KỸ THUẬT LẬP TRÌNH

Chương 5: struct và class

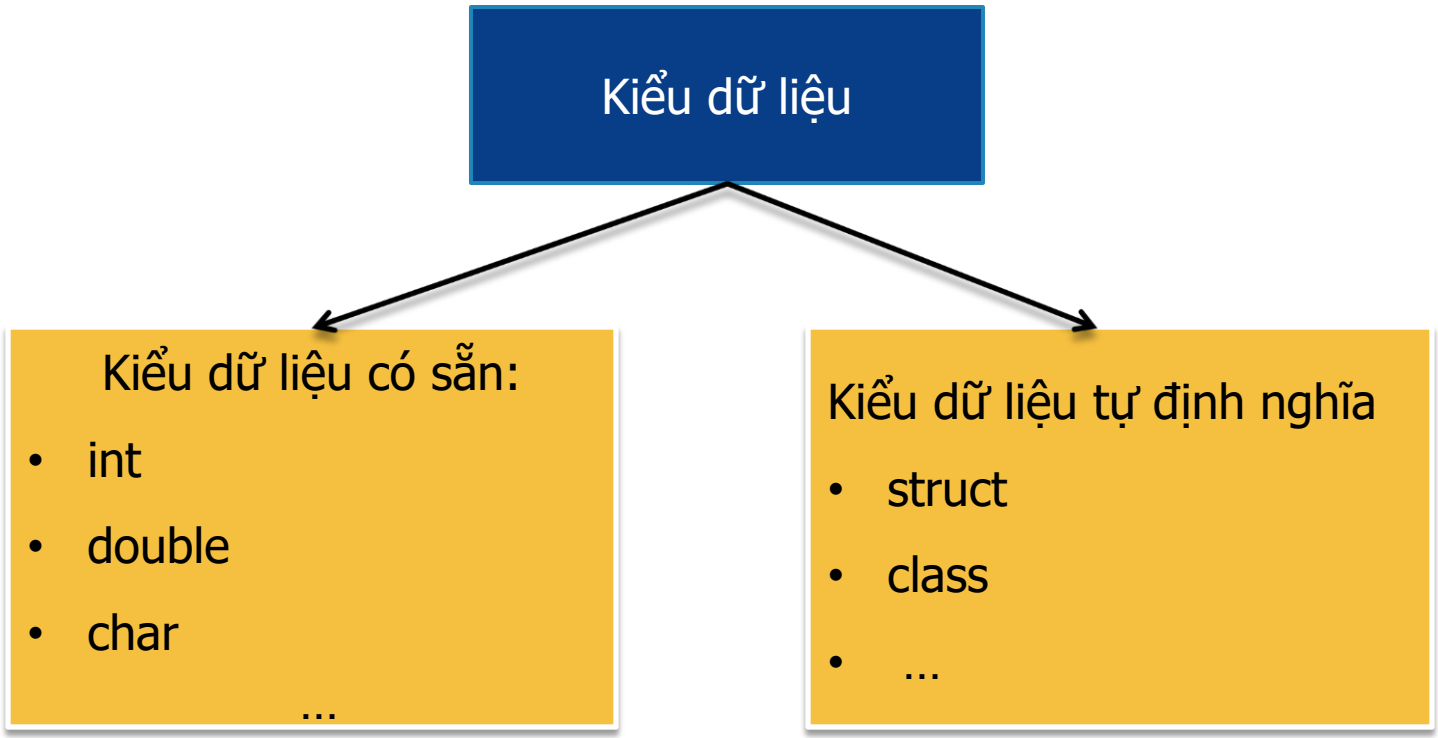
Mục tiêu

- Sau khi học xong chương này, người học có thể:

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

Nội dung

1. Giới thiệu
2. struct
3. class
4. enum


Kiểu dữ liệu
Kiểu dữ liệu có sẵn:
• int
• double
• char
...
Kiểu dữ liệu tự định nghĩa
• struct
• class
• ...

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

- 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

- 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 thànhh viên của struct: họ tên, địa chỉ, tuổi, điểm, ….

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
- CHÚ Ý: kết thúc việc định nghĩa struct phải có dấu ;

Ví dụ:

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

Khai báo biến kiểu struct:

```
structName variableName;
```

Ví dụ:

```
SinhVien sv;
```

Có thể vừa định nghĩa struct vừa khai báo.

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

Truy xuất các thành phần

```
variableName.memberName
```

Trong đó:

- varialeName: 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;
```

- 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.

- Nhập/xuất

Nhập hoặc xuất từng thành phần của biến kiểu struct.

Ví dụ:

```
SinhVien sv;
cout << "Nhap ho ten sinh vien: ";
getline(cin, sv.hoten);
cout << "Nhap dia chi: ";
getline(cin, sv.diachi);
cout << "Nhap tuoi: ";
cin >> sv.tuoi;
cout << "Nhap diem: ";
cin >> sv.diem;

cout << "Thong tin cua sinh vien la:\nTen: " <<
sv.hoten << "\nDia chi la: " << sv.diachi << "\nTuoi
la: " << sv.tuoi << "\nDiem la: " << sv.diem << endl;
```

- Bài tập

1. Viết chương trình định nghĩa `struct NhanVien` gồm các thông tin:
   - Mã nhân viên kiểu **`string`**
   - Họ tên nhân viên kiểu **`string`**
   - Năm sinh kiểu **`int`**
   - Chức vụ kiểu **`string`**
   - Bậc lương kiểu **`double`**

Sau đó khai báo để có thể nhập và xuất thông tin của 1 nhân viên cụ thể.

Trình tự chương trình đề xuất:

```
//chỉ thị tiền xử lý: #include…
//namespace
//định nghĩa struct NhanVien
//hàm main
{
        //khai báo biến kiểu NhanVien
        //Nhập từng thành phần của biến
        //Xuất thông tin từng thành phần
}
```

- 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.

- Bài tập

2. Viết chương trình định nghĩa `struct NhanVien` gồm các thông tin:
   - Mã nhân viên kiểu **`string`**
   - Họ tên nhân viên kiểu **`string`**
   - Năm sinh kiểu **`int`**
   - Chức vụ kiểu **`string`**
   - Bậc lương kiểu **`double`**

Sau đó viết hàm nhập và xuất thông tin của 1 nhân viên cụ thể.

- Gợi ý:

```
void nhap ( NhanVien &nv );

void xuat ( const NhanVien nv);
```

- Bài tập

3. Viết chương trình định nghĩa `struct PhanSo` gồm các thông tin:
   - Tử số kiểu int
   - Mẫu số kiểu int

Sau đó viết các hàm:

- Nhập và xuất thông tin 1 phân số.
- Hàm nhận vào 2 phân số, trả về kết quả cộng của 2 phân số này.
- Hàm main: khai báo phân số a, b, c thuộc kiểu `PhanSo`. Gọi hàm nhập giá trị cho 2 phân số a và b. Xuất thông tin 2 phân số này. Gọi hàm cộng để trả về kết quả tổng 2 phân số a và b. Gán kết quả cho phân số c và xuất c.

- Bài tập

Gợi ý bài 3:

```
void nhap (PhanSo &ps);
void xuat (const PhanSo ps);
PhanSo cong (PhanSo ps1, PhanSo ps2)
{
        PhanSo psKetQua;
        psKetQua.tuso = …;
        psKetQua.mauso = …;
        return psKetQua;
};
```

- Bài tập

4. Phát triển BT3 thành chương trình tính tổng, hiệu, tích, thương của 2 phân số.

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

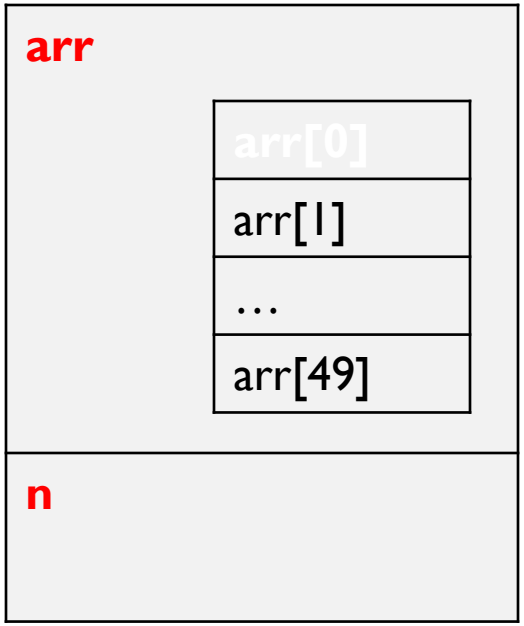
- struct và array
  - struct có thể chứa thành phần là array.

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;
```

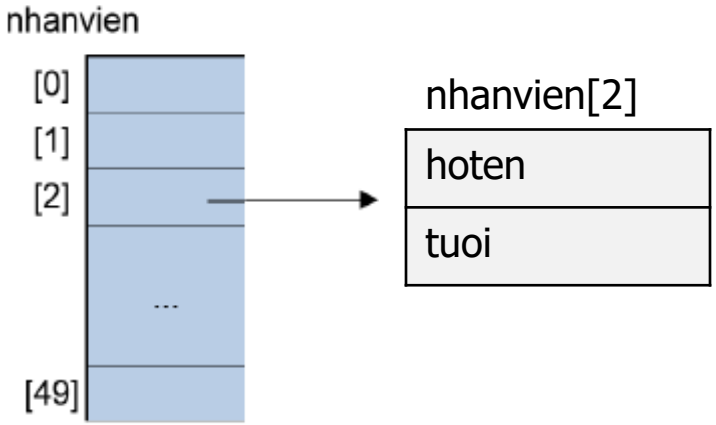


- struct và array
  - Ví dụ: dựa trên ví dụ struct kiểu dãy số nguyên. Viết các hàm nhập, xuất 1 biến kiểu intArr. Viết hàm trả về tổng các phần tử trong dãy số lưu trữ ở biến kiểu intArr.

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

Ví dụ:

```
struct NhanVien
{
    string hoten;
    int tuoi;
};
NhanVien nhanvien[50];
```



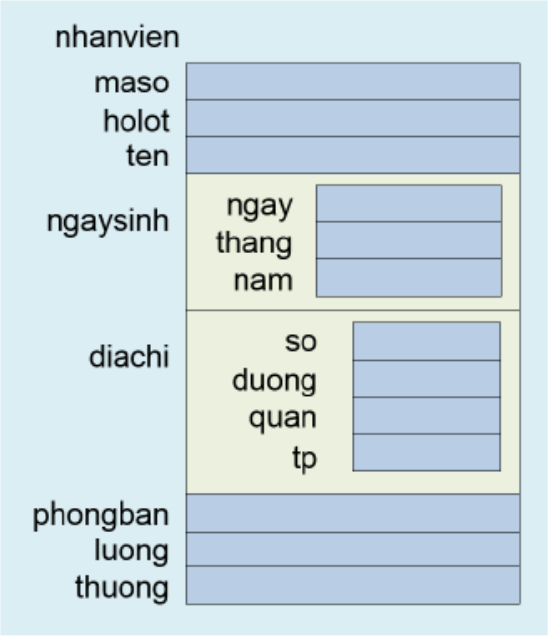
- struct và array
  - Ví dụ: dựa trên ví dụ struct NhanVien. Viết các hàm nhập, xuất thông tin cho 10 nhân viên.

- struct trong struct

```
struct NTN
{
    int ngay, thang, nam;
};
```

```
struct DiaChi
{
    string so;
    string duong;
    string quan;
    string tp;
};
```

```
struct NhanVien
{
    int maso;
    string holot;
    string ten;
    NTN ngaysinh;
    DiaChi diachi;
    string phongban;
    double luong;
    double thuong;
};
```



- Khai báo: **`NhanVien nv;`**
- Truy cập:

```
nv.ho = "Tran Van";
nv.ten = "Son";
nv.ngaysinh.ngay = 8;
nv.ngaysinh.thang = 3;
nv.ngaysinh.nam = 1980;
```

- 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;`

Bài tập struct:

5. Định nghĩa kiểu dữ liệu **SinhVien** gồm các thông tin:

- Họ tên kiểu string;
- Ngày tháng năm sinh kiểu NTN (struct NTN chứa từng thông tin ngày, tháng, năm đều là kiểu số nguyên);
- Điểm toán kiểu số thực.

Viết chương trình:

- Nhập thông tin cho toàn bộ sinh viên của 1 lớp học (tối đa 30 sinh viên). Lưu trữ toàn bộ thông tin 30 sinh viên dưới dạng mảng chứa từng giá trị kiểu SinhVien bằng cấp phát động.
- Xuất lại toàn bộ thông tin của sinh viên nào có điểm toán lớn hơn điểm trung bình của cả lớp.

Bài tập struct:

5. Định nghĩa kiểu dữ liệu **Kho** gồm: tên kho, tải trọng.

Tiến hành dùng cấp phát động để tạo ra nơi lưu trữ cho n kho có thông tin kiểu struct Kho (0 < n < 50).

Viết hàm:

+ Nhập đầy đủ dữ liệu của n kho này.

+ Xuất lại đầy đủ thông tin n kho.

+ Xuất tên kho nào còn trống. Biết rằng nếu tải trọng > 10 là còn trống.

+ main để kiểm chứng toàn bộ chương trình.

- Định nghĩa class
- Public members
- Member functions
- Constructor
- Header file

- 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.

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.
  - Private: được liệt kê bên dưới từ private và dấu :

    Ý nghĩa: 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: được liệt kê bên dưới từ public và dấu :

    Ý nghĩa: các biến liệt kê dạng public có thể được truy xuất trực tiếp từ bên ngoài class.
- Mặc định: private

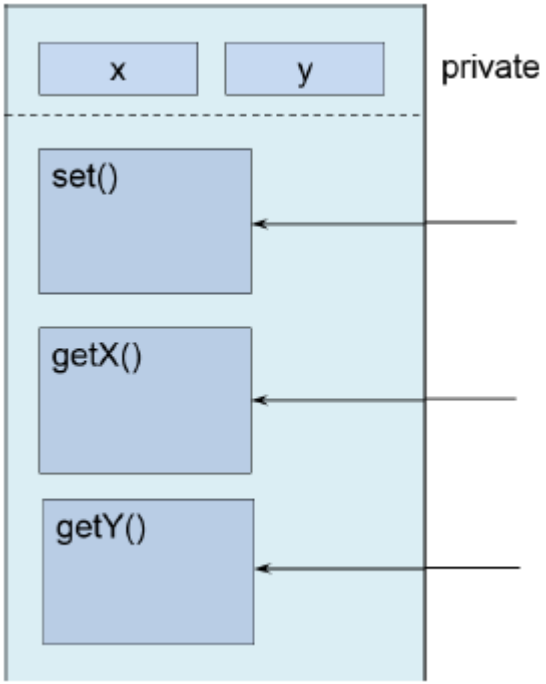
- Public members
- Với các member là biến đơn thì việc truy xuất tương tự như truy xuất struct_member.
- Member là function: Khai báo và định nghĩa bên trong class; cũng có thể dùng function prototype để khai báo trong class. Việc thực hiện định nghĩa function_member có thể diễn ra bên ngoài class với cú pháp:

```
dataType className::functionName([parameterList])
{
    statements
};
```

- Dấu :: là toán tử phân giải phạm vi.

Ví dụ: Xây dựng class Point

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



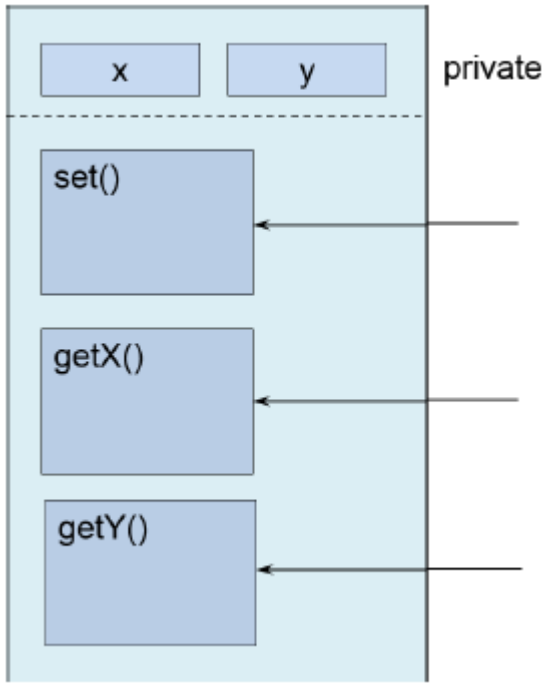
//getX: trả về giá trị của x

//getY: trả về giá trị của y

//set: gán 2 giá trị số nguyên: a cho x, b cho y.

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;
}
```

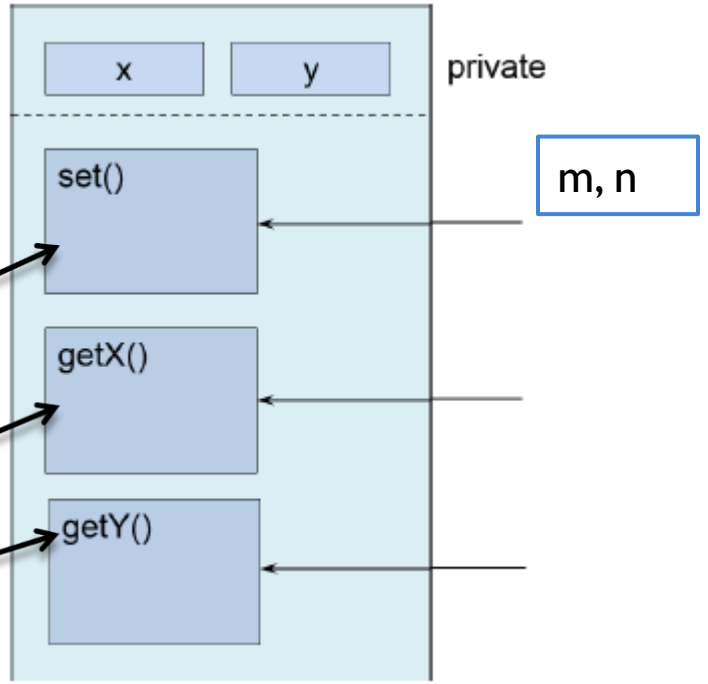


- Ví dụ: Xây dựng class Point bằng cách định nghĩa hàm trực tiếp trong class

```
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;
        }
};
```

Ví dụ: Xây dựng class Point

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



- Bài tập:

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

```
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 << "Nhap tu so va mau so phan so 1: "; cin >> ts1 >> ms1;
   cout << "Nhap tu so va mau so phan so 1: "; 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;
}
```

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**

- Constructor: là hàm tự động thực hiện khi 1 đối tượng thuộc class được tạo.
- Constructor dùng để khởi tạo các giá trị cho biến private trong class.
- Constructor:
  - Trùng tên với class và không có kiểu.
  - Mỗi class có 1 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).

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);
};
```

- Constructor:

Ví dụ:

```
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

- 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"
```

- Bài tập

7. Định nghĩa class PhanSo bao gồm các biến private là tuso, mauso; các hàm public là cong, tru, nhan, chia, ucln, rutgonps.

Sau đó viết chương trình kiểm chứng.

- Định nghĩa enum
- Các phép toán
- Nhập/xuất
- Mảng 2 chiều và kiểu liệt kê

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, …

- Ví dụ:

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

Thu ngay;

ngay = tu;

cout << ngay << endl;

//kết quả:  2
```

- Các phép toán
- 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<DataType>(variableName)`**

- Các phép toán
- Không thể gán giá trị kiểu int cho biến enum.
- Chỉ có thể so sánh giữa biến kiểu enum với giá trị.

- Các phép toán
- Không sử dụng phép toán số học

```
DenGiaoThong x, 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);
```

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

Q & A