مرجع کامل
برنامه‌نویسی C++

تألیف:

دکتر جواد وحیدی (عضو هیات علمی دانشگاه علم و صنعت ایران)

دکتر رمضان عباس نژادورزی

فن‌آوری نوین

شماره کتابشناسی ملی	:	ایران ۷۵٦۵۷٦۰
شابک	:	۹۷۸-٦۲۲-۷۳۹۳-۲۷-۹
سرشناسه	:	وحیدی، جواد، ۱۳٤۸ - Vahidi, Javad
عنوان و نام پدیدآور	:	مرجع کامل برنامه‌نویسی C++[منابع الکترونیکی: کتاب]/تالیف جواد وحیدی، رمضان عباس‌نژادورزی.
مشخصات نشر	:	بابل: فناوری نوین، ۱۳۹۹.
مشخصات ظاهری	:	۱ منبع برخط(۵۷۸ ص.).: مصور(رنگی)، جدول.
وضعیت فهرست نویسی	:	فیپا
نوع منبع الکترونیکی	:	فایل متنی (PDF).
یادداشت	:	دسترسی از طریق وب.
شناسه افزوده	:	عباس‌نژاد ورزی، رمضان، ۱۳٤۸ -
موضوع	:	سی ++ (زبان برنامه‌نویسی کامپیوتر)
موضوع	:	C++ (Computer program language)
رده بندی کنگره	:	QA۷٦/۷۳
رده بندی دیویی	:	۰۰۵/۱۳۳
دسترسی و محل الکترونیکی	:	آدرس الکترونیکی منبع

فن‌آوری نوین بابل، کد پستی ٤۷۱٦۷-۷۳٤٤۸ تلفن: ۳۲۲۵٦٦۸۷-۰۱۱

مرجع کامل برنامه‌نویسی C++

تألیف: جواد وحیدی، رمضان عباس‌نژادورزی

نوبت چاپ: چاپ اول

سال چاپ: زمستان ۹۹

شمارگان: ۲۰۰

قیمت: ۱۳۰۰۰۰ تومان

نام چاپخانه و صحافی: دفتر فنی سورنا

شابک: ۹۷۸-٦۲۲-۷۳۹۳-۲۷-۹

نشانی ناشر: بابل، چهارراه نواب، کاظم‌بیگی، جنب مسجد منصور کاظم‌بیگی، طبقه اول

طراح جلد: کانون آگهی و تبلیغات آبان (احمد فرجی)

فروشگاه و پخش کتاب چاپی: تهران، تلفن ۶۶۴۰۰۲۲۰- ۶۶۴۰۰۱۴۴

تهران، خ اردیبهشت، نبش وحید نظری، پلاک ۱٤۲ تلفکس: ٦٦٤۰۰۱٤٤-٦٦٤۰۰۲۲۰

فهرست مطالب

مقدمه

هنوز هم ماه وامدار خورشید است
و ناامیدی
هویتش را از امید می‌گیرد
و این داستان ادامه دارد...

برنامه‌نویسی کامپیوتر فرآیند نوشتن، **اشکال‌زدایی**[1] و **نگه‌داری کد منبع**[2] برنامه کامپیوتر می‌باشد. این کد منبع با یک زبان برنامه‌نویسی نوشته‌شده است و ممکن است تغییر داده‌شده یک کد قبلی و یا یک کد کاملاً جدید باشد. هدف برنامه‌نویسی ساختن یک برنامه می‌باشد که یک رفتار خواسته‌شده را به نمایش بگذارد.

یکی از دغدغه‌های همیشگی اغلب دانشجویان رشته‌های علوم پایه، فنی مهندسی و بخصوص دانشجویان رشته کامپیوتر این است که، چگونه می‌توانیم یک برنامه‌نویس خوب و حرفه‌ای شویم ؟. برخی از آنان بر این باورند که این کار اصلاً امکان‌پذیر نیست و یا می‌پندارند که آن‌ها برای این کار ساخته نشده‌اند.

برنامه‌نویسی نیز مانند آموزش زبان ،فرایندی زمان‌بر است و نمی‌توانیم در فاصله زمانی اندک انتظار زیادی از خودمان داشته باشیم،درنهایت باید از یک نقطه آغاز کرد . اما نکته حائز اهمیت آن است که :

آب این رود به سرچشمه نمی‌گردد باز بهتر آن است که غفلت نکنیم از آغاز

یکی از عناصر بنیادی برای شروع درست یک درس برنامه‌نویسی، استفاده از یک منبع مناسب است. نگارش کتاب حاضر نیز به همین منظور صورت پذیرفته است.مطالب این کتاب به‌گونه‌ای تنظیم‌شده است که خواننده، با یک ترتیب منطقی به دنیای برنامه‌نویسی قدم بگذارد و با پی گیری کتاب تا انتها بدون آن که فشار زیادی را متحمل شود به یک برنامه‌نویس خوب C++ تبدیل شود.

نکته آخر این‌که این روزها همه حرف از گرانی می‌زنند. از گرانی اقلام خوراکی گرفته تا گرانی پوشاک و مسکن و اتومبیل. در این میان حوزه فرهنگ نیز از این گرانی‌ها بی‌نصیب نمانده است. گرانی کاغذ و اقلام موردنیاز نشر نیز یکی از معضلاتی است که این روزها پایش را روی گلوی یکی از مهم‌ترین عرصه‌های فرهنگی یعنی عرصه نشر گذاشته است؛ تا جایی که شماری از ناشران پای خود را از این عرصه

---

[1] - debug
[2] - source code

عقب کشیده‌اند. با تمام مشکلاتی که در حوزه چاپ و نشر وجود دارد انتشارات فن‌آوری نوین مانند بسیاری از فعالین این عرصه کماکان پیش می‌رود تا آینده بهتری برای کشور عزیزمان رقم زند.

از تمامی اساتید و دانشجویان عزیز تقاضا داریم، هرگونه اشکال، ابهام در متن کتاب، پیشنهاد و انتقادات را به آدرس پست الکترونیک **fanavarienovin@gmail.com** ارسال نمایند.

در پایان امیدوارم این اثر موردتوجه جامعه انفورماتیک کشور، اساتید و دانشجویان عزیز قرار گیرد.

fanavarienovin@gmail.com

مؤلفین - اسفند ۱۳۹۹

فصل ۱ آشنایی با زبان C++

زبان‌های برنامه‌سازی متعددی وجود دارند. یکی از این زبان‌های برنامه‌سازی، c++ می‌باشد. این زبان در سال ۱۹۸۰ توسط بیارنی استراستاپ[۱] در آزمایشگاه بل ابداع گردید. زبان c++ توسعه‌یافته زبان c است. (هر فردی که با کامپیوتر برنامه‌نویسی می‌کند با زبان c آشنا است). زبان c یک زبان برنامه‌سازی ساخت‌یافته است. اما، زبان c++ از شیوه شیءگرایی برای نوشتن برنامه‌ها استفاده می‌کند.

۱ – ۱. سطوح مختلف زبان‌های برنامه‌سازی

زبان‌های برنامه‌سازی با توجه به امکانات و پیچیدگی به سه سطح زیر تقسیم می‌شوند.

۱. سطح پایین
۲. سطح بالا
۳. سطح میانی

۱ – ۱ – ۱. زبان‌های سطح پایین

زبان‌های سطح پایین، همان زبان ماشین نام دارند. در این زبان‌ها، برنامه به‌صورت ۰ یا ۱ (زبان واقعی کامپیوتر) نوشته می‌شود. زیرا، کامپیوتر فقط ۰ یا ۱ را می‌فهمد (شکل ۱ –۱). برنامه نوشته‌شده به این زبان دارای سرعت بالا است. اما، درک و فهم آن برای انسان‌ها بسیار مشکل می‌باشد. به همین دلیل، زبان اسمبلی را ایجاد کردند. زبان اسمبلی به‌جای ۰ یا ۱ از یک‌سری نمادها تشکیل‌شده است.

قطعه برنامه زیر اعداد فرد ۱ تا ۱۰ را جمع کرده، در ثبات AX قرار می‌دهد.

```
  Mov  AX, ۰
  Mov BX, ۱
L۱: Cmp CX, ۱۰
  JG    Exit
  ADD  AX, BX
  ADD  BX, ۲
  JMP   L۱
Exit;
```

```
۱۰۰۰۰۱۱۱۱۰۱۰۰۱۱۱۱۰۱
۱۰۰۱۱۰۱۱۰۰۱۰۰۱۰۱۰۱۱
۱۰۰۱۱۰۱۰۰۱۱۰۰۱۱۰۰۱۱
۱۰۰۱۱۰۰۱۱۱۰۰۱۱۰۰۱۱۰
```

شکل ۱ – ۱ نمونه برنامه زبان ماشین.

همان‌طور که در این برنامه می‌بینید، نسبت به برنامه‌های زبان ماشین قابل فهم‌ترشده است. اما کامپیوتر نمی‌تواند آن را اجرا کند. برای این‌که کامپیوتر بتواند آن را اجرا نماید باید به زبان ماشین تبدیل شود، به همین دلیل اسمبلر نوشته شد. **اسمبلر**، برنامه‌ای است که یک برنامه نوشته‌شده به زبان اسمبلی را گرفته و به زبان ماشین (همان صفر یا یک) تبدیل می‌کند تا کامپیوتر بتواند آن را اجرا نماید (شکل ۲ – ۱).

---

۱. Biarne Stroustrup

شکل ۲ – ۱ وظیفه اسمبلر، تبدیل برنامه زبان اسمبلی به زبان ماشین.

همان‌طور که ملاحظه کردید، نوشتن برنامه‌های اسمبلی خیلی راحت‌تر از نوشتن برنامه‌های زبان ماشین می‌باشد (امروزه نوشتن برنامه‌ها به زبان ماشین (۰ یا ۱) وجود ندارد). بنابراین، زبان سطح پایین، همان زبان اسمبلی می‌باشد. از آنجائی که درک برنامه‌های زبان ماشین نیاز به اطلاعات کافی از سخت‌افزار کامپیوتر می‌باشد و درک آن نیز برای انسان‌ها مشکل است، زبان‌های سطح بالا را تولید کردند. در ادامه شرح این زبان‌ها را می‌بینید.

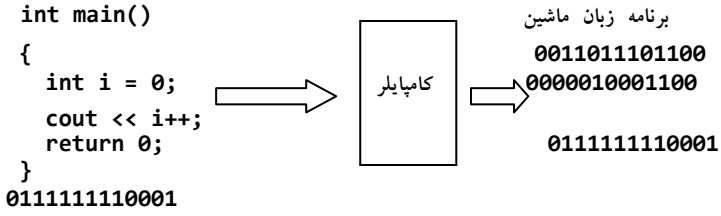
۲ – ۱ – ۱. زبان‌های سطح بالا

در زبان‌های سطح بالا، دستورات به زبان انسان نزدیک‌تر شده‌اند. به‌عنوان‌مثال، در این زبان‌ها می‌توان بیان کرد که اگر شرط خاصی درست باشد، این عمل را انجام بده، در غیر این صورت، عمل دیگر را انجام بده، این کار را ۳۰ بار تکرار کن و غیره.

همانند برنامه‌های اسمبلی، برنامه‌های نوشته‌شده به زبان سطح بالا باید به زبان ماشین تبدیل گردند تا توسط کامپیوتر قابل‌اجرا باشند. برای این منظور، مترجم‌ها نوشته‌شده‌اند. دو نوع مترجم وجود دارند که عبارت‌اند از:

۱. **کامپایلر (Compiler)،** کل برنامه به زبان سطح بالا را می‌گیرد، یک‌باره به زبان ماشین ترجمه می‌کند و یک برنامه کامل به زبان ماشین ایجاد می‌نماید. شکل ۳ –۱ فرآیند انجام ترجمه برنامه سطح بالا را نشان می‌دهد.

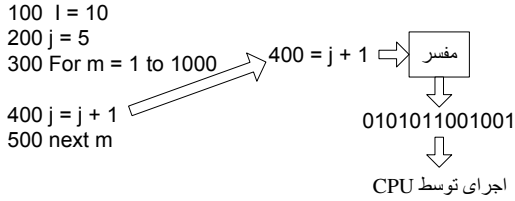
زبان‌هایی مانند C، پاسکال، کوبول، VC++، C#، ویژوال‌بیسیک، زبان‌های کامپایلری هستند.



شکل ۳ – ۱ فرآیند ترجمه با کامپایلر.

۲. **مفسر (Interpreter)**، به‌جای این‌که کل برنامه را به زبان ماشین تبدیل کند، در زمان اجرا، خط به خط دستورات برنامه را خوانده آن خط را به زبان ماشین تبدیل می‌کند و سپس، اجرا می‌نماید. در ادامه، خط بعدی را گرفته، ترجمه و اجرا می‌نماید و این روند را ادامه می‌دهد (شکل ۴ -۱). زبان‌هایی مثل جاوا و بیسیک به زبان مفسری معروف‌اند.

زبان‌های مفسری نسبت به زبان‌های مترجمی، سرعت پایین‌تر و کارایی کمتری دارند. اما، آموزش آن‌ها راحت‌تر بوده و قابلیت اجرای آن‌ها روی پلت‌فرم‌های مختلف بیشتر است.



شکل ۴ – ۱ فرآیند ترجمه زبان‌های مفسری.

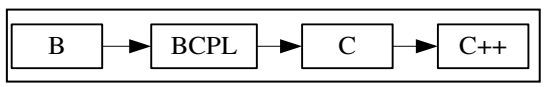
۳ – ۱ – ۱. زبان‌های سطح میانی

زبان‌های سطح میانی، زبان‌هایی هستند که دستورات زبان سطح بالا را با توابع اسمبلی با هم پیاده‌سازی کردند. نمونه‌ای این زبان می‌توان زبان C را نام برد. زبان‌های سطح میانی نیز برای ترجمه برنامه به زبان ماشین از کامپایلر استفاده می‌کنند.

۲ – ۱. ویژگی‌های زبان برنامه‌نویسی C++

زبان C در سال ۱۹۷۲ توسط دنیس ریچی[۲] نوشته شد. زبان C نیز توسعه‌یافته زبان BCPL است. زبان BCPL را مارتین ریچارد[۲] طراحی نمود. زبان‌های BCPL و C توسعه‌یافته B هستند. زبان B در سال ۱۹۷۰ توسط کن تامسون[۳] طراحی گردید.

این زبان‌ها از دسته زبان‌های ساخت‌یافته بودند. اما، زبان C++ در اوایل سال ۱۹۸۰ توسط بیارنی استراستاپ اختراع گردید. این زبان توسعه‌یافته زبان C می‌باشد. برخلاف زبان C در این زبان از روش برنامه‌نویسی شیءگرا (OOP)[۴] استفاده‌شده است. در ادامه این روش برنامه‌نویسی را می‌آموزیم. روند تکامل زبان‌ها در زیر آمده است.



۱. برخی از ویژگی‌های زبان C++ در زیر آمده است:

---

[۲]. Dennis Rich [۲]. Martin Richards [۳]. Ken Thompson [۴]. Object Oriented Program

۲. زبان C++ ، یک زبان سطح میانی است.
۳. زبان C++ از روش برنامه‌نویسی شیءگرا استفاده می‌کند.
٤. زبان C++ طوری طراحی‌شده است که قابل‌حمل می‌باشد.
٥. کلمات کلیدی (دستورات تشکیل‌دهنده) این زبان کم است.
٦. این زبان دارای کتابخانه‌های متعددی است. یعنی، توابع زیادی دارد. این توابع دسته‌بندی‌شده‌اند و برای منظور خاصی طراحی گردیدند. برنامه‌نویس در صورت نیاز می‌تواند کتابخانه مربوط به این توابع را به برنامه اضافه کرده، از توابع آن استفاده نماید.
۷. این زبان نسبت به حروف بزرگ و کوچک حساس است. یعنی، در نام‌گذاری این زبان بین حروف بزرگ و کوچک فرق قائل می‌شود.
۸. زبان C++ طوری طراحی‌شده است که می‌توان تمام برنامه‌های موردنیاز را با آن نوشت.
۹. زبان C++ اجازه می‌دهد که برنامه‌ها را به برنامه‌های کوچک‌تری که تابع نامیده می‌شود تقسیم کنید. این ویژگی‌ها سبب می‌شود که از تکنیک‌های محکمی مانند طراحی از بالا به پایین و مدولار استفاده کنید.

۳ – ۱. آموزش زبان‌های برنامه‌نویسی

آموزش زبان‌های برنامه‌نویسی مانند زبان‌های طبیعی زنده دنیا است. یعنی، برای آموزش زبان‌های برنامه‌نویسی باید مراحل زیر را انجام داد:

۱. مانند هر زبان طبیعی ابتدا باید علائم تشکیل‌دهنده زبان را آموخت. به‌عنوان‌مثال، زبان فارسی از علائم الف تا ی، ارقام ۰ تا ۹ و علائم خاص مانند ! ، : ، ؟ و غیره تشکیل‌شده است. هرکدام از این علائم (نمادها) مفهوم خاصی را دارند. زبان C++، نیز از علائم a تا z ، A تا Z ، ۰ تا ۹ ، علائم ویژه نظیر ; ، : ،]، [، / و غیره تشکیل‌شده است. ابتدا باید مفاهیم هریک از این علائم را در زبان C++ آموخت.
۲. همان‌طور که می‌دانید از ترکیب علائم هر زبان کلمات به وجود می‌آیند. برخی از کلمات دارای معنی و مفهوم هستند و برخی دیگر معنی و مفهوم خاصی ندارند. به‌عنوان‌مثال، کلمات بابا، آب، داد، در زبان فارسی مفهوم خاصی دارند. ولی کلمات تپتانم و بکیاپ مفهوم خاصی ندارند. به کلماتی که در زبان دارای مفهوم خاص هستند، کلمات کلیدی می‌گویند. در زبان C++ کلمات کلیدی نظیر for، if، else، while، int وجود دارند.

در آموزش این زبان ابتدا باید کلمات کلیدی را شناخت. معنی و کاربرد هرکدام از آن‌ها را آموخت.

۳. از ترکیب کلمات کلیدی با یک قواعد خاص در هر زبان طبیعی، جمله ایجاد می‌شود (مانند بابا آب داد). همان‌طور که می‌دانید در زبان فارسی ابتدا فاعل، سپس مفعول و در پایان فعل قرار می‌گیرد. در زبان

C++ نیز برای ایجاد جملات (دستورات) قواعد خاصی وجود دارد. به‌عنوان‌مثال، int برای تعریف داده‌های نوع صحیح به کار می‌رود و به‌صورت، زیر استفاده می‌گردد:

```
int     متغیر ۱ ,متغیر ۲ ,... ,متغیر n
```

٤. همان‌طور که می‌دانید، در زبان‌های طبیعی با کنار هم قرار دادن جملات مرتبط به هم پاراگراف ایجاد می‌شود. در زبان‌های برنامه‌نویسی نیز با کنار هم قرار دادن دستورات مرتبط به هم، بلاک ایجاد می‌شود. در زبان C++، هر بلاک با { شروع و با } خاتمه می‌یابد.

٥. کنار هم قرار دادن پاراگراف‌ها صفحات و فصول را ایجاد خواهند کرد و این روند ادامه می‌یابد تا یک کتاب نوشته شود. در زبان‌های برنامه‌سازی نیز نوشتن برنامه‌ها هم همین روند را دارد. کنار هم قرار دادن بلاک‌ها، فایل، و کنار هم قرار دادن فایل‌های مرتبط به هم، برنامه را ایجاد می‌کند.

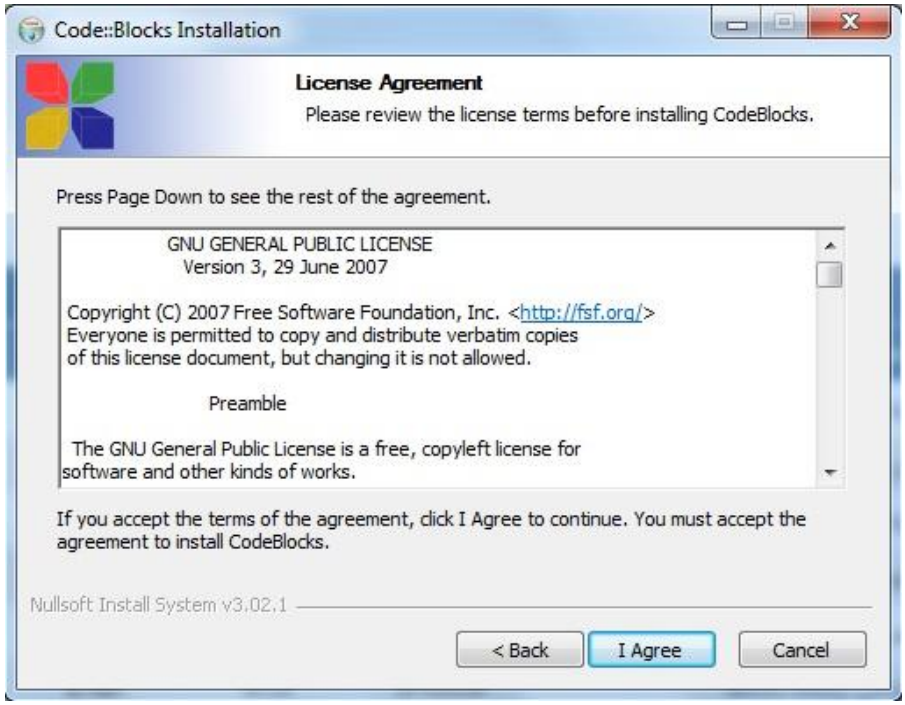
بنابراین، در ادامه کتاب به آموزش زبان C++ را با این شیوه خواهیم پرداخت.

۴–۱. مراحل نصب Code::Blocks

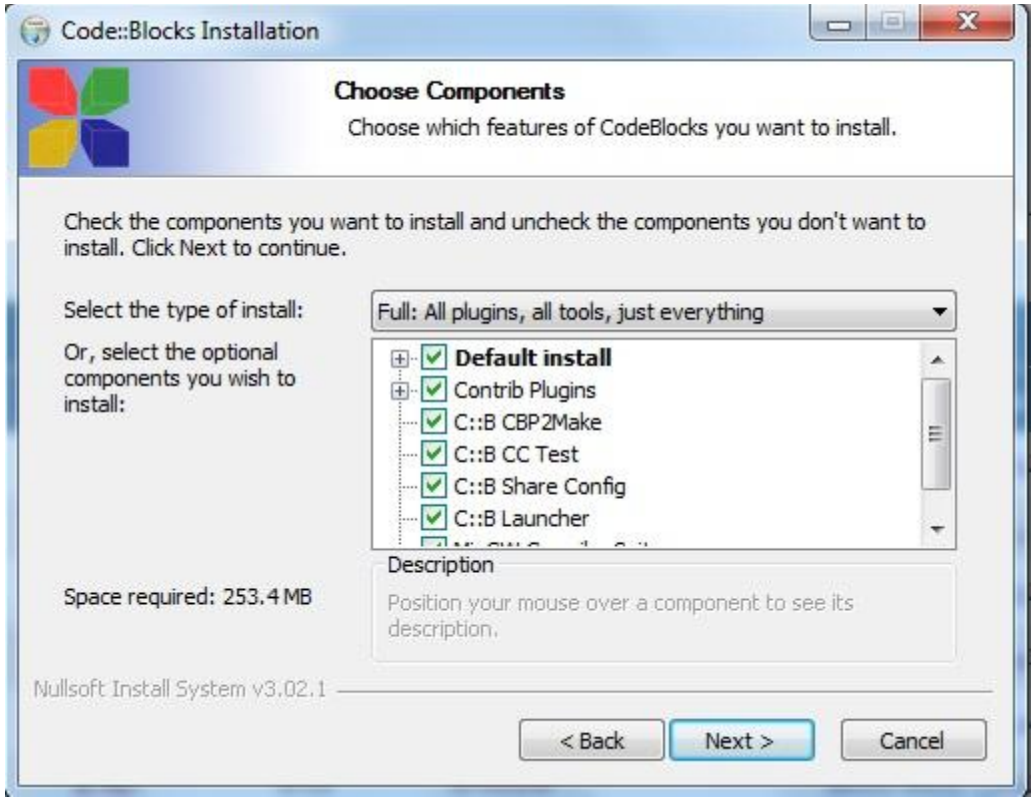
برای نصب Code::Blocks مراحل زیر را انجام دهید:

۱. بر روی فایل نصب Code::Blocks کلیک مضاعف کنید تا شکل زیر ظاهر گردد:

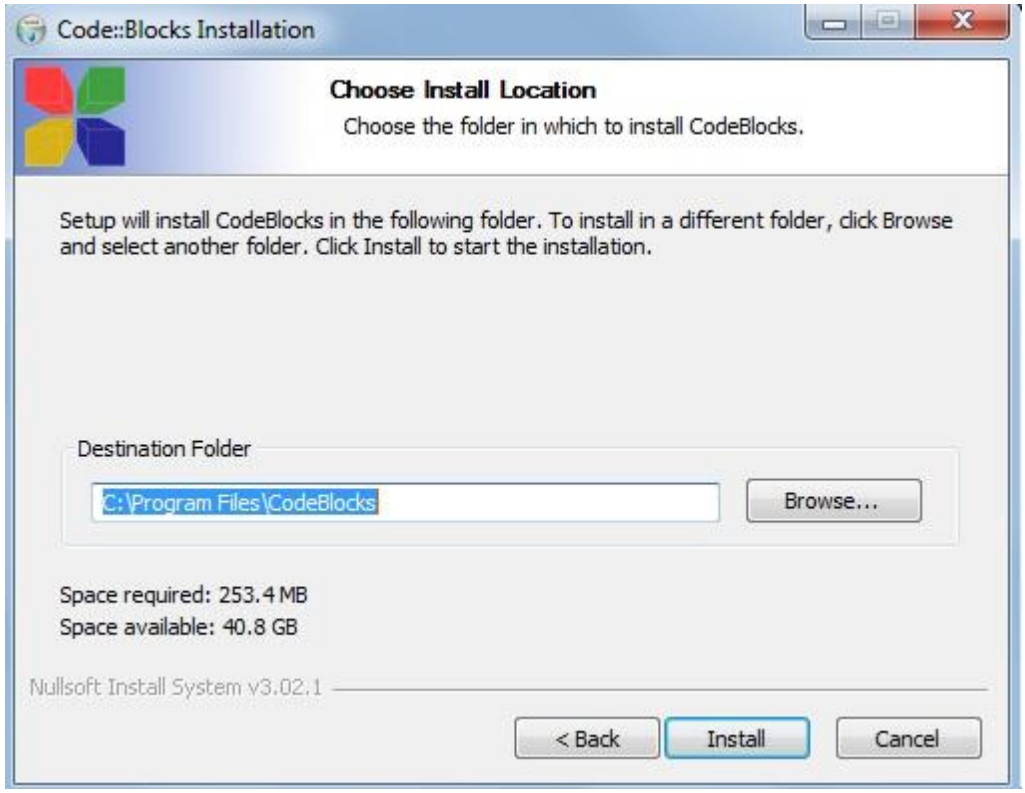
۲. در این شکل دکمه Next را کلیک نموده تا شکل زیر نمایش داده شود:



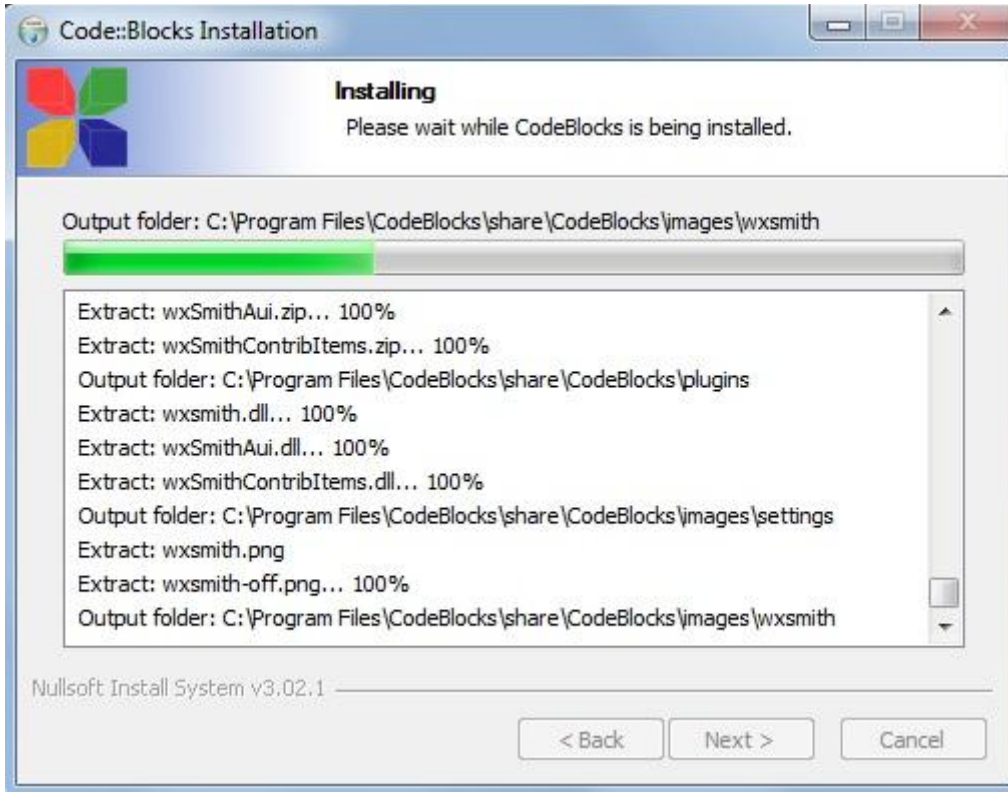
۳. در این شکل دکمه I Agree را کلیک نموده تا شکل زیر ظاهر شود:



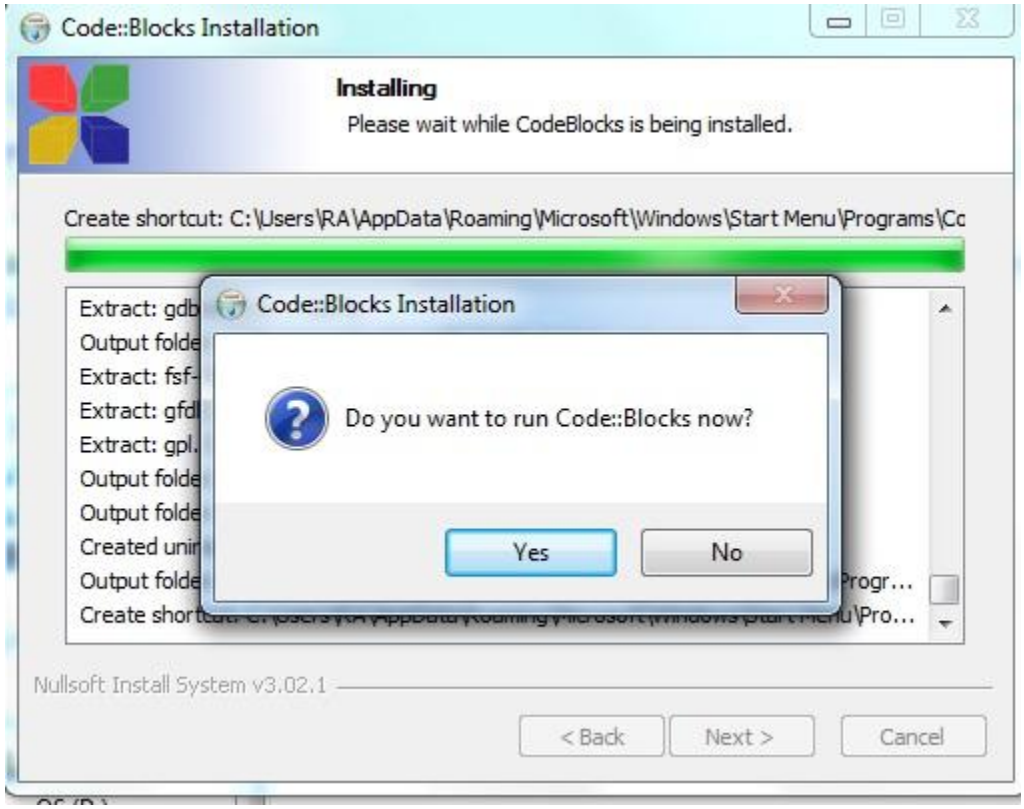
۴. در این شکل دکمه Next را کلیک نموده تا شکل زیر نمایش داده شود:



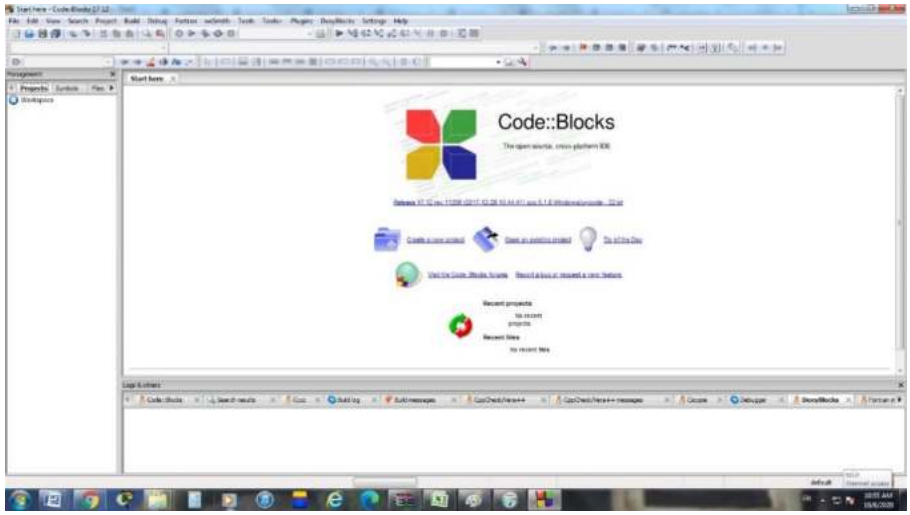
۵. در این شکل مسیر نصب را با دکمه Browse انتخاب نموده و دکمه Install را کلیک کنید تا شکل زیر ظاهر شود:



۶. چند لحظه منتظر بمانید تا شکل زیر ظاهر گردد:

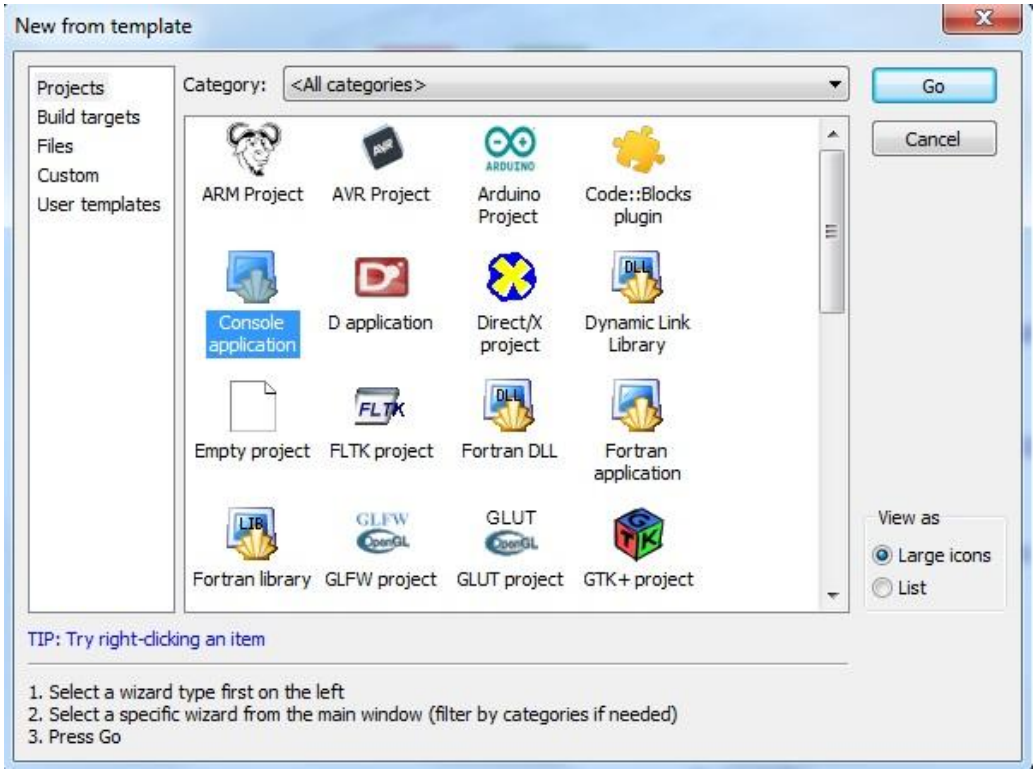


۷. اکنون نرم‌افزار Code::Blocks نصب‌شده است و می‌پرسد که آیا می‌خواهید Code::Blocks را اجرا کنید. دکمه Yes را کلیک کنید تا شکل زیر ظاهر گردد:

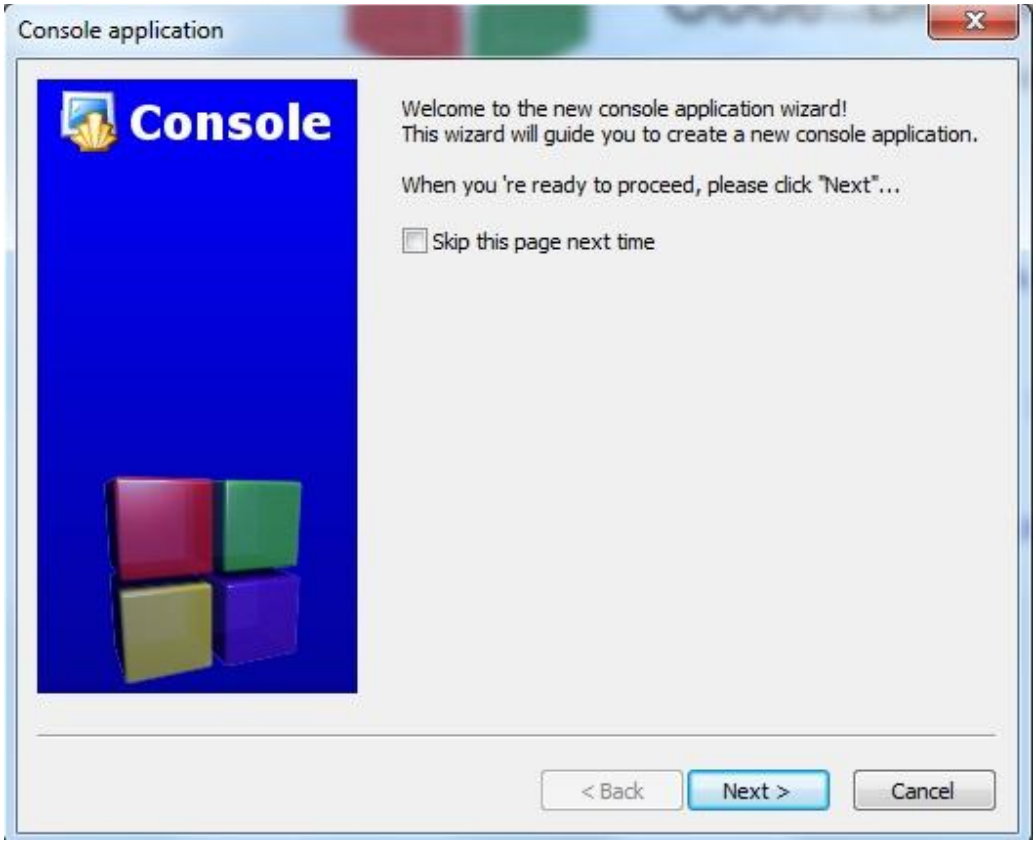


۸. صفحه اصلی Code::Blocks را مشاهده می‌کنید. برای ایجاد پروژه جدید آیکون زیر را کلیک کنید:

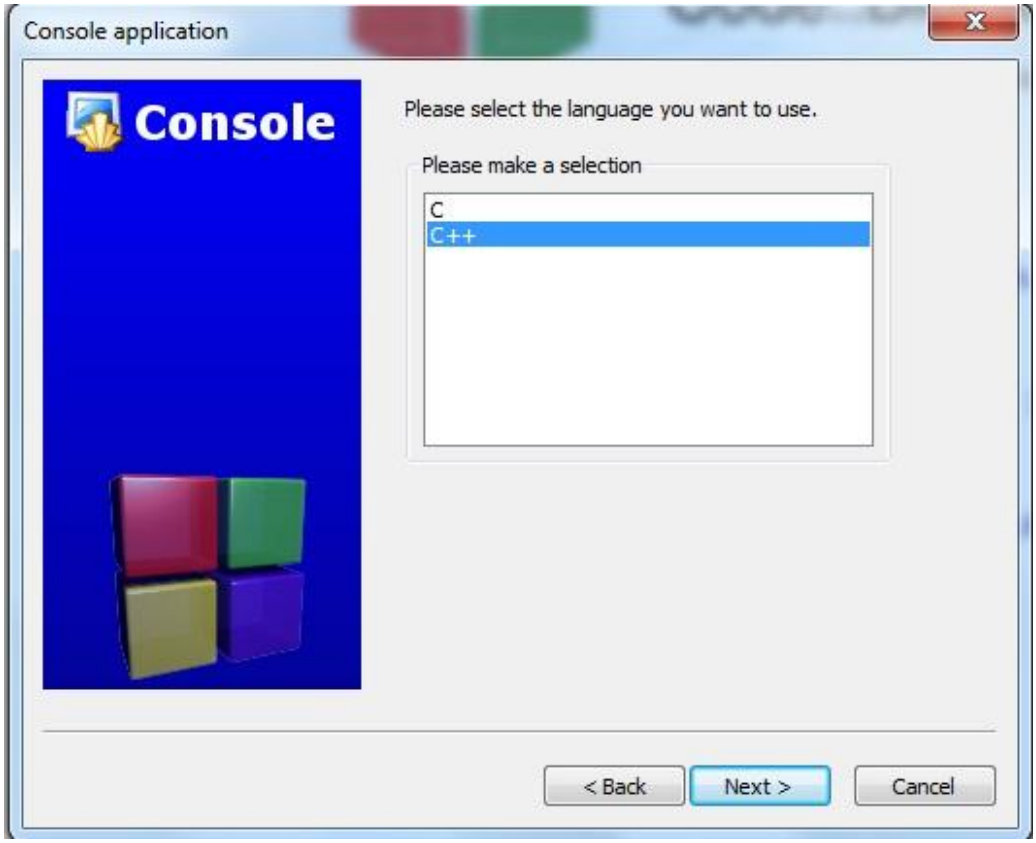
۹. اکنون پنجره New from template ظاهر می‌شود (شکل زیر):



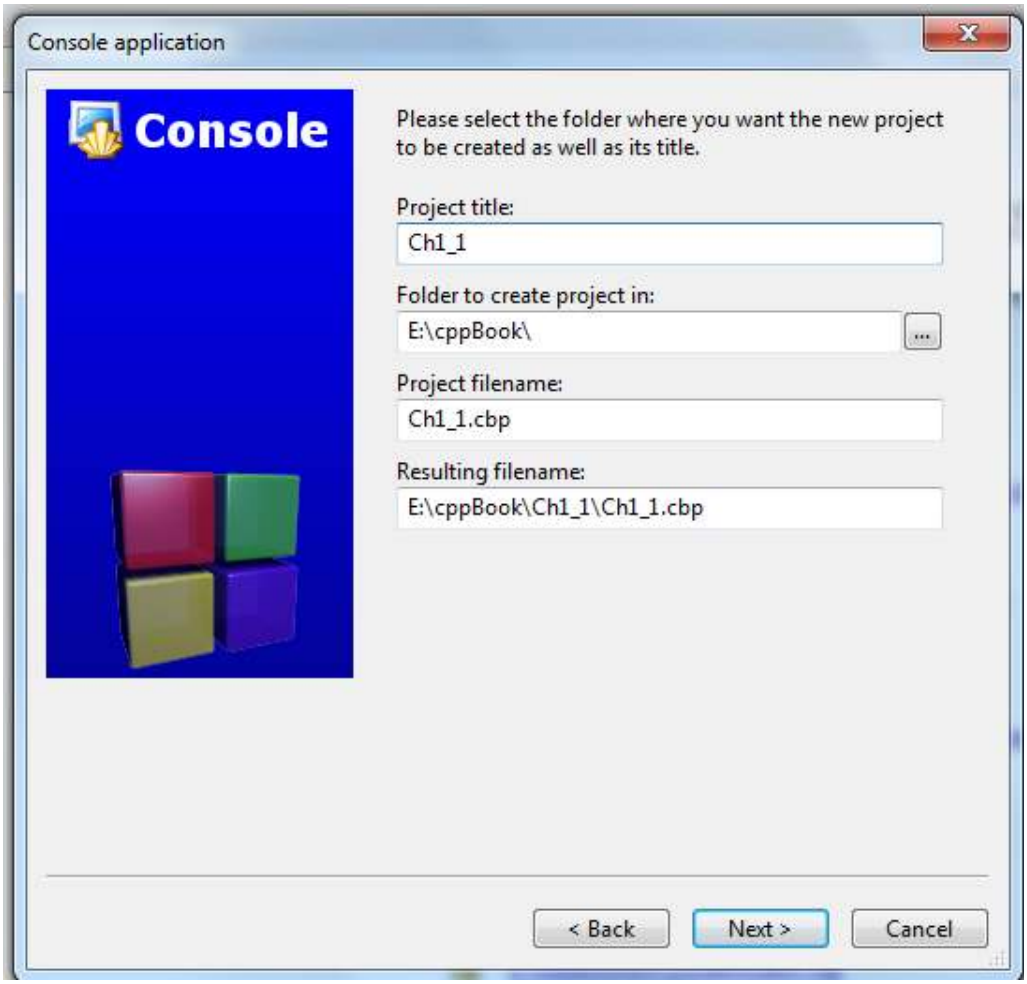
۱۰. در این پنجره، آیکون `Console application` را انتخاب کنید. سپس دکمه Go را کلیک نمایید تا شکل زیر ظاهر شود:



۱۱. در این شکل دکمه Next را کلیک کنید تا شکل زیر ظاهر گردد:



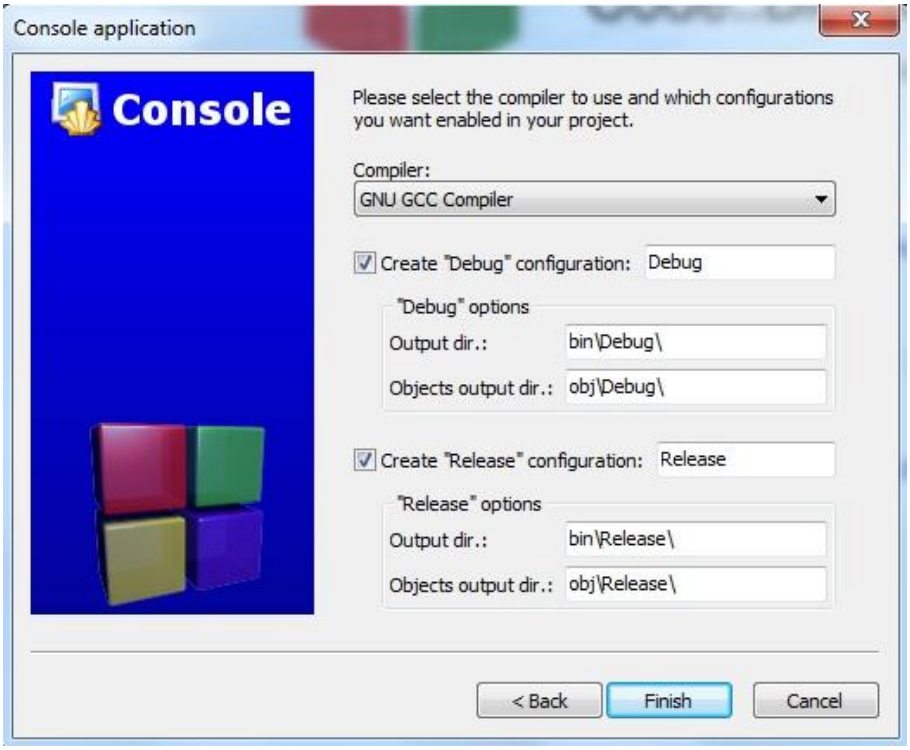
۱۲. در این شکل گزینه C++ را انتخاب کرده و دکمه Next را کلیک کنید تا شکل زیر ظاهر گردد:



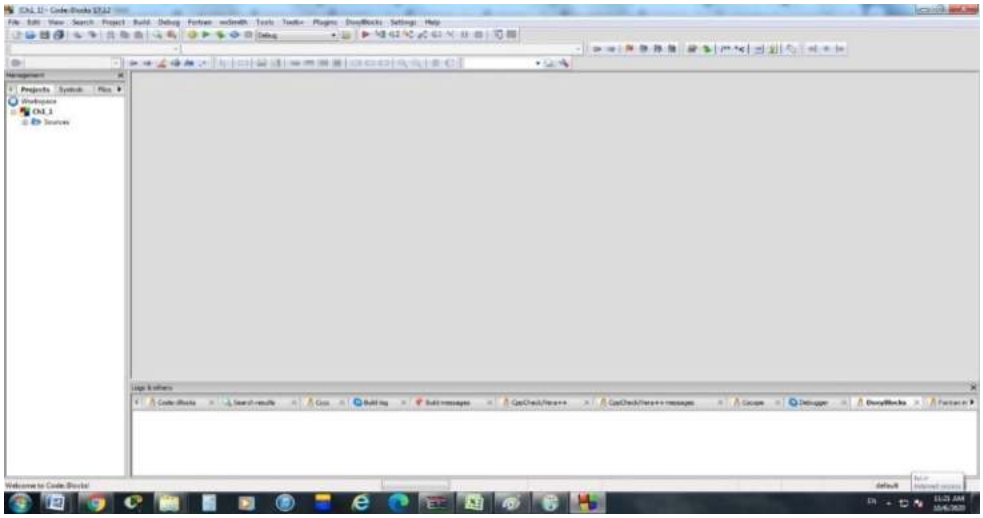
۱۳. در این شکل گزینه‌های زیر را وارد کنید:

- در کادر زیر `Project tilte`، عنوان پروژه را وارد کنید.
- در کادر زیر `Folder to create project in`، مسیر ایجاد پروژه را وارد کنید.
- در کادر `Project filename`، نام فایل پروژه را وارد کنید.
- در کادر زیر `Resulting filename`، مسیر خروجی پروژه را وارد کنید.

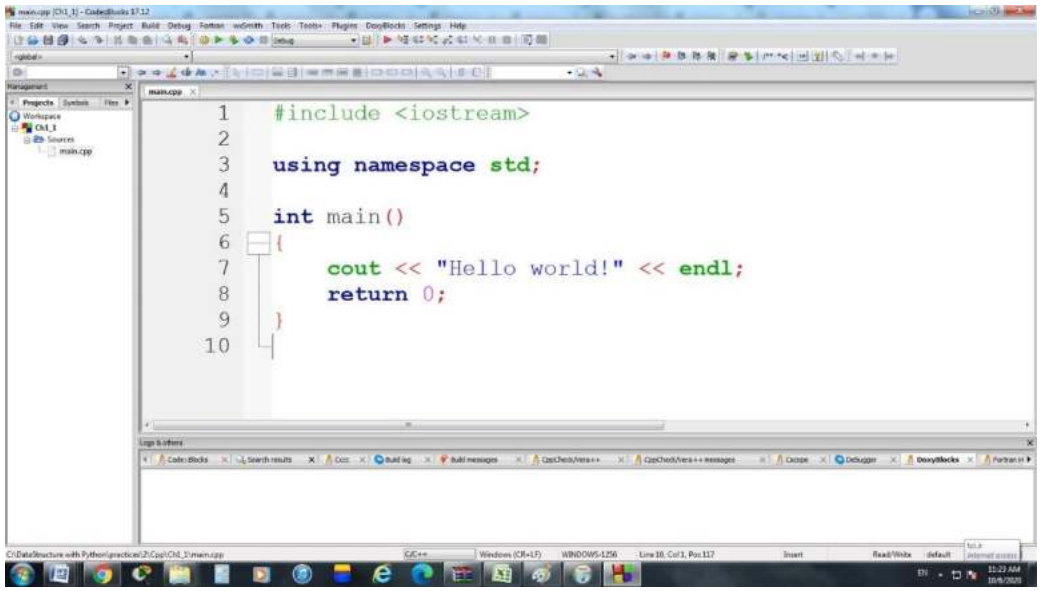
۱۴. اکنون دکمه `Next` را کلیک کنید تا شکل زیر ظاهر گردد:



۱۵. در این شکل دکمه `Finish` را کلیک کرده تا شکل زیر ظاهر شود.



۱۶. اکنون اولین برنامه C++ ایجادشده است. برای مشاهده دستورات این برنامه در زیر Ch1_۱ گزینه Sources و سپس main.cpp را کلیک کرده تا شکل زیر ظاهر شود:

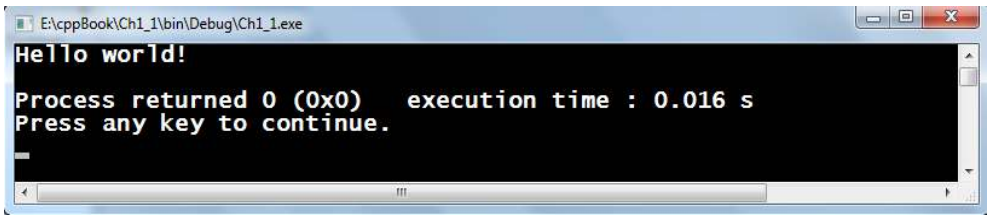


اولین خط برنامه #include <iostream> است. این دستور یک ***رهنمود پیش پردازنده*** است، که پیغامی برای پیش پردازنده C++ می‌باشد. خطوطی که با نماد # آغاز می‌شوند، قبل از این‌که برنامه کامپایل شود توسط پیش پردازنده پردازش می‌شوند. این خط به پیش پردازنده اعلان می‌کند تا محتویات ***سرآیند جریان ورودی/خروجی*** *فایل* ***<iostream>*** را در برنامه وارد سازد. بایستی این فایل در هر برنامه‌ای که می‌خواهد داده‌ای به صفحه‌نمایش انتقال دهد، یا این‌که از صفحه‌کلید و با استفاده از جریان ورودی/خروجی C++ داده‌ای دریافت نماید، وارد شود. *درصورتی‌که افزودن فایل سرآیند <iostream> به برنامه‌ای که مبادرت به دریافت داده از صفحه‌کلید یا ارسال داده به صفحه‌نمایش می‌کند، فراموش شود، کامپایلر یک پیغام خطا ارسال خواهد کرد.* دومین سطر، فقط یک خط خالی است. برنامه‌نویسان برای این‌که خوانایی برنامه را آسان‌تر کنند از خطوط خالی، کاراکترهای فاصله (space) و تب (tab) استفاده می‌کنند. به مجموعه این کاراکترها، **فضای سفید** (white space) می‌گویند. معمولاً این کاراکترها توسط کامپایلر نادیده گرفته می‌شوند.

- با استفاده از خطوط خالی و کاراکترهای فاصله، خوانایی برنامه‌ها را افزایش دهید.

سومین سطر، با دستور using فضای نام std را به برنامه اضافه می‌کند. اگر این دستور را به برنامه اضافه نکنید، به‌جای دستور **std::cout << "Hello World!" << endl;** باید دستور **cout << "Hello World!" << endl;** را استفاده کنید که کار سخت و طاقت‌فرسایی است. این دستور یک *عبارت* گفته می‌شود. هر عبارتی در C++ باید با یک سیمیکولن (;) خاتمه پذیرد (که به آن *خاتمه دهنده عبارت* هم گفته می‌شود). همان‌طور که مشاهده کردید، رهنمودهای پیش پردازنده (همانند **#include**) با سیمیکولن خاتمه نمی‌یابند. دستور بعدی، **int main()** می‌باشد که بخشی از هر برنامه C++ است. پرانتزهای واقع بعد از **main** نشان می‌دهند که **main** یک **بلوک برنامه به نام *تابع*** است. برنامه‌های C++ می‌توانند حاوی یک یا چندین تابع و کلاس باشند، اما باید یکی از آن‌ها حتماً **main** باشد، حتی اگر **main** اولین تابع در برنامه نباشد. کلمه کلیدی **int** که در سمت چپ **main** قرارگرفته، بر این نکته دلالت دارد که **main** یک مقدار صحیح **"برمی‌گرداند"** (عدد بدون اعشار). کلمه کلیدی، کلمه‌ای در کد است که توسط C++ رزرو شده است. لیست کامل کلمات را در ادامه خواهید دید. در فصل سوم مفهوم دقیق "مقدار برگشتی" از سوی یک تابع خواهیم دید. اما برای این لحظه، کافی است بدانید که کلمه کلیدی int در سمت چپ برنامه‌های شما قرار خواهد گرفت. بایستی براکت چپ، { (سطر بعد از دستور int main())، در ابتدای بدنه هر روالی قرار داده شود. براکت متناظر، براکت راست، }، (سطر آخر برنامه) است،که باید آن را در انتهای بدنه هر روالی قرار داد. دستور **std::cout << "Hello World!" << endl;** به کامپیوتر *فرمان* می‌دهد تا رشته‌ای از کاراکترها را که مابین جفت کتیشن قرار دارند بر روی صفحه‌نمایش چاپ کند و endl به کامپیوتر فرمان می‌دهد تا کنترل نمایش را به ابتدای صفحه بعد انتقال دهد. دستور بعدی return ۰; است که برنامه را خاتمه می‌دهد.

۱۷. پروژه را اجرا کنید. برای این منظور آیکون را کلیک کنید تا شکل زیر را ببینید:



همان‌طور که مشاهده می‌کنید، رشته Hello World! نمایش داده‌شده است. جهت خاتمه اجرای برنامه کلیدی را فشار دهید.

در این شکل زمینه پنجره مشکی و نوشته آن سفید است. در خروجی‌های بعدی به ترتیب رنگ زمینه و نوشته را به سفید و مشکی تغییر دادیم تا خروجی با کیفیت بهتر نمایش داده شود.

۵ – ۱. کلمات کلیدی

کلمات کلیدی[۳]، کلماتی هستند که در زبان شناخته‌شده‌اند و مفهوم خاصی در آن دارند. به‌عنوان‌مثال، کلماتی نظیر `if` (اگر)، `else` (در غیر این صورت)، `void` (هیچ)، `for` (تکرار) کلیدی هستند. برخی از کلمات کلیدی C و C++ را در جدول ۱ – ۱ می‌بینید. در ادامه مفاهیم این کلمات را مشاهده خواهید کرد.

- زیاد بودن تعداد کلمات کلیدی نشانه قدرت زبان برنامه‌نویسی نیست. به‌عنوان‌مثال، زبان بیسیک بیش‌تر از ۱۵۰ کلمه کلیدی دارد، ولی توانمندی آن بسیار کم‌تر از زبان C++ است.
- استفاده از کلمه کلیدی به‌عنوان نام یک شناسه یا متغیر خطای نحوی خواهد بود.
- نوشتن یک کلمه کلیدی با حروف بزرگ خطای نحوی است. تمام کلمات کلیدی در C++ فقط شامل حروف کوچک هستند.

۶–۱. انواع داده‌ها

در زبان C++ داده‌های مختلفی وجود دارند که عبارت‌اند از:

۱. **داده‌های اولیه،** داده‌های استاندارد نظیر `int`، `float`، `double`، `char` و غیره که در زبان وجود دارند.

۲. **داده‌هایی که کاربر تعریف می‌کند.** داده‌هایی نظیر آرایه‌ها مجموعه‌ها، ساختمان‌ها، رشته و کلاس‌ها که توسط برنامه‌نویس تعریف می‌شوند. در ادامه با این داده‌ها آشنا خواهید شد.

۳. **داده‌های اشاره‌گر،** برای نگه‌داری آدرس فیزیکی متغیرها به کار می‌روند. این نوع داده‌ها را در ادامه می‌آموزیم.

۱ – ۶ – ۱. داده‌های اولیه

همان‌طور که بیان گردید، انواع داده‌هایی که به‌صورت استاندارد در زبان وجود داشته باشند، انواع داده‌های اصلی یا اولیه نام دارند. این داده‌ها انواع مختلفی دارند که در زیر آمده‌اند:

---

[۳]. keywords

جدول ۱ – ۱ کلمات کلیدی زبان C و ++C.

کلمات کلیدی مشترک زبان C و ++C				کلمات کلیدی فقط زبان ++C				
break	case	char	const	xor_eq	typename	namespasce	bitand	bitor
default	do	double	else	bool	catch	class	compl	asm
extern	float	for	go to	delete	dynamic_cast	explicit	export	false
int	long	register	return	friend	inline	mutable	and_eq	new
singed	siezof	static	struct	private	protected	public		try
typedef	union	unsigned	void	not	not_eq	operator	or_eq	true
while	enum	continue	auto	template	this	throw	typeid	and
volatile	switch	short	if	using	virtual	wchar_t		or
				static_cast	reinterpret_cast	const_cast		xor

۱. نوع داده‌ای عددی

نوع داده‌ای که برای نگه‌داری اعداد به کار می‌روند. انواع داده‌ای عددی نیز دو نوع می‌باشند که عبارت‌اند از:

- **داده‌های عددی صحیح،** برای نگه‌داری مقادیر عددی صحیح مثبت و منفی به کار می‌روند. انواع این داده‌ها، تعداد بایت‌های که اشغال می‌کنند و محدوده مقادیر این داده‌ها را در جدول ۲ – ۱ می‌بینید.
- **داده‌های اعشاری،** برای نگهداری اعداد اعشاری مثبت و منفی به کار می‌روند. این انواع نیز در جدول ۲ – ۱ آمده‌اند.

دقت اعداد اعشاری و نیاز به حافظه، متغیرهای از نوع `float` نشان‌دهنده اعداد با دقت معمولی در نقطه اعشار هستند و دارای هفت رقم معنی‌دار در سیستم‌های ۳۲ بیتی می‌باشند. متغیرهای از نوع `double` نشان‌دهنده دقت مضاعف در نقطه اعشار هستند. این دقت مستلزم دو برابر حافظه موردنیاز برای یک متغیر `float` است و دارای ۱۵ رقم معنی‌دار در سیستم‌های ۳۲ بیتی است (تقریباً دو برابر دقیق‌تر از متغیرهای`float`). برای اکثر محاسبات صورت گرفته در برنامه‌ها، نوع `float` می‌تواند کافی باشد، اما می‌توانید با استفاده از `double` دقت را تضمین کنید. در برخی از برنامه‌ها، حتی متغیرهای از نوع `double` هم کافی نیستند، برنامه‌هایی که خارج از قلمرو بحث این کتاب هستند. اکثر برنامه‌نویسان برای عرضه اعداد اعشاری از نوع `double` استفاده می‌کنند. درواقع ++C به‌طور پیش‌فرض با تمام اعداد اعشاری که در کد برنامه تایپ می‌کنید (همانند ۷,۳۳ و ۰,۰۹۷۵) همانند مقادیر `double` رفتار می‌کند. چنین مقادیری در کد برنامه به‌عنوان ثابت‌های اعشاری شناخته می‌شوند.

غالباً اعداد اعشاری در انجام عملیات تقسیم گسترش زیادی پیدا می‌کنند. برای مثال با تقسیم ۱۰.۰ بر ۳.۰، نتیجه ۳,۳۳۳۳۳۳... با دنباله‌ای از ۳های نامتناهی خواهد بود. کامپیوتر فضای ثابتی برای نگه‌داری چنین مقادیری در اختیار دارد، از این‌رو، ذخیره‌سازی مقادیر اعشاری فقط به‌صورت تخمینی صورت می‌گیرد.

علیرغم این‌که اعداد اعشاری همیشه ۱۰۰ درصد دقیق نیستند، اما کاربردهای بسیاری دارند. برای مثال، هنگامی‌که در مورد حرارت عادی بدن یعنی ۹۸,۶ صحبت می‌کنیم، نیازی نیست تا دقت اعشاری آن را بسیار دقیق بیان کنیم. زمانی که به درجه حرارت در یک دماسنج نگاه می‌کنیم و آن را ۹۸,۶ می‌خوانیم، ممکن است مقدار دقیق آن ۹۸,۵۹۹۹۴۷۳۲۱۰۶۴۳ باشد. اما استفاده از مقدار ۹۸,۶ به‌صورت تخمینی در بسیاری از موارد می‌تواند مناسب و کاربردی باشد.

استفاده از اعداد اعشاری با فرض این‌که این اعداد نشان‌دهنده مقدار کاملاً دقیق هستند (به‌ویژه در عبارات مقایسه‌ای) می‌تواند نتایج اشتباهی به دنبال داشته باشد. اعداد اعشاری تقریباً در تمام کامپیوترها نشان‌دهنده یک مقدار تقریبی هستند.

جدول ۲ - ۱ انواع داده اولیه در C++.

نوع داده	نام نوع	محدوده داده	تعداد بایت	مثال
عددی صحیح با طول کوتاه	**short int**	-۱۲۸.....۱۲۷	۱	**-۱۲۰**
عددی صحیح	**Int**	-۳۲،۷٦۸ ... ۳۲،۷٦۷	۲	**۳۰۱۵**
عدد صحیح مثبت	**unsigned int**	۰ .... ٦۵،۵۳۵	۲	**۲۷۹۸**
عدد صحیح با طول بلند	**long int**	-۲،۱٤۷،٤۸۳،٦٤۸ ... ۲،۱٤۷،٤۸۳،٦٤۷	٤	**۲۹۸۹۷٦**
عدد صحیح مثبت با طول بلند	**unsigned long**	۰...٤۲،٤۹٦۷۲۹۵	٤	**۹۹۹۹۹۹۹**
عدد اعشاری	**float**	۱۰۱۷۵٤۹٤۳۵e-۳۸... ۳۰٤۰۲۸۲۳٤۷e+۳۸	٤	**۱۱۲۰۷۵**
عدد اعشاری‌بادقت مضاعف	**double**	۲۰۲۲۵۰۷۳۸۵۸۵۸۵۰۷۲۰۱٤e-۳۰۸ ... ۱۰۷۹۷٦۹۳۱۳٤۸٦۲۳۱۵۷e+۳۰۸	۸	**۰,۰۰۰۰۰۰۰۰۱**
عدد اعشاری‌بادقت مضاعف و۱۵ رقمی	**long double**	۱۰۷e-۳۰۸...۱۰۷e+۳۰۸	۸	**۱۰۰۰۰۰۰۰۰۰. ۱۲۵۷۹۱۵۷۵**
کاراکتری	**char**	نگهداری یک کاراکتر۱۲۷ ...-۱۲۸	۱	**'a'**
منطقی	**bool**	false یا true	۱	**false**

۱. نوع داده‌ای کاراکتری

داده‌های کاراکتری برای ذخیره یک کاراکتر (هر علامتی که بین تک کوتیشن (' ')قرار می‌گیرد) به کار می‌روند. این نوع داده را نیز در جدول ۲ - ۱ مشاهده می‌کنید.

تذکر: هر داده کاراکتری یک بایت حافظه را اشغال می‌کند.

۲. نوع داده‌ای منطقی

این نوع داده‌ها برای نگه‌داری نتیجه ارزیابی عبارت‌های منطقی و ریاضی (True و False) به کار می‌روند. این نوع داده‌ها را نیز در جدول ۲ - ۱ می‌بینید.

۷ - ۱. متغیر

برای نگه‌داری هر چیز لازم است که از یک ظرف متناسب با آن استفاده نمود. به‌عنوان‌مثال، در خانه برای نگه‌داری مواد غذایی، ظروف مختلفی وجود دارند که هرکدام برای نگه‌داری مواد خاصی به کار می‌روند. مثلاً، بطری برای نگه‌داری آب و غیره به همین ترتیب در برنامه‌نویسی، برای نگه‌داری مقادیر از ظروف مخصوص به خود استفاده می‌شود. ظرف نگه‌داری داده در زبان‌های برنامه‌نویسی **متغیر** نام دارد. بنابراین، **متغیر نامی است برای یک مکان از حافظه که ممکن است که در طول اجرای برنامه مقدار آن تغییر کند. ولی، در یک‌لحظه فقط یک مقدار را دارد.**

برای استفاده از متغیرها سه عمل باید انجام شود که عبارت‌اند از:

۱. نام‌گذاری متغیرها

بعد از این‌که یک بچه به دنیا آمد، برای شناسایی او نامی انتخاب می‌کنید. جهت مراجعه به متغیرها نیز از نام آن‌ها استفاده می‌شود. برای نام‌گذاری بچه‌ها ثبت‌احوال از قوانینی خاصی پیروی می‌کند، به‌عنوان‌مثال، اجازه نمی‌دهد نام بچه را رضا۱ انتخاب کرد. در زبان C++ نیز برای نام‌گذاری متغیرها قوانین زیر وجود دارد:

۱. نام متغیر می‌تواند ترکیبی از حروف a تا z یا A تا Z ارقام، خط ربط (_)، ارقام ۰ تا ۹ باشد.
۲. حرف اول متغیر نمی‌تواند ارقام ۰ تا ۹ باشد.
۳. زبان C++ بین حروف بزرگ و کوچک فرق می‌گذارد. یعنی، متغیرهای Count و count با هم فرق دارند.
٤. نام متغیر نمی‌تواند کلمات کلیدی یا نام توابع انتخاب شود.

- زبان C++ به شناسه‌ها امکان می‌دهد تا هر طولی داشته باشند، اما امکان دارد سیستم شما یا ساختار C++ به‌کاررفته محدودیت‌هایی بر روی طول شناسه‌ها اعمال کند. از این‌رو، برای حفظ سازگاری و قابلیت حمل، از شناسه‌هایی با طول ۳۱ کاراکتر یا کم‌تر استفاده کنید.
- در برنامه‌نویسی ایده‌ال، انتخاب اسامی با معنی به برنامه کمک می‌کند که خود به‌عنوان توضیحی بر برنامه باشد (self-documenting). در چنین حالتی اگر متن برنامه در اختیار دیگران قرار داده شود، بدون این‌که نیازی به راهنما و توضیحات اضافی باشد، عملکرد برنامه مشخص خواهد بود.

- در برنامه‌نویسی ایده‌ال، از اختصار یا کوته‌سازی شناسه‌ها اجتناب کنید، تا خوانایی برنامه افزایش یابد.
- در برنامه‌نویسی ایده‌ال، از ایجاد شناسه‌های که با یک خط زیر (_) یا دو خط زیر (__) آغاز می‌شوند اجتناب کنید، چراکه امکان دارد کامپایلر C++ از اسامی مشابهی برای انجام مقاصد داخلی خود استفاده کرده باشد. در این صورت، از ایجاد تداخل مابین خود و کامپایلر جلوگیری خواهید کرد.

برخی از نام‌های مجاز برای متغیر عبارت‌اند از: area، sum، sum۱، pr-۱ و غیره. اما، نام‌های زیر برای متغیر مجاز نیستند:

- **نام 2test:** نام متغیر نمی‌تواند با ارقام ۰ تا ۹ شروع شود.
- **نام $test:** در نام متغیر نمی‌توان از $ استفاده کرد.
- **نام store 2:** در نام متغیر نمی‌توان کاراکتر فاصله[٤] استفاده کرد.
- **نام jpg.:** نام متغیر نمی‌تواند با کاراکتر نقطه (.) شروع شود.

۲. معرفی متغیرها

همان‌طور که بیان گردید، هر ظرفی برای نگه‌داری نوعی غذا به کار می‌رود. بنابراین، متغیرها نیز باید دارای نوع باشند تا بتوانند انواع داده‌ها را ذخیره کنند. چون داده‌ها دارای انواع مختلف هستند. بنابراین، متغیرها که داده‌ها را نگه‌داری می‌کنند، باید دارای نوع باشند. نوع متغیر تعیین می‌کند اولاً چه نوع داده‌ایی می‌تواند در آن متغیر قرار گیرد و ثانیاً، این متغیر به چند بایت از حافظه نیاز دارد. تعیین نوع متغیر به‌صورت زیر می‌باشد:

```
; لیست متغیرها        نوع داده‌ای
```

نوع داده‌ای، یکی از انواع داده بیان‌شده نظیر int، float، double و غیره در C++ می‌باشد.

- اگر تعداد متغیرها بیش از یکی باشند، با کاما (,) از هم جدا می‌شوند. پس از هر , یک فاصله قرار دهید تا خوانائی برنامه افزایش یابد.
- در برنامه‌نویسی ایده‌ل، برخی از برنامه‌نویسان ترجیح می‌دهند تا هر متغیر را در یک خط جداگانه اعلان کنند. در این حالت قرار دادن یک توضیح در کنار هر اعلان به‌آسانی صورت می‌گیرد.

مثال ۲-۱. برنامه‌ای که متغیرهای a، b و c را با نوع int، d را با نوع double، f۱ و f۲ را با نوع float و ch را با نوع کاراکتر تعریف می‌کند.

۱. پروژه جدیدی به نام Ch1_2 ایجاد کنید.

[٤] .blank(space)

۲. دستورات پروژه را به‌صورت زیر تغییر دهید:

```
#include <iostream>
using namespace std;
int main()
{
    int a, b, c;
    double d;
    float f1, f2;
    char ch;
    return 0;
}
```

دستور پنجم، متغیرهای a، b و c را با نوع int تعریف می‌کند، دستور ششم، متغیر d را با نوع double تعریف می‌نماید، دستور هفتم، متغیرهای f1 و f2 را با نوع اعشاری (float) تعریف می‌کند، دستور هشتم، متغیر ch را با نوع کاراکتری تعریف می‌نماید.

۳. پروژه را ذخیره و اجرا کنید تا خروجی را زیر مشاهده نمایید:

هر متغیر دارای **نام**، **نوع**، **اندازه** و **یک مقدار** است.

سه روش برای مقداردهی به متغیرها وجود دارد، یکی از روش‌ها، مقداردهی به متغیرها در هنگام تعریف آن است.

مثال ۳–۱. برنامه زیر متغیرهای a و b را با نوع int تعریف کرده، مقادیر 10 و 12 را به آن‌ها تخصیص می‌دهد و متغیر PI را با نوع float با مقدار 3.14 تعریف می‌کند، ch را با نوع کاراکتری و مقدار 'F' تعریف کرده، متغیر yes را با نوع منطقی و مقدار true تعریف می‌نماید..

۱. پروژه جدیدی به نام Ch1_3 ایجاد کنید.

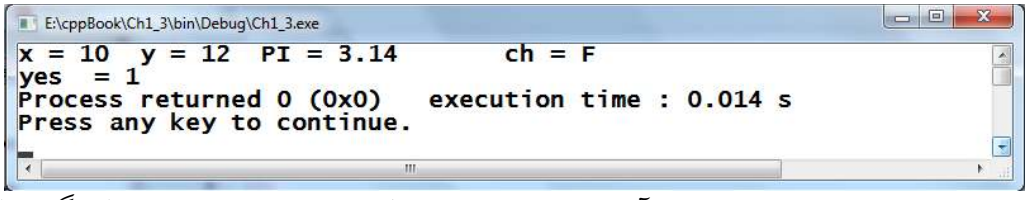
۲. دستورات پروژه را به‌صورت زیر تغییر دهید:

```
#include <iostream>
using namespace std;
int main()
{
    int x = 10, y = 12;
    float PI = 3.14f;
    char ch = 'F';
    bool yes = true;
    cout << "x = " << x <<  "\ty = " << y;
```

```
10      cout << "PI = " <<  PI << "\tch = " <<  ch << " \nyes  = " <<
11          yes;
12      return 0;
13  }
```

دستور پنجم، متغیرهای x را با مقدار اولیه ۱۰ و y را با مقدار اولیه ۱۲ تعریف می‌کند، دستور ششم، متغیر PI را با نوع float و مقدار اولیه 3.14f (f تعیین می‌کند که داده 3.14 اعشاری است) تعریف می‌کند، دستور هفتم، متغیر ch را با نوع char (کاراکتری) و مقدار 'F' تعریف می‌نماید، دستور هشتم، متغیر yes را با نوع bool (منطقی) و مقدار true تعریف می‌نماید، دستور نهم، ابتدا، عبارت = x را نمایش می‌دهد، سپس مقدار متغیر x (یعنی ۱۰) را نمایش می‌دهد، در ادامه کنترل چاپ را با کاراکتر '\t' به تب بعدی انتقال می‌دهد و عبارت = y را نمایش می‌دهد، در پایان، مقدار متغیر y (یعنی ۱۲) را نمایش می‌دهد و دستور دهم، ابتدا کنترل چاپ را به تب بعدی انتقال داده، عبارت = PI را نمایش می‌دهد و سپس مقدار متغیر PI را نمایش می‌دهد، سپس کنترل چاپ را به تب بعدی انتقال می‌دهد، عبارت = ch را نمایش داده و مقدار متغیر ch (یعنی 'F') را چاپ می‌کند و در پایان، با کنترل چاپ با کاراکتر '\n' به خط بعدی می‌رود، عبارت = yes را نمایش داده، مقدار متغیر yes (یعنی، ۱. چون در C++ مقدار true به ۱ و false به ۰ تبدیل می‌گردد) را چاپ می‌کند.

۳. پروژه را ذخیره و اجرا کنید تا خروجی را به شکل زیر مشاهده نمایید:



بعد از تعریف متغیر نیز می‌توان به آن‌ها مقدار داد. برای این منظور می‌توانید از دستور انتساب (عملگر =) یا دستورات ورودی استفاده کنید. نمونه‌ای از کاربرد عملگر = در زیر آمده است. در ادامه، دستورات ورودی جهت تخصیص مقدار به متغیرها را می‌بینید.

مثال ۴ – ۱. برنامه‌ای که سه متغیر به نام‌های a، f و yes را به ترتیب با انواع int، float و bool تعریف می‌کند و سپس، به آن‌ها مقادیر 10، 13.7 و false را تخصیص می‌دهد.

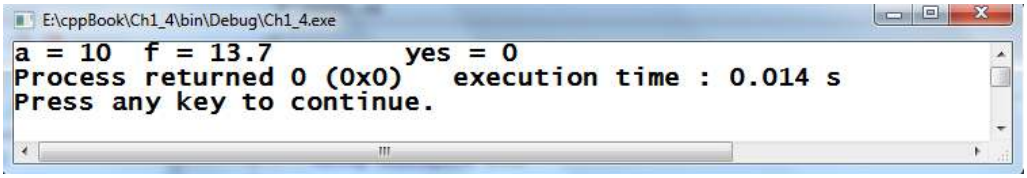
۱. پروژه جدیدی به نام Ch1_4 ایجاد کنید.

۲. دستورات پروژه را به‌صورت زیر تغییر دهید:

```
#include <iostream>
using namespace std;
int main()
{
   int a;
   float f;
   bool yes;
   a = 10;
   f = 13.7f;
   yes = false;
   cout << "a = " << a << "\tf = " << f << "\tyes = " << yes;
   return 0;
}
```

دستورات پنجم تا هفتم به ترتیب متغیرهای a را با نوع int، f را با نوع float و yes را با نوع bool تعریف می‌کنند، دستورات هشتم تا دهم، به ترتیب به متغیرهای a، f و yes مقادیر ۱۰، ۱۳,۷f و false تخصیص می‌دهند و دستور یازدهم، ابتدا عبارت a = را نمایش داده، سپس مقدار ۱۰ (مقدار a) را نمایش می‌دهد، با کاراکتر '\t' به تب بعدی می‌رود و عبارت f = را نمایش داده، جلوی آن مقدار ۱۳,۷ (مقدار متغیر f) را نمایش می‌دهد و در پایان، به تب بعدی می‌رود، عبارت yes = را نمایش داده و روبه‌روی آن مقدار ۰ (مقدار متغیر yes که false است به معادل ۰ تبدیل می‌شود) را نمایش می‌دهد.

۳. پروژه را ذخیره و اجرا کنید تا خروجی را به‌صورت زیر مشاهده نمایید:



نکته: وقتی مقدار جدیدی در متغیر قرار می‌گیرد، این مقدار جایگزین مقدار قبلی می‌شود. یعنی، مقدار قبلی از دست می‌رود (حذف می‌گردد).

۸ – ۱. ثابت‌ها

ثابت شناسه‌ای (نام خانه‌ای از حافظه) است که مقدار آن در طول اجرا برنامه تغییر نمی‌کند. ثابت‌ها انواع مختلف دارند. ثابت‌ها می‌توانند عددی صحیح، اعشاری، کاراکتری، رشته‌ای یا منطقی باشند. ثابت‌های کاراکتری بین تک کتیشن (') قرار می‌گیرند (مانند 'C')، ثابت‌های رشته‌ای بین جفت کتیشن قرار می‌گیرند (نظیر "C++") و ثابت‌های منطقی مقادیر true یا false هستند. به‌عنوان‌مثال، مقدار 3.1415، (عدد Π) را مشخص می‌کند. این عدد یا هر ثابت دیگر ممکن است چندین مرتبه در برنامه استفاده شود. به دلیل راحتی اصلاح و تغییر مقدار ثابت‌ها به آن‌ها نام تخصیص می‌دهند. تعریف ثابت در C++ به صورت‌های زیر انجام می‌شود.

۱.const نوع ثابت نام ثابت = مقدار ثابت ;

	مقدار ثابت	نام ثابت	
۲.		#define	

ساختار اول در هر جای برنامه می‌تواند تعریف شود. اما، ساختار دوم در ابتدای برنامه قبل از تابع اصلی (main) تعریف می‌گردد.

مثال ۵ – ۱. برنامه‌ای که دو مقدار ثابت از طریق #define و const تعریف کرده، آن‌ها را نمایش می‌دهد.

۱. پروژه جدیدی به نام Ch1_5 ایجاد کرده، دستورات برنامه را به‌صورت زیر تغییر دهید:

```
1  #include <iostream>
2  #define ANS true
3  using namespace std;
4  int main()
5  {
6     const float  PI = 3.1415;
7     cout << "Ans = " << ANS << "\tPi = " << PI;
8     return 0;
9  }
```

دستور دوم (#define ANS true) ثابتی به نام ANS با مقدار true تعریف می‌کند. دستور ششم (const float PI 3.1415;) ثابت PI را با مقدار 3.1415 تعریف می‌کند و دستور هفتم توسط cout، ابتدا عبارت Ans = را نمایش داده، سپس مقدار ۱ (مقدار ANS) را نمایش می‌دهد، با کاراکتر '\t' به تب بعدی می‌رود و در پایان، عبارت Pi = را نمایش داده، جلوی آن مقدار 3.1415 (مقدار ثابت PI) را نمایش می‌دهد.

۲. پروژه را ذخیره و اجرا کنید تا خروجی را به‌صورت زیر مشاهده نمایید:

اکنون دستورات زیر را ببینید.

```
ans  =  false;
PI = 3.141505;
```

این دستورات نادرست هستند. زیرا، مقدار ثابت‌ها را نمی‌توان تغییر داد.

دستوراتی که با علامت # شروع می‌شوند. دستورات **پیش پردازنده** نام دارند که مربوط به کامپایلر هستند. پیش پردازنده یک برنامه سیستمی است که قبل از انجام عمل ترجمه توسط کامپایلر تغییراتی را در کد برنامه ایجاد می‌کند.

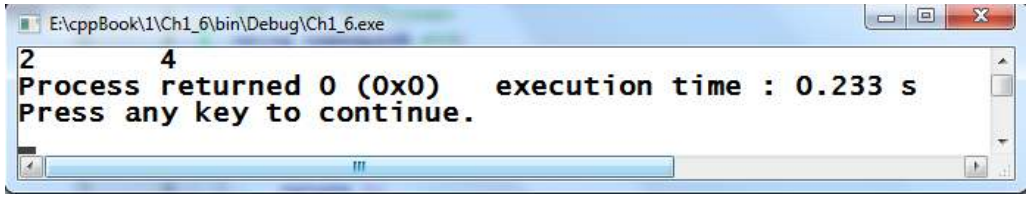
ثابت‌هایی که با define تعریف می‌شوند، اصطلاحاً **ماکرو** نامیده می‌شوند و پس از تعریف یک ماکرو می‌توان از آن‌ها در تعریف ماکروهای دیگر استفاده نمود.

مثال ۶ – ۱. دستوراتی که استفاده از یک ماکرو در تعریف ماکروی دیگر را نمایش می‌دهد.

۱. پروژه جدیدی به نام Ch1_6 ایجاد کرده، دستورات برنامه را به‌صورت زیر تغییر دهید:

```
1  #include <iostream>
2  using namespace std;
3  #define PL 2
4  #define PU PL + PL
5  int main()
6  {
7      cout << PL << "\t" << PU;
8      return 0;
9  }
```

۲. پروژه را ذخیره و اجرا کنید تا خروجی زیر را مشاهده نمایید:



توصیه می‌شود که نام ماکروها را با حروف بزرگ بنویسید. اگرچه می‌توان نام آن‌ها را با حروف کوچک نوشت، ولی برای آن‌که آن‌ها را از متغیرها بدهیم، بزرگ نوشتن نام آن‌ها مفید است و موجب افزایش خوانایی برنامه می‌شود.

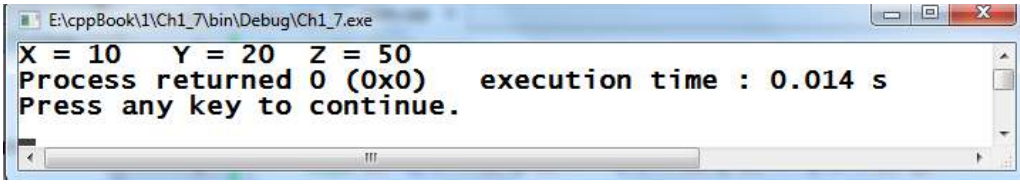
با هر دستور `#define` فقط یک ثابت قابل‌تعریف است اما با استفاده از `const` می‌توان چند ثابت را به‌طور هم‌زمان تعریف نمود. البته دقت داشته باشید ثابت‌هایی که با `const` تعریف می‌شوند، به‌اندازه میزان حافظه‌ای که نوع داده‌ای متناظر نیازمندند، حافظه اشغال می‌کنند.

مثال ۷ – ۱. دستوری که چند ثابت را در یک دستور `const` تعریف می‌کند.

۱. پروژه جدیدی به نام Ch1_7 ایجاد کرده، دستورات برنامه را به‌صورت زیر تغییر دهید:

```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5      const int X = 10,  Y = 20,  Z = 50;
6      cout << "X = " << X << "   Y = " << Y << "  Z = " << Z;
7      return 0;
8  }
```

۲. پروژه را ذخیره و اجرا کنید تا خروجی زیر را مشاهده نمایید:



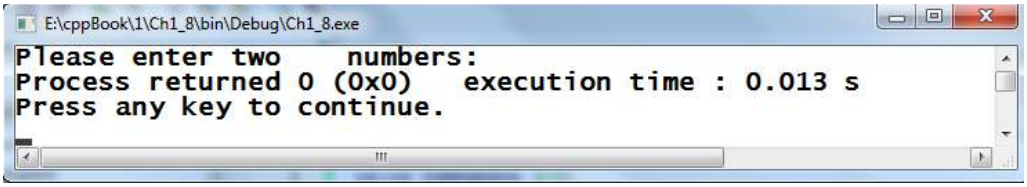
اگر بخواهیم با `#define` به یک رشته نام مجازی بدهیم و رشته در یک خط خا نشود، در انتهای خط کاراکتر \ را قرار می‌دهیم و دنباله رشته را در سطر بعدی می‌نویسیم.

مثال ۸ – ۱. دستوری که تعریف ثابت رشته‌ای را در چند سطر نشان می‌دهد.

۱. پروژه جدیدی به نام Ch1_8 ایجاد کرده، دستورات برنامه را به‌صورت زیر تغییر دهید:

```
1  #include <iostream>
2  using namespace std;
3  #define inpuMess "Please enter two \
4     numbers:"
5  int main()
6  {
7      cout << inpuMess;
8      return 0;
9  }
```

۲. پروژه را ذخیره و اجرا کنید تا خروجی زیر را مشاهده نمایید:



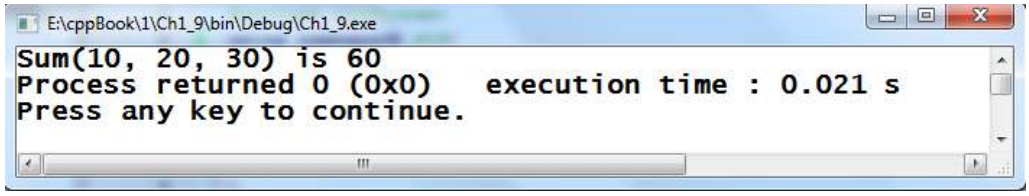
با `#define` می‌توان ماکروی نوشت که مانند تابع عمل کند. با این تفاوت که در هنگام صدازدن توابع کنترل به داخل تابع منتقل می‌شود و پس از اجرای آن به تابع فراخوان برمی‌گردد ولی هنگام فراخوانی ماکرو، دستورات ماکرو در مکان صدازدن عیناً نوشته‌شده و تکرار می‌شود.

مثال ۹ – ۱. دستوری که تعریف تابع را با ماکرو نشان می‌دهد.

۱. پروژه جدیدی به نام Ch1_9 ایجاد کرده، دستورات برنامه را به‌صورت زیر تغییر دهید:

```
1  #include <iostream>
2  using namespace std;
3  #define sum(a, b, c)  a + b + c
4  int main()
5  {
6      cout << "Sum(۱۰, ۲۰, ۳۰) is " << sum(۱۰, ۲۰, ۳۰);
7      return ۰;
8  }
```

۲. پروژه را ذخیره و اجرا کنید تا خروجی زیر را مشاهده نمایید:



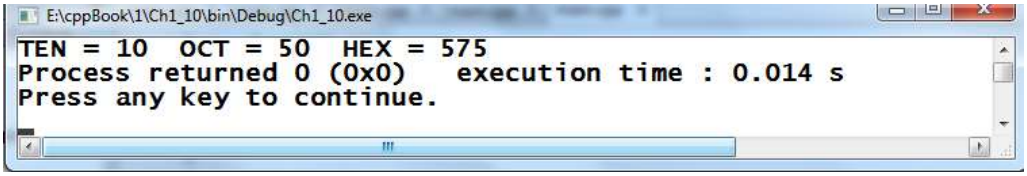
ثابت‌های صحیح را می‌توان در مبنای ده‌دهی (۱۰)، اکتال (۸) و یا هگزا دسیمال (۱۶) نوشت. ثابت‌های صحیح در مبنای ۸ با ۰ شروع می‌شوند و ثابت‌های صحیح در مبنای ۱۶ با 0X شروع می‌شوند.

مثال ۱۰ - ۱. دستوراتی که ثابت‌های در مبنای ۱۰، ۸ و ۱۶ را تعریف می‌کنند.

۱. پروژه جدیدی به نام Ch1_10 ایجاد کرده، دستورات برنامه را به‌صورت زیر تغییر دهید:

```
1  #include <iostream>
2  using namespace std;
3  #define TEN ۱۰
4  #define OCT ۰۶۲
5  #define HEX ۰X۲۳F
6  int main()
7  {
8      cout << "TEN = " << TEN << "  OCT = " << OCT << "  HEX = " <<
9       HEX;
10     return ۰;
11 }
```

۲. پروژه را ذخیره و اجرا کنید تا خروجی زیر را مشاهده نمایید:



۹ – ۱. عملگرها

عملگرها[۵]،نمادهایی هستند که اعمال خاصی را بر روی داده انجام می‌دهند. عملگرها انواع مختلف دارند که برخی از آن‌ها عبارت‌اند از:

۱. عملگرهای محاسباتی ۲. عملگرهای رابطه‌ای (مقایسه‌ای)
۳. عملگرهای ترکیبی ۴. عملگرهای منطقی
۵. عملگرهای خاص

جدول ۳ – ۱ عملگرهای محاسباتی.

عملگر	نام عملگر	مثال	نتیجه	توضیحات
+	جمع	12 + 3	15	عملوند اول را با عملوند دوم جمع می‌کند.
-	تفریق	13.5 – 3	10.5	عملوند دوم را از عملوند اول کم می‌کند.
*	ضرب	12*2.5	30	عملوند اول را در عملوند دوم ضرب می‌کند
/	تقسیم	13/2	6	عملوند اول را بر عملوند دوم تقسیم می‌کند
%	باقی‌مانده تقسیم صحیح	13 % 5	3	باقی‌مانده تقسیم صحیح عملوند اول بر عملوند دوم را محاسبه می‌کند.
++	افزایش	x=10; x++;	11	یک واحد به عملوند اضافه می‌کند.
--	کاهش	x = 10; x --;	9	یک واحد از عملوند کم می‌نماید.

۱ – ۹ – ۱. عملگرهای محاسباتی

این عملگرها برای انجام محاسبات بر روی داده‌های عددی به کار می‌روند. (جدول ۳ – ۱). ازجمله این عملگرها می‌توان عملگرهای + (جمع)، – (تفریق)، * (ضرب)، / (تقسیم)، ٪ (باقی‌مانده تقسیم صحیح)، ++ (افزایش) و -- (کاهش) را نام برد. با عملگرهای +، –، * و / از قبل آشنا هستید. عملگر ٪ برای محاسبه باقی‌مانده تقسیم صحیح به کار می‌رود.

[۵] .operators

- در برنامه‌نویسی ایده‌ال، در هر دو طرف یک عملگر دودویی (عملگرهایی که نیاز به دو عملوند دارند) یک یا چند فاصله قرار دهید. فاصله‌ها باعث متمایز شدن نقش عملگر شده و خوانائی عبارت افزایش می‌یابد.
- برخی از زبان‌های برنامه‌نویسی از عملگرهای ** یا ^ برای نمایش توان استفاده می‌کنند. درحالی‌که زبان C++ از این عملگرها پشتیبانی نمی‌کند، و استفاده از آن‌ها خطای نحوی خواهد بود.

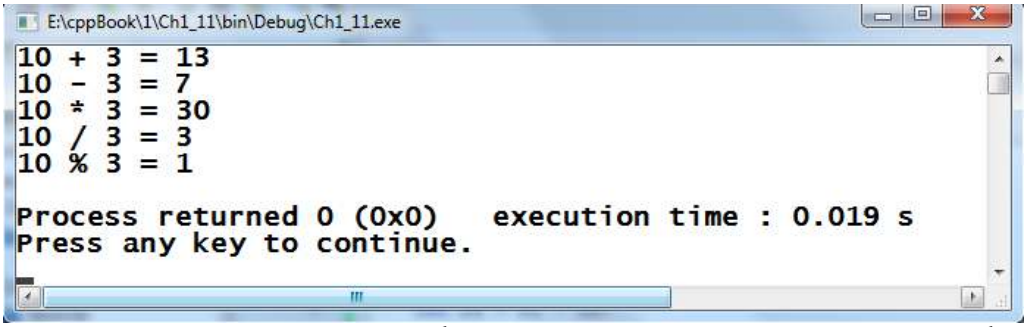
مثال ۱۱ – ۱. برنامه‌ای که کاربرد عملگرهای محاسباتی را نشان می‌دهد.

۱. پروژه جدیدی به نام Ch1_11 ایجاد کرده، دستورات برنامه را به‌صورت زیر تغییر دهید:

```
#include <iostream>
using namespace std;
int main()
{
   int n1 = 10, n2 = 3;
   int n3 = n1 + n2;
   int n4 = n1 - n2;
   int n5 = n1 * n2;
   int n6 = n1 / n2;
   int n7 = n1 % n2;
   cout << n1 << " + " << n2 << " = " <<  n3 << endl;
   cout << n1 << " - " << n2 << " = " <<  n4 << endl;
   cout << n1 << " * " << n2 << " = " << n5 << endl;
   cout << n1 << " / " << n2 << " = " << n6 << endl;
   cout << n1 << " % " << n2 << " = " << n7 << endl;
   return 0;
}
```

دستور پنجم، متغیرهای n1 و n2 را به ترتیب با مقادیر ۱۰ و ۳ تعریف می‌کند، دستور ششم، متغیر n3 را تعریف کرده، مقدار n1 + n2 را در آن قرار می‌دهد، دستور هفتم، متغیر n4 را تعریف کرده، n1 - n2 را در آن قرار می‌دهد، دستور هشتم، n5 را تعریف کرده، مقدار n1 * n2 را در آن قرار می‌دهد، دستور نهم، متغیر n6 را تعریف کرده، حاصل عبارت n1 / n2 را در آن قرار می‌دهد، دستور دهم، متغیر n7 را تعریف کرده، حاصل n1 % n2 (باقی‌مانده تقسیم صحیح n1 بر n2) را در آن قرار می‌دهد، دستور یازدهم، مقدار 10 + 3 = 13 را نمایش می‌دهد، دستور دوازدهم، مقدار 10 - 3 = 7 را نمایش خواهد داد، دستور سیزدهم، مقدار 10 * 3 = ۳۰ را نمایش می‌دهد، دستور چهاردهم، عبارت 10 / 3 = 3 را چاپ خواهد کرد (چون n1 و n2 صحیح هستند، نتیجه تقسیم نیز عددی صحیح خواهد شد) و دستور پانزدهم، عبارت 10 % 3 = 1 را نمایش می‌دهد (چون باقی‌مانده تقسیم صحیح ۱۰ بر ۳ برابر ۱ است).

۲. پروژه را ذخیره و اجرا کنید تا خروجی را به‌صورت زیر مشاهده نمایید:



عملگر ++ یک واحد به محتویات عملوند اضافه می‌کند. اما، عملگر -- یک واحد از محتویات عملوند کم خواهد کرد. چنانچه عملگرهای ++ و -- قبل از عملوند قرار گیرند، ابتدا یک واحد به محتویات عملوند اضافه کرده یا از آن کسر می‌کند. سپس، عبارت ارزیابی می‌گردد (مثال‌های زیر را ببینید).

- مبادرت به استفاده از عملگر ++ یا -- بر روی عبارتی به‌جز نام یک متغیر، همانند (x + 1)++ خطای نحوی خواهد بود.

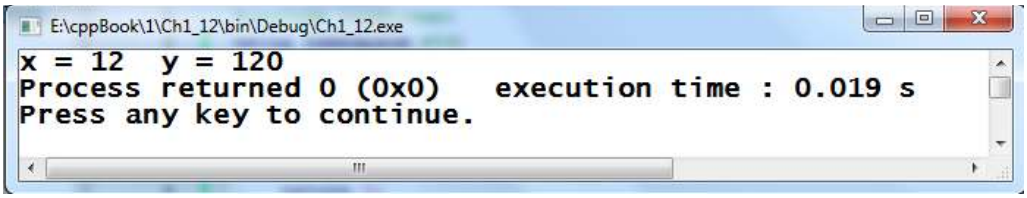
مثال ۱۲ - ۱. پس از اجرای دستورات زیر مقدار x و y چند است؟

۱. پروژه جدیدی به نام Ch1_12 ایجاد کرده، دستورات برنامه را به‌صورت زیر تغییر دهید:

```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5      int x = 10, y;
6      y = x ++ * ++ x;
7      cout << "x = " << x << "\ty = " << y;
8      return 0;
9  }
```

x = 12 و y = 120. چون در دستور ششم ابتدا عملگر * انجام می‌شود، یعنی ۱۰ در ۱۰ ضرب شده مقدار ۱۲۰ در y قرار می‌گیرد. سپس دو واحد به x اضافه‌شده، یعنی x برابر ۱۲ می‌شود.

۲. پروژه را ذخیره و اجرا کنید تا خروجی زیر را مشاهده نمایید:



مثال ۱۳ - ۱. برنامه‌ای که عملکرد عملگرهای ++ و -- را نشان می‌دهد.

۱. پروژه جدیدی به نام Ch1_13 ایجاد کرده، دستورات برنامه را به‌صورت زیر تغییر دهید:

```
#include <iostream>
using namespace std;
int main()
{
    int n1 = 10, n2 = 20;
    int n3 = n1 --;
    int n4 = -- n2;
    int n5 =  n1 ++;
    int n6 = ++ n2 ;
    cout << "n1 = " << n1 << " \t n2 = " << n2 << endl;
    cout << "n3 = " << n3 << " \t n4 = " << n4 << endl;
    cout << "n5 = " << n5 << " \t n6 = " << n6 << endl;
    return 0;
}
```

دستور پنجم متغیرهای n1 و n2 را با نوع int و مقادیر ۱۰ و ۲۰ تعریف می‌کند، دستور ششم، ابتدا متغیر n3 را تعریف کرده، مقدار ۱۰ را در آن قرار می‌دهد و سپس یک واحد از n1 کم می‌کند (یعنی، معادل دستورات زیر است):

```
n3 = n1;
n1 = n1 – 1;
```

دستور هفتم، ابتدا متغیر n4 را با نوع int تعریف کرده، مقدار ۱۹ را در آن قرار می‌دهد. زیرا معادل، دستورات زیر است:

```
n2 = n2 - 1;
n4 = n2;
```

دستور هشتم، ابتدا متغیر n5 را با نوع int تعریف کرده، مقدار ۹ را در آن قرار می‌دهد و سپس به n۱ یک واحد اضافه می‌نماید (معادل دو دستور زیر است):

```
n5 = n1;
n1 = n1 + 1;
```

دستور نهم، متغیر n6 را تعریف کرده، ابتدا به n2 یک واحد اضافه می‌کند (یعنی، n2 برابر با ۲۰ می‌شود) و این مقدار را در n6 قرار می‌دهد (این دستور معادل دستورات زیر است):

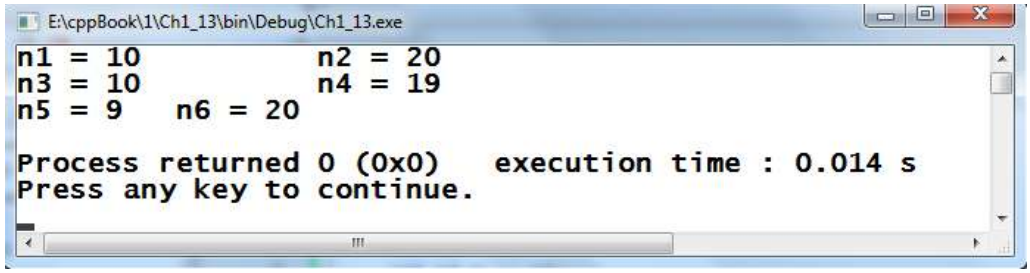
```
n2 = n2 + 1;
n6 = n2;
```

دستور دهم، عبارت، n2 = 20 n1 = 10 را نمایش می‌دهد.

دستور یازدهم، عبارت n4 = 19 n3 = 10 را نمایش خواهد داد.

دستور دوازدهم، عبارت n6 = 20 n5 = 9 را چاپ می‌نماید.

۲. پروژه را ذخیره و اجرا کنید تا خروجی زیر را مشاهده نمایید:



۲ – ۹ – ۱. عملگرهای رابطه‌ای (مقایسه‌ای)

این عملگرها برای مقایسه دو عملوند به کار می‌روند و نتیجه درست یا نادرست را برمی‌گردانند. در C++ هر عدد غیر صفر به معنی درست می‌باشد و عدد صفر نادرست بودن را نشان می‌دهد. عملگرهای رابطه‌ای (مقایسه‌ای) در جدول ۴ – ۱ آمده‌اند. دقت کنید عملگر = = تساوی (مساوی بودن) می‌باشد.

جدول ۴ – ۱ عملگرهای رابطه‌ای (مقایسه‌ای).

عملگر	نام عملگر	مثال	نتیجه	توضیحات
>	بزرگ‌تر	2 > 3	false	اگر عملوند اول بزرگ‌تر از عملوند دوم باشد، نتیجه درست است، وگرنه نتیجه نادرست می‌باشد.
>=	بزرگ‌تر مساوی	5 >= 3	true	اگر عملوند اول بزرگ‌تر یا مساوی عملوند دوم باشد، نتیجه درست است، وگرنه، نتیجه نادرست می‌باشد.
<	کوچک‌تر	5 < 7	true	اگر عملوند اول کوچک‌تر از عملوند دوم باشد، نتیجه درست است. وگرنه، نتیجه نادرست است.
<=	کوچک‌تر یا مساوی	5 <= 3	false	اگر عملوند اول کوچک‌تر یا مساوی عملوند دوم باشد، نتیجه درست است، وگرنه نتیجه نادرست خواهد شد.
!=	نامساوی	! = 5 2	true	اگر عملوند اول مخالف عملوند دوم باشد، نتیجه درست است، وگرنه، نتیجه نادرست خواهد بود.
==	تساوی	2 == 3	false	اگر عملوند اول مساوی عملوند دوم باشد، نتیجه درست است، وگرنه، نتیجه نادرست خواهد شد.

- درصورتی‌که مابین هر کدام‌یک از عملگرهای ==، =!، =< و => فاصله قرار دهید با خطای نحوی مواجه خواهید شد.
- معکوس نوشتن عملگرهای =!، =< و => به‌صورت !=، <= و >= خطای نحوی به دنبال خواهد داشت. در برخی از موارد نوشتن عملگر =! به‌صورت != خطای نحوی تلقی نمی‌شود اما به‌صورت یک خطای منطقی و در زمان اجرای برنامه تأثیر خود را نشان می‌دهد.

- اشتباه گرفتن رفتار عملگر رابطه‌ای == با عملگر تخصیص = می‌تواند خطای منطقی به دنبال داشته باشد.

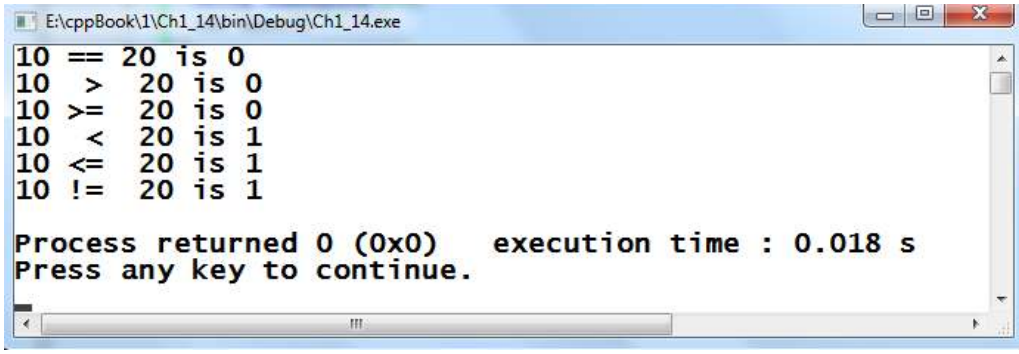
مثال ۱۴ – ۱. برنامه‌ای که عملکرد عملگرهای مقایسه‌ای را نشان می‌دهد.

۱. پروژه جدیدی به نام Ch1_14 ایجاد کرده، دستورات برنامه را به‌صورت زیر تغییر دهید:

```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5     int n1 = 10, n2 = 20;
6     cout << n1 << " == "  << n2 << " is " << (n1 == n2)  << endl;
7     cout << n1 << "  >  " << n2 << " is " <<  (n1 > n2)  << endl;
8     cout << n1 << " >=  " << n2 << " is " <<  (n1 >= n2) << endl;
9     cout << n1 << "  <  " << n2 << " is " <<  (n1 < n2)  << endl;
10    cout << n1 << " <=  " << n2 << " is " <<  (n1 <= n2) << endl;
11    cout << n1 << " !=  " << n2 << " is " << (n1 != n2)  << endl;
12    return 0;
13 }
```

دستور پنجم، متغیرهای n1 و n2 را با نوع int و مقادیر ۱۰ و ۲۰ تعریف می‌کند، دستور ششم، عبارت 10 == 20 is 0 را نمایش می‌دهد، (چون ۱۰ برابر با ۲۰ نیست، نتیجه عملگر == برابر ۰ (همان false) خواهد بود)، دستور هفتم، عبارت 10 > 20 is 0 را نمایش می‌دهد، چون ۱۰ از ۲۰ بزرگ‌تر نیست، دستور هشتم، عبارت 10 >= 20 is 0 را نمایش می‌دهد، دستور نهم، عبارت 10 < 20 is 1 را نمایش می‌دهد (۱۰ از ۲۰ کوچک‌تر است). دستور دهم، عبارت 10 <= 20 is 1 را نمایش می‌دهد و دستور یازدهم، 10 != 20 is 1 را نمایش می‌دهد.

۲. پروژه را ذخیره و اجرا کنید تا خروجی زیر را مشاهده نمایید:



۳ – ۹– ۱. عملگرهای ترکیبی

این عملگرها، ترکیبی از عملگرهای محاسباتی و = هستند. عملکرد این عملگرها را در جدول ۵ – ۱ می‌بینید.

مثال ۱۵ – ۱. برنامه‌ای که عملکرد عملگرهای ترکیبی را نشان می‌دهد.

۱. پروژه جدیدی به نام Ch1_15 ایجاد کرده، دستورات برنامه را به‌صورت زیر تغییر دهید:

```
#include <iostream>
using namespace std;
int main()
{
  int n1 = 10, n2 = 20;
  n1 += n2;
  cout << "n1 = " << n1 << " \t n2 = " << n2 << endl;
  n1 -= n2;
  cout << "n1 = " << n1 << " \t n2 = " << n2 << endl;
  n1 *= n2;
  cout << "n1 = " << n1 << " \t n2 = " << n2 << endl;
  n1 /= n2;
  cout << "n1 = " << n1 << " \t n2 = " << n2 << endl;
  n1 %= n2;
  cout << "n1 = " << n1 << " \t n2 = " << n2 << endl;
  return 0;
}
```

دستور پنجم تابع main()، متغیرهای n1 و n2 را با نوع int و به ترتیب با مقادیر ۱۰ و ۲۰ تعریف می‌کند، دستور ششم، n1 را برابر ۳۰ قرار می‌دهد، زیرا:

n1 + = n2 ⇨ n1 = n1 + n2 = 10 +20

دستور هفتم، عبارت n2 = 20 n1 = 30 را نمایش می‌دهد، دستور هشتم، n۱ را به مقدار ۱۰ برمی‌گرداند، زیرا:

n1 -= n2 ⇨ n1 = n1 – n2 = 30 – 10 = 10

دستور نهم، مقدار n2 = 20 n1 = 10 را نمایش می‌دهد، دستور دهم، مقدار n۱ را به ۲۰۰ تغییر می‌دهد، زیرا:

n1 *= n2 ⇨ n1 = n1 * n2 = 10 *20 = 200

دستور یازدهم، مقدار n2 = 20 n1= 200 را نمایش می‌دهد، دستور دوازدهم، مقدار n۱ را به همان ۱۰ برمی‌گرداند، زیرا:

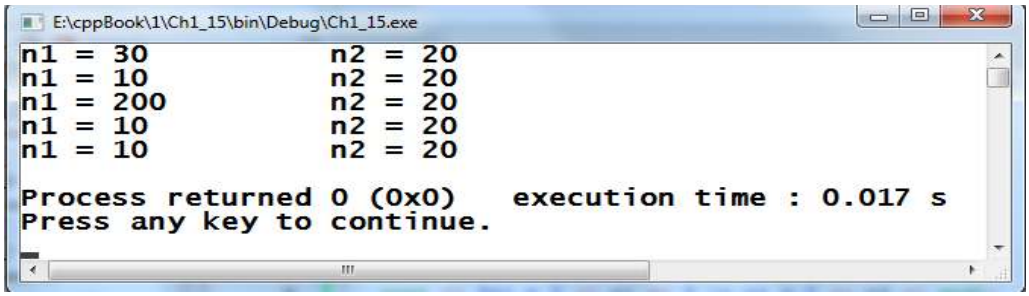
n1 /= n2 ⇨ n1 = n1 / n2 = 200 / 20 = 10

دستور سیزدهم، مقدار n2 = 20 n1= 10 را نمایش می‌دهد، دستور چهاردهم، مقدار n۱ را تغییر نمی‌دهد، زیرا:

n1 % = n2; ⇨ n1 = n1 % n2 = 10 % 20 = 10

دستور پانزدهم، مقدار n2 = 20 n1= 10 را نمایش می‌دهد.

۲. پروژه را ذخیره و اجرا کنید تا خروجی زیر را مشاهده نمایید:



۴ – ۹ – ۱. عملگرهای منطقی

عملگرهای منطقی، بر روی عبارات منطقی درست یا نادرست عمل می‌کنند. نتیجه عملگرهای منطقی در جدول ۶ – ۱ آمده است. همان‌طور که در جدول ۶ – ۱ می‌بینید، هنگامی نتیجه عملگر && (و منطقی) درست است که هر دو عملوند نتیجه درست داشته باشند. اما نتیجه عملگر || (یا منطقی) هنگامی نادرست است که هر دو عملوند نادرست باشند.

جدول ۵ – ۱ عملگرهای ترکیبی.

عملگر	روش استفاده	مثال	نتیجه	عملکرد
+=	x += y;	x = ۳; x += ۵;	8	x = x + y;
-=	x -= y;	x = ۷; x -= ۳;	4	x = x - y;
*=	x *= y;	x = ۳; x *= ۵;	15	x = x * y;
/=	x /= y;	x = ۱۷; x /= ۵;	3	x = x / y;
%=	x%= y;	x = ۱۷; x %= ۵;	2	x = x % y;

جدول ۶ – ۱ عملکرد عملگرهای منطقی.

X	Y	x && y	x || y	!x	!y
False	False	false	false	true	true
True	False	false	true	True	false
False	True	false	true	False	true
True	True	true	true	False	false

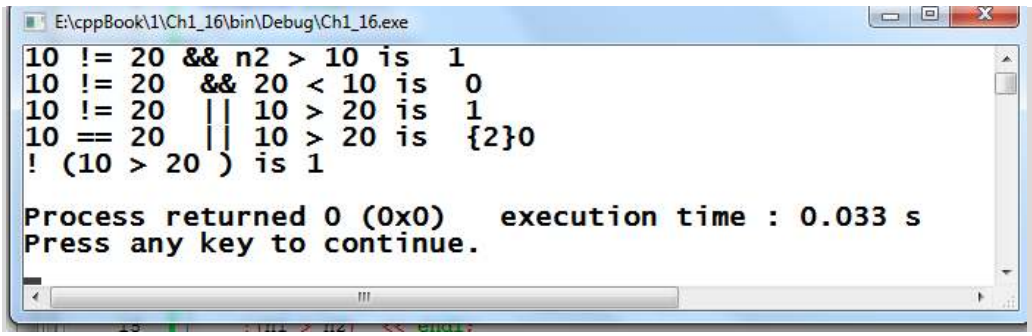
مثال ۱۶ – ۱. برنامه‌ای که عملکرد عملگرهای منطقی را نشان می‌دهد.

۱. پروژه جدیدی به نام Ch1_16 ایجاد کرده، دستورات برنامه را به‌صورت زیر تغییر دهید:

```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5      int n1 = 10, n2 = 20;
6      cout << n1 << " != " <<  n2 << " && n2 > " << n1 <<
7            " is  " << (n1 != n2 && n2 > n1) << endl;
8     cout << n1 << " != " <<  n2 << "  && " <<  n2 << " < "
9             << n1 << " is  " << ( n1 != n2 && n2 < n1) << endl;
10    cout << n1 << " != " <<  n2 << "  || " << n1 << " > "
11            <<  n2 << " is  " << (n1 != n2 || n1 > n2) << endl;
12    cout << n1 << " == " <<  n2 << "  || " << n1 << " > "
13            <<  n2 << " is  {2}"  << ( n1 == n2 || n1 > n2) << endl;
14    cout << "! (" << n1 << " > " <<  n2 << " ) is " <<
15            !(n1 > n2)  << endl;
16    return 0;
17 }
```

دستور پنجم، متغیرهای n1 و n2 را با نوع int تعریف کرده، مقادیر ۱۰ و ۲۰ را به آن‌ها تخصیص می‌دهد، دستور ششم، عبارت 10 != 20 && 20 > 10 is 1 را نمایش می‌دهد (چون 20 != 10 است و 10 < 20 می‌باشد، سپس نتیجه عملگر && برابر ۱ یا همان true است). دستور هشتم، عبارت 10 != 20 && 20 < 10 is 0 را نمایش می‌دهد (چون ۲۰ کوچک‌تر از ۱۰ نیست)، پس نتیجه && نادرست (یعنی ۰ یا همان false) می‌باشد. دستور دهم، عبارت 10 != 20 || 10 > 20 is 1 را نمایش می‌دهد (چون ۱۰ مخالف ۲۰ است) پس نتیجه || برابر true خواهد بود). دستور دوازدهم، عبارت 10 == 20 || 10 > 20 is 0 را نمایش می‌دهد (چون هر دو عبارت 10 == 20 و 10 > 20 نادرست هستند، نتیجه عبارت || نادرست می‌باشد و دستور چهاردهم، عبارت ! (10 > 20) is 1 را نمایش می‌دهد (چون ۱۰ بزرگ‌تر از ۲۰ نیست، پس !(10 > 20) درست (true) می‌باشد.

۲. پروژه را ذخیره و اجرا کنید تا خروجی زیر را مشاهده نمایید:



۵ – ۹ – ۱. عملگرهای خاص

علاوه بر عملگرهای بیان‌شده، برخی از عملگرها در C++ کاربرد خاصی دارند. این عملگرها را در زیر می‌بینید:

۱ **.عملگر؟:** برای بررسی شرط خاصی به کار می‌رود و به‌صورت زیر استفاده می‌شود:

; عبارت ۲ : عبارت ۱ ? (شرط)

در این ساختار ابتدا شرط ارزیابی می‌شود (شرط می‌تواند یک شرط ساده یا ترکیبی باشد)، اگر نتیجه ارزیابی شرط درست باشد، **عبارت ۱** انجام می‌شود، وگرنه، **عبارت ۲** انجام خواهد شد (مثال زیر را ببینید).

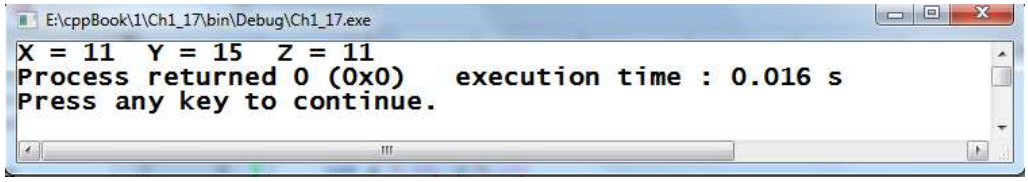
مثال ۱۷ – ۱. برنامه‌ای که کاربرد عملگر ؟ را نشان می‌دهد.

۱. پروژه جدیدی به نام Ch1_17 ایجاد کرده، دستورات برنامه را به‌صورت زیر تغییر دهید:

```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5      int x = 10, y = 15;
6      int z = (x > y || y < 17) ? ++x : ++y;
7      cout << "X = " << x << "\tY = " << y << "\tZ = " << z;
8      return 0;
9  }
```

دستور پنجم، x و y را با نوع int تعریف کرده، مقادیر ۱۰ و ۱۵ را به ترتیب به x و y تخصیص می‌دهد. دستور ششم، ابتدا نتیجه عبارت شرطی (x > y || y <17) را بررسی می‌کند که درست می‌باشد. زیرا، 15 < 10 نیست، اما 17 > 15 است. بنابراین، true || false برابر true می‌باشد. پس، عبارت ۱ (یعنی، ++x) انجام خواهد شد. یعنی، ابتدا به x یک واحد اضافه می‌شود (مقدار x برابر با ۱۱ خواهد شد) و مقدار ۱۱ در z قرار می‌گیرد. دستور هفتم، عبارت X = 11 Y = 15 Z = 11 را نمایش می‌دهد.

۲. پروژه را ذخیره و اجرا کنید تا خروجی زیر را مشاهده نمایید:



مثال ۱۸ – ۱. خروجی دستورات زیر چیست؟

۱. پروژه جدیدی به نام Ch1_18 ایجاد کرده، دستورات برنامه را به‌صورت زیر تغییر دهید: