Bộ môn Công nghệ phần mềm
Khoa Công nghệ thông tin
Trường Đại học Khoa học Tự nhiên

KỸ THUẬT LẬP TRÌNH

ThS. Đặng Bình Phương
dbphuong@fit.hcmus.edu.vn

DỮ LIỆU KIỂU CON TRỎ (NÂNG CAO)

Nội dung

1. Con trỏ cấp 2
2. Con trỏ và mảng nhiều chiều
3. Mảng con trỏ
4. Con trỏ hàm

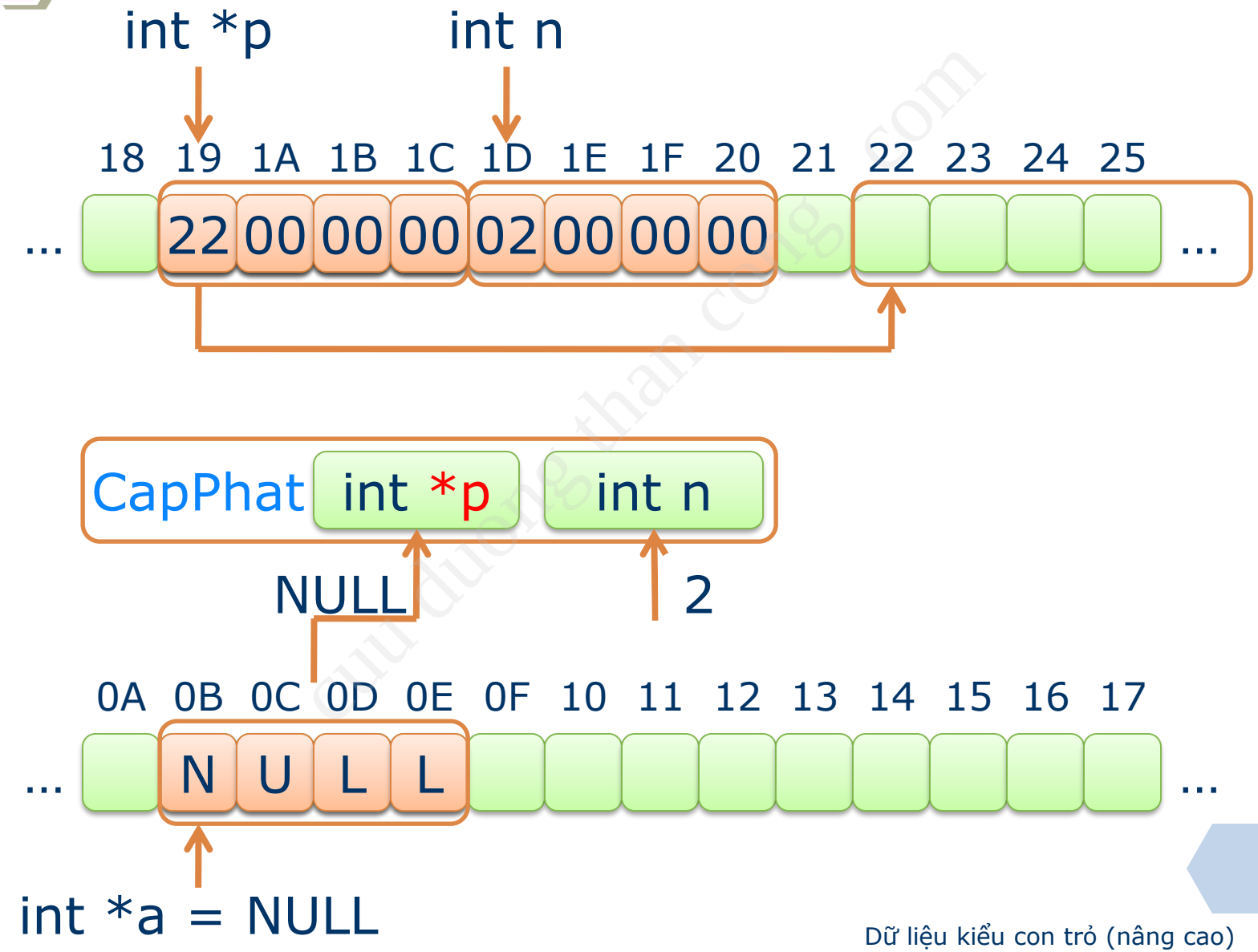
Con trỏ cấp 2 (con trỏ đến con trỏ)

Đặt vấn đề

```
void CapPhat(int *p, int n)
{
	p = (int *)malloc(n * sizeof(int));
}
void main()
{
	int *a = NULL;
	CapPhat(a, 2);
	// a vẫn = NULL
}
```

Làm sao thay đổi giá trị của con trỏ (không phải giá trị mà nó trỏ đến) sau khi gọi hàm?

Con trỏ cấp 2



Con trỏ cấp 2

Giải pháp

- Sử dụng tham chiếu int *&p (trong C++)

```
void CapPhat(int *&p, int n)
{
	p = (int *)malloc(n * sizeof(int));
}
```

- Không thay đổi trực tiếp tham số mà trả về

```
int* CapPhat(int n)
{
	int *p = (int *)malloc(n * sizeof(int));
	return p;
}
```

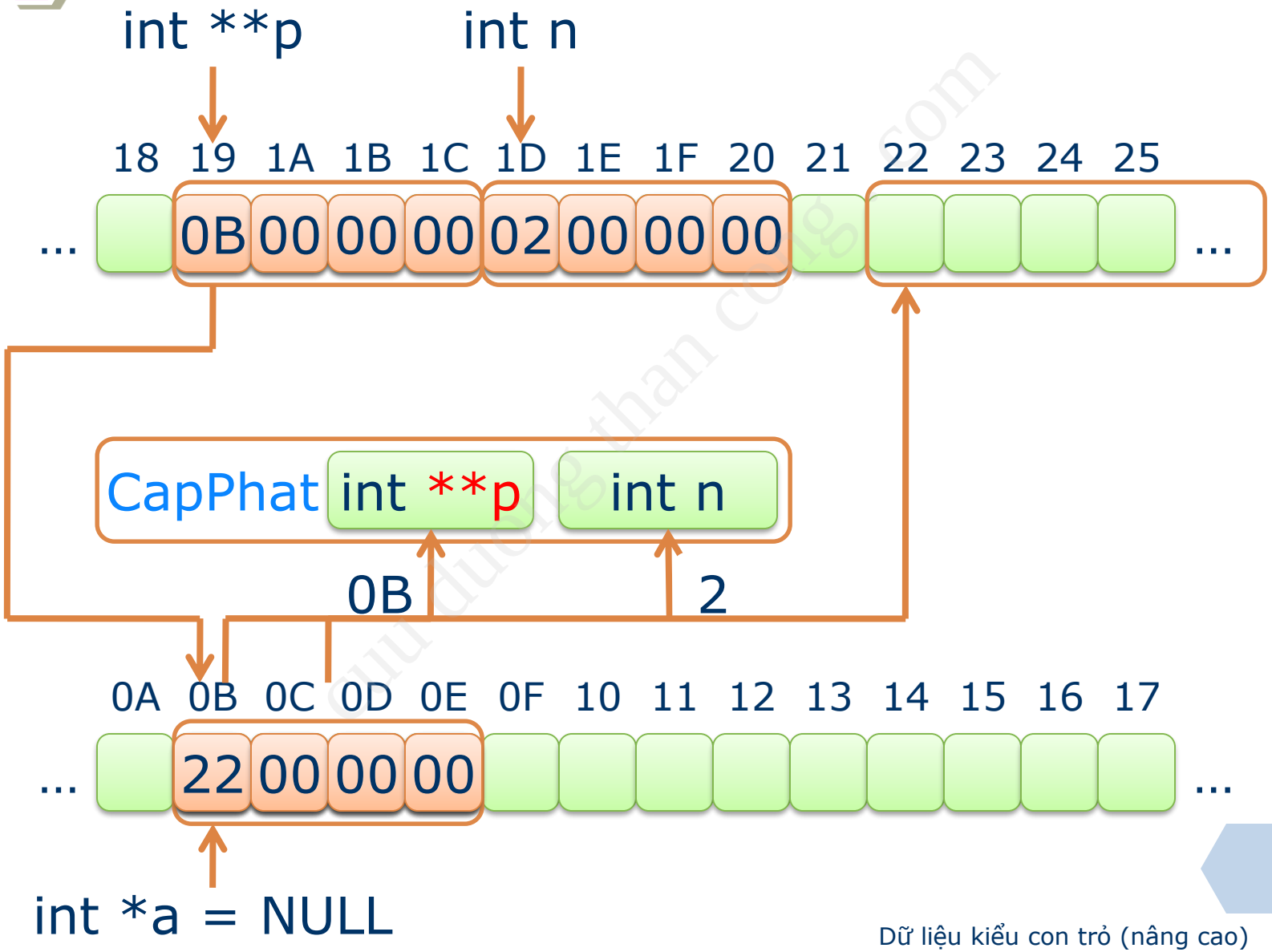
Con trỏ cấp 2

Giải pháp

- Sử dụng con trỏ p trỏ đến con trỏ a này. Hàm sẽ thay đổi giá trị của con trỏ â gián tiếp thông qua con trỏ p.

```
void CapPhat(int **p, int n)
{
	*p = (int *)malloc(n * sizeof(int));
}

void main()
{
	int *a = NULL;
	CapPhat(&a, 4);
}
```

Con trỏ cấp 2



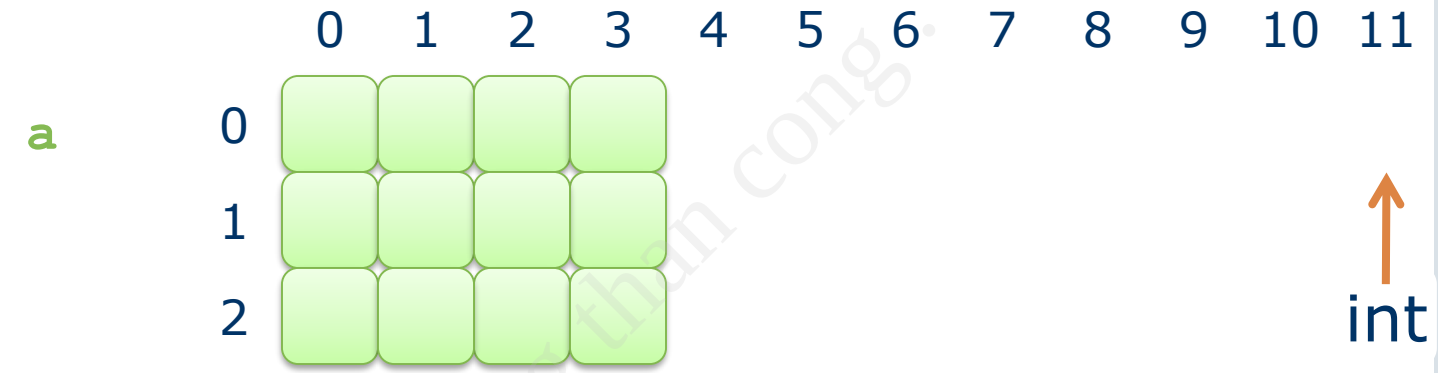
Con trỏ cấp 2

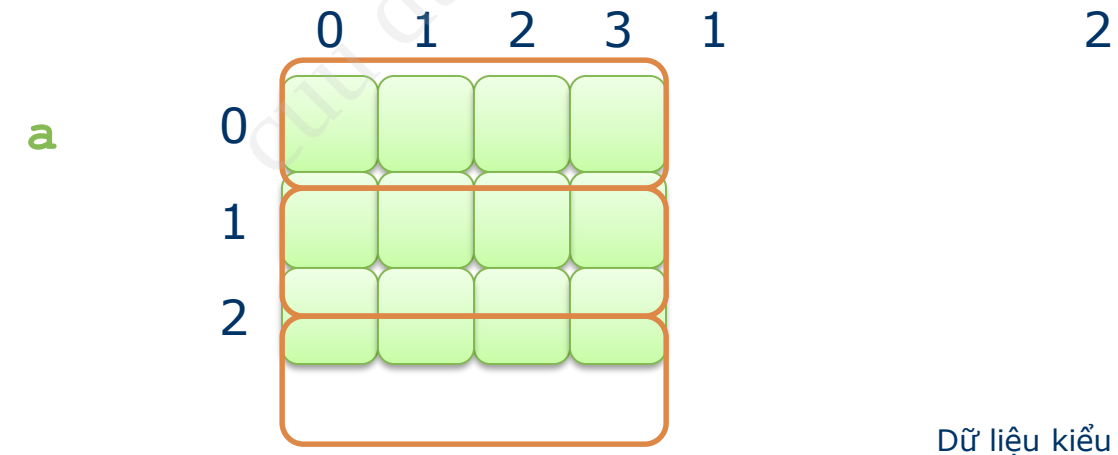
Lưu ý

```
int x = 12;
int *ptr = &x;                  // OK
int k = &x; ptr = k;            // Lỗi

int **ptr_to_ptr = &ptr;        // OK
int **ptr_to_ptr = &x;          // Lỗi

**ptr_to_ptr = 12;              // OK
*ptr_to_ptr = 12;               // Lỗi

printf("%d", ptr_to_ptr);       // Địa chỉ ptr
printf("%d", *ptr_to_ptr);      // Giá trị ptr
printf("%d", **ptr_to_ptr);     // Giá trị x
```

Con trỏ và mảng 2 chiều

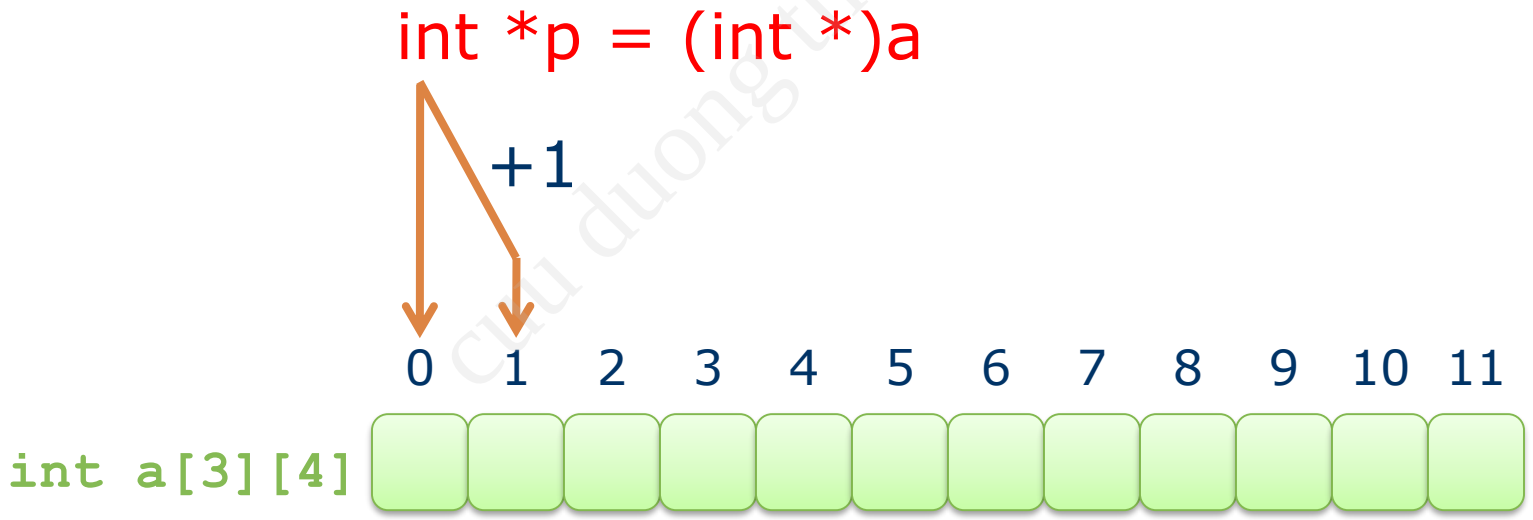
```
int a[3][4];
```





Con trỏ và mảng 2 chiều

- Hướng tiếp cận 1
  - Các phần tử tạo thành mảng 1 chiều
  - Sử dụng con trỏ int * để duyệt mảng 1 chiều



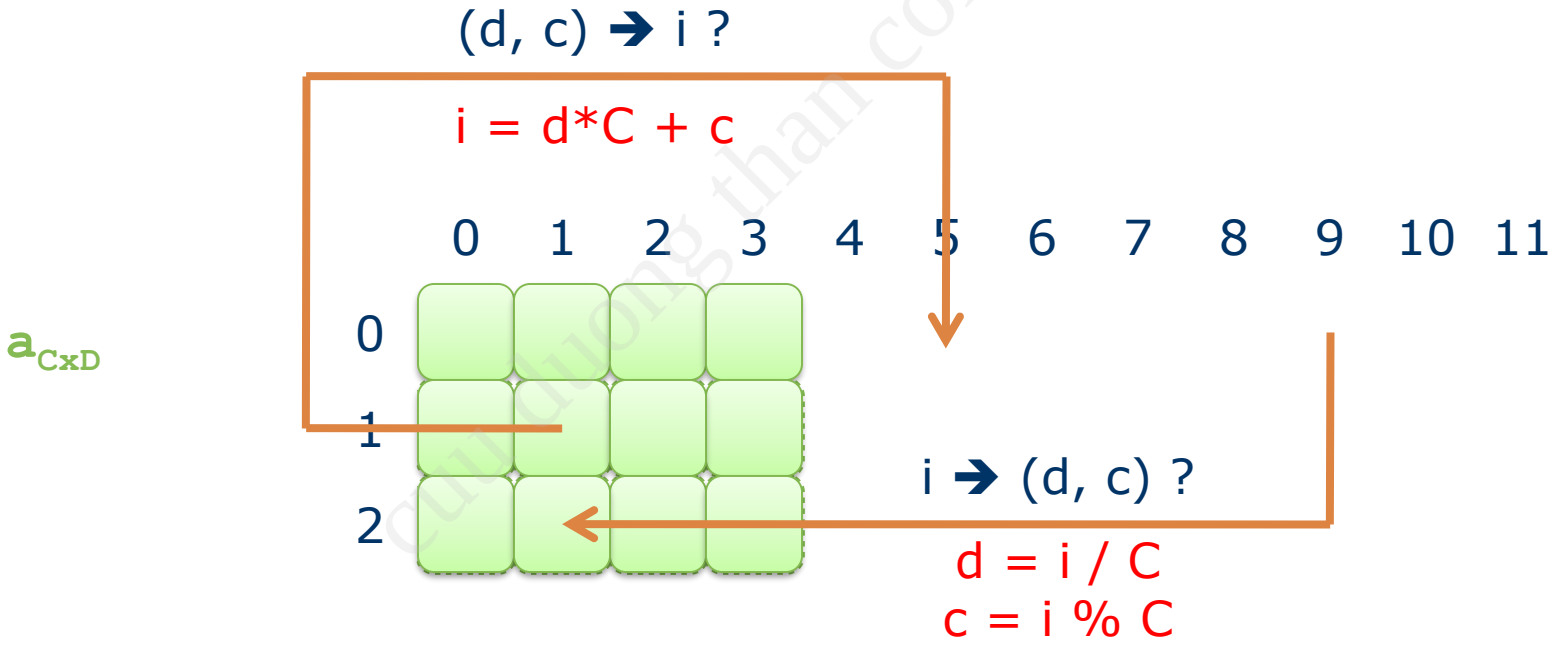
Hướng tiếp cận 1

Nhập / Xuất theo chỉ số mảng 1 chiều

```
#define D 3
#define C 4
void main()
{
	int a[D][C], i;
	int *p = (int *)a;
	for (i = 0; i < D*C; i++)
	{
		printf("Nhap phan tu thu %d: ", i);
		scanf("%d", p + i);
	}

	for (i = 0; i < D*C; i++)
		printf("%d ", *(p + i));
}
```

Hướng tiếp cận 1

- Liên hệ giữa chỉ số mảng 1 chiều và chỉ số mảng 2 chiều



Hướng tiếp cận 1

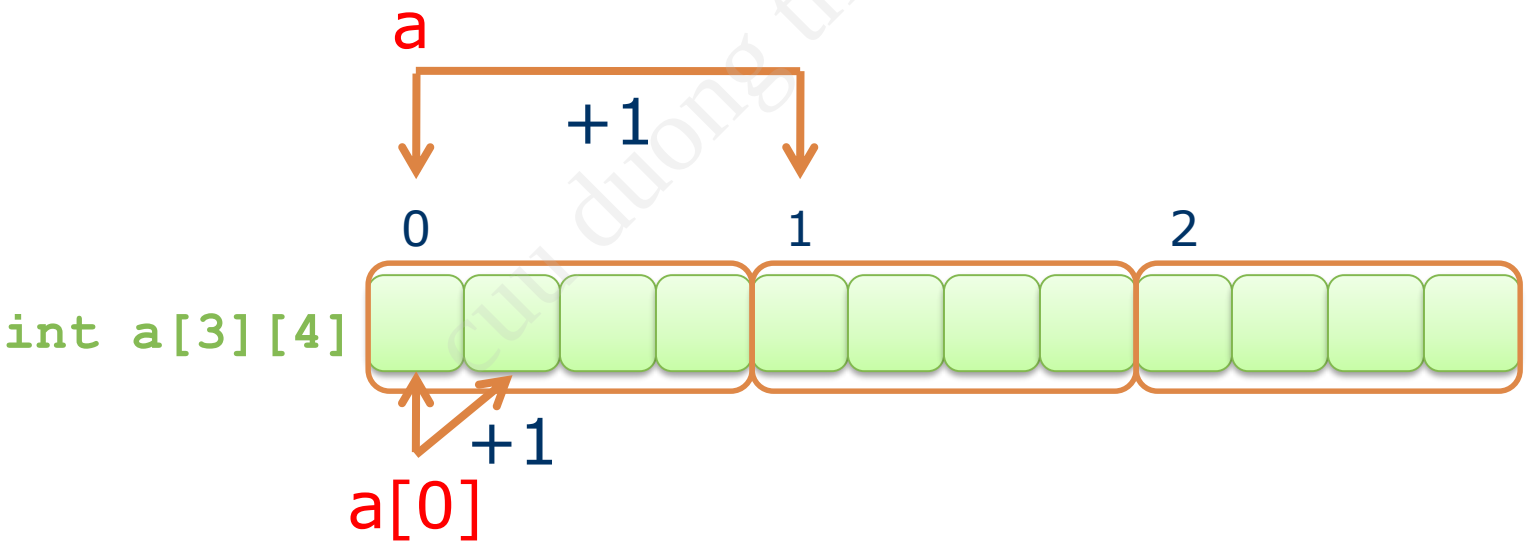
Nhập / Xuất theo chỉ số mảng 2 chiều

```
int a[D][C], i, d, c;
int *p = (int *)a;

for (i = 0; i < D*C; i++)
{
      printf("Nhap a[%d][%d]: ", i / C, i % C);
      scanf("%d", p + i);
}
for (d = 0; d < D; d++)
{
      for (c = 0; c < C; c++)
            printf("%d ", *(p + d * C + c));// *p++
      printf("\n";
}
```

Con trỏ và mảng 2 chiều

Hướng tiếp cận 2

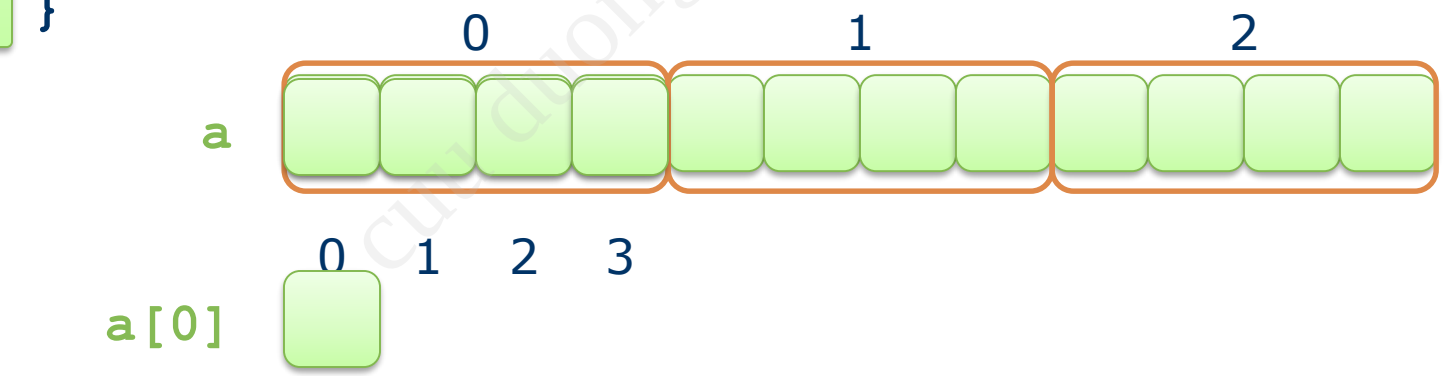
- Mảng 1 chiều, mỗi phần tử là mảng 1 chiều
  - a chứa a[0], a[1], … ➔ a = &a[0]
  - a[0] chứa a[0][0], a[0][1], … ➔ a[0] = &a[0][0]



Hướng tiếp cận 2

Kích thước của mảng

```
void main()
{
        int a[3][4];
        printf("KT của a = %d", sizeof(a));
        printf("KT của a[0] = %d", sizeof(a[0]));
        printf("KT của a[0][0] = %d", sizeof(a[0][0]));
}
```



Hướng tiếp cận 2

- Nhận xét
  - a là con trỏ đến a[0], a[0] là con trỏ đến a[0][0] ➔ a là con trỏ cấp 2.
  - Có thể truy xuất a[0][0] bằng 3 cách:

```
void main()
{
        int a[3][4];
        a[0][0] = 1;
        *a[0] = 1;
        **a = 1;

        a[1][0] = 1; *a[1] = 1; **(a+1) = 1;
        a[1][2] = 1; *(a[1]+2) = 1; *(*(a+1)+2) = 1;
}
```

Hướng tiếp cận 2

- Truyền mảng cho hàm
  - Truyền địa chỉ phần tử đầu tiên cho hàm.
  - Khai báo con trỏ rồi gán địa chỉ mảng cho con trỏ này để nó trỏ đến mảng.
  - Con trỏ này phải cùng kiểu với biến mảng, tức là con trỏ đến vùng nhớ n phần tử (mảng)
- Cú pháp

```
<kiểu dữ liệu> (*<tên con trỏ>)[<số phần tử>];
```

- Ví dụ

```
int (*ptr)[4];
```

Hướng tiếp cận 2

Truyền mảng cho hàm

```
void Xuat_1_Mang_C1(int (*ptr)[4])  // ptr[][4]
{
        int *p = (int *)ptr;
        for (int i = 0; i < 4; i++)
                printf("%d ", *p++);
}
void main()
{
        int a[3][4]={{1,2,3,4},{5,6,7,8},{9,10,11,12}};
        int (*ptr)[4];
        ptr = a;
        for (int i = 0; i < 3; i++)
                Xuat_1_Mang_C1(ptr++);  // hoặc ptr + i
                Xuat_1_Mang_C1(a++);    // sai => a + i
}
```

Hướng tiếp cận 2

Truyền mảng cho hàm

```
void Xuat_1_Mang_C2(int *ptr, int n)        // ptr[]
{
        for (int i = 0; i < n; i++)
                printf("%d ", *ptr++);
}
void main()
{
        int a[3][4]={{1,2,3,4},{5,6,7,8},{9,10,11,12}};
        int (*ptr)[4];
        ptr = a;
        for (int i = 0; i < 3; i++)
                Xuat_1_Mang_C2((int *)ptr++);
                Xuat_1_Mang_C2((int *)(a + i));// a++ sai
}
```

Hướng tiếp cận 2

Truyền mảng cho hàm

```
void Xuat_n_Mang_C1(int (*ptr)[4], int n)
{
        int *p = (int *)ptr;
        for (int i = 0; i < n * 4; i++)
              printf("%d ", *p++);
}
void main()
{
        int a[3][4]={{1,2,3,4},{5,6,7,8},{9,10,11,12}};
        int (*ptr)[4];
        ptr = a;

        Xuat_n_Mang_1(ptr, 3);
        Xuat_n_Mang_1(a, 3);
}
```

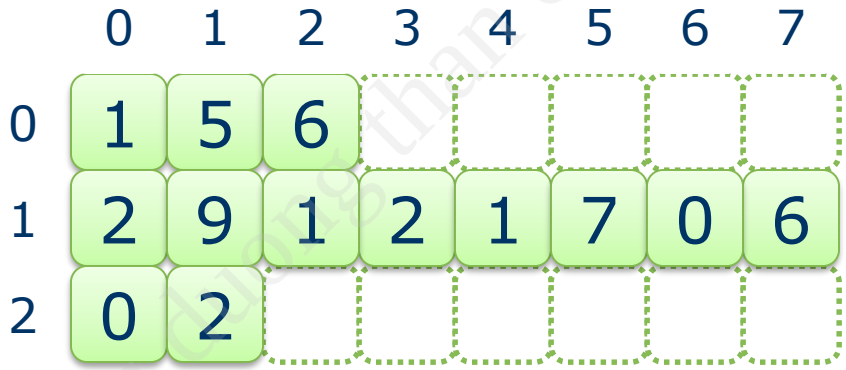
Hướng tiếp cận 2

Truyền mảng cho hàm

```
void Xuat_n_Mang_C2(int (*ptr)[4], int n)
{
        int *p;
        for (int i = 0; i < n; i++)
        {
                p = (int *)ptr++;

                for (int i = 0; i < 4; i++)
                        printf("%d ", *p++);

                printf("\n");
        }
}
```

Mảng con trỏ

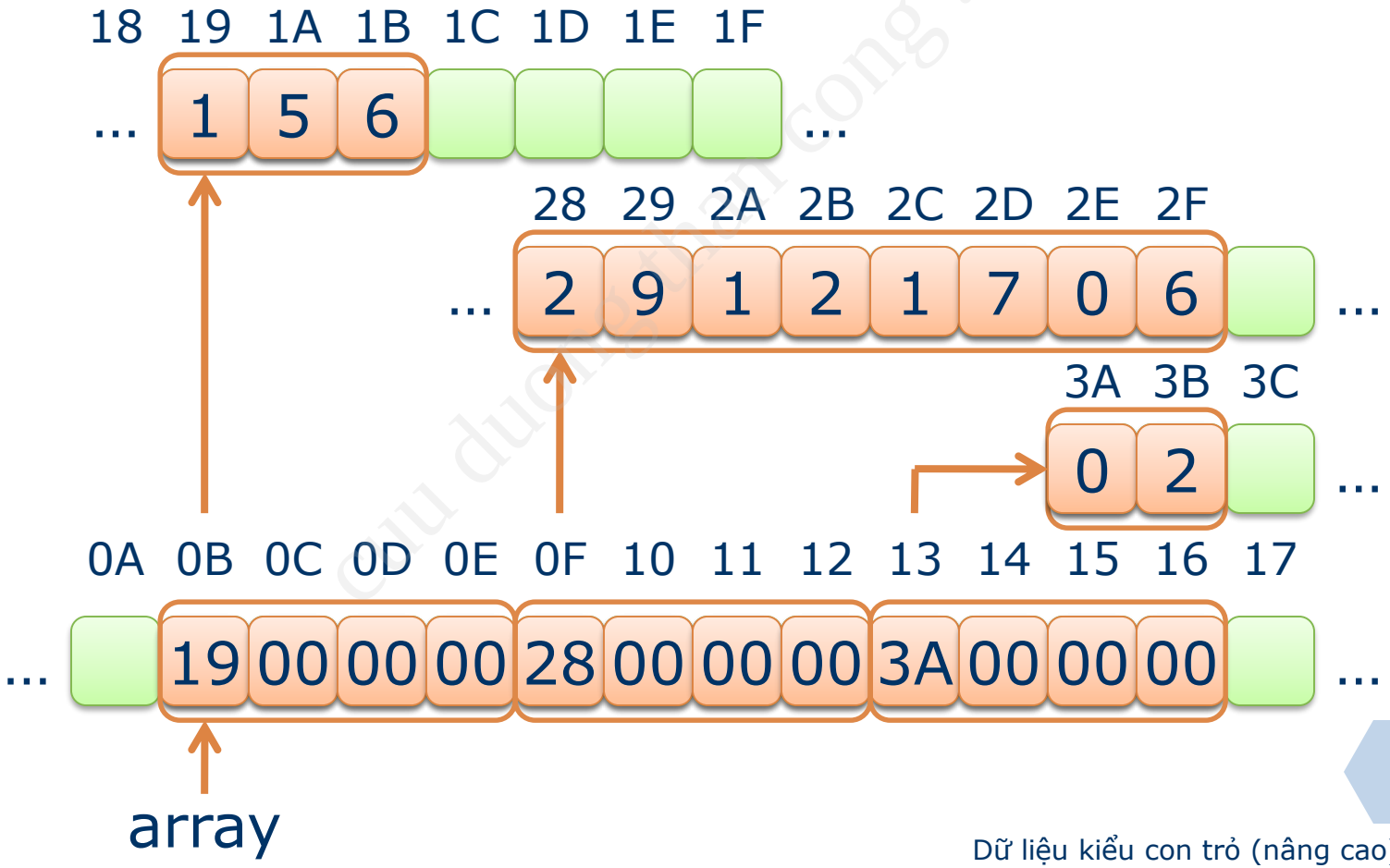
- Đặt vấn đề
  - Sử dụng cấu trúc dữ liệu nào để lưu trữ thông tin sau?



- Giải pháp?
  - Cách 1: Mảng 2 chiều 3x8 (tốn bộ nhớ)

Mảng con trỏ

- Cách 2: Mảng 1 chiều các con trỏ



Mảng con trỏ

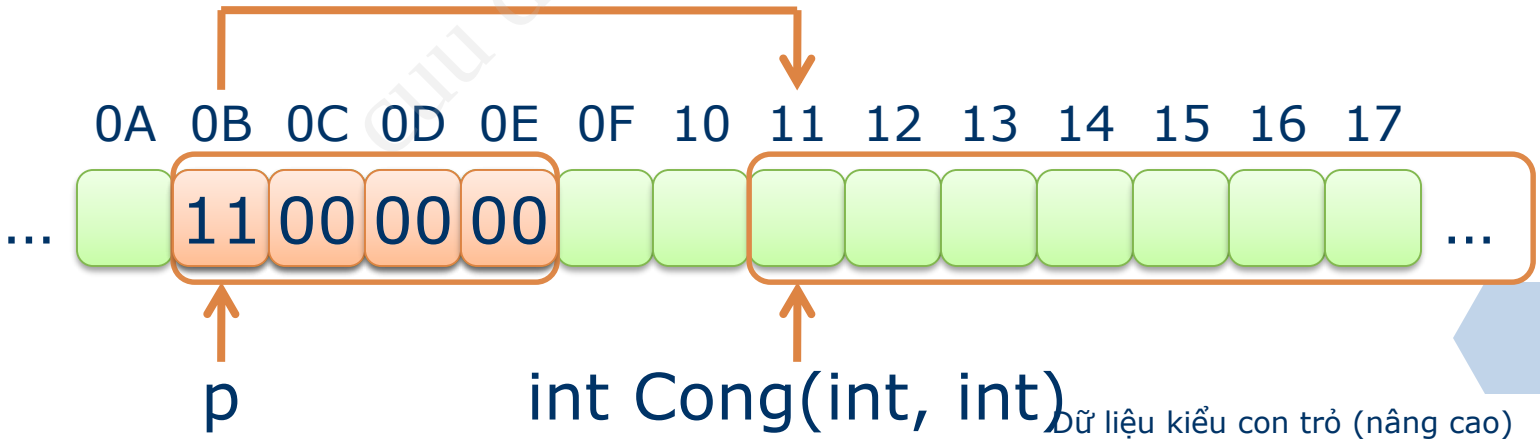
Ví dụ

```
void print_strings(char *p[], int n)
{
      for (int i = 0; i<n; i++)
            printf("%s ", p[i]);
}

void main()
{
      char *message[4] = {"Tin", "Hoc", "Co", "So"};

      print_strings(message, 4);
}
```

Con trỏ hàm

Khái niệm

- Hàm cũng được lưu trữ trong bộ nhớ, tức là cũng có địa chỉ.
- Con trỏ hàm là con trỏ trỏ đến vùng nhớ chứa hàm và có thể gọi hàm thông qua con trỏ đó.



Con trỏ hàm

Khai báo tường minh

```
<kiểu trả về> (* <tên biến con trỏ>)(ds tham số);
```

Ví dụ

```
// Con trỏ đến hàm nhận đối số int, trả về int
int (*ptof1)(int x);

// Con trỏ đến hàm nhận 2 đối số double, không trả về
void (*ptof2)(double x, double y);

// Con trỏ đến hàm nhận đối số mảng, trả về char
char (*ptof3)(char *p[]);

// Con trỏ đến không nhận đối số và không trả về
void (*ptof4)();
```

Con trỏ hàm

Khai báo không tường minh (thông qua kiểu)

```
typedef <kiểu trả về> (* <tên kiểu>)(ds tham số);
<tên kiểu> <tên biến con trỏ>;
```

Ví dụ

```
int (*pt1)(int, int);   // Tường minh

typedef int (*PhepToan)(int, int);

PhepToan pt2, pt3;      // Không tường minh
```

Con trỏ hàm

Gán giá trị cho con trỏ hàm

```
<biến con trỏ hàm> = <tên hàm>;
<biến con trỏ hàm> = &<tên hàm>;
```

- Hàm được gán phải cùng dạng (vào, ra)

Ví dụ

```
int Cong(int x, int y);                    // Hàm
int Tru(int x, int y);                     // Hàm
int (*tinhtoan)(int x, int y);             // Con trỏ hàm

tinhtoan = Cong;  // Dạng ngắn gọn
tinhtoan = &Tru;  // Dạng sử dụng địa chỉ
tinhtoan = NULL;  // Không trỏ đến đâu cả
```

Con trỏ hàm

So sánh con trỏ hàm

```
if  (tinhtoan != NULL)
{
      if (tinhtoan == &Cong)
            printf("Con trỏ đến hàm Cong.");
      else
            if (tinhtoan == &Tru)
                  printf("Con trỏ đến hàm Tru.");
            else
                  printf("Con trỏ đến hàm khác.");
}
else
      printf("Con trỏ chưa được khởi tạo!");
```

Con trỏ hàm

- Gọi hàm thông qua con trỏ hàm
  - Sử dụng toán tử lấy nội dung "*" (chính quy) nhưng trường hợp này có thể bỏ

```
int Cong(int x, int y);
int Tru(int x, int y);

int (*tinhtoan)(int, int);

tinhtoan = Cong;
int kq1 = (*tinhtoan)(1, 2);   // Chính quy
int kq2 = tinhtoan(1, 2);      // Ngắn gọn
```

Con trỏ hàm

Truyền tham số là con trỏ hàm

```
int Cong(int x, int y);
int Tru(int x, int y);
int TinhToan(int x, int y, int (*pheptoan)(int, int))
{
        int kq = (*pheptoan)(x, y);    // Gọi hàm
        return kq;
}

void main()
{
        int (*pheptoan)(int, int) = &Cong;
        int kq1 = TinhToan(1, 2, pheptoan);
        int kq2 = TinhToan(1, 2, &Tru);
}
```

Con trỏ hàm

Trả về con trỏ hàm

```
int (*LayPhepToan(char code))(int, int)
{
	if (code == '+')
		return &Cong;
	return &Tru;
}

void main()
{
	int (*pheptoan)(int, int) = NULL;
	pheptoan = LayPhepToan('+');
	int kq2 = pheptoan(1, 2, &Tru);
}
```

Con trỏ hàm

Trả về con trỏ hàm (khai báo kiểu)

```
typedef (*PhepToan)(int, int);
PhepToan LayPhepToan(char code)
{
	if (code == '+')
		return &Cong;
	return &Tru;
}

void main()
{
	PhepToan pheptoan = NULL;
	pheptoan = LayPhepToan('+');
	int kq2 = pheptoan(1, 2, &Tru);
}
```

Con trỏ hàm

Mảng con trỏ hàm

```
typedef (*PhepToan)(int, int);
void main()
{
        int (*array1[2])(int, int);    // tường minh
        PhepToan array2[2];            // kô tường minh

        array1[0] = array2[1] = &Cong;
        array1[1] = array2[0] = &Tru;

        printf("%d\n", (*array1[0])(1, 2));
        printf("%d\n", array1[1](1, 2));
        printf("%d\n", array2[0](1, 2));
        printf("%d\n", array2[1](1, 2));
}
```

Con trỏ hàm

- Lưu ý
  - Không được quên dấu () khi khai báo con trỏ hàm
    - int (*PhepToan)(int x, int y);
    - int *PhepToan(int x, int y);
  - Có thể bỏ tên biến tham số trong khai báo con trỏ hàm
    - int (*PhepToan)(int x, int y);
    - int (*PhepToan)(int, int);

Bài tập

- Câu 1: Ta có thể khai báo và sử dụng biến con trỏ đến cấp thứ mấy?

➔

- Câu 2: Có sự khác nhau giữa con trỏ đến một chuỗi và con trỏ đến một mảng ký tự không?

➔

Bài tập

- Câu 3: Nếu không sử dụng các kiến thức nâng cao về con trỏ, ta có thể giải quyết một số bài toán nào đó không?

  ➔

- Câu 4: Hãy nêu vài ứng dụng của con trỏ hàm.

  ➔

Bài tập

- Câu 5: Viết đoạn lệnh khai báo biến x kiểu float, khai báo và khởi tạo con trỏ px đến biến x và khai báo và khởi tạo con trỏ ppx đến con trỏ px.

➔

- Câu 6: Ta muốn gán 100 cho x thông qua con trỏ ppx bằng biểu thức gán "ppx = 100;" có được không?

➔

Bài tập

- Câu 7: Giả sử ta khai báo mảng array 3 chiều: int array[2][3][4]. Cho biết cấu trúc của mảng này đối với trình biên dịch C.

➔

Câu 8: Cho biết array[0][0] có nghĩa là gì?

➔

Bài tập

- Câu 9: Xét xem biểu thức so sánh nào sau đây đúng
  a. array[0][0] == &array[0][0][0];
  b. array[0][1] == array[0][0][1];
  c. array[0][1] == &array[0][1][0];

  ➔
- Câu 10: Viết nguyên mẫu của một hàm nhận một mảng con trỏ đến kiểu char làm đối số, và giá trị trả về có kiểu void.

  ➔

Bài tập

- Câu 11: Theo cách viết của câu 10, ta có thể biết được số phần tử của mảng được truyền kô?

  ➔

- Câu 12: Con trỏ đến hàm là gì?

  ➔

- Câu 13: Viết khai báo con trỏ đến một hàm mà hàm đó có giá trị trả về kiểu char, nhận đối số là một mảng con trỏ đến kiểu char.

  ➔

Bài tập

- Câu 13: Ta viết khai báo con trỏ ở câu 12 như vậy có đúng không? char *ptr(char *x[]);

  ➔

- Câu 14: Cho biết ý nghĩa của các khai báo sau:
  a. int *var1;
  b. int var2;
  c. int **var3;

Bài tập

- Câu 15: Cho biết ý nghĩa của các khai báo sau:
  a. int a[3][12];
  ➔
  b. int (*b)[12];
  ➔
  c. int *c[12];
  ➔

Bài tập

- Câu 16: Cho biết ý nghĩa của các khai báo sau:
  a. char *z[10];
  ➔
  b. char *y(int field);
  ➔
  c. char (*x)(int field);
  ➔

Bài tập

- Câu 17: Viết khai báo con trỏ func đến một hàm nhận đối số là một số nguyên và trả về giá trị kiểu float.

➔

- Câu 18: Viết khai báo một mảng con trỏ đến hàm. Các hàm nhận một chuỗi ký tự làm tham số và trả về giá trị kiểu nguyên. Ta có thể sử dụng mảng này để làm gì?

➔

Bài tập

- Câu 19: Viết câu lệnh khai báo một mảng 10 con trỏ đến kiểu char.

➔

- Câu 20: Tìm lỗi sai trong đoạn lệnh sau
  - int x[3][12];
  - int *ptr[12];
  - ptr = x;

➔

Bài tập

- Câu 21: Viết chương trình khai báo mảng hai chiều có 12x12 phần tử kiểu char. Gán ký tự 'X' cho mọi phần tử của mảng này. Sử dụng con trỏ đến mảng để in giá trị các phần tử mảng lên màn hình ở dạng lưới.
- Câu 22: Viết chương trình khai báo mảng 10 con trỏ đến kiểu float, nhận 10 số thực từ bàn phím, sắp xếp lại và in ra màn hình dãy số đã sắp xếp.
- Câu 23: Sửa lại bài tập 22 để người sử dụng có thể lựa chọn cách sắp xếp theo thứ tự tăng hay giảm dần.

Bài tập

- Câu 24: Chương trình cho phép người dùng nhập các dòng văn bản từ bàn phím đến khi nhập một dòng trống. Chương trình sẽ sắp xếp các dòng theo thứ tự alphabet rồi hiển thị chúng ra màn hình.
- Câu 25: Sử dụng con trỏ hàm để viết các hàm sắp xếp sau
  - Tăng dần
  - Giảm dần
  - Dương giảm rồi âm tăng, cuối cùng là số 0
  - …