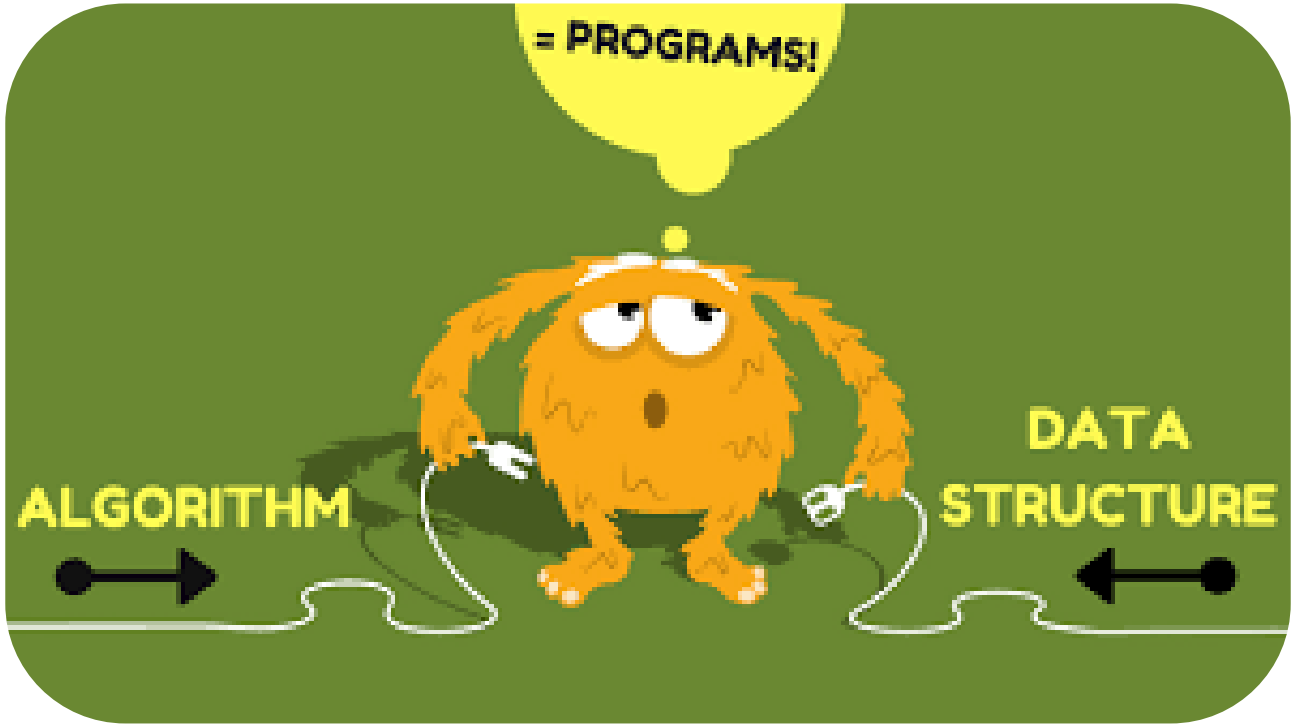
MẢNG VÀ CON TRỎ TRONG C



MỤC TIÊU BÀI HỌC

- Tìm hiểu kiểu dữ liệu mảng trong ngôn ngữ lập trình C.
- Tìm hiểu kiểu dữ liệu con trỏ trong ngôn ngữ lập trình C.
- Mối liên hệ giữa mảng và con trỏ trong ngôn ngữ lập trình C.

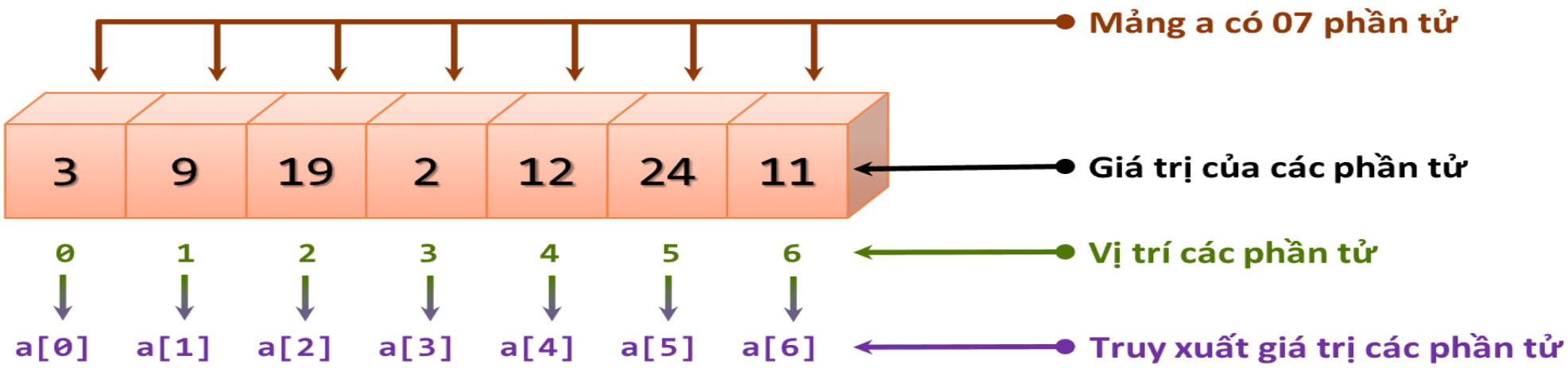
NỘI DUNG BÀI HỌC

1. Mảng trong C
2. Mảng một chiều
3. Mảng nhiều chiều
4. Khai báo và sử dụng biến con trỏ
5. Con trỏ và mảng
6. Con trỏ và tham chiếu hình thức của hàm


1
MẢNG TRONG C

➢ Mảng trong C:

- ✓ Mảng là một tập hợp các phần tử cố định **có cùng một kiểu**, gọi là kiểu phần tử.
- ✓ Kiểu phần tử có thể là: ký tự, số, chuỗi ký tự;
- ✓ Ta có thể chia mảng làm 2 loại: **mảng 1 chiều và mảng nhiều chiều**.



2 MẢNG MỘT CHIỀU

MẢNG 1 CHIỀU (TT)

- **Mảng 1 chiều** là một dãy các phần tử có cùng tên gọi, 1 chỉ số để chỉ thứ tự của phần tử đó trong dãy.
- *Mảng một chiều còn có thể hiểu như một Vector.*
- *Khai báo mảng với số phần tử xác định (khai báo tường minh)*
- Cú pháp:

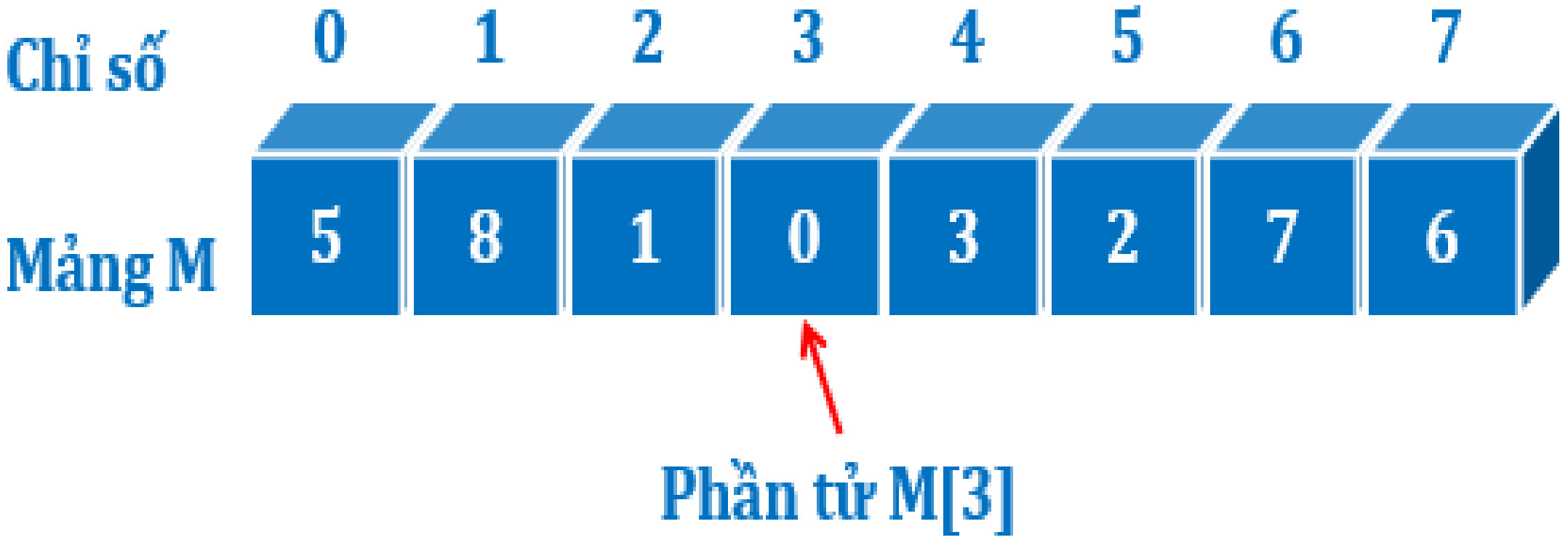
<Kiểu> <Tên mảng >[n]

- Trong đó:
  - ✓ **Tên mảng:** đây là một cái tên đặt đúng theo quy tắc đặt tên của danh biểu.

MẢNG 1 CHIỀU (TT)

- ✓ **n**: là một hằng số nguyên, cho biết số lượng phần tử tối đa trong mảng là bao nhiêu (hay nói khác đi kích thước của mảng là gì).
- ✓ **Kiểu**: mỗi phần tử của mảng có dữ liệu thuộc kiểu gì.
- ✓ Ở đây, ta khai báo một biến mảng gồm có **n** phần tử, phần tử thứ nhất là tên **mảng [0]**, phần tử cuối cùng là tên mảng **[n -1].**

- *Ví dụ*:



MẢNG MỘT CHIỀU (TT)

- *Khai báo mảng với số phần tử không xác định (khai báo không tường minh)*
- **Cú pháp:**

<Kiểu> <Tên mảng> <[]>

- Khi khai báo, không cho biết rõ số phần tử của mảng, kiểu khai báo này thường được áp dụng trong các trường hợp: vừa khai báo vừa gán giá trị, khai báo mảng là tham số hình thức của hàm.

MẢNG 1 CHIỀU (tt)

- *Vừa khai báo vừa gán giá trị*
  - ✓ Cú pháp:

<Kiểu> <Tên mảng> []= {Các giá trị cách nhau bởi dấu phẩy}

- Sử dụng hàm sizeof() để lấy số phần tử của mảng như sau:

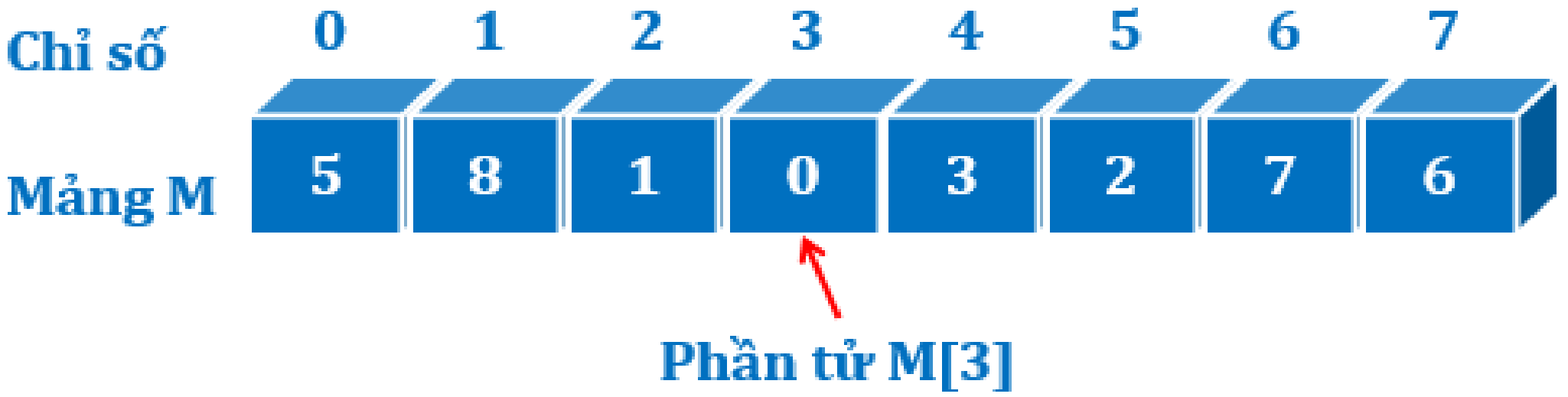
Số phần tử=sizeof(tên mảng)/ sizeof(kiểu)

- Truy xuất từng phần tử của mảng:
  - ✓ Mỗi phần tử của mảng được truy xuất thông qua **Tên biến mảng** theo sau là chỉ số nằm trong cặp dấu ngoặc vuông **[ ]**.

➢ *Ví dụ 1*:

✓ int M[8];

✓ Phần tử thứ 4 (có vị trí 3) là M[3]…



➢ ***Ví dụ 2:*** Vừa khai báo vừa gán trị cho 1 mảng 1 chiều các số nguyên.

```
#include <stdio.h>
#include <conio.h>
int main(){
        int n,i,j,tam;
        int dayso[]={66,65,69,68,67,70};
        n=sizeof(dayso)/sizeof(int); /*Lay so phan tu*/
        printf("\n Noi dung cua mang ");
        for (i=0;i<n;i++)
        printf("%d ",dayso[i]);
        getch();
        return 0;
        }
```

3

MẢNG NHIỀU CHIỀU

MẢNG NHIỀU CHIỀU

- Mảng nhiều chiều là mảng có từ 2 chiều trở lên.
- Người ta thường sử dụng mảng nhiều chiều để lưu các ma trận, các tọa độ 2 chiều, 3 chiều…
- ***Khai báo mảng 2 chiều tường minh.***
- Cú pháp:

```
<Kiểu>  <Tên mảng><[Số phần tử chiều 1]><[Số phần tử chiều 2]>
```

MÃNG NHIỀU CHIỀU(tt)

- *Ví dụ*: Cần lưu trữ thông tin của một ma trận gồm các số thực. khai báo một mảng 2 chiều như sau:

```
float m[8][9];  /* Khai báo mảng 2 chiều có 8*9 phần tử là số thực*/
                /*  khai báo cho một ma trận có tối đa là 8 dòng, mỗi dòng
                    có tối đa là 9 cột. */
```

- **Khai báo mảng 2 chiều không tường minh.**

MÃNG NHIỀU CHIỀU(tt)

- Để khai báo mảng 2 chiều không tường minh, phải chỉ ra số phần tử của chiều thứ hai (chiều cuối cùng).
- Cú pháp:

```
<Kiểu> <Tên mảng> <[]><[Số phần tử chiều 2]>
```

- Truy xuất một phần tử của mảng hai chiều bằng cách viết ra tên mảng theo sau là hai chỉ số đặt trong hai cặp dấu ngoặc vuông, chẳng hạn m[2][3].

4

CON TRỎ

KHAI BÁO VÀ SỬ DỤNG CON TRỎ

- Các biến sử dụng trước đây đều là biến có kích thước và kiểu dữ liệu xác định được **gọi là biến tĩnh**.
- Cần phải cấp một lượng bộ nhớ, mà không cần biết trong quá trình thực thi chương trình có sử dụng hết lượng ô nhớ này hay không.
- Các biến tĩnh dạng này sẽ tồn tại trong suốt thời gian thực thi chương trình dù có những biến mà chương trình chỉ sử dụng 1 lần rồi bỏ.

KHAI BÁO VÀ SỬ DỤNG CON TRỎ

- Khó khăn:
  - ✓ Cấp phát ô nhớ dư, gây ra lãng phí ô nhớ.
  - ✓ Cấp phát ô nhớ thiếu, chương trình thực thi bị lỗi.
- Khắc phục:-> sử dụng biến động
  - ✓ Chỉ phát sinh trong quá trình thực hiện chương trình, không phát sinh lúc bắt đầu chương trình;
  - ✓ Khi chạy chương trình, kích thước của biến, 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 chỗ trong bộ nhớ.

KHAI BÁO VÀ SỬ DỤNG CON TRỎ

- Tuy nhiên các biến động không có địa chỉ nhất định nên ta không thể truy cập đến chúng được.
- Vì thế, ngôn ngữ C lại cung cấp một loại biến đặc biệt để khắc phục tình trạng này, là biến con trỏ (pointer) với các đặc điểm:
  - ✓ Biến con trỏ không chứa dữ liệu mà chỉ chứa địa chỉ của dữ liệu hay chứa địa chỉ của ô nhớ chứa dữ liệu.
  - ✓ Kích thước của biến con trỏ không phụ thuộc vào kiểu dữ liệu, luôn có kích thước cố định là 2 byte.

KHAI BÁO VÀ SỬ DỤNG CON TRỎ

- *Khai báo biến con trỏ*
- Cú pháp:

```
<Kiểu> * <Tên con trỏ> ;
```

- Ý nghĩa: Khai báo một biến có tên là Tên con trỏ dùng để chứa địa chỉ của các biến.

✓***Ví dụ 1*:** Khai báo 2 biến a,b có kiểu int và 2 biến pa, pb là 2 biến con trỏ kiểu int.

```
int a, b, *pa, *pb;
```

✓***Ví dụ 2*:** Khai báo biến f kiểu float và biến pf là con trỏ float

```
float f, *pf;
```

- **Lưu ý:**
  - ✓ Nếu chưa muốn khai báo kiểu dữ liệu mà con trỏ ptr đang chỉ đến, ta sử dụng:

```
void *ptr;
```

- Nếu ta muốn con trỏ ptr chỉ đến kiểu dữ liệu gì cũng được.

  Tác dụng của khai báo này là chỉ dành ra **2 bytes** trong bộ nhớ để cấp phát cho biến con trỏ ptr.

KHAI BÁO VÀ SỬ DỤNG CON TRỎ

➢ Các thao tác với con trỏ:

✓ Gán địa chỉ của biến cho biến con trỏ:

- Toán tử & dùng để định vị con trỏ đến địa chỉ của một biến đang làm việc.

- Cú pháp:

<Tên biến con trỏ>=&<Tên biến>;

- Ví dụ:

Gán địa chỉ của biến a cho con trỏ pa, gán địa chỉ của biến b cho con trỏ pb.

pa=&a; pb=&b;

- Lưu ý: Khi gán địa chỉ của biến tĩnh cho con trỏ cần phải cùng kiểu dữ liệu.

- Ví dụ: sau đây không đúng do không tương thích kiểu:

```
int Bien_Nguyen;
float *Con_Tro_Thuc;
Con_Tro_Thuc=&Bien_Nguyen;
```

✓ Cấp phát vùng nhớ cho biến con trỏ:

- Trước khi sử dụng biến con trỏ, ta nên cấp phát vùng nhớ cho biến con trỏ này quản lý địa chỉ.

- **Cú pháp:**

  + Cấp phát vùng nhớ có kích thước là size.

```
void *malloc(size_t size):
```

  + Cấp phát vùng nhớ có kích thước là nitems*size.

```
void *calloc(size_t nitems, size_t size):
```

✓ *Cấp phát lại vùng nhớ cho biến con trỏ:*

- Nếu ta cần cấp phát thêm vùng nhớ có kích thước lớn hơn vùng nhớ đã cấp phát, ta sử dụng hàm **realloc().**

- *Cú pháp*:

```
void *realloc(void *block, size_t size)
```

✓ Giải phóng vùng nhớ cho biến con trỏ

- Cú pháp:

```
void free(void *block)
```

5

CON TRỎ VÀ MẢNG

KHAI BÁO VÀ SỬ DỤNG CON TRỎ

➢ *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.

✓ Truy cập các phần tử mảng theo dạng con trỏ

&<Tên mảng>[0] tương đương với <Tên mảng>

&<Tên mảng> [<Vị trí>] tương đương với <Tên mảng> + <Vị trí>

<Tên mảng>[<Vị trí>] tương đương với *(<Tên mảng> + <Vị trí>)

➢ *Con trỏ và mảng nhiều chiều:*

✓ Ta có thể sử dụng con trỏ thay cho mảng nhiều chiều như sau: Giả sử ta có mảng 2 chiều và biến con trỏ như sau:

```
int a[n][m];
int *contro_int;
```

Thực hiện phép gán:

```
contro_int=a;
```

6 CON TRỎ VÀ THAM SỐ HÌNH THỨC CỦA HÀM

CON TRỎ VÀ THAM SỐ HÌNH THỨC CỦA HÀM

- Khi tham số hình thức của hàm là một con trỏ, theo nguyên tắc gọi hàm dùng tham số thực tế là 1 con trỏ có kiểu giống với kiểu của tham số hình thức.
- Nếu lúc thực thi hàm, có sự thay đổi trên nội dung vùng nhớ được chỉ bởi con trỏ tham số hình thức thì lúc đó nội dung vùng nhớ được chỉ bởi tham số thực tế cũng sẽ bị thay đổi theo.
- Ví dụ : Xét hàm hoán vị được viết như sau :

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

CON TRỎ VÀ THAM SỐ HÌNH THỨC CỦA HÀM

```
1.  void HoanVi(int *a, int *b)
2.  { int c=*a;
3.  *a=*b;
4.  *b=c;
5.  }
6.  int main()
7.  {   int m=20,n=30;
8.      printf("Truoc khi goi ham m= %d, n= %d\n",m,n);
9.      HoanVi(&m,&n);
10.     printf("Sau khi goi ham m= %d, n= %d",m,n);
11.     getch();
12.     return 0;
13. }
```

TỔNG KẾT

- Trong lập trình, mảng là CTDL hay được dùng nhất.
- Dùng mảng thông thường cho bài toán tính toán trên dữ liệu dạng số.
- Con trỏ được sử dụng linh động và giúp tiết kiệm bộ nhớ cho máy tính.

Câu hỏi multiple choice

Loại hình da phương tiện nào mà có ưu điểm là chi phí thấp, dễ kiểm soát và đo lường được tính hiệu quả trong quảng cáo?

A. Tờ rơi, tời gấp.

B. Thư trực tiếp.

C. Điện thoại trực tiếp.

D. Truyền hình

E. Quảng cáo ngoài trời

F. Truyền thanh


感謝你
Danke
Gracias
Grazie
Merci
Thanks!
THANK
YOU!


VAIO