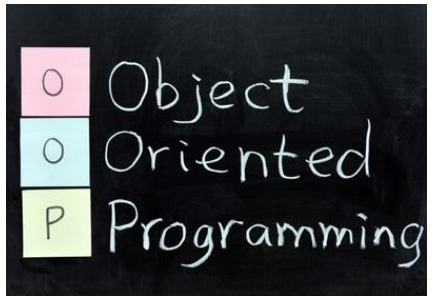



Môn học

Lập trình hướng đối tượng

Đỗ Phúc Hảo

Khoa Công nghệ thông tin




Nội dung

❖Chương 2: Các mở rộng trong C++

- Toán tử nhập xuất
- Toán tử phạm vi
- Tham chiếu
- Chồng hàm
- Chồng toán tử
- Hàm có đối số mặc định
- Các toán tử quản lý bộ nhớ động
- Bài tập




1. Toán tử nhập xuất

❖Đọc dữ liệu từ thiết bị vào chuẩn (bàn phím)

- Cú pháp

  cin>> biến 1 >> … >> biến n;

❖Ghi dữ liệu lên thiết bị chuẩn (màn hình)

- Cú pháp

  cout<< biến 1 << … << biến n;




1. Toán tử nhập xuất (t)

- cin: Đối tượng tương ứng thiết bị vào chuẩn (bàn phím)
- cout: đối tượng tương ứng thiết bị ra chuẩn (màn hình)
- >>: Toán tử nhập
- <<: Toán tử xuất




1. Toán tử nhập xuất (t) – Ví dụ

```
#include "iostream”
using namespace std;
int main(){
        int a, b;
        cout<<"Nhap so a va b: ";
        cin>>a>>b;
        cout<<"So vua nhap la: "<< a << " " << b<<endl;
        return 0;
}
```




2. Toán tử phạm vi

- Bình thường biến *cục bộ* sẽ che lấp biến *toàn cục*
- Trong những trường hợp cần thiết, khi muốn truy xuất đến biến toàn cục ta phải sử dụng toán tử phạm vi :: đặt trước tên biến.

```
int x  = 10;
int main() {
        int x = 20;
        cout<<x<<endl;
        cout<<::x<<endl;
}
```




3. Tham chiếu

❖Biến tham chiếu

- C++ cho phép khai báo một biến tham chiếu dùng để tham chiếu đến một biến đang tồn tại trong bộ nhớ
- Mọi thay đổi trên biến tham chiếu đều làm thay đổi chính biến được tham chiếu. Do đó có thể coi biến tham chiếu như là bí danh khác của một biến.




3. Tham chiếu (t)

```
int main(){
        int n = 10;
        int &r = n;
        r += 10;
        cout<<n<<" "<<r<<endl;
        n += 10;
        cout<<n<<" "<<r<<endl;
}
```




3. Tham chiếu (t)

❖Hằng tham chiếu

- Hằng tham chiếu cũng có thể tham chiếu đến một biến. Tuy nhiên, không cho phép dùng hằng tham chiếu để làm thay đổi giá trị của vùng nhớ mà nó tham chiếu.
- Chú ý: Nếu dùng khái niệm hằng tham chiếu trong khai báo đối số của hàm thì sẽ cho phép hàm sử dụng giá trị của các tham số trong lời gọi hàm nhưng tránh làm thay đổi giá trị các tham số.




3. Tham chiếu (t)

```
int main() {
        int n = 10;
        const int &r = n;
        cout<<n<<" "<<r<<endl;
        n += 10;
        cout<<n<<" "<<r<<endl;
        r += 10; ///????
}
```




3. Tham chiếu (t)

❖Truyền tham số cho hàm bằng tham chiếu

- Trong C++, việc dùng khái niệm tham chiếu trong khai báo đối số của hàm sẽ cho phép hàm thao tác trực tiếp trên vùng nhớ các tham số thực trong lời gọi hàm → Dễ dàng thay đổi giá trị các tham số này khi cần.




3. Tham chiếu (t)

```
void hv1(int a, int b) {
        int t = a; a = b; b = t;
}
void hv2(int *a, int *b) {  // sử dụng biến con trỏ sẽ được đề cập sau
        int t = *a; *a = *b, *b = t;
}
void hv3(int &a, int &b){ // sử dụng biến tham chiếu
        int t = a, a = b, b = t;
}
```




3. Tham chiếu (t)

```
int main(){
        int a = 3, b = 4;
        hv1(a,b); cout<<a<<" "<<b<<endl;
        hv2(&a, &b); cout<<a<<" "<<b<<endl;
        hv3(a,b); cout<<a<<" "<<b<<endl;
        return 0;
}
```




3. Tham chiếu (t)

❖Hàm trả về một tham chiếu

- Hàm cũng có thể trả về một tham chiếu. Khi đó, có thể dùng hàm để truy nhập đến một biến nào đó.

❖Lưu ý:

- Chỉ có thể trả về một tham chiếu đến một biến xác định từ bên ngoài hàm vì khi đó mới có thể sử dụng được giá trị của hàm.




3. Tham chiếu (t)

```
int x;
int &get() {
        return x;
}
int main() {
        get() = 10; // tương đương với việc gán x = 10;
        cout<<x<<endl;
}
```




4. Chồng hàm

- C++ cho phép sử dụng một tên cho nhiều hàm khác nhau, ta gọi đó là sự chồng hàm.
- Các hàm phải có danh sách các đối số khác nhau (về số lượng hoặc kiểu dữ liệu)




4. Chồng hàm (t)

Một vài ví dụ về chồng hàm:

int min(int a, int b); // Số nguyên

int min(int a, int b, int c); // Số nguyên

float min(float a, float b); // Số thực

char min(char a, char b); // Ký tự

int min(int t[], int n); // Số nguyên




4. Chồng hàm (t)

```
int min(int a, int b){ return a < b ? a : b); }
int min(int a, int b, int c){return min(min(a,b), c); }
float min(float a, float b){ return a < b ? a : b); }
char min(char a, char b) { return a < b ? a : b); }
int min(int t[], int n){
        int res = t[0];
        for(int i=1;i<n;i++) res = min(res, t[i]);
        return res;
}
```




4. Chồng hàm (t)

```
int main(){
        int u = 1, v = 2, r = 3;
        double x = 23.33, y = 13.33;
        char c1 = ‘a’, c2 = ‘b’;
        int a[3] = {3,4,1};
        cout<<min(u,v)<<" " <<min(u,v,r)<<endl;
        cout<<min(c1,c2)<<" "<<min(x,y)<<endl;
        cout<<min(a, 3)<<endl;
}
```




5. Chồng toán tử

❖ C++ cho phép định nghĩa các toán tử mới trên cơ sở các ký hiệu toán tử đã có (Chẳng hạn như +, -, *, /, +=, -=, *=, /=, >, >=, <, <=, ==, …) Việc này được gọi là định nghĩa chồng toán tử.

❖ Việc định nghĩa chồng toán tử giống như định nghĩa các hàm, chỉ khác đây là hàm toán tử, hàm toán tử có tên được ghép bởi từ khóa **operator** và phép toán tương ứng.




5. Chồng toán tử (t)

```
struct PS {
        int ts, ms;
};
void Nhap(PS &u);
void Xuat(PS u);
int UCLN(int x, int y);
PS operator+(PS u, PS v);
PS operator-(PS u, PS v);
PS operator*(PS u, PS v);
PS operator/(PS u, PS v);
```




5. Chồng toán tử (t)

```
void Nhap(PS &u){
        cout<<"Nhap tuso: "; cin>>u.ts;
        cout<<"Nhap mauso: "; cin>>u.ms;
}
void Xuat(PS u){
        cout<<u.ts<<"/"<<u.ms<<endl;
}
int UCLN(int x, int y){
        if(y == 0) return x;
        return UCLN(y, x % y);
}
```




5. Chồng toán tử (t)

```
void RutGon(PS &u){
        int t = UCLN(u.ts, u.ms);
        u.ts/=t; u.ms /= t;
}
PS operator+(PS u, PS v){
        PS res;
        res.ts = u.ts * v.ms + v.ts * u.ms;
        res.ms = u.ms * v.ms;
        RutGon(res);
        return res;
}
```

```
PS operator-(PS u, PS v){
        PS res;
        res.ts = u.ts * v.ms - v.ts * u.ms;
        res.ms = u.ms * v.ms;
        RutGon(res);
        return res;
}
```




5. Chồng toán tử (t)

```
PS operator*(PS u, PS v){
        PS res;
        res.ts = u.ts * v.ts;
        res.ms = u.ms * v.ms;
        RutGon(res);
        return res;
}
```

```
PS operator/(PS u, PS v){
        PS res;
        res.ts = u.ts * v.ms;
        res.ms = u.ms * v.ts;
        RutGon(res);
        return res;
}
```




5. Chồng toán tử (t)

```
int main(){
        PS a, b, c, d;
        Nhap(a);
        Nhap(b);
        Nhap(c);
        d = (a - b) * c / (a + b);
        Xuat(d);
        return 0;
}
```




6. Hàm có đối số mặc định

```
void f(int a, int b = 0);
void f(int a, int b){
        cout<<a<<" "<<b<<endl;
}
int main(){
        int u = 10, v = 20;
        f(u,v);
        f(u);
}
```




6. Hàm có đối số mặc định (t)

❖Chú ý:

- Các đối số mặc định phải được đặt ở cuối trong danh sách các đối số của hàm
- Nếu đã sử dụng nguyên mẫu hàm thì các đối tượng mặc định phải được khởi gán trong khai báo nguyên mẫu hàm, không phải trong định nghĩa hàm.




7. Các toán tử quản lý bộ nhớ động

❖Toán tử **new**

- Dạng 1: **new** <type>;

Cấp phát bộ nhớ động cho một biến kiểu type và trả về một con trỏ đến bộ nhớ này nếu như cấp phát thành công, ngược lại nếu thất bại sẽ trả về NULL.

- Dạng 2: **new** <type>[<size>];

Cấp phát bộ nhớ động cho size phần tử type và trả về một con trỏ đến đầu bộ nhớ này nếu như cấp phát thành công, ngược lại nếu thất bại sẽ trả về NULL.




7. Các toán tử quản lý bộ nhớ động (t)

❖Toán tử delete

- Dạng 1: delete <pointer>;

Giải phóng bộ nhớ động chỉ chứa một biến.

- Dạng 2: delete[]<pointer>;

Giải phóng bộ nhớ động chứa một dãy các biến.

Pointer là bên biến con trỏ đang trỏ tới bộ nhớ động cần giải phóng.




7. Các toán tử quản lý bộ nhớ động (t)

```
void Nhap(int *p, int n);
void Xuat(int *p, int n);
void Nhap(int *p, int n){
        for(int i=0;i<n;i++) cin>>p[i];
}
void Xuat(int *p, int n){
        for(int i=0;i<n;i++){
                cout<<p[i]<<" ";
        }
}
```

```
int *p, n;
cin>>n;
p = new int[n];
Nhap(p,n);
Xuat(p,n);
delete []p;
p = NULL;
```




7. Các toán tử quản lý bộ nhớ động (t)

```
void Nhap2(int **p, int m, int n){
        for(int i=0;i<m;i++){
                for(int j=0;j<n;j++)
                        cin>>p[i][j];
        }
}
```

```
void Xuat2(int **p, int m, int n){
        for(int i=0;i<m;i++){
                for(int j=0;j<n;j++){
                        cout<<p[i][j]<<" “;
                }
                cout<<endl;
        }
}
```




7. Các toán tử quản lý bộ nhớ động (t)

```
int sd, sc;
cin>>sd>>sc;
int **arr;
arr = new int *[sd];
for(int i=0;i<sd;i++) arr[i] = new int[sc];
Nhap2(arr,sd,sc);
Xuat2(arr,sd,sc);

for(int i=0;i<sd;i++)
        delete [] arr[i];
delete []arr;
arr = NULL;
```




8. Bài tập

❖Sử dụng C++ để viết các chương trình sau đây:

1. Nhập vào 2 số a và b, in ra màn hình là “yes” nếu a chia hết cho b, ngược lại thì in ra màn hình “no”.
2. Nhập vào một chuỗi s, in ra màn hình là “yes” nếu chuỗi đó có chứa ít nhất một ký tự HOA và ngược lại thì in ra “no”
3. Nhập vào một chuỗi s, in ra màn hình “yes” nếu chuỗi đó chứa ít nhất 2 ký tự HOA và ngược lại thì in ra “no”
4. Nhập vào các giá trị a, b và c. Hãy giải phương trình bậc 2:

$$ax2 + bx + c = 0;$$

5. Giải bài toán vừa gà vừa chó (cổ điển), thay số lượng tổng số con là N và tổng số lượng chân là K. Nếu không tính được thì in ra màn hình là “no”




8. Bài tập (t)

❖Sử dụng C++ để viết các chương trình sau đây:

1. Sử dụng chồng toán tử, xây dựng bài toán thao tác tính cộng, trừ, nhân, chia hai phân số bất kỳ
2. Sử dụng chồng toán tử, xây dựng bài toán thao tác tính cộng, trừ, nhân, chia hai số phức bất kỳ
3. Thao tác mảng một chiều với con trỏ
4. Thao tác mảng hai chiều với con trỏ




Thank you!

Any questions?