ( )
   { char ch ;
    do
     { clrscr( );
       printf(" thêm sinh viên tìm kiếm loại bỏ liệt kê Quit \n");
       ch = toupper (getch());
       switch (ch)
            { case "I' :themsv() ;break ;
              case ' I ' : timkiem( ) ; break ;
              case ' L; : loaibo( ) ;break ;
              case ' D' : lietke( ) ; break ;
              case ' Q ' : exit (1) ; break ;
              default : break ;
           }
  }
while ( ch!= 'Q');
}
void themsv ( )
{ char tensv [30] ; int diem ;
   clrscr ( );
    printf(" thêm sinh viên vào danh sách \n");
    gotoxy(1,10) ; printf(" họ và tên : "); gets( tensv);
     printf("điểm :"); scanf("%d", &diem);
     vitrihv ( tensv, diem);
}
void vitrihv( char st [ ] ) int d )
{ sinhvien *find = NULL , *next = NULL; int kq ; char ch ;
   sv = NULL ;
   if ((sv = ( sinhvien*) malloc ( sizesv )) = = NULL)
        { printf(" không đủ bộ nhớ \n") ; getch( ) ; return }
       strcpy ( svă hoten, st);
      svă diem = d ;
/* nếu danh sách ban đầu là rỗng */
    if ( head = = NULL)
          { head = sv ; headă tiep = NULL ; }
    else
{ /* tìm vị trí mới của phần tử trong danh sách */
         find = head ; next = find ;
          while ((find!=NULL) &&((kq=strcmp(findă hoten, sv ă hoten))< 0)
                { next = find ; find = findătiep ;}
                if ( kq = = 0)
                   { printf("sinh viên đã có trong danh sách . Ghi đè (Y/N) ? \n");
                     ch = getch( ); ch = toupper (ch);
                     if (ch = 'N')
                        { free(sv) ; return ; }
                     else
                          find --> diem = d ;
                       free (sv) ;
                   return ;
                }
/* nếu phần tử thêm vào đầu danh sách */
             if (find = = head )
```

```
                { sv ă tiep = head ; head = sv ; }
            else { sv ă tiep = find ; next ă tiep = sv ; }
    }
}
void timkiem( )
   { char tensv[30] ; int kq ; clrscr ( );
      printf(" tên sinh viên cần tìm :") ; gets(tensv);
      if((tensv !=" " ) && (head1 = NULL))
          { sv = head ;
            while ((sv! = NULL) &&((kq = strcmp(svăhoten, tensv))< 0)
            sv = sv ă tiep ;
            if(kq = = 0);
                 printf (" Họ và tên %s điểm %d", svăhoten, svă diem);
           else printf (" không có sinh viên %s \n", tensv);
          }
       getch( ) ;
   }
void loaibo( )
{ char tensv [30] ; int kq ; sinhvien *next ;
  clrscr ( )
  printf ( " tên sinh viên cần loại bỏ :"); scanf("%s", tensv );
  iF((tensv!=NULL) && (head!= NULL))
    { sv = head ; next = sv ;
      while ((kq = strcmp (svă hoten, tensv )) < 0)
        { next = sv ; sv = sv ă tiep ; }
        if ( kq = = 0)
          { if ( sv = = head )
              { head = head ă tiep ; free (sv) ; return ; }
                next ă tiep = sv ă tiep ;
     free(sv);
              }
       else
          { printf (" không có tên %s \n", tensv );
     }
  }
}
void lietke( )
{ clrscr( )
sv = head ;
while ( sv! = NULL)
    { printf(" Họ và tên : %s \n" , svăhoten );
       printf(" điểm : %d \n\n", svă diem);
       sv = svătiep ;
    }
    getch( );
  }
```

Bài tập : Hãy lập trình quản lý sinh viên sử dụng cấu trúc danh sách. Mỗi phần tử cấu trúc như sau : họ và tên, điểm.

Yêu cầu : - In danh sách sinh viên có điểm >= 7.

- Sắp xếp theo điểm .
- Loại bỏ sinh viên nào đó (nhập tên vào).