



الفبای C++

برگه از هرعت الپیاره کامپیونر

مؤلف:

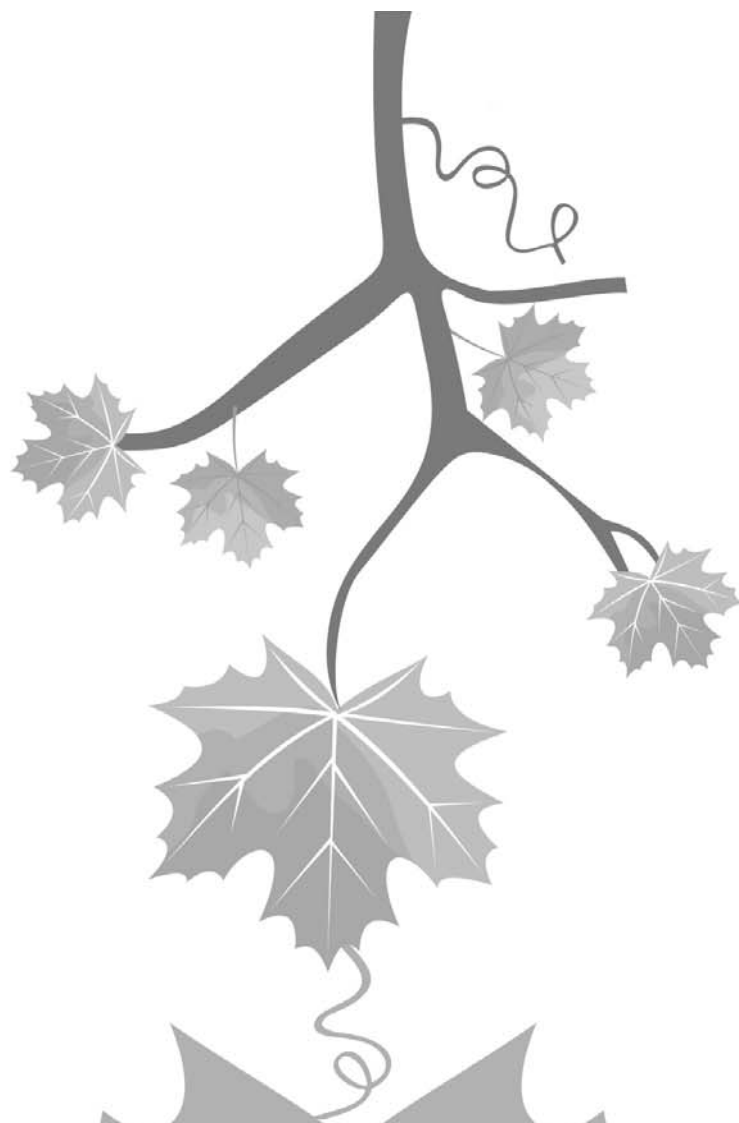
علی امینیان



انتشارات خوشخون

تقدیم بہ:
پدرم بہ استواری کوه
مادرم بہ زلالی چشمہ
و ہمسرم بہ صمیمیت باران





درخت المپیاد درختی است که توسط
انتشارات خوشخوان کاشته شده و هر یک
از کتاب های این پروژه برگگی از آن است.
وظیفه ما نگهداری و آبیاری این درخت است. امیدواریم
با عنایات حضرت حق این درخت، تنومند شده
وبه بار واقعی بنشیند. فراموش نکنید که بار و میوه ی

این درخت شما
عزیزان می باشید.

التماس دعا



پروژه‌ی درخت المپیاد

اعتقاد بر این است که شروع فعالیت‌های المپیاد به صورت حرفه‌ای، باید از ابتدای دوره‌ی دبیرستان شروع شود. اکثر المپیادهای علمی در زمستان سال سوم دبیرستان تعیین تکلیف می‌شوند. بنابراین از شروع دبیرستان تا اواسط سال سوم حدوداً ۸ ترم تحصیلی می‌شود (با احتساب فصل و ترم تابستان) که لازم است برنامه‌ریزی دقیقی برای این چند ترم انجام شود.

انتشارات خوشخوان این برنامه‌ریزی را در قالب پروژه‌ی درخت المپیاد انجام داده است که هر شاخه از درخت، مبحثی از آن المپیاد و هر برگ از آن شاخه شماره‌ای از آن مبحث می‌باشد.

به عنوان مثال اپتیک (۱) کتابی است که در یک ترم تحصیلی در یک کلاس ممتاز می‌توان برای داوطلبان المپیاد فیزیک تدریس کرد.

با عنایات حضرت حق و با کمک تنی چند از همکاران گرامی کتب مربوط به این درخت در هر رشته‌ای از المپیاد معرفی خواهد شد.

منتظر پیشنهادات و نظرات شما سروران هستیم.

گروه المپیاد

انتشارات خوشخوان



مسابقه‌ها، کنکورها و المپیادهای علمی همایش‌هایی هستند که کم و بیش در سرتاسر دنیای پهناور به صورت داخلی و بین‌المللی برگزار می‌شود و سال به سال به تنوع، جذب و عظمت آن‌ها افزوده می‌شود. یکی از این همایش‌های باشکوه که هر سال در چندین رشته در سطح دانش‌آموزان سال‌های آخر دوره‌ی متوسطه برگزار می‌شود المپیادهای علمی است. که قدیمی‌ترین آن المپیاد ریاضی بوده و از سال ۱۹۵۹ آغاز شده و تا به حال ادامه داشته است.

در حال حاضر نتیجه‌ی کسب شده در المپیادهای علمی برای هر کشوری یکی از شاخص‌های قدرت علمی آن کشور محسوب شده و نفرت ممتاز این المپیادها به راحتی جذب دانشگاه‌ها و آکادمی‌های ممتاز جهان شده و پس از گذشت چندین سال به موفقیت‌های چشم‌گیری نائل می‌شوند چنانچه بسیاری از دانشمندان حال حاضر در رشته‌های مختلف از جمله شیمی، فیزیک، IT و . . . در سال‌های نه چندان دور از مدال‌آوران این المپیادها بوده‌اند.

جمهوری اسلامی ایران برای اولین بار در سال ۱۳۶۶ در المپیاد ریاضی جهانی که در کشور کوبا برگزار می‌شد شرکت کرده و با کسب یک مدال برنز به مقام ۲۶ جهان نائل آمد که تعجب همگان را برانگیخت چرا که در آن سال ایران درگیر جنگ تحمیلی بود و جهانیان به غیر از جنگ و درگیری چیزی از ایران سراغ نداشتند و درخشش دانش‌آموزان ایران در آن سال و سال‌های بعد نگاه‌ها را به سمت ایران معطوف کرده و چشم خفته‌ی آن‌ها را تا حدود زیادی بیدار کرد. همان‌طور که از رسانه‌های گروهی مطلع شده‌اید در تمام المپیادهای علمی تیم اعزامی کشور عزیزمان در سال‌های گذشته جزء کشورهای برتر بوده و ضمن کسب مدال‌های رنگارنگ رتبه‌های بسیار درخشانی از جمله رتبه‌ی اول را احراز کرده‌اند.

نحوه‌ی گزینش نفرات اعزامی به المپیادهای جهانی تا حدود زیادی مشابه یکدیگرند به این صورت که در ابتدا در مسابقه‌ی سراسری تحت عنوان مرحله‌ی اول که معمولاً به صورت پرسش‌های چندگزینه‌ای مطرح می‌شود حدوداً هزار نفر پذیرفته شده و در رقابتی معمولاً تشریحی که مرحله‌ی دوم نامیده می‌شود شرکت می‌کنند. در این مرحله در هر رشته حدوداً چهل نفر پذیرفته شده و در دوره‌ی تابستانی در باشگاه دانش‌پژوهان جوان که متولی برگزاری تمام المپیادهای علمی می‌باشد شرکت

کرده و پس از گذراندن این دوره، مرحله‌ی سوم آزمون برگزار شده و عده‌ای (در حدود ده نفر) مدال طلا، عده‌ای مدال نقره و عده‌ای دیگر مدال برنز کسب می‌کنند (در این مرحله معمولاً همه افراد شرکت کننده در دوره مدال کسب می‌کنند) دارندگان مدال طلا حدود یک‌سال در آن باشگاه آموزش دیده و پس از آن اعضای تیم اعزامی شناسایی می‌شوند. دارندگان مدال طلا همگی بدون کنکور و در رشته و دانشگاه دلخواه خود پذیرفته شده و ادامه تحصیل می‌دهند اما دارندگان مدال‌های نقره و برنز همانند سایر داوطلبان در کنکور سراسری شرکت کرده و برای کسب رتبه دلخواه جهت پذیرفته شدن در رشته و دانشگاه مورد علاقه‌ی خود رقابت می‌کنند با این تفاوت که این افراد سهمیه‌ی ویژه‌ای در پذیرفته شدن در رشته و دانشگاه مورد علاقه‌ی خود دارند که جزئیات آن در سایت باشگاه دانش‌پژوهان جوان تشریح شده است.

متأسفانه در سال‌های اخیر در بعضی از مدارس افرادی مثلاً لباس کارشناسی به تن کرده و علیه فعالیت‌های المپیاد جبهه می‌گیرند و ادعا می‌کنند فعالیت برای المپیادهای علمی مانع موفقیت در کنکور سراسری بوده و هرچه دانش‌آموز به سمت المپیاد سوق پیدا کند از کنکور فاصله گرفته و در صورت عدم کسب مدال طلا (که بسیار محتمل است) آینده‌ی خود را تباه کرده است درحالی که با تحقیقی که در سال‌های گذشته انجام شده است فعالیت در زمینه المپیادهای علمی نه تنها مانع فعالیت برای کنکور نیست بلکه مسیر فعالیت برای کسب رتبه مناسب در کنکور را بسیار هموارتر می‌سازد به عنوان مثال می‌توانید تمام مدال آوران نقره و برنز و یا حتی آنهایی که در مرحله اول پذیرفته شده ولی به دوره تابستانی راه پیدا نکرده‌اند را در یک رشته شناسایی کرده و موفقیت‌های تحصیلی آن‌ها را در دانشگاه‌ها جویا شوید که نگارنده‌ی این متن بارها این تحقیق را انجام داده و به مثبت بودن آن یقین پیدا کرده است.

به هر حال ادعا این است که فعالیت دانش‌آموز در یک رشته از رشته‌های المپیاد فواید بسیاری دارد که به تعدادی از آن‌ها به صورت گذرا اشاره می‌شود:

۱. همان طور که خداوند به بشر تن سالم داده و انتظار می‌رود با ورزش‌ها و نرمش‌های مناسب از این نعمت خدادادی محافظت شود به هر دانش‌آموزی نیز استعدادی داده است که باید شکوفا و بهره‌ور شود. اغلب باشگاه‌های کشور اعم از خصوصی و دولتی داوطلب زیادی در رشته‌های متفاوت ورزشی دارند که مشغول فعالیت در یکی از رشته‌های ورزشی مانند کشتی، تکواندو، بدن سازی و... می‌باشند که وقتی از آن افراد راجع به اهدافشان از این فعالیت سؤال می‌شود سالم نگه داشتن بدن را عنوان داشته و انتخاب شدن در تیم ملی را در نهایت عنوان می‌کنند. چه بسا افرادی که در این رشته‌ها فعالیت می‌کنند و هرگز به تیم ملی راه پیدا نمی‌کنند که وقتی از این افراد راجع به موفقیت‌هایشان

سؤال می‌شود هرگز خود را ناموفق معرفی نمی‌کنند و همین که توانسته‌اند از بدن سالم خود به روش مناسب محافظت کنند را پیروزی بزرگی می‌دانند بنابراین فعالیت در یکی از زمینه‌های المپیاد چه در نهایت به کسب مدال منجر شود و یا نشود همین که استعداد خدادادی پرورش می‌یابد موفقیتی است بس بزرگ.

۲. کتب درسی به اذعان اکثر کارشناس‌ها و اساتید سال به سال ساده‌تر شده و برای عموم دانش‌آموزان دلچسب هستند ولی برای دانش‌آموزان ممتاز و تیزهوش به هیچ عنوان اغناکننده نمی‌باشد لذا لازم است این سری از دانش‌آموزان فعالیت ویژه‌ای را در رشته‌ی مورد علاقه‌ی خود داشته باشند تا احساس کنند این فعالیت‌ها برای آن‌ها اغناکننده است.

۳. فعالیت‌های المپیادی که در نهایت به حل سؤالات پیچیده و عمیق در رشته‌ی مربوطه می‌شود باعث می‌شود تا فرد به تمام مسائل جامعه و پیش‌آمده در زندگی به دید یک مسأله‌ی المپیاد نگاه کرده و در حل آن نسبت به سایر رقبا موفق‌تر باشند. تحقیقات نشان می‌دهد افرادی که با علاقه و اشتیاق حداقل یکی از شاخه‌های المپیاد را دنبال می‌کنند (نه به نیت کسب مدال بلکه به نیت پرورش ذهن) نسبت به سایر افراد در زندگی موفق‌ترند.

۴. زیربنای اکثر دروس پیش‌دانشگاهی در دروس المپیاد بنا نهاده می‌شود بنابراین افرادی که به سبک المپیادی دروس خود را مطالعه می‌کنند در دوره پیش‌دانشگاهی با پایه‌ی بسیار قوی‌تری با دروس مواجه می‌شوند و نسبت به رقبای خود راحت‌تر از عهده آن‌ها برمی‌آیند.

۵. با توجه به مصوبه‌های موجود، کسب مدال در یکی از المپیادهای علمی (حتی مدال برتر) باعث اعطای امتیازهای ویژه‌ای برای داوطلبان کنکور در ورود به دانشگاه‌های سراسری می‌شود که جزئیات آن در سایت‌های معتبر مخصوصاً سایت باشگاه دانش‌پژوهان جوان موجود است.

۶. همچنین با توجه به مصوبه‌های موجود اکثر المپیادها به عضویت نهادهای مختلف از جمله بنیاد ملی نخبگان درمی‌آیند که با رجوع به سایت‌های مرتبط با این نهادها و بنیادها امتیازات تعلق یافته به اعضا را مشاهده خواهید کرد.

انتشارات خوشخوان مفتخر است از بدو تأسیس به فکر تدوین و تألیف منابعی مناسب برای دانش‌آموزان ممتاز و داوطلبان المپیاد بوده است که خوشبختانه با یاری خداوند متعال و بهره‌گیری از اساتید مجربی که خود در سنواتی نه چندان دور مدال‌آور یکی از المپیادهای علمی بوده‌اند، کتب متعددی به بازار عرضه شده است که مورد توجه داوطلبان قرار گرفته است. بعد از کسب تجربیات لازم به این نتیجه رسیده‌ایم که لازم است کتبی به صورت کار تدوین و تألیف شود که در آن هر

کتاب مخصوص یک ترم تحصیلی باشد. این پروژه به نام درخت المپیاد نام گرفته است و هر کتاب از این پروژه که در اختیار دارید برگگی از آن درخت خواهد بود.

بدیهی است انجام چنین پروژهی عظیمی نظر و همت دسته‌جمعی می‌طلبد لذا لازم است از تمام دوستان و همکارانی که ما را در انجام این پروژه یاری نموده‌اند، تشکر و قدردانی نمایم و در نهایت نیز از عوامل زحمت‌کش انتشارات اعم از مشاورین، حروف‌چین‌ها، طراحان و کارمندان و کارگران عزیز کمال امتنان را دارم.



با تشکر

رسول حاجی‌زاده مدیر انتشارات خوشخوان



کتابی که ملاحظه می‌کنید مقدماتی از برنامه‌نویسی به زبان ++C است که با مجموعه‌ای از سؤالات و راه‌حل‌ها و همچنین درس‌نامه‌ای مختصر و مفید، تلاش شده تا حدالمقدور، دانش‌پژوهان به صورتی عملی مطالب را فراگیرند.

آنچه واضح است از سال ۱۳۸۰ به بعد با فراگیری کامپیوتر و استفاده از آن در بسیاری از شرکت‌ها و از طرفی با به وجود آمدن امکان استفاده از برنامه‌های کامپیوتری برای سهولت در کارها از جمله استفاده از آن در بانک‌ها و ...، برنامه‌نویسی اهمیت بسیاری را به دست آورده است. به همین خاطر، برنامه‌نویسی به‌عنوان پیش‌نیاز بسیاری از مباحث پیشرفته‌تر در علم کامپیوتر به شمار می‌آید و فراگیری آن به‌شدت کاربردی و مفید خواهد بود.

از سال ۸۹ به بعد، به المپیاد کامپیوتر مرحله سوم اضافه شد که در آن، امتحانی از برنامه‌نویسی گرفته می‌شود و برگزیدگان آن اجازه داشتند تا در آمادگی المپیاد کامپیوتر کشوری شرکت کنند و مدال المپیاد کسب کنند. به همین خاطر، آشنایی با مقدمات برنامه‌نویسی موضوعی شد که دانش‌آموزان مجبور بودند که قبل از ورود به دوره، فراگیرند و بسیاری از دانش‌آموزان و مدارس شروع به یادگیری و آموزش آن کردند.

در حال حاضر، به علت تازگی موضوع، تقریباً اکثر کتاب‌های مناسب، فارسی نیستند و این موضوع باعث می‌شود که بسیاری از دانش‌آموزان، دانش‌جویان و استعدادهای برتر به علت آشنایی نداشتن با زبان انگلیسی، امکان شکوفایی استعدادهایشان و یادگیری برنامه‌نویسی برای آن‌ها میسر نشود. از طرفی کتاب‌های موجود اکثراً به‌صورت تئوری آموزش داده شده ولی در این کتاب تلاش شده تا با آوردن سؤال‌های مناسب و راه‌حل آن‌ها، مفاهیم به‌صورت عملی و تجربی آموزش داده شوند. همچنین، تلاش شده تا مفاهیم ابتدایی برای آمادگی المپیاد کامپیوتر آموزش داده شود.

در هر مبحث، به‌صورت مختصر و مفید درس‌نامه‌ای وجود دارد که توضیحاتی در مورد آن موضوع داده شده است و سپس با آوردن سؤالاتی مناسب در مورد مبحث مذکور، به‌جا افتادن مفاهیم اولیه پرداخته‌ایم. همچنین، بعد از درس‌نامه راه‌حل‌های آن وجود دارد تا دانش‌پژوهان بتوانند راه‌حل‌های خود را با راه‌حل‌های صحیح مطابقت دهند. اکیداً توصیه می‌شود که سؤال‌های هر بخش توسط خود دانش‌پژوه حل شود و پس از حل شدن، یا تلاش برای حل شدن، به راه‌حل سؤال رجوع شود؛ زیرا اکثر مفاهیم کلیدی در سؤالات موجود است و عملاً با مطالعه‌ی درس‌نامه امکان فراگیری آن به‌صورت عمیق و مفهومی میسر نیست. این کتاب، صرفاً به آموزش مطالب اولیه و اساسی پرداخته است و برای کسانی که هیچ دانشی از برنامه‌نویسی ندارند مناسب است. مطالب و مفاهیم پیشرفته در این کتاب آورده نشده است و علاقه‌مندان می‌توانند با مطالعه‌ی منابع پیشرفته‌تر، به فراگیری کامل‌تر بپردازند. در حقیقت، با مطالعه این کتاب، انقبای

برنامه‌نویسی و پایه‌ی آن آموزش داده می‌شود.

این کتاب مانند همه کتاب‌های دیگر، ممکن است ایراداتی از جمله ایراد نگارشی یا ایراد علمی داشته باشد. از دانش‌پژوهان عزیز خواهشمندم مشکلات را از طریق ایمیل aliiaminian@yahoo.com به بنده اطلاع دهند تا امکان تصحیح آن‌ها وجود داشته باشد.



با تشکر

علی امینیان



فهرست مطالب

۱	عملگرها و عملیات‌ها	فصل ۱ 
	۱	عملیات حسابی	۱-۱
	۲	عملیات منطقی	۲-۱
	۴	سؤالات	۳-۱
	۵	راه حل‌ها	۴-۱
۷	انواع داده‌ها	فصل ۲ 
	۹	سؤالات	۱-۲
	۱۱	راه حل‌ها	۲-۲
۱۳	مقدمات نوشتن برنامه	فصل ۳ 
	۱۳	نرم افزار مناسب و اجرای برنامه	۱-۳
	۱۴	مقدمات نوشتن برنامه	۲-۳
	۱۴	بلوک	۳-۳
۱۷	جریان‌های ورودی و خروجی	فصل ۴ 
	۱۷	Cout	۱-۴
	۱۸	Cin	۲-۴
	۱۹	سؤالات	۳-۴
	۲۰	راه حل‌ها	۴-۴
۲۳	دستورهای شرطی	فصل ۵ 
	۲۳	ساختار if	۱-۵
	۲۴	ساختار if else	۲-۵
	۲۴	ساختار else if	۳-۵
	۲۶	محاسبه‌ی عبارت شرطی	۴-۵
	۲۷	سؤالات	۵-۵
	۲۸	راه حل‌ها	۶-۵
۳۵	حلقه‌ها	فصل ۶ 
	۳۵	حلقه‌ی while	۱-۶
	۳۷	سؤالات	۲-۶
	۳۹	راه حل‌ها	۳-۶

	۴۸	حلقه‌ی for	۴-۶
	۵۱	سؤالات	۵-۶
	۵۳	راه حل‌ها	۶-۶
۶۱	آرایه‌ها	فصل ۷ 
	۶۱	یک بُعدی	۱-۷
	۶۳	سؤالات	۲-۷
	۶۴	راه حل‌ها	۳-۷
	۷۱	دو بُعدی	۴-۷
	۷۲	سؤالات	۵-۷
	۷۳	راه حل‌ها	۶-۷
	۷۶	سه بُعدی و بالاتر	۷-۷
۷۷	توابع	فصل ۸ 
	۸۱	سؤالات	۱-۸
	۸۲	راه حل‌ها	۲-۸
۹۱	وکتور	فصل ۹ 
	۹۴	سؤالات	۱-۹
	۹۵	راه حل‌ها	۲-۹
۹۷	گراف‌ها	فصل ۱۰ 
	۹۹	سؤالات	۱-۱۰
	۱۰۰	راه حل‌ها	۲-۱۰
۱.۹	مسئله‌هایی از مرحله‌ی ۳ المپیاد کامپیوتر	فصل ۱۱ 
	۱۱۰	سؤالات	۱-۱۱
	۱۱۲	راه حل‌ها	۲-۱۱
۱۱۵	مسئله‌های اضافه	فصل ۱۲ 
	۱۱۶	سؤالات	۱-۱۲

فصل اول عملگرها و عملیات‌ها

عملیات حسابی

(۱-)

در زبان ++C، عمل اصلی شناخته شده هستند یعنی می‌توانیم ۲ عدد را با علامت + با هم جمع کنیم. هم‌چنین، برای ضرب از *، برای تفریق از - و برای تقسیم از / استفاده می‌کنیم. هم‌چنین، برای گرفتن باقی‌مانده‌ی a بر b از % استفاده می‌کنیم. مثلاً وقتی می‌نویسیم:

$$a = 3 * 5 - 1;$$

ابتدا مقدار ۳×۵ حساب می‌شود سپس یک واحد از آن کم می‌شود و عدد نهایی درون a ریخته می‌شود. دقت کنید که کامپیوتر اولویت‌ها را می‌شناسد و طبق آن‌ها عمل می‌کند یعنی در مثال بالا ابتدا ضرب و سپس تفریق را انجام می‌دهد. اگر می‌خواستیم که اول تفریق انجام شود باید با پرانتز آن را اولویت می‌دادیم؛ یعنی:

$$a = 3 * (5 - 1);$$

هم‌چنین، = به معنی قرار دادن مقدار متغیر سمت راستش در سمت چپش است. برای مثال:

$$a = b;$$

یعنی مقدار داخل متغیر b را داخل a قرار بدهد؛ بنابراین، بعد از این دستور مقدار داخل متغیر a برابر مقدار داخل متغیر b است. دقت کنید = به هیچ عنوان تساوی دو طرف را چک نمی‌کند و برای چک کردن تساوی دو متغیر از نماد دیگری استفاده می‌کنیم که با این اشتباه نشود. برای مثال:

$$a = 3;$$

$$b = 3 * 6;$$

$$c = b * 2;$$

$$b = b + c;$$

$$c = c + 1;$$

$$a = c / 2;$$

در خط اول = وجود دارد، پس مقدار سمت راست آن داخل مقدار سمت چپ ریخته می‌شود؛ یعنی عدد ۳ داخل a ریخته می‌شود و از این پس مقدار آن ۳ است.

در خط دوم، مجدد = وجود دارد، پس باید مقدار سمت راست محاسبه شده و در سمت چپ ریخته شود. برای این کار، مقدار سمت راست، خود یک ضرب دارد و ابتدا باید حساب شود، پس کامپیوتر ابتدا 3×6 را محاسبه می‌کند و به جای آن عدد ۱۸ را سمت راست عبارت جایگزین می‌کند، پس از آن، ۱۸ را داخل b می‌ریزد.

در خط بعد = وجود دارد و سمت راست \times داریم، پس ابتدا مقدار b در ۲ ضرب می‌شود یعنی $18 \times 2 = 36$ و عدد ۳۶ داخل c ریخته می‌شود. در خط بعد، ابتدا سمت راست محاسبه می‌شود یعنی مقادیر b و c باهم جمع می‌شوند، پس $18 + 36 = 54$ و حالا مقدار ۵۴ داخل b ریخته می‌شود. از این پس دیگر مقدار قبلی b از بین می‌رود و داخل آن ۵۴ وجود دارد. این نکته حائز اهمیت است که وقتی داخل b مقداری ریخته می‌شود، دیگر مقدار قبلی‌اش کاملاً از دست می‌رود. در خط بعد، مقدار $c + 1$ یعنی $36 + 1 = 37$ حساب می‌شود و داخل c ریخته می‌شود پس دیگر مقدار ۳۷ را دارد و مقدار قبلی‌اش یعنی ۳۶ از بین رفت. این خط هم خط مهمی است، زیرا برای یک واحد اضافه کردن یک متغیر از این دستور استفاده می‌کنیم. همچنین، برای یک واحد کم یا زیاد کردن متغیری مثلاً d می‌نویسیم:

`d-- ; d++ ;`

در خط بعد هم مقدار c که ۳۷ است تقسیم بر ۲ می‌شود $37 \div 2 = 18/5$ که چون a باید عدد صحیح باشد مقدار ۱۸ توسط خود کامپیوتر در نظر گرفته می‌شود و داخل a ریخته می‌شود. پس در آخر، مقدار a برابر ۱۸، مقدار b برابر ۵۴ و مقدار c برابر ۳۷ است.

خیلی اوقات لازم می‌شود که متغیری را زیاد، کم و یا ضرب کنیم. برای این کار، به طور مثال یک واحد اضافه کردن متغیر a، می‌نویسیم:

`a = a + 1 ;`

در زبان C++، برای این کار مخفی وجود دارد. همچنین، باعث خوانایی برنامه خواهد شد. این دستور به شکل زیر است:

`a += 1 ;`

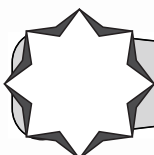
و به طور کلی، برای ضرب کردن a در b می‌نویسیم:

`a *= b`

که دقیقاً مشابه:

`a = a * b ;`

است.



عملیات منطقی

۲-۱

علاوه بر عملگرهای حسابی، عملگرهای منطقی هم موجود است که معمولاً برای چک کردن به کار می‌روند. علامت کوچک‌تر به شکل <، کوچک‌تر مساوی به شکل <=، بزرگ‌تر به شکل > و بزرگ‌تر مساوی به شکل >= است. مثلاً برای این که بفهمیم مقدار b بزرگ‌تر از مقدار a هست یا خیر، می‌نویسیم $b > a$. وقتی کامپیوتر به $b > a$ می‌رسد، بررسی می‌کند که آیا

مقدار b از مقدار داخل a بزرگ‌تر هست یا خیر، اگر بزرگ‌تر باشد به معنی آن هست که $b > a$ درست است پس به جای کل این عبارت مقدار $true$ یا همان 1 گذاشته می‌شود.

بنابراین، زمانی که به‌عنوان یک شرط بیان شده باشد، ابتدا مقدار آن محاسبه می‌شود و اگر صحیح بود با $true$ یا 1 و اگر صحیح نبود با 0 یا $false$ جایگزین می‌شود (توسط خود کامپیوتر) و از آن پس اگر حاصل 1 بود به معنی صحیح بودن شرط، کارهای آتی را انجام خواهد داد.

یکی دیگر از عملگرهای مقایسه‌ای $==$ است که برای برابر بودن دو متغیر استفاده می‌شود. وقتی می‌نویسیم $a == b$ ، یعنی آیا مقدار a برابر مقدار b هست یا خیر، اگر برابر بود، این عبارت صحیح خواهد شد و در غیر این صورت، این عبارت $false$ خواهد شد. همان‌طور که در بالا اشاره شد، $=$ به معنی ریختن مقدار متغیر سمت راست در متغیر سمت چپ است و $==$ به معنی چک کردن تساوی دو متغیر. دقت شود که این دو، جای هم به کار نروند وگرنه باعث درست کار نکردن برنامه خواهد شد. همچنین از $!=$ برای نابرابر بودن استفاده می‌شود. مثلاً وقتی داریم $b != a$ یعنی اگر b با a برابر نبود، یا به عبارتی وقتی این عبارت صحیح است که a با b برابر نباشد و در غیر این صورت این عبارت 0 خواهد شد. توجه کنید که حاصل هر عبارت مقایسه‌ای، در نهایت 0 یا 1 خواهد شد که کامپیوتر خودش پس از چک کردن برقراری رابطه اگر صحیح بود کل آن عبارت را با 1 و در غیر این صورت با 0 جایگزین می‌کند.

سؤالات ۳-۱

۱. مقادیر متغیرها را در پایان برنامه به دست آورید.

```

b = 8;
a = 4 + 1 * 2;
b == 2;
a = 4 - 1 / 2;
c = a + 1;
c = c + 1;
d = c = a;
e = c == 5;
a == a = b;
a = b <= c;
a < b < c;
    
```

۲

۳

۴

۵

۶

۷

۸

۹

۱۰

۱۱

۱۲

۱. در ابتدا ۸ داخل متغیر b ریخته می شود.

سپس در خط بعد سمت راست آن باید حساب شود؛ یعنی ابتدا ۱×۲ حساب می شود.

سپس حاصل آن با ۴ جمع می شود.

سپس حاصل آن یعنی ۶ در a ریخته می شود.

دقت کنید که ترتیبها بسیار مهم هستند و کامپیوتر ابتدا سمت راست را کامل حساب می کند و در هر مرحله مقدار محاسبه شده را جایگزین می کند. برای مثال وقتی ۱×۲ حساب شد و مقدار آن ۲ شد ۱×۲ کامل پاک می شود و جای آن ۲ گذاشته می شود. حال $۴+۲$ حساب می شود و قبل از اینکه در a ریخته شود مقدار پاک شده و جایگزین می شود. سپس سمت راست ۶ و سمت چپ a هست پس ۶ را داخل a می ریزد.

در خط بعد چون دو تا مساوی داریم چک می شود که آیا b برابر ۲ هست یا نه که چون $b, ۸$ است و مساوی نیست پس کل خط $۲ = b = ۰$ جایگزین شده بنابراین، این خط هیچ فایده ای ندارد و مقدار b ثابت می ماند.

در خط بعد $۱ \div ۲$ می شود $۰/۵$ اما در کامپیوتر چون پیش فرض همیشه عدد صحیح بودن هست چون مقدار آن می شود $۰/۵$ که عدد صحیح نیست، قسمت صحیح آن یعنی ۰ در نظر گرفته می شود و ۰ جایش قرار می گیرد.

سپس $۰-۴$ حساب شده و ۴ جایگزین می شود و پس از آن مقدار ۴ در a ریخته می شود. پس الان در $a, ۴$ وجود دارد و دیگر مقدار قبلی خود را از دست داده است. در حقیقت ۴ روی قبلی ریخته شد.

سپس در c مقدار a که ۴ هست با ۱ جمع می شود یعنی ۵ و در c ریخته می شود. دقت کنید که مقدار a ثابت می ماند و با ۱ جمع نمی شود.

سپس مقدار c یعنی ۵ با ۱ جمع می شود و در خودش یعنی c ریخته می شود، یعنی c برابر ۶ می شود. دقت شود برای این که یک متغیر ۱ واحد افزایش پیدا کند این کار را می کنیم.

سپس $d=c=a$ اگر از سمت راست کارها را یکی یکی انجام دهیم، ابتدا a داخل c ریخته می شود یعنی همان ۴ پس $c, ۴$ می شود و سپس در $d, ۴$ ریخته می شود. پس این خط هر ۳ متغیر را برابر a یعنی همان ۴ می کند و از اینجا به بعد هر متغیر مقدار ۴ را دارد.

در خط بعد، از سمت راست، ابتدا $c=۵$ بودن چک شده که چون $c=۵$ است یعنی چک شدن، پس جواب آن یا یک هست یا صفر که چون c برابر ۴ است اما سمت راست ۵ است پس این مساوی برقرار نیست و کل عبارت با ۰ جایگزین می شود، حال ۰ در e ریخته می شود.

در خط بعد، سمت راست $c=۵$ طبق تعریف، باید یک متغیر باشد، نه یک عبارت، پس این خط error خواهد داد و برنامه اجرا نخواهد شد (امتحان کنید و با تصحیح آن، برنامه را اجرا کنید).

در خط بعد، b یعنی عدد ۸ چک می شود که از c کوچک تر یا مساوی باشد که چون $c, ۴$ است پس این شرط برقرار نیست و جایش ۰ می آید. سپس در $a, ۰$ ریخته می شود.

در خط بعد هم که ابتدا $c > b$ بودن چک می‌شود، یعنی $۸ < ۴$ که چون برقرار نیست جای آن ۰ می‌آید و سپس $a < ۰$ بودن چک می‌شود که باز هم برقرار نیست. دقت کنید که متوجه می‌شویم که این خط کلاً اتفاق نادرست و دور از انتظاری می‌افتد و اصلاً مثل ریاضی نیست که این خط وقتی درست باشد که هم a کوچک‌تر از b باشد هم b کوچک‌تر از c .

۲

۳

۴

۵

۶

۷

۸

۹

۱۰

۱۱

۱۲

فصل دوم انواع داده‌ها

برای گرفتن حافظه توسط کامپیوتر، لازم است که نوع حافظه را مشخص کنیم. فرض کنید می‌خواهیم یک عدد صحیح را در کامپیوتر ذخیره کنیم؛ بنابراین، باید به کامپیوتر بگوییم که یک حافظه از نوع عدد صحیح به ما بدهد و پس از آن، هم ما و هم کامپیوتر بدانند که با داده‌ی داخل آن حافظه چطور برخورد کند. از مهم‌ترین انواع داده‌هایی که وجود دارند عبارت است از:

`int` (integer): این نوع داده برای نگهداری عدد صحیح است و حافظه آن ۴ بایت است و از آن جایی که هر بایت ۸ بیت خانه دارد که هر کدام ۰ یا ۱ می‌پذیرند، پس جمعاً $8 \times 4 = 32$ بیت (خانه حافظه) رزرو می‌شود و چون محدوده اعداد مثبت و منفی برای `int` قابل قبول است، پس از -2^{31} تا $2^{31}-1$ محدوده عدد قابل قبول برای `int` است. برای مثال اگر عددی بزرگ‌تر از محدوده مورد نظر به یک حافظه `int` داده شود، از آن جایی که حافظه لازم برای آن وجود ندارد، کامپیوتر به صورت خودکار روی آن تغییراتی ایجاد می‌کند و عدد، هنگام ذخیره به کلی عوض خواهد شد. پس باید حواسمان باشد که محدودیت‌های هر تایپ را رعایت کنیم.

`short int`: این نوع حافظه ۲ بایت یا همان ۱۶ بیت است و حافظه لازم برای نگهداری آن تنها ۱۶ بیت است؛ یعنی برای نگهداری اعدادی که کوچک هستند بهتر است از `short int` استفاده کنیم. محدوده آن از -2^{15} تا $2^{15}-1$ است.

`long long int`: این نوع داده ۸ بایت یا همان ۶۴ بیت است و محدوده آن از -2^{63} تا $2^{63}-1$ است.

`float`: این نوع داده برای نگهداری اعداد اعشاری است. مقداری بیت به نگهداری قسمت صحیح عدد اختصاص می‌دهد و مقداری بیت هم به قسمت اعشاری عدد اختصاص می‌دهد. به علت کم اهمیت بودن، محدوده دقیق اعداد قابل قبول اینجا آورده نشده و در صورت تمایل قابل جستجو در اینترنت است.

`double`: `double` هم مانند `float` مخصوص اعداد اعشاری است با این تفاوت که مقدار حافظه بیشتری می‌گیرد و در عوض اعدادی که می‌تواند ذخیره کند در محدوده بیشتری هستند.

char (character): این نوع داده مخصوص نگهداری یک کاراکتر است و حافظه آن ۱ بایت است. کاراکترها انواع مختلفی دارند و در حقیقت تمام شکلک‌هایی که در کامپیوتر مشاهده می‌شود هر کدام یک کاراکتر هستند مانند حروف کوچک انگلیسی یا حروف بزرگ انگلیسی و همچنین اعداد ۰ تا ۹ و حتی شکل‌هایی نظیر: * & ^ % و ... در حقیقت هر سمبلی که در کامپیوتر قابل مشاهده است به عنوان یک کاراکتر شناخته می‌شود و قابل ذخیره هست. برای هر کاراکتر، عددی استاندارد کرده‌اند که به جای آن، عدد نظیرش را خود کامپیوتر در حافظه قرار می‌دهد و هنگامی که به آن خانه از حافظه نگاه می‌اندازیم بازهم خود کامپیوتر طبق جدولش آن عدد را تبدیل به شکل متناظرش می‌کند و آن را نمایش می‌دهد. به استاندارد یک که وجود دارد و برای هر سمبل یک عدد اختصاص داده‌اند، کد اسکی می‌گویند. در حقیقت طبق جدول زیر، هر سمبلی یک عدد دارد. برای مثال 'A' عدد ۶۵ است به این معنی که اگر بخواهیم در حافظه‌ای 'A' را ذخیره کنیم ما می‌نویسیم 'A' اما در حقیقت خود کامپیوتر بجای 'A' عدد ۶۵ را ذخیره می‌کند.

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	\$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

bool: در این نوع داده، فقط دو نوع حالت قابل نگهداری است و مخصوص وضعیتی است که ما می‌خواهیم صحیح یا غلط بودن چیزی را نگهداری کنیم، یا به‌طور کلی می‌خواهیم متغیری را در آن نگاه داریم که آن متغیر دو حالت بیشتر ندارد و آن ۰ و ۱ است. می‌توانیم درون آن‌ها false یا true بریزیم که false معادل ۰ و true معادل ۱ است. اگر در متغیر bool یک عدد غیر از ۰ و ۱ بریزیم، چون که هر عدد غیر صفری در کامپیوتر درست در نظر گرفته می‌شود پس به‌جای آن ۱ ریخته می‌شود که همان true است.

متغیرهای دیگری هم وجود دارد از جمله unsigned int که مفهوم آن عدد صحیح بدون علامت است که قابل جستجو در اینترنت است و به علت کم اهمیت بودن، اینجا توضیحی در مورد آن‌ها داده نشده است.

۱. در کد زیر چند بایت و چند بیت حافظه گرفته شده است؟

```
int a;
int b;
long long int c, d;
char e;
a = 5;
b = a + 1;
```

۲. مقادیر a, b, c و d را در زیر مشخص کنید.

```
int a = 100;
int b = a + 800;
long long int c = -18;
a = b + c;
char d = 's';
char e = 'r';
e = 65;
```

۳. مقدار متغیرهای زیر را در انتهای برنامه بنویسید.

```
int a = 'c';
int b = 100000000000;
long long int c = 1000000000000;
char d = 'a' + 2;
char e = 'a' + '2';
e = 35 + 30;
int f = d;
float g = 8.8;
int h = g;
bool k = true;
bool z = 5;
```

```
bool m = 0;  
bool n = 1 - m;  
char p = n;
```

۱

۲

۴

۵

۶

۷

۸

۹

۱۰

۱۱

۱۲

انتشارات
C++

۱. می‌دانیم که هر بایت حافظه شامل ۸ بیت خانه حافظه است. هر بیت هم یک خانه حافظه است که یا ۰ است یا ۱.

می‌دانیم هر int ، ۴ بایت است، هر $long\ long\ int$ ، ۸ بایت و هر کاراکتر ۱ بایت. پس ۲ تا ۴ بایت داریم. چون $long\ long\ int$ به نام‌های c و d تعریف شده، ۲ تا ۸ بایت هم داریم و در انتها یک ۱ بایت داریم. در خط آخر چون فقط عملیاتی انجام شده است و حافظه‌ای تعریف نشده است پس حافظه‌ای اضافه‌تر گرفته نمی‌شود. پس کلاً ۲۵ بایت حافظه یا ۲۰۰ بیت حافظه گرفته شده است. (حال بهتر درک می‌کنیم فایل متنی که مثلاً حجمش ۳۰۰ کیلوبایت است منظور چه مقدار حافظه است.)

۲. ابتدا متغیر a تعریف شده و مقدار آن ۱۰۰ گذاشته شده است. سپس متغیر b تعریف شده و مقدار آن $a+800$ گذاشته شده که چون a از خط قبل مقدار ۱۰۰ را دارد پس $a+800$ توسط کامپیوتر محاسبه می‌شود و مقدار ۹۰۰ داخل b ریخته می‌شود. دقت کنید که مقدار ۹۰۰ ریخته می‌شود و دیگر کاری به a ندارد و اگر a عوض شود هم باز مقدار b ، ۹۰۰ می‌ماند.

در خط بعد c را تعریف و مقدار c را -18 گذاشتیم. در خط بعد مقدار a را برابر $c+b$ گذاشتیم که $b=900$ و c برابر -18 است. پس مقدار a که ۱۰۰ بود، ۸۸۲ می‌شود و دیگر مقدار قبلی‌اش از بین می‌رود.

در خط بعد کاراکتر d تعریف شده و کاراکتر s در آن ریخته شده است. دقت کنید که در حقیقت کد اسکی s در آن ریخته می‌شود. توجه کنیم که برای ریختن کاراکتری در متغیر باید آن کاراکتر داخل کوتیشن “ باشد. در خط بعد حافظه e از نوع کاراکتر تعریف شده و کاراکتر r در آن ریخته شده است (کد اسکی آن). در خط بعد داخل e که از نوع کاراکتر است نه عدد (!)، عدد ۶۵ ریخته شده است؛ اما این کار قابل انجام است و کامپیوتر کاری ندارد به e و فکر می‌کند که ما داریم کد اسکی داخل آن می‌ریزیم؛ اما وقتی آن را چاپ می‌کنیم یا کاری انجام می‌دهیم کامپیوتر خودش کاراکتری را پیدا می‌کند که کد اسکی آن ۶۵ است یعنی 'A'. پس در نهایت کاراکتر r پاک می‌شود و A ریخته می‌شود.

۳. خط اول مقدار a برابر کد اسکی c یعنی ۹۹ است.

سپس در خط بعد داخل متغیر از نوع int به نام b عدد مشاهده شده ریخته می‌شود. دقت کنید که int ، ۴ بایت یعنی ۳۲ بیت است پس امکان ذخیره در آن از یک عددی بزرگ‌تر وجود ندارد و در صورت این کار، عددی دیگر ذخیره خواهد شد. پس در c یک عددی ذخیره می‌شود اما عدد داخل b از بین خواهد رفت و عدد دیگری در آن خواهد ماند زیرا b از نوع int است. در صورت علاقه می‌توانید مباحث $one's\ complement$ و $two's\ complement$ را در اینترنت مطالعه کنید تا طریقه نگهداری اعداد توسط کامپیوتر را دقیق‌تر متوجه بشوید.

در خط بعد کاراکتر a با ۲ جمع شده و داخل d ریخته می‌شود. در حقیقت کاراکتر 'a'، کد اسکی آن است که ۹۷ است و وقتی با ۲ جمع شود ۹۹ می‌شود و داخل d ریخته می‌شود؛ و چون d کاراکتر است، کامپیوتر می‌داند که کد اسکی است و کاراکتر آن یعنی 'c' را به ما نشان می‌دهد. در واقع، کاراکترها، کد اسکی‌شان پشت سر هم هستند پس وقتی به a ، ۲ واحد اضافه می‌کنیم 'c' می‌شود.



در خط بعد، داخل کاراکتر e، کاراکتر ۲ و کاراکتر ' a ' جمع شده، ریخته می‌شود. دقت کنید که اینجا ۲، داخل کوتیشن است یعنی منظور کد اسکی ۲ بوده نه عدد ۲. پس اینجا کد اسکی ' ۲ ' با کد اسکی ' a ' جمع می‌شود و یک کاراکتری می‌شود که به داخل e می‌رود. در خط بعد، دو عدد با هم جمع می‌شوند یعنی ۳۵ و ۳۰ و داخل e ریخته می‌شوند که همان ۶۵ یا ' A ' است.

در f که از نوع int است مقدار d که از نوع کاراکتر است ریخته می‌شود. در حقیقت با این کار، کد اسکی d در f ریخته می‌شود؛ یعنی ۹۹ و این بار کامپیوتر با f یعنی ۹۹ مثل int برخورد می‌کند. سپس داخل g مقدار ۸/۸ ریخته شده است. در h که int است مقدار g یعنی ۸/۸ ریخته می‌شود اما چون h، int است، ۰/۸ آن می‌گیرد و ۸ ذخیره می‌شود.

در خط بعد، داخل k مقدار true یا همان ۱ ذخیره می‌شود. در خط بعد داخل z مقدار ۵ ریخته می‌شود که چون bool است و فقط صحیح و غلط یا همان ۰ و ۱ ذخیره می‌شود و چون هر چیزی به جز ۰ صحیح است پس ۵ هم صحیح است پس داخل z صحیح یا همان ۱ ریخته می‌شود و ۵ از بین خواهد رفت. خط بعد داخل m false یا همان ۰ می‌رود. در n مقدار ۱-m یعنی ۱ که همان true است می‌رود پس n، true می‌شود. در حقیقت با عمل $h=1-h$ ، مقدار h برعکس می‌شود (اگر h، bool باشد) و این یک راه برای برعکس کردن یک متغیر بدون چک کردن آن است.

در خط بعد، داخل p مقدار n یعنی ۱ ریخته می‌شود و هنگام چاپ، چون کاراکتر است آن کاراکتری که کد اسکی آن ۱ است چاپ خواهد شد.

بهتر است که این برنامه را خودتان بنویسید و خروجی‌ها را چک کنید. دقت کنید که بعضی از کاراکترها قابل دیدن نیستند مثلاً tab، space و enter این‌ها همگی هنگام چاپ به یک شکل دیده می‌شوند.

فصل سوم

مقدمات نوشتن برنامه

نرم افزار مناسب و اجرای برنامه

۱-۳

حال که با تعریف متغیرها و گرفتن حافظه آشنا شدیم، می‌خواهیم اولین برنامه‌ی خود را بنویسیم. قبل از آن لازم است تا ابتدا طریقه‌ی نوشتن برنامه در کامپیوتر و اجرای آن را کمی توضیح دهیم.

هر برنامه C++ شامل یک فایل با پسوند `cpp` است که داخل آن، متن برنامه نوشته شده است. در واقع، در هر فایل متنی که برنامه نوشته شود و با پسوند `cpp` ذخیره شود، این برنامه قابل قبول است؛ اما هر محیطی به منظور خاصی نوشته شده است، مثلاً، نرم‌افزار `word`، برای نوشتن و ویرایش متن طراحی شده و برای نوشتن برنامه‌ی C++ طراحی نشده و کار را دشوار خواهد کرد. از آن طرف، نرم‌افزارهایی وجود دارد که مخصوص برنامه‌نویسی هستند و کار را برای برنامه‌نویس آسان‌تر خواهند کرد. در ویندوز، نرم‌افزار `dev` یکی از این مثال‌ها است، همچنین، `eclipse`، `gvim`، `xcode` در `mac` و اِدیتورهای فراوانی وجود دارند و قابل استفاده هستند. دقت کنید که بیشتر انتخاب اِدیتور سلیقه‌ای و بر اساس عادت شخص است.

حال که یک اِدیتور مناسب انتخاب کردیم، می‌توانیم داخل آن برنامه بنویسیم و با نام مورد نظر مثلاً `havi j.cpp` آن را ذخیره کنیم.

می‌دانیم که این برنامه به زبان ما نوشته شده است و همچنان کامپیوتر آن را نخواهد فهمید؛ بنابراین، باید به روشی این برنامه را به زبان ماشین تبدیل کنیم (که صفر و یک است). این کار وظیفه برنامه‌ای به نام `compiler` است، یا به عبارتی برنامه ما باید `compile` شود؛ و پس از کامپایل شدن، یک نسخه از آن به زبان ماشین تولید می‌شود و پسوند آن اجرایی است برای مثال در ویندوز `exe` خواهد بود. حال با اجرا کردن آن فایل اجرایی برنامه اجرا خواهد شد. همچنین، هنگام کامپایل کردن، کامپایلر ممکن است ایرادهایی از برنامه ما بگیرد و قابلیت ایجاد فایل اجرایی نباشد که در آن صورت باید ابتدا ایرادهایی را که داشتیم برطرف کنیم.

```
g++ havi j.cpp -o golabi
```

با اجرای این خط در محیط terminal یا همان cmd، فایل `havij.cpp` کامپایل شده و گلابی به عنوان فایل اجرایی به وجود می آید. حال دستور زیر فایل را اجرا می کند:

```
./golabi
```

بعضی از اِدیتورها، داخل خودشان کامپایلر دارند و تمامی این کارها با یک دکمه اتفاق می افتد. در ویندوز، `dev-cpp` این خاصیت را دارد؛ و ما در این کتاب برای راحتی با اِدیتور `dev-cpp` کار می کنیم.

کسانی که در سیستم عامل دیگری برنامه نویسی می کنند یا به کار کردن با اِدیتور دیگری علاقه مند هستند، می توانند با بررسی در اینترنت کار کردن با آن را یاد بگیرند.

در محیط `dev`، برنامه نوشته می شود، در منوی بالا، دکمه `compile` وجود دارد و پس از کامپایل، دکمه `run` موجود است که برنامه را اجرا می کند. برای آشنایی بیشتر آن را نصب کرده و با آن کار کنید و همچنین، در وبسایت آن می توانید توضیحات تکمیلی را مشاهده فرمایید.

۲-۳ مقدمات نوشتن برنامه

حال نوبت به نوشتن برنامه می رسد. هر برنامه C++ خطهایی دارد که همیشه ثابت هستند و نوشته می شوند. در ابتدا کتابخانه هایی که لازم داریم را می نویسیم که معمولاً یکی از آن ها همیشه ضروری خواهد بود. سپس یک خط همیشه ثابت داریم که `std` را اضافه می کند. پس از آن بلوک `main` را می نویسیم. در حقیقت هر برنامه C++ باید یک `main` داشته باشد و هنگام اجرای برنامه، همیشه داخل `main` شروع به اجرا شدن می شود.

۳-۳ بلوک

در بعضی قسمت ها، از جمله خود `main`، در شرطها و همچنین حلقه ها، پس از نوشتن یک دستور (مثلاً دستور شرط که `if` است)، پس از آن { باز می شود و با } انتهای آن مشخص می شود، بنابراین، تمام دستورهای داخل { } مربوط به آن شرط هستند (در مثال ها بهتر متوجه خواهید شد). به هر کدام از این ها یک بلوک می گویند. برای مثال خود `main` هم که در پایین تر خواهیم دید، با { شروع می شود، دستورات داخل آن نوشته می شوند و سپس با } پایان می یابد و کل `main` یک بلوک حساب می شود.

نکات زیر در برنامه ها دقت شود:

- ✓ در C++ فاصله بیشتر از یکی، بی معنی بوده و برای کامپیوتر بی تأثیر است؛ بنابراین، گذاشتن چند فاصله هیچ مشکلی در برنامه ایجاد نمی کند. همچنین همین موضوع برای `new line` یا همان `enter` هم برقرار است.
- ✓ بین کلمات کلیدی باید یک فاصله زده شود تا کامپیوتر دستورهای مجزا را تشخیص دهد.
- ✓ هر دستور باید با `semicolon` (;) خاتمه پیدا کند. در حقیقت کامپیوتر جلو می رود تا به ; برسد و بعد متوجه می شود که یک دستور باید اجرا شود.

با وجود این که در نوشتن برنامه‌ها، آزادی زیادی وجود دارد و باسلیقه‌های مختلف می‌توان نوشت، استانداردهایی وجود دارد که آن‌ها را رعایت کنیم. در غیر این صورت، وقتی برنامه از حدی بیشتر و پیچیده‌تر شود، به‌سختی قابل نوشتن خواهد بود.

۱- هر دستور در یک خط نوشته شود. درست است که از `enter` ها چشم‌پوشی می‌شود و کامپیوتر وجود آن‌ها را در نظر نمی‌گیرد و می‌توان کل برنامه را (۲۰۰ دستور برای مثال) در یک خط نوشت، اما به‌شدت کار را دشوار خواهد کرد؛ بنابراین هر دستور در یک خط نوشته شود.

۲- از `space` زدن اضافه پرهیز شود و فقط بین کلمات معنی‌دار یک `space` زده شود.

۳- داخل هر بلوک، به فاصله یک `tab` از اول بلوک، جلوتر نوشته شود.

حال برنامه ساده زیر را در نظر بگیرید که تقریباً همه برنامه‌های `C++` آن‌را خواهند داشت:

```
#include <iostream>
using namespace std;
int main() {
    return 0;
}
```

همان‌طور که مشاهده می‌کنید، کتابخانه‌ها با اسم‌های مختلف (در این مثال `iostream`) به شکل بالا اضافه می‌شوند. بلوک `main` حتماً باید تعریف شود. داخل `{ }` دستورات نوشته می‌شود؛ و خط آخر `main`، `return 0` را باید داشته باشیم. دقت کنید، همان‌طور که در بالا گفتیم، دستورات داخل بلوک `main` با یک `tab` فاصله نوشته شده‌اند.

دقت کنید که شروع بلوک یعنی همان جایی که `{ }` خواهد داشت، دیگر نداریم زیرا هنوز دستور تمام نشده است. هم‌چنین در انتهای بلوک‌ها هم یعنی جایی که با `{ }` بسته می‌شود، بازهم نخواهیم داشت.

برای مثال:

```
include <iostream>
using namespace std;
int main() {
    int a;
    int b;
    int c;
    a = 8;
    b = 11;
    c = a + b;
    return 0;
}
```



برنامه کوچک بالا، یک main دارد، مقداری دستور شامل تعریف متغیر و همچنین تعدادی عملیات دارد و سپس برنامه تمام می‌شود (این برنامه فقط جنبه آموزشی دارد و با اجرای آن چیزی مشاهده نمی‌شود، زیرا دستوری برای نمایش دادن استفاده نشده است).

بهتر است این چند دستوری که معرفی شد و برای همه برنامه‌ها ضروری است را در ابتدا حفظ کنید تا به مرور زمان، با ادامه درس، دلیل آن‌ها توضیح داده شود.

دقت کنید که گاهی کسانی که از dev استفاده می‌کنند ممکن است با مشکلی مواجه شود که هنگام اجرای برنامه، برنامه به سرعت اجرا شده و صفحه بسته می‌شود و امکان مشاهده وجود نداشته باشد. در این صورت قبل از `return 0` دستور زیر استفاده کنید:

```
system ("pause");
```

با حل سؤالات از فصل بعد مفاهیم را بهتر یاد خواهید گرفت. برای این فصل، برنامه dev را دانلود کرده و نصب کنید و دو برنامه کوچک بالا را نوشته و اجرا کنید.

برای mac و linux هم داخل اِدیتور مورد نظر برنامه‌های بالا را بنویسید و سپس از terminal آن‌را همان‌طور که توضیح داده شد اجرا کنید.