بسم الله الرحمن الرحیم

مقالات

C++

منبع:

www.Dev.ir

نمایش Menu روی Tray Icon

قرار بود نمایش Menu روی Tray Icon رو بگم. کار ساده تر از این حرفاست.

توی کلاس دیالوگتون بصورت public تابع زیر رو اضافه کنید :

```
virtual LRESULT WindowProc(UINT message, WPARAM wParam, LPARAM lParam);
```

این برا اینه که message ها رو handle کنیم (دیگه چون یکم عکس گذاشتن سختم بود دستی میگم انجام بدید که عکس نخواد)

حالا تابع زیر رو توی کد اصلی بنویسید که قراره menu رو اینجا نمایش بدیم :

```
LRESULT اسم کلاس برنامه تونDlg::WindowProc(UINT message, WPARAM wParam,
LPARAM lParam)
{

 return CDialog::WindowProc(message, wParam, lParam);
}
```

حالا باید یه menu ساخته باشید و با استفاده از ID اون و چند خط کد زیر می تونید menu رو نمایش بدید (اینجا من از پیغام کلیک راست روی آیکون استفاده کردم) :

```
CMenu mymenu,*pmenu;
   if (message == myicon.uCallbackMessage)
   {
  if(lParam==WM_RBUTTONDOWN)
  {
   mymenu.LoadMenu(IDR_MENU1);
   pmenu=mymenu.GetSubMenu(0);
   CPoint pos;
   #ifdef _WIN32_WCE
    pos = CPoint(GetMessagePos());
   #else
    GetCursorPos(&pos);
   #endif
   ::TrackPopupMenu(pmenu->m_hMenu, TPM_TOPALIGN, pos.x, pos.y, 0,
       this->GetSafeHwnd(), NULL);
  mymenu.DestroyMenu();
  pmenu->DestroyMenu();
  }
```

**اما myicon.uCallbackMessage چیه ؟!

اگه یادتون باشه برای ایجاد tray icon یه شی با اسم myicon ساخته بودیم همونه که uCallbackMessage رو توش ست کرده بودیم.

**IDR_MENU1 هم ID منوی مورد نظر هستش.

Modless Dialog

مطلب امروز در مورد پنجره های Modless هستش.

شاید بگید یعنی چی !!!

اگه دقت کرده باشید معمولا پیغامهای خطا یا هشدار دهنده به شکلی هستند که وقتی نمایش داده میشوند دیگه نمی تونید به پنجره برنامه اصلی دسترسی داشته باشید و غیر فعال می شود تا زمانی که به اون پیغام خطا یا هشدار یه جواب مناسب (حالا از OK گرفته تا Yes و No) بدهید. به اینگونه نمایش یه پنجره (همون پیغام خطا خودش یه پنجره هستش دیگه) روش Modal میگن.

حالا بعضی مواقع نیاز دارید که پنجره هایی رو نشون بدید که همزمان نیاز باشه که هم روی پنجره اصلی برنامه و هم روی این پنجره جدید نمایش داده شده کار شود (یه مثال خوب و روشن برای این نوع یاهو مسنجر هستش که این صفحه اصلی رو در ابتدا دارید و به ازای هر PM که باز می کنید یا به شما داده میشه یه پنجره باز میشه که باید بتونید روی هرکدوم که خواستید کار کنید) به این نوع نمایش پنجره Modless میگن.

کار خیلی راحت و کوتاه هستش.

اول که خب باید دیالوگتون و کد های مربوط رو بنویسید!

هر موقع خواستید از یه دیالوگ یه دیالوگ دیگه رو نمایش بدید کافی هستش که چند خط زیر رو بنویسید :

```
CDialog *MyDlg=new CDialog();

MyDlg->Create(IDD_ABOUTBOX,0);

MyDlg->ShowWindow(true);
```

توضیحات :

IDD_ABOUTBOX مربوط به ID دیالوگ اصلی شما میشود که حالا اینجا من برای سادگی کار خواستم دیالوگ About که بصورت پیش فرض ساخته میشه رو نشون بدم !

اگه دقت کنید تابع Create از شما ۲ تا پارامتر می خواد که با اولی آشنا شدید. اما دومی چیه ؟! باید بگم اگه دومی رو مقدار ۰ یا همون NULL قرار بدید فرض میکنه که پدر دیالوگی که میخواهید نمایش بدید اون دیالوگی هستش که از اون قصد نمایش این دیالوگ جدید رو دارید و یکم با اینکه یه دیالوگ مستقل باشه فرق میکنه (مثلا اگه دیالوگ اصلی رو مینیمم کنید این هم همراه اون مینیمم میشه و اگه این جدید رو مینیمم کنید دیگه روی TaskBar دیده نمیشه و مثل پروژهای MDI انگار که دیالوگ اصلی شما برنامه Container اصلی باشد دیالوگ جدید به همون شکل مینیمم میشه ! حالا تست کنید بهتر دستگیرتون میشه!). اما اگه بخواهید عین صفحه PM در YAHOO! Messeneger وقتی دیالوگ جدید رو نمایش میدید بر روی TaskBar هم دیده بشه باید یه اشاره گر به Desktop بهش بدید که با تابع GetDesktopWindow که هیچ پارامتر ورودی هم نیاز نداره این کار میسر میشه.

Tray Icon

کنار ساعت ویندوز رو نگاه کن چند تا ازشون میبینی . آره همون آیکون ها رو می گم.

خیلی کار راحتیه!

اول خط زیر یادتون نره :

```
#include <SHELLAPI.H>
```

برا اینکار یه آیکون درست کنید یا از همون آیکون استاندارد فعلا استفاده کنید و ID اونو از قسمت resource و در پوشه Icon پیدا کنید. بصورت استاندارد یه آیکون با ID (IDR_MAINFRAME) ایجاد میکنه که آیکون MFC هستش.

حالا باید در این ۳ خط زیر هر جا IDI_ICON1 که آی دی انتخابی من برا آیکون مورد نظرم بوده, آی دی آیکون خودتون رو بگذارید:

```
HICON hIcon;

HINSTANCE hInst =
AfxFindResourceHandle(MAKEINTRESOURCE(IDI_ICON1),RT_GROUP_ICON);

hIcon =
(HICON)LoadImage(hInst,MAKEINTRESOURCE(IDI_ICON1),IMAGE_ICON,16,16,LR_DE
FAULTCOLOR);
```

hIcon رو قراره به یه تابع بدیم که بدونه کدوم آیکون رو میخواهیم برامون بگذاره.

حالا یه متغییر بصورت زیر تعریف کنید :

```
NOTIFYICONDATA myicon;
```

با این متغییر یکسری اطلاعات به تابع مربوط به کار مورد نظر ما پاس میشه.

حالا مقادیری که باید مقدار دهی کنید رو دونه دونه جلوی هر خط مثال زیر توضیح میدم :

```
myicon.hWnd=this->GetSafeHwnd();//
```

هندلی به دیالوگی می خواید پیغامهای ویندوز به اون منتقل بشه که برای استفاده از منو روی آیکون مورد نظر اهمیت پیدا میکنه

```
myicon.cbSize=sizeof(NOTIFYICONDATA);//
```

اندازه متغییر NOTIFYICONDATAرو که مقدار دهی میکنه که همیشه همینه

```
myicon.uFlags=NIF_MESSAGE | NIF_ICON | NIF_TIP | NIF_INFO;//
```

امکاناتی که می خواید این آیکون داشته باشه از جمله پیغامهای ویندوز و نمایش خود آیکون و نمایش بالون محتوی توضی در مورد آیکون

myicon.uCallbackMessage=IDD_DIALOG1;//
آی دی دیالوگی که پیغامهای ویندوز به اون منتقل میشه

myicon.hIcon=hIcon;//
متغییری که بالا برای آیکون خودمون تنظیم کردیم

myicon.szInfoTitle="My Title";//
تیتر بالونی که روی آیکون ایجاد میشه

myicon.szInfo="My information";//
متن اصلی بالونی که روی آیکون ایجاد میشه

myicon.szTip="Mouse over information (Tool Tip !)";//
متنی که وقتی موس روی آیکون نگهداشته میشه نشمون میده

myicon.uTimeout=20;//
زمانی که بالون نمایش داده میشه

Shell_NotifyIcon(NIM_ADD,&myicon);//
با این تابع آیکون در کنار ساعت نمایش داده میشه

برای مخفی کردن دیالوگ هم از تابع زیر استفاده کنید :

ShowWindow(SW_HIDE);

** نکته:

باید بگم که موارد بالون و ToolTip در VisualC++6 با مشکل همراه هستش و ظاهرا support نمی کنه و باید این رو توی VisualC++.NET انجام بدید

DialUp

امروز میخوایم یه برنامه ساده بنویسیم که یوزر و پسورد و شماره تلفن بگیره و خودش شروع کنه به اینترنت وصل بشه.

اول که باید یه پروژه جدید بسازید و ۳ تا edit box برای اطلاعات یوزر و پسورد و شماره تلفن.

اولین کاری که باید کنید اینه که ۲-۱ کلاس رو به پروژه اضافه کنید که با ۲خط زیر انجام میدید:

```
#include "ras.h"
#include "raserror.h"
```

حالا ۲ تا دکمه بگذارید برای وصل شدن و قطع ارتباط.

برای وصل شدن از چند خط ساده زیر استفاده کنید که هر خط رو جداگانه توضیح میدم:

```
 RASDIALPARAMS rdParams//;
```
متغییری(شئی) است برای دادن اطلاعات مورد نیاز برای کانکت به اینترنت

```
 rdParams.dwSize = sizeof(RASDIALPARAMS);
rdParams.szEntryName[0] = '\0';

lstrcpy( rdParams.szPhoneNumber, m_strPhoneNumber );//
```
در این قسمت شماره تلفن را مقدار دهی میکنیم

```
rdParams.szCallbackNumber[0] = '\0';

lstrcpy( rdParams.szUserName, m_strUserName );//
```
در این قسمت یوزررا مقدار دهی میکنیم

```
lstrcpy( rdParams.szPassword, m_strPassword );//
```
در این قسمت پسورد را مقدار دهی میکنیم

```
rdParams.szDomain[0] = '\0';

HRASCONN hRasConn = NULL;//
```
این متغییر برای نسبت دادن یک هندل به این کانکشن در صورت موفقیت خوهد بود که بتوان در آینده از آن استفاده کرد مثلا آنرا قطع کرد

```
DWORD dwRet = RasDial( NULL, NULL, &rdParams, 0L, NULL, &hRasConn );//
```
در این قسمت شروع به شماره گیری میکند و نتیجه شماره گیری بازگردانده می شود که در صورت موفقیت صفر و در غیر این صورت شماره خطلا خواهد بود.

```
if ( dwRet == 0 )  return true;
    char  szBuf[256];

if ( RasGetErrorString( (UINT)dwRet, (LPSTR)szBuf, 256 ) != 0 )//
```

اگر منجر به خطا شد با استفاده از شماره خطا متن خطا را بدست آمرده نمایش می دهیم و کانکشن را می بندیم.

```
wsprintf( (LPSTR)szBuf, "Undefined RAS Dial Error (%ld).", dwRet );
 RasHangUp( hRasConn );
 MessageBox( (LPSTR)szBuf, "Error", MB_OK | MB_ICONSTOP );
 return false;
```

تا ایجا نحوه خیلی ساده اتصال رو دیدیم. حالا باید در صورت نیاز امکان قطع کردن رو هم فراهم کنیم که اونم به سادگی کد زیر هستش:

```
RASCONN ras[20];
 DWORD  dSize, dNumber;
 char  szBuf[256];

 connected=false;
 ras[0].dwSize = sizeof( RASCONN );
 dSize = sizeof( ras );   // Get active RAS - Connection
 DWORD  dwRet = RasEnumConnections( ras, &dSize, &dNumber );
 if ( dwRet != 0 )
 {
  if ( RasGetErrorString( (UINT)dwRet, (LPSTR)szBuf, 256 ) != 0 )
   wsprintf( (LPSTR)szBuf, "Undefined RAS Enum Connections error (%ld).", dwRet );
  MessageBox( (LPSTR)szBuf, "RasHangUp", MB_OK | MB_ICONSTOP );
  return false;
 }
 bool bOK = true;
 for( DWORD dCount = 0;  dCount < dNumber;  dCount++ )
 {    // Hang up that connection
  HRASCONN hRasConn = ras[dCount].hrasconn;
  DWORD dwRet = RasHangUp( hRasConn );
  if ( dwRet != 0 )
  {
   char  szBuf[256];
   if ( RasGetErrorString( (UINT)dwRet, (LPSTR)szBuf, 256 ) != 0 )
    wsprintf( (LPSTR)szBuf, "Undefined RAS HangUp Error (%ld).", dwRet );
   MessageBox( (LPSTR)szBuf, "RasHangUp", MB_OK | MB_ICONSTOP );
   bOK = false;
  }
 }
 return bOK;
```

با این ۲ تابع ساده میتونید به اینترنت متصل شید. اما نکاتی که هست اینه که :

اولا اصلا کانکشن ویندوزی ساخته نمیشه.

دوماً اینکه ایکون اتصال هم نمایش داده نمیشه و دست شمارو باز میگذاره که هرجور دوست دارید اعمال سلیقه کنید.

مقدمه ای بر پردازش تصویر

اولین گام پردازش تصویر کار با رنگ نقاط (pixel) هستش. اما باید بگم توی حالت عادی شما ۳ درجه رنگی از ۳ رنگ اصلی آبی و سبز و قرمز برای هر نقطه دارید که با کم و زیاد شدن شدت هر مولفه اصلی رنگ حاصل تغییر میکنه . اما تو پردازش تصویر این فرمت اصلا به درد نمی خوره ! دلیلش اینه که شما معمولا نیاز به تشخیص یه محدوده رنگی خاص دارید مثلا رنگ حول و حوش صورتي, حالا با ۳ رنگ اصلی چطور می خواید این محدوده رو تعیین کنید ؟! میشه گفت محاله ممکنه یا بهتر بگم اصلا عقلانی نیست دنبالش برید.!!!

پس باید دنبال شیوه ای برای اون باشیم!

برای هر رنگ ما ۳ مشخصه می تونیم نام ببریم :

۱- نام رنگ (چیز بهتری گیر نیووردم بهش بگم -> Hue) :

این یعنی اینکه رنگ چیه ! مثلا میگی آبی یا آبی یکم متمایل به سبز یا ....

۲- شدت رنگ :

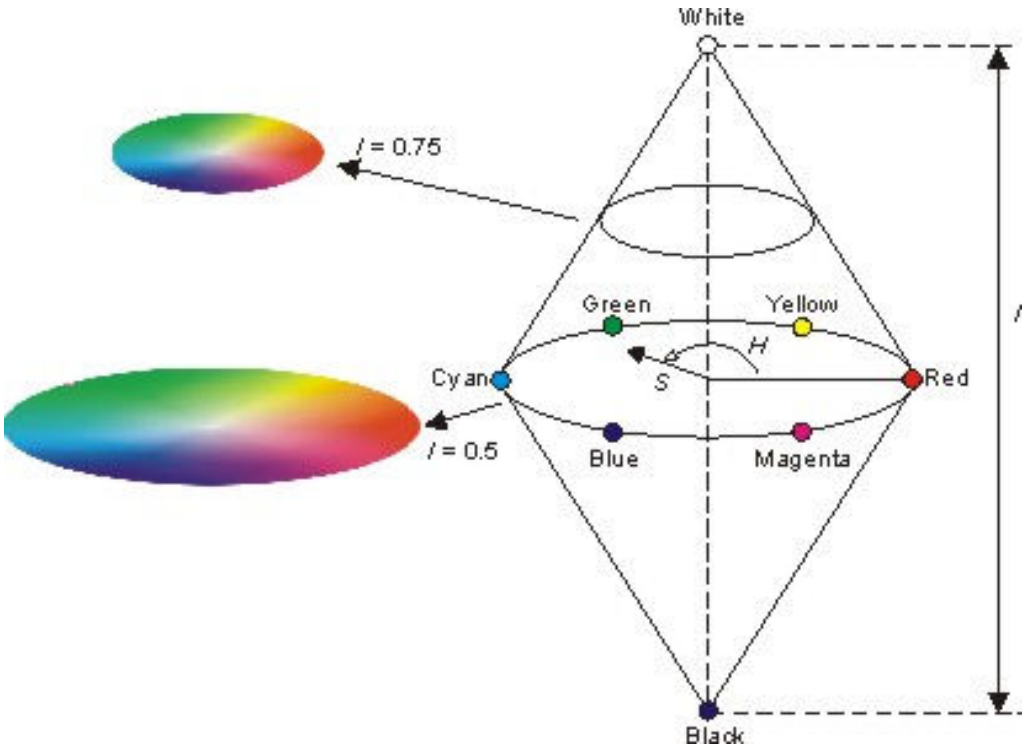
اگه دقت کرده باشید هر رنگ میتونه پر رنگ یا کم رنگ باشه اما ماهیت ذاتی اون یه چیزه و فقط کم رنگ تر یا پررنگ تر شده به این میگن -> Saturation

۳- روشنایی یا تیرگی رنگ :

یه رنگ رو می تونید بهش نور بتابونید که روشناییش زیاد بشه یا نور رو کم کنید که روشناییش کم بشه که میتونه انقدر کم بشه که سیاه بشه !! به این میگن->Intensity

امیدوارم تونسته باشن خوب برسونم منظورم چیه !

پس یه حوضه جدید که الهام گرفته از چشم هستش رو باهاش آشنا شدیم که کار روی رنگها برای تصمیم گیری روی مشخصات اون رو راحت میکنه. به این حوضه رنگی اصطلاحا HSI میگن که مخفف اون ۳کلمه بالاست. شکل زیر نمودار تغییرات رنگ رو با توجه به این ۳ مولفه نشون میده:



H بین ۰ تا ۳۶۰ هستش یعنی از قرمز تا سبز ۱۲۰ درجه و از سبز تا آبی ۱۲۰درجه و از آبی تا قرمز ۱۲۰ درجه در جهت خلاف جهت حرکت عقربه های ساعت.

S بین ۰ تا ۱۰۰ هستش که از کم رنگ (۰) تا پر رنگ (۱۰۰)

I از ۰ تا ۱۰۰ هستش یعنی از تاریک(۰) تا روشن(۱۰۰)

چند نمونه از رنگها در ۲ حوضه مذکور :

رنگ	مقادیر RGB	مقادیر HSI
سیاه	(۲۵۵،۲۵۵،۲۵۵)	(۰،۰،۰)
سفید	(۲۵۵،۲۵۵،۲۵۵)	(۰،۰،۱۰۰)
قرمز	(۲۵۵،۰،۰)	(۰،۱۰۰،۱۰۰)
سبز	(۰،۲۵۵،۰)	(۱۲۰،۱۰۰،۱۰۰)
آبی	(۰،۰،۲۵۵)	(۲۴۰،۱۰۰،۱۰۰)
قهوه ای!	(۶۴،۱۲۸،۱۲۸)	(۱۸۰،۵۰،۵۰)

حالا نیاز به این داریم که RGB مربوط به هر نقطه را به HSI تبدیل کنیم که رابطش رو به صورت سی نوشتم که r,g,b همون ۳ رنگ آبی و سبز و قرمز هر نقطه هستش:

```
int min,max;

//////////////////////
```

```
if(r>g)
{
max=r;
min=g;
}
else
{
max=g;
min=r;
}
if(b>max)
max=b;
if(b<min )

min=b;
//////////////////
if(max==0)
{
i=0;
s=0;
h=0;
}
else
{
```

```
i=max*100/255;
s=(max-min)*100/max;
h=180*acos((((r-g)+(r-b))/2)/sqrt(pow(r-g,2)+(r-b)*(g-b)))/3.1415;
if(b>g)
h=360-h;
}
```

تغییر رنگ پس زمینه دیالوگ

تا حالا سعی کردید توی ویژوال سی رنگ پس زمینه رو عوض کنید و رنگ دلخواهتون رو براش تعیین کنید ؟

این کار مستقیما توی ویزارد نیست اما با یک خط کد نویسی قابل انجام هستش !

پوزه شما ۲ تا فایل cpp اصلی داره که عبارتند از projectname.cpp و projectnameDlg.cpp که منظور از projectname نام پروژه ای هستش که انتخاب کردید.

توی فایل projectname.cpp قبل ار دستور;() int nResponse = dlg.DoModal کافیه این یک خط رو بنویسید :

SetDialogBkColor(RGB(255,0,123),RGB(12,56,212));

RGB اول رنگ پس زمینه را مشخص میکنه و RGB دوم رنگ متن دیالوگ رو تعیین می کنه .

Socket - Socket Programming چیست ؟

در ارتباطات شبکه قراردادهایی استفاده می شود تحت عنوان Protocol , پروتکولهای مختلفی ایجاد شده اما در ارتباطات اینترنتی و شبکه هایی که ارتباط ۲ طرفه است بیشتر از نوع (TCP/IP) استفاده میشه که اساس اون به صورت خیلی مختصر اینه که برای هر کامپیوتر که به شبکه (که اینترنت هم یه شبکه هستش !!) وصله یه IP مخصوص به خودش رو داره و به نوعی آدرس اون کامپیوتر محسوب میشه که تمام خصوصیات یه آدرس رو که میشه متصور شد داراست .
اما چرا گفتم در شبکه هایی که ارتباط ۲ طرفه نیازه از (TCP/IP) استفاده میشه ؟
چون انواع دیگری هم از جمله UDP هستند که تقریبا ارتباط یکطرفه میباشد مثلا برای پخش برنامه های رادیویی .

اصلی ترین عامل در یک ارتباط شبکه ای Socket میباشد که اعمال شبکه را بصورت خواندن و نوشتن در یک فایل شبیه سازی نموده است . سوکت در اصل مانند یک کانال ارتباطی است که میان دو نقطه ایجاد شده و پیغامها رد و بدل میشود.
برای داشتن یک ارتباط شبکه ای باید یک سوکت ایجاد کنیم (منظور نرم افزاری هستشا !!)
که لازمه این کار اینه که بدانیم برای گوش دادن داریم ایجاد میکنیم یا برای فرستادن پیام .
اگه برای فرستادن پیام باشه که پر واضحه که نیاز به آدرس مقصد داریم . اما این کافی نیست !!!
امروزه آپارتمان نشینی رونق گرفته و تو کامپیوتر هم نفوذ کرده !!!!
برای فرستادن نامه امروزه تنها آدرس ساختمون طرف کافی نیست و نیاز به شماره واحد اون هو هست .
این فقط یه تمثیل بود ! درسته تو کامپیوتر هم فقط IP کافی نیست و شماره واحد برنامه ای که گوش به زنگه هم نیازه که به اون Port میگن . یعنی شماره پورت هم نیازه .
اما یه چیز یادم رفت بگم که IP از ۴ عدد ۰تا۲۵۵ تشکیل شده که با نقطه از هم جدا میشن مثلا : ۱۶۸.۲۳.۵۶.۹۸ یه IP هستش . اما پورت اعدادی بین ۱ تا (بصورت رایج) ۶۵۰۰۰ را میتونه شامل بشه .(بله بزرگتر از این عدد هم میتونه شماره پورت بشه اما چه کاریه !!!)

اینجا ممکنه یه سوال پیش بیاد که مگه نه اینکه هر کامپیوتر فقط یه IP داره پس یعنی در یه لحظه آیا فقط به یه کامپیوتر میتونه وصل شه ؟؟
جواب منفی است
درسته که IP یکی بیشتر نیست اما تعداد معتنابهی پورت داریم
یعنی با هر پورت به یه کامپیوتر متصل میشیم و میتونیم تا ۶۵۰۰۰ (می دونم بیشتره ;-)) ارتباط از صدقه سری پورتها داشته باشیم .

اما اگه سوکت برای گوش دادن باشه تنها کافیه که برای ایجاد سوکت بگیم با چه شماره پورتی می خوایم ایجادش کنیم .

اولین نکته ای که لازمه بگم اینه که تو استفاده از سوکت در اول ایجاد پروژه حتما باید چک باکس مربوط به WinSock رو تیک کنید که تو درس بعدی با جزئیات بیشتری خواهم گفت .

در برنامه نویسی سوکت ۲ نوع ارتباط خواهیم داشت :

الف) در هنگام دریافت یا ارسال پیغام تمام برنامه متوقف می شود که در بسیاری از موارد اصلا مطلوب نیست

ب) بر اساس Event پایه گذاری می شود که هر موقع پیغامی منتظر دریافت شدن بود آنرا دریافت کند و هر موقع پیغام آماده فرستادن بود آنرا بفرستد و کل برنامه از کار نیافتد که در اکثر موارد این روش استفاده می شود .

تمام نیازهای مارا یک کلاس به نام CAsyncSocket تامین می کند . که با ساختن شئ از آن می توانیم به امکاناتی نظیر Receive , send ,... دسترسی داشته باشیم که مفصلترش رو توضیح خواهم داد . که هرچی بعد از این از توابع اسم میبرم مربوط به این کلاس هستش

برای این نوع ارتباط که داریم بحث می کنیم ۲ جور سوکت باید بسازیم :

۱- برای فرستادن
۲- برای گوش دادن یا همون دریافت

برای ایجاد سوکت از متد;( ) Create استفاده میشه که ۲ نوع آن به این شکل است که :
۱
- برای نوع فرستادن تنها کافیست که این متد بدون پارامتر صدا زده شود : ;( ) Create
۲- برای نوع دریافت کردن ۱ پارامتر نیاز دارد و آن شماره پورت برای دریافت کردن میباشد که به طور مثال :;(۱۷۵۳) Create

بعد از ایجاد سوکت کار بعدی وصل شدن به برنامه مقصد هستش که منتظر دریافت پیغام هستش .
این کار از جانب فرستنده پیغام شروع میشه که باید متد;( ) Connect صدا زده بشه .
این متد ۲ پارامتر نیاز دارد :

۱- IP کامپیوتر مقصد یا آدرس تحت وب آن مثلا :" www.taraf.com" (اگر در یک شبکه محلی میخواید وصل بشید اسم کامپیوتر هم میتونه جای IP به کار بره) که بصورت یه رشته خواهد بود
۲- شماره پورت کامپیوتر مقصد که گوش به زنگ هستش

مثلا : ;(۱۳۶۵,"۳۹. ۱۲ .۱۶۸.۱۹۲")Connect

اما قبل وصل شدن نیازه که گوش به زنگ بشیم تا بتونیم درخواستهای اتصال رو جواب بدیم برای این کار از متد;( ) Listen بدون هیچ پارامتری استفاده میکنیم تا سوکت دریافت آماده بشه .

حالا که به اینجا رسیدیم وقتی تقاضای وصل شدن از سمت یه کامپیوتر دیگه به ما میرسه رو باید قبول کنیم :
گوش به زنگ هستیم که اگه تقاضایی رسید با متد ;( ) Accept قبولش کنیم
این متد یه پارامتر میگیره و اون یه شئ از کلاس CAsyncSocket هستش که وظیفه دریافت متن پیغامهای بعد از متصل شدن رو به عهده میگیره .

برای فرستادن پیغام از متد ;( ) Send استفاده میکنیم که ۲ پارامتر میگیرد :

۱- اشاره گر به بافر داده مورد نظر برای فرستادن است که با استفاده از تابع;( ) LPCTSTR استفاده میکنیم که یک پارامتر ورودی دارد و آن متغییری است که پیغام ما در آن است .
۲- طول بافر که همان طول پیغام می باشد

مثلا اگه پیغام در متغییری به نام str باشه به شکل زیر میفرستیم :
Send(LPCTSTR(str)وstr.GetLength()) ;

وقتی پیغام به گوش دهنده برسه یه Eventبه نام;( ) OnReceive فعال میشه که با متد ;( )Receive پیغام را دریافت می کند که ۲ پارامتر می گیرد :

۱- اشاره گر به یک بافر که میتواند یک رشته کاراکتری باشد

۲- طول بافر مورد نظر که به چه طولی از پیغام را مي خاهیم دریافت کنیم

بعد از اتمام کار با سوکت باید انرا ببندیم که این کار با متد ;( ) Close صورت میگیرد.

توابع مجازی

توابع مجازی یا Virtual Functions در ارث بری توابع کاربرد دارند که توضیح خواهم داد .
در هنگام تعریف یک تابع در یک کلاس شما می توانید در ابتدای تعریف تابع کلمه کلیدی virtual رو برای مجازی کردن تابع استفاده کنید . مثال :

```
class A{
.
.
.
public:
virtual void setage(int age);
.
.
.
{
```

نکته : کاربرد توابع مجازی مانند توابع حقیقی است و هیچ فرقی ندارد (در حالت عادی)

امااگر یک کلاس از این کلاس شما مشتق شود و تابعی با همین نام اما به صورت تابع غیر مجازی داشته باشد اتفاق زیر رخ می دهد :

اگر شئی از کلاس مشتق شده تعریف شود و تابع مشترک این دو کلاس که در اولین کلاس مجازی است فراخوانی شود : **تابع متعلق به کلاس مشتق شده اجرا خواهد شد .**
این نوع توابع در مواردی کاربد دارد که شما تشخیص می دهید که ممکن است در بعضی شرایط نیاز باشد که تابعی که در کلاس پایه تعریف میکنید تغییر ساختار موقتی داشته باشد برای یک کاربرد خاص.

ارث بری کلاسها (مشتق کردن کلاسی از کلاس دیگر)

ارث بری همونطور که از اسمش کاملا پیداست همانند قانون ارث بردن در موجودات زنده عمل می کند , یعنی همانند طبیعت یک کلاس به عنوان کلاس پدر (مادر) فرض می شود و یک کلاس به عنوان فرزند از این کلاس یک سری خصوصیات و قابلیتها را به ارث می برد (فرزند نیز دارای بعضی امکانات پدر می شود . البته میزان آن بستگی به خواست ما دارد که توضیح خواهم داد)
کلاس پدر را کلاس پایه گویند Base Class .
ارث بری را مشتق کردن نیز می گویند .

نحوه مشتق کردن یک کلاس از کلاس دیگر :

فرض می کنیم کلاسی با نام A وجود دارد :

```
class B :(type) A {
int a,d ;
public:
void Rotate (void);
{
```

نوشته بالا یعنی کلاس B از کلاس A با دسترسی نوع (type) که توضیح خواهم داد که نوع دسترسی ۳ دسته است مشتق شده یا خصوصیات و قابلیتهای آن را به ارث برده است .
۳ نوع دسترسی برای مشتق کردن وجود دارد که عبارتند از :

۱- public
۲- protected
۳- private

یعنی یکی از سه کلمه بالا به جای (type) نوشته خواهد شد.

هر کدام از این انواع دسترسی توضیحات خاص خود را دارد که عبارتند از :

۱- public :
یعنی تمام خواص عمومی و خصوصی کلاس پایه را به همان شکل به ارث می برد . بدین شکل که موارد public در کلاس مبنا (پایه) برای این کلاس جدید نیز وجود دارد و برای این نیز public خواهد بود . و تمام خواص protetted و private نیز به همین شکل می باشد که در کلاس جدید نیز هر کدام protected و private خواهند بود .

نکته مهم : در این نوع دسترسی توابع عضو کلاس جدید اجازه دسترسی به خواص protected کلاس مبنا را دارند اما به خواص private خیر !! دسترسی ندارند .

۲- protected :
در این نوع دسترسی , کلاس جدید خواص کلاس مبنا را به این شکل به ارث می برد که تنها توابع عضو کلاس جدید به فقط خواص public و protected کلاس مبنا دسترسی دارند و به خواص private دسترسی ندارند .

۳- private :
این نوع دسترسی یعنی نه شئی از کلاس جدید و نه تابع عضو کلاس جدید به هیچ چیز از کلاس مبنا دسترسی ندارند !! (عملا یعنی این نوع دسترسی یعنی اصلا مشتق نکنیم سنگین تریم!!!)

نکته بسیار مهم :
ارث بری میتواند به گونه ای باشد که یک کلاس از تعداد بیش از ۱ کلاس ارث ببرد (مشتق شود) . مثلا :

```
class B: public A , public C , protected D , public E , private F{
.
.
.
.
{
```

چند ریختی Polymorphism

چند ریختی یا Polymorphism یکی از خواص جالب در C++ محسوب میشه .
شما نمی تونید توابعی با اسامی یکسان داشته باشید مگر در یک حالت استثناء
اون هم مسئله چند ریختی هستش .
یعنی چند تابع مختلف با یک اسم یکسان تعریف می کنید اما باید دقت داشته باشید که در مقادیر ورودی توابع متفاوت باشند .
مثال :

```
void set(int a){
int b ;
b=a;
{
**************
void set(float s){
float m ;
m=s;
{
**************
int set(int f,int g){
int j,k ;
j=f;
k=g;
return k+j ;
{
**************
```

در مثالهای بالا دقت کنید که اسم توابع یکسان است و تنها در موقع صدا زدن آنها با توجه به نوع ورودی تابع , تابع مورد نظر اجرا می شود .
مثال :
اگر بنویسیم ;(1) set تابع اولی اجرا میشود

اگر بنویسیم ;(5.1) set تابع دومی اجرا میشود

اگر بنویسیم ;(90, 8) set تابع سومی اجرا میشود و مقدار ۹۸ را بر می گرداند.

سازنده ها و مخربها در یک کلاس

در یک کلاس توابعی تعریف میشه که یه دسته درست در هنگام ایجاد یک شئ از آن کلاس و دسته دیگر درست قبل از از بین بردن یک شئ از یک کلاس اجرا می شوند .
دسته اول را سازنده یا Constractor و دسته دوم را مخرب یا Destractor گویند .

اگر توابعی به عنوان سازنده و مخرب تعریف نشود . کامپایلر توابع پیش فرضی را ایجاد می کند که تنها وجود دارند و هیچ کار خاصی انجام نمی دهند اما شما می توانید سازنده یا مخرب دلخواه خود را بنویسید که درست قبل ایجاد شئ اجرا شوند یا درست قبل از تخریب شئ اجرا شوند .

برای روشن شدن مطلب یه مثال می زنم :
شاید شما بخواهید وقتی یک شئ ایجاد می کنید به یه تعداد معلوم یا نا معلومی متغییر به صورت دینامیکی از سیستم بگیرید . برای این کار آن را داخل تابع سازنده خود قرار می دهید .
حال وقتی این برنامه به اتمام می رسد یا این شئ از بین برده می شود باید این حافظه دینامیک به سیستم برگشت داده شود . پس تابعی به نام مخرب می نویسیم که قبل از اینکه شئ از بین برود حافظه را آزاد کند و به سیستم تحویل دهد .

اما چگونه بنویسیم ؟؟؟
در یک کلاس اگر تابعی با نام آن کلاس بنویسید به عنوان سازنده محسوب می شود و قبل ایجاد شئ حتما یک بار اجرا می شود .
مثلا نام کلاس شما Cat.h باشد :
سازنده شما باید تابعی به نام Cat باشد که ممکن است ورودی و یا خروجی داشته ویا نداشته باشد که به اختیار شما است .
مخرب نیز مشابه به سازنده است با یک تفاوت در نام آن که در ابتدای نام تابع حرف(~) و سپس نام کلاس نوشته می شود و بقیه ماجرا ...

اگر به یاد داشته باشید دردرسهای قبل گفته شد که در یک کلاس فقط تعریف تابع آورده می شود و خود تابع در یک فایل با پسوند cpp نوشته می شود . اما یک استثنا وجود دارد به نام توابع In Line یا یک خطی ::
می توان توابع کوتاه در حد یکی دو خط را همانجا در کنار تعریف تابع در کلاس مورد نظر نوشت .

***این معمولا برای سازنده ها و مخربهای کوتاه کاربرد دارد و لطفا از این سوء استفاده نکنید

مثال :

```
class Cat {
public:
void Cat(void){int a=10}
void ~Cat(void){a=20}
{
```

البته این تنها یک مثال است و هر کار دیگری نیز می توانید در سازنده و مخرب به عنوان یک تابع انجام دهید .

سطح دسترسی در کلاسها

در مورد سطح دسترسی به متغییرها و توابع در یک کلاس بحث خواهیم کرد
سطح دسترسی یعنی چی؟
یعنی یک کلاس برای محتویات خود یکسری دسته بندی را رعایت می کند که هر کسی به هر چیزی نتواند دسرسی داشته باشد که در بعضی موارد اگر رعایت نشود می تواند باعث بسیاری مشکلات شود.
در یک کلاس ۳ نوع دسترسی وجود دارد :
۱- Public:در این نوع دسترسی هیچ محدودیتی اعمال نمی شود و هر چیزی چه داخلی و چه خارجی می تواند از ان استفاده کند (فعلا قصد بنده معرفی دسترسی ها می باشد و برای توضیحات عمیق تر لازم به دانستن یک سری مطالب دیگر است که در روزهای آینده ذکر خواهد شد)
۲- Private: این نوع دسترسی بر عکس نوع قبل عمل می کند . یعنی غیر از توابع عضو این کلاس هیچ چیز دیگری نمی تواند به آنها دسترسی داشته باشد . مثلا وقتی یک شئ از این کلاس تعریف می کنیم از طریق شئ نمی توانیم مستقیما به ایم نوع متغییرها یا توابع دسترسی داشته باشیم اما خود توابع عضو این کلاس میتوانند در کد نویسی خود از این نوع استفاده کنند که در آینده بیشتر آشنا خواهیم شد .
۳- Protected:در این نوع نیز شئی که از کلاس تعریف می شود نمی تواند به این نوع دسترسی داشته باشد . این نوع تعریف خاص خود را دارد که بعد از بحث ارث بری قابل ذکر است و در اینجا تنها نامی از آن برای تکمیل بحث آورده شده است .
*نکته قابل ذکر این است که غالبا از دو نوع ۱و۲ استفاده می شود و از نوع ۳ خیلی کم استفاده خواهید کرد .
در کلاس این انواع دسترسی با ۳ کلمه کلیدی ذکر شده تعیین می شوند :
اگر در ابتدای کلاس باشیو و هیچ کدام را ننویسیم متغییرها و توابع تعریفی تا کلمه کلیدی دیگر همه private محسوب می شوند تا زمانی که از یکی از دو کلمه دیگر استفاده شود . بعد از آن نیز بقیه از این کلمه استفاده شده تبعیت می کنند تا کلمه کلیدی بعدی .

در ادامه یک نمونه مثال از سطح دسترسی آورده شده و در ادامه آن مثالهای اشتباه و درست نیز آورده شده است :

```
class Cat
{
int a,b;
void setage(int age);
public:
int c,d;
void setlength(int length);
protected:
int e;
void setwidth(int width);
{
```

در بالا یک نمونه کلاس آورده شده حالا یک شئ از آن تعریف کرده و مثالهای درست و غلط را ذکر میکنم :

```
Cat m;
```

مثالهای درست :

```
m.a=10;
m.b=80;
m.setage();
```

مثالهای غلط :

```
m.b=20;
m.c=13;
m.setlength();
m.e=90;
m.setwidth();
```

include کلاس چیست ؟

۱- کلاسها به صورت فایلهایی با پسوند" h " به طور مثال "cat.h " ذخیره میشوند و برای استفاده از آن در برنامه نیاز است که آنرا به برنامه ضمیمه کنید به طورمثال :(#include " cat.h")
۲- اما در داخل فایل کلاس چه چیزهایی نوشته می شود :
در کلاس دو مسئاله تعیین می شود :
الف) متغییرهای لازم
ب) توابع مورد نیاز
در مورد تعریف متعییرها باید گفت که همانند تعریف متغییر در بر نامه نویسی غیر شئ گرا می باشد اما یک نکته مهم را باید مد نظر داشت و آن این است که در برنامه نویسی شئ گرا در کلاس شما حق مقدار دهی به یک متغییر در حین تعریف آن را ندارید . به مثال زیر توجه کنید :

int a=10; این در برنامه نویسی عادی مشکلی ندارد اما در یک کلاس حق چنین کاری نداریم و باید به شکل زیر بنویسیم :
int a;
چگونگی مقدار دهی به این متغییر در بحث شئ مورد بررسی قرار می گیرد .
و اما توابع مورد نیاز نیز روش خاص خود را داراست .
در یک کلاس شما فقط باید تابع را تعریف کنید و کد نویسی بدنه تابع در داخل کلاس انجام نمی شود . در ادامه یک نمونه از یک کلاس ساده آورده شده است :

```
class cat{

int age;
int weight,length;

void setage(int x);
int getage(void );
void Meow(void);

}
```

همانطور که مشاهده کردید در یک کلاس تنها تعریف متغییرها و توابع صورت می گیرد .
این نوشته به صورت فایلی با پسوند "h " ذخیره می شود بطور مثال : cat.h

حالا باید بتونیم توابع تعریف شده رو بصورت کامل بنویسیم .
برای این کار یک فایل هم نام با نام کلاس می سازند با پسوند cpp مثلا cat.cpp
در این فایل جدید ابتدا نوشته می شود " #include "cat.h ادیگر هدر فایلهای مورد نیاز نیز که جزء ملزومات برناه نویسی است نیز باید نوشته شود
بعد از آن نوشتن بدنه فایلها شروع می شود که همانند سی معمولی است اما با کمی تفاوت جزئی :
بعد از تعیین نوع تابع باید نام کلاس نوشته شده بعد ۲ تا علامت :: گذاشته و اسم تابع و بقیه مجرا نوشته شود . مثلا :

```
void cat::setage (int a)
{
age=a;
{
```