



# LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

## Chương 1: Các đặc điểm mới của C++ (p2)

Trình bày: ThS. Lê Thanh Trọng



## 8. Chuyển kiểu dữ liệu

- ❖ Chuyển từ kiểu này sang kiểu khác
- ❖ 2 dạng
  - ❑ Widening (Nới rộng): Chuyển từ kiểu “bé hơn” sang “lớn hơn”
    - short -> int -> long -> float -> double
  - ❑ Narrowwing(Thu hẹp): Chuyển từ kiểu “lớn hơn” sang “bé hơn”
    - double -> float -> long -> int -> short



# Chuyển kiểu nối rộng

- ❖ Được thực hiện tự động bởi compiler
- ❖ Chuyển từ kiểu “bé hơn” sang “lớn hơn”
- ❖ VD: **float f = 10L;**  
**double d = f;**



# Chuyển kiểu thu hẹp

- ❖ Thực hiện bởi lập trình viên
- ❖ Có thể dẫn đến “thất thoát” dữ liệu
- ❖ VD:

```
float PI = 3.14;  
int y = (int)PI; //y=3
```



# Các loại chuyển đổi khác trong C++

- ❖ **static\_cast**: thường được dùng chuyển kiểu số nguyên sang ký tự, có kiểm tra lúc biên dịch
- ❖ **const\_cast**: có thể thay đổi giá trị các thuộc tính thành phần trong phương thức có khai báo const
- ❖ **dynamic\_cast**: ép con trỏ kiểu cơ sở về con trỏ kiểu dẫn xuất
- ❖ **reinterpret\_cast**: ép các con có thuộc các kiểu hoàn toàn khác nhau



## 9. Không gian tên (namespace)

- ❖ Định nghĩa một vùng không gian **có tên** của riêng mình thông qua từ khóa namespace
- ❖ Nhằm chứa source code, tài nguyên (gọi là tài liệu)
- ❖ Giúp giải quyết vấn đề trùng tên giữa các **tài liệu**
- ❖ Thường được viết thường (std)
- ❖ Phân loại: do người dùng định nghĩa (mới) hoặc do C++ cung cấp (có sẵn)



# Không gian tên (namespace)

## ❖ Truy cập

- ❑ Thông qua toán tử giải phạm vi ( :: )
- ❑ Ví dụ: std::vector

## ❖ Ví dụ:

namespace **my\_namespace**

```
{  
    void print() //  
    {  
        std::cout << "Hello World";  
    }  
}
```



# Ví dụ không gian tên (namespace)

```
void print()
{
    std::cout << "print from the
global";
}
namespace my_namespace
{
    void print()
    {
        std::cout << "print from
my_namespace";
    }
}
```

```
int main()
{
    my_namespace::print();
    ::print();
    return 0;
}
```



# 10. Quản lý bộ nhớ động

- ❖ Toán tử cấp phát bộ nhớ động **new**

```
int *x;
```

```
x = new int;           //x = (int*)malloc(sizeof(int));
```

```
char *y;
```

```
y = new char[100];    //y = (char*)malloc(100);
```



# Quản lý bộ nhớ động

- ❖ Toán tử giải phóng vùng nhớ động **delete**

```
delete x;           // free(x);
```

```
delete []y;        // free(y);
```

- ❖ Cấp phát với **new** phải giải phóng bằng **delete**



# 11. Truyền tham số

## ❖ Truyền theo giá trị (tham trị)

- Giá trị tham số khi ra khỏi hàm **sẽ không thay đổi**

## ❖ Truyền theo địa chỉ (tham chiếu)

- Giá trị tham số khi ra khỏi hàm **có thể thay đổi**



# Tham chiếu

- ❖ Tham chiếu là địa chỉ vùng nhớ được cấp phát cho một biến
- ❖ Ký hiệu **&** đặt trước biến hoặc hàm để xác định tham chiếu của chúng
- ❖ Ví dụ 1:
  - `int x = 10, *px = &x, &y = x;`
  - `*px = 20;`
  - `y = 30;`



# Tham chiếu

## ❖ Ví dụ 2:

- `int arrget(int *a, int i) { return a[i]; }`
- `arrget(a, 1) = 1; // a[1] = 1;`
- `cin >> arrget(a,1); // cin >> a[1];`

## ❖ Ví dụ 3:

- `void swap1(int x, int y) { int t = x; x = y; y = t; }`
- `void swap2(int *x, int *y) { int *t = x; x = y; y = t; }`
- `void swap3(int &x, int &y) { int t = x; x = y; y = t; }`



## 12. Hàm nội tuyến (inline)

- ❖ Hàm inline hay còn gọi là hàm nội tuyến
- ❖ Sử dụng từ khóa **inline** trước khai báo hàm
- ❖ Dùng cho các hàm thực thi thường xuyên
- ❖ Yêu cầu trình biên dịch **copy code vào trong chương trình** thay vì thực hiện lời gọi hàm
- ❖ Tốt hơn so với sử dụng macro (Bjarne Stroustrup)



# Hàm nội tuyến (inline)

## ❖ Lợi ích:

- Giảm thời gian thực thi chương trình
- Giảm không gian bộ nhớ mà các hàm nhỏ chiếm chỗ khi thực hiện thường xuyên

## ❖ Chỉ nên định nghĩa inline khi hàm có kích thước nhỏ

❖ Trình biên dịch quyết định một hàm nội tuyến có thể thực thi như là một hàm nội tuyến hay không



# Hàm nội tuyến (inline)

❖ Ví dụ

```
inline float sqr(float x) {  
    return (x*x);  
}
```

$S = \text{sqr}(a); // S = a * a;$

```
inline int Max(int a, int b) {  
    return ((a>b) ? a : b);  
}
```

$G = \text{Max}(x, y); // G = x > y ? x : y;$



# Tóm tắt bài học về các đặc điểm mới của C++

## ❖ Lưu ý về phong cách lập trình

- ❑ Đặt tên
- ❑ Viết hoa
- ❑ Cú pháp, từ loại
- ❑ Khai báo biến, hằng
- ❑ Viết câu lệnh
- ❑ Sử dụng chú thích

## ❖ C++ là một ngôn ngữ biên dịch, lập trình đa năng bậc cao (high-level), như một phần mở rộng của ngôn ngữ lập trình C, hoặc **C với các class**, hỗ trợ:

- ❑ Lập trình thủ tục
- ❑ Lập trình tổng quát
- ❑ **Lập trình hướng đối tượng**



# Tóm tắt bài học về các đặc điểm mới của C++

## ❖ Quá trình thực thi chương trình C++:

1. Editing
2. Preprocessing
3. Compilation
4. Linking
5. Loading
6. Execution



# Tóm tắt bài học về các đặc điểm mới của C++

## ❖ Một số điểm mới của C++

1. Chú thích
2. Khai báo hằng
3. Phạm vi và khai báo (::)
4. Nhập xuất
5. Các kiểu dữ liệu
6. Tham số mặc nhiên



# Tóm tắt bài học về các đặc điểm mới của C++

## ❖ Một số điểm mới của C++

7. Tái định nghĩa hàm
8. Chuyển kiểu
9. Không gian tên (namespace)
10. Quản lý bộ nhớ
11. Truyền tham số
12. Hàm nội tuyến (inline)
- 13....