Chương 7: Giới thiệu tổng quan về lập trình

Phần a: Dữ liệu có cấu trúc

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 CÓ CẤU TRÚC

Đặt vấn đề

- Khai báo các biến để lưu trữ 1 SV

```
char mssv[8];        // "0912345"
char hoten[30];      // "Nguyen Van A"
char ntns[9];        // "01/01/91"
char phai;           // ‘y’:Nam ;‘n’:Nu
float toan, ly, hoa;// 8.5 9.0 10.0
```

- Truyền thông tin 1 SV cho hàm

```
void xuat(char* mssv,char* hoten,char* ntns,
   char phai,float toan,float ly,float hoa);
```

Đặt vấn đề

- Nhận xét
  - Đặt tên biến khó khăn và khó quản lý
  - Truyền tham số cho hàm quá nhiều
  - Tìm kiếm, sắp xếp, sao chép,… khó khăn
  - Tốn nhiều bộ nhớ
  - …
- Ý tưởng
  - Gom những thông tin của cùng 1 SV thành một kiểu dữ liệu mới => Kiểu struct

Khai báo kiểu cấu trúc

- Cú pháp

```
struct <tên kiểu cấu trúc>
{
   <kiểu dữ liệu> <tên thành phần 1>;
   …
   <kiểu dữ liệu> <tên thành phần n>;
};
```

- Ví dụ

```
struct Point2D
{
   int x;
   int y;
};
```

Khai báo kiểu cấu trúc

- Ví dụ

```
struct Point2D
{
    int x;
    int y;
};
```

- Áp dụng
  - Khai báo các kiểu dữ liệu sau:
    1. Phân số
    2. Hỗn số
    3. Điểm không gian
    4. Đơn thức
    5. Ngày

Khai báo biến

- Cú pháp khai báo tường minh

```
struct <tên kiểu cấu trúc>
{
    <kiểu dữ liệu> <tên thành phần 1>;
    …
    <kiểu dữ liệu> <tên thành phần n>;
} <tên biến 1>, <tên biến 2>;
```

- Ví dụ

```
struct Point2D
{
    int x;
    int y;
} p1, p2;
```

Khai báo biến

- Ví dụ

```
1. struct Point2D
2. {
3.     int x;
4.     int y;
5. } p1, p2;
```

- Áp dụng
  - Khai báo biến của các kiểu dữ liệu sau:
    1. Thời gian (giờ, phút, giây)
    2. Đường thẳng (có dạng ax + by + c = 0)

Khai báo biến

- Cú pháp khai báo không tường minh

```
struct <tên kiểu cấu trúc> {
    <kiểu dữ liệu> <tên thành phần 1>;
    …
    <kiểu dữ liệu> <tên thành phần n>;
};
struct <tên kiểu cấu trúc> <tên biến 1>, <tên biến 2>;
```

- Ví dụ

```
struct Point2D {
    int x;
    int y;
};
struct Point2D p1, p2;     // C++ có thể bỏ từ khóa struct
```

Khai báo biến

- Ví dụ

```
1. struct Point2D {
2.     int x;
3.     int y;
4. };
5. struct Point2D p1, p2; // C++ có thể bỏ từ struct
```

- Áp dụng
  - Khai báo không tường minh các kiểu dữ liệu sau:
    1. Tam giác (với độ dài ba cạnh)
    2. Tam giác (với tọa độ ba điểm)

Sử dụng typedef

- Cú pháp

```
typedef struct {
    <kiểu dữ liệu> <tên thành phần 1>;
    …
    <kiểu dữ liệu> <tên thành phần n>;
} <tên kiểu cấu trúc>;
<tên kiểu cấu trúc> <tên biến 1>, <tên biến 2>;
```

- Ví dụ

```
typedef struct {
    int x;
    int y;
} Point2D;
Point2D p1, p2;
```

Sử dụng typedef

- Ví dụ

```
1. typedef struct {
2.       int x;
3.       int y;
4. } Point2D;
5. Point2D p1, p2;
```

- Áp dụng
  - Sử dụng typedef khai báo các biến dữ liệu sau:
    1. Khai báo hình cầu
    2. Cầu thủ (với các thông tin mã cầu thủ 30 ký tự, tên cầu thủ tối đa 30 ký tự, ngày sinh)

Khởi tạo cho biến cấu trúc

- Cú pháp

```
struct <tên kiểu cấu trúc> {
    <kiểu dữ liệu> <tên thành phần 1>;
    …
    <kiểu dữ liệu> <tên thành phần n>;
} <tên biến> = {<giá trị 1>, <giá trị 2>, …, <giá trị n>};
```

- Ví dụ

```
struct Point2D {
    int x;
    int y;
} p1= {2912, 1706}, p2;
```

Khởi tạo cho biến cấu trúc

- Ví dụ

```
1. struct Point2D {
2.      int x;
3.      int y;
4. } p1= {2912, 1706}, p2;
```

- Áp dụng
  - Khởi tạo cho các biến sau:
    1. Số phức
    2. Điểm không gian

Truy xuất

- Đặc điểm
  - Không thể truy xuất trực tiếp.
  - Thông qua toán tử thành phần cấu trúc . Hay còn gọi là toán tử chấm (dot operation).

```
<tên biến cấu trúc>.<tên thàn phần>
```

- Ví dụ

Truy xuất

1. Ví dụ

```
2. struct Point2D {
3.     int x, y;
4. } p = {2912, 1706};

5. void show(Point2D p)
6. {
7.     printf("x = %d, y = %d\n", p.x, p.y);
8. }
```

Truy xuất

- Ví dụ

```
1. struct Point2D {
2.     int x;
3.     int y;
4. } p = {2912, 1706};

5. void show(Point2D p)
6. {
7.     printf("x = %d, y = %d\n", p.x, p.y);
8. }
```

- Ví dụ áp dụng: Tính khoảng cách hai điểm không gian

Gán dữ liệu

- Có 2 cách

<biến cấu trúc đích> = biến cấu trúc nguồn

<biến cấu trúc đích>.<tên thành phần> = <giá trị>

- Ví dụ


```
struct Point2D {
    int x, y;
} p1 = {2912, 1706}, p2;
void main() {
    p2 = p1;
    p2.x = p1.x;
    p2.y = p1.y * 2;
}
```

Gán dữ liệu

- Ví dụ

```
1. struct Point2D {
2.      int x, y;
3. } p1 = {2912, 1706}, p2;
4. void main() {
5.      p2 = p1;
6.      p2.x = p1.x;
7.      p2.y = p1.y * 2;
   }
```

- Áp dụng : Viết hàm tìm trung điểm của hai điểm trong mặt phẳng

Ví dụ tìm trọng tâm tam giác

- Các định nghĩa hàm

```
1.  void gravCenter(Triangle t, Point2D& p)
2.  {
3.  p.x = (t.ver[0].x + t.ver[1].x + t.ver[2].x) / 3;
4.  p.y = (t.ver[0].y + t.ver[1].y + t.ver[2].y) / 3;
5.  }

6.  void inputTriangle(Point2D& p)
7.  {
8.      for (int i  = 0; i < 3; i++)
9.      {
10.             cout << "Vertex " << i + 1 << ": " << endl;
11.             inputPoint2D(t.ver[i]);
12.     }
13. }
```

Ví dụ về phân số

- Các khai báo cần thiết

```
1. #include <iostream>
2. using namespace std;
3. typedef struct {
4.     long num, denom;
5. } Fraction;
6. void greatestDivisor(long a, long b);
7. void reduce(Fraction& p);
8. Fraction add(Fraction p, Fraction q);
9. Fraction sub(Fraction p, Fraction q);
10.void showFraction(Fraction p);
```

Ví dụ về phân số

- Các định nghĩa hàm

```
1. void greatestDivisor(long a, long b) {
2.      // Viết như các ví dụ trước…
3. }
4. void reduce(Fraction& p) {
5.      long gcd = greatestDivisor(p.num, p.denom);
6.      p.num /= gcd; p.denom /= gcd;
7. }
8. Fraction add(Fraction p, Fraction q) {
9.      Fraction r;
10.     r.num = p.num * q.denom + p.denom * q.num;
11.     r.denom = p.denom * q.denom;
12.     return r;
13.}
```

Ví dụ về phân số

- Các định nghĩa hàm

```
1. Fraction sub(Fraction p, Fraction q)
2. {
3.     q.num = -q.num;
4.     return add(p, q);
5. }

6. void showFraction(Fraction p)
7. {
8.     reduce(p); // Tối giản trước khi in ra
9.     cout << p.num << "/" << p.denom;
10.}
```

Ví dụ áp dụng

- Viết chương trình nhập vào một đơn thức và một giá trị của biến x. Tính giá trị của đơn thức đó và xuất kết quả.