

Nhập Môn Lập Trình

Cấu Trúc **struct**

TS. Lê Nguyên Khôi
Trường Đại học Công nghệ, ĐHQGHN

Nội Dung

- ▶ Cấu trúc **struct**
 - ▶ Kiểu dữ liệu nhóm
 - ▶ Định nghĩa
 - ▶ Khai báo / Khởi tạo
 - ▶ Sử dụng
 - ▶ Truyền biến cho hàm

Kiểu Dữ Liệu Mảng (Nhắc Lại)

- ▶ Quản lý điểm số môn học Toán, Lý, và điểm trung bình của một lớp học

- ▶ 3 mảng một chiều

```
double toan[SO_SINH_VIEN] ;
```

```
double ly[SO_SINH_VIEN] ;
```

```
double diemTB[SO_SINH_VIEN] ;
```

- ▶ 1 mảng hai chiều

```
double sinhVien_diemSo[SSV][3] ;
```

- Sử dụng chỉ số thứ hai để xác định môn học
 - Lập trình viên tự quản lý, khó nhớ, dễ nhầm

Quản Lý Điểm Số – Cấu Trúc Mảng

- ▶ 1 mảng hai chiều quản lý điểm số môn học

```
double sinhVien_diemSo[SSV][3];
```

- ▶ Sử dụng hằng số để quy ước môn học

```
const int TOAN = 0;
```

```
const int LY = 1;
```

```
const int DTB = 2;
```

- ▶ Ví dụ

```
sinhVien_diemSo[i][LY]
```

- ▶ Điểm môn Lý của sinh viên thứ **i** trong danh sách

Quản Lý Điểm Số – Cấu Trúc Mảng

- ▶ Nhập điểm môn học cho 1 sinh viên
- ▶ Tính điểm trung bình cho 1 sinh viên

```
const int TOAN = 0;
const int LY = 1;
const int DTB = 2;

void nhapDiemSo(double diemSo[])
{
    cin >> diemSo[TOAN] >> diemSo[LY];
}

void tinhDiemTB(double diemSo[])
{
    diemSo[DTB] = (diemSo[TOAN]+diemSo[LY]) / 2;
}
```

Quản Lý Điểm Số – Cấu Trúc Mảng

```
const int TOAN = 0;
const int LY = 1;
const int DTB = 2;

void nhapDiemSo(double diemSo[]);
void tinhDiemTB(double diemSo[]);

int main()
{
    double sinhVien_diemSo[SSV][3];
    for (int i = 0; i < SSV; i++) {
        nhapDiemSo(sinhVien_diemSo[i]);
        tinhDiemTB(sinhVien_diemSo[i]);
    }
    return 0;
}
```

Quản Lý Điểm Số – Cấu Trúc **struct**

```
struct DiemSo {  
    double toan, ly, diemTB;  
};  
int main() {  
    const int SSV = 10;  
    DiemSo sv[SSV];  
    for (int i = 0; i < SSV; i++) {  
        cin >> sv[i].toan >> sv[i].ly;  
        sv[i].diemTB = (sv[i].toan+sv[i].ly)/2;  
    }  
    for (int i = 0; i < SSV; i++) {  
        cout << sv[i].toan << " " << sv[i].ly;  
        cout << " " << sv[i].diemTB << endl;  
    }  
    return 0;  
}
```

Kiểu Dữ Liệu Nhóm

- ▶ Kiểu Mảng:
 - ▶ Tập hợp dữ liệu cùng kiểu
 - ▶ Truyền cho hàm:
 - truyền tham chiếu
- ▶ Kiểu cấu trúc **struct**:
 - ▶ Tập hợp dữ liệu có thể khác kiểu
 - ▶ Phải định nghĩa trước khi sử dụng
 - ▶ Truyền cho hàm:
 - truyền tham chiếu
 - truyền giá trị

Quản Lý Vector

- ▶ Vector trong hệ tọa độ 2 chiều:
 - ▶ Cặp tọa độ x-, y-
 - ▶ Phép toán cộng vector
- ▶ Viết chương trình nhập & in vector, và tính tổng hai vector
 - ▶ **nhap**: nhập dữ liệu cho một vector
 - ▶ **in**: in dữ liệu của một vector
 - ▶ **cong**: tính vector tổng của 2 vector

Quản Lý Vector 2D

```
void cong(int _1X, int _1Y,  
          int _2X, int _2Y,  
          int & _sX, int & _sY)  
{  
    _sX = _1X + _2X;  
    _sY = _1Y + _2Y;  
  
}  
  
int main()  
{  
    int aX = 1, aY = 0;  
    int bX = 2, bY = 1;  
    int sX, sY;  
    cong(aX, aY, bX, bY, sX, sY);  
    return 0;  
}
```

Quản Lý Vector 3D

```
void cong(int _1X, int _1Y, int _1Z,  
          int _2X, int _2Y, int _2Z,  
          int & _sX, int & _sY, int & _sZ)  
{  
    _sX = _1X + _2X;  
    _sY = _1Y + _2Y;  
    _sZ = _1Z + _2Z;  
}  
  
int main()  
{  
    int aX = 1, aY = 0, aZ = 2;  
    int bX = 2, bY = 1, bZ = 0;  
    int sX, sY, sZ;  
    cong(aX, aY, aZ, bX, bY, bZ, sX, sY, sZ);  
    return 0;  
}
```

Kiểu Dữ Liệu Cấu Trúc **struct**

- ▶ Định nghĩa sử dụng từ khóa **struct**
 - ▶ Tạo kiểu dữ liệu mới
- ▶ Định nghĩa toàn cục
 - ▶ Ngoài và trước tất cả các hàm
 - ▶ Tất cả các hàm đều hiểu
- ▶ Miêu tả:
 - ▶ Các thành viên và kiểu dữ liệu của chúng

Quản Lý Vector 2D – Sử Dụng **struct**

```
struct Vector {  
    int X, Y;  
  
};  
void cong(Vector v1, Vector v2, Vector & sum) {  
    sum.X = v1.X + v2.X;  
    sum.Y = v1.Y + v2.Y;  
  
}  
int main() {  
    Vector a, b, sum;  
    a.X = 1; a.Y = 0;  
    b.X = 2; b.Y = 1;  
    cong(a, b, sum);  
    return 0;  
}
```

Quản Lý Vector 3D – Sử Dụng **struct**

```
struct Vector {  
    int X, Y;  
    int Z;  
};  
void cong(Vector v1, Vector v2, Vector & sum) {  
    sum.X = v1.X + v2.X;  
    sum.Y = v1.Y + v2.Y;  
    sum.Z = v1.Z + v2.Z;  
}  
int main() {  
    Vector a, b, sum;  
    a.X = 1; a.Y = 0; a.Z = 2;  
    b.X = 2; b.Y = 1; b.Z = 0;  
    cong(a, b, sum);  
    return 0;  
}
```

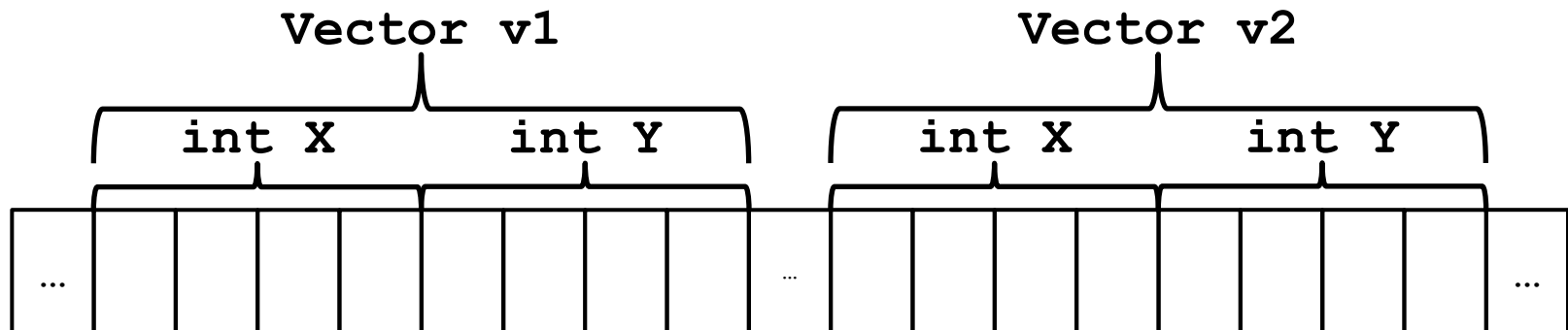
Kiểu Dữ Liệu Cấu Trúc **struct**

- ▶ Tập hợp các kiểu dữ liệu
- ▶ Mỗi dữ liệu được lưu trong một biến
- ▶ Mỗi dữ liệu có kiểu cụ thể
- ▶ Tổ chức dữ liệu phức tạp
- ▶ Thao tác giữa các dữ liệu này

Định Nghĩa Kiểu **Vector**

- ▶ Định nghĩa sử dụng từ khóa **struct**
 - ▶ Tên kiểu dữ liệu mới **Vector**
 - ▶ **Vector** có 02 dữ liệu thành viên kiểu **int**

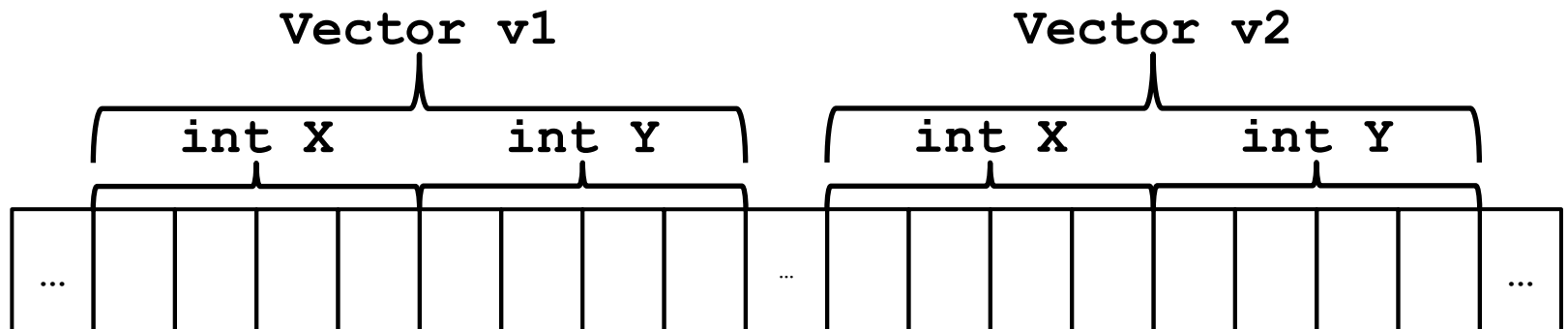
```
struct Vector {  
    int X;  
    int Y;  
};  
Vector v1, v2;
```



Truy Cập Thành Viên Kiểu **Vector**

- ▶ Truy cập **X**, **Y** của kiểu **Vector**

```
struct Vector {  
    int X;  
    int Y;  
};  
Vector v1, v2;  
v1.X = ... ..; v1.Y = ... ..;  
v2.X = ... ..; v2.Y = ... ..;
```



Truy Cập Thành Viên Kiểu **Vector**

- ▶ Sử dụng toán tử chấm để truy cập biến thành viên (**.x**, **.y**) của kiểu **Vector**
- ▶ **x**, **y** là biến thành viên của kiểu **Vector**
 - ▶ Kiểu cấu trúc **struct** khác nhau có thể có cùng tên biến thành viên
 - ▶ Biến thành viên là biến cục bộ

Kiểu **Vector** – Phép Gán

- ▶ Thực hiện phép gán đơn giản hợp lệ
 - ▶ Sao chép giá trị các biến thành viên của vector **v2** cho các biến thành viên của vector **v1**

```
struct Vector
{
    int X;
    int Y;
};
```

```
Vector v1, v2;
v1 = v2 ;
// tương đương 2 phép gán sau
v1.X = v2.X;
v1.Y = v2.Y;
```

Kiểu **Vector** – Phép Toán Khác

- ▶ Các phép toán khác đều chưa được định nghĩa
- ▶ Phải tự định nghĩa các phép toán này

```
struct Vector
{
    int X;
    int Y;
};
Vector v1, v2;
// các phép toán sau đều lỗi dịch
cin >> v1;    cout << v2;
v1 + v2;       v1 == v2;
v1 - v2;       v1 <= v2;
v1 * v2;       v1 >= v2;
v1 / v2;       v1 != v2;
```

Kiểu **Vector** – Truyền Biến Cho Hàm

▶ Truyền tham chiếu:

```
void nhap(Vector & v)
{
    cin >> v.X;
    cin >> v.Y;
}
```

▶ Truyền giá trị:

```
void in(Vector v)
{
    cout << v.X;
    cout << v.Y;
}
```

Kiểu **Vector** – Khai Báo/Khởi Tạo

```
struct Vector {  
    int X, Y;  
};  
  
void nhap(Vector & v);  
void in(Vector v);  
void cong(Vector v1, Vector v2, Vector & sum);  
  
int main() {  
    Vector v1 = { 1 , 2 };  
    Vector v2, sumV;  
    nhap(v2);  
    cong(v1, v2, sumV);  
    in(sumV);  
    return 0;  
}
```

Kiểu ToaDo

```
struct ToaDo {  
    double X, Y;  
};  
  
void nhapToaDo(ToaDo & td) ;  
void inToaDo(ToaDo td) ;  
ToaDo congToaDo(... ..) ;  
  
int main() {  
    ToaDo td1, td2;  
    nhapToaDo(td1) ; inToaDo(td1) ;  
    nhapToaDo(td2) ; inToaDo(td2) ;  
    ToaDo tdTong = congToaDo(... ..) ;  
    inToaDo(tdTong) ;  
    return 0;  
}
```

Kiểu ToaDo

```
struct ToaDo {  
    double X, Y;  
};  
  
ToaDo congToaDo(ToaDo td1, ToaDo td2) {  
    ToaDo tdTong;  
    tdTong.X = td1.X + td2.X;  
    tdTong.Y = td1.Y + td2.Y;  
    return tdTong;  
}  
  
int main() {  
    ToaDo td1, td2;  
    ToaDo tdTong = congToaDo(td1, td2);  
    return 0;  
}
```


Kiểu ToaDo

```
struct ToaDo {  
    double X, Y;  
};  
  
ToaDo congToaDo(ToaDo td1, ToaDo td2);  
ToaDo tinhTrungDiem(ToaDo td1, ToaDo td2);  
double tinhKhoangCach(ToaDo td1, ToaDo td2) {  
    return sqrt((td1.X-td2.X)*(td1.X-td2.X)  
                + (td1.Y-td2.Y)*(td1.Y-td2.Y));  
}  
  
int main() {  
    ToaDo td1, td2;  
    ... ..  
    return 0;  
}
```

Kiểu ²TamGiac

```
struct TamGiac {
    ToaDo dinhA, dinhB, dinhC;
};

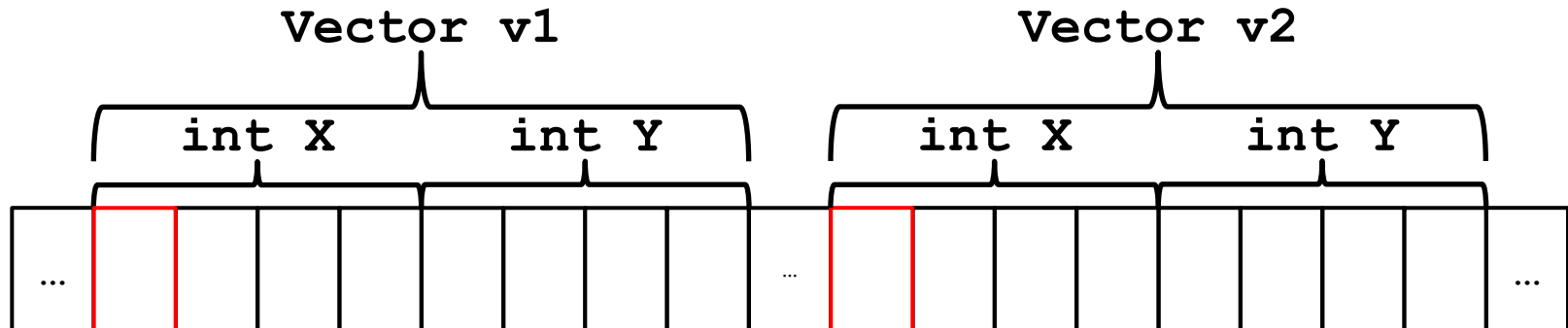
double tinhChuVi(TamGiac tg) {
    return tinhKhoangCach(tg.dinhA, tg.dinhB)
        + tinhKhoangCach(tg.dinhB, tg.dinhC)
        + tinhKhoangCach(tg.dinhC, tg.dinhA);
}

int main() {
    TamGiac tam_giac;
    ... ..
    cout << tinhChuVi(tam_giac);
    return 0;
}
```

Kiểu **Vector** – Con Trỏ

- ▶ Giống các kiểu dữ liệu khác:
 - ▶ **Vector *** là kiểu con trỏ
 - ▶ Toán tử **&** trả về địa chỉ của biến kiểu **Vector**

```
struct Vector {  
    int X;  
    int Y;  
};  
Vector v1, v2;
```



Kiểu **Vector** – Con Trỏ

```
struct Vector
{
    int X, Y;
};

int main()
{
    Vector v1, v2;
    Vector *p1 = &v1;
    Vector *p2 = &v2;
    Vector *pSum = new Vector;
    (*pSum).X = (*p1).X + (*p2).X;
    pSum->Y = p1->Y + p2->Y;
    *pSum.Y = *p1.Y + *p2.Y; // lỗi dịch
    return 0;
}
```

Kiểu **Vector** – Con Trỏ

- ▶ Theo thứ tự ưu tiên: “.” được ưu tiên trước “*”
- ▶ Nếu **p** là con trỏ kiểu **Vector ***:
 - ▶ ***p.X** tương đương ***(p.X)** không hợp lệ
 - Toán tử . áp dụng cho **p**
 - Toán tử . chỉ áp dụng cho kiểu cấu trúc **struct**
 - Kiểu dữ liệu **p** là kiểu con trỏ **Vector ***
 - ▶ Phải sử dụng **(*p).X**
 - ▶ Để thuận tiện, có thể dùng toán tử **->**:
 - kết hợp con trỏ (*****) với truy cập trường (**.**)
 - **p->X** tương đương **(*p).X**

Kiểu **Vector** – Mảng

```
struct Vector {  
    int X, Y;  
};  
  
void nhap(Vector & v);  
  
int main() {  
    Vector mangV1[10];  
    Vector * mangV2 = new Vector[10];  
    for (int i = 0; i < 10; i++)  
    {  
        nhap(mangV1[i]);  
        nhap(mangV2[i]);  
    }  
    return 0;  
}
```

Quản Lý Thông Tin Sinh Viên

- ▶ Thông tin sinh viên:
 - ▶ Mã số sinh viên
 - ▶ Họ và tên
 - ▶ Ngày sinh
- ▶ Quản lý thông tin:
 - ▶ Không dùng được 1 mảng hai chiều
 - ▶ Dùng nhiều mảng một chiều
 - ▶ Định nghĩa kiểu dữ liệu mới

Kiểu ²NgàyThangNam

```
struct NgàyThangNam {  
    int ngay;  
    int thang;  
    int nam;  
};  
  
bool isNgàyHopLe(NgàyThangNam ntn);  
NgàyThangNam tinhNgàyTruoc(NgàyThangNam ntn);  
NgàyThangNam tinhNgàySau(NgàyThangNam ntn);  
  
int main() {  
    NgàyThangNam ntn = {01, 04, 2000};  
    NgàyThangNam ngay_mai = tinhNgàySau(ntn);  
    NgàyThangNam hom_qua = tinhNgàyTruoc(ntn);  
    return 0;  
}
```


Kiểu SinhVien

```
struct SinhVien {
    string      mssv;
    string      ho_ten;
    NgayThangNam ngay_sinh;
};

int main() {
    NgayThangNam ntn = {01, 04, 2000};
    SinhVien sv1 = { "12345678",
                    "Trach Van Doanh",
                    ntn };
    SinhVien sv2 = { "12345678",
                    "Trach Van Doanh",
                    {01, 04, 2000} };

    return 0;
}
```

Kiểu **DaThuc**

- ▶ Định nghĩa kiểu dữ liệu mới **DaThuc** để biểu diễn một đa thức dạng:

$$a_0 + a_1x^1 + \dots + a_nx^n$$

- ▶ n là bậc của đa thức
- ▶ a_0 đến a_n là các hệ số của đa thức

```
struct DaThuc {  
    ... ..  
};  
int main() {  
    DaThuc dt1, dt2;  
    ... ..  
    return 0;  
}
```

Kiểu DaThuc²

```
struct DaThuc {
    int bac;
    double mang_he_so[100];
};

void nhap( DaThuc & )
void quyChuan( DaThuc & );
DaThuc tong( DaThuc , DaThuc );
DaThuc hieu( DaThuc , DaThuc );
double tinhGiaTri( DaThuc , double );

int main() {
    DaThuc dt1, dt2;
    ... ..
    return 0;
}
```

Kiểu DaThuc²

```
struct DaThuc {  
    ... ..  
};  
  
void nhap( DaThuc & )  
void quyChuan( DaThuc & );  
  
int main() {  
    DaThuc dt1, dt2;  
    return 0;  
}  
  
void nhap(DaThuc & dt) {  
    ... ..  
    quyChuan(dt) ;  
}
```