فصل اول

معرفی کامپیوتر، اینترنت و وب

اهداف

- درک مفاهیم سخت‌افزاری و نرم‌افزاری.
- درک مفاهیم اولیه تکنولوژی شی، همانند کلاس‌ها، شی‌ها، صفات، رفتار، کپسوله‌سازی و توارث.
- آشنایی با انواع زبان‌های برنامه‌نویسی.
- آشنایی با زبان‌های برنامه‌نویسی پرکاربرد.
- آشنایی با محیط توسعه برنامه C++.
- تاریخچه زبان مدل‌سازی UML.
- تاریخچه اینترنت و وب.
- استفاده از عملگرهای محاسباتی.
- تست برنامه‌های C++ بر روی لینوکس و ویندوز XP.

رئوس مطالب

۱-۱ مقدمه

به ++C خوش آمدید! ما خیلی سخت تلاش کرده‌ایم تا کتابی به رشته تحریر در آوریم که حاوی اطلاعات مفیدی بوده و سرگرم کننده نیز باشد و شما را وادار به یادگیری کند. اصولا زبان ++C زبانی است که اکثر مخاطبان آن برنامه نویسان با تجربه هستند، از اینرو سعی کرده‌ایم تا این کتاب برای افراد زیر مناسب باشد:

- افرادی که تا حدی دارای تجربه برنامه‌نویسی بوده یا فاقد تجربه هستند.
- افرادی که دارای تجربه برنامه‌نویسی بوده و مایل هستند تا درک عمیقتری نسبت به این زبان پیدا کنند.

نقطه قوت این کتاب وضوح و ترتیب خاص بیان انواع تکنیک‌های برنامه‌نویسی همانند برنامه‌نویسی ساخت‌یافته، و شی‌گرا (OOP) است. هیچ برنامه‌نویسی، برنامه‌نویسی را بخوبی یاد نخواهد گرفت مگر آنکه از ابتدا در مسیر صحیح قرار بگیرد. ما سعی کرده‌ایم که بطور واضح و خیلی سرراست به موضوعات نزدیک شویم. این کتاب حاوی مثال‌های فراوانی است که در یادگیری بسیار موثر واقع می‌شوند. در این کتاب

خروجی تمام برنامه‌ها آورده شده است. تمام برنامه‌های معرفی شده در این کتاب بر روی CD همراه کتاب موجود می باشند. فصل‌های اولیه در ارتباط با مفاهیم بنیادین کامپیوترها، برنامه‌نویسی کامپیوتر و زبان برنامه‌نویسی C++ است.

عموم مردم در مورد اعمالی که کامپیوترها انجام می‌دهند آشنایی دارند. با استفاده از این کتاب با دستوراتی آشنا خواهید شد که اعمال ویژه‌ای در یک موضوع خاص انجام می‌دهند و کامپیوتر انجام این دستورات را برعهده دارد. نرم‌افزار (دستوراتی که بصورت کد نوشته شده و کامپیوتر آن دستورات را انجام داده و در مورد آنها تصمیم‌گیری می‌کند) کامپیوتر را کنترل می‌کند (غالبا از آن بعنوان سخت‌افزار یاد می‌شود). C++ یکی از ابزارهای قدرتمند در توسعه نرم‌افزار است. در این کتاب به معرفی اصول برنامه‌نویسی در زبان C++ استاندارد شده در ایالات متحده بر اساس ANSI[1] و استاندارد جهانی ISO[2] می پردازیم.

استفاده از کامپیوترها مستلزم سعی و کوشش زیادی است. در عصری که هزینه‌ها مرتباً در حال افزایش است کاهش هزینه سخت‌افزار و نرم‌افزار که بسرعت در حال توسعه است می‌تواند جالب توجه باشد. کامپیوترهایی که 25 الی 30 سال پیش فضای زیادی از اطاق‌ها را در بر می‌گرفتند و هزینه‌های آنها بالغ بر میلیون‌ها دلار می‌شد، امروزه بر روی یک سطح تراشه سیلیکونی که به اندازه یک ناخن دست است و فقط چند دلار قیمت دارد جا گرفته‌اند. سیلیکون یکی از فراوان‌ترین فلزات موجود بر روی زمین است و عموماً جزء اجزای ترکیبی ماسه یا شن می‌باشد. تکنولوژی تراشه سیلیکونی باعث ساخته شدن کامپیوترهای مقرون به صرفه شده و بیش از ۲۰۰ میلیون کامپیوتر موجود در سرتاسر جهان از این تکنولوژی استفاده می‌کنند، که مورد استفاده عمومی در زمینه‌های تجاری، صنعتی، دولتی و امور روزمره می‌باشند.

در طول سالیان گذشته، بسیاری از برنامه‌نویسان شروع به یادگیری روشی بنام برنامه‌نویسی ساخت‌یافته کردند. دراین کتاب به آموزش برنامه‌نویسی ساخت‌یافته و همچنین مبحث جالب برنامه‌نویسی *شی‌گرا* (*Object-Oriented*) می‌پردازیم. چرا به آموزش هر دو روش می‌پردازیم؟ برنامه‌نویسی شی‌گرا کلید برنامه‌نویسی دهه بعد است. در این کتاب شی‌های متعددی ایجاد و با آنها کار خواهید کرد. در ضمن مطالعه ساختار داخلی شی‌ها متوجه خواهید شد که در ایجاد این شی‌ها از تکنیک‌های برنامه‌نویسی ساخت‌یافته استفاده شده است. همچنین دستکاری منطقی شی‌ها در بهترین حالت با برنامه‌نویسی ساخت‌یافته صورت می‌گیرد.

[1] American Natiomnal Standards Institute
[2] International Organization for Standardization

۱-۲ کامپیوتر چیست؟

کامپیوتر وسیله‌ای است که توانایی انجام محاسبات و تصمیم‌گیری‌های منطقی با سرعت میلیون‌ها و حتی بیلیون‌ها برابر سریع‌تر از یک انسان را دارد. برای مثال امروزه بیشتر کامپیوترهای شخصی می‌توانند صدها میلیون دستور در هر ثانیه انجام دهند. یک ماشین حساب بایستی تمام عمر کار کند تا بتواند همان عدد محاسبه شده توسط یک کامپیوتر شخصی را که در عرض یک ثانیه انجام داده به دست بیاورد. امروزه سوپر کامپیوترها توانایی انجام صدها بیلیون دستور در هر ثانیه را دارند در حالیکه صدها هزار نفر با استفاده از ماشین حساب این عمل را در طول یک سال می‌توانند انجام دهند. کامپیوترهایی که توانایی انجام تریلیون‌ها دستور در هر ثانیه دارند در آزمایشگاه‌های تحقیقاتی مورد استفاده می‌باشند. کامپیوترها پردازش داده‌ها را تحت کنترل تعدادی از دستورات که برنامه‌های کامپیوتری نامیده می‌شوند، انجام می‌دهند و این برنامه‌ها توسط اشخاصی بنام *برنامه‌نویس* نوشته می‌شوند.

انواع متفاوتی از قطعات همانند صفحه کلید، صفحه نمایش، دیسک‌ها، حافظه و واحدهای پردازش، که یک سیستم کامپیوتری را تشکیل می‌دهند بنام *سخت‌افزار* شناخته می‌شوند. برنامه‌های کامپیوترئی که بتوانند بر روی یک کامپیوتر اجرا شوند بنام *نرم‌افزار* شناخته می‌شوند. هزینه‌های سخت‌افزاری در سال‌های اخیر کاهش یافته و کامپیوترهای شخصی بعنوان یک ابزار مناسب در دسترس قرار گرفته‌اند، ولی متأسفانه هزینه‌های توسعه نرم‌افزار افزایش یافته و برنامه‌نویسان برنامه‌های کاربردی قویتر و پیچیده‌تری ارائه می‌کنند بدون اینکه قادر باشند تکنولوژی توسعه نرم‌افزار را بهبود بخشند. در این کتاب سعی شده است تا بتوانید روش‌های توسعه نرم‌افزار را با استفاده از کاهش هزینه‌های نرم‌افزار از طریق به کارگیری روش‌های برنامه‌نویسی ساخت‌یافته از بالا به پایین (top-down)، برنامه‌های مبتنی بر شی، برنامه‌نویسی شی‌گرا، طراحی شی‌گرا فرا بگیرید.

۱-۳ سازماندهی کامپیوتر

صرفنظر از تفاوت‌های فیزیکی، هر کامپیوتر به شش قسمت منطقی تقسیم می‌شود که عبارتند از:

۱- واحد ورودی (Input unit):

این قسمت "واحد دریافت" برای بدست آوردن اطلاعات (داده و برنامه کامپیوتری) از طریق انواع وسایل ورودی و قرار دادن این اطلاعات در دسترس سایر واحدها برای پردازش اطلاعات می‌باشد. بیشتر اطلاعات ورودی از طریق صفحه کلید و ماوس دریافت می‌شوند. در آینده ممکن است حجم زیادی از اطلاعات بفرم ویدئوی دریافت شوند.

۲- واحد خروجی (Output unit):

این قسمت وظیفه "حمل" را برعهده دارد. این واحد اطلاعات پردازش شده توسط کامپیوتر را دریافت کرده و به انواع وسایل خروجی منتقل می‌کند تا در خارج از کامپیوتر مورد استفاده قرار گیرد. بیشتر اطلاعات خروجی بر روی صفحات نمایش و چاپ بر روی کاغذ منتقل می‌شوند یا از اطلاعات خروجی برای کنترل سایر دستگاه‌ها استفاده می‌شود.

۳- واحد حافظه (Memory unit):

این قسمت "انبار" کامپیوتر محسوب می‌شود و دارای سرعت دسترسی و ظرفیت نسبتاً بالایی است. اطلاعات وارد شده از واحد ورودی در این قسمت نگهداری می‌شوند. همچنین واحد حافظه می‌تواند اطلاعات پردازش شده را در خود نگهداری کند، تا زمانیکه اطلاعات بتوانند بر روی دستگاه خروجی (به وسیله واحد خروجی) قرار گیرند. واحد حافظه اغلب به عنوان حافظه یا حافظه اولیه نامیده می‌شود.

٤- واحد محاسبه و منطق (ALU):

این قسمت "واحد ساخت" کامپیوتر است. این بخش مسئولیت انجام اعمال محاسباتی همانند جمع، تفریق، ضرب و تقسیم را برعهده دارد. همچنین شامل مکانیزم‌های تصمیم‌گیری می‌باشد. بطور مثال به کامپیوتر این امکان را می‌دهد که محتویات دو محل متفاوت از حافظه را باهم مقایسه کرده و تعیین کند که آیا برابرند یا خیر.

٥- واحد پردازش مرکزی (CPU):

این قسمت واحد "اجرائی" کامپیوتر است. این قسمت وظیفه هماهنگ کردن کامپیوتر و مسئولیت نظارت بر نحوه انجام عملیات توسط سایر قسمت‌ها را برعهده دارد. CPU به واحد ورودی اعلان می‌کند که در چه زمانی می‌بایستی اطلاعات به واحد حافظه وارد شده و به ALU اعلان می‌کند در چه زمانی اطلاعات از حافظه برداشته و بکار گرفته شوند و به واحد خروجی اعلان می‌کند که در چه زمانی اطلاعات از حافظه به واحد خروجی مشخص شده ارسال شوند.

٦- واحد ذخیره‌سازی ثانویه:

این قسمت "انبار" کامپیوتر است که دارای طول عمر زیاد و ظرفیت بالا می‌باشد. برنامه‌ها یا داده‌ها تا زمانیکه بر روی وسایل ذخیره‌سازی ثانویه (همانند دیسک‌ها) ذخیره نشوند، نمی‌توانند بدرستی بکار گرفته شوند. اطلاعات قرار گرفته بر روی واحد ذخیره‌سازی ثانویه بدفعات زیاد نسبت به حافظه می‌توانند مورد دستیابی قرار گیرند. هزینه وسایل ذخیره‌سازی ثانویه نسبت به حافظه اولیه بسیار کمتر است.

٤-١ تکامل سیستم عامل

زمانی کامپیوترها فقط می‌توانستند یک عمل یا یک وظیفه را در هر زمان انجام دهند اینحالت در کامپیوترها به عنوان پردازش *دسته‌ای تک‌کاربره* (*single user batch processing*) معروف است. کامپیوتر در هر زمان توانایی اجرای یک برنامه در زمان پردازش را دارد. در این سیستم‌ها کاربران معمولاً کارهایی که

می‌خواستند انجام دهند بر روی کارت پانچ قرار می‌دادند و به کامپیوتر مرکزی ارائه می‌کردند. کاربران اغلب ساعت‌ها و حتی روزها منتظر جواب می‌شد. نرم‌افزارهای سیستم که معروف به سیستم عامل هستند به منظور استفاده آسانتر از کامپیوترها توسعه پیدا کردند. سیستم‌های عامل قدیمی، مدیریت انتقالی بین وظایف محوله را انجام می‌دادند. هنگامی که کامپیوترها قویتر و کاراتر شدند، سیستم‌های تک کاربره در استفاده از منابع سیستم دیگر کارایی قابل قبولی نداشتند. برای مثال، بایستی تعداد متنوعی از وظایف با استفاده از اشتراک منابع برای استفاده بهینه از کامپیوتر مورد استفاده قرار گیرد که بنام *Multiprogramming* نامیده می‌شود. Multiprogramming چندین عملیات را بصورت همزمان در یک کامپیوتر انجام می‌دهد (این قابلیت با عنوان *Multiprocessing* نیز شناخته می‌شود). کامپیوتر با استفاده از اشتراک منابع در میان انواع وظایف به فعالیت خود ادامه می‌دهد. اما هنوز هم در این سیستم‌های عامل، بایستی کاربران ساعت‌ها در انتظار باقی می‌ماندند.

در دهه ۱۹۶۰ چندین گروه از صنایع و دانشگاه‌ها پیش گام توسعه سیستم‌های عامل *اشتراک زمانی* (*Timesharing*) شدند. اشتراک زمانی یک حالت خاص از Multiprogramming می‌باشد، که در آن کاربران از طریق یک ترمینال که نوعاً یک صفحه کلید و صفحه نمایش می‌باشد به کامپیوتر دسترسی داشتند. در نمونه واقعی کامپیوتری که از سیستم اشتراک زمانی استفاده می‌کند ممکن است یک دوجین یا حتی صدها کاربر بصورت مشترک از کامپیوتر استفاده کنند. کامپیوتر نمی‌تواند به درخواست‌های همزمان کاربران واکنش نشان دهد. در اینحالت کامپیوتر یک قسمت از کار یک کاربر را انجام داده و سپس سرویس را به کاربر بعدی انتقال می‌دهد. کامپیوتر این عمل را بسیار سریع انجام می‌دهد و ممکن است به چندین کاربر در هر ثانیه سرویس ارائه کند. در اینحالت کاربران گمان می‌کنند که برنامه‌ها بصورت همزمان اجرا می‌شوند. مزیت اشتراک زمانی این است که به درخواست کاربر سریعاً واکنش نشان داده می‌شود.

۱-۵ محاسبات شخصی، محاسبات توزیع شده و سرویس دهنده‌ها/ سرویس گیرنده‌ها

در سال ۱۹۷۷، کامپیوترهای اپل (Apple)، نماد محاسبات شخصی بودند. کامپیوترها به تدریج ارزان شدند تا مردم آنها را خریداری کرده و در کارهای شخصی یا تجاری مورد استفاده قرار دهند. در سال ۱۹۸۱ شرکت IBM که بزرگترین فروشنده کامپیوتر در جهان است، کامپیوترهای شخصی IBM را به بازار معرفی کرد. بسرعت محاسبات شخصی در تجارت، صنایع و مراکز دولتی وارد شد. اما این کامپیوترها هنوز هم بفرم واحدهای منفرد عمل می‌کردند. کاربران کارهای خود را بر روی سیستم خود انجام می‌دادند و سپس نتایج را بر روی دیسک منتقل می‌کردند و آنرا به اشتراک می‌گذاشتند. با اتصال چندین سیستم به یکدیگر شبکه تشکیل داده شد. شبکه‌های محلی (LAN) از این نوع سازماندهی می‌باشند. این فرآیند سبب هدایت بسوی محاسبات توزیع شده در سازماندهی محاسباتی گردید. کامپیوترهای شخصی بقدر کافی قدرت پیدا کرده

بودند که می‌توانستند محاسبات جداگانه چندین کاربر را انجام داده و وظایف ارتباطی و عبور اطلاعات بصورت الکترونیکی را فراهم نمایند.

امروزه کامپیوترهای شخصی نسبت به کامپیوترهای دهه قبل چندین میلیون برابر، قدرت بیشتر پیدا کرده‌اند. ماشین‌های قدرتمند رومیزی که *ایستگاه کاری* (*Workstation*) نامیده می‌شوند، توانایی بسیار زیادی در ارائه سرویس به کاربران با نیازهای متفاوت دارند.

اطلاعاتی که حالت اشتراکی دارند در کامپیوترهای شبکه موسوم به *سرویس‌دهنده* (*Server*) قرار می‌گیرند. این کامپیوترها اطلاعات و برنامه‌ها را در خود نگهداری می‌کنند که ممکن است توسط *س‌رویس‌گیرنده‌ها* (*Clients*) که در سرتاسر جهان توزیع شده‌اند مورد استفاده قرار گیرند، از اینرو عبارت *سرویس‌دهنده/سرویس‌گیرنده* (*Server/Client*) وارد صحنه گردید. زبان‌های C و C++ به عنوان زبان‌های برنامه‌نویسی، برای نوشتن نرم‌افزار سیستم عامل برای کامپیوترهای شبکه و کاربردهای توزیع شده Server/Client انتخاب شده‌اند، امروزه سیستم‌های عامل پرطرفدار همانند UNIX، Linux، Solaris، MacOs، Windows 2000 و Windows XP دارای قابلیت‌های فراوانی هستند که در مورد آنها صحبت خواهیم کرد.

۶-۱ تاریخچه اینترنت و www

در اواخر دهه ۱۹۶۰، پورفسور H.M.Deitel از دانشجویان فارغ‌التحصیل دانشگاه MIT بود. پروفسور Deitel بر روی پروژه Mac[1] دانشگاه MIT که سبب پیدایش ARPA[2] شده کارکرده است. ARPA میزبان کنفرانسی شد که میهمانان آن مجموعه‌ای از پدید آورندگان ARPA در دانشگاه Illinois بودند و در آن به بحث و بررسی مباحث مختلف پرداخته شد. در این کنفرانس طرح شبکه کردن، کامپیوترهای اصلی دانشگاه‌های سهیم در پروژه ARPA مطرح گردید. کامپیوترهای متصل شده با خطوط ارتباطی با سرعت کمتر 56 kbps کار می‌کردند (kbps 1 معادل، 1024 بیت در هر ثانیه است)، در آن زمان بیشتر مردم (کسانی که به شبکه دسترسی داشتند) از طریق خطوط تلفن با سرعت 110 بیت در هر ثانیه به کامپیوترها متصل می‌شدند. در این کنفرانس در مورد مباحث گوناگونی صحبت شد و سرانجام ARPA به ARPAnet تغییر نام داد که پدر بزرگ اینترنت است.

گروه‌های مختلف بر روی طرح اولیه به روش‌های گوناگونی کار کردند. اگر چه ARPAnet امکان تحقیق به محققان خود را بر روی کامپیوترهای شبکه می‌داد، اما اصلی‌ترین مزیت آن بهبود قابلیت برای ارتباط آسان و سریع بود که امروزه بنام پست الکترونیکی (e.mial) شناخته می‌شود. این قابلیت امروزه نیز در

۱- هم اکنون آزمایشگاه علوم کامپیوتر و منزلگاه کنسرسیوم World Wide Web است.

۲- Advanced Research Project Agency of Department

اینترنت در زمینه پست الکترونیکی و انتقال فایل در میان میلیون‌ها نفر در سرتاسر جهان بکار گرفته می‌شود. شبکه طراحی شدهٔ در آن زمان فاقد یک کنترل مرکزی بود. به این دلیل که اگر بخشی از شبکه از مدار خارج می‌شد، مابقی بخش‌های شبکه هنوز هم قادر به ارسال و دریافت بسته‌های اطلاعاتی از طریق مسیرهای جایگزین بودند.

پروتکل (مجموعه قوانین) برقراری ارتباط بر روی شبکه ARPAnet امروزه بنام *TCP* (*Transmission Control Protocol*) شناخته می‌شود. این پروتکل سبب می‌شود که پیغام‌ها با دقت و به درستی از سوی فرستنده به گیرنده ارسال شوند. برای تشخیص بخش‌های مقابل، شبکه ARPAnet موجب توسعه پروتکل اینترنت یا IP (*Internet Protocol*) شد که بدنبال آن واقعیت "شبکه‌ای از شبکه‌ها" تحقق پیدا کرد و معماری جاری در اینترنت شد. مجموعه‌ای از این پروتکل‌ها بعنوان TCP/IP شناخته می‌شود.

وب گسترده جهانی (*World Wide Web*) به کاربران کامپیوتر امکان می‌دهد تا مستندات مبتنی بر مولتی مدیا را یافته و به آنها نگاه کنند (مستنداتی متشکل از متن، گرافیک، انیمیشن، صوت یا ویدئو). در سال ۱۹۸۹، پورفسور Tim Berners-Lee از گروه CERN (سازمان اروپا در زمینه تحقیقات هسته‌ای) شروع به توسعه تکنولوژی، در زمینه به اشتراک گذاری اطلاعات از طریق فوق لینک‌ها در مستندات متنی کرد. اینکار بر مبنای زبان جدیدی بنام SGML صورت گرفت (استانداردی برای تبادل اطلاعات)، که Berners-Lee آنرا HTML[1] (*HyperText Markup Language*) نامید. البته Lee پروتکل‌های ارتباطی برای سیستم اطلاعاتی فوق متن جدید خود نوشت که او از آن بعنوان World Wide Web نام برد. امروزه اینترنت و WWW از بخش‌های اصلی و ضروری در زندگی انسانها شده‌اند. در گذشته، بیشتر برنامه‌های کامپیوتری روی یک سیستم منفرد به اجرا در می‌آمدند (کامپیوترهای که به کامپیوتر دیگری متصل نبودند). امروزه برنامه‌های کامپیوتری می‌توانند برای برقراری ارتباط مابین میلیون‌ها کامپیوتر نوشته شوند.

۷-۱ زبان ماشین، زبان اسمبلی و زبان سطح بالا

برنامه‌نویس دستورات خود را می‌تواند در انواع متفاوتی از زبان‌های برنامه‌نویسی بنویسد. تعدادی از این زبان‌ها به صورت مستقیم توسط کامپیوتر درک می‌شوند و تعداد دیگری نیاز به ترجمه دارند تا قابل فهم برای کامپیوتر شوند. امروزه صدها زبان کامپیوتری مورد استفاده می‌باشند، که می‌توان آنها را به سه دسته تقسیم کرد:

۱- زبان ماشین (Machine Languages)

۲- زبان اسمبلی (Assembly Languages)

۱- زبان نشانه‌گذاري فوق متن

۳- زبان‌های سطح بالا (High-Level Languages)

هر کامپیوتری می‌تواند بطور مستقیم فقط زبان ماشین خود را درک کند. زبان ماشین، زبان ذاتی و منحصر بفرد یک کامپیوتر می‌باشد و به هنگام طراحی سخت‌افزار کامپیوتر تعریف می‌شود. زبان ماشین عموماً شامل رشته‌ای از اعداد است و موجب می‌شود که کامپیوتر عملیات اصلی را که در ارتباط با خود است درهر بار راه‌اندازی اجرا نماید. زبان ماشین، وابسته به ماشین می‌باشد (زبان ماشین یک دستگاه فقط بر روی همان نوع از ماشین اجرا می‌شود). درک زبان ماشین برای انسان طاقت فرسا و بسیار مشکل است. برای مثال می‌توانید به زبان ماشین که در قسمت زیر آورده شده توجه کنید، این برنامه اضافه کار را بر مبنای حقوق محاسبه و نتیجه بدست آمده را در grosspay ذخیره می‌کند.

```
+1300042774
+1400593419
+1200274027
```

زمانیکه کامپیوترها مورد استفاده عموم قرار گرفتند، مشخص شد برنامه‌نویسی زبان ماشین برای بسیاری از برنامه‌نویسان خسته کننده و ملالت‌آور است. در عوض، بکار بردن رشته‌ای از اعداد که کامپیوتر بتواند بصورت مستقیم آنرا درک کند، برنامه‌نویسان از عبارات کوتاه شده زبان انگلیسی برای فهماندن عملیات ابتدایی به کامپیوتر استفاده کردند. این عبارات مخفف شده شبیه زبان انگلیسی، مبنای زبان اسمبلی هستند. برنامه‌های مترجم بنام اسمبلر مشهور می‌باشند که زبان اسمبلی را بزبان ماشین ترجمه می‌کنند. قطعه برنامه‌ای که در قسمت پایین آورده شده همان عملیات بالا را انجام می‌دهد منتهی با استفاده از زبان اسمبلی که نسبت به زبان ماشین از وضوح (قابل فهم) بیشتری برخوردار است.

```
LOAD BASEPAY
ADD OVERPAY
STORE GROSSPAY
```

اگر چه این کد برای انسان از وضوح بیشتری برخوردار است اما برای کامپیوتر تا زمانی که به زبان ماشین ترجمه نشود معنی ندارد. زبان اسمبلی باعث افزایش سرعت برنامه‌نویسی شد اما هنوز هم مستلزم دستورات فراوانی برای انجام یک عمل ساده بود. برای افزایش سرعت برنامه‌نویسی زبان‌های سطح بالا توسعه پیدا کردند. که با استفاده از یک عبارت می‌توانند وظایف و اعمال وسیع‌تری را انجام دهند. برنامه‌های مترجم که وظیفه تبدیل زبان‌های سطح بالا به زبان ماشین را برعهده دارند کامپایلر نامیده می‌شوند. زبان‌های سطح بالا این امکان را به برنامه‌نویس می‌دهند که دستورات مورد نیاز خود را تقریباً مانند زبان انگلیسی و عملیات ریاضی را به صورت روزمره بنویسد.

```
grossPay = basePay + overTimePay
```

واضح است که زبان‌های سطح بالا نسبت به زبان‌های ماشین یا اسمبلی از محبوبیت بیشتری در نزد برنامه‌نویسان برخوردارند. ویژوال بیسیک به صورت وسیع مورد استفاده می‌باشد و از جمله زبان‌های سطح بالا به شمار می‌آید. عمل کامپایل کردن زبان سطح بالا به زبان ماشین می‌تواند وقت زیادی از کامپیوتر را بگیرد. برنامه‌های مفسر (Interpreter) توسعه یافته می‌توانند به صورت مستقیم برنامه‌های زبان‌های سطح بالا را بدون نیاز به کامپایل به زبان ماشین تبدیل کنند. اگر چه برنامه‌های مفسر نسبت به برنامه‌های کامپایلر آهسته‌تر عمل می‌کنند، اما برنامه‌های مفسر فوراً شروع به فعالیت می‌کنند بدون اینکه تأخیرهای ذاتی از عمل کامپایل را در خود داشته باشند.

۸-۱ تاریخچه C و C++

زبان C++ توسعه یافته زبان C است که از دو زبان برنامه‌نویسی قبلی، بنام‌های BCPL و B منشعب شده است. زبان BCPL در سال 1967 توسط Martin Richards بعنوان زبانی برای نوشتن نرم‌افزار سیستم‌های عامل و کامپایلرها طراحی شده بود.آقای Ken Thompson بسیاری از ویژگیهای زبان B خود را از BCPL اقتباس کرد و از B برای ایجاد نسخه‌های اولیه سیستم عامل UNIX در آزمایشگاهای Bell در سال 1970 بر کامپیوتر DEC PDP-7 استفاده شد. هر دو زبان BCPL و B از نوع زبانهای بدون نوع (typeless) هستند، به این معنی که هر ایتم داده یک "کلمه" در حافظه اشغال می‌کند و مسئولیت رسیدگی به دادها به عهده برنامه نویس خواهد بود.

زبان C از زبان B و توسط Dennis Ritchie در آزمایشگاهای شرکت Bell توسعه یافت و برای اولین بار بر روی کامپیوتر DEC PDP-11 در سال 1972 پیاده سازی گردید. زبان C از مفاهیم اساسی BCPL و B سود می‌برد در حالیکه دارای قابلیت تعریف *نوع داده* (*data type*) و ویژگیهای دیگر بود. زبان C در بدو شروع بکار بطور گسترده‌ای بعنوان زبان توسعه‌دهنده سیستم عامل UNIX بکار گرفته شد. امروزه، اکثر سیستم‌های عامل توسط زبان‌های C یا C++ یا ترکیبی از هر دو نوشته شده‌اند. هم اکنون C بر روی بیشتر کامپیوتر پیدا می‌شود. زبان C، زبان مستقل از سخت‌افزار است. اگر در زمان طراحی دقت کافی بخرج داده شود، می‌توان برنامه‌های C را که از قابلیت حمل (portable) برخودار هستند بر روی اکثر کامپیوترها به اجرا در آورد.

در اواخر دهه 1970، زبان C توسعه پیدا کرد و بنام‌های "C تجاری"،"C کلاسیک"، و " Keringhan and Ritchie C" معروف شد. کتاب "زبان برنامه‌نویسی C" که توسط انتشارات Prentice-Hall در سال 1978 منتشر شد تاثیر بسیار زیادی در گسترش این زبان بازی کرد.

بکارگیری زبان C بر روی مجموعه وسیعی از انواع کامپیوترها (گاهی اوقات از این مطلب بعنوان platform یاد می‌شود) موجب شده تا نسخه‌های متعدی از آن بوجود آید (متاسفانه). با اینکه این نسخه‌ها شبیه هم بودند، اما گاهی اوقات عدم سازگاری مابین آنها رخ می‌داد. این عدم سازگاری یکی از جدی‌ترین مشکلات برنامه‌نویسانی بود که می‌خواستند برنامه‌های قابل حملی بنویسند که بر روی چندین پلات‌فرم به اجرا درآید. در چنین وضعیتی وجود یک نسخه استاندارد C احساس گردید. در سال 1983، کمیته استاندارد X3J11 که تحت نظارت کمیته ملی استاندارد کامپیوتر و پردازش اطلاعات آمریکا[1](ANSCIP) بوجود آمده بود، یک تعریف غیر مبهم از زبان مستقل از ماشین ارائه کرد. در سال 1989، استاندارد رشد پیدا کرده بود و ANSI با همراهی ISO[2] زبان C را در سرتاسر جهان استاندارد کردند. مستند استاندارد در سال 1990 منتشر شد و از آن بعنوان ANSI/ISO 9899:1990 یاد می‌شود. ویرایش دوم کتاب "زبان برنامه‌نویسی C" در سال 1988، چاپ و به نام ANSI C، نامیده شد که هم اکنون در سرتاسر جهان بکار گرفته می‌شود.

قابلیت حمل

بدلیل اینکه زبان C++ استاندارد شده است، مستقل از سخت افزار است و بطرز گسترده ای در دسترس می باشد، غالبا برنامه‌های نوشته شده به زبان C++ را می‌توان با کمی اصلاح و حتی بدون هیچ گونه تغییری، بر روی انواع مختلفی از سیستم‌های کامپیوتری به اجرا در آورد.

زبان C++ بسط یافته زبان C است،که توسط Bjarne Stroustryp در اوایل 1980 و در آزمایشگاه‌های Bell ابداع گردید. زبان C++ حاوی برخی از ویژگیهای C است، اما مهمترین ویژگی و قابلیت این زبان در برنامه‌نویسی شی گرا بودن آن است.

این ویژگی، انقلابی در جامعه نرم‌افزاری بوجود آورد. در چنین حالتی تولید نرم‌افزار بسرعت، با دقت و اقتصادی‌تر صورت می‌گیرد. شی‌ها *کامپونت‌های* نرم‌افزاری با قابلیت *استفاده مجدد* هستند که ایتم‌های حقیقی در دنیا را مدل سازی می‌کنند. توسعه دهنده‌گان نرم‌افزار به این نتیجه رسیده‌اند که بهره گیری از روش مدولار و طراحی شی‌گرا و پیاده سازی به این روش‌ها می‌تواند در بهره‌وری گروه‌های توسعه‌دهنده نرم‌افزاری در مقایسه با تکنیک‌های متداول قدیمی‌تر، همانند برنامه‌نویسی ساخت‌یافته بسیار موثر باشد. درک برنامه‌های شی‌گرا، اصلاح و تغییر آنها راحتر است.

زبانهای شی‌گرای متعددی تا بدین روز پدید آمده‌اند، زبان‌های همانند Smalltalk، که توسط PARC[3] ابداع شده است. زبان Smalltalk یک زبان شی‌گرای محض است، که هر چیزی در آن یک شی می‌باشد. از

[1] American NationalStandards Committee on Computers and Information Processing (X3)
[2] International Standards Organization
[3] Xerox's Palo Alto Research Center

سوی دیگر C++ از جمله زبان‌های هیبرید می‌باشد، به این معنی که می‌توان در این زبان برنامه‌ای نوشت که شبیه C یا شی‌گرا باشد، یا اینکه ترکیبی از هر دو حالت را در بر گیرد.

۹-۱ کتابخانه استاندارد C++

برنامه‌های C++ متشکل از قسمت‌های بنام کلاس‌ها و توابع هستند. می‌توانید برحسب نیاز هر قسمت را به فرم یک برنامه C++ برنامه‌نویسی کنید. با این همه, اکثر برنامه‌نویسان C++ از مزیت کلکسیون‌های غنی از کلاس‌ها و توابع در کتابخانه استاندارد C++ بهره می‌برند. از اینرو, واقعاً دو بخش آموزشی در جهان C++ وجود دارد. بخش اول یادگیری خود زبان C++ است و بخش دوم نحوه استفاده از کلاس‌ها و توابع موجود در کتابخانه استاندارد C++. در سرتاسر این کتاب در ارتباط با تعدادی از این کلاس‌ها و توابع صحبت خواهیم کرد. خواندن کتاب The Standard C Library, نوشته PJ.Plauger را برای کسانی که می‌خواهند درک عمیقی از توابع کتابخانه ANSI C که در برگیرنده C++ نیز می‌باشد و در آن مطالبی در زمینه نحوه پیاده‌سازی و نحوه استفاده از این توابع در ایجاد کدهای قابل حمل وجود دارد, توصیه می‌کنیم. معمولاً کتابخانه‌های استاندارد توسط سازندگان کامپایلر تدارک دیده می‌شوند. البته کتابخانه‌های کلاس با مقاصد خاص نیز وجود دارند که توسط سازندگان مستقل نرم‌افزار تهیه می‌شوند.

مهندسی نرم‌افزار

از روش «ساخت- بلوکی» در ایجاد برنامه‌ها استفاده کنید. از اختراع مجدد چرخ اجتناب کنید. تا حد امکان از قسمت‌های موجود استفاده کنید. به این روش, استفاده مجدد از نرم‌افزار می‌گویند, که یکی از اهداف برنامه‌نویسی شی‌گرا می‌باشد.

مهندسی نرم‌افزار

به هنگام برنامه‌نویسی در C++, از بلوک‌های زیر استفاده خواهید کرد: کلاس‌ها و توابع موجود در کتابخانه استاندارد C++, کلاس‌ها و توابعی که خودتان ایجاد می‌کنید، کلاس‌ها و توابعی که در کتابخانه‌های دیگر وجود دارند.

در کل کتاب با نکاتی در ارتباط با مهندسی نرم‌افزار مشاهده خواهید کرد که به توصیف مفاهیم موثر در بهبود معماری و کیفیت سیستم نرم‌افزاری می‌پردازند. همچنین به برجسته کردن نکات دیگری شامل برنامه‌نویسی ایده‌آل (برای کمک به شما در نوشتن برنامه‌هایی که واضح بوده, درک آنها آسانتر باشد, نگهداری و استفاده راحت‌تری داشته باشند و بتوان خطاهای آنها را به راحتی یافته و اصلاح کرد), خطاهای برنامه‌نویسی (مشکلاتی که باید برای اجتناب از آنها هوشیار باشید), کارایی (تکنیک‌های برنامه‌نویسی که سبب اجرای سریعتر و مصرف کمتر حافظه می‌شوند), قابلیت حمل (تکنیک‌هایی که به کمک آنها می‌توان برنامه‌هایی نوشت که با کمی تغییر یا هیچ تغییری, بر روی انواع کامپیوترها اجرا شوند) و اجتناب از خطا

(تکنیک‌های حذف خطا از برنامه‌ها و روش نوشتن برنامه‌های بدون خطا از همان ابتدای کار) است. تمام این نکات فقط نقش راهنما و هدایت‌کننده دارند.

یکی از مزایا ایجاد توابع و کلاس‌های متلعق بخود این است که دقیقاً از نحوه عملکرد آنها مطلع هستیم و می‌توانیم به بررسی کد C++ آنها بپردازیم. عیب این روش در زمان‌بر بودن و پیچیدگی است که در طراحی, توسعه و نگهداری توابع و تلاش‌های جدید بوجود می‌آید.

کارائی

با استفاده از توابع و کلاس‌های کتابخانه استاندارد C++, بجای نوشتن نسخه‌هایی از آنها, می‌توانید کارایی برنامه را افزایش دهید چرا که آنها بدقت و با توجه به کارایی نوشته شده‌اند. همچنین این تکنیک زمان توسعه و ایجاد برنامه را کوتاهتر می‌سازد.

قابلیت حمل

قابلیت حمل با استفاده از توابع و کلاس‌های کتابخانه استاندارد C++ بجای نوشتن توابع و کلاس‌های متعلق به خود, قابلیت حمل برنامه افزایش و بهبود می‌یابد چرا که این توابع و کلاس‌های در هر پیاده‌سازی C++ وجود دارند.

۱۰-۱ تاریخچه جاوا

بسیاری از افراد بر این باورند که آینده در اختیار ریزپردازنده‌های هوشمند خواهد بود. با در نظر گرفتن این مطلب، شرکت Sun Microsystems یک تیم تحقیقاتی با نام کد Green در سال 1990 تاسیس کرد. نتیجه پروژه که مبتنی بر زبان‌های C و C++ بود توسط James Gosling بنام Oak نامیده شد. پس ملاقات افراد تیم Sun در یک کافه محلی بر سر نام *جاوا* (*Java*) به توافق رسیدند.

اما پروژه Green با مشکلاتی مواجه شد. بازار قطعات هوشمند مطابق با آنچه که شرکت Sun انتظار داشت رشد نکرد. بدتر از آن قراردادی که شرکت Sun بر سر آن رقابت می‌کرد به یک شرکت دیگر واگذار گردید. از اینرو پروژه در وضعیت خطرناک لغو قرار گرفت. از بخت بلند، در سال 1993، گشت و گذار در *وب گسترده جهانی(WWW)* از محبوبیت بسیار زیادی در بین مردم برخوردار شده بود. بنابر این اهالی Sun بلافاصله متوجه کاربرد جاوا و پتانسیل‌های آن در ایجاد *محتویات دینامیک* بر صفحات وب شدند.

شرکت Sun در ماه می 1995 به عرضه تجاری جاوا پرداخت. بلافاصله، جاوا نظر بسیاری از مراکز تجاری را بخود جلب کرد، چراکه علاقه عجیبی به وب گسترده جهانی در نزد مردم پیدا شده بود. هم اکنون از جاوا برای ایجاد صفحات وب با قابلیت دینامیکی و تعاملی، توسعه برنامه‌های کاربردی در مقیاس گسترده، به منظور افزایش کارائی و عملکرد سرویس‌دهنده‌های وب (کامپیوترهائی که محتویات بنمایش درآمده در مرورگرهای وب را فراهم می‌آورند) و همچنین برنامه‌نویسی دستگاه‌های همانند تلفن‌های موبایل، فراخوان و

PDA[1] استفاده می‌شود. در سال 1995، شاهد توسعه جاوا توسط شرکت Sun Microsystems بودیم. در نوامبر 1995، کنفرانسی در بوستون در ارتباط با اینترنت برگزار شد. فردی از طرف شرکت Sun Microsystems مقاله‌ای در ارتباط با جاوا ارائه کرد. در حین بیان مقاله، برای ما مشخص شد که جاوا در حال کار بر روی توسعه صفحات وب با قابلیت‌های مولتی مدیا و تعاملی است. چندی نگذشت که متوجه قابلیت‌های بسیار زیاد این زبان شدیم. دانستیم که جاوا زبان مناسبی برای دانشجویان سال اول در مبحث آموزش زبان برنامه‌نویسی است، چراکه این زبان شامل مباحث گرافیگ، تصویر، انیمیشن، صدا، ویدئو، پایگاه داده، شبکه، چندنخی (Multithreading) و محاسبات همروند است.

علاوه بر اینها، جاوا بطور بارزی تبدیل به زبان انتخابی به منظور پیاده سازی نرم‌افزار دستگاه‌های ارتباطی بر روی یک شبکه شده است (همانند تلفن‌های موبایل، فراخوان، و PDA). از اینکه زمانی فرا برسد که ضبط صوت ودستگاه‌های دیگر منزلتان با استفاده از تکنولوژی جاوا تشکیل یک شبکه را بدهند تعجب نکنید.

۱-۱۱ فرترن، کوبول و پاسکال

تا کنون صدها زبان سطح بالا ایجاد شده اما فقط تعدادی از آنها موفقیت قابل قبولی بدست آورده‌اند.

FORTRAN (*FORmula TRANslator*) توسط شرکت IBM در بین سال‌های ۱۹۵۴ و ۱۹۵۷ ایجاد شده و در کاربردهای عملی و مهندسی که نیاز به محاسبات پیچیده ریاضی دارد بکار گرفته می‌شود. فرترن هنوز هم به صورت گسترده‌ای مورد استفاده قرار می‌گیرد. مخصوصاً در کاربردهای مهندسی.

COBOL (*Common Business Oriented Language*) توسط گروهی از سازنده‌های کامپیوتر، دولت و کارخانه‌هایی که از کامپیوتر استفاده می‌کردند در سال ۱۹۵۹ طراحی و ایجاد شد. کوبول بصورت یک زبان تجاری مورد استفاده قرار گرفت که نیاز به انجام عملیات دقیق بر روی مقادیر زیادی از داده‌ها دارد. امروزه در حدود نیمی از نرم‌افزارهای تجاری موجود توسط کوبول برنامه‌نویسی شده‌اند. به طور تقریبی در حدود یک میلیون نفر با این نرم‌افزار برنامه‌نویسی می‌کنند.

Pascal همزمان با C طراحی گردیده است. این زبان توسط پروفسور Nicklaus Wirth برای استفاده‌های آکادمیک ایجاد شده است.

۱-۱۲ بیسیک، ویژوال بیسیک، ویژوال ++C، #C و .NET

ایجاد و توسعه برنامه‌های کاربردی مبتنی بر ویندوز میکروسافت توسط زبان‌های هانند C و ++C کار مشکلی بوده و فرآیند پیچیده ای دارد. زمانیکه Bill Gates شرکت میکروسافت را تاسیس نمود، مبادرت به

[1] Personal Digital Assistant

پیاده‌سازی BASIC بر روی چندین کامپیوتر شخصی اولیه کرد. زبان BASIC[1] زبان برنامه‌نویسی است که در میانه دهه 1960 توسط پروفسور *John Kemeny* و *Thomas Kurtz* از کالج Dartmouth به منظور نوشتن برنامه‌های ساده ابداع گردید. هدف از BASIC اولیه، آموزش تکنیک‌های برنامه‌نویسی به افراد مبتدی و تازه کار بود.

با توسعه واسط گرافیکی کاربر (GUI) توسط میکروسافت، در اواخر دهه ۱۹۸۰ و اوایل ۱۹۹۰ بیسیک تکامل تدریجی خود را به سوی ویژوال بیسیک انجام داده بود که توسط گروه مایکروسافت در سال ۱۹۹۱ انجام پذیرفت.

اگرچه ویژوال بیسیک از زبان برنامه‌نویسی BASIC مشتق شده است، اما تفاوت‌های بسیار زیادی با BASIC دارد. ویژوال بیسیک از ویژگیهای قدرتمندی همانند واسط گرافیکی کاربر، رسیدگی به رویداد (*event handling*)، دسترسی به Win32 API، برنامه‌نویسی شی‌گرا، رسیدگی به خطا، برنامه‌نویسی ساخت‌یافته و سایر موارد برخوردار است. ویژوال بیسیک از جمله زبان‌های پر طرفدار در برنامه‌نویسی رویداد گرا (*event-driven*) و واسط‌های ویژوال است.

جدیدترین نسخه ویژوال بیسیک، ویژوال بیسیک NET. است، که برای پلات‌فرم جدید برنامه‌نویسی میکروسافت طراحی شده است. نسخه‌های اولیه ویژوال بیسیک دارای قابلیت شی‌گرا بودند، اما ویژوال بیسیک NET. در سطح بالاتری به عرضه برنامه‌نویسی شی‌گرا می‌پردازد و دارای کتابخانه NET. قدرتمندی از کامپونت‌های نرم‌افزاری با قابلیت استفاده مجدد است.

[نکته: خواننده عزیز، در صورتیکه به زبان برنامه‌نویسی ویژوال بیسیک NET. علاقمند باشید کتاب "راهنمای جامع برنامه نویسان ویژوال بیسک NET. ویرایش دوم" که توسط نویسنده‌گان همین کتاب نوشته شده، از سوی انتشارات اتحاد (نشر و پخش آیلار) ترجمه و چاپ شده است.]

ویژوال C++ نسخه پیاده سازی شده C++ توسط شرکت میکروسافت است. در اوایل، برنامه‌نویسی گرافیگی و GUI در ویژوال C++ با استفاده از *کلاس‌های بنیادین مایکروسافت*[2] (MFC) صورت می‌گرفت. در حال حاضر، با معرفی شدن NET. میکروسافت مبادرت به تهیه یک کتابخانه عمومی برای پیاده سازی GUI، گرافیک، شبکه، چندنخی و موارد دیگر کرده است. این کتابخانه مابین ویژوال بیسیک، ویژوال C++ و زبان برنامه‌نویسی جدید میکروسافت بنام C# مشترک است.

[1] Beginner's All-Purpose Symbolic Instruction Code
[2] Microsoft Foundation Classes

ابزارهای برنامه‌نویسی جدید (همانند C++ و جاوا) و دستگاه‌های الکترونیکی مصرف کننده (همانند تلفن‌های موبایل) مسائل و نیازهای خاص خود را دارند. جمع کردن کامپونت‌های نرم‌افزاری از زبانهای مختلف کار مشکلی است، و معمولا مشکل نصب در این بین رخ می‌دهد، چرا که نسخه‌های جدید از کامپونت‌های اشتراکی با نسخه‌های قدیمی سازگاری پیدا نمی‌کنند. علاوه بر این، توسعه دهنده‌گان متوجه نیازهای خود در زمینه برنامه‌های کاربردی مبتنی بر وب شدند که بتوان به آنها از طریق اینترنت دسترسی پیدا کرده و از آنها استفاده کرد. در نتیجه استقبال عمومی از دستگاه‌های الکترونیکی سیار، توسعه دهنده‌گان نرم‌افزار بر این باور رسیدند که مشتریان، دیگر نمی‌خواهند به کامپیوترهای رومیزی محدود باشند. توسعه دهنده‌گان نیاز خود به یک نرم‌افزار که بتواند در اختیار همه بوده و تقریبا برروی هر نوع دستگاه بکار گرفته شود، را احساس کردند. پاسخ میکروسافت به این نیازها NET. (با تلفظ دات نت) و زبان برنامه‌نویسی C# (با تلفظ سی شارپ) بود.

پلات‌فرم موجود در NET. یک مدل جدید از توسعه نرم‌افزاری عرضه می‌کند که به برنامه‌های ایجاد شده توسط زبان‌های مختلف برنامه‌نویسی امکان می‌دهد تا با یکدیگر ارتباط برقرار نمایند. زبان برنامه‌نویسی C# توسط میکروسافت و به سرپرستی *Anders Hejlsberg* و *Scott Wiltamuth* خاص پلات‌فرم NET. بعنوان زبانی که به برنامه نویسان اجازه می‌داد تا بتوانند به آسانی خود را به سطح NET. ارتقاء دهند، توسعه پیدا کرده است. با توجه به این حقیقت که ریشه C# از زبان‌های C, C++ و جاوا است، فرآیند ارتقاء به آسانی می‌تواند صورت گیرد. در حالیکه C# حاوی بهترین ویژگیهای زبان‌های فوق به همراه قابلیت‌های جدید متعلق به خود است. بدلیل اینکه در ساخت C# بطور گسترده‌ای از زبانهای مناسب و خوش ساختار استفاده شده است، برنامه نویسان از کار کردن و آسانی آن لذت می‌برند.

۱-۱۳ تکنولوژی شی

یکی از نویسنده‌گان این کتاب، پورفسور H.M.Deitel شاهد یک فروپاشی در اواخر دهه 1960 توسط سازمان‌های توسعه نرم‌افزار، بویژه در شرکتهای بود که پروژه‌هائی در سطح کلان انجام می‌دادند. زمانیکه Deitel دانشجو بود، فرصت کار کردن در یک گروه توسعه دهنده در بخش اشتراک زمانی و حافظه مجازی سیستم‌های عامل را بدست آورد، که در نوع خود یک تجربه با ارزش محسوب می‌شد. اما در تابستان 1967، شرکت تصمیم به متوقف کردن پروژه بدلایل اقتصادی کرد، پروژهای که صدها نفر بمدت چندین سال بر روی آن کار کرده بودند. این موضوع نشان داد که نرم‌افزار محصول پیچیده‌ای است.

بهبود وضعیت نرم‌افزار از زمانی آغاز شد که مزایای برنامه‌نویسی *ساخت‌یافته* بر همگان هویدا گردید (از دهه 970). اما تا دهه 1990 تکنولوژی برنامه‌نویسی شی‌گرا بطرز گسترده‌ای بکار گرفته نشد، تا اینکه برنامه‌نویسان به وجود ابزاری برای بهبود فرآیند توسعه نرم‌افزار، احساس نیاز کردند.

در واقع، پیدایش تکنولوژی شی به میانه دهه 1960 باز می‌گردد. زبان برنامه‌نویسی C++ در AT&T و توسط Bjarne Stroustrup در اوایل دهه 1980 توسعه یافته که خود مبتنی بر دو زبان C و Simula 67 است. زبان C در بدو کار در AT&T و برای پیاده سازی سیستم عامل یونیکس در اوایل دهه 1970 ایجاد شد. زبان Simula 67، زبان برنامه‌نویسی شبیه سازی است که در سال 1967 و در اروپا توسعه یافته است. زبان C++ در برگیرنده قابلیت‌ها و توانائی زبان‌های C و Simula در ایجاد و دستکاری شی‌ها است.

شی‌ها چیستند و چرا خاص می‌باشند؟ در واقع، تکنولوژی شی، یک الگوی بسته (package) است که به ما در ایجاد واحدهای نرم‌افزاری با معنی کمک می‌کند. شی‌ها تمرکز زیادی بر نواحی خاص برنامه‌ها دارند. شی‌ها می‌توانند از جمله شی‌های تاریخ، زمان، پرداخت، فاکتور، صدا، ویدئو، فایل، رکورد و بسیاری از موارد دیگر باشد. در حقیقت، می‌توان هر چیزی را بفرم یک شی عرضه کرد.

ما در دنیای از شی‌ها زندگی می‌کنیم. کافیست نگاهی به اطراف خود بیاندازیم. اطراف ما پر است از اتومبیل‌ها، هواپیما، انسان‌ها، حیوانات، ساختمان‌ها، چراغ‌های ترافیگ، بالابرها، و بسیاری از چیزهای دیگر. قبل از اینکه زبان‌های برنامه‌نویسی شی‌گرا ابداع شوند، زبان‌های برنامه‌نویسی (همانند FORTRAN، Pascal ، C و Basic) بر روی اعمال یا actions بجای چیزها یا شی‌ها (نام) تمرکز داشتند. با اینکه برنامه‌نویسان در دنیائی از شی‌ها زندگی می‌کردند اما با فعل‌ها سرگرم بودند. خود همین تناقض باعث شد تا برنامه‌های نوشته شده از قدرت کافی برخوردار نباشند. هم اکنون که زبان‌های برنامه‌نویسی شی‌گرا همانند C و جاوا در دسترس هستند، برنامه‌نویسان به زندگی خود در یک دنیای شی‌گرا ادامه می‌دهند و می‌توانند برنامه‌های خود را با اسلوب شی‌گرا بنویسند. فرآیند برنامه‌نویسی شی‌گرا در مقایسه با برنامه‌نویسی *روالی* (*procedural*) ماهیت بسیار طبیعی‌تری دارد و نتیجه آن هم رضایت بخش‌تر است.

اصلی‌ترین مشکل برنامه‌نویسی روالی این است که واحدهای تشکیل دهنده برنامه نمی‌توانند به آسانی نشاندهنده موجودیت‌های دنیای واقعی باشند. از اینرو، این واحدها نمی‌توانند بطرز شایسته‌ای از قابلیت استفاده مجدد برخوردار باشند. برای برنامه نویسان روالی، نوشتن دوباره و مجدد کدها در هر پروژه جدید و با وجود مشابه بودن این کدها با نرم‌افزار پروژه‌های قبلی، چندان غیر عادی نمی‌باشد. نتیجه اینکار تلف شدن زمان و هزینه‌ها است. با تکنولوژی شی، موجودیت‌های نرم‌افزاری (کلاس) ایجاد می‌شوند و در صورتیکه بدرستی طراحی شده باشند، می‌توان در آینده از آنها در پروژه‌های دیگر استفاده کرد.

استفاده از کتابخانه کامپونت‌ها با قابلیت بکارگیری مجدد همانند *MFC(Microsoft Foundation Classes)* و کامپونت‌های تولید شده توسط *Rogue Wave* و سایر شرکتهای نرم‌افزاری، می‌تواند در کاهش هزینه‌ها و نیازهای پیاده‌سازی سیستم‌های خاص بسیار موثر باشد.

با بکارگیری برنامه‌نویسی شی‌گرا، نرم‌افزار تولید شده بسیار قابل فهم‌تر شده، نگهداری و سازماندهی آن اصولی‌تر و اصلاح و خطایابی آن ساده‌تر می‌شود. این موارد از اهمیت خاصی برخوردار هستند چراکه تخمین زده می‌شود که هشتاد در صد هزینه یک نرم‌افزار مربوط به دوره نگهداری و ارتقاء آن در چرخه طول عمرش است و ارتباطی با نوشتن و توسعه اولیه نرم‌افزار ندارد. با تمام این اوصاف، مشخص است که برنامه‌نویسی شی‌گرا تبدیل به یکی از کلیدی‌ترین مفاهیم برنامه‌نویسی در چند دهه آینده خواهد شد.

یکی از دلایل نوشتن کدهای متعلق بخود این است که دقیقا می‌دانید که کد نوشته شده چگونه کار می‌کند، و می‌توانید به بررسی آن بپردازید. عیب اینکار هم در صرف زمان و پیچیدگی طراحی و پیاده‌سازی کد جدید است.

مهندسی نرم‌افزار

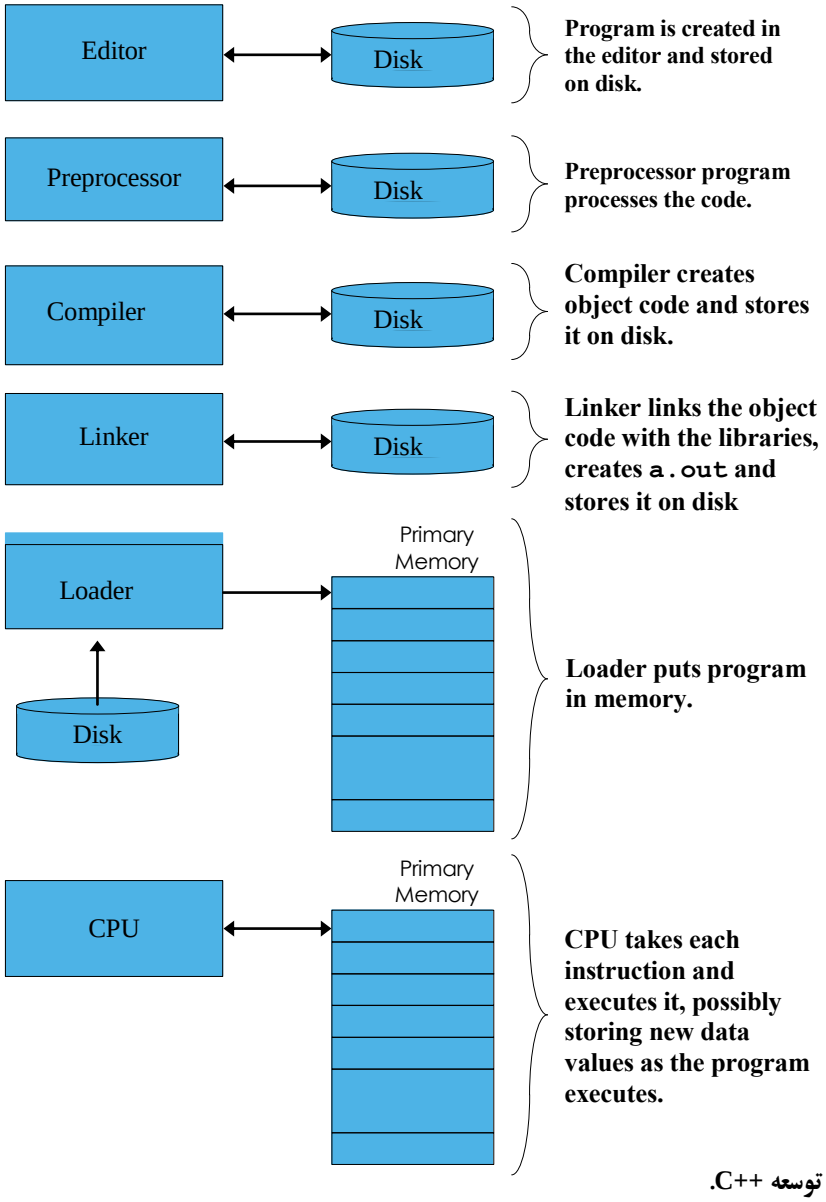
کتابخانه کامپونت‌های نرم‌افزاری با قابلیت استفاده مجدد را می‌توان از طریق اینترنت و وب گسترده جهانی بدست آورد. برای تهیه بسیاری از این کتابخانه‌ها نیازی به پرداخت هزینه نیست.

۱۴-۱ محیط توسعه C++

اجازه دهید تا به بررسی مراحل ایجاد و اجرای یک برنامه کاربردی C++ با استفاده از محیط توسعه C++ بپردازیم (شکل ۱-۱). اصولاً سیستم C++ متشکل از سه بخش است: *محیط توسعه برنامه*, *زبان* و *کتابخانه استاندارد* C++. عموماً برنامه‌های C++ از شش فاز یا مرحله عبور می‌کنند: ویرایش, پیش پردازش، کامپایل, لینک, بار شدن و اجرا. در ادامه به توضیح هر فاز می‌پردازیم.

فاز ۱: ایجاد برنامه

فاز ۱ مرکب از ویرایش یک فایل با استفاده از یک برنامه ویرایشگر است. با استفاده از یک ویرایشگر مبادرت به تایپ برنامه C++ کرده و هرگونه اصلاحات مورد نیاز را در آن اعمال و بر روی یک دستگاه ذخیره‌سازی ثانویه, همانند دیسک سخت ذخیره می‌کنیم. فایل‌های برنامه C++ دارای پسوندهای ***.cpp***, ***.cxx*** , ***.cc*** یا ***.C***. می‌باشند (دقت کنید که C با حرف بزرگ است), که نشان می‌دهند, که فایل حاوی کد منبع C++ است. برای مشاهده پسوند فایل در محیط توسعه C++ به مستندات محیط توسعه C++ بکار رفته بر روی سیستم خود مراجعه کنید.



شکل ۱-۱ | محیط توسعه C++.

دو ویرایشگر که در سیستم UNIX کاربرد بیشتری دارند عبارتند از **vi** و **emacs**. بسته‌های نرم‌افزاری C++ برای ویندوز میکروسافت همانند

Metroworks CodeWarrior (www.metrowerks. com), Borland C++ (www.borland. com)

و Microsoft Visual C++ (www.msdn.microsoft.com/visualc/) دارای ویرایشگر مجتمع در محیط برنامه‌نویسی خود هستند. البته می‌توانید از یک ویرایشگر متنی ساده همانند Notepad در ویندوز, برای نوشتن کد C++ استفاده کنید. فرض ما بر این است که شما حداقل با نحوه ویرایش یک برنامه آشنا هستید.

فاز ۲ و ۳: پیش پردازنده و کامپایل یک برنامه C++

در فاز ۲, برنامه‌نویس, دستوری برای کامپایل برنامه صادر می‌کند. در یک سیستم C++, برنامه *پیش‌پردازنده*, قبل از اینکه فاز ترجمه کامپایلر شروع بکار کند, آغاز یا اجرا می‌شود (از اینرو به پیش‌پردازنده فاز ۲ و به کامپایل فاز ۳ نام گذاشته‌ایم). پیش‌پردازنده فرمانبر دستوراتی بنام *رهنمود پیش‌پردازنده* است که عملیات مشخص بر روی برنامه قبل از کامپایل آن انجام می‌دهند. معمولاً این عملیات شامل کامپایل سایر فایلهای متنی و انجام انواع جایگزینی‌ها است. در فصل‌های آتی با تعدادی از رهنمودهای پیش‌پردازنده آشنا خواهید شد. در فاز ۳, کامپایلر شروع به ترجمه برنامه C++ به کد زبان ماشین می‌کند که گاهاً بعنوان *کد شی* (*object code*) شناخته می‌شود.

فاز ٤: لینک

فاز ۴ معروف به فاز لینک است. معمولاً برنامه‌های C++ حاوی مراجعه‌هایی به توابع و داده‌های تعریف شده در بخش‌های گوناگون هستند, همانند کتابخانه‌های استاندارد یا کتابخانه‌های خصوصی گروهی از برنامه‌نویسان که بر روی یک پروژه کار می‌کنند. کد شی تولید شده توسط کامپایلر C++ حاوی شکاف‌هایی است که از فقدان این بخش‌ها بوجود می‌آیند. برنامه *لینکر* مبادرت به لینک کد شی با کد توابع مفقوده می‌کند تا یک تصویر کامل و بدون شکاف بوجود آید. اگر برنامه بدرستی کامپایل و لینک گردد, یک نماد اجرایی تولید خواهد شد.

فاز ٥: بار شدن

فاز ۵, فاز *بار* شدن نام دارد. قبل از اینکه برنامه بتواند اجرا شود, ابتدا بایستی در حافظه جای گیرد. این کار توسط *بارکننده* (*loader*) صورت می‌گیرد که نماد اجرایی را از دیسک دریافت و به حافظه منتقل می‌کند. همچنین برخی از کامپونت‌ها که از برنامه پشتیبانی می‌کنند, به حافظه بار می‌شوند.

فاز ٦: اجرا

سرانجام, کامپیوتر, تحت کنترل CPU خود, مبادرت به اجرای یک به یک دستورالعمل‌های برنامه می‌کند.

مسائلی که ممکن است در زمان اجرا رخ دهند

همیشه برنامه‌ها در همان بار اول اجرا نمی‌شوند. در هر کدامیک از فازهای فوق امکان شکست وجود دارد، چرا که ممکن است انواع مختلفی از خطاها که در مورد آنها صحبت خواهیم کرد, در این بین رخ دهند. برای مثال, امکان دارد برنامه اقدام به انجام عملیات تقسیم بر صفر نماید (یک عملیات غیرمعتبر). در

چنین حالتی برنامه ++C یک پیغام خطا بنمایش در خواهد آورد. اگر چنین شود, مجبور هستید تا به فاز ویرایش بازگردید و اصلاحات لازم را انجام داده و مجدداً مابقی فازها را برای تعیین اینکه آیا اصلاحات صورت گرفته قادر به رفع مشکل بوده‌اند یا خیر, طی کنید.

بیشتر برنامه‌های نوشته شده در ++C عملیات ورود و یا خروج داده انجام می‌دهند. توابع مشخصی از ++C, از طریق **cin** (استریم ورودی استاندارد, با تلفظ "see-in") که معمولاً از طریق صفحه‌کلید است, اقدام به دریافت ورودی می‌کنند, اما تابع **cin** می‌تواند به دستگاه دیگری هدایت شود. غالباً داده‌ها از طریق دستور **cout** (استریم استاندارد خروجی, با تلفظ "see-out") خارج می‌گردند که معمولاً صفحه‌نمایش کامپیوتر است, اما می‌توان **cout** را به دستگاه دیگری هدایت کرد. زمانیکه می‌گوییم برنامه نتیجه‌ای را چاپ می‌کند, معمولاً منظورمان نمایش نتیجه بر روی صفحه نمایش (مانیتور) است. می‌توان داده‌ها را به دستگاه‌های دیگری همانند دیسک‌ها ارسال کرد و توسط چاپگر از آنها چاپ گرفت. همچنین دستوری بنام **cerr** (استریم استاندارد خطا) وجود دارد, که از آن برای نمایش پیغام‌های خطا استفاده می‌شود, این دستور اکثراً برای نمایش پیغام در صفحه نمایش بکار گرفته می‌شود.

خطای برنامه‌نویسی

خطاهایی همانند, خطای تقسیم بر صفر در زمان اجرای برنامه رخ می‌دهند, از اینرو به آنها خطای زمان اجرا گفته می‌شود. خطاهای زمان اجرای عظیم یا مهلک (fatal runtime errors) سبب می‌شوند تا برنامه‌ها بلافاصله خاتمه یابند بدون اینکه بتوانند با موفقیت وظایف خود را به انجام برسانند. خطاهای زمان اجرای غیرمهلک اجازه می‌دهند تا برنامه‌ها بطور کامل اجرا شوند, اما نتایج اشتباهی تولید می‌کنند.

۱۵-۱ نکاتی در مورد ++C و این کتاب

گاهی اوقات برنامه‌نویسان با تجربه ++C بدون رعایت هیچ‌گونه ساختار و روش مشخص شروع به برنامه‌نویسی می‌کنند, که چنین کاری در برنامه‌نویسی روش ضعیفی است. خواندن, نگهداری, تست و خطایابی چنین برنامه‌هایی بسیار مشکل بوده و گاهاً رفتار عجیبی از خود بروز می‌دهند. این کتاب با در نظر گرفتن برنامه‌نویسان مبتدی سعی در نوشتن واضح برنامه‌ها دارد.

برنامه‌نویسی ایده‌ال

برنامه‌های ++C خود را ساده و سرراست بنویسید. گاهی اوقات به این روش KIS (keep it simple) گفته می‌شود. می‌دانید که C و ++C از جمله زبان‌های قابل حمل می‌باشند و برنامه‌های نوشته شده در این دو زبان قادر به اجرا برای انواع کامپیوترها هستند. قابلیت حمل یک بحث اغواکننده است. مستند ANSIC استاندارد حاوی لیست طولانی از مباحث قابلیت حمل است و کتاب‌های کاملی در این زمینه نوشته شده‌اند.

قابلیت حمل

اگر چه نوشتن برنامه‌های قابل حمل امکان‌پذیر است, چندین مسئله در میان انواع مختلف کامپایلرهای C و

++C و کامپیوترها وجود دارد که می‌توانند قابلیت حمل را مشکل‌ساز کنند. نوشتن برنامه‌ها در C و ++C قابلیت حمل را تضمین نمی‌کند. غالباً برنامه‌نویس نیاز به توجه دقیق به نوع کامپایلر و کامپیوتر دارد، که بصورت سرجمع پلات فرم (platform) نام دارد.

مطالب عرضه شده در این کتاب براساس مستندات استاندارد ANSI/ISO C++ نوشته شده‌اند. با این همه, C++ یک زبان غنی است و برخی از ویژگی‌های آن در این کتاب مورد بررسی قرار نگرفته‌اند. اگر نیاز به جزئیات تکنیکی دیگری در ارتباط با C++ داشته باشید, بایستی به مطالعه مستندات استاندارد C++ بپردازید، که می‌توان از وب سایت ANSI به آن دسترسی پیدا کرد.

webstore.ansi.org/ansidocstore/default.asp

مستند عنوان "Information Technology-Programming Language-C++" داشته و شماره مستند INCITS/ISO/IEC 14882-2003 است.

۱-۱۶ تست یک برنامه C++

در این بخش, مبادرت به اجرا و تعامل با اولین برنامه C++ خود می‌کنیم. کار را با اجرای *بازی حدس عدد* شروع می‌کنیم که عددی از 1 تا 1000 دریافت و شروع به حدس عدد می‌کند. اگر عددی که وارد کرده‌اید یا حدس زده‌اید, صحیح باشد, بازی به پایان رسیده است. اگر حدس شما صحیح نباشد, برنامه نشان خواهد داد که عدد وارد شده بزرگتر یا کوچکتر از عدد صحیح است. برای حدس زدن عدد هیچ محدودیتی در تعداد حدس وجود ندارد.

اجرای یک برنامه C++ را به دو روش به شما نشان خواهیم داد, استفاده از ***خط اعلان*** یا ***دستور*** (***Command Prompt***) ویندوز XP و استفاده از یک ***پوسته*** (***shell***) در لینوکس. اجرای برنامه در هر دو پلات‌فرم یکسان است. محیط‌های توسعه متعددی وجود دارند که شما بعنوان خواننده این کتاب می‌توانید با استفاده از آنها اقدام به کامپایل, ایجاد و اجرای برنامه‌های C++ کنید, محیط‌های همانند Borland C++ Builder, Metrowerks, GNU C++, Microsoft Visual C++. NET و غیره در حالیکه ما تمام این محیط‌ها را تست نکرده‌ایم, اما اطلاعاتی در بخش ۱۹-۱ فراهم آورده‌ایم که در ارتباط با کامپایلرهای مجانی C++ موجود در اینترنت هستند و می‌توان آنها را برداشت کرد. لطفاً با توجه به دستورالعمل مدرس خود اقدام به تهیه محیط توسعه کنید.

با توجه به مراحل معرفی شده در زیر, قادر به اجرای برنامه و وارد کردن اعداد مختلف برای حدث عدد صحیح خواهید بود. عناصر و مراحلی که در این برنامه مشاهده می‌کنید در کل این کتاب کاربرد خواهند داشت.

اجرای یک برنامه C++از طریق خط دستور ویندوز XP

۱- *بررسی setup*. برای اینکه مطمئن شوید مثال‌های کتاب به درستی بر روی دیسک‌سخت کامپیوترتان کپی شده‌اند, به مطالعه بخش «قبل از شروع بکار» در ابتدای کتاب کنید.

۲- *یافتن برنامه کامل*. پنجره خط دستور را باز کنید. برای کاربران در حال استفاده از ویندوز 95, 98 یا 2000, این پنجره از طریق ***Start>Programs>Accessories>Command Prompt*** باز می‌شود. برای کاربران ویندوز XP, این پنجره از طریق ***Start>All Programs>Accessories>Command Prompt*** باز می‌شود. برای رفتن به سراغ شاخه برنامه **GussNumber**, مبادرت به تایپ ***cd C:\examples\ch01\GussNumber\windows*** کرده سپس کلید *Enter* را فشار دهید (شکل ۲-۱). از دستور **cd** برای تغییر شاخه استفاده می‌شود.

۳- *اجرای برنامه **GuessNumber***. اکنون که در شاخه حاوی برنامه **GuessNumber** هستید, دستور **GuessNumber** را تایپ و کلید *Enter* را فشار دهید (شکل ۳-۱). [نکته: **GuessNumber.exe** نام واقعی برنامه است. با این وجود, فرض ویندوز پسوند **exe.** است.]

۴- *وارد کردن اولین حدس*. برنامه پیغام "`Please type first guess.`" را بنمایش در آورده و سپس یک علامت سئوال (?) در خط بعدی ظاهر می‌سازد (شکل ۳-۱). در این خط عدد 500 را وارد کنید (شکل ۴-۱).

۵- *وارد کردن حدس بعدی*. برنامه پیغام "`Too high. Try again`" را به نمایش در می‌آورد, به این معنی که مقدار وارد شده بزرگتر از عدد انتخابی از سوی برنامه به عنوان مقدار صحیح است. از اینرو, باید یک عدد کوچکتر برای حدس بعدی خود وارد کنید. در خط اعلان, مقدار 250 را وارد سازید (شکل ۵-۱). برنامه مجدداً پیغام "`Too high. Try again`" را بنمایش در می‌آورد, چرا که مقدار وارد شده هنوز بزرگتر از عدد صحیح مورد نظر می‌باشد.

شکل ۲-۱ | باز کردن پنجره Command Prompt و تغییر شاخه.

شکل ۳-۱ | اجرای برنامه GuessNumber

شکل ٤-۱ | وارد کردن اولین حدس.

شکل ٥-۱ | وارد کردن دومین حدس و پاسخ دریافتی.

۶- *وارد کردن حدس‌های بعدی*. با وارد کردن مقادیری به بازی ادامه دهید تا موفق به حدس عدد صحیح شوید. پس از حدس پاسخ صحیح, برنامه پیغام "`Excellent! You guessed the number!`" را بنمایش در می‌آورد (شکل ۶-۱).

۷- *بازی مجدد یا خروج از برنامه*. پس از حدس عدد صحیح, برنامه در مورد انجام یک بازی دیگر سئوال می‌کند (شکل ۶-۱). در مقابل پیغام "**Would you like to play again (y or n)?**" وارد کردن کاراکتر **y** سبب می‌شود تا برنامه یک عدد جدید انتخاب کرده و مجدداً با نمایش پیغام "**Please enter your first guess.**" و بدنبال آن یک علامت سئوال در خط بعدی یک بازی جدید را شروع کند (شکل ۷-۱). با وارد کردن کاراکتر **n** برنامه خاتمه یافته و به شاخه برنامه در خط دستور باز می‌گردد (شکل ۸-۱). در هر بار اجرای این برنامه از مرحله ۳, همان اعداد برای حدس انتخاب می‌شوند.

۸- *بستن پنجره* ***Command Prompt***

شکل ٦-١ | وارد کردن اعداد دیگر و حدس عدد صحیح.

شکل ۷-۱ | بازی مجدد.

شکل ۸-۱ | خروج از بازی.

اجرای یک برنامه ++C با استفاده از GNU C++ در لینوکس

برای تست این برنامه، فرض می‌کنیم که شما با نحوه کپی مثال‌ها به شاخه خانگی خود آشنا هستید. اگر سئوالی در این زمینه دارید، می‌توانید از مدرس خود کمک بگیرید. خط اعلان در پوسته سیستم از کاراکتر مد (~) استفاده می‌کند تا نشاندهنده شاخه خانگی (home) باشد و هر اعلان با کاراکتر **$** پایان می‌یابد. خط اعلان در سیستم‌های لینوکس بسیار متفاوت است.

۱- *یافتن برنامه*. از طریق یک پوسته لینوکس، به شاخه برنامه **GuessNumber** بروید (شکل ۹-۱)، با تایپ

```
cd Examples\ch01\GuessNumber\GNU_Linux
```

سپس کلید *Enter* را فشار دهید. دستور **cd** برای تغییر شاخه بکار گرفته می‌شود.

۲- *کامپایل برنامه* **GuessNumber**. برای اجرای برنامه در کامپایلر GNU C++، بایستی ابتدا آن را توسط عبارت زیر کامپایل کنید

```
g++ GuessNumber.cpp -o GuessNumber
```

همانند شکل ۱۰-۱. دستور فوق سبب کامپایل برنامه و تولید یک فایل اجرایی بنام **GuessNumber** در خط اعلان بعدی می‌شود. کلید *Enter* را فشار دهید (شکل ۱۱-۱).

```
~$ cd examples/ch01/GuessNumber/GNU_linux
~/examples/ch01/GuessNumber/GNU_Linux$
```

شکل ۹-۱ | تغییر شاخه برنامه GussNumber پس از ورود به سیستم لینوکس.

```
~/examples/ch01/GuessNumber/GNU_linux$ g++ GuessNumber.cpp -o GuessNumber
```

```
~/examples/ch01/GuessNumber/GNU_linux$
```

شکل ۱۰-۱ | کامپایل برنامه GuessNumber با استفاده از دستور g++.

```
~/examples/ch01/GuessNumber/GNU_linux$  ./GuessNumber
I have a number between 1 and 1000.
Can you guess my number?
Please type your first guess.
?
```

شکل ۱۱-۱ | اجرای برنامه GuessNumber

۴- *وارد کردن حدس اول.* برنامه عبارت "**Pleas type your first guess.**" و سپس نماد علامت سئوال (**?**) را بعنوان اعلان در خط بعدی بنمایش در می‌آورد (شکل ۱۱-۱). در این اعلان عدد 500 را وارد سازید (شکل ۱۲-۱).

۵- *وارد کردن حدس بعدی.* "**Too high. Try again.**" را بنمایش در می‌آورد، به این معنی که مقدار ورودی بزرگتر از عدد انتخابی از سوی برنامه است (شکل ۱۲-۱). در اعلان بعدی، مقدار 250 را وارد کنید (شکل ۱۳-۱). این بار برنامه پیغام "**Too low. Try again**" را نشان می‌دهد. چرا که مقدار وارد شده کوچکتر از مقدار صحیح است.

۶- *وارد کردن حدس‌های بعدی.* به بازی با وارد کردن حدس‌های بعدی ادامه دهید تا عدد صحیح را حدس بزنید. پس از حدس زدن پاسخ، "**Excellent! You guessed the number!**" بنمایش در می‌آید (شکل ۱۴-۱).

```
~/examples/ch01/GuessNumber/GNU_linux$  ./GuessNumber
I have a number between 1 and 1000.
Can you guess my number?
Please type your first guess.
? 500
Too hight.Try again.
?
```

شکل ۱۲-۱ | وارد کردن حدس اول.

```
~/examples/ch01/GuessNumber/GNU_linux$  ./GuessNumber
I have a number between 1 and 1000.
Can you guess my number?
Please type your first guess.
? 500
Too hight.Try again.
? 250

Too low. Try again.

?
```

شکل ۱۳-۱ | وارد کردن حدس دوم و دریافت پاسخ برنامه.

```
Too low. Try again.
?  375
Too low. Try again.
?  437
Too high. Try again.
?  406
```

```
Too high. Try again.
? 391
Too low. Try again.
? 387
Too high. Try again.
? 385
Too high. Try again.
? 384

Excellent! You guessed the number!
Would you like to play again <y or n>?
```

شکل ١٤-١ | وارد کردن حدس‌های بعدی و حدس پاسخ صحیح.

۷- *بازی مجدد یا خروج از برنامه*. پس از حدس عدد صحیح، برنامه در مورد انجام یک بازی دیگر سئوال می‌کند. در مقابل پیغام "**Would you like to play again (y or n)?**" وارد کردن کاراکتر **y** سبب می‌شود تا برنامه یک عدد جدید انتخاب کرده و مجدداً با نمایش "**Please enter your first guess.**" و بدنبال آن یک علامت سئوال در خط بعدی یک بازی جدید را شروع کند (شکل ۱۵-۱). با وارد کردن کاراکتر **n** برنامه خاتمه یافته و به شاخه برنامه در پوسته باز می‌گردید (شکل ۱۶-۱). در هر بار اجرای این برنامه از مرحله ۳، همان اعداد برای حدس انتخاب می‌شوند.

```
Excellent! You guessed the number!
Would you like to play again <y or n>? y

I have a number between 1 and 1000.
Can you guess my number?
Please type your first guess.
?
```

شکل ۱۵-۱ | بازی مجدد.

```
Excellent! You guessed the number.
Would you like to play again <y or n>? n
~/examples/ch01/GuessNumber/GNU_linux$
```

شکل ١٦-١ | خروج از بازی.

۱-۱۷ مبحث آموزشی مهندسی نرم‌افزار: مقدمه‌ای بر تکنولوژی شی و UML

اکنون بحث خود را در ارتباط با شی‌گرا و تفکر طبیعی در ارتباط با جهان و برنامه‌نویسی کامپیوتر آغاز می‌کنیم. در انتهای تمام فصل‌های ۱ الی ۷ و ۱۳ بخشی بعنوان مبحث آموزشی مهندسی نرم‌افزار وجود دارد که در آن به معرفی دقیق بحث شی‌گرا می‌پردازیم. هدف ما از این بخش کمک به شما در توسعه یک روش تفکر شی‌گرا و آشنایی با ***UML (Unified Modeling Language)*** است. یک زبان گرافیکی که به افراد مسئول طراحی سیستم‌های نرم‌افزاری شی‌گرا امکان می‌دهد تا از نمادهای استاندارد شده صنعتی برای نمایش طرح‌ها استفاده کنند. در این بخش که مطالعه آن الزامی است، به معرفی مفاهیم پایه شی‌گرا و اصطلاحات فنی آن خواهیم پرداخت. بخش‌های اختیاری قرار گرفته در فصل‌های ۲ الی ۷، ۹ و ۱۳ نمایشی از یک طراحی شی‌گرا و پیاده‌سازی نرم‌افزاری برای یک ماشین خودپرداز (ATM) ساده ارائه می‌کنند.

بخش‌های مهندسی نرم‌افزار در انتهای فصل‌های ۲ الی ۷ عبارتند از:

- تحلیل نیازمندی‌های یک سیستم نرم‌افزاری (سیستم ATM) برای ایجاد آن
- تعیین صفاتی که شی‌ها باید داشته باشند
- تعیین رفتار این شی‌ها
- تعیین نحوه تعامل شی‌ها با یکدیگر برای برطرف کردن نیازهای سیستم

مفاهیم پایه تکنولوژی شی

مقدمه بحث شی‌گرا را با برخی اصطلاحات کلیدی آغاز می‌کنیم. به هر کجا در دنیای واقعی نگاه کنید. شاهد *شی‌ها* خواهید بود. مردم، حیوانات، گیاهان، اتومبیل‌ها، هواپیماها، ساختمان‌ها، کامپیوترها و بسیاری از چیزهای دیگر. تفکر انسان براساس شی است. تلفن‌ها، خانه‌ها، چراغ‌های ترافیک، اجاق‌های میکروویو و کولرهای آبی تعدادی کمی از بی‌شمار شی‌ها هستند که در زندگی روزمره خود شاهد آنها هستیم.

گاهی اوقات مبادرت به تقسیم شی‌ها به دو طبقه می‌کنیم: متحرک و غیرمتحرک. شی‌های متحرک، زنده هستند و می‌توانند حرکت کرده و کاری انجام دهند. از سوی دیگر، شی‌های غیرمتحرک، در محیط خود حرکت نمی‌کنند. با این همه، شی‌ها از هر دو نوع، در برخی چیزها مشترک هستند. همه آنها دارای *صفات* (همانند سایز، شکل، رنگ و وزن) بوده و از خود *رفتارهای* نشان می‌دهند (مثلاً توپ می‌غلتد، بالا و پایین می‌پرد، یک بچه گربه می‌کند، می‌خوابد، راه می‌رود، یک اتومبیل شتاب می‌گیرد، ترمز کرده و چرخش می‌کند، یک حوله آب را جذب می‌کند).

انسان‌ها با بررسی صفات و رفتار اشیاء، مطالبی در ارتباط با آنها یاد می‌گیرند. شی‌های مختلف می‌توانند صفات و رفتار مشابهی داشته باشند. مقایسه را همیشه می‌توان اعمال کرد، مثلاً مابین کودکان و بزرگسالان.

طراحی نرم‌افزار مدل‌های شی‌گرا ***OOD (Object-oriented design)*** شبیه به روشی است که افراد برای توصیف شی‌ها در دنیای واقعی بکار می‌برند. می‌توان از رابطه موجود مابین کلاس استفاده کرد، آنجا که شی‌هایی از یک کلاس همانند کلاس وسیله نقلیه دارای ویژگی‌های یکسان هستند، اتومبیل‌ها، تریلی‌ها دارای صفات مشترک می‌باشند. OOD از مزیت *ارث‌بری* (یا توارث) سود می‌برد، که در آن کلاس‌های جدید، صفاتی را از کلاس‌های موجود جذب می‌کنند و به صفات خاص خود اضافه می‌نمایند.

طراحی شی‌گرا یک روش طبیعی و ذاتی در نمایش فرآیند طراحی نرم‌افزار بنام مدلسازی شی‌ها توسط صفات، رفتار و رابطه موجود در میان شی‌ها همانند شی‌ها در دنیای واقعی است. همچنین OOD ارتباط مابین شی‌ها را مدلسازی می‌کند. همانند فردی که پیغامی به شخص دیگری می‌فرستد، شی‌ها نیز از طریق پیغام ارتباط برقرار می‌کنند. یک شی حساب بانکی می‌تواند پیغامی برای کاهش موجودی دریافت کند چرا که مشتری مقداری از پول خود را از حساب برداشت کرده است.

OOD مبادرت به *کپسوله‌سازی* صفات و *عملیات* (رفتار) در شی‌ها می‌کند. سرانجام صفات و رفتار یک شی با هم گره می‌خورند. شی‌ها از خصیصه *پنهان‌سازی اطلاعات* برخوردار هستند. به این معنی که شی‌ها از نحوه برقراری ارتباط با یک شی دیگر از طریق یک *واسط* (رابط) مناسب مطلع هستند، اما معمولاً از نحوه پیاده‌سازی شی‌های دیگر اطلاعی ندارند. معمولاً جزئیات پیاده‌سازی در درون خود شی‌ها پنهان می‌باشند. می‌توان این امر را با رانندگی اتومبیل مقایسه کرد که در آن برای راندن اتومبیل نیازی نیست تا با جزئیات موتور، جعبه‌دنده و عملکرد ترمز آشنا بود. در ادامه با نحوه پنهان‌سازی اطلاعات و دلایل آن آشنا خواهید شد، که از جمله موارد مناسب در مهندسی نرم‌افزار است.

زبان‌های همانند C++ *شی‌گرا* هستند. به برنامه‌نویسی در چنین زبان‌های، *برنامه‌نویسی شی‌گرا* (*OOP*) گفته می‌شود و به برنامه‌نویسان کامپیوتری امکان می‌دهند تا یک طرح شی‌گرا را بصورت یک سیستم نرم‌افزاری پیاده‌سازی کنند. از سوی دیگر زبان‌های همانند C، زبان‌های *روالی* هستند و از اینرو برنامه‌نویسان مقید به *عمل* می‌باشند. در C واحد برنامه‌نویسی *تابع* است، در حالیکه در C++ این واحد، *کلاس* می‌باشد. کلاس‌های C++ حاوی توابعی هستند که پیاده‌سازی‌کننده صفات می‌باشند.

برنامه‌نویسان C بر روی نوشتن توابع تمرکز دارند. آنها گروهی از عمل‌ها را که وظایفی انجام می‌دهند در درون یک تابع قرار می‌دهند و سپس گروهی از توابع را به فرم یک برنامه در می‌آورند. بطور مشخص داده‌ها در C بسیار با ارزش هستند، اما از همان ابتدای امر برای پشتیبانی از اعمالی که توابع انجام خواهند داد وجود دارند.

کلاس‌ها، اعضا داده و توابع عضو

تمرکز برنامه‌نویسان C++ بر روی ایجاد «*نوع‌های تعریف شده از سوی کاربر*» که *کلاس* نامیده می‌شوند است. هر کلاس حاوی داده و هم مجموعه‌ای از متدها است که بر روی داده‌ها کار می‌کنند و سرویس‌های برای *سرویس‌گیرنده‌ها* (کلاس‌ها و توابع دیگری که از کلاس استفاده می‌کنند) تدارک می‌بینند، به کامپونت‌های داده از یک کلاس *اعضای داده* گفته می‌شود. برای مثال، یک کلاس حساب بانکی می‌تواند شامل یک شماره حساب و یک موجودی باشد. به کامپونت‌های داده از یک کلاس، *توابع عضو* گفته می‌شود (معمولاً در سایر زبان‌های برنامه‌نویسی شی‌گرا همانند جاوا، به توابع عضو، *متد* گفته می‌شود). برای مثال، یک کلاس حساب بانکی می‌تواند دارای توابع عضوی برای ایجاد یک پس‌انداز (افزایش‌دهنده موجودی)، برداشت (کاهش‌دهنده موجودی) و نمایش موجودی فعلی باشد. برنامه‌نویس از نوع‌های توکار (و سایر نوع‌های تعریف شده توسط کاربر) بعنوان «بلوک‌های سازنده» در ایجاد نوع‌های جدید (کلاس‌ها) استفاده می‌کند. اسامی در ساختار یک سیستم به برنامه‌نویس C++ در تعیین اینکه کدام شی‌ها برای کار با یکدیگر طراحی شده‌آند کمک می‌کنند.

کلاس‌ها همانند نقشه ترسیمی خانه‌ها هستند. یک کلاس، نقشه ایجاد یک شی از کلاس است. همانطوری که می‌توانیم خانه‌های متعددی از روی یک نقشه بسازیم، می‌توانیم تعدادی شی از روی یک کلاس، نمونه‌سازی کنیم. نمی‌توان در نقشه آشپزخانه مبادرت به آشپزی کرد، آشپزی فقط در آشپزخانه خانه امکان‌پذیر است.

کلاس‌ها می‌توانند با کلاس‌های دیگر ارتباط داشته باشند. برای مثال در یک طرح شی‌گرا از بانک، کلاس "bank teller" نیاز به برقراری ارتباط با کلاس‌های دیگر همانند کلاس «مشتری»، کلاس «پرداخت نقدی»، کلاس «پس‌انداز» و غیره دارد. به این روابط، *وابستگی* گفته می‌شود.

بسته کردن (package) نرم‌افزار بصورت کلاس‌ها می‌تواند ویژگی استفاده مجدد از نرم‌افزار را عرضه کند. غالباً کلاس‌های مرتبط با یکدیگر را بصورت *کامپونت‌های* با قابلیت استفاده مجدد بسته‌بندی می‌کنند.

مهندسی نرم‌افزار

استفاده مجدد از کلاس‌های موجود به هنگام ایجاد کلاس‌ها و برنامه‌های جدید سبب صرفه‌جویی در زمان و هزینه‌ها می‌شود. همچنین به برنامه‌نویسان در ایجاد سیستم‌های قابل اعتمادتر و کاراتر کمک می‌کند، چرا که کلاس‌ها و کامپونت‌های موجود غالباً از مرحله تست، خطایابی و میزان کارایی عبور کرده‌اند.

مقدمه‌ای بر تحلیل و طراحی شی‌گرا (OOAD)

بزودی شروع به برنامه‌نویسی در C++ خواهید کرد. چگونه می‌خواهید کد برنامه‌های خود را ایجاد کنید؟ شاید، می‌خواهید همانند هر برنامه‌نویس تازه‌کاری، فقط کامپیوتر را روشن کرده و شروع به تایپ برنامه کنید. این روش می‌تواند در ارتباط با برنامه‌های کوچک کاربرد داشته باشد، اما اگر از شما خواسته شود تا یک سیستم نرم‌افزاری برای کنترل صدها دستگاه ماشین پرداخت اتوماتیک یک بانک ایجاد کنید، چه خواهید کرد؟ یا اگر از شما خواسته شود تا با یک تیم هزار نفری در ارتباط با تولید نسل بعدی سیستم کنترل ترافیک هوایی ایالات متحده کار کنید، چه خواهید کرد؟ برای پروژه‌های به این بزرگی و پیچیدگی، نمی‌توانید به راحتی پشت کامپیوتر نشسته و شروع به برنامه‌نویسی کنید.

برای دستیابی به بهترین راه حل، بایستی بدقت نیازمندیهای پروژه را تجزیه و تحلیل کنید (تعیین کنید که سیستم چه کار می‌خواهد انجام دهد) و طرحی را توسعه دهید که آنها را برآورده سازد (سیستم قادر به تصمیم‌گیری صحیح برای انجام وظایف خود باشد). در حالت ایده‌آل، قبل از هرگونه کدنویسی، باید به سراغ فرآیند رفته و به دقت طراحی را انجام دهید. اگر این فرآیند مستلزم تحلیل و طراحی سیستم از نقطه نظر شی‌گرا باشد، آن را ***object-orientend analysis and design (OOAD)*** یا ***تحلیل و طراحی شی‌گرا*** می‌گویند. برنامه‌نویسان با تجربه مطلع هستند که تحلیل و طراحی می‌تواند در زمان و هزینه ایجاد برنامه بسیار صرفه‌جویی کند، با اجتناب از اعمال طرح‌های ضعیف که در هر بار برنامه را به شکست کشانده و باعث می‌شوند کار از ابتدا آغاز گردد که همان تحمیل هزینه و زمان است.

OOAD یک کلمه کلی برای بیان فرآیند تجزیه و تحلیل یک مسئله و توسعه روش حل آن مسئله است. مسائل کوچکی همانند مسائلی که در چند فصل اولیه با آنها برخورد خواهیم داشت، نیازی به فرآیند OOAD ندارند. برای ایجاد این برنامه‌ها، می‌توان ابتدا مبادرت به نوشتن *شبه کد* آنها کرد و سپس کد C++ آنها را نوشت. در فصل ۴ به معرفی شبه کد خواهیم پرداخت.

با بزرگتر و پیچیده‌تر شدن مسئله، برتری روش OOAD به روش شبه کد کاملاً مشخص می‌شود. اگر چه پردازش‌های مختلفی از OOAD وجود دارد، اما یک زبان گرافیکی برای نمایش نتایج هر فرآیند OOAD بیشتر از همه بکار گرفته شده است. این زبان *UML (Unified Modeling Language)* نام دارد که اواسط دهه ۱۹۹۰ تحت نظر سه نظریه‌پرداز نرم‌افزار بنام‌های Grady Booch، James Rumbaugh و Ivar Jacobson توسعه پیدا کرده است.

تاریخچه UML

در دهه ۱۹۸۰، تعداد سازمان‌هایی که شروع به استفاده از OOP برای ایجاد برنامه‌های کاربردی خود کردند بسیار افزایش یافت و از اینرو نیاز به یک استاندارد OOAD احساس گردید. تعدادی از نظریه‌پردازان شامل Booch، Rumbaugh و Jacobson، بطور جداگانه مبادرت به ایجاد و عرضه فرآیندهای برای برطرف کردن این نیاز کردند. هر فرآیند یا پردازه دارای نمادها یا «زبان» (به فرم دیاگرام‌های گرافیکی) متعلق بخود بود، تا حاصل نتایج تجزیه و تحلیل و طراحی‌ها باشد.

در اوایل دهه ۱۹۹۰، سازمان‌های مختلف و حتی قسمتهای موجود در درون یک سازمان، در حال استفاده از فرآیندهای منحصر بفرد با نمادهای متعلق بخود بودند. همزمان، این سازمان‌ها مایل به استفاده از ابزارهای نرم‌افزاری بودند که از فرآیندهای خاص آنها پشتیبانی کند. سازندگان نرم‌افزار متوجه مشکل تهیه ابزارهایی برای کار با این همه فرآیند متفاوت شدند. واضح بود که نیاز به یک نماد و فرآیند استاندارد بوجود آمده است.

در سال ۱۹۹۴، James Rumbaugh با همکاری Grady Booch در شرکت Rational Softwave (که اکنون بخشی از IBM است)، شروع به متحدالشکل کردن فرآیندهای خود کرد. بلافاصله Ivar Jacobson نیز به آنها پیوست. در سال ۱۹۹۶، گروه مبادرت به عرضه نسخه اولیه از UML به جامعه مهندسین نرم‌افزار کرد و پاسخ آنها را خواستار شدند. در همان زمان، سازمانی که بعنوان *OMG (Object Management Group)* شناخته می‌شود، دعوت به یک زبان مدل‌سازی مشترک از گروه‌های مختلف کرد. OMG (به آدرس www.omg.org) یک سازمان غیرتجاری است که مبادرت به نشر تکنولوژی‌های استاندارد شی‌گرا همانند UML می‌کند. چندین شرکت، از جمله HP، IBM، Microsoft، Oracle و Rational Software تشخیص داده بودند که نیاز به یک زبان مدل‌سازی مشترک است. در پاسخ به تقاضای OMG، این شرکت‌ها، همکاری UML خود را آغاز کردند، کنسرسیوم UML نسخه 1.1 را عرضه و تحویل OMG داد. OMG آن

را پذیرفته و در1997، مسئولیت ادامه و نگهداری UML را قبول کرد. در مارس ۲۰۰۳، OMG مبادرت به عرضه UML نسخه 1.5 کرد. UML نسخه 2 که در زمان نشر این کتاب مراحل پایانی خود را می‌گذارند، اصلی‌ترین نسخه از سال ۱۹۹۷ است. از اینرو بحث ما بر روی UML نسخه 2 خواهد بود.

UML چیست؟

در حال حاضر زبان UML یکی از پرکاربردترین طرح‌های نمایش گرافیکی برای مدل کردن سیستم‌های شی‌گرا است. این زبان انواع فرآیندهای موجود را متحد کرده است. برای مدل کردن سیستم‌ها از این زبان در سراسر کتاب استفاده کرده‌ایم.

یکی از ویژگی‌های جذاب UML در انعطاف‌پذیری آن است. UML بسط‌پذیر بوده (ویژگی‌های جدید را می‌پذیرد) و از هر فرآیند یا طریقه خاص OOAD مستقل است. مدل‌کننده‌های UML برای استفاده در طراحی سیستم‌ها مجانی هستند.

UML یک زبان مرکب با ویژگی‌های گرافیکی مناسب است. در بخش‌های «مبحث آموزشی مهندسی نرم‌افزار» در توسعه نرم‌افزار ماشین پرداخت اتوماتیک (ATM) نمایش ساده و زیرمجموعه‌ای از ویژگیت‌های UML عرضه خواهیم کرد. این مباحث آموزشی به دقت و تحت نظر اساتید دانشگاه و افراد حرفه‌ای طراحی شده‌اند. امیدواریم که از مطالعه و کار کردن با این مباحث لذت ببرید.

منابع اینترنت و وب UML

- **www.uml.org**

این صفحه UML از گروه OMG تدارک بیننده مستنداتی در ارتباط با UML و تکنولوژی‌های دیگر شی‌گرا است.

- **www.ibm.com/software/rational/uml**

این صفحه UML متعلق به IBM Rational است.

کتاب‌های توصیه شده

کتاب‌های متعددی در ارتباط UML به چاپ رسیده است. کتاب‌های توصیه شده در این بخش حاوی اطلاعاتی در مورد طراحی شی‌گرا با UML هستند.

- **Arlow,J.,and I.Neustadt.*UML and the Unified Process:Practical Object-Oriented Analysis and Design*. Lodon:Pearson Education Ltd.,2002.**
- **Fowler,M.*UML Distilled, Third Edition: A Breif Guide to the Standard Object Modeling Language*. Boston:Addison-Wesley,2004.**
- **Rumbaugh,J.,I.Jacobson and G.Booch. *The Unified Modeling Language User Guide*. Reading,MA:Addison-Wesley,1999.**

برای یافتن کتاب‌های دیگری در زمینه UML، می‌توانید به وب‌سایت‌های **www.amazon.com** یا **www.bn.com** مراجعه کنید. همچنین در وب سایت IBM لیستی از کتاب‌های UML وجود دارد:

www.ibm.com/software/rational/info/technical/books.jsp

خودآزمایی بخش ۱۷-۱

۱-۱ سه نمونه از شی‌های دنیای واقعی که در مورد آنها صحبت نکردیم، بیان کنید. برای هر شی، لیستی از صفات و رفتار آن تهیه نمائید.

۱-۲ شبه کد _____

a) کلمه دیگر OOAD است.

b) یک زبان برنامه‌نویسی بکار رفته برای نمایش دیاگرام‌های UML است.

c) یک مفهوم غیررسمی از بیان منطقی برنامه.

d) نمایش گرافیکی برای مدل‌سازی سیستم‌های شی‌گرا.

۱-۳ کاربرد اصلی UML در _____

a) تست سیستم‌های شی‌گرا.

b) طراحی سیستم‌های شی‌گرا.

c) پیاده‌سازی سیستم‌های شی‌گرا.

d) a و b.

پاسخ خودآزمایی بخش ۱۷-۱

۱-۱ [نکته: پاسخ‌ها می‌توانند متفاوت باشند.]. a) صفات یک تلویزیون عبارتند از سایز صفحه نمایش، تعداد رنگ‌هایی که می‌تواند بنمایش درآورد. کانال جاری و صدای آن. تلویزیون می‌تواند روشن یا خاموش شود، کانال‌های آن عوض شود، صدا و تصویر به نمایش در آورد. b) صفات قهوه‌ساز عبارتند از حداکثر آبی که می‌تواند در خود نگهداری کند، زمان تهیه قهوه و درجه حرارت صفحه قهوه‌ساز. قهوه‌ساز می‌تواند روشن و خاموش شود، قهوه را بجوشاند. c) صفات یک لاک‌پشت عبارتند از سن، سایز پوسته (لاک) و وزن آن. یک لاک‌پشت راه می‌رود و در لاک خود پنهان می‌شود.

۱-۲ c.

۱-۳ b.

۱-۱۸ منابع وب

وب سایت Deitel & Associates

- www.deitel.com/books/cppHTP5/index.html
- www.deitel.com
- www.deitel.com/newsletter/subscribe.html
- www.prenhall.com/deitel

کامپایلرها و ابزارهای توسعه

- www.thefreecountry.com/developercity/ccompilers.shtml
- msdn.microsoft.com/visualC
- www.borland.com/bcppbuilder
- www.compilers.net

- www.kai.com/C_plus_plus
- www.symbian.com/developer/development/cppdev.html

منابع

- www.hal9k.com/cug
- www.devx.com
- www.acm.org/crossroads/xrds3-2/ovp32.html
- www.accu.informika.ru/resources/public/terse/cpp.htm
- www.cuj.com
- www.research.att.com/~bs/homepage.html

بازی‌ها

- www.codearchive.com/list.php?go=0708
- www.mathools.net/c_c__/Games/
- www.gametutorials.com/Tutorials/GT/GT_Pg1.htm
- www.forum.nokia.com/main/0,6566,050_20,00.htm

خودآزمایی

۱-۱ جاهای خالی در عبارات زیر را با کلمات مناسب پر کنید:

a) شرکت ..... سبب محبوبیت کامپیوترهای شخصی شد.

b) کامپیوتر ..... به نحوی ساخته شده بود که از قابلیت محاسبات شخصی و کار در صنایع برخوردار بود.

c) کامپیوترها پردازش داده‌ها را تحت کنترل مجموعه‌ای از دستورالعمل‌ها بنام ..... کامپیوتری انجام می‌دهند.

d) شش واحد منطقی و کلیدی در یک کامپیوتر عبارتند از ..... ، ..... ، ..... ، ..... ، ..... ، و .....

e) سه کلاس زبان معرفی شده در این فصل عبارت بودند از ..... ، ..... و .....

f) برنامه‌های که مبادرت به ترجمه برنامه‌های زبان سطح بالا به زبان ماشین می‌کنند، ..... نام دارند.

g) از زبان C بطرز گسترده‌ای در توسعه سیستم عامل ..... استفاده شده است.

h) ..... زبانی است که توسط Wirth توسعه پیدا کرد و هدف از آن آموزش برنامه‌نویسی در دانشگاه‌ها است.

i) دپارتمان دفاع توسعه دهنده زبان Ada با قابلیتی بنام ..... است که به برنامه‌نویسان امکان می‌دهد تا تعدادی از فعالیت‌ها را بصورت موازی به انجام برسانند.

۱-۲ جاهای خالی در عبارات زیر که در ارتباط با محیط C++ هستند با کلمات مناسب پر کنید:

a) معمولا برنامه‌های C++ توسط یک برنامه ..... تایپ شده و وارد کامپیوتر می‌شوند.

b) در یک سیستم C++، برنامه ..... قبل شروع بکار فاز ترجمه کامپایلر اجرا می‌شود.

c) برنامه ..... مبادرت به ترکیب خروجی کامپایلر با انواع مختلفی از توابع کتابخانه‌ای برای تولید یک حالت اجرائی می‌کند.

d) برنامه ..... حالت اجرائی یک برنامه C++ را از روی دیسک به حافظه منتقل می‌کند.

۱-۳ جاهای خالی را که در ارتباط با مبحث شی‌گرا هستند با کلمات مناسب پر کنید:

a) شی‌ها دارای ویژگی ..... هستند، با اینکه شی‌ها از نحوه برقراری ارتباط با یک شی دیگر از طریق واسط مطلع هستند، معمولا از نحوه ساختار داخلی سایر شی‌ها اطلاعی ندارند.

b) برنامه‌نویسان C++ متمرکز بر ایجاد ..... هستند که حاوی اعضای داده و توابع عضو می‌باشند.

c) کلاس‌ها می‌تواند با سایر کلاس‌ها رابطه داشته باشند. این رابطه ..... نامیده می‌شود.

d) از منظر شی‌گرا، فرآیند تحلیل و طراحی یک سیستم ..... نامیده می‌شود.

e) OOD از مزیت رابطه ..... سود می‌برد، که در آن شی‌های جدید صفات کلاس‌های قبلی را جذب می‌کنند.

f) زبان ..... یک زبان گرافیکی است که طراحان سیستم‌های نرم‌افزاری امکان نمایش استاندارد طرح‌ها را فراهم می‌آورد.

g) سایز، شکل، رنگ و وزن یک شی نشاندهنده ..... آن شی است.

پاسخ خودآزمایی

۱-۱ a) اپل، b) کامپیوتر شخصی IBM ، c) برنامه، d) واحد ورودی، واحد خروجی، واحد حافظه، واحد محاسبه و منطق، واحد پردازش مرکزی، واحد ذخیره‌سازی ثانویه. e) زبان ماشین، زبان اسمبلی، زبان سطح بالا. f) کامپایلرها. g) یونیکس. h) پاسکال. i) multitasking

۱-۲ a) ویرایشگر. b) پیش‌پردازنده. c) linker d) loader

۱-۳ a) پنهان‌سازی اطلاعات b) کلاس‌ها c) وابستگی d) تحلیل و طراحی شی‌گرا (OOD). e) توارث f) UML g) صفت

تمرینات

۱-٤ هر کدامیک از ایتم‌های زیر را در دو دسته نرم‌افزار و سخت‌افزار رده‌بندی کنید:

a) CPU

b) کامپایلر C++

c) ALU

d) پیش‌پردازنده C++

e) واحد ورودی

f) برنامه ویرایشگر

۱-۵ به کدام دلیل مایل هستید تا برنامه‌ای بنویسید که مستقل از زبان ماشین باشد بجای اینکه وابسته به زبان ماشین باشد؟ چرا یک زبان وابسته به ماشین در نوشتن برخی از برنامه‌های خاص مناسب است؟

۱-٦ جاهای خالی را در عبارات زیر با کلمات مناسب پر کنید:

a) کدام واحد منطقی کامپیوتر اطلاعات خارج از آن را برای استفاده کامپیوتر دریافت می‌کند؟.....

b) پردازش دستورالعمل توسط کامپیوتر برای حل مسئله‌ای ..... نامیده می‌شود.

c) کدام نوع از زبان کامپیوتر از زبانی شبیه به زبان مخفف شده انگلیسی در دستورالعمل‌های زبان ماشین استفاده می‌کند؟.....

d) کدام واحد منطقی کامپیوتر اطلاعات پردازش شده توسط خود را به به دستگاه‌های دیگر که احتمالا در خارج از کامپیوتر هم می‌توانند قرار داشته باشند، ارسال می‌کند؟.....

e) کدام واحد منطقی کامپیوتر مبادرت به نگهداری اطلاعات می‌کند؟.....

f) کدام واحد منطقی کامپیوتر مسئول انجام محاسبات است؟.....

g) کدام واحد منطقی کامپیوتر مسئول تصمیم‌گیری است؟.....

h) برنامه‌نویسی با سطح زبان ..... برای بسیاری از برنامه‌نویسان مناسب بوده و تولید برنامه در آنها سریعتر صورت می‌گیرد.

i) زبان ..... تنها زبانی است که یک کامپیوتر می‌تواند مستقیما آن را درک کند.

j) کدام واحد منطقی کامپیوتر مسئول هماهنگی مابین سایر قسمتهای منطقی است؟.....

۱-۷ به چه دلایلی این روزها توجه زیادی به برنامه‌نویسی شی‌گرا و انجام آن بویژه توسط C++ می‌شود؟

۱-۸ وجه تمایز مابین عبارت خطای عظیم با خطای غیر عظیم در چیست؟