



**HUTECH**  
Đại học Công nghệ Tp.HCM

**KHOA CÔNG NGHỆ THÔNG TIN**

**Bài giảng:**

# KỸ THUẬT LẬP TRÌNH

Bài 4:

## CON TRỎ



*C Ngôn ngữ lập trình số 1 thế giới*

**Giảng viên: Th.S Dương Thành Phết**

**Email: phetcm@gmail.com**

**Website: <http://www.thayphet.net>**

**Mobile: 0918158670**



## MỤC TIÊU

- ✓ Trình bày được khái niệm về con trỏ;
- ✓ Thực hiện được khai báo và sử dụng biến kiểu con trỏ;
- ✓ Xử lý được các phép toán trên mảng một chiều, hai chiều theo kiểu con trỏ;
- ✓ Thực hiện được các giải thuật trên mảng 1 chiều, 2 chiều như tìm kiếm, sắp xếp, thêm phần tử, xóa phần tử...theo kiểu con trỏ;
- ✓ Thao tác được con trỏ với kiểu dữ liệu có cấu trúc.



# NỘI DUNG

1. Khái niệm về địa chỉ ô nhớ và con trỏ
2. Khai báo và sử dụng biến con trỏ
3. Các phép toán trên con trỏ
4. Sử dụng con trỏ để cấp phát và thu hồi bộ nhớ
5. Con trỏ và mảng một chiều
6. Con trỏ và mảng hai chiều
7. Con trỏ với kiểu có cấu trúc (struct)

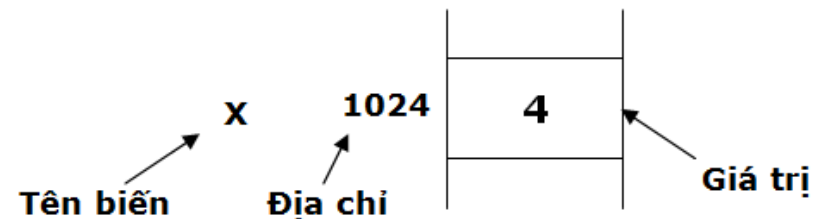


## 4.1. KHÁI NIỆM VỀ ĐỊA CHỈ Ô NHỚ VÀ CON TRỎ

### Biến tĩnh:

- ✓ Các biến có kiểu cơ sở, kiểu cấu trúc là biến tĩnh.
- ✓ Khi khai báo biến tĩnh, một lượng ô nhớ cho các biến này sẽ được cấp phát cố định.
- ✓ Biến tĩnh sẽ tồn tại trong suốt thời gian thực thi chương trình.

**Ví dụ : `int x=4` ;**



### Một số hạn chế khi sử dụng các biến tĩnh:

- ✓ Cấp phát ô nhớ dư, gây ra lãng phí.
- ✓ Cấp phát ô nhớ thiếu, chương trình thực thi bị lỗi.



## 4.1. KHÁI NIỆM VỀ ĐỊA CHỈ Ô NHỚ VÀ CON TRỎ

### **Biến động:**

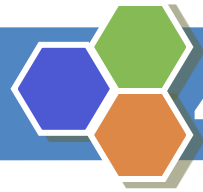
- ✓ Khắc phục những hạn chế của biến tĩnh
- ✓ Chỉ phát sinh trong quá trình thực hiện chương trình (không phát sinh đầu chương trình).
- ✓ Khi chạy chương trình, vùng nhớ và địa chỉ vùng nhớ được cấp phát cho biến có thể thay đổi.
- ✓ Sau khi sử dụng xong có thể giải phóng để tiết kiệm bộ nhớ.



## 4.1. KHÁI NIỆM VỀ ĐỊA CHỈ Ô NHỚ VÀ CON TRỎ

### **Biến con trỏ:**

- ✓ Vì các biến động không có địa chỉ nhất định nên không thể truy cập được.
- ✓ Nên ngôn ngữ C cung cấp loại biến để khắc phục là biến con trỏ (pointer) với các đặc điểm:
  - Con trỏ không chứa dữ liệu mà chỉ chứa địa chỉ của ô nhớ chứa dữ liệu.
  - Kích thước của biến con trỏ không phụ thuộc vào kiểu dữ liệu, cố định là 2 byte.



## 4.2. KHAI BÁO VÀ SỬ DỤNG BIẾN CON TRỎ

### 4.2.1. Khai báo biến con trỏ

Cú pháp:      <Kiểu> \* <Tên con trỏ>

Ví dụ 1: `int a, b, *pa, *pb;`

→ Khai báo 2 biến a,b có kiểu int và 2 biến pa, pb là 2 biến con trỏ trỏ đến 2 biến chứa giá trị kiểu int.

Ví dụ 2: `float f, *pf;`

→ Khai báo biến f kiểu float và biến pf là con trỏ , trỏ đến biến chứa giá trị kiểu float

## 4.2. KHAI BÁO VÀ SỬ DỤNG BIẾN CON TRỎ

### 4.2.2. Các thao tác trên con trỏ

#### ✓ Gán địa chỉ cho biến con trỏ:

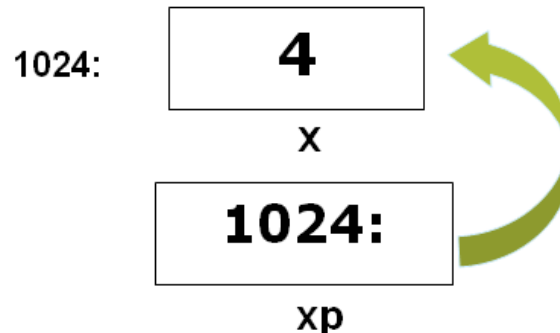
- Toán tử & dùng để định vị con trỏ đến địa chỉ của một biến.
- Cú pháp: **<Tên biến con trỏ>=&<Tên biến>**

→ Gán địa chỉ của biến cho con trỏ.

Ví dụ: **int \*xp , x = 4 ;**

**xp = &x**

→ Gán địa chỉ của biến x cho con trỏ xp.







## 4.2. KHAI BÁO VÀ SỬ DỤNG BIẾN CON TRỎ

### 4.2.2. Các thao tác trên con trỏ

Truy cập nội dung biến con trỏ:

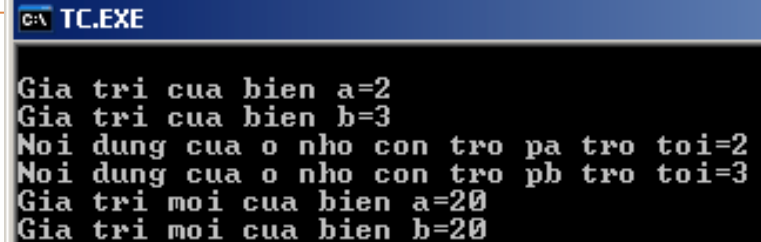
- ✓ Để truy cập nội dung của ô nhớ mà con trỏ chỉ tới, cú pháp: **\*<Tên biến con trỏ>**
- ✓ Vậy **\*<Tên biến con trỏ>** là một biến có kiểu được mô tả trong phần khai báo biến con trỏ.

Ví dụ:

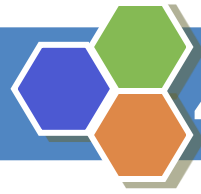
```
int x=100; *ptr;
ptr= &x; //ptr chưa giá trị là địa chỉ biến x
int y= *ptr; //y=100
```

## 4.2. KHAI BÁO VÀ SỬ DỤNG BIẾN CON TRỎ

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int a,b,*pa,*pb;
    a= 2 ;
    b= 3 ;
    printf("\nGia tri cua bien a=%d \nGia tri cua bien b=%d ",a,b);
    pa=&a;
    pb=&b;
    printf("\nNoi dung cua o nho con tro pa tro toi=%d",*pa);
    printf("\nNoi dung cua o nho con tro pb tro toi=%d ",*pb);
    *pa=20; /* Thay doi gia tri  cua *pa*/
    *pb=20; /* Thay doi gia tri  cua *pb*/
    printf("\nGia tri moi cua bien a=%d \n
    Gia tri moi cua bien b=%d ",a,b); /* a, b thay doi theo*/
    getch();
}
```



```
TC.EXE
Gia tri cua bien a=2
Gia tri cua bien b=3
Noi dung cua o nho con tro pa tro toi=2
Noi dung cua o nho con tro pb tro toi=3
Gia tri moi cua bien a=20
Gia tri moi cua bien b=20
```



## 4.3. CÁC THAO TÁC TRÊN CON TRỎ

### 4.3.2. Tăng giảm địa chỉ:

- ✓ Cộng (+), trừ (-) 1 con trỏ với số nguyên N;
- ➔ Trả về là 1 con trỏ trỏ đến vùng nhớ cách vùng nhớ của con trỏ hiện tại N phần tử.

Ví dụ: `float x[30], *px;      px = &x[10];`

- ➔ `px` là con trỏ float trỏ đến phần tử `x[10]`.
  - `px+i` trỏ đến phần tử `x[10+i]`
  - `px-i` trỏ đến phần tử `x[10-i]`.
- ✓ Phép trừ 2 con trỏ cùng kiểu sẽ trả về 1 giá trị nguyên là khoảng cách (số phần tử) giữa 2 con trỏ.
- ✓ *Con trỏ NULL*: là con trỏ không chứa địa chỉ nào cả  
 Lưu ý: Không thể cộng 2 con trỏ với nhau.



## 4.3. CÁC THAO TÁC TRÊN CON TRỎ

### 4.3.3. Truy cập bộ nhớ:

**Nguyên tắc:** Con trỏ float truy nhập 4 byte. Con trỏ int 2 byte. Con trỏ char 1 byte.

Ví dụ float \*pf ;

int \*pi ;

char \*pc ;

*Khi đó :*

- Nếu pf trỏ đến byte 10001, thì \*pf biểu thị vùng nhớ 4 byte liên tiếp từ byte 10001 → 10004.
- Nếu pi trỏ đến byte 10001, thì \*pi biểu thị vùng nhớ 2 byte liên tiếp từ byte 10001 → byte 10002



## 4.3. CÁC THAO TÁC TRÊN CON TRỎ

### 4.3.4. Phép so sánh

Cho phép so sánh 2 con trỏ cùng kiểu:

- ✓  $p1 < p2$ : Địa chỉ  $p1$  trỏ tới thấp hơn địa chỉ  $p2$  trỏ tới
- ✓  $p1 = p2$ : Địa chỉ  $p1$  trỏ tới bằng địa chỉ  $p2$  trỏ tới
- ✓  $p1 > p2$ : Địa chỉ  $p1$  trỏ tới cao hơn địa chỉ  $p2$  trỏ tới.



## 4.4. SỬ DỤNG CON TRỎ C.PHÁT/T.HỒI BỘ NHỚ

- ✓ Để cấp phát bộ nhớ động, sử dụng các hàm trong thư viện `stdlib.h` hoặc `alloc.h`.
  - `malloc`
  - `calloc`
  - `realloc`
  - `new`
- ✓ Khi dùng xong phải giải phòng thu hồi bộ nhớ.
  - khi cấp phát dùng hàm `malloc`, `calloc`, `realloc`, thì khi giải phóng sử dụng hàm `free`.
  - Khi cấp phát sử dụng toán tử `New`, khi giải phóng bộ nhớ ta phải dùng toán tử `delete`



## 4.4. SỬ DỤNG CON TRỎ C.PHÁT/T.HỒI BỘ NHỚ

### 4.4.1. Các hàm cấp phát vùng nhớ

#### ✓ Hàm malloc

Cú pháp: `void *malloc( size_t n);`

- Hàm xin cấp phát vùng nhớ cho n phần tử, mỗi phần tử có kích thước là size\_t.
- Nếu thành công hàm trả về địa chỉ đầu vùng nhớ được cấp phát.
- Khi không đủ vùng nhớ hàm trả về trị NULL.



## 4.4. SỬ DỤNG CON TRỎ C.PHÁT/T.HỒI BỘ NHỚ

```
#include <stdio.h>
#include <string.h>
#include <alloc.h>
#include <process.h>
void main( )
{
    char *str; /* allocate memory for string */
    str = (char *) malloc(10);
    if( str==NULL){
        printf("Not enough memory to allocate buffer\n");
        exit(1); /* terminate program if out of memory */
    }
    strcpy(str, "Hello"); /* copy "Hello" into string */
    printf("String is %s\n", str); /* display string */
    free(str); /* free memory */
}
```





## 4.4. SỬ DỤNG CON TRỎ C.PHÁT/T.HỒI BỘ NHỚ

### 4.4.1. Các hàm cấp phát vùng nhớ

#### ✓ Hàm calloc

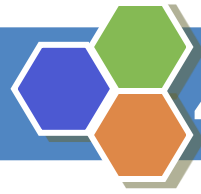
Cú pháp: `void *calloc(size_t nitems, size_t size);`

- ➔ Cấp phát vùng nhớ `nitems*size` byte.
- ➔ Nếu thành công hàm trả về địa chỉ đầu vùng nhớ được cấp phát.
- ➔ Khi không đủ bộ nhớ hàm trả về NULL.

## 4.4. SỬ DỤNG CON TRỎ C.PHÁT/T.HỒI BỘ NHỚ

```
#include <stdio.h>
#include <alloc.h>
#include <string.h>
void main(void)
{
    char *str = NULL;
    /* allocate memory for string */
    str = (char *) calloc(10, sizeof(char));
    strcpy(str, "Hello"); /* copy "Hello" into string */
    printf("String is %s\n", str); /* display string */
    free(str); /* free memory */
}
```

*Hàm calloc cấp phát vùng nhớ và khởi tạo tất cả các bit trong vùng nhớ mới cấp phát về 0. Hàm malloc chỉ cấp phát vùng nhớ.*



## 4.4. SỬ DỤNG CON TRỎ C.PHÁT/T.HỒI BỘ NHỚ

### 4.4.1. Các hàm cấp phát vùng nhớ

#### ✓ Hàm realloc

Cú pháp: `void* realloc(void *ptr, unsigned size);`

- ptr: trỏ đến vùng nhớ đã được cấp phát trước đó.
- size: là số byte cần cấp phát lại.

→ Hàm thay đổi kích thước vùng nhớ đã cấp phát trước đó. Phần dữ liệu trên vùng nhớ cũ được chuyển đến vùng nhớ mới.

Ví dụ: `int a, *pa;`

```
pa = (int*) malloc (10) ;
```

```
/*Xin cấp phát vùng nhớ có kích thước (10 x 2 ) byte*/
```

```
pa = realloc (pa, 16);
```

```
/* Xin cấp phát lại vùng nhớ có kích thước 16 x2 byte*/
```



## 4.4. SỬ DỤNG CON TRỎ C.PHÁT/T.HỒI BỘ NHỚ

### 4.4.2. Thu hồi bộ nhớ động

- ✓ Khi sử dụng con trỏ nếu cấp phát bộ nhớ thì bắt buộc phải trả lại bộ nhớ.
- ✓ Để thu hồi bộ nhớ ta có thể dùng hàm free hoặc toán tử delete.

Lưu ý :

- ✓ Khi cấp phát bộ nhớ nếu dùng hàm **malloc**, **calloc**, **realloc**, thì khi giải phóng dùng hàm **free**.
- ✓ Khi cấp phát dùng toán tử **New**, thì khi giải phóng dùng toán tử **delete**



## 4.4. SỬ DỤNG CON TRỎ C.PHÁT/T.HỒI BỘ NHỚ

### 4.4.3. Toán tử sizeof

- ✓ Toán tử sizeof cho biết kích thước (tính theo byte) của một kiểu dữ liệu hay một đối tượng dữ liệu.
- ✓ Kiểu dữ liệu có thể là kiểu chuẩn hay kiểu dữ liệu được định nghĩa.
- ✓ Đối tượng dữ liệu bao gồm tên biến, tên mảng, biến struct, ...
- ✓ Khai báo :
 

sizeof (<đối tượng dữ liệu>)  
 sizeof (<kiểu dữ liệu>)



## 4.4. SỬ DỤNG CON TRỎ C.PHÁT/T.HỒI BỘ NHỚ

### 4.4.3. Toán tử sizeof

```
typedef float KieuThuc;  
struct Sinhvien  
{  
    char masv[8];  
    char mamh[5];  
    int lanthi;  
    float diem;  
} sv;  
sizeof (int);           //cho trị 2  
sizeof (KieuThuc);     //cho trị 4  
sizeof (sv);           //cho trị 19
```



## 4.5. CON TRỎ VÀ MẢNG 1 CHIỀU

“Giữa mảng và con trỏ có một sự liên hệ rất chặt chẽ. Những phần tử của mảng có thể được xác định bằng chỉ số trong mảng, bên cạnh đó chúng cũng có thể được xác lập qua biến con trỏ.”

### 4.5.1. Truy cập các phần tử mảng theo dạng con trỏ

Quy tắc:

$\&\langle \text{Tên mảng} \rangle[0] \leftrightarrow \langle \text{Tên mảng} \rangle$

$\&\langle \text{Tên mảng} \rangle [\langle \text{Vị trí} \rangle] \leftrightarrow \langle \text{Tên mảng} \rangle + \langle \text{Vị trí} \rangle$

$\langle \text{Tên mảng} \rangle[\langle \text{Vị trí} \rangle] \leftrightarrow *(\langle \text{Tên mảng} \rangle + \langle \text{Vị trí} \rangle)$



## 4.5. CON TRỎ VÀ MẢNG 1 CHIỀU

```
#include <stdio.h>
#include <conio.h>
void NhapMang (int a[], int n) /* Nhập mảng bình thường*/
{
    for( int i=0;i<n ;i++)
    {
        printf("Phan tu thu %d: ",i);
        scanf("%d",&a[i]);
    }
}
void NhapContro (int *a, int n) /* Nhập mảng theo dạng con trỏ*/
{
    for(i=0;i<n ;i++)
    {
        printf("Phan tu thu %d: ",i);
        scanf("%d",a+i);
    }
}
```





## 4.5. CON TRỎ VÀ MẢNG 1 CHIỀU

### 4.5.2. Truy cập từng phần tử đang được quản lý bởi con trỏ theo dạng mảng

$\langle \text{Tên biến} \rangle [\langle \text{Vị trí} \rangle] \iff *(\langle \text{Tên biến} \rangle + \langle \text{Vị trí} \rangle)$

$\&\langle \text{Tên biến} \rangle [\langle \text{Vị trí} \rangle] \iff (\langle \text{Tên biến} \rangle + \langle \text{Vị trí} \rangle)$

Trong đó:

$\langle \text{Tên biến} \rangle$  là biến con trỏ,

$\langle \text{Vị trí} \rangle$  là 1 biểu thức số nguyên.

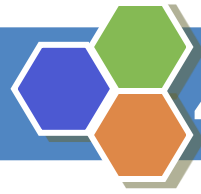
## 4.5. CON TRỎ VÀ MẢNG 1 CHIỀU

### 4.5.2. Truy cập từng phần tử đang được quản lý bởi con trỏ theo dạng mảng

```
#include <stdio.h>
#include <alloc.h>
#include <conio.h>
void main()
{
    int *a;
    a = (int*)malloc (10);
    for(i=0;i<10;i++)
        a[i] = 2*i;
    printf("Truy cap theo kieu mang: ");
    for(i=0;i<10;i++)
        printf("%d ",a[i]);
    printf("\nTruy cap theo kieu con tro: ");
    for(i=0;i<10;i++)
        printf("%d ",*(a+i));
    getch();
}
```

C:\ TC.EXE

Truy cap theo kieu mang: 0 2 4 6 8 10 12 14 16 18  
Truy cap theo kieu con tro: 0 2 4 6 8 10 12 14 16 18



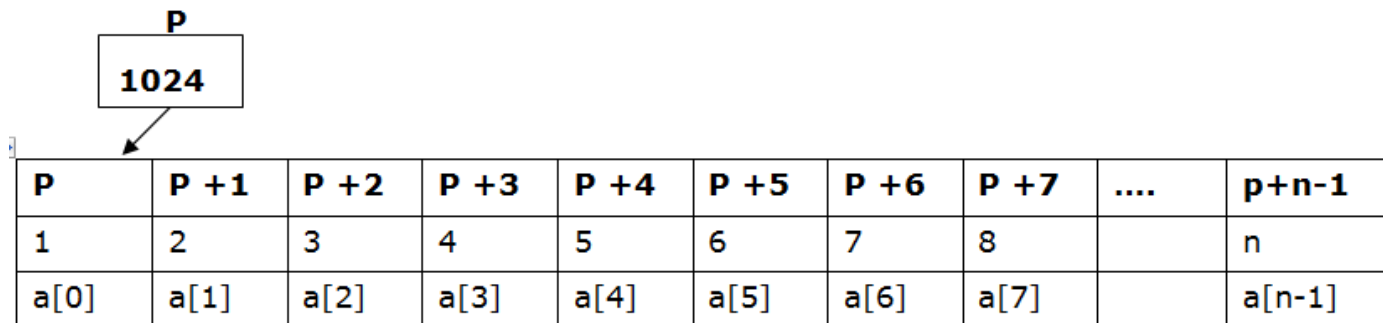
## 4.5. CON TRỎ VÀ MẢNG 1 CHIỀU

**Bài toán 1:** Dùng phương pháp con trỏ. Viết chương trình:

- ✓ Nhập mảng số nguyên gồm  $n$  phần tử ( $n \leq 100$ ).
- ✓ Xuất ra mảng số nguyên vừa nhập.
- ✓ Tính tổng các phần tử có trong mảng.

`int * p;` // khai báo biến con trỏ p để quản lý mảng

`p = ( int *) malloc ( n );` // xin cấp phát ( $n \times 2$ ) byte cho p.





## 4.5. CON TRỎ VÀ MẢNG 1 CHIỀU

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
void Nhapsort ( int &n ){
    do{
        printf ("Nhap so phan tu: ");
        scanf( " %d ", &n);
        if (n<=0)
            printf ("Nhap sai, nhap lai " );
    } while(n<=0);
}
void Capphat ( int *p , int n){
    p = ( int*) calloc ( n , sizeof ( int) );
    if (p==NULL) {
        printf(" ko du bo nho");
        getch( );
        exit(1);
    }
}
```

```
void Nhapmang (int * a , int n){
    for (int i=0 ; i<n ; i++) {
        printf(" nhap a[%d]:", i );
        scanf ( " %d ", (a+ i));
    }
}
void Xuatmang ( int * a, int n){
    for ( int i=0 ; i<n ; i++ )
        printf ("%4d", *( a+i ) );
}
void main()
{
    int *a, n;
    Nhapsort (n);
    Capphat(a,n);
    Nhapmang( a , n);
    Xuatmang(a , n);
    free(a);
}
```



## 4.6. CON TRỎ VÀ MẢNG 2 CHIỀU

**Bài toán 2:** Dùng phương pháp con trỏ. Viết chương trình:

- ✓ Nhập vào một ma trận số nguyên gồm m dòng, n cột (m,  $n \leq 100$  ).
- ✓ Xuất ra ma trận số nguyên vừa nhập.
- ✓ Tính tổng các phần tử có trong ma trận.

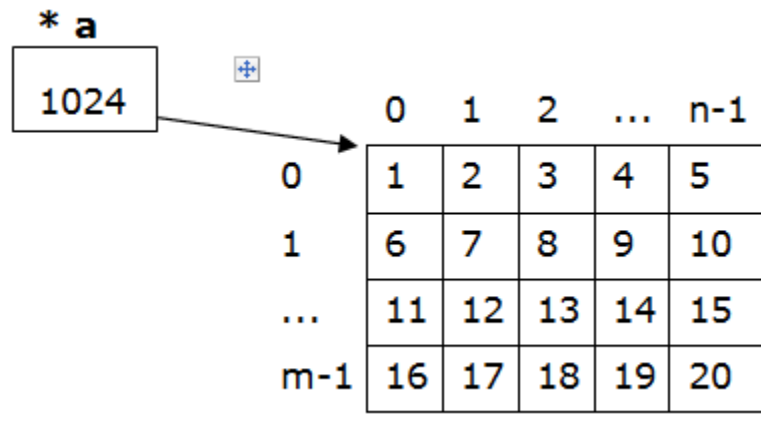


## 4.6. CON TRỎ VÀ MẢNG 2 CHIỀU

### Cách 1:

`int *a ;` // khai báo con trỏ a để quản lý ma trận

`a = (int *) malloc (m*n));` // cấp phát bộ nhớ



A	a +1	a +2	a +3	a +4	a +5	a +6	....	a+mxn-1
1	2	3	4	5	6	7		mxn
a[0][0]	a[0][1]	a[0][2]	a[0][3]	a[0][4]	a[0][5]	a[0][6]		a[m-1][n-1]

Để truy cập đến phần tử a [i][j]: `*(a + i*n + j);`



## 4.6. CON TRỎ VÀ MẢNG 2 CHIỀU

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
void Nhapdongcot ( int  &m , int  &n )
{
    do{
        printf ( " nhap so dong: ");
        scanf( " %d ", &m);
        if (m<=0 || m >100)
            printf ("Nhap sai, nhap lai:");
    } while(m<=0 || m>100);
    do{
        printf ( " nhap so cot: " );
        scanf ( " %d ", &n);
        if( n<=0 || n>100 )
            printf ( "Nhap sai, nhap lai:");
    } while(n<=0 || n>100);
}
```

```
void Capphat ( int *a , int m ,int n)
{
    a=( int*) calloc (m*n, sizeof (int));
    if(a==NULL) {
        printf(" ko du bo nho");
        getch( );
        exit(1);
    }
}
void Nhapmang (int * a , int m , int n)
{
    for (int i=0 ; i<m ; i++)
        for ( int j=0 ; j<n ; j++)
        {
            printf("Nhap a[%d][%d]:",i,j);
            scanf ( " %d ", (a+ i*n + j ) );
        }
}
```



## 4.6. CON TRỎ VÀ MẢNG 2 CHIỀU

```
void Xuatmang ( int * a, int m , int n)
{
    for ( int i=0 ; i<m ; i++ )
    {
        for ( int j=0 ; j<n ; j++)
            printf ("%4d", *( a+i *n +j ) );
        printf("\n");
    }
}

long Tinhtong ( int * a, int m , int n)
{
    for ( int i=0 ; i<m ; i++ )
        for ( int j=0 ; j<n ; j++)
            s = s+ *( a+i *n +j ) ;
    return s;
}
```

```
void main()
{
    int *a, m,n;
    Nhapdongcot(m,n);
    Capphat(a,m,n);
    Nhapmang( a , m, n);
    Xuatmang(a , m, n);
    long kq=Tinhtong(a,m,n);
    printf("Tong cac phan tu:%ld", kq);
    free(a);
}
```





## 4.6. CON TRỎ VÀ MẢNG 2 CHIỀU

### Cách 2:

`int **a ;` // khai báo con trỏ a để quản lý ma trận

`a = (int **) calloc (m,sizeof(int*));` // cấp phát bộ nhớ

- ✓ Con trỏ \*a quản lý các phần tử dòng 0: `a[0][0]`, `a[0][1]`, `a[0][2]`,....`a[0][n-1]`
- ✓ Con trỏ \*(a+1) quản lý các phần tử dòng 1: `a[1][0]`, `a[1][1]`,`a[1][2]`,....`a[1][n-1]`
- ✓ .....
- ✓ Con trỏ \*(a+m-1) quản lý các phần tử dòng m-1:  
`a[m-1][0]`, `a[m-1][1]`, `a[m-1][2]`,....`a[m-1][n-1]`

<code>*a →</code>	<code>a[0][0]</code>	<code>a[0][1]</code>	...	<code>a[0][n-1]</code>
<code>*(a+1) →</code>	<code>a[1][0]</code>	<code>a[1][1]</code>	...	<code>a[1][n-1]</code>
.....	.....	.....	.....	.....
<code>*(a+m-1) →</code>	<code>a[m-1][0]</code>	<code>a[m-1][1]</code>	...	<code>a[m-1][n-1]</code>

Con trỏ `**a` quản lý mảng con trỏ: `*a`, `*(a+1)`, `*(a+2)`..., `*(a+m-1)`

<code>** a →</code>	<code>* a</code>	<code>* (a+1)</code>	...	<code>*(a+m-1)</code>
---------------------	------------------	----------------------	-----	-----------------------

## 4.6. CON TRỎ VÀ MẢNG 2 CHIỀU

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
void Nhapdongcot ( int  &m , int  &n )
{
    do{
        printf ( " nhap so dong: ");
        scanf( " %d ", &m);
        if (m<=0 || m >100)
            printf ( "Nhap sai, nhap lai:");
    } while(m<=0 || m>100);
    do{
        printf ( " nhap so cot: " );
        scanf ( " %d ", &n);
        if( n<=0 || n>100 )
            printf ( "Nhap sai, nhap lai:");
    } while(n<=0 || n>100);
}
```

```
void Capphat ( int **a , int m ,int n)
{
    a = ( int** ) calloc (m,sizeof( int* ));
    if(a==NULL){
        printf(" ko du bo nho");
        getch( );
        exit(1);
    }
    for(int i=0;i<m;i++){
        a[ i]=(int *) calloc(n,sizeof(int));
        if ( a[i] == NULL){
            printf(" ko du bo nho");
            getch( );
            exit(1);
        }
    }
}
```

## 4.6. CON TRỎ VÀ MẢNG 2 CHIỀU

```
void Nhapmang (int** a , int m , int n)
{
    for (int i=0 ; i<m ; i++)
        for ( int j=0 ; j<n ; j++) {
            printf(" nhap a[%d][%d]:", i , j);
            scanf ( " %d ", (*(a+i) + j ) );
        }
}

void Xuatmang ( int ** a, int m , int n)
{
    for ( int i=0 ; i<m ; i++ ) {
        for ( int j=0 ; j<n ; j++)
            printf ("%4d", *( *( a+i ) +j ) );
        printf("\n");
    }
}

void myfree ( int **a , int m )
{
    for ( int i=0 ; i<m ; i++)
        free ( a[ i ] ) ;
    free (a);
}
```

```
long Tinh tong ( int ** a, int m , int n)
{
    long s=0;
    for ( int i=0 ; i<m ; i++ )
        for ( int j=0 ; j<n ; j++)
            s= s+ *( *( a+i ) +j ) ;
    return s;
}

void main()
{
    int **a, m,n;
    Nhapdongcot(m,n);
    Capphat(a,m,n);
    Nhapmang( a , m , n);
    Xuatmang(a , m , n);
    long kq=Tinh tong(a,m,n);
    printf("Tong cac phan tu:%ld", kq);
    myfree(a);
    getch();
}
```



## 4.7. CON TRỎ VỚI KIỂU CÓ CẤU TRÚC

### Ví dụ 1:

```
typedef struct DIEM
{
    char masv[8];
    char mamh[5];
    int lanthi;
    float diem;
};
DIEM *p, x;
/* p là con trỏ kiểu struct và x là biến struct */
p=&x; /* con trỏ p chứa địa chỉ của biến x */
```

- ✓ Để truy cập đến các thành phần của struct thông qua con trỏ ta dùng 1 trong 2 cách sau:



## 4.7. CON TRỎ VỚI KIỂU CÓ CẤU TRÚC

### Cách 1:

Cú pháp:

**<tên con trỏ> -> <tên thành phần>**

Ví dụ :

**printf("%f ", p ->diem);**

### Cách 2:

Cú pháp:

**<(\*<tên con trỏ>).<tên thành phần>**

Ví dụ :

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

## 4.7. CON TRỎ VỚI KIỂU CÓ CẤU TRÚC

**Ví dụ 2:**Viết chương trình nhập/xuất điểm.

```
#include "stdio.h"
#include "conio.h"
struct DIEM{
    char masv[8];
    char mamh[5];
    int lanthi;
    float diem;
};
void main(){
    struct DIEM x,*p;
    float tam;
    p=&x;
    printf("\nNhap ma so sinh vien :");
    gets(p->masv);
```

```
printf("Nhap ma mon hoc : ");
gets(p->mamh);
printf("Lan thi :");
scanf("%d ", &p->lanthi);
printf("diem :");
scanf("%f ", &tam);
p->diem=tam;
printf("\nKet qua:");
printf("\nMa sSV:%s", (*p).masv);
printf("\nMa MH: %s", (*p).mamh);
printf("\nLan thi :%d", (*p).lanthi);
printf("\ndiem :%.2f ", (*p).diem);
getch();
}
```

## 4.7. CON TRỎ VỚI KIỂU CÓ CẤU TRÚC

### Truyền structure sang hàm

```
#include "stdio.h"
#include "conio.h"
typedef struct DIEMTHI {
    char masv[8];
    char mamh[5];
    int lanthi;
    float diem;
};
void Nhap( DIEMTHI *px);
void Xuat( DIEMTHI x);
void main() {
    DIEMTHI x;
    printf("\nNhap diem thi");
    Nhap(&x);
    printf("\nXuat diem thi");
    Xuat(x);
    getch();
}
```

```
void nhap( DIEMTHI *px) {
    float tam;
    printf("\nMa sinh vien :");
    gets(px->masv);
    printf("Ma mon hoc:");
    gets(px->mamh);
    printf("Lan thi:");
    scanf("%d%c", &px->lanthi);
    printf("Diem thi:");
    scanf("%f%c", &tam);
    px->diem=tam;
}

void xuat( DIEMTHI x) {
    printf("\nMa SV :%s", x.masv);
    printf("\nMa MH:%s", x.mamh);
    printf("\nLan thi:%d", x.lanthi);
    printf("\nDiem thi:%.2f", x.diem);
}
```



## 4.8. BÀI TẬP SỬ DỤNG CON TRỎ

### Bài tập 1.

1. Hàm nhập mảng 1 chiều số nguyên gồm n phần tử ( $0 < n < 100$ )
2. Hàm nhập mảng 1 chiều số thực gồm n phần tử ( $0 < n < 100$ )
3. Viết hàm xuất mảng số nguyên n phần tử vừa nhập ở trên
4. Viết hàm xuất mảng số thực n phần tử vừa nhập ở trên
5. Tính tổng các phần tử có trong mảng
6. Tính tổng các phần tử chẵn có trong mảng
7. Tính tổng các phần tử lẻ có trong mảng
8. Tính tổng các phần tử nguyên tố có trong mảng
9. Tìm phần tử chẵn đầu tiên có trong mảng
10. Tìm phần tử lẻ đầu tiên có trong mảng





## 3.8. BÀI TẬP

### Bài tập 2.

11. Tìm phần tử nguyên tố đầu tiên có trong mảng
12. Tìm phần tử chẵn cuối cùng có trong mảng
13. Tìm phần tử chính phương cuối cùng có trong mảng
14. Tìm phần tử lớn nhất có trong mảng
15. Đếm số phần tử chẵn có trong mảng
16. Đếm số phần tử lớn nhất có trong mảng
17. In ra vị trí của phần tử lớn nhất đầu tiên có trong mảng
18. Thêm một phần tử vào đầu mảng.
19. Thêm một phần tử vào cuối mảng.
20. Thêm một phần tử vào vị trí x trong mảng.

## 3.8. BÀI TẬP

### Bài tập 3.

21. Xóa phần tử chẵn đầu tiên.
22. Xóa tất cả các phần tử lớn nhất trong mảng
23. Sắp xếp mảng tăng dần
24. Hàm nhập mảng 2 chiều các số nguyên gồm  $m \times n$  ( $0 < m, n < 100$ )
25. Hàm nhập mảng 2 chiều các số thực gồm  $m \times n$  ( $0 < m, n < 100$ )
26. Hàm xuất mảng 2 chiều các số nguyên  $m \times n$  phần tử
27. Hàm xuất mảng 2 chiều các số thực  $m \times n$  phần tử
28. Tính tổng các phần tử có trong mảng
29. Tính tổng các phần tử chẵn có trong mảng
30. Tính tổng các phần tử lẻ có trong mảng



## 3.8. BÀI TẬP

### Bài tập 4.

31. Tính tổng các phần tử nguyên tố có trong mảng
32. Tính tổng các phần tử đường chéo chính (ma trận vuông)
33. Tính tổng các phần tử đường chéo phụ (ma trận vuông)
34. Tính tổng các phần tử nằm trên đường biên có trong mảng
35. Tìm phần tử chẵn đầu tiên có trong mảng
36. Tìm phần tử lẻ đầu tiên có trong mảng
37. Tìm phần tử nguyên tố đầu tiên có trong mảng
38. Tìm phần tử chẵn cuối cùng có trong mảng
39. Tìm phần tử chính phương cuối cùng có trong mảng
40. Tìm phần tử lớn nhất có trong mảng



## 3.8. BÀI TẬP

### Bài tập 5.

41. Đếm số phần tử chẵn có trong mảng
42. Đếm số phần tử lớn nhất có trong mảng
43. In ra vị trí của phần tử lớn nhất đầu tiên có trong mảng
44. Tính tổng các phần tử nằm trên một dòng
45. Tìm dòng có tổng lớn nhất
46. Sắp xếp mảng tăng dần



## 3.8. BÀI TẬP

### **Bài tập 6.**

- 47. Nhập chuỗi
- 48. Xuất chuỗi
- 49. Đếm số ký tự ' a ' có trong chuỗi
- 50. Cắt khoảng trắng có trong chuỗi
- 51. Đếm khoảng trắng trong chuỗi
- 52. Đếm số từ có trong chuỗi
- 53. Sắp xếp chuỗi tăng dần
- 54. Nhập 3 chuỗi, xuất chuỗi theo thứ tự từ điển



## 3.8. BÀI TẬP

### **Bài tập 7.**

- 55. Viết hàm Nhập vào một phân số
- 56. Viết hàm Nhập vào một dãy phân số
- 57. Viết hàm xuất một phân số
- 58. Viết hàm xuất một dãy phân số
- 59. Viết hàm tìm phân số lớn nhất trong dãy phân số
- 60. Viết hàm tính tổng các phân số có trong dãy



## 3.8. BÀI TẬP

### Bài tập 8.

61. Viết hàm nhập một sinh viên. Thông tin: Họ, Tên, mã số SV, ngày tháng năm sinh, giới tính, lớp, điểm toán, điểm lý, điểm tin.
62. Viết hàm xuất dữ liệu một sinh viên với thông tin vừa nhập ở trên
63. Viết hàm nhập danh sách sinh viên, lưu trên mảng một chiều
64. Viết hàm xuất danh sách sinh viên
65. Xuất thông tin của sinh viên có mã sinh viên là “ X “
66. Xuất danh sách sinh viên thuộc ngành công nghệ thông tin
67. Xuất danh sách sinh viên Nữ thuộc ngành công nghệ thông tin
68. Sắp xếp danh sách sinh viên theo tên
69. Sắp xếp danh sách sinh viên theo MSSV
70. Sắp xếp danh sách sinh viên theo điểm toán



**HUTECH**  
Đại học Công nghệ Tp.HCM

**KHOA CÔNG NGHỆ THÔNG TIN**

**Bài giảng:**

# KỸ THUẬT LẬP TRÌNH

## HẾT BÀI 4 CON TRỎ



*C Ngôn ngữ lập trình số 1 thế giới*