C++ چیست

C++ یک زبان برنامه نویسی است که از paradigm یا سبک‌های مختلف برنامه نویسی پشتیبانی می‌کند.C++ نسخه توسعه یافته زبان C می‌باشد و بیشتر کدهای زبان C به راحتی می‌تواند در C++ کامپایل شود. در C++ از ویژگی‌های مهمی که به C اضافه شده است می‌توان به برنامه نویسی شی گرا ، سربارگذاری عملگرها، وراثت چندگانه و مدیریت خطاها اشاره نمود. توسعه C++ در سال ۱۹۷۹ آغاز شد و ۷ سال پس از زبان C به نمایش گذاشته شد.با وجود قدیمی بودن زبان‌های C و c++ هنوز هم به صورت گسترده ای در نرم افزارهای صنعتی مورد استفاده قرار می‌گیرد. این زبان‌ها برای ساخت هر چیزی از سیستم عامل گرفته تا نرم افزارهای توکار ، برنامه‌های دسکتاپ و بازی‌ها مورد استفاده قرار می‌گیرد.

در مقایسه با زبان‌های جدید تر ، برنامه‌های نوشته شده با C++ اغلب پیچیده تر می‌باشند و زمان بیشتری برای توسعه نیاز دارد.در عوض، C++زبانی است که به شما اجازه می‌دهد که هم به صورت High-level (نزدیک به زبان انسان) و هم به صورت low-level (نزدیک به زبان ماشین) سخت افزار را تحت کنترل خود قرار دهید.همچنین با پشتیبانی از سبک‌ها یا پارادایم‌های مختلف برنامه نویسی از جمله رویه ای ، شی گرا یا عمومی ، دست برنامه نویس را در انتخاب سبک مورد نظرش آزاد می‌گذارد. اکنون ۵ نسخه از استاندارد این زبان منتشر شده است؛ و استاندارد C++17 نیز برای انتشار در سال ۲۰۱۷ برنامه ریزی شده است.

نام غیر رسمی	استاندارد++C	سال
C++98	ISO/IEC 14882:1998	1998
C++03	ISO/IEC 14882:2003	2003
C++07/TR1	ISO/IEC TR 19768:2007	2007
C++11	ISO/IEC 14882:2011	2011
C++14	ISO/IEC 14882:2014	2014
C++17	هنوز تعیین نشده.	2017

برای اجرای کدهای C++ نیاز به یک کامپایلر داریم. کامپایلرها و محیط‌های برنامه نویسی (IDE) گوناگونی برای زبان C++ وجود دارند از بین معروف‌ترین آن‌ها می‌توان موارد زیر اشاره نمود:

- Turbo C
- Turbo C++
- Borland C++
- Microsoft C++/C

زبان C++ وابسته به یک سیستم عامل نیست یعنی شما بعد از نوشتن برنامه خود به زبان++C ، اگر کد استانداری نوشته باشید می‌توانید با توجه به سیستم عامل، کدتان را کامپایل کنید. می‌توان کد C++ را در هر محیطی، مثلاً NotePadدر ویندوز و یا gEdit در گنو/لینوکس نوشته و بعد آن را بوسیله یک کامپایلر کامپایل کنیم، ولی برای راحتی کار ما می‌توانیم از یک IDE مناسب، نیز بهره ببریم. البته در این سری آموزشی ما از راحت‌ترین روش برای کامپایل کدها استفاده می‌کنیم.

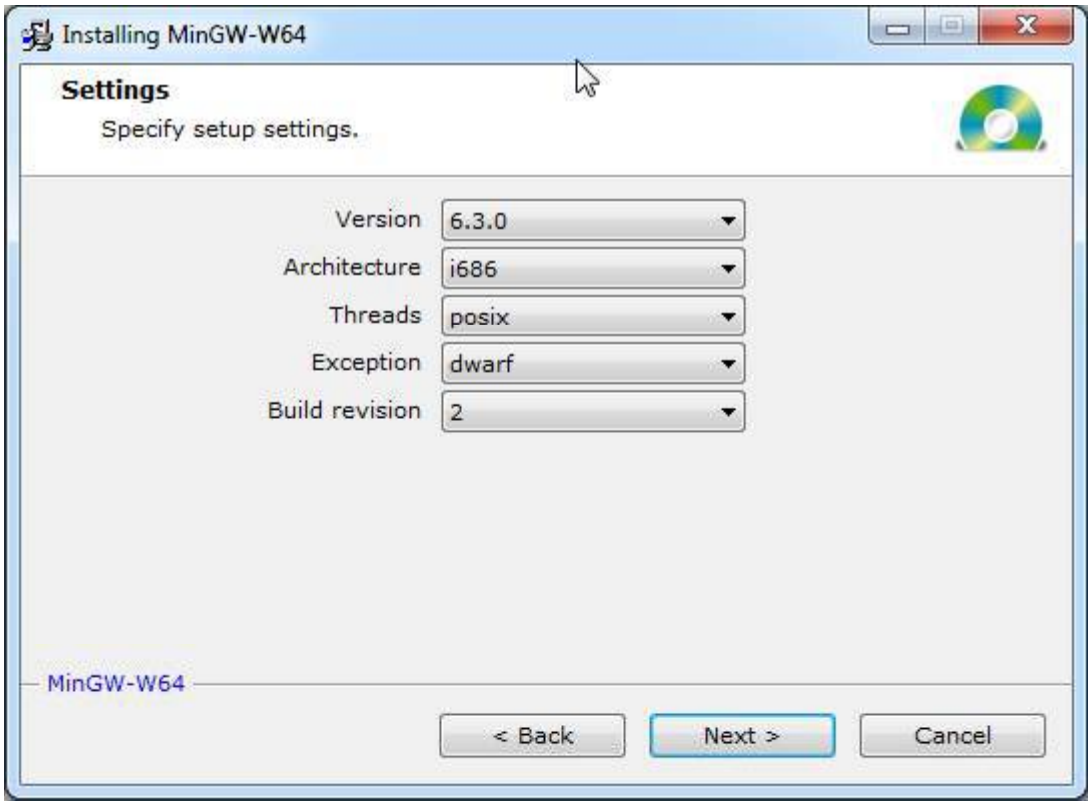
نصب و پیکربندی MinGW

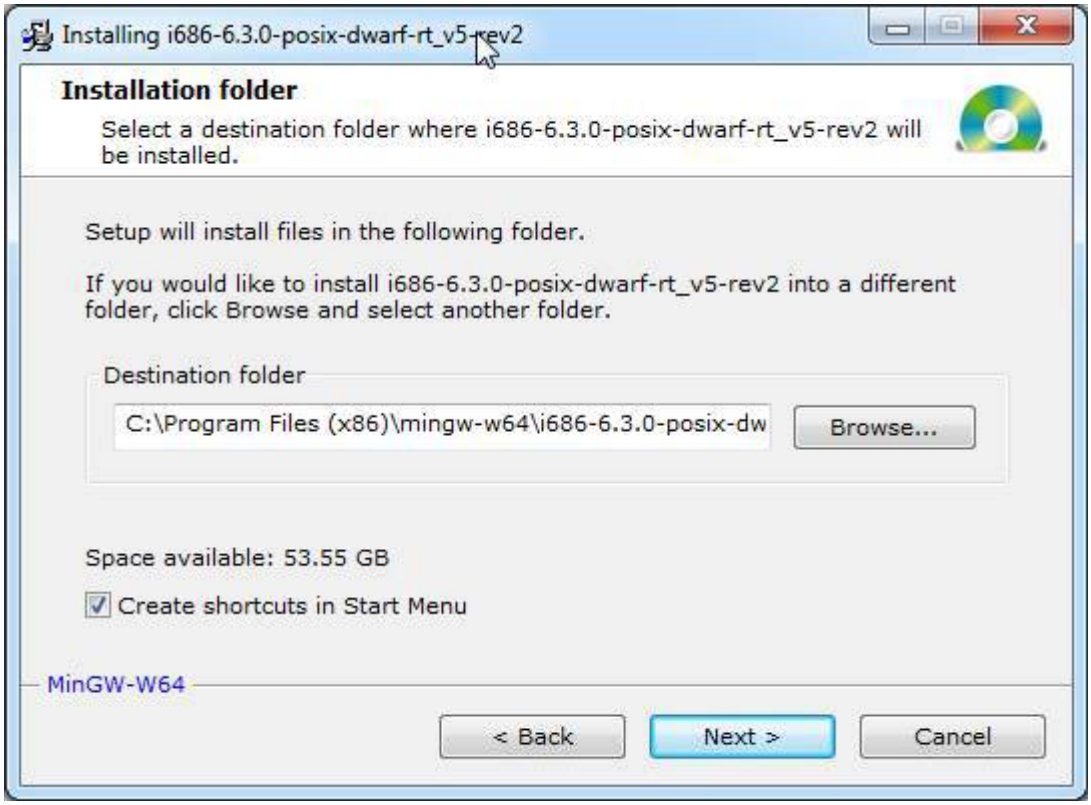
همانطور که اشاره شد برای اجرای کدهای C++ نیاز به یک کامپایلر داریم. کامپایلری که در این سری آموزشی می‌خواهیم از آن استفاده کنیم MinGW می‌باشد که می‌توانید آن را از لینک زیر دانلود کنید:

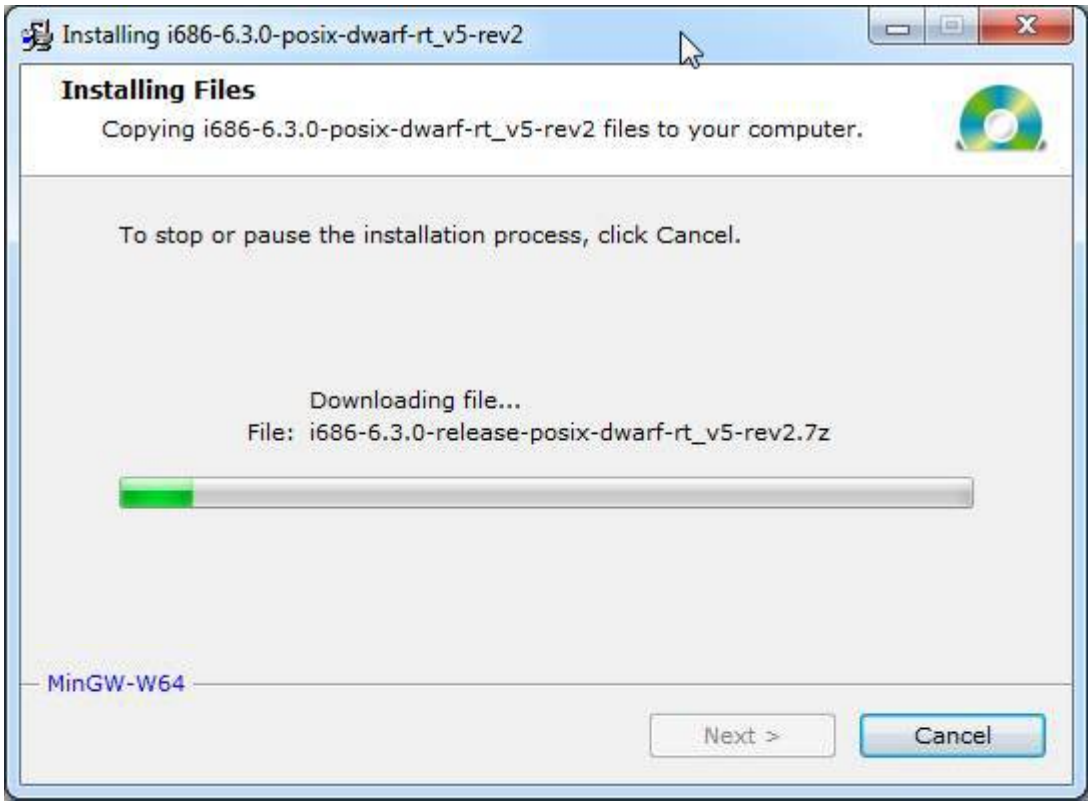
```
dl.w3-farsi.com/Software/Cplusplus/mingw-w64-install.exe
```

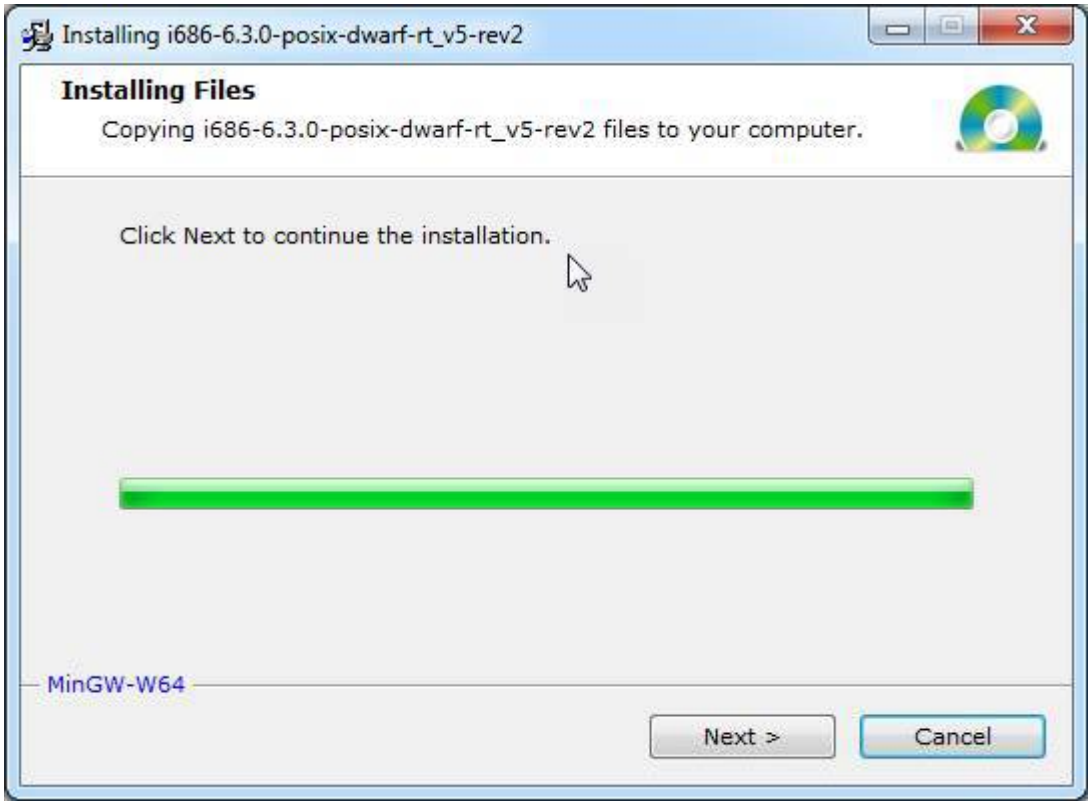
نصب MinGW

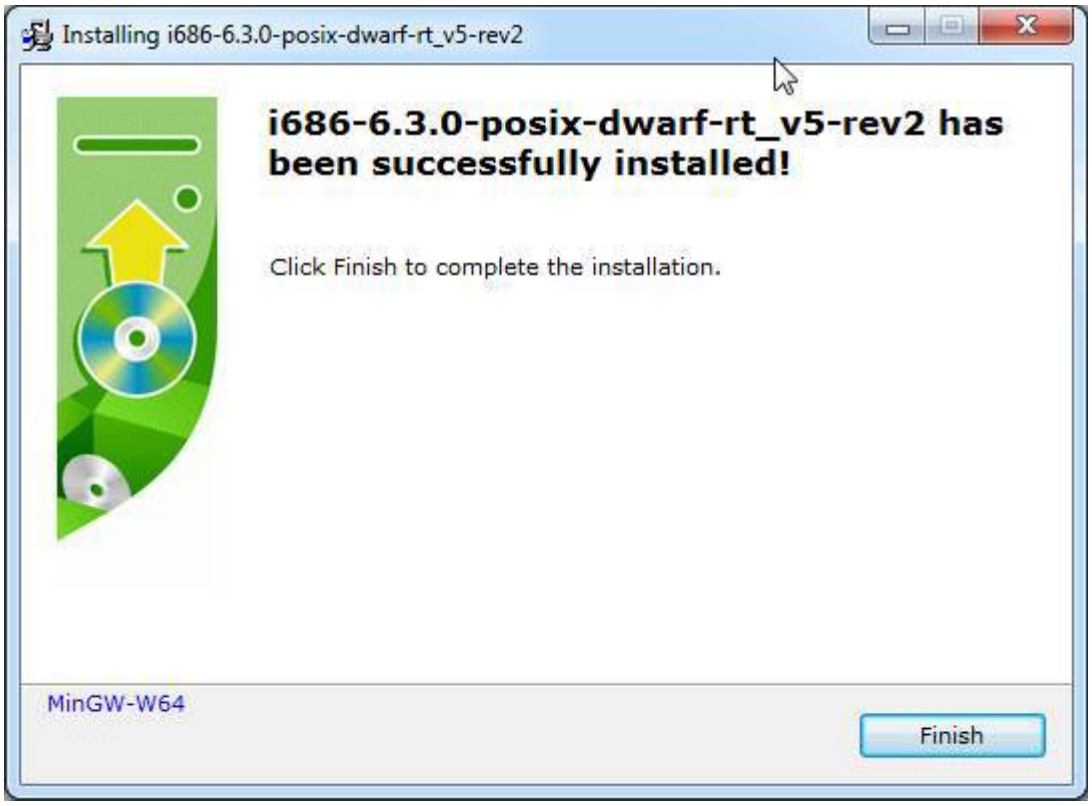
بعد از دانلود فایل بالا بر روی آن دو بار کلیک کرده و مراحل زیر را برای نصب و پیکربندی آن طی کنید:


Installing MinGW-W64
Settings
Specify setup settings.
Version
6.3.0
Architecture
i686
Threads
posix
Exception
dwarf
Build revision
2
MinGW-W64
< Back
Next >
Cancel


Installing i686-6.3.0-posix-dwarf-rt_v5-rev2
Installation folder
Select a destination folder where i686-6.3.0-posix-dwarf-rt_v5-rev2 will be installed.
Setup will install files in the following folder.
If you would like to install i686-6.3.0-posix-dwarf-rt_v5-rev2 into a different folder, click Browse and select another folder.
Destination folder
C:\Program Files (x86)\mingw-w64\i686-6.3.0-posix-dw
Browse...
Space available: 53.55 GB
Create shortcuts in Start Menu
MinGW-W64
< Back
Next >
Cancel


Installing i686-6.3.0-posix-dwarf-rt_v5-rev2
Installing Files
Copying i686-6.3.0-posix-dwarf-rt_v5-rev2 files to your computer.
To stop or pause the installation process, click Cancel.
Downloading file...
File: i686-6.3.0-release-posix-dwarf-rt_v5-rev2.7z
MinGW-W64
Next >
Cancel


Installing i686-6.3.0-posix-dwarf-rt_v5-rev2
Installing Files
Copying i686-6.3.0-posix-dwarf-rt_v5-rev2 files to your computer.
Click Next to continue the installation.
MinGW-W64
Next >
Cancel

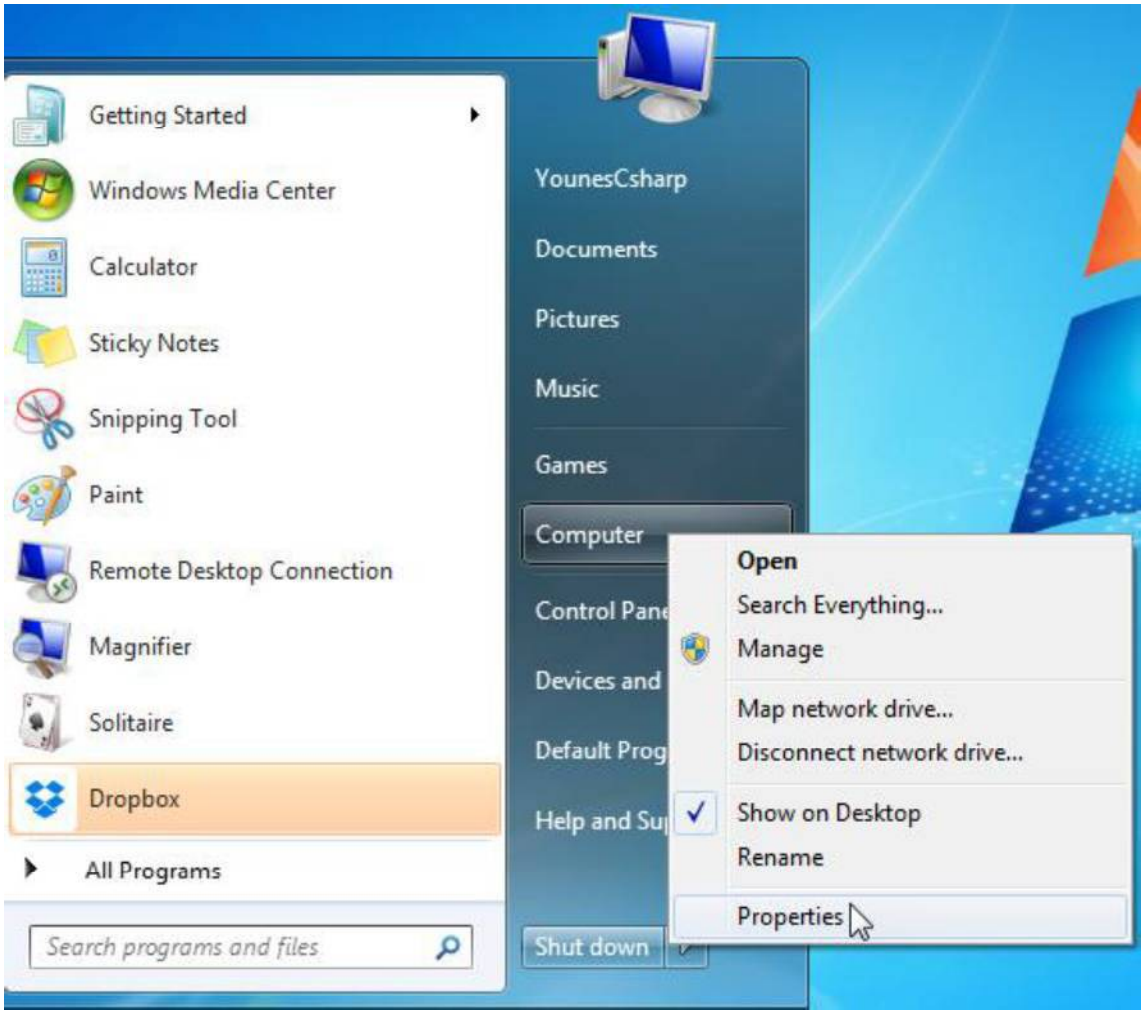


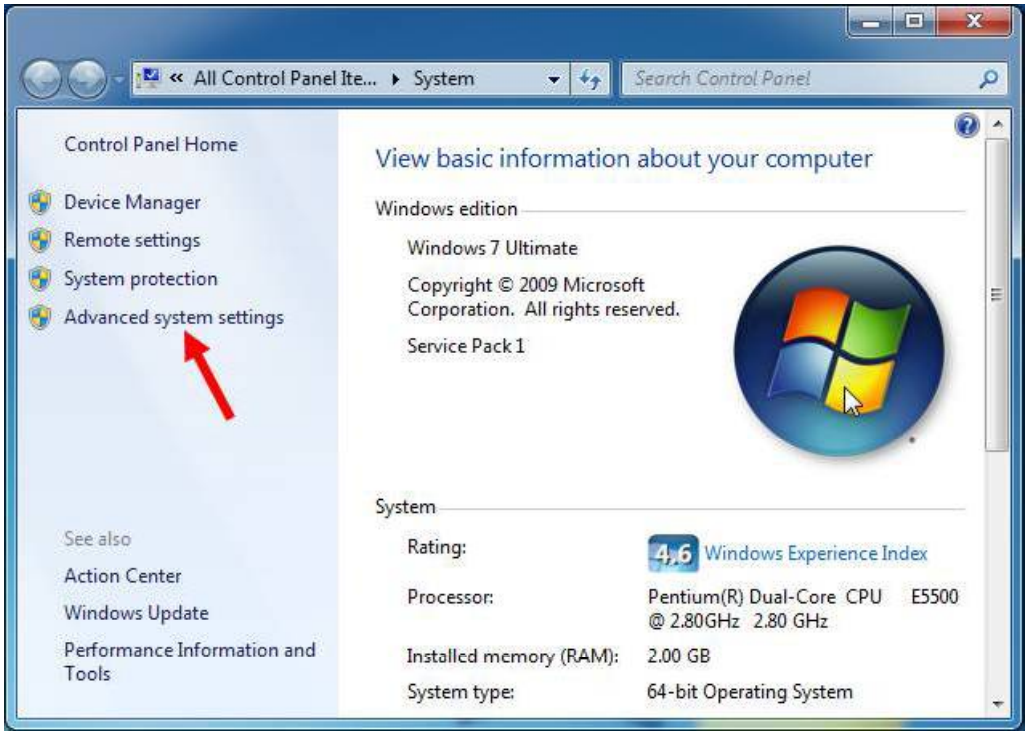
پیکربندی MinGW

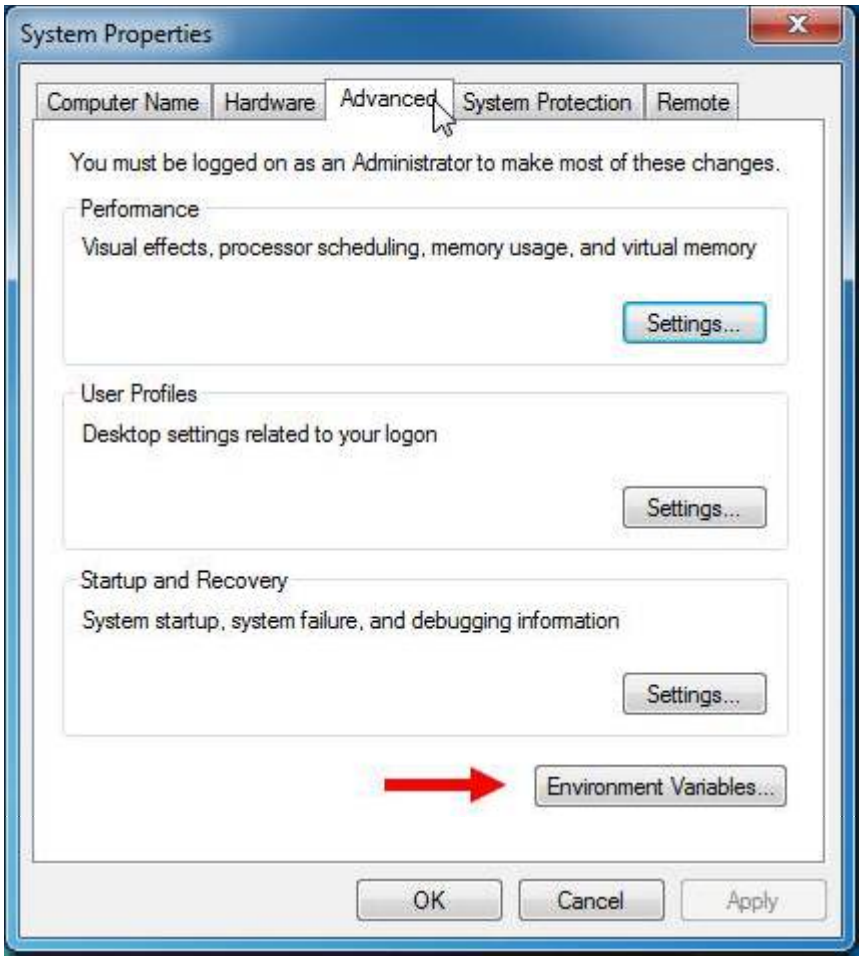
بعد از اتمام نصب نوبت به پیکربندی MinGW می‌رسد. در کل باید مسیر پوشه bin نرم افزار MinGW را در متغیر PATH ویندوز قرار دهید. پوشه bin نرم افزار MinGW ما در مسیر زیر قرار دارد و البته ممکن است برای شما این مسیر متفاوت باشد:

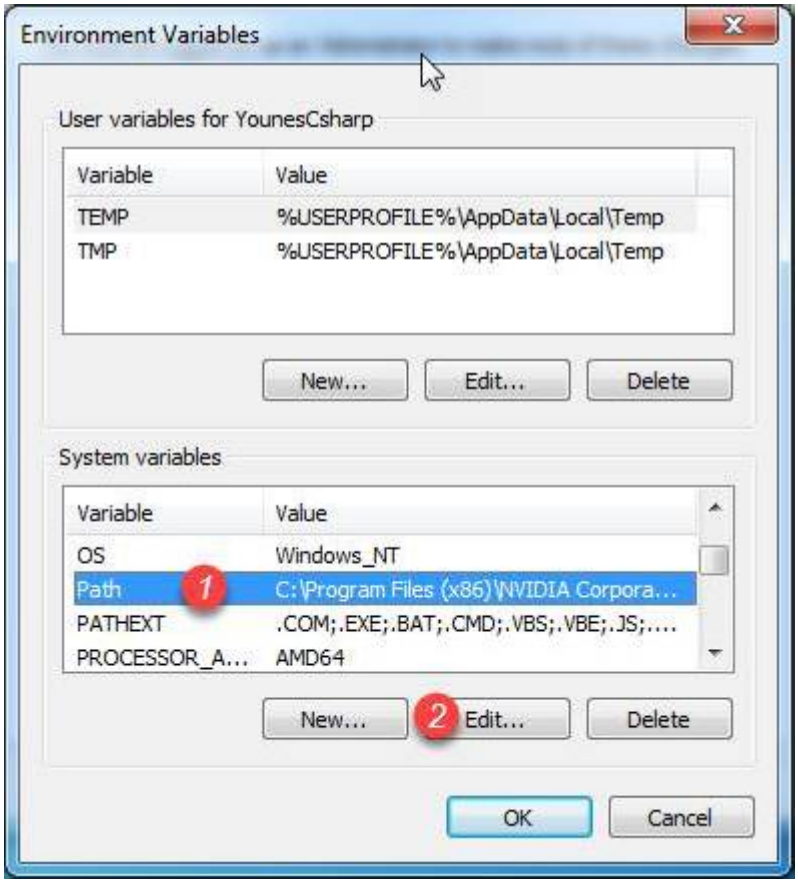
```
C:\Program Files (x86)\mingw-w64\i686-6.3.0-posix-dwarf-rt_v5-rev2\mingw32\bin
```

مسیر پوشه bin را کپی کرده و مراحل زیر را طی کنید:

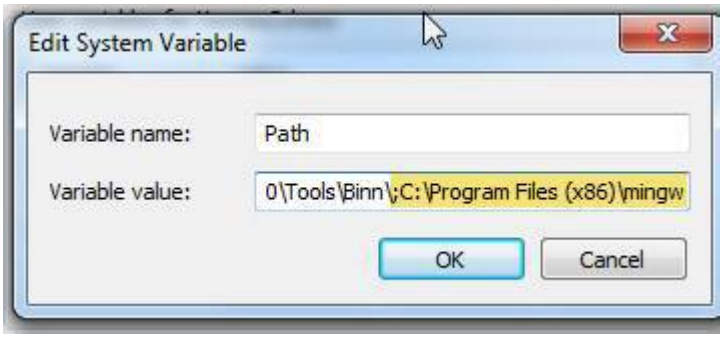

Getting Started
Windows Media Center
Calculator
Sticky Notes
Snipping Tool
Paint
Remote Desktop Connection
Magnifier
Solitaire
Dropbox
All Programs
Search programs and files
YounesCsharp
Documents
Pictures
Music
Games
Computer
Control Pane
Devices and
Default Prog
Help and Su
Shut down
Open
Search Everything...
Manage
Map network drive...
Disconnect network drive...
Show on Desktop
Rename
Properties


« All Control Panel Ite... ▸ System
Search Control Panel
Control Panel Home
Device Manager
Remote settings
System protection
Advanced system settings
View basic information about your computer
Windows edition
Windows 7 Ultimate
Copyright © 2009 Microsoft Corporation. All rights reserved.
Service Pack 1
System
Rating:
4.6 Windows Experience Index
Processor:
Pentium(R) Dual-Core CPU E5500 @ 2.80GHz 2.80 GHz
Installed memory (RAM):
2.00 GB
System type:
64-bit Operating System
See also
Action Center
Windows Update
Performance Information and Tools


System Properties
Computer Name
Hardware
Advanced
System Protection
Remote
You must be logged on as an Administrator to make most of these changes.
Performance
Visual effects, processor scheduling, memory usage, and virtual memory
Settings...
User Profiles
Desktop settings related to your logon
Settings...
Startup and Recovery
System startup, system failure, and debugging information
Settings...
Environment Variables...
OK
Cancel
Apply



همانطور که در شکل بالا مشاهده می‌کنید بعد از کلیک بر روی متغیر `path` سپس دکمه `Edit` پنجره ای به صورت زیر نشان داده می‌شود. نشانگر ماوس را به آخر جعبه متن برده و سپس یک علامت سیمیکالن (;) گذاشته و مسیر پوشه `bin` خود را بعد از علامت سمیکالن کپی کنید:



در نهایت دکمه `OK` تمام پنجره‌های باز را بزنید. با این کار پیکربندی هم به اتمام می‌رسد. در درس بعد نحوه اجرای اولین برنامه با سی پلاس پلاس را به شما آموزش می‌دهیم.

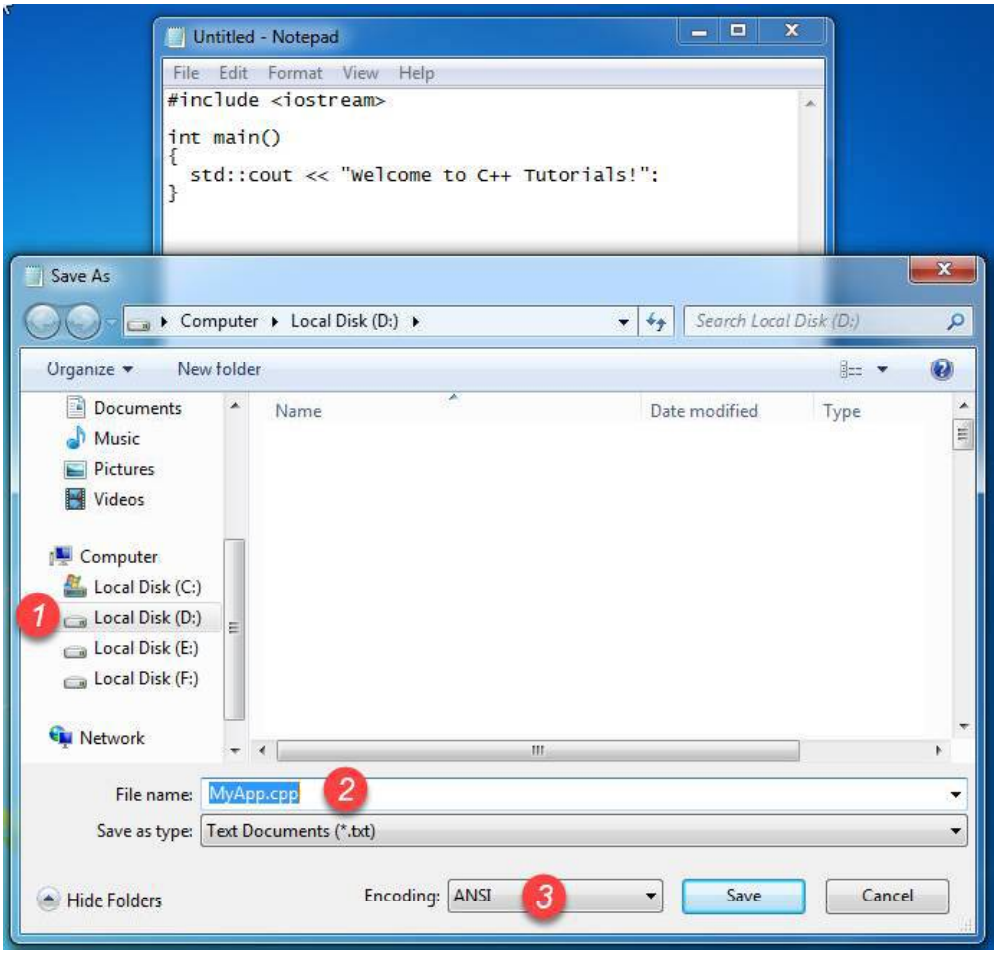
ساخت یک برنامه ساده

اجازه بدهید یک برنامه بسیار ساده به زبان سی پلاس پلاس (C++) بنویسیم. این برنامه یک پیغام را در محیط کنسول نمایش می‌دهد. در این درس می‌خواهم ساختار و دستور زبان یک برنامه ساده C++ را توضیح دهم. هر چند که محیط‌های کدنویسی زیادی برای C++ وجود دارند ولی ما از ساده‌ترین روش برای کدنویسی استفاده می‌کنیم. برنامه NotePad ویندوز را باز کرده و کدهای زیر را در داخل آن بنویسید:

```
#include <iostream>

int main()
{
    std::cout << "Welcome to C++ Tutorials!";
}
```

حال فایل را با نام Myapp و پسوند .cpp به صورت زیر در درایو D ذخیره کنید:



ساختار یک برنامه در C++

مثال بالا ساده‌ترین برنامه ای است که شما می‌توانید در C++ بنویسید. هدف در مثال بالا نمایش یک پیغام در صفحه نمایش است. هر زبان برنامه نویسی دارای قواعدی برای کدنویسی است. اجازه بدهید هر خط کد را در مثال بالا توضیح بدهیم. در خط اول فایل هدر یا سرایند آمده است. فایل‌های سرآیند کتابخانه استاندارد C++ می‌باشند و در این برنامه ما به فایل سرایند iostreamنیاز داریم (در درس‌های آینده در مورد این فایل‌ها به طور مفصل توضیح می‌دهیم). خط ۳ متد main() یا متد اصلی نامیده می‌شود. هر متد شامل یک سری کد است که وقتی اجرا می‌شوند که متد را صدا بزنیم. درباره متد و نحوه صدا زدن آن در فصول بعدی توضیح خواهیم داد. متد main() نقطه آغاز اجرای برنامه است. این بدان معناست که ابتدا تمام کدهای داخل متد main() و سپس بقیه کدها اجرا می‌شود. درباره متد main() در فصول بعدی توضیح خواهیم داد. متد main() و سایر متدها دارای آکولاد و کدهایی در داخل آن‌ها می‌باشند و وقتی کدها اجرا می‌شوند که متدها را صدا بزنیم. هر خط کد در C++به یک سمیکالن (;) ختم می‌شود. اگر سمیکالن در آخر خط فراموش شود برنامه با خطا مواجه می‌شود. مثالی از یک خط کد در C++ به صورت زیر است:

```
std::cout << "Welcome to C++ Tutorials!";
```

این خط کد پیغام Welcome to Visual C++ Tutorials!را در صفحه نمایش نشان می‌دهد. از شیء cout برای چاپ یک رشته استفاده می‌شود. یک رشته گروهی از کاراکترها است، که به وسیله دابل کوتیشن (") محصور شده است. مانند: "Welcome to Visual C++ Tutorials!".

یک کاراکتر می‌تواند یک حرف، عدد، علامت یا ... باشد. در کل مثال بالا نحوه استفاده از شیء cout است که در داخل فضای نام std قرار دارد را نشان می‌دهد. توضیحات بیشتر در درس‌های آینده آمده است. فضای خالی و خطوط جدید را نادیده می‌گیرد. بنابراین شما می‌توانید همه برنامه را در یک خط بنویسید. اما اینکار خواندن و اشکال زدایی برنامه را مشکل می‌کند. یکی از خطاهای معمول در برنامه نویسی فراموش کردن سمیکالن در پایان هر خط کد است. به مثال زیر توجه کنید :

```
std::cout <<
"Welcome to C++ Tutorials!";
```

سی پلاس پلاس فضای خالی بالا را نادیده می‌گیرد و از کد بالا اشکال نمی‌گیرد. اما از کد زیر ایراد می‌گیرد:

```
std::cout << ;
"Welcome to C++ Tutorials!";
```

به سمیکالن آخر خط اول توجه کنید. برنامه با خطای نحوی مواجه می‌شود چون دو خط کد مربوط به یک برنامه هستند و شما فقط باید یک سمیکالن در آخر آن قرار دهید. همیشه به یاد داشته باشید که C++ به بزرگی و کوچکی حروف حساس است. یعنی به طور مثال MAN و man در سی پلاس پلاس با هم فرق دارند. رشته‌ها و توضیحات از این قاعده مستثنی هستند که در درس‌های آینده توضیح خواهیم داد. مثلاً کدهای زیر با خطا مواجه می‌شوند و اجرا نمی‌شوند:

```
std::cOUT << "Welcome to C++ Tutorials!";
STD::cout << "Welcome to C++ Tutorials!";
Std::Cout << "Welcome to C++ Tutorials!";
```

تغییر در بزرگی و کوچکی حروف از اجرای کدها جلوگیری می‌کند. اما کد زیر کاملاً بدون خطا است:

```
std::cout << "Welcome to C++ Tutorials!";
```

همیشه کدهای خود را در داخل آکولاد بنویسید.

```
{
   statement1;
}
```

این کار باعث می‌شود که کدنویسی شما بهتر به چشم بیاید و تشخیص خطاها راحت تر باشد.

کامپایل و اجرای برنامه

برای کامپایل و اجرای برنامه، cmdویندوز را باز کرده و سپس به درایو D یعنی جایی که فایل را ذخیره کرده‌اید بروید:

```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\YounesCsharp>D:

D:\>
```

حال برای کامپایل برنامه کد زیر را نوشته و دکمه Enter را بزنید:

```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\YounesCsharp>D:

D:\>g++ MyApp.cpp -o MyApp.exe
```

و در نهایت برای اجرای برنامه کد زیر را بنویسید:

```
Microsoft Windows [Version 6.1.7601]
```

```
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\YounesCsharp>D:

D:\>g++ MyApp.cpp -o MyApp.exe

D:\>MyApp.exe

Welcome to C++ Tutorials!

D:\>
```

مشاهده می‌کنید که پیغام Welcome to C++ Tutorials! چاپ می‌شود.

وارد کردن فضای نام در برنامه

در برنامه فوق ما یک فضای نام در برنامه مان با نام std داریم، اما سی پلاس پلاس دارای تعداد زیادی فضای نام می‌باشد. یکی از این فضاهای نامی ، فضای نام std است که شیء cout که ما از آن در برنامه بالا استفاده کردیم در این فضای نام قرار دارد.

```
std::cout << "Welcome to C++ Tutorials!";
```

اینکه قبل از استفاده از هر کلاس ابتدا فضای نام آن را مانند کد بالا بنویسیم کمی خسته کننده است. خوشبختانه C++ به ما اجازه می‌دهد که برای جلوگیری از تکرار مکررات، فضاهای نامی را که قرار است در برنامه استفاده کنیم با استفاده از دستور using و کلمه namespace در ابتدای برنامه وارد نماییم:

```
using namespace NameofNameSpace;
```

دستور بالا نحوه وارد کردن یک فضای نام در برنامه را نشان می‌دهد. در نتیجه به جای آنکه به صورت زیر ابتدا نام فضای نام و سپس نام کلاس را بنویسیم:

```
std::cout << "Welcome to C++ Tutorials!";
```

می‌توانیم فضای نام را با دستوری که ذکر شد وارد برنامه کرده و کد بالا را به صورت خلاصه شده زیر بنویسیم:

```
cout << "Welcome to C++ Tutorials!";
```

دستورات using که باعث وارد شدن فضاهای نامی به برنامه می‌شوند عموماً در ابتدای برنامه و قبل از همه کدها نوشته می‌شوند، پس برنامهٔ این درس را می‌توان به صورت زیر نوشت:

```
#include <iostream>
using namespace std;
```

```
int main()
{
    cout << "Welcome to C++ Tutorials!";
}
```

حال که با خصوصیات و ساختار اولیه C++ آشنا شدید در درس‌های آینده مطالب بیشتری از این زبان برنامه نویسی قدرتمند خواهید آموخت.

توضیحات

وقتی که کدی تایپ می‌کنید شاید بخواهید که متنی جهت یادآوری وظیفه آن کد به آن اضافه کنید. در C++ (و بیشتر زبانهای برنامه نویسی) می‌توان این کار را با استفاده از توضیحات انجام داد. توضیحات متونی هستند که توسط کامپایلر نادیده گرفته می‌شوند و به عنوان بخشی از کد محسوب نمی‌شوند.

هدف اصلی از ایجاد توضیحات، بالا بردن خوانایی و تشخیص نقش کدهای نوشته شده توسط شما، برای دیگران است. فرض کنید که می‌خواهید در مورد یک کد خاص، توضیح بدهید، می‌توانید توضیحات را در بالای کد یا کنار آن بنویسید. از توضیحات برای مستند سازی برنامه هم استفاده می‌شود. در برنامه زیر نقش توضیحات نشان داده شده است:

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      // This line will print the message hello world
7      cout << "Hello World!";
8  }
```

در کد بالا، خط ٦ یک توضیح درباره خط ٧ است که به کاربر اعلام می‌کند که وظیفه خط ٧ چیست؟ با اجرای کد بالا فقط جمله Hello World چاپ شده و خط ٧ در خروجی نمایش داده نمی‌شود چون کامپایلر توضیحات را نادیده می‌گیرد . توضیحات بر دو نوع‌اند:

توضیحات تک خطی

```
// single line comment
```

توضیحات چند خطی

```
/* multi
     line
comment */
```

توضیحات تک خطی همانگونه که از نامش پیداست، برای توضیحاتی در حد یک خط به کار می‌روند. این توضیحات با علامت // شروع می‌شوند و هر نوشته ای که در سمت راست آن قرار بگیرد جز توضیحات به حساب می‌آید. این نوع توضیحات معمولاً در بالا یا کنار کد قرار می‌گیرند. اگر توضیح درباره یک کد به بیش از یک خط نیاز باشد از توضیحات چند خطی استفاده می‌شود. توضیحات چند خطی با /*شروع و با */ پایان می‌یابند. هر نوشته ای که بین این دو علامت قرار بگیرد جز توضیحات محسوب می‌شود.

کاراکترهای کنترلی

کاراکترهای کنترلی کاراکترهای ترکیبی هستند که با یک بک اسلش (\) شروع می‌شوند و به دنبال آن‌ها یک حرف یا عدد می‌آید و یک رشته را با فرمت خاص نمایش می‌دهند. برای مثال برای ایجاد یک خط جدید و قرار دادن رشته در آن می‌توان از کاراکتر کنترلی \n استفاده کرد:

```
#include <iostream>
using namespace std;

int main()
{
    cout << "Hello\nWorld";
}
```

```
Hello
World
```

مشاهده کردید که کامپایلر بعد از مواجهه با کاراکتر کنترلی \n نشانگر ماوس را به خط بعد برده و بقیه رشته را در خط بعد نمایش می‌دهد. جدول زیر لیست کاراکترهای کنترلی و کارکرد آن‌ها را نشان می‌دهد:

عملکرد	کاراکتر کنترلی	عملکرد	کاراکتر کنترلی
Form Feed	\f	چاپ کوتیشن	\’
خط جدید	\n	چاپ دابل کوتیشن	\”
سر سطر رفتن	\r	چاپ بک اسلش	\\
حرکت به صورت افقی	\t	چاپ فضای خالی	\0
حرکت به صورت عمودی	\v	صدای بیپ	\a

چاپ کاراکتر یونیکد	\u	حرکت به عقب	\b

ما برای استفاده از کاراکترهای کنترلی از بک اسلش (\) استفاده می‌کنیم.از آنجاییکه \ معنای خاصی به رشته‌ها می‌دهد برای چاپ بک اسلش (\) باید از (\\) استفاده کنیم:

```
cout << "We can print a \\ by using the \\\\ escape sequence.";
```

```
We can print a \ by using the \\ escape sequence.
```

یکی از موارد استفاده از \\، نشان دادن مسیر یک فایل در ویندوز است:

```
cout << "C:\\Program Files\\Some Directory\\SomeFile.txt";
```

```
C:\Program Files\Some Directory\SomeFile.txt
```

از آنجاییکه از دابل کوتیشن (") برای نشان دادن رشته‌ها استفاده می‌کنیم برای چاپ آن از \"استفاده می‌کنیم:

```
cout << "I said, \"Motivate yourself!\".";
```

```
I said, "Motivate yourself!".
```

همچنین برای چاپ کوتیشن (') از \'استفاده می‌کنیم:

```
cout << "The programmer\'s heaven.";
```

```
The programmer's heaven.
```

برای ایجاد فاصله بین حروف یا کلمات از \t استفاده می‌شود:

```
cout << "Left\tRight";
```

```
Left  Right
```

هر تعداد کاراکتر که بعد از کاراکتر کنترلی \r بیایند به اول سطر منتقل و جایگزین کاراکترهای موجود می‌شوند:

```
cout << "Mitten\rK";
```

```
Kitten
```

مثلاً در مثال بالا کاراکتر K بعد از کاراکتر کنترلی \r آمده است.کاراکتر کنترلی حرف K را به ابتدای سطر برده و جایگزین حرف M می‌کند. برای چاپ کاراکترهای یونیکد می‌توان از \u استفاده کرد. برای استفاده از \u،مقدار در مبنای ۱۶ کاراکتر را درست بعد از علامت \u قرار می‌دهیم.برای مثال اگر بخواهیم علامت(-۲) را چاپ کنیم باید بعد از علامت \u مقدار ۰۰ A9 را قرار دهیم مانند :

```
cout << "\u00A9";
```

```
T-
```

برای مشاهده لیست مقادیر مبنای ۱۶ برای کاراکترهای یونیکد به لینک زیر مراجعه نمایید:

http://www.ascii.cl/htmlcodes.htm

اگر کامپایلر به یک کاراکتر کنترلی غیر مجاز برخورد کند، برنامه پیغام خطا می‌دهد.بیشترین خطا زمانی اتفاق می‌افتد که برنامه نویس برای چاپ اسلش (\) از \\ استفاده می‌کند. برای دریافت اطلاعات بیشتر در مورد کاراکترهای کنترلی به لینک زیر مراجعه کنید:

https://msdn.microsoft.com/en-us/library/h21280bw.aspx

متغیر

متغیر مکانی از حافظه است که شما می‌توانید مقادیری را در آن ذخیره کنید.می‌توان آن را به عنوان یک ظرف تصور کرد که داده‌های خود را در آن قرار داده‌اید.محتویات این ظرف می‌تواند پاک شود یا تغییر کند.هر متغیر دارای یک نام نیز هست.که از طریق آن می‌توان متغیر را از دیگر متغیرها تشخیص داد و به مقدار آن دسترسی پیدا کرد.همچنین دارای یک مقدار می‌باشد که می‌تواند توسط کاربر انتخاب شده باشد یا نتیجه یک محاسبه باشد.مقدار متغیر می‌تواند تهی نیز باشد.متغیر دارای نوع نیز هست بدین معنی که نوع آن با نوع داده ای که در آن ذخیره می‌شود یکی است. متغیر دارای عمر نیز هست که از روی آن می‌توان تشخیص داد که متغیر باید چقدر در طول برنامه مورد استفاده قرار گیرد.و در نهایت متغیر دارای محدوده استفاده نیز هست که به شما می‌گوید که متغیر در چه جای برنامه برای شما قابل دسترسی است.ما از متغیرها به عنوان یک انبار موقتی برای ذخیره داده استفاده می‌کنیم. هنگامی که یک برنامه ایجاد می‌کنیم احتیاج به یک مکان برای ذخیره داده ،مقادیر یا داده‌هایی که توسط کاربر وارد می‌شوند داریم.ایم مکان همان متغیر است. برای این از کلمه متغیر استفاده می‌شود چون ما می‌توانیم بسته به نوع شرایط هر جا که لازم باشد مقدار آن را تغییر دهیم. متغیرها موقتی هستند و فقط موقعی مورد استفاده قرار می‌گیرند که برنامه در حال اجراست و وقتی شما برنامه را می‌بندید محتویات متغیرها نیز پاک می‌شود. قبلاً ذکر شد که به وسیله نام متغیر می‌توان به آن دسترسی پیدا کرد. برای نامگذاری متغیرها باید قوانین زیر را رعایت کرد:

- نام متغیر باید با یک از حروف الفبا (a-z or A-Z) شروع شود.
- نمی‌تواند شامل کاراکترهای غیرمجاز مانند . $, ^, ?, #باشد.

- نمی‌توان از کلمات رزرو شده در C++ برای نام متغیر استفاده کرد.
- نام متغیر نباید دارای فضای خالی (spaces) باشد.
- اسامی متغیرها نسبت به بزرگی و کوچکی حروف حساس هستند. در C++ دو حرف مانند a و A دو کاراکتر مختلف به حساب می‌آیند.

دو متغیر با نامهای myNumber و MyNumber دو متغیر مختلف محسوب می‌شوند چون یکی از آن‌ها با حرف کوچک m و دیگری با حرف بزرگ M شروع می‌شود. شما نمی‌توانید دو متغیر را که دقیق شبیه هم هستند را در یک scope (محدوده) تعریف کنید. Scope به معنای یک بلوک کد است که متغیر در آن قابل دسترسی و استفاده است. در مورد Scope در فصل‌های آینده بیشتر توضیح خواهیم داد. متغیر دارای نوع هست که نوع داده ای را که در خود ذخیره می‌کند را نشان می‌دهد. معمول‌ترین انواع داده int, double, string, char, float می‌باشند. برای مثال شما برای قرار دادن یک عدد صحیح در متغیر باید از نوع int استفاده کنید.

انواع ساده

انواع ساده انواعی از داده‌ها هستند که شامل اعداد، کاراکترها و رشته‌ها و مقادیر بولی می‌باشند. به انواع ساده انواع اصلی نیز گفته می‌شود چون از آن‌ها برای ساخت انواع پیچیده تری مانند کلاس‌ها و ساختارها استفاده می‌شود. انواع ساده دارای مجموعه مشخصی از مقادیر هستند و محدوده خاصی از اعداد را در خود ذخیره می‌کنند. در C++ هفت نوع داده وجود دارد که در جدول زیر ذکر شده‌اند:

کلمه کلیدی	نوع
bool	Boolean
char	Character
int	Integer
float	Floating point
double	Double floating point
void	Valueless
wchar_t	Wide character

انواع بالا (به جز void) می‌توانند با عباراتی مثل signed ، long، unsignedو short ترکیب شده و نوع‌های دیگری را به وجود آورند:

نوع	مقدار فضایی که از حافظه اشغال می‌کند	محدوده
char	1byte	-128 to 127 or 0 to 255
unsigned char	1byte	0 to 255
signed char	1byte	-128 to 127
int	4bytes	-2147483648 to 2147483647
unsigned int	4bytes	0 to 4294967295
signed int	4bytes	-2147483648 to 2147483647
short int	2bytes	-32768 to 32767
unsigned short int	2bytes	0 to 65,535
signed short int	2bytes	-32768 to 32767
long int	8bytes	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
signed long int	8bytes	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
unsigned long int	8bytes	0 to 18,446,744,073,709,551,615
float	4bytes	+/- 3.4e +/- 38 (~7 digits)
double	8bytes	+/- 1.7e +/- 308 (~15 digits)
long double	8bytes	+/- 1.7e +/- 308 (~15 digits)
wchar_t	2 or 4 bytes	1 wide character

نوع char برای ذخیره کاراکترهای یونیکد استفاده می‌شود.کاراکترها باید داخل یک کوتیشن ساده قرار بگیرند مانند.('a')

نوع bool فقط می‌تواند مقادیر درست (true) یا نادرست (false) را در خود ذخیره کند و بیشتر در برنامه‌هایی که دارای ساختار تصمیم گیری هستند مورد استفاده قرار می‌گیرد.

نوع stringبرای ذخیره گروهی از کاراکترها مانند یک پیغام استفاده می‌شود. مقادیر ذخیره شده در یک رشته باید داخل دابل کوتیشن قرار گیرند تا توسط کامپایلر به عنوان یک رشته در نظر گرفته شوند. مانند("massage")

استفاده از متغیرها

در مثال زیر نحوه تعریف و مقدار دهی متغیرها نمایش داده شده است:

```
#include <iostream>
using namespace std;

int main()
{
    //Declare variables
    int num1;
    int num2;
    double num3;
    double num4;
    bool boolVal;
    char myChar;
    string message;

    //Assign values to variables
    num1 = 1;
    num2 = 2;
    num3 = 3.54;
    num4 = 4.12;
    boolVal = true;
    myChar = 'R';
    message = "Hello World!";

    //Show the values of the variables
    cout << "num1 = " << num1 << endl;
    cout << "num2 = " << num2 << endl;
    cout << "num3 = " << num3 << endl;
    cout << "num4 = " << num4 << endl;
    cout << "boolVal = " << boolVal << endl;
    cout << "myChar = " << myChar << endl;
    cout << "message = " << message << endl;

}
```

تعریف متغیر

در کد بالا متغیرهایی با نوع و نام متفاوت تعریف شده‌اند. ابتدا باید نوع داده‌هایی را که این متغیرها قرار است در خود ذخیره کنند را مشخص کنیم و سپس یک نام برای آن‌ها در نظر بگیریم و در آخر سیمیکولن بگذاریم.همیشه به یاد داشته باشید که قبل از مقدار دهی و استفاده از متغیر باید آن را تعریف کرد. شاید برایتان این سؤال پیش آمده باشد که کاربرد endl چیست؟ endl برای ایجاد خط جدید مورد استفاده قرار گرفته است. یعنی نشانگر ماوس را همانند کاراکتر کنترلی \n به خط بعد می‌برد، در نتیجه خروجی کد بالا در خطوط جداگانه چاپ می‌شود .

```
//Declare variables
int num1;
int num2;
double num3;
double num4;
bool boolVal;
char myChar;
string message;
```

نحوه تعریف متغیر به صورت زیر است :

```
data_type identifier;
```

date_type همان نوع داده است مانند int ، doubleو ..... dentifier نیز نام متغیر است که به ما امکان استفاده و دسترسی به مقدار متغیر را می‌دهد .برای تعریف چند متغیر از یک نوع می‌توان به صورت زیر عمل کرد:

```
data_type identifier1, identifier2, ... indentifierN;
```

مثال

```
int num1, num2, num3, num4, num5;
string message1, message2, message3;
```

در مثال بالا ۵ متغیر از نوع صحیح و ۳ متغیر از نوع رشته تعریف شده است.توجه داشته باشید که بین متغیرها باید علامت کاما (,)باشد.

نامگذاری متغیرها

- نام متغیر باید با یک حرف یا زیرخط و به دنبال آن حرف یا عدد شروع شود.
- نمی‌توان از کاراکترهای خاص مانند & ,% ,#یا عدد برای شروع نام متغیر استفاده کرد مانند .2numbers

- نام متغیر نباید دارای فاصله باشد.برای نام‌های چند حرفی می‌توان به جای فاصله از علامت زیرخط یا _استفاده کرد.

نامهای مجاز:

```
num1  myNumber  studentCount  total       first_name    _minimum
num2  myChar    average       amountDue   last_name     _maximum
name  counter   sum           isLeapYear  color_of_car  _age
```

نامهای غیر مجاز:

```
123        #numbers#  #ofstudents  1abc2
123abc     $money     first name   ty.np
my number  this&that  last name    1:00
```

اگر به نامهای مجاز در مثال بالا توجه کنید متوجه قراردادهای به کار رفته در نامگذاری آن‌ها خواهید شد.یکی از روش‌های نامگذاری، نامگذاری کوهان شتری است.در این روش که برای متغیرهای دو کلمه ای به کار می‌رود،اولین حرف اولین کلمه با حرف کوچک و اولین حرف دومین کلمه با حرف بزرگ نمایش داده می‌شود مانند :myNumber . توجه کنید که اولین حرف کلمه Number با حرف بزرگ شروع شده است. مثال دیگر کلمه numberOfStudents است.اگز توجه کنید بعد از اولین کلمه حرف اول سایر کلمات با حروف بزرگ نمایش داده شده است.

محدوده متغیر

متغیرهای ابتدای درس در داخل متد main() تعریف شده‌اند. در نتیجه این متغیرها فقط در داخل متد main() قابل دسترسی هستند. محدوده یک متغیر مشخص می‌کند که متغیر در کجای کد قابل دسترسی است. هنگامیکه برنامه به پایان متد main() می‌رسد متغیرها از محدوده خارج و بدون استفاده می‌شوند تا زمانی که برنامه در حال اجراست.محدوده متغیرها انواعی دارد که در درس‌های بعدی با آن‌ها آشنا می‌شوید. تشخیص محدوده متغیر بسیار مهم است چون به وسیله آن می‌فهمید که در کجای کد می‌توان از متغیر استفاده کرد. باید یاد آور شد که دو متغیر در یک محدوده نمی‌توانند دارای نام یکسان باشند. مثلاً کد زیر در برنامه ایجاد خطا می‌کند:

```
int num1;
int num1;
```

از آنجاییکه C++ به بزرگی و کوچک بودن حروف حساس است می‌توان از این خاصیت برای تعریف چند متغیر هم نام ولی با حروف متفاوت (از لحاظ بزرگی و کوچکی) برای تعریف چند متغیر از یک نوع استفاده کرد مانند:

```
int num1;
int Num1;
int NUM1;
```

مقداردهی متغیرها

می‌توان فوراً بعد از تعریف متغیرها مقادیری را به آن‌ها اختصاص داد.این عمل را مقداردهی می‌نامند. در زیر نحوه مقدار دهی متغیرها نشان داده شده است:

```
data_type identifier = value;
```

به عنوان مثال:

```
int myNumber = 7;
```

همچنین می‌توان چندین متغیر را فقط با گذاشتن کاما بین آن‌ها به سادگی مقدار دهی کرد:

```
data_type variable1 = value1, varaible2 = value2, ... variableN, valueN;
int num1 = 1, num2 = 2, num3 = 3;
```

تعریف متغیر با مقدار دهی متغیرها متفاوت است. تعریف متغیر یعنی انخاب نوع و نام برای متغیر ولی مقدار دهی یعنی اختصاص یک مقدار به متغیر.

اختصاص مقدار به متغیر

در زیر نحوه اختصاص مقادیر به متغیرها نشان داده شده است:

```
num1 = 1;
num2 = 2;
num3 = 3.54;
num4 = 4.12;
boolVal = true;
myChar = 'R';
message = "Hello World!";
```

به این نکته توجه کنید که شما به مغیری که هنوز تعریف نشده نمی‌توانید مقدار بدهید. شما فقط می‌توانید از متغیرهایی استفاده کنید که هم تعریف و هم مقدار دهی شده باشند. مثلاً متغیرهای بالا همه قابل استفاده هستند. در این مثال num1 و num2 هر دو

تعریف شده‌اند و مقادیری از نوع صحیح به آن‌ها اختصاص داده شده است. اگر نوع داده با نوع متغیر یکی نباشد برنامه پیغام خطا می‌دهد.

ثابت

ثابت‌ها انواعی هستند که مقدار آن‌ها در طول برنامه تغییر نمی‌کند. ثابت‌ها حتماً باید مقدار دهی اولیه شوند و اگر مقدار دهی آن‌ها فراموش شود در برنامه خطا به وجود می‌آید. بعد از این که به ثابت‌ها مقدار اولیه اختصاص داده شد هرگز در زمان اجرای برنامه نمی‌توان آن را تغییر داد. برای تعریف ثابت‌ها باید از کلمه کلیدی const و #define استفاده کرد. معمولاً نام ثابت‌ها را طبق قرارداد با حروف بزرگ می‌نویسند تا تشخیص آن‌ها در برنامه راحت باشد. نحوه تعریف ثابت در زیر آمده است:

```
const data_type identifier = initial_value;
```

در کد بالا ابتدا کلمه کلیدی const و سپس نوع ثابت و بعد نام ثابت را با حروف بزرگ می‌نویسیم. و در نهایت یک مقدار را به آن اختصاص می‌دهیم و علامت سمیکالن می‌گذاریم.

```
#define data_type identifier initial_value
```

در روش بالا فقط #define را نوشته و سپس نام ثابت و بعد مقداری که قرار است دریافت کند. به این نکته توجه کنید که در روش بالا نه علامت سمیکالن وجود دارد و نه علامت مساوی. مثال:

```
#include <iostream>
using namespace std;

int main()
{
    const int NUMBER = 1;

    NUMBER = 20; //ERROR, Cant modify a constant

    cout << NUMBER;
}
```

```
#include <iostream>
using namespace std;

int main()
{
    #define NUMBER 1

    NUMBER = 20; //ERROR, Cant modify a constant

    cout << NUMBER;
```

```
}
```

در این مثال می‌بینید که مقدار دادن به یک ثابت ،که قبلاً مقدار دهی شده برنامه را با خطا مواجه می‌کند.نکتهٔ دیگری که نباید فراموش شود این است که نباید مقدار ثابت را با مقدار دیگر متغیرهای تعریف شده در برنامه برابر قرار داد.

مثال:

```
int someVariable;
const int MY_CONST = someVariable;
```

ممکن است این سؤال برایتان پیش آمده باشد که دلیل استفاده از ثابت‌ها چیست؟ اگر مطمئن هستید که مقادیری در برنامه وجود دارند که هرگز در طول برنامه تغییر نمی‌کنند بهتر است که آن‌ها را به صورت ثابت تعریف کنید.این کار هر چند کوچک کیفیت برنامه شما را بالا می‌برد.

عبارات و عملگرها

ابتدا با دو کلمه آشنا شوید:

- عملگر : نمادهایی هستند که اعمال خاص انجام می‌دهند.
- عملوند : مقادیری که عملگرها بر روی آن‌ها عملی انجام می‌دهند.

مثلاً X+Y: یک عبارت است که در آن X و Y عملوند و علامت + عملگر به حساب می‌آیند.

زبانهای برنامه نویسی جدید دارای عملگرهایی هستند که از اجزاء معمول زبان به حساب می‌آیند C++ .دارای عملگرهای مختلفی از جمله عملگرهای ریاضی، تخصیصی ، مقایسه ای، منطقی و بیتی می‌باشد. از عملگرهای ساده ریاضی می‌توان به عملگر جمع و تفریق اشاره کرد. سه نوع عملگر در C++ وجود دارد:

- یگانی – (Unary) به یک عملوند نیاز دارد
- دودویی – (Binary)به دو عملوند نیاز دارد
- سه تایی – (Ternary) به سه عملوند نیاز دارد

انواع مختلف عملگر که در این بخش مورد بحث قرار می‌گیرند عبارت‌اند از:

- عملگرهای ریاضی
- عملگرهای تخصیصی

- عملگرهای مقایسه ای
- عملگرهای منطقی
- عملگرهای بیتی
- عملگرهای ریاضی

عملگرهای ریاضی

C++ از عملگرهای ریاضی برای انجام محاسبات استفاده می‌کند. جدول زیر عملگرهای ریاضی سی پلاس پلاس را نشان می‌دهد:

عملگر	دسته	مثال	نتیجه
+	Binary	var1 = var2 + var3;	Var1برابر است با حاصل جمع var2و var3
-	Binary	var1 = var2 - var3;	Var1برابر است با حاصل تفریق var2و var3
*	Binary	var1 = var2 * var3;	Var1برابر است با حاصلضرب var2در var3
/	Binary	var1 = var2 / var3;	Var1برابر است با حاصل تقسیم var2بر var3
%	Binary	var1 = var2 % var3;	Var1برابر است با باقیمانده تقسیم var2و var3
+	Unary	var1 = +var2;	Var1برابر است با مقدار var2
-	Unary	var1 = -var2	Var1برابر است با مقدار var2ضربدر ۱ -

مثال بالا در از نوع عددی استفاده شده است.اما استفاده از عملگرهای ریاضی برای نوع رشته ای نتیجه متفاوتی دارد. همچنین در جمع دو کاراکتر کامپایلر معادل عددی آن‌ها را نشان می‌دهد. دیگر عملگرهای C++ عملگرهای کاهش و افزایش هستند.این عملگرها مقدار ۱ را از متغیرها کم یا به آن‌ها اضافه می‌کنند. از این متغیرها اغلب در حلقه‌ها استفاده می‌شود:

عملگر	دسته	مثال	نتیجه
++	Unary	var1 = ++var2;	مقدار var1 برابر است با var2 بعلاوه ۱
- -	Unary	var1 = - -var2;	مقدار var1 برابر است با var2 منهای ۱

++	Unary	var1 = var2++;	مقدار var1 برابر است باvar2 به متغیر var2 یک واحد اضافه می‌شود
-	Unary	var1 = var2-;	مقدار var1 برابر است باvar2 از متغیر var2 یک واحد کم می‌شود

به این نکته توجه داشته باشید که محل قرار گیری عملگر در نتیجه محاسبات تأثیر دارد. اگر عملگر قبل از متغیر var2 بیاید افزایش یا کاهش var1 اتفاق می‌افتد.چنانچه عملگرها بعد از متغیر var2 قرار بگیرند ابتدا var1 برابر var2می‌شود و سپس متغیر var2افزایش یا کاهش می‌یابد. به مثال‌های زیر توجه کنید:

```
#include <iostream>
using namespace std;

int main()
{
    int x = 0;
    int y = 1;

    x = ++y;

    cout << "x= {0}" << x << endl;
    cout << "y= {0}" << y << endl;
}
```

```
x=2
y=2
```

```
#include <iostream>
using namespace std;

int main()
{
    int x = 0;
    int y = 1;

    x = --y;

    cout << "x= {0}" << x << endl;
    cout << "y= {0}" << y << endl;
}
```

```
x=0
y=0
```

همانطور که در دو مثال بالا مشاهده می‌کنید،درج عملگرهای – و ++ قبل از عملوند y باعث می‌شود که ابتدا یک واحد از y کم و یا یک واحد به y اضافه شود و سپس نتیجه در عملوند x قرار بگیرد. حال به دو مثال زیر توجه کنید:

```
#include <iostream>
using namespace std;

int main()
{
    int x = 0;
    int y = 1;

    x = y--;

    cout << "x= {0}" << x << endl;
    cout << "y= {0}" << y << endl;
}
```

```
x=1
y=0
```

```
#include <iostream>
using namespace std;

int main()
{
    int x = 0;
    int y = 1;

    x = y++;

    cout << "x= {0}" << x << endl;
    cout << "y= {0}" << y << endl;
}
```

```
x=1
y=2
```

همانطور که در دو مثال بالا مشاهده می‌کنید،درج عملگرهای – و ++ بعد از عملوند y باعث می‌شود که ابتدا مقدار y در داخل متغیر x قرار بگیرد و سپس یک واحد از y کم و یا یک واحد به ان اضافه شود. حال می‌توانیم با ایجاد یک برنامه نحوه عملکرد عملگرهای ریاضی در C++ را یاد بگیریم:

```
#include <iostream>
using namespace std;

int main()
{
      //Variable declarations
      int num1, num2;
      string msg1, msg2;
```

```
    //Assign test values
    num1 = 6;
    num2 = 3;

    //Demonstrate use of mathematical operators
    cout << "The sum of num1 and num2 is " << (num1 + num2) << endl;
    cout << "The difference of num1 and num2 is " << (num1 - num2) << endl;
    cout << "The product of num1 and num2 is " << (num1 * num2) << endl;
    cout << "The quotient of num1 and num2 is " << (num1 / num2) << endl;
    cout << "The remainder of num1 and num2 is " << (num1 % num2) << endl;

    msg1 = "Hello ";
    msg2 = "World!";
    cout << msg1 + msg2;
}
```

```
The sum of 6 and 3 is 9.
The difference of 6 and 3 is 3.
The product of 6 and 3 is 18.
The quotient of 6 and 3 is 2.
The remainder of 6 divided by 3 is 0
Hello World!
```

برنامه بالا نتیجه هر عبارت را نشان می‌دهد. در این برنامه از endl برای نشان دادن نتایج در سطرهای متفاوت استفاده شده است. C++ خط جدید و فاصله و فضای خالی را نادیده می‌گیرد. در خط ۲۲ مشاهده می‌کنید که دو رشته به وسیله عملگر + به هم متصل شده‌اند. نتیجه استفاده از عملگر + برای چسباندن دو کلمه "Hello " و "World!" رشته "Hello World!" خواهد بود. به فاصله‌های خالی بعد از اولین کلمه توجه کنید اگر آن‌ها را حذف کنید از خروجی برنامه نیز حذف می‌شوند.

عملگرهای تخصیصی

نوع دیگر از عملگرهای C++ عملگرهای جایگزینی نام دارند. این عملگرها مقدار متغیر سمت راست خود را در متغیر سمت چپ قرار می‌دهند. جدول زیر انواع عملگرهای تخصیصی در سی شارپ را نشان می‌دهد:

عملگر	مثال	نتیجه
=	var1 = var2;	مقدار var1 برابر است با مقدار var2
+=	var1 += var2;	مقدار var1 برابر است با حاصل جمع var1 و var2
-=	var1 -= var2;	مقدار var1 برابر است با حاصل تفریق var1 و var2
*=	var1 *= var2;	مقدار var1 برابر است با حاصل ضرب var1 در var2

/=	var1 /= var2;	مقدار var1 برابر است با حاصل تقسیم var1 بر var2
%=	var1 %= var2;	مقدار var1 برابر است با باقیمانده تقسیم var1 بر var2

استفاده از این نوع عملگرها در واقع یک نوع خلاصه نویسی در کد است. مثلاً شکل اصلی کد var1 += var2به صورت var1 = var1 + var2 می‌باشد. این حالت کدنویسی زمانی کارایی خود را نشان می‌دهد که نام متغیرها طولانی باشد. برنامه زیر چگونگی استفاده از عملگرهای تخصیصی و تأثیر آن‌ها را بر متغیرها نشان می‌دهد.

```
#include <iostream>
using namespace std;

int main()
{
    int number;

    cout << "Assigning 10 to number..." << endl;
    number = 10;
    cout << "Number = " << number << endl;

    cout << "Adding 10 to number..." << endl;
    number += 10;
    cout << "Number = " << number << endl;

    cout << "Subtracting 10 from number..." << endl;
    number -= 10;
    cout << "Number = " << number << endl;
}
```

```
Assigning 10 to number...
Number = 10
Adding 10 to number...
Number = 20
Subtracting 10 from number...

Number = 10
```

در برنامه از ۳ عملگر تخصیصی استفاده شده است. ابتدا یک متغیر و مقدار 10 با استفاده از عملگر = به آن اختصاص داده شده است.سپس به آن با استفاده از عملگر =+ مقدار ۱۰ اضافه و در آخر به وسیله عملگر =- عدد ۱۰ از آن کم شده است.

عملگرهای مقایسه ای

از عملگرهای مقایسه ای برای مقایسه مقادیر استفاده می‌شود.نتیجه این مقادیر یک مقدار بولی(منطقی) است.این عملگرها اگر نتیجه مقایسه دو مقدار درست باشد مقدار ۱ و اگر نتیجه مقایسه اشتباه باشد مقدار ۰ را نشان می‌دهند. این عملگرها به طور

معمول در دستورات شرطی به کار می‌روند به این ترتیب که باعث ادامه یا توقف دستور شرطی می‌شوند.جدول زیر عملگرهای مقایسه ای در C++ را نشان می‌دهد:

عملگر	دسته	مثال	نتیجه
==	Binary	var1 = var2 == var3	var1در صورتی ۱ است که مقدار var2 با مقدار var3 برابر باشد در غیر اینصورت ۰ است
!=	Binary	var1 = var2 != var3	var1در صورتی ۱ است که مقدار var2 با مقدار var3 برابر نباشد در غیر اینصورت ۰ است
<	Binary	var1 = var2 < var3	var1در صورتی ۱ است که مقدار var2 کوچک‌تر از var3 مقدار باشد در غیر اینصورت ۰ است
>	Binary	var1 = var2 > var3	var1در صورتی ۱ است که مقدارvar2 بزرگ‌تر ازمقدار var3باشد در غیر اینصورت ۰ است
<=	Binary	var1 = var2 <= var3	var1در صورتی ۱ است که مقدار var2 کوچک‌تر یا مساوی مقدار var3باشد در غیر اینصورت ۰ است
>=	Binary	var1 = var2 >= var3	var1در صورتی ۱ است که مقدار var2 بزرگ‌تر یا مساوی var3 مقدار باشد در غیر اینصورت ۰ است

برنامه زیر نحوه عملکرد این عملگرها را نشان می‌دهد:

```
#include <iostream>
using namespace std;

int main()
{
    int num1 = 10;
    int num2 = 5;

    cout << num1 << " == " << num2 << " : " << (num1 == num2) << endl;
    cout << num1 << " != " << num2 << " : " << (num1 != num2) << endl;
    cout << num1 << " <  " << num2 << " : " << (num1 < num2) << endl;
    cout << num1 << " >  " << num2 << " : " << (num1 > num2) << endl;
    cout << num1 << " <= " << num2 << " : " << (num1 <= num2) << endl;
    cout << num1 << " >= " << num2 << " : " << (num1 >= num2) << endl;
}
```

```
10 == 5 : 0
```

```
10 != 5 : 1
10 <  5 : 0
10 >  5 : 1
10 <= 5 : 0
10 >= 5 : 1
```

در مثال بالا ابتدا دو متغیر را که می‌خواهیم با هم مقایسه کنیم را ایجاد کرده و به آن‌ها مقادیری اختصاص می‌دهیم.سپس با استفاده از یک عملگر مقایسه ای آن‌ها را با هم مقایسه کرده و نتیجه را چاپ می‌کنیم.به این نکته توجه کنید که هنگام مقایسه دو متغیر از عملگر == به جای عملگر = باید استفاده شود. عملگر = عملگر تخصیصی است و در عبارتی مانند x = y مقدار y را در به x اختصاص می‌دهد. عملگر == عملگر مقایسه ای است که دو مقدار را با هم مقایسه می‌کند مانند x==y و اینطور خوانده می‌شود x برابر است با. y

عملگرهای منطقی

عملگرهای منطقی بر روی عبارات منطقی عمل می‌کنند و نتیجه آن‌ها نیز یک مقدار بولی است. از این عملگرها اغلب برای شرطهای پیچیده استفاده می‌شود. همانطور که قبلاً یاد گرفتید مقادیر بولی می‌توانند false یا true باشند. فرض کنید که var2 و var3دو مقدار بولی هستند.

عملگر	نام	دسته	مثال
&&	منطقی AND	Binary	var1 = var2 && var3;
||	منطقی OR	Binary	var1 = var2 || var3;
!	منطقی NOT	Unary	var1 = !var1;

عملگر منطقی (&&)AND

اگر مقادیر دو طرف عملگر AND ، trueباشند عملگر AND مقدار true را بر می‌گرداند. در غیر اینصورت اگر یکی از مقادیر یا هر دوی آن‌ها false باشند مقدار false را بر می‌گرداند. در زیر جدول درستی عملگر AND نشان داده شده است:

X && Y	Y	X
true	true	true
false	false	true

false	true	false
false	false	false

برای درک بهتر تأثیر عملگر AND یاد آوری می‌کنم که این عملگر فقط در صورتی مقدار true را نشان می‌دهد که هر دو عملوند مقدارشان true باشد.در غیر اینصورت نتیجه تمام ترکیب‌های بعدی false خواهد شد. استفاده از عملگر ANDمانند استفاده از عملگرهای مقایسه ای است. به عنوان مثال نتیجه عبارت زیر درست (true) است اگر سن (age) بزرگ‌تر از ۱۸ و salary کوچک‌تر از ۱۰۰۰ باشد.

```
result = (age > 18) && (salary < 1000);
```

عملگر AND زمانی کارامد است که ما با محدود خاصی از اعداد سرو کار داریم. مثلاً عبارت 100 => x => 10بدین معنی است که x می‌تواند مقداری شامل اعداد ۱۰ تا ۱۰۰ را بگیرد. حال برای انتخاب اعداد خارج از این محدوده می‌توان از عملگر منطقی ANDبه صورت زیر استفاده کرد.

```
inRange = (number <= 10) && (number >= 100);
```

عملگر منطقی (||)OR

اگر یکی یا هر دو مقدار دو طرف عملگر OR ،درست (true) باشد،عملگر OR مقدار true را بر می‌گرداند. جدول درستی عملگر ORدر زیر نشان داده شده است :

X	Y	X || Y
true	true	true
true	false	true
false	true	true
false	false	false

در جدول بالا مشاهده می‌کنید که عملگر OR در صورتی مقدار false را بر می‌گرداند که مقادیر دو طرف آن false باشند. کد زیر را در نظر بگیرید.نتیجه این کد در صورتی درست (true) است که رتبه نهایی دانش آموز (finalGrade) بزرگ‌تر از ۷۵ یا یا نمره نهایی امتحان آن ۱۰۰ باشد.

```
isPassed = (finalGrade >= 75) || (finalExam == 100);
```

عملگر منطقی (!) NOT

برخلاف دو اپراتور OR و AND عملگر منطقی NOT یک عملگر یگانی است و فقط به یک عملوند نیاز دارد. این عملگر یک مقدار یا اصطلاح بولی را نفی می‌کند .مثلاً اگر عبارت یا مقدار true باشد آنرا false و اگر false باشد آنرا true می‌کند. جدول زیر عملکرد اپراتور NOT را نشان می‌دهد:

X	!X
true	false
false	true

نتیجه کد زیر در صورتی درست است که) age سن) بزرگ‌تر یا مساوی ۱۸ نباشد.

```
isMinor = !(age >= 18);
```

عملگرهای بیتی

عملگرهای بیتی به شما اجازه می‌دهند که شکل باینری انواع داده‌ها را دستکاری کنید. برای درک بهتر این درس توصیه می‌شود که شما سیستم باینری و نحوه تبدیل اعداد دهدهی به باینری را از لینک زیر یاد بگیرید :

```
http://www.w3-farsi.com/?p=5698
```

در سیستم باینری (دودویی) که کامپیوتر از آن استفاده می‌کند وضعیت هر چیز یا خاموش است یا روشن. برای نشان دادن حالت روشن از عدد ۱ و برای نشان دادن حالت خاموش از عدد ۰ استفاده می‌شود. بنابراین اعداد باینری فقط می‌توانند صفر یا یک باشند.اعداد باینری را اعداد در مبنای ۲ و اعداد اعشاری را اعداد در مبنای ۱۰ می گویند. یک بیت نشان دهنده یک رقم باینری است و هر بایت نشان دهنده ۸ بیت است. به عنوان مثال برای یک داده از نوع int به ۳۲ بیت یا ٤ بایت فضا برای ذخیره آن نیاز داریم، این بدین معناست که اعداد از ۳۲ رقم ۰ و ۱ برای ذخیره استفاده می‌کنند. برای مثال عدد ۱۰۰ وقتی به عنوان یک متغیر از نوع int ذخیره می‌شود در کامپیوتر به صورت زیر خوانده می‌شود:

```
00000000000000000000000001100100
```

عدد ۱۰۰ در مبنای ده معادل عدد ۱۱۰۰۱۰۰ در مبنای ۲ است. در اینجا ۷ رقم سمت راست نشان دهنده عدد ۱۰۰ در مبنای ۲ است و مابقی صفرهای سمت راست برای پر کردن بیت‌هایی است که عدد از نوع int نیاز دارد. به این نکته توجه کنید که اعداد باینری از سمت راست به چپ خوانده می‌شوند. عملگرهای بیتی C++ در جدول زیر نشان داده شده‌اند:

عملگر	نام	دسته	مثال
&	بیتیAND	Binary	x = y & z;
|	بیتیOR	Binary	x = y | z;
^	بیتیXOR	Binary	x = y ^ z;
~	بیتیNOT	Unary	x = ~y;
&=	بیتیAND Assignment	Binary	x &= y;
|=	بیتیOR Assignment	Binary	x |= y;
^=	بیتیXOR Assignment	Binary	x ^= y;

عملگر بیتی AND(&)

عملگر بیتی AND کاری شبیه عملگر منطقی AND انجام می‌دهد با این تفاوت که این عملگر بر روی بیت‌ها کار می‌کند. اگر مقادیر دو طرف آن ۱ باشد مقدار ۱ را بر می‌گرداند و اگر یکی یا هر دو طرف آن صفر باشد مقدار صفر را بر می‌گرداند. جدول درستی عمگر بیتی AND در زیر آمده است:

X	Y	X AND Y
1	1	1
1	0	0
0	1	0
0	0	0

در زیر نحوه استفاده از عملگر بیتی AND آمده است:

```
int result = 5 & 3;
cout << result;
```

```
1
```

همانطور که در مثال بالا مشاهده می‌کنید نتیجه عملکرد عملگر AND بر روی دو مقدار ۵ و ۳ عدد ۱ می‌شود. اجازه بدهید ببینیم که چطور این نتیجه را به دست می‌آید:

```
5:  00000000000000000000000000000101
3:  00000000000000000000000000000011
---------------------------------
1:  00000000000000000000000000000001
```

ابتدا دو عدد ۵ و ۳ به معادل باینری‌شان تبدیل می‌شوند. از آنجاییکه هر عدد صحیح 32 (int)بیت است از صفر برای پر کردن بیت‌های خالی استفاده می‌کنیم. با استفاده از جدول درستی عملگر بیتی AND می‌توان فهمید که چرا نتیجه عدد یک می‌شود.

عملگر بیتی OR(|)

اگر مقادیر دو طرف عملگر بیتی OR هر دو صفر باشند نتیجه صفر در غیر اینصورت ۱ خواهد شد. جدول درستی این عملگر در زیر آمده است:

X	Y	X OR Y
1	1	1
1	0	1
0	1	1
0	0	0

نتیجه عملگر بیتی OR در صورتی صفر است که عملوندهای دو طرف آن صفر باشند. اگر فقط یکی از دو عملوند یک باشد نتیجه یک خواهد شد. به مثال زیر توجه کنید:

```
int result = 7 | 9;

cout << result;
```

```
15
```

وقتی که از عملگر بیتی OR برای دو مقدار در مثال بالا (۷ و ۹) استفاده می‌کنیم نتیجه ۱۵ می‌شود. حال بررسی می‌کنیم که چرا این نتیجه به دست آمده است؟

```
 7: 00000000000000000000000000000111
 9: 00000000000000000000000000001001
---------------------------------
15: 00000000000000000000000000001111
```

با استفاده از جدول درستی عملگر بیتی OR می‌توان نتیجه استفاده از این عملگر را تشخیص داد. عدد ۱۱۱۱ باینری معادل عدد ۱۵ صحیح است.

عملگر بیتی XOR(^)

جدول درستی این عملگر در زیر آمده است:

X	Y	X XOR Y
1	1	0
1	0	1
0	1	1
0	0	0

در صورتیکه عملوندهای دو طرف این عملگر هر دو صفر یا هر دو یک باشند نتیجه صفر در غیر اینصورت نتیجه یک می‌شود. در مثال زیر تأثیر عملگر بیتی XORرا بر روی دو مقدار مشاده می‌کنید:

```
int result = 5 ^ 7;

cout << result;
```

```
2
```

در زیر معادل باینری اعداد بالا (۵ و ۷) نشان داده شده است.

```
5: 00000000000000000000000000000101
7: 00000000000000000000000000000111
-----------------------------------
2: 00000000000000000000000000000010
```

با نگاه کردن به جدول درستی عملگر بیتیXOR ، می‌توان فهمید که چرا نتیجه عدد ۲ می‌شود.

عملگر بیتی NOT(~)

این عملگر یک عملگر یگانی است و فقط به یک عملوند نیاز دارد. در زیر جدول درستی این عملگر آمده است:

X	NOT X
1	0

1	0

عملگر بیتی NOT مقادیر بیت‌ها را معکوس می‌کند. در زیر چگونگی استفاده از این عملگر آمده است:

```
int result = ~7;

cout << result;
```

```
-8
```

به نمایش باینری مثال بالا که در زیر نشان داده شده است توجه نمایید.

```
 7: 00000000000000000000000000000111
----------------------------------
-8: 11111111111111111111111111111000
```

عملگر بیتی تغییر مکان(shift)

این نوع عملگرها به شما اجازه می‌دهند که بیت‌ها را به سمت چپ یا راست جا به جا کنید. دو نوع عملگر بیتی تغییر مکان وجود دارد که هر کدام دو عملوند قبول می‌کنند. عملوند سمت چپ این عملگرها حالت باینری یک مقدار و عملوند سمت راست تعداد جابه جایی بیت‌ها را نشان می‌دهد.

عملگر	نام	دسته	مثال
>>	تغییر مکان به سمت چپ	Binary	x = y << 2;
<<	تغییر مکان به سمت راست	Binary	x = y >> 2;

عملگر تغییر مکان به سمت چپ

این عملگر بیت‌های عملوند سمت چپ را به تعداد n مکان مشخص شده توسط عملوند سمت راست، به سمت چپ منتقل می‌کند.

به عنوان مثال:

```
int result = 10 << 2;

cout << result;
```

```
40
```

در مثال بالا ما بیت‌های مقدار ۱۰ را دو مکان به سمت چپ منتقل کرده‌ایم، حال بیایید تأثیر این انتقال را بررسی کنیم:

```
10: 00000000000000000000000000001010
--------------------------------
40: 00000000000000000000000000101000
```

مشاهده می‌کنید که همه بیت‌ها به اندازه دو واحد به سمت چپ منتقل شده‌اند. در این انتقال دو صفر از صفرهای سمت چپ کم می‌شود و در عوض دو صفر به سمت راست اضافه می‌شود.

عملگر تغییر مکان به سمت راست

این عملگر شبیه به عمگر تغییر مکان به سمت چپ است با این تفاوت که بیت‌ها را به سمت راست جا به جا می‌کند. به عنوان مثال:

```
int result = 100 >> 4;

cout << result;
```

```
6
```

با استفاده از عملگرتغییر مکان به سمت راست بیت‌های مقدار ۱۰۰ را به اندازه ٤ واحد به سمت چپ جا به جا می‌کنیم. اجازه بدهید تأثیر این جا به جایی را مورد بررسی قرار دهیم:

```
100: 00000000000000000000000001100100
--------------------------------
  6: 00000000000000000000000000000110
```

هر بیت به اندازه ٤ واحد به سمت راست منتقل می‌شود،بنابراین ٤ بیت اول سمت راست حذف شده و چهار صفر به سمت چپ اضافه می‌شود.

تقدم عملگرها

تقدم عملگرها مشخص می‌کند که در محاسباتی که بیش از دو عملوند دارند ابتدا کدام عملگر اثرش را اعمال کند. عملگرها در C++ در محاسبات دارای حق تقدم هستند.به عنوان مثال:

```
number = 1 + 2 * 3 / 1;
```

اگر ما حق تقدم عملگرها را رعایت نکنیم و عبارت بالا را از سمت چپ به راست انجام دهیم نتیجه ۹ خواهد شد (۳=۲+۱ سپس ۹=۳×۳ و در آخر ۹=۹/۱).اما کامپایلر با توجه به تقدم عملگرها محاسبات را انجام می‌دهد.برای مثال عمل ضرب و تقسیم نسبت به

جمع و تفریق تقدم دارند.بنابراین در مثال فوق ابتدا عدد ۲ ضربدر ۳ و سپس نتیجه آن‌ها تقسیم بر ۱ می‌شود که نتیجه ٦ به دست می‌آید.در آخر عدد ٦ با ۱ جمع می‌شود و عدد ۷ حاصل می‌شود. در جدول زیر تقدم عملگرهای C++ از بالا به پایین آمده است:

Level	Precedence group	Operator	Grouping
1	Scope	::	Left-to-right
2	Postfix (unary)	++ -	Left-to-right
		()	
		[]	
		. ->	
3	Prefix (unary)	++ -	Right-to-left
		~ !	
		+ -	
		& *	
		new delete	
		sizeof	
		(*type*)	
4	Pointer-to-member	.* ->*	Left-to-right
5	Aritdmetic: scaling	* / %	Left-to-right
6	Aritdmetic: addition	+ -	Left-to-right
7	Bitwise shift	<< >>	Left-to-right
8	Relational	< > <= >=	Left-to-right
9	Equality	== !=	Left-to-right
10	And	&	Left-to-right
11	Exclusive or	^	Left-to-right
12	Inclusive or	|	Left-to-right
13	Conjunction	&&	Left-to-right
14	Disjunction	||	Left-to-right
15	Assignment-level expressions	= *= /= %= += -= >>= <<= &= ^= |=	Right-to-left

		?:	
16	Sequencing	,	Left-to-right

ابتدا عملگرهای با بالاترین و سپس عملگرهای با پایین‌ترین حق تقدم در محاسبات تأثیر می‌گذارند.به این نکته توجه کنید که تقدم عملگرها ++و – به مکان قرارگیری آن‌ها بستگی دارد (در سمت چپ یا راست عملوند باشند). به عنوان مثال:

```
int number = 3;

number1 = 3 + ++number; //results to 7
number2 = 3 + number++; //results to 6
```

در عبارت اول ابتدا به مقدارnumber یک واحد اضافه شده و ٤ می‌شود و سپس مقدار جدید با عدد ٣ جمع می‌شود و در نهایت عدد ٧ به دست می‌آید.در عبارت دوم مقدار عددی ٣ به مقدار number اضافه می‌شود و عدد ٦ به دست می‌آید.سپس این مقدار در متغیر number2 قرار می‌گیرد. و در نهایت مقدار number به ٤ افزایش می‌یابد. برای ایجاد خوانایی در تقدم عملگرها و انجام محاسباتی که در آن‌ها از عملگرهای زیادی استفاده می‌شود از پرانتز استفاده می‌کنیم:

```
number = ( 1 + 2 ) * ( 3 / 4 ) % ( 5 - ( 6 * 7 ));
```

در مثال بالا ابتدا هر کدام از عباراتی که داخل پرانتز هستند مورد محاسبه قرار می‌گیرند.به نکته ای در مورد عبارتی که در داخل پرانتز سوم قرار دارد توجه کنید.در این عبارت ابتدا مقدار داخلی‌ترین پرانتز مورد محاسبه قرار می‌گیرد یعنی مقدار ٦ ضربدر ٧ شده و سپس از ٥ کم می‌شود. اگر دو یا چند عملگر با حق تقدم یکسان موجود باشد ابتدا باید هر کدام از عملگرها را که در ابتدای عبارت می‌آیند مورد ارزیابی قرار دهید. به عنوان مثال:

```
number = 3 * 2 + 8 / 4;
```

هر دو عملگر * و / دارای حق تقدم یکسانی هستند. بنابر این شما باید از چپ به راست آن‌ها را در محاسبات تأثیر دهید.یعنی ابتدا ٣ را ضربدر ٢ می‌کنید و سپس عدد ٨ را بر ٤ تقسیم می‌کنید. در نهایت نتیجه دو عبارت را جمع کرده و در متغیر number قرار می‌دهید.

گرفتن ورودی از کاربر

سی پلاس پلاس دارای تعدادی شیء و متد برای گرفتن ورودی از کاربر می‌باشد. حال می‌خواهیم درباره cin یکی دیگر از اشیاء کلاس Istraem بحث کنیم که یک مقدار را از کاربر دریافت می‌کند. کار cin این است که تمام کاراکترهایی را که شما در محیط کنسول تایپ می‌کنید تا زمانی که دکمه Enter را می‌زنید می‌خواند. به برنامه زیر توجه کنید:

```
#include <iostream>
#include <string>

using namespace std;

int main()
{
	string name;
	int age;
	double height;

	cout << "Enter your name: ";
	cin >> name;
	cout << "Enter your age: ";
	cin >> age;
	cout << "Enter your height: ";
	cin >> height;

	//Print a blank line
	cout << endl;

	//Show the details you typed
	cout << "Name is " << name << endl;
	cout << "Age is " << age << endl;
	cout << "Height is " << height << endl;
}
```

```
Enter your name: John
Enter your age: 18
Enter your height: 160.5

Name is John.
Age is 18.
Height is 160.5.
```

ابتدا ۳ متغیر را برای ذخیره داده در برنامه تعریف می‌کنیم (خطوط ۸ و ۹ و ۱۰). برنامه از کاربر می‌خواهد که نام خود را وارد کند (خط ۱۲). در خط ۱۳ شما به عنوان کاربر نام خود را وارد می‌کنید. مقدار متغیر نام، برابر مقداری است که توسط cin خوانده می‌شود.از آنجاییکه نام از نوع رشته است باید کتابخانه مربوط به رشته‌ها را در ابتدای برنامه وارد کنیم:

```
#include <string>
```

سپس برنامه از ما سن را سؤال می‌کند(خط ١٤). آن را در خط ۱۵ وارد کرده و در نهایت در خط ١٦ و ۱۷ هم قد را وارد می‌کنیم.

ساختارهای تصمیم