

Nhập Môn Lập Trình Hàm Nâng Cao

TS. Tô Văn Khánh
Trường Đại học Công nghệ, ĐHQGHN

Nội Dung

- ▶ Nhắc lại Hàm – Cơ Bản
- ▶ Nạp chồng hàm
- ▶ Toán tử
- ▶ Nạp chồng toán tử
- ▶ Template

Hàm – Cơ Bản

- ▶ Khai báo hàm
`int bin(int, int) ;`
- ▶ Định nghĩa hàm
`int bin(int sol, int so2) {
// thân hàm miêu tả định nghĩa
// không khai báo lại sol, so2
}`
- ▶ Chứa ký hàm
`int bin(int, int)`
- ▶ Lời gọi hàm
`int a = bin(sol, so2) ;`
- ▶ Sử dụng tên hàm, tên tham số, không có kiểu

Truyền Tham Số

- ▶ Truyền giá trị

`int bin(int, int)`

- ▶ Tạo và truyền một bản sao > tốn chi phí
- ▶ Thay đổi bản sao, bản gốc ngoài hàm không đổi
- ▶ Xóa bản sao khi kết thúc hàm

- ▶ Truyền tham chiếu

`int bin(int&, int&)`

- ▶ Chính bản gốc được truyền (địa chỉ bản gốc)
- ▶ Thay đổi trong hàm, bản gốc thay đổi
- ▶ Không xóa bản gốc khi kết thúc hàm

Sử Dụng **const**

- ▶ Truyền tham chiếu

`int bin(int&, int&)`

- ▶ Thay đổi trong hàm, bản gốc thay đổi
- ▶ Không muốn hàm thay đổi, sử dụng từ khóa **const**
- ▶ Bảo vệ dữ liệu, không cho thay đổi, dữ liệu “chỉ-đọc”

`int bin(const int&, const int&)`

- ▶ Quy ước:

- ▶ Luôn truyền tham chiếu
- ▶ Dữ liệu **không** thay đổi, dùng **const**
- ▶ Dữ liệu thay đổi, **không** dùng **const**

Hàm Nạp Chồng (Function Overloading)

- ▶ Các hàm thực hiện công việc tương tự
- ▶ Trùng tên, nhưng khác nhau về tham số
 - ▶ Số lượng tham số
 - ▶ Kiểu tham số
 - ▶ Chữ ký hàm khác nhau
- ▶ Xác định hàm nào được gọi:
 - ▶ Sử dụng chữ ký của hàm
 - Số lượng tham số
 - Kiểu mỗi tham số

Hàm Nạp Chồng

- ▶ Khác số lượng tham số
- ▶ Hàm tính tổng các số truyền vào
 - ▶ Tính tổng 1 số nguyên
`int tong(int)`
 - ▶ Tính tổng 2 số nguyên
`int tong(int, int)`
 - ▶ Tính tổng 3 số nguyên
`int tong(int, int, int)`
 - ▶ Dựa trên số lượng tham số để gọi hàm tương ứng

Hàm Nạp Chồng

```
int tong(int so1) {  
    return so1;  
}  
int tong(int so1, int so2) {  
    return so1 + so2;  
}  
int tong(int so1, int so2, int so3) {  
    return so1 + so2 + so3;  
}  
  
int main() {  
    int tong1 = tong(1);  
    int tong2 = tong(1, 2);  
    int tong3 = tong(1, 2, 3);  
    return 0;  
}
```


Hàm Nạp Chồng

- ▶ Cùng số lượng tham số
- ▶ Khác kiểu tham số
- ▶ Hàm tính tổng 2 số truyền vào
 - ▶ Tính tổng 2 số nguyên
`int tong(int, int)`
 - ▶ Tính tổng 2 số thực
`double tong(double, double)`
 - ▶ Dựa trên kiểu tham số để gọi hàm tương ứng

Hàm Nạp Chồng

```
int tong(int so1, int so2) {  
    return so1 + so2;  
}  
  
double tong(double so1, double so2) {  
    return so1 + so2;  
}  
  
int main()  
{  
    int tong2int = tong(1, 2);  
    double tong2double = tong(1.0, 2.0);  
    double tong2mix1 = tong(1, 2.0);  
    double tong2mix2 = tong(1.0, 2);  
    return 0;  
}
```

Hàm Nạp Chồng

- ▶ Khi sử dụng hàm nạp chồng
 - ▶ Cùng số lượng tham số
 - ▶ Khác kiểu tham số
- ▶ Nên định nghĩa hàm cho tất cả các kiểu
- ▶ Không nên sử dụng ép kiểu
- ▶ Hàm tính tổng 2 số truyền vào

`int tong(int, int)`

`double tong(double, double)`

`double tong(int, double)`

`double tong(double, int)`

Hàm Nạp Chồng

- ▶ Tham số mặc định

```
int tong(int so1, int so2=0, int so3=0)
{
    return so1 + so2 + so3;
}
```

- ▶ Có thể gọi

```
tong(1, 2, 3)
tong(1, 2)
tong(1)
```

Hàm Nạp Chồng

- ▶ Tính tổng 3 số truyền vào (sử dụng tham số mặc định)

```
int tong(int so1, int so2=0, int so3=0)
{
    return so1 + so2 + so3;
}
```

- ▶ Tính tổng 2 số truyền vào

```
int tong(int so1, int so2)
{
    return so1 + so2;
}
```

- ▶ Hàm nào được gọi
`tong(1, 2)`

Hàm Nạp Chồng

- ▶ Trong C++, áp dụng hàm nạp chồng

- ▶ Hàm tính căn bậc hai **sqrt**

```
double sqrt(double arg);
```

```
double sqrt(IntegralType arg);
```

- ▶ Hàm tính mũ **pow**

```
double pow(double base, int iexp);
```

```
double pow(double base, double exp);
```

Toán Tử

- ▶ Toán tử thực chất là hàm
- ▶ Sử dụng toán tử giống lời gọi hàm
- ▶ Viết theo một cách khác
- ▶ Ví dụ: so sánh bằng
 - ▶ $1 == 2$ thực chất $== (1, 2)$

Toán Tử Nạp Chồng

- ▶ Giống như hàm, toán tử cũng có thể nạp chồng
- ▶ Định nghĩa toán tử cho kiểu dữ liệu mới
- ▶ Nạp chồng toán tử so sánh cho kiểu cấu trúc

```
struct ToaDo { double X, Y; };  
bool operator== (const ToaDo& td1,  
                 const ToaDo& td2 ) {  
    return (td1.X == td2.X  
           && td1.Y == td2.Y);  
}  
bool operator!= (const ToaDo& td1,  
                 const ToaDo& td2 );
```


Toán Tử Nạp Chồng

```
struct ToaDo { double X, Y; };
bool operator== (const ToaDo& td1,
                 const ToaDo& td2) {
    return (td1.X == td2.X && td1.Y == td2.Y);
}
bool operator!= (const ToaDo& td1,
                 const ToaDo& td2) {
    return !(td1 == td2);
}
int main()
{
    ToaDo td1 , td2 ;
    if (td1 == td2) ...
    if (td1 != td2) ...
    return 0;
}
```

Toán Tử Nạp Chồng

```
struct PhanSo { int tu_so, mau_so; };

PhanSo operator+ (const PhanSo&, const PhanSo&);
PhanSo operator+ (const PhanSo&, int);
PhanSo operator+ (int, const PhanSo&);

int main()
{
    PhanSo ps1, ps2, psTong;

    psTong = ps1 + ps2;
    psTong = 1 + ps2;
    psTong = ps1 + 1;

    return 0;
}
```

Toán Tử Nạp Chồng

```
struct PhanSo { int tu_so, mau_so; };

ostream& operator<< (ostream& outStream,
                    const PS& ps ) {
    outStream << ps.tu_so << "/" << ps.mau_so;
    return outStream;
}

int main()
{
    PhanSo ps1, ps2, psTong1, psTong2;
    psTong1 = ps1 + 1;
    psTong2 = 1 + ps2;
    cout << psTong1 << psTong2;
    return 0;
}
```

Toán Tử Nạp Chồng

- ▶ Lớp **string** cài đặt toán tử nạp chồng

operator+

operator==

operator!=

operator<

operator>

operator<=

operator>=

Template

```
int getMax(int so1, int so2)
{
    if (so1 < so2) return so2;
    return so1;
}

double getMax(double so1, double so2)
{
    if (so1 < so2) return so2;
    return so1;
}

PhanSo getMax(PhanSo so1, PhanSo so2)
{
    if (so1 < so2) return so2;
    return so1;
}
```

Template

```
template <class T>
T getMax(T so1, T so2)
{
    if ( so1 < so2 ) return so2;
    return so1;
}

int main() {
    int int1, int2, intMax;
    intMax = getMax(int1, int2);
    double double1, double2, doubleMax;
    doubleMax = getMax(double1, double2);
    PhanSo ps1, ps2, psMax;
    psMax = getMax(ps1, ps2);
    return 0;
}
```

Template

- ▶ Thư viện **algorithm** hàm **sort**
 - ▶ Sắp xếp mảng của bất kỳ kiểu dữ liệu nào
 - ▶ Toán tử so sánh phải được định nghĩa