

# Chương 5: Cài đặt Phần mềm



**GVLT: ThS Nguyễn Thị Thanh Trúc**



# Nội dung

- ❖ Giới thiệu
- ❖ Kỹ năng lập trình
- ❖ Phương pháp
  - Lập trình tuyến tính
  - Lập trình hướng cấu trúc
  - Lập trình hướng đối tượng
- ❖ Kiến trúc mô hình 1 lớp, 2 lớp, 3 lớp
- ❖ Sử dụng công cụ Visual Source Safe để tổ chức, quản lý, chia sẻ source code.



# Giới thiệu

## ❖ Cài đặt?

- Là quá trình chuyển đổi từ **thiết kế** chi tiết sang **mã lệnh**.

## ❖ Lựa chọn **ngôn ngữ lập trình**:

- Phụ thuộc vào cấu hình máy
- Phụ thuộc vào số lượng ngôn ngữ lập trình sẵn có
- Phụ thuộc vào thói quen sử dụng ngôn ngữ lập trình
- Phụ thuộc vào khách hàng
- ...

## ❖ **Đánh giá rủi ro** khi chọn ngôn ngữ lập trình



# Kỹ năng lập trình

- ❖ Hiểu rõ ngôn ngữ (language-specific)
- ❖ Sử dụng tên biến thích hợp và có nghĩa
  - Tên biến phải rõ ràng, tránh nhầm lẫn
  - Có nghĩa theo quan điểm của các nhà lập trình
  - Thống nhất ngôn ngữ để đặt tên biến
  - Dễ dàng hiểu các mã lệnh thông qua tên biến
- ❖ Chú thích tự thân nghĩa là tên biến phải được diễn giải ngay từ đầu, về sau không cần chú thích thêm
- ❖ Nên có các chú thích bên trong mô-đun
- ❖ Mã lệnh chuẩn
  - Thống nhất về cách đặt tên Mô-đun, tên hàm, tên biến,...
- ❖ Mã lệnh dễ đọc, sử dụng các cặp dấu ngoặc, canh đầu dòng
- ❖ Nên có các dòng trống để phân biệt các công việc.
- ❖ Khả năng tái sử dụng



# Kỹ năng lập trình

- ❖ Thông tin tối thiểu của một mô-đun:
  - Tên mô-đun
  - Mô tả vắn tắt các công việc mô-đun phải thực hiện
  - Tên lập trình viên
  - Ngày viết
  - Ngày chỉnh sửa
  - Danh sách các tham số
  - Danh sách các biến
  - ...



# Kỹ năng lập trình

- Ngày module được chấp thuận, chấp thuận bởi ai
- Các tham số
- Danh sách các tên biến (nên theo thứ tự chữ cái), cách sử dụng
- Tên các tập tin mà module có truy xuất
- Tên các tập tin bị thay đổi bởi modul
- Nhập xuất của module
- Các khả năng lỗi xảy ra
- Tên tập tin sẽ được sử dụng để kiểm thử
- Các lỗi đã biết
- Danh sách các cập nhật đã được thực hiện với ngày tương ứng, người thực hiện.



# PHƯƠNG PHÁP LẬP TRÌNH

- ❖ Lập trình tuần tự (tuyến tính)
- ❖ Lập trình có cấu trúc (thủ tục)
- ❖ Lập trình hướng chức năng
- ❖ Lập trình hướng đối tượng
- ❖ Lập trình Logic



# Lập trình tuyến tính

- ❖ Khi các phần mềm còn rất đơn giản:
  - Chương trình được viết tuần tự với các câu lệnh thực hiện từ đầu đến cuối.
- ❖ Tuy nhiên:
  - Khoa học máy tính ngày càng phát triển.
  - Các phần mềm đòi hỏi ngày càng phức tạp và lớn hơn rất nhiều.
- ❖ Phương pháp lập trình tuyến tính kém hiệu quả ?





# Lập trình cấu trúc

- ❖ Phương pháp lập trình **thủ tục** hay lập trình **cấu trúc**
  - Hệ thống chia các chức năng (hàm) thành các chức năng nhỏ hơn.
  - Chương trình được tổ chức thành các chương trình con
  - Chương trình = Cấu trúc dữ liệu + giải thuật
- ❖ Tổ chức dữ liệu như thế nào?
- ❖ Khi thay đổi cấu trúc dữ liệu?



# Lập trình Hướng đối tượng

- ❖ Lập trình hướng đối tượng – Lập trình định hướng đối tượng - OOP
  - Là phương pháp lập trình lấy đối tượng làm nền tảng để xây dựng thuật giải, xây dựng chương trình.
  - Dữ liệu + Hành vi của dữ liệu = Đối tượng
- ❖ Cách tiếp cận gần gũi và thực tế



# LỰA CHỌN NGÔN NGỮ LẬP TRÌNH

*Dựa vào:*

- ❖ Đặc trưng của ngôn ngữ
- ❖ Miền ứng dụng của ngôn ngữ
- ❖ Năng lực, kinh nghiệm của nhóm phát triển
- ❖ Yêu cầu của khách hàng



# NGÔN NGỮ LẬP TRÌNH - Đặc trưng

- ❑ năng lực (kiểu biến, các cấu trúc)
- ❑ tính khả chuyển
- ❑ mức độ hỗ trợ của các công cụ



# NGÔN NGỮ LẬP TRÌNH - Đặc trưng

- Năng lực của ngôn ngữ
  - Có cấu trúc, câu lệnh phong phú
  - Hỗ trợ nhiều kiểu dữ liệu
  - Hỗ trợ con trỏ, đệ quy
  - Hỗ trợ hướng đối tượng
  - Thư viện phong phú



# NGÔN NGỮ LẬP TRÌNH - Đặc trưng

## □ Tính khả chuyển

- thay đổi phần cứng
- thay đổi OS

Ví dụ: C, Java là các ngôn ngữ khả chuyển



# NGÔN NGỮ LẬP TRÌNH - Đặc trưng

## □ Hỗ trợ của công cụ

*editor, debugger, linker, make...*

- biên dịch tốc độ cao
- khả năng tối ưu cao
- khai thác các tập lệnh, kiến trúc phần cứng mới



# NGÔN NGỮ LẬP TRÌNH - Miền ứng dụng

- ❖ Phần mềm nghiệp vụ
  - CSDL: Oracle, DB2, SQL Server, MySQL...
  - ngôn ngữ: FoxPro, COBOL, VB, VC++
- ❖ Trí tuệ nhân tạo
  - Lisp, Prolog, OPS5,...
- ❖ Lập trình Web/CGI
  - Perl, ASP, PHP, Java, Java script, Python...





# NGÔN NGỮ LẬP TRÌNH - Miền ứng dụng

- ❖ Phần mềm hệ thống
  - C, C++
- ❖ Hệ thời gian thực
  - C, C++, Ada, Assembly
- ❖ Phần mềm nhúng
  - C++, Java, Assembly
- ❖ Phần mềm khoa học kỹ thuật
  - Fortran




# PHONG CÁCH LẬP TRÌNH

## Phong cách lập trình tốt

- ❑ Tuân theo các chuẩn thông dụng
- ❑ Chú giải đầy đủ mỗi khi không tuân theo chuẩn



# PHONG CÁCH LẬP TRÌNH

- ❑ Tuân theo chuẩn
    - cách đặt tên hàm và biến
    - cách xây dựng câu lệnh, cấu trúc chương trình
    - cách viết chú thích
    - cách xử lý lỗi
- 
- ❑ Hướng tới phong cách làm cho mã nguồn
    - dễ hiểu, dễ sửa đổi
    - an toàn (ít lỗi)



# PHONG CÁCH LẬP TRÌNH - Đặt tên

*Đặt tên biến, tên hàm có nghĩa, gợi nhớ*

- ❑ Sử dụng các ký hiệu, từ tiếng Anh có nghĩa
- ❑ Làm cho dễ đọc
  - dùng DateOfBirth hoặc date\_of\_birth
  - không viết dateofbirth
- ❑ Tránh đặt tên quá dài
  - không đặt tên dài với các biến cục bộ
- ❑ Thống nhất cách dùng
  - i cho vòng lặp, tmp cho các giá trị tạm thời...



# PHONG CÁCH LẬP TRÌNH – Chú thích

*Mọi điều được Chú thích trong chương trình*

- ❑ Mục đích sử dụng của các biến
- ❑ Chức năng của khối lệnh, câu lệnh
  - các lệnh điều khiển
  - các lệnh phức tạp



# PHONG CÁCH LẬP TRÌNH – Chú thích

*Mọi điều được Chú thích trong chương trình*

- Chú thích các mô đun
  - mục đích, chức năng của mô đun
  - tham số, giá trị trả lại (giao diện)
  - các mô đun thuộc cấp
  - cấu trúc, thuật toán
  - nhiệm vụ của các biến cục bộ
  - tác giả, người kiểm tra, thời gian



# PHONG CÁCH LẬP TRÌNH - Cấu trúc chương trình

- ❑ Chương trình cần được chia thành nhiều mô đun (hàm)
- ❑ Không viết hàm quá dài
  - không quá 2 trang màn hình
  - tạo ra các hàm thứ cấp để giảm độ dài từng hàm
- ❑ Không dùng quá nhiều biến cục bộ
  - không thể theo dõi đồng thời hoạt động của nhiều biến (vd. không quá 7 biến cục bộ)



# PHONG CÁCH LẬP TRÌNH – Câu lệnh

- ❑ Các câu lệnh phải mô tả cấu trúc
  - tụt lề, dễ đọc, dễ hiểu
- ❑ Làm đơn giản các lệnh
  - mỗi lệnh trên một dòng
  - triển khai các biểu thức phức tạp
  - hạn chế truyền tham số là kết quả của hàm, biểu thức  
`printf("%s", strcpy(des, src));`
- ❑ Tránh các cấu trúc phức tạp
  - các lệnh if lồng nhau
  - điều kiện phủ định if not



## ❖ Các lệnh if lồng nhau

Vĩ độ	90°					
	60°			①		
	30°			②		
		90°	120°	150°	180°	
		Kinh độ				

Hình 13.1 Các tọa độ trên bản đồ

```

if (vido >= 30 && kinhdo > 120)
{
    if (vido <= 60 && kinhdo <= 150)
    {
        // Định dạng tốt nhưng qua nhiều if lồng nhau
    }
    else
    {
        System.out.println("Not on the map");
    }
}
else
{
    System.out.println("Not on the map");
}

```

```

if (vido >= 30 && kinhdo > 120) mapSquareNo = 1;
else if (vido <= 60 && kinhdo <= 150) mapSquareNo = 2;
else System.out.println("Not on the map");

```

Định dạng xấu, có quá nhiều if lồng nhau



## Ví dụ if (tt)

### ❖ Các câu if chấp nhận được

```
if (kinhdo>120 && kinhdo<=150 && vido>=30  
&& vido<=60)
```

```
    mapSquareNo = 1;
```

```
else if (kinhdo>120 && kinhdo<=150 &&  
vido>60 && vido<=90)
```

```
    mapSquareNo = 2;
```

```
else
```

```
    System.out.println("Not on the map");
```



# PHONG CÁCH LẬP TRÌNH – Xử lý lỗi

- ❑ Có thể phát hiện lỗi trong khi thực hiện
  - lỗi chia 0
  - lỗi input/output, ...
- ❑ Xử lý lỗi
  - nhất quán trong xử lý
    - phân loại lỗi
    - thống nhất định dạng thông báo,...
  - phân biệt output và thông báo lỗi



# PHONG CÁCH LẬP TRÌNH – Xử lý lỗi

## Ngoại lệ

- ❖ Là cách thức xử lý lỗi tiến tiến trong các ngôn ngữ hướng đối tượng
  - môđun xử lý ném ra một ngoại lệ (đối tượng chứa thông tin lỗi)
  - môđun điều khiển bắt ngoại lệ (nếu có)
- ❖ Tách phần xử lý lỗi khỏi phần cài đặt thuật toán thông thường, làm cho chương trình dễ đọc hơn
- ❖ Dễ dùng hơn, an toàn hơn



# PHONG CÁCH LẬP TRÌNH – Xử lý lỗi

## Ngoại lệ: ném ngoại lệ

```
double MyDivide(double num, double denom)
{
    if (denom == 0.0) {
        throw invalid_argument("The denom cannot be 0.");
    }
    else {
        return num / denom;
    }
}
```



# PHONG CÁCH LẬP TRÌNH – Xử lý lỗi

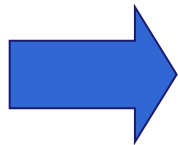
**Ngoại lệ:** bắt ngoại lệ

```
try {  
    result = MyDivide(x, y);  
}  
catch (invalid_argument& e) {  
    cerr << e.what() << endl;  
    ...      // mã xử lý với ngoại lệ  
};
```



# KỸ THUẬT LẬP TRÌNH

- ❖ Tránh lỗi
- ❖ Phòng thủ
- ❖ Thử lỗi
- ❖ Hướng hiệu quả



*Xây dựng hệ thống tin cậy*



# KỸ THUẬT LẬP TRÌNH – tránh lỗi

## *Tránh các cấu trúc nguy hiểm*

- ❖ Số thực
- ❖ Con trỏ
- ❖ Cấp phát bộ nhớ
- ❖ Đệ quy



## *Defensive programming*

### ❖ Dự đoán khả năng xuất hiện lỗi

- Lệnh và

- Các phép

```
FILE* fp;  
fp = fopen("data",  
"r");
```

```
FILE* fp;  
if (NULL == (fp =  
fopen("data", "r"))) {  
    fprintf(stderr, "can not  
open file...");  
    ...  
}
```

- lưu trạng

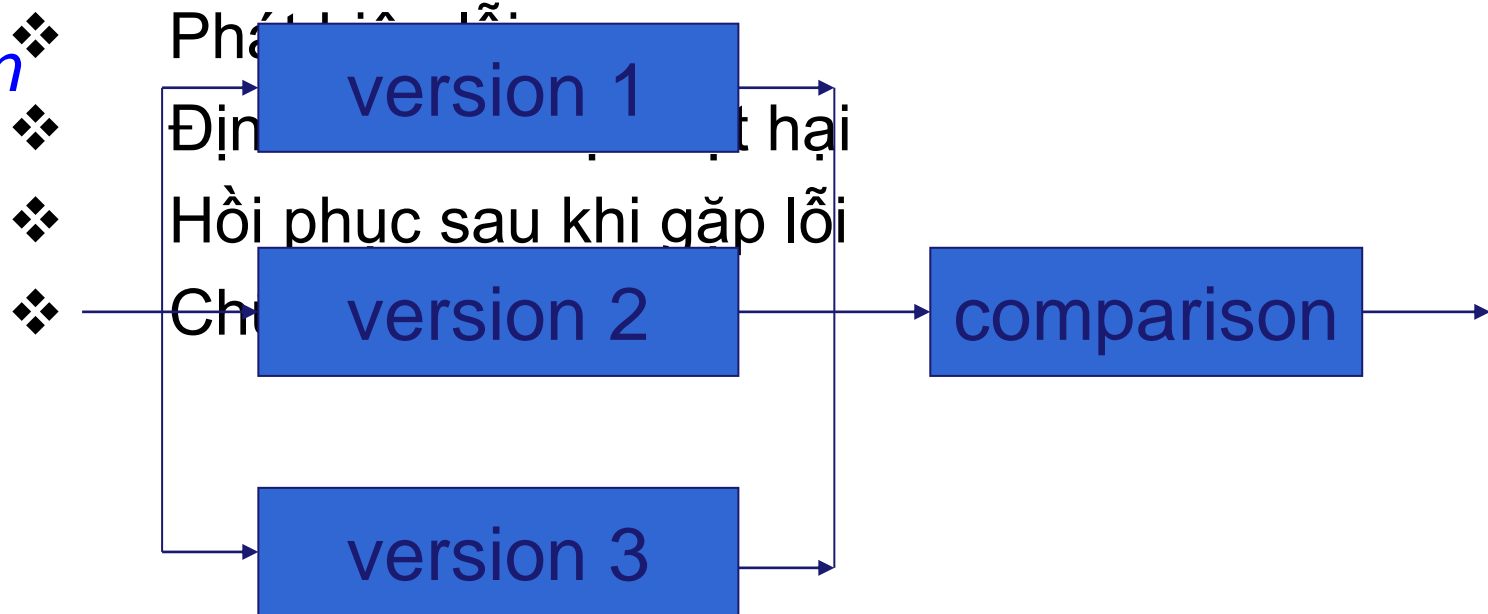
- quay lại trạng thái an toàn gần nhất



# KỸ THUẬT LẬP TRÌNH – thứ lỗi

## *Fault tolerance programming*

*N - version*





# KỸ THUẬT LẬP TRÌNH-hướng hiệu quả

- ❖ Sử dụng bộ nhớ
- ❖ Tốc độ chương trình



# KỸ THUẬT LẬP TRÌNH-hướng hiệu quả

- ❖ Đơn giản hóa các biểu thức số học và logic
- ❖ Giảm các câu lệnh lặp lồng nhau
- ❖ Tránh dùng mảng nhiều chiều
- ❖ Tránh việc dùng con trỏ và danh sách phức tạp
- ❖ Dùng các phép toán số học “nhanh”
- ❖ Không trộn lẫn các kiểu dữ liệu
- ❖ Tận dụng khả năng tối ưu mã của trình biên dịch



# KỸ THUẬT LẬP TRÌNH-hướng hiệu quả

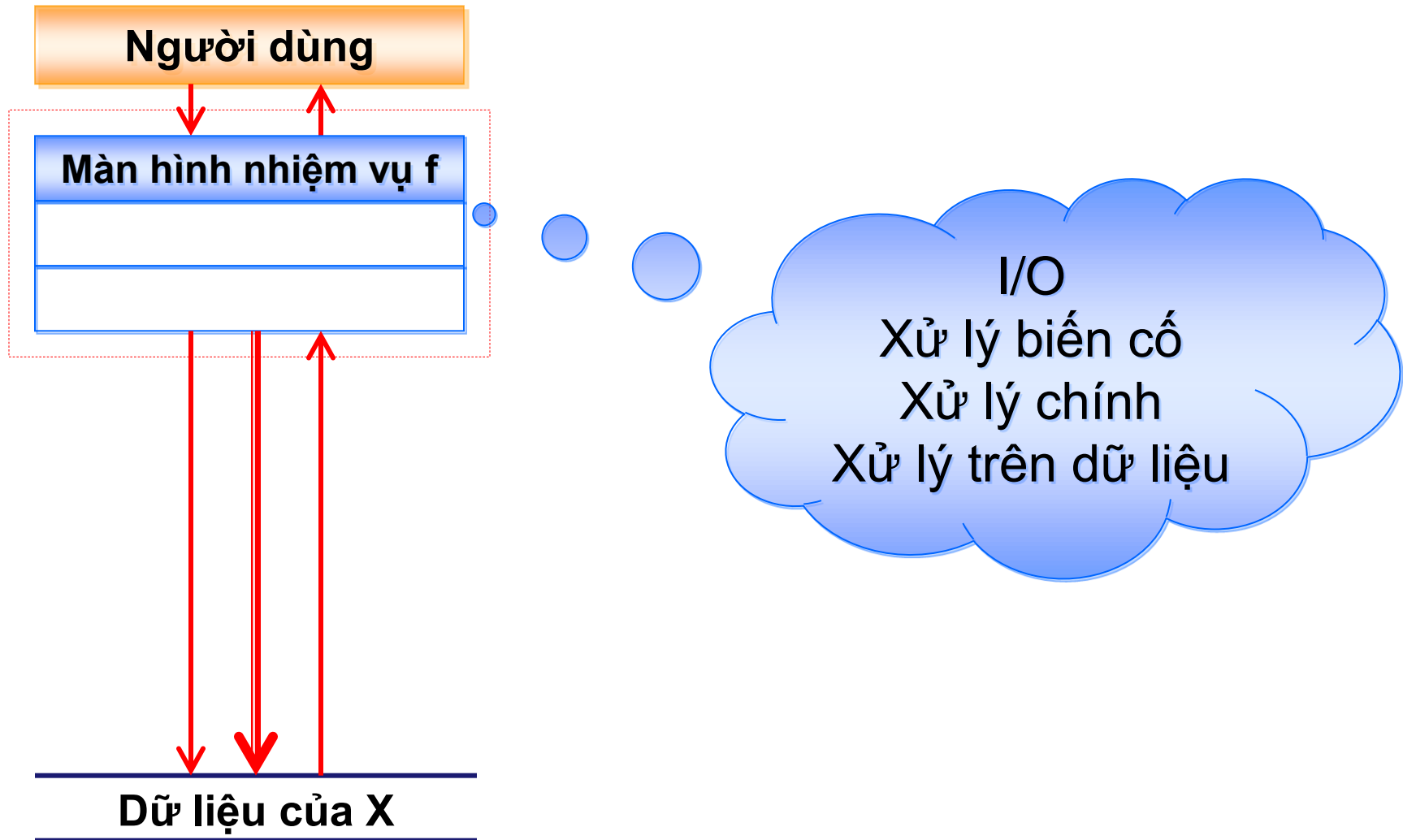
- ❖ Hiệu quả bộ nhớ
  - Phụ thuộc vào đặc trưng phân trang của hệ điều hành. Mô đun hóa và đảm bảo tính cấu trúc của chương trình làm giảm việc phân trang
  - Nếu yêu cầu hệ thống cần tới bộ nhớ tối thiểu (như sản phẩm giá thấp, khối lượng lớn) thì có thể phải dùng tới hợp ngữ.



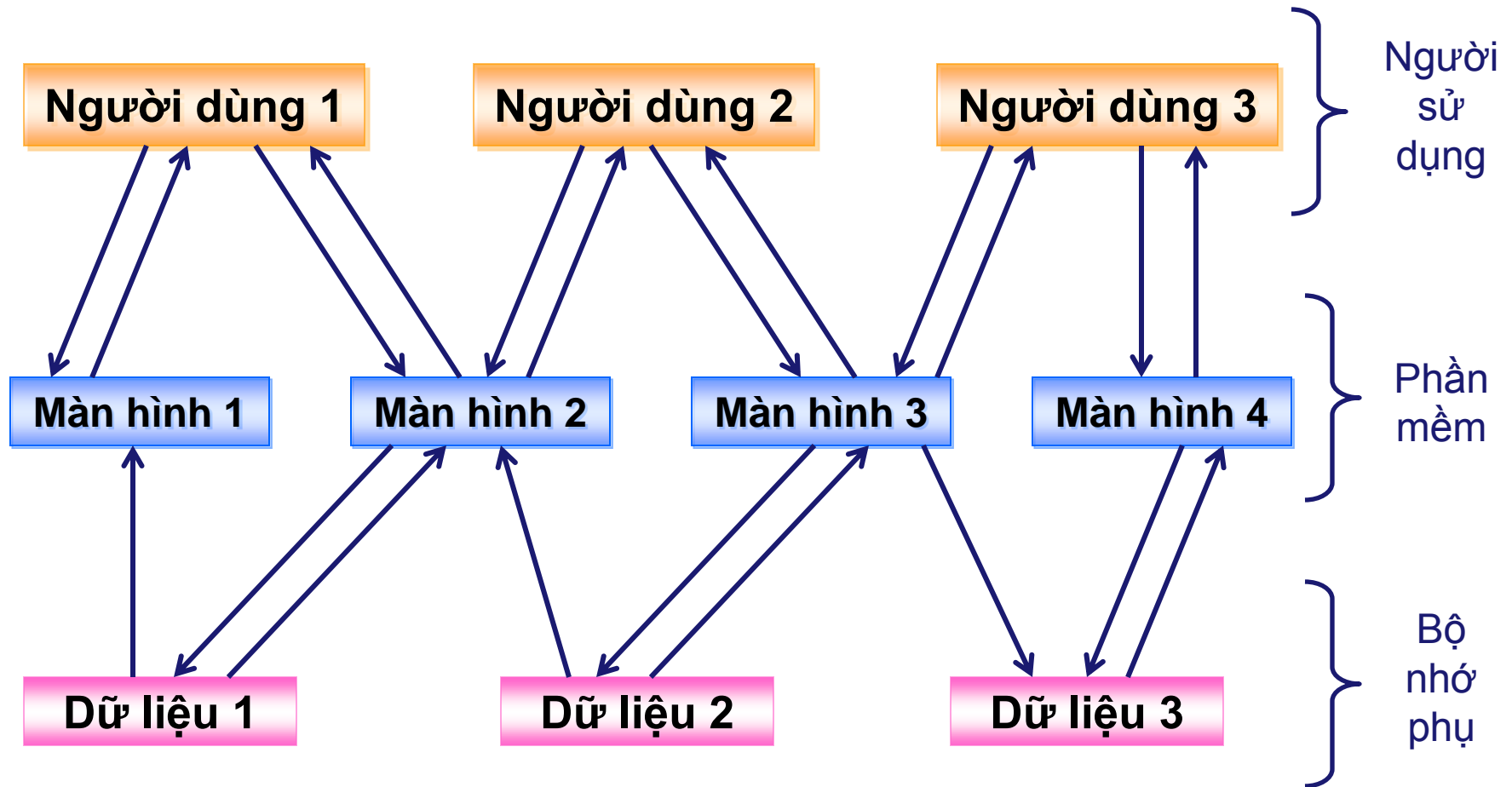
# KỸ THUẬT LẬP TRÌNH-hướng hiệu quả

- ❖ Hiệu quả vào ra
  - Số các yêu cầu vào/ra nên giữ mức tối thiểu
  - Vào/ra nên qua bộ đệm
  - Nên xếp khối vào/ra với các thiết bị bộ nhớ phụ.

# Mô hình kiến trúc 1 tầng (1 layer)



# Mô hình kiến trúc 1 tầng (1 layer)

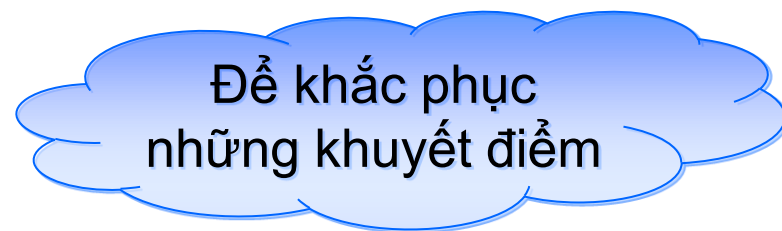






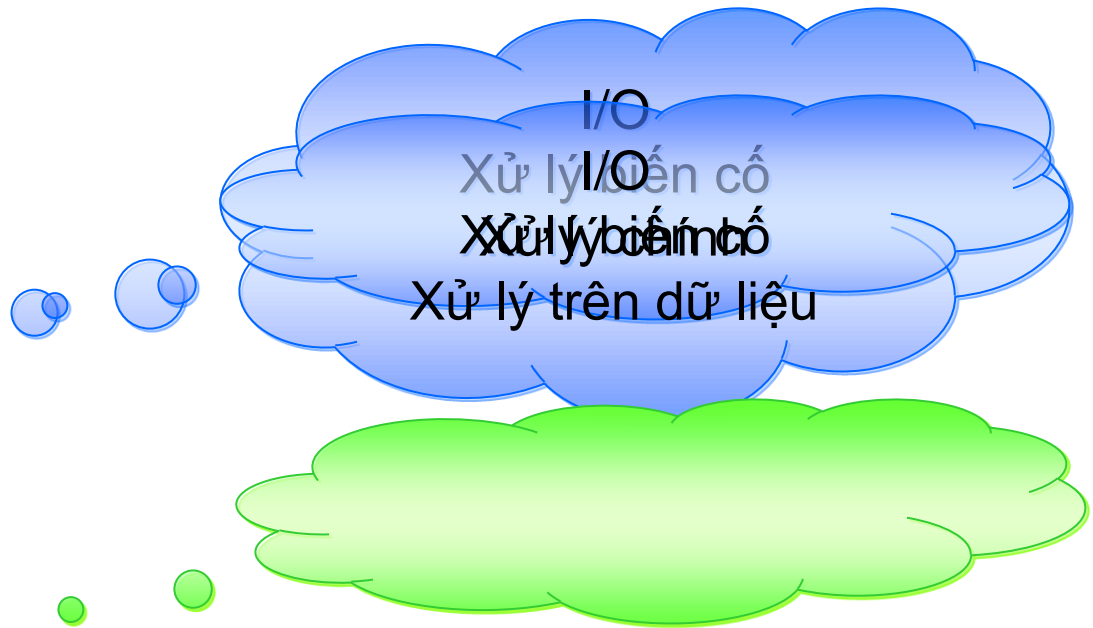
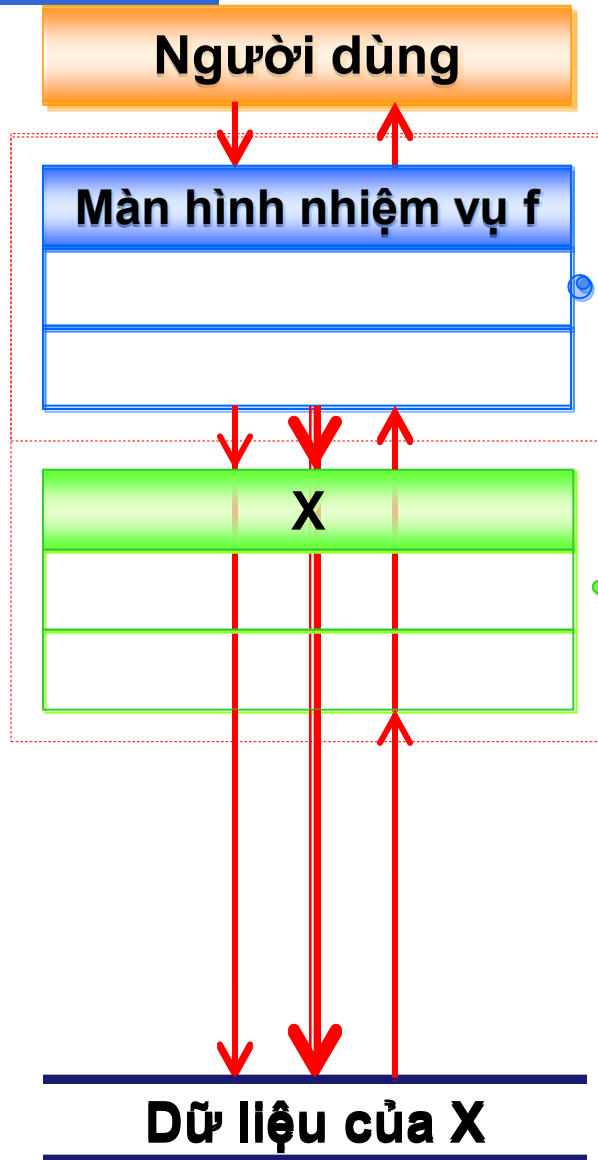
# Mô hình kiến trúc 1 tầng (1 layer)

- ❖ Hệ thống trên bao gồm:
  - 3 người sử dụng
  - 4 đơn vị xử lý
  - 3 đơn vị lưu trữ
- ❖ Đặc điểm: Không có sự phân loại các xử lý
- ❖ Ưu điểm: Thiết kế và lập trình nhanh
- ❖ Nhược điểm:
  - Mỗi đơn vị xử lý phức tạp
  - Khó bảo trì
  - Không có tính tái sử dụng

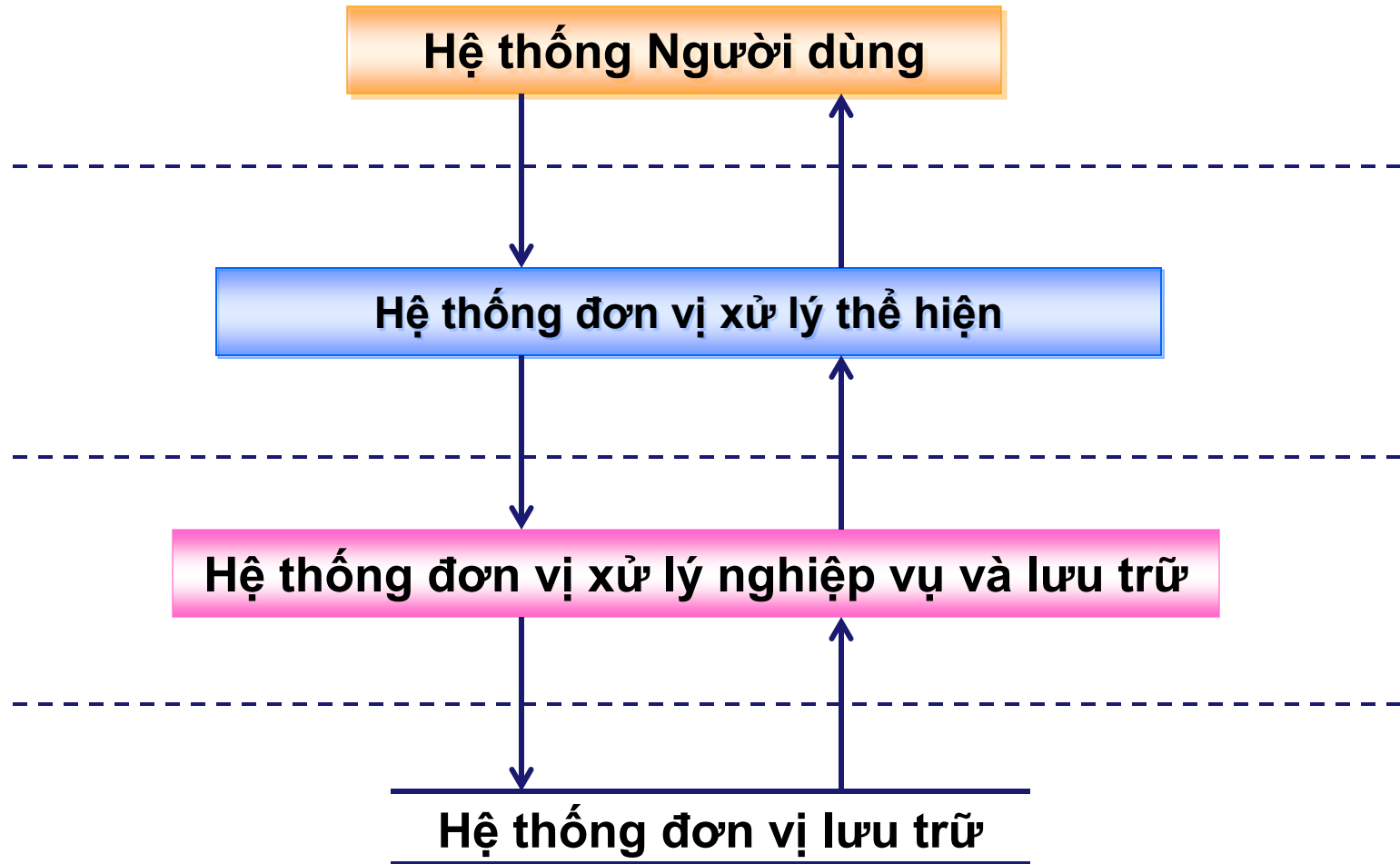


???

# Mô hình kiến trúc 2 tầng (2 layer)



# Mô hình kiến trúc 2 tầng (2 layer)

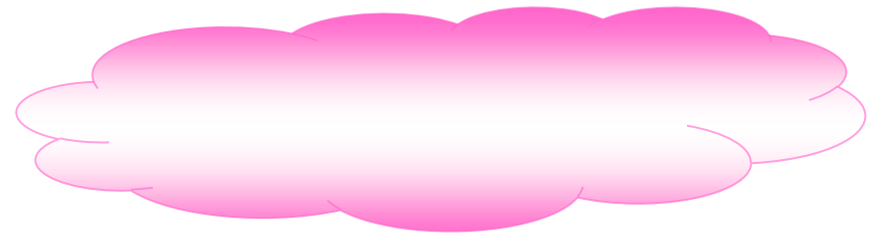
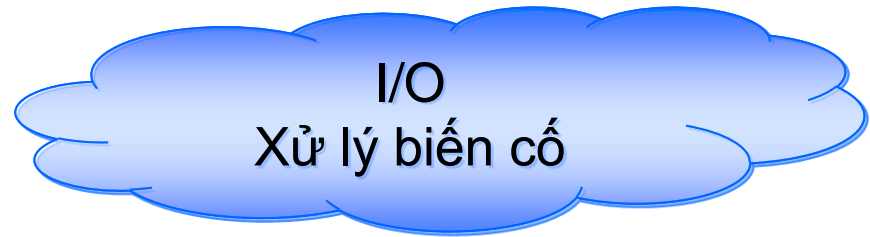
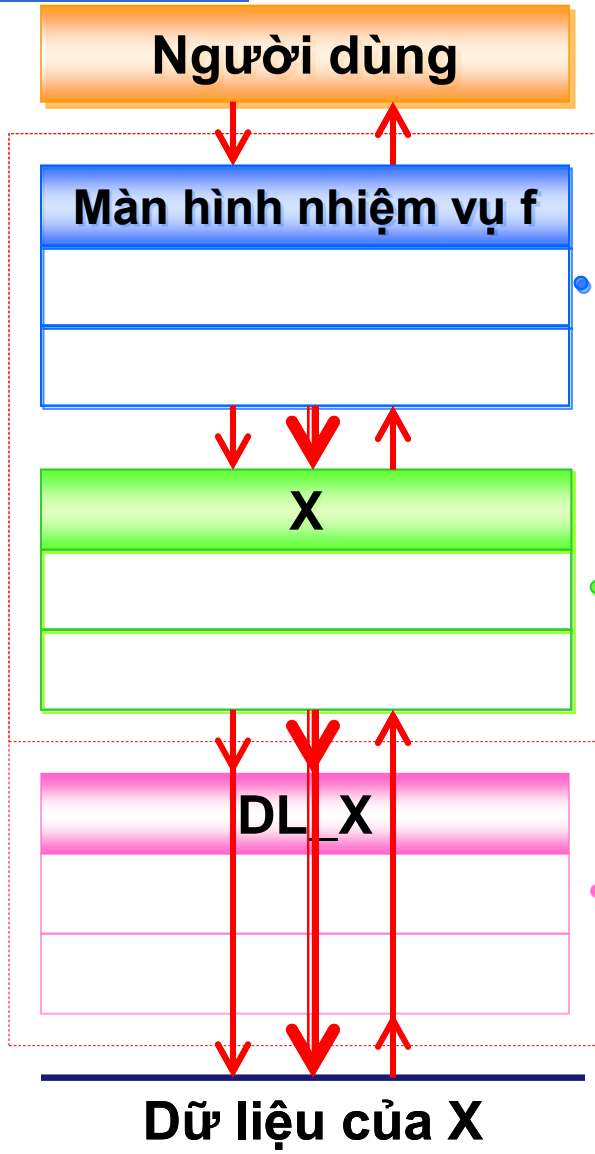




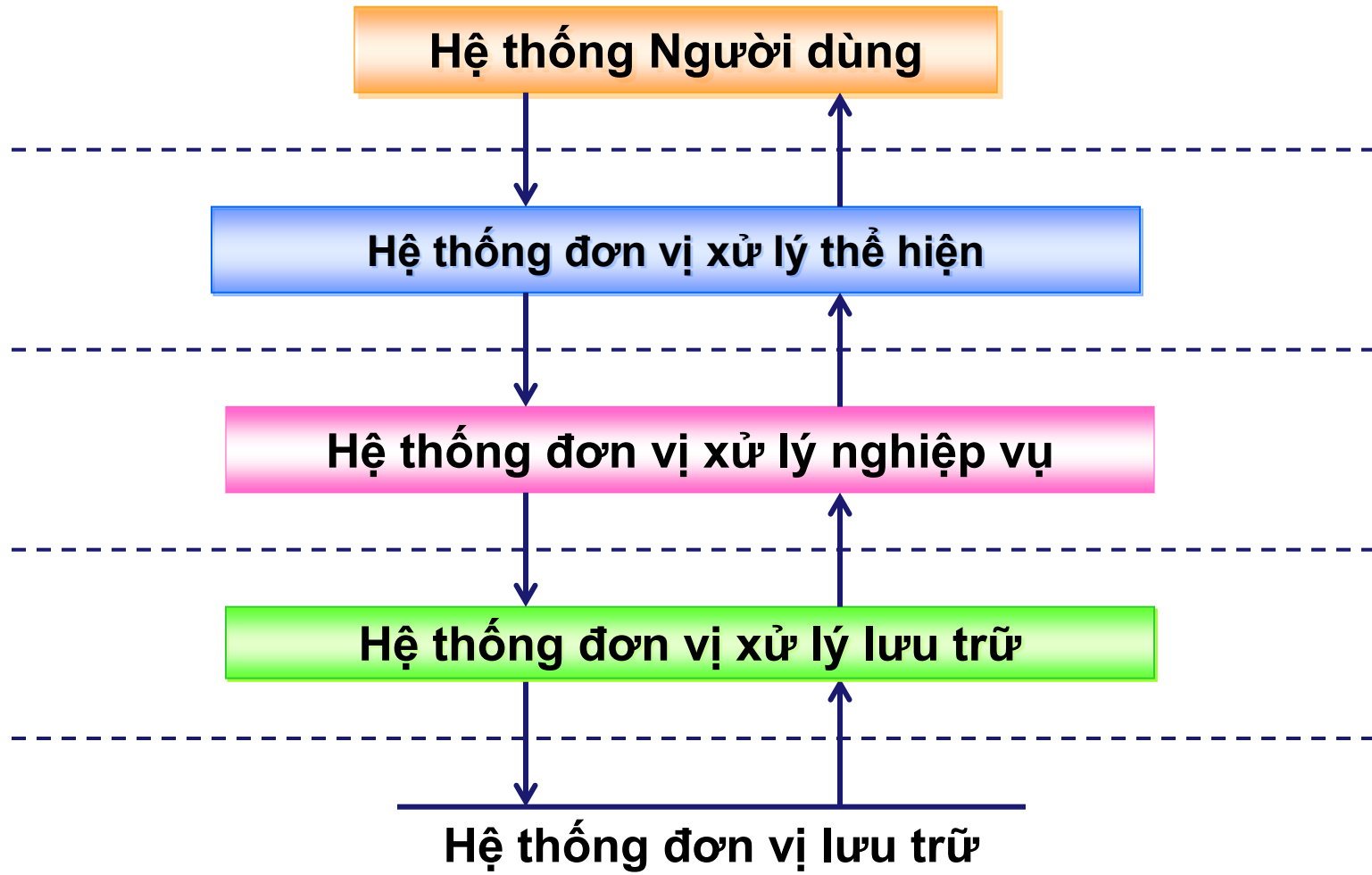
## Mô hình kiến trúc 2 tầng (2 layer)

- ❖ Đặc điểm: Các đơn vị xử lý được phân thành 2 loại
  - Loại 1: Các đơn vị xử lý chuyên biệt về giao tiếp người dùng
  - Loại 2: Các đơn vị xử lý nghiệp vụ (kiểm tra, tính toán), lưu trữ (đọc, ghi)
- ❖ Ưu điểm, khuyết điểm ?

# Mô hình kiến trúc 3 tầng (3 layer)



# Mô hình kiến trúc 3 tầng (3 layer)

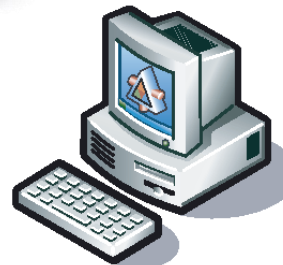


# **Giới thiệu Microsoft Visual Source Safe**



# Giới thiệu

Qui mô dự án phần mềm lớn  
Nhiều người cùng tham gia  
Tổ chức, quản lý ???



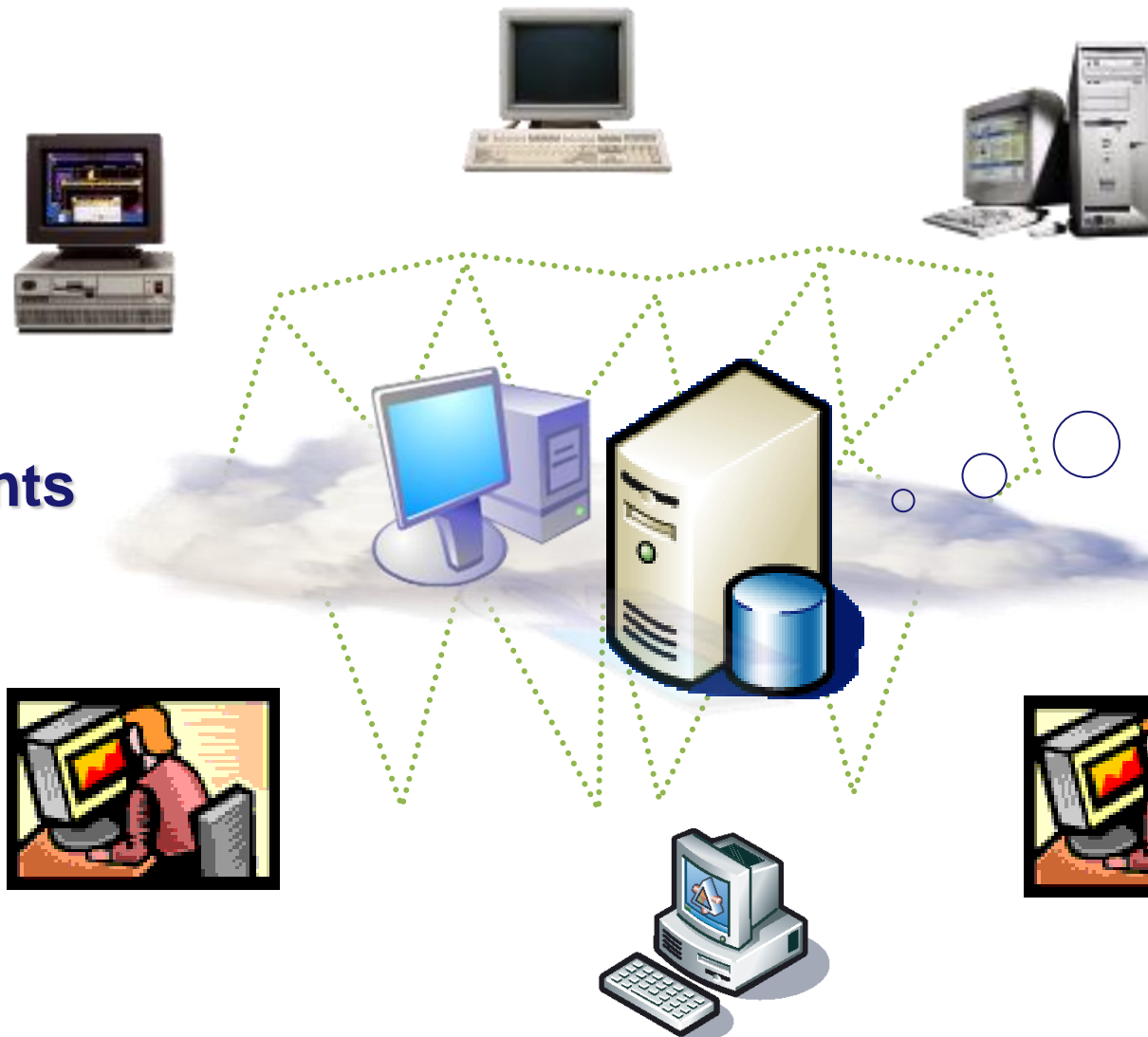
Phần mềm





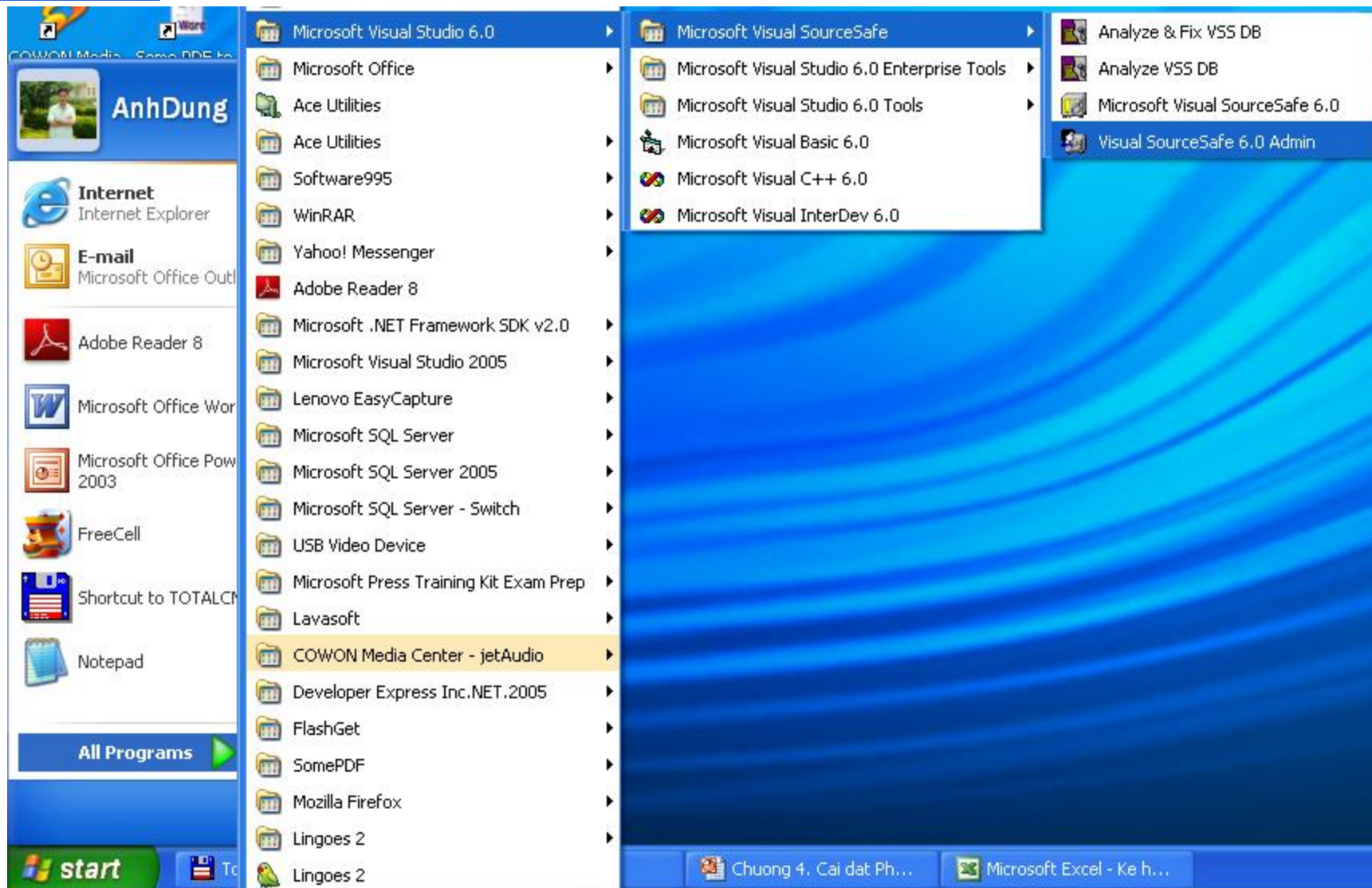
# Giới thiệu

**Clients**

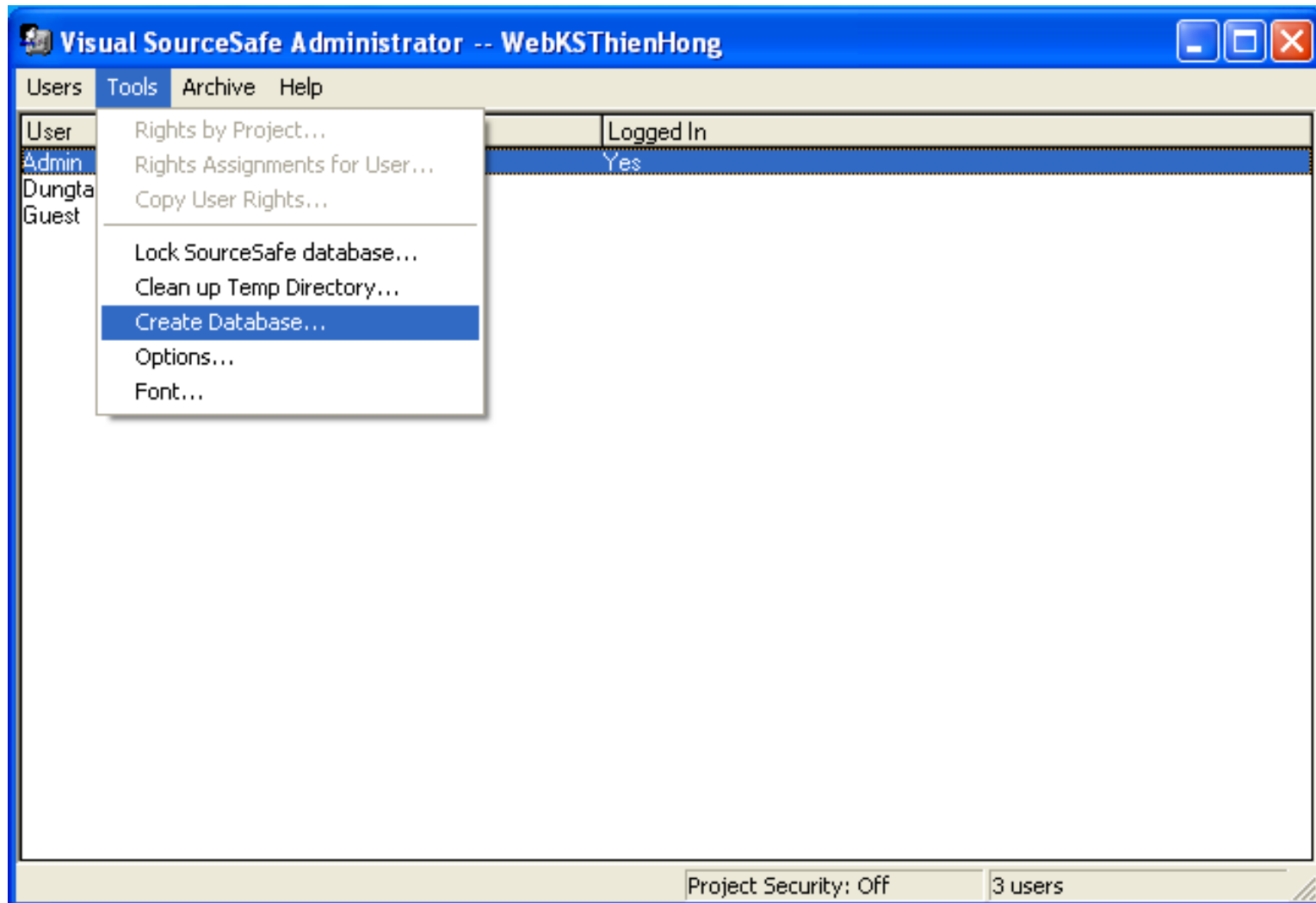


Cần phải  
có công  
cụ hỗ trợ  
quản lý

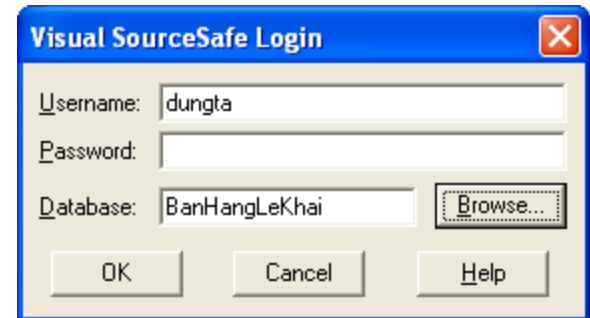
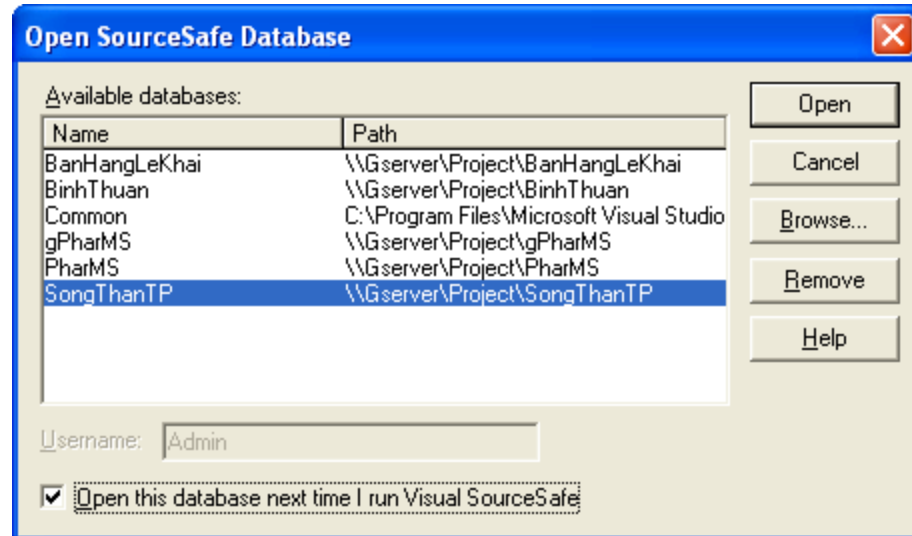
# Giới thiệu



# Visual SourceSafe Admin

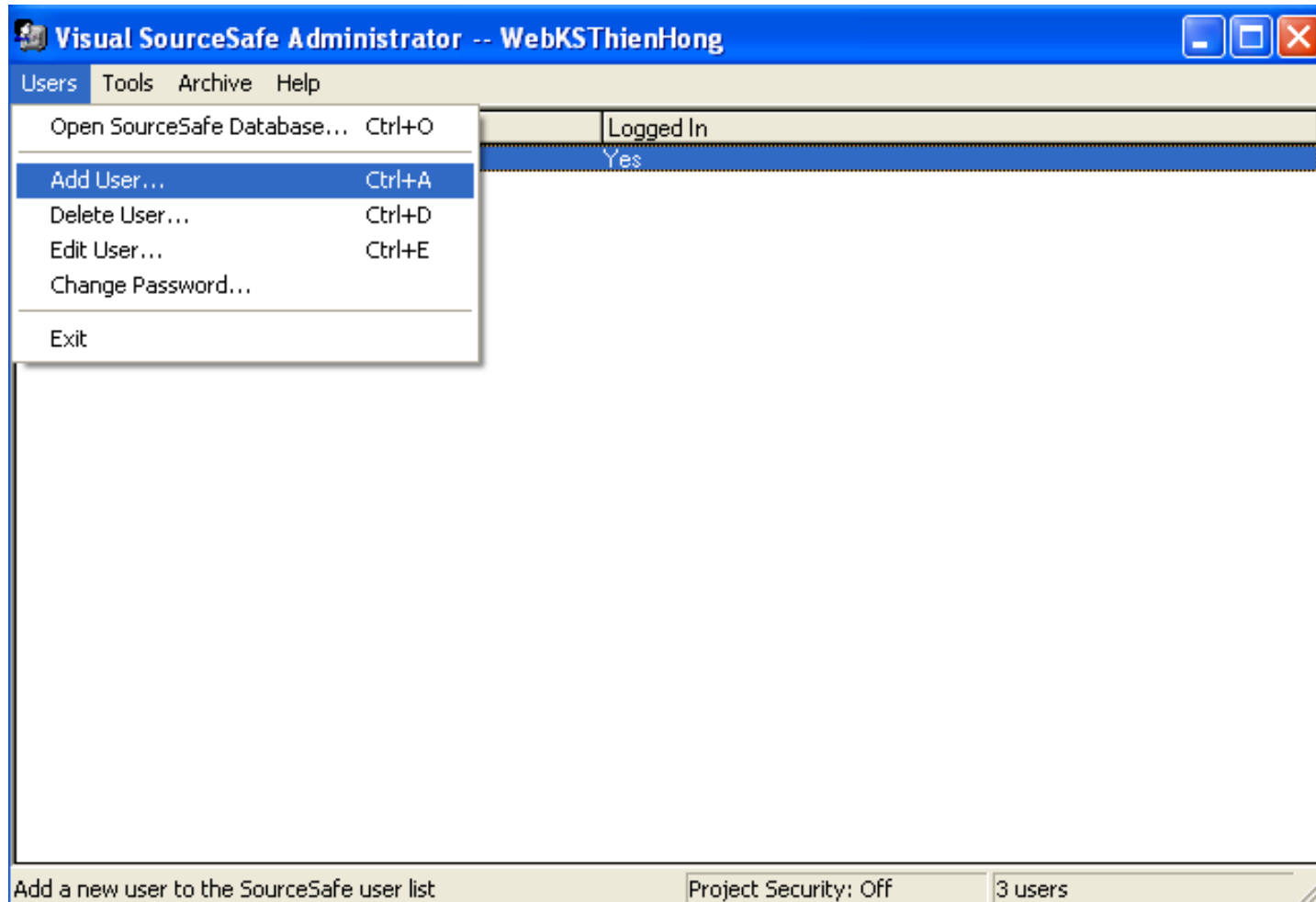


# Visual SourceSafe Admin



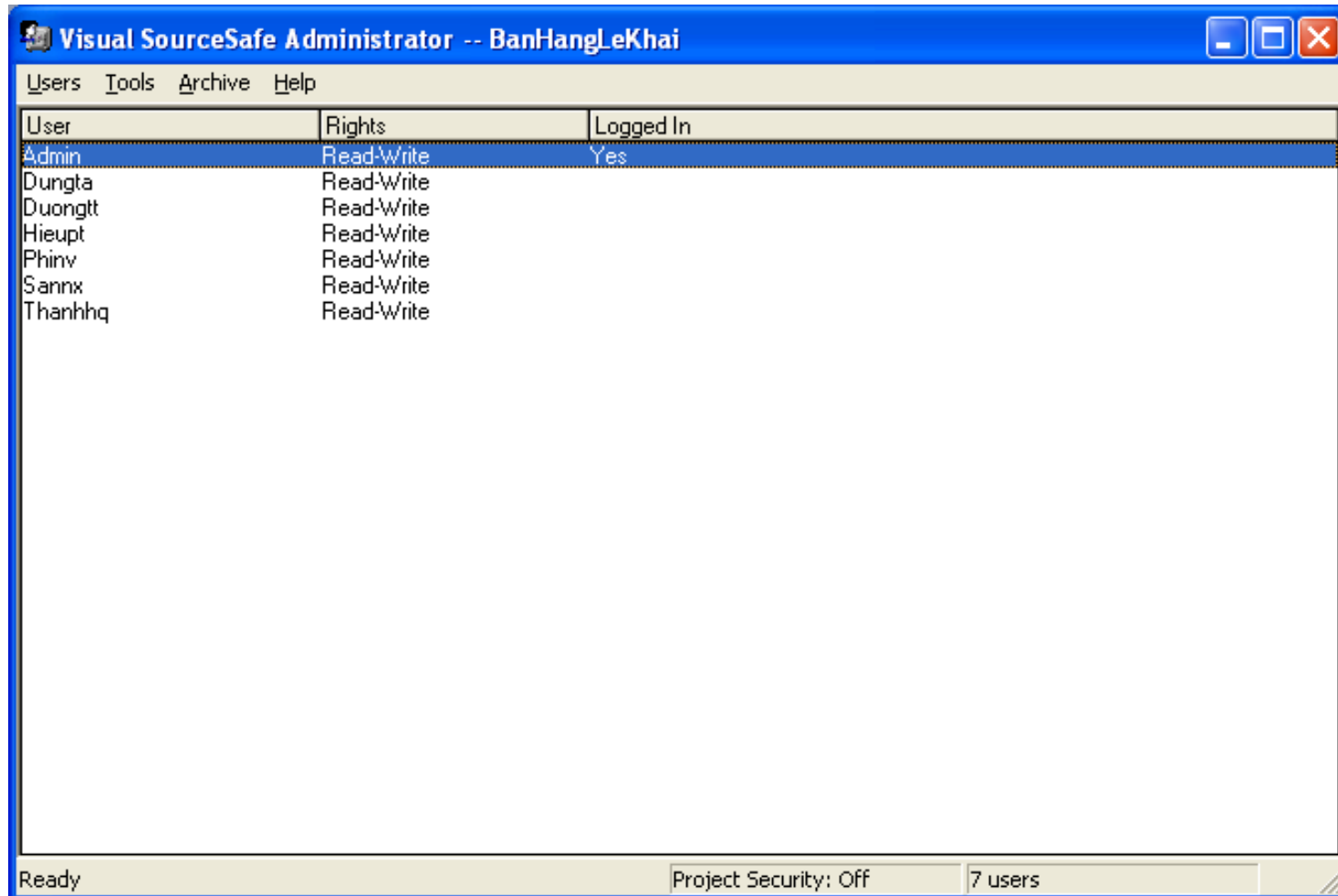
# Visual SourceSafe Admin

## ❖ Quản lý user

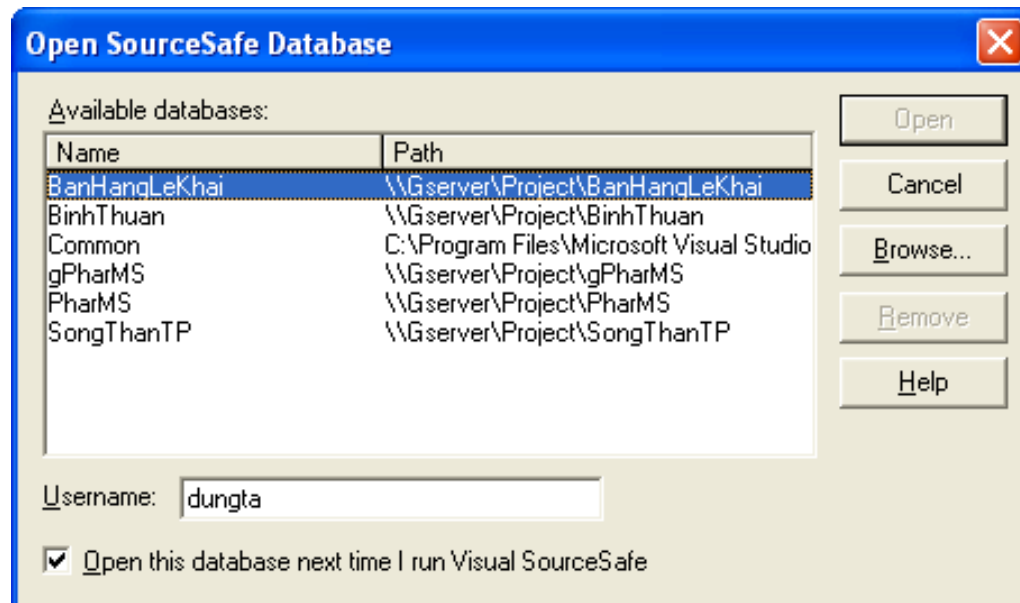


# Visual SourceSafe Admin

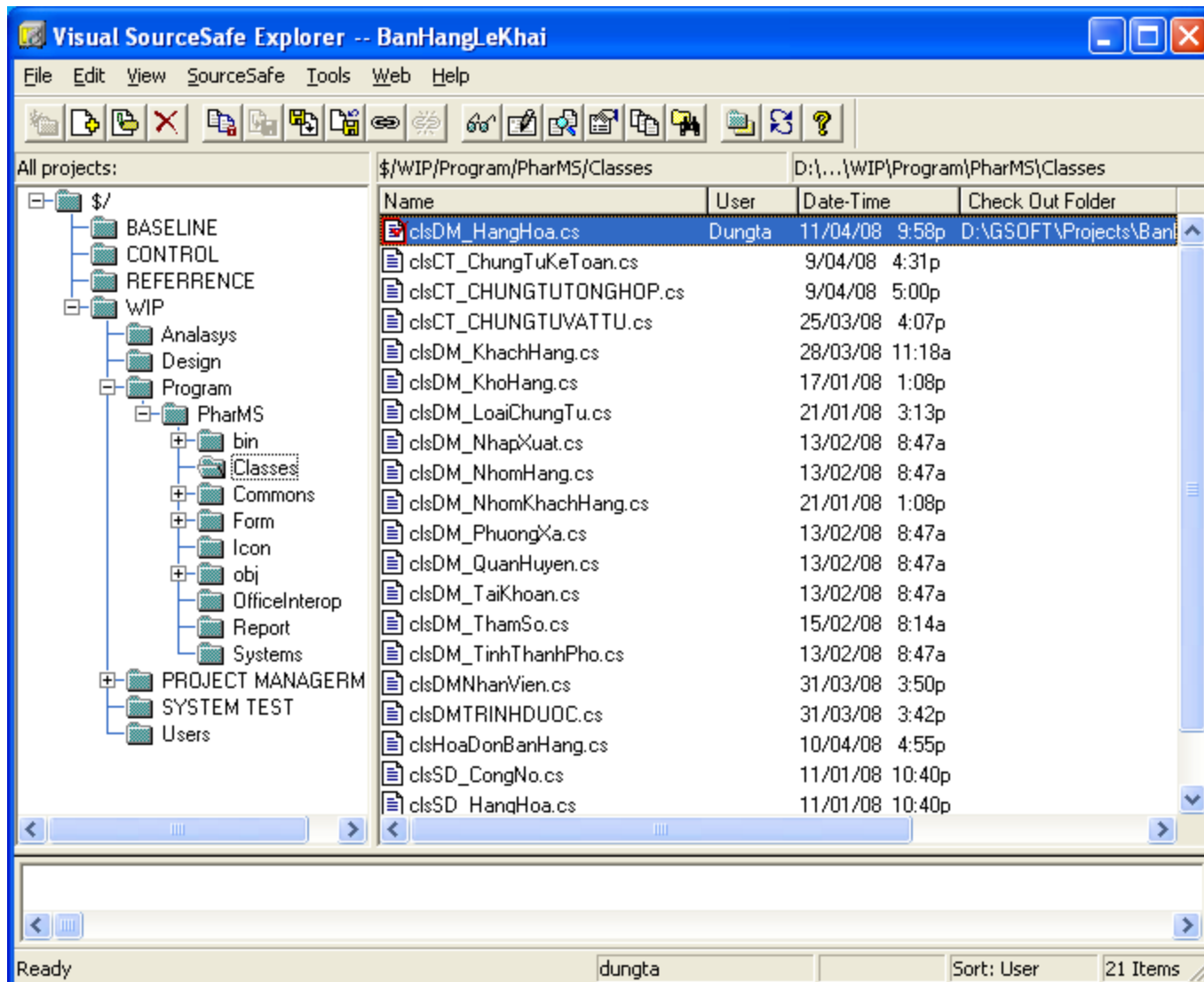
## ❖ Quản lý user



# Microsoft Visual SourceSafe

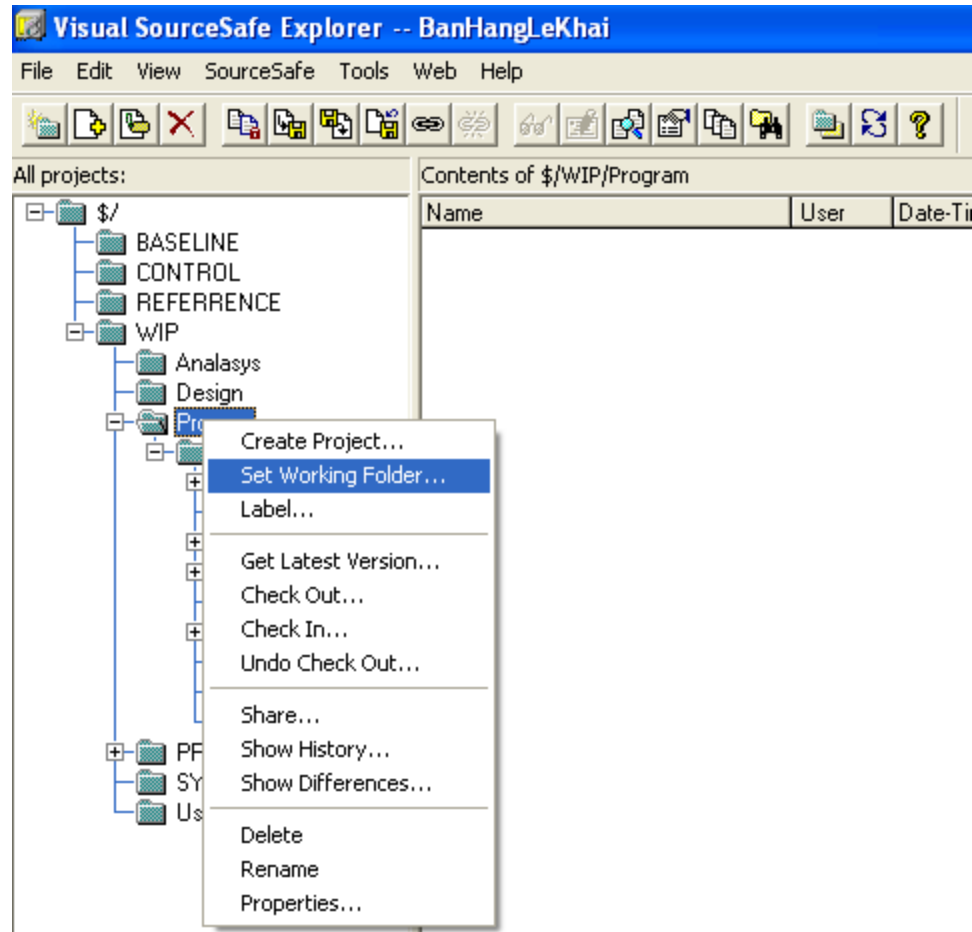


# Microsoft Visual SourceSafe

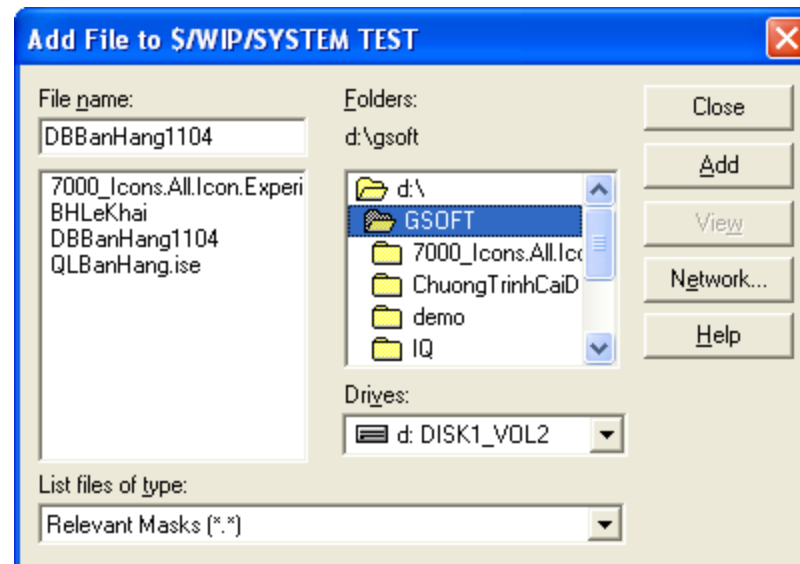




# Microsoft Visual SourceSafe



# Microsoft Visual SourceSafe



# Microsoft Visual SourceSafe

**Visual SourceSafe Explorer -- BanHangLeKhai**

File Edit View SourceSafe Tools Web Help

All projects: Contents of \$/WIP/Program/PharMS/Classes

Tree view structure:

- \$/
  - BASELINE
  - CONTROL
  - REFERENCE
  - WIP
    - Analasys
    - Design
    - Program
      - PharMS
        - bin
        - Classes
        - Commons
        - Form
        - Icon
        - obj
        - OfficelInterop
        - Report
        - Systems
  - PROJECT MANAGERM
  - SYSTEM TEST
  - Users

Table of file contents:

Name	User	Date-Time	Check Out Folder
clsDM_Han		11/04/08 9:58p	D:\GSOFT\Projects\BanHangLe
clsCT_Chur		9/04/08 4:31p	
clsCT_CHU		9/04/08 5:00p	
clsCT_CHU		25/03/08 4:07p	
clsDM_Kha		28/03/08 11:18a	
clsDM_Kho		17/01/08 1:08p	
clsDM_Loi		21/01/08 3:13p	
clsDM_Nha		13/02/08 8:47a	
clsDM_Nho		13/02/08 8:47a	
clsDM_Nho		21/01/08 1:08p	
clsDM_Phu		13/02/08 8:47a	
clsDM_Qua		13/02/08 8:47a	
clsDM_TaiK		13/02/08 8:47a	
clsDM_ThamSo.cs		15/02/08 8:14a	
clsDM_TinhThanhPho.cs		13/02/08 8:47a	
clsDMNhanVien.cs		31/03/08 3:50p	
clsDMTRINHDOUC.cs		31/03/08 3:42p	
clsHoaDonBanHang.cs		10/04/08 4:55p	
clsSD_CongNo.cs		11/01/08 10:40p	
clsSD_HangHoa.cs		11/01/08 10:40p	
clsSD_TaiKhoan.cs		8/04/08 4:53p	

