LỜI CẢM ƠN

Em xin được bày tỏ lòng biết ơn sâu sắc tới Ths.Đỗ Văn Chiểu giảng viên trường Đại học dân lập Hải Phòng đã tận tình hướng dẫn và tạo mọi điều kiện thuận lợi để em hoàn thành bài báo cáo tốt nghiệp của mình.

Em xin chân thành cảm ơn tất cả các thầy, cô giáo khoa Công nghệ thông tin trường Đại học dân lập Hải Phòng đã nhiệt tình giảng dạy và cung cấp những kiến thức quý báu để em có thể hoàn thành tốt luận văn tốt nghiệp này.

Cuối cùng, em xin cảm ơn tất cả các bạn đã động viên, góp ý và trao đổi hỗ trợ cho em trong suốt thời gian vừa qua.

Vì thời gian tìm hiểu luận văn có hạn, trình độ bản thân còn nhiều hạn chế. Cho nên trong đề tài khó tránh khỏi những thiếu sót, em rất mong nhận được sự đóng góp ý kiến quý báu của các thầy cô giáo cũng như các bạn để đề tài của em được hoàn thiện hơn.

Em xin chân thành cảm ơn!

Hải Phòng, Tháng 10 năm 2010

Sinh viên thực hiện

Đặng Bá Hậu

MỤC LỤC

LỜI NÓI ĐẦU

Trong thời đại ngày này, công nghệ thông tin đóng vài trò quan trọng hầu như trong tất cả các lĩnh vực. Do vậy con người phải không ngừng học tập để mở mang, trao dồi kiến thức. Khi mạng internet xuất hiện, nhu cầu trao đổi thông tin ngày càng cao, nhu cầu học hỏi kiến thức không chỉ gói gọn trong nhà trường, hoặc trong lớp học., giờ đây với máy vi tính cùng với mạng internet, chúng ta có thể tham gia vào các lớp học trực tuyến

Có nhiều website hỗ trợ việc học trực tuyến nhưng giá thành mắc, không hỗ trợ người học tập tham gia trực tiếp vào lớp học. Trong những năm trước đây, các dịch vụ truyền thông đa phương tiện đều rất khó thực hiện bởi ít có sự hỗ trợ về phần cứng, đặc biệt băng thông chính là điều khó khăn nhất trong việc truyền tín hiệu âm thanh, và hình ảnh. Tuy nhiên, với kỹ thuật phát triển hiện nay, các tín hiệu âm thanh và hình ảnh có thể được nén lại một cách dễ dàng, tiết kiệm được băng thông. Do vậy em chọn đề tài "**Tìm hiểu kĩ thuật multicast xây dựng ứng dụng hỗ trợ giảng dạy trên mạng LAN** " nhằm xây dựng một hệ thống đào tạo từ xa, có hỗ trợ âm thanh và hình ảnh để giúp cho giáo viên có thể giáo tiếp trực tiếp với sinh viên giúp cho học viên có thể tiếp thu bài tốt hơn

Mục tiêu của đề tài :

Ở nước ta hiện nay, hình thức đào tạo thông dụng là học viên trực tiếp trên truyền hình, các bài giảng được các giáo viên thu lại và phát trên truyền hình vào một thời điểm nhất định. Hình thức này giúp cho học viên có thể tiếp thu bài tốt hơn nhưng lại thiếu sự giao tiếp trực tiếp với giáo viên. Do vậy, chúng em đã nghiên cứu và được sự chỉ bảo tận tình của các thầy,cô và các bạn tìm hiểu các phương tiện đa truyền thông hiện nay để tạo ra một hệ thống giúp cho việc dạy học trực tuyến, giao tiếp giữa học viên và giáo viên được tốt hơn.

Trong đồ án này, em đi sâu vào giải quyết bài toán " Tìm hiểu kỹ thuật multicast xây dựng ứng dụng giảng dạy trên mạng LAN" nội dung của đồ án được bao quát trong ba chương như sau :

Chương 1: Trình bày các kiến thức cơ bản về mạng máy tính.

Chương 2: Trình bày các kiến thức cơ bản về lập trình C#

Chương 3: xây dựng chương trình thử nghiệm

CHƯƠNG 1: CÁC KIẾN THỨC CƠ BẢN VỀ MẠNG MÁY TÍNH

❖ **Giới thiệu :**

IP multicast là một sự mở rộng của IP. Tổ chức IETF đưa ra khuyến nghị RFC 1112, định nghĩa các thành phần mở rộng cho IP. Một hướng đi mới cho IP, IP Multicast là giao thức dùng để truyền gói tin IP từ một nguồn đến nhiều đích đến khác nhau trong mạng LAN hay WAN. Nhóm những thành viên muốn nhận thông tin này thì phải tham gia vào một nhóm multicast. Với IP multicast, ứng dụng gửi một bản sao của thông tin đến một nhóm. Thông tin này đến tất cả những người nào muốn nhận nó.

Kĩ thuật Multicast đánh địa chỉ các gói là địa chỉ nhóm thay vì địa chỉ của từng người nhận; Các gói tin này phụ thuộc vào các mạng chuyển tiếp để chuyển đến mạng cần nhận nó. Một nút có khả năng - Multicast chạy giao thức TCP/IP có thể nhận được thông điệp multicast. Multicast là kĩ thuật đẩy thông tin, trong đó một máy chủ sẽ gửi dữ liệu đến người sử dụng mà không cần người sử dụng phải yêu cầu trước.

1.1. Mô hình tham khảo 7 tầng OSI

Mô hình kết nối hệ thống mở được Tổ chức quốc tế về tiêu chuẩn hoá ISO (International Organizaiton for Standardization) đưa ra nhằm cung cấp một mô hình chuẩn cho các nhà sản xuất và cung cấp sản phẩm viễn thông áp dụng theo để phát triển các sản phẩm viễn thông. Ý tưởng mô hình hoá được tạo ra còn nhằm hỗ trợ cho việc kết nối giữa các hệ thống và modun hoá các thành phần phục vụ mạng viến thông.

a. Chức năng của mô hình OSI:

- Cung cấp kiến thức về hoạt động của kết nối liên mạng

- Đưa ra trình tự công việc để thiết lập và thực hiện một giao thức cho kết nối các thiết bị trên mạng.Mô hình OSI còn có một số thuận lợi sau :

+ Chia nhỏ các hoạt động phức tạp của mạng thành các phần công việc đơn giản.

+ Cho phép các nhà thiết kế có khả năng phát triển trên từng modun chức năng.

+ Cung cấp các khả năng định nghĩa các chuẩn giao tiếp có tính tương thích cao“plug and play” và tích hợp nhiều nhà cung cấp sản phẩm.

b. Cấu trúc mô hình OSI:

Mô hình OSI gồm 7 lớp (level), mỗi lớp thực hiện các chức năng riêng cho hoạt động kết nối mạng.

Hình 1-1 Mô tả 7 lớp OSI. 4 lớp đầu định nghĩa cách thức cho đầu cuối thiết lập kết nối với nhau để trao đổi dữ liệu. 3 lớp trên dùng để phát triển các ứng dụng để đầu cuối kết nối với nhau và người dùng.

<table>
<tr><td rowspan="3">Application</td><td>Aplication</td><td rowspan="3"></td></tr>
<tr><td>Presentation</td></tr>
<tr><td>Session</td></tr>
<tr><td rowspan="4"></td><td>Transport Layer</td><td rowspan="4">Data Lower Layer</td></tr>
<tr><td>Network Layer</td></tr>
<tr><td>Data Link</td></tr>
<tr><td>Physical</td></tr>
</table>

3lớp trên cùng của mô hình OSI thường được gọi là các lớp ứng dụng (Application layers) hay còn gọi là các lớp cao. Các lớp này thường liên quan tới giao tiếp với người dùng, định dạng của dữ liệu và phương thức truy nhập các ứng dụng đó.

Hình 1-2 Mô tả các lớp trên và cung cấp thông tin với các chức năng của nó qua ví dụ sau:

<table>
<tr><td>Application</td><td>- lớp ứng dụng: Chức năng giao tiếp giữa người sử dụng và các chương trình ứng dụng</td><td>Telnet, HTTP</td></tr>
<tr><td>Presentation</td><td>- lớp trình bày: Cách thức chuẩn hóa dữ liệu và trình bày số liệu
- Có chức năng đặc biệt là mã hóa dữ liệu người sử dụng</td><td>ASSCII
EBCDIC
JPEC</td></tr>
<tr><td>Session</td><td>- Lớp phiên: thiết lập duy trì và hủy bỏ một phiên làm việc</td><td>NFS, SQL</td></tr>
<tr><td>Transport Layer</td><td colspan="2" rowspan="4"></td></tr>
<tr><td>Network Layer</td></tr>
<tr><td>Data Link</td></tr>
<tr><td>Physical</td></tr>
</table>

- **Application layer :** đây là lớp cao nhất trong mô hình. Nó là nơi mà người sử dụng hoặc kết nối các chương trình ứng dụng với các thủ tục cho phép truy nhập vào mạng.
- **Presentation layer :** Lớp presentation cung cấp các mã và chức năng để chuyển đổi mà được cung cấp bởi lớp ứng dụng. Các chức năng đó đảm bảo rằng dữ liệu từ lớp ứng dụng trong một hệ thống có thể được đọc bởi lớp ứng dụng của một hệ thống khác. VD : dùng để mã hoá dữ liệu từ lớp ứng dụng : như mã hoá ảnh jpeg , gif. Mã đó cho phép ta có thể hiện lên trang web .
- **Session layer :** được sử dụng để thiết lập, duy trì và kết thúc phiên làm việc giữa các lớp presentation. Việc trao đổi thông tin ở lớp này bao gồm yêu cầu dịch vụ và đáp ứng yêu cầu của các ứng dụng trên thiết bị khác.Các lớp dưới :

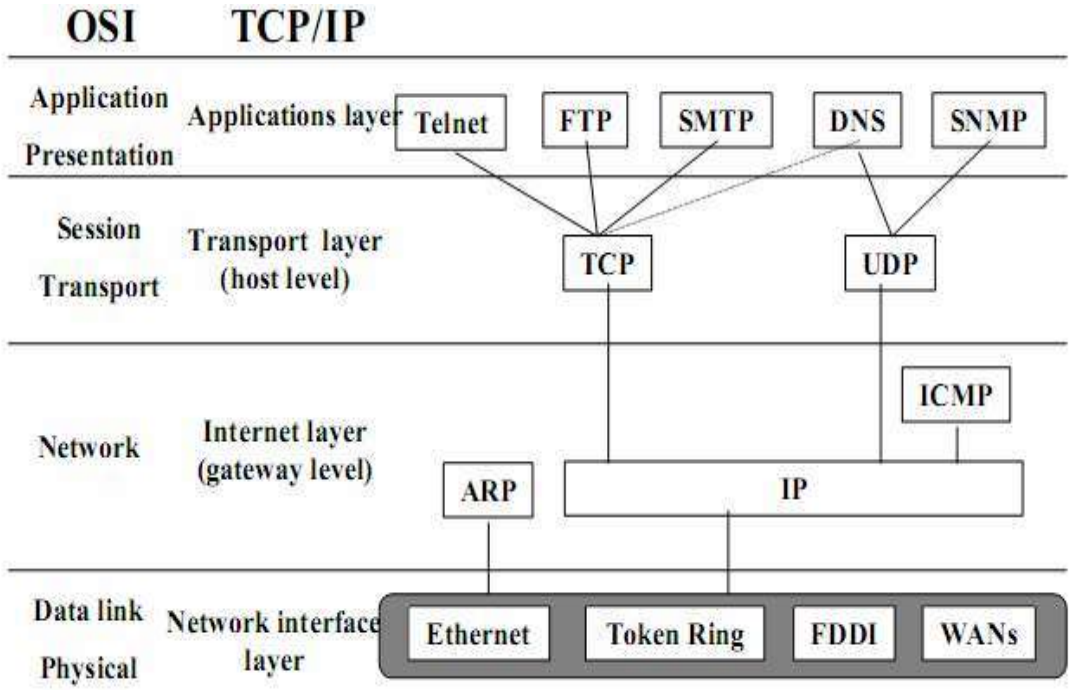
Bốn lớp dưới của mô hình OSI sử dụng để định nghĩa làm thế nào để dữ liệu được truyền đi trong các dây nối vật lý, các thiết bị mạng và đi đến trạm đầu cuối cuối cùng là đến các lớp ứng dụng. Quấn sách này ta chỉ quan tâm đến 4 lớp cuối. Và sẽ xem xét từng lớp một cách chi tiết giao thiếp giữa các lớp trong mô hình OSI:

Sử dụng phương pháp protocal stack để kết nối giữa hai thiết bị trong mạng. Protocal stack là một tập hợp các quy định dùng để định nghĩa làm thế nào để dữ liệu truyền qua mạng.Ví dụ với : TCP/IP mỗi Layer cho phép dữ liệu truyền qua. Các lớp đó trao đổi các thông tin để cung cấp cuộc liên lạc giữa hai thiết bị trong mạng. Các lớp giao tiếp với nhau sử dụng Protocal Data Unit (PDU). Thông tin điểu khiển của PDU được thêm vào với dữ liệu ở lớp trên. Và thông tin điều khiển này nằm trong trường gọi là trường header và traile

			Application	
			Presentation	
	Upper Layer Data		Session	
			Transport	segment
TCP Header	Upper Layer Data			
IP Header	Data		Network Layer	packet
LLC Header	Data	FCS	Data Link	Frame
		FCS		
01011101010010000010				

1.2. Họ giao thức TCP/IP

Hình 1-3 Data encapsulation



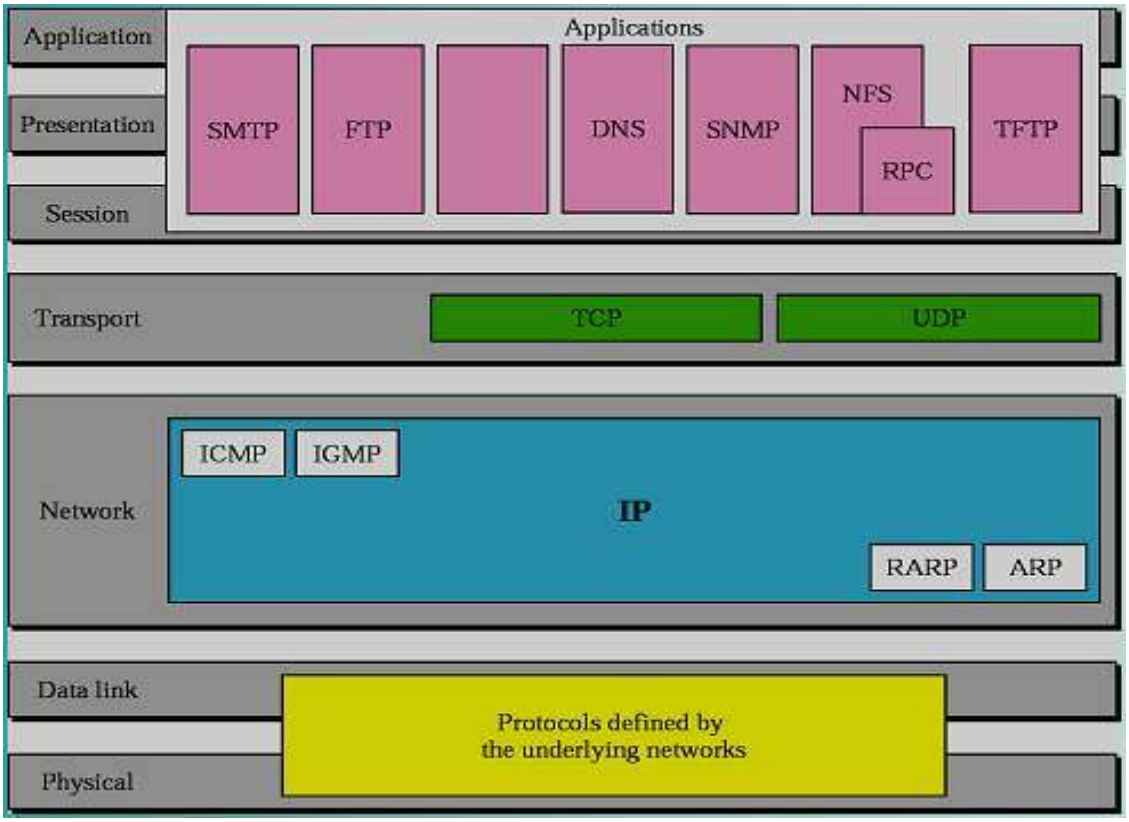
Các tầng của giao thức TCP/IP so với cấc tầng của mô hình OSI

Application: Xác nhận quyền, nén dữ liệu và các dịch vụ cho người dùng

Transport: Xử lý dữ liệu giữa các hệ thống va cung cấp việc truy cập mạng cho các ứng dụng

Network: Tìm đường cho các packet

Link: Mức OS hoặc các thiết bị giao tiếp mạng trên một máy tính



Một số điểm khác nhau của TCP/IP và mô hình OSI

+ Lớp ứng dụng trong TCP/IP xử lý chức năng của lớp 5,6,7 trong mô hình OSI

+ Lớp transport trong TCP/IP cung cấp cớ chế UDP truyền không tin cậy, transport trong OSI luôn đảm bảo truyền tin cậy

+ TCP/IP là một tập của các protocols (một bộ giao thức)

+ TCP/IP xây dựng trước OSI

Quy trình đóng gói dữ liệu trong mô hình TCP/IP như sau:

1.3. So sánh giữa hai giao thức TCP và UDP

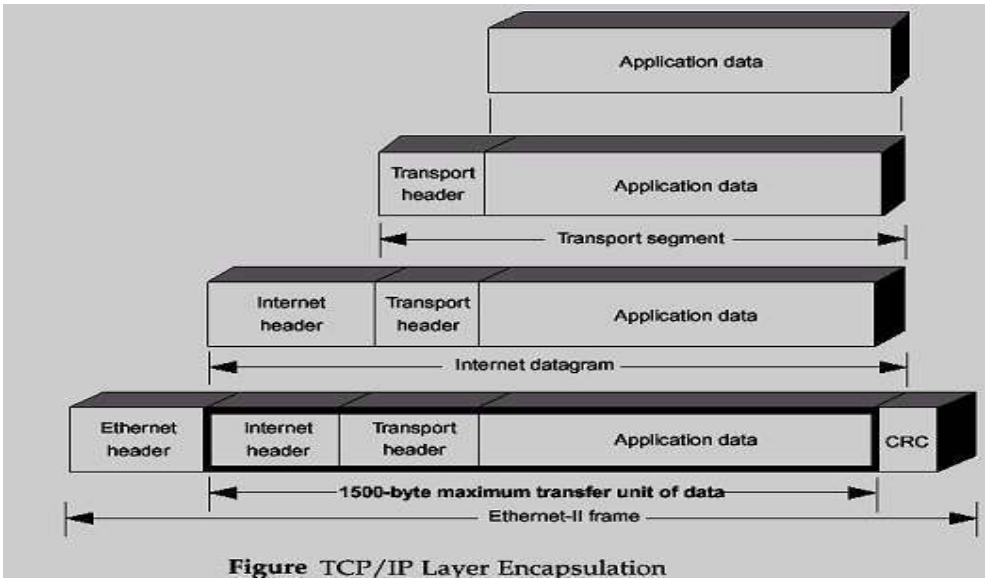
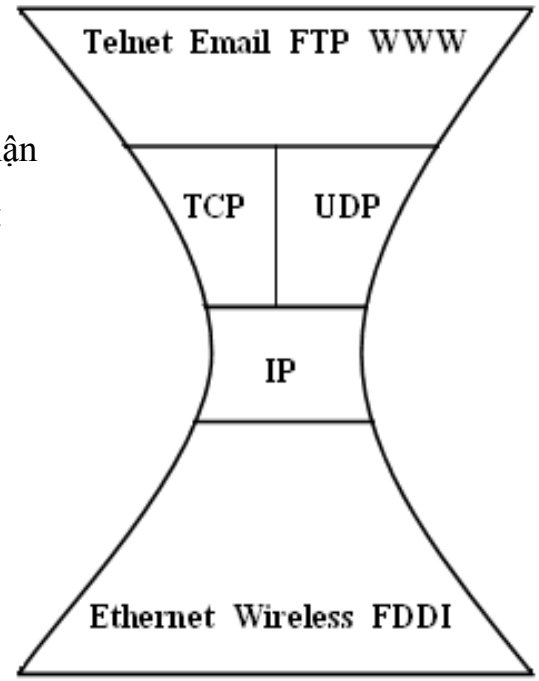


Figure TCP/IP Layer Encapsulation

Tầng giao vận: Dịch vụ TCP



- Phân kênh / Dồn kênh
- Truyền tin cậy
  - Giữa tiến trình Gửi và tiến trình Nhận
  - Hai bên phải thiết lập trước kết nối: Dịch vụ hướng kết nối
- Điều khiển lưu lượng
  - Bên gửi không gửi quá nhiều
- Kiểm soát tắc nghẽn
  - Giảm tốc độ gửi khi mạng quá tải
- Phát hiện lỗi
- Không cung cấp
  - Đảm bảo về thời gian và băng thông

1.4. Cổng giao thức

Là một số năm trong khoảng 1..65535 dùng để phân biệt giữa 2 ứng dụng mạng với nhau gắn với địa chỉ IP và Socket

Một số cổng và các giao thức thông dụng:

+ FTP: 21

+ Telnet: 23

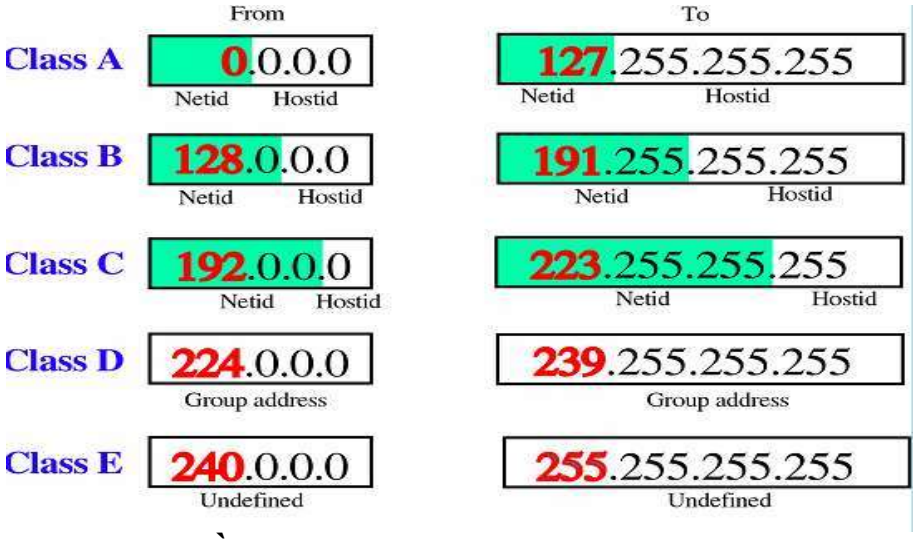
+ SMTP: 25

+ POP3: 110

+ HTTP:80

1.5. Địa chỉ IP, các địa chỉ IP dành riêng

	0	1	2	3	4				8		16	24	
Class A	0	Netid							Hostid				
Class B	1	0	Netid								Hostid		
Class C	1	1	0	Netted								Hostid	
Class D	1	1	1	0	Multicast Address								
Class E	1	1	1	1	0	Reverved for future use							



1.6. Địa chỉ tên miền: loại A, loại MX..

- ***Type = A***
  - ***Name: Hostname***
  - ***Value: IP address***
- ***Type = NS***
  - ***Name: Domain (vi dụ foo.com)***
  - ***Value: Địa chỉ IP của authoritative name server ứng với miền đó***
- ***Type = CNAME***
  - ***Name: Tên bí danh cho một tên thực nào đó: vi dụ www.ibm.com la tên bí danh của Servereast.backup2.ibm.com***
  - ***Value: Tên thực***
- ***Type = MX***
  - ***Value: Tên của mailserver***

Prefix	Suffix	Type Of Address	Purpose
all-0s	all-0s	this computer	used during bootstrap
network	all-0s	network	identifies a network
network	all-1s	directed broadcast	broadcast on specified net
all-1s	all-1s	limited broadcast	broadcast on local net
127	any	loopback	testing

Maximum number of unique addresses in each class

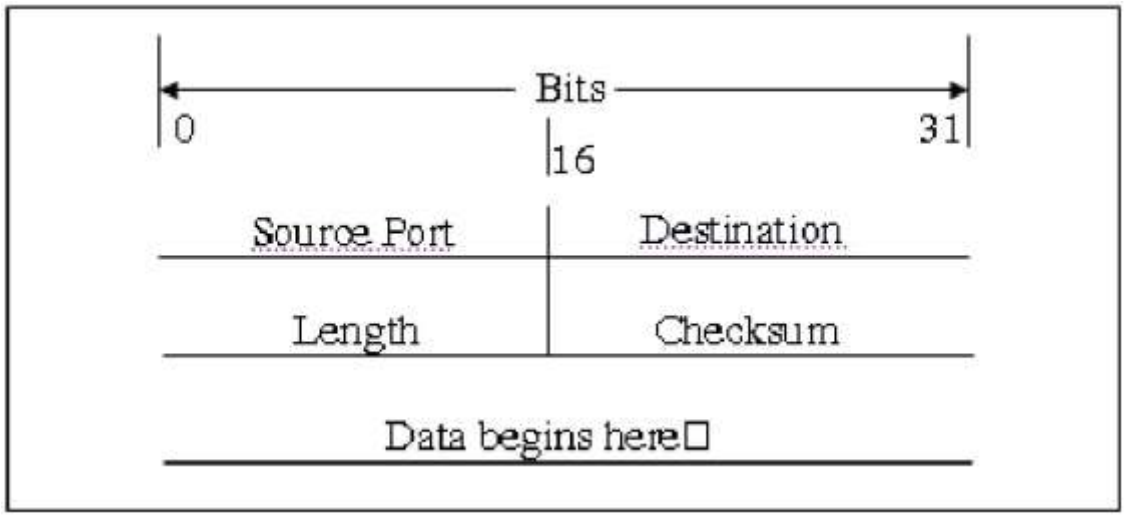
Class A 2^7 – 2 = 126

Class B 2^14 – 2 = 16,382

Class C 2^21 – 2 = 2,097,150

1.7. Giao thức hiệu năng UDP(User Datagram Protocol)

UDP là giao thức không liên kết , cung cấp dịch vụ giao vận không tin cậy đ- ợc, sử dụng thay thế cho TCP trong tầng giao vận. Khác với TCP, UDP không có chức năng thiết lập và giải phóng liên kết, không có cơ chế báo nhận (ACK), không sắp xếp tuần tự các đơn vị dữ liệu (datagram) đến và có thể dẫn đến tình trạng mất hoặc trùng dữ liệu mà không hề có thông báo cho ng- ời gửi. Khuân dạng của UDP datagram đ- ợc mô tả nh- sau:



- Số hiệu cổng nguồn (Source Port -16 bit): số hiệu cổng nơi đã gửi datagram.

- Số hiệu cổng đích (Destination Port – 16 bit): số hiệu cổng nơi datagram đã chuyển tới.

- Độ dài UDP (Length – 16 bit): độ dài tổng cộng kể cả phần header của UDP datagram.

- UDP Checksum(16 bit): dùng để kiểm soát lỗi, nếu phát hiện lỗi thì UDP datagram sẽ bị loại bỏ mà không có một thông báo nào trả lại cho trạm gửi.

- UDP có chế độ gán và quản lý các số hiệu cổng (port number) để định danh duy nhất cho nên UDP có xu thế hoạt động nhanh hơn so với TCP. Nó th- ờng dùng cho các ứng dụng không đòi hỏi độ tin cậy cao trong giao vận.

1.8. Giao thức RTP (Real-time Transport Protocol) :

Realtime Protocol là một chuẩn Internet để truyền các luồng thông tin giữa các thành phần tương tác trên mạng. RTP cung cấp các dịch vụ về dữ liệu mang tính thời gian thực như video và audio. Thông thường các ứng dụng chạy RTP dựa trên UDP để tận dụng khả năng multiplexing và kiểm lỗi. RTP hỗ trợ việc truyền dữ liệu đến nhiềuđịa chỉ đích bằng cách dùng cơ chế multicast nếu được hỗ trợ bởi hệ thống mạng.Giao thức truyền thời gian thực (RTP) là một thủ tục dựa trên kỹ thuật IP tạo ra các hỗ trợ để truyền tải các dữ liệu yêu cầu thời gian thực, ví dụ như các dòng dữ liệu hình ảnh và âm thanh. Các dịch vụ cung cấp bởi RTP bao gồm các cơ chế khôi phục thời gian, phát hiện các lỗi, bảo an và xác định nội dung. RTP được thiết kế chủ yếu cho việc truyền đa đối tượng nhưng nó vẫn có thể được sử dụng để truyền cho một đối tượng. RTP có thể truyền tải một chiều như dịch vụ video theo yêu cầu cũng như các dịch vụ trao đổi qua lại như điện thoại Internet.

Hoạt động của RTP được hỗ trợ bởi một thủ tục khác là RCTP để nhận các thông tin phản hồi về chất lượng truyền dẫn và các thông tin về thành phần tham dự các phiên hiện thời.

- **Hoạt động của giao thức.**

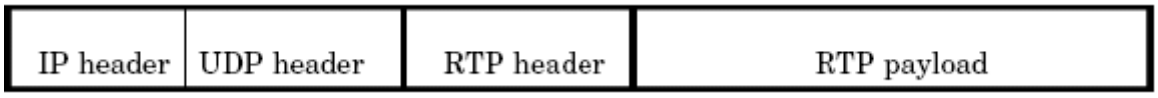
RTP không có sẵn các cơ chế để đảm bảo việc truyền theo thời gian hay các kỹ thuật về QoS mà dựa vào các dịch vụ ở lớp dưới để thực hiện những khả năng này. RTP không đảm bảo an toàn hay thứ tự các packet khi truyền, số thứ tự trong RTP packet cho phép bên nhận sắp xếp lại các packet theo thứ tự khi truyền của bên gửi. Ngoài ra số thứ tự cũng có thể được tận dụng để xác

định vị trí thích hợp của một packet, ví dụ trong việc giải mã video, mà không cần phải giải mã các packet theo thứ tự.

Các gói tin truyền trên mạng Internet có trễ và jitter không dự đoán được. Nhưng các ứng dụng đa phương tiện yêu cầu một thời gian thích hợp khi truyền các dữ liệu và phát lại. RTP cung cấp các cơ chế bảo đảm thời gian, số thứ tự và các cơ chế khác liên quan đến thời gian. Bằng các cơ chế này RTP cung cấp sự truyền tải dữ liệu thời gian thực giữa các đầu cuối qua mạng.

Tem *thời* gian (time-stamping) là thành phần thông tin quan trọng nhất trong các ứng dụng thời gian thực. Người gửi thiết lập các "tem thời gian" ngay thời điểm octet đầu tiên của gói được lấy mẫu. "Tem thời gian" tăng dần theo thời gian đối với mọi gói. Sau khi nhận được gói dữ liệu, bên thu sử dụng các "tem thời gian" này nhằm khôi phục thời gian gốc để chạy các dữ liệu này với tốc độ thích hợp.

Ngoài ra, nó còn được sử dụng để đồng bộ các dòng dữ liệu khác nhau (chẳng hạn như giữa hình và tiếng). Tuy nhiên RTP không thực hiện đồng bộ mà các mức ứng dụng phía trên sẽ thực hiện sự đồng bộ này. Bộ phận nhận dạng tải xác định kiểu định dạng của tải tin cũng như các phương cách mã hoá và nén. Từ các bộ phận định dạng này, các ứng dụng phía thu biết cách phân tích và chạy các dòng dữ liệu tải tin. Tại một thời điểm bất kỳ trong quá trình truyền tin, các bộ phát RTP chỉ có thể gửi một dạng của tải tin cho dù dạng của tải tin có thể thay đổi trong thời gian truyền (thay đổi để thích ứng với sự tắc nghẽn của mạng).Một chức năng khác mà RTP có là xác định nguồn . Nó cho phép các ứng dụng thu biết được dữ liệu đến từ đâu. Ví dụ thoại hội nghị, từ thông tin nhận dạng nguồn một người sử dụng có thể biết được ai đang nói.



Hình 3-7: Mã hoá gói tin RTP trong gói IP

Các cơ chế trên được thực hiện thông qua mào đầu của RTP. Cách mã hoá gói tin RTP trong gói tin IP được mô tả **trên** hình vẽ.

RTP nằm ở phía trên UDP, **sử** dụng các chức năng ghép kênh và kiểm tra của UDP.

UDP và TCP là hai giao thức được sử dụng chủ yếu trên Internet. TCP cung cấp các kết nối định hướng và các dòng thông tin với độ tin cậy cao trong khi UDP cung cấp các dịch vụ không liên kết và có độ tin cậy thấp giữa hai trạm chủ. Sở dĩ UDP được sử dụng làm thủ tục truyền tải cho RTP là bởi vì 2 lí do:

- Thứ nhất, RTP được thiết kế chủ yếu cho việc truyền tin đa đối tượng, các kết nối có định hướng, có báo nhận không đáp ứng tốt điều này.
- Thứ hai, đối với dữ liệu thời gian thực, độ tin cây không quan trọng bằng truyền đúng theo thời gian. Hơn nữa, sự tin cậy trong TCP là do cơ chế báo phát lại, không thích hợp cho RTP. Ví dụ khi mạng bị tắc nghẽn một số gói có thể mất, chất lượng dịch vụ dù thấp nhưng vẫn có thể chấp nhận được. Nếu thực hiện việc phát lại thì sẽ gây nên độ trễ rất lớn cho chất lượng thấp và gây ra sự tắc nghẽn của mạng.

Thực tế RTP được thực hiện chủ yếu trong các ứng dụng mà tại các mức ứng dụng này có các cơ chế khôi phục lại gói bị mất, điều khiển tắc nghẽn.

1.9. Giao thức RTCP (Real-time Transport Control Protocol):

RTCP (Real-time Transport Control Protocol) là giao thức hỗ trợ cho RTP cung cấp các thông tin phản hồi về chất lượng truyền dữ liệu. Các dịch vụ mà RTCP cung cấp là:

- Giám sát chất lượng và điều khiển tắc nghẽn: Đây là chức năng cơ bản của RTCP.Nó cung cấp thông tin phản hồi tới một ứng dụng về chất lượng phân phối dữ liệu. Thông tin điều khiển này rất hữu ích cho các bộ phát, bộ thu và giám sát. Bộ phát có thể điều chỉnh cách thức truyền dữ liệu dựa trên các thông báo phản hồi của bộ thu.Bộ thu có thể xác định được tắc nghẽn là cục bộ, từng phần hay toàn bộ. Người quản lí mạng có thể đánh giá được hiệu suất mạng.
- Xác định nguồn: Trong các gói RTP, các nguồn được xác định bởi các số ngẫu nhiên có độ dài 32 bít. Các số này không thuận tiện đối với người sử dụng RTCP cung cấp thông tin nhận dạng nguồn cụ thể hơn ở dạng văn bản. Nó có thể bao gồm tên người sử dụng, số điện thoại, địa chỉ e-mail và các thông tin khác.
- Đồng bộ môi trường: Các thông báo của bộ phát RTCP chứa thông tin để xác

định thời gian và nhãn thời gian RTP tương ứng. Chúng có thể được sử dụng để đồng bộ giữa âm thanh với hình ảnh.

- Điều chỉnh thông tin điều khiển: Các gói RTCP được gửi theo chu kỳ giữa những người tham dự. Khi số lượng người tham dự tăng lên, cần phải cân bằng giữa việc nhận thông tin điều khiển mới nhất và hạn chế lưu lượng điều khiển. Để hỗ trợ một nhóm người sử dụng lớn, RTCP phải cấm lưu lượng điều khiển rất lớn đến từ các tài nguyên khác của mạng. RTP chỉ cho phép tối đa 5% lưu lượng cho điều khiển toàn bộ lưu lượng của phiên làm việc. Điều này được thực hiện bằng cách điều chỉnh tốc độ phát của RTCP theo số lượng người tham dự.

CHƯƠNG 2: KIẾN THỨC CƠ BẢN VỀ LẬP TRÌNH C#

2.1. Ngôn ngữ C#

C# là một ngôn ngữ rất đơn giản, với khoảng 80 từ khoá và hơn mười kiểu dữ liệu dựng sẵn, nhưng C# có tính diễn đạt cao. C# hỗ trợ lập trình có cấu trúc, hướng đối tượng, hướng thành phần (component oriented).

Trọng tâm của ngôn ngữ hướng đối tượng là lớp. Lớp định nghĩa kiểu dữ liệu mới, cho phép mở rộng ngôn ngữ theo hướng cần giải quyết. C# có những từ khoá dành cho việc khai báo lớp, phương thức, thuộc tính (property) mới. C# hỗ trợ đầy đủ khái niệm trụ cột trong lập trình hướng đối tượng: đóng gói, thừa kế, đa hình.

C# hỗ trợ khái niệm giao diện, interfaces (tương tự Java). Một lớp chỉ có thể kế thừa duy nhất một lớp cha nhưng có thế thực thi nhiều giao diện.

C# có kiểu cấu trúc, struct (không giống C++). Cấu trúc là kiểu nhẹ hơn và bị giới hạn.Cấu trúc không thể thừa kế lớp hay được kế thừa nhưng có thể thực thi giao diện.

C# cung cấp những đặc trưng lập trình hướng thành phần như property, sự kiện và dẫn hướng khai báo (được gọi là attribute). Lập trình hướng component được hỗ trợ bởi CLR thông qua siêu dữ liệu (metadata). Siêu dữ liệu mô tả các lớp bao gồm các phương thức và thuộc tính, các thông tin bảo mật ….

Assembly là một tập hợp các tập tin mà theo cách nhìn của lập trình viên là các thư viện liên kết động (DLL) hay tập tin thực thi (EXE). Trong .NET một assembly là một đơn vị của việc tái sử dụng, xác định phiên bản, bảo mật, và phân phối. CLR cung cấp một số các lớp để thao tác với assembly.

C# cũng cho truy cập trực tiếp bộ nhớ dùng con trỏ kiểu C++, nhưng vùng mã đó được xem như không an toàn. CLR sẽ không thực thi việc thu dọn rác tự động các đối tượng được tham chiếu bởi con trỏ cho đến khi lập trình viên tự giải phóng.

2.2. Lớp, đối tượng và kiểu

Bản chất của lập trình hướng đối tượng là tạo ra các kiểu mới. Một **kiểu (type)** biểu diễn một "điều" gì đó. Đôi khi, "điều" đó là những gì trừu tượng như một bảng dữ liệu hay một chuỗi. Khi khác lại là những gì hữu hình hơn như một nút trong cửa sổ Windown. Một kiểu là định nghĩa những thuộc tính chung và cách hoạt động của "điều" đó.

Giống với các ngôn ngữ lập trình hướng đối tượng khác, một kiểu trong C# cũng định nghĩa bằng từ khoá **class (lớp)** còn thể hiện của lớp được gọi là **đối tượng (object)**.

2.3. Phương thức

Các hành vi của một lớp được gọi là các phương thức thành viên (gọi tắt là phương thức) của lớp đó. Một phương thức là một hàm (phương thức thành viên còn gọi là hàm thành viên). Các phương thức định nghĩa những gì mà một lớp có thể làm.

Phương thức thường đưa ra tên của hành động như là WriteLine( ) hay AddNumbers( ). Tuy nhiên có một lớp phương thức có tên dặc biệt Main(), nó không diễn tả một hành động nhưng được chỉ định rõ với CLR đó là phương thức chính đầu tiên cho lớp của bạn. Khi một chương trình bắt đầu, CLR sẽ gọi hàm main() đầu tiên và bất cứ chương trình C# nào cũng phải có hàm main().

Sự khai báo phương thức là một liên hệ giữa người tạo ra phương thức và người thực hiện phương thức. Giống như là người viết phương thức và người sử dụng phương thức là một, nhưng thực tế thì không hoàn toàn như vậy. Nó có thể là do một thành viên trong đội phát triển sẽ tạo ra phương thức và người lập trình viên khác sẽ sử dụng lại nó.

Để khai báo một phương thức, bạn phải chỉ định giá trị trả về của nó. Khi khai báo phương thức phái có dấu ngoặc đơn (), và lúc có chấp nhân truyền tham biến, lúc không. Giá trị trả về cho người sử dụng biết kiểu dữ liệu đó trả về khi phương thức chạy xong. Một số phương thức không trả về một giá trị cụ thể, gọi là trả về kiểu void và được khai báo bằng từ khóa void. Và trong C#, một phương thức bắt buộc phải trả về một kiểu giá trị cụ thể hoặc kiểu void.

2.4. Các kiểu

C# buộc phải khai báo kiểu của đối tượng được tạo. Khi kiểu được khai báo rõ ràng, trình biên dịch sẽ giúp ngăn ngừa lỗi bằng cách kiểm tra dữ liệu được gán cho đối tượng có hợp lệ không, đồng thời cấp phát **đúng** kích thước bộ nhớ cho đối tượng.

C# phân thành hai loại: loai dữ liệu dựng sẵn (intrinsic (built-in)) và loại do người dùng định nghĩa (user-defined).

C# cũng chia tập kiểu thành hai loại: giá trị và tham chiếu. Biến kiểu giá trị được lưu trong vùng nhớ stack, còn biến kiểu tham chiếu được lưu trong vùng nhớ heap.

C# hỗ trợ kiểu con trỏ của C++, nhưng ít khi được sử dụng. Thông thường con trỏ chỉ được sử dụng khi làm việc trực tiếp với Win API hay các đối tượng COM.

Loại dữ liệu định sẵn

C# có nhiều kiểu dữ liệu định sẵn, mỗi kiểu ánh xạ đến một kiểu được hỗ trợ bởi CLS (Commom Language Specification), ánh xạ để đảm bảo rằng đối tượng được tạo trong C# không khác gì đối tượng được tạo trong các ngôn ngữ .NET khác Mỗi kiểu có một kích thước cố định được liệt kê trong bảng sau:

Các kiểu dựng sẵn:

Kiểu	*Kích thước (byte)*	*Kiểu Net*	*Mô tả - Giá trị*
Byte	***1***	***Byte***	***Không dấu***
Char	***1***	***Char***	***Mã ký tự Unicode***
Bool	***1***	***Boolean***	***True hay False***
Sbyete	***1***	***Sbyte***	***Có dấu (-128..127)***
Short	***2***	***Int 16***	***Có dấu (-32768…32767)***
Ushort	***2***	***Unit 16***	***Không dấu(-2147483647…2147483647)***
Int	***4***	***Int 32***	***Không có dấu (0…4294967295)***
Uint	***4***	***Uint 32***	***Số thực (≈ ± 1.5*10-45.. ≈ ±3.4*1038)***
Float	***4***	***Single***	***Số thực (≈ ±5.0*10324.. ≈ ±1.7*10308)***
Double	***8***	***Double***	***Số có dấu chấm tĩnh với 28 ký số và dấu chấm***
Decimal	***8***	***Decimal***	***Số nguyên có dấu(-9223372036854775808.. 9223372036854775808)***

Long	***8***	***Int 64***	***Số nguyên không dấu (0..0xffffffffffffffff)***
Ulong	***8***	***Uint 64***	***Không dấu (0..65535)***

2.4.1. Chọn một kiểu định sẵn

Tuỳ vào từng giá trị muốn lưu trữ mà ta chọn kiểu cho phù hợp. Nếu chọn kiểu quá lớn so với các giá trị cần lưu sẽ làm cho chương trình đòi hỏi nhiều bộ nhớ và chạy chậm. Trong khi nếu giá trị cần lưu lớn hơn kiểu thực lưu sẽ làm cho giá trị các biến bị sai và chương trình cho kết quả sai.

Kiểu char biểu diễn một ký tự Unicode. Ví dụ "\u0041" là ký tự "A" trên bảng Unicode. Một số ký tự đặc biệt được biểu diễn bằng dấu "\" trước một ký tự khác.

Các ký tự đặc biệt thông dụng:

Ký tự	Nghĩa
\'	dầu nháy đơn
\"	dấu nháy đôi
\\	dấu chéo ngược "\"
\0	Null
\a	Alert
\b	lùi về sau
\f	Form feed
\n	xuống dòng
\r	về đầu dòng
\t	Tab ngang
\v	Tab dọc

2.4.2. Chuyển đổi kiểu định sẵn

Một đối tượng có thể chuyển từ kiểu này sang kiểu kia theo hai hình thức: ngầm hoặc tường minh. Hình thức ngầm được chuyển tự động còn hình thức tường minh cần sự can thiệp trực tiếp của người lập trình:

```
short x = 5;
int y ;
y = x; // chuyển kiểu ngầm định - tự động
x = y; // lỗi, không biên dịch được
x = (short) y; // OK
```

2.5. Biến và hằng

Biến dùng để lưu trữ dữ liệu. Mỗi biến thuộc về một kiểu dữ liệu nào đó.

2.5.1. Khởi tạo trước khi dùng

Trong C#, trước khi dùng một biến thì biến đó phải được khởi tạo nếu không trình biên dịch sẽ báo lỗi khi biên dịch. Ta có thể khai báo biến trước, sau đó khởi tạo và sử dụng; hay khai báo biến và khởi gán trong lúc khai báo.

```
int x; // khai báo biến trước
x = 5; // sau đó khởi gán giá trị và sử dụng
int y = x; // khai báo và khởi gán cùng lúc
```

2.5.2. Hằng

Hằng là một biến nhưng giá trị không thay đổi theo thời gian. Khi cần thao tác trên một giá trị xác định ta dùng hằng. Khai báo hằng tương tự khai báo biến và có thêm từ khóa const ở trước. Hằng một khi khởi động xong không thể thay đổi được nữa.

```
const int HANG_SO = 100;
```

2.5.3. Kiểu liệt kê

Enum là một cách thức để đặt tên cho các trị nguyên (các trị kiểu số nguyên, theo nghĩa nào đó tương tự như tập các hằng), làm cho chương trình rõ ràng, dễ hiểu hơn. Enum không có hàm thành viên. Ví dụ tạo một enum tên là Ngay như sau:

```
enum Ngay {Hai, Ba, Tu, Nam, Sau, Bay, ChuNhat};
```

Theo cách khai báo này enum ngày có bảy giá trị nguyên đi từ 0 = Hai, 1 = Ba, 2 = Tư … 7 = ChuNhat.

Mặc định enum gán giá trị đầu tiên là 0 các trị sau lớn hơn giá trị trước một đơn vị, và các trị này thuộc kiểu int. Nếu muốn thay đổi trị mặc định này ta phải gán trị mong muốn.

Cú pháp chung cho khai báo một kiểu enum như sau

```
[attributes] [modifiers] enum identifier [:base-type]
{
enumerator-list
};
```

2.5.4. Chuỗi

Chuỗi là kiểu dựng sẵn trong C#, nó là một chuỗi các ký tự đơn lẻ. Khi khai báo một biến chuỗi ta dùng từ khoá string. Ví dụ khai báo một biến string lưu chuỗi "Hello World"

string myString = "Hello World";

2.5.5. Định danh

Định danh là tên mà người lập trình chọn đại diện một kiểu, phương thức, biến, hằng, đối tượng… của họ. Định danh phải bắt đầu bằng một ký tự hay dấu "_".

Định danh không được trùng với từ khoá C# và phân biệt hoa thường.

2.6. Biểu thức

Bất kỳ câu lệnh định lượng giá trị được gọi là một biểu thức (expression). Phép gán sau cũng được gọi là một biểu thức vì nó định lượng giá trị được gán (là 32)

x = 32;

Vì vậy phép gán trên có thể được gán một lần nữa như sau

y = x = 32;

Sau lệnh này y có giá trị của biểu thức x = 32 và vì vậy y = 32.

2.7. Câu lệnh

Cũng như trong C++ và Java một chỉ thị hoàn chỉnh thì được gọi là một câu lệnh (statement). Chương trình gồm nhiều câu lệnh, mỗi câu lệnh kết thúc bằng dấu ";".

Ví dụ:

int x; // là một câu lệnh

x = 23; // một câu lệnh khác

Ngoài các câu lệnh bình thường như trên, có các câu lệnh khác là: lệnh rẽ nhánh không điều kiện, rẽ nhánh có điều kiện và lệnh lặp.

2.7.1. Các lệnh rẽ nhánh không điều kiện

Có hai loại câu lệnh rẽ nhánh không điều kiện. Một là lệnh gọi phương thức: khi trình biên dịch thấy có lời gọi phương thức nó sẽ tạm dừng phương thức hiện hành và nhảy đến phương thức được gọi cho đến hết phương thức này sẽ trở về phương thức cũ.

Cách thứ hai để tạo các câu lệnh rẽ nhánh không điều kiện là dùng từ khoá: goto,break, continue, return, hay throw. Cách từ khóa này sẽ được giới thiệu trong các phần sau.

2.7.2. Lệnh rẽ nhánh có điều kiện

Các từ khóa if-else, while, do-while, for, switch-case, dùng để điều khiển dòng chảy chương trình. C# giữ lại tất cả các cú pháp của C++, ngoại trừ switch có vài cải tiến.

a. Lệnh If .. else …

Cú pháp:

if (biểu thức logic)

khối lệnh;

hoặc

if (biểu thức logic)

khối lệnh 1;

else

khối lệnh 2;

Ghi chú: Khối lệnh là một tập các câu lện trong cặp dấu "{…}". Bất kỳ nơi đâu có câu lệnh thì ở đó có thể viết bằng một khối lệnh.

Biểu thức logic là biểu thức cho giá trị dúng hoặc sai (true hoặc false). Nếu "biểu thức logic" cho giá trị đúng thì "khối lệnh" hay "khối lệnh 1" sẽ được thực thi, ngược lại "khối lệnh 2" sẽ thực thi. Một điểm khác biệt với C++ là biểu thức trong câu lệnh if phải là biểu thức logic, không thể là biểu thức số.

b. Lệnh switch

Cú pháp:

switch (biểu_thức_lựa_chọn)

{

case biểu_thức_hằng :

khối lệnh;

lệnh nhảy;

[default :

khối lệnh;

lệnh nhảy;]

}

Biểu thức lựa chọn là biểu thức sinh ra trị nguyên hay chuỗi. Switch sẽ so sánh biểu_thức_lựa_chọn với các biểu_thức_hằng để biết phải thực hiện với khối lệnh nào. Lệnh nhảy như break, goto…để thoát khỏi câu switch và bắt buộc phải có.

2.7.3. Lệnh lặp

C# cung cấp các lệnh lặp giống C++ như for, while, do-while và lệnh lặp mới foreach. Nó cũng hổ trợ các câu lệnh nhảy như: goto, break, continue và return.

a. Lệnh goto

Lệnh goto có thể dùng để tạo lệnh nhảy nhưng nhiều nhà lập trình chuyên nghiệp khuyên không nên dùng câu lệnh này vì nó phá vỡ tính cấu trúc của chương trình.

Cách dùng câu lệnh này như sau: (giống như trong C++)

1. Tạo một nhãn

2. goto đến nhãn đó.

b. Vòng lặp while

Cú pháp:

while (biểu_thức_logic)

khối_lệnh;

Khối_lệnh sẽ được thực hiện cho đến khi nào biểu thức còn đúng. Nếu ngay từ đầu biểu thức sai, khối lệnh sẽ không được thực thi.

c. Vòng lặp do … while

Cú pháp:

do

khối_lệnh

while (biểu_thức_logic)

Khác với while khối lệnh sẽ được thực hiện trước, sau đó biệu thức được kiểm tra.Nếu biểu thức đúng khối lệnh lại được thực hiện.

d. Vòng lặp for

Cú pháp:

for ([khởi_tạo_biến_đếm]; [biểu_thức]; [gia_tăng_biến_đếm])

khối lệnh;

e. Câu lệnh break, continue, và return

Cả ba câu lệnh break, continue, và return rất quen thuộc trong C++ và Java, trong C#, ý nghĩa và cách sử dụng chúng hoàn toàn giống với hai ngôn ngữ này.

2.8. Toán tử

Các phép toán +, -, *, / là một ví dụ về toán tử. Áp dụng các toán tử này lên các biến kiểu số ta có kết quả như việc thực hiện các phép toán thông thường.

```
int a = 10;
int b = 20;
int c = a + b; // c = 10 + 20 = 30
```

C# cung cấp cấp nhiều loại toán tử khác nhau để thao tác trên các kiểu biến dữ liệu, được liệt kê trong bảng sau theo từng nhóm ngữ nghĩa.

2.8.1. Toán tử gán (=)

Toán tử này cho phép thay đổi các giá trị của biến bên phải toán tử bằng giá trị bên trái toán tử.

2.8.2. Nhóm toán tử toán học

C# dùng các toàn tử số học với ý nghĩa theo đúng tên của chúng như: + (cộng), – (trừ) , * (nhân) và / (chia). Tùy theo kiểu của hai toán hạng mà toán tử trả về kiểu tương ứng. Ngoài ra, còn có toán tử % (lấy phần dư) được sử dụng trong các kiểu số nguyên.

2.8.3. Các toán tử tăng và giảm

C# cũng kế thừa từ C++ và Java các toán tử: +=,-=, *=, /= , %= nhằm làm đơn giản hoá. Nó còn kế thừa các toán tử tiền tố và hậu tố (như biến++, hay ++biến) để giảm bớt sự cồng kềnh trong các toán tử cổ điển.

2.8.4. Các toán tử quan hệ

Các toán tử quan hệ được dùng để so sánh hai giá trị với nhau và kết quả trả về có kiểu Boolean. Toán tử quan hệ gồm có: == (so sánh bằng), != (so sánh khác), > (so sánh lớn hơn), >= (lớn hơn hay bằng), < (so sánh nhỏ hơn), <= (nhỏ hơn hay bằng).

2.8.5 Các toán tử logic

Các toán tử logic gồm có: && (và), || (hoặc), ! (phủ định). Các toán tử này được dùng trong các biểu thức điều kiện để kết hợp các toán tử quan hệ theo một ý nghĩa nhất định.

2.8.6. Thứ tự các toán tử

Đối với các biểu thức toán, thứ tự ưu tiên là thứ tự được qui định trong toán học. Còn thứ tự ưu tiên thực hiện của các nhóm toán tử được liệt kê theo bảng dưới đây

Thứ tự ưu tiên của các nhóm toán tử (chiều ưu tiên từ trên xuống)

Nhóm toán tử	Toán tử	Ý nghĩa
Primary (chính)	{x} x.y f(x) a[x] x++x--	
Unary	+ - ! ~ ++x –x (T)x	
Nhân	* / %	Nhân, chia, lấy phần dư
Cộng	+ -	cộng, trù
Dịch bít	<< >>	Dịch trái, dịch phải
Quan hệ	< > <= >= is	nhỏ hơn, lớn hơn, nhỏ hơn hay bằng, lớn hơn hay bằng và là
Bằng	== !=	bằng, khác
Logic trên bit AND	&	Và trên bit.
XOR	^	Xor trên bit
OR	|	hoặc trên bit
Điều kiện AND	&&	Và trên biểu thức điều kiện
Điều kiện OR	||	Hoặc trên biểu thức điều kiện
Điều kiện	?:	điều kiện tương tự if
Assignment	= *= /= %= += -= <<=	

2.9. Namespaces

Namespaces là một cách tổ chức mã nguồn thành các nhóm có ngữ nghĩa liên quan. Bạn có thể thoải mái tạo một namespace của bạn được hỗ trợ trong.NET Framework hoặc từ các dịch vụ khác. Để khai báo bạn phải dùng từ khóa namespace và sau đó là tên namespace mà bạn muốn tạo, kèm theo đó là các đối tượng của namespace được đặt trong dấu {}.

2.10. Lớp và đối tượng

Một lớp định nghĩa một tập các đối tượng hoặc các thể hiệncủa lớp đó. Đối tượng là một trị có thể được tạo ra, lưu giữ và sử dụng. Trong C# tất cả các biến đều là đối tượng. Các biến kiểu số, kiểu chuỗi … đều là đối tượng. Mỗi một đối tượng đều có các biến thành viên để lưu giữ dữ liệu và có các phương thức (hàm) để tác động lên biến thành viên. Mỗi đối tượng thuộc về một lớp đối tương nào đó. Các đối tượng có cùng lớp thì có cùng các biến thành viên và phương thức.

2.10.1. Định nghĩa lớp

Cú pháp:

[attribute][bổ từ truy xuất] class **định danh** [:lớp cơ sở]

{

thân lớp

}

a. Bổ từ truy xuất

Bổ từ truy xuất xác định thành viên (nói tắt của biến thành viên và phương thức thành viên) nào của lớp được truy xuất từ lớp khác. Có các loại kiểu truy xuất sau:

- public: Truy xuất mọi nơi
- protected: Truy xuất trong nội bộ lớp hoặc trong các lớp con
- internal: Truy xuất nội trong chương trình (assembly)
- protected internal: Truy xuất nội trong chương trình (assembly) và trong các lớp con
- private (mặc định): Chỉ được truy xuất trong nội bộ lớp

b. Các tham số của phương thức

Mỗi phương thức có thể không có tham số mà cũng có thể có nhiều tham số. Các tham số theo sau tên phương thức và đặt trong cặp ngoặc đơn.

2.10.2. Tạo đối tượng

a.Constructor

Constructor là phương thức đầu tiên được triệu gọi và chỉ gọi một lần khi khởi tạo đối tượng, nó nhằm thiết lập các tham số đầu tiên cho đối tượng. Tên Constructor trùng tên lớp, còn các mặt khác như phương thức bình thường.

Nếu lớp không định nghĩa hàm Constructor, trình biên dịch tự động tạo một Constructor mặc định. Khi đó các biến thành viên sẽ được khởi tạo theo các giá trị mặc định:

Kiểu	Giá trị mặc định
số (int, long, …)	0
bool	false
char	‘\0’ (null)
enum	0
Tham chiếu	null

b. Khởi tạo

Ta có thể khởi tạo giá tri các biến thành viên theo ý muốn bằng cách khởi tạo nó trong constructor của lớp hay có thể gán vào trực tiếp lúc khai báo. Với giá trị khởi tạo này thì khi một đối tượng khai báo kiểu của lớp này thì giá trị ban đầu là các giá trị khởi tạo chứ không phải là giá trị mặc định.

c.Copy constructor

Hàm dựng sao chép (copy constructor) là sao chép toàn bộ nội dung các biến từ đối tượng đã tồn tại sang đối tượng mới khởi tạo.

d. Từ khoá this

Từ khoá this được dùng để tham chiếu đến chính bản thân của đối tượng đó. Ví dụ:

```
public void SomeMethod (int hour)
{
    this.hour = hour;
}
```

2.10.3. Sử dụng các thành viên tĩnh

Ta có thể truy cập các thành viên static thông qua tên của lớp mà nó được khai báo trong đó.

a. Cách gọi một thành viên tĩnh

Phương thức tĩnh (static) được nói là hoạt động trong lớp. Do đó, nó không thể được tham chiếu this chỉ tới. Phương thức static cũng không truy cập trực tiếp vào các thành viên không static được mà phải dùng qua thể hiện của đối tượng(đối với hàm main()).

b. Sử dụng Constructors tĩnh

Constructor tĩnh (static constructor) sẽ được chạy trước khi bất kỳ đối tượng nào tạo ra.Ví dụ:

```
static Time( )
{
    Name = "Time";
}
```

Khi dùng constructor tĩnh phải khá thận trọng vì nó có thể có kết quả khó lường.

c. Constructor private

Khi muốn tạo một lớp mà không cho phép tạo bất kỷ một thể hiện nào của lớp thì ta dùng constructor private.

d. Sử dụng các trường tĩnh

Cách dùng chung các biến thành viên tĩnh là giữ vết của một số các thể hiện mà hiện tại nó đang tồn tại trong lớp đó.

2.10.4. Truyền tham số

C# cung cấp các tham số ref để h iệu chỉnh giá trị của những đối tượng bằng các tham chiếu.

a. Truyền bằng tham chiếu

Một hàm chỉ có thể trả về một giá trị. Trong trường hợp muốn nhận về nhiều kết quả, ta sử dụng chính các tham số truyền cho hàm như các tham số có đầu ra (chứa trị trả về). Ta gọi tham số truyền theo kiểu này là tham chiếu.

Trong C#, tất cả các biến có kiểu tham chiếu sẽ mặc định là tham chiếu khi các biến này được truyền cho hàm. Các biến kiểu giá trị để khai báo tham chiếu, sử dụng từ khóa ref.

b. Truyền tham số đầu ra (out parameter)

Để sử dụng được, một biến phải được khai báo và khởi tạo giá trị ban đầu. Các biến theHour, theMinute, theSecond phải được khởi tạo giá trị 0 trước khi truyền cho hàm GetTime. Sau lời gọi hàm thì giá trị các biến sẽ thay đổi ngay, vì vậy C# cung cấp từ khóa out để không cần phải khởi tạo tham số trước khi dùng.

2.11. Kế thừa và Đa hình

Kế thừa là cách tạo mới một lớp từ những lớp có sẵn. Tức là nó cho phép tái sử dụng lại mã nguồn đã viết trong lớp có sẵn. Đa hình là việc lớp B thừa kế các đặc tính từ lớp A nhưng có thêm một số cài đặt riêng.

2.11.1 Sự kế thừa

Trong C#, mối quan hệ chi tiết hoá là một kiểu kế thừa. Sự kế thừa không cho mang ý nghĩa chi tiết hoá mà còn mang ý nghĩa chung của tự nhiên về mối quan hệ này. Khi ta nói rằng ListBox kế thửa từ Window có nghĩa là nó chi tiết hoá Window. Window được xem như là lớp cơ sở (base class) và ListBox được xem là lớp kế thừa (derived class). Lớp ListBox này nhận tất cả các đặc tính và hành vi của Window và chi tiết hoá nó bằng một số thuộc tính và phương thức của nó cần.

Trong C#, khi ta tạo một lớp kế thừa bằng cách công một thêm dấu ":" và sau tên của lớp kế thừa và theo sau đó là lớp cơ sở như sau:

public class ListBox : Window có nghĩa là ta khai báo một lớp mới ListBox kế thừa từ lớp Window.

Lớp kế thừa sẽ thừa hưởng được tất các phương thức và biến thành viên của lớp cơ sở, thậm chí còn thừa hưởng cả các thành viên mà cơ sở đã thừa hưởng.

2.11.2. Đa hình

Đa hình là việc lớp B thừa kế các đặc tính từ lớp A nhưng có thêm một số cài đặt riêng. ListBox và Button đều là một Window, ta muốn có một form để giữ tập hợp tất cả các thể hiện của Window để khi một thể hiện nào được mở thì nó có thể bắt Window của nó vẽ lên. Ngắn gọn, form này muốn quản lý mọi cư

xử của tất cà các đối tượng đa hình của Window.Tạo phương thức đa hình, ta cần đặt từ khoá virtual trong phương thức của lớp cơ sở. Ví dụ như:

public virtual void DrawWindow( )

Trong lớp kế thừa để nạp chồng lại mã nguồn của lớp cơ sở ta dùng từ khoá override khi khai báo phương thức và nội dung bên trong viết bình thường. Ví dụ về nạp chồng phương thức DrawWindow:

```
public override void DrawWindow( )
{
base.DrawWindow( ); // gọi phương thức của lớp co sở
Console.WriteLine ("Writing string to the listbox: {0}",
listBoxContents);
}
```

Dùng hình thức đa hình phương thức này thì tuỳ kiểu khai báo của đối tượng nào thì nó dùng phương thức của lớp đó.

Khi cần viết lại một phương thức trong lớp kế thừa mà đã có trong lớp cơ sở nhưng ta không muốn nạp chồng lại phương thức virtual trong lớp cơ sở ta dùng từ khoá new đánh dấu trước khi từ khoá virtual trong lớp kế thừa.

```
public class ListBox : Window
{
    public new virtual void Sort( ) {...}
}
```

2.12. Cấu trúc

Một cấu trúc (struct) là một kiểu do người dùng định nghĩa, nó tương tự như lớp như nhẹ hơn lớp.

Định nghĩa cấu trúc

Cú pháp

```
[thuộc tính] [kiểu truy cập] struct <định danh> [: <danh sách các giao diện >]
{
// Các thành viên của cấu trúc
}
```

Không giống như lớp, cấu trúc không hỗ trợ kế thừa. Tất cả các cấu trúc thừa kế ngầm định object nhưng nó không thể thừa kế từ bất kỳ lớp hay cấu trúc nào khác. Các cấu trúc cũng ngầm định là đã niêm phong. Tuy nhiên, nó có điểm giống với lớp là cho phép cài đặt đa giao diện.

Cấu trúc không có hủy tử cũng như không thể đặt các tham số tuỳ ý cho Constructor. Nếu ta không cài đặt bất kỳ hàm dựng nào thì cấu trúc được cung cấp hàm dựng mặc định, đặt giá trị 0 cho tất cả các biến thành viên.

Do cấu trúc được thiết kế cho nhẹ nhàng nên các biến thành viên đều là kiểu private và được gói gọn lại hết. Tuỳ từng tình huống và mục đích sử dụng mà ta cần cân nhắc chọn lựa dùng lớp hay cấu trúc.

2.13. Windows Form

Hầu hết mọi ứng dụng Windows Form mở rộng chức năng của *System.Windows.Forms*. Chức năng cơ bản của lớp Form không thể tạo một cửa sổ có thể sống và tương tác trong môi trường Windows một cách đúng đắn. Đây là một thuận lợi như một điểm khởi đầu và bằng việc mở rộng lớp Form và thêm các control tuỳ biến và các bộ điều khiển sự kiện tuỳ biến, một ứng dụng rất hữu ích được tạo để có thể tương tác với người dùng và dữ liệu hiện tại thông qua một giao diện người dùng tinh vi.

Các lớp thừa kế từ *System.Windows.Forms:*

- *System.Windows.Forms.Control* - hành động này như lớp cơ bản cho phần lớn các lớp trong namespace. Nó chứa chức năng cơ bản của thao tác xử lý bàn phím và nhập từ chuột và xử lý tin nhắn window.
- *System.Windows.Forms.ButtonBase* - Lớp này hỗ trợ chức năng cơ bản của một nút mà mọi lớp thừa hưởng sử dụng trong các cách khác nhau.
- *System.Windows.Forms.TextBoxBase* - một lần nữa, lớp này là một lớp cơ sở được sử dụng để cung cấp chức năng và thuộc tính thông thuờng cho các lớp thừa hưởng. Cả hai lớp TextBox và RichTextBox sử dụng chức năng cung cấp bởi TextBoxBase.

- *System.Windows.Forms.ScrollableControl* - đây là một lớp cơ bản khác cung cấp hỗ trợ cho các lớp thừa hưởng. Lớp này quản lý sự phát sinh và hiển thị của các thanh cuộn đến người dùng để truy cập đến gốc của một hiển thị.
- *System.Windows.Forms.ContainerControl* - Lớp này quản lý chức năng yêu cầu cho một control để hành động như một sự chứa đựng những control khác.
- *System.Windows.Forms.Panel* - đây là control khác có thể chứa các control thêm vào, nhưng khác với lớp *ContainerControl*, nó phân loại các control một cách đơn giản.
- *System.Windows.Forms.Form* - Đây là lớp mà phân phát với việc tạo ra và hiển thị các cửa sổ. Lớp này có thể được dùng để tạo bất kỳ loại cửa sổ nào: standard, toolbox, borderless, even modal dialog boxes và multi-document interfaces.
- *System.Windows.Forms.UserControl* - Đây là lớp có thể được dùng để thừa hưởng từ việc tạo một custom control đến việc được dùng trong một nơi phức tạp trong một ứng dụng hay tổ chức.

2.14. Truy cập dữ liệu

Trong phần này chúng ta sẽ khảo sát sang trình cung cấp dữ liệu **OLE DB .NET Data Provider**, với trình cung cấp dữ liệu này ta có thể kết nối đến bất kỳ hệ quản trị cơ sở dữ liệu nào có hỗ trợ trình cung cấp dữ liệu **OLE DB Providers**, cụ thể là **Microsoft Access** với tập CSDL northwind.mdb**.**

Đầu tiên là chuỗi kết nối :

```
string connectionString = "provider=Microsoft.JET.OLEDB.4.0; "
+ "data source = c:\\northwind.mdb";
```

Chuỗi trên sẽ kết nối đến cơ sở dữ liệu northwind trên ổ đĩa C.

Kế tiếp ta là đối tượng OleDbDataAdapter

```
OleDbDataAdapter DataAdapter = new OleDbDataAdapter(
commandString, connectionString);
```

Chúng ta phải đảm bảo là namespace OleDb được thêm vào ứng dụng :

```
using System.Data.OleDb;
```

Sau đây sẽ trích ra một đoạn mã chính phục vụ cho việc kết nối theo cách này :

```
public ADOForm1( )
{
InitializeComponent( );
// chuỗi kết nối đến cơ sở dữ liệu
string connectionString = "provider=Microsoft.JET.OLEDB.4.0;"
+ "data source = c:\\nwind.mdb";
// chuỗi truy vấn dữ liệu
string commandString =
"Select CompanyName, ContactName from Customers";
// tạo đối tượng OleDbDataAdapter và DataSet mới
OleDbDataAdapter DataAdapter = new OleDbDataAdapter(
commandString, connectionString);
DataSet dataSet = new DataSet( );
// đẩy dữ liệu vào dataSet
DataAdapter.Fill(DataSet,"Customers");
// lây về bảng dữ liệu Customers
DataTable dataTable = DataSet.Tables[0];
// duyệt qua từng dòng dữ liệu
foreach (DataRow dataRow in dataTable.Rows)
{
lbCustomers.Items.Add(dataRow["CompanyName"] +
" (" + dataRow["ContactName"] + ")" );
}
}
```

CHƯƠNG 3: CHƯƠNG TRÌNH ỨNG DỤNG

3.1. Chức năng của chương trình

Chương trình được viết trên nền tảng ngôn ngữ C#, nhằm tạo ra một chương trình ứng dụng cho các đối tượng :

Những đối tượng sử dụng hệ thống gồm có:

- Giáo viên: Người làm công tác giảng dạy
- Sinh viên: Người học thông qua hệ thống mạng LAN

3.1.1. Chức năng dành cho giáo viên:

3.1.1.1. Quản lý lớp học của mình:

Mặc định là các lớp được tạo ra là không có mật khẩu, mọi sinh viên đều có thể tham gia vào lớp học được.

Giáo viên có thể thay đổi địa chỉ IP của lớp mình

Giáo viên là người chủ của một lớp học cũng có thể xoá đi lớp học đó

3.1.1.2. cho xem – không cho xem bài giảng của giáo viên

Khi giáo viên bật chức năng cho phép xem thì mới có thể thấy được hình ảnh của bài giảng trên giao diện.

3.1.2. Chức năng dành cho sinh viên:

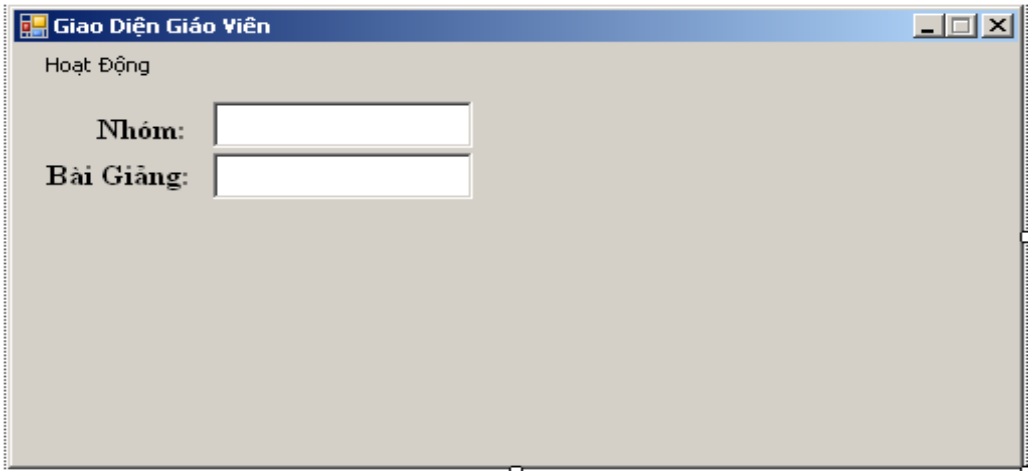
3.1.2.1. Đăng nhập vào học

Sinh viên được cấp địa chỉ IP, sau khi đăng nhập vào hệ thống sinh viên đó mới có quyền tham gia vào lớp học được mở, bài giảng sẽ được phát trên màn hình giao diện của sinh viên.

3.2. Thiết kế giao diện.

3.2.1. Giao diện của giáo viên

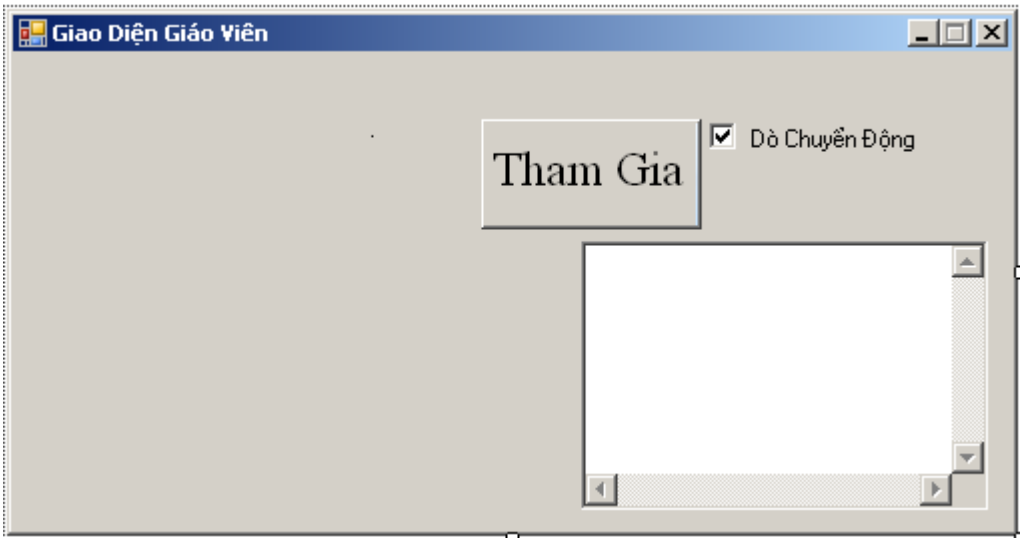
3.2.1.1. Giao diện nhập địa chỉ và tên bài giảng của giáo viên



Hình 3.2.1 Địa chỉ và tên bài giảng của giáo viên

STT	Tên	Mô tả
1	Title	Hiển thị tiêu đề của giao diện
2	Hoat động	Chỉ trạng thái của giao diện
3	Nhóm	Là một TextBbox để nhập địa chỉ IP của lớp học
4	Bài giảng	Là một TexBox nhập bài giảng cho giáo viên đứng lớp

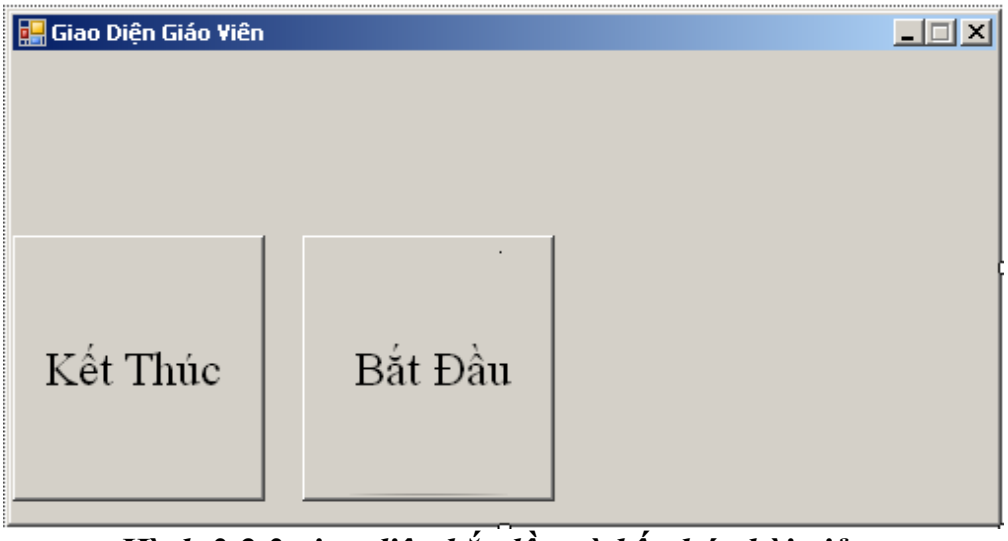
3.2.1.2. Giao diện hiển thị nội dung và thời gian bài giảng của giáo viên



Hình 3.2.2- Địa chỉ và tên bài giảng của giáo viên

STT	Tên	Mô tả
1	Tham gia	Kích vào nút Button để chuẩn bị bài giảng
2	Dò chuyển động	Chỉ trạng thái hoạt động của bài giảng
3	ListBox	Là một ListBbox hiển thị nội dung và thời gian bài giảng

3.2.1.3. Giao diện bắt đầu và kết thúc bài giảng

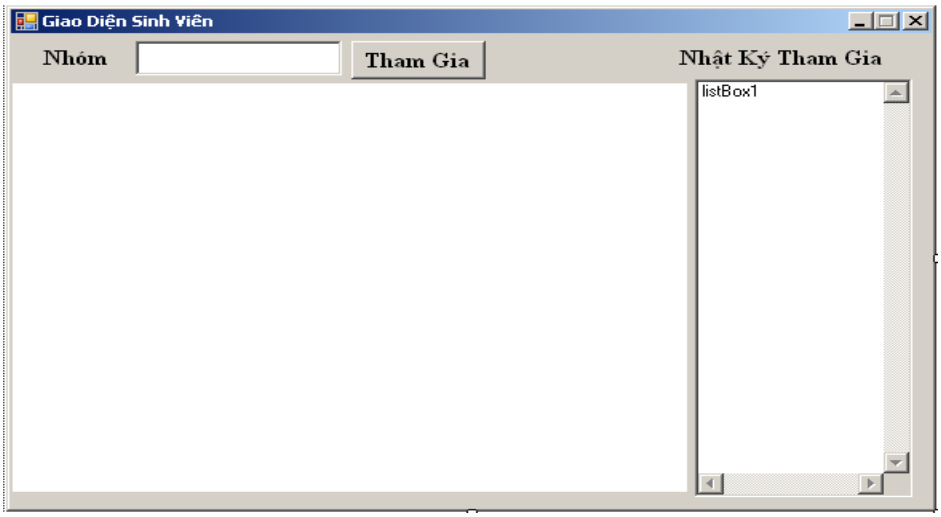


Hình 3.2.3 giao diện bắt đầu và kết thúc bài giảng

STT	Tên	Mô tả
1	Bắt Đầu	Bắt đầu bài giảng
2	Kết Thúc	Kết thúc bài giảng

3.2.2. Giao diện sinh viên

3.2.2.1. giao diện sinh viên tham gia bài giảng



Hình 3.2.4 – Giao diện sinh viên tham gia bài giảng

STT	Tên	Mô tả
1	Nhóm	TexBox nhập địa chỉ IP tham gia vào lớp học
2	Tham gia	Kích vào Nút Button tham gia vào lớp học
3	Nhật ký tham gia	ListBox hiển thị tên và thời gian bài giảng
4	Màn hình giao diện	Hiển thị nội dung đầy đủ của bài giảng

3.3.Thiết kế modul chương trình

3.3.1 Modul chương trình giáo viên

```
private void btnThamGiaRoiKhoi_Click(object sender, System.EventArgs e)
      {

         ep = new IPEndPoint(IPAddress.Parse(text_IP_Multicast.Text), 5000);
         if(btnThamGiaRoiKhoi.Text == "Tham Gia")
         {
               HookRtpEvents(); // 1
               JoinRtpSession(textLecture.Text + " Bắt đầu lúc " +
DateTime.Now.ToShortTimeString ()); // 2

               // Change the UI
               btnThamGiaRoiKhoi.Text = "Rời khỏi";
               //txtSend.Enabled = true;

               text_IP_Multicast.Enabled = false;
               button1.Enabled = true;
               textLecture.Enabled = false;

         }
         else
         {
            Cleanup(); // 6

            // Change the UI
            btnThamGiaRoiKhoi.Text = "Tham gia";
                //txtReceive.Clear();
            text_IP_Multicast.Enabled = true;
            text_IP_Multicast.Enabled = true;
            button1.Enabled = false;
            textLecture.Enabled = true;
            button3.Enabled = false;
            sendingSt.ForeColor = Color.Red;
            sendingSt.Text = "Dừng";
            notifyIcon1.Text = "RTP Presenter – Dừng";

          }
      }
```

3.3.1.2 Modul ngừng truyền bài giảng

```
        private void RoiPhien()
      {
```

```
        if(rtpSession != null)
        {
            // Clean up all outstanding objects owned by the RtpSession
            rtpSession.Dispose();
            rtpSession = null;
            rtpSender = null;
        }
    }
```

3.3.1.3 Modul hiển thị thông tin bài giảng qua ListBox

```
private void ShowMessage(string msg)
    {
        listBox1.Items.Add(msg);
    }
```

Nhận dữ liệu từ mạng

```
private void RtpParticipantAdded(object sender,
RtpEvents.RtpParticipantEventArgs ea)
    {
        ShowMessage(string.Format("{0} đã tham gia",
ea.RtpParticipant.Name));
    }
```

Dữ liệu bị remove

```
private void RtpParticipantRemoved(object sender,
RtpEvents.RtpParticipantEventArgs ea)
    {
        ShowMessage(string.Format("{0} đã rời khỏi",
ea.RtpParticipant.Name));
    }
```

3.3.1.4. Modul hiển thị nội dung bài giảng trên màn hình sinh viên

```
void send_img()
    {
        try
        {

            if (MotionFlag.Checked)
            {
                Image oldimage = scr.Get_Resized_Image(100,
100,scr.GetDesktopBitmapBytes());

                while (true)
                {

                    Image newimage = scr.Get_Resized_Image(100, 100,
scr.GetDesktopBitmapBytes());
```

```
                float difference = scr.difference(newimage, oldimage);
                differencelab.Text = difference.ToString() + "%";
                if (difference >= 1)
                {
                   sendingSt.ForeColor = Color.Green;
                   sendingSt.Text = "Đang gửi ...";
                   notifyIcon1.Text = "RTP Presenter – Đang gửi...";
                   rtpSender.Send(scr.GetDesktopBitmapBytes());
                   oldimage = scr.Get_Resized_Image(100, 100,
scr.GetDesktopBitmapBytes());
                }
                else { sendingSt.ForeColor = Color.Red; sendingSt.Text = "Tạm
dừng"; notifyIcon1.Text = "RTP Presenter – Tạm dừng"; }
             }
          }
          else
          {
             sendingSt.ForeColor = Color.Green;
             sendingSt.Text = "Đang gửi ...";
             notifyIcon1.Text = "RTP Presenter – Đang gửi...";

             while (true)
             {
                rtpSender.Send(scr.GetDesktopBitmapBytes());
             }
          }
       }
       catch (Exception ) {}
    }
```

3.3.2. Modul giao diện chương trình sinh viên

```
private void btnThamGiaRoiKhoi_Click(object sender, System.EventArgs e)
    {
       ep = new IPEndPoint(IPAddress.Parse(text_IP_Multicast.Text), 5000);
       if(btnThamGiaRoiKhoi.Text == "Tham gia")
       {
            HookRtpEvents(); // 1
            JoinRtpSession(Dns.GetHostName()); // 2
            // Change the UI
            btnThamGiaRoiKhoi.Text = "Rời khỏi";
            //txtSend.Enabled = true;
            //txtSend.Focus();
            text_IP_Multicast.Enabled = false;
       }
```

```
        else
        {
            Cleanup(); // 6
            // Change the UI
            btnThamGiaRoiKhoi.Text = "Tham gia";
            //txtReceive.Clear();
            text_IP_Multicast.Enabled = true;
        }
    }
```

3.4. Giao diện chương trình thực nghiệm

3.4.1. giao diện giáo viên:

H3.4.1.1. Giao diện giáo viên

H3.4.1.2. Giao diện bắt đầu khi nhập giảng

H3.4.1.3 giao diện cập nhật thời gian khi giảng

H3.4.1.4. Giao diện kết thúc bài giảng

3.4.2. Giao diện bài học của sinh viên

3.4.2.1 Giao diện bài học sinh viên

H3.4.2.1. Giao diện bài học của sinh viên

3.2.2. Giao diện sinh viên khi tham gia bài giảng

H3.4.2.2. giao diện sinh viên khi tham gia bài giảng

KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

1. Kết luận

Dựa trên tìm hiểu và nghiên cứu công nghệ truyền thông đa phương tiện ,em bước đầu xây dựng hệ thống hỗ trợ cho việc đào tạo từ xa thông qua mạng LAN.

Hệ thống cung cấp cho giáo viên những công cụ giúp việc thu hình, phát hình, và âm thanh. Hệ thống đã giúp cho giáo viên thực sự tham gia vào một lớp học ảo, giúp cho sinh viên có điều kiện giao tiếp trực tiếp với giáo viên đứng lớp.

Hệ thống cũng giúp cho giáo viên tạo ra một lớp học riêng của chính mình hoặc tạo ra một diễn đàn thảo luận. Đồng thời cũng giúp cho sinh viên có thể đăng ký làm thành viên của lớp học.Hệ thống được tổ chức thành hai phần :

- Phần Server : Giữ trách nghiệm tạo, quản lý các lớp học trong hệ thống. Làm nhiệm vụ giữ kết nối với các thành viên khác trên hệ thống.Hỗ trợ người dùng có thể truyền nhận được tín hiệu âm thanh và hình ảnh.

- Phần Client : Giữ trách nhiệm kết nối với server. Hỗ trợ người sử dụng liên lạc với các thành viên khác trong hệ thống. Nhận và phát tín hiệu âm thanh, hình ảnh cho các thành viên khác có tham gia hệ thống.

2. Hướng phát triển :

- Cải tiến chất lượng truyền thông về hình ảnh, âm thanh
- Cho phép sinh viên lưu lại bài giảng của giáo viên lên máy tính.
- Mở rộng ra mô hình toàn cầu là mạng WAN, Internet.

TÀI LIỆU THAM KHẢO

[1]. Phạm Hồng Tài, "Tự học C#" NXB Thống Kê, 01/2003

[2]. Jesse Liberty & O'Reilly, "Programming C#".

[3]. Nguyễn Gia Thiều, "Mạng máy tính", Nhà xuất bản thông tin.

[4]. Giáo Trình Thiết Kế và Xây Dựng Mạng LAN - WAN , Trung Tâm Khoa Hoạc Tự Nhiên Và Công Nghệ Quốc Gia, Viện Công Nghệ Thông Tin Quốc Gia, 2004.