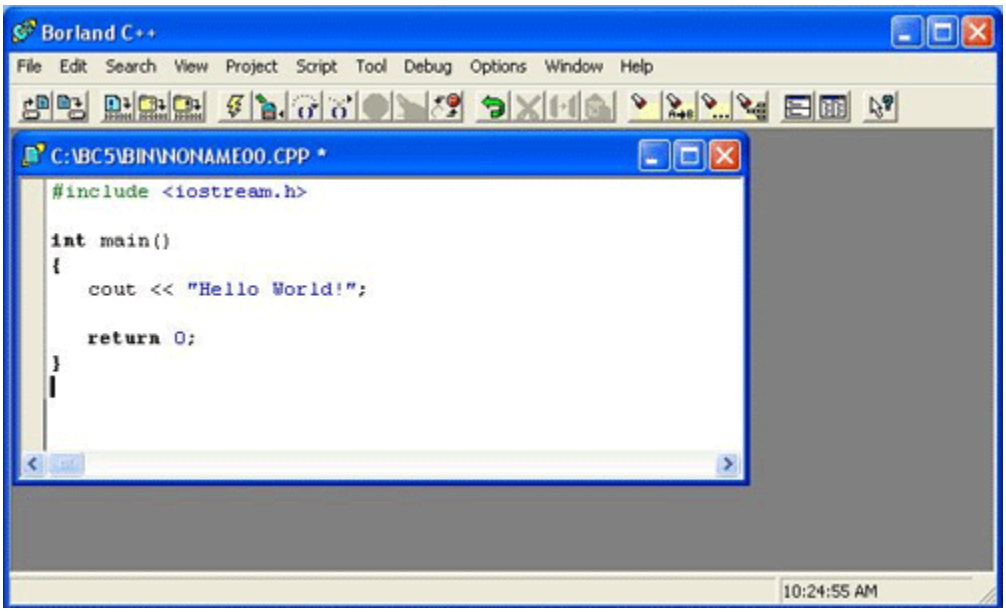
شاید بهترین راه برای یادگیری یک زبان برنامه نویسی، نوشتن یک برنامه ابتدایی و ساده در محیط یک نرم افزار برای آشنایی با ساختار آن زبان باشد.

در آغاز عکسی از برنامه Borland C++ را نشان می‌دهم تا با محیط نرم افزار C++ آشنا شوید. هرچند نرم افزارهای دیگه هم در این خصوص وجود دارند مانند Visual C++ یا Turbo C++ .



برنامه ای را که در بالا می بینید سورس کد اولین برنامه ما و همچنین ابتدایی ترین ساختار برنامه C++ می باشد. قبل از هر چیز به بررسی سطحی و آشنایی اولیه با خطوط برنامه Hello world! می‌پردازیم:

```
// my first program in c++
#include <iostream.h>
int main()
{
  cout << "Hello world" ;
  return 0 ;
}
```

```
// my first program in c++
```

این یک خط Comment یا توضیحات می‌باشد. در برنامه نویسی، توضیحات خطوطی هستند که کامپایل نمی‌شوند و فقط برای خوانایی برنامه بکار برده می‌شوند. در برنامه های بزرگتر و با زیاد شدن خطوط برنامه، توضیحات به فهم راحتتر برنامه برای برنامه نویس و دیگر کسانی که کد را مورد بررسی قرار می‌دهند کمک شایانی می‌نماید.

برای بوجود آوردن توضیحات در برنامه C++از دو الگو می‌توان استفاده نمود:

- اگر بخواهیم توضیحات را در یک خط قرار دهیم از علامت // قبل از توضیحات استفاده می کنیم. در مثال قبل ما از این الگو بهره بردیم.
- در صورت زیاد بودن توضیحات واشغال چند سطر از برنامه توسط آن از علامت /* */ استفاده کرده و توضیحات را در بین آن قرار می دهیم.

```
/* my
first program
in c++ */
```

در C++ خطوطی که با علامت # شروع شده و در بالای برنامه قرار می گیرند خطوط فرمان به پردازشگر می باشند که در اصطلاح فایل سرآیند نامیده می شوند.

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

از فایلهای سرآیند بعنوان کتابخانه های C++یاد می کنند که از قبل نوشته شده اند و ما برای استفاده از برخی از توابع و روالها از آنها استفاده می کنیم . کامپایلر فقط کلمات کلیدی را می شناسد و همانطور که گفته شد برای استفاده از یک سری دستورات و توابع مانند دستورات ورودی و خروجی و ... باید از این فایلهای سرآیند استفاده نماییم و اگر استفاده نکنیم امکان برنامه نویسی بوجود نخواهد آمد که به تفصیل در آینده در مورد این فایلها و مورد استفادشان صحبت خواهیم کرد . این نکته را هم خاطر نشان می کنم که پسوند این فایلها h می باشد. در انتهای این فصل هم توضیحات تکمیلتری در این مورد آورده شده است.

```
int main()
```

این خط تعریفی برای تابع اصلی برنامه می باشد. در واقع برنامه با کامپایل از این نقطه شروع و پردازش می شود. هر برنامه C++باید دارای تابع main() باشد. در این تابع است که بلوکها و خطوط برنامه نوشته میشود. باید گفته شود هرچند خطوطی بالاتر از تابع اصلی نوشته شده اما بخاطر داشته باشید که برنامه از این نقطه شروع خواهد شد. عبارت int به معنی integer یا اعداد یک کلمه کلیدی می باشد و برای تعریف متغیرهای از نوع صحیح بکار می رود که در ادامه آموزش برنامه نویسی بطور مفصل مورد بررسی قرار می گیرد Main .نامی برای تابع اصلی است که تغییر نمی کند و () علامتی است که در ادامه هر تابعی قرار می گیرد که توابع نیز مفصلا در ادامه مورد بحث قرار می گیرند و اطلاعات فوق جنبه آشنایی با آنها را دارد.

```
{
```

آکولاد باز در واقع شروع یک بلوک از دستورات را تعریف می کند که در این برنامه بدنه تابع اصلی (main) را در بر می گیرد و با علامت

```
}
```

یا آکولاد بسته پایان بلوک را تعیین می کنیم.

cout << "Hello world" ;

تابع cout << در تابع سرآیند iostream در زبان C++ قرار دارد و موجب ارسال اطلاعات به خروجی و چاپ و نمایش آن بر روی مانیتور کاربر می شود. با نوشتن این دستور عبارت Hello world! برروی صفحه نمایش پس از کامپایل بدون خطای برنامه به نمایش در می آید. از علامت " " برای متغیرهای رشته ای که در اصطلاح string گفته می شوند استفاده می شود. از علامت ; نیز در انتهای هر دستور در C++ استفاده می گردد تا بوسیله آن پایان آن دستور را به پردازشگر اعلام نمود.

return 0 ;

این عبارت مقدرا صفر را به تابع در برگیرنده خود که در این مثال تابع اصلی است برمی گرداند که این مورد در مبحث توابع و انواع بازگشتی آن توضیح داده خواهد شد.

namespace

باید بگم که فضاهای نام هم در ساختار C++ نقش اساسی دارند. این فضاها مجموعه دیگری از کتابخانه های C++می باشند که در استفاده از بعضی عناصر مانند رشته ها و ... کمک شایانی به کاربران می کنند.

فضا های نام با استفاده از الگوی زیر قابل استفاده اند و بعد از فایلهای کتابخانه ای include در برنامه قرار می گیرند که در بخش انواع یک نمونه از اونا رو استفاده مکنیم.

```
// using namespace std
#include <iostream>
using namespace std;
int main()
{
  string st = "Hello world!" ;
  cout << st ;
  return 0 ;
}
```

کد بالا را میشود به شکل دیگری هم نوشت که در اینصورت عبارت <iostream> به <iostream.h> تغییر می کنه و using حذف شده و بجاش از عبارت <string> استفاده می کنیم به صورت زیر:

```
// using namespace std
#include <iostream.h>
#include <string>
int main()
{
  string st = "Hello world!" ;
  cout << st ;
  return 0 ;
}
```

⭐ ذکر این مطلب لازم است که متاسفانه فایل سرآیند string در محیط Borland C++ کار نمی کند.

برخی از ویژگیهای زبان : C++

- انعطاف پذیری و غنای بالا
- زبان برنامه نویسی سیستم است و با آن می توان برنامه های سیستمی را نوشت، بدین معنی که مستقیما می تواند با سخت افزار و نرم افزار ارتباط برقرار نماید.
- زبان شی گراست
- Case sevsitive است ، یعنی نسبت به کوچکی و بزرگی حروف حساس بوده وبین این دو تمایز قائل است. توصیه می شود که برنامه ها را با حروف کوچک بنویسید .While برابر نیست با WHILE

برخی از ویژگیهای دستورات C++

- هر دستور باید به ; ختم شود.
- حداکثر طول یک دستور،۲۵۵ کاراکتر است.
- هر دستور می تواند در یک و یا چند سطر نوشته شود.
- در هر سطر می توان چندین دستور را نوشت.(این کار توصیه نمی شود.)
- توضیحات می توانند بین /* و */ در چندین سطر و یا بعد از // و در فقط یک سطر نوشته شوند.

کلمات کلیدی در C++

auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
const	float	short	unsigned
continue	for	signed	void
default	goto	sizeof	volatile
do	if	static	while

این کلمات، کلمات کلیدی هستند و کامپایلر فقط این کلمات را می شناسد و هرآنچه غیر از کلمات کلیدی در برنامه C++نوشته شود باید برای کامپایلر درست تعریف شود. در این بین توابعی برای گرفتن اطلاعات و یا چاپ اطلاعات و چیزهای دیگری وجود دارد که برای استفاده از آنها باید از فایل سرآیند مربوطه استفاده نماییم که از قبل نوشته شده اند مثل توابع cout ,cin که توابع ورودی خروجی هستند که در فایل سرآیند iostream وجودارند و همچنین تابع getch که برای زدن یک کلید از صفحه کلید است و در فایل سرآیند conio قرار دارد.

انواع داده ها در : C++

در C++شش نوع داده وجود دارد. منظور از داده، متغیری است که در قالب متن یا عدد در طول برنامه مورد استفاده قرار می گیرد.

داده های موجود در C++عبارتند از:

char, int, float, double, void, bool ,string

نوع char برای ذخیره داده های کاراکتری مانند 'W' , 'z' , 'a' : بکار می رود.

از نوع int برای ذخیره اعداد صحیح مانند ۱۲۸، ٥، ٤٥٠٨ استفاده می شود.

نوع float برای ذخیره اعداد اعشاری مثل ۱۲,٥، ۷۸۰٥,۱۱ بکار می رود.

نوع double برای اعداد اعشاری بزرگتر از float استفاده می شود.

از boolean برای ذخیره مقادیر منطقی استفاده می شود (درستی یا نادرستی).

نوع void هیچ مقداری را نمی گیرد

نوع دیگری از داده وجود دارد که برای استفاده از رشته ها مورد استفاده قرار میگیرد که string گفته میشود اما در برخی از نسخه های کامپایلر زبان برنامه نویسی C++پشتیبانی نمی شود، لذا مجبور به استفاده از آرایه ای از کاراکترها برای این منظور خواهیم بود.

متغیرها

در طول برنامه نویسی، کاربران با متون و اعداد زیادی کار می کنند، به همین دلیل آنها را در متغیرها ذخیره می کنند. در واقع متغیر ها نامهایی برای کلمات ذخیره شده در حافظه سیستم هستند.

برای استفاده از یک متغیر ابتدا باید آن را در برنامه تعریف نماییم که روش تعریف متغیر بصورت زیر است:

نوع متغیر نام متغیر ;

```
int  count ;
```

در قطعه کد بالا می بینیم که هر متغیر باید دارای نام منحصر بفردی باشد که برای نامگذاری متغیر ها باید توجه داشته باشیم که:

- برای نامگذاری متغیرها از ترکیبی از حروف a تا z یا A تا Z ، ارقام و خط ربط (_) استفاده می شود.
- اولین کاراکتر نام نباید از ارقام باشد.
- نام می تواند هر طولی داشته باشد اما فقط ۳۱ کاراکتر ابتدایی استفاده می شوند.

```
اسامی مجاز          اسامی غیر مجاز
-------------------------------------
count3              3count
count               .count
co_unt              co.unt
```

مقدار دادن به متغیر ها

بعد از تعریف یک متغیر باید مقداری را به آن نسبت دهیم که به یکی از چهار روش زیر می توان اینکار را انجام داد:

- هنگام تعریف متغیر

```
int  x = 4;     // initial value = 4
char  char1 = 'a' , char2 = char3 = 'y';    // initial values = 'a' and 'y'
bool  b = true;     // initial value = true
```

- بعد از تعریف و با عمل انتساب.(=)

```
int  y;
y = 12;     // initial value = 12
```

- با استفاده از قالب سازنده.

```
float float_1 (2);     // initial value = 2
```

- دستورات ورودی که در فصل مربوط به ورودی / خروجی گفته خواهد شد.

```
float a, b;
cin >> a >> b;
```

ثوابت و عملگرها در C++ - (Constants)

ثوابت مقادیری در برنامه هستند که مقدارشون در طول برنامه قابل تغییر نیست و اگر که بخوایم مقدار ثوابت رو تغییر بدیم با خطایی از طرف کامپایلر مواجه می شیم.

برای تعریف ثوابت در c++ دو الگو وجود دارد:

- 1.با استفاده از دستور #define

```
#define  <name>  <value>
// For example :
#include <iostream>
#define  P  3.14
int main()
{
   cout >> P;
   return  0 ;
}
```

```
3.14
```

به محل استفاده از این دستور دقت کنید که در کجای برنامه مورد استفاده قرار گرفته است (بعد از فایلهای سرآیند).

نکته ای که باید در اینجا توجه نمود و در مثال بالا هم مشخص بود این است که در پایان دستور #define از ; (سمی کالن) استفاده نمی کنیم.

- 2.با استفاده از کلمه کلیدی: const

```
const  <مقدار> = <نام ثابت>  <نوع داده>;
// For example :
#include <iostream>
int main()
{
   const  float  P = 3.14 ;
   cout >> P;
   return  0 ;
}
```

```
3.14
```

می بینیم که محل این دستور درون خود تابع main هست اما دستور #define در بیرون از تابع main و در بالای برنامه.

عملگر ها(Operators)

برای انجام عملیات بر روی داده ها از عملگرها استفاده می کنیم. عملگرها نمادهایی هستند که عملیاتی مانند جمع،ضرب، کوچکتری و از این قبیل را روی داده ها انجام می دهند که عبارتند از:

- انتساب(=) (Assignment)

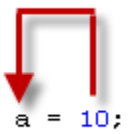
از این عملگر برای نسبت دادن یک مقدار به یک داده استفاده می شود.

```
#include <iostream>
int main()
{
   int a, b ;         // a:?,  b:?
   a = 10 ;           // a:10, b:?
   b = 4 ;            // a:10, b:4
   a = b ;            // a:4,  b:4
   b = 7 ;            // a:4,  b:7
   cout >> "a:" ;
   cout >> a ;
   cout>> "b:" ;
   cout >> b ;
   return  0 ;
}
```

```
a:4  b:7
```

نحوه عملکرد این عملگرد به این شکله که مقدار سمت راست تساوی را در سمت چپ قرار میدهد.

.



- عملگرهای محاسباتی(% , / , * , - , +) (Arithmetic Operators)

پنج عملگر محاسباتی موجود در C++عبارتند از:

جمع	+
تفریق	-
ضرب	*
تقسیم	/
باقیمانده تقسیم	%

با ٤ عملگرد اول آشنا هستید اما آخرین که خوانده می شود (مد "با واو تلفظ کنید") عملوند سمت چپ را بر عملوند سمت راست تقسیم کرده و باقیمانده آنرا بدست می آورد.

```
#include <iostream>
int main()
{
  int  a = 11;
  int  b = 3;
  int c = a % b;
  cout >> "c:" ;
  cout >> c ;
  return  0 ;
}
```

```
c: 2
```

- عملگرهای ترکیبی(/= , *= , -= , +=) (Compund Operators)

عبارت	برابر است با
a += b	a=a+b
a -= b	a=a-b
a *= b+1	a=a*(b+1)
a /= b	a=a/b

در واقع جواب این نوع از عملگرها برابر حاصل عمل عملگر، بر خود عبارت سمت چپ و عبارت سمت راست تساوی است . علت اینگونه نوشتار هم مختصرنویسی است.

عملگرهای ترکیبی دیگری نیز وجود دارند که در ادامه در موردشان بحث می کنیم مثل <= : و>=

```
#include <iostream>
int main()
{
  int  a ,b = 3;     // a=?, b=3
  a = b;            // a=3, b=3
  a += 2;           // a=a+2=3+2=5
  cout >> a;
  return  0 ;
}
```

```
5
```

- عملگرهای افزایش کاهش (-- , ++) (Increase , Decrease)

این عملگرها یک واحد به عملوند خود اضافه می کنند و عمل اونها به اینکه در سمت چپ یا راست عملوند خود قرار بگیرند متفاوت است.

```
#include <iostream>
int main()
{
  int  a = 2, b = 3;        // a=2, b=3
  a += b++;                 // a+=(b+1) ---> a=a+(b+1) ---> a=2+4=6
  cout >> "a:";
  cout >> a;
  return  0;
}
```

```
a:6
```

اگر عملگر سمت راست یا چپ عملوند خود باشه در هر دو صورت یک واحد به عملوند اضافه می شود . اما تفاوت این دوحالت در عبارات محاسباتی خود را نشان می دهد . عبارات محاسباتی ترکیبی از متغیرها، ثوابت و عملگرها هستند مثل ٤*٥-١٠/٥ و x/y٦-

```
int  A , B = 3;       // A=?, B=3
A = ++B;              // A=(++B) ---> A=(B+1) , B=B+1 ---> A=4, B=4
```

```
A=4   ,B=4
```

در مثال بالا چون افزایش قبل B قرار دارد ابتدا یک واحد به B اضافه شده، پس در همینجا B می شود ٤ و در پایان مقدار فوق در A قرار می گیرد.

```
int  A , B = 3;       // A=?, B=3
A = B++;              // A=(B++) ---> A=B, B=B+1 ---> A=3, B=4
```

```
A=3   ,B=4
```

اما در مثال بالا چون افزایش بعد از B قرار دارد اول مقدار B که ۳ هست در A ریخته میشود و بعد یک واحد به B اضافه میشود.

- عملگرهای رابطه ای و تساوی(= = , =! , > , < , =< , =>) (Relational and equality operators)

از این نوع عملگرها برای مقایسه دو عبارت استفاده میشود که کاربرد اونها بیشتر در عبارات شرطی است که بعدا در موردشون بحث می کنیم . فعلا اینو بدونید که این عملگرها در صورت درست بودن مقایسه، مقدار درستی و در غیر این صورت مقدار نادرستی را برمی گردانند.

```
int  a = 10 , b = 7;     //a=10, b=7
(a == b) ;               //a=10 and not equal to b so return false
(a >= b) ;               //a=10 greater than b so return false
(a > b) ;                //a=10 greater than b so return true
```

عملگرهای دیگه ای هم وجود دارند که در آینده و با برخورد به اونها در موردشون صحبت می کنیم تا مبحث کسل کننده و طولانی نشود.