**LAB 2**

1. Mục đích ................................................................................................................................................ 2
2. Yêu cầu ................................................................................................................................................. 2
3. Hướng dẫn ........................................................................................................................................... 2
   1. Tạo một chương trình server đơn giản ......................................................................................... 2
      1. Hướng dẫn lập trình ............................................................................................................... 2
      2. Test chương trình: .................................................................................................................. 3
      3. Trả lời câu hỏi: ....................................................................................................................... 4
   2. Tạo chương trình Server đơn giản, khi Client kết nối vào thì gởi câu chào cho client ................ 5
      1. Hướng dẫn lập trình ............................................................................................................... 5
      2. Test chương trình ................................................................................................................... 5
      3. Trả lời câu hỏi: ....................................................................................................................... 7
   3. Xây dựng chương trình client đơn giản ........................................................................................ 7
      1. Hướng dẫn lập trình ............................................................................................................... 7
      2. Test chương trình ................................................................................................................... 8
      3. Trả lời câu hỏi: ....................................................................................................................... 9
   4. Cải tiến chương trình client ........................................................................................................ 11
      1. Hướng dẫn lập trình ............................................................................................................. 11
      2. Test chương trình ................................................................................................................. 11
   5. Thêm khả năng gởi dữ liệu lên server và nhận lại dữ liệu từ server gởi về ............................... 12
      1. Hướng dẫn lập trình ............................................................................................................. 12
   6. Thêm khả năng nhận dữ liệu từ client gởi lên và gởi trả dữ liệu ngược lại client ...................... 12
      1. Hướng dẫn lập trình ............................................................................................................. 12
      2. Test chương trình ................................................................................................................. 12
      3. Trả lời câu hỏi: ..................................................................................................................... 14
4. Bài tập ............................................................................................................................................... 15

# I. Mục đích

Ban đầu làm quen với lập trình Socket

Hiểu được cách hoạt động của bộ đệm hệ thống và bộ đệm chương trình

Xây dựng chương trình client-server đơn giản

**II. Yêu cầu**

Xây dựng các chương trình client-server từ mức đơn giản nhất rồi phức tạp dần

# III. Hướng dẫn

## III.1. Tạo một chương trình server đơn giản

### III.1.1. Hướng dẫn lập trình

Để lập trình socket ta sử dụng 2 namespace:

using System.Net;

using System.Net.Sockets;

Tạo Server EndPoint, EndPoint này sẽ tham chiếu đến địa chỉ IP và Port của Server:

IPEndPoint serverEndPoint = new IPEndPoint(IPAddress.Any, 5000);

Tạo Server Socket, Socket này sẽ được kết nối với Server EndPoint:

Socket serverSocket = new Socket(AddressFamily.InterNetwork, SocketType.Stream,

ProtocolType.Tcp);

Lắng nghe kết nối trên Server Socket, khi có client kết nối tới sẽ trả về Socket để trao đổi dữ liệu với client:

serverSocket.Listen(10);

Socket clientSocket = serverSocket.Accept();

Khi client kết nối tới trên server sẽ xuất thông tin của client kết nối tới (địa chỉ IP, Port)

EndPoint clientEndPoint = clientSocket.RemoteEndPoint; Console.WriteLine(clientEndPoint.ToString());

**III.1.2. Test chương trình:**

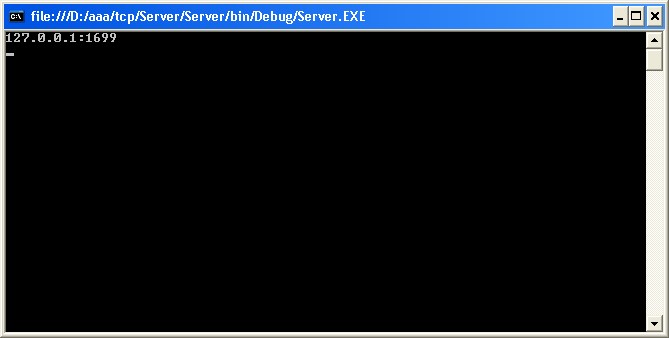
Chạy chương trình Server:



Chạy chương trình CMD và telnet vào server trên với địa chỉ IP loopback và port 5000:



Kết quả trả về trên Server:



A computer screen with text and images

Description automatically generated

**III.1.3. Trả lời câu hỏi:**

1. Khi telnet vào Server bằng port 5000 tại sao port client dùng để kết nối vào lại không phải là port 5000 ?

Vì giao thức TCP sẽ sử dụng 2 cặp port destination/source và ip destination/source vì vậy cổng kết nối sẽ khác nhau khi truy cập vào

Port của Server (Server Port): Đây là port mà server đang lắng nghe các kết nối đến. Trong trường hợp của bạn, đây là port 5000.

Port của Client (Client Port): Đây là port trên máy client mà kết nối sẽ được thiết lập. Port này được chọn ngẫu nhiên bởi hệ điều hành của máy client từ dải port động.

Port của client không phải là port 5000 vì giao thức TCP cần phải phân biệt giữa các kết nối khác nhau. Port 5000 là port mà server lắng nghe các kết nối đến, còn port của client là một port ngẫu nhiên mà hệ điều hành chọn để gửi dữ liệu đến server. Điều này đảm bảo rằng các kết nối từ cùng một client đến cùng một server có thể được phân biệt và xử lý đúng cách.

1. Như hình trên, giải thích vì sao có client dùng port giá trị 1699 ?

Port 1699 được chọn vì nó là một port ngẫu nhiên nằm trong dải port động mà hệ điều hành của client đã chọn để thiết lập kết nối TCP. Việc chọn port này giúp phân biệt các kết nối khác nhau từ cùng một client đến cùng một server và đảm bảo rằng các kết nối được quản lý một cách hiệu quả mà không gây xung đột.

1. Có phải lúc nào client cũng mở port 1699 để kết nối với Server không ?

Client không giữ port 1699 mở liên tục. Port này chỉ được sử dụng cho một kết nối cụ thể tại thời điểm thiết lập kết nối. Sau khi kết nối kết thúc, port sẽ được giải phóng và có thể được sử dụng cho các kết nối khác hoặc không sử dụng nữa. Port được chọn cho mỗi kết nối là ngẫu nhiên và không cố định, giúp hệ điều hành quản lý kết nối TCP một cách hiệu quả và bảo mật.

1. Ta có thể chạy 2 chương trình CMD để telnet vào Server trên được không ? Giải thích.

Có thể chạy hai hoặc nhiều cửa sổ CMD để thực hiện kết nối Telnet đồng thời đến cùng một server. Mỗi kết nối sẽ có một port nguồn khác nhau và sẽ không xung đột với các kết nối khác, nhờ vào cách mà hệ điều hành quản lý các port và kết nối TCP.

## III.2. Tạo chương trình Server đơn giản, khi Client kết nối vào thì gởi câu chào cho client

### III.2.1. Hướng dẫn lập trình

Sử dụng chương trình Server đơn giản ở trên, thêm phần gởi dữ liệu xuống client

byte[] buff;

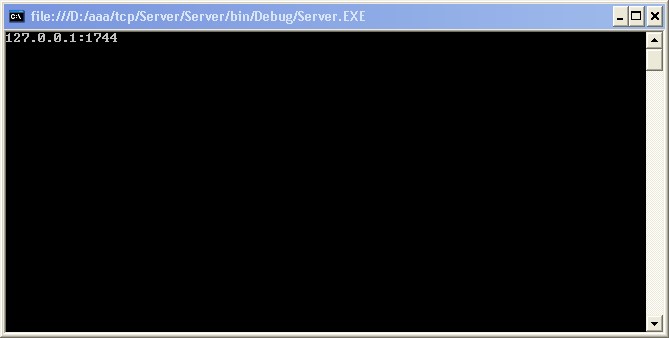
string hello = "Hello Client"; buff = Encoding.ASCII.GetBytes(hello); clientSocket.Send(buff, 0, buff.Length, SocketFlags.None); **III.2.2. Test chương trình** Chạy chương trình Server:



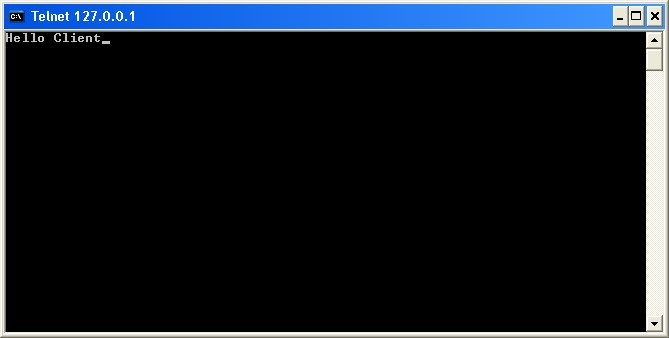
Chạy chương trình CMD và telnet vào server trên với địa chỉ IP loopback và port 5000:



Kết quả trả về trên Server:



Kết quả trả về trên client :



A screen shot of a computer

Description automatically generated

**III.2.3. Trả lời câu hỏi:**

1. Tạo sao trong đoạn code viết thêm vào ta không cần phải khởi tạo độ lớn ban đầu cho buffer ?

Việc không cần phải khởi tạo độ lớn ban đầu cho buffer thường là nhờ vào tính năng của ngôn ngữ lập trình hoặc cấu trúc dữ liệu mà bạn đang sử dụng. Các ngôn ngữ lập trình hiện đại và các cấu trúc dữ liệu động giúp bạn dễ dàng quản lý bộ nhớ mà không cần phải xác định kích thước ban đầu, từ đó tăng tính linh hoạt và đơn giản hóa việc lập trình.

1. Giải thích ý nghĩa các tham số của phương thức clientSocket.Send .

Phương thức Send gửi dữ liệu từ mảng buff thông qua kết nối socket.

Tham số buff là mảng byte chứa dữ liệu.

Tham số 0 chỉ định vị trí bắt đầu trong mảng để gửi dữ liệu.

Tham số buff.Length xác định số byte cần gửi.

Tham số SocketFlags.None chỉ định không có cờ điều khiển đặc biệt.

## III.3. Xây dựng chương trình client đơn giản

### III.3.1. Hướng dẫn lập trình

Để lập trình socket ta using 2 namespace:

using System.Net;

using System.Net.Sockets;

Tạo Server EndPoint, EndPoint này sẽ tham chiếu đến địa chỉ IP và Port của Server:

IPEndPoint serverEndPoint = new IPEndPoint(IPAddress.Loopback, 5000);

Tạo socket dùng để kết nối với server:

Socket serverSocket = new Socket(AddressFamily.InterNetwork, SocketType.Stream,

ProtocolType.Tcp);

Console.WriteLine("Dang ket noi voi server..."); serverSocket.Connect(serverEndPoint);

Nếu kết nối thành công, nhận câu chào từ server và hiển thị câu chào lên màn hình:

if (serverSocket.Connected)

{

Console.WriteLine("Ket noi thanh cong voi server ...");

byteReceive = serverSocket.Receive(buff, 0, buff.Length, SocketFlags.None);

str = Encoding.ASCII.GetString(buff, 0, byteReceive)

Console.WriteLine(str);

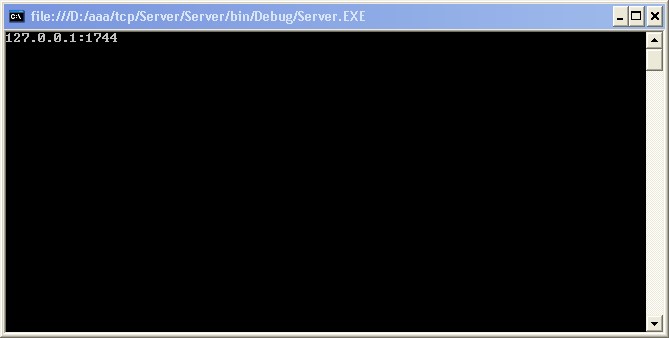
}

**III.3.2. Test chương trình**

Chạy chương trình Server:

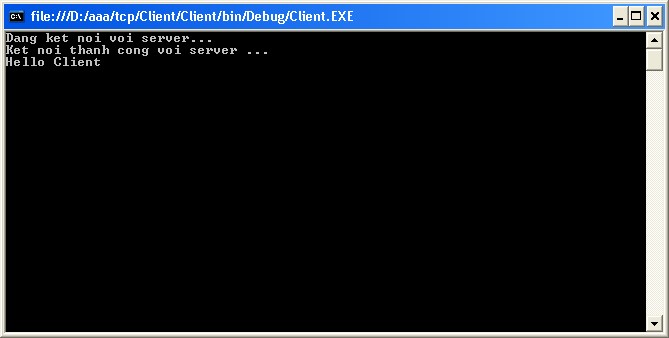


Kết quả trả về trên Server:



Kết quả trả về trên client :

A screenshot of a computer

Description automatically generated

**III.3.3. Trả lời câu hỏi:**

1. Khi client kết nối đến server mà server chưa chạy hoạc là đường truyền mạng bị rớt thì có hiện tượng gì xảy ra ?, giải thích.

 **Không thể truy cập dịch vụ:** Client sẽ không thể truy cập được bất kỳ dịch vụ nào mà server cung cấp. Ví dụ:

* **Website:** Trang web sẽ không mở được, hiển thị thông báo lỗi như "Không thể kết nối đến máy chủ", "Trang web này không khả dụng" hoặc thông báo tương tự.
* **Ứng dụng:** Ứng dụng sẽ báo lỗi kết nối, không thể đăng nhập hoặc thực hiện các tác vụ yêu cầu kết nối đến server.

 **Thời gian chờ kết nối quá lâu:** Client sẽ chờ đợi trong một khoảng thời gian nhất định để thiết lập kết nối, nhưng cuối cùng sẽ nhận được thông báo về việc kết nối thất bại.

 **Lỗi timeout:** Nếu client chờ quá lâu mà không nhận được phản hồi từ server, sẽ xuất hiện lỗi timeout (hết thời gian chờ).

 **Lỗi DNS:** Trong một số trường hợp, client có thể không giải quyết được địa chỉ IP của server, dẫn đến lỗi DNS.

 **Kết nối không ổn định:** Nếu đường truyền mạng bị gián đoạn từng lúc, client có thể kết nối được một lúc rồi lại bị ngắt kết nối.

1. Từ đoạn code ở chương trình client: str = Encoding.ASCII.GetString(buff, 0, byteReceive)

Nếu ta thay byteReceive bằng buff.Lengh thì kết quả sẽ như thế nào? Giải thích.

 Việc thay byteReceive bằng buff.Length thường không được khuyến khích.

 byteReceive nên được xác định chính xác bằng số lượng byte dữ liệu thực sự đã nhận được từ server.

 Điều này giúp đảm bảo rằng chỉ có dữ liệu hợp lệ được chuyển đổi thành chuỗi và tránh các vấn đề không mong muốn.

## III.4. Cải tiến chương trình client

### III.4.1. Hướng dẫn lập trình

Chương trình client này khi kết nối đến server mà server chưa bật hoạc đường truyền bị lỗi thì chương trình sẽ bị lỗi.

Sửa lại chương trình bằng cách thêm try-catch vào:

try {

serverSocket.Connect(serverEndPoint);

}

catch (SocketException se)

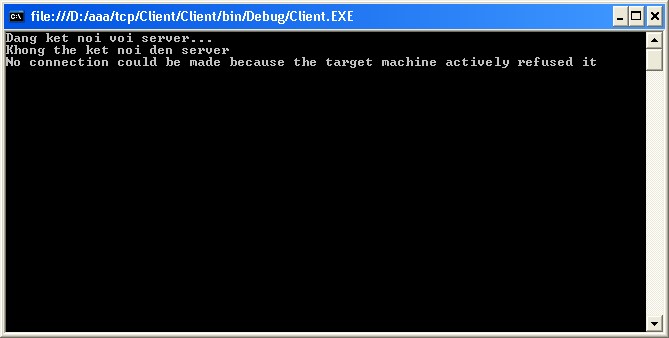
{

Console.WriteLine("Khong the ket noi den server"); return;

}

### III.4.2. Test chương trình

Chạy chương trình ở client mà không chạy server:



A screenshot of a computer program

Description automatically generated

## III.5. Thêm khả năng gởi dữ liệu lên server và nhận lại dữ liệu từ server gởi về

### III.5.1. Hướng dẫn lập trình

Việc này được thực hiện bằng cách thêm vòng lặp vô hạn

while (true)

{

str = Console.ReadLine(); buff = Encoding.ASCII.GetBytes(str);

serverSocket.Send(buff, 0, buff.Length, SocketFlags.None); buff = new byte[1024];

byteReceive = serverSocket.Receive(buff, 0, buff.Length, SocketFlags.None);

str = Encoding.ASCII.GetString(buff, 0, byteReceive);

Console.WriteLine(str); }

## III.6. Thêm khả năng nhận dữ liệu từ client gởi lên và gởi trả dữ liệu ngược lại client

### III.6.1. Hướng dẫn lập trình

Việc này được thực hiện bằng cách thêm vòng lặp vô hạn

while (true)

{

buff = new byte[1024];

byteReceive = clientSocket.Receive(buff, 0, buff.Length, SocketFlags.None);

str = Encoding.ASCII.GetString(buff, 0, byteReceive); Console.WriteLine(str);

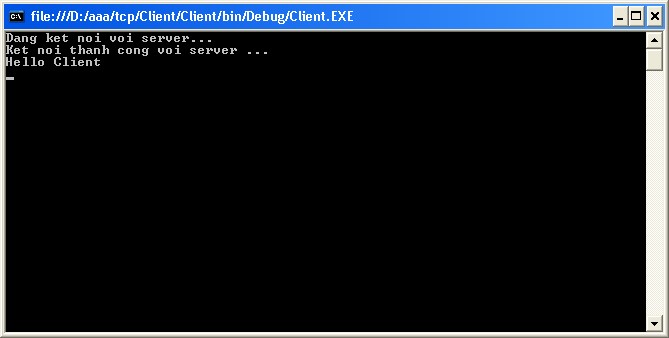
clientSocket.Send(buff, 0, byteReceive, SocketFlags.None); }

### III.6.2. Test chương trình

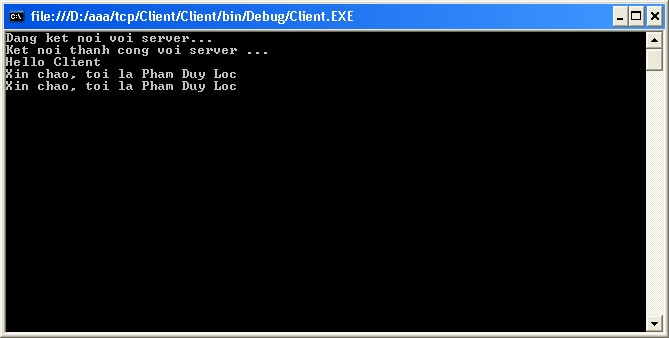
Chạy chương trình server:



Chạy chương trình client:



Nhập dữ liệu và gởi lên server:



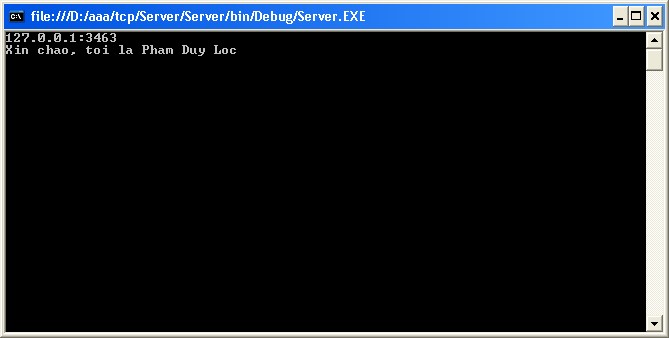
A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

Kết quả hiển thị trên server:



**III.6.3. Trả lời câu hỏi:**

1. Khi thay đổi kích thước bộ đệm từ 1024 xuống 10 thì có hiện tượng gì xảy ra ?. Giải thích

Việc thay đổi kích thước bộ đệm có thể ảnh hưởng đáng kể đến hiệu suất và độ ổn định của ứng dụng. Việc lựa chọn kích thước bộ đệm phù hợp phụ thuộc vào nhiều yếu tố, bao gồm:

* **Kích thước dữ liệu cần xử lý**
* **Tốc độ xử lý của CPU và đĩa**
* **Các thuật toán sử dụng**
* **Các yêu cầu về hiệu suất và độ ổn định của ứng dụng**

1. Nếu ta không reset lại bộ đệm thì có hiện tượng gì xảy ra? Giải thích.

**Không reset bộ đệm** có nghĩa là dữ liệu cũ trong bộ đệm sẽ không được xóa đi trước khi ghi dữ liệu mới vào. Điều này có thể dẫn đến một số vấn đề sau:

### 1. ****Ô nhiễm dữ liệu:****

* **Dữ liệu cũ lẫn lộn với dữ liệu mới:** Nếu không xóa bỏ dữ liệu cũ, khi ghi dữ liệu mới vào bộ đệm, dữ liệu cũ có thể bị lẫn lộn với dữ liệu mới, dẫn đến kết quả không chính xác.
* **Khó khăn trong việc xác định dữ liệu:** Việc phân biệt đâu là dữ liệu cũ, đâu là dữ liệu mới trở nên khó khăn, gây ra lỗi khó phát hiện.

### 2. ****Tràn bộ đệm:****

* **Bộ đệm đầy:** Nếu liên tục ghi dữ liệu vào bộ đệm mà không xóa dữ liệu cũ, bộ đệm sẽ nhanh chóng bị đầy.
* **Mất dữ liệu:** Khi bộ đệm đầy, dữ liệu mới ghi vào có thể ghi đè lên dữ liệu cũ, dẫn đến mất mát dữ liệu.
* **Lỗi chương trình:** Tràn bộ đệm có thể gây ra các lỗi nghiêm trọng trong chương trình, thậm chí dẫn đến crash.

### 3. ****Giảm hiệu suất:****

* **Tìm kiếm dữ liệu lâu hơn:** Nếu cần tìm kiếm một phần tử trong bộ đệm, việc có quá nhiều dữ liệu cũ sẽ làm chậm quá trình tìm kiếm.
* **Tăng tiêu tốn bộ nhớ:** Bộ đệm luôn chiếm một lượng bộ nhớ nhất định. Nếu không reset, bộ đệm sẽ luôn chiếm dụng bộ nhớ, gây ảnh hưởng đến các quá trình khác.

### 4. ****Rò rỉ bộ nhớ:****

* **Bộ nhớ không được giải phóng:** Trong một số trường hợp, nếu bộ đệm không được quản lý đúng cách, bộ nhớ cấp phát cho bộ đệm có thể không được giải phóng, dẫn đến rò rỉ bộ nhớ.

# IV. Bài tập

1. Chương trình trên khi tắt client thì trên server sẽ bị lỗi, hãy cải tiến lại chương để khi client tắt đột ngột trên server sẽ thông báo client đã ngắt kết nối

Dựa vào mô hình đã có em sẽ bổ sung them đoạn code sau

Thêm đoạn try catch vào để bắt các lỗi ngoại lệ

while (true)

{

try

{

buff = new byte[1024];

byteReceive = clientSocket.Receive(buff, 0, buff.Length, SocketFlags.None);

str = Encoding.ASCII.GetString(buff, 0, byteReceive); Console.WriteLine(str);

clientSocket.Send(buff, 0, byteReceive, SocketFlags.None);

}

catch (SocketException ex)

{

if (ex.SocketErrorCode == SocketError.ConnectionReset)

{

Console.WriteLine("Client đã ngắt kết nối!");

break; // Thoát khỏi vòng lặp

}

else

{

Console.WriteLine("Lỗi kết nối khác: " + ex.Message);

}

}

}

Với đoạn code bổ sung này, khi client ngắt kết nối đột ngột, server sẽ in ra thông báo và thoát khỏi vòng lặp, giúp chương trình chạy ổn định hơn.

**SocketError.ConnectionReset:** Kiểm tra xem lỗi có phải do client ngắt kết nối đột ngột không (thường xảy ra khi client đóng socket).

A screenshot of a computer

Description automatically generated

1. Cải tiến chương trình: khi client nhập chữ “exit” thì sẽ tắt client, trên server thông báo client đã ngắt kết nối

Nếu vậy thì Ái Mừng sẽ thêm điều kiện này vào server lồng vào vòng while

if (str.ToLower() == "exit")

{

Console.WriteLine("Client đã gửi lệnh thoát!");

break;

}

serverSocket.Close();

Bên client thì ta bổ sung thêm đoạn sau:

while (true)

{

str = Console.ReadLine();

if (str.ToLower()=="exit")

{

buff = Encoding.ASCII.GetBytes(str);

serverSocket.Send(buff, 0, buff.Length, SocketFlags.None);

break;

}

buff = Encoding.ASCII.GetBytes(str);

serverSocket.Send(buff, 0, buff.Length, SocketFlags.None); buff = new byte[1024];

byteReceive = serverSocket.Receive(buff, 0, buff.Length, SocketFlags.None);

str = Encoding.ASCII.GetString(buff, 0, byteReceive);

Console.WriteLine(str);

}

serverSocket.Close(); // Explicitly close the socket

Console.WriteLine("Da ngat ket noi voi server!");

A screenshot of a computer

Description automatically generated

1. Viết chương trình thực hiện các phép tính số học, việc thực hiện tính toán được thực hiện trên server, phía client sẽ nhập vào các toán hạng và các phép tính