

MỤC TIÊU

1

**Hiểu và vận dụng struct, class
vào bài toán thực tế**

2

**Phân biệt được các thành phần,
phép toán cụ thể với tùy kiểu dữ
liệu định nghĩa**



Giới thiệu
Struct
Class
Enum

CHƯƠNG 5: **STRUCT VÀ CLASS**



GIỚI THIỆU

Kiểu dữ liệu

Kiểu dữ liệu có sẵn:

- Int
- Char
- Float
- ...

• Kiểu dữ liệu tự định nghĩa:

- Struct
- Class
- enum
- ...

GIỚI THIỆU

Khi nào cần kiểu dữ liệu tự định nghĩa? Kiểu dữ liệu tự định nghĩa giải quyết những vấn đề gì?

- **Tạo cấu trúc dữ liệu phức tạp:** Khi cần biểu diễn một đối tượng hoặc cấu trúc dữ liệu phức tạp.
- **Đóng gói dữ liệu liên quan lại với nhau:** để đóng gói các biến và hàm liên quan lại với nhau trong một đối tượng hoặc lớp, giúp quản lý dữ liệu và logic của đối tượng dễ dàng hơn.
- **Tạo ra kiểu liên kết mới:** có thể tạo ra một kiểu dữ liệu mới bằng cách kết hợp các kiểu dữ liệu có sẵn.



Giới thiệu
Struct
Class
Enum

CHƯƠNG 5: **STRUCT VÀ CLASS**



STRUCT

- Định nghĩa struct
- Truy xuất các thành phần
- Các phép toán cơ bản
- Nhập/xuất
- struct và function
- struct và array
- struct trong struct
- struct và pointer

STRUCT

❓ Định nghĩa struct

Struct là kiểu dữ liệu do người lập trình **tự định nghĩa**. Struct có thể chứa nhiều thành phần dữ liệu có kiểu khác nhau. Mỗi thành phần của struct gọi là thành viên (member) và có tên (memberName).

Ví dụ:

Struct lưu trữ thông tin của sinh viên.

Các thành viên của struct: họ tên, địa chỉ, tuổi, điểm,

STRUCT

?] Cú pháp định nghĩa struct:

```
struct structName {  
    dataType memberName;  
    dataType memberName;  
    ...  
    dataType memberName;  
};
```

Trong đó:

- structName: tên của kiểu dữ liệu được định nghĩa
- dataType: kiểu dữ liệu của từng thành viên của struct
- membername: tên của từng thành viên của struct

- Lưu ý:

- Khi Struct được định nghĩa, không có vùng nhớ nào được cấp cho tới khi struct đó được khởi tạo

STRUCT

Ví dụ: khai báo một cú pháp struct:

```
struct SinhVien {  
    string hoten;  
    string diachi;  
    int tuoi;  
    double diem;  
};
```

STRUCT

- ?] Cú pháp khai báo biến kiểu struct:

structName variableName;

Ví dụ:

SinhVien sv;

- ?] Lưu ý: có thể vừa định nghĩa struct vừa khai báo như sau:

```
?] struct SinhVien {  
    string hoten;  
    string diachi;  
    int tuoi;  
    double diem;  
} sv;
```

Khai báo biến **sv** kiểu **SinhVien**

STRUCT

Truy xuất các thành phần

variableName.memberName;

Trong đó:

- variableName: tên biến có kiểu struct.
- memberName: tên thành phần trong struct.

Ví dụ:

```
Sinhvien sv1;  
sv1.hoten = "Tran Van Minh";  
sv1.diachi = "Thanh pho Ho Chi Minh";  
sv1.tuoi = 20;  
sv1.diem = 7.2;
```

Lưu ý:

- Các biến thành viên trong struct có kiểu truy cập mặc định là công khai (public)

STRUCT

Các phép toán cơ bản

- **Phép gán:** có thể gán 2 biến struct cho nhau nếu cùng kiểu. Tương ứng với lệnh gán từng thành phần cho nhau.
- **Phép so sánh:** không thể so sánh 2 biến struct với nhau dù cùng kiểu. Chỉ có thể so sánh từng thành phần.

STRUCT

Nhập/xuất kiểu struct

Nhập hoặc xuất từng thành phần của biến kiểu struct.

Ví dụ:

```
SinhVien sv;
cout << "Nhập họ tên sinh viên: ";
getline(cin, sv.hoten);
cout << "Nhập địa chỉ: ";
getline(cin, sv.diachi);
cout << "Nhập tuổi: ";
cin >> sv.tuoi;
cout << "Nhập điểm: ";
cin >> sv.diem;
cout << "Thông tin của sinh viên là:\nTen: " <<
sv.hoten << "\nDia chi la: " << sv.diachi << "\nTuoi la: "
<< sv.tuoi << "\nDiem la: " << sv.diem << endl;
```

STRUCT

Liên hệ giữa struct và function

- ?] Nên định nghĩa các struct trước khi định nghĩa các hàm.
- ?] Struct truyền cho hàm dưới các dạng:
 - Tham trị
 - Tham chiếu
 - Con trỏ
- ?] Hàm có thể trả về kiểu struct.

STRUCT

Liên hệ giữa struct và array

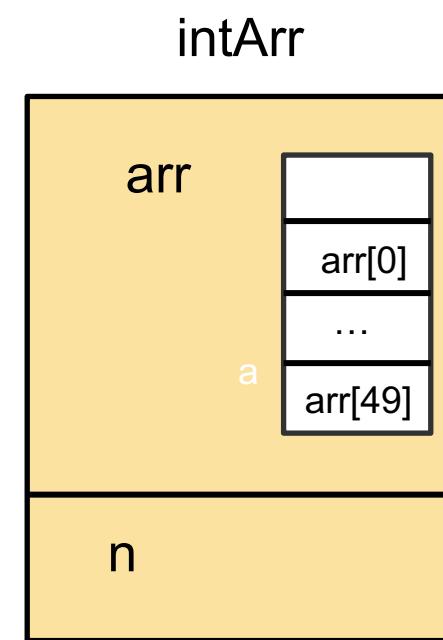
- struct có thể chứa thành phần là array.
- Có thể khai báo 1 biến array kiểu struct.

STRUCT

Liên hệ giữa struct và array

struct có thể chứa thành phần là array.

Ví dụ: struct lưu dãy số nguyên



```
const int MAXSIZE = 50;  
struct intArr  
{  
    int arr[MAXSIZE];  
    int n;  
};  
Cách truy xuất từng phần tử:  
intArr a;  
a.arr[1] = 3;  
a.n = 15;
```

STRUCT

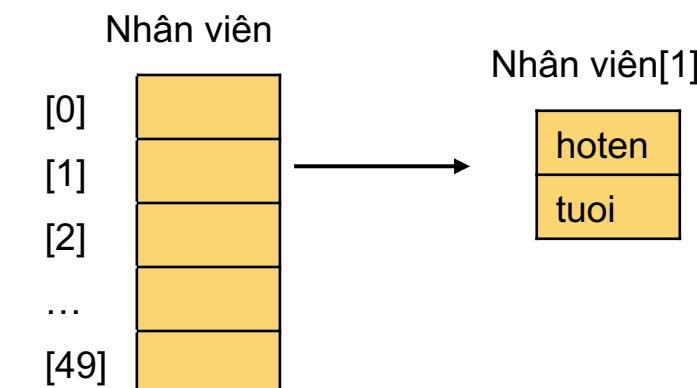
Liên hệ giữa struct và array

- ☒ Có thể khai báo 1 biến array kiểu struct.

Ví dụ:

```
struct NhanVien {  
    string hoten;  
    int tuoi;  
};
```

```
NhanVien nhanvien[50];
```



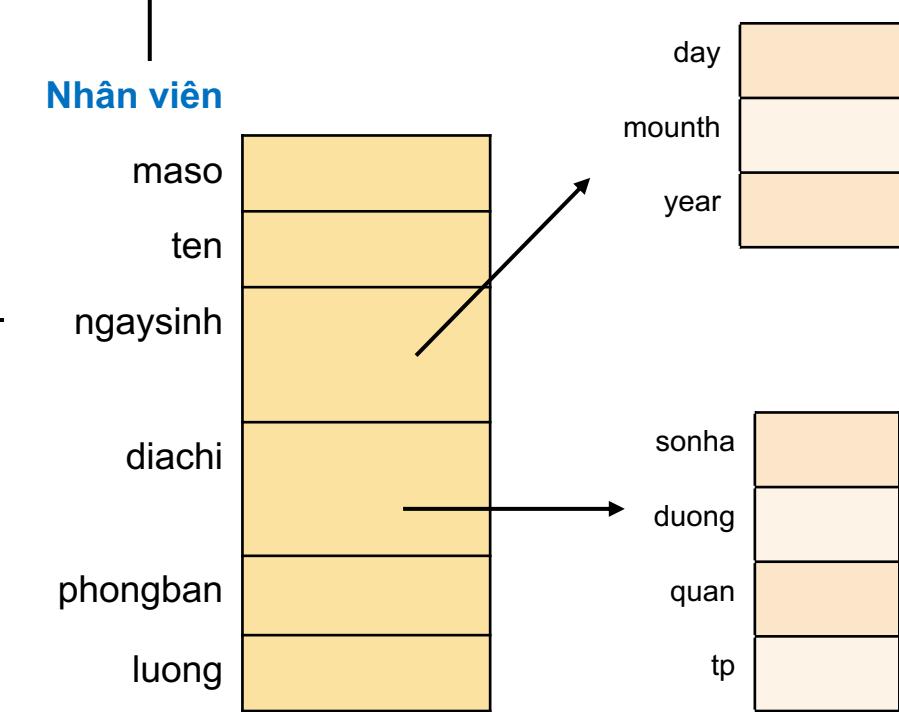
STRUCT

Liên hệ giữa struct và array

```
struct Address{  
    int sonha;  
    string duong;  
    string quan;  
    string tp;  
};
```

```
struct Date{  
    int day;  
    int mounth;  
    int year;  
};
```

```
struct Nhanvien{  
    int maso;  
    string ten;  
    Date ngaysinh;  
    Address diachi;  
    string phongban;  
    int luong;  
};
```



STRUCT

Khai báo và truy cập kiểu struct

```
Nhanvien nv;  
nv.maso=2023;  
nv.ten="Nguyen Van A";  
nv.ngaysinh.day=1;  
nv.ngaysinh.mounth=1;  
nv.ngaysinh.year=1990;  
nv.diachi.so=37;  
nv.luong=7000000;
```

STRUCT

Liên hệ giữa struct và pointer

- Có thể khai báo con trỏ kiểu struct. Sau đó cho trỏ đến địa chỉ của 1 biến kiểu struct đã khai báo.
- Có thể dùng cấp phát động.

Ví dụ: với struct NhanVien đã khai báo

NhanVien nv;

NhanVien *pnv;

pnv = &nv;

Truy xuất thành phần: (*pnv).maso = 5;

Hoặc

pnv -> maso = 5;



Giới thiệu
Struct
Class
Enum

CHƯƠNG 5: STRUCT VÀ CLASS



CLASS

Định nghĩa class

Public members

Member functions

Constructor

Header file

CLASS

?] Định nghĩa class

Tương tự struct nhưng có bao gồm nhiều thành phần trong đó có hàm.

?] Cú pháp khai báo class:

```
class className {  
    memberList;  
};
```

Trong memberList:

- Nếu là biến thì khai báo như cú pháp khai báo biến.
- Nếu là hàm thì khai báo như function prototype.

LƯU Ý: với memberlist phải xác định mức độ truy cập.

CLASS

② Định nghĩa class

- Khi định nghĩa class phải lưu ý đến mức độ truy xuất các member.
- Mức độ truy xuất các member trong class: **private, public, protected**.
- Mặc định: private

CLASS

Cú pháp class:

```
Class nameclass{  
    private:  
        // code  
    public:  
        //code  
    ...  
};
```

- **Private**: các biến liệt kê dạng private không thể liệt kê trực tiếp bên ngoài class (muốn truy xuất có thể thông qua member là function).
- **Public** các biến liệt kê dạng public có thể được truy xuất trực tiếp từ bên ngoài class.
- **Protected** các biến liệt kê dưới dạng protected có thể truy xuất từ trong class và các class kế thừa

CLASS

?) Public members

Với các member là biến đơn thì việc truy xuất tương tự như truy xuất struct_member.

?) **Hàm thành viên (Member):** Khai báo và định nghĩa bên trong class; cũng có thể dùng nguyên mẫu hàm (function prototype) để khai báo trong class. Việc thực hiện định nghĩa function_member có thể diễn ra bên ngoài class với cú pháp:

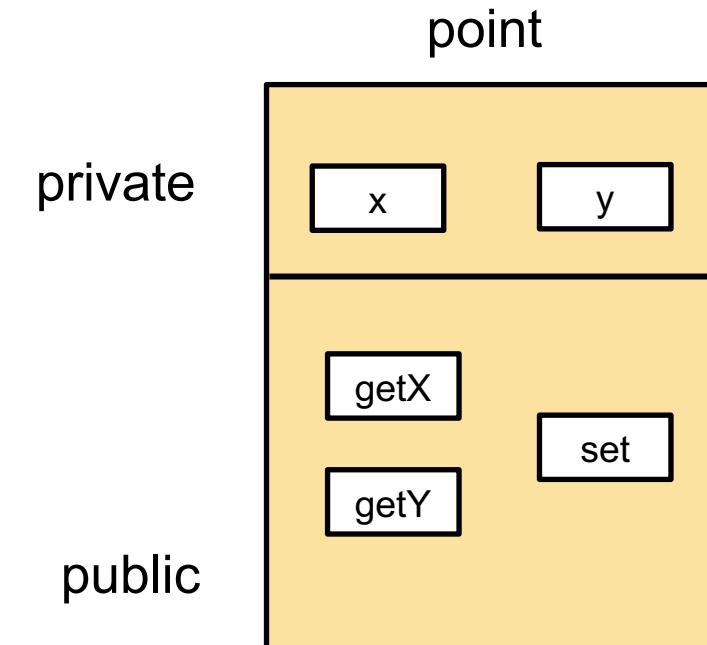
```
Datatype classname :: functionname( parameter ){  
    Statements;  
}
```

Dấu :: là toán tử phân giải phạm vi

CLASS

Ví dụ: xây dựng class point.

```
class Point
{
    private:
        int x, y;
    public:
        int getX();
        int getY();
        void set(int a, int b);
};
```

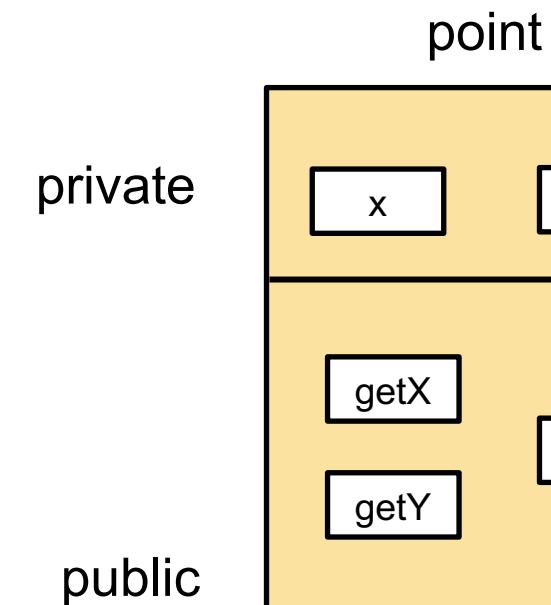


CLASS

Ví dụ: xây dựng class point.

```
int Point::getX()
{
    return x;
}
int Point::getY()
{
    return y;
}
void Point::set(int a, int b)
{
    x = a;
    y = b;
}
```

Truy cập các thuộc tính của class



CLASS

Ví dụ: Xây dựng class Point bằng cách định nghĩa hàm trực tiếp trong class.

```
class Point
{
    private:
        int x, y;
    public:
        int getX()
        {
            return x;
        }
        int getY()
        {
            return y;
        }
        void set(int a, int b)
        {
            x = a;
            y = b;
        }
};
```

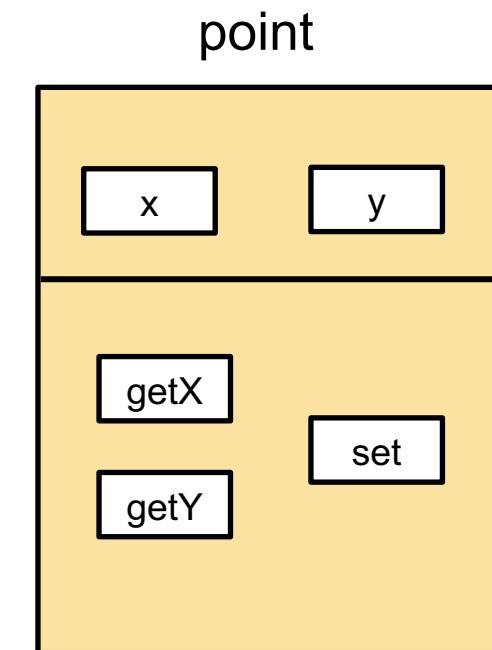
CLASS

Ví dụ: xây dựng class point.

```
int main()
{
    Point s;
    int m, n;
    cout << "Nhập hai số nguyên: ";
    cin >> m >> n;
    s.set(m, n);
    cout << s.getX() << endl;
    cout << s.getY() << endl;
}
```

private

public



CLASS

Viết chương trình xây dựng class PhanSo gồm:

Private: tử số, mẫu số

Public:

- Hàm trả về giá trị của tử số
- Hàm trả về giá trị của mẫu số
- Hàm thiết lập giá trị cho tử số
- Hàm thiết lập giá trị cho mẫu số
- Hàm tính tích hai phân số.

Sau đó tạo hàm main để kiểm chứng.

CLASS

Định nghĩa class phanso:

```
class phanso
{
    private:
        int tuso, mauso;
    public:
        int getts();
        int getms();
        void setts(int a);
        void setms(int b);
        phanso tich(phanso ps1, phanso ps2);
};
```

CLASS

```
int phanso::getts()
{
    return tuso;
}
int phanso::getms()
{
    return mauso;
}
void phanso::setts(int a)
{
    tuso = a;
}
void phanso::setms(int b)
{
    mauso = b;
}
```

```
phanso phanso::tich(phanso ps1, phanso ps2)
{
    phanso kq;
    kq.setts( ps1.tuso * ps2.tuso );
    kq.setms( ps1.mauso * ps2.mauso);
    return kq;
}

int main()
{
    phanso ps1, ps2, kq;
    int ts1, ms1, ts2, ms2;
    cout << "Nhập tử số và mẫu số phân số 1: "; cin >> ts1 >> ms1;
    cout << "Nhập tử số và mẫu số phân số 2: "; cin >> ts2 >> ms2;
    ps1.setts(ts1);
    ps1.setms(ms1);
    ps2.setts(ts2);
    ps2.setms(ms2);
    kq = kq.tich(ps1, ps2);
    cout << kq.getts() << "/" << kq.getms() << endl;
}
```

CLASS

Vấn đề tồn tại ???

//Định nghĩa class

```
int main() {  
    phanso ps1, ps2, kq;  
  
    cout << ps1.getts() << endl;  
}
```

//Sẽ xuất giá trị rác

- Vậy ta cần khởi tạo giá trị cho các biến private trong class.
- class hỗ trợ khởi tạo giá trị tự động cho các biến private, gọi là constructor

CLASS

- ② **Constructor:** là hàm tự động thực hiện khi một đối tượng thuộc class được tạo.
- ② Constructor dùng để **khởi tạo** các giá trị cho biến private trong class.
- ② **Lưu ý:**
 - Nên đặt trùng tên với class và không được định dạng kiểu.
 - Mỗi class có một constructor mặc định: constructor không có tham số.
 - Mỗi class nên có ít nhất 1 constructor có tham số (đều trùng tên với class nhưng số tham số khác nhau).

CLASS

?] Constructor:

Ví dụ:

```
class phanso
{
    private:
        int tuso, mauso;
    public:
        phanso() { tuso = 0; mauso = 0;};
        phanso (int a, int b);
        ~phanso(); //huy constructor
        int getts();
        int getms();
        void setts(int a);
        void setms(int b);
        phanso tich(phanso ps1, phanso ps2);
};
```

```
phanso::phanso(int a, int b)
{
    setts(a);
    setms(b);
}
//khi đó trong hàm main:
phanso ps1, ps2, kq;
cout << ps1.getts() << endl;
//kết quả sẽ là 0
```

CLASS

Header file

- Tạo file kiểu header file (.h) ở folder header file của project.
- Định nghĩa class ở header file.
- Ở file .cpp sử dụng class thêm chỉ thị: `#include “tên_file_header.h”`



Giới thiệu
Struct
Class
Enum

CHƯƠNG 5: STRUCT VÀ CLASS



ENUM

Định nghĩa enum

Các phép toán

Nhập/xuất

Mảng 2 chiều và kiểu liệt kê

ENUM

② Định nghĩa enum

Là kiểu dữ liệu do người dùng định nghĩa đơn giản nhất, gọi là kiểu liệt kê, dùng để lưu trữ một tập giá trị hằng được đặt tên giúp chương trình dễ đọc và tránh lỗi.

Cú pháp định nghĩa enum: `enum typename { value1, value2,...};`

Trong đó:

- `typeName`: tên kiểu dữ liệu
- `value1, value2, ...` tên của các giá trị, đặt giữa `{` và `}`
- Giá trị mặc định theo thứ tự value là `0, 1, ..., n-1`

ENUM

Ví dụ:

```
enum Thu {hai, ba, tu, nam, sau, bay, chunhat};
```

```
Thu ngay;
```

```
ngay = tu;
```

```
cout << ngay << endl;
```

```
//kết quả: 2
```

ENUM

Các phép toán với Enum:

- Phép gán: gán giữa các biến cùng kiểu enum với nhau, gán giá trị cho biến kiểu enum.
- Chuyển đổi kiểu dữ liệu khi xuất giá trị lưu trữ: câu lệnh `static_cast(variableName)`
- Không thể gán giá trị kiểu int cho biến enum.
- Chỉ có thể so sánh giữa biến kiểu enum với giá trị.

ENUM

Các phép toán với Enum

- Không sử dụng phép toán số học

```
DenGiaoThong a,y;  
  
x = xanh;  
y = x + 1; //error  
x++; // error
```

- Muốn tăng hay thay đổi phải chuyển đổi kiểu dữ liệu

```
x = static_cast<DenGiaoThong( x + 1)>;
```

ENUM

- ② Nhập/xuất: gián tiếp thông qua biến kiểu enum đã khai báo và cú pháp lựa chọn trường hợp để gán dữ liệu.
- ② Mảng 2 chiều và kiểu liệt kê: tận dụng chỉ số mảng là chỉ số giá trị kiểu liệt kê do enum trả về.



Thank you