



دانشگاه ساه نور  
په

## کارگاه مبانی کامپیوتر و برنامه سازی

رشته مهندسی کامپیوتر

کمال محمدی اصل

زهرا کمال

## بسم الله الرحمن الرحيم

### پیشگفتار

از آنجایی که منبع مشخص و منسجمی جهت تدریس کد درس ۱۳۲۲۰۸۸ با عنوان "کارگاه مبانی کامپیوتر و برنامه‌سازی" برای دانشجویان رشته مهندسی کامپیوتر در دسترس نیست، بر آن شدیم، چالش‌هایی که دانشجویان عزیز با آن مواجه هستند و همچنین اشکالاتی که ممکن است یک برنامه‌نویس، در ابتدای راه با آن مواجه شود، را با استفاده از تجربه حاصل از کلاس درس و کارگاه برنامه‌نویسی، در قالب این مجموعه‌ی گردآوری شده، رفع نماییم.

در این مجموعه سعی بر آن بوده که از مفاهیم اولیه آغاز و به تدریج مطالب پیچیده‌تر مطرح شود. امتیاز بارز این مجموعه، کاربردی و عملیاتی بودن مطالب آن می‌باشد.

در نهایت، از همه شما اساتید گرامی، که وقت شریف خود را برای مطالعه این اثر می‌گذارید، تقاضا داریم ما را از نظرات سازنده، پیشنهادات و انتقادات خود، بهره‌مند سازید.

با تشکر

## فهرست مطالب:

### فصل اول: فلوجارت‌ها و نرم‌افزار فلوگوریتیم

۱-۱	مقدمه‌ای راجع به فلوگوریتیم.....	۱
۱-۱-۱	خصوصیات نرم‌افزار فلوگوریتیم.....	۱
۱-۱-۲	نصب نرم‌افزار فلوگوریتیم.....	۳
۱-۲	برنامه‌سازی با فلوگوریتیم.....	۴
۱-۲-۱	انواع داده‌ها در فلوگوریتیم.....	۴
۱-۲-۲	شناسه‌ها.....	۴
۱-۲-۳	عملگرها.....	۵
۱-۲-۴	معرفی اشکال هندسی در فلوگوریتیم.....	۷
۱-۳	رسم فلوجارت در نرم‌افزار فلوگوریتیم.....	۸
۱-۳-۱	اجرای فلوجارت.....	۱۱
۱-۳-۲	تبدیل فلوجارت به سورس کد.....	۱۴
۱-۴	ذخیره‌سازی و تبدیل فلوجارت به تصویر.....	۳۰
۱-۵	تغییر زبان برنامه.....	۳۱
	تمرینات آخر فصل.....	۳۲

### فصل دوم: برنامه‌نویسی به زبان ++C در محیط توسعه کدبلاکس

۲-۱	مقدمه‌ای راجع به کامپایلر و IDE.....	۳۴
۲-۲	راهنمای نصب کامپایلر و اجرای برنامه‌های C, C++.....	۳۷
۲-۲-۱	کامپایلر روی کامپیوترهای شخصی.....	۳۷
۲-۲-۲	کامپایلر روی گوشی‌های هوشمند و یا تبلت.....	۳۷
۲-۳	نحوه ایجاد کردن پروژه جدید در نرم‌افزار کدبلاکس و اجرای آن.....	۳۸
۲-۴	خطایابی و انواع خطاها.....	۴۶
۲-۴-۱	نمونه‌هایی از خطاهای زمان کامپایل.....	۴۸

- ۲-۴-۲. نمونه‌هایی از خطاهای زمان اجرا..... ۵۱
- ۲-۴-۲-۱. سرریزی عدد اعشاری ..... ۵۱
- ۲-۴-۲-۲. خطای گرد کردن..... ۵۲
- ۲-۴-۳. روش‌های یافتن خطاهای منطقی و برطرف کردن آنها ..... ۵۴
- ۲-۴-۳-۱. نمونه‌هایی از خطاهای منطقی ..... ۵۹
- ۲-۵. کامپایل جداگانه توابع و فایل DLL ..... ۶۶
- ۲-۵-۱. نحوه ساخت فایل DLL ..... ۶۸
- ۲-۵-۲. نحوه استفاده از فایل DLL در برنامه‌های دیگر..... ۷۰
- ۲-۶. ساختن فایل کتابخانه‌ای (فایل سرآیند)..... ۷۲
- ۲-۷. تفاوت‌های فایل کتابخانه‌ای و فایل DLL ..... ۷۶
- تمرینات برنامه‌نویسی آخر فصل ..... ۷۷

# فصل اول

## فلوچارت‌ها و نرم‌افزار فلوگوریتم

### هدف کلی

درک عمیق مسئله و رسم فلوچارت به صورت کاملاً بصری

### هدف‌های رفتاری

انتظار می‌رود پس از مطالعه این فصل بتوانید:

- ویژگی‌های نرم‌افزار فلوگوریتم را شرح دهید.
- با انواع داده‌ها در فلوگوریتم آشنا باشید.
- نحوه تعریف شناسه‌ها را بدانید.
- انواع عملگرها را بشناسید.
- اشکال هندسی نرم‌افزار فلوگوریتم را بشناسید.
- توانایی رسم و اجرای فلوچارت در نرم‌افزار فلوگوریتم را داشته باشید.
- توانایی تبدیل فلوچارت به کد برنامه را داشته باشید.
- توانایی پیاده‌سازی انواع ساختارهای شرطی و حلقه‌های تکرار را داشته باشید.
- توانایی تعریف و پیاده‌سازی انواع توابع فرعی را داشته باشید.
- با روش‌های تبدیل فلوچارت به تصاویر با فرمت‌های مختلف آشنا باشید.



## ۱-۱. مقدمه‌ای راجع به فلوگوریتیم

فلوگوریتیم<sup>۱</sup> یک ابزار کمک آموزشی برای کسانی است که می‌خواهند برنامه نویسی با کامپیوتر را یاد بگیرند. فلوگوریتیم یک نرم‌افزار رایگان و مبتنی بر نمودارهای گرافیکی یعنی همان فلوجارت است.

واژه فلوگوریتیم (Flowgorithm) از ترکیب دو واژه فلوجارت (Flowchart) و الگوریتیم (Algorithm) تشکیل شده است. در واقع این نرم‌افزار به ما کمک می‌کند تا با استفاده از فلوجارت و تاکیدی که روی الگوریتیم دارد، به حل مسئله بپردازیم.

این نرم‌افزار قابلیت اجرای فلوجارتهی که توسط کاربر کشیده شده است را دارد و همچنین قابلیت تبدیل فلوجارت به زبان‌های برنامه‌نویسی سطح بالا مثل C++، Java، C# و ... را دارد. این نرم‌افزار کاربردی در سال ۲۰۱۴ توسط آقای دیون کوک<sup>۲</sup> در کالج ساکرومنتواستیت<sup>۳</sup> طراحی شده است.

لازم به ذکر است، نرم‌افزارهای دیگری هم در این زمینه وجود دارند که تعدادی از بهترین‌های آن‌ها عبارتند از:

- [AlgoBuild](#)
- [LARP](#)
- [PSeInt](#) - Spanish
- [Raptor](#)
- [Visual Logic](#)

### ۱-۱-۱. خصوصیات نرم‌افزار فلوگوریتیم

❖ محیط کنسولی آن برخلاف محیط کنسولی IDE ها، سیاه و سفید نیست و کاربر با یک محیط ساده و کلاسیک سروکار ندارد.

❖ در حین اجرای برنامه، می‌توان متغیرها را مشاهده نمود. (همانطور که می‌دانیم وقتی فلوجارت را روی کاغذ یا در محیط‌های گرافیکی دیگر رسم می‌کنیم نیازی به تعریف نوع متغیرها نداریم. اما در این نرم‌افزار باید نوع متغیرها را مشخص نماییم.

نوع متغیرهایی که در فلوگوریتیم داریم عبارتند از: Boolean و String، Real، Integer

<sup>۱</sup>.Flowgorithm

<sup>۲</sup>.Devin Cook

<sup>۳</sup>.Sacramento State

❖ فلوگوریتیم می‌تواند، فلوچارت شما را به بیش از ۲۱ زبان برنامه‌نویسی تبدیل کند.

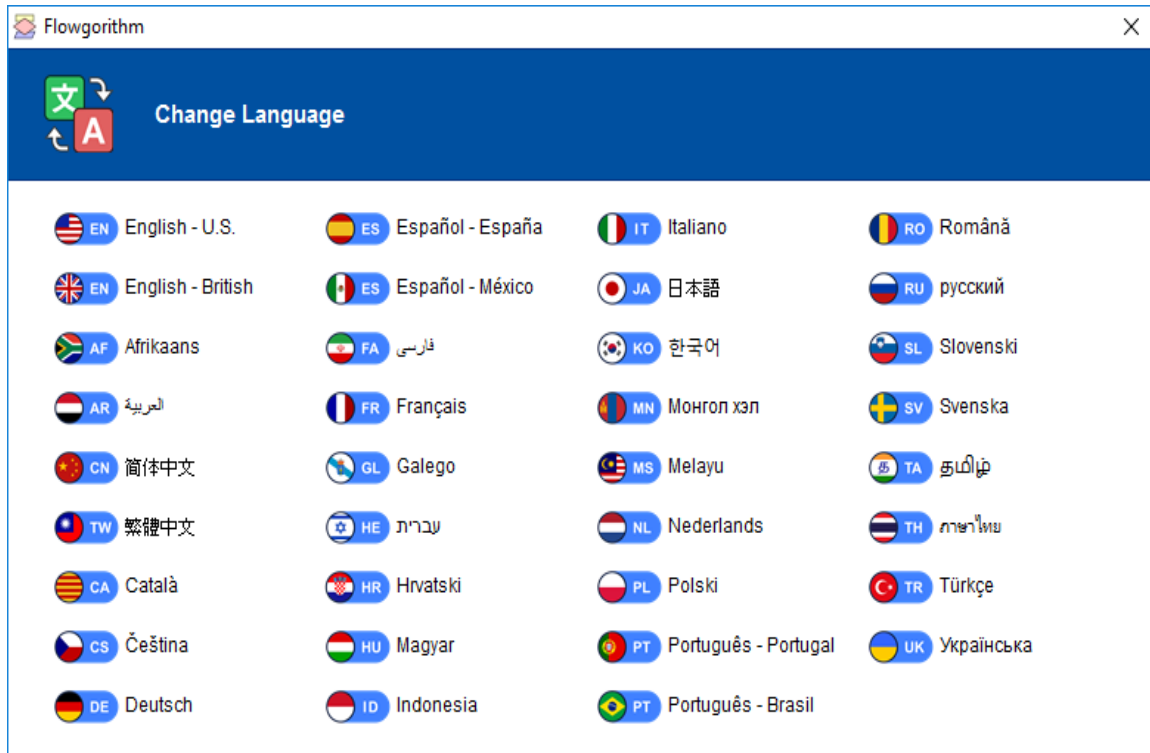


شکل ۱-۱: زبان‌های برنامه‌نویسی در فلوگوریتیم

- ❖ طرح‌های رنگی فلوچارت‌ها بسیار متنوع است.
- ❖ امکان تعریف آرایه‌های تک بعدی را دارد.
- ❖ امکان استفاده از ۳ حلقه معروف For، While و Do\_While را دارد.
- ❖ از پشته سیستم عامل استفاده نمی‌کند.
- ❖ از اپراتورهای خانواده C و خانواده Basic، همزمان پشتیبانی می‌نماید.
- ❖ امکان تعریف تابع توسط کاربر وجود دارد.
- ❖ ۲۰ تابع داخلی نیز وجود دارد.
- ❖ سبک‌های مختلف فلوچارت‌ها را پشتیبانی می‌کند. مثل IBM، SDL، Classic و ...
- ❖ از فلوچارت‌ها می‌توان، به صورت فایل PNG و یا SVG (بررداری) خروجی گرفت.



- ❖ خود نرم‌افزار هم فلوجارت‌ها را با فرمت \*.fprg ذخیره می‌کند.
- ❖ این نرم‌افزار Multi Language (چند زبانه) است و از زبان فارسی نیز پشتیبانی می‌کند.



شکل ۱-۲: تغییر زبان نرم‌افزار فلوگوریتم

### ۱-۲-۱. نصب نرم‌افزار فلوگوریتم

برای نصب نرم‌افزار فلوگوریتم، کافی است به سایت <http://flowgorithm.org/download> مراجعه کرده و بر اساس نسخه ویندوزتان، لینک مربوطه را دانلود و سپس آن را نصب نمایید.

## ۱-۲. برنامه‌سازی با فلوگوریتیم

برای برنامه‌نویسی با فلوگوریتیم یکسری قواعد و اصولی وجود دارد که ابتدا باید آنها را فرا بگیریم.

### ۱-۲-۱. انواع داده‌ها<sup>۴</sup> در فلوگوریتیم

از آنجایی که در نرم‌افزار فلوگوریتیم قرار است تبدیل از فلوچارت به یک زبان برنامه‌نویسی را داشته باشیم، لذا باید نوع متغیرها مشخص باشند.

جدول ۱-۱: انواع داده‌ها


نوع داده	توضیحات
Boolean	Stores either Boolean true or false (منطقی (True یا False)
Real	Stores a real number. (اعداد اعشاری)
Integer	Stores an integer number. (اعداد صحیح)
String	Stores textual data. (رشته‌ای یا متنی)

### ۱-۲-۲. شناسه‌ها<sup>۵</sup>

شناسه اسمی ست که به عناصر مختلف برنامه مثل متغیرها، توابع، آرایه‌ها و ... اختصاص می‌یابد. برای تعریف این اسامی باید از قوانینی پیروی کرد.

#### قوانین تعریف شناسه‌ها (قوانین نام‌گذاری‌ها)

- ❖ باید با یک حرف شروع شود.
- ❖ در ادامه می‌توانیم فقط حروف یا اعداد را داشته باشیم.
- ❖ استفاده از کلمات کلیدی مجاز نیست.

نکته:  فلوگوریتیم نسبت به بزرگ و کوچک بودن حروف حساسیتی ندارد.

<sup>۴</sup> . Data Types

<sup>۵</sup> . Identifiers

## ۱-۲-۳. عملگرها

از آنجایی که فلوگوریتم یک ابزار کمک آموزشی است بنابراین به گونه‌ای طراحی شده که هم از عملگرهای مشتق شده از زبان C و هم از عملگرهای مشتق شده از زبان Basic پشتیبانی می‌کند.

جداول زیر لیست عملگرها را نشان می‌دهند:

جدول ۱-۲: عملگرهای فلوگوریتم

Operator (عملگر)	C Family (خانواده سی)	BASIC Family (خانواده بیسیک)	توضیحات
Equality	==	=	تساوی
Inequality	!=	<>	عدم تساوی (نامساوی)
Less Than or Equal	<=	<=	کوچک‌تر از یا مساوی
Greater Than Or Equal	>=	>=	بزرگتر از یا مساوی
Logical Not	!	not	Not منطقی
Logical And	&&	and	And منطقی
Logical Or		or	Or منطقی
Multiply	*	*	ضرب
Divide	/	/	تقسیم
Modulo	%	mod	باقی مانده

جدول ۱-۳: ادامه عملگرها

Operator (عملگر)	BASIC Family (خانواده بیسیک)	توضیحات
String Concatenation	&	الحاق دو یا چند رشته به یکدیگر
Exponent	^	توان

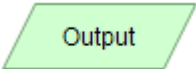

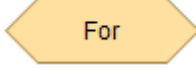
اولویت عملگرها به شرح زیر می‌باشد:

جدول ۱-۴: اولویت عملگرها

Level (سطح)	نام	عملگر
۸	Unary	- ! not ¬
۷	Exponent	^ ↑
۶	Multiply	* x / ÷ % mod
۵	Addition	+ -
۴	Concatenate	&
۳	Relational	> < >= ≥ <= ≤ == = != <> ≠
۲	Logical And	and && ∧
۱	Logical Or	or    ∨

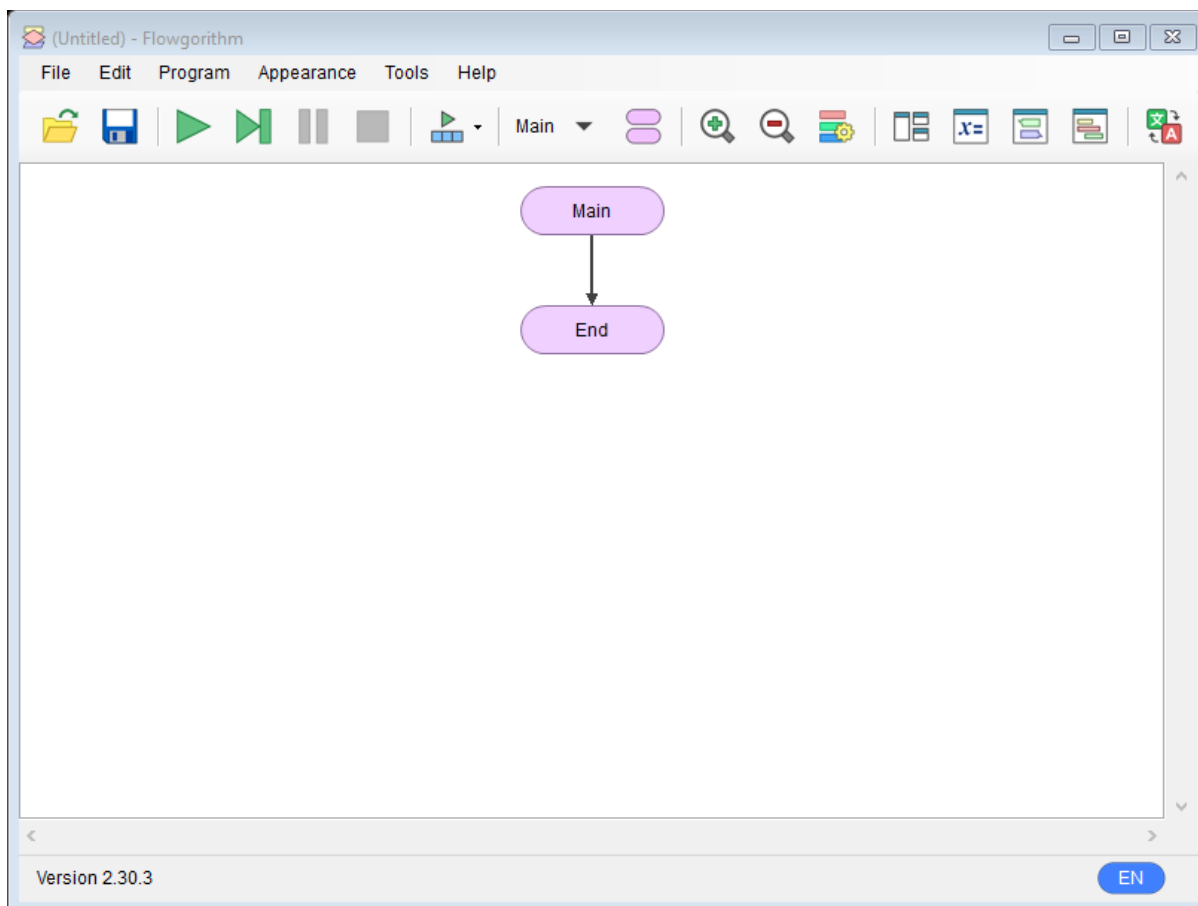
## ۱-۲-۴. معرفی اشکال هندسی در فلوگوریتیم (Shapes)

جدول ۱-۵: معرفی اشکال هندسی

نام شکل	شکل موردنظر	توضیحات
Assignment		عملیات محاسباتی، مقداردهی به یک متغیر
Declare		تعریف نوع متغیر
Input		دریافت ورودی
Output		چاپ خروجی
If		دستور شرطی If
While		حلقه While
For		حلقه For
Do		حلقه Do-While
Call		فراخوانی تابع
Comment		درج توضیحات در برنامه
Breakpoint		نقطه انفصال (برنامه تا این نقطه اجرا می‌شود)

## ۱-۳. رسم فلوچارت در نرم‌افزار فلوگوریتم

وقتی برنامه فلوگوریتم را اجرا می‌کنیم با پنجره زیر مواجه می‌شویم.



شکل ۱-۳: صفحه آغازین نرم‌افزار فلوگوریتم


این پنجره به ترتیب از نوار عنوان، نوار منو، نوار ابزار و محیطی برای رسم فلوچارت تشکیل شده است.

در محیط رسم فلوچارت، به طور پیش فرض، اشکال مربوط به شروع (Main) و پایان (End) فلوچارت که با یک فلش به هم متصل شده‌اند، وجود دارند.

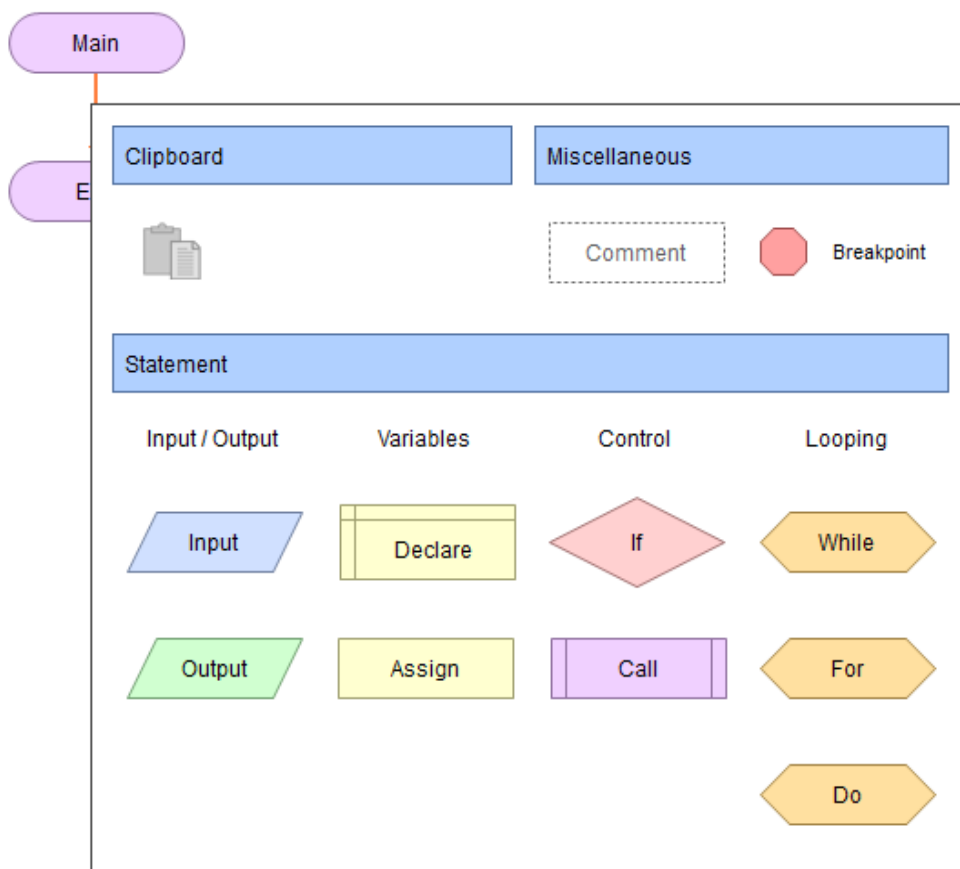
سایر توضیحات را در قالب مثال‌هایی ارائه می‌دهیم.

مثال ۱-۱: 

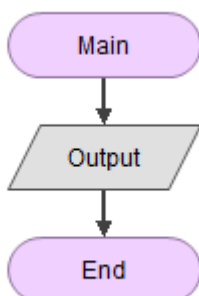
می‌خواهیم فلوجارتی رسم نماییم که عبارت "Hello World" را نمایش دهد.

روش کار: 

۱- برای اضافه کردن شکل جدید به فلوجارت، کافیست روی فلش موردنظر کلیک کنیم، با این کار پنجره‌ای باز می‌شود که همه اشکال هندسی موردنیاز ما در آن موجود است فقط باید روی شکل دلخواه کلیک کرده تا به فلوجارت اضافه شود.



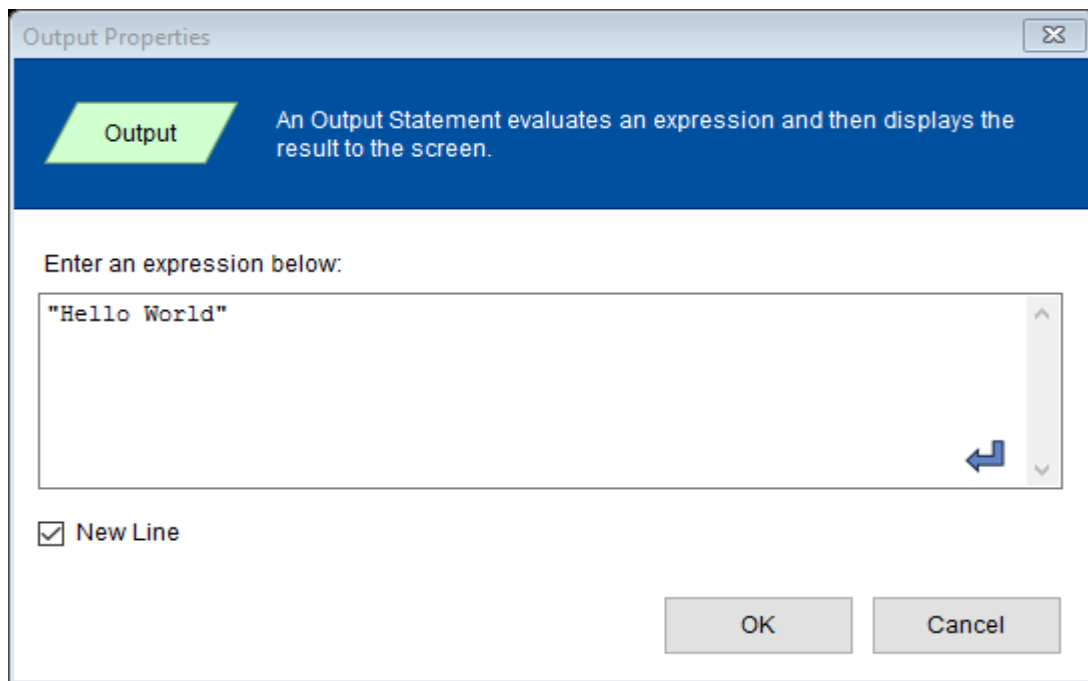
شکل ۱-۴: اضافه کردن شکل جدید به فلوجارت



۲- حال روی شکل مربوط به Output کلیک می‌کنیم تا در فلوجارت ما قرار بگیرد.

شکل ۱-۵: نمای اولیه از فلوجارت

۳- سپس روی شکل Output دابل کلیک می‌کنیم تا پنجره خصوصیات (Output properties) آن باز شود.



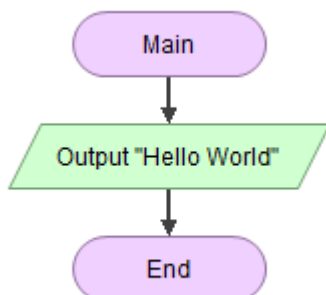
شکل ۱-۶: پنجره خصوصیات دستور خروجی

در این پنجره در قسمت Enter an expression below: متنی که عینا قرار است چاپ شود را داخل گیومه (دابل کوتیشن) قرار می‌دهیم.

اگر گزینه New Line فعال باشد، اشاره‌گر ماوس به ابتدای خط بعدی منتقل می‌شود.

در صورتی که بخواهیم متنی به همراه مقدار متغیری در خروجی نمایش داده شود، کافیست از عملگر & جهت اتصال آنها (Concat) استفاده نماییم.

در نهایت فلوچارت زیر را داریم.



شکل ۱-۷: فلوچارت کامل شده مثال ۱

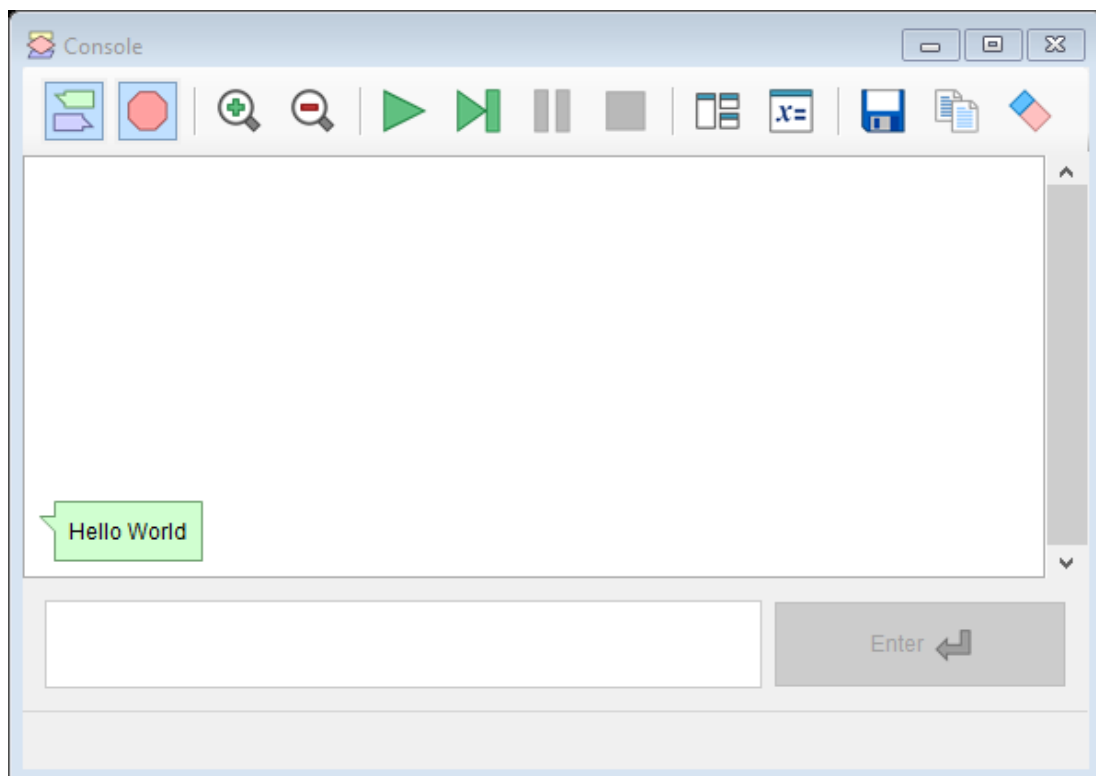


### ۱-۳-۱. اجرای فلوجارت

برای اجرای فلوجارت می‌توان از ۳ روش زیر استفاده کرد:

- منوی Program گزینه Run
- کلید F۵
- گزینه Run موجود در نوار ابزار 

در صورتیکه برنامه خطا نداشته باشد، با موفقیت اجرا می‌شود و پنجره خروجی (Console) باز و خروجی نمایش داده می‌شود.



شکل ۱-۸: پنجره خروجی مثالی

اگر بخواهیم روند اجرای برنامه را به صورت کندتر مشاهده نماییم، می‌توانیم از منوی Program و گزینه Run Speed ، سرعت اجرای برنامه را تنظیم نماییم.

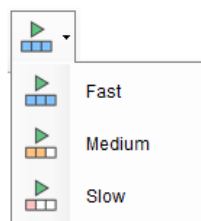
Run Speed روی ۳ حالت زیر تنظیم می‌شود:

❖ Fast (تند - سریع)

❖ Medium (متوسط)

❖ Slow (کند - آرام)


گزینه فوق در نوار ابزار برنامه نیز وجود دارد.



در صورتی که بخواهیم اجرای برنامه، مرحله به مرحله صورت پذیرد، می‌توان از روش‌های زیر استفاده کرد:

- منوی Program گزینه Step

- کلید F۶

-  گزینه Step موجود در نوار ابزار

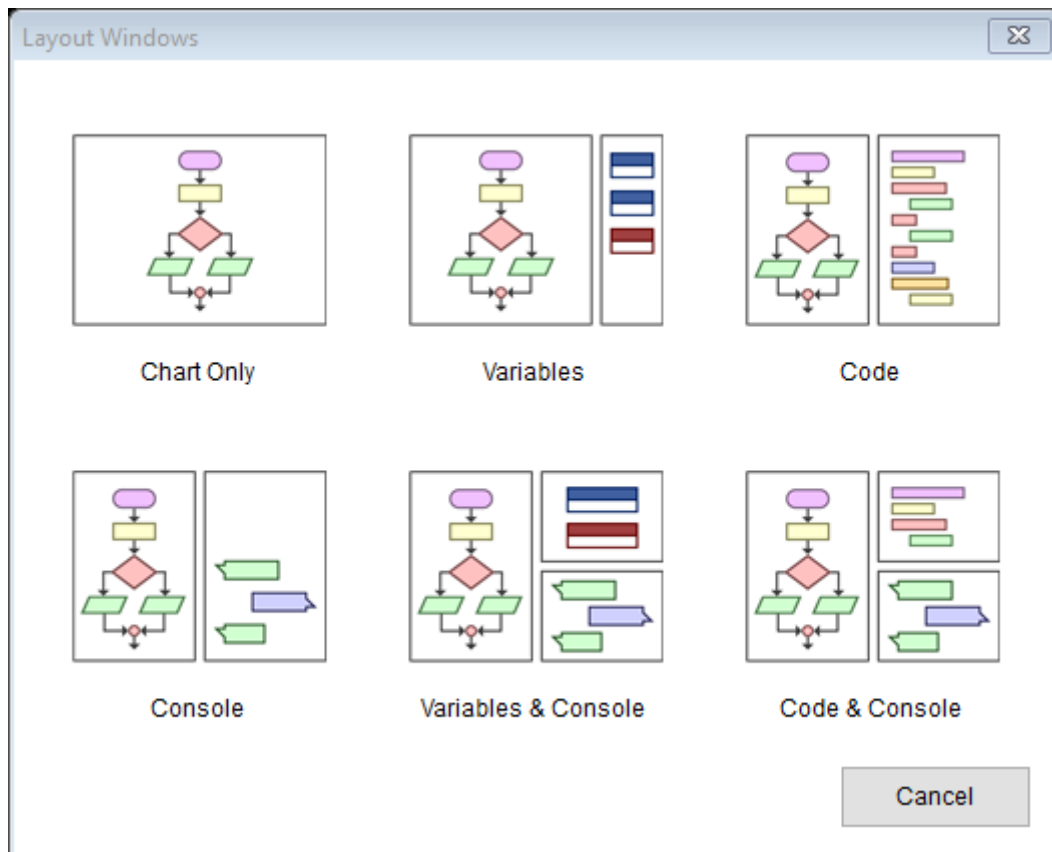
ما می‌توانیم به هنگام اجرای برنامه، به جز پنجره کنسول به پنجره متغیرها (Variables) و کد برنامه (Code) هم دسترسی داشته باشیم. برای مشاهده این پنجره‌ها می‌توان از ۳ روش زیر استفاده کرد:

- از منوی Tools گزینه Layout windows

-  گزینه Layout windows موجود در نوار ابزار

- کلیدهای میانبر Ctrl+L

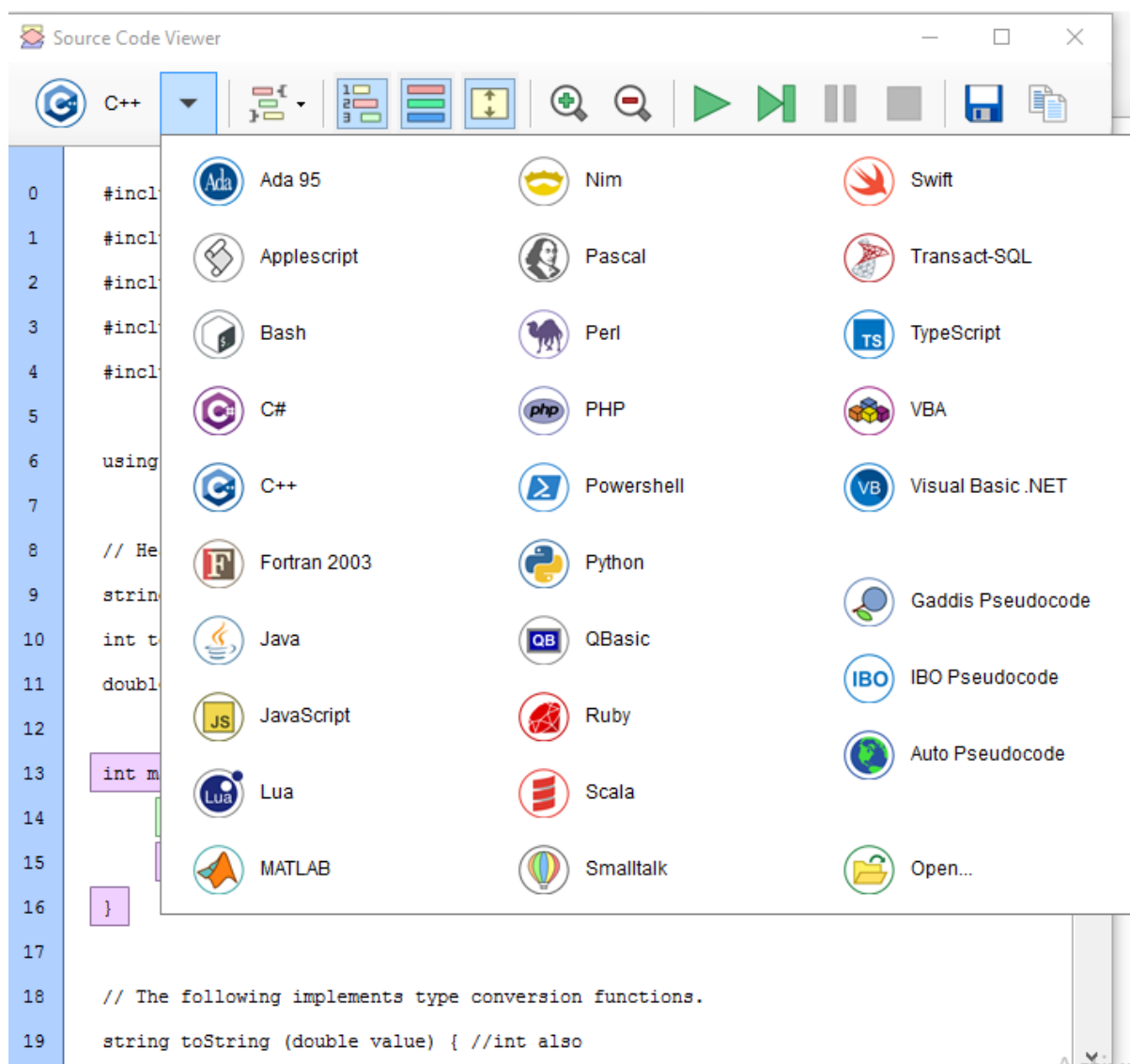
وقتی روی گزینه فوق کلیک می‌نماییم، پنجره زیر باز می‌شود. در این پنجره حالت‌های مختلف کنار هم قرار گرفتن پنجره‌های مربوط به فلوجارت، کنسول، متغیرها و کد برنامه به نمایش گذاشته شده است. که می‌توان هر یک را انتخاب نمود.




شکل ۱-۹: حالت‌های مختلف کنار هم قرار گرفتن پنجره‌های مربوط به فلوجارت، کنسول، متغیرها و کد برنامه

### ۱-۳-۲. تبدیل فلوچارت به سورس کد

برای مشاهده کد برنامه، ابتدا وارد پنجره Source Code Viewer می‌شویم (از روش‌های ذکر شده در صفحه قبل یا گزینه Source Code Viewer موجود در نوار ابزار). سپس با کلیک روی گزینه Language موجود در نوار ابزار این پنجره می‌توانیم زبان برنامه‌نویسی مورد نظرمان را انتخاب نماییم تا فلوچارت ما تبدیل به کد زبان مورد نظر شود. در ادامه می‌توان این برنامه را ذخیره کرده و در محیط‌های توسعه دیگر (IDE) مشاهده نمود و خروجی گرفت.



شکل ۱-۱۰: تبدیل فلوچارت به کد برنامه

مثال ۱-۲: 

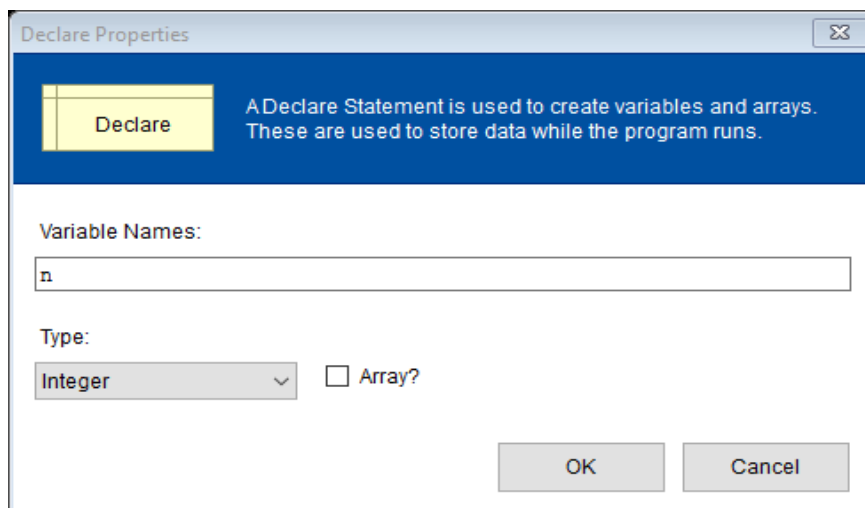
**عنوان:** فلوجارت برنامه‌ای را رسم نمایید که عددی را از کاربر دریافت نموده، تشخیص دهد عدد ورودی زوج است یا فرد.

**هدف:** استفاده از ساختار شرطی If

روش کار: 

مانند مثال قبل، اشکال مورد نظرمان را اضافه می‌کنیم و با دابل کلیک روی هر شکل به پنجره تنظیمات آن هدایت می‌شویم. در این فلوجارت پنجره تنظیمات If، Declare و Input را مورد بررسی قرار می‌دهیم.

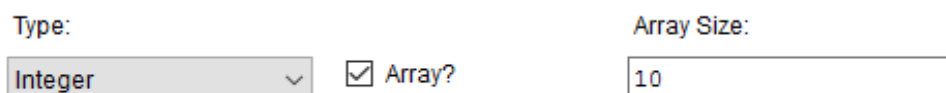
## پنجره تنظیمات Declare



شکل ۱-۱۱: پنجره تنظیمات Declare

در این پنجره نوع متغیرها را تعریف می‌کنیم. در کادر مربوط به Variable Names نام متغیرها را وارد می‌کنیم و از بخش Type نوع آنها را مشخص می‌کنیم.

اگر بخواهیم متغیرمان را به صورت آرایه تعریف کنیم، تیک گزینه Array را فعال می‌نماییم. با فعال کردن این گزینه، سائز آرایه نیز از ما پرسیده می‌شود.



## پنجره تنظیمات ورودی

شکل ۱-۱۲: پنجره تنظیمات ورودی

در کادر مربوط به: Enter a variable name below: نام متغیری که می‌خواهیم از ورودی دریافت کنیم را وارد می‌کنیم.

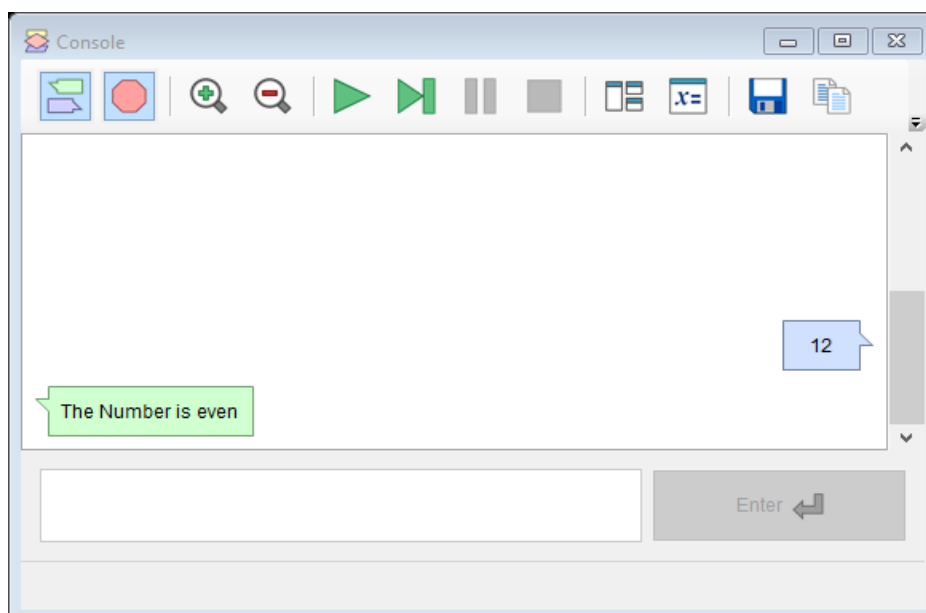
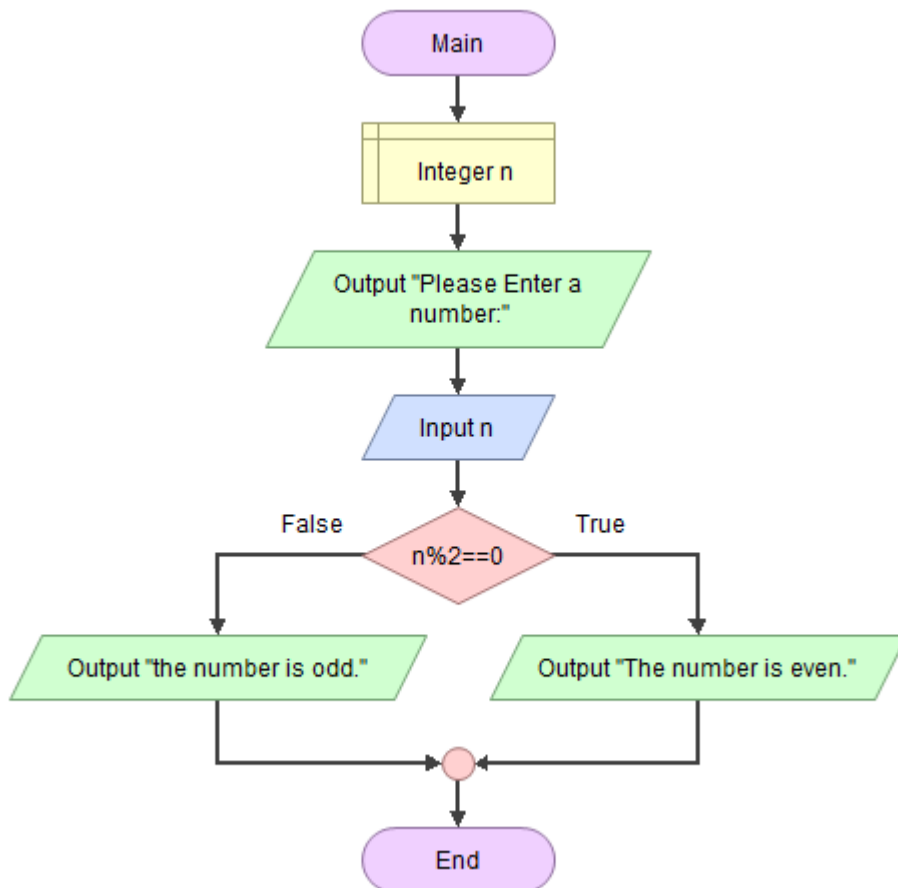
## پنجره تنظیمات If

شکل ۱-۱۳: پنجره تنظیمات If

در کادر مربوط به: Enter a conditional expression below: عبارت شرطی مورد نظر را وارد می‌نماییم.

لازم به ذکر است وقتی از شکل لوزی که مربوط به ساختار شرطی If می‌باشد، استفاده می‌کنیم، دو فلش از آن خارج می‌شود که یکی از آنها به معنی درست بودن (True) شرط است و دیگری به معنای نادرست بودن (False) شرط است.

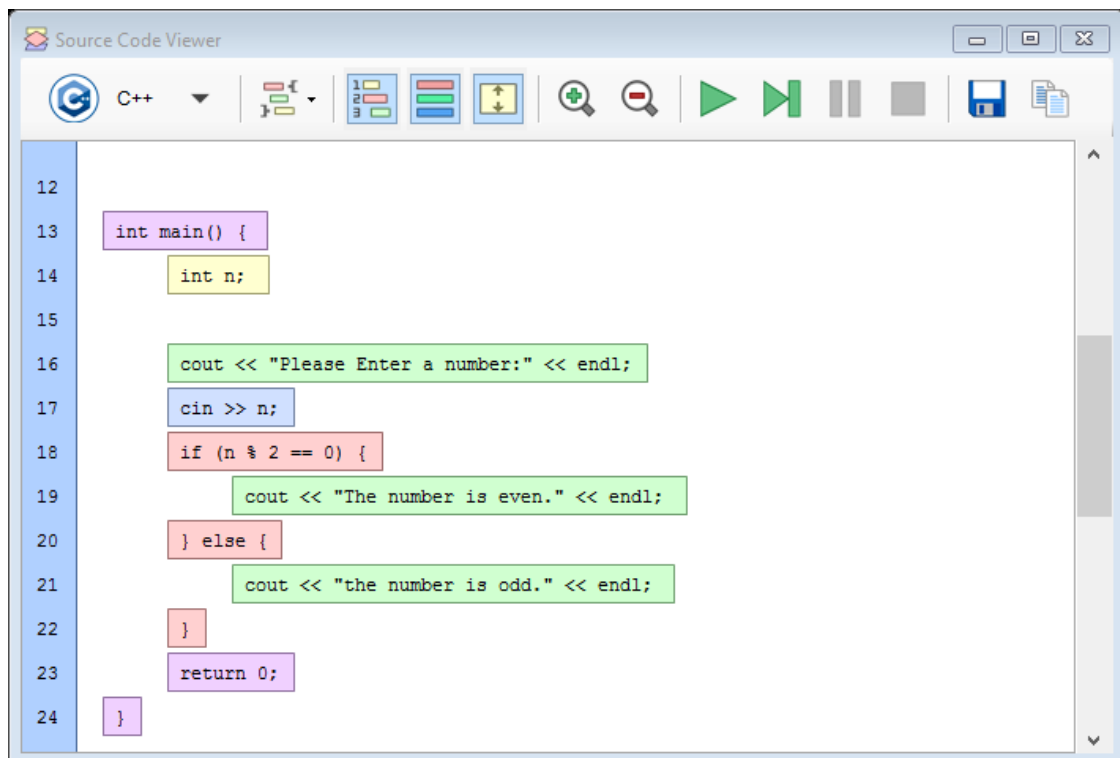
در نهایت فلوچارت و خروجی برنامه به شکل زیر درمی‌آید:



شکل ۱-۱۴: فلوچارت و خروجی مثال ۲

نمونه‌هایی از کدهای برنامه‌نویسی مثال ۱-۲:

### ▪ زبان C++

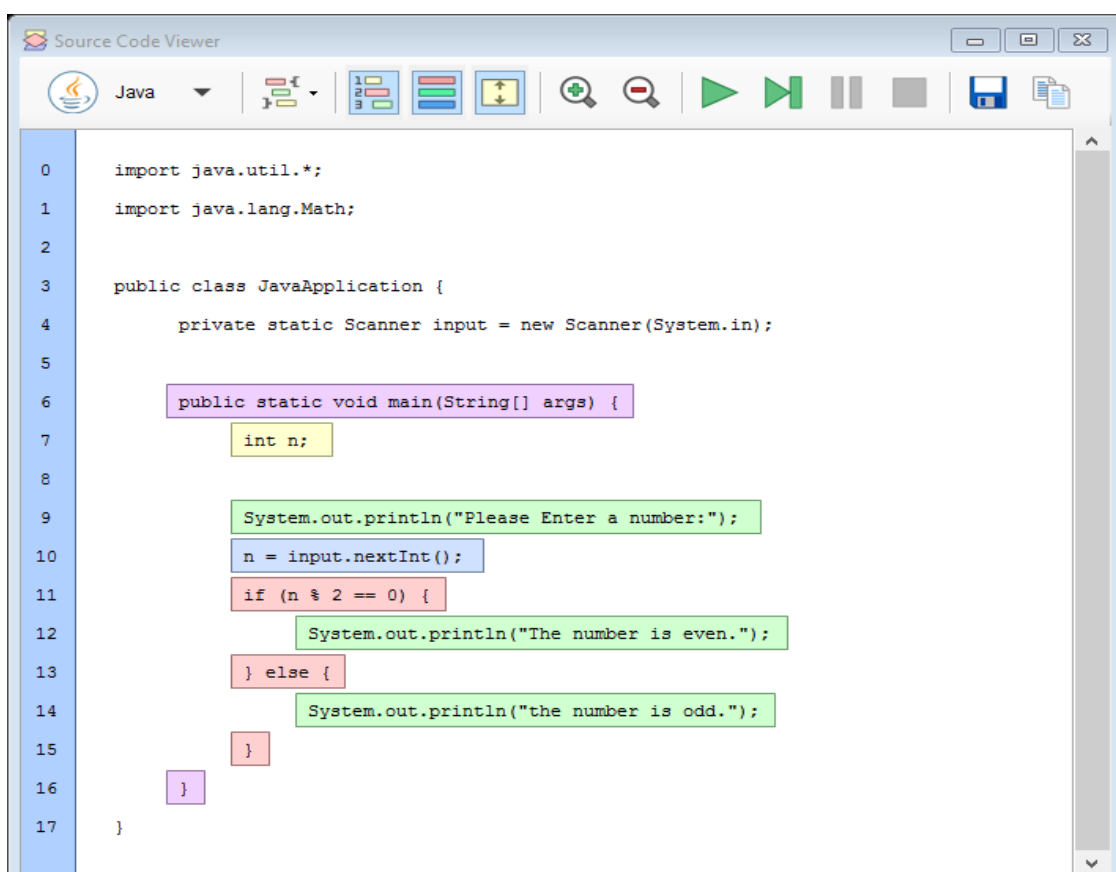


```

Source Code Viewer
C++
12
13 int main() {
14     int n;
15
16     cout << "Please Enter a number:" << endl;
17     cin >> n;
18     if (n % 2 == 0) {
19         cout << "The number is even." << endl;
20     } else {
21         cout << "the number is odd." << endl;
22     }
23     return 0;
24 }

```

### ▪ زبان جاوا



```

Source Code Viewer
Java
0 import java.util.*;
1 import java.lang.Math;
2
3 public class JavaApplication {
4     private static Scanner input = new Scanner(System.in);
5
6     public static void main(String[] args) {
7         int n;
8
9         System.out.println("Please Enter a number:");
10        n = input.nextInt();
11        if (n % 2 == 0) {
12            System.out.println("The number is even.");
13        } else {
14            System.out.println("the number is odd.");
15        }
16    }
17 }

```



تمرین ۱-۱ 

**عنوان:** فلوجارت برنامه‌ای را رسم نمایید که ۳ عدد از ورودی دریافت نموده، سپس ماکزیمم مقدار آنها را چاپ نماید

**هدف:** استفاده از ساختار شرطی تو در تو

مثال ۱-۳ 

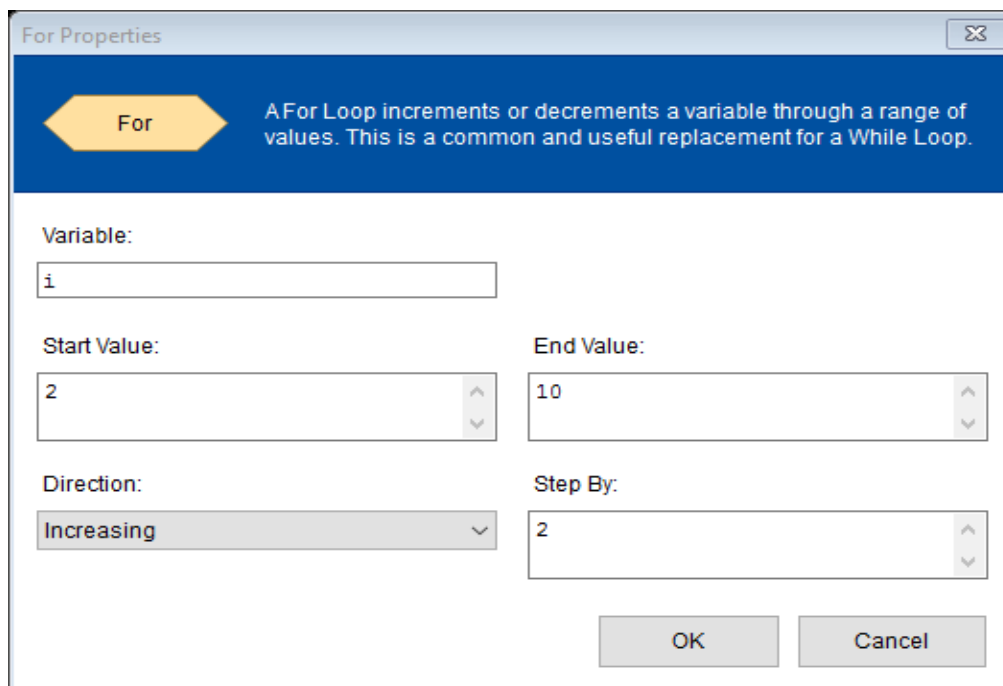
**عنوان:** فلوجارت برنامه‌ای را رسم نمایید که اعداد زوج بین ۱ تا ۱۰ را در خروجی چاپ نماید.

**هدف:** استفاده از حلقه‌ها

روش کار: 

مانند مثال‌های قبل عمل کرده، مضاف بر اینکه در این فلوجارت پنجره تنظیمات حلقه For را مورد بررسی قرار می‌دهیم.

پنجره تنظیمات  
حلقه For

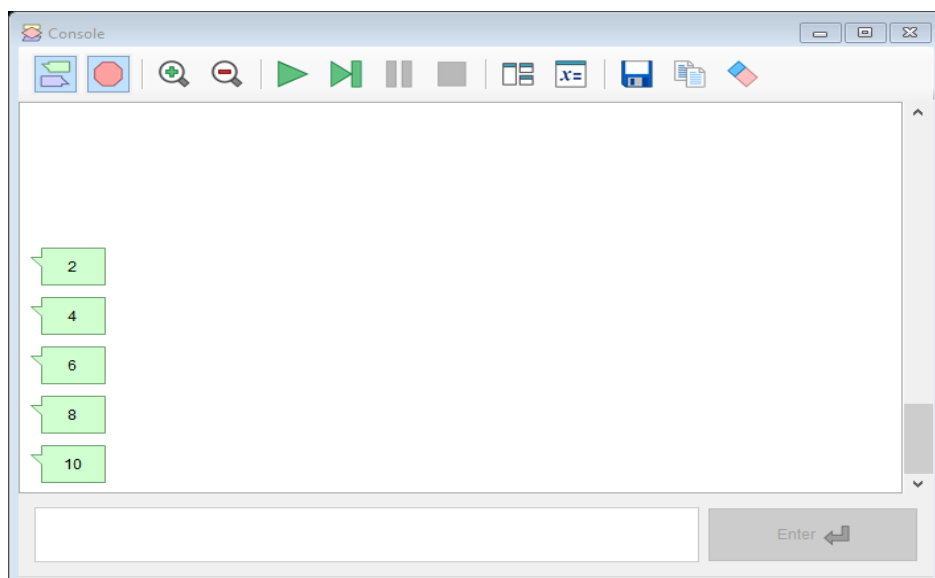
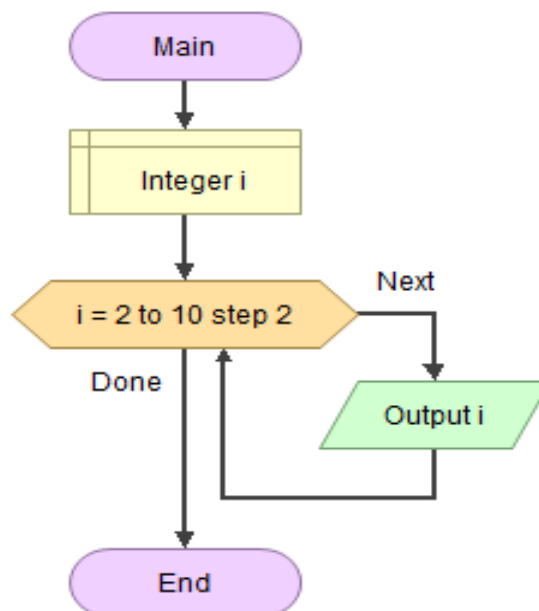


شکل ۱-۱۵: پنجره تنظیمات حلقه For


مواردی که باید در این پنجره مشخص شوند، عبارتند از:

- ❖ Variable: متغیری را به عنوان اندیس حلقه (شمارنده حلقه) انتخاب می‌کنیم. مثل  $i, j, k$
- ❖ Start Value: مقدار اولیه برای اندیس حلقه
- ❖ End Value: مقدار نهایی برای اندیس حلقه
- ❖ Step By: مقدار افزایشده یا کاهشده برای اندیس حلقه
- ❖ Direction: شمارنده حلقه افزایشی یا کاهششی باشد.

در نهایت فلوچارت برنامه به شکل زیر درمی‌آید:




شکل ۱-۱۶: فلوچارت و خروجی مثال ۳

تمرین ۱-۲ 

**عنوان:** مثال ۳ را این بار با حلقه For و ساختار شرطی If پیاده‌سازی نمایید. به گونه‌ای که شرط زوج بودن در دستور If بررسی شود.

**هدف:** استفاده از ساختار شرطی و حلقه در کنار هم

مثال ۱-۴ 

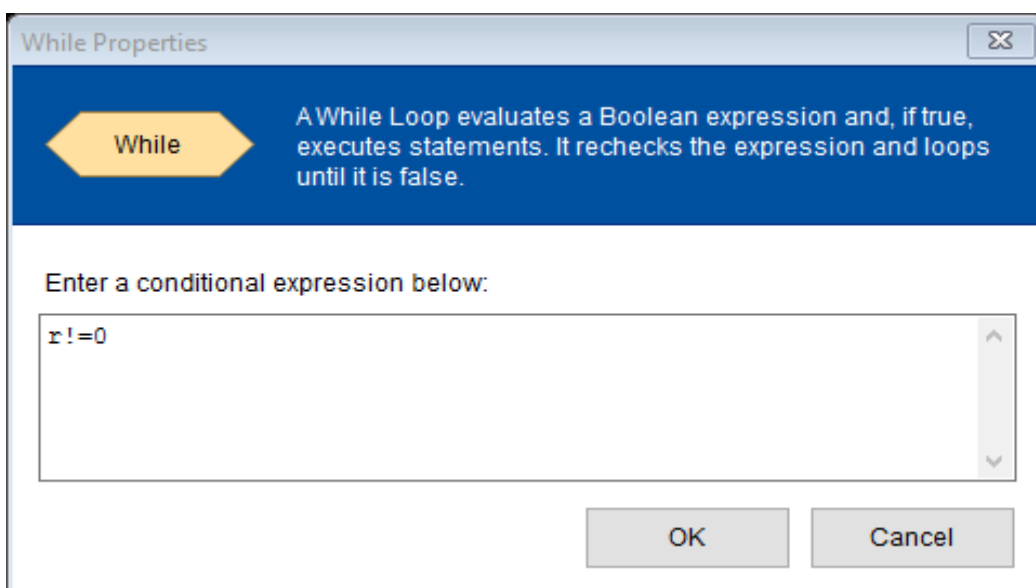
**عنوان:** فلوجارت برنامه‌ای را رسم نمایید که ۲ عدد از ورودی دریافت کرده سپس ب.م.م آنها را محاسبه و چاپ نماید.

**هدف:** استفاده از حلقه While

روش کار: 

مانند مثال‌های قبل عمل کرده، مضاف بر اینکه در این فلوجارت پنجره تنظیمات حلقه While و Assign را مورد بررسی قرار می‌دهیم.

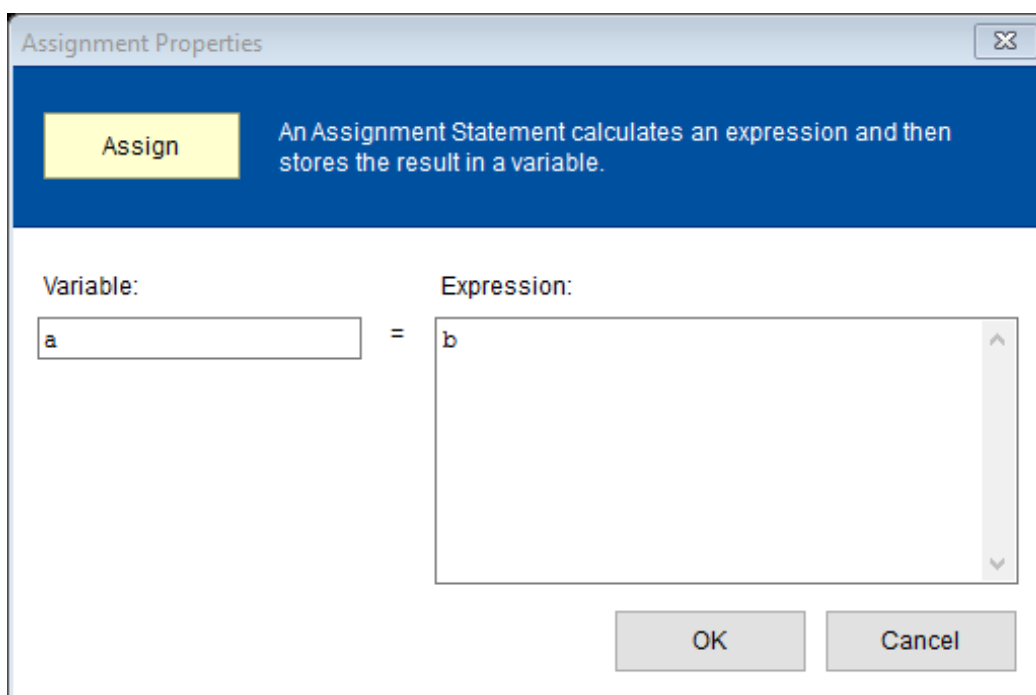
پنجره تنظیمات  
حلقه While



شکل ۱-۱۷: پنجره تنظیمات حلقه While

در کادر مربوط به: Enter a conditional expression below: عبارت شرطی مورد نظر را وارد می‌نماییم. حلقه While تا زمانی که شرط درست باشد ادامه می‌یابد و به محض نادرست شدن شرط، از حلقه خارج می‌شود و به سراغ اولین دستور بعد از حلقه می‌رود.

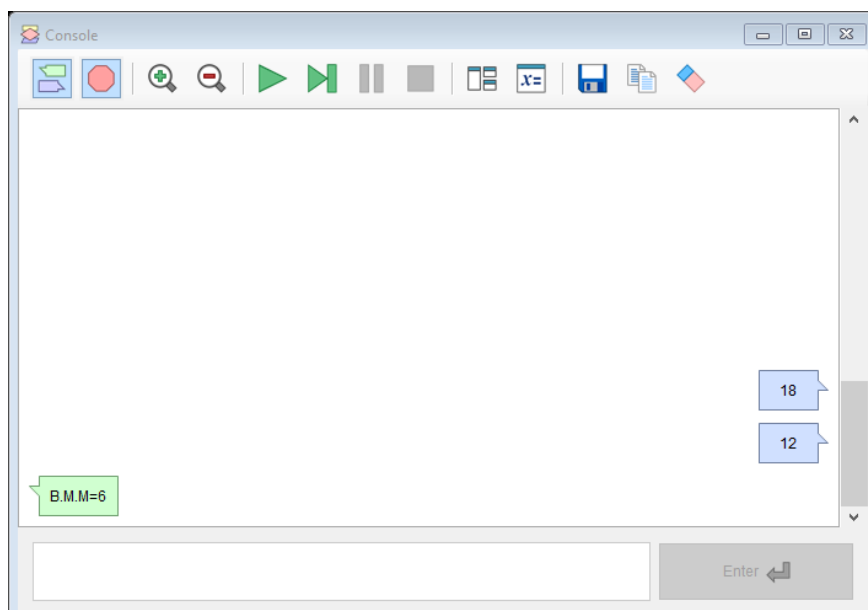
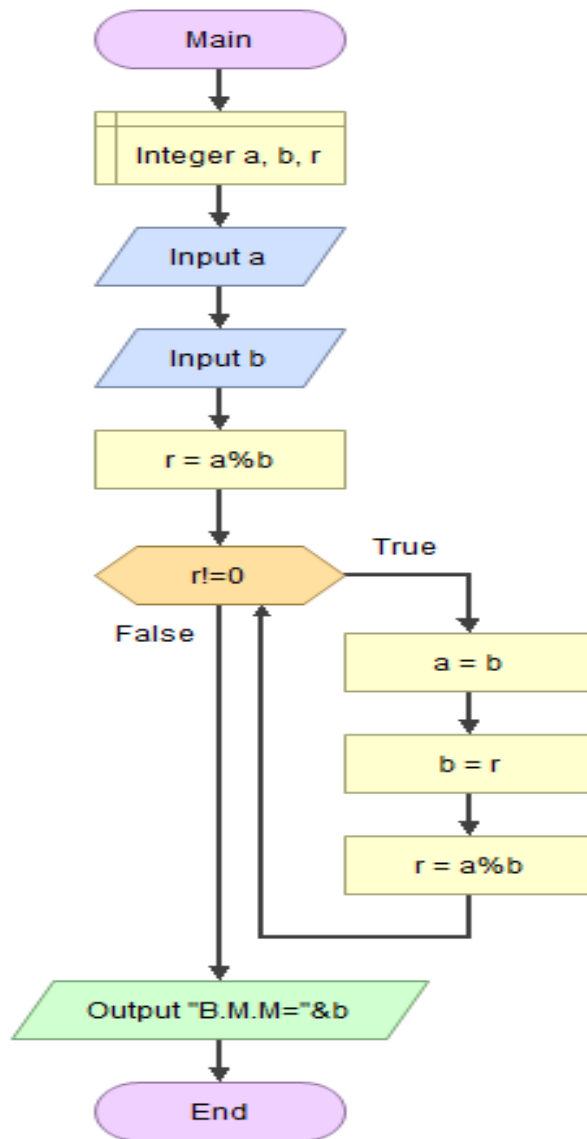
پنجره تنظیمات انتساب  
(Assign)



شکل ۱-۱۸: پنجره تنظیمات دستور انتساب

در این پنجره در کادر مربوط به: Variable: نام متغیری که می‌خواهیم مقدار یا عبارتی را به آن نسبت دهیم، وارد می‌کنیم و در کادر مربوط به: Expression: آن مقدار یا عبارت مورد نظر را وارد می‌کنیم.

در نهایت فلوچارت و خروجی برنامه به شکل زیر درمی‌آید:




شکل ۱-۱۹: فلوچارت و خروجی مثال ۴

تمرین ۱-۳ 

عنوان: مثال ۴ را این بار با حلقه do \_ While پیاده‌سازی نمایید.

هدف: استفاده از حلقه do \_ While

مثال ۱-۵ 

عنوان: فلوچارت برنامه‌ای را رسم نمایید که اعداد ۱ تا ۵ را به صورت زیر نمایش دهد.

```

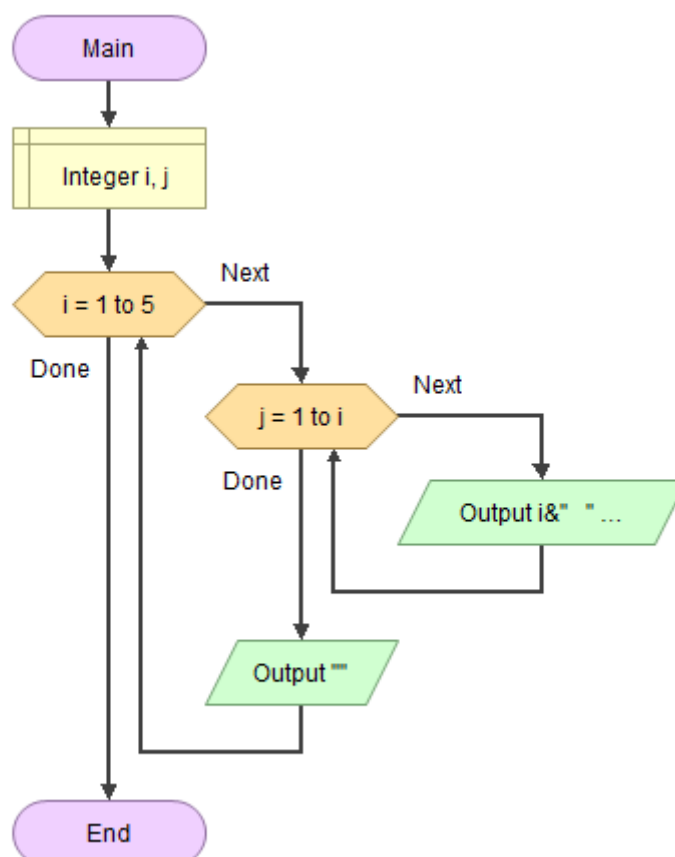
1
2 2
3 3 3
4 4 4 4
5 5 5 5 5
    
```

هدف: استفاده از حلقه‌های تو در تو

روش کار: 

با استفاده از دو حلقه For تو در تو، این کار را انجام می‌دهیم.

فلوچارت و خروجی برنامه به شکل زیر درمی‌آید:



```

1
2 2
3 3 3
4 4 4 4
5 5 5 5 5

```

شکل ۱-۲۰: فلوجارت و خروجی مثال ۵



### خودآزمایی :

در صورتی که در مثال ۵، به جای چاپ متغیر  $i$ ، متغیر  $j$  در دستور خروجی (output) قرار بگیرد، اعداد در خروجی به چه صورت چاپ خواهند شد.

## مثال ۱-۶:

**عنوان:** فلوچارت برنامه‌ای را رسم نمایید که با استفاده از یک تابع فرعی به نام pow،  $a^b$  را محاسبه نماید.

**هدف:** استفاده از توابع فرعی در فلوچارت

## روش کار:

ابتدا باید تابع مورد نظرم را تعریف کنیم. برای این کار می‌توان از یکی از روش‌های زیر استفاده کرد:



- منوی Program گزینه Add Function...
- گزینه Add Function... موجود در زیرمجموعه Main در نوار ابزار برنامه

سپس در پنجره باز شده، تنظیمات مربوط به تعریف تابع را انجام می‌دهیم.

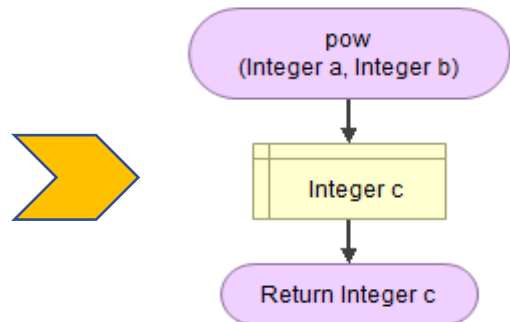
شکل ۱-۲۱: تنظیمات تابع



- ❖ **Function Name:** نامی برای تابع انتخاب می‌کنیم.
- ❖ **Parameters:** با استفاده از کلید Add ، اسامی پارامترهای موجود در تابع را وارد می‌کنیم.
- ❖ **Return Type:** نوع بازگشتی تابع را انتخاب می‌نماییم.
- ❖ **Return Variable:** نام متغیری که قرار است به تابع اصلی یا تابع فراخوانده شده، بازگشت داده شود را وارد می‌کنیم.

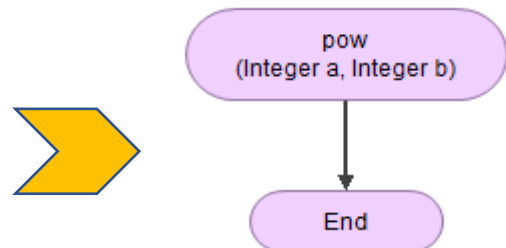
در این مثال، نام تابع را `pow` می‌گذاریم و دو پارامتر به نام‌های `a` و `b` از نوع صحیح تعریف می‌کنیم. حال اگر بخواهیم تابع `pow`، مقداری را به تابع اصلی بازگشت دهد باید نام متغیر و نوع آن را مشخص نماییم مثلاً متغیر `c` از نوع صحیح (روش اول) و در غیر این صورت نوع بازگشتی را روی `None` قرار می‌دهیم. (روش دوم)

The screenshot shows the 'Function Properties' dialog box. At the top, it says 'Function: A function allows programs to both reuse code and simplify logic. Data is passed into functions using parameters.' Below this, the 'Function Name' field contains 'pow'. The 'Parameters' list contains 'Integer a' and 'Integer b'. The 'Return Type' dropdown is set to 'Integer' and the 'Return Variable' field contains 'c'. There are 'Add', 'Edit', and 'Remove' buttons for parameters, and 'OK' and 'Cancel' buttons at the bottom.



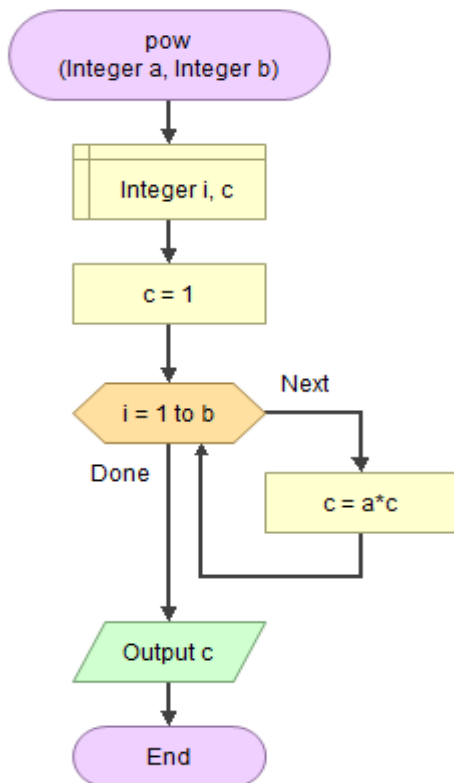
شکل ۱-۲۲: تنظیمات تابع در مثال ۶ و فلوجارت اولیه آن (روش اول)

The screenshot shows the 'Function Properties' dialog box. The 'Function Name' field contains 'pow'. The 'Parameters' list contains 'Integer a' and 'Integer b'. The 'Return Type' dropdown is set to 'None' and the 'Return Variable' field is empty. There are 'Add', 'Edit', and 'Remove' buttons for parameters, and 'OK' and 'Cancel' buttons at the bottom.

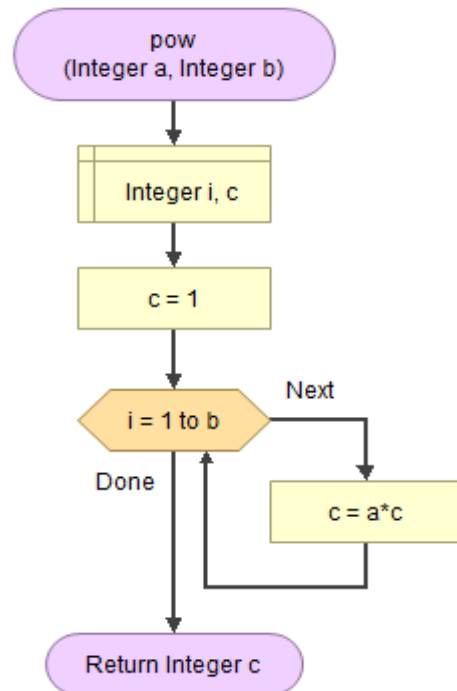


شکل ۱-۲۳: تنظیمات تابع در مثال ۶ و فلوجارت اولیه آن (روش دوم)

حال باید فلوچارت تعریف تابع را با توجه به آنچه که برنامه از ما خواسته است، کامل نماییم.

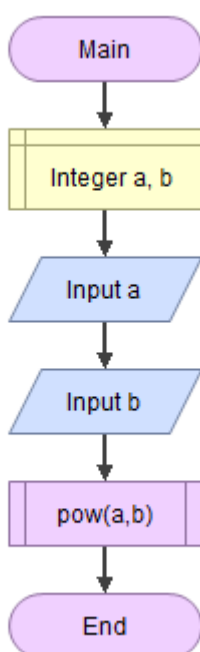


شکل ۱-۲۵: تعریف تابع به روش دوم

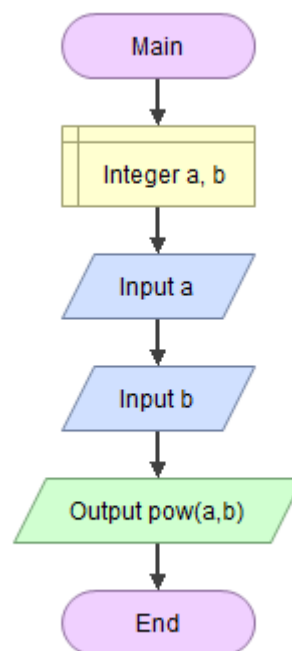


شکل ۱-۲۴: تعریف تابع به روش اول


در ادامه به سراغ تابع Main می‌رویم تا تابع pow را فراخوانی کنیم.



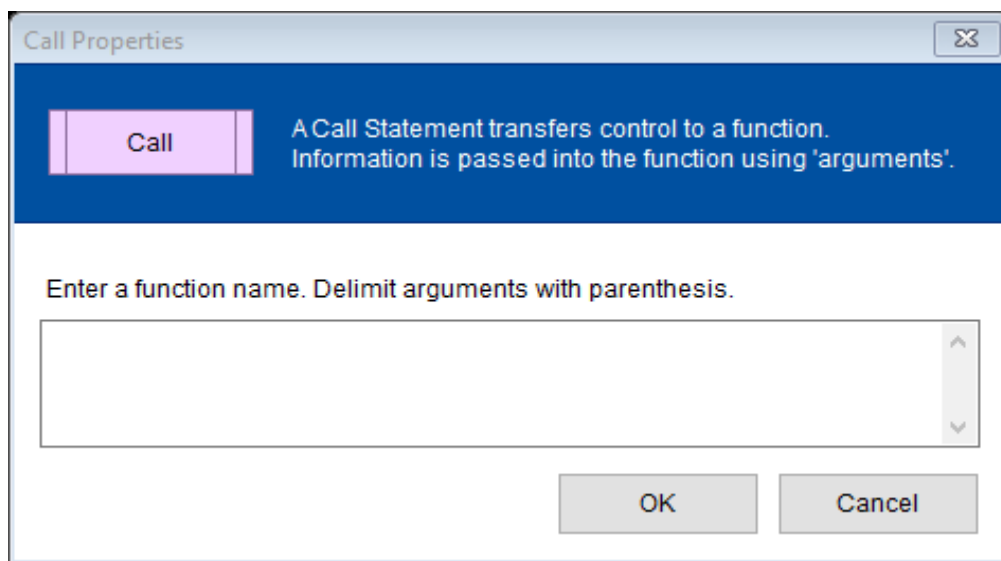
شکل ۱-۲۷: تابع اصلی در روش دوم



شکل ۱-۲۶: تابع اصلی در روش اول

نکته: همان طور که در فلوجارت‌های رسم شده در بالا مشاهده می‌کنید، اگر فراخوانی تابع به تنهایی صورت گیرد از شکل مربوط به فراخوانی تابع  استفاده می‌کنیم.

پنجره تنظیمات  
فراخوانی تابع (Call)



شکل ۱-۲۸: تنظیمات فراخوانی تابع

در این پنجره در کادر مربوط به Enter a function name. Delimit arguments with parenthesis. نام تابع به همراه آرگومان‌هایش را وارد می‌کنیم. (فراخوانی تابع)

تمرین ۱-۴ 

عنوان: فلوجارت برنامه‌ای را رسم نمایید که با استفاده از یک تابع فرعی با نام دلخواه، فاکتوریل عدد  $x$  را محاسبه نماید.

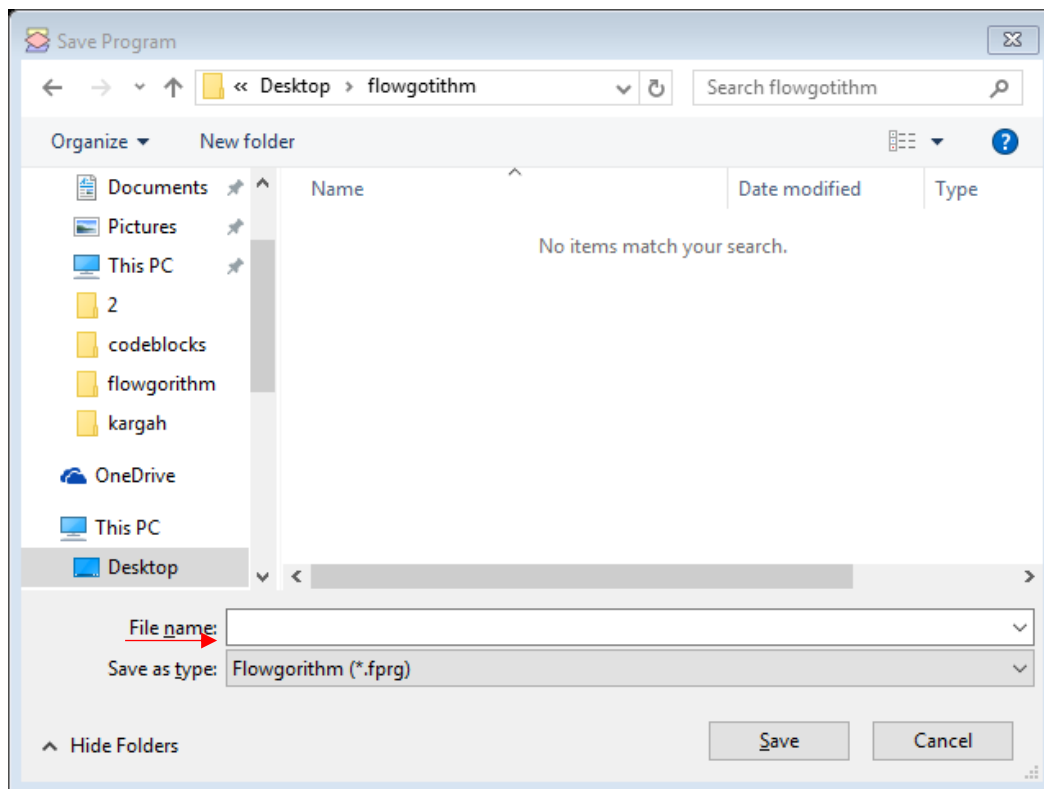
هدف: استفاده از توابع فرعی

## ۴-۱. ذخیره‌سازی و تبدیل فلوچارت به تصویر

یکی از قابلیت‌های خوب نرم‌افزار فلوگوریتم این است که می‌توان از فلوچارت‌ها، به صورت فایل PNG و یا SVG (برداری) خروجی گرفت.

برای این کار به منوی Tools رفته و از گزینه‌های Export to an Image File... برای تبدیل شدن فلوچارت به فرمت PNG و Export to a Vector Graphics File... برای فرمت SVG (برداری) استفاده می‌کنیم.

برای ذخیره‌سازی فلوچارت نیز از منوی File گزینه Save را انتخاب می‌کنیم. در پنجره باز شده در بخش File name: نامی دلخواه برای فایل مورد نظر وارد می‌کنیم و محل ذخیره‌سازی را نیز مشخص می‌کنیم.



شکل ۱-۲۹: پنجره ذخیره‌سازی

در صورتی که بخواهیم نام فایل یا محل ذخیره‌سازی را تغییر دهیم از گزینه Save as استفاده می‌کنیم.

نکته: پسوند فایل‌های فلوگوریتم \*.fprg می‌باشند.



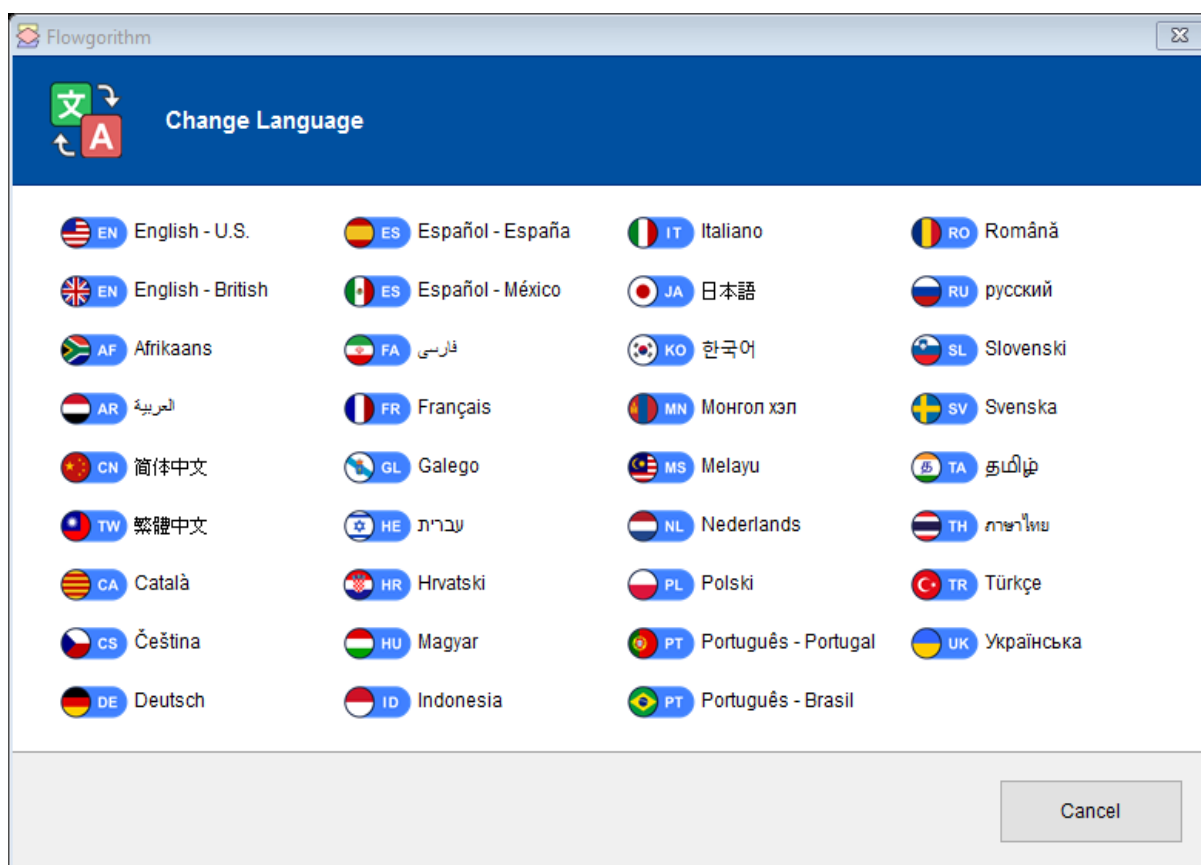
## ۱-۵. تغییر زبان برنامه

نرم‌افزار فلوگوریتیم Multi Language (چند زبانه) است و از زبان فارسی نیز پشتیبانی می‌کند.

برای تغییر زبان برنامه:

- از منوی Appearance گزینه Change Language...
- کلید F۱۲
- کلید Change Language... موجود در پایین صفحه

حال می‌توان در پنجره باز شده زبان مورد نظر را انتخاب نمود.



شکل ۱-۳۰: تغییر زبان برنامه

## تمرینات آخر فصل

۱. فلوچارتی رسم نمایید که شعاع دایره‌ای را از ورودی دریافت کرده، محیط و مساحت آن را محاسبه و چاپ نماید.
۲. فلوچارتی رسم نمایید که دو عدد از ورودی دریافت کرده سپس محتویات آنها را جابجا نماید.
۳. فلوچارتی رسم نمایید که دو عدد از ورودی دریافت کرده سپس بزرگترین عدد را در خروجی نمایش دهد.
۴. فلوچارتی رسم نمایید که دو عدد و یک عملگر را از ورودی دریافت کرده، کار یک ماشین حساب ساده را شبیه‌سازی نماید.
۵. فلوچارتی رسم نمایید که ضرایب یک معادله درجه دوم را از ورودی دریافت کرده، ریشه‌های آن را محاسبه و چاپ نماید.
۶. فلوچارتی رسم نمایید که عدد  $n$  را از ورودی دریافت کرده، مجموع اعداد از ۱ تا  $n$  را محاسبه کند.
۷. فلوچارتی رسم نمایید که ۱۰ عدد از ورودی دریافت کرده، مجموع و میانگین ۱۰ عدد را محاسبه و چاپ نماید.
۸. فلوچارتی رسم نمایید که عدد  $n$  و  $x$  (۲ عدد صحیح مثبت) را از ورودی دریافت کرده، سپس  $x$  به توان  $n$  را محاسبه نماید.
۹. فلوچارتی رسم نمایید که عدد  $n$  را از ورودی دریافت کرده، فاکتوریل آن را محاسبه نماید.
۱۰. فلوچارتی رسم نمایید که عددی را از ورودی دریافت کرده، سپس تعداد ارقام آن را شمرده و در خروجی چاپ نماید.
۱۱. فلوچارتی رسم نمایید که عددی را از ورودی دریافت کرده، اول بودن عدد را بررسی نماید.
۱۲. فلوچارتی رسم نمایید که عددی را از ورودی دریافت کرده، کامل بودن عدد را بررسی نماید.
۱۳. فلوچارتی رسم نمایید که یک آرایه حداکثر ۱۰۰ عنصری را از ورودی دریافت کرده، سپس با خواندن عنصری از ورودی، عنصر جدید را در آرایه جستجو نماید.

# فصل دوم

## برنامه‌نویسی به زبان C++ در محیط توسعه کد بلاکس

### هدف کلی

کسب مهارت در زمینه برنامه‌نویسی و خطایابی

### هدف‌های رفتاری

انتظار می‌رود پس از مطالعه این فصل بتوانید:

- با مفهوم کامپایلر آشنا شوید و انواع آنها را بشناسید.
- با مفهوم IDE آشنا شوید و انواع آنها را بشناسید.
- توانایی نصب نرم‌افزار کد بلاکس را داشته باشید.
- توانایی اجرای برنامه‌های C++ را داشته باشید.
- انواع خطاها را بشناسید و توانایی رفع آنها را نیز داشته باشید.
- با روش‌های یافتن خطاهای منطقی و برطرف کردن آنها آشنا باشید.
- مهارت خطایابی در کد بلاکس را داشته باشید.
- توانایی ساخت فایل DLL را داشته و با نحوه به‌کارگیری آن آشنا باشید.
- توانایی ساخت فایل کتابخانه‌ای یا هدر فایل را داشته باشید.
- با تفاوت‌های فایل کتابخانه‌ای و فایل DLL آشنا باشید.

## ۲-۱. مقدمه ای راجع به کامپایلر و IDE

### کامپایلر چیست؟

کامپایلر، برنامه یا مجموعه‌ای از برنامه‌های کامپیوتری است که متنی از زبان برنامه‌نویسی سطح بالا (زبان مبدا) را به زبانی سطح پایین (زبان مقصد)، مثل اسمبلی یا زبان ماشین، تبدیل می‌کند. خروجی این برنامه ممکن است برای پردازش شدن توسط برنامه دیگری مثل پیوند دهنده<sup>۷</sup> مناسب باشد یا فایل متنی باشد که انسان نیز بتواند آن را بخواند.

### IDE چیست؟

یک IDE یا به طور کامل محیط توسعه یکپارچه که مخففی از Integrated Development Environment میباشد. برنامه نرم‌افزاری است که برای کمک به برنامه‌نویسان و توسعه دهندگان جهت ساخت نرم افزار طراحی شده است. اکثر IDEها شامل یک ویرایشگر<sup>۸</sup> کد منبع، یک یا چند کامپایلر<sup>۹</sup> و یک اصلاح کننده خطا<sup>۱۰</sup> میباشد. البته لازم به ذکر است که IDE هایی نیز وجود دارند که فقط یک محیط ادیتوری یا ویرایشگری برای کدها دارند و بسیاری از آنها نیز هم یک محیط ویرایشی برای کدها دارند و هم قابلیت اجرا گرفتن و خطایابی از کدها را دارند.

### تفاوت بین کامپایلر و IDE چیست؟

کامپایلر وظیفه ی تبدیل کد های برنامه نویسی به زبان قابل فهم ماشین را برعهده دارد اما IDE یک نرم افزار کمکی برای راحت تر شدن برنامه نویسی است. بی شک زبان C و ++C جزء قدرتمندترین و مشهورترین زبان های برنامه نویسی جهان هستند و کامپایلر ها و IDE های بسیاری برای آن ها عرضه شده است. که تعداد محدودی از آن ها دارای محبوبیت و قدرت کافی هستند.

در زیر بهترین کامپایلر ها و IDE های جهان به اختصار توضیح داده شده اند.

<sup>۷</sup>.Linker

<sup>۸</sup>.Editor

<sup>۹</sup>.Compiler

<sup>۱۰</sup>.Debugger



## کامپایلر ها

**MinGw**: نرم افزار MinGw کامپایلر مخصوص مایکروسافت میباشد که فقط از ویندوز پشتیبانی میکند و برای C Runtime و برخی دیگر از زبان های Runtime می باشد .

**GCC**: نرم افزار GCC یک کامپایلر رایگان زیر نظر GNU میباشد که نه تنها کد های C++ را کامپایل میکند بلکه از زبان های Ada , Objective\_C , Java و ..... نیز پشتیبانی می کند .

**Tiny C Compiler**: نرم افزار TCC یکی از بهترین کامپایلر های C میباشد که از تمامی پیش پردازنده ها پشتیبانی میکند و در آن از اسمبلر GNU استفاده شده است . لازم به ذکر است که اسمبلر GNU یکی از بهترین اسمبلر های جهان است .

**Ideone**: نرم افزار Ideone یک IDE و کامپایلر آنلاین میباشد که از C++ - C و ۶۰ زبان دیگر پشتیبانی میکند .

چهار کامپایلر نام برده شده در بالا از بهترین کامپایلر های جهان هستند که کمترین اشکالات را در میان دیگر کامپایلر ها داشته اند.

## معرفی انواع IDE

**Visual studio**: از قابلیت های VS میتوان به برنامه نویسی برای موبایل , وب و دسکتاپ اشاره کرد و پشتیبانی از زبان های بسیاری هم چون , Asp.net , Basic , C# , C++ , C , Css , Python , JavaScript , Ruby , Xml و ..... و هم چنین قابلیت های بی شمار دیگر .

اما از بدی های آن میتوان پشتیبانی نکردن از دیگرسستم عامل ها و کامپایلر ها , حجم بسیار زیاد و قیمت سرسام آور آن اشاره کرد.

**Code blocks**: ادیتور C::B یک ادیتور مخصوص C++-C است که البته در نگارش جدید آن Fortran نیز اضافه شده است سرعت بالا , پشتیبانی از تمام سیستم عامل ها , حجم بسیار کم و همچنین رایگان و متن باز (Open Source) بودن آن , این IDE را در بین برنامه نویسان بسیار محبوب کرده است.

**Kdevelop**: ادیتور Kd یک ادیتور C++ - C رایگان متن باز و کم حجم برای سیستم عامل های خانواده ی لینوکس و Mac می باشد . این ادیتور از فریم ورک قدرتمند Qt نیز پشتیبانی می کند و البته نسخه های مختلفی از آن برای پشتیبانی از زبان های Php و Python نیز ارائه شده است . از بدی های این ادیتور میتوان پشتیبانی نکردن از سیستم عامل محبوب ویندوز نام برد.

**Anjuta Devstudio**: نرم افزار AD یکی دیگر از ادیتور های رایگان C++-C می باشد که دارای امکانات بسیاری نظیر مدیریت پروژه، طراح، GUI کنترل نسخه و ... است و دارای رابط کاربری خوب و حجم کم می باشد.

**Code lite**: ابزار CL نیز یکی دیگر از ادیتورهای محبوب در بین برنامه‌نویسان زبان‌های C++ می باشد که متن باز و رایگان است. همچنین دارای حجم کم و پشتیبانی خوبی از سیستم عامل های مختلف و محیطی آسان و راحت می باشد.

**Dev C++**: ابزار Dev یک ادیتور رایگان با ظاهری قدیمی است که فقط از دو کامپایلر MinGw و GCC پشتیبانی می کند. البته دارای امکانات خوبی می باشد اما پشتیبانی نکردن از تمامی نسخه های ویندوز و لینوکس و همچنین پشتیبانی نکردن از Mac و نمایش کدها بصورت تک رنگ آن را ادیتوری ضعیف جلوه می دهد. اما لازم به ذکر است که برنامه‌نویسان زیادی از این ادیتور در سرتاسر جهان استفاده می کنند.

## ۲-۲. راهنمای نصب کامپایلر و اجرای برنامه های C , C++

در این بخش، به روش نصب کامپایلرهای C++ روی کامپیوترهای شخصی، گوشی های هوشمند و تبلت می پردازیم.

### ۲-۲-۱. کامپایلر روی کامپیوترهای شخصی

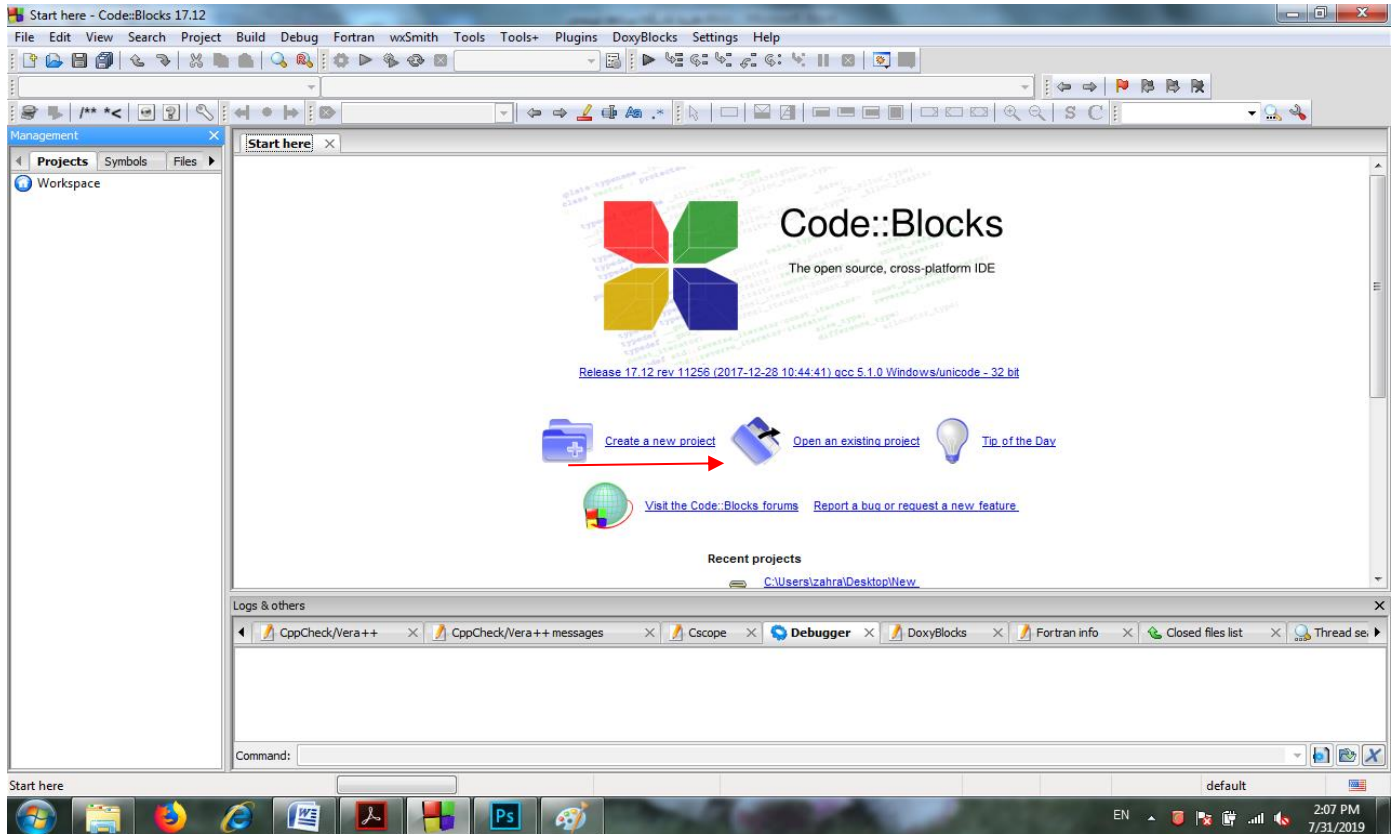
برای نصب روی کامپیوتر شخصی می توان از هر یک از نرم افزارهای نام برده شده (IDE) در قسمت قبل استفاده نمود. در این جزوه آموزشی، نرم افزار مورد استفاده، نرم افزار کدبلاکس می باشد.

### ۲-۲-۲. کامپایلر روی گوشی های هوشمند و یا تبلت

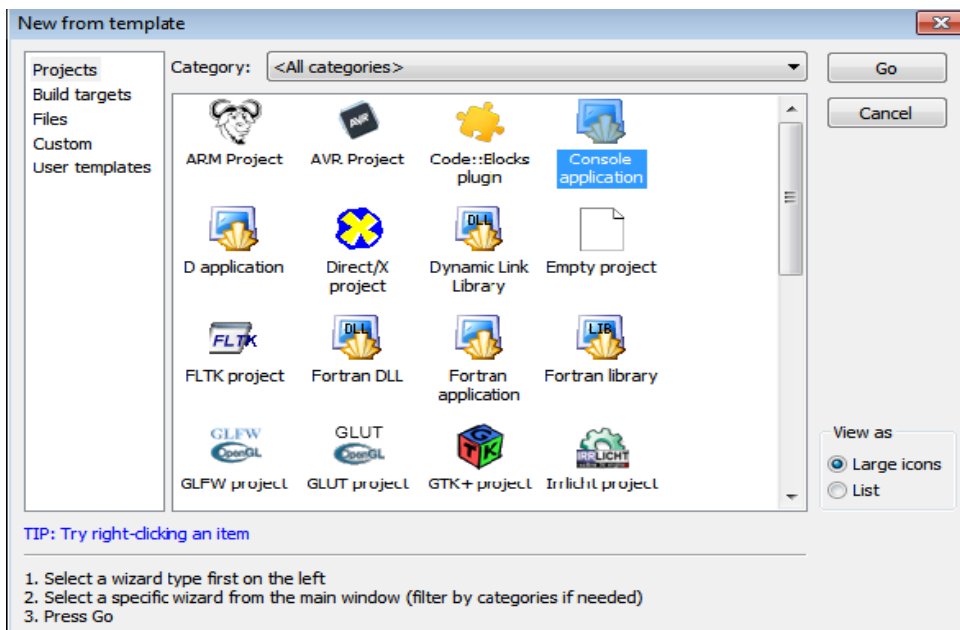
اپلیکیشن های زیادی در حوزه برنامه نویسی وجود دارند و روز به روز به تعداد آنها افزوده می شود . برای نصب نرم افزار روی گوشی های هوشمند خود می توانید به play store و یا بازار مراجعه نمایید سپس اپلیکیشن هایی نظیر Dcoder، Code Editor یا Cxxdroid را جستجو نموده و آن را نصب نمایید.

## ۲-۳. نحوه ایجاد کردن پروژه جدید در نرم‌افزار کد بلاکس و اجرای آن

۱- نرم‌افزار کد بلاکس را اجرا نموده، روی گزینه Create a new project کلیک نمایید.



شکل ۲-۱: صفحه آغازین نرم‌افزار کد بلاکس



۲- در پنجره باز شده انواع گوناگونی از قالب‌های پروژه، که هر یک کارایی و عملکرد خاص خود را دارند، مشاهده می‌کنید.

شکل ۲-۲: انتخاب قالب برنامه‌نویسی

مثلاً:

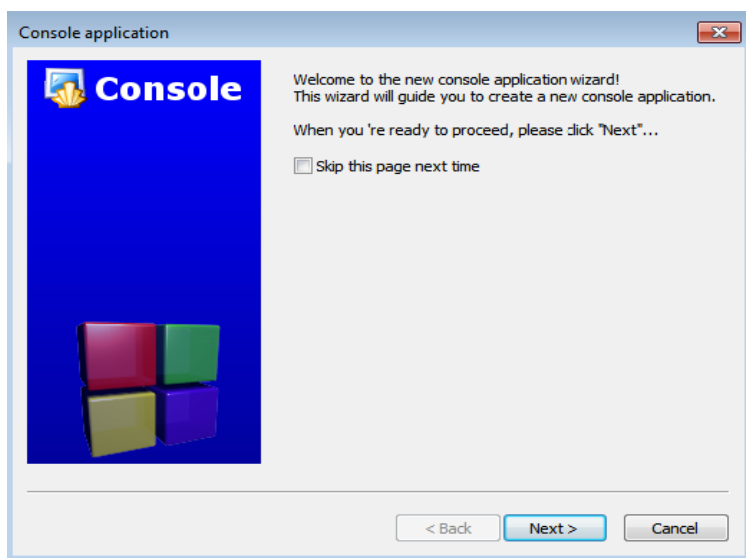
- AVR Project یا ARM Project، جهت برنامه نویسی میکروکنترلرها به کار می‌روند.
- Empty Project صرفاً یک پروژه خالی ایجاد می‌کند.
- Dynamic Link Library برای ایجاد کردن فایل‌های DLL (در بخش‌های بعدی توضیح داده خواهد شد.) به کار می‌رود.
- Consol Application برای برنامه نویسی کنسولی به کار می‌رود.

برنامه کنسولی یک نوع از برنامه‌های کامپیوتری است که در ساده‌ترین حالت گرافیکی موجود در یک سیستم عامل اجرا می‌شود. در همه سیستم‌عامل‌ها یک محیط تقریباً بدون گرافیک وجود دارد که مبتنی بر متن می‌باشد. گاهی این محیط را Terminal و یا Consol نیز می‌نامند. سیستم‌عامل داس نمونه‌ای از یک محیط کنسولی است. محیط‌های کنسولی GUI (رابط کاربر گرافیکی) ندارند و از طریق Command Line اجرا می‌شوند.

از آنجاییکه محیط کنسول به دور از پیچیدگی‌های گرافیکی است، یادگیری برنامه نویسی در این محیط، راحت‌تر است.

حال با توجه به توضیحات بالا، بر روی Consol Application کلیک می‌کنیم.

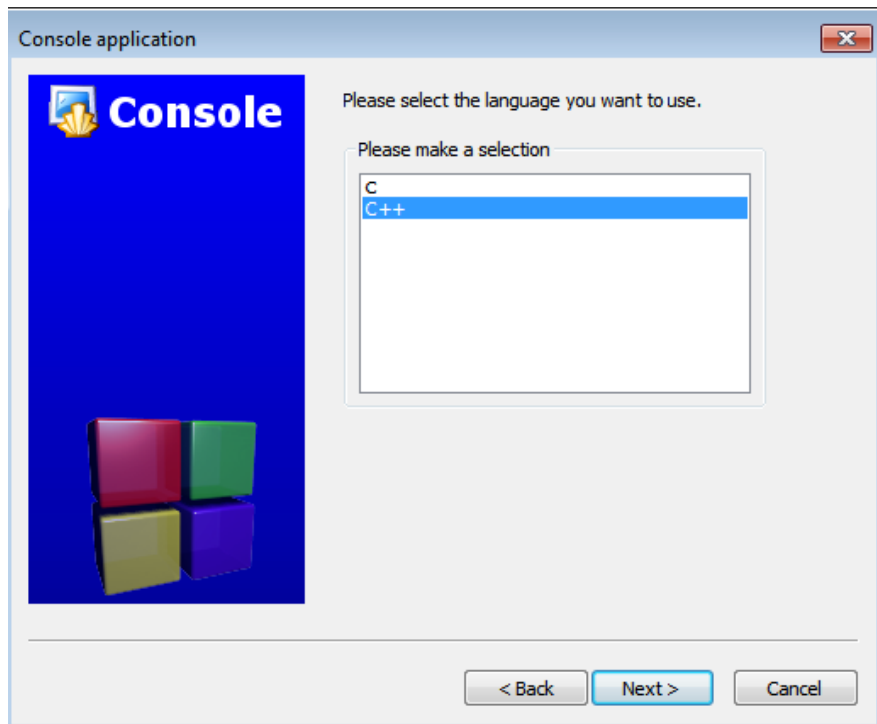
پیشنهاد می‌شود دانشجو دربارہ سایر قالب‌ها نیز تحقیق کرده و نتایج خود را به کلاس ارائه دهد.



۳- در این پنجره روی Next کلیک کنید.

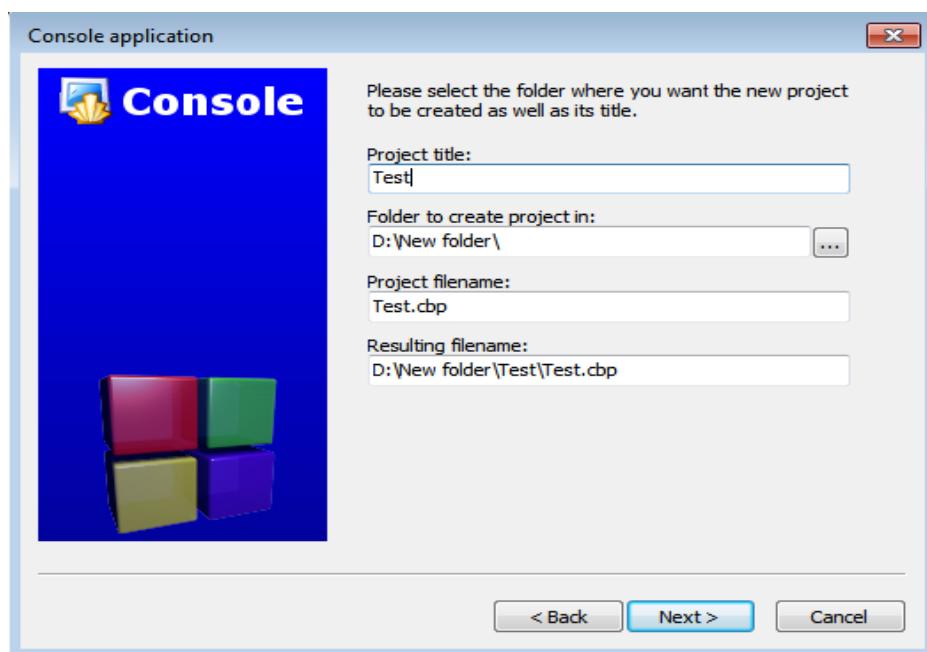
شکل ۲-۳

۴- در پنجره بعدی زبان انتخابی را روی C++ می‌گذاریم و روی Next کلیک می‌کنیم.




شکل ۲-۴: انتخاب زبان برنامه‌نویسی مورد نظر

۵- در پنجره بعدی در قسمت Project title : نامی برای عنوان پروژه خود وارد نمایید. و در قسمت Folder to create project in : می‌توانید محل ذخیره سازی پروژه خود را مشخص نمایید. و در نهایت روی Next کلیک کنید.



شکل ۲-۵: وارد کردن نام و محل ذخیره‌سازی پروژه

نکته: پسوند فایل های کد بلاکس .cbp می باشند. 

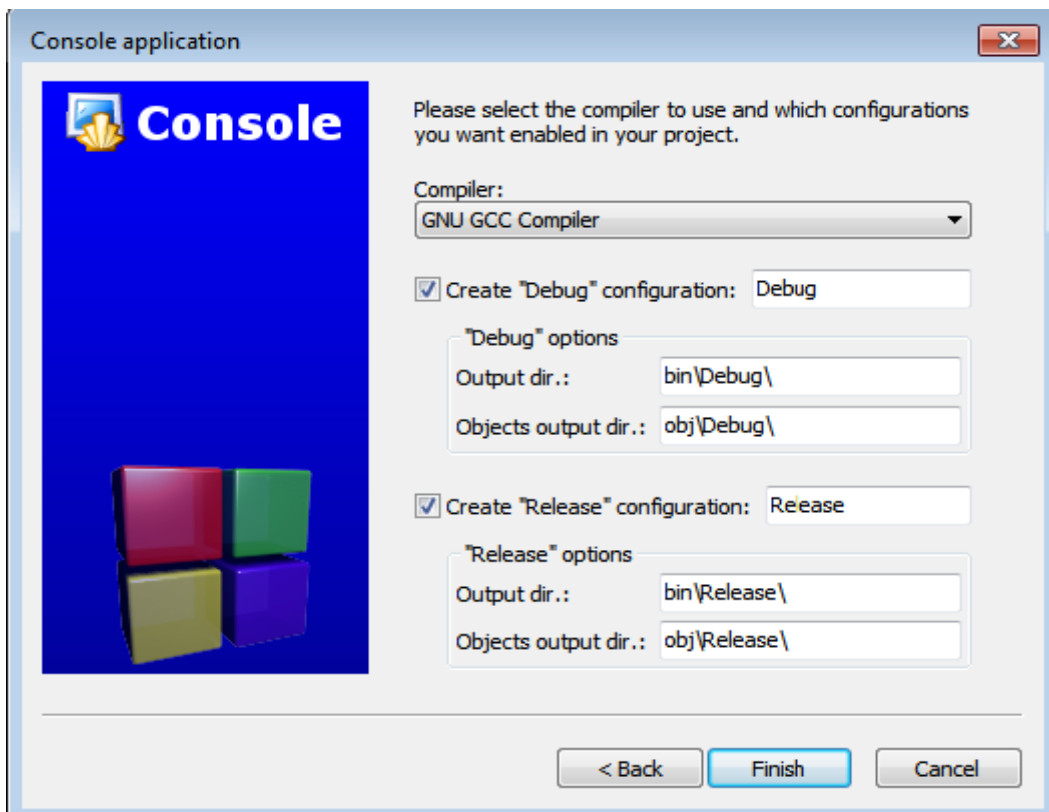
### نکته‌ای راجع به عنوان پروژه:

نامی که برای عنوان پروژه خود انتخاب می کنید باید:

- حتما انگلیسی باشد.
  - در مسیر ذخیره سازی، فقط باید از نام های انگلیسی استفاده شده باشد.
- در صورتی که هر یک از موارد بالا رعایت نشود برای رفتن به مراحل بعدی، خطایی داده نمی شود اما زمانی که می خواهید پروژه تان را کامپایل کنید به شما پیغام خطا می دهد.

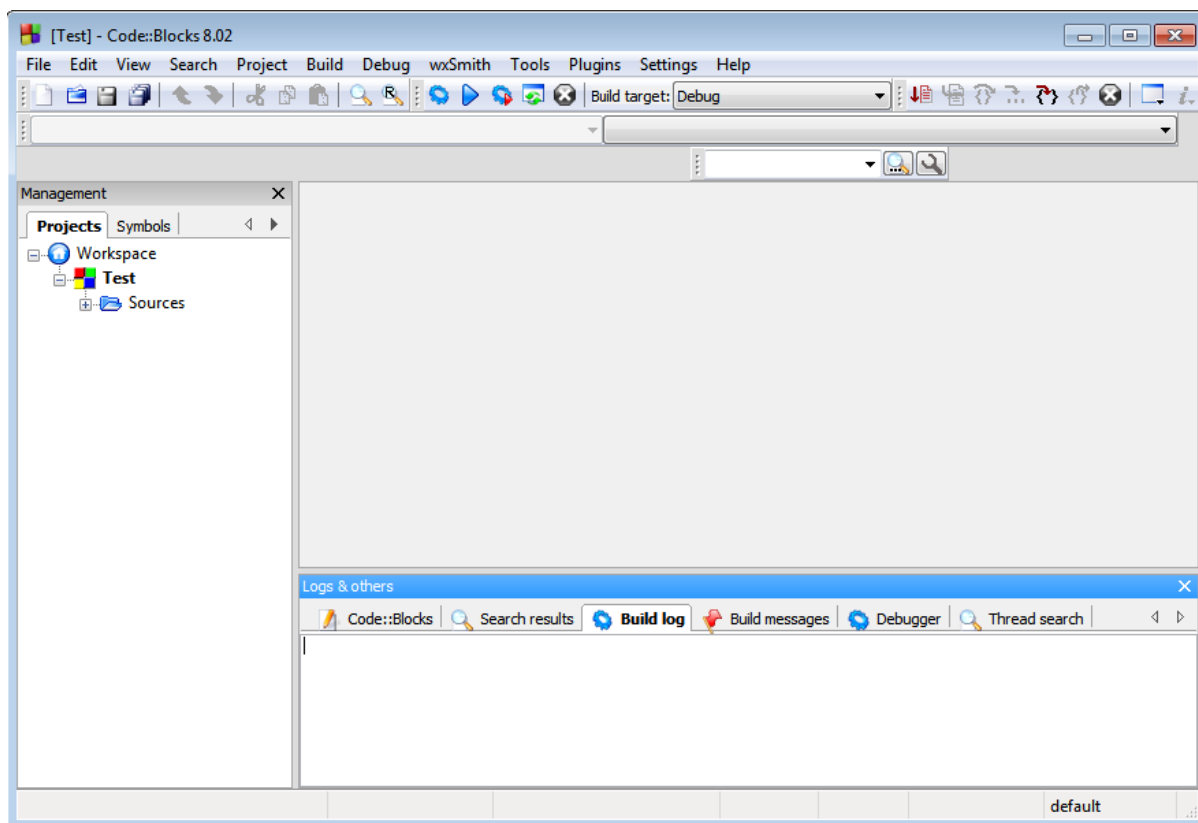
۶- در این پنجره پیش فرض ها را تغییر نمی دهیم و فقط روی finish کلیک می کنیم.

نکته: کامپایلر انتخابی، GNU GCC Compiler می باشد.



شکل ۲-۶: پنجره تنظیمات کامپایلر

حال وارد محیط اصلی نرم افزار شده اید و می توانید شروع به برنامه نویسی کنید.



شکل ۲-۷: محیط برنامه‌نویسی نرم‌افزار

در سمت چپ محیط این نرم افزار، پنل Management را مشاهده می کنید. در این پنل نامی که برای پروژه خود انتخاب کرده بودید را مشاهده می کنید که زیرمجموعه آن پوشه Sources قرار دارد و داخل آن پوشه فایل main.cpp وجود دارد. این فایل در زمان ایجاد پروژه به طور خودکار ساخته می شود. با کلیک کردن روی این فایل و باز کردن آن، محیط برنامه نویسی برای شما در سمت راست باز می شود.

برنامه Hello world به عنوان نمونه برای شما به نمایش درمی آید. حال شما می توانید برنامه مورد نظرتان را در این قسمت تایپ نموده و آن را اجرا نمایید.

نکته: در صورتی که پنل Management وجود نداشت، می توانید آن را از منوی View فعال نمایید.



```

1  #include <iostream>
2
3  using namespace std;
4
5  int main()
6  {
7      cout << "Hello world!" << endl;
8      return 0;
9  }
10

```

شکل ۲-۸: نمایش برنامه Hello world



روش ترجمه و اجرای برنامه:

وقتی برنامه ای نوشته می شود ابتدا باید آن را کامپایل یا ترجمه (ترجمه برنامه به زبان ماشین) نمایید این کار به عهده کامپایلر یا مترجم است.

برای این کار می توانید از سه روش استفاده کنید:

- از منوی Build اولین گزینه را اجرا کنید.
- کلید ترکیبی  $Ctrl+F9$
- از آیکن موجود در بالای صفحه (Build) که به شکل یک چرخ دنده می باشد.

پس از کامپایل کردن برنامه در صورتی که برنامه خطا نداشته باشد نوبت به Run کردن برنامه است.

برای این کار می توانید از سه روش استفاده کنید:

- از منوی Build گزینه run را اجرا کنید.
- کلید ترکیبی  $Ctrl+F10$
- از آیکن موجود در بالای صفحه (Run) که به شکل یک مثلث سبز رنگ می باشد.

البته می‌توان کار کامپایل و اجرا را توامان با هم نیز انجام داد. برای این کار هم می‌توانید از سه روش زیر استفاده کنید:

- از منوی Build گزینه Build and run را اجرا کنید.
- کلید F۹
- از آیکن موجود در بالای صفحه (Build and Run) استفاده نمایید. (چرخ دنده و مثلث سبز رنگ)

با اجرای برنامه، خروجی در صفحه ای مشکی رنگ به نمایش در می‌آید.

```

D:\New folder\Test\bin\Debug\Test.exe
Hello world!
Process returned 0 (0x0) execution time : 0.016 s
Press any key to continue.
  
```

شکل ۲-۹: نمایش صفحه خروجی

در صورتی که برنامه خطا داشته باشد، ابتدای خط یا خطوطی از برنامه که دارای اشکال هستند با مربع قرمز رنگ به نمایش در می‌آیند البته ممکن است خطای مورد نظر واقعا مربوط به آن خط نباشد. مثل خطای عدم گذاشتن سمی کولون در انتهای یک خط.

در قسمت پایین پنجره و در تب Build message خطاها نمایش داده می‌شوند.

نکته: در صورتی که به هر دلیلی پنل Logs در دسترس نبود، می‌توانید آن را از منوی View فعال نمایید.

### نکته ای مهم راجع به کد اجرایی برنامه :

وقتی به سراغ برنامه ذخیره شده می رویم و می خواهیم فایل اجرایی (exe) آن را باز کنیم بعد از باز کردن، فایل اجرایی سریع بسته می شود.

دلیل این اتفاق این است که برنامه پس از رسیدن به خط آخر خود به خود بسته می شود و پنجره آن هم همینطور. اگر می خواهید صفحه مربوط به خروجی تان را ببینید و باز نگه دارید، می توانید به آخر برنامه تان (قبل از ; Return ۰) ، یکی از دستورات ورودی را اضافه نمایید.

Scanf() – getch() – getche() – getchar() – system(“pause”) –

```

1  #include <iostream>
2  #include <stdio.h>
3
4  using namespace std;
5
6  int main()
7  {
8      cout << "Hello world!" << endl;
9
10
11     getchar();
12
13
14
15     return 0;
16 }
17

```

شکل ۲-۱۰

## ۲-۴. خطایابی و انواع خطاها

خطاها به ۳ دسته کلی تقسیم می‌شوند:

۱- **خطاهای زمان کامپایل Compile Error:** خطاهایی که در نوشتن دستورات برنامه نویسی ممکن است رخ دهند. به طور کلی رعایت نکردن قواعد زبان برنامه نویسی، سبب بروز چنین خطاهایی می‌شوند. مثل فراموش کردن سمی کولون در انتهای یک دستور که بسیار رایج است. البته اگر خطا به نوشتن؛ مربوط باشد نرم افزار شماره خط خطا را خط بعد مشخص می‌کند.

این دسته از خطاها را می‌توان در ۳ بخش جای داد:

- **خطای لغوی:** اگر هر یک از کلمات کلیدی موجود در برنامه نادرست تایپ شوند با خطای لغوی مواجه خواهیم شد. (غلط املائی در کلمات کلیدی یا کلمات رزو شده‌ی زبان مورد نظر)
- **خطای دستوری یا گرامری:** همه زبان‌های برنامه‌نویسی یک ساختار دستوری مشخصی دارند. به این ساختار اصطلاحاً syntax گفته می‌شود. اگر در نوشتن یک برنامه به زبان مورد نظر، اصول (دستورات گرامری) آن زبان را رعایت نکنیم، با خطای دستوری، مواجه خواهیم شد.
- **خطای مربوط به عدم وجود فایل‌های کتابخانه‌ای:** یکی دیگر از مقدماتی که به ما مربوط است، نوشتن کلیه هدر فایل‌های یک برنامه و در دسترس بودن کتابخانه‌های استفاده شده در برنامه است. عدم وجود هر یک از فایل‌های کتابخانه‌ای باعث بوجود آمدن خطا در برنامه می‌شود.

اگر در هنگام کامپایل کردن کد، یک یا چند مورد از موارد ذکر شده، فراهم نباشد، با کامپایل ارور مواجه خواهیم شد. وجود هرگونه خطای زمان کامپایل مانع از اجرای برنامه خواهد شد.

## ۲- خطاهای زمان اجرا **Run Time Error** : هر وقفه‌ای که در جریان عادی اجرای برنامه

پیش آید که معمولاً باعث سقط یا توقف برنامه می‌شود. مثل تقسیم عددی بر صفر

برخی دیگر از خطاهای زمان اجرا برنامه را از کار نمی‌اندازند بلکه برنامه همچنان کار می‌کند اما پاسخ‌های عجیب و نادرست می‌دهد. مثل عمل سرریزی<sup>۱۱</sup> برای یک متغیر یا خطای گرد کردن (در ادامه درباره هر یک بحث خواهد شد).

## ۳- خطاهای منطقی **Logic Error** : در این نوع خطاها کامپایلر ارور نمی‌دهد و برنامه اجرا


می‌شود اما نتیجه مورد نظر تولید نمی‌شود. (خروجی نادرست است) در همچنین شرایطی باید برنامه را از لحاظ منطقی و معنایی چک کنیم و با محدود کردن اجرای برنامه و مشاهده مقادیر متغیرهای تعریف شده می‌توانید خطا را شناسایی کنید. که یافتن چنین خطاهایی نسبت به خطاهای گرامری مشکل‌تر است. (معمولاً به آن باگ<sup>۱۲</sup> هم گفته می‌شود).

---

<sup>۱۱</sup> . overflow

<sup>۱۲</sup> .bug


## ۲-۴-۱. نمونه‌هایی از خطاهای زمان کامپایل

مثال ۲-۱: برنامه زیر را اجرا نمایید و خطاهای اعلام شده را برطرف نمایید. 

```
main.cpp X
1      #include <iostream>
2
3      using namespace std;
4
5      int main()
6      {
7          int x, y
8          cin>>x>>y;
9          z=x+y;
10         cout<<z;
11         getche();
12         return 0;
13     }
14
```

File	Line	Message
C:\Users\zahra...	8	error: expected initializer before 'cin'
C:\Users\zahra...	9	error: 'z' was not declared in this scope
C:\Users\zahra...	9	error: 'y' was not declared in this scope
C:\Users\zahra...	11	error: 'getche' was not declared in this scope
=== Build failed: 4 error(s), 0 warning(s) (0 minute(s), 0 second(s)) ===		

شکل ۲-۱۱: برنامه و خطاهای مثال ۲-۱

پاسخ: 

همانطور که قبلاً اشاره کرده بودیم، اگر برنامه خطا داشته باشد، در تب Build message، خطای مورد نظر یا لیستی از خطاها نمایش داده می‌شوند. محتوای این تب از ۳ ستون تشکیل شده است:

- ستون اول File: آدرس محل ذخیره‌سازی پروژه را نشان می‌دهد.
- ستون دوم Line: شماره خطی از برنامه که خطا دارد را نشان می‌دهد. با انتخاب هر خط، خطای مربوطه در برنامه با مربع قرمز رنگ نمایش داده می‌شود.

- ستون سوم Message: پیام مربوط به خطا را نشان می‌دهد. (باید توجه داشته باشید که پیغام‌های نمایش داده شده توسط IDE‌های مختلف لزوماً یکسان نیستند.)


حال به بررسی خطاهای برنامه بالا می‌پردازیم:

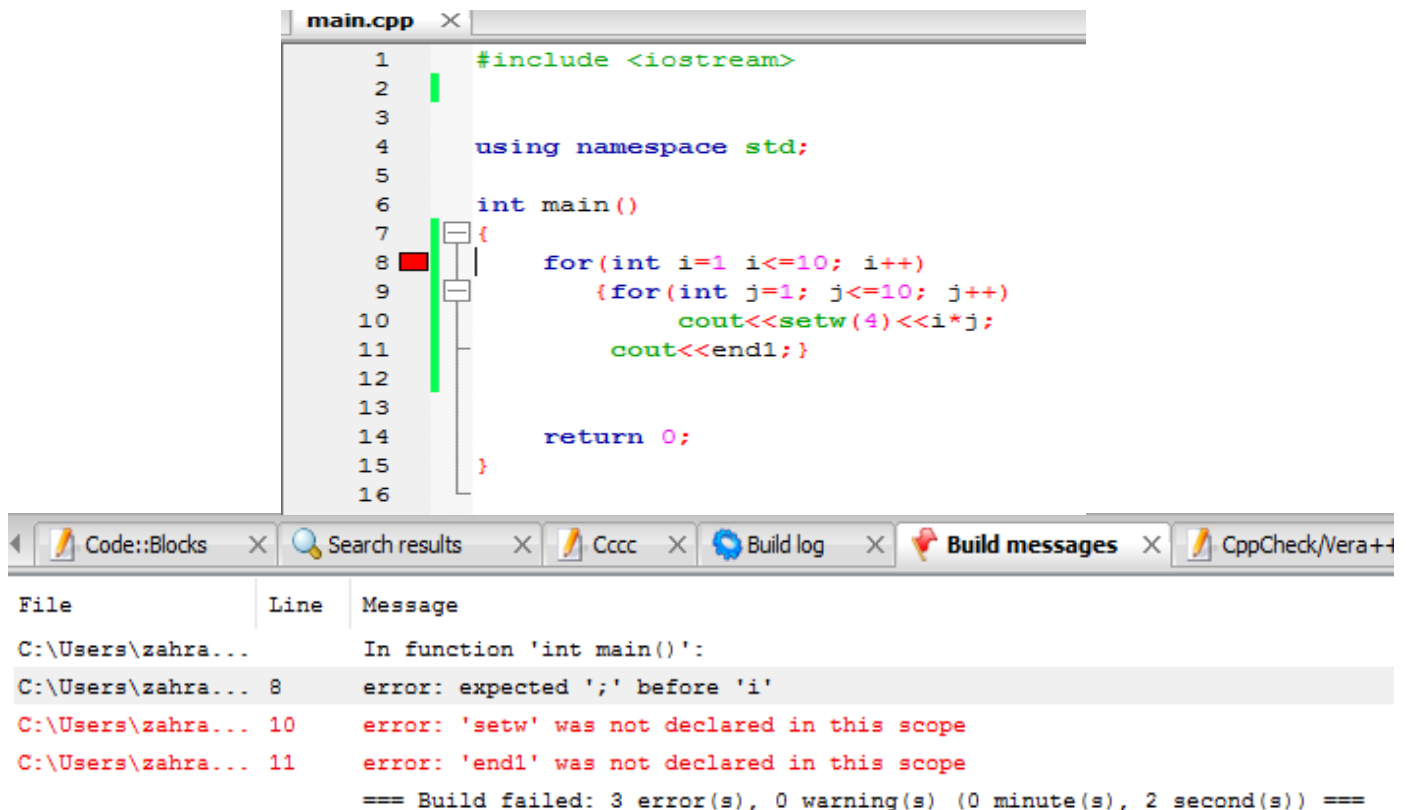
خط ۸: عدم رعایت سمی کولون در خط قبلی را متذکر می‌شود.

خط ۹: عدم تعریف متغیر z را تذکر می‌دهد.

خط ۱۱: عدم تعریف فایل کتابخانه‌ای تابع getch را تذکر می‌دهد. (این تابع در فایل کتابخانه‌ای conio.h تعریف شده است.)

مثال ۲-۲: برنامه زیر قرار است یک جدول ضرب ۱۰\*۱۰ را نمایش دهد اما به هنگام کامپایل کردن با خطاهایی مواجه می‌شویم که باید آنها را برطرف نماییم.

پاسخ به عنوان تمرین به عهده دانشجو می‌باشد. 



```


main.cpp
1  #include <iostream>
2
3
4  using namespace std;
5
6  int main()
7  {
8      for(int i=1 i<=10; i++)
9          {for(int j=1; j<=10; j++)
10             cout<<setw(4)<<i*j;
11             cout<<endl;}
12
13
14     return 0;
15 }
16

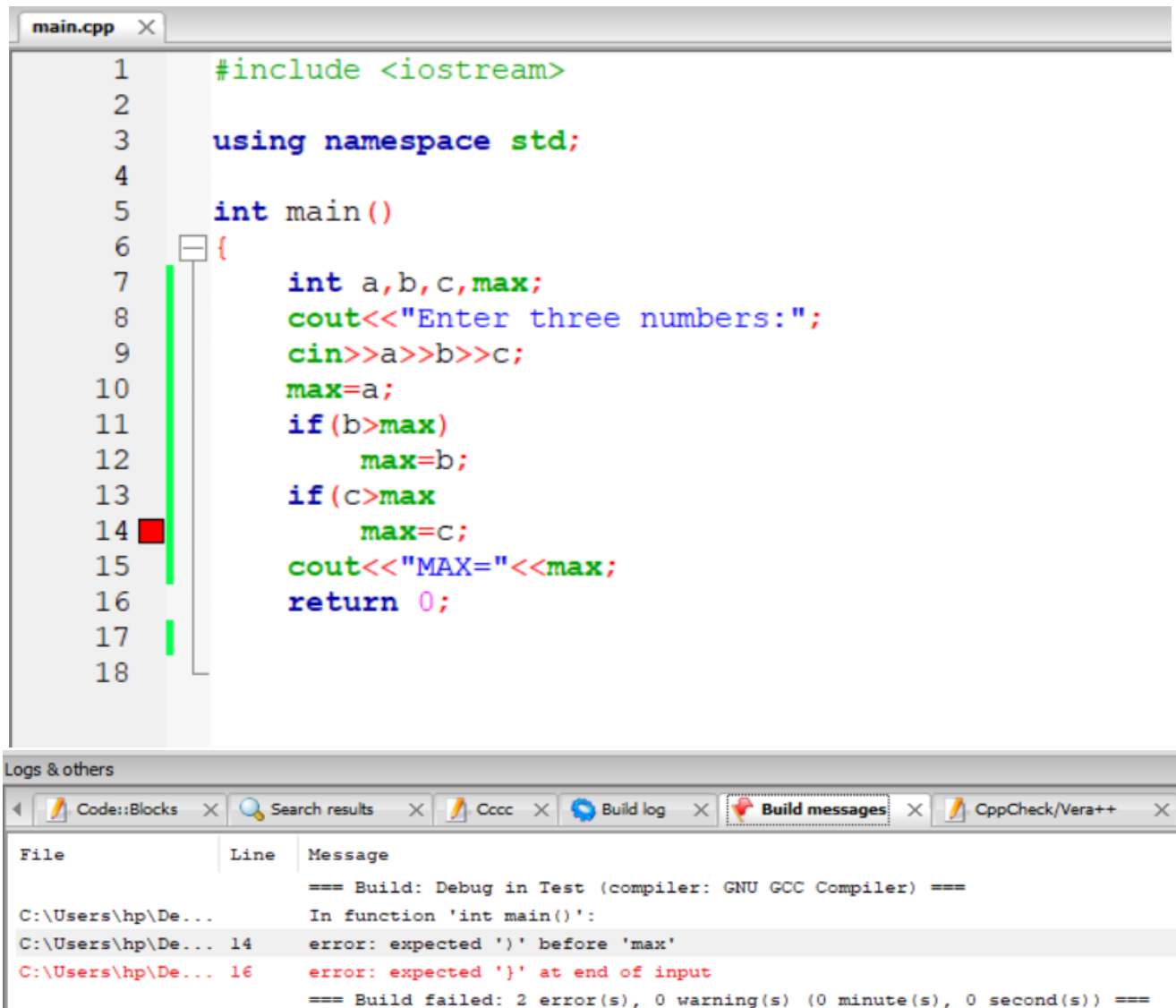
```

File	Line	Message
C:\Users\zahra...		In function 'int main()':
C:\Users\zahra...	8	error: expected ';' before 'i'
C:\Users\zahra...	10	error: 'setw' was not declared in this scope
C:\Users\zahra...	11	error: 'endl' was not declared in this scope
=== Build failed: 3 error(s), 0 warning(s) (0 minute(s), 2 second(s)) ===		

شکل ۲-۱۲: برنامه و خطاهای مثال ۲-۲

مثال ۲-۳: برنامه زیر قرار است ۳ عدد صحیح از ورودی دریافت کرده سپس ماکزیمم مقدار آن را نمایش دهد، اما به هنگام کامپایل کردن با خطاهایی مواجه می‌شویم که باید آنها را برطرف نماییم.

پاسخ به عنوان تمرین به عهده دانشجو می‌باشد. 



```

1  #include <iostream>
2
3  using namespace std;
4
5  int main()
6  {
7      int a,b,c,max;
8      cout<<"Enter three numbers:";
9      cin>>a>>b>>c;
10     max=a;
11     if(b>max)
12         max=b;
13     if(c>max
14         max=c;
15     cout<<"MAX="<<max;
16     return 0;
17
18

```

File	Line	Message
=== Build: Debug in Test (compiler: GNU GCC Compiler) ===		
In function 'int main()':		
C:\Users\hp\De...	14	error: expected ')' before 'max'
C:\Users\hp\De...	16	error: expected ')' at end of input
=== Build failed: 2 error(s), 0 warning(s) (0 minute(s), 0 second(s)) ===		

شکل ۲-۱۳: برنامه و خطاهای مثال ۲-۳



## ۲-۴-۲. نمونه هایی از خطاهای زمان اجرا

### سرریزی عددی

نوع صحیح **long** یا نوع ممیز شناور **double** محدوده وسیعی از اعداد را می‌توانند نگهداری کنند. به بیان ساده‌تر، متغیری که از نوع **long** یا **double** باشد، گنجایش زیادی دارد. اما حافظه رایانه‌ها متناهی است. یعنی هر قدر هم که یک متغیر گنجایش داشته باشد، بالاخره مقداری هست که از گنجایش آن متغیر بیشتر باشد. اگر سعی کنیم در یک متغیر مقداری قرار دهیم که از گنجایش آن متغیر فراتر باشد، متغیر سرریزه می‌شود. مثل یک لیوان آب که اگر بیش از گنجایش آن در لیوان آب بریزیم، سرریز می‌شود. در چنین حالتی می‌گوییم که خطای سرریزی رخ داده است.

جدول ۲-۱: محدوده انواع داده‌ها در C++

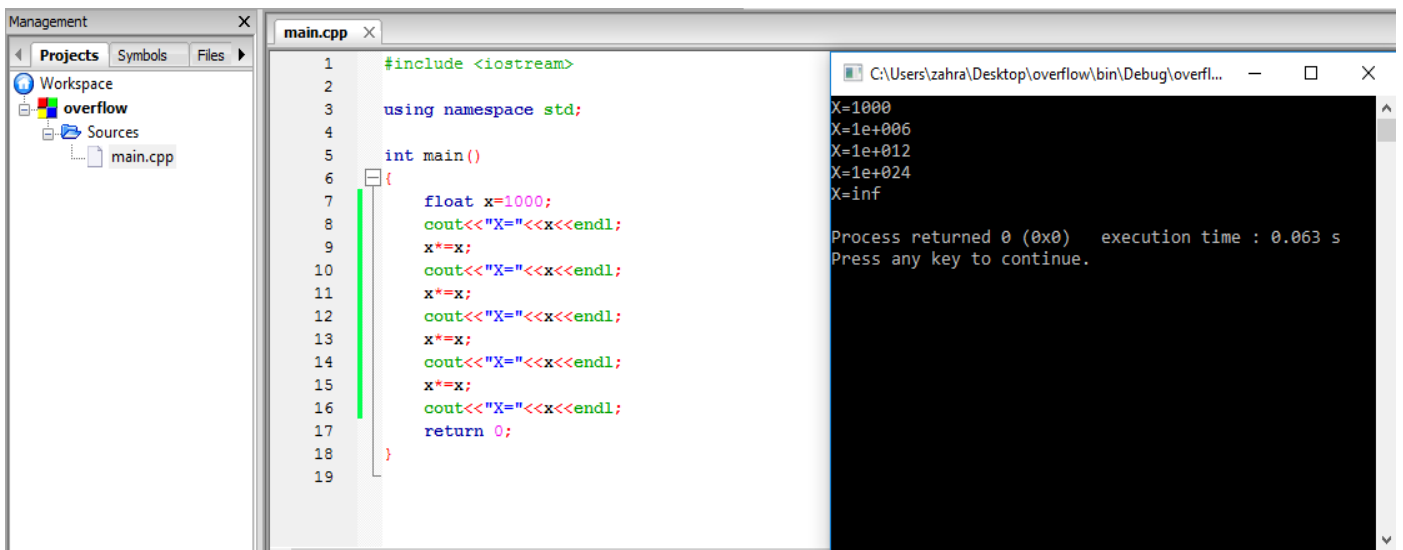
Data Types	Sizes in byte	Sizes in bits	Range formula $2^n-1$	Ranges
int	4 bytes	32bits	$2^{32}-1$	-2,147,483,648 to 2,147,483,647
unsigned int	4 bytes	32 bits	$2^{32}-1$	0 to 4294967295
float	4 bytes	32 bits	$2^{32}-1$ (5 points)	$3.4 \times 10^{-38}$ to $3.4 \times 10^{+38}$
double	8 bytes	64 bits	$2^{64}-1$ (15 points)	$1.7 \times 10^{-308}$ to $1.7 \times 10^{+308}$
long double	10 bytes	80 bits	$2^{80}-1$ (19 points)	$1.7 \times 10^{-4932}$ to $1.7 \times 10^{+4932}$
char	1 byte	8 bits	$2^8-1$	0 to 255

### ۲-۴-۲-۱. سرریزی عدد اعشاری

این برنامه  $x$  را به طور مکرر به توان می‌رساند تا اینکه سرریز شود. در آخرین خط خروجی **inf** به معنی بی‌نهایت<sup>۱۳</sup> یا بی‌انتهاست.

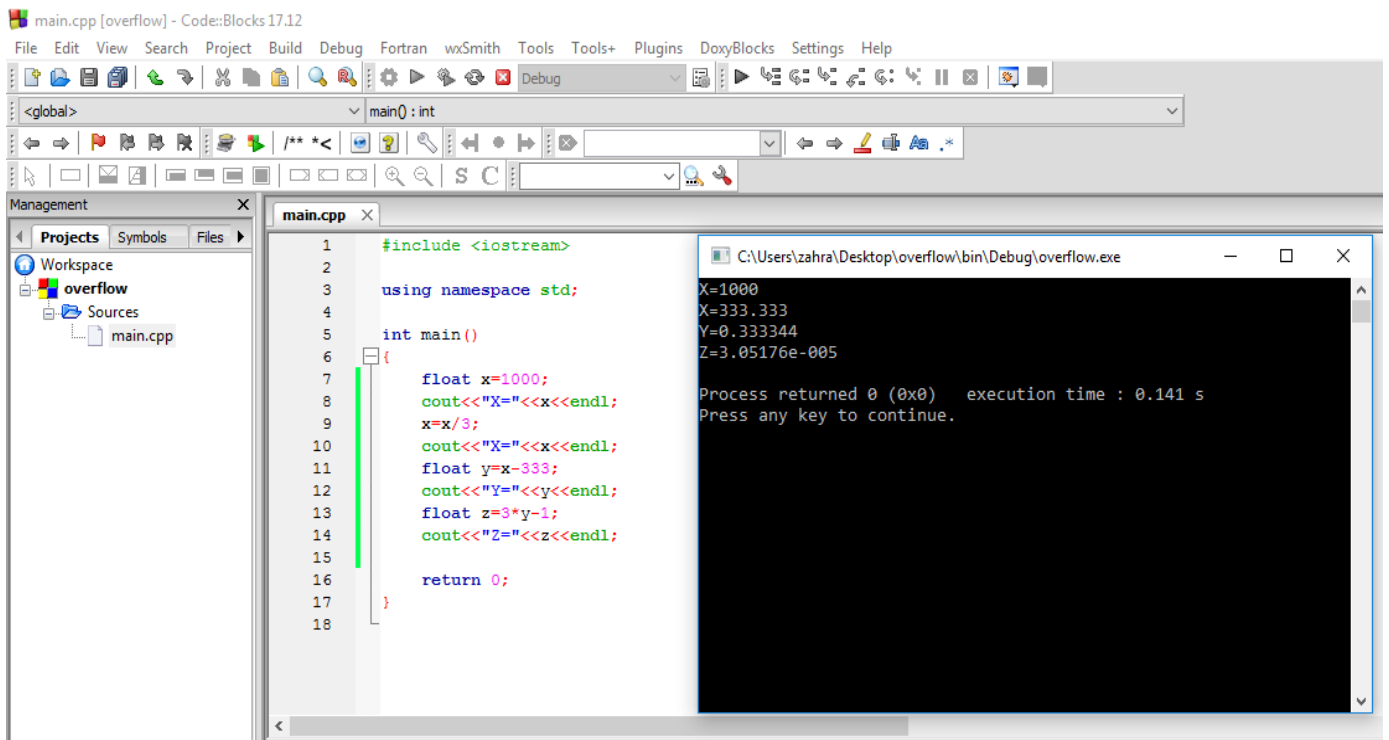
سرریزی عدد صحیح به عنوان تمرین به عهده دانشجو می‌باشد.

<sup>۱۳</sup>.infinity



شکل ۲-۱۴: سرریزی عدد اعشاری

## ۲-۲-۴-۲. خطای گرد کردن



شکل ۲-۱۵: نمونه‌ای از خطای گرد کردن

در این برنامه، مقدار متغیر  $x$  برابر با ۱۰۰۰ است، سپس این مقدار بر ۳ تقسیم می‌شود در صورتی که حاصل را به شکل عدد مخروطن در نظر بگیریم، حاصل این تقسیم برابر با  $\frac{1}{3}$  می‌شود. در ادامه برنامه، از مقدار  $x$ ، ۳۳۳ واحد کم می‌شود و حاصل در متغیر  $y$  قرار می‌گیرد بنابراین مقدار  $y$  برابر با  $\frac{1}{3}$  می‌شود. سپس مقدار  $y$  در ۳ ضرب می‌شود و یک واحد از آن کم می‌شود در اینجا حاصل این عبارت محاسباتی برابر با صفر و در متغیر  $z$  ریخته می‌شود. در صورتی که مقدار متغیر  $z$  در خروجی، به دلیل گرد کردن اعداد حاصل در محاسبات، مقداری نزدیک به صفر است.

### تمرین ۱-۲

عنوان: به دست آوردن ریشه‌های یک معادله درجه دو، با ضرایب اعشاری

هدف: بررسی خطای گرد کردن در محاسبات

## ۲-۴-۳. روش‌های یافتن خطاهای منطقی و برطرف کردن آنها (خطایابی در کد بلاکس)

خطایابی منطقی یکی از دشوارترین مراحل خطایابی است. این مرحله از خطایابی، یکی از پرهزینه‌ترین مراحل برنامه‌سازی است. یکی از ساده‌ترین و در عین حال پر استفاده‌ترین راهکارهای خطایابی منطقی، آزمودن برنامه در برابر تعداد مشخصی ورودی و خروجی آزمون است. به عبارت دیگر، داده آزمون برای یک برنامه عبارت است از تعدادی مشخص ورودی به همراه خروجی‌های مرتبط. در صورتی که برنامه حتی به ازای یک مورد از ورودی‌های آزمون، خروجی مناسب تولید نکند، می‌توان مطمئن بود که برنامه دارای خطای منطقی است.

نکته مهم در خطایابی این است که هنگامی که به یک خطا رسیدیم، نباید ناامید شویم! درست است که یک برنامه ایده آل باید بدون خطا باشد، اما رسیدن به چنین ایده آلی در عمل، اگر نگوییم ناممکن، اما دشوار است. هنگامی که شما خطای برنامه خود را فهمیده‌اید، در راستای تصحیح آن تلاش می‌کنید و این نکته، خود ارزشمند است.

ساده‌ترین و معمول‌ترین شیوه برای خطایابی برنامه‌ها، اضافه کردن عبارت‌های متوالی `printf()` یا `cout` در طول برنامه است. به‌چنین قطعه‌کدهایی، قطعه‌کدهای رهگیر گفته می‌شود. (از طریق دستورات خروجی یا دستورات چاپی می‌توانید مقادیر متغیرها را در هر قسمت از برنامه به دست آورید و اینگونه به محل بروز خطا برسید.)

استفاده از روش فوق، فرایندی زمانبر است، به ویژه برای مواردی که تعداد خطاها بیشتر یا پیچیدگی بیشتری داشته باشند. به همین منظور، لازم است تا یک روش مناسب برای خطایابی استفاده کنیم. اولین چیزی که به ذهن می‌رسد، برنامه‌ای است که خط به خط برنامه مورد نظر را اجرا کند و پس از اجرای هر خط، مقادیر متغیر مورد نیاز را به ما نشان دهد. شاید بهتر باشد، اجرای برنامه را تا خط مشخصی ادامه داده و در آن خط برای واری مقادیر متغیرهای برنامه متوقف شود و پس از بررسی، اجرای برنامه را از سر گیرد. خوشبختانه چنین برنامه‌ای وجود دارد که به آن خطایاب یا `Debugger` گفته می‌شود. یکی از خطایاب‌های بسیار قدرتمند و البته متن‌باز که در `Code::Blocks` نیز استفاده می‌شود، `GDB` یا `GNU DeBugger` است.

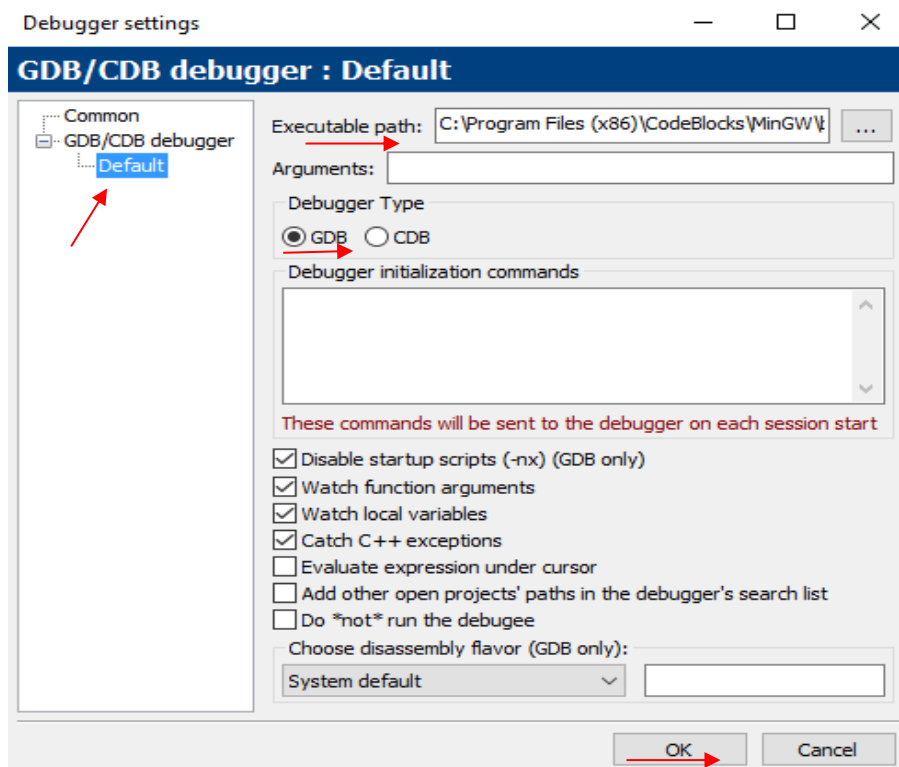
## تنظیمات لازم برای خطایاب یا Debugger:

- In the Code::Blocks IDE, navigate **Settings** -> **Debugger**
- In the tree control at the right, select **Common** -> **GDB/CDB debugger** -> **Default**.
- Then in the dialog at the left you can enter **Executable path** and choose **Debugger type** = GDB or CDB, as well as configuring various other options.

C:\Program Files\CodeBlocks\MinGW\bin\gdb۳۲.exe

در صورتیکه خطایاب (Debugger) در کد بلاکس شما فعال نبود، می‌توانید به روش زیر آن را فعال نمایید:

- وارد منوی **Settings** شده و گزینه **Debugger...** را انتخاب نمایید.
- در سمت چپ پنجره باز شده به ترتیب روی **Common** -> **GDB/CDB debugger** -> **Default** کلیک می‌کنیم.
- سپس در فیلد **Executable path**، مسیر ذخیره‌سازی فایل اجرایی خطایاب ( **C:\Program Files\CodeBlocks\MinGW\bin\gdb۳۲.exe** ) را می‌دهیم.
- و در نهایت نوع خطایاب ( **Debugger type** ) را روی گزینه **GDB** تنظیم و **Ok** می‌کنیم.




شکل ۲-۱۶: پنجره تنظیمات خطایاب

این روش با محدود کردن اجرای برنامه از طریق Breakpoint ها و مشاهده مقادیر متغیرهای تعریف شده، خطا را شناسایی می‌کند. برای دیدن مقدار متغیرها از منوی Debug و زیرمنوی Debugging Windows گزینه Watches را انتخاب کنید. در پنجره باز شده نام متغیرها و مقادیر آنها را می‌توانید مشاهده کنید و به این ترتیب محل بروز خطا را تشخیص دهید.

در قدم بعد باید محدوده اجرای برنامه را مشخص کرد. به این منظور از Breakpoint ها استفاده می‌شود. زمانی که برای دستوری Breakpoint مشخص شود برنامه با رسیدن به این دستور مکث می‌کند و وارد محیط دیباگ می‌شود. سپس می‌توان از دستورات Step و سایر موارد استفاده کرد. برای تعیین Breakpoint یک دستور می‌توان از روش‌های زیر استفاده کرد:

- بر روی خط دستور راست کلیک کرده و گزینه Add Breakpoint
  - کلیک در ابتدای خط دستور
  - بر روی خط دستور کلیک کرده و کلید F5
  - بر روی خط دستور کلیک کرده، از منوی Debug گزینه Toggle breakpoint
- در ابتدای خطی که برای آن breakpoint تعیین می‌کنید یک دایره قرمز رنگ قرار می‌گیرد. بعد از تعیین Breakpoint ها، برای اجرای برنامه می‌توانید:

- از منوی Debug گزینه Start/Continue را انتخاب کنید
- از کلید F8 استفاده نمایید.
- یا از آیکن Debug/Continue که در بالای صفحه به شکل مثلث قرمز رنگ می‌باشد

 استفاده کنید.

برنامه تا رسیدن به اولین Breakpoint اجرا می‌شود و صفحه خروجی نمایش داده می‌شود.

در ادامه، موارد زیر به شما این امکان را می‌دهند که مراحل پیشروی دستورات را همان گونه که تمایل دارید انتخاب کنید:



- **Next line**: این گزینه برنامه را خط به خط اجرا می‌کند. می‌توانید از منوی Debug انتخاب کنید یا کلید F7 را فشار دهید.
  - **Step into**: عملکرد این گزینه شبیه Next line هست با این تفاوت که در Next line زمانی که با دستور فراخوانی یک تابع روبرو شود، وارد آن تابع نمی‌شود و مکان نما به خط بعدی منتقل می‌شود. اما در Step into اجرای برنامه به آن تابع منتقل می‌شود و مکان نما ابتدای اولین دستور تابع قرار می‌گیرد و بعد از اجرای کامل آن تابع به تابع فراخوانی کننده بر می‌گردد. این گزینه را می‌توانید از منوی Debug انتخاب کنید یا از کلیدهای ترکیبی Shift+F7 استفاده کنید.
  - **Step out**: این گزینه از جهت منطقی عملکردی بر عکس Step into دارد. اگر با اجرای Step into یا دلیل دیگری وارد یک تابع شده‌اید می‌توانید با اجرای Step out از آن خارج شده و مکان نما به تابع فراخوانی کننده منتقل کنید. این گزینه را می‌توانید از منوی Debug انتخاب کنید یا از کلیدهای ترکیبی Ctrl+F7 استفاده کنید.
  - **Next instruction**: این گزینه سراغ دستورات عمل بعدی می‌رود. و همه دستورات را مرحله به مرحله چک می‌کند.
  - **Step into instruction**: عملکردی مشابه Step into دارد.
  - **Run to cursor**: مکان نما را روی خطی که قصد دارید عمل اجرا تا آن مکان انجام شود قرار دهید. سپس از منوی Debug گزینه Run to cursor را انتخاب کنید یا کلید F4 را فشار دهید. در آغاز خط مشخص شده، مثلث زرد رنگی قرار می‌گیرد و برنامه بلافاصله تا این دستور اجرا شده و سپس مکث می‌کند.
- هنگامی که خطایاب اجرای برنامه در یک خط را متوقف می‌کند، در کنار آن خط یک نشانگر زردرنگ قرار می‌گیرد.

در این قسمت، همزمان با مشاهده چگونگی اجرای برنامه، می‌توان از پنجره Watches مقادیر متغیرها را نیز مشاهده نمود.

در نهایت برای خروج از این حالت باید روی آیکن Stop Debugger موجود در بالای صفحه یا از منوی Debug گزینه مورد نظر را انتخاب نمود.

لازم به ذکر است، تمامی دستورات ذکر شده در بالا، در نوار ابزار برنامه نیز وجود دارند.

در این بخش، ما از خطایاب‌ها برای یافتن خطای برنامه در حین اجرا استفاده کردیم. یکی دیگر از مزایایی که خطایاب‌ها می‌توانند داشته باشند، فهمیدن عملکرد برنامه‌ها و الگوریتم‌ها با استفاده از آن‌ها است.




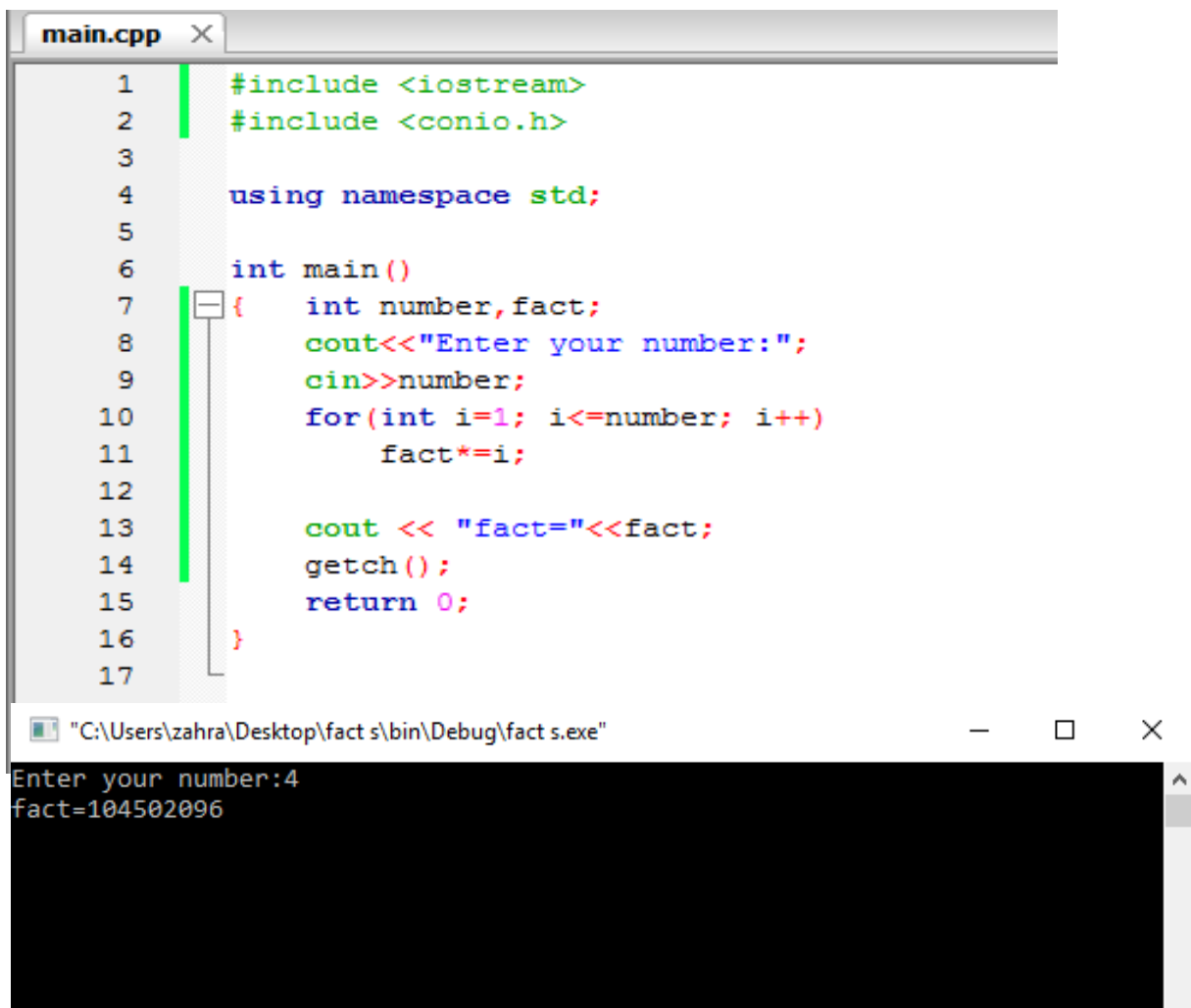
## ۲-۴-۳-۱. نمونه هایی از خطاهای منطقی

مثال ۲-۴:

عنوان برنامه: فاکتوریل

هدف: محاسبه‌ی فاکتوریل عدد وارد شده

 توضیح برنامه: برای بدست آوردن فاکتوریل یک عدد، لازم است حاصلضرب آن عدد تا یک را محاسبه نماییم. از این رو کفایت از یک حلقه For یا While استفاده نماییم و زیرمجموعه آن، دستور ضرب اعداد (به این صورت که هر دو عدد در هم ضرب می‌شوند و حاصل آن در متغیر دیگری مثل fact قرار می‌گیرند تا اینکه به عدد انتهای بازه حلقه برسیم و حلقه خاتمه یابد) را داشته باشیم.



```

main.cpp x
1  #include <iostream>
2  #include <conio.h>
3
4  using namespace std;
5
6  int main()
7  {
8      int number, fact;
9      cout<<"Enter your number:";
10     cin>>number;
11     for(int i=1; i<=number; i++)
12         fact*=i;
13
14     cout << "fact="<<fact;
15     getch();
16     return 0;
17 }

```

"C:\Users\zahra\Desktop\fact s\bin\Debug\fact s.exe"

```

Enter your number:4
fact=104502096

```

شکل ۲-۱۷: برنامه و خروجی مثال ۲-۴

اما وقتی به خروجی برنامه دقت می‌کنیم، مشاهده می‌شود که محاسبه فاکتوریل عدد وارد شده، صحیح نمی‌باشد. در نتیجه باید به دنبال یک یا چند خطای منطقی در روند اجرای برنامه باشیم.

حال به شرح مراحل خطایابی که در بخش قبلی توضیحات کامل آن داده شده است می‌پردازیم:

۱- ابتدا یک Breakpoint در یکی از خطوط ابتدایی برنامه (مثلا خط ۹) قرار می‌دهیم تا خط به خط برنامه را اجرا کنیم.

۲- برای اجرای برنامه بر روی Debug/Continue کلیک می‌کنیم تا پنجره خروجی باز شود.

۳- برای اینکه تغییرات مربوط به مقدار متغیرها را نیز مشاهده کنیم، پنجره Watches را نیز باز می‌کنیم.

۴- حال با استفاده از کلید Next Line خط به خط برنامه را اجرا می‌کنیم و در عین حال، پنجره خروجی و Watches را نیز مورد توجه قرار می‌دهیم تا به خطایمان پی ببریم.

در همان تکرار اول حلقه، با توجه به پنجره Watches، متوجه مقدار زباله ای که در متغیر fact قرار گرفته می‌شویم و همین مقدار زباله، منجر به تولید نتیجه نادرست می‌شود. پس نتیجه می‌گیریم که باید متغیر fact را حتما مقداردهی اولیه نماییم.


Watches	
Function arguments	
Locals	
number	4
fact	4354254
i	1

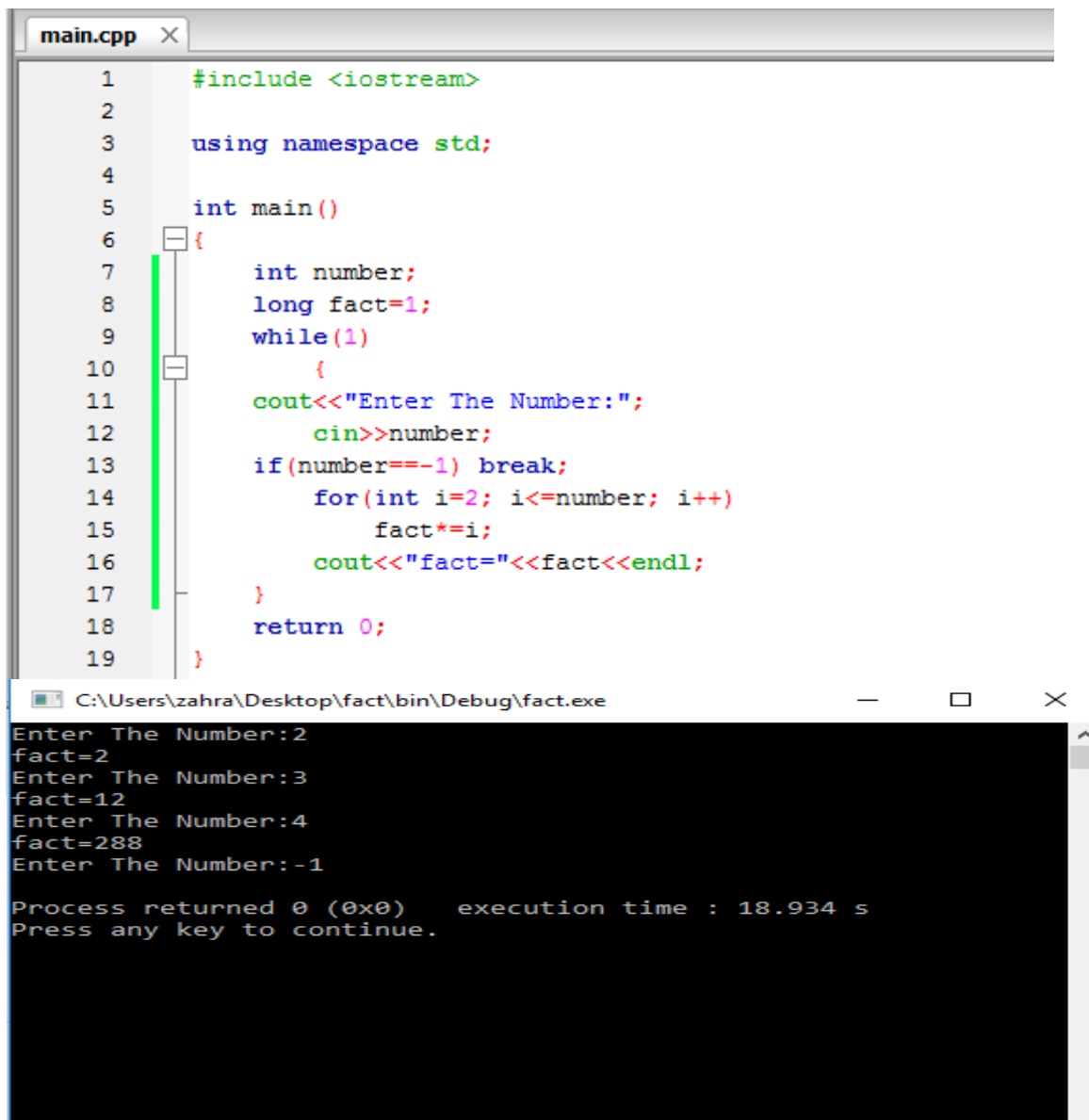
شکل ۲-۱۸: پنجره Watches مثال ۲-۴

مثال ۲-۵: 

## عنوان برنامه: محاسبه فاکتوریل

هدف: محاسبه‌ی فاکتوریل هر عددی که وارد می‌شود، تا زمانی که عدد ۱- وارد شود.

 توضیح برنامه: برای اینکه اعداد، پشت سر هم وارد شوند، به یک حلقه همواره درست نیاز داریم که زیرمجموعه آن، اعداد را از ورودی دریافت کنیم و شرط خاتمه حلقه را وارد شدن عدد ۱- قرار می‌دهیم. در صورتی که عدد وارد شده مخالف ۱- باشد، وارد حلقه دیگری جهت محاسبه فاکتوریل می‌شود. و در نهایت مقدار محاسبه شده یعنی مقدار متغیر fact چاپ می‌شود.



```

main.cpp x
1  #include <iostream>
2
3  using namespace std;
4
5  int main()
6  {
7      int number;
8      long fact=1;
9      while(1)
10     {
11         cout<<"Enter The Number:";
12         cin>>number;
13         if(number== -1) break;
14         for(int i=2; i<=number; i++)
15             fact*=i;
16         cout<<"fact="<<fact<<endl;
17     }
18     return 0;
19 }

```

```

C:\Users\zahra\Desktop\fact\bin\Debug\fact.exe
Enter The Number:2
fact=2
Enter The Number:3
fact=12
Enter The Number:4
fact=288
Enter The Number:-1

Process returned 0 (0x0)   execution time : 18.934 s
Press any key to continue.

```

شکل ۲-۱۹: برنامه و خروجی مثال ۲-۵

اما وقتی به خروجی برنامه دقت می‌کنیم، مشاهده می‌شود که محاسبه فاکتوریل اولین عدد درست و مابقی نادرست هستند. در نتیجه باید به دنبال یک یا چند خطای منطقی در روند اجرای برنامه باشیم.

حال به شرح مراحل خطایابی می‌پردازیم:

۱- ابتدا یک Breakpoint در یکی از خطوط ابتدایی برنامه (مثلا خط ۷) قرار می‌دهیم تا خط به خط برنامه را اجرا کنیم.

۲- برای اجرای برنامه بر روی Debug/Continue کلیک می‌کنیم تا پنجره خروجی باز شود.

۳- برای اینکه تغییرات مربوط به مقدار متغیرها را نیز مشاهده کنیم، پنجره Watches را نیز باز می‌کنیم.

۴- حال با استفاده از کلیدهای Next Line یا Step Into خط به خط برنامه را اجرا می‌کنیم و در عین حال، پنجره خروجی و Watches را مورد توجه قرار می‌دهیم تا به خطایمان پی ببریم. وقتی عدد اول را وارد می‌کنیم، برنامه به درستی اجرا شده و خروجی درست را نیز تولید می‌کند اما وقتی عدد دوم را وارد می‌کنیم و به خط ۱۵ برنامه (محاسبه fact) می‌رسیم از آنجایی که مقدار متغیر fact همان مقدار مربوط به عدد قبل است در نتیجه محاسبات و خروجی ما نادرست می‌شوند.

The screenshot shows a C++ IDE with a file named 'main.cpp'. The code is as follows:

```

1 #include <iostream>
2
3 using namespace std;
4
5 int main()
6 {
7     int number;
8     long fact=1;
9     while(1)
10    {
11        cout<<"Enter The Number:";
12        cin>>number;
13        if(number==-1) break;
14        for(int i=2; i<=number; i++)
15            fact*=i;
16        cout<<"fact="<<fact<<endl;
17    }
18    return 0;
19 }
20

```

A red dot indicates a breakpoint is set at line 15. The console window shows the following output:

```

Enter The Number:3
fact=6
Enter The Number:4

```

The Watches window shows the following variables and their values:

Function argument	
number	4
fact	6
i	2

شکل ۲-۲۰: خطایابی مثال ۲-۵

پس ما باید در مقدار متغیر fact تجدید نظر کنیم و در انتهای حلقه یا ابتدای حلقه، هر بار مقدار متغیر fact را به مقدار ۱ تغییر دهیم تا محاسبات سایر اعداد نیز ارزش درستی پیدا کنند.

```

main.cpp X
1      #include <iostream>
2
3      using namespace std;
4
5      int main()
6      {
7          int number;
8          long fact=1;
9          while(1)
10         {
11             cout<<"Enter The Number:";
12             cin>>number;
13             if(number==-1) break;
14             for(int i=2; i<=number; i++)
15                 fact*=i;
16             cout<<"fact="<<fact<<endl;
17             fact=1;
18         }
19         return 0;
20     }
21


```

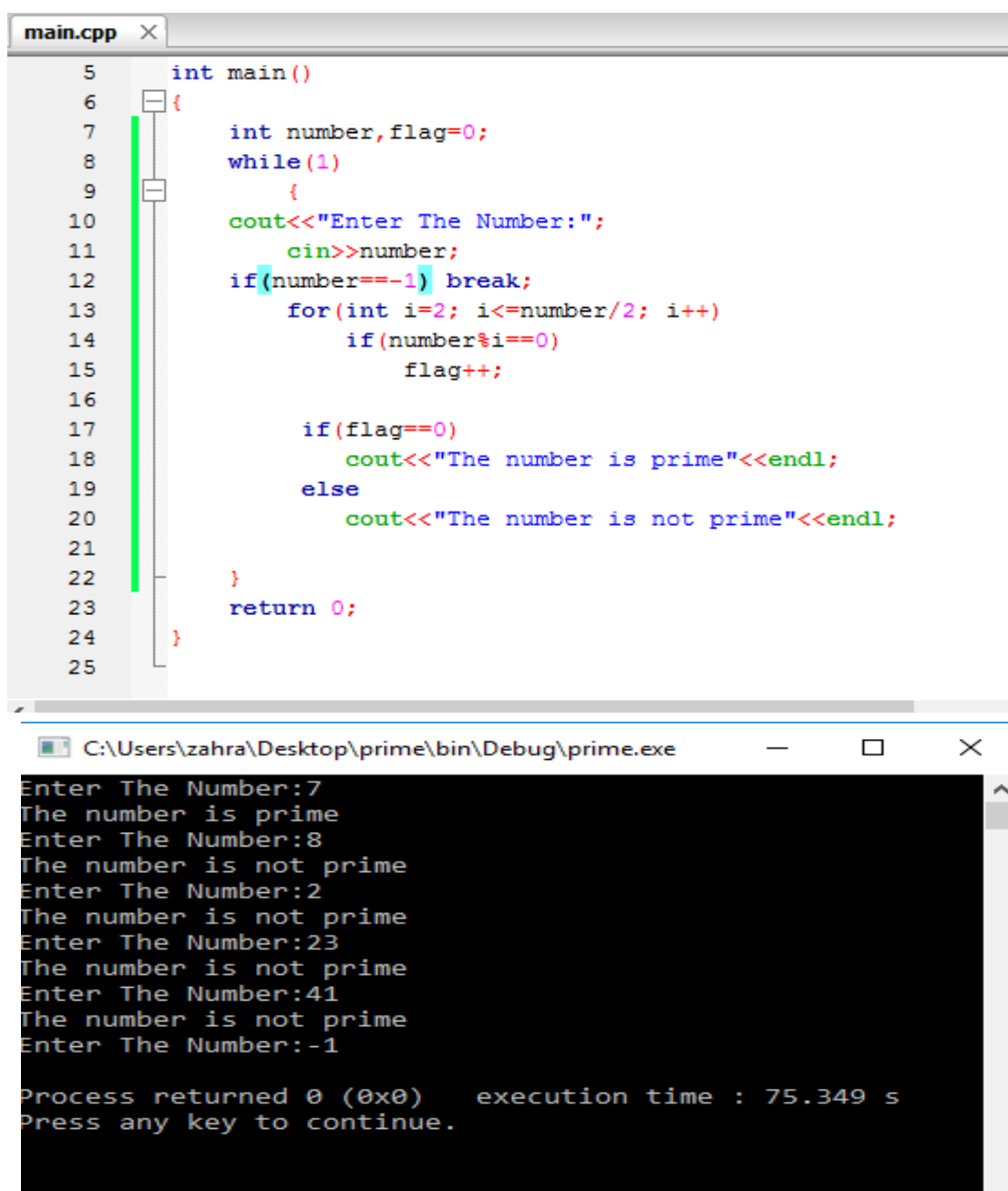
شکل ۲-۲۱: برنامه اصلاح شده مثال ۲-۵

مثال ۲-۶: 

عنوان برنامه: عدد اول

هدف: تشخیص اول بودن هر عددی که وارد می‌شود، تا زمانی که عدد ۱ - وارد شود.

 توضیح برنامه: کد برنامه به همراه خروجی آن در ادامه آمده است. توضیح برنامه و تشخیص خطای آن به عهده دانشجو می‌باشد.



```

main.cpp x
5  int main()
6  {
7      int number, flag=0;
8      while(1)
9      {
10     cout<<"Enter The Number:";
11     cin>>number;
12     if(number==-1) break;
13     for(int i=2; i<=number/2; i++)
14         if(number%i==0)
15             flag++;
16
17     if(flag==0)
18         cout<<"The number is prime"<<endl;
19     else
20         cout<<"The number is not prime"<<endl;
21
22     }
23     return 0;
24 }
25
C:\Users\zahra\Desktop\prime\bin\Debug\prime.exe
Enter The Number:7
The number is prime
Enter The Number:8
The number is not prime
Enter The Number:2
The number is not prime
Enter The Number:23
The number is not prime
Enter The Number:41
The number is not prime
Enter The Number:-1
Process returned 0 (0x0)   execution time : 75.349 s
Press any key to continue.


```

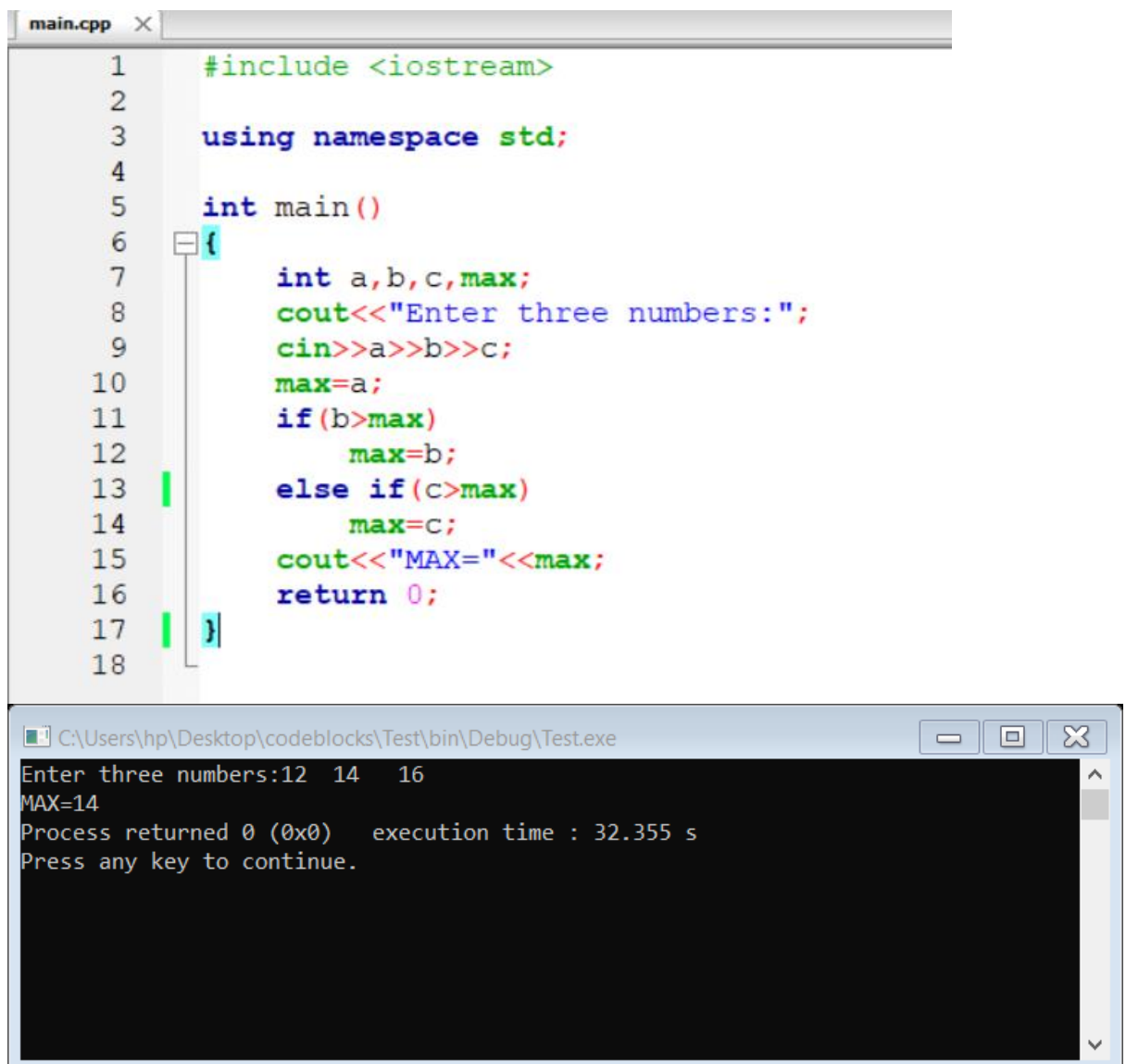
شکل ۲-۲۲: برنامه و خروجی مثال ۲-۶

مثال ۲-۷: 

عنوان برنامه: محاسبه ماکزیمم مقدار

هدف: به دست آوردن بیشترین مقدار بین ۳ عدد وارد شده

 توضیح برنامه: کد برنامه به همراه خروجی آن در ادامه آمده است. توضیح برنامه و تشخیص خطای آن به عهده دانشجو می باشد.



The image shows a screenshot of a C++ IDE with two windows. The top window, titled 'main.cpp', contains the following code:

```

1  #include <iostream>
2
3  using namespace std;
4
5  int main()
6  {
7      int a,b,c,max;
8      cout<<"Enter three numbers:";
9      cin>>a>>b>>c;
10     max=a;
11     if(b>max)
12         max=b;
13     else if(c>max)
14         max=c;
15     cout<<"MAX="<<max;
16     return 0;
17 }
18

```

The bottom window shows the execution output for the program:

```

C:\Users\hp\Desktop\codeblocks\Test\bin\Debug\Test.exe
Enter three numbers:12 14 16
MAX=14
Process returned 0 (0x0)   execution time : 32.355 s
Press any key to continue.

```

شکل ۲-۲۳: برنامه و خروجی مثال ۲-۷

## ۲-۵. کامپایل جداگانه توابع و فایل DLL

گاهی اوقات لازم می‌شود که ما از توابعی که قبلاً ساخته ایم در یک برنامه جدید استفاده کنیم و فقط فراخوانی آن تابع را به برنامه اضافه کنیم بدون اینکه تعریف و بدنه تابع را مجدداً بنویسیم. برای این کار باید تعریف و بدنه توابع را در فایل‌های جداگانه‌ای قرار داد و سپس به برنامه اصلی که آن توابع را به کار می‌گیرد الصاق کرد.

اغلب این طور است که تعریف و بدنه توابع در فایل‌های جداگانه‌ای قرار می‌گیرد. این فایل‌ها به طور مستقل کامپایل می‌شوند و سپس به برنامه اصلی که آن توابع را به کار می‌گیرد الصاق<sup>۱۴</sup> می‌شوند. توابع کتابخانه ++C استاندارد به همین شکل پیاده‌سازی شده‌اند و هنگامی که یکی از آن توابع را در برنامه‌هایتان به کار می‌برید باید با دستور راهنمای پیش‌پردازنده، فایل آن توابع را به برنامه تان ضمیمه کنید مانند توابع ورودی و خروجی `cin` و `cout`، که در فایل کتابخانه‌ای `iostream` تعریف شده‌اند. این کار چند مزیت دارد. اولین مزیت «مخفی‌سازی اطلاعات» است. یعنی این که توابع لازم را در فایل جداگانه‌ای تعریف و کامپایل کنید و سپس آن فایل را به همراه مشخصات توابع به برنامه‌نویس دیگری بدهید تا برنامه اصلی را تکمیل کند. به این ترتیب آن برنامه‌نویس از جزئیات توابع و نحوه اجرای داخلی آنها چیزی نمی‌داند و فقط می‌داند که چطور می‌تواند از آنها استفاده کند. در نتیجه اطلاعاتی که دانستن آنها برای برنامه‌نویس ضروری نیست از دید او مخفی می‌ماند. تجربه نشان داده که پنهان‌سازی اطلاعات، فهمیدن برنامه اصلی را آسان می‌کند و پروژه‌های بزرگ با موفقیت اجرا می‌شوند.

مزیت دیگر این است که توابع مورد نیاز را می‌توان قبل از این که برنامه اصلی نوشته شود، جداگانه آزمایش نمود. وقتی یقین کردید که یک تابع مفروض به درستی کار می‌کند، آن را در یک فایل ذخیره کنید و جزئیات آن تابع را فراموش کنید و هر وقت که به آن تابع نیاز داشتید با خیالی راحت از آن در برنامه‌هایتان استفاده نمایید. نتیجه این است که تولید توابع مورد نیاز و تولید برنامه اصلی، هم زمان و مستقل از هم پیش می‌رود بدون این که یکی منتظر دیگری بماند. به این دیدگاه «بسته‌بندی نرم افزار» می‌گویند.

سومین مزیت این است که در هر زمانی به راحتی می‌توان تعریف توابع را عوض کرد بدون این که لازم باشد برنامه اصلی تغییر یابد. فرض کنید تابعی برای مرتب کردن فهرستی از اعداد ایجاد کرده

<sup>۱۴</sup> . Linking



اید و آن را جداگانه کامپایل و ذخیره نموده اید و در یک برنامه کاربردی هم از آن استفاده برده اید. حالا هر گاه که الگوریتم سریع تری برای مرتب سازی یافتید، فقط کافی است فایل تابع را اصلاح و کامپایل کنید و دیگر نیازی نیست که برنامه اصلی را دست کاری نمایید.

چهارمین مزیت هم این است که می توانید یک بار یک تابع را کامپایل و ذخیره کنید و از آن پس در برنامه های مختلفی از همان تابع استفاده ببرید. وقتی شروع به نوشتن یک برنامه جدید می کنید، شاید برخی از توابع مورد نیاز را از قبل داشته باشید. بنابر این دیگر لازم نیست که آن توابع را دوباره نوشته و کامپایل کنید. این کار سرعت تولید نرم افزار را افزایش می دهد.

برای این منظور بهتر است با نحوه ساخت و بکارگیری فایل DLL<sup>۱۵</sup> (کتابخانه پیوند پویا) که در ادامه توضیح داده شده است، آشنا شویم.

---

<sup>۱۵</sup> . Dynamic Link Library

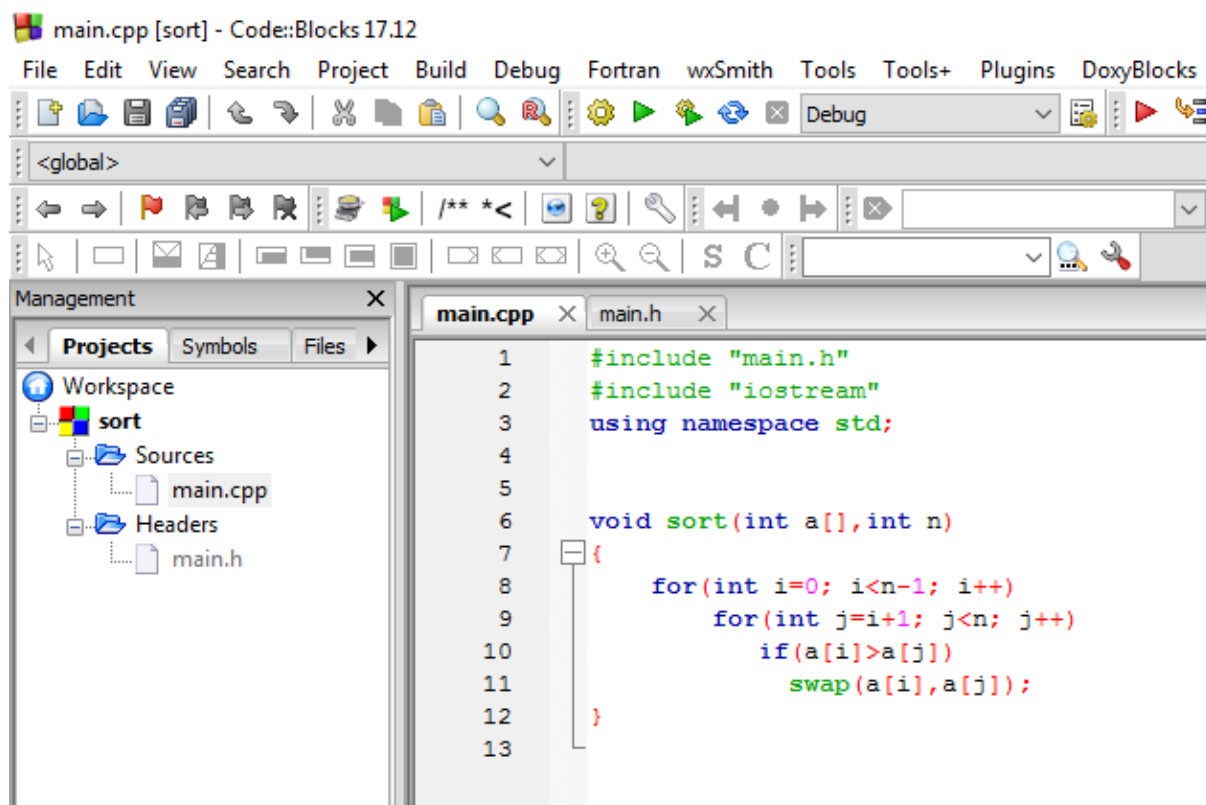
## ۲-۵-۱. نحوه ساخت فایل DLL

برای ساخت فایل DLL به روش زیر عمل می‌کنیم:

File → New Project → Dynamic Link Library

ادامه مراحل شبیه به ساخت پروژه جدید برنامه نویسی می‌باشد که آنها را ادامه می‌دهیم.  
Finish ←

در پنل Management مشاهده می‌کنیم که بر اساس نامی که برای فایل dll انتخاب کرده‌ایم (مثلا sort)، یک فایل dll ساخته شده است. که شامل دو زیرمجموعه به نام‌های Sources و Headers می‌باشد.



شکل ۲-۲۴: ساخت فایل DLL

در ادامه به سراغ فولدر Sources می‌رویم و فایل main.cpp را باز می‌کنیم سپس همه کدها به جز "#include 'main.h'" را پاک می‌کنیم و کدهای مربوط به تعریف تابع (مثلا تابع مرتب‌سازی) را می‌نویسیم. (می‌توانیم تعریف چندین تابع را در یک فایل dll داشته باشیم).

نکته : ممکن است تعریف یک تابع به فایل‌های کتابخانه‌ای خاص خودش احتیاج داشته باشد که باید آنها را نیز در ابتدای برنامه داشته باشیم.

در نهایت تابع موردنظر را کامپایل (compile) کرده و در صورتی که با موفقیت کامپایل شود، به این ترتیب فایل dll ما آماده استفاده در هر برنامه دیگری می‌باشد.

```

mingw32-g++.exe -shared -Wl,--output-def=bin\Debug\libsord.def -Wl,--out-implib=bin\Debug\libsord.a -Wl,--dll obj\Debug\main.o -o bin\Debug\sort.dll -luser32
Output file is bin\Debug\sort.dll with size 1.52 MB
Process terminated with status 0 (0 minute(s), 3 second(s))
0 error(s), 0 warning(s) (0 minute(s), 3 second(s))

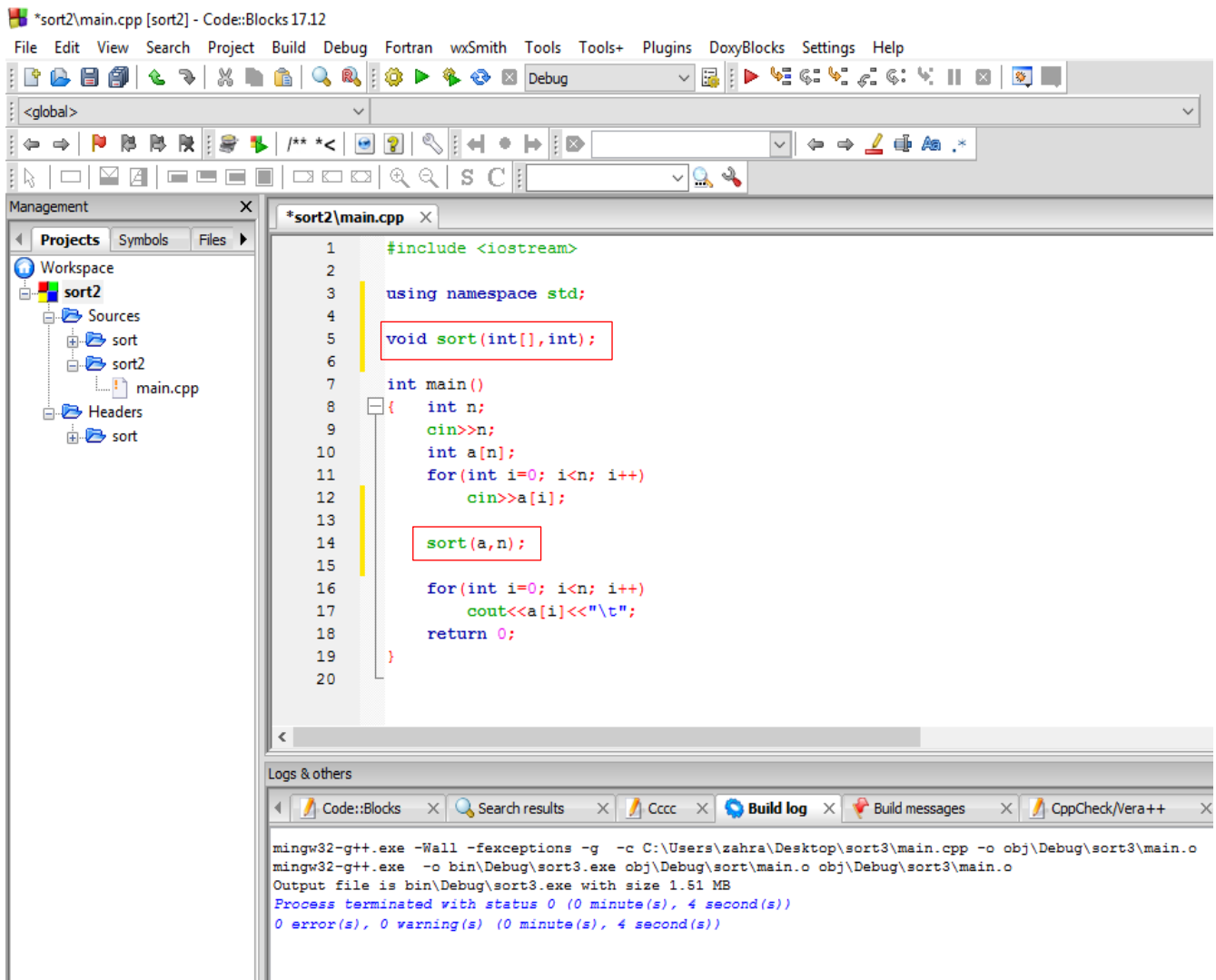
```

شکل ۲-۲۵: نتیجه کامپایل فایل DLL

## ۲-۵-۲. نحوه استفاده از فایل DLL در برنامه‌های دیگر

ابتدا پروژه برنامه نویسی موردنظرمان را ایجاد می‌کنیم (مثلا با نام sort2). سپس روی نام پروژه در پنل Management راست کلیک کرده و گزینه Add Files را انتخاب می‌کنیم. حالا باید آدرس فایل DLL که قبلا درست کرده‌ایم را بدهیم و دو فایل main.cpp و main.h را Open کنیم تا در زیرمجموعه پروژه اصلی ما قرار گیرند. حال در برنامه اصلی، تابع موردنظر را اعلان و فراخوانی کرده و سپس پروژه را اجرا (run) می‌کنیم.

نکته: ما هر تعداد فایل dll می‌توانیم به پروژه مان Add کنیم و هر تابع را در هر جایی که خواستیم (مثلا در برنامه اصلی یا در هر تابع دیگری (dll)) می‌توانیم فراخوانی کنیم.



شکل ۲-۲۶: مثالی از درج فایل DLL در پروژه دیگر

برنامه بالا به اندازه  $n$  عنصر از ورودی دریافت کرده و آن‌ها را در آرایه  $a$  قرار می‌دهد. سپس بوسیله تابع  $sort$ ، عناصر آرایه را مرتب می‌کند. لازم به ذکر است که تعریف تابع  $sort$  در یک فایل  $DLL$  به پروژه ضمیمه شده است. در نهایت نیز عناصر آرایه به صورت مرتب چاپ می‌شوند.

## ۲-۶. ساختن فایل کتابخانه‌ای (فایل سرآیند<sup>۱۶</sup>)

یک فایل کتابخانه‌ای، شامل توابعی است که در برنامه منبع بکار رفته اند و تعریف آن توابع در آن فایل کتابخانه‌ای یا فایل سرآیند آمده است. لذا وقتی در برنامه‌ای از توابع از پیش تعریف شده استفاده می‌کنیم باید ابتدای برنامه، کد مربوط ارجاع به کتابخانه را داشته باشیم. در غیر این صورت کامپایلر ارور می‌دهد.

در جدول زیر تعدادی از معروفترین کتابخانه‌ها را مشاهده می‌کنید:

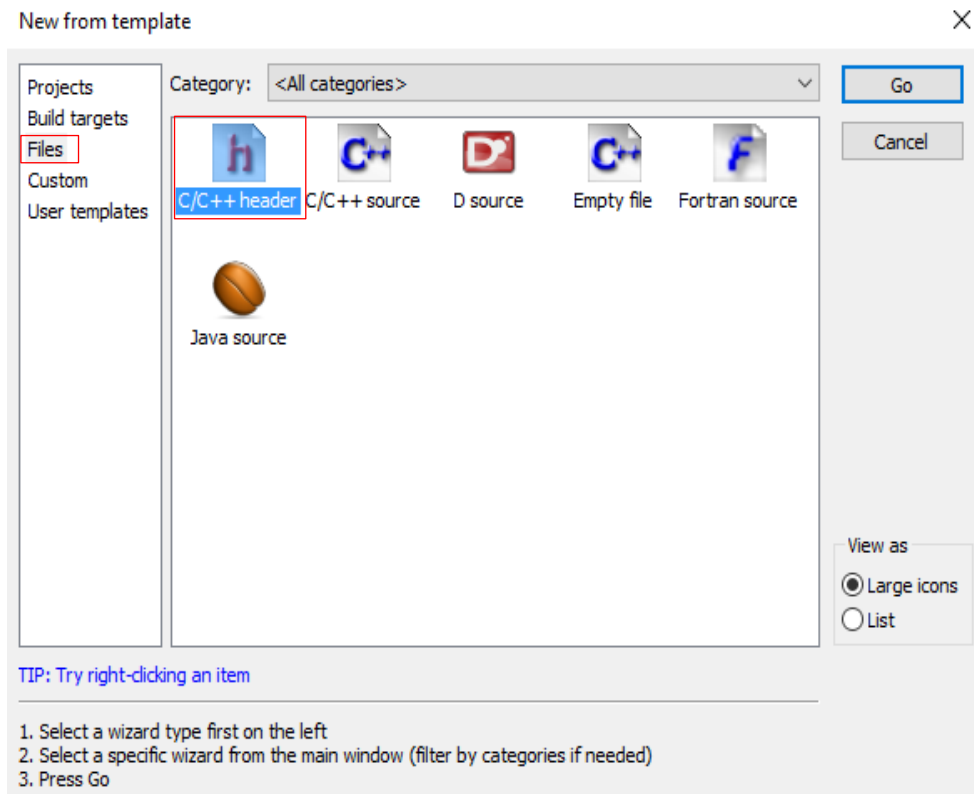
جدول ۲-۲: معرفی تعدادی از فایل‌های کتابخانه‌ای

<b>iostream</b>	تعریف توابع ورودی - خروجی
<b>stdio.h</b>	
<b>conio.h</b>	
<b>stdlib.h</b>	تعریف توابع کاربردی مثل <code>system("pause")</code>
<b>ioomanip</b>	تعریف توابع فرمت بندی مثل <code>setw()</code>
<b>cmath</b>	تعریف توابع ریاضی
<b>math.h</b>	

پیشنهاد می‌شود دانشجو در باره سایر فایل‌های کتابخانه‌ای نیز تحقیق کرده و نتایج خود را به کلاس ارائه دهد.

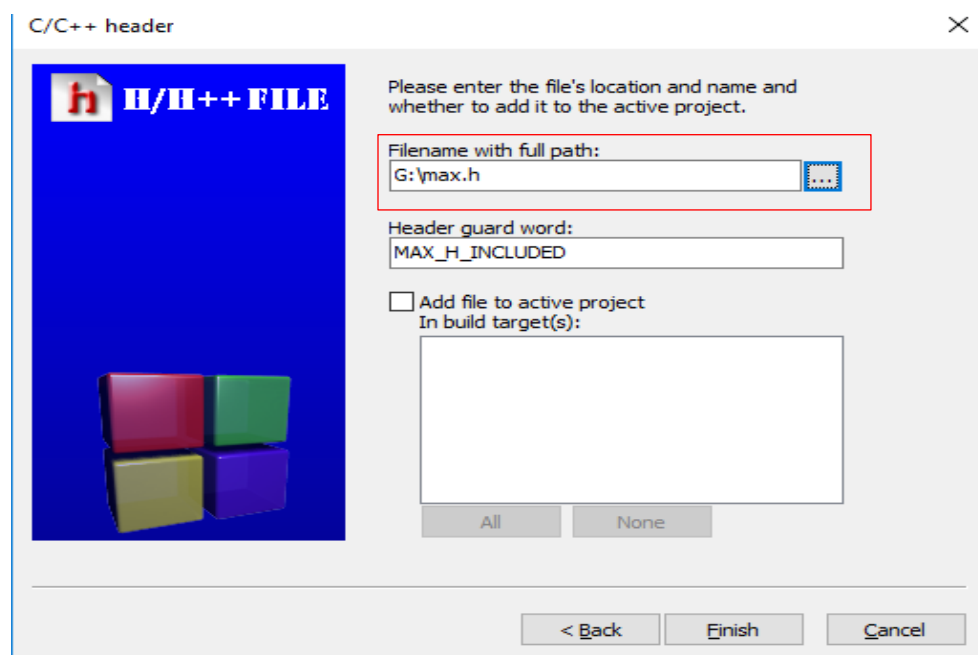
ما می‌توانیم به جز فایل‌های کتابخانه‌ای از پیش تعریف شده، خودمان نیز فایل کتابخانه‌ای بسازیم. برای این کار کافی است به هنگام ساختن پروژه جدید، از بخش Files بر روی گزینه C/C++ header کلیک کنیم. (مانند شکل پایین)

<sup>۱۶</sup>. Header file



شکل ۲-۲۷: ساختن فایل کتابخانه‌ای جدید

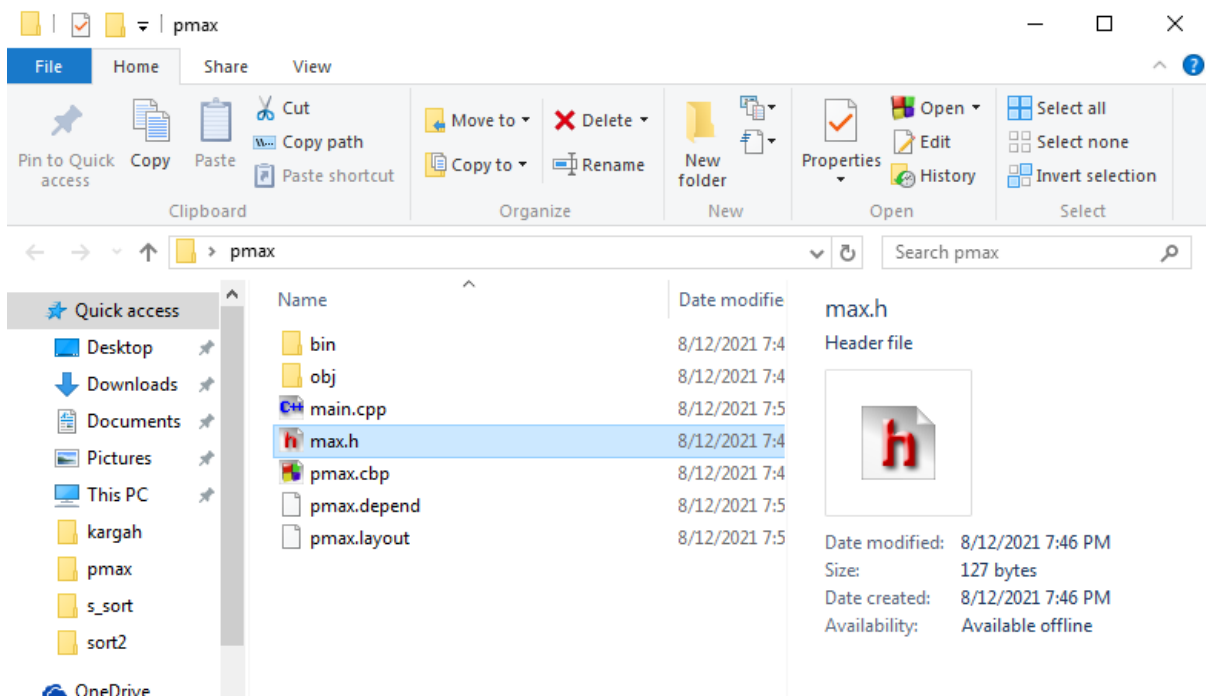
سپس در پنجره باز شده در فیلد **Filename with full path** ، نامی برای هدر فایل انتخاب نموده و محل ذخیره سازی را نیز مشخص می‌کنیم. (مانند شکل زیر) و در نهایت بر روی **Finish** کلیک می‌کنیم.



شکل ۲-۲۸: انتخاب نام و مسیر ذخیره‌سازی برای هدر فایل

بعد از ساختن فایل کتابخانه ای مورد نظر، برای استفاده از آن کافیت :

- هدر فایل مربوطه را (مثلا max.h) را به فولدر پروژه ضمیمه نماییم.



شکل ۲-۲۹: ضمیمه کردن هدر فایل به فولدر پروژه

- در خطوط ابتدایی برنامه نیز هشتگ مربوط به فایل سرآیند را اضافه نماییم. (مثلا "#include "max.h"). حتما توجه داشته باشید که نام هدر فایل باید در گیومه قرار بگیرد. گذاشتن گیومه به این معناست که کامپایلر در مسیر خود پروژه به دنبال هدر فایل مورد نظر باشد.

main.cpp	max.h
1 #include <iostream>	1 #ifndef MAX_H_INCLUDED
2 #include "max.h"	2 #define MAX_H_INCLUDED
3 using namespace std;	3
4	4 int max(int a, int b)
5 int main()	5 {
6 { int a=5, b=7;	6 return(a>b?a:b);
7 cout <<max(a,b);	7 }
8 return 0;	8
9 }	9 #endif // MAX_H_INCLUDED
10	10

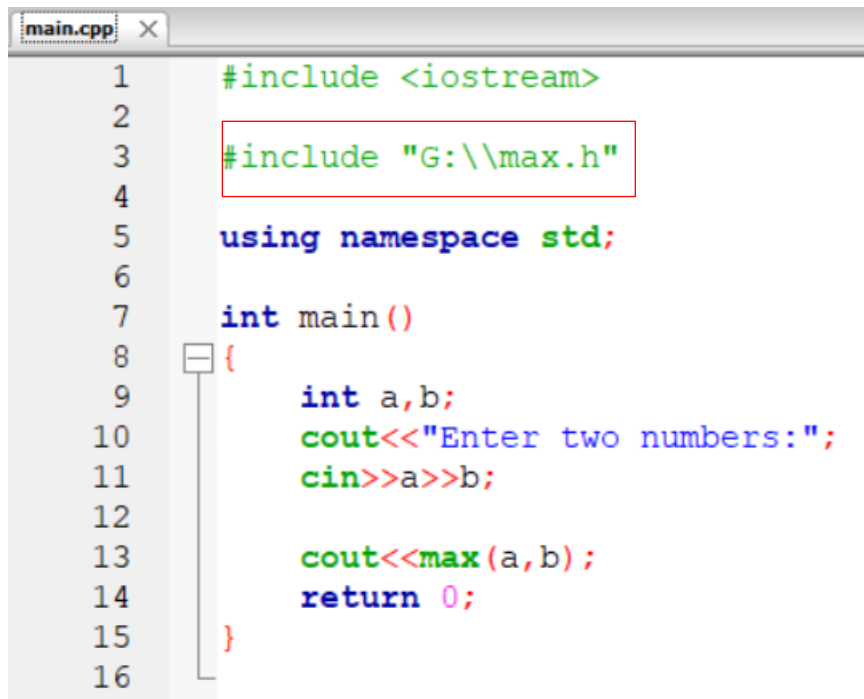
شکل ۲-۳۱: پروژه اصلی

شکل ۲-۳۰: هدر فایل یا فایل سرآیند



**نکته:** پیرو مطلب ذکر شده در صفحه قبل، لازم است بدانید، الزامی به قرار دادن فایل کتابخانه‌ای در فولدر پروژه اصلی نیست. در این شرایط کفایت آدرس مسیر ذخیره‌سازی هدر فایل (فایل کتابخانه‌ای) را به صورت کامل در گیومه قرار دهیم.

مانند تصویر زیر:



```

1  #include <iostream>
2
3  #include "G:\\max.h"
4
5  using namespace std;
6
7  int main()
8  {
9      int a,b;
10     cout<<"Enter two numbers:";
11     cin>>a>>b;
12
13     cout<<max(a,b) ;
14     return 0;
15 }
16

```

شکل ۲-۳۲: استفاده از هدر فایل با آدرس کامل در ابتدای برنامه

## ۲-۷. تفاوت‌های فایل کتابخانه‌ای و فایل DLL

### مکانیزم اجرا و سرعت آن‌ها:

از آنجایی که فایل کتابخانه‌ای به صورت کامل و یکجا به برنامه الحاق می‌شود در نتیجه در روند اجرای برنامه نیازی به مراجعه مداوم به هدر فایل و بازگشت به برنامه اصلی نیست لذا سرعت اجرای برنامه نیز افزایش می‌یابد اما سربار برنامه زیاد می‌شود.

مکانیزم اجرا در فایل DLL به اینگونه است که، هر بار که در برنامه فراخوانی‌ای صورت گیرد، به فایل DLL مراجعه می‌کند و سپس به برنامه برمی‌گردد و این عمل با توجه به برنامه، ممکن است بارها صورت گیرد در نتیجه این مراجعه به فایل DLL و بازگشت‌ها به برنامه اصلی، سرعت اجرا کاهش می‌یابد.

## تمرینات برنامه‌نویسی آخر فصل

۱- برنامه ای بنویسید که با استفاده از کاراکترهای فرمت‌بندی عبارت "Hello world" را چاپ نماید.

۲- برنامه ای بنویسید که مجموع اعداد زوج از ۱ تا  $n$  را محاسبه نماید.

۳- برنامه ای بنویسید که  $n$  عنصر از ورودی دریافت نموده سپس تعداد اعداد زوج و فرد را شمارش نماید.

۴- برنامه ای بنویسید که یک عدد از ورودی گرفته کامل بودن آن را بررسی نماید. (عدد کامل عددی است که مجموع مقسوم علیه های آن به جز خودش، با خود عدد برابر است. مثل  $6 = 1 + 2 + 3$ )

۵- برنامه ای بنویسید که جدول ضرب اعداد ۱ تا ۱۰ را چاپ نماید.

۶- برنامه ای بنویسید که یک رقم بین ۰ تا ۹ از ورودی دریافت نموده سپس معادل حرفی آن را چاپ نماید. (با استفاده از switch-case)

۷- برنامه ای بنویسید که اعداد رو به رو را چاپ نماید.

```

۱
۲   ۲
۳   ۳   ۳
۴   ۴   ۴   ۴

```

۸- برنامه ای بنویسید که دو عدد  $x$ ,  $y$  را از ورودی خوانده سپس بزرگترین مقسوم علیه بین آنها را محاسبه و چاپ نماید. (محاسبه ب.م.م)

۹- برنامه ای بنویسید که عددی را از ورودی دریافت کرده سپس قدرمطلق آن را محاسبه و چاپ نماید.

۱۰- برنامه ای بنویسید که عددی بین ۱ تا ۷ را از ورودی دریافت کرده و روز معادل آن را چاپ نماید. (با استفاده از دستور switch-case)

۱۱- برنامه ای بنویسید که  $n$  عدد از ورودی دریافت کرده سپس حاصل ضرب اعداد مثبت آن را محاسبه کند.

- ۱۲- برنامه‌ای بنویسید که عددی را از ورودی دریافت کرده سپس فاکتوریل آن را محاسبه نماید.  
( فاکتوریل یک عدد برابر است با حاصلضرب متوالی عدد یک تا خود آن عدد )
- ۱۳- برنامه‌ای بنویسید که یک عدد از ورودی دریافت کرده سپس میانگین اعداد از ۱ تا آن عدد را محاسبه کند.
- ۱۴- برنامه‌ای بنویسید که عددی را از ورودی دریافت کرده سپس آن را به مبنای ۲ ببرد.
- ۱۵- برنامه‌ای بنویسید که عددی را از ورودی دریافت کرده سپس مشخص کند آیا عدد اول است یا خیر و آن را با پیغام مناسبی نمایش دهد. (عدد اول عددی است که جز خودش و ۱ مقسوم علیه دیگری ندارد مثل ۳، ۷، ۱۱ و ...)
- ۱۶- برنامه‌ای بنویسید که  $n$  جمله اول سری فیبوناچی را نمایش دهد. (در سری فیبوناچی جمله اول و دوم برابر ۱ است و بقیه جمله‌ها از جمع دو جمله قبلی آن بدست می‌آید.  
۱ ۱ ۲ ۳ ۵ ۸ ۱۳ ...)
- ۱۷- برنامه‌ای بنویسید که جمله  $n$ ام سری فیبوناچی را نمایش دهد.
- ۱۸- برنامه‌ای بنویسید که فاکتوریل عدد صحیح  $n$  را محاسبه نماید. (با استفاده از تابع)
- ۱۹- برنامه‌ای بنویسید که مجموع اعداد طبیعی از ۱ تا  $n$  را محاسبه نماید. (با استفاده از تابع)
- ۲۰- برنامه‌ای بنویسید که عنصر  $x$  را در آرایه‌ای به طول  $n$  جستجو نماید.
- ۲۱- برنامه‌ای بنویسید که اسامی  $n$  دانشجو را از ورودی دریافت نموده سپس آنها را مرتب نماید.