Ngôn ngữ lập trình C

B12: Con trỏ và mảng

Khoa Công nghệ thông tin

Con trỏ và mảng 1 chiều

- Xét câu lệnh:

```
int a[10];
```

- Khai báo này hệ thống sẽ cấp cho mảng a 10 khối nhớ liên tiếp: `a[0], a[1], a[2], a[3], .... , a[9]` (trong trường hợp này mỗi khối nhớ 2 byte).

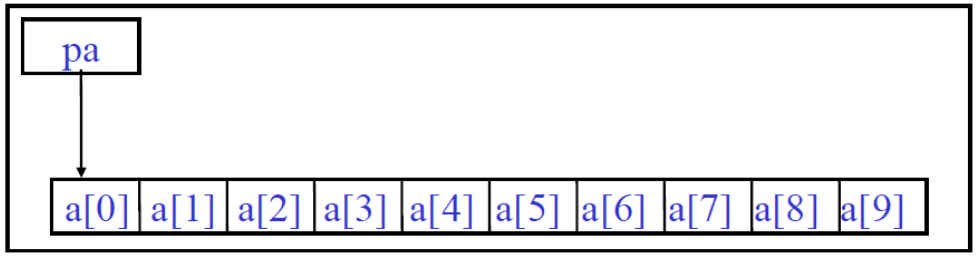
a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]

Con trỏ và mảng 1 chiều

- Chú ý ở đây là tên mảng là một hằng địa chỉ, nó chính là địa chỉ của phần tử đầu tiên của mảng; điều này có nghĩa là:

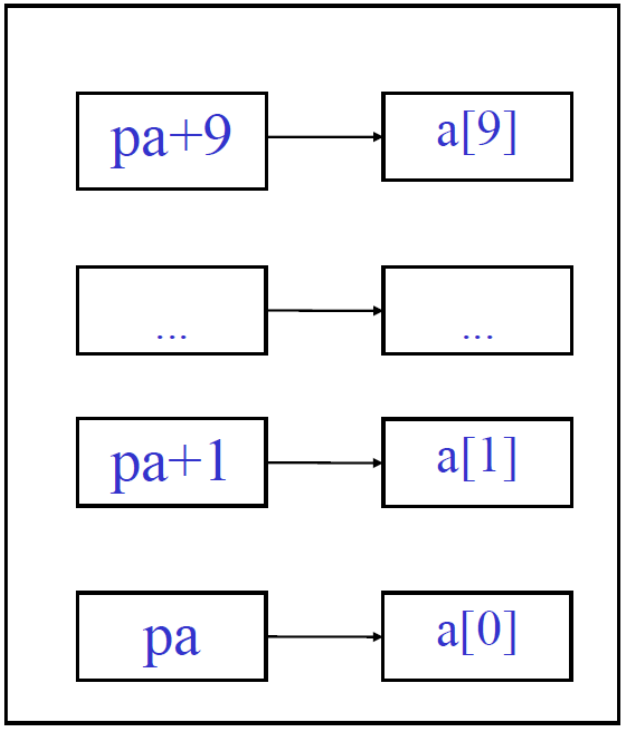
`a` tương đương với `&a[0]`

  - Nếu pa là một con trỏ kiểu nguyên: `int *pa;`
  - Khi đó phép gán: `pa=&a[0];`
  - Đem pa trỏ đến phần tử thứ nhất (có chỉ số là 0) của a; nghĩa là pa chứa địa chỉ của `a[0]`.



Con trỏ và mảng 1 chiều

- `pa+i` là địa chỉ của `a[i]` và `*(pa+i)` là nội dung của `a[i]`



Con trỏ - tương đương mảng

- Mảng thực sự là một loại con trỏ!
- Khi một mảng được định nghĩa, lượng bộ nhớ cấp cho mảng được xác định và không thay đổi (Kích thước của mảng): biến mảng được đặt để trỏ đến đầu của đoạn bộ nhớ đó (có giá trị là địa chỉ đầu của phần tử mảng).
- Khi một con trỏ được khai báo, nó không được khởi tạo (giống như một biến thông thường).
- Không giống như con trỏ, giá trị của biến mảng không thể thay đổi (giá trị của biến mảng là một địa chỉ cố định, hằng số, static).

Phép toán số học đối với biến con trỏ

- Biến con trỏ có thể tăng/giảm giá trị (như biến thông thường).
- Nếu `p` là con trỏ tới một kiểu dữ liệu cụ thể thì `p+1` trỏ tới địa chỉ chính xác của biến tiếp theo cùng kiểu dữ liệu
- `p++`, `p+i`, and `p += i` là tương đương.

Phép toán số học đối với biến con trỏ

- Nếu `p` và `q` trỏ đến các phần tử trong một mảng, `q-p` cho kết quả là số phần tử giữa `p` và `q`.
- Tuy nhiên có một sự khác biệt giữa phép toán số học của con trỏ với phép toán số học thông thường.

Phép toán số học đối với biến con trỏ

- Ví dụ 12.1

```c
#include<stdio.h>
void main(){
 int a[7]={13, -355, 235, 47, 67, 943, 1222}, *p, *q;
 int i;
 p = a;
 q = p+4;
 printf("Gia tri cua a dang hexa: %p\t dang so nguyen: %ld\n", a,(long int)a);
 printf("Dang hexa p la: %p,\tq la: %p\t&a[4] la: %p\n", p,q,&a[4]);
 printf("Dang so nguyen p la: %ld,\tq la: %ld\t&a[4] la: %ld\n", (long int)p,(long int)q,(long int)&a[4]);
 printf("p tro den phan tu co gia tri: %d,\tq tro den phan tu co gia tri: %d\n", *p, *q);
 printf("Khoang cach giua p va q tinh theo don vi con tro la: %d\n", (int)(q-p));
 printf("Khoang cach giua p va q tinh theo don vi so nguyen la: %ld\n", (long int)q-(long int)p);
}
```

- Kết quả:

```
Gia tri cua a dang hexa: 0x7fff5cd39ae0  dang so nguyen: 140734750759648
Dang hexa p la: 0x7fff5cd39ae0, q la: 0x7fff5cd39af0    &a[4] la: 0x7fff5cd39af0
Dang so nguyen p la: 140734750759648,   q la: 140734750759664   &a[4] la: 140734
750759664
p tro den phan tu co gia tri: 13,       q tro den phan tu co gia tri: 67
Khoang cach giua p va q tinh theo don vi con tro la: 4
Khoang cach giua p va q tinh theo don vi so nguyen la: 16
```

Một số cách sử dụng mảng và con trỏ một cách tương đương

```
#include <stdio.h>
void main()
{
  float a[5],s;
  int i;
  for(i=0;i<5;i++)
  {
    printf("\na[%d] = ",i) ;
    scanf("%f",&a[i]);
   }
}
```

```
#include <stdio.h>
void main()
{
  float a[5],s;
  int i;
  for(i=0;i<5;i++)
  {
    printf("\na[%d] = ",i) ;
    scanf("%f",a+i);
   }
}
```

Một số cách sử dụng mảng và con trỏ một cách tương đương

```
#include <stdio.h>
void main()
{
  float a[5],s, *p;
  int i; p=a;
  for(i=0;i<5;i++)
  {
    printf("\na[%d] = ",i) ;
    scanf("%f",&p[i]);
   }
}
```

```
#include <stdio.h>
void main()
{
  float a[5],s,*p;
  int i; p=a;
  for(i=0;i<5;i++)
  {
    printf("\na[%d] = ",i) ;
    scanf("%f",p+i);
   }
}
```

Phép toán số học đối với biến con trỏ

- **Tóm lại:**
  - Một biểu thức chứa một mảng và một chỉ số tương đương với một cách viết khác sử dụng một con trỏ cùng một độ lệch.
  - Có sự khác nhau giữa tên của mảng và con trỏ.
  - Một con trỏ là một biến và vì thế, các câu lệnh `pa=a;` và `pa++;` là hợp lệ.
  - Nhưng một tên mảng không phải là một con trỏ, do đó các câu lệnh kiểu như `a=pa; a++;` là không hợp lệ.
  - Về thực chất, tên mảng là một hằng con trỏ do đó chúng ta không thể thay đổi giá trị của nó được.

Truyền mảng cho hàm

- Do con trỏ tương đương mảng và có thể gán mảng cho con trỏ nên một cách khác để truyền màng cho hàm là sử dụng con trỏ.
- Trong thực tế, chúng ta chỉ cần truyền địa chỉ mảng hay chính xác hơn là một con trỏ chỉ tới mảng.
- **Ví dụ 12.2**: viết hàm tính tổng các thành phần của một mảng được truyền vào.

```
#include <stdio.h>
int addNumbers(int *fiveNumber, int size){
      int i,sum=0;
      for(i=0; i<size; i++, fiveNumbers++){
            sum+= *fiveNumbers
      }
      return sum;
}
```

Bài tập 12.1

- Viết hàm `countEven(int*, int)` nhận vào một mảng các số nguyên và kích thước của nó và trả về số các số chẵn trong mảng đó.

Bài tập 12.2

- Viết một hàm trả về một con trỏ chỉ tới giá trị `maximum` của một mảng số `double` được truyền vào. Nếu mảng trống thì trả về giá trị `NULL` (`return NULL`).

```
double* maximum(double* a, int size);
```

Con trỏ và mảng nhiều chiều

- **Phép cộng địa chỉ trong mảng 2 chiều:**
  - Xét khai báo: `float a[2][3];`
  - Với khai báo này hệ thống cấp sáu phần tử liên tiếp trong bộ nhớ theo thứ tự sau:

    `a[0][0] a[0][1] a[0][2] a[1][0] a[1][1] a[1][2]`
  - Cũng như mảng một chiều tên của mảng hai chiều được hiểu như địa chỉ của phần tử đầu tiên của nó.
  - Phép cộng địa chỉ phải hiểu như sau: C cho rằng mảng hai chiều là mảng của mảng. Như vậy với khai báo trên thì `a` là mảng mà mỗi phần tử của nó là một dãy gồm 3 số thực. Vì vậy:
    - a trỏ tới đầu hàng thứ nhất (phần tử a[0][0])
    - a+1 trỏ tới đầu hàng thứ hai (phần tử a[1][0])

Con trỏ và mảng nhiều chiều

- **Con trỏ và mảng 2 chiều:**
  - Để truy xuất vào các phần tử của mảng hai chiều ta vẫn có thể dùng con trỏ theo cách sau:

```
float *p, a[2][3];
p=(float*)a;
```

  - Khi đó:
    - p trỏ tới a[0][0]
    - p+1 trỏ tới a[0][1]
    - p+2 trỏ tới a[0][2]
    - p+3 trỏ tới a[1][0]
    - p+4 trỏ tới a[1][1]
    - p+5 trỏ tới a[1][2]

Con trỏ và mảng nhiều chiều

- **Minh họa cách đọc dữ liệu cho mảng 2 chiều.**
- Ví dụ 12.3:

```
#include <stdio.h>
void main()
  {
  float a[2][3],*p;
  int i,j,m,n;
  p=(float*)a;
  for(i=0;i<2;i++)
        for(j=0;j<3;j++)
                {
                printf("a[%d][%d] = ",i,j);
                scanf("%f",p+i*3+j);
                }
                 for(i=0;i<2;i++)
        for(j=0;j<3;j++)
                {
                 printf("%6.2f",a[i][j]);
                   if(j==2) printf("\n");

                }
  }
```

Con trỏ và mảng nhiều chiều

- **Con trỏ và mảng 2 chiều:**
  - Để ý rằng a là một hằng con trỏ chỉ đến các dòng của một ma trận hai chiều, vì vậy:
    - `a` trỏ đến dòng thứ nhất
    - `a+1` trỏ đến dòng thứ hai
  - Để tính toán được địa chỉ của phần tử ở dòng `i`, cột `j` chúng ta phải dùng phép chuyển đổi kiểu bắt buộc đối với `a`: `(float*)a`, đây là con trỏ chỉ đến thành phần `a[0][0]` của ma trận. Và vì vậy thành phần `a[i][j]` sẽ có địa chỉ là `(float*a)+i*n+j` (`n` là số cột).

Bài tập 12.3

- Viết hàm đảo ngược một mảng sử dụng 2 cách: dùng chỉ số và dùng con trỏ.

Đáp án Bài tập 12.3: dùng mảng

```
void reversearray(int arr[], int size){
   int i, j, tmp;
   i=0; j= size -1;
   while(i<j){
       tmp=a[i];
       a[i]=a[j];
       a[j]= tmp;
       i++; j--;
   }
}
```

Đáp án Bài tập 12.3: dùng con trỏ

```
void reversearray(int *arr, int size){
  int i, j, tmp;
  i=0; j= size -1;
  while(i<j){
     tmp=*(a+i);
     *(a+i)=*(a+j);
     *(a+j)= tmp;
     i++; j--;
  }
}
```

Bài tập 12.4: Mô phỏng trò chơi

- Một game đối kháng cho phép tạo ra các đấu thủ (mỗi đấu thủ có ba chỉ số sức mạnh, nhanh nhẹn, và máu)
- Viết chương trình tạo ra ba đấu thủ với các chỉ số ban đầu ngẫu nhiên từ 0 đến 100. Chương trình sử dụng hàm: `arena` nhận tham số là các đấu sĩ, đưa ra kết quả là người thắng cuộc nếu tổng 3 chỉ số lớn hơn. Sau một trận đấu người thắng cuộc các chỉ số tăng 2%. Người thua cuộc tăng 1%. Ngoài ra một đấu sĩ còn được điểm % thưởng ngẫu nhiên.
- Cho ba người thi đầu vòng tròn 1, 2, 3, …, n số lần tùy chọn và in ra kết quả cùng chỉ số cuối cùng của mỗi đấu thủ.

Bài tập 12.5

- Cho một mảng số thực. Viết hàm sử dụng con trỏ làm tham số trả về con trỏ trỏ tới phần tử có giá trị gần nhất với giá trị trung bình của mảng. Kiểm tra bằng chương trình.