Chương V

Kiểu dữ liệu cấu trúc

ĐẠI HỌC GTVT TP HỒ CHÍ MINH

Nội dung chính

5.1 Khai báo, khởi tạo

5.2 Truy nhập các thành phần

5.3 Con trỏ với cấu trúc

5.4 Hàm với cấu trúc

5.5 Câu lệnh typedef và Hàm sizeof()

5.1 Khai báo, khởi tạo

Khai báo cấu trúc:

```
struct Tên_Cấu_trúc {
        Kiểu   Thành_phần_1;
        Kiểu   Thành_phần_2;
        …
} Danh sách biến;
```

- *Mỗi thành phần giống như một biến thông thường.*
- *Phần tên cấu trúc và danh sách biến có thể có hoặc không.*
- *Các kiểu cấu trúc được phép khai báo lồng nhau.*

5.1 Khai báo, khởi tạo

- 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;

Hoặc:

Tên cấu trúc Danh sách biến ;

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

  Tên cấu trúc Biến = { giá trị khởi tạo } ;

Ví dụ: Khai báo kiểu cấu trúc phân số:

```
struct Fraction {
        int numerator  ; // tử số
        int denominator ; //mẫu số      } ;
```

Ví dụ :

Khai báo kiểu cấu trúc Diem, sau đó khai báo 2 biến a, b kiểu Diem

```
struct Point {
        int x;    int y;
};
Point  a, b; //Khai báo biến
```

Khai báo cấu trúc, biến và khởi tạo cấu trúc mô tả đối tượng ngày

```
struct Date {
        int day;
        int month;
        int year;
} holiday = { 1,5,2016 } ;
```

Ví dụ :

Kiểu cấu trúc Employee (nhân viên)

```
struct Employee {
        char name[20]; // Tên nhân viên
        int age; // Tuổi nhân viên
        char role[20]; // Chức vụ của nhân viên
        float salary; // Lương của nhân viên
};
```

Khai báo và khởi tạo cho một biến::

```
Employee myEmployee1 = { “Nguyen Van A”, 27,
                         “Nhan vien”,
                         300 };
```

Ví dụ :

- Khai báo và khởi tạo biến có cấu trúc lồng nhau

  Ví dụ:

```
struct  Date {     int day;
                   int month;
                   int year };
struct Employee {  char name[20]; // Tên nhân viên
                   Date birthDay; // Ngày sinh của nhân viên
                   char role[20]; // Chức vụcủa nhân viên
                   float salary; // Lương của nhân viên  };
Employee myEmployee1 = { “Nguyen Van A”,
                   {15, 05, 1980}, // Khởi tạo cấu trúc con
                   “Nhan vien”,  300  };
```

5.2 Truy nhập các thành phần

- Đối với biến thường:

  Tên biến.Tên thành phần

- Đối với biến con trỏ :

  Tên biến -> Tên thành phần

- Đối với biến mảng: Truy nhập thành phần mảng rồi đến thành phần cấu trúc.

  Tên mảng[chỉ số].Tên thành phần

- Đối với cấu trúc lồng nhau. Truy nhập thành phần ngoài rồi đến thành phần của cấu trúc bên trong, sử dụng các phép toán . hoặc -> (các phép toán lấy thành phần) một cách thích hợp.

Ví dụ :

❖ Chương trình minh họa việc tạo lập và sử dụng biến cấu trúc

```
#include …
struct Employee {
        char name[20]; // Tên nhân viên
        int age; // Tuổi nhân viên
        char role[20]; // Chức vụcủa nhân viên
        float salary; // Lương của nhân viên
};

// Khai báo khuôn mẫu hàm
void Display(Employee myEmployee);
```

Ví dụ :

```
// định nghĩa hàm
void Display(Employee myEmployee) {
    cout << “Name: ” << myEmployee.name << endl;
    cout << “Age: ” << myEmployee.age << endl;
    cout << “Role: ” << myEmployee.role << endl;
    cout << “Salary: ” << myEmployee.salary << endl;
    return; }
int main(){
// Hiển thị giá trị mặc định
    Employee myEmployee = {“Nguyen Van A”, 27, “Nhan vien”, 300};
    cout << “Thông tin mặc định:” << endl;
    Display(myEmployee);
```

Ví dụ :

```
// Thay đổi giá trị cho các thuộc tính
    cout << “Name: ”;
    cin >> myEmployee.name;
    cout << “Age: ”;
    cin >> myEmployee.age;
    cout << “Role: ”;
    cin >> myEmployee.role;
    cout << “Salary: ”;
    cin >> myEmployee.salary;
    cout << “Thông tin sau khi thay đổi:” << endl;
    Display(myEmployee);
    return 0; }
```

Ví dụ :

❖ Ví dụ minh họa việc tạo lập và sử dụng biến cấu trúc lồng nhau

```
#include …
struct Date {
    int day;
    int month;
    int year;
};
struct Employee {
    char name[20]; // Tên nhân viên
    Date birthDay; // Ngày sinh của nhân viên
    char role[20]; // Chức vụcủa nhân viên
    float salary; // Lương của nhân viên
};
```

Ví dụ :

```
// Khai báo khuôn mẫu hàm
void Display(Employee myEmployee);
// Định nghĩa hàm
void Display(Employee myEmployee) {
    cout << “Name: ” << myEmployee.name << endl;
    cout << “Birth day: ” << myEmployee.birthDay.day << “/” <<
                             myEmployee.birthDay.month << “/”
                             << myEmployee.birthDay.year << endl;
    cout << “Role: ” << myEmployee.role << endl;
    cout << “Salary: ” << myEmployee.salary << endl;
    return;
}
```

Ví dụ :

```
int main() {
  // Hiển thị giá trị mặc định
   Employee myEmployee =
            {“Nguyen Van A”, {15, 5, 1980}, “Nhan vien”, 300};
   cout << “Thông tin mặc định:” << endl;
   Display(myEmployee);
   // Thay đổi giá trịcho các thuộc tính
   cout << “Name: ”;
   cin >> myEmployee.name;
   cout << “Day of birth: ”;
   cin >> myEmployee.birthDay.day;
   cout << “Month of birth: ”;
```

Ví dụ :

```
    cin  >> myEmployee.birthDay.month;
    cout << “Year of birth: ”;
    cin >> myEmployee.birthDay.year;
    cout << “Role: ”
    cin >> myEmployee.role;
    cout << “Salary: ”;
    cin >> myEmployee.salary;
    cout << “Thông tin sau khi thay đổi:” << endl;
    Display(myEmployee);
    return 0;
}
```

5.3 Con trỏ với cấu trúc

- **Khai báo con trỏ cấu trúc**

  Tên kiểu cấu trúc *Tên biến;

Ví dụ:

```
struct Employee {
    char name[20]; // Tên nhân viên
    Date birthDay; // Ngày sinh (kiểu Date đã khai báo ở vd trước)
    char role[20]; // Chức vụ của nhân viên
    float salary; // Lương của nhân viên
};
Employee *ptrEmployee; // con trỏ trỏ đến kiểu cấu trúc Employee
```

Con trỏ với cấu trúc

- **Gán địa chỉ cho con trỏ cấu trúc**

Tên biến con trỏ = &Tên biến cấu trúc;

Ví dụ: Employee *ptrEmployee, myEmployee;

ptrEmployee = &myEmployee;

/* con trỏ ptrEmployee trỏ đến địa chỉ của biến cấu trúc myEmployee.*/

- **Cấp phát bộ nhớ động cho con trỏ cấu trúc**

Tên biến con trỏ = new Kiểu cấu trúc;

Ví dụ

```
//Khai báo và cấp phát bộ nhớ
Employee *ptrEmployee;
ptrEmployee = new Employee;
/* Cấp phát ngay khi khai báo:
Employee *ptrEmployee = new Employee; */
```

Con trỏ với cấu trúc

- **Giải phóng vùng nhớ đã cấp phát cho con trỏ:**
  delete Tên biến con trỏ;

Ví dụ: Employee *ptrEmployee = new Employee;
…
delete ptrEmployee;

- **Chú ý:**

Thao tác delete chỉ được thực hiện đối với con trỏ đã được cấp phát bộ nhớ động thông qua thao tác new:

Employee *ptrEmployee = new Employee;
delete ptrEmployee; //đúng

Không thể delete với con trỏ chỉ trỏ đến địa chỉ của một biến cấu trúc khác: Employee *ptrEmployee, myEmployee;
ptrEmployee = &myEmployee;
delete ptrEmployee; //lỗi

Con trỏ với cấu trúc

- **Truy nhập thuộc tính của con trỏ cấu trúc**
  - Cách 1:

    Tên biến con trỏ -> Tên thuộc tính;
  - Cách 2:

    (*Tên biến con trỏ). Tên thuộc tính;

Ví dụ:

```
Employee *ptrEmployee = new Employee;
cin >> ptrEmployee -> name;
hoặc:
cin >> (*ptrEmployee).name;
```

Ví dụ

❖ Ví dụ khởi tạo và hiển thị nội dung của một con trỏ cấu trúc.

```
#include<iostream>
#include<string.h>
using namespace std;

struct Date {
        int day;
        int month;
        int year; };
struct  Employee {
   char name[20]; // Tên nhân viên
   Date birthDay; // Ngày sinh của nhân viên
   char role[20]; // Chức vụcủa nhân viên
   float salary; // Lương của nhân viên };
```

Ví dụ

```
// Khai báo khuôn mẫu hàm
void InitStruct(Employee *myEmployee);
void Display(Employee *myEmployee);
//Định nghĩa hàm
void InitStruct(Employee *myEmployee){
    myEmployee = new Employee;
    cout << “Name: ”;
    cin >> myEmployee->name;
    cout << “Day of birth: ”;
    cin >> myEmployee->birthDay.day;
    cout << “Month of birth: ”;
    cin >> myEmployee->birthDay.month;
    cout << “Year of birth: ”;
    cin >> myEmployee->birthDay.year;
    cout << “Role: ”;
```

Ví dụ

```
cin >> myEmployee->role;
    cout << “Salary: ”;
    cin >> myEmployee->salary;
}
void Display(Employee myEmployee){
    cout << “Name: ” << myEmployee->name << endl;
    cout << “Birth day: ” << myEmployee->birthDay.day << “/”
                          << myEmployee->birthDay.month << “/”
                          << myEmployee->birthDay.year << endl;
    cout << “Role: ” << myEmployee->role << endl;
    cout << “Salary: ” << myEmployee->salary << endl;
    return;
}
```

Ví dụ

```
int main(){
    Employee *myEmployee;
    InitStruct(myEmployee);
    Display(myEmployee);
    return 0;
}
```

Mảng cấu trúc

- **Khai báo mảng tĩnh các cấu trúc**

  Tên kiểu cấu trúc Tên biến mảng[Số phần tử];

  Ví dụ: Employee employees[10];
- **Khai báo mảng động các cấu trúc**

  Tên kiểu cấu trúc *Tên biến;
- **Cấp phát bộ nhớ**

  Tên biến mảng = new Kiểu cấu trúc[Số lượng phần tử];

  Ví dụ: Employee *employees = new Employee[10]
- **Truy nhập đến phần tử của mảng cấu trúc**

  Việc truy nhập đến các phần tử của mảng cấu trúc được thực hiện như truy cập đến phần tử của mảng thông thường.

  Ví dụ: Employee *employees = new Employee[10];

  …

  cin >> employees[i].name;

Ví dụ

- Ví dụ: Cài đặt việc khởi tạo một mảng các nhân viên của một phòng trong một công ty. Tìm và in ra thông tin về nhân viên có lương cao nhất và nhân viên có lương thấp nhất trong phòng.

```
#include …
struct Date {
    int day;  int month;  int year; };
struct Employee {
    char name[20]; // Tên nhân viên
    Date birthDay; // Ngày sinh của nhân viên
    char role[20]; // Chức vụcủa nhân viên
    float salary; // Lương của nhân viên
};
```

Ví dụ

```
/* Khai báo khuôn mẫu hàm */
void InitArray(Employee *myEmployee, int length);
Employee searchSalaryMax(Employee *myEmployee, int length);
Employee searchSalaryMin(Employee *myEmployee, int length);
void Display(Employee myEmployee);
// Định nghĩa hàm
void InitArray(Employee *myEmployee, int length){
    myEmployee = new Employee[length];
    for (int i=0; i<length; i++) {
        cout << “Nhan vien thu ” << i << endl;
        cout << “Name: ”;
        cin >> myEmployee[i].name;
        cout << “Day of birth: ”;
        cin >> myEmployee[i].birthDay.day;
        cout << “Month of birth: ”;
```

Ví dụ

```
        cout << “Month of birth: ”;
        cin >> myEmployee[i].birthDay.month;
        cout << “Year of birth: ”;
        cin >> myEmployee[i].birthDay.year;
        cout << “Role: ”;
        cin >> myEmployee[i].role;
        cout << “Salary: ”;
        cin >> myEmployee[i].salary;
    }
    return;
}
```

Ví dụ

```
Employee searchSalaryMax(Employee *myEmployee, int length) {
    int index = 0;
    int maxSalary = myEmployee[0].salary;
    for (int i=1; i<length; i++)
        if(myEmployee[i].salary > maxSalary) {
            maxSalary = myEmployee[i].salary;
            index = i;
        }
    return myEmployee[index];
}
```

Ví dụ

```
Employee searchSalaryMin(Employee *myEmployee, int length) {
    int index = 0;
    int minSalary = myEmployee[0].salary;
    for (int i=1; i<length; i++)
        if(myEmployee[i].salary < minSalary) {
            minSalary = myEmployee[i].salary;
            index = i;
        }
    return myEmployee[index];
}
```

Ví dụ

```
void Display(Employee myEmployee) {
    cout << “Name: ” << myEmployee.name << endl;
    cout << “Birth day: ” << myEmployee.birthDay.day << “/”
                          << myEmployee.birthDay.month << “/”
                          << myEmployee.birthDay.year << endl;
    cout << “Role: ” << myEmployee.role << endl;
    cout << “Salary: ” << myEmployee.salary << endl;
    return;
}
```

Ví dụ

```
int main() {
    Employee *myEmployee, tmpEmployee;
    int length = 0;
    cout << “So luong nhan vien: ”;
    cin >> length;
    InitArray(myEmployee); // Khởi tạo danh sách nhân viên
    tmpEmployee = searchSalaryMax(myEmployee, length);
                                                    // NV có lương cao nhất
    Display(tmpEmployee);
    tmpEmployee = searchSalaryMin(myEmployee, length);
                                                     // NV có lương thấp nhất
    Display(tmpEmployee);
    // Giải phóng vùng nhớ
    delete [] myEmployee;
    return; }
```

C. 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.*

Ví dụ: Sử dụng cấu trúc làm đối số của hàm

```
#include<iostream>
using namespace std;
struct so_thuc{ //Khai báo cấu trúc so_thuc
        int phan_nguyen;
        unsigned thap_phan;};
void hien(so_thuc a);
int main() {
        so_thuc x={-222,178}; // x là cấu trúc so_thuc
        hien(x); // Đối số x là một cấu trúc
}
void hien(so_thuc a) // Hàm hiện số thực
{
        cout<<a.phan_nguyen<<"."<<a.thap_phan;
}
```

Hàm với cấu trúc

D. Giá trị hàm là cấu trúc

Cũng tương tự như các kiểu dữ liệu cơ bản, giá trị trả lại của một hàm cũng có thể là các cấu trúc dưới các dạng sau:

- *Là một biến cấu trúc.*
- *Là một con trỏ cấu trúc.*
- *Là một tham chiếu cấu trúc.*

Ví dụ

```
#include<iostream>
using namespace std;
struct so_phuc{ float thuc;  float ao;};
so_phuc tong(so_phuc a, so_phuc b) {
        so_phuc kq;
        kq.thuc=a.thuc+b.thuc;
        kq.ao=a.ao+b.ao;
        return kq;
}
int main() {
        so_phuc x={1,23}, y={2,22}, z;
        z = tong(x, y);
        cout<<z.thuc<<"+"<<z.ao<<"*i";
}
```

5.5 Câu lệnh typedef

Để thuận tiện trong sử dụng, thông thường các kiểu được người sử dụng tạo mới sẽ được gán cho một tên kiểu bằng câu lệnh typedef như sau:

typedef Kiểu Tên_kiểu ;

Ví dụ: Có thể đặt tên cho kiểu ngày tháng là Date với khai báo sau:

```
typedef struct Date {
        int ng;
        int th;
        int nam;
        };
```

5.5 Hàm sizeof()

Hàm sizeof() trả lại kích thước của một biến hoặc kiểu.

Ví dụ:

```
#include<iostream>
using namespace std;
int main(){
    double x;
    struct date{
        int ngay; int thang; int nam;
    }y;
cout<<sizeof(x)<<endl; // Xuất ra 8
cout<<sizeof(y); //Xuất ra 12 ( kiểu int kích thước là 4)
}
```


Thank You !