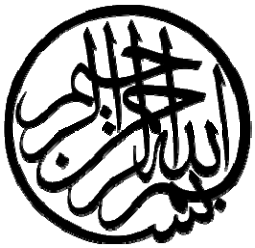
برنامه نویسی به زبان C
با ابزارهای تحت ویندوز

عبدالله آراسته



برنامه نویسی به زبان C

با ابزارهای تحت ویندوز

عبدالله آراسته

مجموعه کتابهای تکمیلی سمپاد

فهرست مطالب

مقدمه

به نام خداوند گردون سپهر
فروزنده‌ی ماه و ناهید و مهر

دانش و فناوری کامپیوتر، امروزه به یکی از پیچیده‌ترین و حساس‌ترین مرزهای گستره‌ی دانش بشر تبدیل شده است و شاید بتوان گفت هر جامعه‌ای در این زمینه پیشرو باشد، می‌تواند زمینه‌ی گسترش مرزهای دانش را در سایر علوم فراهم سازد و هر جامعه‌ای از این بخش غافل شود، پیشروی کلان در سایر علوم نیز برایش دشوار خواهد شد. بنابراین، یادگیری علوم این حوزه، در کنار سایر علوم، از ملزومات پیشرفت محسوب می‌شود و هر کس از این امر غافل شود، صرف نظر از رشته‌ای که به آن علاقه دارد، امکان پیشرفت خود را محدود کرده است.

اکنون که به یاری خداوند متعال نگارش این کتاب به پایان رسیده است، بر خود لازم می‌دانم از زحمات کلیه‌ی عزیزانی که مرا در طی نگارش کتاب یاری کردند، از جمله **همسرم**، که کلیه‌ی زحمات صفحه بندی بر عهده‌ی ایشان بود، **جناب آقای فرهاد مقیمی**، که در طراحی جلد و گرافیک بخش‌های مختلف کتاب زحمت زیادی کشیدند و سایر دوستان که با نظرات خود چراغ راه بودند، تشکر کنم. همچنین جا دارد تشکر ویژه‌ای از کلیه‌ی دوستان عزیز در **مرکز ملی پرورش استعدادهای درخشان و دانش پژوهان جوان** که در راه نشر این کتاب زحمت زیادی کشیدند، بنمایم. امیدوارم پایان این کار، آغاز راهی برای استعدادهای کشور و امیدهای آینده‌ی این مرزبوم باشد.

دانش آموخته‌ی سمپاد

عبدالله آراسته

بهار ۱۳۹۰

سخنی با دانش‌آموز

کتابی که پیش رو دارید از ده فصل تشکیل شده است. هر فصل حاوی مطالب مختلفی است که در فهرست مطالب می‌توانید آن‌ها را به‌سرعت بیابید. در کتاب از برخی کادرها استفاده شده که مهم‌ترین آن‌ها «توجه!» است که معمولاً توجه شما را به نکته‌ای مهم در مورد مطلب مورد بحث جلب می‌کند. کادرهای «لوح» برای مشخص نمودن آدرس یک فایل بر روی لوح فشرده همراه کتاب است و «وب» برای مشخص کردن آدرس یک سایت برای دانلود یک فایل و یا خواندن و دنبال کردن یک مبحث مربوط به درس.

کادر «سؤال» که کمتر آن را در طول کتاب می‌بینید، برای مطرح کردن سؤالاتی است که به نظر نگارنده، ذهن خواننده باید به سمت آن معطوف شود. کلیه‌ی شکل‌ها و کدها دارای شماره‌ای هستند که با شماره‌ی فصل شروع می‌شود، مثلاً کد ۵-۳ یعنی کد سوم از فصل پنجم و یا شکل ۷-۲ یعنی شکل دوم از فصل هفتم. کلیه‌ی کدهای نوشته شده در کتاب در لوح فشرده‌ی همراه کتاب موجود است. در اولین کد، در بخش «لوح» آدرس آن داده شده است، برای بقیه‌ی کدها نیز آدرسی مشابه وجود دارد که برای پرهیز از اطناب، در کنار سایر کدها درج نشده است. توصیه می‌شود برنامه‌های نوشته شده در متن درسی کتاب را خوب خوانده و درک کنید، سپس به ایجاد تغییر در بخش‌های مختلف آن اقدام کنید و نترسید! زیرا همواره یک کپی از برنامه‌ی اصلی در لوح فشرده‌ی کتاب در دسترس شماست! پس در ایجاد تغییرات و رفع اشکالات برنامه‌های جدید ایجاد شده **شجاع** و **کوشا** باشید.

در برخی فصول، بخش‌هایی با علامت (*) مشخص شده است که مفهوم اختیاری بودن دارد، یعنی دبیر درس مختار است این بخش‌ها را درس بدهد یا خیر و این تصمیم وابسته به ارزیابی وی از کلاس است. این موضوع در مورد فصل ۱۰ نیز صدق می‌کند (این فصل کلاً ستاره دار است). برای هر فصل تمارینی در سطوح مختلف در نظر گرفته شده که در کتاب دبیر موجود است و به صلاحدید وی به دانش‌آموزان به عنوان تمرین یا پروژه ابلاغ خواهد شد.

امید است پس از اتمام این کتاب، توانایی شما در حل مسایل کامپیوتری افزایش یابد و بتوانید در برخورد با آن‌ها، یک روال منطقی برای رسیدن به راه حل را طی کنید.

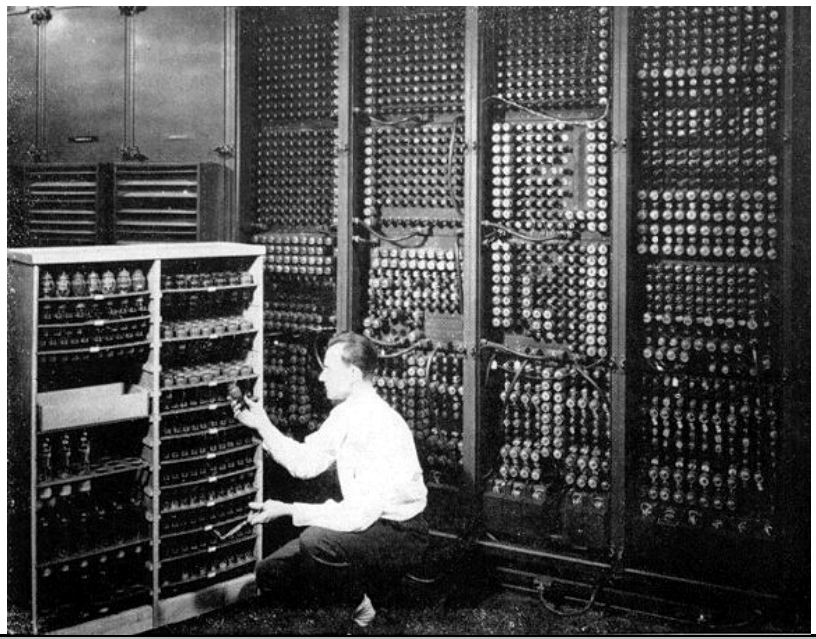
۱

فصل اول
با ما باشید

آشنایی با زبان‌های برنامه نویسی و مفاهیم اولیه

۱. تاریخچه‌ی توسعه‌ی کامپیوترها و زبان‌های برنامه‌نویسی

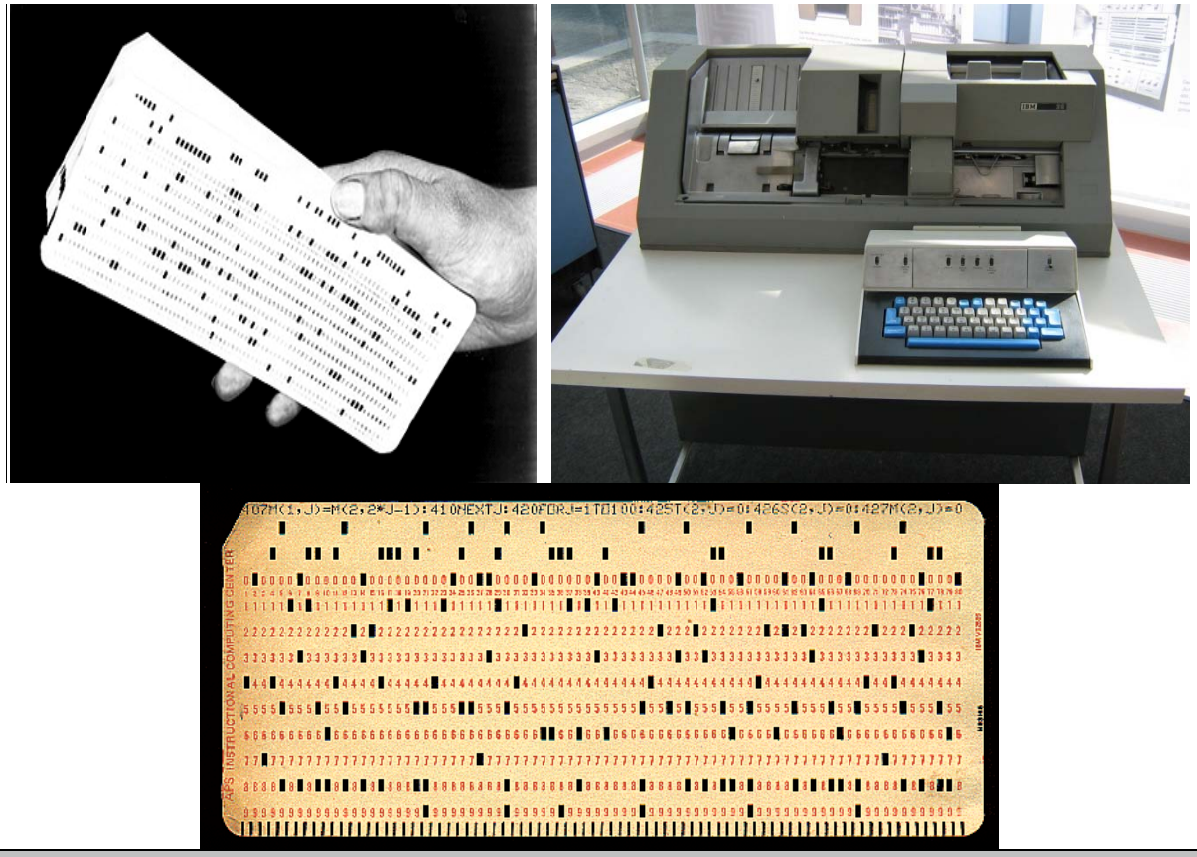
هیچ‌کس نمی‌داند اولین بار چه کسی به فکر ایجاد ماشینی برای انجام محاسبات افتاد، اما تلاش‌های مهمی که در این‌باره انجام شده در تاریخ ثبت شده است. اگر از وسایل مکانیکی ساخته شده برای محاسبات بگذریم، نخستین کامپیوتر الکترونیکی که ساخته شد، اِنیاک[1] بود (شکل ۱-۱). این کامپیوتر الکترونیکی قادر بود برنامه‌های محاسباتی مختلفی را اجرا کند، البته هدف اصلی طراحی آن محاسبه‌ی جدول تیر توپ‌ها و اسلحه‌های پرتابه‌ای ایالات متحده بود. اِنیاک که در سال ۱۹۴۶ توسط محققان دانشگاه پنسیلوانیا ساخته شد ۱۷۴۶۸ لامپ خلأ، ۷۲۰۰ دیود کریستالی، ۱۵۰۰ رله، ۷۰۰۰۰ مقاومت و ۱۰۰۰۰ خازن داشت و مساحتی حدود ۶۳ متر مربع را اشغال کرده و ۱۵۰ کیلو وات توان مصرف می‌کرد! سرعت این کامپیوتر حداکثر تا ۳۸۵ عمل ضرب در ثانیه بود! شاید با خواندن این آمار و ارقام، ضمن آشنا شدن با تاریخچه‌ی کامپیوترها، کمی بیشتر قدر کامپیوترتان را بدانید!



شکل ۱-۱: عکسی از ENIAC (یک تکنیسین در حال تعویض یکی از لامپ‌ها)

[1] Electronic Numerical Integrator And Computer

نحوه‌ی نوشتن برنامه برای اِنیاک، از طریق کارت‌های سوراخ دار[۲] بود که توسط دستگاه‌های کارت خوان IBM خوانده می‌شد (شکل ۱-۲). این کارت‌ها به گونه‌ای، حاوی کد دودویی[۳] دستوراتی بودند که اِنیاک باید اجرا می‌کرد.



شکل ۱-۲: عکس‌هایی از دستگاه کارت خوان IBM و کارت‌های سوراخ دار

پس از اِنیاک تا به امروز پیشرفت‌های فراوانی در توسعه‌ی کامپیوترها و زبان‌های برنامه‌نویسی اتفاق افتاده است، به گونه‌ای که تقریباً در هر خانه یک کامپیوتر شخصی (PC[۴]) وجود دارد و

[2] Card Punch
[3] Binary
[4] Personal Computer

در صورت داشتن اندکی دانش برنامه‌نویسی، هرکسی می‌تواند در خانه‌ی خود اقدام به برنامه‌نویسی و حل مسایل مختلف کند. اولین زبانی که پس از کدهای ماشین (که بسیار شبیه کدهای درج شده بر روی کارت‌های سوراخ دار بود) برای برنامه‌نویسی در نظر گرفته شد زبان اسمبلی[۵] بود. این زبان بسیار ساده‌تر از حفظ کردن کدهای ماشین بود و دستورات پایه‌ای برای انجام عملیات ریاضی و مدیریت حافظه را شامل می‌شد. به عنوان مثال به یک برنامه‌ی ضرب ساده به زبان اسمبلی توجه کنید:

```
mov ax, 14
mov bx, 5
mul bx
```

پس از انجام این دستورات نتیجه‌ی ضرب ۵ در ۱۴ در ax ذخیره می‌شود. همان‌طور که می‌بینید گرچه نسبت به کد ماشین و کارت سوراخ دار فهم این کد آسان‌تر است، اما همچنان برنامه‌نویسی با آن سخت است! طراحان زبان‌های برنامه‌نویسی نیز به این مشکل آگاه بودند و بنابراین اقدام به ایجاد زبان‌هایی کردند که به زبان‌های طبیعی انسان‌ها و زبان‌های محاوره‌ای نزدیک‌تر باشند. به عنوان مثال ضرب فوق که در زبان اسمبلی انجام شد، در چند زبان دیگر آمده است:

```
A = 14
B = 5      } BASIC
C = A*B
```

[5] Assembly

```
A := 14;
B := 5;      } PASCAL
C := A*B;
```

هرچه زبان‌های برنامه‌نویسی به گفتار انسان نزدیک‌تر باشد، اصطلاحاً به آن زبان سطح بالا می‌گویند. بر همین اساس زبان‌های برنامه‌نویسی را به سه دسته تقسیم می‌کنند:

(۱) زبان‌های سطح پایین: زبان‌هایی که مشابه زبان اسمبلی، به سخت افزار نزدیک‌تر بوده و معمولاً اعمال پایه‌ای ریاضی در آن‌ها تعریف شده و انجام عملیات پیچیده در آن‌ها از پیش تعریف نشده است.

(۲) زبان‌های سطح بالا: زبان‌هایی که عموماً به زبان محاوره‌ای نزدیک هستند و فهم آن‌ها با خواندن آن، معمولاً خیلی آسان‌تر از زبان‌های سطح پایین است و معمولاً هیچ ردپایی از سخت افزار در آن‌ها دیده نمی‌شود.

(۳) زبان‌های سطح میانی: زبان‌هایی که هم انعطاف‌پذیری زبان‌های سطح بالا و هم قابلیت ارتباط با سخت‌افزار زبان‌های سطح پایین را دارند.

در بین زبان‌های برنامه‌نویسی، زبان C جزء زبان‌های سطح میانی طبقه‌بندی می‌شود و به همین خاطر، هم در توسعه‌ی نرم‌افزارها و هم جهت ارتباط با سخت‌افزار به طور جدی از آن استفاده می‌شود. در بخش‌های بعدی بیشتر با خصوصیات این زبان برنامه‌نویسی آشنا خواهیم شد.

۲. لزوم فراگیری برنامه‌نویسی

در بخش قبلی در مورد خصوصیات کامپیوتر اِنیاک توضیح دادیم. سؤالی که در ذهن پیش می‌آید این است که چرا باید با صرف هزینه‌های گزاف، یک ماشین محاسبه‌گر ساخت و برای محاسبه به آن برنامه داد؟ پاسخ این است که اولاً محاسبات انسان همواره همراه با خطاست، خصوصاً اگر این محاسبات به دفعات زیاد و پشت سر هم تکرار شود. نکته‌ی مهم‌تر از آن این که بعضاً نتیجه‌ی این محاسبات باید در مواردی نظیر ساخت ابزار یا وسیله‌ای حساس استفاده شود و هرگونه خطا در این محاسبات ممکن است منجر به یک فاجعه شود! دیگر این که با فرض عدم وجود خطای محاسباتی، سرعت ماشین (که موجودی بی‌احساس و خستگی‌ناپذیر است) بسیار فراتر از سرعت انسان در محاسبه است، حتی اگر این ماشین اِنیاک باشد! این موضوع در بسیاری از کاربردهایِ امروزیِ کامپیوترها بیشتر مشهود است. شاید به جرأت بتوان گفت تعداد ضرب و جمع‌هایی که یک کامپیوتر در عرض چند ماه انجام می‌دهد از کل محاسبات عددی یک انسان در طول عمرش فراتر است و این در حالی است که در برخی کاربردها نظیر شبیه‌سازی‌های پیچیده یا یادگیری ماشین[۶] کامپیوترهای پرقدرت برای تولید خروجی، ساعت‌ها به محاسبه می‌پردازند که اگر قرار بود این محاسبات توسط انسان انجام شود معلوم نیست چندین سال طول می‌کشید، و شاید اصلاً دانشمندان چنین مسایلی با بار محاسباتی بالا را کنار می‌گذاشتند.

فایده‌ی یادگیری زبان برنامه‌نویسی و اصولاً هنر برنامه‌نویسی، تنها در سرعت بخشیدن به محاسبات برای رسیدن به جواب پرسش‌های علمی نیست، بلکه در حین تبدیل یک مسأله به یک برنامه که ورودی‌ها و خروجی‌های مشخص دارد و طراحی یک روش حل مسأله یا الگوریتم[۷] برای حل مسأله، فرد با تفکر منطقی و حل قدم به قدم حل مسایل آشنا می‌شود و

[6] Machine Learning
[7] Algorithm

این توانایی نه تنها در حل مسایل در کامپیوتر، که در تحلیل برخی از مسایل روزمره نیز به فرد کمک می‌کند. شاید به همین دلیل است که اغلب افرادی که برنامه‌نویسان خوبی هستند، در تحلیل بسیاری از مسایل، حتی مسایل اجتماعی، فلسفی و ...نسبت به افراد هم‌سطح خود که فاقد این مهارت هستند، قدرت بیشتری دارند.

یکی دیگر از محاسن برنامه‌نویسی به صورت عام و برای علاقه‌مندان به هر رشته‌ای از علم (مثل فیزیک، شیمی، ریاضیات، زیست و ...) این است که امروزه در بسیاری از رشته‌ها، مسایل محاسباتی وجود دارد که لزوماً ابزار یا نرم‌افزار آماده‌ای برای حل آن‌ها وجود ندارد و یا تمامی خواسته‌ها را در مورد یک مسأله‌ی خاص برآورده نمی‌کند. در این حالت چنانچه فرد یک دانش اولیه از برنامه‌نویسی داشته باشد، می‌تواند مسایل علمی خود را حل کند و دیگر احتیاج به سفارش برنامه به شرکت‌های برنامه نویسی و یا شخص دیگری نیست و در وقت و هزینه صرفه‌جویی فراوانی می‌شود.

۳. زبان C و تاریخچه‌ی آن

در بخش‌های قبلی راجع به کامپیوترها و زبان‌های برنامه‌نویسی صحبت شد و اشاره‌ی کوتاهی نیز به زبان C به عنوان زبانی سطح میانی (یعنی زبانی که هم قابلیت ارتباط با سطوح پایین سخت افزاری و هم قابلیت پیاده‌سازی برنامه‌های سطح بالا را دارد) شد، در این بخش نحوه‌ی به وجود آمدن زبان C و گسترش دامنه‌ی کاربرد آن در علوم و صنعت، با مروری بر تاریخچه‌ی آن مطرح خواهد شد. زبان برنامه نویسی C در آزمایشگاه بل، یکی از پر افتخارترین مراکز علمی دنیا متولد شد. خالق این زبان برنامه‌نویسی دنیس ریچی[۸] است که در

[8] Dennis Ritchie

این مرکز تحقیقاتی مشغول کار بر روی توسعه‌ی سیستم‌عامل‌ها و زبان‌های برنامه‌نویسی بود. خلق زبان C توسط دنیس ریچی و نقش آن در توسعه‌ی سیستم‌عامل یونیکس[9] در کنار کِن تامپسون[10]، او را در زمره‌ی افراد پیشرو در زمینه‌ی محاسبات جدید[11] قرار داد.

وب سایت آزمایشگاه بل:

http://www.bell-labs.com

وب سایت دنیس ریچی در آزمایشگاه بل:

http://cm.bell-labs.com/who/dmr

همچنین دنیس ریچی به همراه برایان کرنیگان[12] در سال ۱۹۷۸ اولین کتاب در مورد زبان C را که به صورت جامع به ابعاد مختلف این زبان برنامه‌نویسی پرداخته بود و در دسترس عموم قرار گرفت، نوشتند. در سال ۱۹۸۸، دومین ویرایش این کتاب که شامل آخرین تغییرات زبان C و شرح برخی کتابخانه‌های استاندارد[13] بود، توسط آن‌ها منتشر شد. این ویرایش کتاب تا سال ۲۰۱۰ به بیش از بیست زبان ترجمه شده و در بسیاری از مراکز آموزشی تدریس شده و می‌شود. متن این کتاب اندکی سنگین است، اما ایجازهای زیبایی در نوشتن برنامه‌های آن به زبان C به کار گرفته شده و مطالعه‌ی آن به تمامی علاقه‌مندان به برنامه‌نویسی توصیه می‌گردد.

[9] Unix
[10] Ken Thompson
[11] Modern computing
[12] Brian Kernighan
[13] Standard Libraries

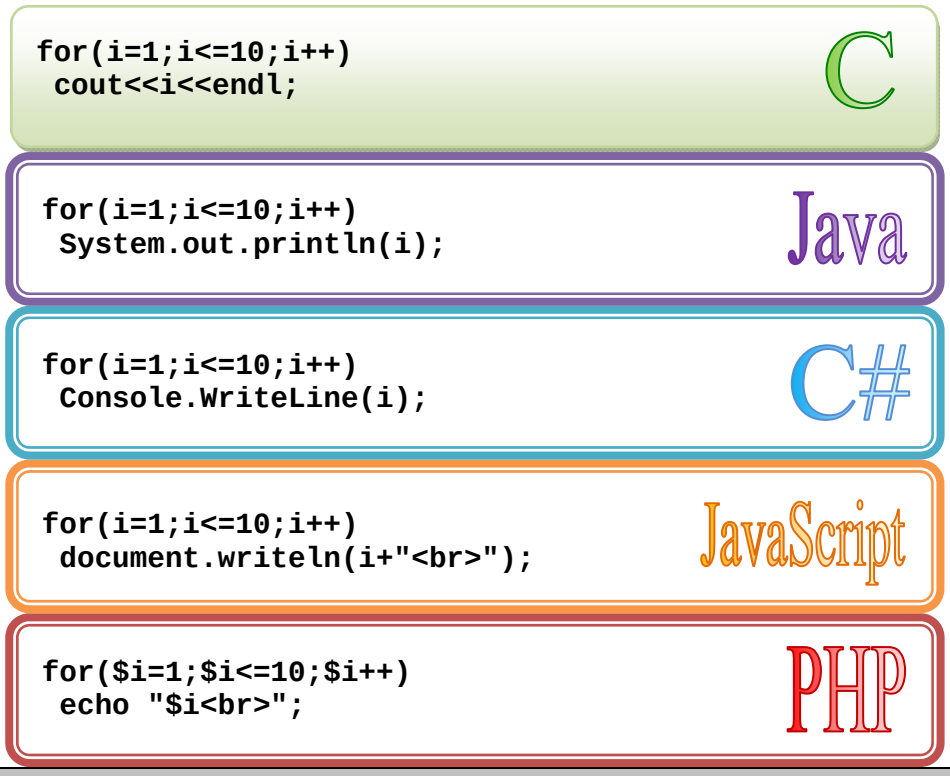
لوح

نسخه‌ی الکترونیکی کتاب The C programming language نوشته‌ی رابرت کرنیگان و دنیس ریچی در لوح فشرده‌ی همراه کتاب در آدرس زیر موجود است:

CDROM:\Resources\Books\K&R.pdf

پس از به وجود آمدن زبان C تا به امروز، هیچ‌گاه آموزش و به‌کارگیری زبان C به عنوان ابزاری در توسعه‌ی بسیاری از سیستم‌های سخت‌افزاری و نرم‌افزاری، قطع نشده و به هیچ‌وجه دستور زبان[۱۴] به کار رفته در این زبان برنامه‌نویسی منسوخ نشده است، بلکه برعکس زبان C در بسیاری از حوزه‌ها نفوذ کرده و امروز خانواده‌ی بزرگی از زبان‌های برنامه‌نویسی را زبان‌های «شبیه C[۱۵]» می‌نامند. از جمله‌ی آن‌ها می‌توان به Java، PHP، JavaScript و C# اشاره کرد. شباهت برخی از این زبان‌ها به C به قدری است که اگر فردی تنها زبان C را بلد باشد و نگاهی گذرا به متن برنامه‌ی نوشته شده با این زبان‌ها بیاندازد، متوجه نخواهد شد که برنامه با زبانی غیر از C نوشته شده است. به عنوان نمونه در شکل ۱-۳ قطعه کدهای مربوط به چاپ اعداد از ۱ تا ۱۰ را به زبان‌های C، Java، Java script، C# و PHP می‌بینید و می‌توانید شباهت بیش از اندازه‌ی چهار برنامه‌ی آخر را به برنامه‌ی اول که به زبان C نوشته شده درک کنید:

[14] Syntax
[15] C-Like



شکل ۱-۳: قطعه کدهای مربوط به چاپ اعداد از ۱ تا ۱۰ به زبان‌های C، Java، Java script، C# و PHP

زبان C به قدری در صنعت نفوذ کرده که برخی پردازنده‌های پرکاربرد (نظیر AVR ها) طوری ساخته شده‌اند که برنامه‌نویسی به زبان C، بهینه‌ترین حالت برای برنامه‌ریزی آن‌هاست. به عنوان نمونه می‌توان به نرم‌افزار Code Vision برای برنامه‌نویسی میکروکنترلرهای AVR اشاره کرد.

علاوه بر تمام موارد گفته شده، C به عنوان ابزار اصلی توسعه‌ی سیستم‌های عامل و ابزارهای متن باز[۱۶] به کار می‌رود و هسته[۱۷]ی سیستم‌عامل‌های لینوکس[۱۸] و نرم‌افزارهای کاربردی نصب شونده بر روی آن‌ها بر پایه‌ی زبان C است که باعث رویکرد میلیون‌ها علاقه‌مند به

[16] Open Source
[17] Kernel
[18] Linux

توسعه‌ی نرم‌افزارهای متن باز در دنیا به زبان C شده است. امروزه اهمیت نرم‌افزارهای متن باز و نقش اساسی آن‌ها در بین ابزارهای مورد استفاده‌ی کاربران بر کسی پوشیده نیست.

وب

در سایت زیر می‌توانید نرم افزار Code Vision و برخی برنامه‌هایی که با آن نوشته شده ملاحظه کرده و شباهت آن را با سایر برنامه‌های C استاندارد بسنجید:

http://www.codevision.be/

در انتهای این بخش این نکته نیز باید ذکر شود که زبان برنامه‌نویسی C++ هیچ تفاوت مبنایی با C ندارد و همان زبان C است که قابلیت‌های برنامه‌نویسی شئ‌گرا[19] به آن اضافه شده است. C++ توسط بیران استراستروپ[20] در سال ۱۹۷۹ در آزمایشگاه بل به عنوان پروژه‌ی بهبود زبان C توسعه یافت و در ابتدا C به انضمام کلاس‌ها[21] نامیده می‌شد و در سال ۱۹۸۳ به C++ تغییر نام داد. استراستروپ در مورد C++ کتابی نیز نوشته است که یکی از مراجع اصلی در این زمینه است. البته یکی دیگر از بهترین کتاب‌ها در این زمینه نیز نوشته‌ی هربرت شیلد[22] است که با مثال‌های متنوع به آموزش ساده و روان C++ پرداخته است. البته از ایرادهای این گونه کتاب‌ها می‌توان به حجم بالا و نداشتن مثال‌های مناسب برای دانش‌آموزان اشاره کرد که پیگیری و دنبال کردن آن را فقط محدود به افراد علاقه‌مند در این زمینه می‌کند.

[19] Object Oriented Programming
[20] Biorne Straustrup
[21] C with classes
[22] Herbert Schildt

وب

آدرس وب‌سایت شخصی استراستروپ:

http://www.straoustrup.com

http://www2.research.att.com/~bs/homepage.html

لوح

نسخه‌ی الکترونیکی کتاب C++ نوشته‌ی استراستروپ در لوح فشرده‌ی همـراه کتاب در آدرس زیر موجود است:

CDROM:\Resources\Books\bsC++.pdf

همچنین کتاب C++ نوشته‌ی هربرت شیلد نیز در آدرس زیر قرار دارد:

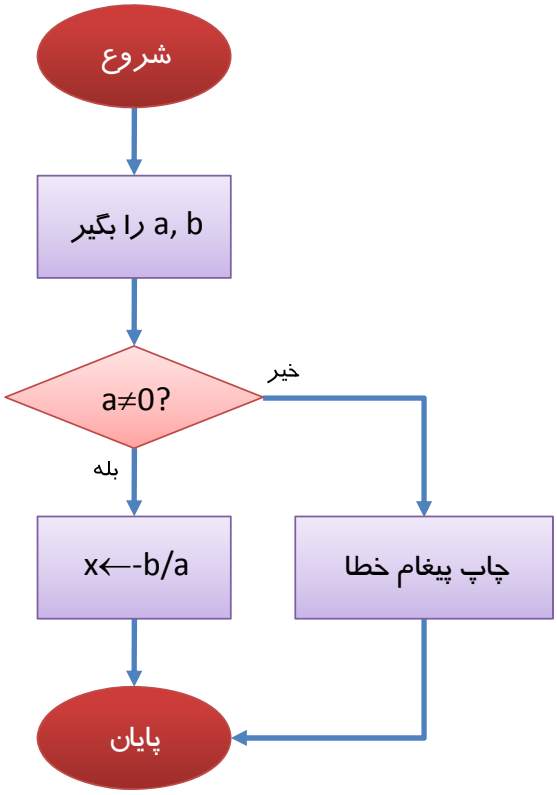
CDROM:\Resources\Books\hsC++.pdf

۴. گام‌های برنامه‌نویسی

یکی دیگر از مشکلات اساسی در نوشتن یک برنامه این است که عمدتاً افراد نمی‌دانند پس از برخورد با یک مسأله در دنیای واقعی، چطور آن را به شکلی در بیاورند که نوشتن برنامه‌ی آن آسان‌تر شود. بسیاری از افراد حتی مسایل ساده را که در حالت عادی به سادگی حل می‌کنند نمی‌توانند بر روی کامپیوتر پیاده‌سازی و اجرا نمایند. به عنوان مثال معادله‌ی درجه اول $ax+b=0$ را در نظر بگیرید، تقریباً هر دانش‌آموزان اول دبیرستانی جواب معادله که $x=-\frac{b}{a}(a\neq 0)$ را می‌داند، اما اگر بخواهد برنامه‌ای بنویسد که چنین معادله‌ای را حل کند، ممکن است نداند که از کجا باید شروع کند و یا اصولاً نداند گام‌های اصلی برای حل

چنین مسأله‌ای توسط کامپیوتر چیست؟ یک فرق اساسی حل مسایل توسط کامپیوتر با انسان این است که مسایلی که به کامپیوتر داده می‌شود با داده‌ها و دستورات بیان می‌شود، حال آن که مسایلی که برای انسان مطرح می‌شود با خواسته‌ها و قوانین مطرح می‌شود. به عنوان مثال همین مسأله‌ی معادله‌ی درجه اول را در نظر بگیرید، چنانچه یک معلم ریاضی بخواهد یک مسأله‌ی معادله‌ی درجه اول به دانش‌آموزان خود بدهد، به طور مثال آن را به صورت $5x+10=0$ مطرح می‌کند و دانش‌آموزان مطابق قوانینی که قبلاً به آن‌ها آموزش داده شده می‌دانند که تساوی در همه حال باید برقرار باشد و با کسر 10 از دو طرف تساوی و تقسیم دو طرف بر 5 به جواب $x=-2$ می‌رسند. اما در مورد حل این مسأله توسط کامپیوتر اوضاع کمی متفاوت است. کامپیوتر مانند انسان نمی‌تواند به یادگیری قواعد و حل مسایل بپردازد و باید قدم به قدم برای آن مشخص کرد که دقیقاً چه کار باید بکند؟ به عنوان مثال نمی‌توان به کامپیوتر تصویر یک معادله را نشان داد و انتظار حل آن را از آن داشت! بلکه باید به شیوه‌ی دیگری عمل کرد. ابتدا باید در هر مسأله مشخص شود ورودی‌های مسأله چیست؟ ورودی‌های مسأله همان‌هایی هستند که ما در ریاضی به آن‌ها معلومات می‌گوییم و قرار است از روی آن‌ها و با استفاده از قوانین مشخص، مجهولات را به دست آوریم. همچنین پس از به دست آوردن مجهولات، باید آن‌ها را به عنوان خروجی برنامه به کاربر اعلام کنیم. عناصر اصلی در نوشتن یک برنامه برای حل مسأله توسط کامپیوتر ورودی‌ها و خروجی‌ها هستند. به عنوان مثال در معادله‌ی درجه اول $ax+b=0$ ورودی‌ها a,b هستند، یعنی باید مقدار آن‌ها مشخص باشد تا بتوان جواب نهایی را محاسبه کرد. خروجی نیز همان x است. اما چه چیزی از روی ورودی، خروجی را می‌سازد؟ پاسخ این پرسش همان برنامه‌ای است که ما می‌نویسیم، یعنی ربط دادن خروجی به ورودی‌ها، وظیفه‌ی برنامه است. در برخی موارد مثل همین مسأله‌ی معادله‌ی درجه اول، ارتباط خروجی به ورودی به سادگی به دست می‌آید، اما

در برخی مسایل دیگر، برای تولید خروجی مناسب از روی ورودی‌ها، به تعمق و تفکر بیشتر احتیاج است. در بسیاری از موارد به دست آوردن گام‌هایی که باید به ترتیب توسط کامپیوتر اجرا شوند، مستلزم صرف ساعت‌ها وقت و بعضاً محاسبات زیاد است. برای روشن‌تر شدن موضوع، در شکل (۱-۴) نمودار گردشی[۲۳] حل مسأله‌ی معادله‌ی درجه اول آمده است:



شکل ۱-۴: نمودار گردشی حل مسأله‌ی معادله‌ی درجه اول

همان‌طور که ملاحظه می‌شود، این نمودار به خوبی نحوه‌ی گرفتن ورودی از کاربر و تولید خروجی را با شرط‌های مناسب $(a\neq0)$ نمایش می‌دهد. نمودار گردشی یک نوع از روش‌های

[23] Flow Chart

بازنمایی حل مسایل به روش کامپیوتری و گام به گام است که در فصل بعدی کتاب به تفصیل به آن پرداخته شده است. روش‌های دیگری نیز برای بازنمایی روش حل مسایل توسط کامپیوتر وجود دارد، نظیر توصیف گام به گام روش حل مسأله یا همان الگوریتم[۲۴] که یک نمونه از آن برای حل مسأله‌ی معادله‌ی درجه اول در شکل ۱-۵ آمده است.

(۱) a و b را بگیر

(۲) اگر $a\neq0$ است، x را برابر $-b/a$ قرار بده و به (۴) برو در غیر این صورت به (۳) برو

(۳) پیغام خطایی مبنی بر صفر بودن a چاپ کن و به (۵) برو

(۴) x را به عنوان جواب چاپ کن

(۵) پایان الگوریتم

شکل ۱-۵: توصیف گام به گام روش حل مسأله برای حل مسأله‌ی معادله‌ی درجه اول

یک نکته که در آزمون درستی یا نادرستی الگوریتمی که ارائه می‌دهیم به کار می‌آید، استفاده از مثال‌های مناسب است. به عنوان مثال اگر $a=10$ و $b=40$ باشد، جواب راه حل ما $-\frac{b}{a}$ یا -4 خواهد بود که از روش ریاضی و حل دستی نیز همین به دست می‌آید، بنابراین راه حل ما درست بوده است. البته شاید درستی الگوریتم در مورد مسأله‌ی معادله‌ی درجه اول بسیار ساده و واضح باشد، اما در مورد بسیاری از مسایل چنین نیست و این مثال زدن‌های عددی و آزمودن الگوریتم در طراحی و تصحیح آن، نقش به سزایی دارند. در پایان این بخش ذکر این نکته ضروری است که با طی تمامی این مراحل، نمی‌توان مطمئن بود که الگوریتم به دست

[24] Algorithm

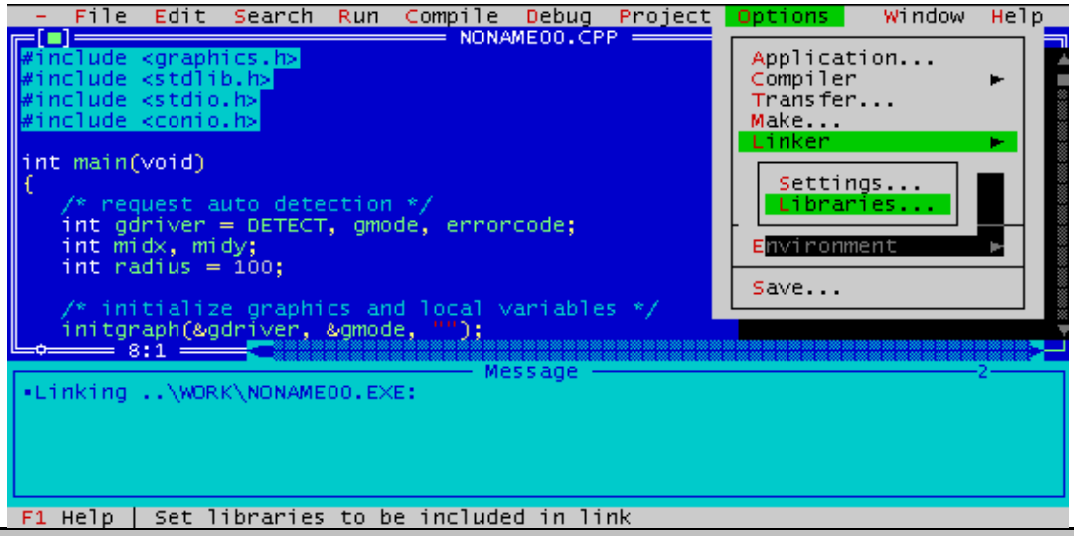
آمده صحیح باشد و تنها راه آزمودن آن، این است که برنامه را نوشته به ازای ورودی‌های مختلف اجرا کنیم تا ببینیم برنامه به درستی کار می‌کند یا خیر؟ اما این که برنامه را چطور بنویسیم و اجرا کنیم موضوع بخش بعدی است.

۵. آشنایی با انواع ویرایشگر[۲۵] ها و مقایسه‌ی آن‌ها

برای اینکه برنامه‌ی نوشته شده به هر زبان برنامه‌نویسی، به شکل قابل اجرا برای کامپیوتر، یعنی همان کد ماشین در بیاید، باید متن برنامه، به زبان ماشین ترجمه[۲۶] شود. برای همین منظور از برنامه‌هایی به اسم مترجم[۲۷] استفاده می‌شود. هر زبان برنامه‌نویسی مترجم مخصوص به خودش را دارد. البته ممکن است برای یک زبان نظیر C، چندین نسخه از مترجم‌های مختلف که تولید شرکت‌های مختلف برنامه‌نویسی است موجود باشد که معمولاً خود مترجم‌ها تفاوت عمده‌ای با هم ندارند. اما آنچه که در مورد ابزارهای مختلف برنامه‌نویسی متفاوت است، وجود ویرایشگرهای مختلف برای یک زبان برنامه نویسی است. گرچه لفظ ویرایشگر باید به برنامه‌هایی اطلاق شود که تنها به ویرایش متن برنامه می‌پردازند، اما امروزه به محیط‌های مجتمع توسعه[۲۸] برنامه‌ها نیز اصطلاحاً ویرایشگر می‌گویند. محیط‌های مجتمع توسعه یا همان IDE ها، ابزارهای مجتمع برای نوشتن، ویرایش، ترجمه و اشکال‌زدایی برنامه‌هایی که نوشته می‌شود هستند. برای هر زبان برنامه‌نویسی IDE های مختلفی وجود دارد و زبان C نیز از این قاعده مستثنی نیست. از جمله IDE های ساده‌ی زبان C می‌توان به Turbo C++ اشاره کرد (شکل ۱-۶) که یک برنامه‌ی قوی در زمان خودش محسوب می‌شد. خصوصیت این

[25] Editor
[26] Compile
[27] Compiler
[28] Integrated Development Environment (IDE)

IDE سادگی کار با آن در نوشتن برنامه‌هاست به طوری که یادگیری برنامه‌نویسی با آن به سادگی امکان‌پذیر است. اما اشکال عمده‌ی این IDE تحت DOS بودن و قدیمی بودن آن است که در کنار محاسن آن کار کردن با آن را خصوصاً به عنوان یک ابزار آموزشی سخت می‌کند، زیرا با سیستم‌عامل‌های جدید نظیر ویندوز ویستا و ویندوز 7 به درستی کار نمی‌کند. این مسأله در مورد نوشتن برنامه‌های گرافیکی بیشتر مشهود بوده و معمولاً خروجی مناسب تولید نمی‌شود و خطاهای سیستم‌عاملی پی‌درپی، عملاً نوشتن برنامه را امکان ناپذیر می‌کند.



شکل ۱-۶: محیط Turbo C++

یک راه حل که برای حل معضل چنین برنامه‌هایی پیشنهاد شده، استفاده از نرم افزار DOS Box است که کار آن شبیه‌سازی محیط DOS برای برنامه‌های قدیمی DOS نظیر Turbo C++ در محیط‌های جدید ویندوز است. اما این برنامه نیز کارکرد درست صد در صد نداشته و در بسیاری موارد اجرای آن با شکست مواجه شده است، مخصوصاً در مورد سیستم عامل ویندوز 7.

از دیگر محدودیت‌های Turbo C++ می‌توان به محدودیت استفاده از حافظه اشاره کرد که موقع کار با آرایه‌های بزرگ، برنامه‌نویس را دچار مشکل می‌کند. همچنین امکانات گرافیکی مثل استفاده از کل جدول رنگ[۲۹] در دسترس در ویندوز و یا بارگذاری و نمایش تصاویر یا استفاده از موس، در Turbo C++ به صورت پیش‌فرض وجود ندارد و تعداد رنگ‌ها در آن محدود به ۱۶ رنگ و یا با استفاده از دستورات خاص تا ۲۵۶ رنگ است، در حالی که تقریباً همه‌ی مانیتورها و کارت‌های گرافیکی روز، از حداقل ۱۶ میلیون رنگ پشتیبانی می‌کنند!

یک راه برداشتن این محدودیت‌ها استفاده از ابزارهای برنامه‌نویسی تحت ویندوز است، اما باید در تفسیر این جمله دقت زیادی کرد. بسیاری از برنامه‌نویسان این عبارت را به معنای برنامه‌نویسی تحت ***معماری ویندوز*** و با ابزارهایی نظیر Microsoft Visual Studio یا Borland C++ Builder و یا Borland Delphi می‌دانند. اگر چنین برداشتی از این جمله شود، باید متذکر شد شروع برنامه‌نویسی معمولاً باید با برنامه‌نویسی ترتیبی[۳۰] با اجرای خط به خط باشد. یادگیری برنامه‌نویسی ترتیبی خود مستلزم زمان زیادی است، حال اگر این مسأله با عنوان کردن معماری ویندوز و مفاهیمی نظیر رخداد[۳۱] ها، اشیاء، برنامه‌نویسی رویدادگرا[۳۲] و... ترکیب شود، باعث گیج شدن اغلب افرادی می‌شود که می‌خواهند برنامه‌نویسی را شروع کنند. در بسیاری از کتب برنامه‌نویسی با معماری ویندوز نیز فصول زیادی در ابتدای کتاب، به مطرح کردن برنامه‌نویسی ترتیبی، با رویکرد آموزش دستور زبان مربوطه اختصاص داده می‌شود. بنابراین استفاده از چنین ابزارهایی، گرچه محدودیت‌های گفته شده را برطرف می‌سازد، اما هدف اصلی که آموزش همگانی با کم‌ترین هزینه‌ی زمانی است را با تردید جدی رو به رو می‌کند.

[29] Palette
[30] Sequential
[31] Event
[32] Event Oriented programming

راه حلی که پیشنهاد می‌شود استفاده از یک ابزار بینابین است که هم تحت ویندوز اجرا شده و محدودیت‌های حافظه‌ای، گرافیکی و... برنامه‌های تحت DOS نظیر Turbo C++ را نداشته و هم این که فاصله‌ی زیادی با مدل برنامه‌نویسی ترتیبی محیط‌هایی نظیر Turbo C++ را نداشته باشد تا فراگیری آن دشوار نشود. در بین ابزارهای برنامه‌نویسی، یک IDE با نام Dev C++ وجود دارد که تقریباً تمامی خصوصیات ذکر شده را دارد. این ابزار برنامه‌نویسی بر روی جدیدترین سیستم‌عامل‌ها نیز بدون مشکل نصب می‌شود.

اصل این برنامه به صورت متن باز و تحت لینوکس است که نسخه‌هایی از آن برای ویندوز نیز نوشته شده و به راحتی و به صورت مجانی از اینترنت قابل دریافت و نصب است. متن باز بودن این برنامه باعث شده است که به روز[۳۳] شدن آن سریع بوده و با سیستم عامل‌های جدید سازگار باشد، همچنین افرادی که به برنامه‌نویسی تحت سیستم عامل لینوکس علاقه دارند می‌توانند نسخه‌های تحت لینوکس این نرم افزار را دریافت و نصب کنند.

وب

آدرس وب‌سایت اصلی نرم‌افزار Dev C++:

http://www.bloodshed.net/dev/devcpp.html

البته با جستجو در گوگل نیز می‌توان سایت‌های بسیار زیادی برای دانلود آخرین نسخه‌ی این نرم افزار پیدا کرد.

[33] Update

لوح

یک نسخه از فایل قابل نصب نرم افزار Dev C++ در لوح فشـرده و در آدرس زیر قرار دارد:

CDROM:\IDE\ Dev C++

همچنین راهنمای نحوه‌ی نصب و مراحل آن در آدرس زیر موجود است:

CDROM:\Resources\Others\InstallDevcpp.pdf

البته IDE های مشابه دیگری نظیر Code Blocks نیز وجود دارد که تفاوت‌هایی جزئی با Dev C++ دارد و معمولاً انتخاب بین IDE هایی نظیر Dev C++ و Code Blocks بیشتر سلیقه‌ای است و تفاوت‌های بنیادینی بین آن‌ها وجود ندارد. در این کتاب مبنا Dev C++ است، اما با فراگیری Dev C++ و با کمی تلاش بیشتر، می‌توان به راحتی با Code Blocks نیز کار کرد.

وب

آدرس وب‌سایت اصلی نرم‌افزار Code Blocks :

http://www.codeblocks.org/downloads/26

لوح

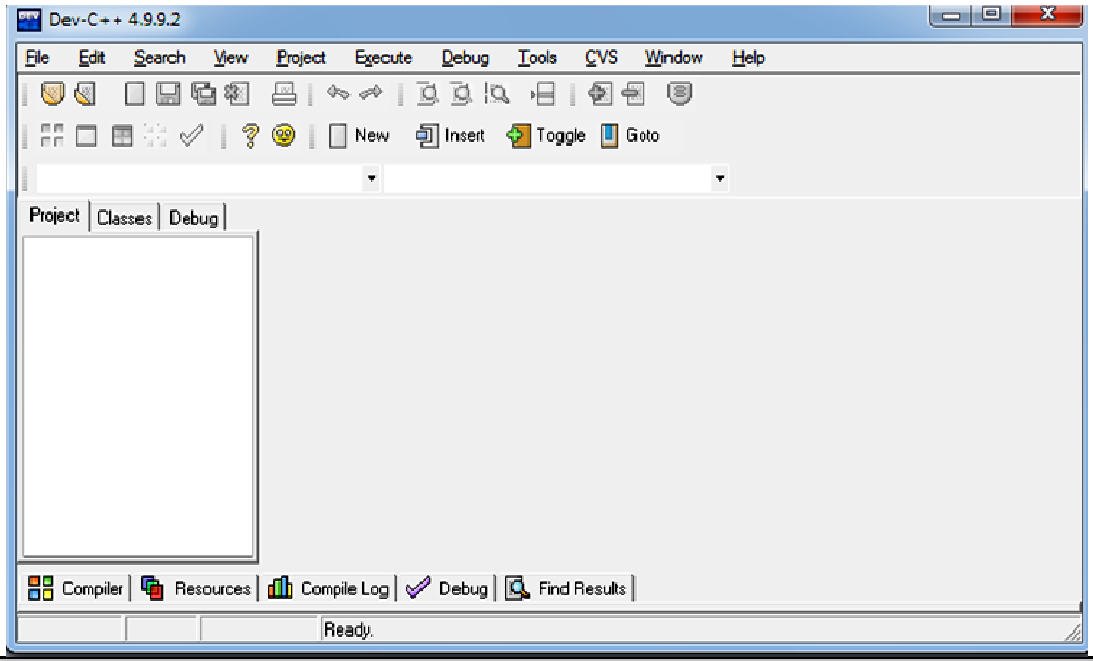
یک نسخه از فایل قابل نصـب نـرم‌افـزار Code blocks در لـوح فشـرده و در آدرس زیر قرار گرفته است.

CDROM:\IDE\Code Blocks

در ادامه‌ی بحث و در بخش بعدی، به مرور امکانات محیط Dev C++، پرداخته خواهد شد.

۶. آشنایی با محیط برنامه‌نویسی Dev C++

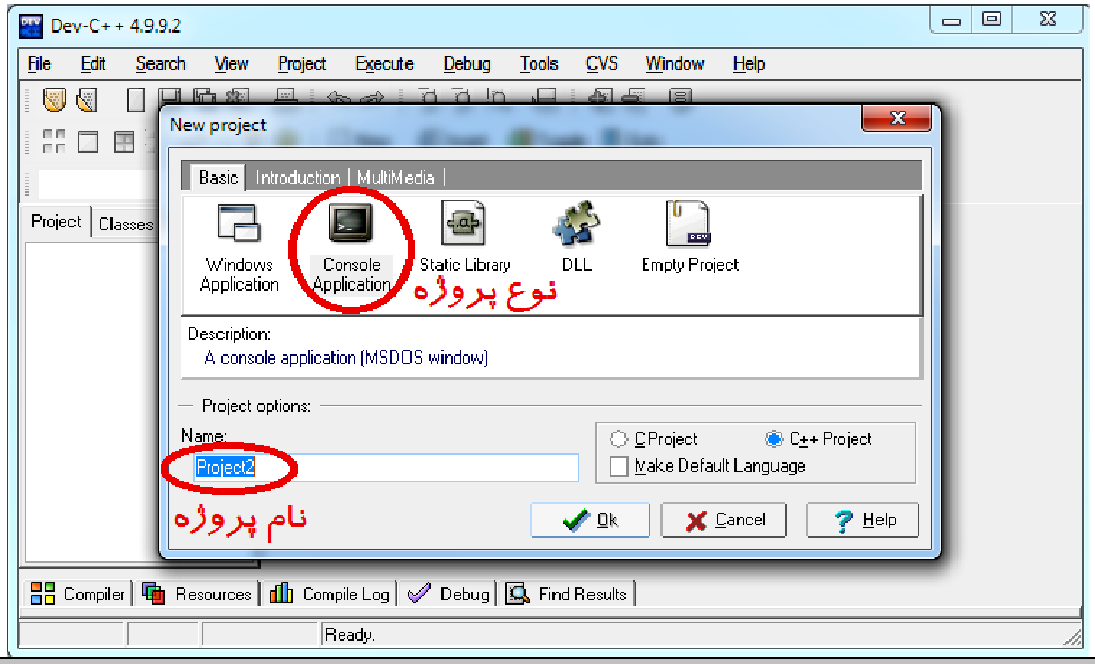
پس از نصب و اجرای برنامه (که راهنمای مرحله به مرحله‌ی آن در لوح فشرده‌ی همراه کتاب آمده) به محیط Dev C++ برخورد خواهید کرد (شکل ۱-۷).



شکل ۱-۷: محیط Dev C++

در همین محیط است که باید برنامه‌ها نوشته‌شده، اجرا شوند و در صورت لزوم مورد اشکال‌زدایی قرار بگیرند. برای ایجاد یک برنامه‌ی ساده، کافی است از منوها File→New→Source File انتخاب شود و یا از کلیدهای ترکیبی Ctrl+N استفاده شود. سپس می‌توان در فایل ایجاد شده برنامه‌ی مورد نظر را تایپ کرده و اجرا نمود. می‌توان در یک زمان چند فایل برنامه‌ی باز در Dev C++ داشت و هر کدام را که مورد نظر است

انتخاب و سپس ویرایش یا اجرا نمود. همچنین می‌توان از منوها File→New→Project را انتخاب کرد که در این صورت پنجره‌ای نمایش داده می‌شود (شکل ۱-۸) که از بین انواع پروژه‌ها باید یک کدام انتخاب شود و اسم آن نیز در قسمت مشخص شده وارد شود.

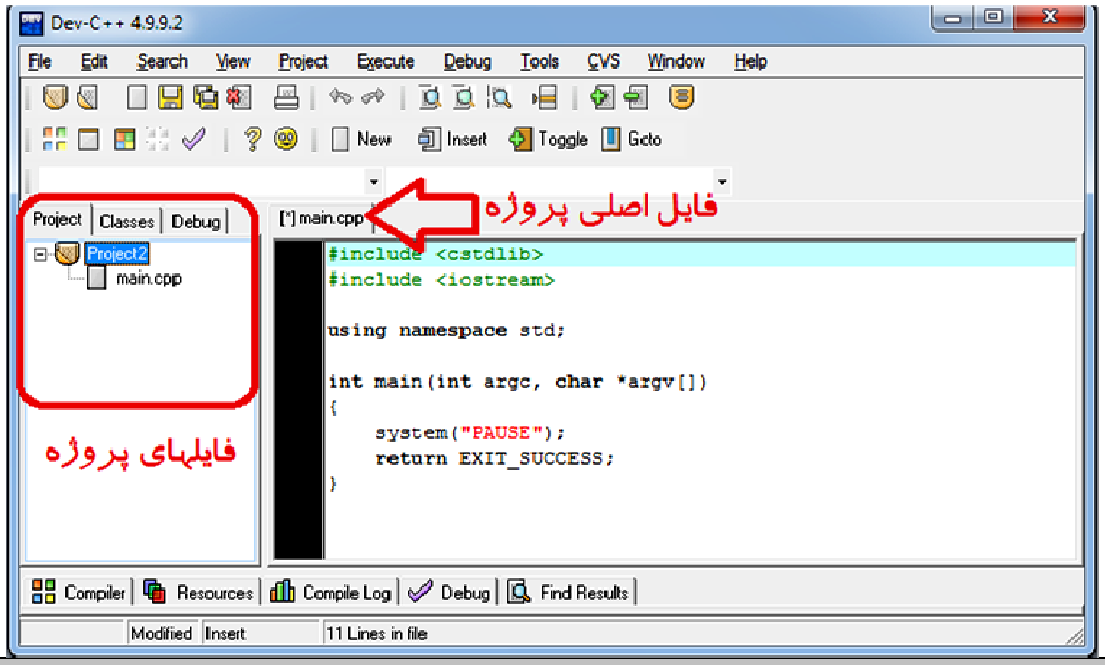


شکل ۱-۸: ایجاد یک پروژه‌ی جدید در Dev C++

سپس پروژه‌ی مورد نظر ایجاد شده و باید آن را در جایی ذخیره کرد. توصیه می‌شود همواره برای هر پروژه یک فولدر جدا ساخته و همه‌ی فایل‌های مربوط به آن پروژه را در آن فولدر ذخیره کنید.

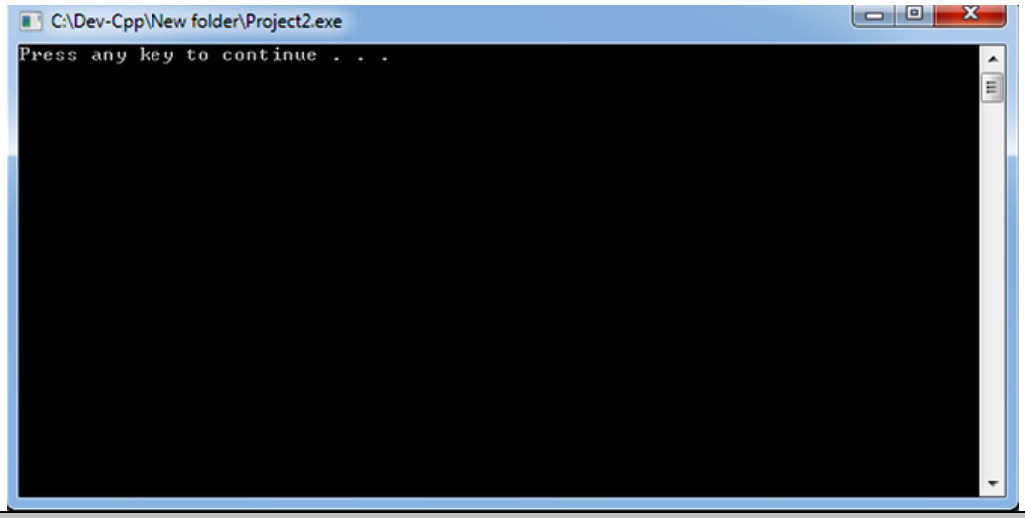
پس از اتمام ایجاد پروژه، نمودار درختی پروژه و فایل‌های مربوط به آن در قسمت سمت چپ برنامه‌ی Dev C++ نمایش داده می‌شود (شکل ۱-۹) و فایل اصلی پروژه که معمولاً با نام main.cpp ساخته می‌شود، در بخش نمایش فایل‌ها نمایش داده می‌شود. چنانچه بخواهید این

پروژه اجرا شود باید کلید F9 را زده و یا از منوها Execute→Compile & Run را انتخاب کنید و پس از انجام این کار پنجره‌ای باز می‌شود که از شما می‌خواهد برنامه‌ی اصلی یا همان main.cpp را ذخیره کنید.



شکل ۱-۹: نمودار درختی فایل‌های پروژه و فایل اصلی پروژه

چنانچه main.cpp را ذخیره کرده و اگر در ابتدا console application را به عنوان نوع پروژه انتخاب کرده باشید (در پنجره‌ای که در شکل ۱-۸ آمده) پروژه اجرا شده و خروجی آن نمایش داده می‌شود (شکل ۱-۱۰).



شکل ۱-۱۰: خروجی یک پروژه‌ی Console Application ساده

توجه!

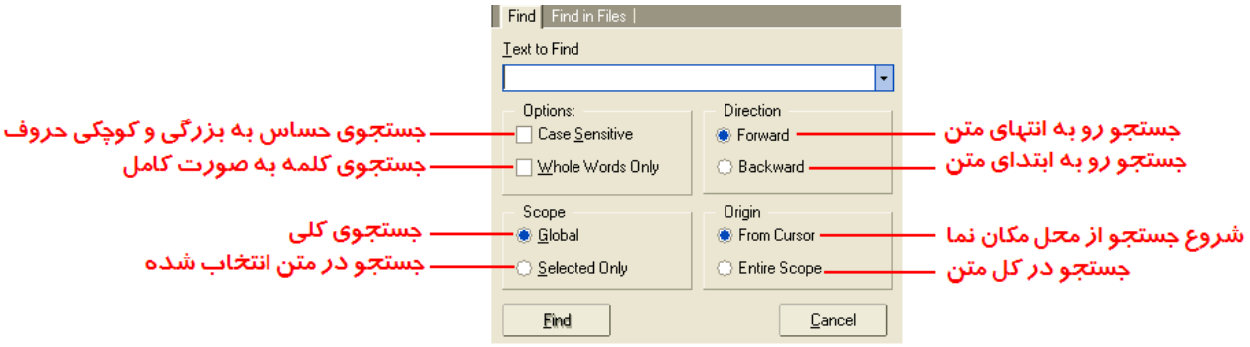
چنانچه پروژه‌ای باز باشد، تا زمانی که آن پروژه باز باشد، هر فایل جدیدی که ایجاد شود و یا هر فایلی که باز است انتخاب شود و سپس برای اجرای آن دکمه‌ی F9 زده شود، تنها همان پروژه اجرا می‌شود. برای جلوگیری از این کار باید از منوها File→Close Project انتخاب شود تا پروژه‌ی فعلی بسته شود. این اتفاق به خاطر آن است که پروژه نسبت به سایر فایل‌ها برای اجرا اولویت دارد. این قضیه در رابطه با فایل‌های ساده‌ی برنامه برقرار نیست، یعنی اگر پروژه‌ای باز نباشد، هر فایلی که انتخاب شده و در حال نمایش است، اجرا می‌شود.

لازم به تذکر است برای عملیاتی که زیاد تکرار می‌شود، نظیر اجرا کردن برنامه و یا ایجاد برنامه‌های جدید، دکمه‌هایی در نوار ابزار برنامه گنجانده شده. در شکل ۱-۱۱ برخی از این دکمه‌ها نمایش داده شده‌اند.

شکل ۱-۱۱: برخی دکمه‌های پرکاربرد در نوار ابزار برنامه‌ی Dev C++

از دیگر امکانات بسیار مفید در برنامه‌نویسی، جستجو و جستجو و جایگزینی هستند. فرض کنید در فایلی به دنبال کلمه‌ی int هستید، در این صورت کافی است تا از منوها Search→Find را انتخاب کرده و یا از کلیدهای ترکیبی Ctrl+F استفاده کنید تا کادر شکل ۱-۱۲ ظاهر شود. در این شکل توضیحات کافی در مورد گزینه‌های پر کاربرد داده شده است. برای جستجوی مورد بعدی، کافیست دکمه‌ی F3 زده شود و یا از منوها Search→Search Again و یا از نوار ابزار دکمه مربوطه انتخاب شود. عملیات جستجو و جایگزینی نیز مشابه جستجو است با این تفاوت که پس از یافتن عبارتی، آن را با عبارت دیگری جایگزین می‌کند.

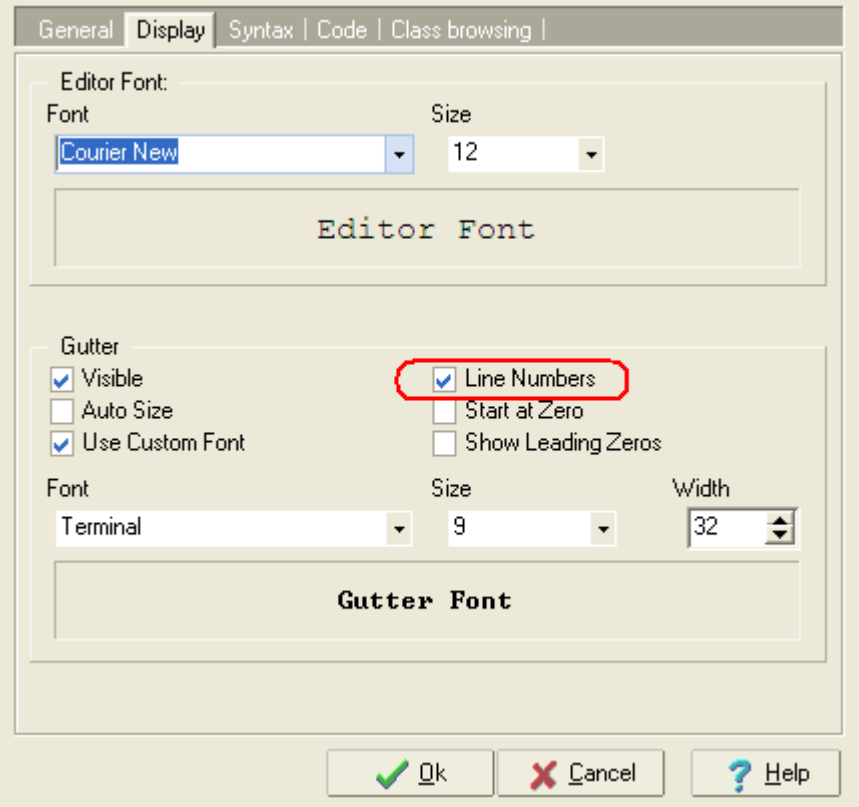
از امکانات مفید دیگر منوی Search، Goto line است که با Ctrl+G نیز در دسترس است و زمانی که خطوط برنامه زیاد باشد مفید است و می‌توان با استفاده از آن به خطی خاص از برنامه رفت.



شکل ۱-۱۲: کادر تبادلی جستجو در Dev C++

چنانچه بخواهید شماره‌ی خطوط برنامه نمایش داده شود، باید از منوها Tools→Editor Options را انتخاب کرده و از پنجره‌ای که ظاهر می‌شود به قسمت Display رفته و جلوی گزینه‌ی Line Numbers، تیک بزنید (شکل ۱-۱۳). همچنین در همین قسمت می‌توانید نوع قلم و سایز آن را که برای نمایش و ویرایش برنامه‌ها استفاده می‌شود، تعیین کنید.

برای مشخص کردن نوع رنگ و حالتِ قلمِ نوشتارهایِ متفاوتِ موجود در برنامه نیز می‌توانید پس از انتخاب Tools→Editor Options، به قسمت Syntax رفته و هرکدام از رنگ‌ها را که می‌خواهید عوض کنید و یا از تنظیمات از پیش آماده شده استفاده کنید (شکل ۱-۱۴). برای آن که آنچه در کتاب می‌بینید با محیط Dev C++ که در آن تایپ می‌کنید هماهنگ باشد، تنها تغییری که نسبت به حالت پیش‌فرض باید بدهید تغییر Forground مربوط به کاراکتر به قرمز است (شکل ۱-۱۴).



شکل ۱-۱۳: برخی تنظیمات ویرایشگر متن Dev C++ و نحوه نمایش شماره خطوط

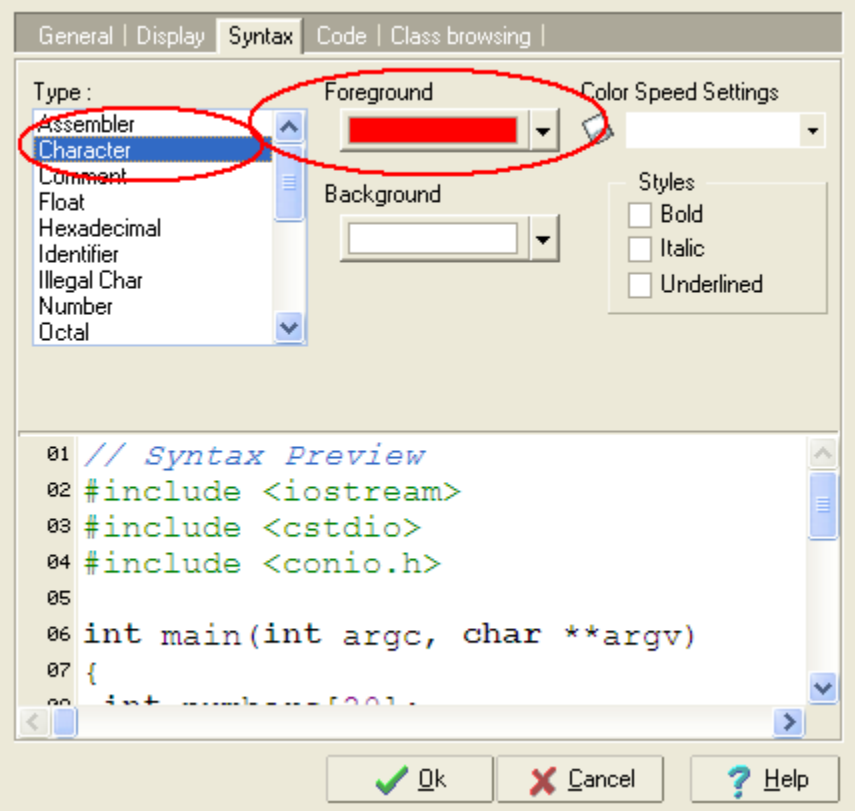
آن‌چه در این بخش گفته شد جهت آشنایی مختصر و سریع با محیط Dev C++ بود و برای مطالعه‌ی بیشتر می‌توانید به منابع دیگر رجوع کنید.

وب

در آدرس‌های زیر، نکاتی پیرامون Dev C++ آمده است:

http://www.bloodshed.net/dev/doc/index.html

http://sourceforge.net/projects/dev-cpp/forums/forum/128327



شکل ۱-۱۴: برخی تنظیمات ویرایشگر متن Dev C++ و نحوه تغییر رنگ متن نمایش داده

شاید بهتر باشد در آغاز فصل ۳ که برنامه‌نویسی به صورت جدی آغاز می‌شود، این بخش را برای آمادگی هر چه بیشتر در برنامه‌نویسی با Dev C++، مجدداً مرور نمایید.