Kỹ Thuật Lập Trình

KIỂU CẤU TRÚC (STRUCTURE)

1

2

cấu trúc (structure)

- Cấu trúc dùng lưu tập hợp các đối tượng không cùng kiểu.
- Khai báo:

struct <tên kiểu>
{
các thành phần ;
} <danh sách biến>;

- Mỗi thành phần giống như một biến riêng của kiểu, nó gồm kiểu và tên thành phần. Một thành phần cũng còn được gọi là trường.
- Các kiểu cấu trúc được phép khai báo lồng nhau
- Một biến có kiểu cấu trúc sẽ được phân bố bộ nhớ sao cho các thực hiện của nó được sắp liên tục theo thứ tự xuất hiện trong khai báo.

3

3

cấu trúc (structure)

- Khai báo biến kiểu cấu trúc cũng giống như khai báo các biến kiểu cơ sở dưới dạng:

struct <tên cấu trúc> <danh sách biến> ; // kiểu cũ trong C
<tên cấu trúc> <danh sách biến> ; // trong C++

Các biến được khai báo cũng có thể đi kèm khởi tạo:

<tên cấu trúc> biến = { giá trị khởi tạo } ;

Ví dụ:

Kiểu ngày tháng gồm 3 thành phần nguyên chứa ngày, tháng, năm.

```
struct Ngaythang {
    int ng ;
    int th ;
    int nam ;
} holiday = { 1,5,2000 } ;
```

4

4

cấu trúc (structure)

- **Ví dụ:**

```
struct Sinhvien {
    char hoten[25] ;
    Ngaythang ns;
    int gt;     //qui ước 1:Nam; 2:Nữ
    float diem ;
} x, *p, K41T[60];

Sinhvien y = {"NVA", {1,1,1980}, 1} ;
```

5

5

cấu trúc (structure)

- **Truy nhập các thành phần kiểu cấu trúc**
  - Đối với biến thường: **<tên biến>.<tên thành phần>**
- Ví dụ:

```
struct Sinhvien {
    char hoten[25] ;
    Ngaythang ns;
    int gt;
    float diem ;
} x, *p, K41T[60];
Sinhvien y = {"NVA", {1,1,1980}, 1} ;
y.diem = 5.5 ;
```

  - Đối với biến con trỏ: **tên biến ->tên thành phần**

```
p = new Sinhvien ;              // cấp bộ nhớ chứa 1 sinh viên
strcpy(p->hoten, y.hoten) ;     // gán họ tên của y cho sv trỏ bởi p
```

6

6

cấu trúc (structure)

- **Truy nhập các thành phần kiểu cấu trúc**
  - **Đối với biến mảng: truy nhập thành phần mảng rồi đến thành phần cấu trúc.**

Ví dụ:

```
strcpy(K41T[1].hoten, p->hoten) ; // gán họ tên cho sv
                                  //đầu tiên của lớp
K41T[1].diem = 7.0 ;          // gán điểm cho sv đầu tiên
```

  - **Đối với cấu trúc lồng nhau. Truy nhập thành phần ngoài rồi đến thành phần của cấu trúc bên trong**

```
x.ns.ng = y.ns.ng ;    // gán ngày,
x.ns.th = 10 ;                // tháng,
x.ns.nam = 1981 ;             // năm sinh của y cho x.
```

7

7

cấu trúc (structure)

- **Phép toán gán cấu trúc**

Cũng giống các biến mảng, để làm việc với một biến cấu trúc chúng ta phải thực hiện thao tác trên từng thành phần của chúng

```
struct Sinhvien {
    char hoten[25] ;
    Ngaythang ns;
    int gt;  float diem ;
    } x, y;
cout << " Nhập dữ liệu cho sinh viên x:" << endl ;
fflush(stdin); gets(x.hoten);
cin >> x.ns.ng >> x.ns.th >> x.ns.nam;
cin >> x.gt;  cin >> x.diem
```

8

8

cấu trúc (structure)

- **Phép toán gán cấu trúc**

```
cout << "Thông tin về sinh viên x là:" << endl ;
cout << "Họ và tên: " << x.hoten << endl;
cout << "Sinh ngày: " << x.ns.ng << "/" << x.ns.th << "/" << x.ns.nam ;
cout << "Giới tính: " << (x.gt == 1) ? "Nam": "Nữ ;
cout << x.diem
```

- Tuy nhiên, khác với biến mảng, **đối với cấu trúc chúng ta có thể gán giá trị của 2 biến cho nhau**

```
y = x ;  // Đối với biến mảng, phép gán này là không thực hiện được
p = new Sinhvien[1] ; *p = x ;
```

- Chú ý: *không gán bộ giá trị cụ thể cho biến cấu trúc. Cách gán này chỉ thực hiện được khi khởi tạo.*

```
Sinhvien y,x = { "NVA", {1,1,1980}, 1, 7.0} ;          // được
y = { "NVA", {1,1,1980}, 1, 7.0};                      // không được
y = x;                                                 // được
```

9

9

cấu trúc (structure)

- **Các ví dụ minh hoạ**

Nhập mảng K41T. Tính điểm trung bình của sinh viên nam, nữ. Hiện danh sách của sinh viên có điểm thi cao nhất.

```
#include <iostream.h>
#include <conio.h>
void main()
{
    struct Sinhvien {
        char hoten[25] ;
        Ngaythang ns;
        int gt;
        float diem ;
    } x, K41T[60];
    int i, n;
```

10

10

cấu trúc (structure)

- **Các ví dụ minh hoạ (cont)**

```
// nhập dữ liệu
cout << "Cho biết số sinh viên: "; cin >> n;
for (i=1, i<=n, i++)
{   cout << "Nhap sinh vien thu " << i);
    cout << "Ho ten: " ; fflush(stdin); gets(x.hoten);
    cout << "Ngày sinh: " ; cin >> x.ns.ng >> x.ns.th
    >>x.ns.nam ;
    cout << "Giới tính: " ; cin >> x.gt ;
    cout << "Điểm: " ; cin >> x.diem ;
    K41T[i] = x ;
}
```

11

11

cấu trúc (structure)

- **Các ví dụ minh hoạ (cont)**

```
// Tính điểm trung bình

    float tbnam = 0, tbnu = 0;
    int sonam = 0, sonu = 0 ;
    for (i=1; i<=n; i++)
        if (K41T[i].gt == 1) { sonam++ ; tbnam += K41T[i].diem ; }
        else { sonu++ ; tbnu += K41T[i].diem ; }
    cout << "Điểm trung bình của sinh viên nam là " <<
    tbnam/sonam ;
    cout << "Điểm trung bình của sinh viên nữ là " << tbnu/sonu ;
```

12

12

cấu trúc (structure)

- **Ví dụ minh hoạ (cont)**

```
// In danh sách sinh viên có điểm cao nhất
    float diemmax = 0;
    for (i=1; i<=n; i++)            // Tìm điểm cao nhất
        if (diemmax < K41T[i].diem) diemmax = K41T[i].diem ;
    for (i=1; i<=n; i++)            // In danh sách
    {   if (K41T[i].diem < diemmax) continue ;
        x = K41T[i] ;
        cout << x.hoten << '\t' ;
        cout << x.ns.ng << "/" << x.ns.th << "/" << x.ns.nam << '\t' ;
        cout << (x.gt == 1) ? "Nam": "Nữ" << '\t' ;
        cout << x.diem << endl;
    }
}   //end main()
```

13

13

Mảng các cấu trúc

- Khai bao biến mảng các cấu trúc cũng giống như khai báo biến mảng trên các kiểu dữ liệu cơ bản khác.
- Dùng tên mảng khi truy cập từng phần tử trong mảng các cấu trúc và toán tử thành viên để truy cập các trường dữ liệu của từng thành phần cấu trúc.

14

14

Hàm với cấu trúc

- ***Con trỏ và địa chỉ cấu trúc***
  - Một con trỏ cấu trúc cũng giống như con trỏ trỏ đến các kiểu dữ liệu khác, có nghĩa nó chứa địa chỉ của một biến cấu trúc hoặc một vùng nhớ có kiểu cấu trúc nào đó
  - Gán địa chỉ của một biến cấu trúc, một thành phần của mảng, tương tự nếu địa chỉ của mảng gán cho con trỏ thì ta cũng gọi là con trỏ mảng cấu trúc
  - Ví dụ:

15

15

Hàm với cấu trúc

- ***Con trỏ và địa chỉ cấu trúc***

```
struct Sinhvien {
    char hoten[25] ;
    Ngaythang ns;
    int gt;
    float diem ;
} x, y, *p, lop[60];
p = &x ;                // cho con trỏ p trỏ tới biến cấu trúc x
P->diem = 5.0;          // gán giá trị 5.0 cho điểm của biến x
p = &lop[10] ;          // cho p trỏ tới sinh viên thứ 10 của lớp
cout << p->hoten;       // hiện họ tên của sinh viên này
*p = y ;                // gán lại sinh viên thứ 10 là y
(*p).gt = 2;            // sửa lại gt của sinh viên thứ 10 là nữ
```

16

16

Hàm với cấu trúc

- ***Con trỏ và địa chỉ cấu trúc***
  - Chú ý: Để truy nhập các thành phần của x được trỏ bởi con trỏ p. Khi đó *p là tương đương với x, do vậy ta dùng toán tử *p để lấy thành phần như (*p).hoten, (*p).diem,…
  - Con trỏ được khởi tạo do xin cấp phát bộ nhớ.
  - Ví dụ:

```
Sinhvien *p, *q ;
p = new Sinhvien[1];
q = new Sinhvien[60];
```

17

17

Hàm với cấu trúc

- ***Con trỏ và địa chỉ cấu trúc***

Đối với con trỏ p trỏ đến mảng a, chúng ta có thể sử dụng một số cách sau để truy nhập đến các trường của các thành phần trong mảng:

p[i].hoten;
(p+i)->hoten;
**(p+i).hoten;*

18

18

Hàm với cấu trúc

- ***Con trỏ và địa chỉ cấu trúc***

```
struct Sinhvien {
        char hoten[25] ;
        Ngaythang ns;
        int gt;
        float diem ;
} lop[60] ;

strcpy(lop[10].hoten, "NVA");
lop[10].gt = 1; lop[10].diem = 9.0 ;
Sinhvien *p ;                  // khai báo thêm biến con trỏ Sinh viên
p = lop ;                      // cho con trỏ p trỏ tới mảng lop
cout << p[10].hoten ;          // in họ tên sinh viên thứ 10
```

19

19

Hàm với cấu trúc

- ***Con trỏ và địa chỉ cấu trúc***

```
cout << (p+10) ->gt ;          // in giới tính của sinh viên thứ 10
cout << (*(p+10)).diem ;// in điểm của sinh viên thứ 10
```

Chú ý: *Độ ưu tiên của toán tử lấy thành phần (dấu chấm) là cao hơn các toán tử lấy địa chỉ (&) và lấy giá trị (*) nên cần phải viết các dấu ngoặc đúng cách.*

20

20

Hàm với cấu trúc

- ***Địa chỉ của các thành phần của cấu trúc***

Các thành phần của một cấu trúc cũng giống như các biến, do vậy cách lấy địa chỉ của các thành phần này cũng tương tự như đối với biến bình thường

- Ví dụ:

21

21

Hàm với cấu trúc

- ***Địa chỉ của các thành phần của cấu trúc***

Ví dụ:

```
struct Sinhvien {
      char hoten[25] ;
      Ngaythang ns;
      int gt; float diem ;
} lop[60], *p, x = { "NVA", {1,1,1980}, 1, 9.0) };
  lop[10] = x; p = &lop[10] ;// p trỏ đến sv thứ 10 trong lop
  char *ht; int *gt; float *d;// các con trỏ kiểu thành phần
  ht = x.hoten ;          // cho ht trỏ đến thành phần hoten của x
    gt = &(lop[10].gt) ;                  // gt trỏ đến gt của sinh viên thứ 10
  d = &(p->diem) ;              // d trỏ đến diem của sv p đang trỏ
  cout << ht ;                  // in họ tên sinh viên x
  cout << *gt ;                 // in giới tính của sinh viên thứ 10
  cout << *d ;                  // in điểm của sinh viên p đang trỏ.
```

22

22

Hàm với cấu trúc

- ***Đối của hàm là cấu trúc***

Một cấu trúc có thể được sử dụng để làm đối của hàm dưới các dạng sau đây:

- Là một biến cấu trúc, khi đó tham đối thực sự là một cấu trúc.
- Là một con trỏ cấu trúc, tham đối thực sự là địa chỉ của một cấu trúc.
- Là một tham chiếu cấu trúc, tham đối thực sự là một cấu trúc.
- Là một mảng cấu trúc hình thức hoặc con trỏ mảng, tham đối thực sự là tên mảng cấu trúc.

23

23

Hàm với cấu trúc

- ***Đối của hàm là cấu trúc***

Ví dụ: Chương trình đơn giản về quản lý sinh viên.

Khai báo.

```
struct Sinhvien {              // cấu trúc sinh viên
    char hoten[25] ;
    Ngaythang ns;
    int gt;
    float diem ;
} ;
Sinhvien lop[3];               // lớp chứa tối đa 3 sinh viên
```

24

24

Hàm với cấu trúc

- ***Đối của hàm là cấu trúc***

Ví dụ: Chương trình đơn giản về quản lý sinh viên.

Hàm in thông tin về sinh viên sử dụng biến cấu trúc làm đối.
Trong lời gọi sử dụng biến cấu trúc để truyền cho hàm.

```
void in(Sinhvien x)
{
    cout << x.hoten << "\t" ;
    cout << x.ns.ng << "/" << x.ns.th << "/" << x.ns.nam <<
    "\t" ;
    cout << x.gt << "\t";
    cout << x.diem << endl;
}
```

25

25

Hàm với cấu trúc

- ***Đối của hàm là cấu trúc***

Ví dụ: Chương trình đơn giản về quản lý sinh viên.

Hàm nhập thông tin về sinh viên sử dụng con trỏ sinh viên làm đối.
Trong lời gọi sử dụng địa chỉ của một cấu trúc để truyền cho hàm.

```
void nhap(Sinhvien *p)
{
    cout << "Họ tên: "; fflush(stdin);gets(p->hoten) ;
    cout << "Ngày sinh: ";
    cin >> (p->ns).ng >> (p->ns).th >> (p->ns).nam ;
    cout << "Giới tính: "; cin >> (p->gt) ;
    cout << "Điểm: "; cin >> (p->diem) ;
}
```

26

26

Hàm với cấu trúc

- ***Đối của hàm là cấu trúc***

Hàm sửa thông tin về SV sử dụng tham chiếu cấu trúc làm đối.
Trong lời gọi sử dụng biến cấu trúc để truyền cho hàm.

```
void sua(Sinhvien &r)
{
    int chon;
    do {
        cout << "1: Sửa họ tên" << endl ;
        cout << "2: Sửa ngày sinh" << endl ;
        cout << "3: Sửa giới tính" << endl ;
        cout << "4: Sửa điểm" << endl ;
        cout << "0: Thôi" << endl ;
        cout << "Sửa (0/1/2/3/4) ? ; cin >> chon ;
```

27

27

Hàm với cấu trúc

- ***Đối của hàm là cấu trúc***

```
do {
cout << "1: Sửa họ tên" << endl ;
cout << "2: Sửa NS" << endl
cout << "3: Sửa giới tính" << endl ;
cout << "4: Sửa điểm" << endl ;
cout << "0: Thôi" << endl ;
cout << "Sửa (0/1/2/3/4) ? ;
cin >> chon ;
```

```
switch (chon) {
  case 1: cin.getline(r.hoten, 25) ;
          break;
  case 2: cin >> r.ns.ng >> r.ns.th  >>
r.ns.nam ;          break;
  case 3: cin >> r.gt ; break;
  case 4: cin >> r.diem ; break;
}
} while (chon) ;
}
```

28

28

Hàm với cấu trúc

- ***Đối của hàm là cấu trúc***

Hàm **nhapds** nhập thông tin của mọi sinh viên trong mảng, sử dụng con trỏ mảng Sinhvien làm tham đối hình thức. Trong lời gọi sử dụng tên mảng để truyền cho hàm.

```
void nhapds(Sinhvien *a)
{
    int sosv = sizeof(lop) / sizeof(Sinhvien) -1; // bỏ phần tử 0
    for (int i=1; i<=sosv; i++) nhap(&a[i]) ;
}
```

29

29

Hàm với cấu trúc

- ***Đối của hàm là cấu trúc***

Hàm **suads** cho phép sửa thông tin của sinh viên trong mảng, sử dụng con trỏ mảng Sinhvien làm tham đối hình thức. Trong lời gọi sử dụng tên mảng để truyền cho hàm.

```
void suads(Sinhvien *a)
{
    int chon;
    cout << "Chọn sinh viên cần sửa: " ;
    cin >> chon ;
    sua(a[chon]) ;
}
```

30

30

Hàm với cấu trúc

- ***Đối của hàm là cấu trúc***

  Hàm **inds** hiện thông tin của mọi sinh viên trong mảng, sử dụng hằng con trỏ mảng Sinhvien làm tham đối hình thức. Trong lời gọi sử dụng tên mảng để truyền cho hàm.

```
Void  inds(const Sinhvien *a, int n)
{
//int sosv = sizeof(lop) / sizeof(Sinhvien) -1;   // bỏ phần tử 0
    for (int i=0; i<n; i++) in(a[i]) ;
    //if(a[i].hoten=="Nguyen Van An")
    //if(strcmpi(a[i].hoten,"Nguyen Van An")==0)
     //         in(a[i]) ;
}
```

31

31

Hàm với cấu trúc

- ***Đối của hàm là cấu trúc***

  Hàm **main()** gọi chạy các hàm trên để nhập, in, sửa danh sách sinh viên.

```
void main()
{
    nhapds(lop) ;
    inds(lop);
    suads(lop);
    inds(lop);
}
```

32

32