Bài 3: Quá tải

Nội dung

1. Quá tải hàm
2. Quá tải phương thức
3. Quá tải toán tử
4. Hàm bạn (friend function)
5. Ôn tập
6. Bài tập

Ôn lại

1. Con trỏ this
2. Cấu tử và sử dụng cấu tử
3. Huỷ tử và sử dụng huỷ tử
4. Cấu tử và huỷ tử ngầm định
5. Quá tải và quá tải cấu tử
6. Cấu tử sao chép

3

Ví dụ Bài toán xây dựng lớp Phân số

- Bài toán: Xây dựng lớp Phân số
  - Một phân số gồm tử số và mẫu số.
  - Có thể rút gọn phân số, cộng 2 phân số...

4

Lớp phân số

- Xây dựng lớp Phân số cơ bản:
  - class Phanso
  - {
  - private:

```
int tuso;     // tử số
int mauso;  // mẫu số
```

  - public:

```
Phanso();         // cấu tử mặc định, đặt phân số 1/1
Phanso(int a);  // đặt phân số a/1
Phanso(int a, int b); // đặt phân số a/b
Phanso(Phanso &ps);  // cấu tử sao chép
void DatTS(int ts);     // đặt giá trị cho tử số
void DatMS(int ms);  // đặt giá trị cho mẫu số
int LayTS();              // lấy giá trị tử số
int LayMS();             // lấy giá trị mẫu số
void InPS();              // in phân số dưới dạng a/b
Phanso  Rutgon();        //Rút gọn phân số
```

5

Lớp phân số

- Sau khi cài đặt xong lớp Phanso, kiểm tra lớp này:

```
int main()
{
      Phanso  x, y(5), z(2,7), k(y);
      x.DatTS(3);
      x.DatMS(5);
      cout<<"Phan so x co tu so la "<<x.LayTS()
            <<" va mau so la "<<x.LayMS()<<endl;
      cout<<"Phan so nay co dang : ";
      x.InPS();
}
```

6

Lớp phân số

- Xây dựng phương thức cộng 2 phân số
  - class Phanso
  - {
  - public:
  - ...
  - Phanso Cong(Phanso ps);
  - };
- Kiểm tra:
  - Phanso x(3,6), y(2,5), z;
  - z = x.Cong(y);
  - z = z.Rutgon();
  - cout<<"Tong cua "; x.InPS();
  - cout<<" va "; y.InPS();
  - cout<<" la "; z.InPS();

7

Quá tải (Overloading)

- Quá tải trong C++ có 2 dạng:
  - Dùng nhiều hàm có cùng một tên nhưng có đối số khác nhau (quá tải hàm)
  - Dùng các cài đặt của riêng lớp cho các toán tử, ví dụ như +, -, ++… (quá tải toán tử)
- Hai khả năng của quá tải đều rất mạnh, nhưng để sử dụng chúng một cách hiệu quả đòi hỏi người lập trình phải thận trọng.

8

Quá tải hàm

- C++ cho phép tạo ra các hàm có tên trùng nhau nhưng khác nhau ở các đối số. Điều này gọi là quá tải hàm (function overloading).
- Ví dụ:
  - int abs(int i);
  - long abs(long i);
  - double abs(double d);
  - int Tong(int a, int b);
  - int Tong(int a, int b, int c);
  - long Tong(long a, long b);

9

Quá tải hàm

- Chú ý:
  - Không thể viết quá tải hàm có giá trị trả về khác nhau và các đối số giống nhau. Ví dụ khai báo sau sẽ bị thông báo lỗi khi biên dịch:
    - int Tong(int a, int b);
    - long Tong(int a, int b);
  - Cần đảm bảo kiểu của các đối số là khác nhau thực sự. Kiểu định nghĩa bởi **typedef** chỉ là một bí danh của một kiểu đã có, vì vậy đoạn chương trình sau sẽ bị lỗi:
    - typedef int Songuyen;
    - int Tang(int a);
    - int Tang(Songuyen a);
    - …

10

Quá tải phương thức

- Các phương thức của lớp cũng có thể được quá tải theo cách của quá tải hàm
- Đối tượng sẽ tự động lựa chọn phương thức thích hợp vào lúc gọi
- Ví dụ: Lớp Phanso có 2 phương thức cộng
  - *// Cộng một phân số với một phân số*
  - Phanso Cong(Phanso ps);
  - *// Cộng một phân số với một số nguyên*
  - Phanso Cong(int n);
- Đặc biệt, nên quá tải các cấu tử của lớp để cung cấp nhiều cách khởi tạo đa dạng.

11

Quá tải toán tử

- Với mỗi kiểu dữ liệu được xây dựng sẵn như int, float, char…, ta có các toán tử thao tác trên đó như ++, +, -…
- Với các lớp, ta vẫn thường xây dựng phương thức để thực hiện các thao tác trên đó. Ví dụ:
  - Lớp Phanso có phương thức Cong
  - Để cộng 2 phân số, ta dùng:

    z = x.Cong(y); *// cách viết này không "tự nhiên" !*

12

Quá tải toán tử

- C++ cho phép xây dựng các lớp trong đó có thể định nghĩa lại thao tác của các toán tử. Điều này gọi là quá tải toán tử (operator overloading).
- Các toán tử có thể quá tải là:
  - ++, --, +, -, *, /, +=, -=, *=, /=, <, >, <<…
- Cú pháp quá tải toán tử: *kiểu_trả_về* ***operator*** *op(đối_số);*
  - kiểu_trả_về: Là kiểu dữ liệu cơ bản int, float… hoặc một lớp
  - operator: Từ khoá
  - op: Một trong các toán tử có thể quá tải
  - đối số: Danh sách các đối số theo cách dùng toán tử
- Ví dụ:
  - int operator++(); *// quá tải toán tử ++*
  - Phanso operator+(Phanso ps); *// quá tải toán tử +*

13

Quá tải toán tử

- Ta sẽ quá tải toán tử + trong lớp Phanso để thay cho phương thức Cong

```
class Phanso
{
  public:
        …
        Phanso operator+(Phanso ps);
};

Phanso operator+(Phanso ps)
{
        int kqts = ts*ps.ms + ps.ts*ms;
        int kqms = ms*ps.ms;
        return Phanso(kqts, kqms); // trả về đối tượng không tên
}
```

14

Quá tải toán tử

- Kiểm tra 1:
  - int main()
  - {
  - Phanso x(4,5), y(3,8), z;
  - z = x + y;
  - z.InPS();
  - }
- Kiểm tra 2:
  - int main()
  - {
  - Phanso x(4,5), z;
  - z = x + 2; *// câu lệnh này báo lỗi !*
  - z.InPS();
  - }

15

Quá tải toán tử

- Để có thể cộng một phân số với một số nguyên, ta tiếp tục quá tải toán tử + với đối số kiểu int:
  - class Phanso
  - {
  - public:
  - ...
  - Phanso operator+(int n);
  - };
- Kiểm tra 2:
  - int main()
  - {
  - Phanso x(4,5), z;
  - z = x + 2; *// bây giờ câu lệnh này là hợp lệ*
  - z.InPS();
  - }

16

Quá tải toán tử

- Ta sẽ quá tải toán tử ++ trong lớp Phanso:

```
class Phanso
{
  public:
        ...
        void operator++();
};

void operator++()
{
        ts = ts + ms;
}
```


Quá tải toán tử

- Kiểm tra 1:

```
int main()
{
        Phanso x(4,5);
        ++x;
        x.InPS();
}
```

- Kiểm tra 2:

```
int main()
{
        Phanso x(4,5), z;
        z = ++x; // câu lệnh này báo lỗi !
        z.InPS();
}
```

Quá tải toán tử

- Để có thể sử dụng toán tử ++ trong câu lệnh gán, ta sẽ định nghĩa lại toán tử ++ như sau:

```
class Phanso
{
   public:
          ...
          Phanso operator++();
};

Phanso operator++()
{
          ts = ts + ms;
          return  operator+(1);  // trả về kết quả của toán tử + với đối số giá trị 1
}
```

- Kiểm tra 2:

```
          ...
          z = ++x; // câu lệnh này bây giờ hợp lệ
```


Quá tải toán tử

- Nguyên lý quá tải toán tử khá đơn giản. Về thực chất toán tử quá tải của lớp có vai trò như một phương thức, nó không hoàn toàn mang ý nghĩa gốc.
- Không nên lạm dụng khả năng quá tải làm cho chương trình khó hiểu
  - Đảm bảo rằng các phép toán vẫn giữ được mục đích nguyên thuỷ của nó
  - Nên chú thích vào những toán tử quá tải
- Nên quá tải toán tử = nếu lớp có liên quan đến cấp phát bộ nhớ động.

Giới hạn của quá tải toán tử

- Một số toán tử không thể quá tải được là:
  - **. .* :: ?: sizeof**
- Không thể thay đổi thứ tự ưu tiên và ngôi của toán tử
  - Toán tử * luôn được ưu tiên hơn toán tử +
  - Toán tử + luôn là 2 ngôi và toán tử ++ luôn là 1 ngôi
- Không thể tạo ra các kiểu toán tử mới.

Hàm bạn (Friend function)

- Hàm bạn (**friend** function)
  - là một hàm không phải là thành viên của lớp nhưng có quyền truy cập đến các thành viên private hoặc protected của một lớp.
  - Một hàm friend của một lớp được định nghĩa bên ngoài phạm vi của lớp đó.
- Cách khai báo một hàm là một friend của một lớp
  - friend <Tên hàm>;

```
class A
{
private:
// Khai báo các thuộc tính
public:
...
// Khai báo các hàm bạn của lớp A
friend void f1(...);
...
} ;
// Xây dựng các hàm f1, f2, f3
void f1(...) { ... }
```

Hàm bạn (Friend function)

- Ví dụ

```
#include <iostream.h>
class Count
{
friend void SetX(Count &, int); //Khai báo friend
public:
Count()//Constructor
 {
        X = 0;
 }
 void Print() const //Xuất
 {
    cout << X << endl;
 }
 private:
    int X;
};
```


Hàm bạn (Friend function)

```
//Có thể thay đổi dữ liệu private của lớp Count vì SetX() khai báo là
    một hàm friend của lớp Count
void SetX(Count &C, int Val)
{
    C.X = Val; //Hợp lệ: SetX() là một hàm friend của lớp Count
 }
 int main()
       {
        Count Object;
        cout << "Object sau khi khai bao: ";
        Object.Print();
        cout << "Object sau khi goi ham ban SetX ";
        SetX(Object, 8); //Thiết lập X với một friend
        Object.Print();
        return 0;
       }
```

Ôn lại

1. Quá tải hàm là gì ? Tại sao cần quá tải hàm ?
2. Quá tải toán tử là gì ? Tại sao cần quá tải tt ?
3. Toán tử += là toán tử mấy ngôi ?
4. Toán tử phủ định ! là toán tử mấy ngôi ?
5. Toán tử ?: là toán tử mấy ngôi ?
6. Có thể quá tải tất cả mọi toán tử không ?
7. Hàm bạn là gì?
8. Sự khác nhau giữa hàm bạn và hàm thông thường?

25

Bài tập về nhà

1. Xây dựng và cài đặt lớp phân số trên máy tính.
2. Viết chương trình nhập vào 10 phân số và sắp xếp theo thứ tự tăng dần.

26