# LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG C++

## Cài đặt phép toán

**Giảng viên:** ThS. Lương Trần Ngọc Khiết
**E-mail:** khietltn@hcmue.edu.vn

# Nội dung

- **Đặt vấn đề**
- **Cài đặt phép toán**
- **Một số bài tập đơn giản**

# Đặt vấn đề

- **Xét bài toán: cho 2 vector a và b trong không gian Oxy, hãy tính tổng, hiệu hai vector trên.**
- **Phân tích:**
  - **Đối tượng: vector**
  - **Xây dựng lớp: VECTOR**

# Đặt vấn đề

- **Phân tích (tt):**
  - **Dữ liệu:**
    - **Hai tọa độ x, y của vector**
  - **Phương thức:**
    - **Phương thức thiết lập, phương thức hủy bỏ**
    - **Phương thức SET(int, int) để đặt dữ liệu cho vector**
    - **Phương thức Print() để xuất vector**
    - **Phương thức Add(VECTOR, VECTOR) để cộng hai vector**
    - **Phương thức Sub(VECTOR, VECTOR) để trừ hai vector**

# Cách 1

## VECTOR

```
int x;          // Tọa độ x
int y;          // Tọa độ y
```

```
VECTOR(int xx=0, int yy=0);
VECTOR(VECTOR &v);
~VECTOR();
.....

void Set(int, int);
void Print();
void Add(VECTOR, VECTOR);
void Sub(VECTOR, VECTOR);
```

# Cách 1(tt)

```
class VECTOR
{
      int x, y;
   public:
      VECTOR(int xx = 0, int yy = 0)
      {
          printf("Vector duoc tao \n");
          x = xx;     y = yy;
      }
      VECTOR(VECTOR &v)
      {
          printf("Vector duoc tao \n");
          x = v.x;   y = v.y;
      }
      void Set(int xx, int yy)
      {    x = xx;     y = yy;    }
      void Print()
      {
            printf("Vector (%d,%d)\n",x,y);
      }
      void Add(VECTOR, VECTOR);
      void Sub(VECTOR, VECTOR);
};
```

```
void VECTOR::Add(VECTOR a, VECTOR b)
{
   x = a.x + b.x;
   y = a.y + b.y;
}
void VECTOR::Sub(VECTOR a, VECTOR b)
{
   x = a.x - b.x;
   y = a.y - b.y;
}

void main()
{
   VECTOR a(2,5), b(-2,4);
   a.Print();          b.Print();
   VECTOR c;
   c.Add(a,b);         c.Print();
   c.Sub(a,b);         c.Print();
   getch();
}
```

# Cách 2

## VECTOR

```
int x;          // Tọa độ x
int y;          // Tọa độ y
```

```
VECTOR(int xx=0, int yy=0);
VECTOR(VECTOR &v);
~VECTOR();
.....

void Set(int, int);
void Print();
VECTOR Add(VECTOR);
VECTOR Sub(VECTOR);
```

# Cách 2 (tt)

```
class VECTOR
{
      int x, y;
   public:
      VECTOR(int xx = 0, int yy = 0)
      {
          printf("Vector duoc tao \n");
          x = xx;     y = yy;
      }
      VECTOR(VECTOR &v)
      {
          printf("Vector duoc tao \n");
          x = v.x;   y = v.y;
      }
      void Set(int xx, int yy)
      {    x = xx;     y = yy;    }
      void Print()
      {
            printf("Vector (%d,%d)\n",x,y);
      }
      VECTOR Add(VECTOR);
      VECTOR Sub(VECTOR);
};
```

```
VECTOR VECTOR::Add(VECTOR a)
{
   VECTOR c(x + a.x, y + a.y);
   return c;
}
VECTOR VECTOR::Sub(VECTOR a)
{
   VECTOR c(x - a.x, y - a.y);
   return c;
}

void main()
{
   VECTOR a(2,5), b(-2,4);
   a.Print();          b.Print();
   VECTOR c;
   c = a.Add(b);      c.Print();
   c = a.Sub(b);      c.Print();
   getch();
}
```

# Đặt vấn đề (tt)

- **Nhận xét về các cách tiếp cận trên???**
  - **Cách thể hiện khi muốn lấy vector c là tổng vector a và vector b:**
    - **C1: c.Add(a,b);**
    - **C2: c = a.Add(b);**
    - **Nhận xét: cách gọi không tự nhiên**
  - **Cải thiện: liệu có thể thực hiện lệnh gọi:**
    - **c = a + b;**

# Cài đặt phép toán

- Sử dụng các phép toán cho phép trong C/C++ để định nghĩa phép toán theo mục đích riêng
- Các phép toán được sử dụng: +, -, *, /,++,--,!, %, &&, ||
- Cú pháp: sử dụng từ khóa

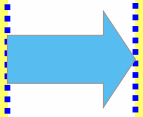
operator <phép toán>

thay cho tên hàm

# Ví dụ:

```
VECTOR VECTOR::Add(VECTOR a)
{
   VECTOR c(x + a.x, y + a.y);
   return c;
}
VECTOR VECTOR::Sub(VECTOR a)
{
   VECTOR c(x - a.x, y - a.y);
   return c;
}

void main()
{
   VECTOR a(2,5), b(-2,4);
   a.Print();          b.Print();
   VECTOR c;
   c = a.Add(b);       c.Print();
   c = a.Sub(b);       c.Print();
   getch();
}
```



```
VECTOR VECTOR::operator +(VECTOR a)
{
   VECTOR c(x + a.x, y + a.y);
   return c;
}
VECTOR VECTOR::operator - (VECTOR a)
{
   VECTOR c(x - a.x, y - a.y);
   return c;
}

void main()
{
   VECTOR a(2,5), b(-2,4);
   a.Print();          b.Print();
   VECTOR c;
   c = a+b;            c.Print();
   c = a-b;            c.Print();
   getch();
}
```

# Hàm toán tử

**Hàm toán tử có thể khai báo là hàm thành phần của lớp hoặc là hàm bạn của lớp.**

- **Khi hàm toán tử là hàm thành phần của lớp thì toán hạng thứ nhất trong phép toán phải là đối tượng thuộc lớp đó và số tham số hình thức của hàm toán tử phải bằng số ngôi của phép toán trừ đi một.**
- **Khi hàm toán tử là hàm bạn của lớp thì toán hạng thứ nhất trong phép toán không nhất thiết phải là đối tượng thuộc lớp đó và số tham số hình thức của hàm toán tử phải bằng số ngôi của phép toán.**

# Ví dụ

```
class SoPhuc {
    private:
        float re, im; //phần thực và phần ảo
    public:
        SoPhuc(float r = 0, float i = 0);
        void Xuat();
        //Hàm toán tử cộng hai đối tượng số phức (hai ngôi)
        SoPhuc operator+(const SoPhuc &u);
        /*Hàm toán tử cộng một số thực với một đối tượng số phức (hai ngôi)*/
        friend SoPhuc operator+(float x, const SoPhuc &u);
        //Hàm toán tử đảo dấu số phức (một ngôi)
        void operator-();
    };
```

# Ví dụ (tt)

```
//Định nghĩa các hàm thành phần
SoPhuc::SoPhuc(float r, float i) {
    cout << "Goi SoPhuc::SoPhuc(float, float) \n";
    re = r; im = i;
}
SoPhuc SoPhuc::operator+(const SoPhuc &u) {
    cout << "Goi SoPhuc::operator+(const SoPhuc &)\n";
    SoPhuc ret(re + u.re, im + u.im);  /*Gọi hàm thiết lập                    hai đối số*/
    return ret;
}
```

# Ví dụ (tt)

```
void SoPhuc::operator-() {
    cout << "Goi operator-() \n";
    re = -re; im = -im;
}
void SoPhuc::Xuat() {
    cout << re << (im >= 0 ? "+" : "-") << fabs(im)
<< "*i\n";
}
//Định nghĩa hàm tự do
SoPhuc operator+(float x, const SoPhuc &u) {
    cout << "Goi operator+(float, const SoPhuc &)
\n";
    SoPhuc ret(x + u.re, u.im);
    return ret;
}
```

# Ví dụ (tt)

```
//Hàm chính
void main() {
   SoPhuc a(-2, 4);
   cout << "So phuc a:"; a.Xuat();
   SoPhuc b(8, -6);
   cout << "So phuc b:"; b.Xuat();
   SoPhuc c = a + b;          //c = a.operator+(b)
   cout << "So phuc c:"; c.Xuat();
   SoPhuc d = a + 3; /*d = a.operator+(3) với 3 tự
            động chuyển kiểu thành SoPhuc với re = 3,
im = 0 */
   cout << "So phuc d:"; d.Xuat();
```

# Ví dụ (tt)

```
    SoPhuc e = 3 + a; // e = operator+(3, a)
    cout << "So phuc e:";
    e.Xuat();
    SoPhuc f = a + b + c; /* f =   (a.operator+(b)).
                              operator+(c)*/
    cout << "So phuc f:";
    f.Xuat();
    -a; //a.operator-();
    cout << "So phuc a:";
    a.Xuat();
}
```

# Chú ý

**Thay vì phải khai báo cả 2 hàm toán tử:**

```
SoPhuc operator+(const SoPhuc &u);
friend SoPhuc operator+(float x,
                        const SoPhuc &u);
```

**để chạy được với cả 3 câu lệnh:**

```
SoPhuc c = a + b;
SoPhuc d = a + 3;
SoPhuc e = 3 + a;
```

# Chú ý (tt)

ta có thể thay bằng chỉ một hàm toán tử:

```
friend SoPhuc operator+(const SoPhuc &u, const SoPhuc &v);
```

khi đó:

```
SoPhuc c = a + b; //operator+(a, b)
SoPhuc d = a + 3; /*operator+(a, 3), với 3 tự động chuyển kiểu thành SoPhuc*/
SoPhuc e = 3 + a; /*operator+(3,a), với 3 tự động chuyển kiểu thành SoPhuc*/
```

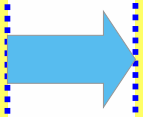
# Phép toán 1 ngôi

- **Ví dụ: cài đặt phép toán một ngôi " - " để tìm vector nghịch đảo của một vector**
- **Cụ thể, phải cài đặt sao cho có thể sử dụng lệnh c = -a**

# Phép toán 1 ngôi (tt)

```
VECTOR VECTOR::Add(VECTOR a)
{
   VECTOR c(x + a.x, y + a.y);
   return c;
}
VECTOR VECTOR::Inverse()
{
   VECTOR c(-x, -y);
   return c;
}

void main()
{
   VECTOR a(2,5), b(-2,4);
   a.Print();          b.Print();
   VECTOR c;
   c = a.Add(b);       c.Print();
   c = a.Inverse();    c.Print();
   getch();
}
```



```
VECTOR VECTOR::operator +(VECTOR a)
{
   VECTOR c(x + a.x, y + a.y);
   return c;
}
VECTOR VECTOR::operator - ()
{
   VECTOR c(-x, -y);
   return c;
}

void main()
{
   VECTOR a(2,5), b(-2,4);
   a.Print();          b.Print();
   VECTOR c;
   c = a+b;            c.Print();
   c = -a;             c.Print();
   getch();
}
```

# Phép toán 1 ngôi (tt) – sử dụng con trỏ this

```
VECTOR VECTOR::operator +(VECTOR a)
{
   VECTOR c(x + a.x, y + a.y);
   return c;
}
VECTOR VECTOR::operator - ()
{
   VECTOR c(-x, -y);
   return c;
}

void main()
{
   VECTOR a(2,5), b(-2,4);
   a.Print();          b.Print();
   VECTOR c;
   c = a+b;            c.Print();
   c = -a;             c.Print();
   getch();
}
```

```
VECTOR VECTOR::operator +(VECTOR a)
{
   VECTOR c(x + a.x, y + a.y);
   return c;
}
VECTOR VECTOR::operator - ()
{
   x = -x;  y = -y;
   return *this;
}

void main()
{
   VECTOR a(2,5), b(-2,4);
   a.Print();          b.Print();
   VECTOR c;
   c = a+b;            c.Print();
   c = -a;             c.Print();
   getch();
}
```

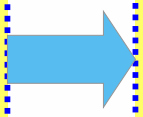
# Phép toán giữa hai kiểu khác nhau

- **Ví dụ: xây dựng phép toán để nhân vector với một số**
- **Cụ thể, phải cài đặt sao cho có thể sử dụng lệnh: c = a*2;**

# Phép toán giữa hai kiểu khác nhau (tt)

```
VECTOR VECTOR::(VECTOR a)
{
   VECTOR c(x + a.x, y + a.y);
   return c;
}
VECTOR VECTOR::Mul(int k)
{
   VECTOR c(x*k, y*k);
   return c;
}

void main()
{
   VECTOR a(2,5), b(-2,4);
   a.Print();          b.Print();
   VECTOR c;
   c = a.Add(b);       c.Print();
   c = a.Mul(2);       c.Print();
   getch();
}
```



```
VECTOR VECTOR::operator +(VECTOR a)
{
   VECTOR c(x + a.x, y + a.y);
   return c;
}
VECTOR VECTOR::operator * (int k)
{
   VECTOR c(x*k, y*k);
   return c;
}

void main()
{
   VECTOR a(2,5), b(-2,4);
   a.Print();          b.Print();
   VECTOR c;
   c = a+b;            c.Print();
   c = a*2;            c.Print();
   getch();
}
```

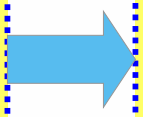
# Phép toán [ ]

- **Sử dụng để rút trích phần tử trong một mảng, danh sách.**
- **Ví dụ: xây dựng phép toán để trích thành phần của một vector**
- **Cụ thể, nếu a là một vector thì a[1] sẽ cho giá trị x của a, a[2] sẽ cho giá trị y của a**

# Phép toán [ ] (tt)

```
int VECTOR::TP(int k)
{
    if (k==1) return x;
    return y;
}
VECTOR VECTOR::Mul(int k)
{
    VECTOR c(x*k, y*k);
    return c;
}

void main()
{
    VECTOR a(2,5), b(-2,4);
    a.Print();          b.Print();
    VECTOR c;
    c = a.Add(b);       c.Print();
    int x = a.TP(2);    printf("%d",x);
    getch();
}
```



```
int VECTOR::operator [ ](int k)
{
    if (k==1) return x;
    return y;
}
VECTOR VECTOR::operator * (int k)
{
    VECTOR c(x*k, y*k);
    return c;
}

void main()
{
    VECTOR a(2,5), b(-2,4);
    a.Print();          b.Print();
    VECTOR c;
    c = a+b;            c.Print();
    int x = a[2];       printf("%d",x);
    getch();
}
```

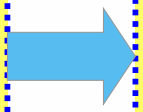
# Phép toán gán =

- **Việc khai báo tường minh hàm toán tử gán chỉ cần thiết khi các đối tượng tham gia phép gán có thành phần dữ liệu kiểu con trỏ.**
- **Khi không khai báo tường minh một hàm toán tử gán nào cho lớp thì trình biên dịch sẽ tự động cung cấp cho lớp một hàm toán tử gán mặc định để thực hiện câu lệnh gán hai đối tượng.**

# Phép toán gán = (tt)

```
void VECTOR::Gan(VECTOR u)
{
   x=u.GetX();
   y=u.GetY();
}

void main()
{
 VECTOR a(2,5);
    a.Print();
    VECTOR b;
    b.Gan(a);          b.Print();
    getch();
}
```



```
VECTOR VECTOR::operator =(VECTOR u)
{
    VECTOR c(u);
    return c;
}

void main()
{
    VECTOR a(2,5);
    a.Print();
    VECTOR b;
    b = a;   b.Print();
    getch();
}
```

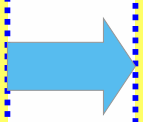
# Phép toán nhập >> và xuất <<

- **Nhằm cho phép các đối tượng đứng bên phải toán tử khi thực hiện các thao tác nhập xuất.**
- **Hàm toán tử nhập/xuất không được phép khai báo như là hàm thành phần mà phải được khai báo như là hàm bạn.**

# Phép toán nhập >> và xuất << (tt)

```
void VECTOR::Nhap()
{
   cout<<“Nhap toa do x, y”;
   cin>>x>>y;
}
void VECTOR::Xuat()
{
   cout<<“(“<<u.x<<“,”<<u.y<<“)”;
}

void main()
{
    VECTOR a;
    a.Nhap();
    a.Xuat();
    getch();
}
```



```
istream& operator >>(istream &is, VECTOR
&u)
{
    cout<<“Nhap toa do x, y”;
    is>>u.x>>u.y;
    return is;
}
ostream& operator <<(ostream &os, const
VECTOR &u)
{
    os<<“(“<<u.x<<“,”<<u.y<<“)”;
    return os;
}

void main()
{
    VECTOR a;
    cin>>a;
    cout<<a;
    getch();
}
```

# Phép toán nhập >> và xuất << (tt)

**<u>Chú ý:</u> Giá trị trả về của hàm toán tử nhập/xuất được chọn là trả về tham chiếu đến đối tượng cin/cout nhằm cho phép thực hiện nhập/xuất liên tiếp nhiều đối tượng.**

# Một số bài tập đơn giản

- **Hãy cài đặt >>, <<, =, +, -, * cho lớp số phức**
- **Hãy cài đặt phép toán ++ (tăng x, y lên 1), -- (giảm x, y xuống 1), * (tính tích vô hướng hai vector) cho lớp Vector**

# Câu hỏi và thảo luận

# Chân thành cảm ơn !