آموزش مقدماتی ویژوال سی پلاس پلاس ۲۰۰۸

بهزاد جناب

مقدمه

دنیای هر انسانی به اندازه وسعت فکر اوست

سلام

از آنجایی که کتاب الکترونیک خوبی به زبان فارسی در زمینه ویژوال سی پلاس پلاس در دست نیست تصمیم گرفتم تا در کنار یادگیری این زبان آموخته های خودم را به صورت یک کتاب درآمورم تا علاقه مندان به برنامه نویسی مانند خودم بتوانند از آن برای یادگیری این زبان استفاده کنند.در تهیه این کتاب از سه منابع زیر استفاده کرده ام.

۱. کتاب آموزش ویژوال C++ 6 در ۲۱ روز دیویس چاپمن انتشارات نص
۲. کتاب Ivor Horton’s 2008 Visual C++ Beginning نوشته
۳. سایت www.barnamenevis.org

این روزها زیاد هستند افرادی که می خواهند با یادگیری زبان C++ اقدام به نوشتن ویروس نمایند تا مثلا سطح مهارت خود را به رخ دیگران بکشند و حس غرور درون خود را به اینصورت ارضا نمایند ، در حالی که تفاوتی بین خط انداختن روی یک ماشین با تخریب اطلاعات کامپیوتر یک شخص ، یا سرقت از کیف پول یا موبایل یک شخص با سرقت اطلاعات شخصی از کامپیوتر فرد وجود ندارد و نام همه آنها مردم آزاری یا دزدی است که باعث مدیونی و عواقب مادی و معنوی شما خواهد شد.

شما می توانید سطح مهارت خود را با نوشتن برنامه های مفید برای دیگران نیز افزایش دهید.دانشمندان و کسانی که از علمی استفاده میکنند باید به تعهدات اخلاقی و وجدانی پایبند باشند. متاسفانه به دلیل بی توجهی به این قضیه دراین زمان مثلا پزشکانی میبینیم که در اتاق عمل کلیه بیمار را میدزدند یا بیمار را بدون نیاز به جراحی و فقط برای دریافت پول عمل می نمایند ، حالا نظر شما درباره این افرادچیست؟ این هم نوع دیگری سوء استفاده از علم است.

دوستان در صورت تمایل به مکاتبه لطفا از طریق بخش نظرات وبلاگ من، نظرات و پیشنهادات خود را بیان نمایید.

در پایان از خوانندگان به دلیل وجود اشکالات احتمالی عذر خواهی می نمایم.

با تشکر

بهزاد جناب

مرداد ۱۳۸۸

چرا از C++ استفاده می کنیم؟

زبان برنامه نویسی C اوایل دهه هفتاد اختراع شد و به سرعت به یکی از محبوبترین زبانهای برنامه نویسی حرفه ای گردید. سی پلاس پلاس زبان مورد انتخاب برای ساخت نرم افزارهایی با کارآیی بالا است که به طور مستقیم به منابع یا تجهیزات و ابزار های ویندوز دسترسی دارند. پس شما با آن می توانید به قابلیت های سطح پایین سیستم دسترسی داشته باشید و البته به خاطر اینکه این زبان به شما قدرت بسیار بالایی می دهد ، لذا در مقایسه با سایر زبان ها مثل visual Basic یا #C شما بایستی از جزییات بیشتری اطلاع داشته باشید. بنابراین از جمله کاربرد های بدون شک این زبان برنامه هایی می باشد که نیاز به دقت بالا ، تاخیر زمانی کوتاه (Low-latency) و استفاده مستقیم از سخت افزار دارند ، نرم افزار هایی همچون:

نرم افزارهای گرافیگی و طراحی ۲ بعدی و ۳ بعدی ، محیط ها و موتورهای توسعه بازی های کامپیوتری ، نرم افزارهای صوتی / تصویری ، پیشرفته سیستمی و غیره ... که چیزی فراتر از طراحی واسط کاربر باشد .

در ذیل چند دلیل موفقیت C++ را آورده ام:

- **C++ فایلهای متکی به خود می سازد. همین که برنامه تان را کامپایل و لینک کردید دیگر می توانید فایل exe را بدون هیچ دغدغه ای به دیگران بدهید.**
- **سرعت اجرای فایل های اجرایی C++ بسیار خوب است.**
- **فایل های exe تولید شده توسط C++ کوچک هستند.**
- **سرعت کامپایل و لینک شدن برنامه های C++ بسیار زیاد است.**
- **C++ زبانی مطمئن، ساده و قدرتمند است.**

به طور کلی دو نوع شیوه های برنامه نویسی برای سی پلاس پلاس وجود دارد.

بومی (Native) و مدیریت شده (Managed)

۱- در نوع native که قدرتمند ترین نوع برنامه نویسی می باشد (مدیریت نشده) ، برنامه شما به طور مستقیم توسط پردازنده مرکزی (CPU) اجرا می شود و می تواند بر روی نسخه های مختلف سیستم عامل ویندوز اجرا شود این مورد شامل ویندوز CE و ویندوز mobile برای تلفن های همراه نیز می شود .لذا برنامه های native دسترسی مستقیم به سیستم عامل و سخت افزار دارند و این به شما قدرت و کارآیی (Performance) بسیار بالایی می دهد. اما نکته ای که باید توجه کنید این است که قدرت بالا ، به دقت ، مسئولیت پذیری و تمرین بیشتری نیاز دارد تا موارد به درستی انجام شود .

شیوه Native نیز به دو نوعMFC و win32 که هر دو مختص پلتفرم ویندوز هستند تقسیم میشود.

در **Win32** که قلب ویندوز شناخته می شود و شیوه برنامه نویسی سطح پایین می باشد (پشتیبانی از ۱۶ بیت تا ۶۴ بیت) ، سرعت توسعه نسبت به سایر زبان ها مانند **VB** و **C#** کمتر است و زمان و انرژی بیشتری صرف خواهد شد، اما در عوض همه چیز در اختیار شما قرار دارد ، از کنترل دقیق حافظه تا کنترل تمامی منابع سیستم و البته با نهایت کارآیی.

اکثر نرم افزار های تجاری شرکت های بزرگ و متوسط سراسر دنیا که در منازل از آن ها استفاده می کنید (و نیازی به نام بردن آن ها نیست)و تقریبا هسته اصلی تمامی بازی ها در نسخه ویندوز آن ها از این **API** ها به طور مستقیم استفاده می کنند.

MFC یا همان **Microsoft Foundation Class** ، یک**framework** می باشد که **API** های **win32** را در قالب کلاس هایی برای برنامه نویسان **C++** ارائه می کند ، تا زمان توسعه را کاهش دهد ، کار با پایگاه داده را آسان تر می کند و با وجودی که تقریبا تمامی قابلیت های سایر زبان ها را در اختیار شما قرار می دهد ، و جدا از اینکه کارآیی در مقایسه با شیوه قبل کمی کاهش می یابد ، تمامی ناحیه های **win32**را نیز در بر نخواهد گرفت و لذا نیاز به آشنایی با خود **API** های ویندوز نیز می باشد .

ضمنا **MFC** در کشورمان کاربران بیشتری دارد.

از جمله محصولات شرکت **Nero** و همین طور ابزار های همراه آن که در سال های اخیر عرضه شده اند ، همچون کپی **CD/DVD** ، پخش فیلم ، ویرایش موسیقی و غیره ... از **MFC** استفاده می کنند.

- مزیت – کارآیی بالا
- مزیت – کمترین میزان نیاز به منابع سخت افزاری مانند حافظه **Ram** و فضای دیسک و ...
- نقص – پیچیدگی بیشتر و دارای زمانبری بیشتر برای کارکردن و نوشتن با آن
- نقص – وابسته به پلتفرم ویندوز

۲- در نوع **managed** که یک محیط **runtime** به نام **CLR** برای شما فراهم می کند ، شما را از این پیچیدگی کار و قرار دادن تحت سیستم عامل و سخت افزار جدا می کند و برنامه نویسی را بسیار سریع تر و آسان تر می کند. اما در هر حال این جدایی ، انعطاف پذیری **(flexibility)** و به احتمال غریب به یقین کارآیی **(Performance)** کمتری دارد ، که البته این موارد بستگی به پروژه مورد نظر دارد که آیا کارآیی ، مورد اهمیت می باشد یا خیر .ضمنا این مورد نیاز به نصب **Net.** در سیستم مورد نظر دارد.

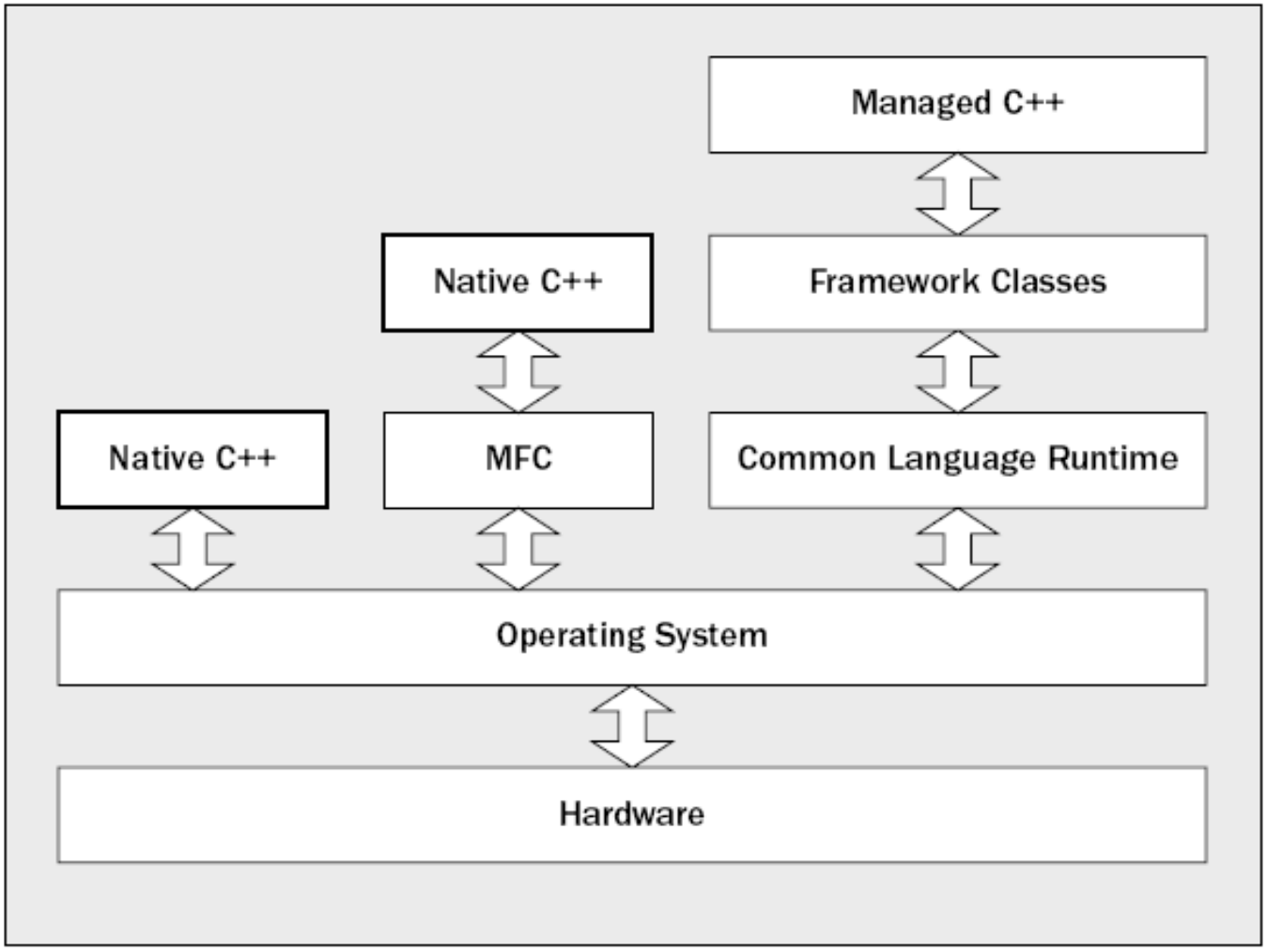
- مزیت – قابلیت حمل)قابل استفاده در هر سیستم دارای **CLR**)
- مزیت – سهولت در این نوع برنامه نویسی
- نقص – احتمال اجرای کند تر
- نقص – امکان نیاز بیشتر به منابع سیستمی ، حافظه و فضای دیسک و ...

طبق گفته های مکرر تیم طراحی **Visual C++** ، طبق سیاست جدید مایکروسافت ، تصمیم بر آن گرفته شده که برای راضی نگه داشتن مشتریان ، توسعه بخش **Native** توسط **C/C++** ، به سرعت و همراه با پیشرفت تکنولوژی انجام شود (و با ارائه بیش از ۷۰۰۰ **API** جدید برای ویندوز **vista** تمامی شک های احتمالی بر طرف شد و امید پشتیبانی بخش **native** برای سال های آینده به واقعیت قطعی تبدیل شد و

دیگر اجباری در کوچ کردن به دات نت نمی باشد)، که این کار باعث ایجاد تاخیر در توسعه بخش دات نت برای **C++/CLI** می شود و بیان شده است که:

کاربران انتظار نداشته باشند کاری که قبلا توسط سایر زبان ها انجام پذیرفته است حتما و فورا برای **C++/CLI** نیز انجام شود و به کسانی که می خواهند که با این روش **managed** برنامه نویسی کنند توسیه می شود از **C#** (سی شارپ) استفاده کنند.

همانطور که در عکس زیر ترتیب اجرای یک فرمان در روشهای مختلف برنامه نویسی در **C++** را مشاهده میکنید، روش **Native** بسیار به سخت افزار نزدیکتر بوده تا روش **Managed** و بنابراین سرعت آن نیز بالاتر است.



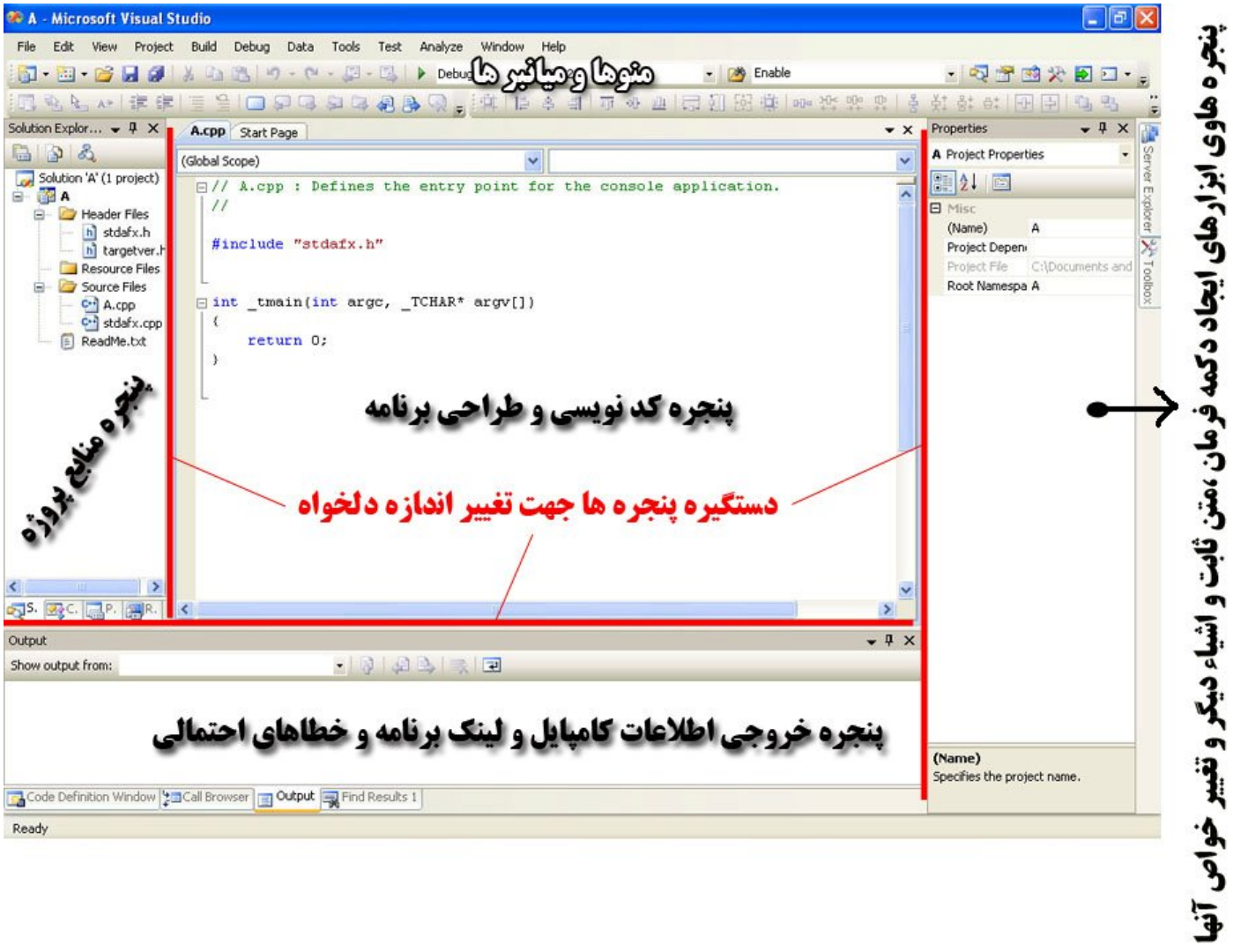
آموزش مقدماتی ویژوال سی پلاس پلاس ۲۰۰۸

بهزاد جناب

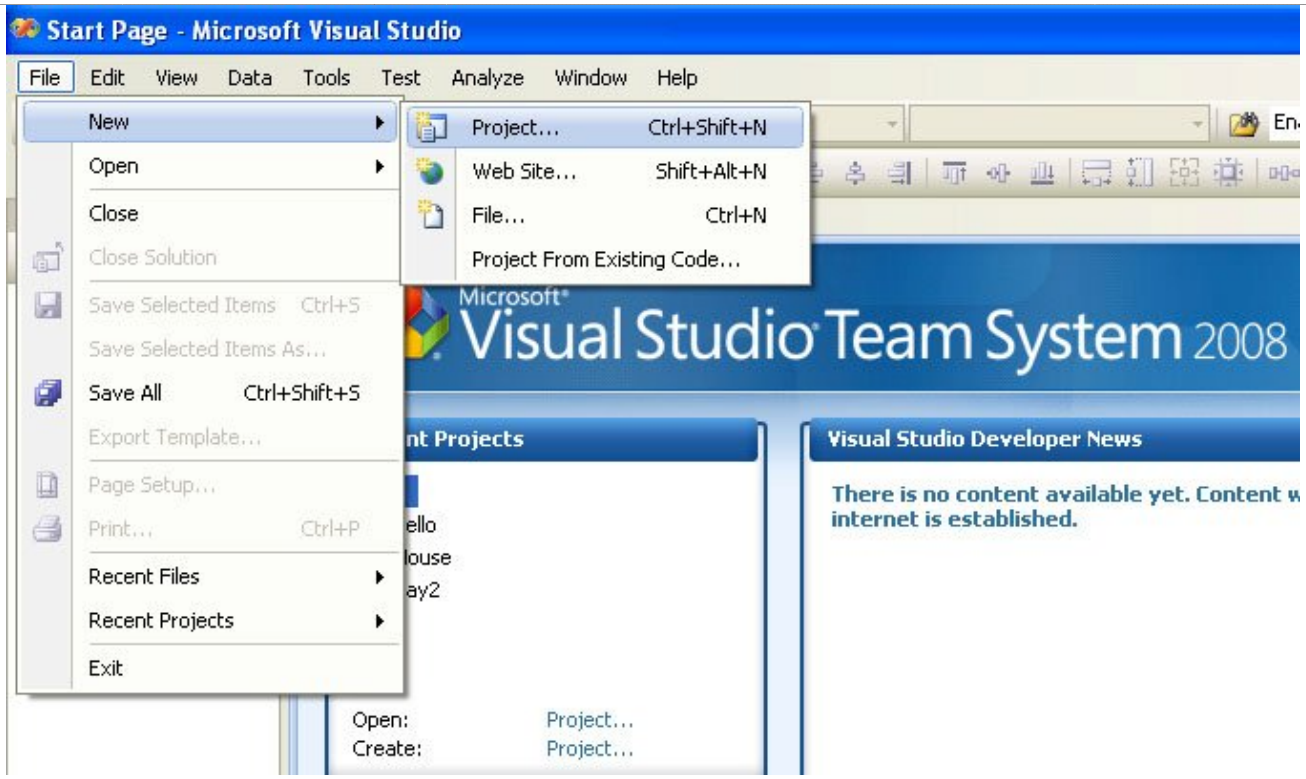
فصل اول

آشنایی با کلیات و مفاهیم زبان سی پلاس پلاس

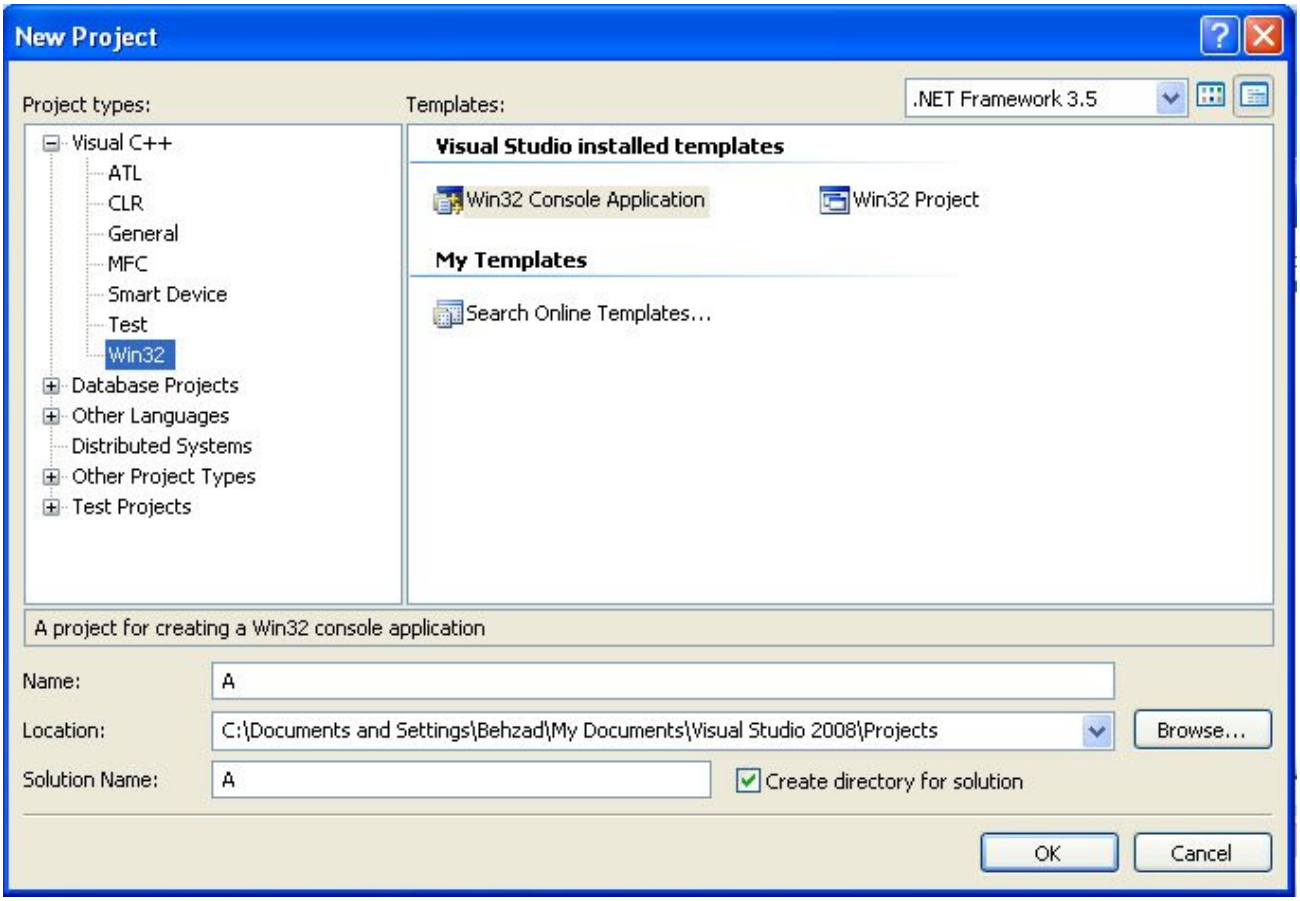
قبل از اینکه شروع به برنامه نویسی و دادن توضیحاتی مربوط به کد ها نماییم ابتدا با محیط Visual Studio 2008 بوسیله عکس زیر کمی آشنا میشویم.در قسمت بالایی منوها و میانبرها و در پائین چهار پنجره به هم چسپیده ، محیط ویژوال استدیو را تشگیل میدهند که بوسیله دستگیره مشخص شده در شکل قابلیت تغییر اندازه به صورت دلخواه ما را دارا هستند.هر کدام از این پنجره ها نیز دارای قسمتهای(Tab) مختلفی هستند.



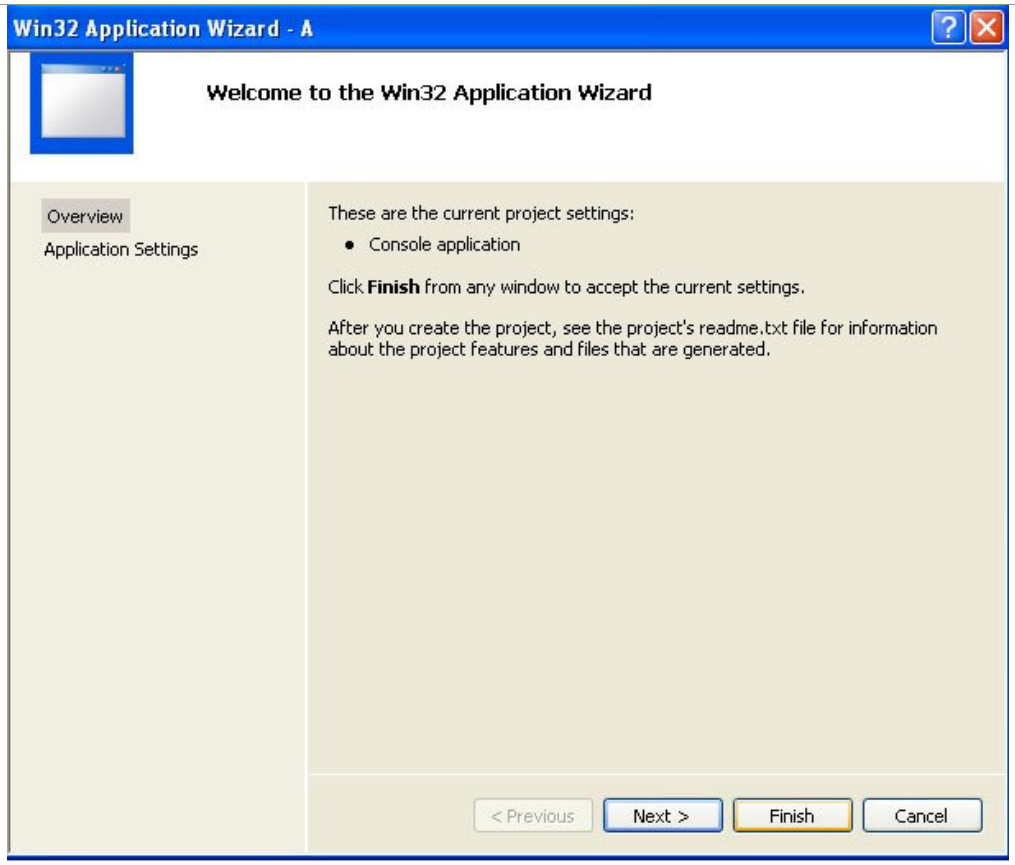
برای آشنایی با اصول اولیه برنامه نویسی با سی پلاس پلاس یک برنامه ساده DOS می نویسیم. ابتدا از منویFile گزینه New و سپس Project را انتخاب کنید.



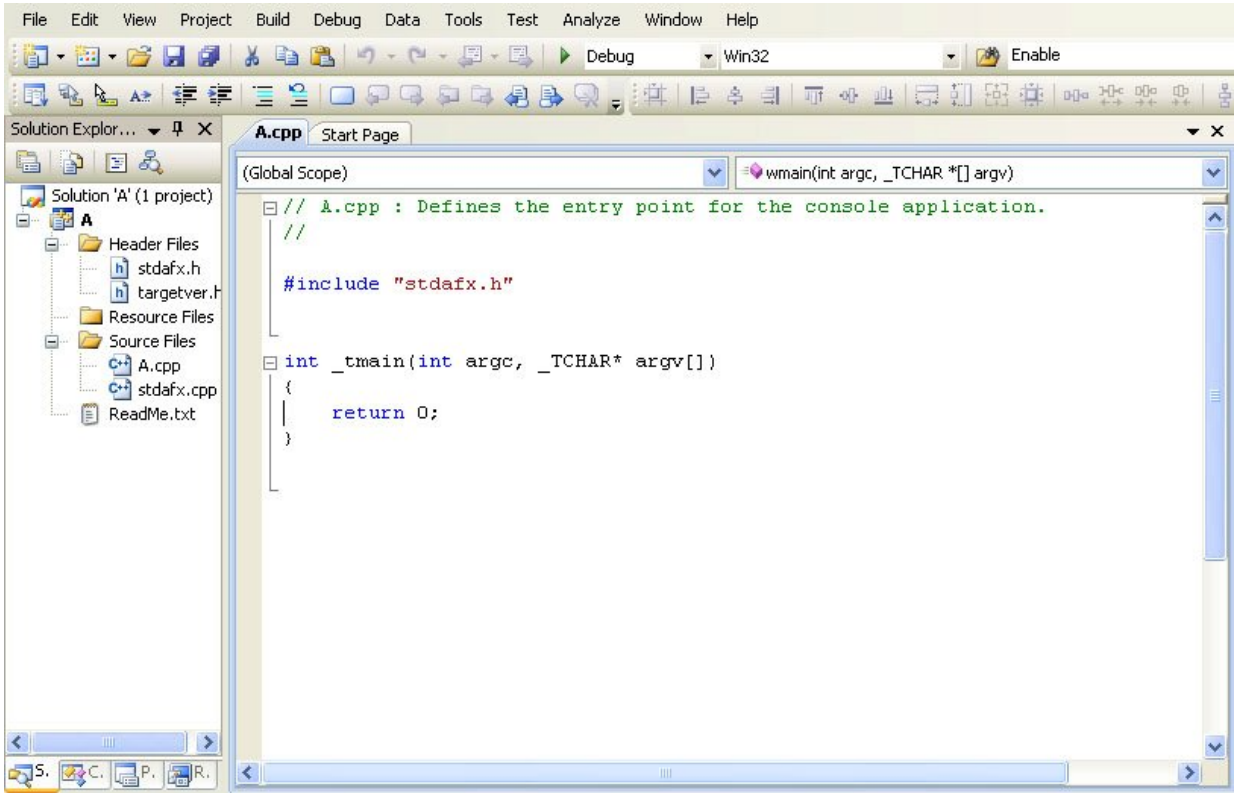
سپس **Win32 Console Application** را انتخاب کرده و نام آن را **A** بگذارید و بر روی **OK** کلیک کنید.



سپس در پنجره دوم به نمایش درآمده بر روی **Finish** کلیک کرده تا پروژه ساخته شود.



در این مرحله شما متن درون فایل A.cpp را مشاهده میکنید.(پسوند cpp به معنای CPlusPlus یا همان C++ است که در اینجا مشاهده میکنید)



اگر تا به حال برنامه ای به زبان C++ ندیده باشید این کدها برایتان بسیار نامفهوم خواهد بود اما نگران نباشید با تمای آنها آشنا خواهید شد.

در دو خط اول از علامت // استفاده شده است که این علامتها به معنای شروع توضیحاتی هستند که برای خوانا شدن برنامه نوشته میشود و هر عبارتی که جلوی آنها نوشته شود هنگام تبدیل پروژه به یک فایل اجرائی توسط کامپایلر نادیده گرفته میشوند.

حال سئوال این است که فایده استفاده این توضیحات در چیست؟ وقتی یک برنامه نوشته میشود دارای بخشهای زیادی است که حتی خود نویسنده هم بعد از مدتی برای اصلاح و تغییر و برسسی برنامه بدون این توضیحات گیج می شود و کلا برای پیدا کردن خطا ها و بخشهای مختلف و ویرایش مجدد یک برنامه بسیار مفید هستند.

حال قصد داریم توضیحاتی به برنامه اضافه کنیم، خط زیر را به برنامه اضافه نمایید.

```
// اولین پروژه من با ویژوال سی پلاس پلاس
```

در روشی دیگر برای نوشتن توضیحات در C++ توضیحات با /* شروع و با */ پایان می یابد .

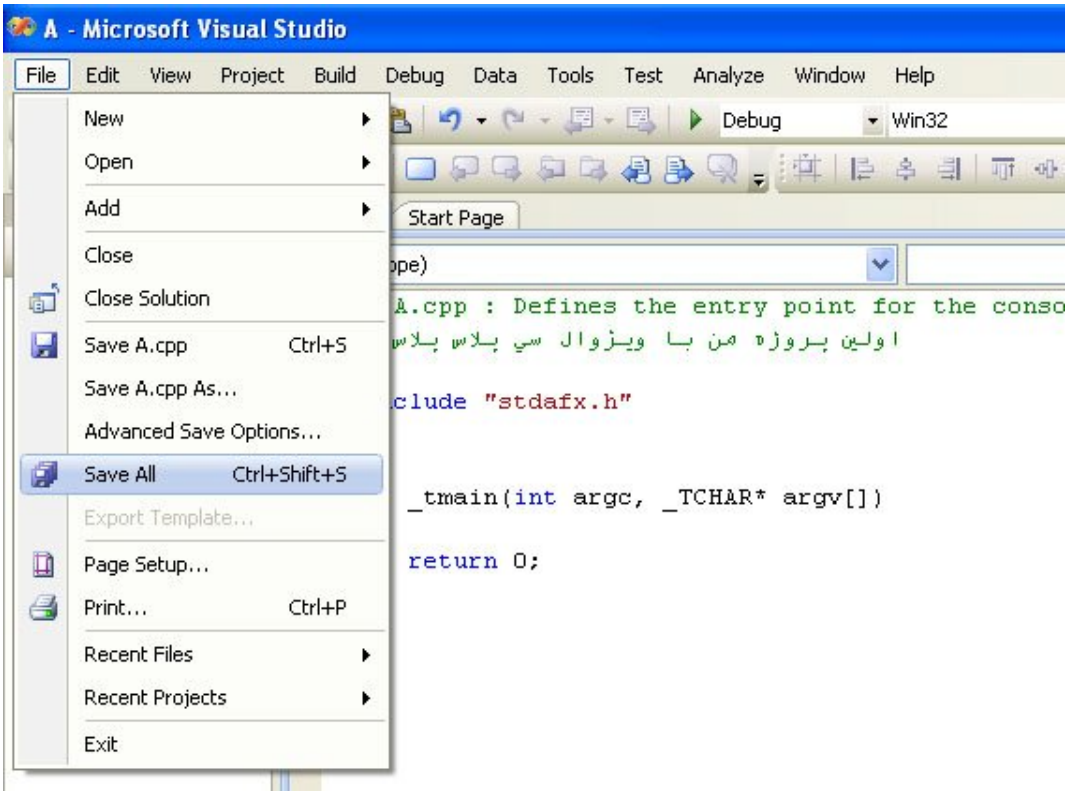
استفاده از دو روش زیر نیر صحیح است.

```
/* اولین پروژه من با ویژوال سی پلاس پلاس */
```

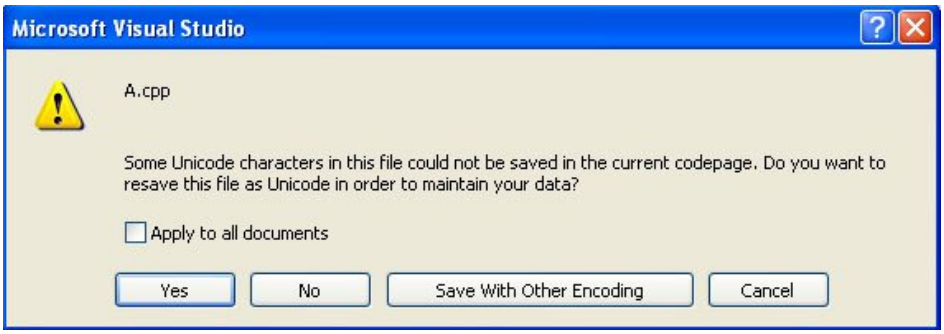
```
/*****************************
*                 بهزاد جناب                *
*    اولین پروژه من با ویژوال سی پلاس پلاس    *
*****************************/
```

حالا می خواهیم برنامه را قبل از اجرا ذخیره نمایید. برای ذخیر کل پروژه می توانید به دو روش زیر اقدام نمایید.

۱. از منوی File گزینه Save ALL را انتخاب.
۲. کلید های Ctrl+Shift+S را فشار دهید.

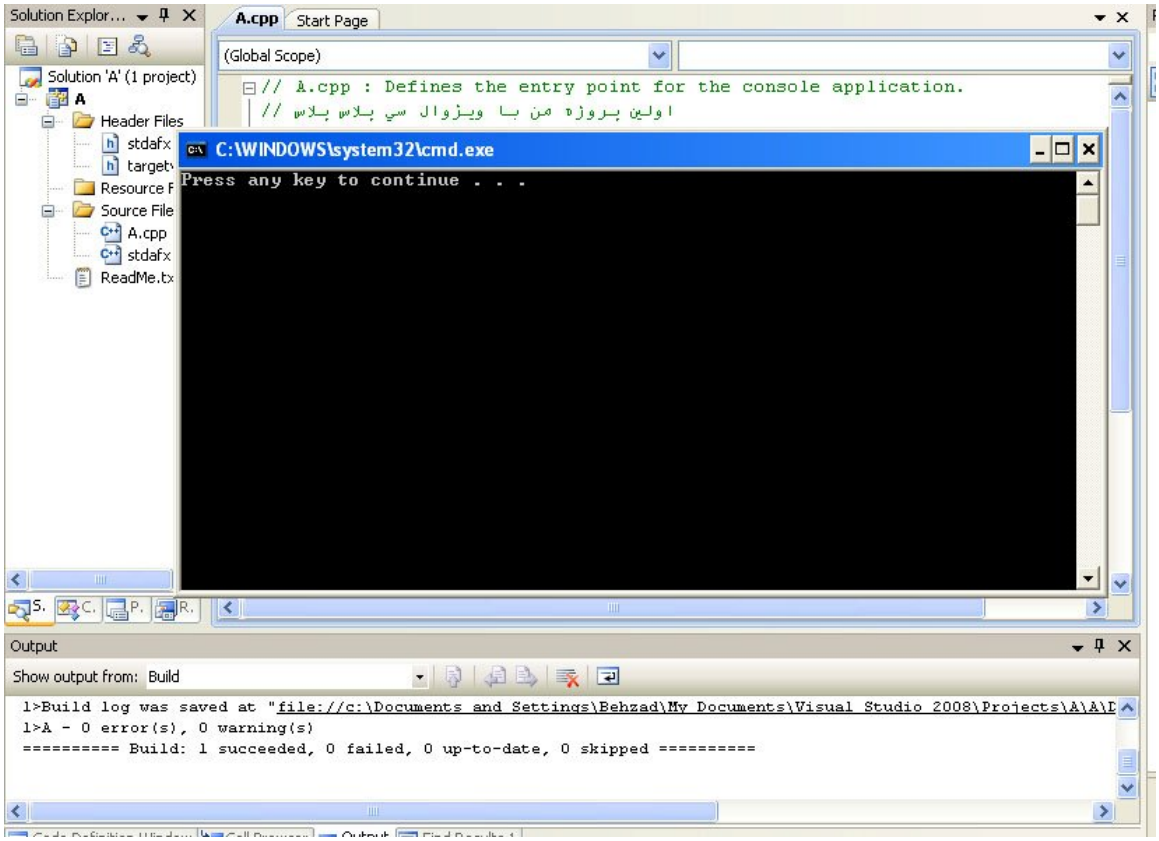


بعد از اقدام ما برای ذخیره پروژه به خاطر استفاده از زبان فارسی که کارکترهای آن به صورت یونی کد (Unicode) است هشداری از ویژوال استدیو دریافت میکنیم که پروژه ما باید به صورت یونی کد ذخیره شود دریافت میکنید که شما باید گزینه Yes را انتخاب کنید تا نوشته های فارسی شما در به درستی ذخیره شده و در مراجعات بعدی به صورت علامتهای ؟ نشان داده نشوند.



حالا برای تبدیل این برنامه به یک فایل اجرایی (exe) و مشاهده خروجی برنامه از کلید Ctrl+F5 استفاده می کنیم.در پنجره خروجی در قسمت پایین شما شاهد اطلاعات لینک و کامپایل این برنامه هستید که به معنای وجود هیچ خطا وهشداری به هنگام کامپایل برنامه شماست و همچنین موفقیت در تبدیل این پروژه به فایل اجرائی(exe).

از آنجایی که هنوز در این برنامه هیچ کدی ننوشته ایم هیچ کاری هم توسط این برنامه انجام نمیشود و در پنجره بوجود آمده توسط برنامه هیچ نوشته ای مشاهده نمیکنیم به غیر از عبارت "Press any key to continue . . ." که توسط خود کامپایلر نوشته شده است نه بوسیله برنامه شما ، آن هم به خاطر اینکه برنامه نویس قبل از بسته شدن پنجره شاهد خروجی برنامه خود باشد،چون در حالت عادی به محز اتمام برنامه این پنجره نیز بسته میشود.



در خطهای بعدی بدنه اصلی برنامه که تابع **main** است را مشاهده می کنید.ولی این تابع به صورت **_tmain** نوشته شده که هیچ تفوتی ندارد و فقط برای حل مشگل استفاده از حروف یونی کد(**Unicode**) مانند زبان فارسی است.

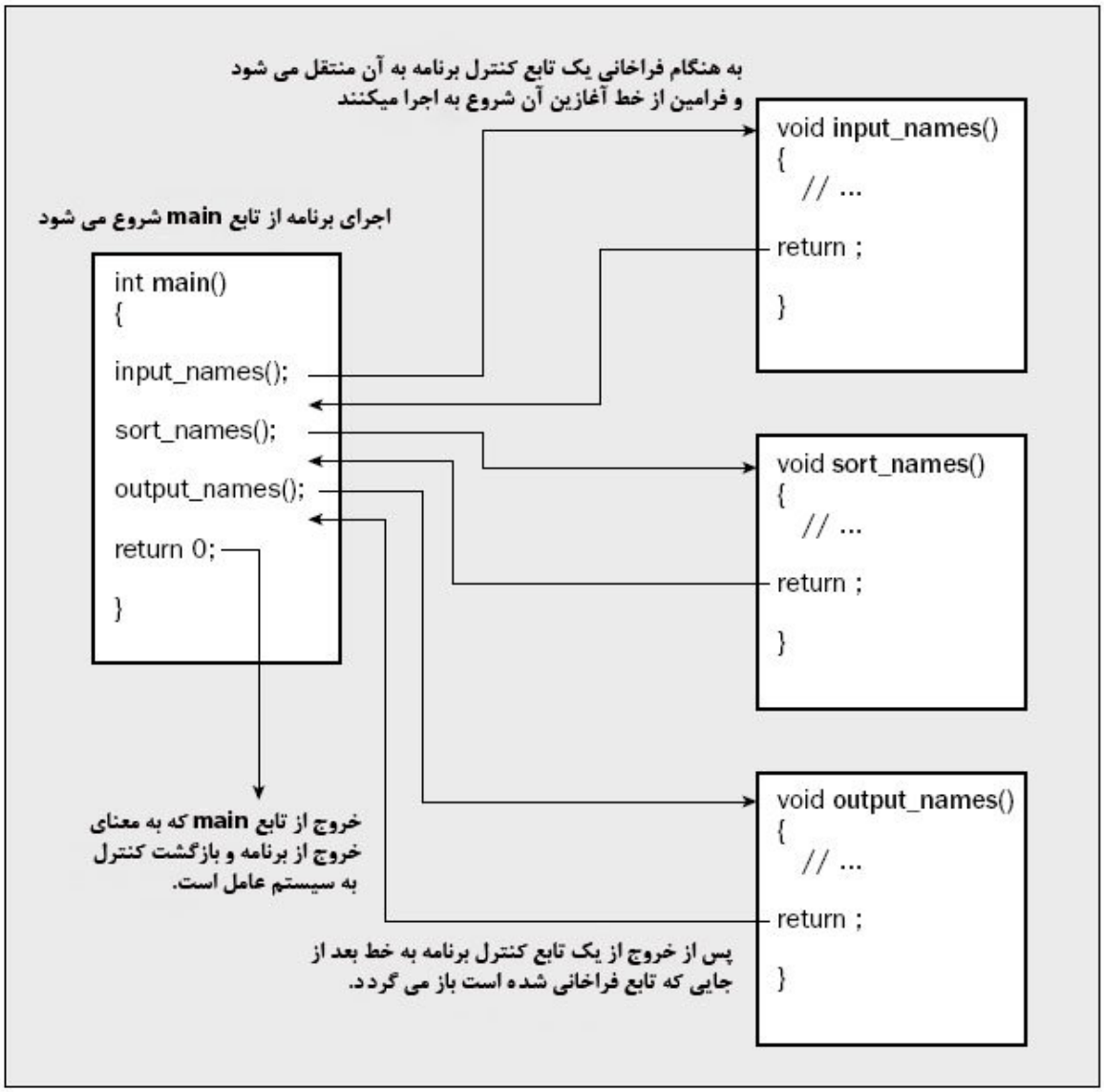
```
int _tmain(int argc, _TCHAR* argv[])
{
      return 0;
}
```

تابع یک بلوک کد است که بین دو{} قرار می گیرد.نمونه ذیل هم یک تابع است:

```
void MyFunction();
{
   کد تابع در این قسمت قرار می گیرد //
}
```

در برنامه های **C++** حتما باید یک تابع **()main** وجود داشته باشد. تابع **()main** اولین تابعی است که در یک برنامه **C++** اجرا می شود.
با اجرای این برنامه دستورات خط به خط بعد از علامت **{** تابع **main** شروع به اجرا و با رسیدن به عبارت **return 0;** خاتمه پیدا میکند و از برنامه خارج میشود.

شکل زیر نحوه اجرای توابع و در برنامه های **C++** را نشان می دهد.



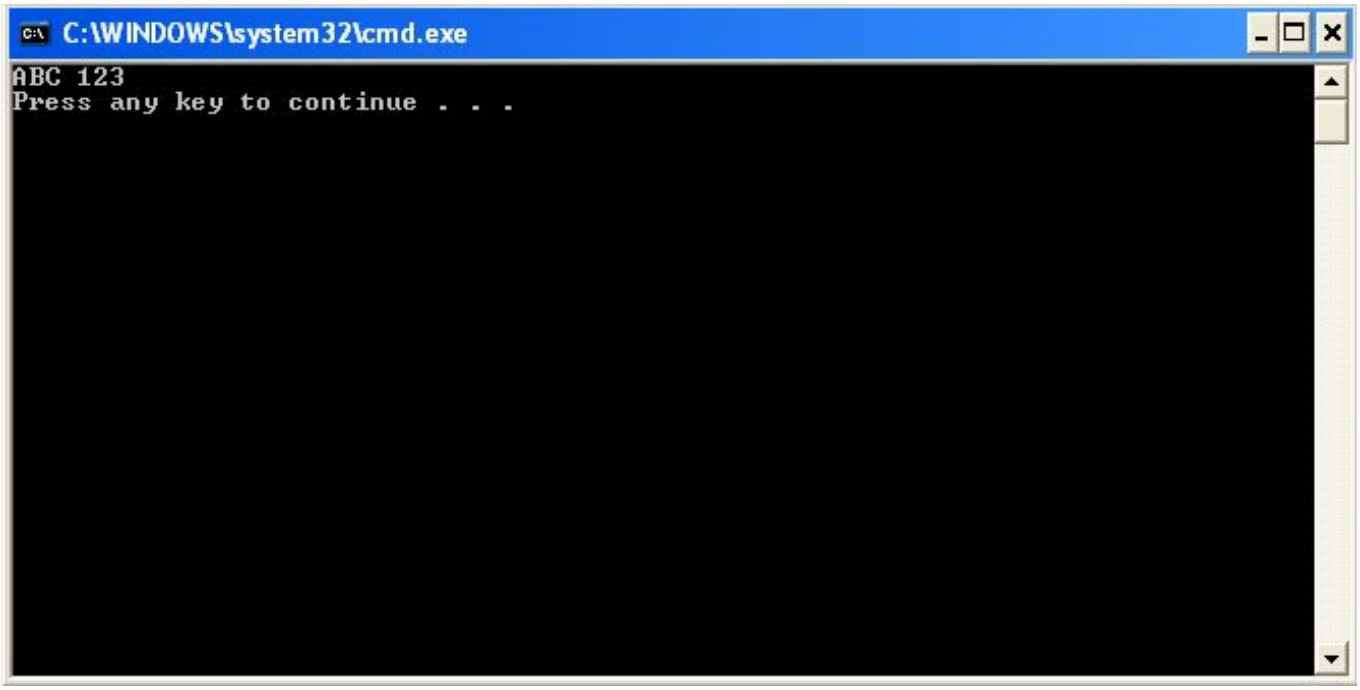
حالا دو خط جدید به این برنامه اضافه می کنیم و به شکل زیر تغییر میکند

```
// A.cpp : Defines the entry point for the console application.

#include "stdafx.h"
#include <iostream>

int _tmain(int argc, _TCHAR* argv[])
{
	std::cout << "ABC 123 \n";
	return 0;
}
```

دستور #include <iostream> باعث میشود که محتویات فایلی به نام iostream.h به برنامه ما اضافه شود چون برای چاپ نوشته مان نیاز به تابع std::cout داریم که در درون فایل ذکر شده قرار دارد، دستور دوم هم که عبارت std::cout << "ABC 123 \n"; است که باعث چاپ عبارت داخل گیومه در پنجره خروجی برنامه می شود، دستور cout به کامپیوتر می گوید که هر چه بعد از << وجود دارد نمایش دهد.کاراکتر \n که چاپ نشده است در واقع به معنای برگشت به سرخط است و کار فشرده شدن کلید اینتر(Enter) را انجام می دهد.

نکته مهم این است که در پایان هر خط کد نوشته شده در C++ باید یک علامت ; قرار گیرد.

برنامه را با Ctrl+F5 کامپایل و اجرا نمایید، خروجی به شکل زیر است.

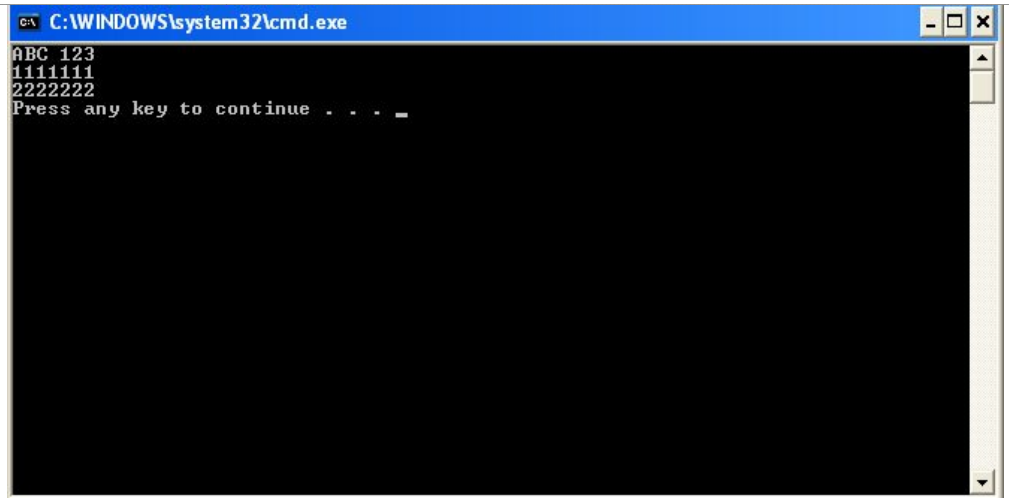


حالا دوخط دیگر نیز به برنامه اضافه میکنیم.

```
	std::cout << "1111111" << std::endl;
	std::cout << "2222222" << std::endl;
```

تنها قسمت جدید عبارت std::endl است که در واقع همان کار عبارت \n داخل گیومه("") که رفتن به خط جدید است را انجام میدهد.

دوباره برنامه را با Ctrl+F5 کامپایل و اجرا نمایید، خروجی به شکل زیر خواهد بود.



جمع کردن در برنامه A.CPP:

در این قسمت کاری می کنیم تا برنامه بتواند دو عدد را با هم جمع کند.

۱-محتوی فایل را به صورت ذیل تغییر دهید:

```
#include "stdafx.h"
#include <iostream>

int _tmain(int argc, _TCHAR* argv[])
{
	int a;
	int b;
	int c;

	a = 2;
	b = 3;
	c = a + b;

	std::cout << "a=" << a << std::endl;
	std::cout << "b=" << b << std::endl;
	std::cout << "a+b=" << c << std::endl;

	return 0;
}
```

می بینید که تمام کدهای قبلی را حذف کرده ایم، در اولین دستور یک متغییر بنام a تعریف شده است. در C++ قبل از استفاده از هر متغیری باید آن را تعریف(declare) کرد.متغییرها می توانند داده ها را در خود نگهدارند.هنگام تعریف یک متغییر باید نام و نوع آن را مشخص کنید، به هر حال کامپایلرها باید بدانند که چه نوع داده ای قرار است در این متغییر قرار گیرد.کلمه int به کامپایلر می گوید که این متغییر داده ای از نوع عدد صحیح را خواهد گرفت.دو متغییر دیگر هم به نامهای b و c تعریف کرده ایم سپس به متغییر های a و b مقدار داده شده است.

در سی پلاس پلاس انواع مختلفی از متغییر وجود دارد که هر کدام برای ذخیره نوع خاصی از داده ساخته شده است. در زیر لیستی از انواع آنها مشاهده می کنید.در نامگزاری متغییر در C++ فقط می توان از حروف انگلیسی ، عدد و علامت _ استفاده کرد.ولی نباید اولین حرف تشگیل دهنده نام متغییر عدد باشد مثلا متغییر 1A خطا است ولی A12h صحیح می باشد، و یک نکته بسیار مهم اینکه حروف کوچک و بزرگ در C++ متفاوت هستند یعنی دو متغییر به نامهای A و a با هم متفاوت هستند و یا دو متغییر با نامهای Behzad و behzad نیز باهم متفاوتند.

تعریف متغییر از فرمول زیر انجام پذیر است:

نوع متغییر نام متغییر;

مثلا برای تعریف متغییری به اسم a برای ذخیره عدد صحیح از دستور int a; و برای ذخیره اعداد اعشاری در متغییری به نام b از دستور double b; استفاده می کنیم.از روی جدول زیر می توانید انواع داده در C++ مشاهده نمایید.

نوع داده	کاربرد	بایت مصرفی	محدوده
bool	ذخیره درست و غلط	۱	مقدار صفر یا یک true or false
char	ذخیره یک کارکتر	۱	–128 to +127
signed char	ذخیره یک کارکتر	۱	–128 to +127
unsigned char	ذخیره یک کارکتر	۱	0 to 255
wchar_t	ذخیره یک کارکتر یونیکد	۲	0 to 65,535
short	ذخیره عدد صحیح کوچک	۲	–32,768 to +32,767
unsigned short	ذخیره عدد مثبت صحیح کوچک	۲	0 to 65,535
int	ذخیره عدد صحیح متوسط	۴	–2,147,483,648 to 2,147,483,647
unsigned int	ذخیره عدد مثبت صحیح متوسط	۴	0 to 4,294,967,295
long	ذخیره عدد صحیح بلند	۴	–2,147,483,648 to 2,147,483,647
unsigned long	ذخیره عدد مثبت صحیح بلند	۴	0 to 4,294,967,295
float	ذخیره اعداد اعشاری	۴	با ۷ رقم دقت در اعشار ±3.4·10±38
double	ذخیره اعداد اعشاری بلند	۸	با ۱۵ رقم دقت در اعشار ±1.7·10±308
long double	ذخیره اعداد اعشاری بلند	۸	با ۱۵ رقم دقت در اعشار ±1.7·10±308

اما چه نوع داده هایی را می توان در این نوع داده ها ذخیره نمود، در جدول زیر مثالهایی برای آشنایی شما آورده ام.

نوع داده	مثال هایی از انواع ورودی های قابل قبول برای ذخیره در این نوع داده
char, signed char و unsigned char	‘A’, ‘Z’, ‘8’, ‘*’
wchar_t	L’A’, L’Z’, L’8’, L’*’
int	-77, 65, 12345, 0x9FE
unsigned int	10U, 64000u
long	-77L, 65L, 12345l
unsigned long	5UL, 999999999UL, 25ul, 35Ul
float	3.14f, 34.506F
double	1.414, 2.71828
long double	1.414L, 2.71828l
bool	true, false

می توان هنگام تعریف متغییر به دو روش زیر مقدار اولیه ای نیز به آنها اختصاص داد.

```
int a = 0;
int b = 10;
int abc = 5;
```

یا

```
int a(0);
int b(10);
int abc(5);
```

نکته: در C++ فاصله های بین دستورات به کلی نادیده گرفته می شود ، به عبارسی دیگر هر چهار دستور ذیل معادل یکدیگر و صحیح هستند.

```
a                    = 2;
a =                  2;
a  =
                   2;
```

نکته: در C++ پایان یک خط را علامت ; مشخص می کند یعنی شما در یک خط مانند زیر می توانید از چندین دستور استفاده کنید و این دستورات هرکدام یک خط جداگانه به حساب می آیند.

```
int a,b,c;   a=20;    b=30; c=a+b;
```

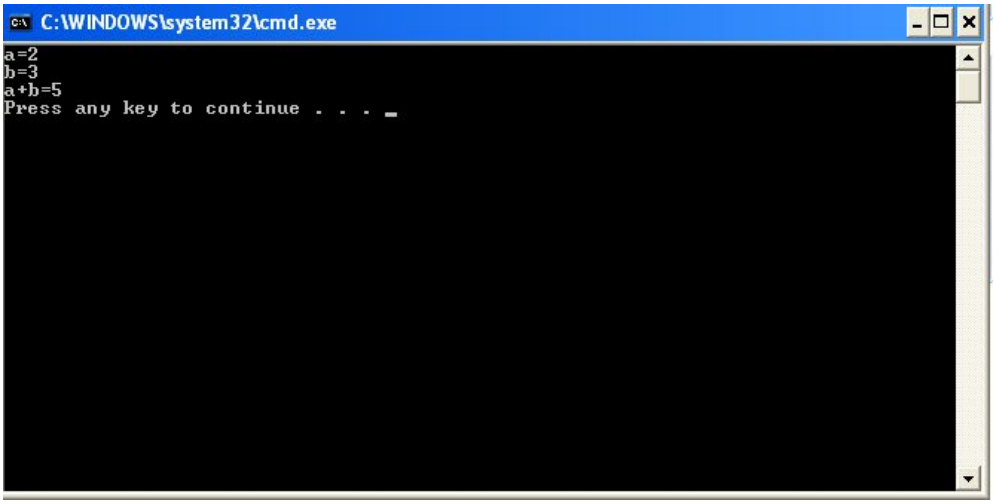
البته یک استثناء وجود دارد و آن عبارت بین گیومه(" ") است، دستور ذیل خطا است:

```
std::cout << " I am
                  Behzad";
```

جمع دو عدد با دستور c=a+b انجام می شود.سپس حاصل جمع با دستورات:

```
std::cout << "a=" << a << std::endl;
std::cout << "b=" << b << std::endl;
std::cout << "a+b=" << c << std::endl;
```

نمایش داده می شود. چون عبارات a،b و c درون " " قرار ندارند مقدار آن نمایش داده می شود نه کلمه a،b و یا c.
پس از اجرای برنامه نتیجه کار به شکل زیر خواهد بود.



عملگرهای ریاضی در زبان C++ به صورت زیر تعریف میشوند

علامت عملگر در C++	نام عملگر
+	جمع
-	تفریق
*	ضرب
/	تقسیم
%	باقیمانده تقسیم
++	افزایش یک واحد به متغییر مورد نظر
--	کاهش یک واحد از متغییر مورد نظر

در مورد دو عملگر آخر یعنی ++ و -- روش استفاده به شکل زیر است و باعث کاهش و یا افزایش یک واحد از متغییر می شود. مثلا دستور a++ معادل با دستور a=a+1 است و دستور —b معادل دستور b=b-1 است، و به شکل زیر مورد استفاده قرار می گیرد.

```
a++;
b--;
```

توابع در C++

برنامه ما فعلا یک تابع به نام main() دارد حالا می خواهیم یک تابع دیگر هم اضافه کنیم ، این تابع را Add() نامیده ایم.

۱-برنامه را چنین تغییر دهید:

```
#include "stdafx.h"
#include <iostream>

int _tmain(int argc, _TCHAR* argv[])
{
	int a,b,c;

	a = 2;
	b = 3;
	c = Add(a,b);

	std::cout << "a=" << a << std::endl;
	std::cout << "b=" << b << std::endl;
	std::cout << "a+b=" << c << std::endl;

	return 0;
}
```

نکته: در خط اول تابع(int a,b,c;) اینبار یکجا سه متغییر از نوع int (عدد صحیح) تعریف کرده ایم که هیچ تفاوتی با روش قبلی ندارد، فقط جهت آشنایی شما با این روش که به کد نویسی کمتری نیاز دارد نوشته شده است.

در اینجا فقط به جای دستور c=a+b دستور c=Add(a,b) را نوشته ایم تابع Add() را بعدا می نویسیم، فقط توجه کنید که این تابع دو پارامتر میگیرد آنها را با هم جمع کرده و سپس حاصل جمع را به متغییری که با آن برابر نهاده شده می دهد.

اگر برنامه را در این مرحله کامپایل کنید با خطا مواجه خواهیم شد چرا؟ چون C++ هیچ چیز درباره تابع Add() نمی دادند و این وظیفه شماست که این را به او بفهمانید.

تعریف پیش اعلام تابع Add()

وقتی کامپایلر به دستوری که Add() در آن است می رسد باید بداند که این تابع است که دو پارامتر ورودی می گیرد(پارامتر هایی از نوع int) و یک عدد صحیح بر می گرداند. این کار با تعریف پیش اعلام(Prototype) انجام خواهد شد:

```
int Add(int , int);
```

حالا تعریف پیش اعلام تابع Add() را به برنامه اضافه کنید:

```
#include "stdafx.h"
#include <iostream>

int Add(int , int);

int _tmain(int argc, _TCHAR* argv[])
{

	int a;
	int b;
	int c;

	a = 2;
	b = 3;
	c = Add(a,b);

	std::cout << "a=" << a << std::endl;
	std::cout << "b=" << b << std::endl;
	std::cout << "a+b=" << c << std::endl;

	return 0;
}
```

حالا دیگر کامپایلر درباره تابع Add() کاملا چشم و گوش بسته نیست. در قسمت آینده خود این تابع را هم تعریف خواهیم کرد.(تنها تابعی که به پیش اعلام نیاز ندارد همان تابع main() است، اما سایر توابع حتما باید پیش اعلام داشته باشند.)

نوشتن کد تابع Add()

کد ذیل که مربوط به تابع Add() است را به برنامه اضافه کنید:

```
int Add(int num1,int num2)
{
	int addnum;
	addnum = num1 + num2;
	return addnum;
}
```

این کد بسیار شبیه تعریف پیش اعلام آن است با این تفاوت که در انتهای خط تعریف تابع ; ندارد. در پیش اعلام نام پارامترها قید نشده و فقط نوع آنها مشخص شده است، اما در این جا نام ها هم قید شده اند.
و در نهایت کد برنامه به صورت ذیل خواهد شد:

```
#include "stdafx.h"
#include <iostream>

int Add(int , int);

int _tmain(int argc, _TCHAR* argv[])
{

	int a;
	int b;
	int c;

	a = 2;
	b = 3;
	c = Add(a,b);

	std::cout << "a=" << a << std::endl;
	std::cout << "b=" << b << std::endl;
	std::cout << "a+b=" << c << std::endl;

	return 0;
}

int Add(int num1,int num2)
{
	int addnum;
	addnum = num1 + num2;
	return addnum;
}
```

با اجرای این برنامه دقیقا همان خروجی را مشاهده میکنید ولی این بار جمع دو متغییر با استفاده از فراخانی یک تابع انجام می شود.

ارسال پارامتر به یک تابع

در تابع main() دیدید که پارا مترهای ارسالی به تابع Add() دو عدد صحیح a و b بودند اما در اینجا نام آنها num1 و num2 است. همچنین دقت کنید که مقدار برگشتی این تابع addnum است در حالی که در تابع main() مقدار برگشتی را به c نسبت دادیم. در کل باید گفت که کامپایلر هنگام برخورد به تابع Add() پارامتر ها را به همان ترتیب گفته شده به آن تابع می فرستد و مقدار برگشتی را همانطور که از آن خواسته شده به یک متغییر نسبت می دهد. یعنی در اینجا مقدار a با num1 ،مقدار b با num2 و مقدار c با addnum برابر هستند.
برنامه را ساخته و اجرا کنید. خواهید دید که همانطور که انتظار داریم کار خواهد کرد.

میدان دید متغییر ها

یکی از مهمترین مفاهیم C++، میدان دید(Scope) متغیرهاست. هر متغییر را تنها در تابعی می توان به کار برد که در آن تعریف شده است و به این نوع متغییر ها محلی (Local) گفته می شود و اگر متغییری با نام a در تابع main() تعریف شده باشد فقط درون این تابع قابل استفاده است

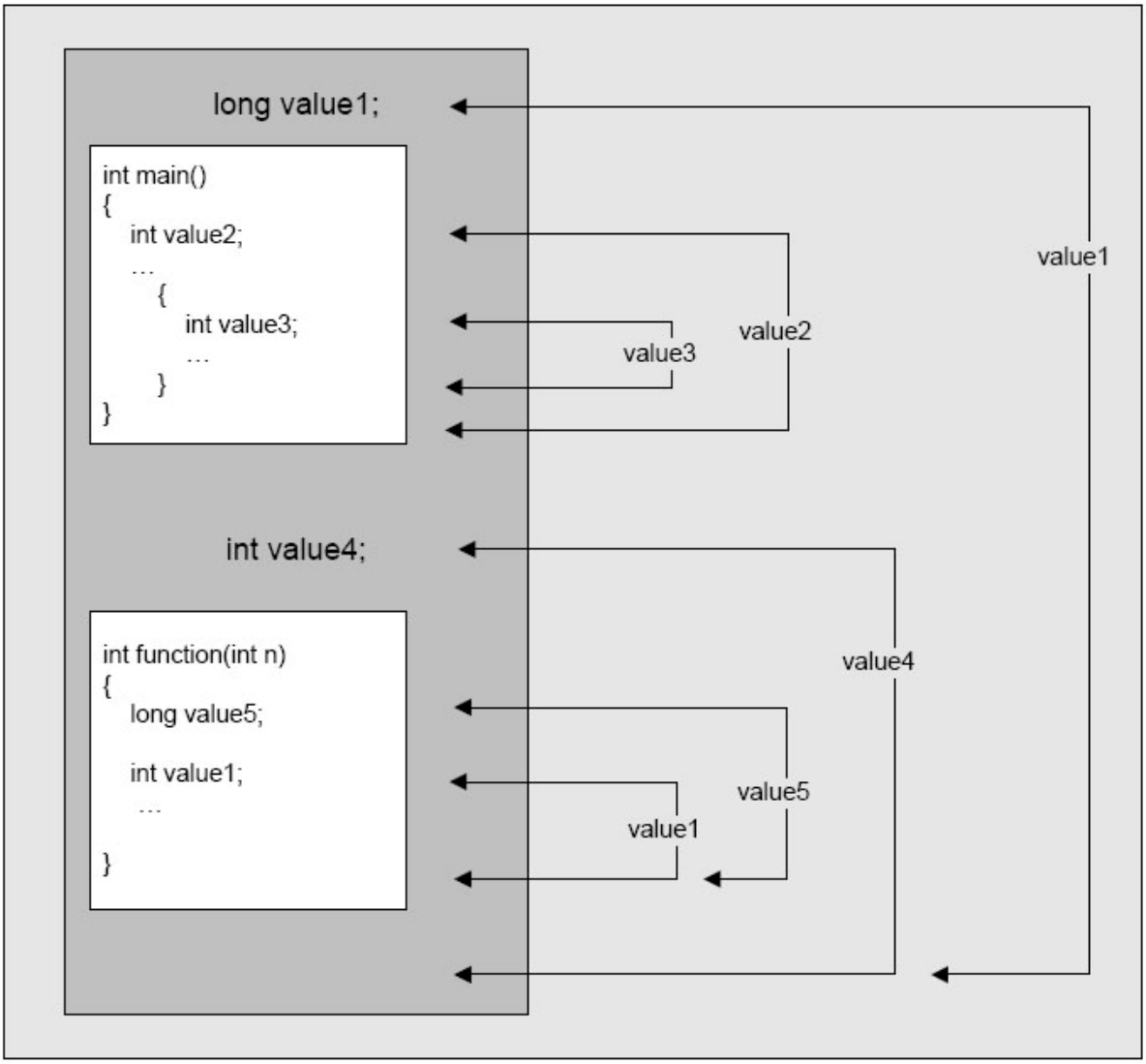
و اگر خارج از این تابع که تعریف شده به کار گرفته شود در یک تابع دیگر، کامپایلر تولید خطا خواهد کرد، یعنی متغییر **a** تعریف شده در دو تابع **()main** و **()Add** هرکدام مجزا هستند و با اجرای تابع تولید و با اتمام و خروج از آن ،از بین میروند.

نوع دیگری از متغییر ها با عنوان متغییر های همگانی (**Global**) وجود دارند که از تمام توابع و نقاط مختلف برنامه می شود آنها را تغییر داد و نیازی به تعریف مجدد در توابع ندارند، که محل تعریف آنها خارج از تابع **()main** و بالای آن است.

در مثال زیر متغییر **a** به صورت همگانی تعریف می شود سپس مقدار **1** در آن قرار گرفته و نمایش داده می شود.

```
#include "stdafx.h"
#include <iostream>

int a;

int _tmain(int argc, _TCHAR* argv[])
{
       a=1;
       std::cout << a << std::endl;
      return 0;
}
```

در عکس زیر میدان دید چند متغییر که در جاهای مختلف برنامه تعریف شده اند را مشاهده میکنید.



دستور if

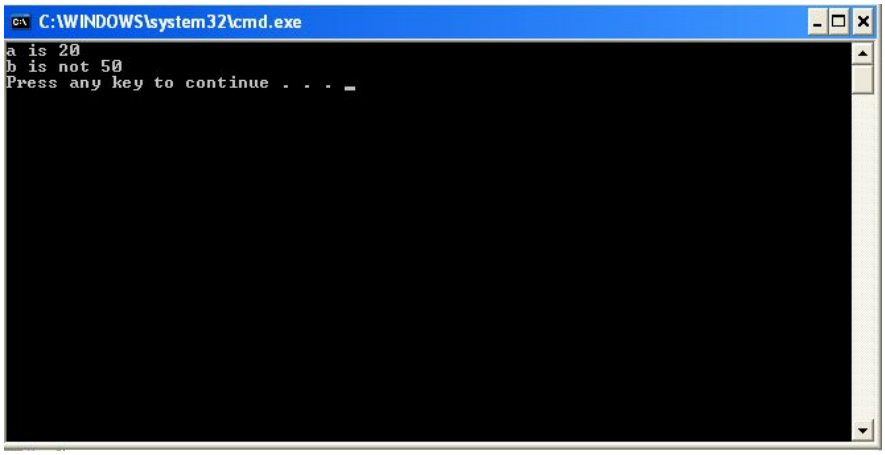
در این قسمت به دستور **if** در C++ می پردازیم. دستورات **if** مقدار یک متغییر را سنجیده و بر اساس آن کارهای مختلفی را انجام می دهد.
برای آشنا شدن با **if** به مثال زیر توجه کنید.

۱-تابع **main()** را مانند زیر اصلاح کنید:

```
#include "stdafx.h"
#include <iostream>


int _tmain(int argc, _TCHAR* argv[])
{
	int a,b;

	a=20;
	b=30;

	if(a == 20)
	{
		std::cout << "a is 20" << std::endl;
	}

	if(b == 50)
	{
		std::cout << "b is 50" << std::endl;
	}
	else
	{
		std::cout << "b is not 50" << std::endl;
	}

	return 0;
}
```

در اینجا دو متغییر تعریف شده و سپس به آنها مقدار داده شده است. اولین دستور **if** مقدار متغییر **a** را می سنجد و اگر مقدار آن ۲۰ بود (که هست) پیام "a is 20 " را می دهد. دستور **if** دوم متغییر دیگر یعنی **b** را می سنجد و اگر مقدار آن ۵۰ باشد (که نیست) پیام "b is 50 " را نشان می دهد و در غیر این صورت که چنین است پیام "b is not 50 " را نشان خواهد داد(توجه کنید که در دستور **if** برای سنجیدن تساوی از == استفاده کرده ایم نه یک =).

۲-کارتان را ذخیره کرده و برنامه را اجرا کنید، نتیجه اجرای برنامه به شکل زیر خواهد بود.



نکته:درک تفاوت = و == در C++ اهمیت زیادی دارد. علامت = برای مقداردهی بکار کی رود یعنی مقدار عبارت سمت راست علامت = در متغییر سمت چپ علامت = قرار داده می شود. ولی علامت == برای مقایسه دو مقدار بکار می رود یعنی هنگام تست شرط ها باید از == استفاده کنید.

نکته: دستورات مقایسه ای در C++ به صورت زیر است.

کوچکتر یا مساوی با	<=	کوچکتر از	<
بزرگتر یا مساوی با	>=	بزرکتر از	>
نا مساوی با	!=	مساوی با	==

استفاده از AND و OR منطقی در برنامه

در برنامه ممکن است در جایی بخواهیم در صورت برابری دوشرط با هم کدی را اجرا نماییم، مثلا در صورت برابری متغیر **a** با **b** و متغیر **c** با **d** دستوراتی را اجرا نماییم.خوب این دستور را می توان با دو دستور **if** تو در تو نوشت مانند کد زیر:

```
if(a == b)
{
     if(c == d)
     {
        std::cout << "Ok" << std::endl;
     }
}
```

اما راه حل بهتر که باعث خواناتر شدن کد برنامه می شود استفاده از دستور **AND** منطقی است،این دستور به صورت علام **&&** بین دو دستور مقایسه اي قرار مي گيرد و كارايي آن نيز به اين صورت است كه در صورت برابری دو مقدار مقایسه ای در دو طرف آن شرط اجرا می شود. حالا دستور بالا را بوسیله استفاده از **AND** منطقی و با یک دستور **if** می نویسیم:

```
if(a == b && c == d)
{
     std::cout << "Ok"  << std::endl;
}
```

حال فرض کنید که می خواهیم کدی بنویسیم که در صورت برابری **a** با **b** و یا **c** با **d** کاری را انجام دهد. در حالت معمول برای انجام این کار باید به صورت زیر عمل کنید:

```
if(a == b)
{
     std::cout << "Ok" << std::endl;
}
else
{
     if(c == d)
```

```
	{
		std::cout << "Ok" << std::endl;
	}
}
```

دستور **OR** منطقی در صورت برابری هر کدام و یا دو طرف شرط باعث اجرای آن می شود ، و شکل دستوری آن علامت || است.حال با استفاده از دستور **OR** منطقی این کد را بازنویسی می کنیم.

```
if(a == b || c == d)
{
	std::cout << "Ok" << std::endl;
}
```

می بینید که کد جدید نوشته شده چقدر ساده تر و خواناتر است.

دستور using

تمامی توابع از قبل تعریف شده موجود در کتابخانه های استاندارد **c++** با نام **std** شروع می شوند مانند تابع **cout** که باعث نمایش یک مقدار در پنجره خروجی میشد و برای استفاده از آن باید نام کامل آن را به صورت **std::cout** نوشت. این کار باعث وجود تفاوت در اسامی این توابع شده و امکان استفاده برنامه نویس را از همان نامها در طول برنامه برای متغییر و یا توابع تعریفی خود و در نتیجه بروز خطا از بین میبرد.
برای راحتی کار و کدنویسی کمتر و بدلیل اینکه برنامه نویس از تمامی این اسامی اطلاع ندارد پس فقط آنهایی را که میشناسد و قصد استفاده از آنها در برنامه خود دارد را بوسیله دستور **using** که به صورت زیر استفاده می شود ساده کرده و دیگر نیازی به استفاده از نام کامل آن در برنامه نمی باشد، در مثال زیر دیگر نیازی به استفاده از عبارت **std::** در دستورات **cout** و **endl** نیست.

```
#include "stdafx.h"
#include <iostream>

using std::cout;
using std::endl;

int _tmain(int argc, _TCHAR* argv[])
{
	int a,b;

	a=20;
	b=30;

	cout << "a = " << a << endl;
	cout << "b = " << b << endl;

	return 0;
}
```

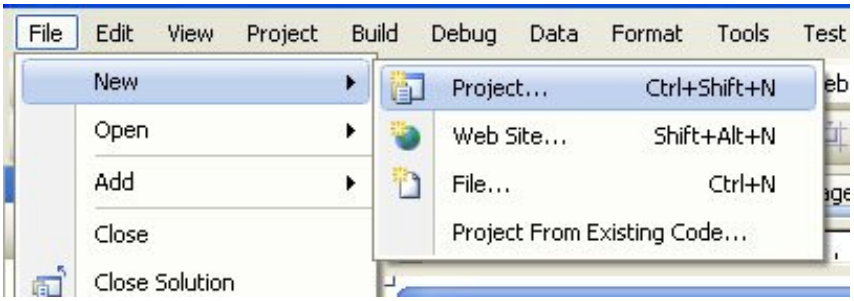
آموزش مقدماتی ویژوال سی پلاس پلاس ۲۰۰۸

بهزاد جناب

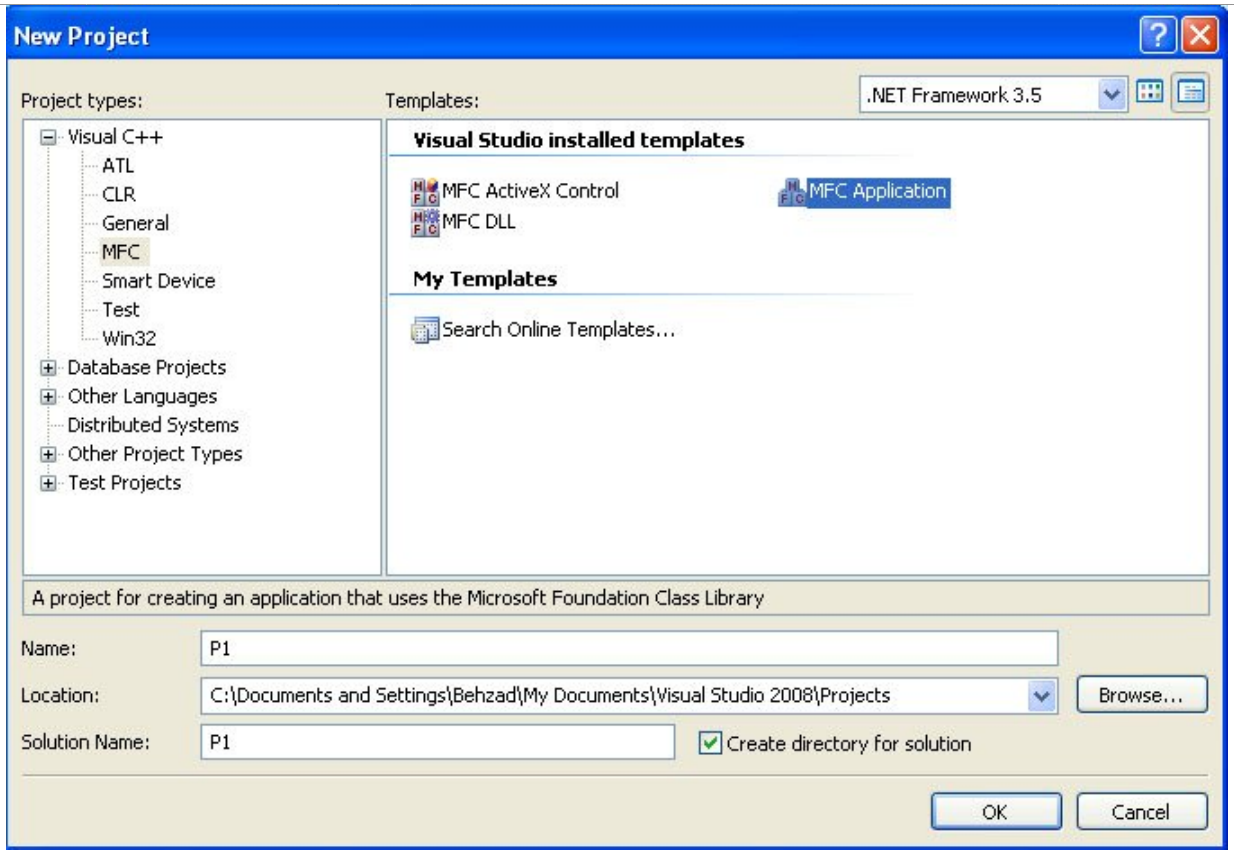
فصل دوم

اولین پروژه با کلاسهای بنیادی مایکروسافت (MFC)

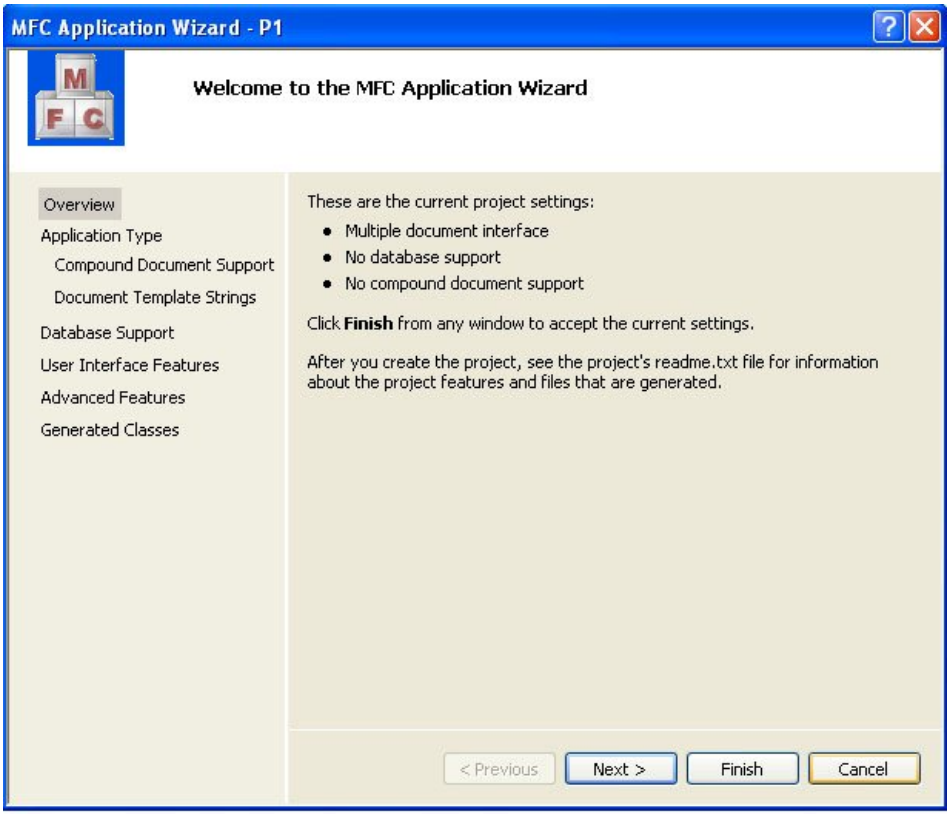
اولین برنامه **MFC** که با **C++** می نویسیم یک برنامه ساده است که یک دکمه دارد برای بستن برنامه. برای ایجاد پروژه از منوی **File** گزینه **Project** را انتخاب کنید.



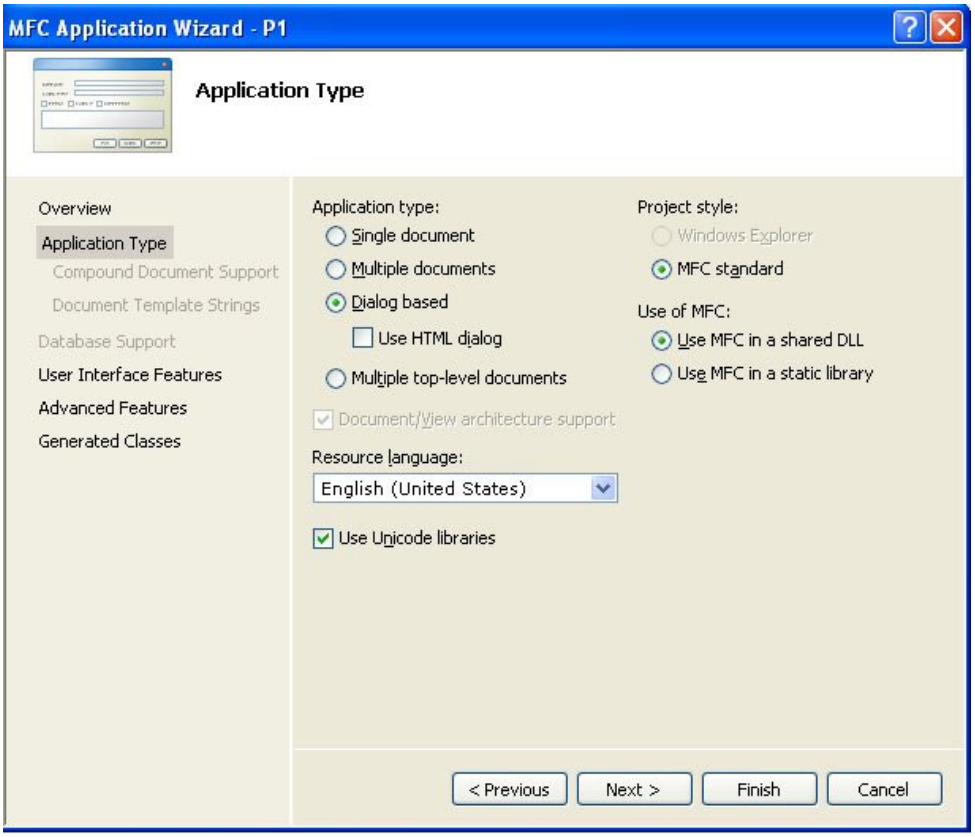
در پنجره به نمایش در آمده در سمت چپ آن گزینه **MFC** را انتخاب و سپس در سمت راست بر روی گزینه **MFC Application** کلیک نمایید، نام برنامه را **P1** بگزارید و بر روی **Ok** کلیک کنید.



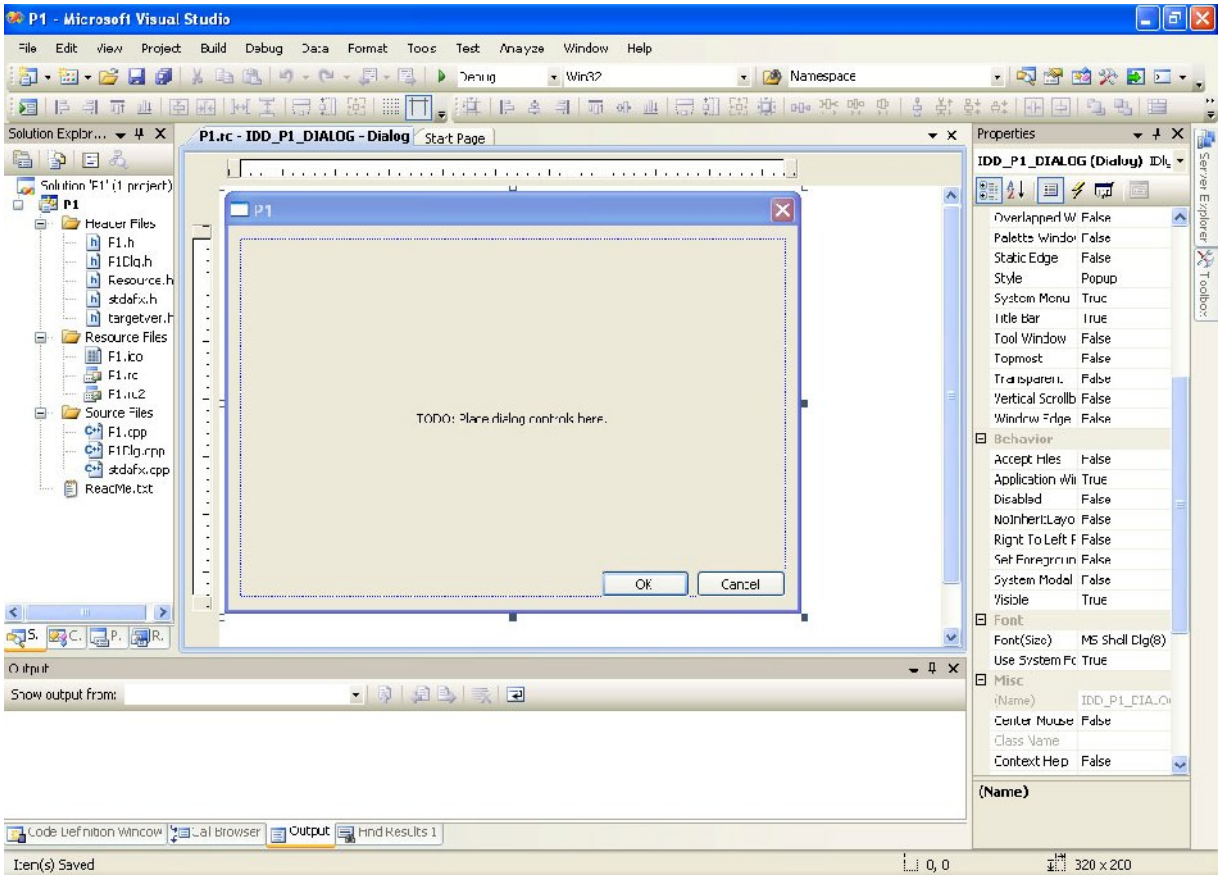
در بنجره بعدی چون قصد داریم تغییراتی در نوع تنظیمات پروژه اعمال کنیم بر روی گزینه **Next>** کلیک می نماییم.



حالا عنوان **Application Type** برنامه را به **Dialog based** تغییر داده و بر روی کزینه **Finish** کلیک می کنیم تا برنامه ساخته شود.



بعد از پایان کار به محیط ویژوال استدیو باز می گردید و پنجره برنامه شما نشان داده می شود(شکل زیر).



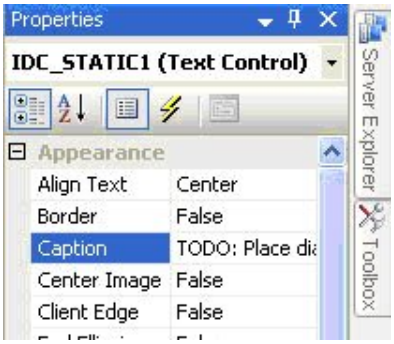
در پنجره برنامه به طور پیشفرض یک پیام TODO و دو دکمه فرمان وجود دارد ، حالا برنامه را با استفاده از انتخاب منوی Debug و گزینهStart Without Debugging و یا با فشردن کلیدهای میانبر Ctrl+F5 اجرا نمایید.خروجی برنامه به شکل زیر است که با فشردن هر کدام از دکمه های Ok یا Cancel از برنامه خارج می شود.

حالا به محیط ویژوال استدیو بر می گردیم و می خواهیم تغییراتی در شکل ظاهری پنجره برنامه اعمال کنیم.

۱. دکمه فرمان Ok را از روی پنجره حذف نماییم. برای اینکار با کلیک بر روی شکل دکمه Ok آن را انتخاب کرده و سپس کلید Delete را از روی کی بورد می فشاریم تا حذف شود و یا با راست کلیک کردن بر روی آن و انتخاب گزینه Delete از منوی باز شده نیز می توان این کار را انجام داد.

۲. پیام TODO را که در وسط پنجره مشاهده می کنید به بالای آن انتقال دهید. برای این کار کافی است آن را نتخاب کنید و در حالی که دکمه چپ ماوس را پایین نگه داشته اید آن را به بالا یا هر جای دیگر دلخاه انتقال دهید.

۳. متن پیام TODO را تغییر دهید. برای تغییر متن پیام ابتدا آن را انتخاب کرده سپس در پنجره Properties سمت راست محیط ویژوال استدیو اطلاعات جلوی فیلد Caption را به متن مورد نظر یعنی اولین پروژه با کلاسهای بنیادی مایکروسافت تغییر دهید.

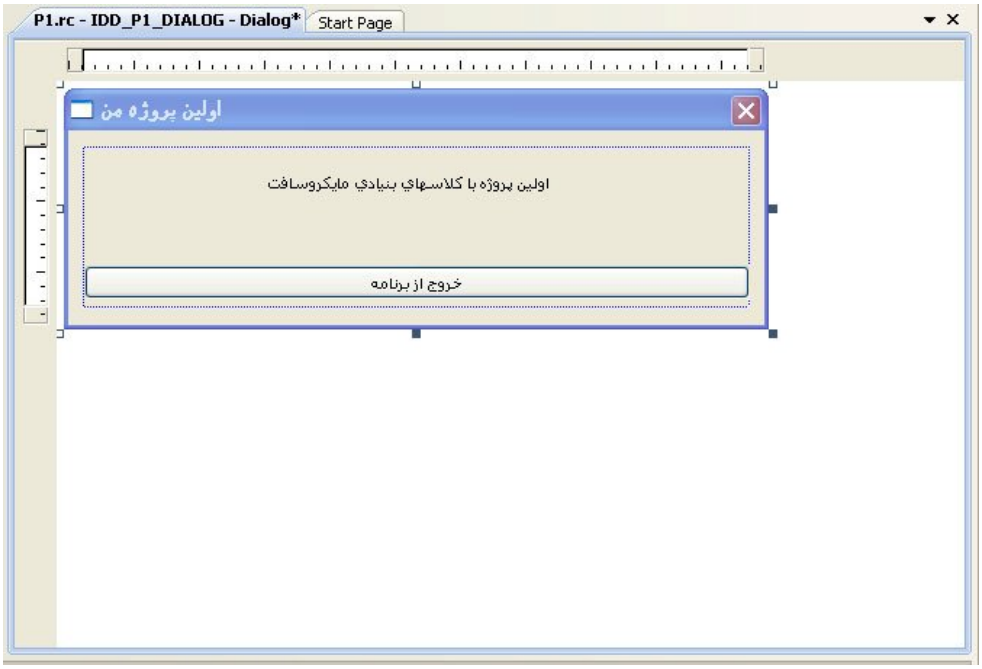


۴. متن بالای پنجره برنامه که P1 نوشته شده را تغییر دهید. برای تغییر متن بالای پنجره شبیه بالا ابتدا آن را انتخاب کرده سپس در پنجره Properties سمت راست محیط ویژوال استدیو اطلاعات جلوی گزینه Caption را به متن مورد نظر یعنی اولین پروژه من تغییر دهید.

۵. اندازه پنجره برنامه را کوچک کنید. با انتخاب پنجره برنامه دستگیره های تغییر اندازه آن فعال می شود، سپس با استفاده از آن دستگیره ها پنجره را کوچک نمایید.

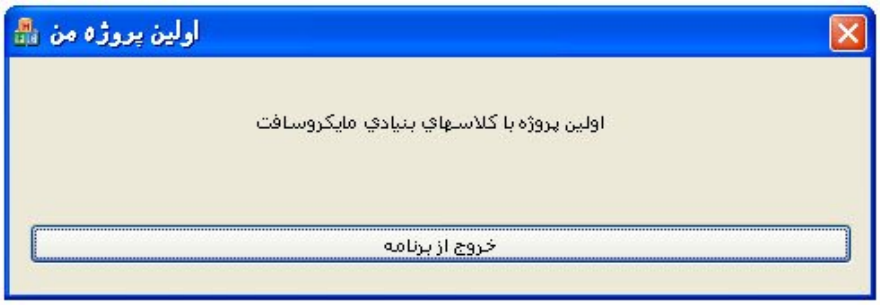
۶. دکمه **Cancel** موجود در روی پنجره را کشیده تر کنید و نام آن را نیز با استفاده از عنوان **Caption** به خروج از برنامه تغییر دهید.

پنجره برنامه چیزی شبیه به شکل زیر خواهد بود.



حالا برنامه را ذخیره و سپس اجرا نمایید.پس از اجرا خواهید دید که تمامی نوشته های فارسی شما به علامت ؟ تبدیل شده اند اما دلیل آن چیست؟ برای استفاده از زبان فارسی در این پنجره باید تنظیمات زبان فارسی را برای آن انتخاب نمایید.برای انجام این کار از پنجره سمت چپ محیط ویژوال استدیو سربرگ **Resource View** را انتخاب کنید، سپس با کلیک بر روی علامتهای مثبت موجود که به صورت درختی هستند با جستجوی محتوای آن به دنبال عبارت **IDD_P1_DIALOG** بگردید و آن را انتخاب کنید.مواضب باشید که به هنگام انتخاب آن بر روی آن دابل کلیک نکنید.سپس در فیلد **Language** در قسمت **Properties** زبان را **Farsi** قرار دهید.

حالا دو باره برنامه را اجرا کنید(اگر برنامه تان را نبسته اید حتما قبل از اجرای مجدد آن را ببندید)، می بینید که مشگل حل شده است.



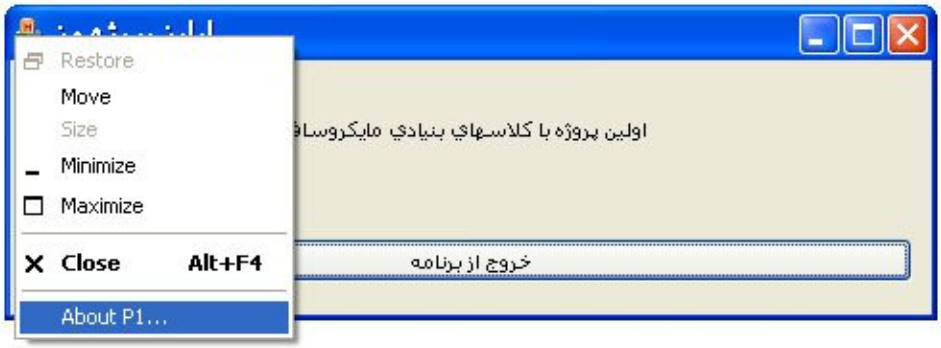
برنامه را بوسیله گزینه **Save All** از منوی **File** ذخیره کنید و از محیط ویژوال استدیو خارج شوید ، حالا ویژوال استدیو را دوباره اجرا نمایید و برنامه تان را بوسیله گزینه **Open** از منوی **File** باز کنید. اما حالا پنجره ای که طراحی کرده اید را نمی بینید، برای مشاهده پنجره برنامه باید از پنجره سمت چپ محیط ویژوال استدیو از سر برگ **Resource View** بر روی عبارت **IDD_P1_DIALOG[Farsi]** دابل کلیک نمایید.

اضافه کردن دکمه های حداقل و حداکثر

برای اضافه کردن دکمه های حداقل و حداکثر به میله عنوان پنجره برنامه ابتدا پنجره دیالوگ برنامه را انتخاب کنید، سپس در پنجره **Properties** محتوای دو فیلد **Maximize Box** و **Minimize Box** را از **False** به **True** تقییر دهید. عبارت **False** به معنای نادرستی و **True** به معنای درستی است.

پنجره توضیحات برنامه

در برنامه شما پنجره دیگری نیز به نام **About** وجود دارد که با راست کلیک کردن بر روی عنوان پنجره و انتخاب گزینه نمایش داده می شود.



در این پنجره معمولا توضیحاتی درباره نویسنده ، سال تولید و نصخه برنامه قرار داده می شود.

شما یا باید این پنجره را از برنامه خود حذف نمایید و یا به شکلی صحیح آنرا ویرایش کنید. برای انتخاب آن از پنجره سمت چپ در سر برگ **Resource View** عبارت **IDD_ABOUTBOX** را انتخاب کنید(دابل کلیک نکنید). برای فارسی سازی این پنجره شبیه بالا عمل می کنیم ،از پنجره **Properties** در سمت راست محیط ویژوال استدیو و از فیلد **Language** زبان را **Farsi** قرار دهید.در این مرحله با دابل کلیک کردن بر روی عبارت **IDD_ABOUTBOX** این پنجره نماین می گردد، حالا آن را به شکل دلخواه ویراش کنید.

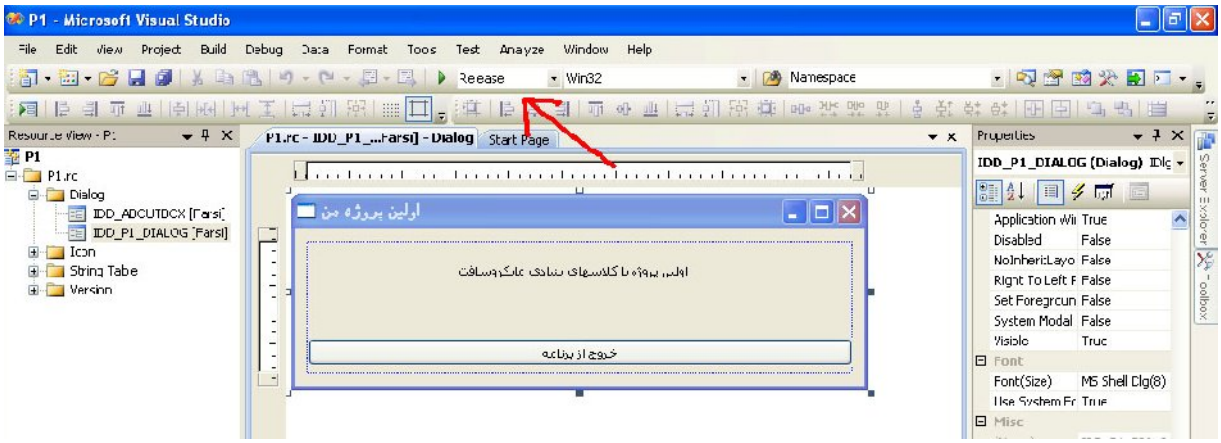
طراحی آیکونها

اگر به آیکون برنامه که در کوشه چپ، بالای پنجره برنامه است نگاه کنید خواهید دید که این آیکون از سه مکعب با حروف F،M و C تسگیل شده است. این سه حرف معرف کلاسهای بنیادی مایکروسافت (Microsoft Foundation Classes) هستند و میگویند که این برنامه با استفاده از کتابخانه های C++ ساخته شده است، ولی شما که میل ندارید این مطلب را همه جا جار بزنید؟ پس باید آیکون برنامه را عوض کنیم. برای تغییر شکل آیکونهای برنامه باید از پنجره سمت چپ ویژوال استدیو سربرگ Solution Explorer را انتخاب کرده ،سپس بر روی فایل P1.ico دابل کلیک نمایید تا پنجره آیکونهای برنامه نمایش داده شود. حالا می توانید آنها را ویرایش کنید.

کامپایل نهایی

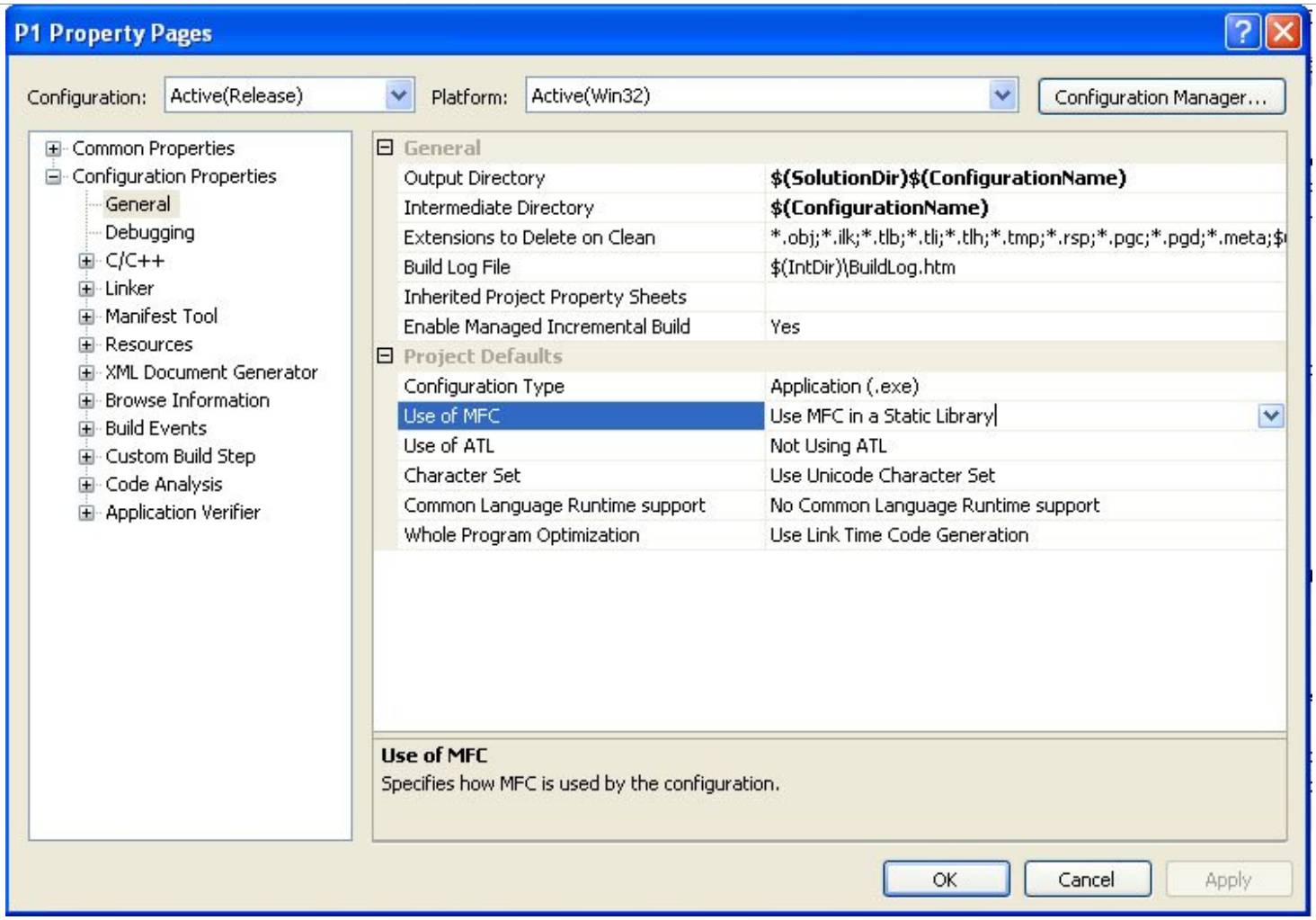
حالا فرض کنید که قصد دارید این برنامه را به عنوان اولین برنامه ای که با C++ نوشته این به خانه دوستتان برده و به او نشان دهید. به طور پیش فرض هر برنامه ای در C++ بر روی حالت Debug کامپایل می شود، برنامه ای که در این حالت کامپایل شود ،غیر از بر روی کامپیوتر خودتان بر روی هیچ کامپیوتر دیگری اجرا نخواهد شد، پس ابتدا باید روش کامپایل برنامه را بر روی حالت Release قرار دهید و سپس آنرا اجرایی نمایید.

(به عکس زیر توجه کنید)



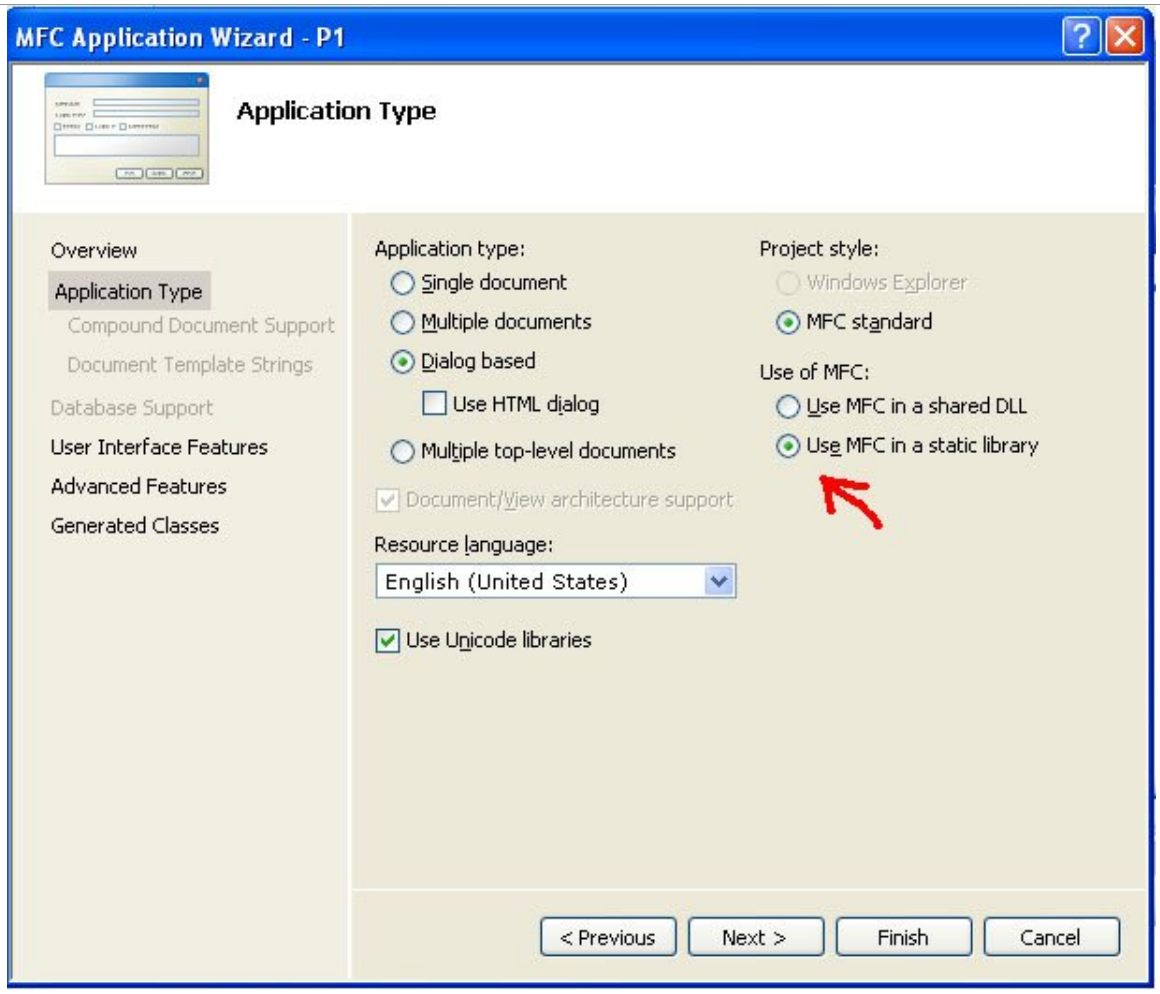
اما برنامه شما هنوز هم یک برنامه مستقل نیست، چون در این برنامه از توابعی استفاده شده است که در کتابخانه های(DLL) مختلفی وجود دارند و شما می بایست آن کتابخانه ها را در پوشه ای که برنامه وجود دارد و یا پوشه سیستم ویندوز کپی نمایید تا برنامه به توابع آن که توسط برنامه فراخانی می شود دسترسی داشته باشند که در این حالت کتابخانه ها به صورت اشتراکی استفاده می شوند(Shared DLL). اما برای حل این مشگل می توانید کتابخانه های مورد نیاز برنامه را به آن پیوست کنید که در این حالت کتابخانه ها به همراه برنامه کامپایل می شود(Static Library)، در این صورت کمی به حجم برنامه اضافه می شود ولی برنامه شما دیگر به راحتی در هر کامپیوتری ، مستقل و بدون نیاز یه نصب (Portable) اجرا خواهد شد.

برای انجام این کار از منوی Project گزینه P1 Properties... را انتخاب کنید. سپس از پنجره نمایش داده شده ، سمت چپ از عبارت Configuration Properties گزینه General را انتخاب نمایید و از تنظیمات آن که در سمت راست نمایش داده شده، فیلد Use of MFC را از Use MFC in a Shared DLL به Use MFC in a static library تغییر دهید و بر روی Ok کلیک کنید،



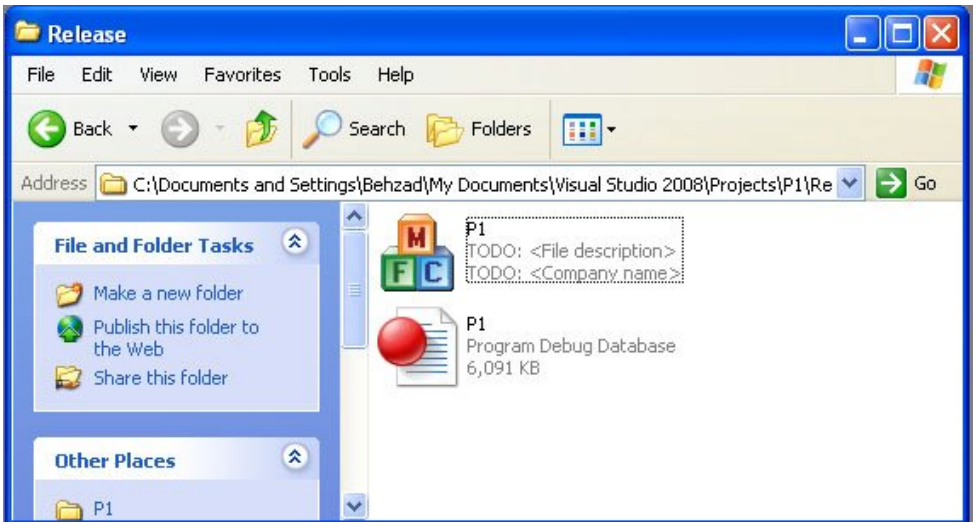
دوباره برنامه را کامپایل نمایید. اگر به دنبال برنامه تان هستید نصخه نهایی و اجرایی پروژه شما با نام **P1.exe** در پوشه **My Documents** ویندوز در مسیر **Visual Studio 2008\Projects\P1\Release** است.حالا می توانید با خیال راحت برنامه را برای اجرا به یک کامپیوتر دیگر منتقل کنید.اما روش دیگری نیز برای پیوست کردن توابع به برنامه در هنگام ساخت پروژه جدید وجود دارد .در این روش هنگام ساخت یک پروژه جدید تنظیمات برنامه را بر روی گزینه **Use MFC in a static library** قرار میدهیم تا هنگام کامپایل و اجرایی شدن برنامه کتابخانه های مورد نیاز به آن پیوست شوند.

به عکس زیر دقت کنید.

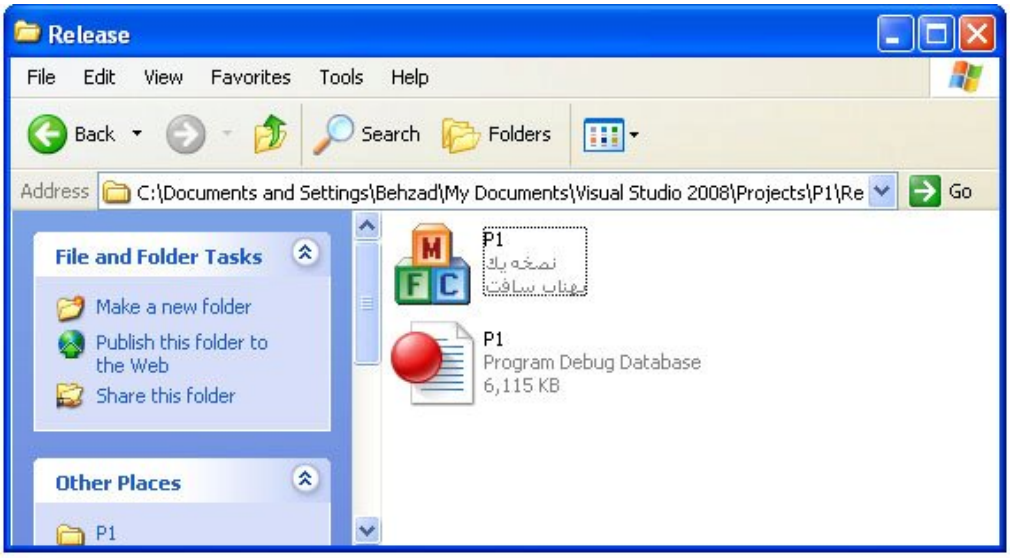


تغییر مشخصات فایل P1.exe :

اگر فایل اجرایی خود را بوسیله My Computer ویندوز تماشا کنید خواهید دید که مشخصاتی که با کلمه TODO شروع می شود در زیر نام فایل وجود دارد، این مشخصات در واقع باید نشان دهنده شرکت تولید کننده و شماره نصخه برنامه شما و یا از این قبیل باشد ولی چگونه می شود آنها را به شکلی زیبا تغییر و ویرایش نمود تا مشخصات شما و برنامه تان را نشان دهد؟



برای این کار از پنجره سمت چپ ، سر برگ **Resource View** را انتخاب کرده و بر روی عبارت **VS_VERSION_INFO** دابل کلیک کنید، سپس اطلاعات این بخش نمایش داده می شود و شما می توانیید آنها را به طور دلخواه ویرایش نمایید.چنانچه قصد دارید مشخصات فایل خود را به زبان فارسی ویرایس نمایید یک بار بر روی عبارت **VS_VERSION_INFO** کلیک نمایید سپس از پنجره **Properties** فیلد **Language** را به **Farsi** تغییر دهید.حالا مشخصات شما در پنجره **My Computer** نشان داده می شود.



آموزش مقدماتی ویژوال سی پلاس پلاس ۲۰۰۸

بهزاد جناب

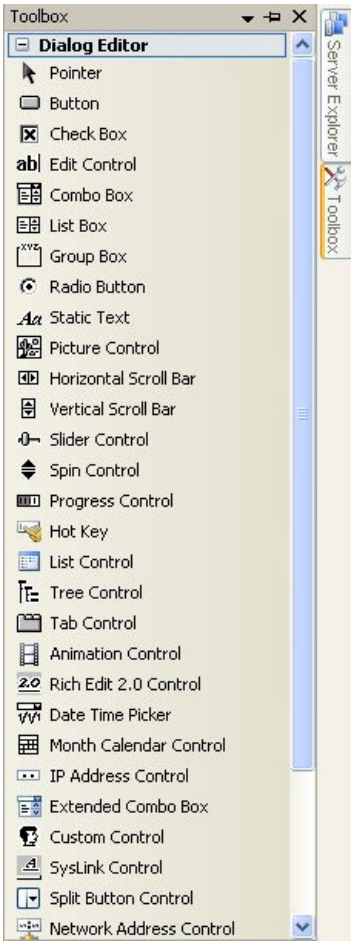
فصل سوم

کنترل های اصلی ویندوز

ویندوز دارای چندین کنترل استاندارد، از جمله کنترل های لیست(**List**) ، لغزنده (**Slider**) ها ، میله های پیشرفت(**Progres Bar**) و از این قبیل است. در درس امروز با تعدادی از اساسی ترین کنترل های ویندوز کار خواهیم کرد:

- متن ثابت (**static text**)
- جعبه ادیت (**edit box**)
- دکمه فرمان (**command button**)
- جعبه چک (**check box**)
- دکمه رادیویی (**radio button**)
- جعبه لیست باز شو (**drop-down list box**) یا جعبه ترکیبی (**combo box**)

استفاده از این کنترلها در **Visual C++** بسیار ساده است. این کنترل ها را می توانید. در پنجره کشویی سمت راست با عنوان **Toolbox** مشاهده کنید.(شکل زیر)



کنترل متن ثابت

کنترل متن ثابت معمولا برای ارایه اطلاعات به کابر بکار می رود. کاربر قادر به دستکاری متن این کنترل نیست و در واقع این یک کنترل فقط خواندنی است. اما متن این کنترل را از طریق کد برنامه می توان تغییر داد.

کنترل جعبه ادیت

کنترل جعبه ادیت به کاربر امکان میدهد تا متن را وارد کرده و یا آن را دستکاری کند. این کنرل یکی از اصلیترین ابزارهای دریافت اطلاعات از کاربر و برقراری ارتباط با وی است. این کنترل فقط متن را برمی گرداند هیچگونه اطلاعاتی درباره فرمت این متن منتقل نخواهد کرد.

کنترل دکمه فرمان

دکمه فرمان کنترلیست که کاربر می تواند بکمک آن عملی را انجام دهد. دکمه فرمان معمولا ارای عنوانی است که عملکرد آنرا توضیح می دهد. عنوان دکمه فرمان می تواند تصویر یا ترکیبی از متن و تصویر باشد.

کنترل جعبه چک

جعبه چک کنترلیست که کاربر می تواند آن را فعال یا غیر فعال کند. با فعال یا غیر فعال شدن جعبه چک مقدار این کنترل تغییر خواهد کرد. از جعبه چک معمولا برای کنترل متغییر هایی که فقط دو مقدار می گیرند استفاده می شود.

کنترل دکمه رادیویی

دکمه رادیویی هم کنترلیست که می تواند دو وضعیت فعال و غیر فعال داشته باشد و از این نظر شبیه جعبه چک است. تفاوت این دو درآن است که از یک گروه دکمه رادیوئی در هر لحظه فقط یکی می تواند فعال باشد. دکمه های رادیویی معمولا گروهی مورد استفاده قرار می گیرند که هر گروه رادیویی دارای عملکرد مستقلی است.

کنترل جعبه لیست باز شو

کنترل جعبه لیست باز شو یا کنترل جعبه ترکیبی یک جعبه ادیت است که به یک لیست متصل شده و کاربر می تواند یک گزینه را از میان چندین گزینه آن انتخاب کند. اگر گزینه مورد نظر کاربر در میان گزینه های لیست نباشد، وی می تواند آنرا در جعبه ادیت وارد کند.

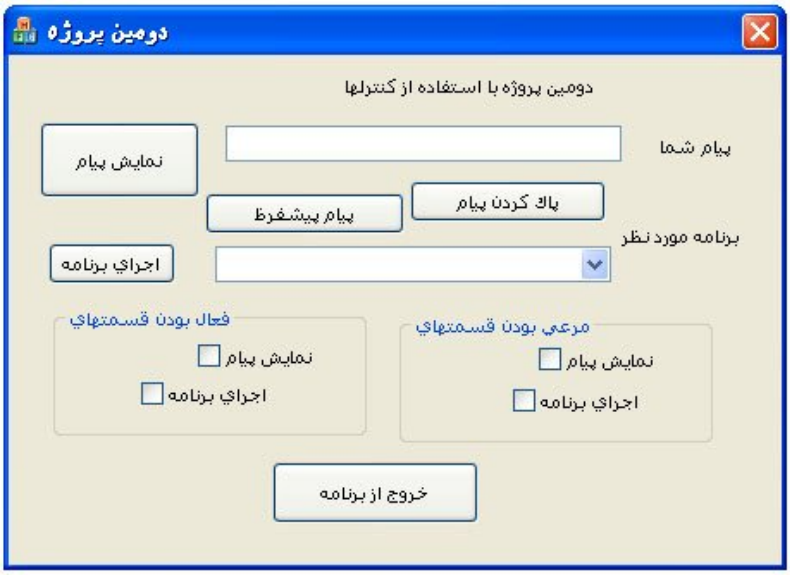
استفاده از کنترل ها در برنامه

برنامه ای که امروز قصد داریم آنرا بنویسیم برنامه ایست که از تعدادی کنترل های ویندوز در یک دیالوگ استفاده می کند.این کنترل ها عملکردهای متفاوتی دارند. در قسمت بالای پنجره جعبه ادیتی وجود دارد که کاربر می تواند پیام مورد نظرش را درآن بنویسد ، این پیام هنگام کلیک شدن دکمه کنار جعبه ادیت نمایش داده خواهد شد. در زیر این جعبه ادیت دو دکمه دیگر هستند که یکی جعبه ادیت را پاک می کند و دیگری یک پیام از پیش تعیین شده را در آن می نویسد. در زیر این دکمه ها یک جعبه لیست باز شو قرار دارد که در آن تعدادی از برنامه های ویندوز را قرار داده ایم، کابر می تواند با انتخاب یکی از این برنامه ها و کلیک کردن دکمه کنار لیست آن برنامه را اجرا کند.در زیر این جعبه لیست دو گروه جعبه چک قرار دارند که هرکدام عملکرد کنترلهایی دیکر موجود بر روی پنجره برنامه را تاحت تاثیر قرار می دهند.گروه سمت چپ ، کنترلهای دیگر را فعال یا غیر فعال می کنند و گروه سمت راست ، این کنترلها را مرئی یا نا مرئی می کنند. در منتهی الیه پایین دیالوگ هم یک دکمه قرار دارد که با آن می توانید برنامه را ببندید.

طراحی و ایجاد برنامه

با استفاده از آنچه که در فصل قبل آموخته اید، برنامه امروز را با دنبال کردن مراحل زیر ایجاد کنید:

۱. یک پروژه جدید MFC Application با نام P2 ایجاد کرده و تنظیمات آن را بر روی Dialog based و Use MFC in a static library قرار دهید.
۲. تنظیمات زبان پنجره دیالوگ برنامه را به فارسی تغییر دهید.
۳. تمامی کنترلهای موجود روی پنجره برنامه را حذف کرده و کنترلهای جدیدی به شکل زیر بر روی آن قرار دهید.



طراحی پنجره برنامه

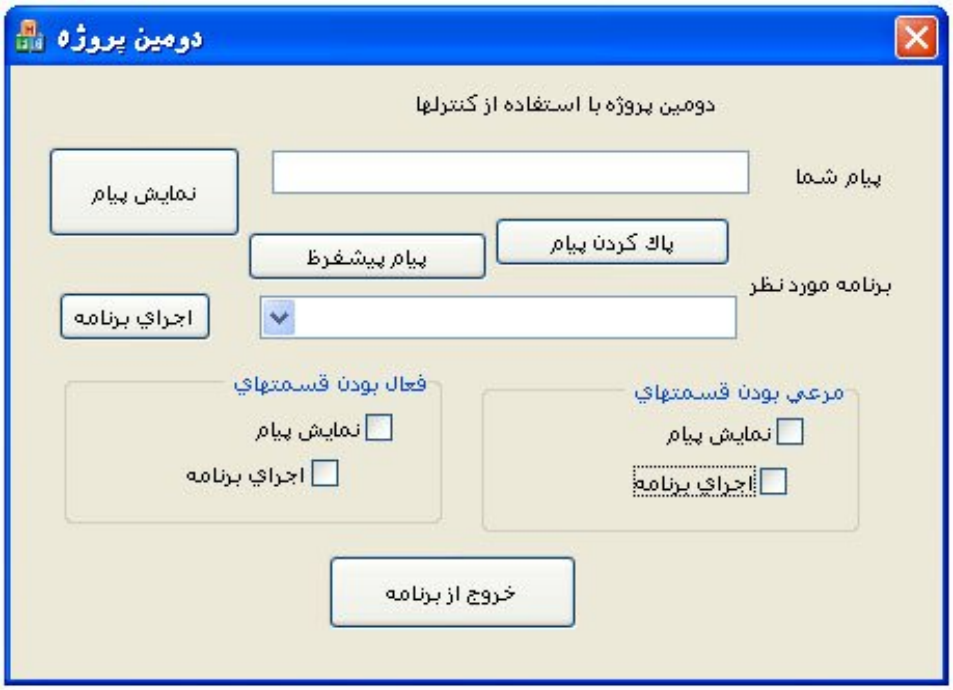
مثلا برای قرار دادن یک دکمه فرمان به برنامه ابتدا بر روی سربرگ Toolbox واقع در پنجره Properties سمت راست محیط ویژوال استدیو بروید تا پنجره کشویی آن باز شود ، حالا بر روی عبارت Button که برای ایجاد دکمه فرمان است کلیک کنید.

حالا یک دکمه فرمان بر روی پنجره برنامه رسم کنید و با استفاده از فیلد Caption واقع در پنجره Properties نام آن را عوض نمایید.
برای قرار دادن دیگر کنترلها بر روی پنجره برنامه نیز بوسیله همین روش ابتدا از پنجره کشویی Toolbox کنترلهای مورد نظر را انتخاب کنید، سپس با کلیک کردن روی پنجره دیالوگ برنامه آنرا ایجاد نمایید و با استفاده از فیلد Caption واقع در پنجره Properties نام هر یک از کنترلها را شبیه عکس بالا تغییر دهید. اما پیش فرض این کنترلها برای زبان انگلیسی که از چپ به راست است تعریف شده و چون ما از زبان فارسی استفاده کرده ایم که از راست به چپ است باید تغییراتی در نحوه نمایش این کنترلها برای زیباتر شدن آن اعمال کنیم و تنظیمات بعضی از کنترلهای برنامه را از راست به چپ تغییر دهیم.مراحل زیر را انجام دهید

در حالی که کلید کنترل از روی صفحه کلید پایین نگهداشته اید بوسیله کلید چپ ماوس

۱. کنترل متن (Edit Control) را انتخاب نمایید،
۲. کنترل جعبه لیست باز شو(Combo Box Control) را انتخاب نمایید،
۳. دو عدد کنترل گروه (Group Box Control) را انتخاب نمایید،
۴. چهار عدد کنترل جعبه چک (Check-box Control) را نتخاب نمایید.

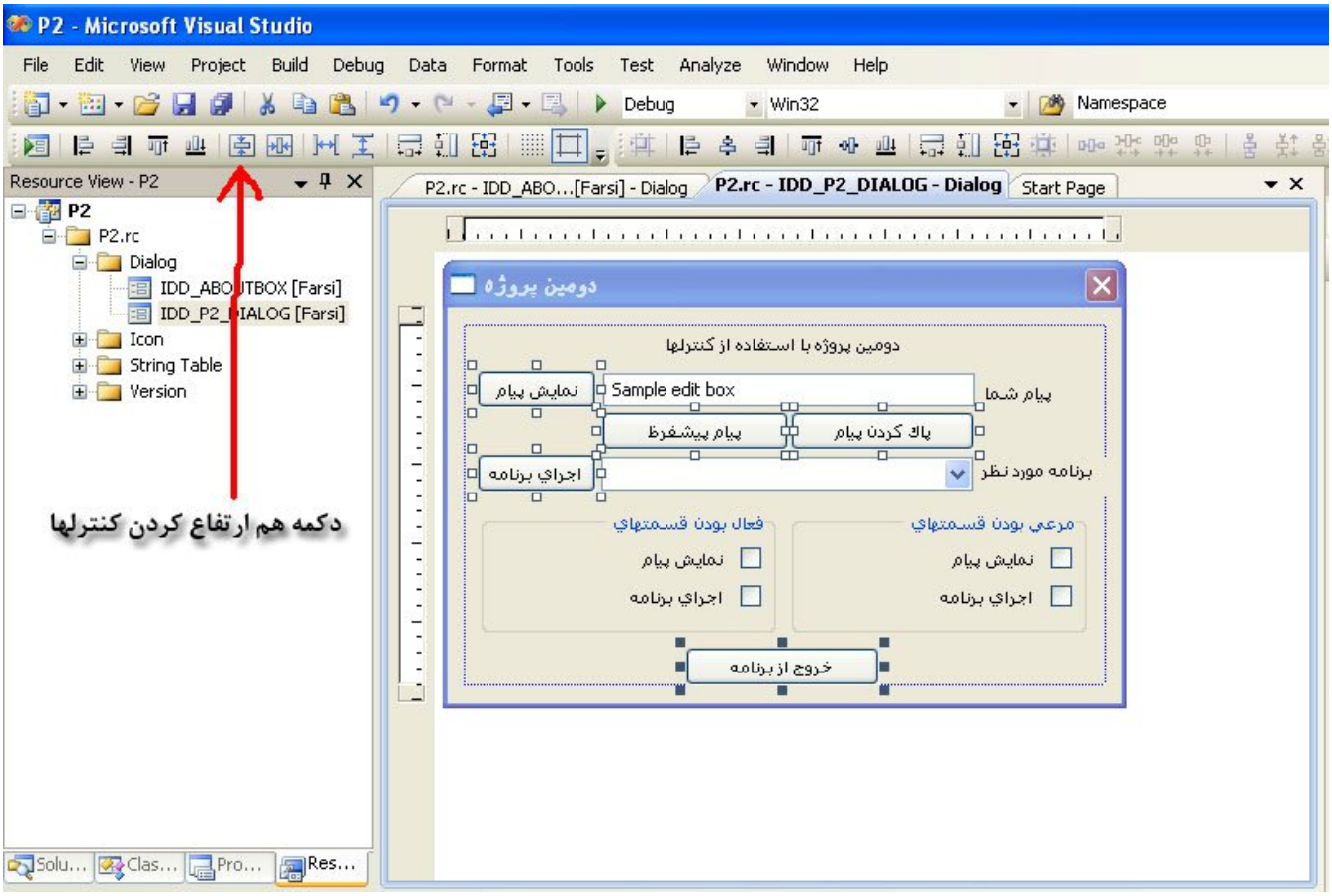
سپس از پنجره Properties دو فیلد Right Align Text وRight To Left Reading Order را به True تغییر دهید. چون این دو فیلد در تمامي اين كنترلها وجود دارد لذا با اينكار به يكباره تمامي آنها را با هم تنظيم ميكنيم و صرفه جويي در وقت مي نماييم.در اين مرحله پنجره برنامه به شكل زير خواهد شد.



پس از قرار دادن ، تغییر نام و تنظیم کردن خواص کنترلها نوبت به چیدن آنها می رسد

در این مرحله مرتب کردن اینهمه کنترل بوسیله استفاده از چشم و تغییر اندازه دستی کار مشگلی است که احتمال خطا نیز وجود دارد. ویژوال استدیو برای انجام این کار ابزارهای مفیدی دارد که بوسیله آنها می شود چند کنترل را یک اندازه کرد و یا بر روی یک خط در پنجره مرتب نمود و از این قبیل کارای هایی که برای زیبا شدن پنجره برنامه مفید هستند. این ابزارها به صورت آیکون های کوچکی در بالای محیط ویژوال استدیو قرار دارند. برای مثال شما می خواهید ارتفاع تمامی دکمه های فرمان موجود در پنجره برنامه تان را یک اندازه کنید ، برای این کار کلید کنترل (Ctrl) از روی کیبورد را نگه داشته و سپس بر روی تک تک دکمه های فرمان کلیک می کنیم تا تمامی آنها انتخاب شوند ، حالا بر روی دکمه هم ارتفاع کردن کلیک می کنیم تا ارتفاع تمامی دکمه فرمانها به یک اندازه درآیند. دکمه های مفید دیگری نیز در این قسمت جهت مرتب کردن از یک چهار جهت و وسط چین کردن و ... وجود دارند که شما خودتان با آزمایش و خطا از عملکرد آنها مطلع شوید.

به عکس زیر دقت کنید.



پنجره برنامه پس از مرتب شدن به شکل زیر در خواهد آمد

ست کردن نام شناسایی (ID) کنترلهای برنامه

در این مرحله می خواهیم نام کنترلهای برنامه را عوض کنیم،این نامی است که در کد های برنامه برای دسترسی به آن کنترل از آن استفاده می کنیم. هر کنترل ایجاد شده در ابتدا یک نام به صورت پیشفرض دارد. مثلا نام(ID) اولین دکمه فرمان ایجاد شده توسط شما **BUTTON1**

دومین دکمه فرمان BUTTON2 و ... می باشد و در کل به همین شکل برای بقیه کنترلها نیز صدق میکند.اما برای ساده تر کردن و آشنایی با روش تغییر نام کنترلها قصد داریم آنها تغییر دهیم.

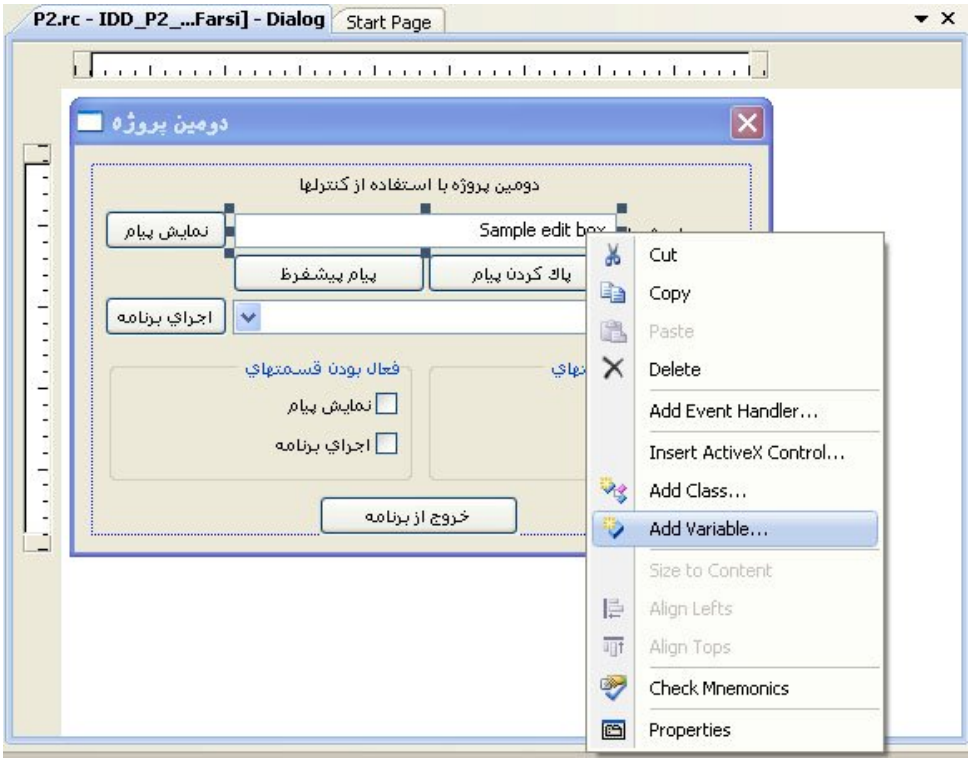
تمامی کنترلها را تک به تک انتخاب کرده و از پنجره Properties مقدار فیلد ID آنها را با جدول زیر برابر نمایید.

نوع کنترل و عنوان آن	نام کنترل ID	نوع کنترل و عنوان آن	نام کنترل ID
کنترل متن ثابت بالای پنجره	IDC_STATIC	کنترل دکمه فرمان پاک کردن پیام	IDC_CLRMSG
کنترل متن پیام شما	IDC_STATICMSG	کنترل دکمه فرمان اجرای برنامه	IDC_RUNPGM
کنترل متن برنامه مورد نظر	IDC_STATICPGM	کنترل دکمه فرمان خروج از برنامه	IDC_EXIT
کنترل ویرایش متن جهت نمایش پیام	IDC_MSG	کنترل جعبه چک مرعی شدن نمایش پیام	IDC_CKSHWMSG
کنترل جعبه لیست باز شو جهت اجرای برنامه	IDC_PROGTORUN	کنترل جعبه چک مرعی شدن اجرای برنامه	IDC_CKSHWPGM
کنترل دکمه فرمان نمایش پیام	IDC_SHWMSG	کنترل جعبه چک فعال شدن نمایش پیام	IDC_CKENBLMSG
کنترل دکمه فرمان پیام پیشفرظ	IDC_DFLTMSG	کنترل جعبه چک فعال شدن اجرای برنامه	IDC_CKENBLPGM

نسبت دادن متغیر به کنترل ها

اگر قبلا با Visual Basic برنامه نویسی کرده باشید شاید تصور کنید که در این مرحله آماده اید تا کد بنویسید.خوب ، در مورد Visual C++ چنین چیزی نیست.قبل از شروع کد نویسی باید به تمام کنترلها به جز متن ثابت و دکمه فرمان متغییر نسبت دهیم. کد نویسی در VC++ یعنی کار با این متغییرها. مقداری که کاربر در هر کنترل وارد می کند به این متغییر ها داده می شود. تا بعدا در برنامه مورد استفاده قرار گیرد. برنامه هم برای تغییر دادن محتوی کنترلها باید مقادیر مورد نظر را در این متغییرها قرار دهد.

اما این متغیرها را چگونه باید تعریف کرد؟ بر روی کنترل جعبه ویرایش متن که قصد تعریف متغییر برای آن دارید راست کلیک کنید و از منوی باز شده گزینه ADD Variable... را انتخاب نمایید.



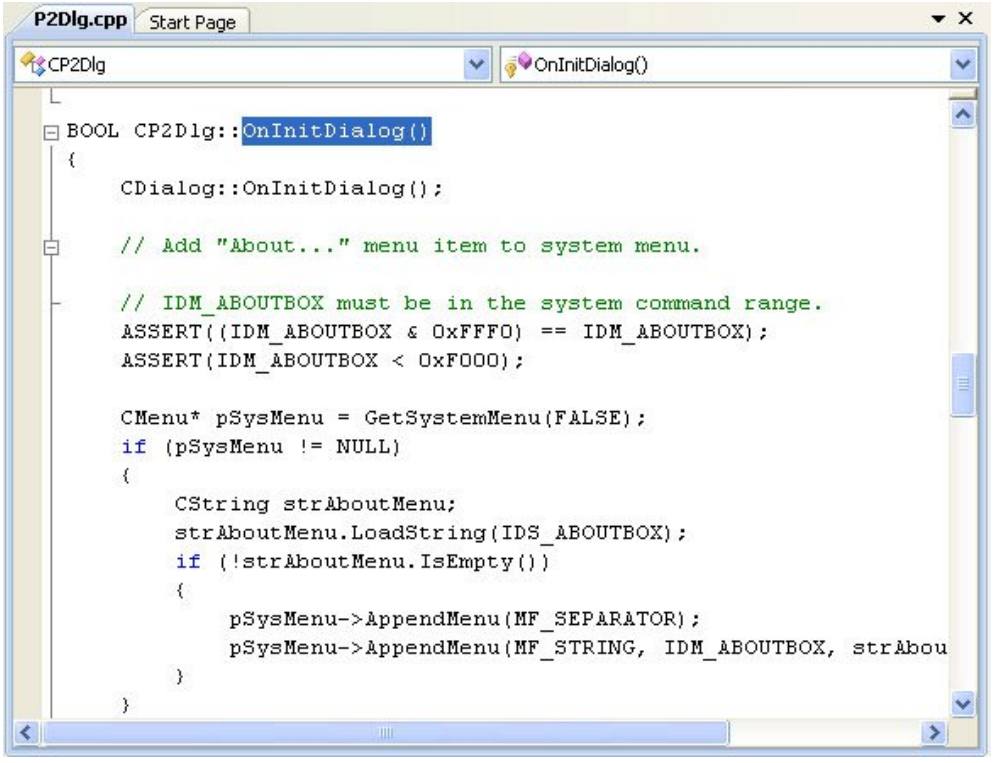
سپس در پنجره به نمایش در آمده عنوان Category را به Value تغییر دهید و فیلد Variable name نام متغییر را m_strMessage قرار دهید و بر روی Finish کلیک کنید تا متغییر برای این کنترل تعریف شود.
کنترلهایی که به تعریف متغییر نیاز دارند را با همین روش طبق جدول زیر تعریف و نامگذاری نمایید.

نام کنترل (ID)	نام متغییر (Variable name)	مقوله (Category)	نوع متغییر (Variable Type)
IDC_MSG	m_strMessage	Value	CString
IDC_PROGTORUN	m_strProgToRun	Value	CString
IDC_PROGTORUN	m_strCB1	Control	CComboBox
IDC_CKENBLMSG	m_bEnableMsg	Value	BOOL
IDC_CKENBLPGM	m_bEnablePgm	Value	BOOL
IDC_CKSHWMSG	m_bShowMsg	Value	BOOL
IDC_CKSHWPGM	m_bShowPgm	Value	BOOL

توجه: در دنیای MFC نام متغیرهای عضو کلاس با m_ شروع می شود. بعد از آن با چند حرف نوع متغییر مشخص مشود،(مثلا b یعنی متغیر منطقی و str یعنی متغیر رشته ای و غیره ...) و به دنبال آن نام متغییر آورده می شود. این استاندارد در تمام کتابهای برنامه نویسی VC++ و MFC رعایت می شود.

عملیاتی کردن کنترلها

قبل از شروع کد نویسی برای برای کنترلها ابتدا باید به تعدادی از متغییر های عضو مقدار داده و آنها را آماده سازی کنید. برای انجام این کار ابتدا از پنجره سمت چپ محیط ویژوال استدیو سربرگ Solution Explorer را انتخاب نمایید، سپس بر روی فایل P2Dlg.cpp دابل-کلیک کنید تا کدهای درون فایل را مشاهده نمایید.حالا به دنبال تابع `OnInitDialog()` بگردید(عکس زیر).این تابع آماده سازی نام دارد و جزء اولین توابعی است که با اجرایی برنامه شما اجرا خواهد شد و معمولا از آن برای مقدار دهی اولیه در برنامه به متغییر ها و آماده سازی اولیه برنامه استفاده میکنند.



به آخر تابع رفته و کدهای زیر را بعد از عبارت TODO به تابع اضافه نمایید دهید.

```
m_strCB1.InsertString( 0, _T("ماشین حساب") );
m_strCB1.InsertString( 1, _T("دفترچه یادداشت") );
m_strCB1.InsertString( 2, _T("پاستور") );

m_strMessage=L"یک پیام بگذارید";

m_bShowMsg=TRUE;
m_bShowPgm=TRUE;
m_bEnableMsg=TRUE;
m_bEnablePgm=TRUE;

UpdateData(FALSE);
```

اگر قبلا با C یا C++ برنامه نوشته باشید، توجه کرده اید که روش مقدار دادن به متغیر m_strMessage بیشتر شبیه Visual Basic است تا C. کلاس CString کلاسیست که اجازه می دهد تا با رشته ها بسیار راحتتر و مانند زبان Visual Basic کار کنید.

- در سه خط اول به جعبه لیست باز شو سه عنوان اضافه می نماییم، چون از حروف فارسی که به صورت یونی کد است استفاده کرده ایم رشته ها را مستقیما در گیومه قرار ندادیم و به صورت _T("Text") نوشته ایم.
- در خط چهارم درون جعبه ویرایش متن یک پیام اولیه قرار داده ایم که به علت فارسی بودن از عبارت L"Text" به جای دو گیومه استفاده کرده ایم.
- خط پنجم تا هشتم هم کنترلهای جعبه چک را علامت دار می کند.
- مهمترین قسمت این کد خط آخر آن است، تابع UpdateData در واقع کلید کار با متغیر کنترل هاست. این تابع از یک طرف مقدار کنترل را گرفته و به متغیرهای وابسته می دهد و از طرف دیگر با استفاده از مقدار متغیرها کنترل ها را به روز در می آورد. جهت کار را آرگومان این تابع مشخص می کند. با آرگومان FALSE حالت کنترل ها با مقدار متغیرها به روز می شود و آرگومان TRUE باعث می شود تا حالت کنترل ها در متغیرهای وابسته نوشته شود.بعد از هر تغییر که در کنترل ها یا متغیرهای وابسته به آنها می دهیم باید این تابع را با آرگومان مناسب اجرا کنید تا تغییرات خواسته شده اعمال شوند.

بستن برنامه

اولین گام در هر برنامه ای آنست که مطمئن شویم می توان برنامه را بست. برای فعال کردن دکمه خروج از برنامه ابتدا بر روی آن راست کلیک کرده و از منوی به نمایش درآمده گزینه Add Event Handler... را انتخاب نمایید. در پنجره بعدی مقدار Message Type باید بر روی عبارت BN_CLICKED (که به صورت پیشفرض نیز روی همین گزینه قرار دارد) باشد تا تابعی که در حال ساخت آن هستیم با کلیک بر روی این دکمه فعال شود. با کلیک بر روی دکمه Add and Edit تابعی با نام OnBnClickedExit() تولید شده و به پنجره ای برای ویرایش کد آن راهنمایی می شوید. کنترل برنامه در صورت فشرده شدن دکمه خروج از برنامه به این تابع منتقل می شود، در نتیجه هر دستوری که در آن بنویسیم به محظ کلیک کردن آن اجرا میشود.دستور زیر که باعث بستن برنامه می شود را در این تابع بنویسید.

```
OnOK();
```

نمایش پیام کاربر

برای نمایش پیامی که کاربر در جعبه ادیت نوشته است باید تابعی برای دکمه نمایش پیام بسازیم که با کلیک بر روی آن تابع اجرا شده و پیام کاربر را نمایش دهد. طبق روشی که در بالا توضیح داده شده یک تابع برای کلیک کردن بر روی دکمه نمایش پیام ساخته و کد زیر را درآن بنویسید.

```
UpdateData(TRUE);
MessageBox(m_strMessage);
```

خط اول باعث می شود که مقدار درون کنترل جعبه ادیت به متغییر وابسته به آن یعنی m_strMessage منتقل خواهد شد، سپس دستور بعدی آن را در یک پنجره نمایش می دهد.

پاک کردن پیام کاربر

تابعی برای کلیک کردن بر روی دکمه پاک کردن پیام ایجاد نمایید و کد زیر را درآن بنویسید.

```
m_strMessage="";
UpdateData(FALSE);
```

پیام پیشفرظ

پس از ایجاد یک تابع برای دکمه پیام پیشفرظ کد زیر را درآن بنویسید.

```
m_strMessage=L"سلام";
UpdateData(FALSE);
```

اجرای برنامه های دیگر

اگر به یاد داشته باشید نام سه تا از برنامه های ویندوز را در لیست جعبه ترکیبی نوشته ایم و وقتی برنامه در حال اجرا باشد می توان با باز کردن این لیست یکی از آیتم ها را انتخاب کرد. تابع دکمه فرمان اجرای برنامه باید این نام را گرفته و برنامه انتخاب شده را اجرا کند. کد زیر را در تابع این دکمه بنویسید.

```
UpdateData(TRUE);

if(m_strProgToRun == (L"ماشین حساب")) WinExec("calc.exe",SW_SHOW);
if(m_strProgToRun == (L"دفترچه یادداشت")) WinExec("notepad.exe",SW_SHOW);
if(m_strProgToRun == (L"پاستور")) WinExec("sol.exe",SW_SHOW);
```

شرطهای موجود در صورت برابری متغیر m_strProgToRun با هر کدام از نوشته های درون لیست برنامه متناضر با آن را بوسیله دستور WinExec اجرا می نمایند.

غیر فعال یا فعال نمودن کنترلها

برای انجام این کار دو تابع برای کلیک کردن بر روی جعبه چکهای درون کنترل گروه با عنوان فعال بودن قسمتهای می سازیم و کدهای زیر را در دو تابع ساخته شده قرار می دهیم.

کد درون جعبه چک نمایش پیام:

```
UpdateData(TRUE);
if(m_bShowMsg == TRUE)
{
     GetDlgItem(IDC_MSG)->ShowWindow(TRUE);
     GetDlgItem(IDC_SHWMSG)->ShowWindow(TRUE);
     GetDlgItem(IDC_DFLTMSG)->ShowWindow(TRUE);
     GetDlgItem(IDC_CLRMSG)->ShowWindow(TRUE);
     GetDlgItem(IDC_STATICMSG)->ShowWindow(TRUE);
}
else
{
     GetDlgItem(IDC_MSG)->ShowWindow(FALSE);
     GetDlgItem(IDC_SHWMSG)->ShowWindow(FALSE);
     GetDlgItem(IDC_DFLTMSG)->ShowWindow(FALSE);
     GetDlgItem(IDC_CLRMSG)->ShowWindow(FALSE);
     GetDlgItem(IDC_STATICMSG)->ShowWindow(FALSE);
}
```

کد درون جعبه چک اجرای برنامه:

```
UpdateData(TRUE);
if(m_bShowPgm == TRUE)
{
     GetDlgItem(IDC_RUNPGM)->ShowWindow(TRUE);
     GetDlgItem(IDC_PROGTORUN)->ShowWindow(TRUE);
     GetDlgItem(IDC_STATICPGM)->ShowWindow(TRUE);
}
else
{
     GetDlgItem(IDC_RUNPGM)->ShowWindow(FALSE);
     GetDlgItem(IDC_PROGTORUN)->ShowWindow(FALSE);
     GetDlgItem(IDC_STATICPGM)->ShowWindow(FALSE);
}
```

مرعی یا نا مرعی کردن کنترلها

شبیه بالا عمل کرده و دو تابع برای کلیک کردن بر روی جعبه چکهای درون کنترل گروه با عنوان مرعی بودن قسمتهای می سازیم و کدهای زیر را در آن دو تابع قرار می دهیم.

کد درون جعبه چک نمایش پیام:

```
UpdateData(TRUE);
if(m_bEnableMsg == TRUE)
{
     GetDlgItem(IDC_MSG)->EnableWindow(TRUE);
     GetDlgItem(IDC_SHWMSG)->EnableWindow(TRUE);
     GetDlgItem(IDC_DFLTMSG)->EnableWindow(TRUE);
     GetDlgItem(IDC_CLRMSG)->EnableWindow(TRUE);
     GetDlgItem(IDC_STATICMSG)->EnableWindow(TRUE);
}
else
{
     GetDlgItem(IDC_MSG)->EnableWindow(FALSE);
     GetDlgItem(IDC_SHWMSG)->EnableWindow(FALSE);
     GetDlgItem(IDC_DFLTMSG)->EnableWindow(FALSE);
     GetDlgItem(IDC_CLRMSG)->EnableWindow(FALSE);
     GetDlgItem(IDC_STATICMSG)->EnableWindow(FALSE);
}
```

کد درون جعبه چک اجرای برنامه:

```
UpdateData(TRUE);
if(m_bEnablePgm == TRUE)
{
     GetDlgItem(IDC_RUNPGM)->EnableWindow(TRUE);
     GetDlgItem(IDC_PROGTORUN)->EnableWindow(TRUE);
     GetDlgItem(IDC_STATICPGM)->EnableWindow(TRUE);
}
else
{
     GetDlgItem(IDC_RUNPGM)->EnableWindow(FALSE);
     GetDlgItem(IDC_PROGTORUN)->EnableWindow(FALSE);
     GetDlgItem(IDC_STATICPGM)->EnableWindow(FALSE);
}
```

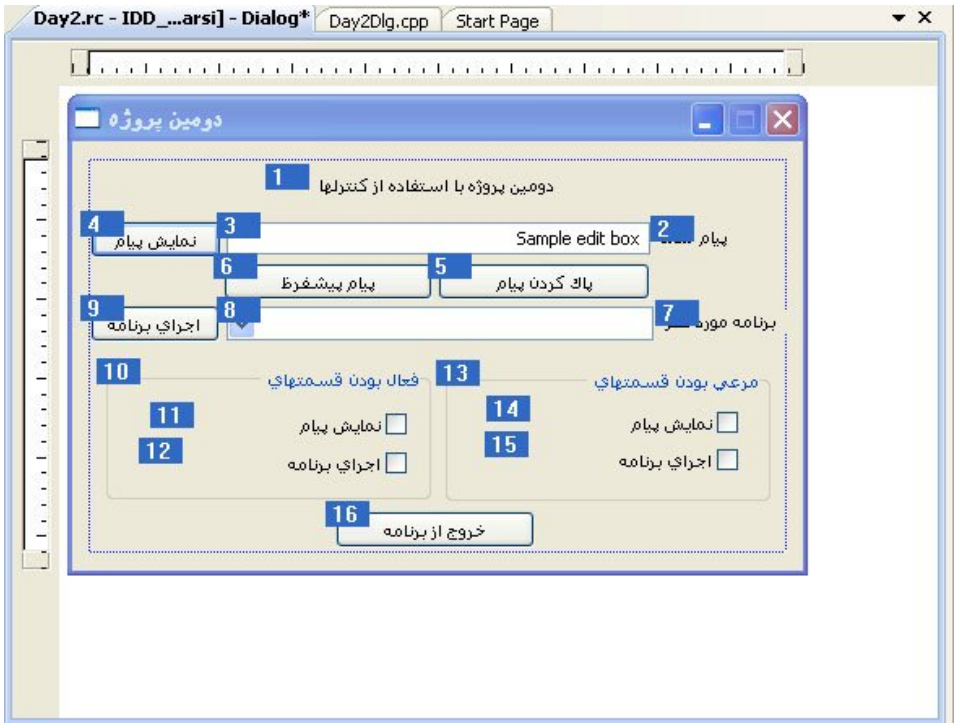
توضیح نهایی برنامه

در قسمت ابتدایی این توابع مقدار کنترل های روی پنجره در متغیرهای برنامه نوشته می شود. سپس مقدار متغییر های متناظر با جعبه چک ها مورد بررسی قرار می گیرد. اگر مقدار این متغییر ها TRUE باشد، کنترل ها باید فعال و یا مرعی شوند و اکر مقدار آنها FALSE باشد، کنترلها باید غیر فعال و یا نامرعی شوند. اگر برنامه را به درستی و بدون خطا نوشته باشید برنامه شما پس از کامپایل و اجرا به درستی کار خواهد کرد.

تعیین ترتیب حرکت بین کنترلها

بعد از قرار دادن کنترلها روی پنجره برنامه، باید مطمعن شویم که کاربر با زدن کلید **Tab** آنطور که شما می خواهید بین کنترلها حرکت خواهد کرد. برای تعیین این ترتیب حرکت که **tab order** نامیده میشود باید پس از انتخاب پنجره دیالوگ برنامه از منوی **Format** گزینه

Tab Order را انتخاب نمایید تا اعدادی در اطراف کنترلها نمایش داده شوند، که در واقع این اعداد ترتیب حرکت بین کنترلها را نشان می دهند. حالا شما با کلیک کردن بر روی کنترلها دوباره ترتیب حرکت بین انها را به صورت دلخاه مشخص کنید.پس از انجام کار دوباره از منوی Format گزینه Tab Order را انتخاب کنید تا از این حال خارج شوید.



آموزش مقدماتی ویژوال سی پلاس پلاس ۲۰۰۸

بهزاد جناب

فصل چهارم

استفاده از ماوس و کی بورد

اغلب پیش می آید که یک برنامه باید کارهایی را با ماوس انجام دهد، مثلا بسته به محل کلیک شدن و نحوه حرکت آن اقداماتی را انجام دهد. یا گاهی لازم است برنامه بداند که کاربر کدام دکمه را کلیک کرده یا در حین نگه داشتن دکمه ماوس چه کارهایی انجام داده است.
اتفاقاتی که روی کی برد هم می افتد می تواند مورد توجه خاص برنامه باشد. مثلا ، کابر چه کلیدی را زده ، چقدر آنرا نگه داشته، یا چه زمانی آنرا رها کرده است. امروز یاد خواهید گرفت که

- رویدادهای ماوس و به کارگیری آنها در برنامه
- چگونگی کشف رویدادهای ماوس
- رویدادهای کی بورد و نحوه تحریک آنها
- روش استفاده از رویدادهای کی بورد

آشنایی با رویدادهای ماوس

در فصل قبل دیدید که هر کنترل تعداد مشخص و محدودی رویداد دارد.تعداد رویدادهای ماوس هم بسیار محدود است و به کلیک و دو-کلیک محدود می شود. اما یک نگاه به ماوس نشان می دهد که ماوس بیش از اینها امکانات دارد.مثلا ، چگونه می توان با دکمه راست ماوس کار کرد و فهمید که آن چه زمانی فشرده شده است؟ یا مثلا چیزهایی را روی صفحه کامپیوتر جابجا کرد؟
مهمترین رویدادهای ماوس در جدول زیر نمایش داده شده است. به کمک این رویدادها می توانید هر کاری که با ماوس لازم باشد در برنامه تان انجام دهید.

پیام Messages	مفهوم
WM_LBUTTONDOWN	دکمه چپ ماوس فشرده شده است
WM_LBUTTONUP	دکمه چپ ماوس رها شده است
WM_LBUTTONDBLCLK	دکمه چپ ماوس دو-کلیک شده است
WM_RBUTTONDOWN	دکمه راست ماوس فشرده شده است
WM_RBUTTONUP	دکمه راست ماوس رها شده است
WM_RBUTTONDBLCLK	دکمه راست ماوس دو-کلیک شده است
WM_MOUSEMOVE	ماوس در فضای پنجره برنامه حرکت کرده است
WM_MOUSEWHEEL	چرخک ماوس چرخانده شده است

نقاشی با ماوس

امروز برای نمایش قابلیت های ماوس و طرز استفاده از آنها یک برنامه ساده نقاشی با ماوس خواهیم نوشت. این برنامه عمدتا از رویداد WM_MOUSEMOVE ، که حرکت ماوس را آشکار می کند، استفاده خواهد کرد.در این برنامه خواهید دید که چگونه می توان فهمید ماوس در کجای پنجره برنامه قرار دارد. نسبتا ساده بنظر می رسد، پس بیایید شروع کنیم.

۱. یک پروژه MFC Application به صورت Dialog Based ساخته و نام آنرا Mouse بگذارید.

۲. بعد از ایجاد پروژه، تمام کنترلهای روی آنرا حذف کنید تا صفحه دیالوگ برنامه ما بوم نقاشی شود. برای بدام انداختن رویدادهای ماوس و کی برد باید هیچ کنترلی روی پنجره برنامه نباشد ، در غیر اینصورت رویدادهای ماوس و کیبورد به کنترل دارای فوکوس(کنترلی که فعال است) خواهد رفت.

۳. نام کلاس دیالوگ مورد نظر که در اینجا CMouseDlg است را از Class View انتخاب کرده (با موس highlight کنید)

و در پنجره Properties آیکن Messages را برای مشاهده پیام ها انتخاب نمایید

سپس پیام WM_MOUSEMOVE را از لیست انتخاب و در پایان برای ساختن تابع مرد نظر برای این رویداد از Combo box کنار آن گزینه add را انتخاب نمایید تا به تابع مورد نظر در پنجره ادیتور جهت نوشتن کد هدایت شوید.

۴. در پنجره ادیتور تابع مرد نظر با عنوان OnMouseMove ایجاد می شود که باید کد زیر را در این تابع بنویسید.

```
if((nFlags & MK_LBUTTON) == MK_LBUTTON)
{
     CClientDC dc(this);
     dc.SetPixel(point.x , point.y ,RGB(0,0,0)));
}
```

اگر به خط اول تابع نگاه کنید خواهید دید که این تابع دو آرگومان ورودی دارد ، ورودی اول مجموعه ایست از چند پرچم که نشان می دهد کدام دکمه ماوس در حین حرکت آن فشرده شده است. بررسی این حالت در اولین خط کدی که نوشته ایم صورت می گیرد. یعنی کد زیر:

```
if((nFlags & MK_LBUTTON) == MK_LBUTTON)
```

نیمه دوم این دستور بررسی می کند که آیا دکمه چپ ماوس فشرده شده است یا خیر. نیمه اول دستورا **if** پرچمیست که بررسی این موضوع را به عهده دارد. اگر این دو نیمه منطبق باشند، دکمه چپ ماوس فشرده شده تلقی خواهد شد.
ورودی دوم تابع **OnMouseMove** ، مکان ماوس را بر می گرداند. این آرگومان مختصات مکان فعلی ماوس را در خود دارد.از این اطلاعات برای رسم نقطه روی پنجره دیالوگ استفاده می شود.
قبل از آنکه بتوانیم چیزی روی دیالوگ رسم کنیم، باید محتوای ابزار(**device context**) آنرا بدست آوریم.این کار با تعریف یک وهله از کلاس **CClientDC** انجام می شود. این کلاس علاوه بر محتوای ابزار ، تمام توابع لازم برای ترسیم را هم در خود دارد. محتوای ابزار در واقع مانند یک بوم است که برنامه می تواند روی آن نقاشی کند. تا زمانی که محتوای ابزار یک پنجره را نداشته باشید نمی توانید هیچ چیزی روی آن رسم کنید. تابعی که برای نقاشی از آن استفاده کرده ایم تابع **SetPixel** است. این تابع سه ورودی می گیرد که عبارتند از مختصات **X** و **Y** نقطه ای که باید رنگ آن تغییر یابد و رنگ مورد نظر برای آن نقطه. اگر برنامه را کامپایل و اجرا کنید خواهید دید که می توان با گرفتن دکمه چپ ماوس و حرکت دادن آن روی پنجره دیالوگ نقاشی کرد.

نکته: در ویندوز هر رنگ یک عدد است، این عدد مقدار رنگهای قرمز، سبز و آبی را تعیین میکند. تابع **RGB** که مخفف اول نامهای این سه رنگ است سه عدد جداگانه را گرفته و آنها را با هم ترکیب می کند تا عدد رنگ مناسب با مقادیر داده شده ساخته شود. هر کدام از این سه رنگ می تواند مقداری بین ۰ تا ۲۵۵ بگیرد.

استفاده از AND و OR باینری

اگر تازه وارد دنیای **C++** شده اید، باید با تفاوت انواع **AND** و **OR** آشنا شوید. دو نوع **AND** و **OR** وجود دارد، منطقی و باینری. **AND** و **OR** منطقی در دستورات شرطی مانند دستورات **if** به کار می روند چنانچه در فصل اول آن را توضیح دادیم. درحالیکه **AND** و **OR** باینری برای ترکیب اعداد باینری مورد استفاده قرار می گیرند.
برای **AND** باینری از علامت **&** استفاده می کنیم. یک علامت **&** برای **AND** باینری و دو علامت **&&** برای **AND** منطقی بکار می رود. رفتار **AND** منطقی شبیه عملکرد **AND** در زبان **Visual Basic** است. یک عبارت **AND** منطقی زمانی درست است که هر دو قسمت آن ارزش درست داشته باشند.**AND** باینری فقط باعث تغییر حالت بیت ها خواهد شد و ارزش درست و نادرست در آن وقطی دو بیت با یکدیگر **AND** می شوند، نتیجه فقط زمانی ۱ خواهد شد که هر دو بیت ۱ باشند و در غیر اینصورت نتیجه صفر خواهد شد. برای درک بهتر مطلب به یک مثال توجه کنید. دو عدد باینری ذیل را که با هم **AND** باینری کرده ایم مشاهده کنید.

عدد اول	01011001
عدد دوم	00101001
باینری AND	00001001

توجه کنید که در عدد حاصله فقط بیت هایی 1 هستند که هر دو بیت متناظر با آنها در اعداد اولیه 1 باشند و تمام بیت های دیگر حتی آهایی که یکی از آنها 1 است 0 شده اند.

OR با علامت | نمایش داده می شود و مانند AND، یک علامت | نشان دهنده OR باینری و دو علامت || نشاندهنده OR منطقی است. OR منطقی هم در عبارات شرطی کاربرد دارد و عملکرد آن بسیار شبیه OR در Visual Basic است. از OR باینری برای ترکیب بیت ها استفاده می شود. حاصل OR شدن دو بیت فقط زمانی 0 است که هر دو قسمت آن مقدار 0 داشته باشند و در غیر این صورت مقدار 1 خواهد داشت. باز هم به یک مثال توجه کنید.ملاحضه می کنید که فقط بیت هایی 0 شده اند که هر دو جزء آنها 0 بوده است.

عدد اول	01011001
عدد دوم	00101001
باینری OR	01111001

پرچمهای باینری

AND و OR باینری در VC++ عمداتا برای خواندن و یا ست کردن پرچمها به کار می روند. پرچم(Flag) مقداریست که هر بیت آن وضعیت یا حالتی را نشان می دهند و برنامه نویس می تواند از این پرچمه برای تشخیص اوضاع استفاده کند یا آنرا تحت کنترل خود در آورد. از آنجایی که هر بیت می تواند یک حالت و موقییت را نشان دهد، یک پرچم می تواند چندین حالت را در خود حمل کند. برای ترکیب حاتها در یک پرچم معمولا از OR باینری استفاده می شود. مثلا اگر دو پرچم داشته باشیم که هر کدام حالاتی را نشان می دهد، می توانیم آنها را با OR کردن با هم ترکیب کنیم.

پرچم اول	00001000
پرچم دوم	00100000
ترکیب پرچم ها	00101000

بدین ترتیب می توان دو پرچم را در یکدیگر ادغام و در جا صرفه جویی کرد. در حقیقت اکثر کنترلهای ویندوز چنین می کنند و هر پرچم خواص متعددی را در خود نگه میدارد. خواصی که حالت روشن یا خاموش (بله یا خیر) دارند بسیار مستعد این وضعیت هستند.از طرفی دیگر، اکر بخواهید مقدار یک پرچم خواص را در ترکیب از چند پرچم بخوانید، باید از AND استفاده کنید.

ترکیب چند پرچم	00101000
پرچم مورد نظر	00001000
حاصل	00001000

با این روش می توان یک پرچم خواص را از میان ترکیب چندین پرچم بیرون کشید(و به اصطلاح فیلتر کرد). اگر عدد حاصل معادل آنچه ما انتظار داریم باشد پرچم مورد نظر در آن ترکیب وجود. اگر حاصل صفر شود، پرچم مورد نظر رد آن ترکیب وجود ندارد، در نتیجه می توان این عدد را در یک دستور if تست کرد. بدین ترتیب دستور if کد زیر را می توان ساده تر کرد.

```
if (nFlags & MK_LBUTTON)
```

اگر می خواهید عدم وجود یک پرچم را تست کنید می توانید از دستوری مانند زیر استفاده کنید.

```
if ( !(nFlags & MK_LBUTTON) )
```

استفاده از هر یک از این روشها به شرایط بستگی دارد و این برنامه نویس است که باید نوع مناسب را انتخاب کند.

اصلاح برنامه نقاشی

اگر برنامه را اجرا کرده باشید احتمالا متوجه یک اشکال کوچک در آن شده اید، برای رسم یک خط ممتد باید ماوس را بسیار آهسته حرکت دهید اما چرا این اشکال در دیگر برنامه های نقاشی وجود ندارد؟ چون آنها بین دو نقطه خط می کشند و نقطه ها (پیکسل ها) را ست نمی کنند، در واقع اکثر این قبیل برنامه ها چنین عمل می کنند. این قبیل برنامه ها در حین حرکت ماوس مکان آنرا چک می کنند و چون نمی توانند تمام مسیر آنرا بررسی کنند مجبورند بین خود فرضیاتی بکنند و سپس بین نقاط بدست آمده خط رسم کنند.

برای اینکه برنامه ما هم مانند این قبیل برنامه ها عمل کند، چه باید کرد؟ ابتدا باید مکان قبلی ماوس را بطریقی ذخیره کنیم. برای این منظور به دو متغییر جدید برای ذخیره کردن مختصات X و Y مکان قبلی ماوس نیاز داریم. برای تعریف این متغیر ها باید بر روی پنجره دیالوگ برنامه راست کلیک کنید و از منوی باز شده گزینه **Add Variable** را انتخاب نمایید، سپس در پنجره باز شده دو متغییر از نوع **int** و با دسترسی **private** با نامهای **m_iPrevX** و **m_iPrevY** ایجاد نمایید. پس از افضودن این دو متغییر تابع **OnMouseMove** را به شکل زیر تغییر دهید.

```
if((nFlags & MK_LBUTTON) == MK_LBUTTON)
{
	CClientDC dc(this);

	dc.MoveTo(m_iPrevX , m_iPrevY);
	dc.LineTo(point.x , point.y);

	m_iPrevX = point.x;
	m_iPrevY = point.y;
}
```

به کد رسم خط بین دو نقطه توجه کنید، ملاحضه می کنید که ابتدا باید به مکان قبلی ماوس رفته و سپس خطی به مکان فعلی آن رسم کنیم. گام امل مهم است چون بدون آن ویندوز نمی تواند بداند که خط را از کجا باید شروع کند. اگر برنامه را اجرا کنید، خواهید دید که عملکرد آن کمی بهترشده است. ولی در ضمن رفتار عجیبی هم از خود نشان می دهد، هر بار که دکمه چپ ماوس را میگیرید تا چیزی رسم کنید، برنامه خطی از انتهای ترسیم قبلی به شروع خط جدید رسم می کند!

آخرین اصلاحات

برنامه ما زمانی شروع به رسم می کند که دکمه چپ ماوس را فشار دهیم. با ست کردن متغییر های مکان قبلی ماوس در لحضه کلیک ماوس، می توانید رفتار برنامه را اصلاح کنیم. برای انجام اصلاحات شبیه روش ساخت تابع **OnMouseMove** عمل می کنیم، نام کلاس دیالوگ برنامه یعنی **CMouseDlg** است را از **Class View** انتخاب کرده (با موس **highlight** کنید) و در پنجره **Properties** آیکن **Messages** را برای مشاهده پیام ها انتخاب نمایید. سپس پیام **WM_LBUTTONDOWN** را از لیست انتخاب و در پایان برای ساختن تابع از **Combo box** کنار آن گزینه **OnLButtonDown** **<add>** را انتخاب نمایید.

سپس در پنجره ادیتور تابع مرد نظر با عنوان **OnLButtonDown** ایجاد شده است که باید کد زیر را در این تابع بنویسید.

```
m_iPrevX = point.x;
m_iPrevY = point.y;
```

حال اگر برنامه را کامپایل و اجرا کنید خواهید دید که برنامه بسیار بهتر عمل می کند.

بدام انداختن رویدادهای کی بورد

خواندن رویدادهای کی بورد بسیار شبیه رویدادهای ماوس است. در مورد کی بورد هم رویدادهای برای فشردن و رها کردن کلیدهای کی بورد وجود دارد. رویدادهای کی بورد را در جدول زیر مشاهده نمایید.

پیام	مفهوم
WM_KEYDOWN	کلیدی زده شده است
WM_KEYUP	کلیدی رها شده است

کی بورد دارای پیام های اندکیست، اما باید بدانید که با کی بورد هم می توان کارهای زیادی انجام داد. علاوه بر کنترل ها حتی خود دیالوگ هم می تواند پیام کلیدهای کی بورد را بگیرد، البته مشروط به اینکه هیچ کنترلی فوکوس را در اختیار نداشته باشد، در غیر این صورت تمام پیام های کی بورد به کنترل دارای فوکوس خواهند رسید(به همین دلیل قبلا تمامی کنترلهای برنامه را حذف کرده ایم).

تغییر دادن کرسر

برای آشنایی بیشتر با پیام های کی بورد، در این قسمت سعی می کنیم با زدن کلیدی خواص کرسر برنامه نقاشی را تغییر دهیم. مثلا کاری میکنیم که با زدن کلید A کرسر پیشفرض انتخاب شود، با زدن کلید B کرسر به I تبدیل شود و با زدن کلید C کرسر تبدیل به ساعت شنی تغییر حالت دهد. برای این منظور طبق روشهای بالا تابعی برای پیام WM_KEYDOWN سخته و کد زیر را در آن تابع یعنی OnKeyDown بنویسید.

```
char IsChar;
HCURSOR IhCursor;
IsChar = char(nChar);

if(IsChar == 'a' || IsChar == 'A')
{
     IhCursor = AfxGetApp()->LoadStandardCursor(IDC_ARROW);
     SetCursor(IhCursor);
}

if(IsChar == 'b' || IsChar == 'B')
{
     IhCursor = AfxGetApp()->LoadStandardCursor(IDC_IBEAM);
     SetCursor(IhCursor);
}

if(IsChar == 'c' || IsChar == 'C')
{
     IhCursor = AfxGetApp()->LoadStandardCursor(IDC_WAIT);
     SetCursor(IhCursor);
}

if(IsChar == 'x' || IsChar == 'X')
{
     IhCursor = AfxGetApp()->LoadStandardCursor(IDC_ARROW);
     SetCursor(IhCursor);
     OnOK();
}
```

در خط اول تعریف تابع مشاهده می کنید که تابع OnKeyDown دارای سه ورودی است. ورودی اول کلید زده شده است. این ورودی در واقع کد کلید زده شده را برمی گرداند و قبل از هر کاری باید آنرا به یک کارکتر تبدیل کرد. بعد از این تبدیل است که می توان روی آن مقایسه انجام داد. دومین ورودی تابع OnKeyDown تعداد دفعات زده شدن کلید است. معمولا وقتی کلیدی زده و بلافاصله رها می شود این ورودی یک خواهد بود، اما اگر کلیدی را نگه دارید این ورودی به سرعت افزایش خواهد یافت و با رها شدن کلید تعداد تکرارها در این ورودی به ویندوز گزارش خواهد شد. ورودی سوم تابه OnKeyDown پرچمیست که نشان می دهد که آیا کلید Alt همزمان با کلید دیگر زده شده یا خیر، این پرچم نمی تواند حالت کلیدهای Ctrl یا Shift را گزارش کند. در دستورهای if مشخص می شود که کدام کلید زده شده است و چون روشن بودن کلید Caps Lock مشخص نیست هردو کارکتر کوچک و بزرگ را به وصیله OR منطقی تست می نماییم.

پس از مشخص شدن اینکه کدام کلید زده شده است نوبت به تغییر دادن کرسر می رسد که فرایندی دو مرحله ای است. اولین مرحله عبارت است از بار کردن کرسر در حافضه، این کار با تابع LoadStandardCursor انجام می شود. این تابع بعد از بار کردن کرسر استاندارد ویندوز شماره شناسائی آنرا برمی گرداند. بعد از بار شدن کرسر در حافظه، شماره شناسایی آن به تابع SetCursor داده می شود تا شکل کرسر عوض شود. اگر برنامه را کامپایل و اجرا کنید، خواهید دید که با زدن کلیدهای مزبور می توانید شکل کرسر را تغییر دهید. اما به محض شروع رسم، کرسر دوباره به حالت اولیه خود باز می گردد. در قسمت بعدی خواهید دید که چگونه می توان این مشکل را برطرف کرد.

ثابت کردن شکل کرسر

مشکل برنامه نقاشی آنست که با هر حرکت ماوس کرسر آن از نو روی صفحه رسم می شود. باید راهی برای متوقف کردن این وضعیت وجود داشته باشد. هر بار که کرسر به هر علتی مانند حرکت کردن، جابجا شدم پنجره ها و ... به تر سیم مجدد نیاز داشته باشد، یک پیام WM_SETCURSOR به برنامه شما فرستاده می شود. اگر رفتار پیش فرض این رویداد را تغییر دهیم، شکل کرسر ثابت باقی خواهد ماندالبته تا زمانی که مجدد آنرا تغییر دهیم.

برای انجام این کار به روش قبلی متغییر جدیدی در کلاس CmouseDlg از نوع BOOL و با نام m_bCursor و به صورت Private تعریف کنید. سپس باید مقدار آنرا در تابع OnInitDialog برابر با FALSE قرار دهید، یعنی کد زیر را به تابع اضافه نمایید.

```
m_bCursor = FALSE;
```

و سپس، هنگام تغییر دادن شکل کرسر، در تابع OnKeyDown پرچم m_bCursor را به TRUE ست کنید. یعنی این تابع را به شکل زیر اصلاح نمایید.

```
 char IsChar;
HCURSOR IhCursor;
IsChar = char(nChar);
if(IsChar == 'a' || IsChar == 'A')
{
     IhCursor = AfxGetApp()->LoadStandardCursor(IDC_ARROW);
     SetCursor(IhCursor);
}

if(IsChar == 'b' || IsChar == 'B')
{
     IhCursor = AfxGetApp()->LoadStandardCursor(IDC_IBEAM);
     SetCursor(IhCursor);
}
```

```
if(IsChar == 'c' || IsChar == 'C')
{
     IhCursor = AfxGetApp()->LoadStandardCursor(IDC_WAIT);
     SetCursor(IhCursor);
}

if(IsChar == 'x' || IsChar == 'X')
{
     IhCursor = AfxGetApp()->LoadStandardCursor(IDC_ARROW);
     m_bCursor = TRUE;
     SetCursor(IhCursor);
     OnOK();
}
else
{
     m_bCursor = TRUE;
     SetCursor(IhCursor);
}
```

در آخر برای پیام WM_SETCURSOR پنجره دیالوگ برنامه یک تابع بسازید و تابع را به شکل زیر اصلاح نمایید.

```
BOOL CMouseDlg::OnSetCursor(CWnd* pWnd, UINT nHitTest, UINT message)
{
     // TODO: Add your message handler code here and/or call default
     if(m_bCursor)
          return TRUE;
     else
          return CDialog::OnSetCursor(pWnd, nHitTest, message);
}
```

تابع OnSetCursor یا همیشه TRUE بر میگرداند یا تابعی را که خود از آن مشتق شده اجرا می کند. تابعی که OnSetCursor از آن مشتق شده کرسر ماوس را به حالت اول برمیگرداند. به همین دلیل است که مقدار متغییر m_bCursor قبل از آنکه کابر کلیدی را بزند باید FALSE باشد تا تابع OnSetCursor رفتار طبیعی خود را داشته باشد. اما وقتی کابر شکل کرسر را با زدن کلید مربوطه تغییر داد، ست کردن پرچم مزبور باعث می شود تا کرسر ثابت بماند و در واقع جلوی رفتار طبیعی OnSetCursor گرفته می شود. بدین ترتیب کابر می تواند شکل کرسر را تغییر داده و با آن نقاشی کند.

نکته: برای استفاده از کرسر های غیر استاندارد ویندوزۀ یا آنهایی که خودتان ساخته اید، باید از تابع LoadCursor استفاده کنید. مثلا اگر کرسری را با استفاده از ادیتور منبع Visual C++ ساخته و نام آنرا IDC_MYCURSOR گذاشته اید، می توانید با دستور ذیل از آن استفاده کنید.

```
IhCursor = AfxGetApp()->LoadCursorW(IDC_MYCURSOR);
```