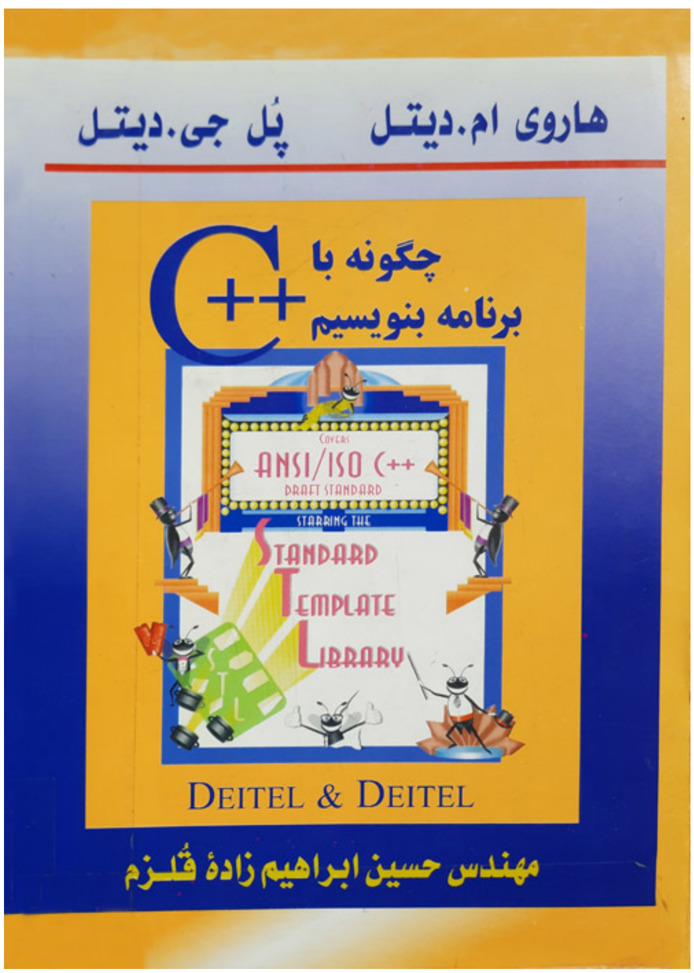
C ǯƘƛƸ Ǵƛ ÆǨȿǽƥ ǴǬƘƝƸƛ



معرفی ساختاری زبان C

پس از نوشتن یک الگوریتم باید آن را با استفاده از یک زبان برنامه نویسی تبدیل به یک برنامه قابل اجرا برای کامپیوتر نماییم. این زبانها به سه دسته کلی تقسیم میگردند :

۱- زبان ماشین (سطح پایین) : این زبان مستقیما با صفر و یک نوشته می شود و بدون هیچ واسطه ای برروی کامپیوتر قابل اجرا است. طراحان سخت افزار هر کامپیوتر، زبان ماشین خاص خود را برای آن ماشین طراحی می نمایند. به همین دلیل هر برنامه ای که به زبان ماشین نوشته شود، فقط برروی همان ماشین خاص کار می کند، بهمین دلیل برنامه های نوشته شده به زبان ماشین را غیر قابل حمل می نامند. از طرف دیگر یادگیری این زبان بسیار مشکل بوده و برنامه نویسی با آن نیز بسیار سخت است و همچنین احتمال بروز خطا نیز در آن زیاد است.

۲- زبان اسمبلی : این زبان شکل ساده تر زبان ماشین است، بدین صورت که برای هر دستورالعمل زبان ماشین، یک اسم نمادین انتخاب شده است (مانند دستور ADD بجای کد دودویی دستورالعمل جمع) که بخاطر سپردن و برنامه نویسی با آنها برای انسانها ساده تر است. اما این برنامه ها برای ماشین قابل فهم نیست و باید قبل از اجرا شدن توسط برنامه مترجمی بنام اسمبلر به زبان ماشین تبدیل شود. توجه کنید که از آنجا که هر دستور زبان اسمبلی معادل یک دستور زبان ماشین است، این زبان نیز وابسته به ماشین می باشد و برنامه های نوشته شده به این زبان فقط برروی همان کامپیوتری که برای آن نوشته شده اند قابل اجرا است. علاوه براین کار با این زبانها هنوز هم نسبتا مشکل بود و فقط متخصصین کامپیوتر قادر به استفاده از آنها بودند.

۳- زبانهای سطح بالا : دستورالعملهای این زبانها بسیار نزدیک به زبان انسانها (بطور مشخص زبان انگلیسی) می باشد و بهمین دلیل برنامه نویسی به آنها بسیار ساده تر بوده و می توان الگوریتمها را به راحتی به این زبانها تبدیل کرد. از آنجا که این زبانها به هیچ ماشین خاصی وابسته نیستند، برنامه های نوشته شده با این زبانها (تا حد زیادی) قابل حمل می باشند. مثالهایی از این زبانها عبارتند از :

- بیسیک (Basic): برای کاربردهای آموزشی
- فرترن (Fortran) : برای کاربردهای علمی و مهندسی
- پاسکال (Pascal) : برای کاربردهای آموزشی و علمی

و بالاخره زبان برنامه نویسی C که درمورد آن بیشتر صحبت خواهیم کرد. البته برنامه های نوشته شده به این زبانها ابتدا باید به زبان ماشین ترجمه شوند تا برروی کامپیوتر قابل اجرا باشند. برای ترجمه این زبانها از کامپایلرها و یا مفسرها استفاده می شود.

تاریخچه C

برای بررسی تاریخچه زبان C باید به سال ۱۹۶۷ بازگردیم که مارتین ریچاردز زبان BCPL را برای نوشتن نرم افزارهای سیستم عامل و کامپایلر در دانشگاه کمبریج ابداع کرد. سپس در سال ۱۹۷۰ کن تامپسون زبان B را بر مبنای ویژگیهای زبان

BCPL نوشت و از آن برای ایجاد اولین نسخه های سیستم عامل Unix در آزمایشگاههای بل استفاده کرد. زبان C در سال ۱۹۷۲ توسط دنیس ریچی از روی زبان B و BCPL در آزمایشگاه بل ساخته شد و ویژگیهای جدیدی همچون نظارت بر نوع داده ها نیز به آن اضافه شد. ریچی از این زبان برای ایجاد سیستم عامل Unix استفاده کرد اما بعدها اکثر سیستم عاملهای دیگر نیز با همین زبان نوشته شدند. این زبان با سرعت بسیاری گسترش یافت و چاپ کتاب "The C Programming Language" در سال ۱۹۷۸ توسط کرنیگان و ریچی باعث رشد روزافزون این زبان در جهان شد.

متاسفانه استفاده گسترده این زبان در انواع کامپیوترها و سخت افزارهای مختلف باعث شد که نسخه های مختلفی از این زبان بوجود آید که با یکدیگر ناسازگار بودند. در سال ۱۹۸۳ انستیتوی ملی استاندارد آمریکا (ANSI) کمیته ای موسوم به X3J11 را را مامور کرد تا یک تعریف فاقد ابهام و مستقل از ماشین را از این زبان تدوین نماید.در سال ۱۹۸۹ این استاندارد تحت عنوان ANSI C به تصویب رسید و سپس در سال ۱۹۹۰، سازمان استانداردهای بین المللی (ISO) نیز این استاندارد را پذیرفت و مستندات مشترک آنها تحت عنوان ANSI/ISO C منتشر گردید.

در سالهای بعد و با ظهور روشهای برنامه نویسی شی گرا نسخه جدیدی از زبان C بنام C++ توسط بیارنه استراوستروپ در اوایل ۱۹۸۰ در آزمایشگاه بل توسعه یافت. در C++ علاوه بر امکانات جدیدی که به زبان C اضافه شده است، خاصیت شی گرایی را نیز به آن اضافه کرده است.

با گسترش شبکه و اینترنت، نیاز به زبانی احساس شد که برنامه های آن بتوانند برروی هر ماشین و هر سیستم عامل دلخواهی اجرا گردد. شرکت سان مایکروسیستمز در سال ۱۹۹۵ میلادی زبان Java را برمبنای C و C++ ایجاد کرد که هم اکنون از آن در سطح وسیعی استفاده می شود و برنامه های نوشته شده به آن برروی هر کامپیوتری که از Java پشتیبانی کند(تقریبا تمام سیستمهای شناخته شده) قابل اجرا می باشد. شرکت مایکروسافت در رقابت با شرکت سان، در سال ۲۰۰۲ زبان جدیدی بنام C# (سی شارپ) را ارائه داد که رقیبی برای Java بشمار می رود.

برنامه نویسی ساخت یافته

در دهه ۱۹۶۰ میلادی توسعه نرم افزار دچار مشکلات عدیده ای شد. در آن زمان سبک خاصی برای برنامه نویسی وجود نداشت و برنامه ها بدون هیچگونه ساختار خاصی نوشته می شدند. وجود دستور پرش (goto) نیز مشکلات بسیاری را برای فهم و درک برنامه توسط افراد دیگر ایجاد می کرد، چرا که جریان اجرای برنامه مرتبا دچار تغییر جهت شده و دنبال کردن آن دشوار می گردید. لذا نوشتن برنامه ها عملی بسیار زمان بر و پرهزینه شده بود و معمولا اشکال زدایی, اعمال تغییرات و گسترش برنامه ها بسیار مشکل بود. فعالیتهای پژوهشی در این دهه باعث بوجود آمدن سبک جدیدی از برنامه نویسی بنام روش ساختیافته گردید؛ روش منظمی که باعث ایجاد برنامه هایی کاملا واضح و خوانا گردید که اشکال زدایی و خطایابی آنها نیز بسیار ساده تر بود.

اصلی ترین نکته در این روش عدم استفاده از دستور پرش (goto) است. تحقیقات بوهم و ژاکوپینی نشان داد که می توان هر برنامه ای را بدون دستور پرش و فقط با استفاده از ۳ ساختار کنترلی ترتیب، انتخاب و تکرار نوشت.

ساختار ترتیب، همان اجرای دستورات بصورت متوالی (یکی پس از دیگری) است که کلیه زبانهای برنامه نویسی در حالت عادی بهمان صورت عمل می کنند.
ساختار انتخاب به برنامه نویس اجازه می دهد که براساس درستی یا نادرستی یک شرط، تصمیم بگیرد کدام مجموعه از دستورات اجرا شود.
ساختار تکرار نیز به برنامه نویسان اجازه می دهد مجموعه خاصی از دستورات را تا زمانیکه شرط خاصی برقرار باشد، تکرار نماید.
هر برنامه ساختیافته از تعدادی بلوک تشکیل می شود که این بلوکها به ترتیب اجرا می شوند تا برنامه خاتمه یابد(ساختار ترتیب). هر بلوک می تواند یک دستورساده مانند خواندن، نوشتن یا تخصیص مقدار به یک متغیر باشد و یا اینکه شامل دستوراتی باشد که یکی از ۳ ساختار فوق را پیاده سازی کنند. نکته مهم اینجاست که درمورد دستورات داخل هر بلوک نیز همین قوانین برقرار است و این دستورات می توانند از تعدادی بلوک به شرح فوق ایجاد شوند و تشکیل ساختارهایی مانند حلقه های تودرتو را دهند.
نکته مهم اینجاست که طبق قوانین فوق یک حلقه تکرار یا بطور کامل داخل حلقه تکرار دیگر است و یا بطور کامل خارج آن قرار می گیرد و هیچگاه حلقه های روی هم افتاده نخواهیم داشت.
از جمله اولین تلاشها در زمینه ساخت زبانهای برنامه نویسی ساختیافته، زبان پاسکال بود که توسط پروفسور نیکلاس ویرث در سال ۱۹۷۱ برای آموزش برنامه نویسی ساختیافته در محیطهای آموزشی ساخته شد و بسرعت در دانشگاهها رواج یافت. اما بدلیل نداشتن بسیاری از ویژگیهای مورد نیاز مراکز صنعتی و تجاری در بیرون دانشگاهها موفقیتی نیافت.
کمی بعد زبان C ارائه گردید که علاوه بر دارا بودن ویژگیهای برنامه نویسی ساختیافته بدلیل سرعت و کارایی بالا مقبولیتی همه گیر یافت و هم اکنون سالهاست که بعنوان بزرگترین زبان برنامه نویسی دنیا شناخته شده است.

مراحل اجرای یک برنامه C

برای اجرای یک برنامه C ابتدا باید آن را نوشت. برای اینکار می توان از هر ویرایشگر متنی موجود استفاده کرد و سپس فایل حاصل را با پسوند .C ذخیره نمود (فایلهای C++ با پسوند .CPP ذخیره می گردند). به این فایل، کد مبدا (source code) گفته می شود. مرحله بعدی تبدیل کد مبدا به زبان ماشین است که به آن کد مقصد (object code) گفته می شود. همانطورکه قبلا نیز گفته شد برای اینکار از یک برنامه مترجم بنام کامپایلر استفاده می شود. کامپایلرهای متعددی برای زبان C توسط شرکتهای مختلف و برای سیستم عاملهای مختلف نوشته شده است که می توانید برحسب نیاز از هریک از آنها استفاده نمایید. اما هنوز برنامه برای اجرا آماده نیست. معمولا برنامه نویسان از در برنامه های خود از یک سری از کدهای از پیش آماده شده برای انجام عملیات متداول (مانند محاسبه جذر و یا سینوس) استفاده می کنند که برنامه آنها قبلا نوشته و ترجمه شده است. این برنامه ها یا در قالب کتابخانه های استاندارد توسط شرکتهای ارائه کننده نرم افزار عرضه شده است و یا توسط دیگر همکاران برنامه نویس اصلی نوشته و در اختیار وی قرار داده شده است. در این مرحله باید کد مقصد برنامه اصلی با کدهای مربوط به این برنامه های کمکی پیوند زده شود. برای اینکار نیاز به یک پیوند زننده (Linker) داریم و نتیجه این عمل یک فایل قابل اجرا خواهد بود (در ویندوز این فایل پسوند .EXE خواهد داشت). مرحله بعدی اجرای برنامه و دادن ورودیهای لازم به آن و اخذ خروجیها می باشد. در شکل زیر این مراحل نشان داده شده اند.

مسلما طی مراحل بالا برای اجرای هر برنامه زمانبر می باشد، بهمین دلیل اکثر تولید کنندگان کامپایلرها، محیطهایی را برای برنامه نویسی ارائه کرده اند که کلیه مراحل بالا را بطور اتوماتیک انجام می دهند.

به این محیطها IDE (Integrated Development Environment) یا محیط مجتمع توسعه نرم افزار گفته می شود. این محیطها دارای یک ویرایشگر متن می باشند که معمولا دارای خواص جالبی همچون استفاده از رنگهای مختلف برای نشان دادن اجزای مختلف برنامه مانند کلمات کلیدی، و یا قابلیت تکمیل اتوماتیک قسمتهای مختلف برنامه می باشد. پس از نوشتن برنامه و با انتخاب گزینه ای مانند Run کلیه عملیات فوق بطور اتوماتیک انجام شده و برنامه اجرا می گردد. علاوه براین، این محیطها معمولا دارای امکانات اشکالزدایی برنامه (Debug) نیز می باشند که شامل مواردی همچون اجرای خط به خط برنامه و یا دیدن محتویات متغیرها در زمان اجرا است. چند محیط معروف برنامه نویسی عبارتند از :

Borland C++ 3.1 برای محیط DOS
Borland C++ از نسخه ۴ به بالا برای Windows
Microsoft Visual C++ برای محیط Windows
Borland C++ Builder برای محیط Windows

برای شروع ما از محیط Borland C++ 3.1 تحت Dos که نحوه کار ساده تری نسبت به سایرین دارد استفاده می کنیم.
پس از نصب این نرم افزار، برنامه BC.exe را اجرا کنید تا وارد محیط borland c شوید
همانطور که می بینید، این محیط از ۳ قسمت اصلی تشکیل شده است :
- **بخش ویرایش برنامه :** بخش آبی رنگ وسط می باشد که در حقیقت یک ویرایشگر است که برنامه در آن تایپ می شود. همانطور که می بینید در این ویرایشگر از رنگهای مختلف برای نشان دادن قسمتهای مختلف برنامه استفاده می شود. مثلا برای کلمات کلیدی از رنگ سفید استفاده شده است.
- **بخش منوهای کاری :** این بخش که در قسمت بالا واقع شده است،- حاوی تعدادی منو (گزینه) برای انجام وظایف مختلف است. خلاصه این عملیات عبارتند از :

- منوی File : عملیاتی مانند باز کردن و یا ذخیره یک برنامه
- منوی Edit : عملیات ویرایش مانند حذف،o کپی و یا چسباندن یک قسمت از برنامه
- منوی Search : جستجوی و یا تعویض یک متن در برنامه
- منوی Run : اجرای برنامه بصورت کامل یا دستور به دستور
- منوی Compile : عملیات مربوط به کامپایل و پیوند برنامه
- منوی Debug : عملیات مربوط به اشکالزدایی مانند دیدن مقادیر متغیرها در زمان اجرا
- منوی Project : عملیات مربوط به مدیریت برنامه هایی که شامل چندین فایل مستقل هستند (پروژه)
- منوی Options : عملیات مربوط به تنظیمات سیستم مانند نحوه کامپایل و یا رنگ پیش فرض محیط
- منوی Windows : عملیات مربوط به پنجره های باز فعلی (مربوط به چندین برنامه یا نمایش متغیرها و ...)

خطاهای برنامه نویسی

بنظر می رسد خطاها جزء جداناپذیر برنامه ها هستند. بندرت می توان برنامه ای نوشت که در همان بار اول بدرستی و بدون هیچگونه خطایی اجرا شود. اما خطاها از لحاظ تاثیری که بر اجرای برنامه ها می گذارند، متفاوتند. گروهی ممکن است باعث شوند که از همان ابتدا برنامه اصلا کامپایل نشود و گروه دیگر ممکن است پس از گذشت مدتها و در اثر دادن یک ورودی خاص به برنامه، باعث یک خروجی نامناسب و یا یک رفتار دور از انتظار (مانند قفل شدن برنامه) شوند. بطور کلی خطاها به دو دسته تقسیم می شوند:

خطاهای نحوی (خطاهای زمان کامپایل): این خطاها در اثر رعایت نکردن قواعد دستورات زبان C و یا تایپ اشتباه یک دستور بوجود می آیند و در همان ابتدا توسط کامپایلر به برنامه نویس اعلام می گردد. برنامه نویس باید این خطا را رفع کرده و سپس برنامه را مجددا کامپایل نماید. لذا معمولا این قبیل خطاها خطر کمتری را در بردارند.

خطاهای منطقی (خطاهای زمان اجرا): این دسته خطاها در اثر اشتباه برنامه نویس در طراحی الگوریتم درست برای برنامه و یا گاهی در اثر درنظر نگرفتن بعضی شرایط خاص در برنامه ایجاد می شوند. متاسفانه این دسته خطاها در زمان کامپایل اعلام نمی شوند و در زمان اجرای برنامه خود را نشان می دهند. بنابراین، این خود برنامه نویس است که پس از نوشتن برنامه باید آن را تست کرده و خطاهای منطقی آن را پیدا کرده و رفع نماید. متاسفانه ممکن است یک برنامه نویس خطای منطقی برنامه خود را تشخیص ندهد و این خطا پس از مدتها و تحت یک شرایط خاص توسط کاربر برنامه کشف شود. بهمین دلیل این دسته از خطاها خطرناکتر هستند. خود این خطاها به دو دسته تقسیم می گردند:

a. خطاهای مهلک: در این دسته خطاها کامپیوتر بلافاصله اجرای برنامه را متوقف کرده و خطا را به کاربر گزارش می کند. مثال معروف این خطاها خطای تقسیم بر صفر می باشد.

b. خطاهای غیرمهلک: در این دسته خطا اجرای برنامه ادامه می یابد ولی برنامه نتایج اشتباه تولید می نماید. بعنوان مثال ممکن است دراثر وجود یک خطای منطقی در یک برنامه حقوق و دستمزد حقوق کارمندان اشتباه محاسبه شود و تا مدتها نیز کسی متوجه این خطا نشود!

با توجه به آنچه گفته شد، در می یابیم که رفع اشکال برنامه ها بخصوص خطاهای منطقی از مهمترین و مشکلترین وظایف یک برنامه نویس بوده و گاهی حتی سخت تر از خود برنامه نویسی است! بهمین دلیل است که بسیاری از شرکتها(همانند مایکروسافت) ابتدا نسخه اولیه نرم افزار خود را در اختیار کاربران قرار می دهند تا اشکالات آن گزارش شده و رفع گردد. بسیار مهم است که در ابتدا سعی کنید برنامه ای بنویسید که حداقل خطاها را داشته باشد، در گام دوم با آزمایش دقیق برنامه خود هرگونه خطای احتمالی را پیدا کنید و در گام سوم بتوانید دلیل بروز خطا را پیدا کرده و آنرا رفع نمایید. هر سه عمل فوق کار سختی بوده و نیاز به تجربه و مهارت دارد.

آخرین نکته اینکه در اصطلاح برنامه نویسی به هر گونه خطا، bug و به رفع خطا debug گفته می شود.

زبان C از قدرتمندترین زبانهاست که سالهاست در صنایع و تجارت و خیلی چیزهای دیگر مورد استفاده قرار گرفته است. زبان C یک زبان سطح میانی میباشد. یعنی نه سطح بالاست(مانند پاسکال)و نه سطح پایین (مانند زبان اسمبلی). بلکه قابلیتهای هر دو اینها را دارد. به همین دلیل هم است که زبان C برای نوشتن برنامه های سیستمی مانند سیستم عامل و کامپایلر و... بسیار مناسب است.

C++ عموماً از سه بخش تشکیل شده است:

- محیطی برای نوشتن برنامه و ویرایش آن.
- کامپایلر C++.
- کتابخانه استاندارد C++.

یک برنامه زبان C++ برای رسیدن به مرحله اجرا از شش مرحله عبور می کند.

مرحله اول : برنامه نویس، برنامه را در محیط ویرایشگر می نویسد و آن را بر روی دیسک ذخیره می کند

مرحله دوم : برنامه پیش پردازنده، خطوط برنامه را از لحاظ ایردات نگارشی بررسی می کند، و در صورت وجود اشکال در برنامه پیغام خطائی داده می شود، تا برنامه نویس نسبت به رفع آن اقدام نماید.

مرحله سوم : کامپایلر، برنامه را به زبان ماشین ترجمه می کند و آن را بر روی دیسک ذخیره می نماید.

مرحله چهارم : پیوند دهنده، کدهای زبان ماشین را، به فایلهای کتابخانه هایی که مورد استفاده قرار گرفته اند پیوند می دهد و یک فایل قابل اجرا بر روی دیسک می سازد.

مرحله پنجم : بار کننده برنامه را در حافظه قرار می دهد.

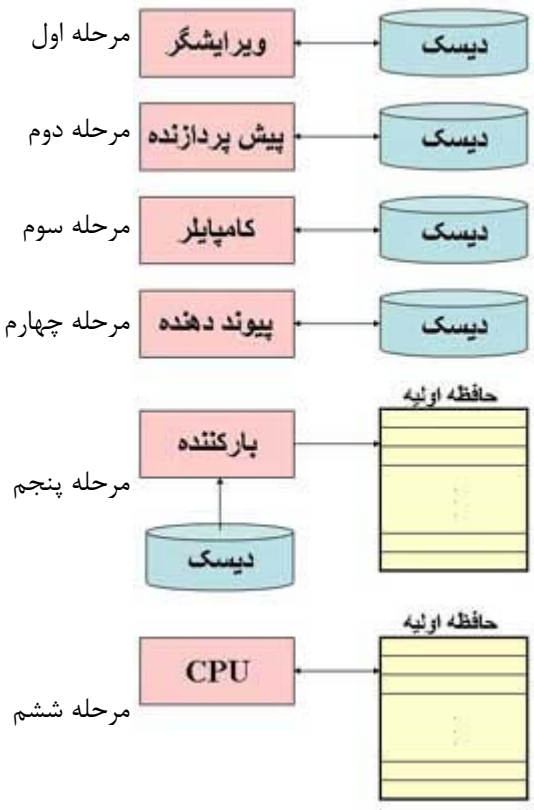
مرحله ششم : واحد پردازش مرکزی کامپیوتر دستورات برنامه را اجرا می کند.

مرحله اول

مرحله دوم

مرحله سوم

مرحله چهارم



نکته : همانطور که گفته شد پیش پردازنده ایرادات برنامه را بررسی می کند و در صورتی که برنامه مشکلی نداشت در نهایت به زبان ماشین ترجمه می شود و قابلیت اجرا پیدا می کند اما در هنگام اجرای برنامه نیز ممکن است خطایی بروز کند به عنوان مثال تقسیم بر صفر بوجود آید. این خطا قابل تشخیص توسط پیش پردازنده نیست و در زمان اجرای برنامه رخ می دهد و باعث خروج ناگهانی از برنامه می شود. به اینگونه خطاها، خطای زمان اجرا گفته می شود. تقسیم بر صفر جزء خطاهای مهلک است. خطای غیر مهلک خطایی است که اجازه اجرای ادامه برنامه را می دهد ولی ممکن است نتایج غیر صحیحی را به ما بدهد.

مفاهیم حافظه و انواع داده

همانطور که در سازمان کامپیوتر گفتیم یکی از واحدهای کامپیوتر، واحد حافظه می باشد.

این واحد که به آن **RAM** (حافظه با دسترسی تصادفی Memory Random Access) نیز می گویند، برای ذخیره موقت داده ها و دستورالعملها تا هنگامی که به آنها احتیاج شود استفاده می شود. اطلاعاتی که در RAM قرار دارند قابل پاک

شدن و جایگزین شدن با داده های دیگر است. فضایی که ما در برنامه نویسی برای متغیرها و داده ها استفاده می کنیم در RAM قرار دارد. برای درک بهتر مطلب ، واحدهای اندازه گیری حافظه را بررسی می کنیم:

Bit بیت: یک بیت عنصری الکترونیکی در کامپیوتر است که دارای دو حالت روشن (۱) و خاموش(۰) می باشد و کوچکترین واحد اطلاعاتی است.

Byte بایت: چون بیتها واحدهای اطلاعاتی کوچکی هستند و فقط می توانند دو حالت را انتقال دهند، بنابراین آنها را در واحدهای بزرگتری سازماندهی می کنند تا اطلاعات بیشتری هر بار قابل انتقال باشد. این واحد بزرگتر بایت است که واحد اصلی اطلاعات در سیستمهای کامپیوتری می باشد. هر ۸ بیت ، یک بایت را تشکیل می دهند.

از واحدهای زیر برای اندازه گیری حافظه استفاده می شود:

1 **KB** = 1024 **B** = 2^{10} **B** یک کیلو بایت

1 **MB** = 1024 **KB** = 2^{20} **B** یک مگا بایت

1 **GB** = 1024 **MB** = 2^{30} **B** یک گیگا بایت

ما در برنامه نویسی نیاز به خانه های حافظه داریم. در تعریف خانه حافظه باید نام و نوع اطلاعاتی که در آن قرار می گیرد معین شود.

```
نام متغییر        نوع داده  ;
int               i1,i2,index;
```

دستور فوق سه خانه حافظه با نامهای i1 وi2 وindex از نوع اعداد صحیح تعیین می کند، یعنی در هر کدام از خانه های حافظه فوق می توان یک عدد صحیح در بازه ۳۲۷۶۷ تا ۳۲۷۶۸- قرار داد. نوع داده int به دو بایت حافظه نیاز دارد.

نکته :

- هر دستور زبان C++ به **;** ختم می شود.
- برای نام گذاری خانه های حافظه فقط می توان از حروف، اعداد و ... استفاده کرد و نیز حرف اول نام یک متغیر باید یک حرف باشد. به عنوان مثال نامهای **test۱** و **test!num** و **mark.1** اسامی غیر مجاز می باشند.

  - توضیحات در C بین //(برای یک خط) و /* ..... */ (برای چند خط) قرار می گیرند.

- بین حروف نام متغیر نمی توان از کاراکتر فاصله استفاده کرد.
- زبان C++ دارای تعدادی کلمات کلیدی است که نمی توان از این کلمات به عنوان نام متغیر استفاده کرد. کلمات کلیدی زبان C++ عبارتند از:

char	cdecl	case	break	auto	asm
delete	default	_cs	continue	const	class
_es	enum	else	_ds	double	do
for	float	_fastcall	far	_export	extern
int	inline	if	huge	goto	friend
operator	new	near	long	_loadds	interrupt
return	register	public	protected	private	pascal
_ss	sizeof	signed	short	_seg	_saveregs
typedef	this	template	switch	struct	static
while	volatile	void	virtual	unsigned	union

-
- زبان C++ نسبت به حروف حساس است.(case sensitive) یعنی بین حروف کوچک و بزرگ تفاوت قائل می شود. در این زبان تمام کلمات کلیدی با حروف کوچک نوشته می شوند، به عنوان مثال short یک کلمه کلیدی می باشد ولی SHORT یا shoRT کلمات کلیدی نیستند. توصیه می شود که تمام برنامه های این زبان با حروف کوچک نوشته شوند.

در زبان C++ چهار نوع داده اصلی وجود دارد که عبارتند از :

۱- **char** : این نوع داده برای ذخیره داده های کاراکتری مانند 'a' ، '۱' ، '.' به کار می رود و بازه قابل قبول آن از ۱۲۸- تا ۱۲۷ می باشد. در حقیقت خانه های char نیز از نوع اعداد صحیح می باشند که یک بایت طول دارند و کد اسکی کاراکتر مورد نظر را در خود حفظ می کنند. به عنوان مثال کد اسکی کاراکتر A عدد ۶۵ می باشد.

۲- **int** : این نوع داده برای ذخیره اعداد صحیح مانند ۱۳۰۰، ۳۲۰۰۰ ، ۸۵۰- به کار می رود و بازه قابل قبول آن ۳۲۷۶۸- تا ۳۲۷۶۷ می باشد.

۳- **float** : این نوع داده برای ذخیره اعداد اعشاری مانند ۱۲,۵۲۴۱ ، ۱۵۰۱,۳- ، ۱۴۱۵,۱۲۳۴ به کار می رود و دقت آن تا ۷ رقم اعشاری می باشد.

۴- **double** : این نوع داده برای ذخیره سازی اعداد اعشاری بزرگ به کار می رود و دقت آن از float بیشتر می باشد.

باکلماتی مانند signed (علامت دار) ، unsigned (بدون علامت)، short (کوتاه) و long (بلند) انواع داده های جدیدی می توان ایجاد کرد. نوع int با هر چهار کلمه فوق می تواند مورد استفاده قرار گیرد. نوع char می تواند با signed و unsigned به کار رود و نوع double می تواند با long به کار رود. به جدول زیر توجه کنید:

نوع داده	طول داده	بازه
unsigned char	8 bits	0 to 255
char	8 bits	-128 to 127
enum	16 bits	-32,768 to 32,767
unsigned int	16 bits	0 to 65,535
short int	16 bits	-32,768 to 32,767
int	16 bits	-32,768 to 32,767
unsigned long	32 bits	0 to 4,294,967,295
long	32 bits	-2,147,483,648 to 2,147,483,647
float	32 bits	3.4 * (10**-38) to 3.4 * (10**+38)
double	64 bits	1.7 * (10**-308) to 1.7 * (10**+308)
long double	80 bits	3.4 * (10**-4932) to 1.1 * (10**+4932)

اعمال ریاضی و محاسباتی

در مبحث حافظه با انواع داده و شیوه اختصاص دادن حافظه به متغیرها آشنا شدیم حال می توانیم متغیرها را در محاسبات به کار ببریم. برای نیل به این هدف C++ عملگرهایی را در اختیار ما قرار داده است.

عملگر انتساب (=)

عملگر تساوی جهت اختصاص دادن یک مقدار به یک متغیر به کار می رود ، مانند **a = 5** که عدد ۵ را به متغیر **a** تخصیص می دهد. جزئی که در سمت چپ تساوی قرار دارد همواره باید نام یک متغیر باشد، وجزء سمت راست تساوی می تواند یک عدد، یک متغیر و یا ترکیبی از هر دو باشد. مانند: **a=b+5** ، که در ایجا حاصل **b + 5** در متغیر **a** قرار می گیرد. توجه داشته باشید که همواره مقدار سمت راست تساوی در مقدار سمت چپ قرار می گیرد. به دستورات زیر توجه کنید.

```
int a,b;
a = 10;
b = 4;
a = b;
b = 7;
```

اگر از دستورات فوق استفاده کنیم در نهایت مقدار **a** برابر **۴** و مقدار **b** برابر **۷** خواهد بود. **C++** قابلیتهای زیادی دارد یکی از این قابلیتها اینست که می توانیم چند دستور را در یک دستور خلاصه کنیم ، به عنوان مثال دستور :

```
a = 2 + (b = 5);
```

برابر است با:

```
b = 5;
a = 2 + b;
```

که هر دو عبارت در نهایت عدد **۷** را در متغیر **a** قرار می دهند.

ضمناً استفاده از عبارت زیر نیز در **C++** مجاز می باشد:

```
a = b = c = 5
```

عبارت فوق عدد **۵** را به سه متغیر **a** و **b** و **c** اختصاص می دهد.

عملگر های محاسباتی

پنج عملگر محاسباتی که قابل استفاده در زبان **C++** هستند عبارتند از:

+	جمع
-	تفریق
*	ضرب
/	تقسیم
%	باقیمانده تقسیم

تنها عملگری که ممکن است برای شما ناشناس باشد عملگر ٪ است. این عملگر باقیمانده تقسیم دو عدد صحیح را به ما می دهد، به عنوان مثال اگر از دستور زیر استفاده کنیم:

```
a = 11 % 3;
```

متغیر **a** حاوی عدد ۲ خواهد شد. چون عدد ۲ باقیمانده تقسیم ۱۱ بر ۳ می باشد.

عملگر های انتساب مرکب

عملگرهای انتسال مرکب عبارتند از += ، -= ، *= ، /= ، %= .این عملگرها دو کار را با هم انجام می دهند و در کم شدن کد نویسی به ما کمک می کنند، به جای توضیح اضافی به مثال های زیر که فهم مطلب را ساده تر می کند توجه کنید:

```
value += increase;«برابر است با»value=value+increase;
a -= 5;           «برابر است با»a = a - 5;
a /= b;           «برابر است با»a = a / b;
price*=units+1;   «برابر است با»price=price*(units+1);
x %= y * z;       «برابر است با»x = x % (y * z);
```

عملگرهای افزایش و کاهش

گونه ای دیگر از عملگرها که در کم شدن کد نویسی به ما کمک می کنند عملگر افزایش(++) و عملگر کاهش(--) می باشند. عملگر افزایش(++) یک واحد به مقدار قبلی که در متغیر بود اضافه می کند و عملگر کاهش(--) یک واحد از مقدار قبلی که در متغیر بود کم می کند.

```
++a;      a++;      a += 1;      a = a + 1;
```

هر چهار دستور فوق یک واحد به مقدار قبلی متغیر اضافه می کنند.

```
--a;      a--;      a -= 1;      a = a - 1;
```

هر چهار دستور فوق یک واحد از مقدار قبلی متغیر کم می کنند.

اگر از دستورات ++**a** و **a**++ به تنهایی استفاده کنیم فرقی ندارد که ++ قبل از متغیر قرار گیرد یا بعد از متغیر. اما اگر از ++ در کنار عملگرهای دیگر استفاده شود، اگر ++ قبل از متغیر قرار گیرد ابتدا یک واحد به متغیر اضافه شده سپس در محاسبه استفاده می شود، ولی اگر ++ بعد از متغیر قرار گیرد ابتدا متغیر در محاسبه استفاده می شود سپس یک واحد به آن اضافه می شود. همین روال برای عملگر -- نیز برقرار است. به مثال های زیر توجه کنید:

b = 3; a = b++;	b = 3; a = ++b;

در مثال سمت چپ ابتدا یک واحد به **b** اضافه می شود، یعنی **b** مقدار ۴ را می گیرد سپس عدد ۴ در **a** قرار می گیرد؛ اما در مثال سمت راست ابتدا مقدار **b** یعنی عدد ۳ در **a** قرار می گیرد سپس یک واحد به **b** اضافه می شود و مقدار ۴ را می گیرد.

در این مثال عدد ۶ در **m** قرار می گیرد:

```
a = 2;
b = 3;
m = ++a + b--;
```

b مقدار **۲** و **a** مقدار ۳ را می گیرد.

حال که با انواع عملگرهای محاسباتی آشنا شدید عبارت زیر را در نظر بگیرید.

```
y = 5 * 3 + 2 - 1 * 3 / 2;
```

مقداری که در **y** قرار می گیرد چه عددی می تواند باشد؟ **۳۰** یا **۲۴** یا **۱۵,۵** یا **۱۷,۵** . نظر شما چیست؟ شما مقدار **y** را چگونه حساب می کنید؟

کامپیوتر برای بررسی و محاسبه چنین عبارتی برای اینکه با چندین جواب مواجه نشود قواعدی را در نظر می گیرد و طبق قوانین تقدم عملگرها عمل می کند. این قوانین که مشابه قوانین جبر می باشند به ترتیب عبارتند از:

۱- عملگرهایی که درون پرانتز قرار دارند اول محاسبه می شوند. در صورتی که پرانتزها تودرتو باشند ابتدا داخلی ترین پرانتز مورد بررسی و محاسبه قرار می گیرد.

۲- اگر عبارتی حاوی * ، / و ٪ باشد پس از پرانتز این عملگرها در اولویت قرار دارند. اگر عبارتی حاوی ترکیبی از این عملگرها باشد چون این عملگرها در تقدم یکسانی نسبت به یکدیگر قرار دارند، محاسبه به ترتیب از چپ به راست انجام می شود.

۳- اعمال + و - در انتها انجام می شوند. اگر عبارتی شامل ترکیبی از این دو عملگر باشد چون این دو عملگر در تقدم یکسانی نسبت به هم هستند، محاسبه به ترتیب از چپ به راست انجام می شود.

با توجه به قواعد گفته شده حاصل عبارت فوق عدد ۱۵,۵ خواهد بود.

```
y = 5 * 3 + 2 - 1 * 3 / 2;      ----»   y = 15.5
  6   1   4   5   2   3
```

به مثال های زیر توجه کنید:

```
x = (2 + 1) * 3 + 5;            ----»   x = 14
  4     1     2   3
```

```
z = 5 % 3 * (3 + 1);          ----»  z = 8
  4   2   3    1
y = p * r % q + w / x - y;
  6   1   2   4   3   5
```

عبارات منطقی

یک عبارت منطقی، عبارتی است با مقدار درست یا نادرست. به عنوان مثال ۵ بزرگتر از ۳ است، یک عبارت منطقی است با مقدار درست و ۵ کوچکتر از ۳ است، نیز یک عبارت منطقی است اما با مقدار نادرست. در کامپیوتر نتیجه عبارات منطقی درست عدد یک و نتیجه عبارات منطقی نادرست عدد صفر خواهد بود.

ضمناً کامپیوتر هر عدد مخالف صفر را به عنوان یک عبارت منطقی درست در نظر می گیرد.

عملگرهای رابطه ای

برای مقایسه دو متغیر یا دو عبارت از عملگرهای رابطه ای استفاده می کنیم که همانطور که گفته شد دارای نتیجه درست یا نادرست می باشد. عملگرهای رابطه ای عبارتند از ==(مساوی)، =!(متفاوت)، <(بزرگتر از)، >(کوچکتر از)، =<(بزرگتر مساوی از)، =>(کوچکتر مساوی از). به مثال های زیر توجه کنید.

ضمناً به جای اینکه فقط از اعداد در عبارتهای رابطه ای استفاده کنیم می توانیم از عبارتهایی شامل متغیرها و یا ترکیبی از متغیرها و اعداد استفاده کنیم به عنوان مثال فرض کنید a = 2 و b=3 و c=6 خواهیم داشت:

```
(a==5)          -----> نادرست
(a*b>=c)        -----> درست
(b+4<a*c))      -----> نادرست
((b=2)==a))     -----> درست
```

توجه کنید که عملگر = همانند عملگر == نمی باشد. اولی عملگر انتساب است که مقدار سمت راست را در متغیر سمت چپ قرار می دهد و دیگری عملگر رابطه ای است که برابر بودن یا نبودن دو مقدار را با هم مقایسه می کند. بنابراین در عبارت ((b=2)==a)) ما ابتدا مقدار ۲ را در متغیر b قرار دادیم سپس آن را با a مقایسه کردیم، لذا نتیجه این مقایسه درست بود.