

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN &
TRUYỀN THÔNG VIỆT HÀN

Khoa Khoa Học Máy Tính



**ĐỒ ÁN HỌC PHẦN LẬP TRÌNH MẠNG
XÂY DỰNG CHƯƠNG TRÌNH CHIA SẺ MÁY IN
QUA MẠNG**

Sinh viên thực hiện: Hoàng Vũ Dạ Quỳnh 19IT6

Thái Thị Thu Xuân 19IT5

Giảng viên hướng dẫn: ThS. Nguyễn Thanh Cẩm

Đà Nẵng, tháng 11 năm 2021

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN &
TRUYỀN THÔNG VIỆT HÀN

Khoa Khoa Học Máy Tính



ĐỒ ÁN HỌC PHẦN LẬP TRÌNH MẠNG
**XÂY DỰNG CHƯƠNG TRÌNH CHIA SẺ
MÁY IN QUA MẠNG**

Sinh viên thực hiện:	Hoàng Vũ Dạ Quỳnh	19IT6
	Thái Thị Thu Xuân	19IT5
Giảng viên hướng dẫn:	ThS. Nguyễn Thanh Cẩm	

Đà Nẵng, tháng 11 năm 2021

NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN

.....

.....

.....

.....

.....

.....

.....

LỜI CẢM ƠN

Em xin trân trọng cảm ơn

Sinh viên

MỤC LỤC

PHẦN 1: GIỚI THIỆU	6
PHẦN 2: CƠ SỞ LÝ THUYẾT	7
1. Giao thức TCP/IP	7
2. Mô hình Client/Server	8
PHẦN 3: NỘI DUNG CHƯƠNG TRÌNH	10

PHẦN 1: GIỚI THIỆU

Thế kỷ 21 được mệnh danh là thế kỷ của công nghệ thông tin, với sự bùng nổ mạnh mẽ về khoa học công nghệ. Đây là kỷ nguyên của nền văn minh dựa trên cơ sở công nghiệp trí tuệ. Ngày nay, tin học đã trở thành một môn khoa học quan trọng trên thế giới. Sự phát triển mạnh mẽ như vậy thì công việc lập trình các ứng dụng nhằm phục vụ nhu cầu, lợi ích của con người trở nên cấp thiết. Máy tính đã trở thành công cụ đắc lực và không thể thiếu của con người.

Trong công việc hàng ngày của nhân viên văn phòng thường phải in rất nhiều tài liệu khác nhau phục vụ cho hợp hành hoặc tiếp xúc khách hàng nhưng mỗi văn phòng thường chỉ có 1 máy in. Chính vì thế, mỗi lần in sẽ rất mất thời gian, nhất là nếu phải in với số lượng lớn. Do đó nhóm em chọn đề tài “Xây dựng chương trình chia sẻ máy in qua mạng” với giao thức TCP để làm bài báo cáo cuối kỳ môn Lập Trình Mạng.

Do thời gian, kiến thức và hiểu biết còn hạn chế, chương trình chỉ cài đặt một vài chức năng đơn giản nhất của ứng dụng chia sẻ file.

PHẦN 2: CƠ SỞ LÝ THUYẾT

Quá trình truyền dữ liệu trên mạng diễn ra khá phức tạp, chi tiết quá trình này diễn ra tương tự như trong thực tế ta gửi thư hay bưu phẩm, trước hết cần phải ghi rõ địa chỉ nơi đến (trường này là địa chỉ IP của máy chủ), sau đó có thể gửi thông thường hay bảo đảm (tùy theo cách gửi mà thư hay bưu phẩm có chắc chắn đến được tay người nhận hay không), người nhận sau khi nhận được có thể hồi âm trả lời đã nhận đủ hay mất mát gì trong quá trình chuyển tải, người gửi có thể gửi tiếp những phần bị mất (hoặc không cần gửi nữa).

Cách chuyển dữ liệu thông thường, không đảm bảo tương ứng với giao thức UDP (User Datagram Protocol), còn cách chuyển dữ liệu đảm bảo tương ứng với giao thức TCP (Transmission Control Protocol).

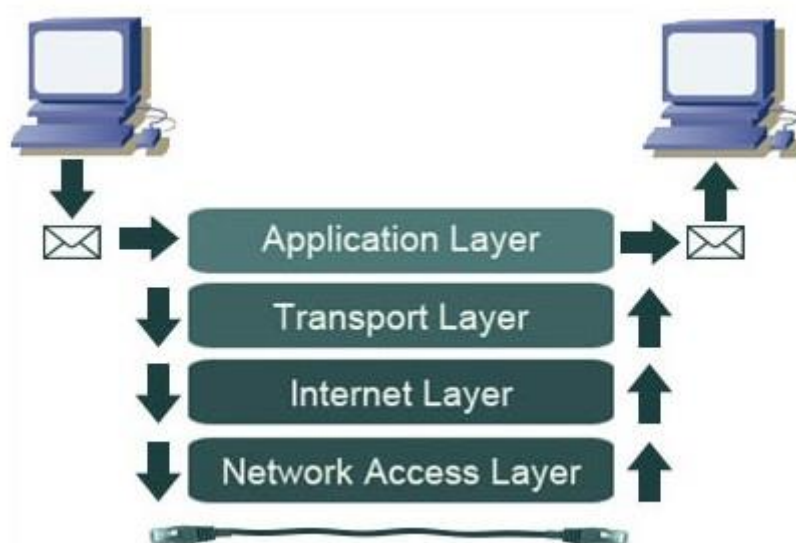
Chương trình “Ứng dụng chia sẻ file trong mạng LAN” sử dụng giao thức TCP để đảm bảo tính chuẩn xác trong quá trình truyền dữ liệu.

1. Giao thức TCP/IP

TCP/IP (Transmission Control Protocol/ Internet Protocol) là bộ giao thức cho phép kết nối các hệ thống mạng không đồng nhất với nhau. Ngày nay, TCP/IP được sử dụng rộng rãi trong các mạng cục bộ cũng như trên mạng Internet toàn cầu.

TCP/IP được xem là giản lược của mô hình tham chiếu OSI với bốn tầng như sau:

- Tầng liên kết mạng (Network Access Layer)
- Tầng Internet (Internet Layer)
- Tầng giao vận (Host-to-Host Transport Layer)
- Tầng ứng dụng (Application Layer)



- **Tầng liên kết mạng (Network Access Layer)**

Tầng liên kết (còn được gọi là tầng liên kết dữ liệu hay là tầng giao tiếp mạng) là tầng thấp nhất trong mô hình TCP/IP, bao gồm các thiết bị giao tiếp mạng và chương trình cung cấp các thông tin cần thiết để có thể hoạt động, truy nhập đường truyền vật lý qua thiết bị giao tiếp mạng đó.

- **Tầng Internet (Internet Layer)**

Tầng Internet (còn gọi là tầng mạng) xử lý quá trình truyền gói tin trên mạng. Các giao thức của tầng này bao gồm: IP (Internet Protocol), ICMP (Internet Control Message Protocol), IGMP (Internet Group Messages Protocol).

- **Tầng Tầng giao vận (Host-to-Host Transport Layer)**

Tầng giao vận phụ trách luồng dữ liệu giữa hai trạm thực hiện các ứng dụng của tầng trên. Tầng này có hai giao thức chính: TCP (Transmission Control Protocol) và UDP (User Datagram Protocol)

TCP cung cấp một luồng dữ liệu tin cậy giữa hai trạm, nó sử dụng các cơ chế như chia nhỏ các gói tin của tầng trên thành các gói tin có kích thước thích hợp cho tầng mạng bên dưới, báo nhận gói tin, đặt hạn chế thời gian time-out để đảm bảo bên nhận biết được các gói tin đã gửi đi. Do tầng này đảm bảo tính tin cậy, tầng trên sẽ không cần quan tâm đến nữa.

UDP cung cấp một dịch vụ đơn giản hơn cho tầng ứng dụng. Nó chỉ gửi các gói dữ liệu từ trạm này tới trạm kia mà không đảm bảo các gói tin đến được tới đích. Các cơ chế đảm bảo độ tin cậy cần được thực hiện bởi tầng trên.

- **Tầng ứng dụng (Application Layer)**

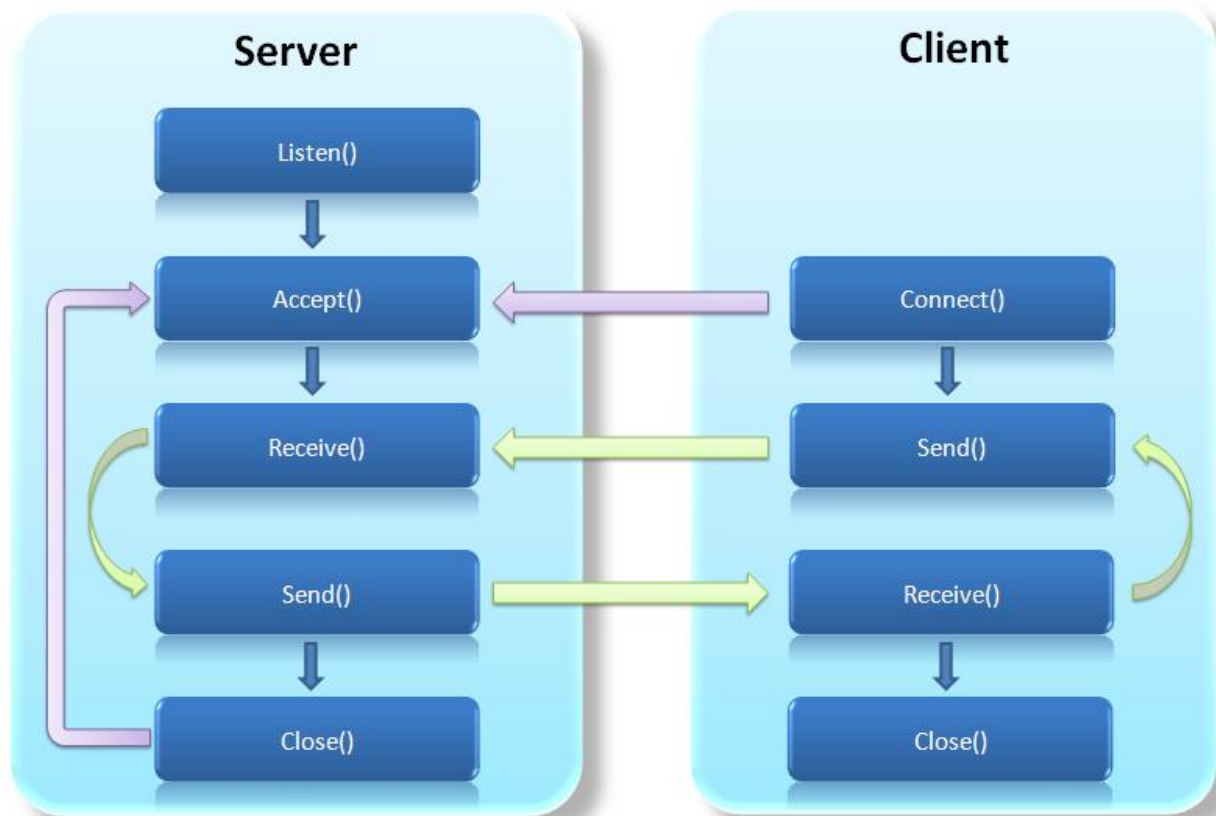
Tầng ứng dụng là tầng trên cùng của mô hình TCP/IP bao gồm các tiến trình và các ứng dụng cung cấp cho người sử dụng để truy cập mạng. Có rất nhiều ứng dụng được cung cấp trong tầng này, mà phổ biến là: Telnet: sử dụng trong việc truy cập mạng từ xa, FTP (File Transfer Protocol): dịch vụ truyền tệp, Email: dịch vụ thư tín điện tử, WWW (World Wide Web).

2. Mô hình Client/Server

Mô hình được phổ biến nhất và được chấp nhận rộng rãi trong các hệ thống phân tán là mô hình client/server. Trong mô hình này sẽ có một tập các tiến trình mà mỗi tiến trình đóng vai trò như là một trình quản lý tài nguyên cho một tập hợp các tài nguyên cho trước và một tập hợp các tiến trình client trong đó mỗi tiến trình thực hiện một tác vụ nào đó cần truy xuất tới tài nguyên phần cứng hoặc phần mềm dùng chung. Bản thân các trình quản lý tài nguyên cần phải truy xuất tới các tài nguyên dùng chung được quản lý bởi một tiến trình khác, vì vậy một số tiến trình vừa là tiến trình client vừa là tiến trình server.

Các tiến trình phát ra các yêu cầu tới các server bất kỳ khi nào chúng cần truy xuất tới một trong các tài nguyên của các server. Nếu yêu cầu là đúng đắn thì server sẽ thực hiện hành

động được yêu cầu và gửi một đáp ứng trả lời tới tiến trình client. Mô hình client/server cung cấp một cách tiếp cận tổng quát để chia sẻ tài nguyên trong các hệ thống phân tán.

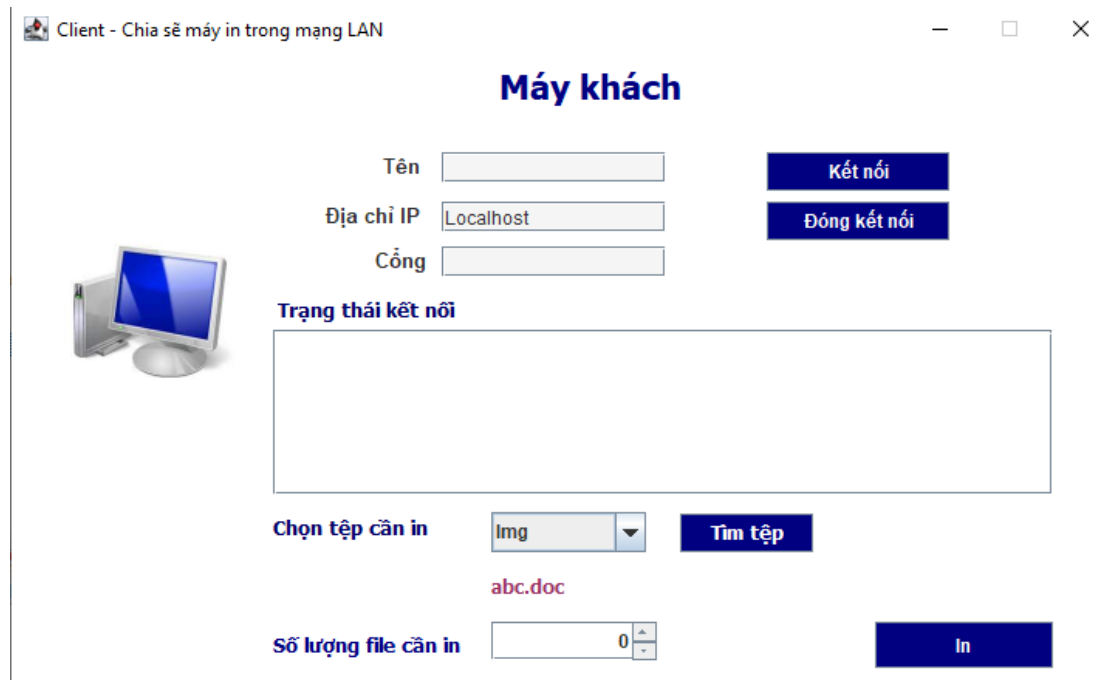


Mô hình này có thể được cài đặt bằng rất nhiều môi trường phần cứng và phần mềm khác nhau. Các máy tính được sử dụng để chạy các tiến trình client/server có nhiều kiểu khác nhau và không cần thiết phải phân biệt giữa chúng; cả tiến trình client và tiến trình server đều có thể chạy trên cùng một máy tính. Một tiến trình server có thể sử dụng dịch vụ của một server khác.

PHẦN 3: NỘI DUNG CHƯƠNG TRÌNH

1. Giao diện chương trình

- *Giao diện Client:*



Client - Chia sẻ máy in trong mạng LAN

Máy khách

Tên

Địa chỉ IP

Cổng

Kết nối

Đóng kết nối

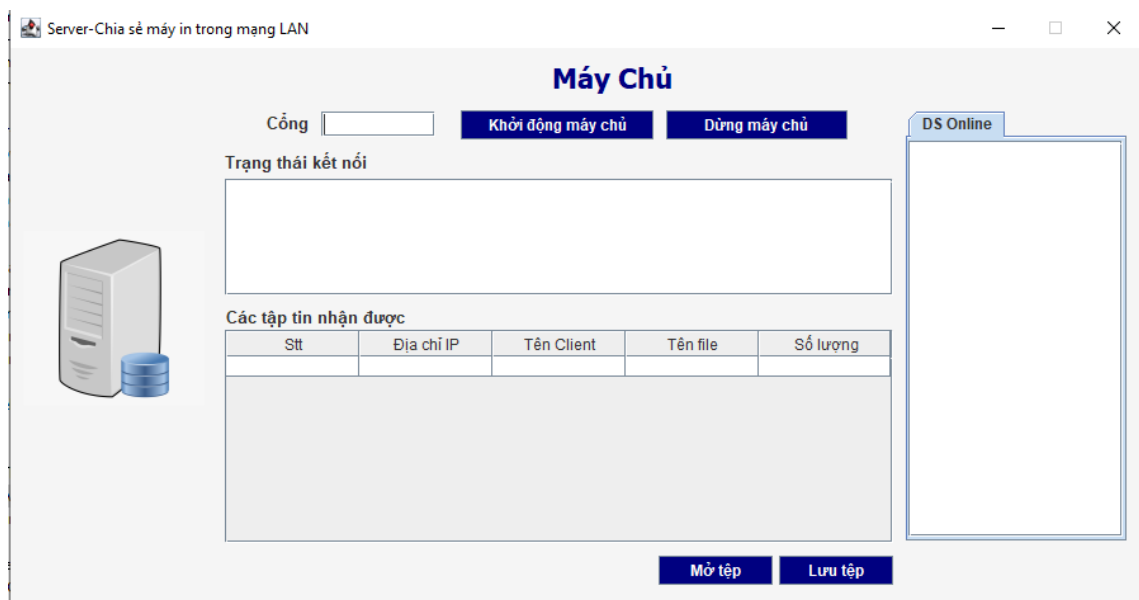
Trạng thái kết nối

Chọn tệp cần in **Tìm tệp**

abc.doc

Số lượng file cần in **In**

- *Giao diện Server:*



Server-Chia sẻ máy in trong mạng LAN

Máy Chủ

Cổng

Khởi động máy chủ **Dừng máy chủ**

Trạng thái kết nối

Các tập tin nhận được

Stt	Địa chỉ IP	Tên Client	Tên file	Số lượng

DS Online

Mở tệp **Lưu tệp**

2. Chức năng của từng lớp

Cần using thêm vào cả Client và Server:

```
using System.Net;
using System.Net.Sockets;
using System.IO;
using System.Threading;
```

2.1. Server

- Lớp StartReceiving() gồm các phương thức xử lý lắng nghe, thiết lập kết nối, nhận tập tin, xuất ra thông báo, cập nhật thanh tiến trình. Phương thức này sau đó được gọi từ một Thread.

```
private void StartReceiving() {
    try {
        if (tlsServer == null) {
            tlsServer = new TcpListener(IPAddress.Parse(txtHost.Text),
Convert.ToInt32(txtPort.Text));
        }
        this.Invoke(new UpdateStatusCallback(this.UpdateStatus),
            new object[] { "Starting the server...\r\n" });
        tlsServer.Start();
        this.Invoke(new UpdateStatusCallback(this.UpdateStatus),
new object[] { "OK! Địa chỉ là: " + txtHost.Text + " cổng: " + txtPort.Text + "\r\n" });
        TcpClient tclServer = tlsServer.AcceptTcpClient();
        this.Invoke(new UpdateStatusCallback(this.UpdateStatus),
            new object[] { "Đã có kết nối tới!\r\n" });
        strRemote = tclServer.GetStream();
        this.Invoke(new UpdateStatusCallback(this.UpdateStatus),
            new object[] { "Đã có luồng Stream!\r\n" });

        int bytesize = 0;
        byte[] downBuffer = new byte[2048];
        bytesize = strRemote.Read(downBuffer, 0, 2048);
        string FileName = System.Text.Encoding.ASCII.GetString(downBuffer, 0,
bytesize);
        FileStream strLocal = new FileStream(@txtDuongDan.Text + "/" + FileName,
FileMode.OpenOrCreate, FileAccess.Write);
        downBuffer = new byte[2048];
        bytesize = strRemote.Read(downBuffer, 0, 2048);
        long FileSize =
Convert.ToInt64(System.Text.Encoding.ASCII.GetString(downBuffer, 0, bytesize));
        this.Invoke(new UpdateStatusCallback(this.UpdateStatus),
            new object[] { "Đang nhận: " + FileName + " (" + FileSize + " bytes)\r\n" });
        downBuffer = new byte[2048];
        while ((bytesize = strRemote.Read(downBuffer, 0, downBuffer.Length)) > 0) {
            strLocal.Write(downBuffer, 0, bytesize);
        }
    }
}
```

```

        this.Invoke(new UpdateProgressCallback(this.UpdateProgress), new object[]
        { strLocal.Length, FileSize });
    }
}
finally {
    this.Invoke(new UpdateStatusCallback(this.UpdateStatus),
        new object[] { "Đã nhận được file. Đang đóng luồng Streams.\r\n" });
    strRemote.Close();
    this.Invoke(new UpdateStatusCallback(this.UpdateStatus),
        new object[] { "Đóng luồng Stream.\r\n" });
    StartReceiving(); } }

```

- Phía Server tạo lắng nghe trên cổng mà người dùng cài đặt:

```

private void btnStart_Click(object sender, EventArgs e)
{
    thrDownload = new Thread(StartReceiving);
    thrDownload.Start();
}

```

- Server cần chọn đường dẫn lưu file khi kích vào “Save as”

2.2. Client

- Client kết nối tới máy chủ Server thông qua địa chỉ IP và cổng của Server. Chỉ có thể kết nối tới Server khi Server đang ở trạng thái lắng nghe.

```

private void ConnectToServer(string ServerIP, int ServerPort)
{
    tcpClient = new TcpClient();
    try
    {
        tcpClient.Connect(ServerIP, ServerPort);
        txtLog.Text += "Kết nối thành công tới Server !!!\r\n";
    }
    catch (Exception exMessage)
    {
        txtLog.Text += exMessage.Message;
    }
}

private void btnConnect_Click(object sender, EventArgs e)
{
    ConnectToServer(txtServer.Text, Convert.ToInt32(txtPort.Text));
}

```

- Sau khi đã kết nối thành công tới Server, Client sẽ tiến hành gửi dữ liệu sang Server

```
private void btnSend_Click(object sender, EventArgs e)
{
    if (tcpClient.Connected == false)
    {
        ConnectToServer(txtServer.Text, Convert.ToInt32(txtPort.Text));
    }
    if (openFile.ShowDialog() == DialogResult.OK)
    {
        txtLog.Text += "Gửi thông tin file.\r\n";
        strRemote = tcpClient.GetStream();
        byte[] byteSend = new byte[tcpClient.ReceiveBufferSize];
        fstFile = new FileStream(openFile.FileName, FileMode.Open,
FileAccess.Read);
        BinaryReader binFile = new BinaryReader(fstFile);
        FileInfo fInfo = new FileInfo(openFile.FileName);
        string FileName = fInfo.Name;
        byte[] ByteFileName = new byte[2048];
        ByteFileName =
System.Text.Encoding.ASCII.GetBytes(FileName.ToCharArray());
        strRemote.Write(ByteFileName, 0, ByteFileName.Length);
        long FileSize = fInfo.Length;
        byte[] ByteFileSize = new byte[2048];
        ByteFileSize =
System.Text.Encoding.ASCII.GetBytes(FileSize.ToString().ToCharArray());
        strRemote.Write(ByteFileSize, 0, ByteFileSize.Length);
        txtLog.Text += "Đang gửi file " + FileName + " (" + FileSize + " bytes)\r\n";
        int bytesize = 0;
        byte[] downBuffer = new byte[2048];
        while ((bytesize = fstFile.Read(downBuffer, 0, downBuffer.Length)) > 0)
        {
            strRemote.Write(downBuffer, 0, bytesize);
        }
        txtLog.Text += "File đã gửi thành công.\r\n";
        tcpClient.Close();
        strRemote.Close();
        fstFile.Close();
        txtLog.Text += "Đóng luồng Stream !!!\r\n";
    }
}
```

- Sau khi tập tin được gửi, các kết nối được đóng lại để giải phóng tài nguyên. Để chuẩn bị cho một kết nối sắp tới, Server sẽ bắt đầu lại từ đầu.

2.3. Demo chương trình