電腦網路與程式課程

BC++Builder 程式設計

BCB 程式組教材

授課人員：葉書詠

目錄

第一章 - C++ Builder 程式教學

1.C 語言複習

while

```
        while（條件式）
          {  動作  }
－－－－－－－－－－－－
```

do / while

```
        do
          {  動作  }
        while（條件式）
－－－－－－－－－－－－
```

for

```
        for（起始條件；結束條件；增量）
          {  動作  }
－－－－－－－－－－－－
```

if

```
        if（條件式）
          {  動作  }
－－－－－－－－－－－
```

if...else

```
        if（條件式）
          {  動作１  }
        else
          {  動作２  }
－－－－－－－－－－－－
```

if...else if...else

```
        if（條件式１）
            {  動作１  }
        else if（條件式２）
            {  動作２  }
        else if（條件式３）
            {  動作３  }
        else
            {  動作４  }
－－－－－－－－－－－－
```

switch / case

```
        switch（條件式）
        {
          case  條件值１：  {動作１}
                            break；
          case  條件值２：  {動作２}
                            break；
```

```
    case  條件值３：   {動作３}
                       break；
    case  條件值４：   {動作４}
                       break；
   default：           {動作５}
}
```

2.C++ Builder 簡介

project manager
tool palette
strucure
object inspector

3.名詞解釋

類別(Class)
簡單的說就是模子，一個類別會定義其所屬的成員(Member)，包括屬性和方法等。

物件(Object)
物件，也就是類別的實體，照字面上的意義來說，就是一個一個的東西。例如汽車是一個物件。
在 BCB 裡面的物件，是一個一個的 VCL 元件，例如按鈕是個物件，選單也是物件。

屬性(Property)
屬性就是物件的特性、特徵。例如：汽車有顏色這個特徵，但是不同汽車可能也不同的顏色，其餘的有汽車的馬力、排氣量等等。
在 BCB 中，舉例來講，按鈕的屬性有它的顏色、位置、大小等等。

事件(Event)
事件就是會發生的事情。對於一台汽車來講，正在駕駛，就是一個事件，轉彎、煞車也是。
對 BCB 來說，按下按鈕，就是一個事件。

方法(Method)
也就是操作物件的方法
例如將汽車從紅色變成藍色。
在 BCB 來講，要將字串轉成整數，可以用 StrToInt 這個方法。

4.屬性設定簡介

Top
元件與視窗上緣的距離。

Left
元件與視窗左邊的距離。

Caption
表單上的標題，或是按鈕上的文字，或是標籤上顯示的字。

Font
更改 Caption 的字型。

Cursor
當滑鼠移到這個元件上面時會出現的游標形狀。

Enabled
設定該元件是否被啓動。舉按鈕爲例，若 Enabled 爲 True，則按下按鈕之後，會有事件被驅動。

Visible
設定該元件是否要在執行的階段爲"可見的"狀態。

Height
該元件的高度。

Width
該元件的寬度。

AtuoSize
自動設定該元件的大小

ShowHint / Hint
是否顯示提示。若 ShowHint 爲 true，則執行時，滑鼠移到這個元件上面，會出現小小的視窗顯示 Hint。
Hint 這個屬性就是顯示的內容。

Name
元件的名稱。像是 Button1、Label2、Edit1 等等。
Name 跟 Caption 沒有絕對的關係，Caption 是顯示給大家看的，Name 則是讓程式看的。

其他還有很多屬性，像是 Color、Icon 等，有些屬性還有附屬屬性，像是 Font 底下還有 Size、Color 等等

5.VCL 元件介紹

AnsiString 類別
宣告：AnsiString str;
方法：int Pos(const AnsiString& Str)：傳回 Str 第一個字元的位置，若不存在則回傳 0
AnsiString SubString(int start , int count) ：傳回 start 後 count 個字元的字串
StrToInt(AnsiString Str)：將數字字串轉爲整數，其他還有 StrToFloat 等轉換方法，通常亦會有 IntToStr(int i)等反轉換方法
範例：str.Pos("@");

Label
Edit
Button

TStringList
宣告：TStringList *List1=new TStringList;
屬性：Count ：總列數
Strings[i]：第 i 列的字串
方法：Add(AnsiString)：加入字串
Delete(i)：刪除第 i 列字串
Clear()：清除所有字串
LoadFromFile(檔名) ：讀檔
SaveToFile(檔名)：存檔

TOpenDialog
ex：

```
if(OpenDialog1->Execute())
{
    Path=OpenDialog1->FileName;
}
```

ComboBox
屬性：Item：TStringList 型態的物件，ComboBox 的控制項
ItemIndex：int 型態的變數，被選擇列的索引值
ex：被選擇的字串就是，ComboBox1->Items->Strings[ComboBox1->ItemIndex]
事件：OnSelect
OnEnter

TMemo
屬性：Lines ：TStringList 類別的物件，即是 memo 顯示出來的部分，請參考上面的 TStringList

6.程式示範

1.ShowMessage("Hello World!!");
2.Edit 輸入，Label 輸出

7.作業

(I)作業一

(II)作業二

第二章-VCL 元件介紹-(續)

1.VCL 元件介紹

LabeledEdit

GroupBox

Timer：

屬性：Enabled：bool 值，決定 timer 是否啓動
Interval：執行間隔，單位爲 1/1000 秒
事件：OnTimer

IdTCPClient：

屬性：

BoundIP：自己本機的 IP，預設爲空，也就是讓程式自己抓
BoundPort：本機的 port，預設爲 0 ，也是讓程式自己抓
Host：遠端 IP，也就是伺服器端 IP
Port：伺服器端監聽的 port
Connected：bool 值，若以連線即爲 true，反之爲 false

方法：

Connect()
DisConnect()
WriteLn(AnsiString buf)： 於 buf 字串尾加入換行字元後送出
AnsiString ReadLn(AnsiString ATerminator,int ATimeout,int AMaxLineLength)：
讀取字串， 當讀到 ATerminator 即停止，預設爲換行字元 ，當執行超過 ATimeout(微秒)終止函式，所能接收的最大字元數爲 AMaxLineLength

事件：

OnConnected OnDisconnected

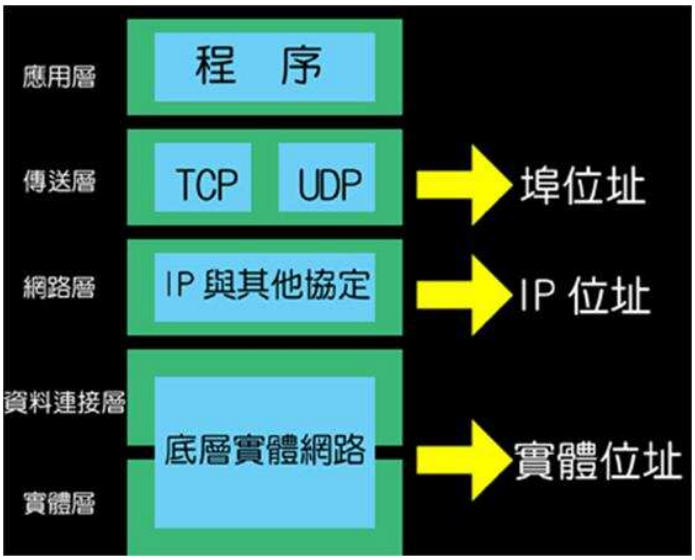
2.程式示範

3.作業

(I)作業三

第三章-使用 socket 撰寫 BCB 網路程式

TCP/IP 分層架構



- 應用層：包括有一般常見的 http，ftp，snmp 等服務的協定。
- 傳送層：
  - TCP：TCP 所採用的方式是所謂的連接導向模式，也就是好像在互相傳輸的兩點建立一條虛擬電纜一樣，TCP 提供嚴謹，可靠的傳輸。
  - UDP：UDP 採用的是無連接模式，傳輸雖然較不可靠，但卻相對簡單，處理速度較快，在錯誤率低的短距離區網內， 或是在大量傳輸中可以允許有少量錯誤的情況中，UDP 的效果就可能比 TCP 來得好。
- 網路層：
  - IP：TCP 及 UDP 負責將來自某個程序的訊息送至另一個程序，而 IP 則負責將來自某台電腦的程序送往另一台電腦。
- 底層實體網路：負責實際上物理訊號的傳輸，包括了像是光的波長，頻率，或是無線電的周波數，強度之類的定義。

Socket

socket 介於應用層與傳輸層之間 ，是一個網路系統的通訊函式庫，在任何作業系統中可以通用

主要的函式：

socket() <-> 電話
bind() <-> 線路(第幾分機?)
listen() <-> 準備好接聽(啓用鈴聲)
connect() <-> 撥電話出去
accept() <-> 接聽
send()/recv() <-> 通話
closesocket() <-> 掛斷

在快樂的使用函式之前，有幾件事情要先注意

#include<winsock2.h>

winsock2.h 內定義了一些有用的巨集以及 socket 資料結構

int WSAStartup(WORD *wVersionRequested*,LPWSADATA *lpWSAData*);

- 參數：
  - wVersionRequested 可使用的 Windows Sockets API 最高版本
  - lpWSAData 指向 WSADATA 資料的指標
- 傳回值：
  - 成功： 0
  - 失敗： WSASYSNOTREADY / WSAVERNOTSUPPORTED / WSAEINVAL

- 說明：WSAStartup()是連結應用程式與 Windows Sockets DLL 的第一個函式。 此函式「必須」是應用程式呼叫到 Windows Sockets DLL 函式中的第一個， 也唯有此函式呼叫成功後，才可以再呼叫其他 Windows Sockets DLL 的函式。 此函式亦讓使用者可以指定要使用的 Windows Sockets API 版本，及獲取設計者的一些資訊。

範例：出自 msdn library：http://msdn2.microsoft.com/en-us/library/ms742213.aspx

```
WORD wVersionRequested;
WSADATA wsaData;
int err;
wVersionRequested = MAKEWORD( 2, 2 );
err = WSAStartup( wVersionRequested, &wsaData );
if ( err != 0 ) {
/* Tell the user that we could not find a usable */
/* WinSock DLL. */
return 1;
}
/* Confirm that the WinSock DLL supports 2.2.*/
/* Note that if the DLL supports versions greater */
/* than 2.2 in addition to 2.2, it will still return */
/* 2.2 in wVersion since that is the version we */
/* requested. */
if ( LOBYTE( wsaData.wVersion ) != 2 ||
HIBYTE( wsaData.wVersion ) != 2 ) {
/* Tell the user that we could not find a usable */
/* WinSock DLL. */
WSACleanup( );
return 1;
}
/* The WinSock DLL is acceptable. Proceed. */
```

int WSACleanup(void);

- 參 數： 無
- 傳回值：
  - 成功： 0
  - 失敗：SOCKET_ERROR (呼叫 WSAGetLastError() 可得知原因)

- 說明： 應用程式在使用 Windows Sockets DLL 時必須先呼叫 WSAStartup() 來向 Windows Sockets DLL 註冊； 當應用程式不再需要使用 Windows Sockets DLL 時，須呼叫此一函式來註銷使用，以便釋放其占用的資源。

開始使用 socket

int socket(int af, int type, int protocol);

- 參 數：
  - af ：位址資料族系(family)，用不同方式表示網路位址，請用 AF_INET，表示是普通的 internet 位址
  - type：通訊方式
    - SOCKET_STREAM：代表 TCP
    - SOCKET_DGRAM：代表 UDP
  - Protocal：傳輸協定編號 選擇 IPPROTO_TCP (TCP 通訊協定) 或寫入 0，交由系統設定
- 回傳值：-1 表示建立 socket 發生錯誤 若成功則回傳非負整數，稱爲 socket descriptor(socket 描述子)

範例：

```
SOCKET sock;                                    //宣告

sock=socket(AF_INET,SOCK_STREAM,0);     //設定
```

int bind(SOCKET s, const struct sockaddr* name, int namelen);

- 參 數：
  - s : 指定好通訊協定的 socket
  - name : 指定本地端位址，資料格式爲 sockaddr
  - namelen : name 之資料長度(單位 byte)

- 回傳值：-1 表錯誤，否則爲 0

sockaddr 原型(保留各協定自行定義空間)

```
struct sockaddr{
u_short sa_family; // address family
char sa_data[14]; // 位址內容（未定義）
}
```

sockaddr for IPv4：

```
struct sockaddr_in {
short sin_family;
u_short sin_port;
struct in_addr sin_addr;
char sin_zero[8]; };
```

sin_family:位址資料族系，同樣設定爲 AF_INET
sin_port:主機開啓的通訊埠號 用 htons() 寫入
sin_addr:主機 IP 位址 in_addr 資料格式
sin_zero[8]:目前沒用處，保留以後使用

範例：

```
struct sockaddr_in sa;            //先宣告一個 sockaddr_in 作爲引入值
sa.sin_family=AF_INET;            //設定爲 internet 位址族系
sa.sin_port=htons(31800);         //htons()是將數字轉換成網路位元排列
sa.sin_addr.s_addr=INADDR_ANY;    //INADDR_ANY 表示我不在意 Local IP，由系統自行決定
                                  //若要指定 IP，則使用 inet_addr 將 IP 字串轉爲網路位元排列，
                                  //例如：sa.sin_addr.s_addr = inet_addr("140.115.65.1");
                                  //如需要反轉換，有 inet_ntoa 函式可用
bind(sock,(sockaddr *)&sa,sizeof(sa)); //bind
```

int listen(SOCKET s, int backlog);

- 參 數：
  - s：設定好 bind(),並且尚未連線的 socket
  - Backlog：等待 Server 接受連線前，同時最大連線數
- 回傳值：-1 表錯誤，否則爲 0

範例：listen(sock,1);

SOCKET accept(SOCKET s, struct sockaddr* addr, int* addrlen);

- 參 數：
  - s：一個設定爲 listen 狀態的 socket
  - addr：Client 端位址資訊，由函式自動產生填入
  - addrlen：sddr 長度，由函式自動產生
- 回傳值：-1 表示錯誤，否則傳回另一個包含 Client 端資訊的新 socketdescriptor，作爲傳送資料用

傳進 accept()的 listen socket 本身並沒有辦法作資料的傳輸，所以必須透過 accept()產生一個包含通訊協定、Server、Client 資訊的新 socket，利用他就可以進行資料的傳輸了

範例：accept(Message.WParam,NULL,NULL);

int connect(SOCKET s,const struct sockaddr* name,int namelen);

設定方式請參照 bind()函式

回傳值：-1 表錯誤，否則回傳 0

int recv(SOCKET s, char* buf, int len, int flags);

- 參 數：
  - s：一個建立連線成功的 socket
  - buf：呼叫 recv，用來儲存收到資料的暫存器
  - len：buf 的長度(byte)
  - flags：選擇工作模式，一般填入 0
- 回傳值：-1 表錯誤，否則傳回接受到資料的長度(byte)

範例：

```
#defined MSG_LEN 100                    //定義接收訊息的最大長度
//-------------------------------------------------------------------------------
int bytes_of_read;                      //宣告變數儲存 recv 的回傳值
char buf[MSG_LEN];                      // 宣告暫存區
bytes_of_read=recv(sock,buf,1024,0);    //recv
```

int send(SOCKET s,const char* buf, int len, int flags);

- 參 數：
  - s：一個建立連線成功的 socket
  - buf：用來儲存將送出資料的暫存器
  - len：buf 的長度(byte)
  - flags：選擇工作模式，一般填入 0
- 回傳值：-1 表錯誤，否則傳回送出資料的長度(byte)

範例：

```
int bytes_of_sent;                  //宣告變數儲存 send 的回傳值
bytes_of_sent=send(sock,Edit1->Text.c_str(),Edit1-
```

```
>Text.Length,0);    //send
```

int closesocket(SOCKET s);
int shutdown(SOCKET s,int how);

- 參數：
  - s：使用中的 socket
  - how：控制 socket 工作的方式
    - SD_RECEIVE 禁止輸入(disable recv()函式)
    - SD_SEND 禁止輸出(disable send()函式)
    - SD_BOTH 雙向禁止
- 回傳值：-1 表錯誤，否則傳回 0
- 說明
  closesocket()可以用來終止 TCP 連線，但不會馬上關閉，必須等到該 socket 不在動作後才切斷連線，這是用完再關的函式， 而 shutdown()是有強制性質的中斷連線函式，用來控制 socket 的 IO。

一個好的中斷連線作法應有四步：

1. 結束傳送資料
2. 使用 shutdown()，設定爲禁止送出資料
3. 呼叫 recv()，確定收到的資料長度爲 0，避免遺漏資訊
4. closesocket() 來關閉 socket

範例：

```
void __fastcall TForm1::SocketGracefulClose(SOCKET s)
{
 int i=0;
 bool flag=true;
 AnsiString buf;
 shutdown(s,SD_SEND);
 i=recv(s,buf.c_str(),1024,0);
 if(i==-1);
 else
 {
  while(flag)
  {
   i=recv(s,buf.c_str(),1024,0);
   if(i==0)
   {
    flag=false;
   }
  }
 }
 closesocket(s);
//Memo1->Lines->Add("socket closed");
}
```

第四章-作業 - 聊天室 參考協定

0:取得列表
命令格式:
```
+---+----+
| 0 | SP |
+---+----+
```
流程:
```
client----------------------server
==================================
0lspace---------------------->
        <-------------------ack(0)
        ack(0)---------------->
        <-------------------count
ack(0)----------------------->
        <-------------------(loop)writeln
ack(0)----------------------->
```

1:一般談話:
命令格式:
```
+---+----+-----+
| 1 | SP | MSG |
+---+----+-----+
```
流程:
```
client----------------------server
==================================
1lspacelmsg---------------------->
        <-------------------ack(0)
```

2:悄悄話:
命令格式:
```
+---+----+-----------+----+-----+
| 2 | SP | target id | SP | MSG |
+---+----+-----------+----+-----+
```
流程:
```
client----------------------server
==================================
2lspaceltargetlspacelmsg------>
        --------------------find(id)(get ip&port)
        <-------------------ack(0)
```

3:送出暱稱及驗證:
命令格式:

+---+----+----+
| 3 | SP | id |
+---+----+----+
流程:
client----------server
==========
3lspacelid---------->
<----------IDtest(1/0)//回傳 1 即爲通過,0 則爲已有重複的暱稱

4:送出震動:
命令格式:
+---+----+------------+
| 4 | SP | target id |
+---+----+------------+
流程:
client----------server
==========
4lspaceltarget id---------->
<----------ack(0)