CHƯƠNG 3

PHƯƠNG THỨC TOÁN TỬ

3.1. Phân loại toán toán tử

- **Toán tử một ngôi:**
  - Là những toán tử thực hiện trên một toán hạng.
  - Gồm có:
    - Phép phủ định (!)
    - Phép tăng 1 đơn vị (++)
    - Giảm một đơn vị (--)
    - Phép đổi dấu (-) …
- **Toán tử hai ngôi:**
  - Là những toán tử thực hiện trên 2 toán hạng.
  - Gồm các toán tử: cộng (+), trừ (-), nhân (*), chia (/)….

3.2. Cài đặt hàm toán tử

- Trong lập trình cấu trúc một hàm toán tử có đặc điểm sau:
  - Hàm toán tử được cài đặt tương tự hàm thông thường, chỉ khác ở tên hàm và cách sử dụng.
  - Tên hàm: được viết theo dạng: operator <Ký hiệu toán tử>
- Cú pháp của hàm:

```
<Kiểu trả về> operator <Ký hiệu toán tử> (các đối số)
{
        Thân hàm toán tử;
}
```

- Ví dụ: Hàm toán tử cộng hai số thực bất kỳ

```
float operator+(float x, float y){
      return x + y;
}
```

3.3. Sử dụng hàm toán tử

- Cách 1: gọi như hàm thông thường. VD: để cộng hai số thực a, b ta có thể viết:

```
cout<<"Tong cua hai so a va b la: ";
cout<<operator+(a,b);
```

- Cách 2: gọi như một toán tử: Ta có thể sử dụng hàm toán tử như một toán tử, tức là ta có thể viết:

```
cout<<"Tong hai so S1 va S2 la"<<S1+S2;
```

- Ví dụ: Một số phức có dạng: <Phần thực> + i * <Phần ảo>. Cho hai số phức X = a + i*b và Y = c + i * d. Khi đó X + Y sẽ cho số phức có dạng: X+Y = (a+c) + i * (b + d). Hãy định nghĩa hàm toán tử để thực hiện cộng hai số phức bất kỳ.

Sử dụng hàm toán tử (tt)

```
struct SoPhuc {
   float phanThuc;
   float phanAo;
};
//Dinh nghia ham toan tu cong hai so phuc
SoPhuc operator+(SoPhuc x, SoPhuc y) {
   SoPhuc tg;
   tg.phanThuc = x.phanThuc + y.phanThuc;
   tg.phanAo = x.phanAo + y.phanAo;
   return tg;
}
```

Sử dụng hàm toán tử (tt)

```
int main() {
//Khai bao hai so phuc x va y va tong T
   SoPhuc x, y, T;
   x.phanThuc = 2; x.phanAo = 3;
   y.phanThuc = 3; y.phanAo = 5;
//Cong hai so phuc va in ket qua
   T = operator+(x, y);
//Co the viet T = x + y
   cout<<"Ket qua "<<T.phanThuc;
   cout<<"+ i * "<<T.phanAo;
}
```

3.4. Cài đặt phương thức toán tử

- **Cài đặt phương thức toán tử một ngôi**
  - Phương thức toán tử cũng tương tự như hàm toán tử.
- **Ví dụ: Cài đặt lớp số phức bao gồm**
  - Các thuộc tính: phần thực và phần ảo
  - Các phương thức:
    - PT khởi tạo có đối khởi gán giá trị cho phần thực và ảo
    - Phương thức khởi tạo không đối.
    - Phương thức đổi dấu.
    - Phương thức hiển thị số phức.
  - Viết chương trình chính để tạo một số phức và in kết quả sau khi đã đổi dấu số phức ra màn hình.

Cài đặt phương thức toán tử (tt)

```
class SoPhuc {
   private:
      float phanThuc, phanAo;
   public:
      SoPhuc() {
         phanThuc = 0; phanAo = 0;
       }
      SoPhuc(float a, float b) {
         phanThuc = a;
         phanAo   = b;
      }
      void xuat(){
         cout<<"So phuc la "<<phanThuc;
         cout<<" + i * "<<phanAo;
      }
```

Cài đặt phương thức toán tử (tt)

```
//Phuong thuc toan tu doi dau
   SoPhuc operator-(){
      SoPhuc tg;
      tg.phanThuc = -phanThuc;
      tg.phanAo   = -phanAo;
      return tg;
   }
};
int main(){
   SoPhuc x(2, 3);
   SoPhuc y = x.operator-();
   //co the viet y = -x;
   y.xuat();
   getch();
}
```

Cài đặt phương thức toán tử (tt)

- **Nhận xét:**
  - Phương thức toán tử một ngôi không có đối vào. Thực chất phương thức toán tử đổi dấu trên đã bao gồm một đối mặc định, đó là con trỏ this.
  - Con trỏ this luôn là đối mặc định của các phương thức toán tử. Như vậy, hai cách viết sau là tương đương.

tg.phanThuc = -phanThuc; tg.phanAo = -phanAo;	tg.phanThuc = -this->phanThuc; tg.phanAo = -this->phanAo;

Khi sử dụng PTTT một ngôi ta cũng có 2 cách như với hàm toán tử. Như vậy, hai cách viết sau là tương đương:

SoPhuc y = x.operator-();	SoPhuc y = -x;

Cài đặt phương thức toán tử (tt)

- **Cài đặt phương thức toán tử hai ngôi**
  - Trong PT toán tử, con trỏ this luôn là một đối số mặc định.
  - Với phương thức toán tử hai ngôi, thay vì có hai đối vào, ta chỉ cần một đối, đối còn lại là con trỏ this.
- **Ví dụ:**

  Cài đặt lớp số phức ở trên với phương thức toán tử hai ngôi cộng hai số phức bất kỳ.

```
SoPhuc operator+(SoPhuc y)
{
    SoPhuc tg;
    tg.phanThuc = this->phanThuc + y.phanThuc;
    tg.phanAo   = this->phanAo + y.phanAo;
    return tg;
}
```

3.5. Cài đặt hàm toán tử nhập - xuất

```
ostream& operator<<(ostream& out, SoPhuc S)
{
  out<<S.phanThuc<<" + "<<S.phanAo<<"i"<<endl;
  return out;
}

istream& operator>>(istream& in, SoPhuc &P)
{
  cout<<"Nhap phan thuc: ";
  in>>P.phanThuc;
  cout<<"Nhap phan ao: ";
  in>>P.phanAo;
  return in;
}
```