دانشگاه پیام نور

زبان برنامه‌نویسی ۲

(رشتهٔ علم اطلاعات و دانش شناسی)

مهندس صلاح رستمی **مهندس یونس زارعی**

«امروزه کتاب‌خوانی و علم‌آموزی، نه تنها یک وظیفه‌ی ملی، که یک واجب دینی است.»[1]

در عصر حاضر یکی از شاخصه‌های ارزیابی رشد، توسعه و پیشرفت فرهنگی هر کشوری میزان تولید کتاب، مطالعه و کتاب‌خوانی مردم آن مرز و بوم است. ایران اسلامی نیز از دیرباز تاکنون با داشتن تمدنی چندهزارساله و مراکز متعدد علمی، فرهنگی، کتابخانه‌های معتبر، علما و دانشمندان بزرگ با آثار ارزشمند تاریخی، سرآمد دولت‌ها و ملت‌های دیگر بوده و در عرصه‌ی فرهنگ و تمدن جهانی به‌سان خورشیدی تابناک همچنان می‌درخشد و با فرزندان نیک‌نهاد خویش هنرنمایی می‌کند. چه کسی است که در دنیا با دانشمندان فرزانه و نام‌آور ایرانی همچون ابوعلی سینا، ابوریحان بیرونی، فارابی، خوارزمی و ... همچنین شاعران برجسته‌ای نظیر فردوسی، سعدی، مولوی، حافظ و ... آشنا نباشد و در مقابل عظمت آنها سر تعظیم فرود نیاورد. تمامی این افتخارات ارزشمند، برگرفته از میزان عشق و علاقه فراوان ملت ما به فراگیری علم و دانش از طریق خواندن و مطالعه منابع و کتاب‌های گوناگون است. به شکرانه‌ی الهی، تاریخ و گذشته ما، همیشه درخشان و پربار است. ولی اکنون در این زمینه در چه جایگاهی قرار داریم؟ آمار و ارقام ارائه‌شده از سوی مجامع و سازمان‌های فرهنگی در مورد سرانه‌ی مطالعه‌ی هر ایرانی، برایمان چندان امیدوارکننده نمی‌باشد.

کتاب، دروازه‌ای به سوی گستره‌ی دانش و معرفت است و کتاب خوب، یکی از بهترین ابزارهای کمال بشری است. همه‌ی دستاوردهای بشر در سراسر عمر جهان، تا آنجا که قابل کتابت بوده است، در میان دست‌نوشته‌هایی است که انسان‌ها پدید آورده و می‌آورند. در این مجموعه‌ی بی‌نظیر، تعالیم الهی، درس‌های پیامبران به بشر، و همچنین علوم مختلفی است که سعادت بشر بدون آگاهی از آنها امکان‌پذیر نیست. کسی که با دنیای زیبا و زندگی‌بخش کتاب ارتباط ندارد بی‌شک از مهم‌ترین دستاورد انسانی و نیز از بیشترین معارف الهی و بشری محروم است. با این دیدگاه، به‌روشنی می‌توان ارزش و مفهوم رمزی عمیق در این حقیقت تاریخی را دریافت که اولین خطاب خداوند متعال به پیامبر گرامی اسلام(ص) این است که »بخوان!« و در اولین

---

۱. پیام مقام معظم رهبری به مناسبت آغاز هفته کتاب ۷۲/۱۰/۴

سوره‌ای که بر آن فرستاده‌ی عظیم‌الشأن خداوند، فرود آمده، نام «قلم» به تجلیل یاد شده‌است: «إِقْرَأْ وَ رَبُّکَ الْاَکْرَمُ. اَلَّذی عَلَّمَ بِالْقَلَم» در اهمیت عنصر کتاب برای تکامل جامعه‌ی انسانی، همین بس که تمامی ادیان آسمانی و رجال بزرگ تاریخ بشری، از طریق کتاب جاودانه مانده‌اند.

دانشگاه پیام‌نور با گستره‌ی جغرافیایی ایران‌شمول خود با هدف آموزش برای همه، همه‌جا و همه‌وقت، به‌عنوان دانشگاهی کتاب‌محور در نظام آموزش عالی کشورمان، افتخار دارد جایگاه اندیشه‌سازی و خردورزی بخش عظیمی از جوانان جویای علم این مرز و بوم باشد. تلاش فراوانی در ایام طولانی فعالیت این دانشگاه انجام پذیرفته تا با بهره‌گیری از تجربه‌های گرانقدر استادان و صاحب‌نظران برجسته کشورمان، کتاب‌ها و منابع آموزشی درسی شاخص و خودآموز تولید شود. در آینده هم، این مهم با هدف ارتقای سطح علمی، روزآمدی و توجه بیشتر به نیازهای مخاطبان دانشگاه پیام‌نور با جدیت ادامه خواهد داشت. به‌طور قطع استفاده از نظرات استادان، صاحب‌نظران و دانشجویان محترم، ما را در انجام این وظیفه‌ی مهم و خطیر یاری‌رسان خواهد بود. پیشاپیش از تمامی عزیزانی که با نقد، تصحیح و پیشنهادهای خود ما را در انجام این وظیفه‌ی خطیر یاری می‌رسانند، سپاسگزاری می‌نماییم. لازم است از تمامی اندیشمندانی که تاکنون دانشگاه پیام‌نور را منزلگه اندیشه‌سازی خود دانسته و ما را در تولید کتاب و محتوای آموزشی درسی یاری نموده‌اند، صمیمانه قدردانی گردد. موفقیت و بهروزی تمامی دانشجویان و دانش‌پژوهان عزیز آرزوی همیشگی ما است.

دانشگاه پیام‌نور

فهرست مطالب

پیشگفتار

در دنیای امروز، نرم‌افزارهای رایانه‌ای نقش بسیار مهمی را در زندگی بشر ایفا کرده به‌طوری‌که با یک نگاه ساده به اطراف خود می‌توان انبوه برنامه‌ها و نرم‌افزارهای رایانه‌ای را مشاهده کرد. تمامی برنامه‌های موجود در گوشی‌های همراه، رایانه‌های شخصی، سیستم‌های کنترلی، سیستم‌های مخابراتی، لوازم صوتی و تصویری، خودروها و بسیاری موارد دیگر تنها بخشی از مثال‌های موجود هستند.

از طرفی، تنوع و گسترهٔ تکنولوژی باعث شده که علم برنامه‌نویسی به یک علم کاملاً بزرگ و پیچیده تبدیل شده و شاخه‌های بسیار زیادی را شامل شود. بسیاری از زبان‌های مختلف برنامه‌نویسی عمر چندان زیادی نداشتند و تنها برخی از آن‌ها توانسته‌اند نیازهای بشر در تولید نرم‌افزار را برآورده کنند. در این میان، زبان C/C++، یکی از معروف‌ترین و پرکاربردترین زبان‌های موجود در دنیا بوده که بسیاری از نرم‌افزارهای کنونی، به‌وسیلهٔ آن نوشته شده است. وجود خواص و ابزارهایی نظیر سطح میانی، سرعت پردازش بالا، قابلیت برنامه‌نویسی شیءگرا، کار با سخت‌افزار و موارد دیگر را می‌توان از جمله جذابیت‌های این زبان دانست.

نگارش این کتاب به‌گونه‌ای است که در آن هدف اصلی آموزش زبان برنامه‌نویسی C/C++ برای دانشجویان عمل اطلاعات و دانش شناسی است که اصولاً با این مفاهیم آشنایی نداشته‌اند. تمامی مطالب بیان شده در این کتاب به‌صورت قدم‌به‌قدم و با ذکر انواع مثال‌های مختلف تنظیم شده است به‌گونه‌ای که زبان‌آموزان بتوانند به‌راحتی با این زبان مهم ارتباط برقرار کنند.

تقدیم به مهربان فرشتگانی که لحظات ناب باور بـودن، لـذت و غـرور دانسـتن، جسارت خواستن، شکوه توانستن، عظمت رسیدن و تمـام تجربـه‌هـای یکتـا و زیبـای زندگی‌ام، مدیون حضور سبز آن‌هاست.

مؤلفان

فصل اول

تابع

هدف کلی

در این فصل دانشجویان با انواع تابع آشنا خواهند شد، خصوصیات، ویژگی‌ها و مزایای تابع مورد بحث قرار خواهد گرفت، به‌طوری‌که در پایان فصل دانشجو بتواند برنامه‌ای بنویسید که از مفهوم تابع بهره ببرد.

هدف‌های یادگیری

۱. آشنایی با سبک‌های برنامه‌نویسی را یاد خواهیم گرفت.
۲. نحوهٔ نوشتن توابع ساده را مرور خواهیم کرد.
۳. نحوهٔ نوشتن توابع بازگشتی را یاد خواهیم گرفت.
۴. مفهوم چندریختی[1] را مرور خواهیم کرد.

مقدمه

استفاده از توابع در برنامه‌نویسی با افزایش تجربه و مهارت کد نویسی بیشتر می‌شود. تجربه نشان داده‌شده که کد نویسان مبتدی علاقهٔ چندانی به استفاده از توابع ندارند اما هرچه تبحر برنامه‌نویسی بیشتر می‌شود، اهمیت و قدرت توابع در توسعه کد بیشتر نمایان خواهد شد به‌طوری‌که برنامه نویسان حرفه‌ای اکثر کدهای خود را با استفاده از توابع می‌نویسند، در این فصل سعی می‌کنیم با فرمت کلی تابع آشنا شده، سپس اعلان

1. Polymorphism

توابع، انواع توابع، استفاده از متغیرها و اشاره‌گرها، مفاهیم چندریختی در توابع و مثال‌هایی از هرکدام را در ادامه یاد خواهیم گرفت.

۱-۱ نکات و مفاهیم پایه

زبان‌های برنامه‌نویسی متعددی وجود دارند و برنامه نویسان، دستورالعمل‌های برنامه را به زبان‌های مختلفی می‌نویسند که بعضی از آن‌ها مستقیماً توسط کامپیوتر قابل‌درک هستند ولی بعضی دیگر باید به زبان ماشین (زبانی که کامپیوتر آن را درک می‌کند)، ترجمه شوند. صدها زبان برنامه‌سازی وجود دارند که هرکدام از آن‌ها برای اهداف خاصی طراحی‌شده‌اند. تمام این زبان‌ها را می‌توان به سه دسته تقسیم کرد:

- زبان‌های ماشین (زبان‌هایی که فقط از کدهای ۰ و ۱ استفاده می‌کنند).
- زبان‌های اسمبلی (زبان‌هایی که از نمادها و علامت‌های خاصی استفاده می‌کنند).
- زبان‌های سطح بالا (زبان‌های برنامه‌نویسی‌ای مانند C، پاسکال و یا Fortran هستند که برنامه‌نویس را قادر می‌سازند برنامه‌ای بنویسد که مستقل از کامپیوتر است). چنین زبان‌هایی سطح بالا در نظر گرفته می‌شوند چرا که آن‌ها به زبان انسان نزدیک‌تر و از زبان ماشین دور هستند یا به عبارتی زبان‌های سطح بالا زبان‌های هستند که مستقل از ماشین‌اند و نوشتن، خواندن، ویرایش و درک آن‌ها راحت است.

C++ یکی از زبان‌های سطح بالاست که به زبان محاوره‌ای نزدیک است. این زبان در اوایل دهه ۱۹۸۰، از زبان C توسعه یافت. زبان C++ برای اغلب کامپیوترها وجود دارد و مستقل از سخت‌افزار است. دو سبک برنامه‌نویسی متداول وجود دارد که برای هرکدام از این سبک‌ها، زبان‌هایی طراحی و پیاده‌سازی شده‌اند:

- سبک برنامه‌نویسی ساخت‌یافته[1]
- سبک برنامه‌نویسی شیءگرا[2]

۱-۲ برنامه‌نویسی ساخت‌یافته

در دهۀ ۱۹۶۰ میلادی، تولید بسیاری از نرم‌افزارها با مشکل مواجه شدند. زمان‌بندی تولید نرم‌افزار به تأخیر می‌افتاد، هزینه‌ها بالابود و درنتیجه بودجۀ تولید نرم‌افزار افزایش

1. Structured Programing
2. Object Oriented Programing

می‌یافت و نرم‌افزار تولیدی نیز از قابلیت اعتماد بالایی برخوردار نبوده است. تولیدکنندگان نرم‌افزار به این نتیجه رسیدند که تولید نرم‌افزار مشکل‌تر از چیزی است که در مورد آن تصور می‌شود. تحقیقاتی که برای برطرف کردن مشکلات به عمل آمد، منجر به برنامه‌نویسی ساخت‌یافته شد. برنامه‌نویسی ساخت‌یافته، روش منظمی برای نوشتن برنامه‌ها است و منجر به نوشتن برنامه‌هایی می‌شود که خوانایی آن‌ها بالا است، تست و اشکال‌زدایی آن‌ها راحت‌تر و اصلاح آن‌ها آسان‌تر است.

در برنامه‌نویسی ساخت‌یافته، برنامه به‌صورت مجموعه‌ای از فعالیت‌ها تصور می‌شود که باید بر روی‌داده‌ها انجام شوند. در این روش، هر مسئله پیچیده‌ای، به مجموعه‌ای از مسئله‌های کوچک تجزیه می‌شود تا اینکه قابل‌درک باشد. به‌عبارت‌دیگر، برنامه‌نویس سعی می‌کند رویه‌هایی بنویسد که نیازمندی‌های سیستم را برآورده کنند.

به‌عنوان مثالی از برنامه‌نویسی ساخت‌یافته، محاسبهٔ میانگین حقوق کارمندان یک شرکت را در نظر بگیرید. این کار، فعالیتی پیچیده است و به‌کارهای کوچک‌تری تقسیم می‌شود[2]:

۱. حقوق هر کارمند را مشخص کنید.
۲. تعداد کارکنان را تعیین کنید.
۳. مجموع حقوق تمام افراد را تعیین کنید (محاسبهٔ مجموع حقوق).
۴. مجموع حقوق را بر تعداد افراد تقسیم کنید.

اما فعالیت محاسبهٔ مجموع حقوق می‌تواند به‌کارهای کوچک‌تری تقسیم شود:

۱. رکورد کارمند را بازیابی کنید (دستیابی به رکورد کارمند)
۲. حقوق کارمند را بیابید.
۳. این حقوق را به مجموع حق‌هایی که تاکنون به‌دست آوردید اضافه کنید.
۴. رکورد کارمند بعدی را بازیابی کنید.
۵. درصورتی‌که کارمندی باقی‌مانده است، به مرحلهٔ ۱ بروید.

به همین ترتیب، فعالیت دستیابی به رکورد کارمند می‌تواند به‌کارهای کوچک‌تری تقسیم شود:

۱. فایل کارکنان را بازکنید.
۲. به رکورد موردنظر بروید.
۳. رکورد را از روی دیسک بخوانید.

برنامه‌نویسی ساخت‌یافته، روش موفقی برای حل مسائل پیچیده است، اما مشکلات خاص خودش را دارد. در این روش، داده‌ها از فعالیت (متدهایی) که آن‌ها را پردازش می‌کنند جدا است. وقتی حجم داده‌ها زیاد می‌شود، نگهداری آن‌ها مشکل می‌شود.

۱-۳ برنامه‌نویسی شیءگرا

برنامه‌نویسی شیءگرا شیوهٔ نوینی است که قطعات نرم‌افزاری را ایجاد می‌کند که در برنامه‌های مختلف مورداستفاده قرار می‌گیرند. همان‌طور که کامپیوتر از قطعات سخت‌افزاری ساخته می‌شود، در برنامه‌نویسی شیءگرا، برنامه از قطعات نرم‌افزاری ساخته می‌شود. به‌این‌ترتیب، سرعت تولید نرم‌افزار افزایش می‌یابد. قابلیت خوانایی برنامه‌هایی که در این روش نوشته می‌شوند بالا بوده، تست، عیب‌یابی و اصلاح آن‌ها آسان است.

با بعضی از اصطلاحات مهم در برنامه‌نویسی شیءگرا شروع می‌کنیم. به دنیای اطراف خود بنگرید. به هر جا که نگاه می‌کنید، اشیایی را می‌بینید: مردم، حیوانات، گیاهان، اتومبیل‌ها، هواپیماها، کامپیوترها و مانند آن‌ها. انسان‌ها، بر اساس اشیا فکر می‌کنند. ما توانایی عجیبی از انتزاع[1] داریم که ما را قادر می‌سازد تا به‌جای اینکه تصاویر صفحه‌نمایش را به‌صورت نقاط منفردی از رنگ‌ها (که در گرافیک، پیکسل نامیده می‌شود) در نظر بگیریم، آن‌ها را به‌عنوان اشیایی مثل مردم، هواپیماها، درخت‌ها و کوه تصور کنیم. در صورت لزوم، می‌توانیم به‌جای اینکه به ذرات شن فکر کنیم، به ساحل فکر کنیم و به‌جای اینکه به درخت‌ها فکر کنیم به جنگل فکر کنیم و به‌جای اینکه به آجرها فکر کنیم، به ساختمان‌ها فکر کنیم.

اشیا را می‌توان به دو دسته تقسیم کرد:

- اشیای بی‌جان
- اشیای جاندار.

اشیای جاندار زنده‌اند، حرکت می‌کنند و کارهایی را انجام می‌دهند. اشیای بی‌جان، مانند سنگ به نظر نمی‌رسند که کاری انجام دهند. تمام اشیا (چه جان‌دار و چه

1. Abstraction

بی‌جان)، چیزهایی مشترک دارند. آن‌ها صفاتی[1] مثل اندازه، شکل و وزن دارند و همهٔ آن‌ها رفتارهایی[2] را از خود نشان می‌دهند. به‌عنوان‌مثال، توپ می‌غلتد، بالا و پایین می‌رود، پُرباد می‌شود یا باد آن خالی می‌شود. بچه گریه می‌کند، می‌خوابد، می‌خندد، راه می‌رود و چشمک می‌زند. اتومبیل شتاب می‌گیرد، ترمز می‌کند و روشن می‌شود.

انسان اشیا را از طریق مطالعهٔ صفات و مشاهدهٔ رفتار آن‌ها می‌شناسد. اشیای مختلف می‌توانند صفات مشابهی داشته باشند و رفتارهای یکسانی را از خود نشان دهند. به‌عنوان‌مثال، می‌توان بچه‌ها و بزرگ‌ترها را باهم و یا انسان را با حیوان مقایسه کرد. اتومبیل‌های سواری و کامیون‌ها نیز چیزهای مشترکی دارند[7].

۱-۴ چرا در برنامه از توابع استفاده می‌کنیم؟

استفاده از توابع در برنامهٔ کامپیوتری باعث می‌شود که هر مسئله به بخش‌های منطقی کوچک‌تری تقسیم شود. بدین ترتیب، به‌جای یک مسئلهٔ بزرگ، با مسئله‌های کوچک‌تری سروکار خواهیم داشت و آن‌ها را با نوشتن توابع حل خواهیم کرد. امتیازات نوشتن توابع را می‌توان به‌صورت زیر برشمرد:

۱. برنامه‌های پیچیده به بخش‌های کوچک‌تری تقسیم می‌شوند و هر بخش توسط تابعی نوشته می‌شود. دستورالعمل‌ها و داده‌های موجود در تابع، مستقل از سایر بخش‌های برنامه است.

۲. همکاری بین افراد را فراهم می‌کند، به‌طوری‌که افراد مختلف می‌توانند بخش‌های مختلفی از برنامه را بنویسند.

۳. اشکال‌زدایی برنامه‌های حاوی توابع، ساده‌تر است. اگر برنامه مشکلی داشته باشد، بررسی تابعی که این اشکال در آن به‌وجود آمده است، ساده است.

۴. برنامه‌نویسی با استفاده از توابع، موجب صرفه‌جویی در وقت می‌شود. بدین ترتیب که اگر تابعی بنویسید که عملی را در برنامه‌ای انجام دهد، می‌توانید آن تابع را در برنامهٔ دیگری که به این عمل نیاز دارد، به‌کار ببرید. حتی اگر با تغییر اندکی در توابع نوشته‌شده، بتوانید آن‌ها را در برنامه‌های دیگر به‌کار ببرید، بازهم مقرون‌به‌صرفه است.

---

1. Attributes
2. Behavior

۱-۵ توابع

باید به این نکته اشاره کرد که تنها کلمات کلیدی و سوئیچ‌های کامپایلر (یکسری دستورات شناخته شده هستند که مربوط به کامپایلر می‌شوند. این دستورها با علامت # مشخص می‌شوند) نیاز به تعریف ندارند و تمامی دستورهای دیگر را لازم است یا تعریف کنیم یا آنکه از قبل تعریف کرده و درون کتابخانه قرارداد. دستورهای موجود درون این کتابخانه‌ها در حقیقت همگی توابعی هستند که هنگام نیاز می‌توان از آن‌ها استفاده کرد.

نکتهٔ دیگری که در مورد زبان C/C++ بیان‌شده این است که این زبان به‌صورت خط به خط برنامه را اجرا می‌کند و بنابراین هنگامی‌که قرار است از یک تابع استفاده شود، لازم است قبل از آنکه برنامه به خط استفاده برسد، این تابع تعریف‌شده باشد. این نکته هنگام تعریف توابع و استفاده از آن‌ها بسیار مهم بوده و لازم است که همواره مدنظر قرار بگیرد.

۱-۶ نکاتی در مورد نوشتن توابع

ابتدا بدون پرداختن به جزئیات پیاده‌سازی توابع، آرگومان‌ها و نتیجه‌ای را که از توابع انتظار دارید، مشخص کرده، برنامه اصلی را بنویسید؛ به‌عبارت‌دیگر، در قدم اول لازم نیست به جزئیات پیاده‌سازی تابع بپردازید. پس از نوشتن برنامه اصلی، توابع دیگر را بنویسید. تابع چه وظیفه‌ای بر عهده دارد؟ ورودی‌های تابع چیست؟ خروجی‌های تابع کدام‌اند؟

توابع را طوری طراحی و پیاده‌سازی کنید که هر تابع فقط به آنچه نیاز دارد دسترسی داشته باشد و بقیهٔ قسمت‌های برنامه و سایر اطلاعات، توسط توابع غیر مرتبط، قابل‌دستیابی نباشد. این موضوع را پنهان‌سازی اطلاعات[1] گویند. برای این منظور، هر تابع باید یک نقطهٔ ورودی و یک نقطهٔ خروج داشته باشد. برای ارتباط بین توابع، از آرگومان‌ها و پارامترها استفاده کنید. این مفهوم از همان واژهٔ Capsule گرفته شده است؛ به همان کپسولی اطلاق می‌شود که وقتی مریض می‌شویم، میل می‌کنیم. کپسول چیزی است که غشایی دور چیزی یا چیزهای خاصی ایجاد می‌کند تا علاوه بر دور هم نگه‌داشتن آن‌ها کنار یکدیگر، از آن‌ها محافظت نیز کند.

---

1. Encapsulation

۷-۱ فرمت کلی یک تابع

در زبان برنامه‌نویسی C/C++، یک تابع بسیار مهم وجود دارد که به نام main معروف بوده و همواره برنامهٔ C/C++ با این تابع شروع به‌کار می‌کند؛ بنابراین هنگام نوشتن یک برنامه به زبان C/C++، لازم است این تابع حتماً موجود باشد؛ اما دیگر توابع را بسته به نیاز می‌توان تعریف کرده و استفاده کرد[8].

توابعی که در برنامه‌نویسی تعریف می‌شوند مشابه با توابع ریاضی بوده و دارای سه خاصیت اساسی هستند که عبارت‌اند از:

۱. هر تابع دارای یک نام منحصربه‌فرد است که توسط آن شناخته می‌شود.
۲. هر تابع همچنین دارای تعدادی آرگومان است. این آرگومان‌ها می‌توانند از صفرتا هر تعداد دلخواه انتخاب شوند. همچنین آرگومان‌های یک تابع می‌توانند از نوع‌های متفاوتی انتخاب شوند.
۳. درنهایت هر تابع می‌تواند دارای یک مقدار بازگشتی باشد. این بدان معنی است که توابع بعدازآنکه محاسبات خود را انجام دادند، نتیجهٔ خود را که یک متغیر از نوع خاص است، برمی‌گردانند و یا اینکه توابعی وجود دارند که هیچ مقداری را برنمی‌گردانند.

به‌عنوان‌مثال توابع sin(x) و log(x,n) در ریاضی هرکدام دارای یک اسم خاص بوده و توسط آن اسم نیز شناخته می‌شوند. تابع sin(x) دارای یک آرگومان از نوع عدد حقیقی بوده درحالی‌که تابع log(x,n) دارای دو متغیر است. متغیر اول یک عدد حقیقی بوده که قرار است مقدار لگاریتم آن محاسبه شود. آرگومان دوم یک عدد طبیعی بوده که پایهٔ لگاریتم را تعیین می‌کند. به‌عنوان‌مثال log(100,10)=2 معرف مقدار لگاریتم عدد ۱۰۰ بر مبنای ۱۰ است. البته می‌توان پایه را نیز عدد حقیقی در نظر گرفت؛ ولی در اینجا هدف تنها نشان دادن یک مثال است. درنهایت هردوی این توابع یک مقدار حقیقی را برمی‌گردانند. با توجه به این مقدمه می‌توان فرمت کلی تعریف یک تابع را در زبان C/C++ بدین صورت بیان کرد:

```
1  <آرگومان‌های ورودی>  نام تابع  <نوع تابع>
2  {
3  دستورات تابع //
4  }
```

مقدار بازگشتی یک تابع می‌تواند از هر نوعی از متغیرهای موجود در زبان C/C++ باشد. حتی می‌تواند از جنس متغیرهای تعریف‌شده توسط Struct و غیره نیز باشد. این مقدار را می‌توان برابر با متغیری که هم‌جنس با آن است، قرارداد. به‌عنوان‌مثال می‌توان نوشت:

```
1  float x = sin (4.0f);
```

ازآنجاکه مقدار بازگشتی sin(4.0f) یک مقدار حقیقی است، با دستور فوق هنگامی‌که متغیر X تعریف می‌شود، مقدار آن نیز برابر با sin(4.0f) قرار می‌گیرد.

۸-۱ تعریف یک تابع

گفتیم که زبان C/C++ به‌صورت خط به خط اجرا می‌شود؛ بنابراین اگر قرار باشد یک تابع تعریف شود و در تابع main از آن استفاده شود، حتماً لازم است این تابع قبل از تابع main تعریف‌شده باشد. البته نه‌تنها در مورد تابع main بلکه چنانچه در هر تابع دیگری نیز یک تابع دیگر استفاده شود، لازم است ترتیب تعریف به‌درستی صورت بگیرد.

این موضوع دارای دو مشکل اساسی است:

۱. نوشتن یک برنامه در یک فایل که در آن تمام توابع تعریف‌شده باشند، بسیار حجیم و غیرمنطقی است.
۲. گاهی ممکن است حالتی رخ دهد که دو تابع بخواهند همدیگر را فراخوانی کنند. در این صورت مشخص نیست که کدام تابع باید اول نوشته شود.

برای رفع این مشکل در زبان C/C++ دو اصطلاح موجود عبارت‌اند از:

۱. اعلام تابع
۲. تعریف تابع

۸-۱-۱ اعلام تابع

اعلام یک تابع عبارت است از نوشتن مشخصات تابع بدون آنکه بدنه‌ای برای آن نوشته‌شده باشد. هنگامی‌که یک تابع اعلام می‌شود، از آن به بعد آن تابع برای کامپایلر

شناخته می‌شود؛ اما همان‌گونه که بیان شد، اعلام یک تابع بدون ذکر بدنه‌ای اصلی تابع بوده و بنابراین لازم است حتماً درجایی دیگر بدنهٔ تابع تعریف بشود.

اعلام یک جمله خبری است و هیچ حافظه‌ای نمی‌گیرد. همچنین ازآنجاکه اعلام تنها یک جمله خبری است، می‌توان آن را بارها و بارها به کاربرد بدون آنکه مشکلی به‌وجود بیاید.

۱-۸-۲ تعریف تابع

تعریف تابع عبارت است از نوشتن بدنه‌ای آن، ازآنجاکه هر تابع در زبان C/C++ باید تنها یک بدنه داشته باشد، لازم است که تعریف یک تابع فقط و فقط یک‌بار انجام شود. برای روشن شدن این موضوع برنامهٔ نوشته شده در شکل ۱-۱ را در نظر بگیرید. در این برنامه و در خطوط ۵ تا ۸ توابع موردنظر فقط اعلام‌شده‌اند. اعلام این توابع دقیقاً مانند تعریف آن‌هاست با این تفاوت که در اعلام، بدنه‌ای برای تابع نوشته‌نشده و در انتهای آن علامت نقطه‌ویرگول; استفاده‌شده است. همچنین دقت شود که تابع Power بیش از یک‌بار اعلام‌شده و همان‌گونه که قبلاً اشاره شد، تکرار آن هیچ مسئلهٔ برای کامپایلر ایجاد نمی‌کند[2].

```
1   #include "stdafx.h"
2   #include <iostream>
3   using namespace std;
4   //-----Declarations
5   void PrintFunction (int n);
6   float Power (float x, int n);
7   float Power (float x, int n);
8   float Power (float x, int n);
9
10  int main()
11  {
12   int j=5;
13   float x=8.0f;
14   float y=7.0f;
15  //
16   cout<<"j = "<<j<<" x = "<<x<<" y = "<<y<<endl;
17   PrintFunction(j);
18   y=Power(x,j);
```

```
19  cout<<"j = "<<j<<" x = "<<x<<" y = "<<y<<endl;
20   return 0;
21  }
22   //-----Definitions
23   float Power (float x, int n)
24   {
25    int i;
26    float pow=1;
27    for (i=0;i<n;i++)
28   {
29      pow=pow * x;
30   }
31   PrintFunction(n);
32   return pow;
33   }
34   void PrintFunction (int n)
35   {
36    int i;
37    for (i=0;i<n;i++)
38     {
39      cout<<i<<endl;
40     }
41  }
```

شکل ۱-۱. کاربرد اعلام و تعریف در زبان C/C++

در ادامه و بعد از تابع main توابع موردنظر نیز تعریف شده‌اند. همان‌طور که مشخص است، تعریف تابع یعنی آنکه تابع به‌طور کامل به همراه بدنهٔ آن نیز نوشته شود. دقت شود که اعلام و تعریف دقیقاً باید مشابه یکدیگر باشند.

```
j = 5  x = 8  y = 7
0
1
2
3
4
0
1
2
3
4
j = 5  x = 8  y = 32768
```

شکل ۱-۲. خروجی برنامه اعلان توابع

۱-۹ توابعی که هیچ مقداری برنمی‌گردانند

در برخی از موارد هدف از نوشتن توابع تنها انجام یک سری محاسبات است و نیازی نیست که این تابع مقداری را برگرداند. در این صورت از کلمه کلیدی void برای مقدار بازگشتی تابع می‌توان استفاده کرد. به‌عنوان‌مثال تابع زیر را در نظر بگیرید[1]:

```
1  void Function (void)
2  {
3  //Body Of the Function
4  }
```

این تابع هیچ آرگومانی نداشته و هیچ مقدار بازگشتی نیز برنمی‌گرداند. استفاده از چنین توابعی بسیار متداول بوده و وظیفهٔ آن‌ها تنها دسته‌بندی انجام محاسبات است.

برای انتخاب اسم توابع تنها می‌توان از حروف و اعداد و خط زیر استفاده کرد. اگرچه از عدد هم می‌توان در نام‌گذاری توابع استفاده کرد، ولی باید دقت کرد که اسم تابع نمی‌تواند با عدد شروع شود. تعداد حروف به‌کار رفته برای اسم نیز ۱۲۸ حرف بوده که کامپایلر در عمل ۶۴ حرف آن را بیشتر نمی‌شناسد؛ بنابراین لازم است هنگام استفاده از اسامی بلند به طول آن نیز دقت داشت. همچنین بهتر است در انتخاب اسم توابع همواره نامی را برگزید که معرف وظیفهٔ آن تابع بوده تا هنگام برنامه‌نویسی از ایجاد سردرگمی پرهیز شود.

۱-۱۰ توابعی که مقداری برمی‌گردانند

به‌عنوان‌مثال، برنامهٔ نوشته شده در شکل ۱-۳ را در نظر بگیرید که یک عدد را گرفته و مقسوم‌های آن را محاسبه می‌کند.

همان‌طور که در برنامه نیز نمایش داده‌شده است می‌توان به سه روش پارامتر را به تابع ارسال کرد:

- مقداردهی مستقیم تابع در برنامه اصلی maghsom(4)، خط ۱۶ برنامه.
- مقداردهی تابع با مقدار اولیه متغیر در برنامه اصلی maghsom(x)، خط ۱۸ برنامه. مقداردهی تابع به‌وسیله مقدار ورودی متغیر از صفحه‌کلید در برنامه اصلی، خط ۲۱ برنامه.

```
1   #include "stdafx.h"
2   #include <iostream>
3   #include <conio.h>
4   using namespace std;
5   void maghsom (int n)
6   {
7      int i;
8   for (i=1; i<n;i++)
9        if(n%i == 0)
10       cout<<i<<"\t";
11       cout<<n<<"\n";
12  }
13  int main()
14  {
15  int x;
16    maghsom(4);
17    x = 10;
18    maghsom(x);
19    cout<<"enter x: ";
20    cin>>x;
21    maghsom(x);
22    cin.get();
23   return 0;
24  }
```

شکل ۱-۳. تعریف برنامه پیدا کردن مقسوم‌های یک عدد

در شکل ۱-۴ می‌توان نشان داد که چطور تابعی بنویسیم که متغیر x را از ورودی دریافت کرده مقدار متغیر y را از معادله‌های زیر به‌دست آورد.

$$y = \begin{cases} 3x - \frac{5}{2.0} & x < 0 \\ 2 & x = 1 \\ 3x + \frac{5}{2.0} & x > 1 \end{cases}$$

همان‌طور که ملاحظه می‌شود ابتدا تابع و دستورات آن نوشته می‌شود و سپس در سطر ۲۱ برنامه مستقیم تابع با مقدار ورودی که از صفحه‌کلید دریافت شده است، فراخوانی شده و مقدار محاسبه‌شده، چاپ می‌شود.