Kỹ thuật lập trình(6): ngôn ngữ lập trình C

Khoa Công nghệ thông tin
Học viện Kỹ thuật Quân sự
100-Hoàng Quốc Việt – Hà Nội

Kiểu mảng

- Khi làm việc với các cấu trúc dữ liệu dạng dãy hay danh sách các phần tử, ta sử dụng kiểu mảng (array)
  - Mảng 1 chiều: một vec-tơ các phần tử
  - Mảng nhiều chiều: một bảng các phần tử
- Mảng một chiều
  - Dãy các phần tử có cùng kiểu dữ liệu
  - Các phần tử được sắp xếp theo trật tự nhất định

Kiểu mảng

- Cú pháp khai báo mảng một chiều

  kiểu_dữ_liệu tên_mảng[số_phần_tử_của_mảng];
- Ví dụ
  - int ai[10];
  - float af[100];
- Số phần tử mảng được xác định khi khai báo
- Sử dụng toán tử [] để truy cập phân tử của mảng
  - Ví dụ: ai[2], af[10], …
- Chỉ số các phần tử mảng được **đánh số từ 0**

Kiểu mảng

- Ví dụ
  - Nhập danh sách các giá trị nguyên vào một mảng, sau đó tìm phần tử có giá trị nhỏ nhất trong mảng

```
#include <stdio.h>
#define N 10
main()
{
   int x[N], min;
    int i;
   for (i=0; i <= N-1; i++){
            printf(" x[%d]= ", i);
            scanf("%d", &x[i]);
   }
   min = x[0];
   for (i=1; i < N; i++)
            if (min > x[i]) min = x[i];
   printf("\n min= %d", min);
}
```

Kiểu mảng

- Khởi tạo mảng
  - Mảng có thể được khởi tạo giá trị ngay khi khai báo
  - Cú pháp

    kiểu_dữ_liệu tên_mảng[số_phần_tử_của_mảng] = {danh_sách_các_giá_trị_khởi_tạo};
  - Khi khai báo mảng có khởi tạo giá trị thì có thể không cần chỉ ra số phần tử mảng
  - Ví dụ

    int ai[3] = {2, 4, 5};
    - Hoặc

      int ai[] = {2, 4, 5}; /*không khai báo số phần tử mảng*/

Kiểu mảng

- Định nghĩa kiểu mới – từ khóa **typedef**
  - Có thể sử dụng từ khóa **typedef** để định nghĩa các kiểu dữ liệu mới
  - Kiểu dữ liệu mới sẽ được sử dụng để khai báo dữ liệu
  - Ví dụ
    - typedef int kieunguyen;
    - typedef float mangthuc10[10];

    sử dụng
    - kieunguyen x, a[100];
    - mangthuc10 x, y;

Kiểu mảng

- Mảng và địa chỉ
  - Toán tử & dùng để lấy địa chỉ một biến
  - Toán tử & cũng được dùng để lấy địa chỉ của một phần tử mảng
  - Các phần tử trong mảng được bố trí các ô nhớ liên tiếp nhau trên bộ nhớ
    - Nếu biết được địa chỉ phần tử thú i sẽ xác định được địa chỉ phần tử thú i+1
    - Địa chỉ phần tử đầu tiên là địa chỉ của mảng
  - Tên mảng mang địa chỉ của mảng đó

Kiểu mảng

- Mảng và địa chỉ
  - Ví dụ

    float a[100];

    float *pa;
    - Các cách viết sau là tương đương:

      a ⇔ &a[0]

      a + i ⇔ &a[i]

      *(a + i) ⇔ a[i]
    - Các phép gán hợp lệ

      pa = a;

      pa = &a[0];

Kiểu mảng

- Mảng là tham số của hàm
  - Khi sử dụng mảng là tham số của hàm, ta có thể khai báo, chẳng hạn:

    int a[]
    - Hoặc

      int *a
  - Như thế, hai cách sau là tương đương:

    f(int a[]) { … }

    f(int *a) { … }
  - Khi sử dụng, có thể gọi:

    f(a);

    Hoặc

    f(&a[0]);

Kiểu mảng

- Mảng là tham số của hàm
  - Ví dụ

```
void nhap_mang(int *x, int n)
{
        int i;
        /* Đọc các giá trị mảng */
        for (i=0; i <= n-1; i++)
        {
                printf(" x[%d]= ", i);
                scanf("%d", &x[i]);
        }
}
```

Kiểu mảng

- Mảng là tham số của hàm
  - Ví dụ

```
void xuat_mang(int *x, int n)
{
        int i;
        /* In các giá trị mảng */
        for (i=0; i <= n-1; i++)
                printf(" x[%d]= %d\n", i, x[i]);
}
```


Kiểu mảng

- Sắp xếp mảng
  - Sắp xếp các phần tử của mảng sao cho giá trị chúng theo thứ tự tăng dần hay giảm dần
  - Vấn đề thường gặp trong tin lập trình
  - Có nhiều cách sắp xếp khác nhau
    - Sắp xếp lựa chọn
    - Sắp xếp nổi bọt
    - Sắp xếp nhanh
    - Sắp xếp vun đống
    - …
  - Giả sử các phần tử của mảng có kiểu nguyên hoặc thực

Kiểu mảng

- Sắp xếp lựa chọn
  - Lấy phần tử đầu so sánh với các phần tử còn lại, nếu nó lớn hơn (nhỏ hơn) thì đổi chỗ giá trị của phần tử đầu tiên với phần tử đang so sánh. Kết quả sau lượt đầu, phần tử đầu tiên sẽ giữ giá trị nhỏ nhất.
  - Tiếp tục lượt hai, lấy phần tử thứ hai so sánh với các phần tử tiếp theo, nếu nó lớn hơn thì đổi chỗ giá trị của phần tử thứ hai với phần tử đang so sánh.
  - Việc này được tiến hành cho đến khi ta gặp phần tử cuối cùng.

Kiểu mảng

- Sắp xếp lựa chọn

```
#define N 50

…

int x[N];
int i, j, tam;

/* Đọc các giá trị mảng */

/* Sắp xếp mảng theo chiều tăng dần */
for (i=0; i < N-1; i++)
  for (j=i+1; j < N; j++)
  {
        if (x[i] > x[j])
        {
              tam=x[i];/* Hoán đổi giá trị 2 biến */
              x[i]=x[j];
              x[j]=tam;
        }
 }
```

Kiểu mảng

- Sắp xếp lựa chọn
  - Cải tiến: ở một lượt i nào đó, thay vì đổi chỗ liên tục phần tử thứ i với phần tử có giá trị nhỏ hơn, thì ta chỉ thực hiện việc đổi chỗ phần tử nhỏ nhất ở lượt i với phần tử thứ i.

```
/* Sắp xếp mảng theo chiều tăng dần */
for (i=0; i < kich_thuoc - 1; i++)
{
   m = i;
          for (j = i+1; j < kich_thuoc; j++)
          {
                    if (x[m] > x[j]) m = j;
          }
          if (m != i)
          {
                    tam=x[m];
                    x[m]=x[i];
                    x[i]=tam;
          }
}
```


Kiểu mảng

- Sắp xếp nổi bọt
  - Duyệt các phần tử của mảng từ cuối mảng lên đến đầu mảng
  - Gặp hai phần tử kế cận ngược thứ tự thì đổi chỗ cho nhau
  - Như thế, lượt đầu sẽ chuyển phần tử nhỏ nhất lên đầu mảng phần tử
  - Tiếp tục, lượt thứ hai phần tử nhỏ thứ hai sẽ được chuyển đến vị trí thứ hai
  - …
  - Hình dung mảng được xếp thẳng đứng thì sau từng lượt các phần tử nhỏ dần sẽ được nỗi lên như “bọt nổi lên trong nồi nước đang sôi”

Kiểu mảng

- Sắp xếp nổi bọt

```
/* Sắp xếp nổi bọt */
for (i = 0; i < kich_thuoc - 1; i++)
{
for (j = kich_thuoc - 1; j > i + 1; j--)
{
        if (x[j] < x[j-1])
        {
                tam=x[j];
                x[j]=x[j-1];
                x[j-1]=tam;
        }
}
}
```


Kiểu mảng

- Sắp xếp nhanh (quicksort)
  - Chọn một phần tử làm "chốt"
  - So sánh các phần tử còn lại với chốt và thực hiện hoán đổi sao cho các phần tử nhỏ hơn chốt được xếp trước chốt, các phần tử nhỏ hơn chốt được xếp sau chốt
  - Sau bước này mảng gồm
    - Phân đoạn các phần tử nhỏ hơn chốt
    - Chốt (cũng là vị trí thực của chốt sau khi mảng đã sắp xếp)
    - Phân đoạn các phần tử lớn hơn chốt
  - Thực hiện lại các bước trên cho hai phân đoạn trước và sau chốt, cho đến khi phân đoạn chỉ gồm một phần tử thì dừng lại

Kiểu mảng

- Sắp xếp nhanh (quicksort)

```
void quicksort(int a[], int l, int r)
{
        int i, j, chot;
        if (l < r){
                i = l+1;
                j = r;
                chot = a[l];
                while (i < j){
                        while(a[i] < chot) i++;
                        while(a[j] > chot) j--;
                        if (i < j) hoanvi(&a[i], &a[j]);
                }
                hoanvi(&a[j], &a[l]);
                quicksort(a, l, j-1);
                quicksort(a, j+1, r);
        }
}
```


Kiểu mảng

- Tìm kiếm phần tử trong mảng
  - Tìm sự xuất hiện của một phần tử trong mảng
  - Hai phương pháp cơ bản
    - Tìm kiếm tuần tự
    - Tìm kiếm nhị phân

Kiểu mảng

- Tìm kiếm tuần tự
  - Duyệt từ đầu mảng đến cuối mảng để tìm sự xuất hiện của một phần tử

```
int timkiem_tuantu(int a[], int n, int x)
{
        int i;
        i = 0;
        while ((i < n) && (a[i] != x)) i++;
        /* nếu i == n thì không tìm thấy x */
        return(i);
}
```


Kiểu mảng

- Tìm kiếm nhị phân
  - Áp dụng đối với mảng đã được sắp xếp
  - Ý tưởng
    - Giả sử mảng đã được sắp xếp tăng dần
    - Lấy phần tử cần tìm so sánh với phần tử giữa mảng (gọi là *g*), có các khả năng xảy ra:
      - Nếu phần tử cần tìm lớn hơn *g*, thì chỉ tìm nữa cuối mảng
      - Nếu phần tử cần tìm nhỏ hơn *g*, thì chỉ tìm nữa đầu mảng
      - Nếu không, *g* chính là phần tử cần tìm

Kiểu mảng

- Tìm kiếm nhị phân

```
int timkiem_nhiphan(int a[], int n, int x)
{
        int t, p, g;
        t = 0;
        p = n-1;
        while (t <= p)
        {
                g = (t + p)/2;
                if (x < a[g]) p = g - 1;
                else if (x > a[g]) t = g + 1;
                        else return(g);
        }
        return(n); /* trường hợp không tìm thấy x */
}
```


Kiểu mảng

- Mảng nhiều chiều
  - Ví dụ, khai báo mảng hai chiều

    int a[4][10];

    là mảng có 4 hàng, 10 cột
  - Truy cập các phần tử của mảng

    a[0][0], a[0][1], a[i][j]…
  - Ví dụ khác

    float arr[3][4][5];

    char arrc[4][4];

Kiểu mảng

- Mảng nhiều chiều
  - Ví dụ

```
/* Hàm nhập mảng các số nguyên */
void nhap_ma_tran(int a[][Max_Cot], int m, int n)
{
    int i, j;
    int x;
    for (i=0; i < m; i++){
          printf("\n Nhap hang thu %2d\n", i);
          for (j=0; j < n; ++j){
                    printf("pt[%d][%d]", i, j);
                    scanf("%d", &x);
                    a[i][j] = x;
          }
  }
}
```


Kiểu mảng

- Mảng nhiều chiều
  - Ví dụ

```
/* Hiển thị các phần tử của mảng */
void in_ma_tran (int a[][Max_Cot], int m, int n);
{
  int i, j;
  for (i=0; i < m; i++){
          for (j=0; j < n; ++j)
                    printf("%4d", a[i][j]);
          printf("\n");
  }
}
```

Kiểu mảng

- Mảng nhiều chiều
  - Ví dụ

```
/* Tính tổng 2 mảng các số nguyên */
void tinh_tong(int a[][Max_Cot], int b[][Max_Cot],
               int c[][Max_Cot], 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];
}
```

  - Bài tập ?