PHẦN 2: LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG (OBJECT-ORIENTED PROGRAMMING)

NỘI DUNG

- Lịch sử các phương pháp lập trình
- Khai báo và sử dụng lớp
- Hàm tạo, Hàm hủy
- Quan hệ bạn bè: hàm bạn, lớp bạn
- **Định nghĩa toán tử trong lớp**
- Kế thừa và đa hình
- Khuôn mẫu (template)

ĐỊNH NGHĨA TOÁN TỬ TRONG LỚP

GIỚI THIỆU CHUNG

- Nhận xét chung về phép toán cộng trong biểu thức:

  a + b và a++

- Trong lớp hoàn toàn có thể định nghĩa các phép toán cho các đối tượng.
- Các phép toán được định nghĩa bằng một trong hai cách:
  - ✓ Là một hàm thành phần của lớp.
  - ✓ Là hàm bạn của 1 lớp.

TÊN CỦA HÀM TOÁN TỬ

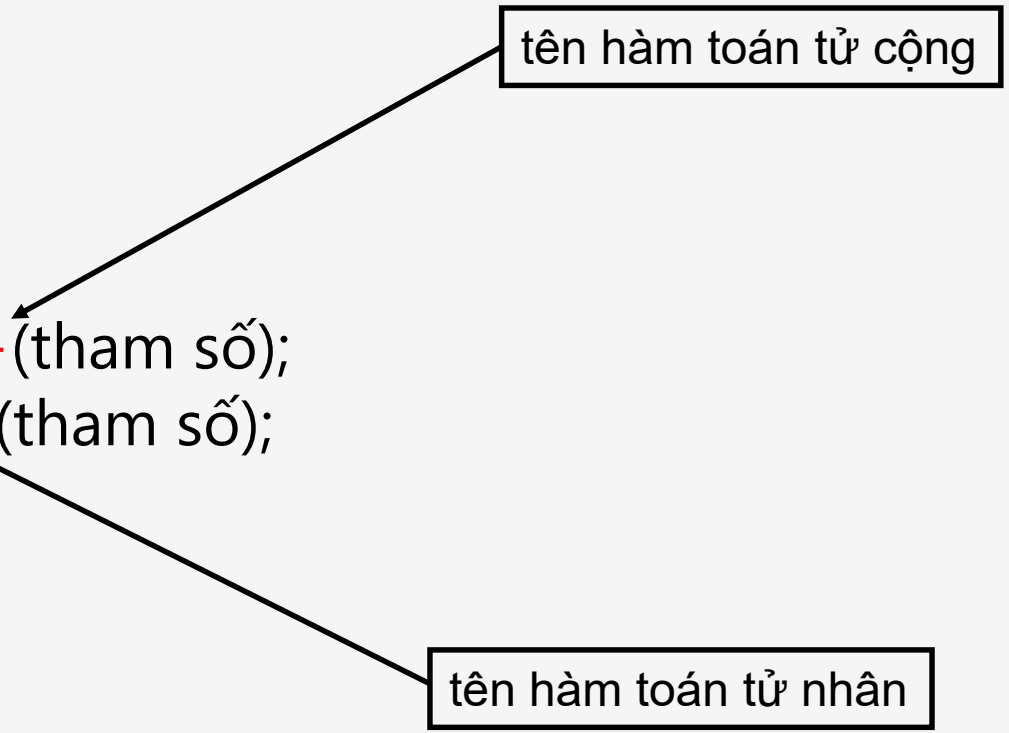
- Tên hàm toán tử = operator + toán tử
- Trong đó: Toán tử phải là một trong các phép toán với các kiểu dữ liệu cơ sở như: +, - , = , +=,....
- Một số tên hàm toán tử quen thuộc:

Tên hàm	Ý nghĩa
operator+	Định nghĩa phép cộng
operator-	Định nghĩa phép trừ
operator/	Định nghĩa phép chia
operator*	Định nghĩa phép nhân
operator+=	Định nghĩa phép cộng

HÀM TOÁN TỬ LÀ HÀM THÀNH PHẦN CỦA LỚP

KHAI BÁO

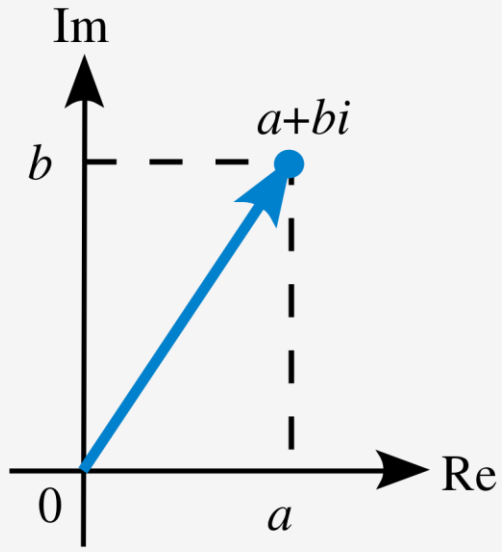
```
class  tên_lớp
{
// các thành phần dữ liệu
// các hàm thành phần
kiểu_của_hàm   operator+(tham số);
kiểu_của_hàm   operator*(tham số);
……
};
```



VÍ DỤ 7.1

Xây dựng lớp số phức (complex)

- Thành phần dữ liệu:
  - ✓ double a, b.
- Hàm thành phần:
  - ✓Hàm tạo 2 tham số
  - ✓Hàm hiển thị
  - ✓Hàm toán tử cộng
  - ✓Hàm toán tử trừ



```cpp
// vi du 7.1
#include <iostream>
#include <math.h>
using namespace std;
//  dinh nghia lop so phuc
class complex {
      double a, b;    // x = a + b * i
      public:
      complex() { a = 0; b = 0;}
      complex(double, double);
      complex operator+(complex); // ham toan tu c
      complex operator-(complex);  // ham toan tu
      void display();
};
// trien khai cac ham thanh phan
complex::complex(double a, double b) {
    this->a = a;
    this->b = b;
}
complex complex::operator+(complex x) {
   complex tmp;
   tmp.a = a + x.a;
   tmp.b = b + x.b;
   return tmp;
}
complex complex::operator-(complex x) {
   complex tmp;
   tmp.a = a - x.a;
   tmp.b = b - x.b;
   return tmp;
}
void complex::display() {
     cout << a << " + " << b << " * i" << endl;
}
main() {
  complex A(4, 8);
  complex B(3, 4);
  complex C;
  cout << " A = ";
  A.display();
  cout << " B = ";
  B.display();
  C = A + B; //C = A.operator+(B);
  cout << " C = ";
  C.display();
  return 0;
}
```

NHẬN XÉT

- Với lớp complex đã định nghĩa như trên thì:
  phép toán c = a+b $\Longleftrightarrow$ c = a.operator+(b)
- Hàm toán tử phải có thuộc tính public.
- Trong lời gọi a.operator+(b) thì a đóng vai trò là tham số ngầm định, b đóng vai trò là tham số tường minh.
- Chương trình dịch sẽ không hiểu được c = 10 + a

HÀM TOÁN TỬ LÀ HÀM BẠN

KHAI BÁO

```
class  tên_lớp
{
// các thành phần dữ liệu
// các hàm thành phần
friend kiểu_của_hàm   operator+(tham số);
friend kiểu_của_hàm   operator*(tham số);
.....
};
```

```
//triển khai các hàm bạn ngoài lớp
kiểu_của_hàm  operator+(tham số)
{
.....  // nội dung của hàm toán tử
}
```

VÍ DỤ 7.2

- Giống Ví dụ 7.1, nhưng thêm vào hàm toán tử để thực hiện phép toán giữa 1 hằng số và một đối tượng. Tức là thực hiện được phép toán:

```
c = 10 + a;
```

```cpp
// vi du 7.2
#include <iostream>
#include <math.h>
using namespace std;
class complex {
      double a, b;    // x = a + b * i;
      public:
      complex() { a = 0; b = 0;}
      complex(double, double);
      complex operator+(complex);
      complex operator-(complex);
      friend complex operator+(double, complex);
      void display();
};
// trien khai cac ham thanh phan
complex::complex(double a, double b) {
   this->a = a;
   this->b = b;
}
complex complex::operator+(complex x) {
   complex tmp;
   tmp.a = a + x.a;
   tmp.b = b + x.b;
   return tmp;
}
complex complex::operator-(complex x) {
   complex tmp;
   tmp.a = a - x.a;
   tmp.b = b - x.b;
   return tmp;
}
complex operator+(double z, complex x) {
      complex tmp;
      tmp.a = x.a + z;
      tmp.b = x.b;
      return tmp;
}
void complex::display() {
    cout << a << " + " << b << " * i" << endl;
}
main() {
  complex A(4, 8);
  complex B(3, 4), C;
  cout << " A = ";  A.display();
  cout << " B = ";  B.display();
  A = 10 + B;
  cout << " A = ";
  A.display();
  return 0;
}
```

LƯU Ý

- Hàm toán tử 1 ngôi, thì nên khai báo là hàm thành phần của lớp
- Hàm toán tử 2 ngôi thì phải khai báo là hàm bạn của lớp

TOÁN TỬ NHẬP (>>) VÀ XUẤT (<<)

- Chúng ta có thể định nghĩa toán tử nhập (>>) và toán tử xuất (<<) cho các đối tượng của một lớp.
- Các hàm này không là hàm thành viên của lớp, thường phải khai báo là hàm bạn của lớp
- Khuôn dạng:

```
istream & operator>>(istream &is, ClassName &x);
ostream & operator<<(ostream &os, ClassName &x);
```

Hàm toán tử ">>" trả về tham chiếu chỉ đến dòng nhập istream
Hàm toán tử "<<" trả về tham chiếu chỉ đến dòng xuất ostream

VÍ DỤ 7.3

- Bổ sung hàm toán tử “>>” và “<<” cho lớp complex

```cpp
#include <iostream>
#include <math.h>
// class: complex
using namespace std;
//  dinh nghia lop so phuc
class complex {
      double a, b;    // x = a*i + b
      public:
      complex() { a = 0; b = 0;}
      complex(double, double);
      complex operator+(complex); // ham toan tu cong
      complex operator-(complex);  // ham toan tu tru
      complex operator+=(complex);  // ham toan tu +=
      complex operator+(double);
      friend istream & operator>>(istream &is, complex &x);
      friend ostream & operator<<(ostream &os, complex &x);
      friend complex operator+(double, complex);
      void display();
};
// trien khai cac ham thanh phan
complex::complex(double a, double b) {
   this->a = a;
   this->b = b;
}
```

```
25  complex complex::operator+(double z) {
26          complex tmp;
27          tmp.a = this->a + z;
28          tmp.b = this->b;
29  }
30  complex complex::operator+=(complex x) {
31      a = a + x.a;
32      b = b + x.b;
33      return *this;
34  }
35  complex complex::operator+(complex x) {
36      complex tmp;
37      tmp.a = a + x.a;
38      tmp.b = b + x.b;
39      return tmp;
40  }
41  ostream &operator<<(ostream &os, complex &x) {
42          os << "*****" << endl;
43          os << x.a << " + " << x.b << " * i" << endl;
44          os << "******" << endl;
45          return os;
46  }
```

```
47  istream &operator>>(istream &is, complex &x) {
48        cout << "Cho biet a: ";
49        is >> x.a;
50        cout << "Cho biet b: ";
51        is >> x.b;
52        return is;
53  }
54  complex complex::operator-(complex x) {
55     complex tmp;
56     tmp.a = a - x.a;
57     tmp.b = b - x.b;
58     return tmp;
59  }
60  complex operator+(double z, complex x) {
61        complex tmp;
62        tmp.a = z + x.a;
63        tmp.b = x.a;
64        return tmp;
65  }
66  void complex::display() {
67        cout << a << " + " << b << " * i" << endl;
68  }
```

```
69  main() {
70     complex A(4, 8);
71     complex B(3, 4);
72     complex C;
73     cout << " A = ";
74     A.display();
75     cout << " B = ";
76     B.display();
77     C = A + B; //C = A.operator+(B);
78     cout << " C = ";
79     C.display();
80     A += C;    // A = A.operator+=(B);
81     A.display();
82     A.display();
83     complex D;
84     cin >> D;  // goi ham toan tu nhap
85     cout << D;  // goi ham toan tu xuat
86     return 0;
87  }
```

BÀI TẬP TẠI LỚP

- Xây dựng lớp PHANSO để biểu diễn khái niệm phân số như sau:
- Dữ liệu: a, b (tử số, mẫu số)
- Hàm thành phần:
  - Các hàm tạo
  - Nhập phân số
  - Phép cộng, trừ, nhân, chia hai phân số
  - Đưa phân số ra màn hình
  - Thử nghiệm chương trình: Nhập hai phân số và thực hiện các phép toán với hai phân số đã nhập, đưa kết quả ra màn hình.

BÀI TẬP TẠI LỚP

- Hãy xây dựng lớp DATE gồm các thông tin và các hàm như sau
  - Dữ liệu: d, m, y : lưu ngày, tháng, năm.
  - Hàm tạo không tham số
  - Hàm tạo 3 tham số
  - Hàm toán tử nhập (>>)
  - Hàm toán tử cộng một ngày với 1 số nguyên
  - Hàm toán tử hiển thị ngày dạng dd/mm/yyyy (<<)
  - Các hàm toán tử so sánh hai ngày d1, d2 (>,<,==)
  - Hàm trả về ngày tiếp theo
  - Hàm trả về ngày, hàm trả về tháng, hàm trả về năm