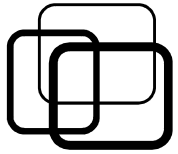
Giới thiệu môn học

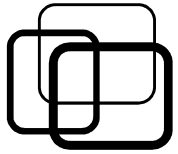
GV. Nguyễn Minh Huy

Nội dung



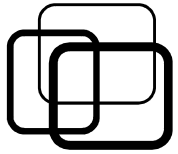
- Chuẩn và quy ước lập trình.
- Nạp chồng hàm.
- Lập trình tổng quát.

Nội dung



- **Chuẩn và quy ước lập trình.**
- Nạp chồng hàm.
- Lập trình tổng quát.

Chuẩn và quy ước lập trình



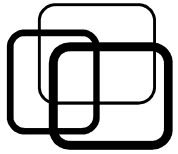
- Vì sao phải có chuẩn và quy ước?
  - Làm việc một mình:
    - Tự làm tự hiểu.
    - **Mình luôn hiểu mình?**

  - Làm việc nhóm:
    - Mỗi người một việc.
    - Ráp nối công việc.
    - **Mọi người luôn hiểu nhau?**

Để phối hợp công việc hiệu quả

Phải áp đặt kỷ luật!!
Phải có chuẩn!!

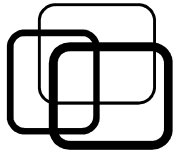
Chuẩn và quy ước lập trình



- **Quy ước #0: Không có chuẩn chung!!**
  - Tùy thuộc ngôn ngữ lập trình.
  - Tùy thuộc doanh nghiệp.
  - Tùy thuộc cộng đồng.
- Một vài quy ước thông dụng:
  - Quy ước đặt tên (naming convention).
  - Quy ước viết câu lệnh (statement convention).
  - Quy ước viết chú thích (comment convention).

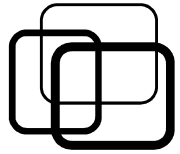
  (Xem Coding Convention Guidelines)

Nội dung



- Chuẩn và quy ước lập trình.
- **Nạp chồng hàm.**
- Lập trình tổng quát.

Nạp chồng hàm



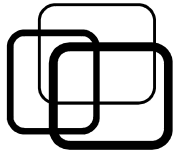
- Chữ ký hàm (Function Signature/Prototype):
  - Định danh một hàm.
  - Thành phần của chữ ký:
    - Tên hàm.
    - Danh sách tham số.

      double sapXep(***int danhSach[ ], int kichThuoc***);

      double sapXep(***float danhSach[ ], int kichThuoc***);
    - **Giá trị trả về không thuộc chữ ký hàm!!**

Chương trình có thể có nhiều hàm cùng tên!!

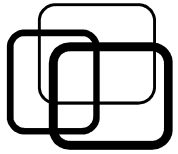
Nạp chồng hàm



- Những trường hợp nạp chồng hàm nào không hợp lệ?

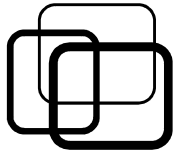
1. int tinhToan(int a, int b);
2. int tinhToan(int x, int y);
3. int tinhToan(int a, float b);
4. float tinhToan(int u, int v);
5. int tinhToan(int a, long b);

Nội dung



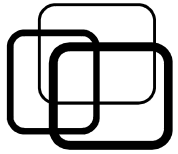
- Chuẩn và quy ước lập trình.
- Nạp chồng hàm.
- **Lập trình tổng quát.**

Lập trình tổng quát



- Mức độ tổng quát của chương trình:
  - Khả năng hoạt động trong nhiều ngữ cảnh.
  - Không cố định vào một trường hợp.
  - Write once, use everywhere.
  - Dùng phương pháp tham số hóa:
    - Tham số hóa dữ liệu (truyền tham số thông thường).
    - Tham số hóa kiểu (function template).
    - Tham số hóa xử lý (con trỏ hàm).

Lập trình tổng quát

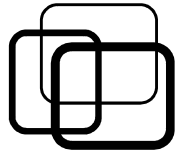


- Tham số hóa dữ liệu:
  - Bài toán: tính tổng hai số nguyên 3 và 5.
  - Tổng quát 1: tính tổng hai số nguyên bất kỳ.

```
// Giải pháp không tổng quát:
// Không dùng tham số.
int tinhToan( )
{
        return 3 + 5;
}
```

```
// Giải pháp tổng quát 1:
// Tham số hóa dữ liệu.
int tinhToan( int x, int y )
{
        return x + y;
}
void main( )
{
        tinhToan( 3, 5 );
        tinhToan( 4, 6 );
        tinhToan( -2, 7 );
}
```

Lập trình tổng quát

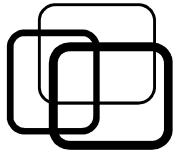


- Tham số hóa kiểu:
  - Tổng quát 2: tính tổng hai số kiểu bất kỳ.
    - ➔ Dùng Function Template.

```
// Giải pháp tổng quát 2:
// Tham số hóa kiểu.
template <class T>
T  tinhToan( T  x, T  y )
{
        return x + y;
}
```

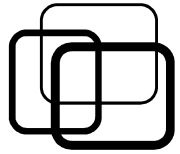
```
void main( )
{
        int x = tinhToan( 3, 5 );

        float y = tinhToan( 4.2, 6.3 );

        PhanSo  p1, p2;
        PhanSo  p3 = tinhToan( p1, p2 );
}
```

Lập trình tổng quát



- Đặc điểm của Function Template:
  - Tham số hóa kiểu dữ liệu.
  - "Khuông hàm" tổng quát cho nhiều kiểu dữ liệu.
  - "Đúc" thành từng hàm với kiểu cụ thể khi gọi hàm.
- Ghi chú:
  - Từ khóa **"class"** có thể thay bằng **"typename"**.
  - Phần khai báo và cài đặt hàm đều phải có template.
  - Phần cài đặt hàm phải nằm cùng file:
    - Hoặc phần khai báo hàm.
    - Hoặc phần gọi sử dụng hàm.

Lập trình tổng quát

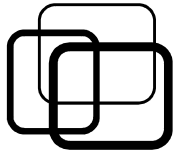


- Tham số hóa xử lý:
  - Tổng quát 3: tính toán bất kỳ giữa hai số nguyên.
    - ➔ Dùng Con trỏ hàm.

```
// Giải pháp tổng quát 3:
// Tham số hóa xử lý.
typedef int ( * PhepTinh )( int, int );
int  tinhToan( int  x, int  y, PhepTinh p )
{
    return p( x, y );
}
```

```
int cong( int x, int y ) {
        return x + y;
}
int nhan( int x, int y ) {
        return x * y;
}
void main( )
{
        int x = tinhToan( 3, 5, cong );
        int x = tinhToan( 4, 6, nhan );
}
```

Lập trình tổng quát



- Cách sử dụng con trỏ hàm:
  - B1: tạo kiểu hàm, dùng **typedef**.

    typedef int **(* PhepTinh) (** int, int **);**
  - B2: tạo hàm tổng quát có đối số là kiểu hàm đã tạo.

```
int tinh( int x, int y, PhepTinh p ) {
    return  p( x, y );
}
```

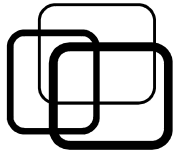
  - B3: tạo hàm xử lý cụ thể.

```
int cong( int x, int y ) {
   return x + y;
}
```

  - B4: gọi hàm tổng quát, truyền hàm xử lý cụ thể vào:

    tinh(3, 5, **cong**);

Lập trình tổng quát

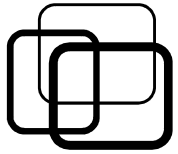


- Đặc điểm của con trỏ hàm:
  - Tham số hóa xử lý.
  - Hàm làm đối số của hàm khác.
  - Hàm có xử lý tổng quát tùy xử lý truyền vào.
  - Xử lý cụ thể được quyết định khi gọi hàm.
- Ghi chú:
  - Dùng con trỏ hàm trực tiếp, không dùng **typedef**:

    int **tinhToan**(int x, int y, **int (*p) (int, int)**);
  - Sử dụng Function Template phải dùng con trỏ hàm trực tiếp.

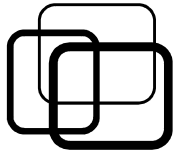
    ➔ Tổng quát 4: tính toán bất kỳ giữa hai số kiểu bất kỳ.

Tóm tắt



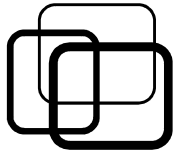
- Chuẩn và quy ước lập trình:
  - Áp đặt kỷ luật lên việc lập trình.
  - Tạo quy ước để phối hợp hiệu quả.
  - Quy ước đặt tên: có nghĩa, ngắn gọn.
  - Quy ước viết câu lệnh: rộng rãi, rõ ràng.
  - Quy ước viết chú thích: đầy đủ, dễ hiểu.
- Nạp chồng hàm:
  - Nhiều hàm cùng tên, khác tham số.

Tóm tắt



- Lập trình tổng quát:
  - Chương trình hoạt động trong nhiều ngữ cảnh.
  - Dùng phương pháp tham số hóa:
    - Tham số hóa dữ liệu (tham số thông thường).
    - Tham số hóa kiểu (function template).
    - Tham số hóa xử lý (con trỏ hàm).

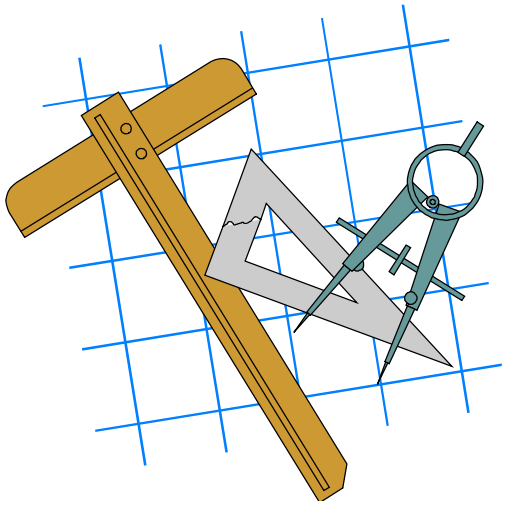
Bài tập



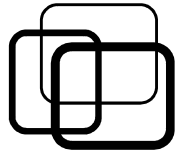
- **Bài tập 1.1:**

  Viết chương trình cho phép thực hiện các thao tác trên kiểu **phân số**:

  - Nhập, xuất phân số.
  - Rút gọn phân số.
  - Cộng, trừ, nhân, chia, so sánh hai phân số.

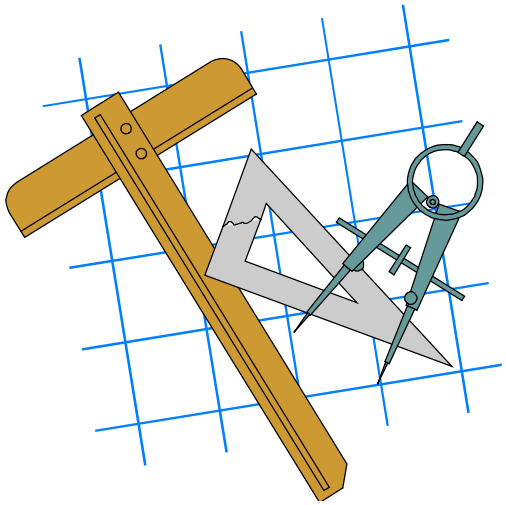


Bài tập

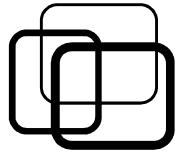


- ## Bài tập 1.2:

  Viết chương trình cho phép sắp xếp mảng **phân số** theo thứ tự tăng/giảm/điều kiện bất kỳ.



Bài tập



- Bài tập 1.3:

  Mở rông bài 1.2, sắp xếp mảng **phân số/số nguyên/kiểu bất kỳ.**