Chương 7: Giới thiệu tổng quan về lập trình

Phần b: Mảng

Nhập môn lập trình

Trình bày: Nguyễn Sơn Hoàng Quốc
Email: nshquoc@fit.hcmus.edu.vn

Nội dung

- Dữ liệu có cấu trúc
- Dữ liệu mảng với kích thước cố định
- Ứng dụng mảng trong lập trình
- Các vấn đề tìm hiểu mở rộng kiến thức nghề nghiệp
- Thuật ngữ và bài đọc thêm tiếng Anh

DỮ LIỆU MẢNG VỚI KÍCH THƯỚC CỐ ĐỊNH

Dữ liệu kiểu mảng

- Khái niệm
  - Là một kiểu dữ liệu có cấu trúc do người lập trình định nghĩa.
  - Biểu diễn một dãy các biến có cùng kiểu. Ví dụ: dãy các số nguyên, dãy các ký tự…
  - Kích thước được xác định ngay khi khai báo và không bao giờ thay đổi.
  - NNLT C luôn chỉ định một khối nhớ liên tục cho một biến kiểu mảng.

Khai báo biến mảng 1 chiều

- Cú pháp tường minh

```
<kiểu cơ sở> <tên biến mảng>[<số phần tử>];
```

- Ví dụ

```
int a[100], b[200], c[100];
float d[50];
```


- Lưu ý
  - Phải xác định <số phần tử> cụ thể (hằng) khi khai báo.
  - Bộ nhớ sử dụng = <tổng số phần tử> * sizeof(<kiểu cơ sở>)
  - Là một dãy liên tục có chỉ số từ 0 đến <tổng số phần tử> - 1

Khai báo biến mảng 1 chiều

- Cú pháp tường minh

```
<kiểu cơ sở> <tên biến mảng>[<số phần tử>];
```

- Ví dụ

```
int a[100], b[200], c[100];
float d[50];
```

- Áp dụng
  - Khai báo mảng một chiều:
    1. Các phần tử kiểu số nguyên không dấu
    2. Các phần tử kiểu phân số

Khởi tạo mảng 1 chiều

- Sử dụng một trong 4 cách sau:
  - Khởi tạo giá trị cho mọi phần tử của mảng

```
int a[4] = {2912, 1706, 1506, 1904};
```

  - Khởi tạo giá trị cho một số phần tử đầu mảng

```
int a[4] = {2912, 1706};
```

  - Tự động xác định số lượng phần tử

```
int a[] = {2912, 1706, 1506, 1904};
```

Khởi tạo mảng 1 chiều

- Sử dụng một trong 4 cách sau:
  - Khởi tạo giá trị cho mọi phần tử của mảng

```
int a[4] = {2912, 1706, 1506, 1904};
```

  - Khởi tạo giá trị cho một số phần tử đầu mảng

```
int a[4] = {2912, 1706};
```

  - Tự động xác định số lượng phần tử

```
int a[] = {2912, 1706, 1506, 1904};
```

- Áp dụng
  1. Khai báo mảng 12 phần tử chứa số ngày trong tháng

Truy xuất mảng 1 chiều

- Thông qua chỉ số:

<tên biến mảng>[<chỉ số>]

- Ví dụ cho mảng `int a[4];`
  - Các truy xuất hợp lệ: `a[0],a[1],a[2],a[3]`
  - Các truy xuất không hợp lệ:`a[-1],a[4],a[5]`

Truy xuất mảng 1 chiều

- Thông qua chỉ số:

<tên biến mảng>[<chỉ số>]

- Ví dụ cho mảng `int a[4];`
  - Các truy xuất hợp lệ: `a[0],a[1],a[2],a[3]`
  - Các truy xuất không hợp lệ:`a[-1],a[4],a[5]`
- Áp dụng
  - Viết đoạn chương trình khai báo mảng 3 phần tử có các giá trị 1, 2, 3, tính tổng của chúng

Gán dữ liệu mảng 1 chiều

- Không được sử dụng phép gán thông thường mà phải gán trực tiếp giữa các phần tử tương ứng
- Ví dụ

```
1. int a[3] = {1, 2, 3}, b[3];

2. void main()
3. {
4.     b = a;        // sai
5.     for (int i = 0; i < 3; i++)
6.           b[i] = a[i];
7. }
```

Truyền mảng 1 chiều cho hàm

- Tham số kiểu mảng truyền cho hàm chính là địa chỉ của phần tử đầu tiên của mảng:
  - Có thể bỏ số lượng phần tử (hoặc sử dụng con trỏ), số lượng phần tử thực sự truyền kèm theo.
  - Mảng có thể thay đổi nội dung sau khi thực hiện hàm.
- Ví dụ

```
void sort(int a[100], int n);
```

Truyền mảng 1 chiều cho hàm

- Ví dụ

```
void sort(int a[100], int n);
```

- Áp dụng
  - Viết hàm tính tích các phần tử mảng một chiều gồm n phần tử

Chương trình

- Viết chương trình nhập và xuất mảng một chiều n phần tử các số nguyên. Tính tổng của chúng và xuất kết quả

Khối khai báo

```
1. #include <iostream>
2. using namespace std;
3. #define MAXN 100

4. void Input(int a[MAXN], int &n);
5. void Output(int a[MAXN], int n);
6. int CalculateSum(int a[MAXN], int n);
```

Khối hàm main

```
1. void main()
2. {
3.    int a[MAXN], n, sum;
4.
5.    cout << "Input array = " << endl;
6.    Input(a, n);
7.    cout << "Array = " << endl;
8.    Output(a, n);
9.
10.   sum = CalculateSum(a, n);
11.
12.   cout << "Sum = " << sum;
13.}
```

Khối định nghĩa hàm

```
1. void Input(int a[MAXN], int &n)
2. {
3.     cout << "Number of elements = ";
4.     cin >> n;
5.
6.     for (int i = 0; i < n; i++)
7.     {
8.         cout << "a[" << i << "] = ";
9.         cin >> a[i];
10.    }
11.}
```

Khối định nghĩa hàm

```
1. void Output(int a[MAXN], int n)
2. {
3.     for (int i = 0; i < n; i++)
4.         cout << a[i] << "\t";
5. }
```

Khối định nghĩa hàm

```
1. int CalculateSum(int a[MAXN], int n)
2. {
3.     int sum = 0;
4.
5.     for (int i = 0; i < n; i++)
6.         sum = sum + a[i];
7.
8.     return sum;
9. }
```

Ví dụ áp dụng

- Áp dụng
  - Viết chương trình nhập vào mảng một chiều n phần tử ($n \geq 1$) các số thực, tìm giá trị trung bình của mảng và xuất kết quả

Xử lý mảng 1 chiều

- Một số thao tác cơ bản
  - Nhập/xuất mảng
  - Tìm kiếm một phần tử trong mảng
  - Kiểm tra tính chất của mảng
  - Chia/gộp mảng
  - Tìm giá trị nhỏ nhất/lớn nhất trong mảng
  - Sắp xếp mảng
  - Thêm/xóa/sửa một phần tử trong mảng

Nhập/xuất mảng

- Nhập mảng một chiều các số nguyên

```
1. void Nhap(int a[100], int &n)
2. {
3.    cout << "Nhap n = ";
4.    cin >> n;
5.    for (int i = 0; i < n; i++)
6.    {
7.          cout << "a[" << i << "]=";
8.          cin >> a[i];
9.    }
10.}
```

Nhập/xuất mảng

```
1. void Nhap(int a[100], int &n)
2. {
3.    cout << "Nhap n = ";
4.    cin >> n;
5.    for (int i = 0; i < n; i++)
6.    {
7.          cout << "a[" << i << "]=";
8.          cin >> a[i];
9.    }
10.}
```

Áp dụng: Nhập mảng một chiều các số thực

Nhập/xuất mảng

```
1. void Nhap(int a[100], int &n)
2. {
3.    cout << "Nhap n = ";
4.    cin >> n;
5.    for (int i = 0; i < n; i++)
6.    {
7.          cout << "a[" << i << "]=";
8.          cin >> a[i];
9.    }
10.}
```

Áp dụng: Xuất mảng một chiều các số nguyên

Tìm kiếm một phần tử trong mảng

- Viết hàm trả về chỉ số phần tử x đầu tiên trong mảng

```
1. int TimPhanTu(int a[100], int n, int x)
2. {
3.    int chiSo = -1;
4.    for (int i = 0; i < n; i++)
5.         if (chiSo == -1 && a[i] == x)
6.              chiSo = i;
7.    return chiSo;
8. }
```

Tìm kiếm một phần tử trong mảng

- Viết hàm trả về chỉ số phần tử x đầu tiên trong mảng

```
1. int TimPhanTu(int a[100], int n, int x)
2. {
3.    int chiSo = -1;
4.    for (int i = 0; i < n; i++)
5.          if (chiSo == -1 && a[i] == x)
6.                chiSo = i;
7.    return chiSo;
8. }
```

- **Áp dụng:** Tìm chỉ số phần tử chẵn cuối cùng trong mảng

Kiểm tra tính chất của mảng

- Kiểm tra một mảng có tăng dần hay không

```
1. bool KiemTraTang(int a[100], int n)
2. {
3.     bool tang = true;
4.
5.     for (int i = 1; i < n; i++)
6.         if (a[i] < a[i - 1])
7.             tang = false;
8.
9.     return tang;
10.}
```

Kiểm tra tính chất của mảng

- Kiểm tra một mảng có tăng dần hay không

```
1.  bool KiemTraTang(int a[100], int n)
2.  {
3.      bool tang = true;
4.
5.      for (int i = 1; i < n; i++)
6.          if (a[i] < a[i - 1])
7.              tang = false;
8.
9.      return tang;
10. }
```

- Áp dụng: Kiểm tra một mảng có chứa toàn số chẵn hay không

Chia/gộp mảng

- Chia a thành 2 mảng dương và mảng âm

```
1.  void ChiaMang(  int a[100], int n, int duong[100],
                    int &n1, int am[100], int &n2)
2.  {
3.      n1 = 0;
4.      n2 = 0;
5.      for (int i = 0; i < n; i++)
6.          if (a[i] > 0)
7.          {
8.              duong[n1] = a[i];
9.              n1++;
10.         }
11.         else if (a[i] < 0)
12.         {
13.             am[n2] = a[i];
14.             n2++;
15.         }
16. }
```

Chia/gộp mảng

Áp dụng: Gộp mảng a có n phần tử và mảng b có m phần tử thành mảng c có n+m phần tử với các phần tử đầu là mảng a, các phần tử cuối là mảng b.

Tìm giá trị nhỏ nhất/lớn nhất trong mảng

- Viết hàm tìm kiếm chỉ số phần tử nhỏ nhất trong mảng

```
1. int TimNhoNhat(int a[100], int n)
2. {
3.    int chiSo = -1;
4.
5.    for (int i = 1; i < n; i++)
6.         if (chiSo == -1 || a[i] < a[chiSo])
7.              chiSo = i;
8.
9.    return chiSo;
10.}
```

Tìm giá trị nhỏ nhất/lớn nhất trong mảng

- Viết hàm tìm kiếm chỉ số phần tử nhỏ nhất trong mảng

```
1.  int TimNhoNhat(int a[100], int n)
2.  {
3.      int chiSo = -1;
4.
5.      for (int i = 1; i < n; i++)
6.          if (chiSo == -1 || a[i] < a[chiSo])
7.              chiSo = i;
8.
9.      return chiSo;
10. }
```

- **Áp dụng:** Tìm chỉ số phần tử có trị tuyệt đối lớn nhất trong mảng

Sắp xếp mảng

- Sắp mảng tăng

```
1.  void SapTang(int a[100], int n)
2.  {
3.     for (int i = 0; i < n - 1; i++)
4.           for (int j = i + 1; j < n; j++)
5.                 if (a[i] > a[j])
6.                 {
7.                       int tam = a[i];
8.                       a[i] = a[j];
9.                       a[j] = tam;
10.                }
11. }
```

Sắp xếp mảng

- Sắp mảng tăng

```
1.  void SapTang(int a[100], int n)
2.  {
3.     for (int i = 0; i < n - 1; i++)
4.           for (int j = i + 1; j < n; j++)
5.              if (a[i] > a[j])
6.              {
7.                    int tam = a[i];
8.                    a[i] = a[j];
9.                    a[j] = tam;
10.             }
11. }
```

- Áp dụng : Sắp các phần tử của mảng tăng dần theo trị tuyệt đối

Thêm/xóa/sửa một phần tử trong mảng

- Thêm phần tử x và vị trí chiSo của mảng

```
1. void ThemPhanTu(  int a[100], int &n,
                     int x, int chiSo)
2. {
3.    if (chiSo <= n-1)
4.    {
5.          for (int i = n - 1; i > chiSo; i--)
6.                a[i] = a[i - 1];
7.          a[chiSo] = x;
8.          n++;
9.    }
10.}
```

Thêm/xóa/sửa một phần tử trong mảng

- Thêm phần tử x và vị trí chiSo của mảng

```
1.  void ThemPhanTu(     int a[100], int &n,
                         int x, int chiSo)
2.  {
3.      if (chiSo <= n-1)
4.      {
5.              for (int i = n - 1; i > chiSo; i--)
6.                      a[i] = a[i - 1];
7.              a[chiSo] = x;
8.              n++;
9.      }
10. }
```


- Áp dụng : Xóa phần tử tại vị trí chiSo của mảng

Thêm/xóa/sửa một phần tử trong mảng

- Thêm phần tử x và vị trí chiSo của mảng

```
1.  void ThemPhanTu(int a[100], int &n, int x, int chiSo)
2.  {
3.      if (chiSo <= n-1)
4.      {
5.              for (int i = n - 1; i > chiSo; i--)
6.                      a[i] = a[i - 1];
7.              a[chiSo] = x;
8.              n++;
9.      }
10. }
```


- Áp dụng : Đổi dấu các phần tử âm thành dương, dương thành âm

Mảng 2 chiều

- Mảng 2 chiều giống như một ma trận gồm nhiều dòng và nhiều cột giao nhau tạo thành các ô, mỗi ô là một phần tử mảng.
- Mọi thao tác xử lý trên mảng 2 chiều hoàn toàn tương tự trên mảng 1 chiều.
- Tạm thời giới hạn trong phạm vi mảng 2 chiều tĩnh (số dòng và cột cố định).

(Xem trong giáo trình NMLT trang 203-221)

ỨNG DỤNG MẢNG TRONG LẬP TRÌNH

Một số ứng dụng

- Kỹ thuật dùng bảng tra cứu trong bộ nhớ để cải tiến tính toán và xử lý.
- Kỹ thuật dùng cờ hiệu khi xử lý mảng.
- Thuật toán tìm kiếm và tính toán trên mảng.
- Thuật toán xáo trộn, sắp xếp các phần tử của mảng.

CÁC VẤN ĐỀ TÌM HIỂU MỞ RỘNG KIẾN THỨC NGHỀ NGHIỆP

Tìm hiểu thêm

- Sử dụng mảng kích thước biến động.
- Qui hoạch động và ứng dụng để giải các bài toán tối ưu.
- Các thuật toán chia để trị.

THUẬT NGỮ
VÀ BÀI ĐỌC THÊM TIẾNG ANH

Thuật ngữ tiếng Anh

- ***array parameter(s), array argument(s)***: tham số mảng
- ***array size***: kích thước mảng
- ***column***: cột
- ***copy***: sao chép
- ***data type declaration, data type definition***: khai báo kiểu dữ liệu
- ***dynamic array***: mảng động
- ***element***: phần tử
- ***implementation***: cài đặt (viết mã nguồn)
- ***index***: chỉ số
- ***insert***: chèn vào
- ***one-dimension array***: mảng một chiều
- ***two-dimension array***: mảng hai chiều
- ***merge***: trộn lại

Thuật ngữ tiếng Anh

- ***remove, delete***: xóa đi
- ***row***: dòng
- ***split***: tách ra
- ***static array***: mảng tĩnh
- ***structured data***: dữ liệu có cấu trúc nói chung

Bài đọc thêm tiếng Anh

- **Thinking in C**, Bruce Eckel, E-book, 2006.
- **Theory and Problems of Fundamentals of Computing with C++**, John R.Hubbard, Schaum's Outlines Series, McGraw-Hill, 1998.

PHỤ LỤC

- Khai báo mảng dùng typedef

Khai báo biến mảng 1 chiều

- Cú pháp (không tường minh)

```
typedef <kiểu cơ sở> <tên kiểu mảng>[<số lượng phần tử>];
<tên kiểu mảng> <tên biến mảng>;
```

- Ví dụ

```
typedef int Arr100int[100];
typedef int Arr200int[200];
typedef float Arr50float[50];
Arr100int a, c;        // int a[100], c[100];
Arr200int b;           // int b[200];
Arr50float d;          // float d[50];
```

Khai báo biến mảng 1 chiều

- Cú pháp (không tường minh)

```
typedef <kiểu cơ sở> <tên kiểu mảng>[<số lượng phần tử>];
<tên kiểu mảng> <tên biến mảng>;
```

- Ví dụ

```
typedef int Arr100int[100];
typedef float Arr50float[50];
Arr100int a, c;        // int a[100], c[100];
```

- Áp dụng
  - Khai báo
    1. Mảng một chiều tối đa 150 ký tự
    2. Mảng một chiều tối đa 50 số thực dài