مبانی رایانه و برنامه نویسی به زبان

C++

به همراه ۳۵۰ برنامه حل شده

برای رشته های: مهندسی کامپیوتر، فناوری اطلاعات، ICT
رشته های فنی و مهندسی وعلوم

تالیف:
مهندس رمضان عباس نژادورزی
مهندس فاطمه عبدی سقاواز
مهندس باقر رحیم پور کامی
محمد علی شمعلی زاده بائی

برخی از عناوین مهم

آشنایی با رایانه، الگوریتم و فلوچارت
ساختار تصمیم و تکرار
توابع
آرایه ها، رشته ها و اشاره گرها
ساختمان، کلاس و وراثت
حل بیش از ۳۵۰ برنامه
بیان حدود ۳۰۰ برنامه حل نشده

مبانی رایانه و
برنامه‌نویسی به زبان
C++

تالیف:

مهندس رمضان عباس‌نژادورزی– مهندس فاطمه عبدی سقاواز
محمدعلی شمع‌علیزاده بائی – مهندس باقر رحیم‌پورکامی

فن‌آوری نوین

عنوان و نام پدیدآور	: مبانی ریانه و برنامه‌نویسی به زبان ++C/تالیف رمضان عباس‌نژادورزی... و[دیگران].
مشخصات نشر	: بابل:فن‌آوری نوین، ۱۳۹۳
مشخصات ظاهری	: ۳۰۴ص.:مصور،
شابک	: ۱۷۵۰۰۰ ریال:۵-۰۱-۷۲۷۲-۶۰۰-۹۷۸
یاداشت	: تالیف:رمضان عباس‌نژادورزی،فاطمه عبدی سقاواز، محمدعلی شمعلی زاده، باقر رحیم پور کامی
موضوع	: سی ++(زبان برنامه نویسی کامپیوتر)
موضوع	: الگوریتم‌های کامپیوتری
موضوع	: زبان برنامه نویسی کامپیوتر
شناسه افزوده	: عباس‌نژاد ورزی، رمضان، ۱۳۴۸ -
رده بندی کنگره	: ۲۳۱۳۹۲ م ۹۳ س/۷۶/۷۳ QA
رده بندی دیویی	: ۰۰۵ / ۷۳
شماره کتابشناسی ملی	: ۳۳۴۴۳۴۱

فن‌آوری نوین بابل، کدپستی ۴۷۱۶۷-۷۳۴۴۸ www.fanavarienovin.net تلفن: ۲۲۵۶۶۸۷-۰۱۱۱

مبانی رایانه و برنامه‌نویسی به زبان C++

تألیف: مهندس رمضان عباس‌نژادورزیی – مهندس فاطمه عبدی سقاواز- محمد علی شمعلی‌زاده- مهندس باقر رحیم‌پور کامی

نوبت چاپ: چاپ دوم

سال چاپ: زمستان ۱۳۹۳

شمارگان: ۱۰۰۰ جلد

قیمت: ۱۷۵۰۰ تومان

نام چاپخانه و صحافی: فرنگاررنگ

شابک: ۵-۰۱-۷۲۷۲-۶۰۰-۹۷۸

نشانی ناشر: بابل، چهارراه نواب، کاظم‌بیگی، جنب حسینیه منصور کاظم‌بیگی، طبقه همکف

طراح جلد: کانون آگهی و تبلیغات آبان (احمد فرجی)

تهران، خ اردیبهشت، نبش وحید نظری، پلاک ۱۴۲ تلفکس: ۶۶۴۰۰۲۲۰-۶۶۴۰۰۱۴۴

فهرست مطالب

فصل چهارم: توابع

فصل پنجم: آرایه‌ها، رشته‌ها و اشاره‌گرها

فیزیک الکتریسته	فن آوری نوین
درس و کنکور پایگاه داده	فن آوری نوین

مقدمه

گسترش و نقش آفرینی نرم‌افزارهای رایانه‌ای و استفاده روز افزون آن‌ها در بخش‌های مختلف نظیر صنعت، تجارت، پزشکی، علوم، اداری، حسابداری و غیره، روز به روز بر اهمیت یادگیری زبان مناسب برنامه‌نویسی می‌افزاید.

زبان‌های برنامه‌نویسی زیادی وجود دارند. در بین این زبان‌ها C++ از ویژگی‌های بخصوصی از لحاظ آموزشی و کاربردی برخودار است. به طوری که این زبان یکی از جذاب‌ترین و قدرت‌مندترین زبان‌های برنامه‌نویسی **شیءگرا** است. از طرف دیگر، زبان C++ به عنوان سرفصل درس مبانی برنامه‌سازی و برنامه‌سازی پیشرفته در رشته-های کامپیوتر، فناوری اطلاعات، ICT، علوم کامپیوتر و رشته‌های فنی و مهندسی تدریس می‌شود.

کتاب‌های زیادی در زمینه برنامه‌نویسی C++ نوشته شده‌اند که جای تقدیر و تشکر دارد. اما، کتاب حاضر با سال-ها تجربه در زمینه تالیف کتب برنامه‌نویسی و تدریس این زبان تدوین شده است و دارای ویژگی‌های زیر است:

- **بیان الگوریتم و فلوچارت** که پایه و اساس یادگیری هر زبان برنامه‌نویسی است.
- **ارائه و حل برنامه‌های متعدد،** در این کتاب حدود 506 مسئله برنامه‌نویسی بیان گردیده، که 180 برنامه آن در متن کتاب حل شده است. حل حدود 142 برنامه در سایت انتشارات فن‌آوری نوین آمده است. اما، 184 برنامه بدون حل باقی مانده است که به عهده دانشجویان محترم می‌باشد.
- در پیوست الف کتاب **حدود ۵۰ سوال تکمیلی و امتحانی** با توضیحات کامل حل گردیده است.
- در پیوست ب کتاب **پروژه مین روب** به عنوان یک پروژه عملی با توضیحات به طور کامل پیاده‌سازی شده است.
- **سادگی و روانی کتاب،** کتاب به صورت گام‌به‌گام با جملات کوتاه و ساده بیان گردیده است.

امیدواریم این اثر نیز مورد توجه اساتید و دانشجویان عزیز واقع شود.

مسائل حل شده در سایت که انتهای هر فصل کتاب آمده است را می‌توانید از آدرس سایت انتشارات فن-آوری نوین به آدرس www.fanavarienovin.net دریافت نمایید.

برنامه‌های کتاب در یکی از محیط‌های TC(Turbo C) و Borland C++ تحت ویندوز(این کامپایلر را می‌توانید در سیستم عامل ویندوز ۷ نیز نصب کنید) تست و اجرا شده‌اند که نمونه خروجی آن‌ها در کتاب آمده است. برنامه‌های که در آن‌ها محیط گرافیک فعال می‌شوند را باید با کامپایلر TC تست کرده و خروجی بگیرید.

در پایان از تمامی خوانندگان عزیز (اساتید و دانشجویان) تقاضا داریم، هرگونه اشکال، ابهام در متن کتاب، پیشنهادات و انتقادات را به آدرس پست الکترونیک fanavarienovin@gmail.com ارسال نمایند.

بابل، زمستان۱۳۹۳

مولفین

فصل ۱ آشنایی با رایانه، الگوریتم و فلوچارت

اولین قدم برای استفاده از هر ابزاری شناخت و بیان کاربردهای آن است. بنابراین، برای استفاده از رایانه باید آن را شناخت و کاربردهای آن را دانست. یعنی، ابتدا باید تعیین کرد رایانه چیست؟، چه ویژگی‌ها و کاربردهای دارد؟ یا مهم‌تر این که با چه دیدی باید به رایانه نگاه نمود تا سودمند باشد و ضرر نداشته باشد. بسیاری از اشخاص، ادارات، سازمان‌ها، حتی مهندسین، ... رایانه را به شکل فانتزی نگاه می‌کنند. آن‌ها انتظار دارند تنها با خرید یک رایانه، مشکلات سازمان و ادارات‌شان برطرف شود. این افراد رایانه را نشناختند و کاربردهای آن را نمی‌دانند. برای این که بتوان رایانه را شناخت، باید مشخص نمود چرا رایانه اختراع شد؟ چون به آن نیاز داشتند، آن را ساختند. پس، نیاز است که موجب اختراع تکنولوژی‌ها و ابزارهای جدید می‌شود. بنابراین، باید تعیین کرد که رایانه چه نیازهایی از بشر را برطرف می‌کند. پیچیدگی و سختی کارها، دقت و صحت، تکرار، حجم زیاد اطلاعات، انتقال اطلاعات، سرعت بالا و کاهش هزینه، مهم‌ترین عواملی هستند که **ریزی است که توسط ابزارهای ورودی داده را از** نیاز به رایانه را بیان می‌کنند. **رایانه دستگاه الکترونیکی قابل برنامه دنیای خارج دریافت نموده، توسط پردازش گر پردازش می‌نماید تا اطلاعات تولید گردد و اطلاعات را در حافظه نگهداری کرده، در پایان توسط ابزارهای خروجی اطلاعات را به دنیای خارج منتقل می‌کند.**

۱– ۱. انواع رایانه‌ها

برای آشنایی با انواع رایانه، فرض کنید بخواهید از نقطه‌ای به نقطه دیگر مسافرت کنید. برای انجام این کار وسائل نقلیه از قبیل ماشین، هواپیما، دوچرخه، قطار و غیره وجود دارند. اگر فاصله سفرتان طولانی نباشد، از ماشین استفاده خواهید کرد. ولی، اگر سفرتان طولانی باشد، از هواپیما استفاده می‌کنید. رایانه‌ها نیز کاربردهای مختلفی دارند. با توجه به انواع کاربردها، رایانه‌های متعددی ساخته شده‌اند. برخی از انواع رایانه‌ها عبارت‌اند از:

- **ابر رایانه‌ها**[1]، قدرتمندترین رایانه‌هایی هستند که تاکنون ساخته شده‌اند. در ساختمان این رایانه‌ها، تعداد زیادی پردازنده وجود دارند که با همکاری هم کار می‌کنند. در سال، تعداد اندکی از این رایانه تولید می‌شوند، چون سازمان‌های کمی در دنیا به چنین توانایی و قدرت پردازش نیاز دارند. از طرف دیگر، هزینه‌های تولید این رایانه‌ها بسیار بالا است. بنابراین، هر سازمانی نمی‌تواند چنین هزینه‌هایی را بپردازد.

[1] .Supper Computers [2].Main Frame Computers

رایانه‌های بزرگ[2]، یک پردازنده قدرتمند دارند که به طور موازی چندین کاربر می‌توانند از آن استفاده کنند. این رایانه‌ها، دیگر تولید نمی‌شوند و به تدریج از رده خارج شدند. افزایش قدرت کارایی و کاهش قیمت رایانه‌های شخصی امروزی، این رایانه‌ها را غیر قابل استفاده کرده است.

رایانه‌های کوچک[2]، برای استفاده در سازمان‌ها، ادارات و شرکت‌های متوسط ساخته شده‌اند. این اندازه‌های متعددی دارند. برخی از آن‌ها رومیزی هستند یا اندازه آن‌ها تا یک کابینت متغیر است.

رایانه‌های شخصی[2]، رایانه‌های امروزی هستند که به رایانه‌های رومیزی معروفند. امروزه، این رایانه‌ها در اکثر منازل، ادارات و سازمان‌ها وجود دارند و کاربردهای عمومی دارند. زیرا، نه تنها قیمت آن‌ها بسیار پایین است، بلکه ممکن است صدها برابر یک رایانه بزرگ قدرت داشته باشند.

رایانه‌های کیفی[3]، رایانه‌های قابل حمل هستند که می‌توانند با باطری و برق شهر کار کنند. ویژگی قابل حمل بودن این رایانه‌ها موجب شده است که انسان‌ها بتوانند در مسافرت، کارهای خودشان را انجام دهند. یعنی، کارمندان بدون این که به اداره محل خدمتشان مراجعه کنند، می‌توانند کارهایشان را انجام دهند. این روش انجام کار، **دور کاری** نام دارد.

رایانه‌های جیبی[4]، همان‌طور که از نام آن‌ها پیداست، اندازه آن‌ها کوچک و وزن آن‌ها نیز کم است. از طرف دیگر، در جیب جا می‌شوند. بنابراین، به سادگی قابل حمل هستند.

۲– ۱. سخت‌افزار و نرم‌افزار

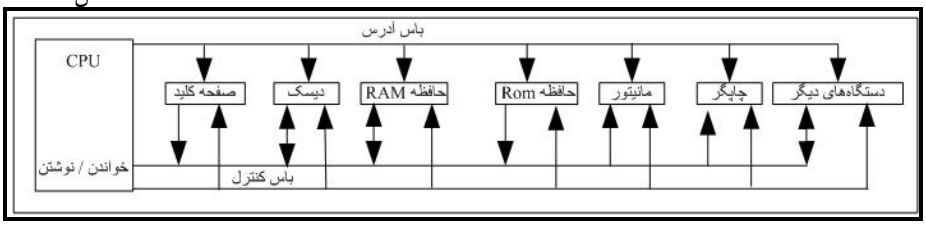
هر فردی که با رایانه آشنایی دارد، واژه‌های سخت‌افزار و نرم‌افزار را شنیده است. برای درک بهتر این مفاهیم یک کلاس آموزش رایانه را در نظر بگیرید. در این کلاس، میز، صندلی، تخته، ویدئو پروژکتور، رایانه، استاد، دانشجو و ابزارهای دیگر وجود دارند. تمام این ابزارها و افرادی که در کلاس وجود دارند، سخت‌افزار کلاس درس می‌باشند. چون، قابل رویت و لمس هستند. بنابراین، تمام چیزهایی که قابل لمس و رویت باشند، **سخت‌افزار**[5] نام دارند. در رایانه، دستگاه‌هایی از قبیل صفحه کلید، صفحه نمایش، ماوس، دسته بازی، حافظه، CD ،DVD ، پردازنده مرکزی، کارت‌های گرافیکی و ابزارهای دیگر که قابل لمس و رویت هستند، سخت‌افزار رایانه می‌باشند. اکنون، برمی‌گردیم به مثال کلاس درس، همان‌طور که بیان گردید، تمام ابزارها و افرادی که در کلاس درس وجود دارند، سخت‌افزار هستند. حال، سوال این است که نرم‌افزار کلاس درس چیست؟ برای پاسخ به این سوال، فرض کنید کلاس درس و تمام امکانات آن آماده باشد، آیا شما می‌توانید آن کلاس را اداره کرده و آموزش دهید؟ آیا استاد دیگری که تخصص آن رایانه نباشد می‌تواند از امکانات کلاس استفاده کند و رایانه آموزش دهد؟ پاسخ به این سوالات منفی است. پس، نرم‌افزار کلاس درس، همان علم استاد رایانه است که ابزارهای فراهم شده در کلاس درس را مورد بهره برداری قرار می‌دهد. بنابراین، **نرم‌افزار رایانه**[6]، علمی است که سخت‌افزار رایانه را راه‌اندازی کرده، مورد بهره‌برداری قرار می‌دهد.

---

[2].Mini Computers [2]. Personal Computers [3] .Laptop Computers [4].Portable Computers
[5] .Hardware [6].Computer Software

۳ – ۱. پردازنده مرکزی

پردازنده مرکزی، قسمتی از رایانه است که عملیات محاسباتی و منطقی را انجام می‌دهد. نام دیگر پردازنده، **CPUواحد پردازشگر مرکزی** (CPU) است. یکی از قطعات بسیار مهم رایانه است که بر روی مادربرد (Mainboard) قرار می‌گیرد. وظیفه و عملکرد آن در رایانه، مشابه با مغز انسان است، همان‌گونه که مغز انسان وظیفه پردازش اطلاعات دریافتی و نظارت بر سایر اعضای بدن را به عهده دارد، یک CPU نیز پردازش داده‌ها را انجام می‌دهد و همچنین نظارت بر کار دیگر اجزای رایانه را به عهده دارد. ارتباط پردازنده با سایر اجزای رایانه از طریق باس‌ها (BUS) انجام می‌شود. هر پردازنده سه نوع باس دارد. این باس‌ها عبارت‌اند از (شکل ۱ – ۱).



شکل ۱ – ۱ ارتباط CPU با اجزای رایانه از طریق باس‌ها.

- **باس داده (Data Bus)**، برای انتقال داده بین اجزای رایانه به کار می‌رود.
- **باس آدرس (Address Bus)**، برای تعین آدرس حافظه مورد نیاز اجزای رایانه به کار می‌رود.
- **باس کنترل (Control Bus)**، برای کنترل و هدایت عملیات خواندن و نوشتن توسط اجزای رایانه به کار می‌رود.

یک پردازنده از سه بخش اصلی تشکیل شده است که عبارت‌اند از (شکل ۲ – ۱).

۱. واحد حافظه (MU)[3]، وظیفه آن نگهداری داده‌ها است. این واحد از ثبات‌ها[2]، حافظه RAM و ROM تشکیل شده است. ثبات‌ها، از چند عنصر الکترونیکی تشکیل می‌شوند. این عناصر به طور منطقی در مجاور هم قرار می‌گیرند. وظیفه ثبات‌ها نگهداری موقت داده و دستورالعمل‌ها است. رایانه‌های امروزی ثبات‌های ۸، ۱۶، ۳۲، ۶۴ بیتی دارند. با حافظه‌های RAM و ROM در ادامه آشنا خواهید شد.

۲. واحد محاسبه و منطق(ALU)[3]، وظیفه انجام عملیات محاسباتی و منطقی را بر عهده دارد. اعمالی از قبیل جمع، ضرب، تفریق و تقسیم، عملیات محاسباتی نام دارند. ولی، اعمالی از قبیل مقایسه دو مقدار، عملیات منطقی نامیده می‌شوند. یعنی، از وظایف این واحد، تصمیم‌گیری در اعمال منطقی (مقایسه‌ای) مثل بزرگ‌تر، مساوی، بزرگ‌تر مساوی و ... است. به عنوان مثال، دستورات زیر را در نظر بگیرید:

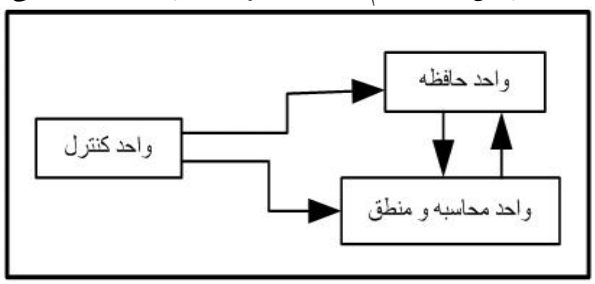
- اگر حقوق کوچک‌تر یا مساوی ۸۳۳۰۰۰۰ ریال است، مالیات برابر با صفر است. وگرنه، مالیات برابر با ۸۳۳۰۰۰۰ – حقوق) ضرب در ۰,۱ است.
- مالیات را چاپ کن.

این تصمیم‌گیری جهت محاسبه مالیات و چاپ آن به کار می‌رود.

---

[3] .Memory Unit [2].Registers [3].Arithmatic and Logical Unit [4].Control Unit

واحد کنترل (CU)[4] ، مسئول دریافت دستورات، داده‌ها، رمزگشایی آن‌ها، سازماندهی و کنترل واحد محاسبه و منطق است. یعنی، بدون اجازه واحد کنترل، هیچ عملی انجام نمی‌شود. واحد کنترل اجازه خواندن دستورات، خواندن داده، نوشتن داده، انجام عملیات (اجرای دستورات) را صادر می‌کند.



شکل ۲ – ۱ ساختار پردازشگر مرکزی (CPU).

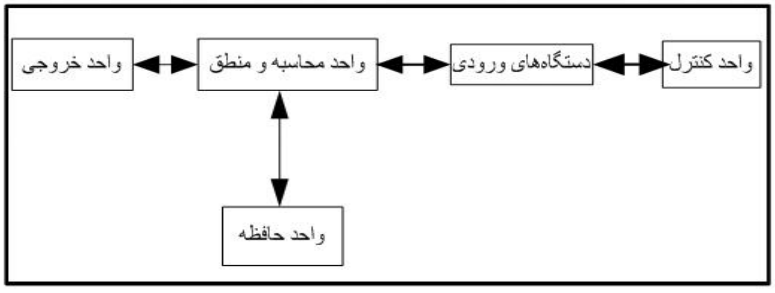
۴ – ۱. اجزای تشکیل‌دهنده رایانه

قبل از این که به اجزای تشکیل‌دهنده رایانه بپردازیم، فرض کنید، رایانه بخواهد حقوق کارمندان را دریافت و محاسبه کند. بنابراین، باید واحدی داشته باشد که بتواند احکام کارکنان را دریافت نماید. این **واحد ورودی**[4] نام دارد. علاوه بر دریافت احکام باید دارای واحدی باشد که بتواند این احکام را ذخیره نماید. این **واحد حافظه**[2] نام دارد. پس از انتقال احکام به واحد حافظه، رایانه باید بتواند قوانین محاسبه حقوق را بر روی احکام اعمال کرده، بیمه، مالیات، اضافه کار و غیره را محاسبه کند. انجام این اعمال وظیفه **واحد پردازش‌گر مرکزی CPU** ([3]) است. بعد از محاسبه حقوق، رایانه باید بتواند نتایج را به کارمندان تحویل دهد و لیست‌های متعددی را تهیه نماید. این کار را **واحد خروجی**[4] انجام می‌دهد. بنابراین، اجزای اصلی رایانه عبارت‌اند از:

۱. واحد ورودی ۲. واحد حافظه ۳. واحد محاسبه و منطق ۴. واحد خروجی (شکل ۳ – ۱).

۱ – ۴– ۱. واحد ورودی

واحد ورودی، وظیفه دریافت داده‌ها از دنیای خارج و انتقال آن به رایانه را دارد. از آنجایی که باید بتوان داده‌های مختلف(از قبیل متن، اعداد، تصاویر، امضاء، بارکد و غیره) را به رایانه انتقال داد، ابزارها و دستگاه‌های متعدد ورودی ساخته شده‌اند. برخی از این دستگاه‌ها را در ادامه می‌بینید.



[4].Input Unit [2]. Memory Unit [3].Central Processing Unit [4].Output Unit
[5] .Keyboard

شکل ۳ – ۱ اجزای تشکیل دهنده رایانه.

صفحه کلید[۵]، رایج‌ترین دستگاه ورودی در رایانه است. صفحه کلید از تعدادی کلید از قبیل کلیدهای عددی، حرفی، کنترلی، تابعی (Fها)، جهت نماها، نشانه‌ها و غیره تشکیل شده است. برخی از این کلیدها در تمام نرم‌افزارها یک کاربرد دارند (نظیر کلیدهای حرفی و عددی). اما، بعضی دیگر از کلیدها مانند کلیدهای تابعی و کنترلی در نرم‌افزارهای مختلف کاربردهای متفاوتی دارند. تعداد کلیدهای صفحه کلید متفاوت است (صفحه کلیدهای قدیمی ۱۰۱ کلید داشتند. اما، صفحه کلیدهای جدید ۱۰۴ یا بیشتر کلید دارند). صفحه کلید از مهم‌ترین دستگاه‌های ورودی یک برنامه نویس است.

ماوس[۵]، پس از صفحه کلید، پرکاربردترین دستگاه ورودی است. ماوس‌ها دو یا سه کلید دارند. ماوس، وظیفه انتقال مکان‌نما و انتخاب گزینه‌ها روی صفحه نمایش یا اجزای برنامه را دارد. یعنی، با حرکت آن روی یک سطح صاف، هم‌زمان مکان‌نما (اشاره‌گر ماوس) روی صفحه نمایش جابه‌جا شده و با فشار دکمه‌ای از ماوس بر روی آیکون یا برنامه کار خاصی انجام می‌شود. به عنوان مثال، اگر بر روی برنامه‌ای، دکمه سمت چپ ماوس را فشار دهید[۲]، آن برنامه اجرا خواهد شد.

گوی‌های کنترل، همانند ماوس است. با این تفاوت که گوی (در گوی کنترل) در قسمت بالای آن قرار گرفته و توسط حرکت دست در جهات مختلف چرخیده و مکان‌نمای آن روی صفحه نمایش جابه‌جا می‌شود.

صفحه نمایش‌های لمسی[۳]، به گونه‌ای طراحی و ساخته شده‌اند که به حرکت و فشار دست بر روی آن حساس هستند. بنابراین، مانند ماوس می‌توان با حرکت دست، برنامه‌ای را انتخاب کرده یا آن را اجرا نمود. برخی از موبایل‌ها نیز صفحه نمایش لمسی دارند.

قلم نوری، یک قلم حساس به نور است که با کشیدن آن بر روی صفحه نمایش (مانند ماوس) می‌توان اعمالی از قبیل انتخاب و رسم اشکال را انجام داد.

جوی استیک(دسته بازی)[۴]، دارای دسته‌ای است که در بازی‌های رایانه‌ای برای انتقال مکان‌نما و اجرای دستورات به کار می‌رود (شکل۴ – ۱).

صفحه ورود اطلاعات، تخته‌ای است که به حرکت و فشار بر روی آن حساس است. به طوری که با حرکت بر روی این تخته و فشردن دست، تصویری (نقاشی) در رایانه رسم خواهد شد. این دستگاه برای ترسیم نقاشی، از ماوس کاراتر و آسان‌تر است.

کارت خوان‌ها[۵]، کارت‌های مغناطیسی یا کارت‌هایی که بارکد یا علامت خاصی بر روی آن‌ها ذخیره شده باشد (مانند کارت‌های ورود و خروج ادارات، بارکدهای چاپ شده بر روی کالاها، کارت‌های اعتباری، کارت‌های هوشمند) را می‌خوانند. کارت‌ها، مغناطیسی یا نوری هستند. برخی از کارت خوان‌های مغناطیسی

---

[5].Mouse [2].Click [3].Touch Screens [4].Joy Stick [5].Card Readers

به تماس مستقیم کارت به کارت خوان نیاز دارند. این کارت خوان‌ها، **تماسی** نام دارند. ولی، بعضی از آن‌ها نیاز به تماس کارت به کارت خوان ندارند (مانند کارت خوان‌های نصب شده در متروها، ایستگاه‌های قطار). این کارت خوان‌ها، **بدون تماس** نامیده می‌شوند. امروزه اکثر فروشگاه‌ها از کارت خوان استفاده می‌کنند.

شکل ۴ – ۱ نمونه‌هایی از جوی استیک.

۸ - ۱. مراحل برنامه‌نویسی

برنامه‌نویسی دارای مراحل زیر است:

۱. **شناخت مسئله**، که شامل مراحل زیر می‌باشد:
- شناخت فرضیات و داده‌های ورودی مسئله (Data)
- تشخیص خواسته‌ها یا مجهولات مسئله
- تعیین ارتباط بین فرضیات و مجهولات

۲. **ارائه طرح یا نقشه حل مسئله**

روش‌های مختلفی برای حل مسئله وجود دارد. برخی از این روش‌ها عبارت‌اند از:
- طراحی الگوریتم
- طراحی فلوچارت

۳. نوشتن برنامه با یکی از زبان‌های برنامه‌نویسی (نظیر C، پاسکال، C++ یا C#).

۹- ۱. الگوریتم چیست؟

وقتی می‌خواهید کاری را انجام دهید، به ویژه اگر آن کار پیچیده باشد، ابتدا، باید روش و مراحل انجام آن کار را مشخص نمایید. به عنوان مثال، فرض کنید مسئول برگزاری جشنی هستید، در این صورت باید خودتان را برای انجام کارهای لازم جهت برگزاری این جشن آماده کنید. در آن صورت چه می‌کنید؟ مطمئناً لیستی از کارهایی که باید انجام دهید، تعیین کرده، ترتیب انجام آن‌ها را مشخص می‌کنید. یعنی، روش و مراحل انجام کار را به دقت تنظیم خواهید کرد. چنانچه این کار را به نحو کامل و بدون **نقص** انجام دهید که تحت هر شرایطی با هر نوع امکانات و خصوصیات بتوان از این رویه استفاده نمود، اکنون توانسته‌اید الگوریتم برگزاری مراسم را طراحی کنید. حتی می‌توانید این الگوریتم را به دیگران بدهید تا آن‌ها نیز بتوانند از این الگوریتم برای برگزاری جشن‌شان استفاده کنند.

الگوریتم، عبارت است از مجموعه‌ای از دستورالعمل‌ها که مراحل انجام کاری را **به زبان دقیق** با **جزئیات کافی** بیان نماید، ترتیب **اجرای دستورات** و **شرط خاتمه** آن مشخص باشد.

برای این که با مفهوم الگوریتم آشنا شویم، دستورالعمل استفاده از تلفن همگانی را در نظر بگیرید. این دستورالعمل در زیر آمده است:

۱. گوشی را بردارید.
۲. یک سکه در داخل تلفن قرار دهید.
۳. منتظر شنیدن بوق آزاد باشید.
٤. شماره را گرفته، صحبت کنید.
٥. در پایان، گوشی را در جایش قرار دهید.

مراحل بیان شده، دستورالعمل استفاده از تلفن همگانی است (الگوریتم نیست). زیرا:

۱. مرحله ۲ به زبان دقیق بیان نشده است. یعنی، استفاده کننده تلفن نمی‌داند چه سکه‌ای را در داخل تلفن قرار دهد. برای این که این مرحله به یک دستورالعمل الگوریتم تبدیل شود، این دستورالعمل باید به صورت زیر تغییر یابد:

- یک سکه ۲۵۰ یا ۱۰۰ ریالی سالم در داخل تلفن قرار دهید.

۲. شرط خاتمه الگوریتم مشخص نشده است. زیرا، اگر تلفن خراب باشد، در مرحله ۳ هیچ‌گاه بوق آزاد شنیده نمی‌شود. بنابراین، استفاده‌کننده تلفن تا کی منتظر شنیدن بوق آزاد باشد. این مرحله را به صورت زیر تغییر دهید تا به یک مرحله الگوریتم تبدیل شود:

- ۳۰ ثانیه منتظر شنیدن بوق آزاد باشید. چنانچه بوق آزاد را نشنیده‌اید، گوشی را در جایش قرار دهید. مراحل ۱ تا ۳ را اجرا کنید. چنانچه ۳ بار این مراحل را انجام داده‌اید و بوق آزاد را نشنیده‌اید، گوشی را در جایش قرار دهید.

بنابراین، الگوریتم استفاده از تلفن همگانی به صورت زیر است:

۱. شروع
۲. گوشی را بردارید.
۳. یک سکه ۱۰۰ یا ۲۵۰ ریالی سالم در داخل تلفن قرار دهید.
٤. ۳۰ ثانیه منتظر شنیدن بوق آزاد باشید، چنانچه بوق آزاد را نشنیده‌اید، گوشی را در جایش قرار دهید. مراحل ۱ تا ۴ را اجرا کنید. اگر ۳ مرتبه این مراحل را انجام داده‌اید و بوق آزاد را نشنیده‌اید، گوشی را جایش قرار دهید و دنبال تلفن دیگر بگردید.
٥. شماره‌گیری نموده و صحبت کنید
٦. در پایان گوشی را جایش قرار دهید.

۷. پایان

مثال ۱ – ۱. الگوریتمی که مراحل آماده‌سازی نیمرو را بیان می‌کند.

۱. شروع
۲. ماهی‌تابه را بردارید.
۳. در ماهی‌تابه روغن بریزید.
٤. اجاق را با حرارت ملایم روشن نمایید.
۵. منتظر داغ شدن روغن بمانید.
٦. پس از داغ شدن روغن، تخم مرغ را در ماهی‌تابه بشکنید.
۷. به تخم مرغ نمک و فلفل اضافه نمایید.
۸. تا زمانی که نیمرو به اندازه کافی بپزید، صبر کنید.
۹. اجاق را خاموش نمایید.
۱۰. نیمرو را در بشقاب قرار دهید.
۱۱. پایان.

همان‌طور که در این الگوریتم مشاهده می‌کنید، هر الگوریتم یک **نقطه شروع** و یک **نقطه پایان** دارد.

۱۰ - ۱. تفکر الگوریتمیک

آیا مفهوم الگوریتم فقط در دنیای برنامه‌نویسی تعریف می‌شود؟ آیا تنها برنامه‌نویس برای طراحی و حل برنامه‌اش باید از الگوریتم استفاده کند؟ یا این که برای انجام تمام کارها، می‌توان از ایده الگوریتمی استفاده نمود؟

در جواب این سوالات باید بگوییم که الگوریتم یک مفهوم عام و فراگیر است. یعنی، ایده الگوریتم، علاوه بر آن که در برنامه‌نویسی به کار می‌رود، می‌تواند در زندگی روزمره نیز استفاده شود.

یک برنامه‌نویس خوب، علاوه بر آن که به الگوریتم برنامه‌هایی که می‌نویسد فکر می‌کند، با الگوریتم زندگی می‌کند. او تمام زندگی خودش را با الگوریتم انجام می‌دهد. با الگوریتم فکر می‌کند. یعنی، تفکر او حتی در امور ساده زندگی نیز الگوریتمیک است. این فرد احتمالاً برنامه‌نویس موفقی خواهد بود. این تنها کافی نیست که بخواهید الگوریتم یک کار را پیدا کنید. بلکه ماهیتاً باید الگوریتمیک فکر کنید. حتی به تمام مسائل زندگی نیز باید به این شیوه نگاه کنید. مثال برنامه نویس که دارای تفکر الگوریتمیک است، نسبت به فردی که این ویژگی را ندارد، مانند نسبت کسی است که ماهیتاً کارش دارای نظم است و کسی که کارش نظم خاصی ندارد. فرد اول، در پایان کار، کارگاه خود را مرتب و منظم می‌کند. اما، کارگاه فرد دوم نه تنها در هنگام کار نامرتب است، بلکه بعد از انجام کار نیز نامنظم خواهد بود.

> وقتی می‌خواهید کاری را انجام دهید، باید به الگوریتم مناسب برای انجام آن کار فکر کنید. پس از یافتن الگوریتم مناسب، انجام آن کار برای‌تان بسیار ساده خواهد شد. چیزی را فراموش نمی‌کنید و در انجام آن کار، همه ابعاد آن را در نظر خواهید گرفت و مهم‌تر از همه این‌ها، ساختار کار برای

شما روشن خواهد بود. برنامه‌نویس باید با الگوریتم زندگی کند و آن را به عنوان روش زندگی خود بشناسد.

ارائه راه حل مناسب، نیاز دقیق به شناخت مسئله دارد.

مثال ۲ – ۱. الگوریتمی که سه عدد را خوانده، میانگین آن‌ها را نمایش می‌دهد.

۱. شروع

۲. a، b و c را بخوان

۳. $ave \leftarrow \frac{a+b+c}{3}$

٤. ave را چاپ کن

۵. پایان

داده : a , b , c (۳ عدد)

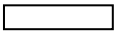
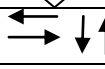
خواسته : ave (میانگین)

رابطه : $ave \leftarrow \frac{a+b+c}{3}$

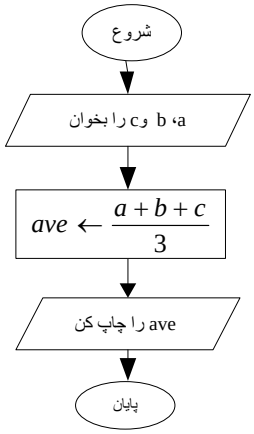
۱۱ – ۱. تعریف فلوچارت

به ارائه‌ی یک طرح منطقی با جزئیات کافی، دقیق و رعایت ترتیب مراحل اجرا، از روش حل یک مسأله فلوچارت (نمودار عملیاتی) آن مسأله می‌گویند. ارائه فلوچارت به کمک علائم قراردادی انجام می‌شود علائم رسم فلوچارت در جدول ۱ – ۱ آمده‌اند.

جدول ۱ – ۱ علائم رسم فلوچارت

شکل	توضیح
	**بیضی:** دستورات شروع و پایان
	**متوازی‌الاضلاع:** دستورات ورودی و خروجی را تعیین می‌کند.
	**مستطیل:** برای دستورات محاسباتی یا جایگزینی استفاده می‌شود.
	**لوزی:** برای دستورات شرطی یا تصمیم‌گیری استفاده می‌شود.
	**فلش‌های جهت‌دار:** جهت تعیین مسیر عملیات استفاده می‌شوند.
	**دایره کوچک:** رابط

به عنوان مثال، فلوچارت میانگین سه عدد در زیر آمده است:



مثال۳ –۱. الگوریتم و فلوچارتی که ابتدا اضلاع مثلثی را از ورودی خوانده، مساحت آن را محاسبه و نمایش می‌دهد.

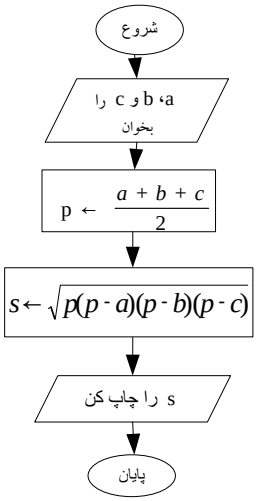
داده : a, b, c (**اضلاع مثلث**)

خواسته : s (**مساحت**)

رابطه : (**قاعده هرون**)

$$s = \sqrt{p(p-a)(p-b)(p-c)} \qquad p = \frac{a+b+c}{2}$$

۱. شروع
۲. اضلاع مثلث را بخوان (b , c و a را بخوان)
۳. $p \leftarrow \frac{a+b+c}{2}$
۴. $p = (x + y) * 2$
۵. $s \leftarrow \sqrt{p(p-a)(p-b)(p-c)}$
۶. s را چاپ کن
۷. پایان



۱۳ - ۱. ساختار حلقه

در برخی از برنامه‌ها نیاز است مجموعه‌ای از دستورات چند بار تکرار شوند. برای تکرار این دستورات از حلقه‌های تکرار استفاده می‌شود. زیرا:

۱. از نوشتن دستورات تکراری جلوگیری می‌کند.
۲. از تعریف متغیرهای اضافی جلوگیری خواهد کرد.

مثال ۸ – ۱. الگوریتمی را در نظر بگیرید که بخواهد اعداد ۱ تا ۵ را نمایش دهد. برای این منظور، سه روش وجود دارد:

۱. **بدون استفاده از حلقه تکرار**، این الگوریتم به صورت زیر است:

۱. شروع
۲. یک را در i قرار بده
۳. i را چاپ کن.
۴. i + 1 را در i قرار بده

۵. i را چاپ کن.
٦. 1 + i را در i قرار بده
۷. i را چاپ کن.
۸. 1 + i را در i قرار بده
۹. i را چاپ کن.
۱۰. 1 + i را در i قرار بده
۱۱. i را چاپ کن.
۱۲. پایان

همان‌طور که می‌بینید، اگر تعداد تکرار زیاد باشد (به عنوان مثال، اگر الگوریتم بخواهد اعداد ۱ تا ۱۰۰ را نمایش دهد)، این فرآیند طولانی و غیر قابل انجام خواهد شد (یعنی، کدهای برنامه زیاد می‌شود).

۲. **استفاده از متغیرهای جداگانه**، در این روش برای هر یک از مقادیر ۱ تا ۵ متغیر مجزایی در نظر گرفته می‌شود. این الگوریتم در زیر آمده است.

۱. شروع
۲. ۱ را در a، ۲ را در b، ۳ را در c، ۴ را در d و ۵ را در e قرار بده.
۳. a، b، c، d و e را چاپ کن.
٤. پایان.

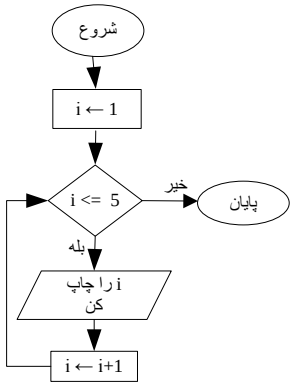
در این روش اگر تعداد تکرار زیاد باشد، نیاز به متغیرهای زیادی است. بنابراین، حافظه زیادی مصرف خواهد شد.

۳. **استفاده از حلقه تکرار**، برای ایجاد حلقه تکرار اعمال زیر باید انجام شود:

۱. یک شمارنده در نظر گرفته شود.
۲. مقدار اولیه حلقه در شمارنده قرار گیرد.
۳. شرط ادامه حلقه تست گردد (ابتدای حلقه). اگر شرط ادامه حلقه درست باشد، گام‌های ۴ تا ۶ اجرا گردد، وگرنه حلقه خاتمه یابد.
٤. دستورات بدنه حلقه اجرا شوند.
۵. شمارنده به اندازه گام افزایش، اضافه شود.
٦. به ابتدای حلقه برود (به مرحله ۳ برود).

بنابراین، الگوریتم مثال ۸ – ۱ با استفاده از حلقه تکرار به صورت زیر است:

۱. شروع
۲. یک را در i قرار بده (i شمارنده حلقه است).
۳. تا وقتی که 5 => i تکرار کن
الف: i را چاپ کن
ب: 1 + i را در i قرار بده.



۴. پایان

این روش، نه تنها از یک متغیر برای نوشتن برنامه استفاده کرده است (شمارنده i)، بلکه از تکرار کدهای اضافی نیز جلوگیری نموده است.

همان‌طور که مشاهده کردید، در الگوریتم مثال ۸ – ۱ تعداد تکرار از قبل مشخص بود.

گاهی اوقات اتفاق می‌افتد که تعداد تکرار متغیر است (مشخص نیست) و بستگی به عددی دارد که کاربر وارد کرده است. در این صورت حلقه، همان حلقه قبلی می‌باشد، با این تفاوت که در شرط حلقه عدد وارد شده قرار می‌گیرد (مثال زیر را ببینید).

مثال ۹ – ۱. الگوریتم و فلوچارتی که عدد مثبتی را خوانده، اعداد ۱ تا عدد خوانده شده را نمایش می‌دهد.

داده: n (عدد خوانده شده) **خواسته:** نمایش ۱ تا n

رابطه: ایجاد حلقه‌ایی با شمارنده i که مقادیر ۱ تا n را می‌پذیرد.

۱. شروع

۲. n را بخوان

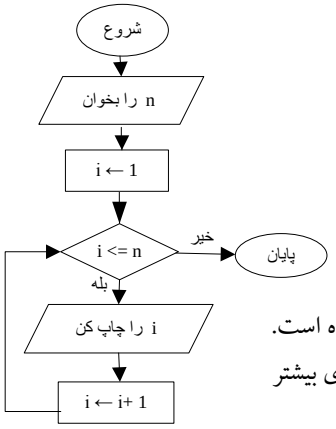
۳. ۱ را در i قرار بده

۴. تا وقتی که n => i تکرار کن

الف: i را چاپ کن

ب: i + 1 را در i قرار بده

۵. پایان



این حلقه‌هایی که مشاهده کردید، گام افزایش آن‌ها یک بوده است. گاهی ممکن است گام افزایش حلقه عدد یک نباشد. یعنی، عددی بیشتر یا کمتر از یک باشد (مثال ۱۰– ۱ را ببینید).

مثال ۱۰ – ۱. الگوریتمی که عددی مثبت را خوانده، اعداد فرد ۱ تا عدد خوانده شده (۱، ۳، ...، n) را نمایش می‌دهد.

داده: n (عدد خوانده شده)

خواسته: نمایش اعداد فرد ۱ تا n

رابطه: تشکیل حلقه تکرار از ۱ تا n به طوری که در هر مرحله ۲ واحد به شمارنده آن اضافه می‌شود.

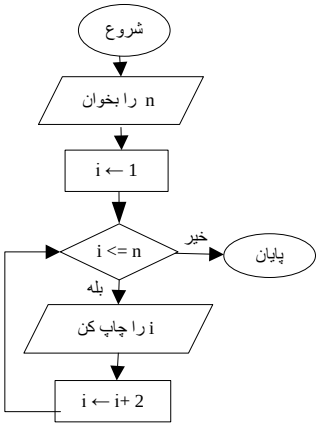
۱. شروع

۲. n را بخوان

۳. ۱ را در i قرار بده

٤. تا وقتی که n => i تکرار کن

الف: i را چاپ کن



ب: $i + 2$ را در i قرار بده

۵. پایان

این الگوریتم مانند الگوریتم ۹ – ۱ است. با این تفاوت که در گام پنجم به جای این که $i + 1$ در i قرار گیرد، $i + 2$ را در i قرار می‌دهد.

حلقه‌هایی که تاکنون دیدید، بدنه حلقه آن‌ها از یک دستور تشکیل می‌گردیدند.

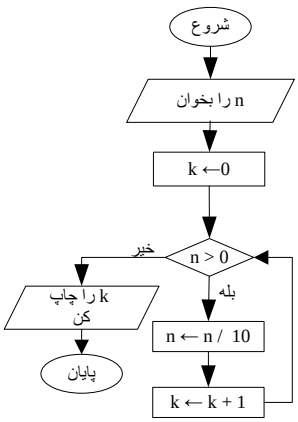
بدنه حلقه می‌تواند دارای بیش از یک دستور باشد (الگوریتم‌های زیر را ببینید):

مثال ۲۵ – ۱. الگوریتم و فلوچارتی که عدد صحیح و مثبت n را از ورودی می‌خواند، سپس:

الف: تعیین کند n چند رقمی است. ب: مجموع ارقام n را به دست آورد.

ج: وارون n را از لحاظ ارزش مکانی ارقام بدست آورد.

نکته: اگر دو عدد صحیح در C++ با / برهم تقسیم شوند نتیجه خارج تقسیم صحیح می‌باشد.



۱. شروع

۲. n را بخوان

۳. $k \leftarrow 0$

٤. تا هنگامی که $n > 0$ تکرار کن

الف: $n \leftarrow n / 10$

ب: $k \leftarrow k+1$

۵. k را چاپ کن

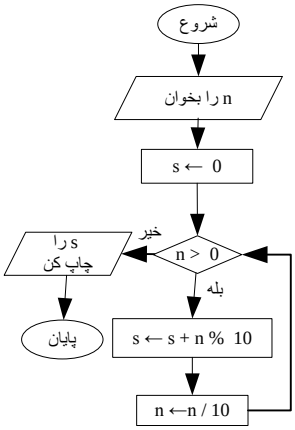
٦. پایان

این الگوریتم ابتدا n (عددی که باید تعداد ارقام آن شمارش شود) را خوانده، تعداد ارقام (k) را برابر صفر قرار می‌دهد. سپس تا زمانی که n بزرگ‌تر از صفر باشد، اعمال زیر را انجام می‌دهد.

- n تقسیم بر ۱۰ را در n قرار می‌دهد تا رقم یکان n حذف شود.
- به k یک واحد اضافه می‌کند (چون یک رقم حذف گردید).

به محض این که n صفر شود، k را چاپ می‌کند.

پاسخ ب:



۱. شروع

۲. n را بخوان

۳. $s \leftarrow 0$

٤. تا هنگامی که $n > 0$ تکرار کن.

الف: $s \leftarrow s + (n \ \% \ 10)$

ب: $n \leftarrow n / 10$

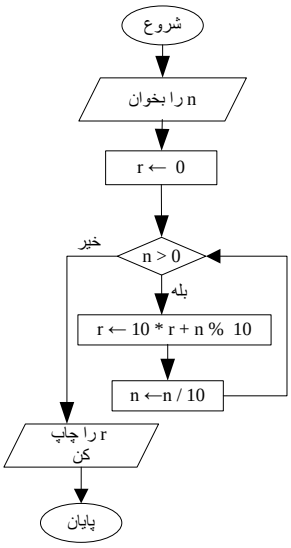
۵. s را چاپ کن

۶. پایان

این الگوریتم ابتدا n را خوانده، سپس مجموع ارقام را برابر صفر قرار می‌دهد و تا زمانی که n بزرگ‌تر از صفر باشد، اعمال زیر را انجام می‌دهد:

- ابتدا باقی مانده تقسیم صحیح n بر ۱۰ (ارقام یکان) را با مجموع ارقام (s) جمع کرده، در s قرار می‌دهد.
- n تقسیم بر ۱۰ را در n قرار می‌دهد تا رقم یکان حذف شود.

به محض این که n صفر شود، s را چاپ می‌کند.



پاسخ ج:

۱. شروع

۲. n را بخوان

۳. $r \leftarrow 0$

۴. تا هنگامی که $n > 0$ تکرار کن

الف: $r \leftarrow 10 * r + n \ \% \ 10$

ب: $n \leftarrow n \ / \ 10$

۵. r را چاپ کن

۶. پایان

این الگوریتم ابتدا n را خوانده، صفر را در وارون (n) یعنی r قرار می‌دهد و تا زمانی که n بزرگ‌تر از صفر باشد، اعمال زیر را انجام می‌دهد:

- وارون عدد (r) را ضرب در ۱۰ می‌کند و با رقم یکان n (یعنی، باقی مانده صحیح تقسیم n بر ۱۰) جمع می‌کند و در r قرار می‌دهد (این عبارت موجب می‌شود تا n وارون گردد).
- n تقسیم بر ۱۰ را در n قرار می‌دهد تا رقم یکان حذف شود.

به محض این که n صفر گردید، r را چاپ می‌کند.

مثال ۲۶ – ۱. الگوریتم و فلوچارتی که n را خوانده خروجی زیر را نمایش می‌دهد:

توضیح: برای حل این مسئله از حلقه‌های تودرتو استفاده می‌شود.

۱

شمارنده حلقه خارجی (شمارنده سطرها)از ۱ تا n تغییر می‌کند. اما شمارنده حلقه

۱ ۲

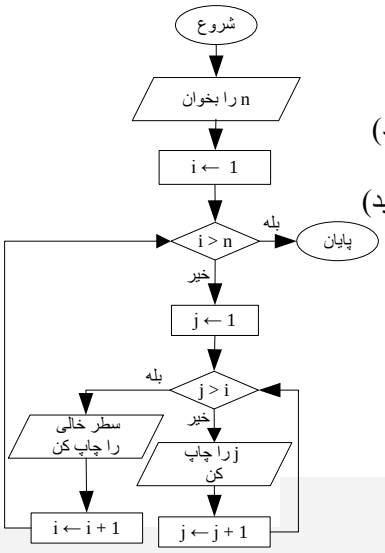
داخلی (شمارنده ستون‌های هر سطر) از ۱ تا شمارنده حلقه خارجی تغییر خواهد

۱ ۲ ۳

کرد (به اندازه شماره سطر تکرار می‌شود و در داخل حلقه داخلی شمارنده
۱ ۲ ۳ ۴

۴ ... n
۱ ۲ ۳

حلقه داخلی نمایش داده می‌شود.

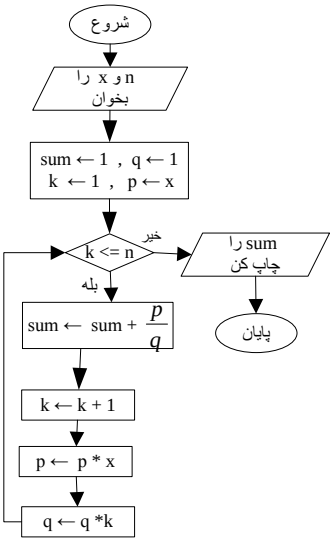


۱. شروع
۲. n را بخوان
۳. i ← 1 (شمارنده حلقه خارجی) (شمارنده سطرها)
۴. اگر i > n است برو به ۱۳(تعداد سطرها به پایان رسید)
۵. j←1
۶. اگر j > i بود برو به ۱۰(چاپ یک سطر به پایان رسید)
۷. در همان سطر j را چاپ کن
۸. j ← j+ 1
۹. برو به ۶
۱۰. در چاپ برود به سطر بعد
۱۱. i ← i +1
۱۲. برو به ۴
۱۳. پایان

مثال ۶ – ۱. الگوریتم و فلوچارتی که ابتدا عدد صحیح و مثبت n و سپس عدد دلخـواه x را از ورودی می‌خواند و حاصل هر یک از عبارت‌های زیر را محاسبه و در خروجی نمایش می‌دهد.

الف: $e^x \approx 1+\frac{x^1}{1!}+\frac{x^2}{2!}+\ldots+\frac{x^n}{n!}$

ب: $\cos x \approx 1-\frac{x^2}{2!}+\frac{x^4}{4!}-\ldots\pm\frac{x^{2n}}{(2n)!}$



پاسخ (الف):

۱. شروع
۲. n و x را بخوان
۳. k ← 1 ،q ← 1 , p ← x , sum ← 1
۴. تا هنگامی که k <= n تکرار کن

الف: sum ← sum + $\frac{p}{q}$

ب: k ← k+1

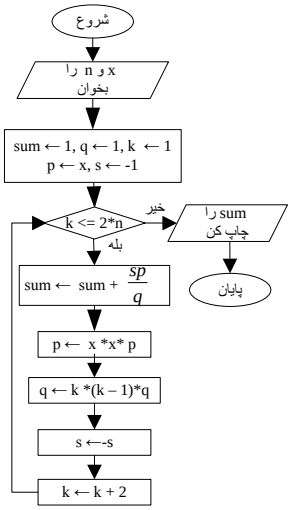
ج: p ← p * x

د: $q \leftarrow q * k$

۵. sum را چاپ کن

۶. پایان

پاسخ (ب):



۱. شروع

۲. x و n را بخوان

۳. $p \leftarrow x * x$, $sum \leftarrow 1$, $s \leftarrow -1$
$k \leftarrow 2$, $q \leftarrow 1$

۴. تا هنگامی که $k <= 2 * n$ تکرار کن

الف: $sum \leftarrow sum + \frac{s * p}{q}$

ب: $p \leftarrow p * x * x$

ج: $q \leftarrow q * k * (k - 1)$

د: $s \leftarrow -s$

هـ: $k \leftarrow k + 2$

۵. sum را چاپ کن

۶. پایان

مثال ۷ –۱. الگوریتم و فلوچارتی که دو عدد طبیعی (بزرگ‌تر از 0) را خوانده، عدد اول را به توان عدد می‌رساند (فقط با عملگر جمع).

توضیح: برای انجام این الگوریتم از حلقه تودرتو استفاده می‌گردد. حلقه خارجی، به تعداد عدد دوم منهای یک مرتبه (توان منهای یک) اجرا می‌شود و حلقه داخلی به تعداد عدد اول (پایه) اجرا می‌گردد. در داخل حلقه دوم، ابتدا پایه با خودش جمع می‌شود و در تکرارهای بعدی، مجموع محاسبه شده مرحله قبل پایه مرتبه با خودش جمع خواهد شد.

$۳ \times ۳ \times ۳ \times ۳ = ۳^۵$ برای انجام این کار، گام‌های زیر انجام می‌شود:

- $۳ \times ۳ = ۳ + ۳ + ۳ = ۹$
- $(۳ \times ۳) \times ۳ = ۹ + ۹ + ۹ = ۲۷$
- $((۳ \times ۳) \times ۳) \times ۳ = ۲۷ + ۲۷ + ۲۷ = ۸۱$
- $(((۳ \times ۳) \times ۳) \times ۳) = ۸۱ + ۸۱ + ۸۱ = ۲۴۳$

۱. شروع

۲. x و y را بخوان

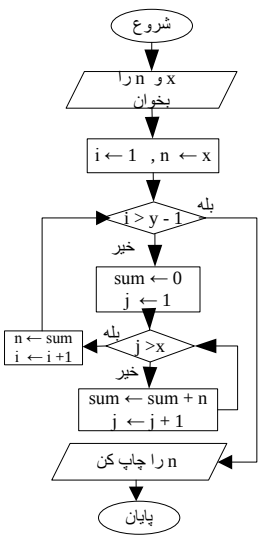
۳. $i \leftarrow 1$ و $n \leftarrow x$

۴. اگر $i > y - 1$ آنگاه بروبه ۱۴

۵. $sum \leftarrow 0$

۶. $j \leftarrow 1$

۷. اگر $j > x$ آنگاه بروبه ۱۱



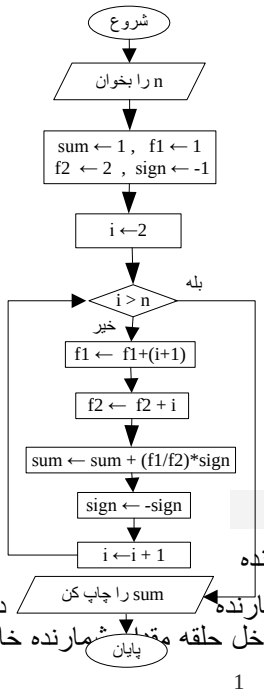
۸. sum ← sum + n
۹. j ← j + 1
۱۰. بروبه ۷
۱۱. n ← sum
۱۲. i ← i + 1
۱۳. بروبه ۴
۱۴. n را چاپ کن
۱۵. پایان

مثال ۸ – ۱. الگوریتمی و فلوچارتی که n را خوانده (n > = 1) و مجموع n جمله اول سری زیر را محاسبه کرده، نمایش می‌دهد:

$$s = 1 - \frac{1+3}{2+4} + \frac{1+3+5}{2+4+6} - \frac{1+3+5+7}{2+4+6+8} + \ldots$$

توضیح: این الگوریتم نیز مانند سری‌های حل شده قبلی است. با این تفاوت که صورت و مخرج باید جداگانه حساب شده در متغیرهای مستقل قرار گیرند.

۱. شروع
۲. n را بخوان
۳. sum ←1 (اولین جمله)
۴. f1 ←1 (صورت)
۵. f1 ←2 (مخرج)
۶. sign ← -1
۷. i ←2
۸. اگر i > n بود بروبه ۱۵
۹. f1 ← f1 + (i + 1)
۱۰. f2 ← f2 +(i+2)
۱۱. sum ← sum + (f1/f2) × sign
۱۲. sign ← -sign
۱۳. i ← i + 2
۱۴. بروبه ۸
۱۵. sum را چاپ کن
۱۶. پایان

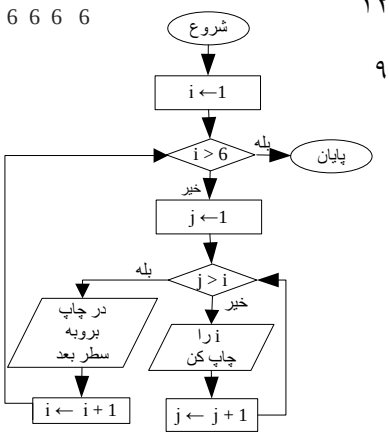


مثال ۹ – ۱. الگوریتم و فلوچارتی که خروجی زیر را نمایش می‌دهد:

```
1
2 2
3 3 3
4 4 4 4
5 5 5 5 5
6 6 6 6 6 6
```

توضیح: این الگوریتم از دو حلقه تودرتو تشکیل می‌شود شمارنده حلقه خارجی (شماره سطر) (i) از ۱ تا ۶ تغییر خواهد کرد و شمارنده داخلی (شمارنده ستون در هر سطر) (j) از ۱ تا i تغییر می‌کند. در داخل حلقه متناظر شمارنده خارجی نمایش داده می‌شود.

۱. شروع
۲. i ← 1
۳. اگر i > 6 آنگاه بروبه ۱۲
۴. j ← 1
۵. اگر j > i آنگاه بروبه ۹



۶. i را در همان سطر چاپ کن
۷. $j \leftarrow j + 1$
۸. بروبه ۵
۹. در چاپ بروبه سطر بعد
۱۰. $i \leftarrow i + 1$
۱۱. بروبه ۳
۱۲. پایان

۱۷–۱. سیستم‌های عدد نویسی

هر مبنایی به اندازه شماره مبنا، نماد دارد. این نمادها از صفر شروع شده و تا عدد مبنا منهای یک ادامه دارد. مثلاً مبنای ۸ دارای ۸ نماد است که از صفر شروع شده تا ۷ ادامه می‌یابد. مبنای ۱۰ دارای ۱۰ نماد(۰ تا ۹) است. مبنای ۱۶ دارای ۱۶ علامت می‌باشد که از ۰ تا ۱۵ ادامه دارد. چون هر نماد باید با یک علامت نشان داده شود، اعداد دو رقمی در این مبنا با حروف الفبا نشان داده می‌شوند:

مبنای ۱۰	۰	۱	۲	۳	٤	٥	٦	۷	۸	۹	۱۰	۱۱	۱۲	۱۳	۱٤	۱٥
مبنای ۱٦	۰	۱	۲	۳	٤	٥	٦	۷	۸	۹	A	B	C	D	E	F

سیستم ده دهی : روش عدد نویسی که در محاسبات روزمره استفاده می‌کنیم، با ده علامت ۰، ۱، ۲، ...، ۹ و ارزش گذاری متفاوت ارقام در محل‌های مختلف(یکان، دهگان، صدگان و ...) می‌توانیم همه‌ی اعداد را بخوانیم و بنویسیم. در این سیستم، ارزش هر رقم بستگی به محلی دارد که رقم در آن قرار گرفته است و هر مکان در بخش صحیح عدد، ارزشی معادل ده برابر ارزش مکانی رقم سمت راست دارد. اما، هر مکان در بخش اعشاری عدد، ارزشی معادل یک دهم برابر ارزش مکانی رقم سمت راست دارد؛ مثلاً در عدد ۴۹۳۷۶٫۸۲۳، ارزش‌های ارقام به صورت زیر است:

$$49376.823 = 4 \times 10^4 + 9 \times 10^3 + 3 \times 10^{2\,+} 7 \times 10^1 + 6 \times 10^0 + 8\times 10^{-1} + 2 \times 10^{-2} + 3 \times 10^{-3}$$

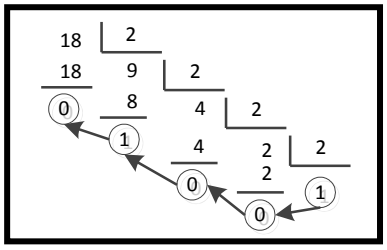
$$= 40000 + 9000 + 300 + 70 + 6 + 0.8 + 0.02 + 0.003 = 49376.823$$

سیستم دودویی: اگر بخواهیم برای نمایش داده‌ها و اطلاعات از سیستم ده دهی استفاده کنیم، در پیاده-سازی سخت افزار رایانه‌ها حداقل به ده سیگنال الکتریکی با سطوح متفاوت جهت تشخیص ده رقم مختلف سیستم ده دهی از یکدیگر نیاز است. این وضعیت، پیاده‌سازی سخت افزار را خیلی پیچیده می‌کند که هم هزینه‌ی طراحی سخت افزار را افزایش می‌دهد و هم احتمال خطا را زیاد می‌کند. به همین دلیل، باید دنبال روشی برای عدد نویسی بگردیم که پیاده‌سازی آن، به کم‌ترین تعداد سیگنال‌ها نیاز داشته باشد و چون ساده-ترین وضع سیگنال‌ها، وجود و یا وجود نداشتن آن‌ها است، باید روشی را پیدا کنیم که فقط دو نماد برای نوشتن اعداد در آن به کار بروند. اگر بخواهیم مفاهیم مربوط به روش معمولی عدد نویسی را برای این دستگاه بازسازی کنیم، باید از دو علامت برای نوشتن رقم‌ها استفاده کنیم و ارزش مکانی هر رقم را دو برابر ارزش

مکانی رقم سمت راستش در نظر بگیریم. با این حساب، مثلاً عددی که در این روش به صورت 11001.01 نوشته می‌شود، در روش معمولی عدد نویسی به صورت زیر محاسبه می‌شود:

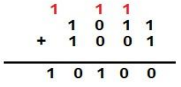
$$1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 0 \times 2^{-1} + 1 \times 2^{-2} = 16 + 8 + 0 + 0 + 1 + 0 + 0.25 = 25.25$$

در حالت کلی، داده‌هایی که وارد رایانه می‌شوند، ابتدا به کد دودویی تبدیل می‌گردند و عملیات محاسبه و پردازش در مبنای دو صورت می‌گیرد و در نهایت، در هنگام نمایش در خروجی، نتیجه‌ها به کد ده دهی تبدیل می‌شوند. برای این که عدد مبنای ۱۰ را به مبنای ۲ ببریم، قسمت صحیح و اعشاری عدد را جدا کرده، قسمت صحیح را با تقسیم‌های متوالی بر عدد ۲ به مبنای دو تبدیل می‌کنیم و عمل تقسیم را تا زمانی ادامه می‌دهیم که خارج قسمت از مبنا بزرگ‌تر باشد و بعد، آخرین خارج قسمت را می‌نویسیم و باقی مانده‌ها را از انتها به ابتدا می‌نویسیم. اما برای قسمت اعشاری از ضرب متوالی در ۲ استفاده می‌کنیم و عمل ضرب را تا زمانی ادامه می‌دهیم تا بخش اعشار پر شود یا عدد صفر شود :



روشی که برای نوشتن اعداد در مبنای ۲ به کار بردیم، برای هر عدد دیگری (غیر از دو) هم قابل استفاده است؛ مثلاً $(18)_{10} = (22)_8$.

عملیات پردازش در سیستم دودویی، مثل قوانین کلی محاسبه‌ی معمولی است؛ با این تفاوت که رقم نقلی و قرضی در محاسبات، به جای عدد ۱۰، عدد ۲ است. بنابراین، جمع دو عدد $(1011)_2$ و $(1001)_2$ در مبنای ۲ به صورت مقابل انجام می‌شود:



توجه کنید که مجموع دو رقم ۱ و ۱ در مبنای دو به صورت ۱۰ نوشته می‌شود که ۰ را به عنوان حاصل جمع می‌نویسیم و ۱ را به عنوان رقم نقلی به واحد بعد منتقل می‌کنیم.

جمع عددهای یک رقمی در مبنای دو			
$\begin{array}{r} 0 \\ +\ 0 \\ \hline 0 \end{array}$	$\begin{array}{r} 0 \\ +\ 1 \\ \hline 1 \end{array}$	$\begin{array}{r} 1 \\ +\ 0 \\ \hline 1 \end{array}$	$\begin{array}{r} 1 \\ +\ 1 \\ \hline 10 \end{array}$

تفریق عدد $(101)_2$ از عدد $(10011)_2$ هم به صورت زیر انجام می‌شود:

همان طور که در این مثال مشاهده می‌کنید، اگر در طبقه‌ای، رقم بالایی از رقم پایینی کم‌تر باشد، یک واحد از طبقه‌ی سمت چپ (که معادل دو واحد در طبقه‌ی فعلی است) به این طبقه منتقل می‌شود.

جمع عددهای یک رقمی در مبنای دو

1 - 1 1 10	1 - 0 1	با یک رقم فرضی 0 - 1 1	0 - 0 0

همان طور که مشاهده می‌کنید در حالت ۱ – ۰ به یک رقم قرضی نیاز است، بنابراین حاصل تفریق ۱ شده اما یک رقم قرضی نیز وجود دارد.

```
  1
  022
  10011
- 00101
-------
  01110
```

سیستم هشت تایی: در سیستم هشت تایی، برای نمایش اعداد از ارقام ۰ تا ۷ استفاده می‌شود و مثل دستگاه دودویی، برای تبدیل مبنا از ۸ به ۱۰ از عمل ضرب و برای تبدیل از مبنای ۱۰ به ۸ از عمل تقسیم استفاده می‌کنیم.

```
  1 1
  605
+ 376
-----
1 203
```

```
    7
  3 8 14
  4 0 6
- 1 5 7
-------
  2 2 7
```

در این مثال مجموع دو عدد ۵ و ۶ برابر ۱۱ است که از بزرگ‌ترین نماد در مبنای ۸ (یعنی عدد ۷) بزرگ‌تر شده است؛ پس ۸ واحد از آن کم می‌کنیم و یک واحد به طبقه‌ی بالاتر اضافه می‌کنیم و عدد ۳ یعنی حاصل تفریق را در پایین می‌نویسیم و این کار را برای سایر سطوح نیز انجام می‌دهیم. همچنین در تفریق عدد ۷ از عدد ۶ ابتدا ۸ واحد به صورت قرضی به عدد ۶ داده می‌شود تا عدد ۱۴ حاصل شود(بنابراین باید یک واحد از طبقه‌ی سمت چپ کم شود اما چون رقم سمت چپ، صفر است و کم کردن از آن امکان پذیر نیست ابتدا یک واحد از عدد ۴ کم می‌کنیم، سپس ۸ واحد به عدد صفر اضافه کرده و در نهایت یک واحد از آن کم می‌کنیم)

سیستم شانزده تایی: سیستم شانزده تایی، کمی با سیستم‌های قبلی فرق دارد. چون مبنا از ده بزرگ‌تر است، ارقام معمولی برای نمایش اعداد در این پایه کافی نیستند. بنابراین، برای نمایش اعداد در این پایه کافی نیستند. بنابراین، برای نمایش ارقام این سیستم از ارقام ۰ تا ۹ و شش نماد (که به ترتیب E (14) , F (14) , A (10) , B (11) , C (12) , D (13)) استفاده می‌شود. در این سیستم ارزش هر طبقه، ۱۶ برابر ارزش طبقه‌ی سمت راست آن است. پس، عدد ۱۲۵۹ مبنای ۱۰ برابر 4EB مبنای ۱۶ می‌باشد.

```
1259 | 16
1248   78 | 16
       64
 11    14    4
```

جدول زیر اعداد ۰ تا ۱۵ را در مبناهای ۲، ۸ و ۱۶ نشان می‌دهد.

ده دهی	باینری	اکتال	هگزا دسیمال	ده دهی	باینری	اکتال	هگزا دسیمال
0	0000	0	0	8	1000	10	8
1	0001	1	1	9	1001	11	9
2	0010	2	2	10	1010	12	A

B	13	1011	11	3	3	0011	3
C	14	1100	12	4	4	0100	4
D	15	1101	13	5	5	0101	5
E	16	1110	14	6	6	0110	6
F	17	1111	15	7	7	0111	7

تبدیل مستقیم توان‌های دو: با توجه به آن که اعداد ۸ و ۱۶ توان‌هایی از ۲ هستند، می‌توان آن‌ها را به روش ساده‌ای به هم تبدیل کرد. در این روش، ابتدا عدد را در مبنای ۲ و سپس به مبنای مورد نظر تبدیل می‌کنیم. مثلاً در تبدیل از مبنای ۲ به مبنای ۱۶، هر چهار رقم در مبنای ۲ معادل یک رقم در مبنای ۱۶ است. به همین ترتیب هر سه رقم در مبنای ۲ معادل یک رقم در مبنای ۸ است.

تبدیل مبنای ۲ به ۱۶ : هر چهار رقم مبنای ۲ را با هم یک دسته در نظر گرفته (بخش صحیح را از سمت راست چهار رقم چهار رقم جدا کرده، اما، بخش اعشاری را سمت چپ چهار رقم چهار رقم می‌کنیم و دسته‌ای که از چهار رقم کم‌تر باشد، آنقدر صفر اضافه می‌کنیم تا چهار رقمی گردد) و معادل هر دسته را طبق جدول به دست آورده و جایگزین می‌نماییم. به عنوان مثال، مقدار ۱۰۱۱۱۱۱۰۱۰۱,۰۰۰۱۱۰۰۰۱۱ به زیر دسته‌بندی می‌کنیم: ۰۱۰۱ ۱۱۱۱ ۰۱۰۱ . ۰۰۰۱ ۱۰۰۰ ۱۱۰۰

سپس هر دسته را جداگانه از مبنای ۲ به ۱۰ تبدیل می‌کنیم و به ازی اعداد از ۱۰ بالاتر معادل مبنای ۱۶ آن را قرار می‌دهیم: 010111110101.0001100011➔5 15 5.1 8 12➔5F5.18C

تبدیل مبنای ۱۶ به ۲ : هر رقم مبنای ۱۶ را جداگانه به مبنای ۲ تبدیل می‌کنیم و به ازای هر رقم مبنای ۱۶باید ۴ رقم در مبنای ۲ قرار دهیم:

$(7DE.4B)_2=(0111,1101,1110.0100,1011)_2$

مکمل یک: برای به دست آوردن مکمل یک عدد باینری کافی است تمام ۱ها به ۰ و همه ۰ها به ۱ تبدیل شوند. به عنوان مثال، مکمل یک عدد ۱۰۰۱۰۰۱۱ برابر با ۰۱۱۰۱۱۰۰ است.

مکمل دو: برای به دست آوردن مکمل دو عدد باینری باید مکمل یک را حساب کرده، یک واحد به مکمل یک اضافه نمود. به عنوان مثال، برای محاسبه مکمل دو ۱۰۰۱۰۰۱۱ مکمل یک را به دست آورده، که برابر با ۰۱۱۰۱۱۰۰ است. اکنون به مقدار ۰۱۱۰۰۱۱۰۰ یک واحد اضافه می‌کنیم تا ۰۱۱۰۰۱۱۰۰ به دست آید. روش دیگر به دست آوردن متمم دو این است که تا اولین ۱ را از سمت راست به همان صورت می‌نویسیم. بقیه ارقام باقی مانده ۰ها به ۱ و ۱ها به ۰ تبدیل می‌گردند. مثلاً، متمم دو مقدار ۱۰۰۰۱۰۱۰ برابر با ۰۱۱۱۰۱۱۰ است.

نمایش اعداد منفی: معمولاً آخرین بیت(سمت چپ‌ترین بیت) به عنوان بیت علامت در نظر گرفته می‌شود که ۱ باشد عدد منفی است. برای به دست آوردن مقدار عدد کافی است عدد را متمم دو کرده و آن را به مبنای

۱۰ تبدیل نماییم. اکنون این عدد را در یک منفی ضرب می‌کنیم. به عنوان مثال، مقدار ۱۱۱۱۱۰۱۰ معادل منفی ۰۰۰۰۰۱۱۰(متمم دو ۱۱۱۱۱۰۱۰) است (یعنی۶-).

۱۸ - ۱. تمرین‌ها

۱. الگوریتم و فلوچارتي بنویسید که از بین ۵۰ عدد ورودي، اعداد زوج دو رقمي را در خروجي چاپ کند.

۲. الگوریتم و فلوچارتي بنویسید که از بین n عدد ورودي، تعداد اعدادي که مضرب ۲ و۳ باشند را پیدا کند و در خروجي بنویسد.

۳. الگوریتم و فلوچارتي بنویسید که عددي صحیح و مثبت را از ورودي بخواند و:

الف: تشخیص دهد که این عدد چند رقم زوج دارد.

ب: آیا ارقام یکان و صدگان آن با هم برابرند یا خیر؟ (عدد حداقل سه رقمي است)

۴. الگوریتم و فلوچارتي بنویسید که ۱۰۰ عدد صحیح را از ورودي بخواند و:

الف: تعداد اعدادي را چاپ کند که حداقل دو رقم زوج داشته باشند.

ب: اعدادي را چاپ کند که تمام ارقامشان فرد باشند.

۵. الگوریتم و فلوچارتي بنویسید که از بین ۲۰ عدد ورودي تنها اعداد اول را در خروجي بنویسد.

٦. الگوریتم و فلوچارتي بنویسید که n عدد صحیح را بخواند و:

الف: اعدادي را چاپ کند که حداقل دو رقم یکسان داشته باشند.

ب: اعدادي را چاپ کند که تعداد ارقام زوج و فرد یکساني داشته باشند.

۷. الگوریتم و فلوچارتي بنویسید که مقدار عبارات زیر را محاسبه کند:

الف: $\frac{3}{2+7} - \frac{4}{3+5} + \frac{5}{4+3} - \frac{6}{5+1} + \frac{7}{6-1} - \ \ldots$ (۱۰۰ جمله)

ب: $\frac{3!}{7} + \frac{5!}{8} + \frac{7!}{9} + \ \ldots$ (۱۰ جمله)

۸. الگوریتم و فلوچارتی بنویسید که تعدادی عدد صحیح و مثبت را از ورودی بخواند که آخرین عدد صفراست، سپس:

الف: اعدادی که رقم سمت راست آن‌ها صفراست را در خروجی بنویسد.

ب: اعدادی که رقم یکان آن‌ها ۵ است را در خروجی بنویسد.

ج: تعداد اعداد فرد را تعیین کند.

۹. الگوریتمی و فلوچارتی بنویسید که عدد صحیح و مثبت n را از ورودی خوانده، آن را به مبنای دو تبدیل کند و در خروجی بنویسد.

۱۰. الگوریتم و فلوچارتی بنویسید که یک عدد را به مبنای دو از ورودی خوانده، آن را به مبنای ده تبدیل کرده و در خروجی بنویسد.

۱۱. الگوریتمی و فلوچارتی بنویسید که ۲۰ عدد صحیح و مثبت را از ورودی بخواند سپس:

الف: اعدادی که مجموع ارقام آن‌ها کم‌تر از ۵ است را در خروجی بنویسد.

ب: اعدادی که رقم دهگان آن‌ها زوج است را در خروجی بنویسد.

۱۲. الگوریتم و فلوچارتی بنویسید که دو عدد را از ورودی بخواند و عدد بزرگ‌تر را در خروجی بنویسد.

۱۳. الگوریتم و فلوچارتی بنویسید که سه عدد را از ورودی بخواند و عدد بزرگ‌تر را در خروجی بنویسد.

۱٤. الگوریتم و فلوچارتی بنویسید که دو عدد صحیح و مثبت را از ورودی بخواند سپس آن‌ها را با استفاده از عمل تفریق برهم تقسیم نماید.

۱۵. الگوریتم و فلوچارتی بنویسید که n عدد را خوانده، تعداد اعداد مثبت، منفی و صفر را نمایش دهد.

۱٦. الگوریتم و فلوچارتی بنویسید که n را خوانده، با استفاده از یک زیر الگوریتم تمام اعداد اول قبل از n را نمایش دهد.

۱۷. الگوریتم و فلوچارتی رسم کنید که n را خوانده، با استفاده از یک زیر الگوریتم تمام اعداد تام (کامل) قبل از عدد خوانده شده را نمایش دهد (عددی کامل است که مجموع مقسوم علیه‌هایش به جز خودش برابر با آن عدد باشد مانند ۶ و ۲۸).

۱۸. الگوریتم و فلوچارتی رسم کنید که عددی را خوانده، با استفاده از یک زیر الگوریتم فاکتوریل، مجموع فاکتوریل ارقام آن عدد را نمایش دهد.

۱۹. الگوریتم و فلوچارتی رسم کنید که n را خوانده، مجموع تمام اعداد فیبوناچی تا عدد خوانده شده (n) را نمایش دهد.

۲۰. الگوریتم و فلوچارتی رسم کنید که n عدد را خوانده، در آرایه قرار دهد. سپس، کوچک‌ترین عنصر و یکی مانده به کوچک‌ترین عنصر و مکان آن‌ها را نمایش دهد.

۲۱. الگوریتم و فلوچارتی رسم کنید که اضلاع مثلثی را گرفته تشخیص دهد متساوی اضلاع است یا خیر.

۲۲. الگوریتم و فلوچارتی رسم کنید که اضلاع مثلثی را خوانده تعیین کند قائم‌الزاویه است یا خیر.

۲۳. الگوریتم و فلوچارتی رسم کنید که مجموع ۱۰ جمله سری زیر را نمایش دهد:

$$\frac{1}{250}+\frac{3}{245}+\frac{4}{240}+\frac{5}{235}+\dots$$

۲۴. الگوریتم و فلوچارتی رسم کنید که n را خوانده، حاصل n جمله سری زیر را نمایش دهد:

$$4-\frac{4}{3}+\frac{4}{5}-\frac{4}{7}+\frac{4}{9}-\dots$$

۲۵. الگوریتم و فلوچارتی رسم کنید که n را خوانده، حاصل n جمله سری زیر را نمایش دهد (برای محاسبه توان و فاکتوریل از الگوریتم فرعی استفاده شود): $\sum_{k=1}^{n}\frac{3^k}{k!(k-1)!}$

۲۶. الگوریتم و فلوچارتی رسم کنید که عدد n را خوانده، مقلوب آن را نمایش دهد.

۲۷. الگوریتم‌ها و فلوچارت‌های رسم کنید که خروجی‌های زیر را نمایش دهند:

```
1
1  2
1  2  3                                    *
1  2  3  4                                 *  *  *
1  2  3  4  5                           *  *  *  *  *
1  2  3  4                           *  *  *  *  *  *  *
1  2  3                           *  *  *  *  *  *  *  *  *
1  2                           *  *  *  *  *  *  *  *  *  *  *
1
```

۲۸. الگوریتم و فلوچارتی رسم کنید که تمام اعداد چهار رقمی که در آن‌ها حداقل دو رقم یک وجود داشته باشد را نمایش دهد.

۲۹. الگوریتم و فلوچارتی رسم کنید که کلیه اعداد سه رقمی که حداقل دو رقم آن‌ها برابر باشند را نمایش دهد.

۳۰. الگوریتم و فلوچارتی رسم کنید که n عدد را خوانده، تعیین کند هر کدام چند رقمی می‌باشد، چند رقم آن‌ها فرد و چند رقم آن‌ها زوج است.

۳۱. الگوریتم و فلوچارتی رسم کنید که x و n را خوانده، مجموع n جمله سری زیر را نمایش دهد (برای محاسبه توان و فاکتوریل از دو الگوریتم فرعی استفاده کنید): $x-\frac{x^2}{2!}+\frac{x^4}{4!}-\frac{x^6}{6!}+\frac{x^8}{8!}-\dots$

۳۲. الگوریتم و فلوچارتی رسم کنید که تمام اعداد متقارن ۵ رقمی را نمایش دهد.

۳۳. الگوریتم و فلوچارتی رسم کنید که تعدادی عدد را خوانده، هر یک از اعداد که بر ۹ بخش‌پذیراند را در خروجی چاپ کند (عددی بر ۹ بخش‌پذیر است که مجموع ارقام آن بر ۹ بخش‌پذیر باشد).

۳۴. الگوریتم و فلوچارتی رسم کنید که x و n را خوانده، حاصل n جمله سری زیر را چاپ کند:

$$s=\frac{1}{x}-\frac{1}{x+2x^2}+\frac{1}{x+2x^2+3x^3}-\dots$$

۳۵. الگوریتم‌ها و فلوچارت‌هایی رسم کنید که خروجی‌های زیر را نمایش دهند:

```
1 1 1 1 1 1 1 1      $ $ $ $ $ $ $ $
2 2 2 2 2 2 2        $ $ $ $ $ $ $ $
3 3 3 3 3 3          $ $ $ $ $ $ $ $
4 4 4 4 4            $ $ $ $ $ $ $ $
5 5 5 5              $ $ $ $ $ $ $ $
6 6 6                $ $ $ $ $ $ $ $
7 7                  $ $ $ $ $ $ $ $
8                    $ $ $ $ $ $ $ $
```

۳۶. الگوریتم و فلوچارتی رسم کنید که عددی را خوانده تشخیص دهد این عدد جزء سری فیبوناچی است یا خیر (سری فیبوناچی به صورت زیر می‌باشد):

1 1 2 3 5 8 13 21 34 ...

۳۷. الگوریتم و فلوچارتی رسم کنید که تمام اعداد سه رقمی که حاصل ضرب ارقام آن‌ها بیش از نصف خودشان است را نمایش دهد.

۳۸. الگوریتم و فلوچارتی رسم کنید که کلیه اعداد چهار رقمی که مجموع ارقام دوم و سوم برابر حاصل ضرب رقم‌های اول و چهارم است را نمایش دهد.

۳۹. الگوریتم و فلوچارتی رسم کنید که کلیه اعداد بین x1 و x2 که مجموع ارقام آن‌ها برابر x3 است را نمایش دهد (x1، x2 و x3 را از ورودی بخواند).

۴۰. الگوریتم و فلوچارتی رسم کنید که x1 و x2 را خوانده، اعداد اولی که بین x1 و x2 هستند را نمایش دهد (برای تعیین اعداد اول از یک الگوریتم فرعی استفاده کنید).

۴۱. الگوریتم و فلوچارتی رسم کنید که ۱۰ عنصر از یک آرایه را خوانده، سپس تمام اعدادی که بیشتر از میانگین عناصر آرایه هستند را نمایش دهد.

۴۲. الگوریتم و فلوچارتی رسم کنید که عددی را خوانده، تشخیص دهد مربع کامل است یا خیر. چند عدد مربع کامل عبارت‌اند از: 0 1 4 9 16 25 36 49 ...

۴۳. الگوریتم و فلوچارتی رسم کنید که x و n را خوانده، حاصل n جمله سری زیر را چاپ کند:

$$s = \frac{1}{x} - \frac{1}{x - 2x^2} + \frac{1}{x - 2x^2 + 3x^3} - \dots$$

۴۴. الگوریتم و فلوچارتی رسم کنید که x و n را خوانده، حاصل n جمله سری زیر را چاپ کند:

$$s = \frac{1!}{x} - \frac{2!}{x - 2x^2} + \frac{3!}{x - 2x^2 + 3x^3} - \dots$$

۴۵. الگوریتم و فلوچارتی رسم کنید که n را خوانده، رقم اول عدد n را بتوان رقم دوم، نتیجه را به توان رقم سوم و همین طور ادامه داده، نتیجه به دست آمده را به توان رقم آخر برساند. به عنوان مثال اگر n عدد ۷۲۳ وارد شود، مقدار ۷ به توان ۲ و نتیجه آن به توان ۳ را محاسبه نماید.

فصل ۲

آشنایی با زبان C++

زبان‌های برنامه‌سازی متعددی وجود دارند. یکی از این زبان‌های برنامه‌سازی، C++ است. این زبان در سال ۱۹۸۰ توسط بیارنی استراستاپ[6] در آزمایشگاه بل ابداع گردید. زبان C++ توسعه یافته زبان C است (هر فردی که با کامپیوتر برنامه‌نویسی می‌کند با زبان C آشنا است). زبان C ، یک زبان برنامه‌سازی ساخت یافته است. اما، زبان C++ از شیوه شی‌گرایی برای نوشتن برنامه‌ها استفاده می‌کند.

۱ – ۲. سطوح مختلف زبان‌های برنامه سازی

زبان‌های برنامه‌سازی با توجه به امکانات و پیچیدگی به سه سطح زیر تقسیم می‌شوند:

۱. سطح پایین
۲. سطح بالا
۳. سطح میانی

۱ – ۱ – ۲ . *زبان‌های سطح پایین*

زبان‌های سطح پایین، همان **زبان ماشین** نام دارند. در این زبان‌ها، برنامه به صورت 0 یا 1 (زبان واقعی کامپیوتر) نوشته می‌شود. زیرا، کامپیوتر فقط 0 یا 1 را می‌فهمد (شکل ۱ - ۲). برنامه نوشته شده به این زبان دارای سرعت بالای است. اما، درک و فهم آن برای انسان‌ها بسیار مشکل می‌باشد. به همین دلیل، زبان اسمبلی را ایجاد کردند. زبان اسمبلی به جای 0 یا 1 از یک سری نمادها تشکیل شده است.

قطعه برنامه شکل ۱ – ۲ اعداد فرد 1 تا 10 را جمع کرده، در ثبات AX قرار می‌دهد.

```
  MOV  AX, 0
  MOV  BX, 1
L1:CMP CX, 10
  JG    Exit
  ADD  AX, BX
  ADD  BX, 2
  JMP   L1
Exit:
```

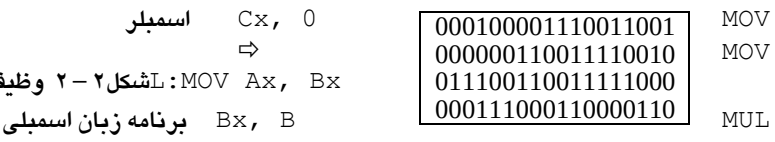
```
1000011110100111101
1001101100100101011
1001101001100110011
1001100111001100110
```

شکل ۱ – ۲ نمونه برنامه به زبان ماشین و اسمبلی.

همان‌طور که در شکل ۱ – ۲ می‌بینید، برنامه نوشته شده به زبان اسمبلی نسبت به برنامه‌های زبان ماشین قابل فهم‌تر شده است. اما کامپیوتر نمی‌تواند آن را اجرا کند. برای این که کامپیوتر بتواند آن را اجرا نماید باید به زبان ماشین تبدیل شود. به همین دلیل اسمبلر نوشته شده است. **اسمبلر**، برنامه‌ای است که برنامه نوشته شده به

[6].Biarne Stroustrup

زبان اسمبلی را گرفته و به زبان ماشین (همان صفر یا یک) تبدیل می‌کند تا کامپیوتر بتواند آن را اجرا نماید (شکل ۲ – ۲).



اسمبلر ⇨

```
L: MOV Ax, Bx
   MOV Cx, 0
   MOV Ax, 1
   MUL Bx, B
```

شکل۲ – ۲ وظیفه اسمبلر، تبدیل برنامه زبان اسمبلی به زبان ماشین.

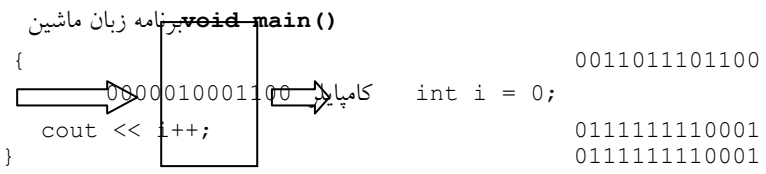
همان‌طور که ملاحظه کردید، نوشتن برنامه‌های اسمبلی خیلی راحت‌تر از نوشتن برنامه‌های زبان ماشین می‌باشد (امروزه برنامه‌ها را به زبان ماشین (0 یا 1) نمی‌نویسند). بنابراین، زبان سطح پایین، همان زبان اسمبلی می‌باشد. چون درک برنامه‌های زبان ماشین نیاز به اطلاعات کافی از سخت‌افزار کامپیوتر دارد و از طرف دیگر، درک آن‌ها نیز برای انسان‌ها مشکل است، زبان‌های سطح بالا را تولید کردند. در ادامه شرح این زبان‌ها را می‌بینید.

۲ – ۱ – ۲ . زبان‌های سطح بالا

در زبان‌های سطح بالا، دستورات به زبان انسان نزدیک‌تر شده‌اند. به عنوان مثال، در این زبان‌ها می‌توان بیان کرد که **اگر شرط خاصی درست باشد، این عمل را انجام بده، در غیر این صورت، عمل دیگر را انجام بده**، این کار را ۳۰ بار تکرار کن و غیره.

همانند برنامه‌های اسمبلی، برنامه‌های نوشته شده به زبان سطح بالا باید به زبان ماشین تبدیل گردند تا توسط کامپیوتر قابل اجرا باشند. برای این منظور، مترجم‌ها نوشته شده‌اند. دو نوع مترجم وجود دارد که عبارت‌اند از:

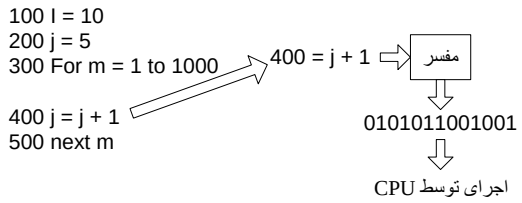
۱. کامپایلر (Compiler)، فایل برنامه به زبان سطح بالا را می‌گیرد، یک‌باره به زبان ماشین ترجمه می‌کند و برنامه‌ای به زبان ماشین ایجاد می‌نماید. شکل ۳ – ۲ فرآیند انجام ترجمه برنامه سطح بالا را نشان می‌دهد.
زبان‌هایی مانند C، پاسکال، کوبول، ++VC، #C، ویژوال‌بیسیک، زبان‌های کامپایلری هستند.



شکل ۳ – ۲ فرآیند ترجمه با کامپایلر.

۲. مفسر (Interpreter)، به جای این که کل برنامه را به زبان ماشین تبدیل کند، در زمان اجرا، اولین خط برنامه را خوانده آن را به زبان ماشین تبدیل می‌کند و سپس، اجرا می‌نماید. در ادامه، خط بعدی را گرفته، ترجمه و اجرا می‌نماید و این روند را ادامه می‌دهد (شکل ۴ – ۲). زبان‌هایی مثل جاوا و بیسیک به زبان مفسری معروفند.

زبان‌های مفسری نسبت به زبان‌های مترجمی، سرعت پایین‌تر و کارایی کمتری دارند. اما، آموزش آن‌ها راحت‌تر بوده و قابلیت اجرای آن‌ها روی پلت فرم‌های(محیط‌های) مختلف بیشتر است.



شکل ۴ – ۲ فرآیند ترجمه زبان‌های مفسری.

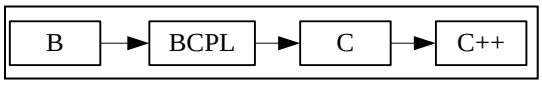
۳ – ۱ – ۲ . زبان‌های سطح میانی

زبان‌های سطح میانی، زبان‌هایی هستند که دستورات زبان سطح بالا را با توابع اسمبلی با هم پیاده‌سازی کردند. نمونه‌ای از این زبان می‌توان زبان C را نام برد. زبان‌های سطح میانی نیز برای ترجمه برنامه به زبان ماشین از کامپایلر استفاده می‌کنند.

۲ – ۲. ویژگی‌های زبان برنامه‌نویسی ++C

زبان C در سال ۱۹۷۲ توسط دنیس ریچی[۷] نوشته شد. زبان C نیز توسعه یافته زبان BCPL است. زبان BCPL را مارتین ریچارد[۲] طراحی نمود. زبان‌های BCPL و C توسعه یافته B هستند. زبان B در سال ۱۹۷۰ توسط کن تامسون[۳] طراحی گردید.

این زبان‌ها از دسته زبان‌های ساخت یافته بودند. اما، زبان ++C در اوایل سال ۱۹۸۰ توسط بیارنی استراستاپ اختراع گردید. این زبان توسعه یافته زبان C می‌باشد. برخلاف زبان C در این زبان از روش برنامه‌نویسی شی‌گرا (OOP)[۴] استفاده می‌شود. روند تکامل زبان‌ها در زیر آمده است.



برخی از ویژگی‌های زبان ++C در زیر آمده است:

۱. زبان ++C، یک زبان سطح میانی است.
۲. زبان ++C از روش برنامه‌نویسی شی‌گرا استفاده می‌کند.
۳. زبان ++C طوری طراحی شده است که قابل حمل باشد.
٤. کلمات کلیدی (دستورات تشکیل دهنده) این زبان کم است.
۵. این زبان دارای کتابخانه‌های زیادی است. یعنی، توابع زیادی دارد. این توابع دسته‌بندی شده‌اند و برای منظور خاصی طراحی گردیدند. برنامه‌نویس در صورت نیاز می‌تواند کتابخانه مربوط به این توابع را به برنامه اضافه کرده، از توابع آن استفاده نماید.
٦. این زبان نسبت به حروف بزرگ و کوچک حساس است. یعنی، این زبان در نام‌گذاری شناسه‌ها بین حروف بزرگ و کوچک فرق قائل می‌شود.

---

[7] .Dennis Rich [2]. Martin Richards [3]. Ken Thompson [4]. Object Oriented Program

۷. زبان C++ طوری طراحی شده است که می‌توان تمام برنامه‌های مورد نیاز را با آن نوشت.

۳ – ۲. آموزش زبان‌های برنامه‌نویسی

آموزش زبان‌های برنامه‌نویسی مانند زبان‌های طبیعی و زنده دنیا است. یعنی، مانند زبان‌های طبیعی برای آموزش زبان‌های برنامه‌نویسی باید مراحل زیر را انجام داد:

۱. مانند هر زبان طبیعی ابتدا باید علائم تشکیل دهنده زبان را آموخت. به عنوان مثال، زبان فارسی از علائم الف تا ی، ارقام ۰ تا ۹ و علائم خاص مانند ! ، : ، ؟ و غیره تشکیل شده است. هرکدام از این علائم (نمادها) مفهوم خاصی دارند. زبان C++، نیز از علائم a تا z ، A تا Z ، ۰ تا ۹ ، علائم ویژه نظیر ; ، : ،] ، [، / تشکیل شده است. ابتدا باید مفاهیم هر یک از این علائم را در زبان C++ آموخت.

۲. همان‌طور که می‌دانید از ترکیب علائم هر زبان کلمات به وجود می‌آیند. برخی از کلمات دارای معنی و مفهوم هستند و برخی دیگر معنی و مفهوم خاصی ندارند. به عنوان مثال، کلمات **بابا، آب و داد**، در زبان فارسی مفهوم خاصی دارند. ولی کلمات **تپتانم و بکیاپ** مفهوم خاصی ندارند. به کلماتی که در زبان دارای مفهوم خاص هستند، **کلمات کلیدی** می‌گویند. در زبان C++ کلمات کلیدی نظیر for، if، else، while، int وجود دارند. در آموزش یک زبان ابتدا باید کلمات کلیدی آن را شناخت و معنی و کاربرد هرکدام از آن‌ها را آموخت.

۳. با ترکیب کلمات کلیدی با یک قواعد خاص در هر زبان طبیعی، جمله ایجاد می‌شود (مانند بابا آب داد). همان‌طور که می‌دانید در زبان فارسی ابتدا فاعل، سپس مفعول و در پایان فعل قرار می‌گیرد. در زبان C++ نیز برای ایجاد جملات (دستورات) قواعد خاصی وجود دارد. به عنوان مثال، int برای تعریف داده‌های نوع صحیح به کار می‌رود که به صورت، زیر استفاده می‌گردد:

int متغیر۱ , متغیر۲n , ... , متغیر;

٤. همان‌طور که می‌دانید، در زبان‌های طبیعی با کنار هم قرار دادن جملات مرتبط به هم پاراگراف ایجاد می‌شود. در زبان‌های برنامه‌نویسی نیز با کنار هم قرار دادن دستورات مرتبط به هم، بلاک ایجاد می‌شود. در زبان C++، هر بلاک با } شروع و با { خاتمه می‌یابد.

۵. کنار هم قرار دادن پاراگراف‌ها صفحات و فصول را ایجاد خواهند کرد و این روند ادامه می‌یابد تا یک کتاب نوشته شود. در زبان‌های برنامه‌سازی نیز نوشتن برنامه‌ها هم همین روند را دارد. کنار هم قرار دادن بلاک‌ها، فایل، و کنار هم قرار دادن فایل‌های مرتبط به هم، برنامه را ایجاد می‌کند. بنابراین، در ادامه کتاب به آموزش زبان C++ با این شیوه خواهیم پرداخت.

۴ – ۲. کلمات کلیدی

کلمات کلیدی[8]، کلماتی هستند که در زبان شناخته شده‌اند و مفهوم خاصی دارند. به عنوان مثال، کلماتی نظیر if (اگر)، else (در غیر این صورت)، void (هیچ)، for (تکرار) کلیدی هستند. برخی از کلمات کلیدی C و C++ را در جدول ۱ – ۲ می‌بینید. در ادامه مفاهیم این کلمات را مشاهده خواهید کرد.

۵- ۲. انواع داده‌ها

در زبان C++ داده‌های مختلفی وجود دارند که عبارتند از:

۱. داده‌های اولیه، داده‌های استاندارد نظیر int، float، double، char و غیره که در زبان وجود دارند.

۲. داده‌هایی که کاربر تعریف می‌کند. داده‌هایی نظیر آرایه‌ها، مجموعه‌ها، ساختمان‌ها، رشته و کلاس‌ها که توسط برنامه‌نویس تعریف می‌شوند. در ادامه با برخی از این داده‌ها آشنا خواهید شد.

۳. داده‌های اشاره‌گر، برای نگهداری آدرس فیزیکی متغیرها به کار می‌روند. این نوع داده‌ها را در ادامه می‌آموزیم.

جدول ۱ – ۲ کلمات کلیدی زبان C و C++.

کلمات کلیدی مشترک زبان C و C++				کلمات کلیدی مختص زبان C++				
break	case	char	const	_eqxor	bool	catch	class	spascename	bitand	bitor
default	do	double	else	compl	delete	dynamic_cast	explicit	export	false	asm
extern	float	for	go to	friend	inline	mutable	and_eq	new	try	
int	long	register	return	private	protected	public	true	xor		
signed	eofsiz	static	struct	not	not_eq	operator	or_eq	static_cast		
deftype	union	unsigned	void	template	this	throw	typeid	or		
while	enum	continue	auto	using	virtual	wchar_t	and			
volatile	switch	short	if	const_cast	reinterpret_cast					

۱ – ۵ – ۲. داده‌های اولیه

همان‌طور که بیان گردید، داده‌هایی که به صورت استاندارد در زبان وجود داشته باشند، انواع داده‌های **اصلی یا اولیه** نام دارند. این داده‌ها انواع مختلفی دارند که در زیر آمده‌اند:

۱. **نوع داده عددی**، نوع داده‌ای که برای نگهداری اعداد به کار می‌رود. انواع داده‌ای عددی نیز دو نوع می‌باشند که عبارت‌اند از:

- **داده‌های عددی صحیح**، برای نگهداری مقادیر عددی صحیح مثبت و منفی به کار می‌روند. انواع این داده‌ها، تعداد بایت‌هایی که نیاز دارند و محدوده مقادیر آن‌ها را در جدول ۲ – ۲ می‌بینید.
- **داده‌های اعشاری**، برای نگهداری اعداد اعشاری مثبت و منفی به کار می‌روند. این انواع نیز در جدول ۲ – ۲ آمده‌اند.

۱. نوع داده‌ای کاراکتری

[8]. keywords

داده‌های کاراکتری برای ذخیره یک کاراکتر (هر علامتی که بین دو تک کوتیشن قرار می‌گیرد) به کار می‌روند. این نوع داده را نیز در جدول ۲ – ۲ مشاهده می‌کنید.

هر داده کاراکتری به یک بایت حافظه نیاز دارد.

۲. **نوع داده‌ای منطقی**، این نوع داده‌ها برای نگهداری نتیجه ارزیابی عبارت‌های منطقی (true و false) به کار می‌روند. این نوع داده‌ها را نیز در جدول ۲ – ۲ می‌بینید.

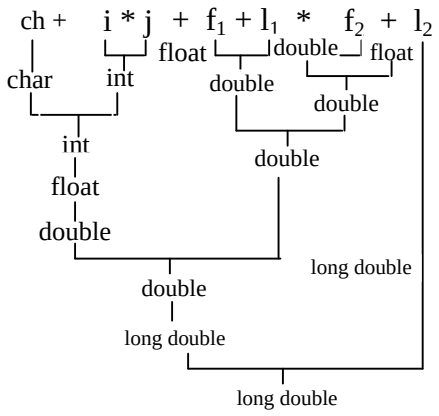
جدول ۲ – ۲ انواع داده اولیه در C++.

نوع داده	نام نوع	محدوده داده	تعداد بایت	مثال
عددی صحیح با طول کوتاه	short int	-128…..127	1	-120
عددی صحیح	int	-32,768 … 32,767	2	3015
عدد صحیح مثبت	unsigned int	0 …. 65,535	2	2798
عدد صحیح با طول بلند	long int	**-2,147,483,648 … 2,147,483,647**	4	298976
عدد صحیح مثبت با طول بلند	unsigned long	0…42,4967295	4	9999999
عدد اعشاری	float	1017549435e-38… 3040282347e+38	4	112075
عدد اعشاری با دقت مضاعف	double	2022507385858507201 4e-308 … 107976931348623157e+308	8	0.000000001
عدد اعشاری با دقت مضاعف(۱۵ رقم اعشار)	long double	107e-308…107e+308	8	1000000000. 125791575
کاراکتری	char	نگهداری یک کاراکتر127 …128-	1	'a'
منطقی	bool	true یا false	1	false

۱۰ – ۲. تبدیل نوع

همان‌طور که بیان گردید، زبان C++ از انواع مختلف داده تشکیل شده است. در عبارت محاسباتی و انتساب، داده‌ها به یکدیگر تبدیل خواهند شد. در عبارت محاسباتی انواع کوچک‌تر به انواع بزرگ‌تر تبدیل می‌شوند. به عنوان مثال، دستورات زیر را در نظر بگیرید:

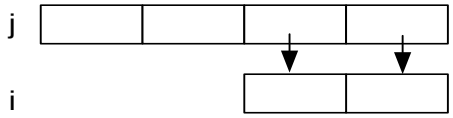
```
char ch;
int i, j;
float f1, f2;
double l1;
long double l2;
```



اما در انتساب، عبارت سمت راست به نوع داده سمت چپ تبدیل می‌گردد. در این تبدیل باید دقت داشته باشید که اگر طول داده سمت راست بیشتر از طول داده سمت چپ باشد، وگرنه بخش با ارزش داده سمت راست در انتساب از دست خواهد رفت. این عمل **سرریز**[9] نام دارد. به عنوان مثال، دستورات زیر را ببینید:

```
int i;
long int j = 123798;
i = j;
```

در این انتساب، چون j چهار بایتی است و i دو بایتی می‌باشد. بنابراین، انتساب به صورت زیر انجام خواهد شد:



همان‌طور که می‌بینید، دو بایت با ارزش j در هنگام انتساب در i از دست خواهد رفت.

[9] .Overflow

روش دیگر تبدیل نوع، تبدیل صریح می‌باشد. این روش، به صورت زیر انجام می‌شود:

متغیر = (نوع متغیر) متغیر;

به عنوان مثال، دستورات زیر را ببینید:

```
int a;
float f = 10.2;
```

دستور سوم مقدار f را به نوع int تبدیل کرده (10) و در a قرار می‌دهد. `a = (int) f;`

```
#include < ۱ فایل سرآیند >
...
#include < n فایل سرآیند >
نوع main ()
{
    main دستورات;
    return (مقدار);
}
1نوع f1 ()
 {
     1 دستورات تابع ;
       return (مقدار);
 }
   ...
n نوع fn ()
{
   n دستورات تابع;
   return (مقدار);
 }
```

۱۱ – ۲. ساختار برنامه C++

ساختار برنامه‌های C++ در شکل ۱ – ۲ آمده است. همان‌طور که در این شکل می‌بینید، ابتدا فایل‌های سرآیند[10] (کتابخانه‌ای)، سپس تعریف ثوابت و در ادامه کدهای برنامه قرار می‌گیرند. در فایل‌های سرآیند توابع مختلفی وجود دارند. برنامه‌نویس با توجه به نیاز خودش فایل‌های کتابخانه‌ای را به برنامه اضافه می‌کند. این فایل‌ها با عبارت #include شروع می‌شوند و نام آن‌ها بین < > قرار می‌گیرد (پسوند فایل‌های کتابخانه‌ای معمولاً h. است). برخی از فایل‌های سرآیند C++ و کاربرد آن‌ها را در جدول ۹ – ۲ می‌بینید. بعد از معرفی فایل‌های کتابخانه‌ای تابع اصلی main قرار می‌گیرد.

این تابع به صورت زیر به کار می‌رود:

```
int  main ()
{
  دستورات;
  return مقدار;
}
```

شکل ۱ – ۲ ساختار برنامه‌های C++.

در این ساختار، چون نوع تابع main، int است. بنابراین، می‌تواند عدد صحیح 0 یا هر مقدار دیگری را برگرداند. اگر این تابع مقدار 0 را برگرداند، بدین معنی است که تابع کارش را با موفقیت به پایان رسانده است.

همان‌طور که می‌بینید در C++، هر برنامه از یک یا چند تابع تشکیل شده است که یکی را تابع اصلی (()main) و بقیه را فرعی هستند. در حالت ساده برنامه فقط از یک تابع اصلی (یعنی ()main) تشکیل می‌گردد.

جدول ۹ – ۲ برخی از فایل‌های سرآیند و کاربرد آن‌ها.

نام فایل	هدف
iostream.h	شامل توابع ورودی و خروجی از قبیل cin و cout است.
math.h	شامل توابع ریاضی از قبیل sin، cos، sqrt و غیره است.
string.h	شامل توابع کار با رشته‌ها از قبیل strcpy، strcmp و غیره است.
conio.h	شامل توابعی برای پاک کردن صفحه نمایش (clrscr)، انتقال مکان‌نما (gotoxy) و غیره است.
graphics.h	شامل توابع گرافیکی از قبیل کشیدن خط (line). کشیدن دایره (circle) و غیره است.
dos.h	شامل توابع سیستم عامل Dos از قبیل کار با دستور سیستم (system) و غیره است.

ساختار دوم تابع ()main در زیر آمده است:

[10] Header File

```
void  main()
{
  دستورات;
}
```

از این ساختار زمانی استفاده می‌گردد که تابع اصلی main نمی‌خواهد هیچ مقداری را برگرداند. همان طور که در این ساختار مشاهده می‌گردد، تابع main از نوع void تعریف شده است. بنابراین، هیچ مقداری را برنمی‌گرداند. اگر نوع تابع اصلی یا فرعی void باشد می‌توان دستور return را حذف کرد.

همان‌طور که در شکل ۱ – ۲ مشاهده کردید، هر تابع از جمله تابع اصلی با } شروع می‌گردد و با { خاتمه می‌یابد. در هنگام تایپ دستورات برنامه C++، به نکات زیر دقت کنید:

۱. بین # و include هیچ فاصله‌ای قرار نمی‌گیرد.

۲. فایل‌های سرآیند بین < , > یا جفت کوتیشن قرار می‌گیرند.

۳. به جزء معرفی فایل‌های سرآیند، تعریف توابع، }، { و توضیحات، بقیه دستورات با علامت ; خاتمه می‌یابند.

۴. بین نوع تابع (void یا int) و نام تابع اصلی (main) حداقل یک فاصله قرار می‌گیرد.

۵. توضیحات یک خطی با علامت // شروع می‌شوند. به عنوان مثال، دستور زیر را ببینید:

```
//this is a comment in C++
```

توضیحات برای افزایش خوانایی برنامه به کار می‌روند.

۶. توضیحات چند خطی بین */ و /* قرار می‌گیرند.

به عنوان مثال، دستورات زیر را ببینید:

```
/*this program, uses
The cast operator*/
int  n = 10;
n += 13;
```

۷. چنانچه تابع اصلی (main) را از نوع void تعریف نمایید، نیازی نیست دستور return را در انتهای آن قرار دهید.

۸. اگر نوع تابع اصلی (main) با نوع int تعریف شود، حتماً با دستور return یک مقدار صحیح را برگردانید. اگر مقدار صفر را برگردانید سیستم‌عامل متوجه می‌شود که برنامه کارش را با موفقیت به پایان رسانده است.

۹. سعی کنید سرآیندی که از توابع آن‌ها در برنامه استفاده نمی‌کنید، در ابتدای برنامه قرار ندهید.

۱۰. متغیرهایی را که می‌خواهید در برنامه از آن‌ها استفاده کنید، حتماً تعریف نمایید.

۱۱. متغیرهایی که در برنامه به آن‌ها نیازی ندارید، تعریف نکنید.

مثال ۱۳ – ۲. برنامه‌ای که کاراکتری را از ورودی می‌خواهد (هدف برنامه آشنایی با ساختار برنامه است).

```
#include  "conio.h"
      void   main()
                    {
            getch();
                    }
```

توضیح: در این برنامه ابتدا، فایل conio.h با دستور #include معرفی می‌گردد تا بتوان از تابع ;()getch استفاده کرد. سپس، تابع ()main تعریف می‌شود. در ادامه با کاراکتر { بلاک باز می‌شود (بلاک برنامه اصلی) و تابع ;()getch فراخوانی می‌شود تا کاراکتری را از کاربر دریافت نماید. در پایان، برنامه با کاراکتر } خاتمه می‌یابد.

۱۲ – ۲. دستورات ورودی و خروجی

همان‌طور که در فصل اول بیان گردید، هر برنامه از سه بخش تشکیل شده است:

۱. **دستورات ورودی**، همان فرضیات برنامه را از کاربر دریافت می‌کنند.

۲. **دستورات پردازش**، رابطه‌ها و پردازش‌های مورد نیاز را بر روی ورودی انجام می‌دهند تا خروجی مناسب را تولید کنند. به این مرحله **پردازش** می‌گویند. با برخی از دستورات پردازش قبلاً آشنا شدید.

۳. **دستورات خروجی**، اطلاعات و نتایج تولید شده را به کاربر نمایش می‌دهند.

۱ – ۱۲ – ۲. *دستورات ورودی*

در C++ برای خواندن اطلاعات از صفحه کلید از شیء cin استفاده می‌شود. این شیء به صورت زیر به کار می‌رود:

<< cin نام متغیر۱ << نام متغیر n۲ << ... << نام متغیر;

این شیء در فایل سرآیند iostream.h قرار دارد. یعنی، برای استفاده از این شیء باید دستور زیر را در ابتدای برنامه قرار دهید:

```
#include <iostream.h>
```

در هنگام ورود داده باید به نکات زیر توجه کنید:

۱. نوع داده نظیر به نظیر با نوع متغیرهای دستور cin یکی باشد.

۲. بین داده‌ها حداقل یک فاصله قرار گیرد.

۳. در پایان ورود داده‌ها، جهت خاتمه، کلید Enter را فشار دهید.

مثال ۱۴ – ۲. دستورات زیر دو مقدار عددی صحیح و اعشاری را از کاربر دریافت کرده، در متغیرهای x و f قرار می‌دهند.

```
int  x;
float f;
cin >> x >> f;
```

۲ – ۱۲ – ۲. *دستورات خروجی*

در C++ برای نمایش اطلاعات در صفحه نمایش از شیء cout به صورت زیر استفاده می‌گردد:

cout << عبارت ۱ >> عبارت ۲ >> ... >> عبارت n ;

این شیء نیز مانند cin در فایل سرآیند iostream.h قرار دارد. در هنگام استفاده از cout به نکات زیر توجه کنید:

۱. برای چاپ مقادیر رشته‌ای آن‌ها را در بین جفت کوتیشن (") قرار دهید.

۲. جهت چاپ مقادیر کاراکتری آن‌ها را بین تک کوتیشن (') قرار دهید.

۳. در هنگام چاپ می‌توان از کاراکترهای کنترلی استفاده کرد. قبل از کاراکترهای کنترلی کاراکتر \ قرار می‌گیرد. برخی از این کاراکترها را در جدول ۱۰ – ۲ می‌بینید.

جدول ۱۰ – ۲ کاراکترهای کنترلی C++.

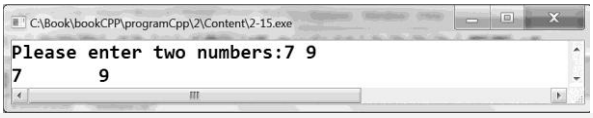
کاراکتر	هدف	کاراکتر	هدف
\\	برای چاپ \ به کار می‌رود.	\?	برای چاپ ؟ به کار می‌رود.
\n یا endl	در چاپ از سطر فعلی عبور می‌کند.	\t	کنترل را در چاپ به tab بعدی منتقل می‌کند.
\"	برای چاپ کاراکتر " به کار می‌رود.	\b	کاراکتر قبلی را حذف می‌کند.
\a	بوق سیستم را به صدا در می‌آورد.	\r	معادل کلید enter است.

مثال ۱۵ – ۲. برنامه‌ای که دو عدد را خوانده، نمایش می‌دهد (هدف این برنامه آشنایی با دستورات cin، cout و تعریف متغیرها است).

توضیح: این برنامه ابتدا متغیرهای a و b را تعریف می‌کند. سپس، دستور cout پیام Please enter two numbers: را نمایش می‌دهد و از طریق دستور cin دو عدد را از کاربر دریافت کرده، در متغیرهای a و b قرار می‌دهد. در پایان، دستور cout، مقدار متغیرهای a و b را نمایش می‌دهد.

متغیر	هدف
a	عدد اول
b	عدد دوم

```
#include  "conio.h"
#include  "iostream.h"
void   main()
{
 int a, b;
 cout << "Please enter two numbers:";
 cin >> a  >>  b ;
 cout << a << "\t" << b;
 getch();
}
```

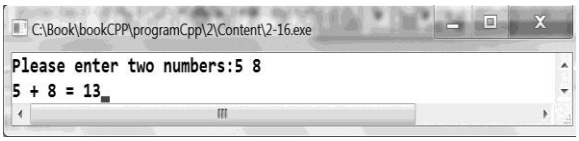


مثال ۱۶ – ۲. برنامه‌ای که دو عدد را خوانده، مجموع آن‌ها را نمایش می‌دهد.

توضیح: این برنامه ابتدا متغیرهای a و b را عددی صحیح تعریف کرده (نوع int)، سپس پیام Please enter two numbers: را با دستور cout نمایش می‌دهد. در ادامه با دستور cin مقادیر متغیرهای a و b را از کاربر دریافت می‌کند. در پایان، مقادیر a، b و مجموع آن‌ها را نمایش می‌دهد.

متغیر	هدف
a	عدد اول
b	عدد دوم

```
#include "iostream.h"
#include "conio.h"
void main()
{
  int a, b;
  cout << "Please enter two numbers:";
  cin >> a >> b;
  cout << a   << " + " << b << " = " << a + b;
  getch();
}
```

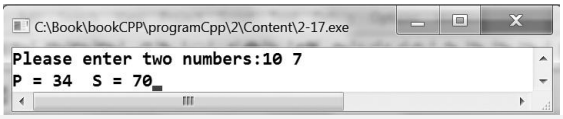


مثال ۱۷ – ۲. برنامه‌ای که اضلاع مستطیل را خوانده، محیط و مساحت آن را نمایش می‌دهد (مساحت مستطیل برابر با طول ضرب در عرض است و محیط آن برابر با طول + عرض ضرب در ۲ می‌باشد).

توضیح: این برنامه ابتدا، متغیرهای a (طول)، b (عرض)، p (محیط) و s (مساحت) را از نوع int تعریف کرده، سپس پیام :Please enter two numbers را نمایش می‌دهد. در ادامه، با دستور cin مقادیر a و b را از کاربر دریافت می‌نماید. دستور ;p = (a + b) * 2 محیط را محاسبه کرده، در p قرار می‌دهد و دستور ;s = a * b مساحت را محاسبه می‌کند، در s قرار می‌دهد. در پایان، محیط و مساحت نمایش داده می‌شوند.

```
#include "iostream.h"
#include "conio.h"
void main()
{
  int a, b, p, s;
  cout << "Please enter two numbers:";
  cin >> a >> b;
  p = (a + b ) * 2;
  s = a * b;
  cout << "P = "<< p << "\tS="<< s;
  getch();
}
```

متغیر	هدف
a	طول مستطیل
b	عرض مستطیل
p	محیط مستطیل
s	مساحت مستطیل

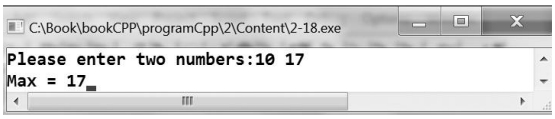


مثال ۱۸ – ۲. برنامه‌ای که دو عدد را خوانده، عدد بزرگ‌تر را نمایش می‌دهد (هدف برنامه کاربرد عملگر ؟ است).

توضیح: این برنامه ابتدا متغیرهای a (عدد اول)، b (عدد دوم) و max (عدد بزرگ‌تر) را از نوع int تعریف می‌کند. سپس، با دستور cout پیام :Please enter two numbers را نمایش می‌دهد. در ادامه با دستور cin مقادیر متغیرهای a و b را با cin می‌خواند و با دستور ;max = (a > b) ? a : b عدد بزرگ‌تر را در متغیر max قرار می‌دهد. در پایان، با یک پیام مقدار max را نمایش می‌دهد.

```
#include "iostream.h"
#include "conio.h"
void main()
{
  int a, b, max;
  cout << "Please enter two numbers:";
  cin >> a >> b;
  max = (a >  b ) ? a : b;
  cout << "Max = "   << max;
  getch();
}
```

متغیر	هدف
a	عدد اول
b	عدد دوم
max	عدد بزرگ‌تر



منظور خواهد شد.intاگر نوع تابعی ذکر نگردد، به طور پیش‌فرض

فصل ۴ توابع

ساختار یک برنامه C++ خیلی شبیه به ساختار یک سازمان است. یعنی، در هر سازمان ساختار سلسله مراتبی حاکم است. در بالاترین سطح سازمان، مدیریت قرار دارد. هر مدیر می‌تواند چند معاون داشته باشد. هر یک از معاونین نیز می‌توانند چندین کارمند داشته باشند. در برنامه‌های C++، تابع main نقش مدیریت را بازی می‌کند. در برنامه‌های که تاکنون نوشته شده‌اند، کل عملیات برنامه در تابع main انجام شده است. این برنامه‌ها ساده بودند. این برنامه‌ها مانند شرکت‌های کوچک هستند که مدیر شرکت همه کارهای شرکت را انجام می‌دهد. ولی، در بسیاری از سازمان‌ها چنین وضعیتی حاکم نیست. برنامه‌های واقعی و کاربردی مانند سازمان‌های بزرگ طولانی و پیچیده هستند.

تابع main برای انجام هر وظیفه‌اش (مانند مدیر) یکی از توابع (معاونین) خودش را صدا می‌زند و احتمالاً پارامترهای (پرونده‌های) را در اختیار او قرار داده، از او می‌خواهد کار را انجام داده، نتیجه را برگرداند. هر یک از توابع (معاونین) نیز خود توابع دیگر (کارمندان خودش) را صدا می‌زنند تا بخشی از کار را به آن‌ها محول نمایند و این روند تا انجام کار ادامه دارد. با این تفاسیر، کاربری که از برنامه استفاده می‌نماید، نقش مشتری را بازی خواهد کرد که می‌تواند اطلاعاتی را در اختیار سازمان (برنامه) قرار داده، نتایجی را دریافت کند.

استفاده از تابع در برنامه‌نویسی دارای مزایای زیر است:

۱. برنامه‌نویسی ساخت یافته را امکان‌پذیر می‌کند.

۲. خوانایی برنامه را افزایش می‌دهد. همچنین تست، اشکال‌زدایی و خطایابی برنامه نیز آسان‌تر خواهد شد. چون، برنامه‌ها به بخش‌های کوچک‌تری تبدیل می‌شوند. لذا، خطایابی و اصلاح برنامه‌های کوچک‌تر آسان‌تر خواهد بود.

۳. می‌توان توابع مورد نیاز را در یک برنامه نوشت و از آن‌ها در برنامه‌های دیگر نیز استفاده کرد. این امر، **استفاده مجدد**[۲] نام دارد. بدین ترتیب، کد نویسی کمتر خواهد شد و تولید نرم‌افزار سریع‌تر انجام می‌شود.

۴. توابع، امکان کار گروهی را فراهم می‌کنند. زیرا، پس از این که برنامه به بخش‌های کوچک‌تری تقسیم شدند، هر یک از اعضای گروه وظیفه نوشتن و تست توابع مشخص را بر عهده می‌گیرند. بدین ترتیب، اعضای گروه به صورت هم‌زمان روی بخش‌های مختلف برنامه کار می‌کنند (بدون این که منتظر همدیگر باشند). انجام کار به صورت گروهی موجب می‌شود تا برنامه‌ها سریع‌تر آماده شوند.

۵. توابع امکان استفاده از کارهایی که دیگران انجام داده‌اند، را فراهم می‌کنند. یعنی، در برنامه‌هایتان می‌توانید از توابعی که دوستانتان آماده کرده‌اند، استفاده کنید.

۶. توابع، امکان ایجاد کتابخانه را فراهم می‌کنند. کتابخانه، مجموعه توابعی هستند که مورد نیازتان می‌باشند (مجموعه توابعی که به هم مرتبط‌اند). بنابراین، می‌توان مجموعه توابع مرتبط به هم را در یک فایل کتابخانه قرار داد و در برنامه‌ها از این فایل کتابخانه استفاده نمود.

۱ – ۴. انواع توابع

در هر زبان برنامه‌نویسی دو نوع تابع وجود دارد که عبارت‌اند از:

۱. **توابع کتابخانه‌ای**، توابعی می‌باشند که همراه کامپایلر وجود دارند. این توابع را توابع عمومی نیز می‌نامند. زیرا، کاربردهای زیادی دارند. توابع کتابخانه‌ای را با توجه به کاربرد آن‌ها دسته‌بندی کردند و هر یک از دسته‌ها را در فایل سرآیند خاصی قرار داده‌اند. تاکنون با برخی از این توابع نظیر getche()، getch() آشنا شدید (این توابع در فایل سرآیند conio.h قرار دارند). در ادامه با بعضی از توابع مهم کتابخانه‌ای به همراه کاربرد آن‌ها آشنا خواهید شد.

۲ – ۴. توابعی که برنامه نویس می‌نویسد

همان‌طور که می‌دانید C++ شامل کتابخانه متنوعی است. اما، با این وجود، توابع موجود در کتابخانه C++، پاسخ‌گوی همه در خواست‌های برنامه‌نویس نیستند. لذا، برنامه‌نویس باید بتواند توابعی را نوشته، از آن‌ها استفاده کند. برای این منظور، برنامه‌نویس باید دو کار زیر را انجام دهد:

۱. نوشتن تابع **۲. فراخوانی تابع**

۱ – ۲ – ۴. *نوشتن تابع*

نوشتن تابع خود نیز دارای دو بخش بسیار مهم است. این دو بخش عبارت‌اند از:

۱. اعلان الگوی تابع ۲. تعریف تابع

اعلان الگوی تابع

قبل از این که تابعی را بنویسید باید الگوی آن را اعلان کنید. الگوی تابع تعیین می‌کند، این تابع چه ورودی‌های دارد، چه چیزی را برمی‌گرداند (خروجی تابع چیست) و چه عملی را انجام می‌دهد. الگوی (امضای) تابع، به صورت زیر است:

نوع تابع نام تابع (لیست پارامترها) ;

☒ **نوع تابع**، نوع مقداری را تعیین می‌کند که تابع برمی‌گرداند. این نوع می‌تواند یکی از انواع تعریف شده در C++ باشد، توابع از لحاظ مقداری که برمی‌گردانند به سه نوع زیر تقسیم می‌شوند:

۱. **توابعی که هیچ مقداری را برنمی‌گردانند**. به جای نوع این توابع کلمه کلیدی void قرار می‌گیرد (مثال ۱ – ۴ را ببینید).

۲. **توابعی که فقط یک مقدار را برمی‌گردانند.** نوع مقداری که این توابع برمی‌گردانند، در قبل از نام تابع قرار می‌گیرد. برخی از این توابع عبارت‌اند از:

۱. تابعی که بزرگ‌ترین مقدار بین سه عدد را برمی‌گرداند.

۲. تابعی که تعیین می‌کند عددی اول است یا خیر.

۳. تابعی که تعیین می‌کند عددی کامل (تام) است یا خیر.

۴. تابعی که حاصل ضرب دو عدد را برمی‌گرداند.

۵. و غیره

می‌کنند. دستور return به صورت‌های زیر به کار می‌رود:

```
1. return    مقدار;
2. return (عبارت);
3. return;
```

ساختار اول، یک مقدار را برمی‌گرداند. به عنوان مثال، دستورات زیر را ببینید:

```
return false;
return 10;
```

دستور اول، مقدار false را برمی‌گرداند و دستور دوم، مقدار ۱۰ را برگشت خواهد داد.

اما، ساختار دوم، یک عبارت را ارزیابی کرده، نتیجه ارزیابی عبارت را برگشت خواهد داد. به عنوان مثال، دستورات زیر را مشاهده کنید:

```
return (2 * i – 3);
return ( (a > b) ? a : b);
```

دستور اول، ۲ را در i ضرب کرده، ۳ واحد از این حاصل کم می‌کند و برمی‌گرداند. اما، دستور دوم، بین دو عدد a و b، عدد بزرگ‌تر را برمی‌گرداند.

ساختار سوم، بدون این که تابع مقداری را برگرداند، از تابع برمی‌گردد. این دستور معمولاً در انتها تابع نوع void به کار می‌رود.

۳. **توابعی که چندین مقدار را برمی‌گردانند.** این مقادیر از طریق پارامتر برگردانده می‌شوند. بنابراین، پارامترها باید به صورت ارجاع تعریف شوند.

☒ **نام تابع،** از قانون نام‌گذاری شناسه‌ها و متغیرها پیروی می‌کند و برای دسترسی به تابع به کار می‌رود.

☒ **پارامترهای تابع،** اطلاعاتی هستند که در هنگام فراخوانی تابع باید به آن ارسال گردند. توابع می‌توانند از لحاظ تعداد پارامترهایی که می‌پذیرند به گروه‌های زیر تقسیم گردند:

۱. **توابع بدون پارامتر،** این توابع معمولاً برای چاپ پیغام مشخصی به کار می‌روند. برای توابع بدون پارامتر بهتر است به جای پارامترها کلمه کلیدی void نوشته شود. اگر کلمه کلیدی void را ذکر نکنید، کامپایلر خطایی نمی‌گیرد.

۲. **تابع ممکن است یک پارامتر داشته باشد.** در این صورت باید نوع و نام پارامتر تعیین گردد.

برخی از این توابع عبارت‌اند از:

☒ تابعی که عددی را دریافت کرده، تعیین می‌کند اول است یا خیر.

- تابعی که عددی را دریافت کرده، تعیین می‌کند تام است یا خیر.
- تابعی که عددی را دریافت کرده، فاکتوریل آن را برمی‌گرداند.
- تابعی که عددی را دریافت کرده، مجموع ارقام آن را برمی‌گرداند.
- **تابع ممکن است چندین پارامتر داشته باشد.** در این صورت باید تعریف پارامترها با استفاده از کاما (,) از هم جدا شوند. برخی از این توابع در زیر آمده‌اند:
- تابعی که دو عدد را دریافت کرده، بزرگ‌ترین مقسوم علیه مشترک آن‌ها را برمی‌گرداند (این تابع دارای دو پارامتر است).
- تابعی که سه عدد را دریافت کرده، کوچک‌ترین عدد را برمی‌گرداند (این تابع سه پارامتر دارد).
- تابعی که دو عدد را دریافت کرده، اولین عدد را به توان عدد دوم رسانده و برمی‌گرداند.
- تابعی که دو عدد را گرفته، محتویات آن‌ها را تعویض می‌کند.
- و غیره.

تعریف تابع

تعریف تابع علاوه بر این که الگوی (امضای) تابع را تعیین می‌کند، مشخص می‌نماید تابع چه عملی را باید انجام دهد. عملی که تابع باید انجام دهد، در بدنه تابع قرار می‌گیرد. **بدنه تابع،** مجموعه دستوراتی هستند که تابع باید اجرا کند. تعریف تابع به صورت زیر است:

```
(لیست پارامترها) نام تابع    نوع تابع
{
      ;دستور ۱
      ;دستور ۲
          ...
      ;دستورn
      return    [مقدار];
}
```

نکته	اعلان توابع قبل از تابع ()mian، اما، تعریف آن‌ها بعد از تابع ()mian قرار می‌گیرد.

۲ ـ ۲ ـ ۴. فراخوانی تابع

دستوری که تابع را صدا زده، از آن استفاده می‌کند، **فراخوانی تابع** نام دارد. فراخوانی تابع به صورت زیر انجام می‌شود:

```
;(لیست آرگومان‌ها ) نام تابع
```

در هنگام نوشتن و استفاده از توابع باید به نکات زیر توجه کنید:

۱. الگوی (امضای) کلیه توابع قبل از تابع اصلی (()main) قرار گیرد.

۲. نوع تابع را ذکر کنید. اگر نوع تابعی ذکر نشود، int منظور می‌گردد.

۳. الگوی تابع باید در هنگام اعلان الگو و تعریف تابع یکی باشد. در غیر این صورت، کامپایلر C++ خطا صادر می‌کند.

۴. تعداد و نوع پارامترها (در هنگام تعریف تابع) باید با تعداد و نوع و آرگومان‌ها (در هنگام فراخوانی) یکسان باشند (ممکن است نام آن‌ها یکی نباشد).

۵. در هنگام اعلان الگوی (امضای) تابع می‌توانید نام پارامترها را حذف کنید (نوع آن‌ها ذکر شود).

۶. در C++ نمی‌توانید تابعی را در داخل تابع دیگر تعریف کنید.

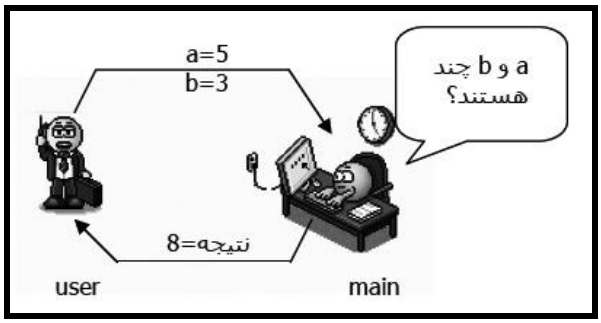
۷. سعی کنید تمام توابع را بعد از تابع اصلی (()mian) تعریف کنید. چنانچه تابعی قبل از تابع ()main معرفی کرده‌اید، نیازی نیست الگوی تابع را ذکر نمایید.

درک عملکرد تابع

برای درک عملکرد توابع، برنامه‌ای را بدون استفاده از توابع و سپس از طریق توابع پیاده‌سازی می‌کنیم. با همین مثال نیز چگونگی تبدیل یک برنامه معمولی به توابع را می‌آموزیم. فرض کنید، بخواهید برنامه‌ای بنویسید که دو عدد را خوانده، حاصل جمع آن‌ها را نمایش دهد. همان‌طور که قبلاً دیدید، این برنامه به صورت زیر پیاده‌سازی می‌شود (روش اول):

```
#include "iostream.h"
#include "conio.h"
void main()
{
  int a, b, c;
  cout << "Enter a, b:";
  cin >> a >> b;
  c = a + b;
  cout << a << " + " << b << " = " << c;
}
```

در این برنامه، تابع main همه کاره است. یعنی، دو عدد صحیح را از کاربر (مشتری) گرفته، خودش حاصل جمع دو عدد را محاسبه کرده، چاپ می‌کند (در اختیار مشتری قرار می‌دهد). این فرآیند در شکل ۱-۴ آمده است.



شکل ۱-۴ فرآیند اجرای برنامه.

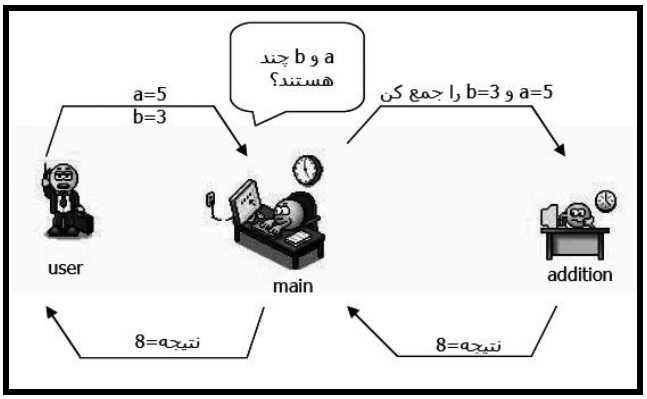
در روش دوم پیاده‌سازی، تابع main (مدیریت) دو عدد a و b را از کاربر گرفته، در اختیار تابع addition (کارمند خودش) قرار می‌دهد تا حاصل جمع این دو عدد را حساب کند. تابع addition پس از محاسبه حاصل جمع دو عدد (مانند کارمند) نتیجه را در اختیار تابع main (مدیریت) قرار می‌دهد و تابع main این نتیجه را به کاربر (مشتری) می‌دهد. پیاده‌سازی این برنامه به صورت زیر است:

```
#include "iostream.h"
#include "conio.h"
int addition (int a, int b)
{
  return a + b;
}
void main()
{
   int a, b, c;
   cout << "Enter a, b:";
   cin >> a >> b;
   c = addition (a, b);
   cout << a << "+" << b << "=" << c;
}
```

این فرآیند در شکل ۲-۴ آماده است. در این فرآیند اتفاق زیر می‌افتد:

```
int addition(int  a, int  b)
↓8              ↑       ↑
c = addition (    5,      3);
```

یعنی، مقدار ۵ در a و مقدار ۳ در b تابع addition قرار خواهد گرفت و نتیجه جمع a و b به c برگردانده خواهد شد.



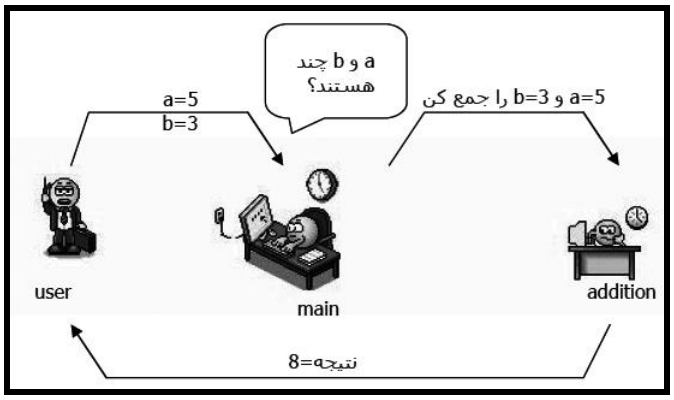
شکل ۲-۴ فرآیند حالت دوم اجرای برنامه.

روش سوم پیاده‌سازی که ممکن است اتفاق افتد، تابع main (مدیریت) دو عدد را از کاربر (مشتری) دریافت کرده، در اختیار تابع addition (کارمند) قرار می‌دهد، تابع addition حاصل جمع دو عدد را محاسبه کرده، به کاربر (مشتری) نشان می‌دهد و هیچ چیزی را به تابع main (مدیریت) برنمی‌گرداند. این روش به صورت زیر پیاده‌سازی می‌شود:

```
#include "iostream.h"
#include "conio.h"
void addition (int a, int b)
{
  int c = a + b;
  cout << a << " + " << b << " = " << c;
}
void main()
{
   int a, b;
   cout << "Enter a, b:";
```

```
    cin >> a >> b;
    addition (a, b);
    getch();
}
```

فرآیند انجام این روش در شکل ۳-۴ آمده است.

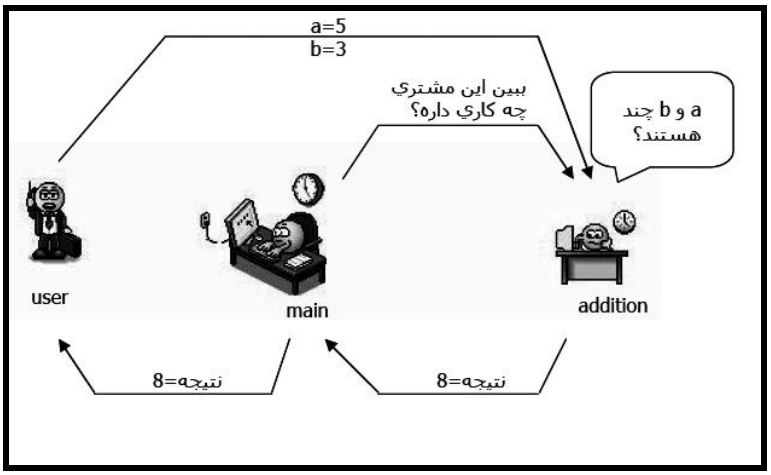


شکل ۳-۴ فرآیند پیاده‌سازی برنامه با روش سوم.

روش چهارم که برای پیاده‌سازی ممکن است اتفاق افتد، تابع main (مدیریت)، تابع addition (کارمند خودش) را صدا می‌زند، تابع addition دو عدد را از کاربر می‌گیرد (با مشتری صحبت می‌کند) و در a و b قرار می‌دهد، ولی نتیجه را در اختیار تابع main (مدیریت) قرار می‌دهد تا تابع main (مدیریت) به کاربر نشان دهد. این روش به صورت زیر پیاده‌سازی می‌شود:

```
#include "iostream.h"
#include "conio.h"
int addition ()
{
   int a, b;
   cout << "Enter a, b:";
   cin >> a >> b;
   return a + b;
}
void main()
{
  int c = addition();
  cout << c;
  getch();
}
```

فرآیند اجرای این روش در شکل ۴-۴ آمده است.

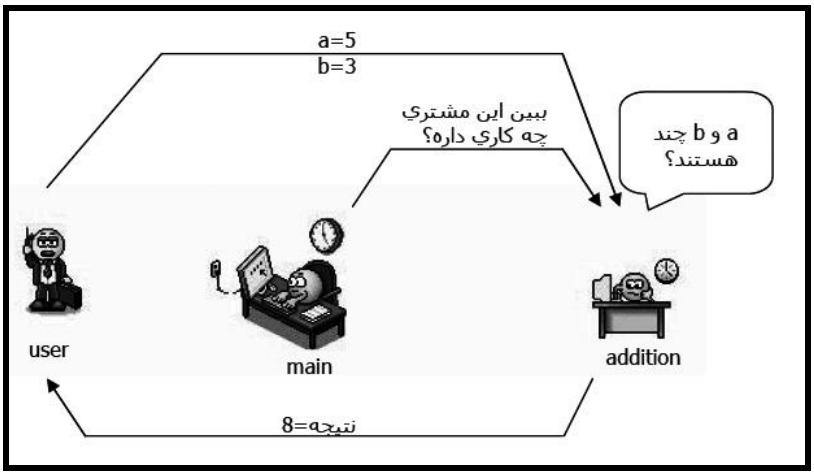


شکل ۴-۴ چهارمین روش پیاده‌سازی برنامه نمونه.

حالت دیگر (پنجمین روش) که ممکن است رخ دهد، تابع main (مدیریت) هیچ کار خاصی را انجام ندهد و تنها تابع addition (کارمند خودش) را صدا می‌زند، تابع addition اعداد a و b را از کاربر (مشتری) دریافت کرده، حاصل جمع آن‌ها را محاسبه کرده، به کاربر (مشتری) نشان می‌دهد. بنابراین، نیازی نیست هیچ نتیجه‌ای را به تابع main (مدیریت) برگرداند. پیاده‌سازی این برنامه به صورت زیر است.

```
#include "iostream.h"
#include "conio.h"
void addition (int a, int b)
{
   int a, b, c;
   cout << "Enter a, b:";
   cin >> a >> b;
   c = a + b;
   cout << a << " + " << b << " = " << c;
}
void main()
{
   addition ();
   getch();
}
```

فرآیند اجرای این روش پیاده‌سازی در شکل ۵-۴ آمده است.



شکل ۵-۴ خروجی پنجمین روش پیاده‌سازی برنامه نمونه.

مثال ۱ – ۴. برنامه‌ای که با استفاده از یک تابع پیغام First function. را نمایش می‌دهد 💻

توضیح: این برنامه از طریق یک تابع پیغامی را نمایش می‌دهد. بنابراین، تابع هیچ مقداری را برنمی‌گرداند و از طرف دیگر، هیچ پارامتری ندارد. بنابراین، الگوی این تابع به صورت زیر است:

void display (void);

نوع این تابع void است(هیچ مقداری را برنمی‌گرداند)، نام آن display می‌باشد و هیچ پارامتری ندارد. این الگو باید قبل از تعریف تابع اصلی (()main) قرار گیرد.

چون این تابع پیغام First function را نمایش می‌دهد، تعریف تابع به صورت زیر خواهد بود:

```
void    display (void)
{
  cout << "\nFirst function";
}
```

همان‌طور که در این تعریف می‌بینید، بدنه تابع فقط شامل یک دستور زیر است:

cout << " \nFirst function ";

چون این تابع هیچ مقداری را برنمی‌گرداند (نوع void است)، پس دستور retrun نیز در انتهای آن وجود ندارد. تعریف تابع باید بعد از تابع اصلی (()main) قرار گیرد (در کد برنامه ملاحظه فرمایید).

در ادامه جهت استفاده از تابع آن را فراخوانی کرده است. این عمل با دستور زیر در تابع اصلی (()main) انجام شده است:

display();

هدف این برنامه آشنایی با اعلان الگو، تعریف و فراخوانی تابع است.

```
#include "iostream.h"
#include "conio.h"
void display(void);
void main()
{
  display();
  getch();
}
void display(void)
{
  cout << "First function\n";
}
```

۳ – ۴. ارسال پارامترها

در هنگام فراخوانی توابع مقادیر آرگومان‌های مجازی به پارامترهای واقعی ارسال می‌گردند. این عمل با دو روش انجام می‌شود:

۱. **ارسال پارامتر از طریق مقدار** ۲. **ارسال پارامتر از طریق ارجاع**

۱ – ۳ -۴. ارسال پارامتر از طریق مقدار

در این روش، در هنگام فراخوانی مقدار آرگومان مجازی در پارامتر واقعی نظیرش کپی می‌گردد. از این به بعد ارتباط بین آرگومان مجازی و پارامتر واقعی قطع خواهد شد. یعنی، آرگومان مجازی یک خانه از حافظه و پارامتر واقعی خانه دیگری از حافظه می‌باشد. تغییر در پارامتر هیچ تأثیری بر روی آرگومان متناظر آن نمی‌گذارد(پارامتر واقعی از نوع فقط خواندنی در نظر گرفته می‌شود). در مثال ۲ – ۴ نمونه‌ای از ارسال پارامتر از طریق مقدار را می‌بینید.

۲ – ۳ – ۴. ارسال پارامتر از طریق ارجاع

در این روش، در هنگام فراخوانی تابع آدرس آرگومان مجازی در پارامتر واقعی نظیرش کپی خواهد شد. یعنی، پارامتر به آرگومان اشاره می‌کند. بنابراین، هر تغییری که در پارامتر انجام می‌شود، مستقیماً تأثیر آن بر روی آرگومان اعمال خواهد شد. یعنی، پارامتر واقعی از نوع خواندنی و نوشتنی در نظر گرفته می‌شود. در ادامه چگونگی ارسال پارامتر از طریق ارجاع را می‌بینید.

مثال ۲ – ۴. برنامه‌ای که عددی را خوانده، با استفاده از یک تابع توان ۲ عدد خوانده شده را محاسبه کرده، به برنامه اصلی برمی‌گرداند و برنامه اصلی آن را نمایش می‌دهد.

توضیح: همان‌طوری که در الگوی تابع می‌بینید، تابع مقداری از نوع double را برمی‌گرداند و دارای یک پارامتر از نوع double می‌باشد (الگوی زیر):

```
double  square (double);
```

در تعریف تابع نیز می‌بینید که نوع تابع double است. بنابراین، تابع باید مقداری از نوع double را برگرداند. این مقدار با دستور return برگشت داده شده است (تعریف زیر را ببینید):

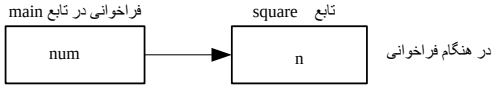
```
double   square(double   n)
{
    return    (n * n);
}
```

همان‌طور که در دستور return مشاهده می‌کنید، این تابع مقدار پارامتر را در خودش ضرب کرده، برمی‌گرداند. در فراخوانی تابع نیز مشاهده می‌گردد که مقدار num از طریق آرگومان برای پارامتر n تابع square() ارسال می‌گردد. یعنی، فراخوانی تابع به صورت زیر می‌باشد:

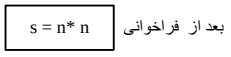
```
double  s = square (num);
```

در هنگام اجرای این دستور اعمال زیر انجام می‌شود:

۱. مقدار آرگومان num در پارامتر n کپی می‌گردد(مقدار متغیر num در m قرار می‌گیرد). ارسال پارامتر به صورت زیر است:



۲. تابع مقدار n را در n ضرب کرده، در متغیر s قرار می‌دهد. یعنی:




```
#include "iostream.h"
#include "conio.h"
double square(double);
void main()
{
  double num;
  cout << "Enter a number:";
  cin >> num;
  double s = square(num);
  cout << "Square(" << num <<"):" << s;
  getch();
}
double square(double n)
{
  return (n * n);
}
```

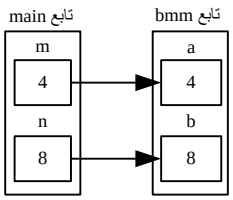
مثال ۳ – ۴. برنامه‌ای که دو عدد را خوانده، بزرگ‌ترین و کوچک‌ترین مقسوم علیه مشترک آن‌ها را نمایش می‌دهد.

توضیح: در این برنامه، تابع bmm() دو عدد را از طریق پارامترهای a و b دریافت کرده، بزرگ‌ترین مقسوم علیه مشترک آن‌ها را برمی‌گرداند و تابع kmm()، سه مقدار (a، b و بزرگ‌ترین مقسوم علیه مشترک آن‌ها) را به عنوان پارامتر دریافت کرده، کوچک‌ترین مقسوم علیه مشترک آن‌ها را برمی‌گرداند. کوچک‌ترین مقسوم-علیه مشترک برابر با حاصل ضرب دو عدد تقسیم بر بزرگ‌ترین مقسوم علیه مشترک است.

```
#include "iostream.h"
#include "conio.h"
int bmm(int, int);
int kmm(int, int, int);
void main()
{
  int m, n;
  cout << "Enter two numbers:";
  cin >> m >> n;
  cout << "bmm = "<<bmm(m,n)<<"\tkmm = " << kmm(m,n, bmm(m, n));
```

```
  getch();
}
int bmm(int a, int b)
{
for (int i= a; i >= 1; i--)
  if(a % i == 0 && b  % i == 0)
  {
    return i;
  }
}
int kmm(int a, int b, int c)
{
   return ( a * b / c);
}
```

همان‌طور که در کد این برنامه می‌بینید، تابع bmm() به صورت مقابل فراخوانی می‌شود: bmm (m , n);



یعنی، مقادیر m و n در a و b کپی می‌گردند (ارسال پارامتر از طریق مقدار است). ارسال پارامترها به صورت زیر است:

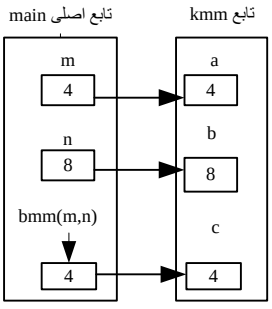
همان‌طور که در این شکل می‌بینید، مقادیر آرگومان‌های m و n به ترتیب در پارامترهای a و b کپی می‌گردند.

در ادامه تابع kmm() به صورت زیر فراخوانی می‌گردد:

kmm (m, n, bmm (m , n));

در هنگام برگشت از تابع kmm()، مقدار ۸ برگردانده می‌شود. زیرا، کوچک‌ترین مقسوم علیه مشترک برابر است با: a * b / bmm (a , b) = 4 * 8 / 4 = 8

روش فراخوانی تابع kmm و ارسال پارامتر های آن در شکل زیر آمده است:



۴ ـ ۴. طول عمر و محدوده حضور متغیرها

متغیرها علاوه بر ویژگی‌های از قبیل نام، نوع، مقدار و اندازه،

دو ویژگی دیگر دارند که عبارت‌اند از:

۱. طول عمر متغیر **۲. محدوده حضور متغیر**

۱ ـ ۴ ـ ۴. طول عمر متغیر

متغیرها در نقطه‌ای از برنامه که تعریف می‌شوند، به آن‌ها فضا اختصاص می‌یابد و حداکثر تا انتهای برنامه وجود خواهند داشت. متغیرها از لحاظ طول عمر به سه دسته تقسیم می‌گردند که عبارت‌اند از:

۱. متغیرهای auto، متغیرها به طول پیش‌فرض auto در نظر گرفته می‌شوند.

۲. متغیرهای static، متغیرهایی هستند که یک بار مقدار اولیه می‌گیرند و آخرین مقدار خودشان را نگهداری می‌کنند. در ادامه این متغیرها را می‌آموزیم.

۳. متغیرهای پویا (دینامیک)، متغیرهایی هستند که در هنگام اجرای برنامه به آن‌ها حافظه تخصیص می‌یابد و در ادامه برنامه حافظه تخصیص یافته به آن‌ها پس گرفته می‌شود.

۲ ـ ۴ ـ ۴ . محدوده حضور متغیر

محدوده حضور[11]، بخشی از برنامه است که " نام متغیر" معتبر و قابل استفاده می‌باشد. اگر بخواهید خارج از محدوده حضور متغیر به آن دست‌یابی داشته باشید، با خطا مواجه خواهید شد. برخی از متغیرها می‌توانند خارج از محدوده خودشان وجود داشته باشند، ولی دسترسی به آن‌ها امکان پذیر نیست (مانند متغیر static).

متغیرهای اتوماتیک (auto)

برای تعریف متغیرهای اتوماتیک، قبل از نوع متغیر از کلمه کلیدی atuo به صورت زیر استفاده می‌شود:

auto نام متغیر نوع متغیر;

اگر کلمه کلیدی auto ذکر نگردد، متغیر به طور پیش‌فرض auto در نظر گرفته می‌شود. متغیرهای اتوماتیک دو نوع‌اند که عبارت‌اند از:

۱. متغیرهای محلی **۲. متغیرهای سراسری**

متغیرهای محلی

محدوده این نوع متغیرها از نقطه تعریف تا پایان بلاک فعلی است. به همین دلیل این متغیرها، **بلاکی** نیز نام دارند. یعنی، در نقطه‌ی تعریف به آن‌ها حافظه تخصیص می‌یابد و در پایان بلاک فعلی حافظه تخصیص یافته به آن‌ها پس گرفته می‌شود. هر باری که بلاک حاوی متغیر اجرا می‌گردد، دوباره به آن‌ها فضا تخصیص می‌یابد. به همین دلیل، این متغیرها در فضای از **پشته**[2] قرار می‌گیرند. تمام متغیرهای که در داخل تابع تعریف می‌شوند، از نوع متغیر محلی هستند.

متغیرهای سراسری

متغیرهایی که در خارج از تمام بلاک‌ها تعریف می‌گردند و از آن نقطه‌ای که تعریف شده‌اند تا پایان فایل محدوده حضور آن‌ها می‌باشد. این متغیرها، **سراسری** نام دارند. متغیرهای سراسری دارای ویژگی‌های زیر هستند:

۱. در تمام توابع داخل همین فایل از نقطه‌ای که تعریف شده‌اند، قابل دست‌یابی‌اند.
۲. اگر مقدار اولیه به آن‌ها تخصیص ندهید، مقدار آن‌ها صفر منظور خواهد شد.
۳. متغیرهای سراسری فقط باید در یک فایل تعریف یا اعلان گردند.

---

[11].Scope [2]. Stack

متغیرهای سراسری و محلی می‌توانند همنام باشند. در این صورت، در بلاک فعلی (نقطه‌ی که متغیر محلی تعریف شده است)، متغیر سراسری مخفی خواهد شد. برای دسترسی به متغیر سراسری در بلاکی که متغیر محلی همنام در آن تعریف شده است، باید از عملگر :: استفاده کنید. به عنوان مثال، دستورات زیر را ببینید:

```
#include "iostream.h"
int a;
void main()
{
   int a = 1, b;
   b = a + :: a;
   cout << b;
}
```

این برنامه، ابتدا متغیر a را به صورت سراسری تعریف کرده، مقدار صفر را در آن قرار می‌دهد (اگر متغیر سراسری در هنگام تعریف مقدار اولیه نگیرد، مقدار صفر به آن تخصیص می‌یابد). در داخل تابع main()، متغیر a را به صورت محلی تعریف کرده و به آن مقدار ۱ تخصیص داده است. در ادامه، مقدار متغیر محلی a (مقدار ۱) و سراسری a (مقدار صفر) را با هم جمع کرده، در متغیر b قرار می‌دهد و در پایان، مقدار متغیر b را نمایش می‌دهد (یک را نمایش می‌دهد).

متغیرهای static

متغیرهای static به صورت زیر تعریف می‌شوند:

static نوع متغیر نام متغیر = [مقدار اولیه] ;

این متغیرها دارای ویژگی‌های زیر هستند:

۱. اگر مقدار اولیه آن‌ها ذکر نشود، مقدار اولیه صفر به آن‌ها تخصیص می‌یابد.

۲. اگر مقدار اولیه به آن‌ها تخصیص دهید، فقط یک بار این عمل انجام خواهد شد.

۳. آخرین مقدارشان را نگهداری می‌کنند.

به عنوان مثال، دستورات زیر را ببینید:

```
#include "iostream.h"
void main()
{
  for (int i = 1; i <= 3; i++) f1();
}
void f1()
{
  int a = 1;
  static int s = 5;
  cout << s++ << "\t" << a++ << "\t";
}
```

با اجرای این دستورات مقادیر زیر نمایش داده می‌شوند:

5 1 6 1 7 1

زیرا، در داخل تابع main() حلقه تکرار ۳ بار اجرا می‌شود و در هر مرتبه تابع f1() فراخوانی می‌شود. در اولین فراخوانی، متغیر a به صورت اتوماتیک تعریف می‌شود و مقدار یک در آن قرار می‌گیرد، سپس متغیر s به صورت static تعریف گردیده، مقدار ۵ در آن قرار می‌گیرد. در ادامه، مقادیر s و a (۵ و ۱) را نمایش

می‌دهد و یک واحد به آن‌ها اضافه می‌کند. در هنگام برگشت، متغیر a از بین می‌رود. چون این متغیر به صورت اتوماتیک تعریف شده است. اما، متغیر s از بین نمی‌رود (آخرین مقدار خودش را نگهداری می‌کند (۶)). در فراخوانی دوم تابع ()f1، مجدداً a تعریف شده، مقدار یک به آن تخصیص می‌یابد. ولی، متغیر s مجدداً تعریف نمی‌شود (مقدار قبلی خودش را دارد). در ادامه، مقادیر s و a (۶ و ۱) را نمایش می‌دهد و یک واحد به s و a اضافه می‌کند. در هنگام برگشت فراخوانی دوم نیز a از بین خواهد رفت و مقدار s حفظ می‌شود. در فراخوانی سوم، نیز a را مجدداً تعریف کرده، مقدار یک را به آن تخصیص می‌دهد و s مقدار قبلی خودش را حفظ می‌نماید (مقدار اولیه نمی‌گیرد) و در ادامه، مقادیر ۷ و ۱ را نمایش می‌دهد.

مثال ۴ – ۴. برنامه‌ای که حوزه دسترسی، طول عمر متغیرهای سراسری، محلی، static و همنام را نشان می‌دهد.

توضیح: در این برنامه چهار متغیر به نام‌های x تعریف شده است.

۱. **متغیر سراسری x (قبل تابع ()main)**، چون سراسری است به آن مقدار اولیه 0 تخصیص می‌یابد. از این متغیر می‌توان در توابع ()main و ()f1 استفاده کرد.

۲. **متغیر محلی x (داخل تابع ()main)**، در هنگام تعریف به آن مقدار ۱۰ تخصیص می‌یابد. از این متغیر فقط می‌توان در تابع ()main استفاده نمود.

۳. **متغیر استاتیک x (در داخل تابع ()f1)**، در هنگام تعریف مقدار اولیه 0 را به آن تخصیص می‌دهد و آخرین مقدارش را نگهداری می‌کند (یک بار تعریف شده و مقدار اولیه می‌گیرد). از این متغیر می‌توان در تابع ()f1 استفاده نمود.

۴. **متغیر محلی y (در تابع ()f1)**، در هنگام تعریف مقدار ۵ به آن تخصیص می‌یابد و هر بار وارد تابع می‌شود، دوباره تعریف شده، مقدار ۵ به آن تخصیص می‌یابد.

برنامه به صورت زیر عمل می‌کند:

قبل از تابع ()main متغیر x را تعریف کرده، در برنامه ()main نیز متغیر محلی x را تعریف می‌نماید، مقدار ۱۰ را در آن قرار می‌دهد. در ادامه، پیام x is و مقدار x را نمایش می‌دهد. یعنی، مقدار x تعریف شده در تابع ()main (۱۰) نمایش داده می‌شود. سپس، تابع ()f1 را فراخوانی می‌کند. در این تابع، متغیر x از نوع static تعریف می‌گردد، مقدار اولیه 0 به آن تخصیص می‌یابد، متغیر محلی y تعریف شده مقدار ۵ به آن تخصیص می‌یابد و مقدار متغیر محلی x و y را نمایش داده (مقدار 0 و 5). سپس به x و y یک واحد اضافه می‌کند. در ادامه، مقدار متغیر سراسری x را یک واحد اضافه می‌کند (::x++) و مقدار x سراسری (۱) را نمایش می‌دهد و برنامه اصلی نیز مقدار متغیر سراسری x (۱) را نمایش می‌دهد. سپس، تابع ()f1 فراخوانی می‌گردد. در تابع ()f1، دستور static int x =0 اجرا نمی‌شود. چون x به صورت static تعریف شده است. آخرین مقدار خودش را حفظ می‌کند و متغیر y تعریف شده، مقدار ۵ در آن قرار می‌گیرد. دستور cout مقدار

متغیر استاتیک x (۱) و مقدار متغیر y (۵) را نمایش می‌دهد. مقدار متغیر سراسری x را یک واحد اضافه می‌کند و مقدار آن (۲) را نمایش می‌دهد. بنابراین، خروجی زیر را نمایش می‌دهد:

مثال ۵ – ۴. برنامه‌ای که اعداد اول بین ۱ تا ۱۰۰ را نمایش می‌دهد. برای تعیین این که عددی اول است یا خیر، از روش زیر استفاده می‌کنیم:

۱. اگر عدد کوچک‌تر از ۲ باشد، اول نیست.

۲. اگر عدد کوچک‌تر از ۴ باشد، اول است.

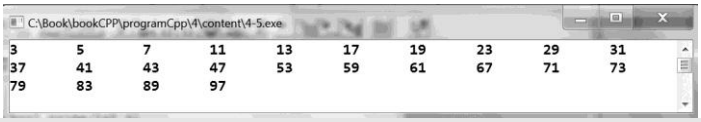
۳. اگر عدد زوج باشد، اول نیست.

۴. اگر هیچ یک از شرایط بیان شده را نداشته باشد، به اعداد فرد از ۳ تا نصف خودش تقسیم می‌گردد. اگر بر هیچ عدد فردی بخش‌پذیر نباشد، اول است.

توضیح: تابع prime، عدد n را به عنوان پارامتر دریافت کرده، اگر n اول باشد، true، وگرنه false را برمی‌گرداند.

تابع	متغیر	هدف
prime	nپارامتر i	عددی که باید مشخص گردد اول است یا خیر nشمارنده اعداد فرد سه تا نصف
main	i	شمارنده اعداد ۱ تا ۱۰۰

```
#include "iostream.h"
#include "conio.h"
bool prime(int n)
{
 if(n<2)  return false;
 if(n<4)  return true;
 if(n%2==0)return false;
 for (int i = 3; i <= n /2; i += 2)
     if (n % i == 0) return false;
 return true;
}
void main()
{
 for(int i = 0; i<= 100; i++)
 {
   if (prime(i)== true) cout << i << "\t";
 }
 getch();
}
```



مثال ۵. برنامه‌ای که یک عدد را خوانده، اگر عدد فرد باشد، مجموع و حاصل ضرب اعداد فرد تا آن عدد، وگرنه (اگر عدد زوج باشد)، مجموع و حاصل ضرب اعداد زوج تا آن عدد را محاسبه کرده، نمایش می‌دهد (تابع محاسبه مجموع (sum) با استفاده از حلقه for و تابع محاسبه حاصل ضرب (mul) به صورت بازگشتی نوشته شده است).

```
#include "iostream.h"
#include "conio.h"
long sum(int  n)
{
   long s = 0;
   for ( int i =n; i>= 1; i -= 2)
     {
```

```
        s += i;
        }
 return s;
 }
long  mul(int a)
{
  if ( a  < 1 )
  {
      return 1;
  }
  else return ( a * mul(a - 2) );
}
void main()
{
  int  a;
  cout << "Enter a number:";
  cin >> a;
  cout << "Sum is " << sum(a);
  cout << "\nMultiply is " << mul(a);
  getch();
}
```

مثال ۶. برنامه‌ای که عددی را خوانده، مجموع و حاصل ضرب ارقام آن را نمایش می‌دهد (این برنامه برای محاسبه مجموع ارقام از تابعی به نام sumDigit() استفاده می‌کند. این تابع از حلقه while برای محاسبه مجموع ارقام استفاده می‌نماید و برای محاسبه حاصل ضرب ارقام از یک تابع بازگشتی به نام mulDigit() استفاده می‌کند).

```
#include "iostream.h"
#include "conio.h"
int sumDigit(int  n)
{
  int sum  = 0;
  while ( n > 0)
  {
    sum += n % 10;
    n = n / 10;
  }
 return sum;
}
long  mulDigit(int a)
{
  if ( a  == 0  )
  {
      return 1;
  }
  else return ( (a  % 10) * mulDigit(a /10) );
}
void main()
{
  int  a;
  cout << "Enter a number:";
  cin >> a;
  cout << "Sum digit is " << sumDigit( a );
  cout << endl << "Multiply digit is " << mulDigit( a );
  getch();
}
```

مثال ۷. برنامه‌ای که عددی را خوانده، حاصل ضرب ارقام فرد و مجموع ارقام زوج آن را محاسبه کرده، نمایش می‌دهد (در این برنامه، برای محاسبه حاصل ضرب ارقام فرد از یک تابع بازگشتی و برای محاسبه مجموع ارقام زوج از تابع معمولی استفاده شده است).

```
                                        #include "iostream.h"
#include "conio.h"
int sumEvenDigit(long  n)
{
   int sum  = 0;
   while ( n > 0)
   {
     if ( n % 10 % 2 == 0 ) sum += n % 10;
     n /= 10;
   }
   return sum;
 }
long  mulOddDigit(long a)
{
  if ( a  == 0  )
  {
      return 1;
  }
  if ( a % 10 % 2 == 0 ) return (mulOddDigit(a /10));
  else return ( (a  % 10) *  mulOddDigit(a /10) );
}
void main()
{
  long  a;
  cout << "Enter a number:";
  cin >> a;
  cout << "Sum is " << sumEvenDigit( a );
  cout << endl << "Multiply is " <<mulOddDigit(a);
 getch();
}
```

مثال ۸. برنامه‌ای که دو عدد n و k را خوانده، تمام اعداد مضرب k و مجموع آن‌ها از ۱ تا n را نمایش می‌دهد (برای محاسبه مضرب k و مجموع آن‌ها از دو تابع استفاده شده است).

```
#include<iostream.h>
#include<conio.h>
long sumDividByK(long int, int k);
void dividByK(long int n, int k);
int main()
{
  long int n;
  int k;
  clrscr();
  cout << "Please enter n, k:";
  cin >> n >> k;
  cout << "Result is ";
  dividByK(n, k);
  cout << "\t = " <<  sumDividByK(n, k) ;
  getch();
  return 0;
}
long sumDividByK(long int n, int k)
{
  long sum = 0;
  while(n>0)
  {
     if((n % k)==0) sum += n;
     n--;
```

```
  }
  return sum;
}
void dividByK(long int n, int k)
{
   if (n == 0 ) return ;
   else
   {
     if((n % k) == 0)  cout << n <<"\t";
     dividByK(--n, k);
   }
}
```

مثال ۹. برنامه‌ای که n و x را خوانده، مجموع n جمله سری زیر را محاسبه کرده، نمایش می‌دهد:

$$sum = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{2\times 4} + \frac{x^5}{1\times 3\times 5} - \frac{x^6}{2\times 4\times 6} + \ldots$$

```
                                        #include<iostream.h>
#include<conio.h>
double pow(double x, int n);
double mul(int s, int n);
void main()
{
  int n, sign= 1;
  double x;
  clrscr();
  cout << "Please enter x, n:";
  cin >> x >> n;
  double s = 0;
  for(int i= 1; i<= n; i++)
  {
    s += sign * pow(x,i)/ mul(i % 2, i);
    sign = -sign;
  }
  cout << "Sum is " << s;
  getch();
}
double pow(double x, int n)
{
   if (n == 0 ) return (1.0);
   else
   {
      return ( x * pow ( x, n-1));
   }
}
double mul(int s, int n)
{
 double m = 1.0;
 for(int i = s + 2; i <= n ; i +=2)
 {
    m *= i;
 }
 return m;
}
```

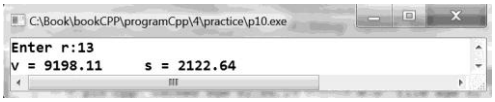
توضیح: در این برنامه، تابع pow()، x و n را دریافت کرده، x به توان n را برمی‌گرداند و تابع mul()، s و n را دریافت کرده، مخرج کسر را محاسبه کرده، برمی‌گرداند. اگر s یک باشد، اعداد فرد ۱ تا n را ضرب می‌نماید ولی، اگر s برابر صفر باشد، اعداد زوج ۲ تا n را با هم ضرب کرده، برمی‌گرداند.

مثال ۱۰. برنامه‌ای که شعاع کره‌ای را گرفته، مساحت و حجم آن را با استفاده از یک تابع محاسبه می‌نماید که به صورت ارجاع به برنامه اصلی برمی‌گرداند و در برنامه اصلی آن‌ها را نمایش می‌دهد.

حجم کره $= 3.1415 * r^3 * 4/3$ مساحت کره $= 3.1415 * r^2 * 4$

```
#include<iostream.h>
#include<conio.h>
void computeSphere(float,float&,float&);
void main()
{
 float v,s,r;
 cout << "Enter r:";
 cin >> r;
 computeSphere(r, v, s);
 cout << "v = " << v << "\t" << "s = " <<s << endl;
 getch();
}
void computeSphere(float r, float& v,float& s)
{
 const float PI =3.14;
```

```
    v = PI * r * r * r * 4/3;
    s = PI * r * r * 4;
}
```



توضیح: در این برنامه، تابع ComputeSpere()، سه پارامتر به نام‌های r (شعاع)، &v (حجم) و &s (شعاع) دارد. پارامتر r از نوع مقدار است. اما، پارامترهای &s و &v از نوع ارجاع می‌باشند که تغییرات‌شان را به برنامه (اصلی (man) برمی‌گردانند.

۱۲ ـ ۴. مسائل حل شده در سایت

۱. برنامه‌ای که x و y را خوانده، حاصل عبارت $\sqrt{|x-y|^{|y|}}$ را محاسبه کرده، نمایش می‌دهد.

۲. برنامه‌ای که نمره را به صورت عددی از ۰ تا ۱۰۰ خوانده، توسط تابعی مقدار حرفی معادل آن را برمی‌گرداند و نمایش می‌دهد (اگر نمره بین ۸۰ تا ۱۰۰ باشد، حرف A، اگر نمره بین ۶۰ تا ۸۰ باشد، حرف B، چنانچه نمره بین ۵۰ تا ۶۰ باشد، حرف C، وگرنه حرف F را نمایش می‌دهد).

۳. برنامه‌ای که سن فردی را به سال، ماه، روز دریافت کرده، توسط توابعی به روز، ساعت، دقیقه و ثانیه تبدیل کرده، برمی‌گرداند و نمایش می‌دهد (هر سال، ۳۶۵ روز و هر ماه ۳۰ روز است).

۴. برنامه‌ای که حقوق کارمندی را خوانده، توسط تابعی مالیات بر حقوق کارمند را محاسبه کرده، نمایش می‌دهد (مالیات از طریق جدول زیر محاسبه می‌شود):

از	تا	درصد مالیات
0	483000	0
483001	600000	%10
600001	1000000	%15
1000001	2000000	%20
2000001	99999999	%30

۵. برنامه‌ای که حاصل ضرب ارقام بالای ۵ عدد را از طریق تابع محاسبه کرده، نمایش می‌دهد.

۶. برنامه‌ای که حاصل جمع ارقام زوج عددی را از طریق تابع بازگشتی محاسبه کرده، نمایش می‌دهد.

۷. برنامه‌ای که n را خوانده و با استفاده از یک تابع در n سطر ۱۰ کاراکتر x را نمایش می‌دهد.

۸. برنامه‌ای که نمره پایان ترم و میان‌ترم دانشجویی را گرفته، با استفاده از یک تابع نمره نهایی دانشجو را محاسبه و برگرداند (نمره نهایی دانشجو برابر با نمره میان‌ترم * ۰,۴ + نمره پایان ترم * ۰,۶ است).

۹. برنامه‌ای که استفاده از متغیرهای محلی و سراسری را نمایش می‌دهد.

۱۰. برنامه‌ای که اطلاعات جعبه‌ای از قبیل طول، عرض و ارتفاع را دریافت کرده، با استفاده از تابعی حجم جعبه را محاسبه می‌کند. حجم جعبه برابر با طول * عرض * ارتفاع است.

۱۱. برنامه ای که قیمت کالا و درصد تخفیف را دریافت کرده، به تابعی ارسال کند، و این تابع میزان تخفیف کالا را برگرداند و در برنامه اصلی میزان تخفیف نمایش داده می‌شود.

۱۲. برنامه‌ای که میزان موجودی حساب بانکی را به همراه درصد بهره سالانه دریافت کرده، تعیین می‌کند پس از چند سال موجودی حساب (بدون برداشت بهره و هیچ مبلغی از حساب) به مبلغ خاصی می‌رسد (موجودی انتهای هر سال توسط تابعی محاسبه می‌گردد).

۱۳. برنامه‌ای که کاراکتری را از ورودی خوانده، توسط توابعی تعیین می‌کند کاراکتر خوانده شده رقم، حرف بزرگ یا حرف کوچک است.

۱۴. برنامه‌ای که n را خوانده، حاصل عبارت زیر را نمایش می‌دهد:

$1! + 2! + 3! + \ldots + n!$

این برنامه تابعی برای محاسبه فاکتوریل و تابع دیگری برای محاسبه مجموع دارد.

۱۵. برنامه‌ای که n را خوانده، حاصل عبارت زیر را نمایش می‌دهد:

$$\frac{1}{1!}+\frac{2}{2!}+\frac{3}{3!}+\ldots+\frac{n}{n!}$$

این برنامه تابعی برای محاسبه فاکتوریل و تابع دیگری برای محاسبه مجموع دارد.

۱۶. برنامه‌ای که x و n را خوانده، حاصل عبارت زیر را نمایش می‌دهد:

$$\frac{x}{1!}+\frac{x^2}{2!}+\frac{x^3}{3!}+\ldots+\frac{x^n}{n!}$$

این برنامه توابعی برای فاکتوریل، توان (x^n) و مجموع دارد.

۱۷. برنامه‌ای که x و n را خوانده، حاصل عبارت زیر را نمایش می‌دهد:

$$\frac{x}{1}+\frac{x^2}{2}+\frac{x^3}{1\times3}+\frac{x^4}{2\times4}+\frac{x^5}{1\times3\times5}+\ldots$$

۱۸. برنامه‌ای که x و n را خوانده، حاصل عبارت زیر را نمایش می‌دهد:

$$\frac{x}{1}-\frac{x^2}{2}+\frac{x^3}{1\times3}-\frac{x^4}{2\times4}+\frac{x^5}{1\times3\times5}-\frac{x^6}{2\times4\times6}+\ldots$$

۱۹. برنامه‌ای که دو مقدار عددی را خوانده، سپس توسط تابعی به عدد اول یک واحد اضافه نماید و از عدد دوم یک واحد کم کرده آن‌ها را در برنامه اصلی نمایش می‌دهد (در این برنامه دو تابع نوشته شده است. تابع اول، incdec که پارامترها را به صورت ارسال پارامتر با مقدار فراخوانی می‌کند. در این صورت تغییرات آرگومان مجازی برای پارامترهای واقعی ارسال نمی‌گردد. اما، تابع incdec1 دو پارامتر نوع ارجاع را می‌پذیرد و مقادیر تغییر یافته آن‌ها را به برنامه اصلی برمی‌گرداند).

۲۰. برنامه‌ای که x و k را خوانده، لگاریتم x در مبنای k را با یک تابع محاسبه کرده، نمایش می‌دهد.

۲۱. برنامه‌ای که عددی را خوانده ریشه دوم عدد را نمایش می‌دهد.

۲۲. برنامه‌ای که سه عدد را خوانده، توسط تابعی تشخیص می‌دهد، سه عدد تشکیل مثلث را نمی‌دهند (عدد صفر را برمی‌گرداند)، سه عدد تشکیل مثلث قائم‌الزاویه را می‌دهند (عدد یک را برمی‌گرداند)، سه عدد تشکیل مثلث متساوی‌الساقین را می‌دهند (عدد ۲ را برمی‌گرداند)، سه عدد تشکیل مثلث متساوی‌الاضلاع را می‌دهند (عدد ۳ را برمی‌گرداند)، وگرنه عدد ۱- را برمی‌گرداند.

۲۳. برنامه‌ای که حرکت جت را نمایش می‌دهد.

۲۴. برنامه‌ای که چرخش منظومه‌های مختلف دور خورشید را نمایش می‌دهد.

۲۵. برنامه‌ای که ابتدا دایره‌ای رسم می‌کند و این دایره تدریج به استوانه تبدیل می‌شود. در ادامه استوانه بزرگ‌تر شده به تدریج به نقطه شروع رسم دایره وصل می‌گردد تا یک تصویر زیبا ایجاد گردد.

۲۶. برنامه‌ای که اعداد n و b را خوانده، عدد n را به روش بازگشتی از مبنای ۱۰ به مبنای b تبدیل می‌کند (b، مبنای کوچک‌تر از ۱۰ است).

۲۷. برنامه‌ای که با روش بازگشتی مقدار مشتق nام را محاسبه می‌کند:

$$\frac{d(n)}{d_x n}(e^{ax})$$

اگر از تابع $(e^{ax})^n$ مشتق بگیریم، رابطه بازگشتی زیر به دست می‌آید:

$(e^{ax})^n = n \times (e^{ax})^{n-1}$

۲۸. برنامه‌ای که sin، cos و tan زاویه‌های زوج ۱ تا ۹۰ درجه را در خروجی با فرمت جدول بندی مناسب تا سه رقم اعشار نمایش می‌دهد ($\frac{R}{3.1415} = \frac{D}{180^o}$ آن گاه $R = \frac{3.1415 * D}{180}$ که D زاویه بر حسب درجه و R زاویه بر حسب رادیان است).

۲۹. برنامه‌ای که ۱۵ جمله دنباله زیر را به کمک تابع بازگشتی نمایش می‌دهد:

اگر n == 1 F(1) = 2

وگرنه، F(n) = 2 * F(n – 1) + 1

۳۰. برنامه‌ای که حاصل عبارت $F(x) = 5x^2 - 3x + 4$ را برای مقادیر صحیح و اعشاری x با دو تابع همنام محاسبه می‌کند.

۳۱. برنامه‌ای که قاعده و ارتفاع مثلث را خوانده، با استفاده از دو تابع همنام مساحت مثلث را حساب می‌کند (مساحت مثلث برابر با قاعده ضرب در نصف ارتفاع است). ارتفاع و قاعده می‌توانند دو عدد صحیح یا دو عدد اعشاری باشند.

۳۲. برنامه‌ای که کاراکتری را خوانده، بدون استفاده از عملگر + و if آن را به حروف کوچک تبدیل می‌کند (این عمل توسط تابعی انجام می‌شود).

۳۳. برنامه‌ای که یک کاراکتر را خوانده، بدون استفاده از عملگر if (با عملگر XOR) حرف کوچک بود آن را به بزرگ تبدیل کند یا اگر حرف بزرگ بود آن را به حرف کوچک تبدیل می‌کند.

۳۴. برنامه‌ای که x و n را خوانده، حاصل تابع $F(x, n) = x^{n!}$ را نمایش می‌دهد.

۳۵. برنامه‌ای که با استفاده از تابعی طول و عرض پنجره را دریافت کرده، برفک تلویزیون را ایجاد می‌نماید.

۳۶. برنامه‌ای که x و n را خوانده، حاصل عبارت زیر را نمایش می‌دهد:

$$F(x,n) = \sum_{k=1}^{n} \frac{x^k}{k!}$$

۳۷. برنامه‌ای که اطلاعات زمین بازی جورچین را خوانده، با استفاده از تابعی اندیس خانه‌های خالی را نمایش می‌دهد و تابع دیگری شماره سطرهایی که در آن سطر خانه خالی قرار ندارد را نمایش می‌دهد.

۳۸. توابعی که اعمال تعیین شده را انجام می‌دهد. تابعی که دنباله کلیک‌های ماوس را به هم وصل می‌نماید (نمایش گرافیکی)، تابعی که تعیین می‌کند دنباله‌ای از کلیک‌های ماوس شکل بسته ایجاد می‌کند یا نه و تابعی یک نقطه را دریافت کرده و تعیین می‌نماید آیا در داخل ناحیه کلیک‌های ماوس قرار دارد یا خیر.

۳۹. برنامه‌ای که دو مقدار منطقی (یک و صفر) را از ورودی خوانده، توسط توابعی حاصل AND، OR، NAND، XOR و NOR را محاسبه کرده، نمایش می‌دهد.

۴۰. برنامه‌ای که با استفاده از یک تابع بازگشتی خروجی زیر را نمایش می‌دهد:

```
*********
*******
*****
***
*
```

۴۱. برنامه‌ای که با استفاده از تابع بازگشتی خروجی زیر را نمایش می‌دهد:

```
******
*****
****
***
```

**
*

۱۳ـ ۴. تمرین‌ها

۱. برنامه‌ای بنویسید که n جمله سری فیبوناچی را به روش بازگشتی تولید کند.

۲. برنامه‌ای بنویسید که فاصله‌ای را بر حسب فوت و اینچ دریافت کرده، معادل آن را بر حسب متر و سانتیمتر بیان کند. هر فوت ۰/۳۰۴۸ متر، یک متر ۱۰۰ سانتیمتر و هر فوت ۱۲ اینچ است. حداقل از سه تابع استفاده کنید. یکی برای ورودی، یکی برای انجام محاسبات و یکی برای خروجی.

۳. نیروی جاذبه بین دو جسم به صورت زیر محاسبه می‌گردد: $f = \frac{g * m1 * m2}{d^2}$

m1 جرم جسم اول، m2 جرم جسم دوم، d فاصله بین دو جسم و g ثابت جهانی جاذبه با مقدار $6.693*10^{-8}$ cm(g,sec) است. برنامه‌ای بنویسید که با استفاده از دو تابع مقدار نیروی جاذبه را محاسبه کند.

۴. یکی از مشکلات هواشناسی محاسبه شاخص سردی باد است. برای این منظور، فرمول زیر وجود دارد:

$$w = 33 - \frac{(10\sqrt{V} - V + 10.5)(33 - t)}{23 - 1}$$

V، سرعت باید بر حسب متر بر ثانیه، t دما بر حسب درجه سانتی‌گراد، (t <= 10) و W شاخص سردی باد است. برنامه‌ای بنویسید که با استفاده از یک تابع، سرعت باد و دما را خوانده شاخص سردی باد را نمایش دهد.

۵. برنامه‌ای بنویسید که یک عدد و یک رقم را خوانده، با استفاده از تابعی تعداد تکرار رقم در عدد را برگرداند و چاپ کند.

۶. برنامه‌ای بنویسید که x و n را خوانده، مقدار سینوس زاویه x را محاسبه کند.

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x7}{7!} + \frac{x^9}{9!} - \ldots + \frac{x^n}{n!}$$

۷. خروجی قطعه برنامه زیر چیست؟

```
#include  <iostream.h>
#include  <conio.h>
int x, y;
void  f1();
void main(void)
        {
  cout << "\n x = " << x;
  f1();
  cout << "\n y = " << ++y;
  cout << "\n x = " << x++;
 }
void f1()
```

```
{
  int y = 10;
  cout << "\n y = " << y;
  x += 10;
  cout << "\n x = " << x++;
}
```

۸. برنامه‌ای بنویسید که اعداد سه رقمی را چاپ کند که مجموع فاکتوریل ارقام آن عدد برابر با خود عدد باشد. به عنوان مثال، فرمول زیر را در نظر بگیرید:

n3n2n1 = n1! + n2! + n3!

۹. برنامه‌ای بنویسید که کلیه اعداد چهار رقمی را چاپ کند که مجموع رقم اول به توان ۱ و رقم چهارم به توان ۴ برابر با مجموع رقم دوم به توان ۲ و رقم سوم به توان ۳ باشد. به عنوان مثال، داریم.

۲۱۴۱ = $۲^۴ + ۱^۱ = ۱^۳ + ۴^۲ = ۱۶ + ۱ = ۱ + ۱۶ = ۱۷$

۱۰. برنامه‌ای بنویسید که دو عدد را خوانده، به تابعی ارسال کند. تابع، تفاضل حاصل ضرب و حاصل تقسیم آن‌ها را محاسبه کرده، به برنامه برگرداند. توابع را به صورت قالب‌های تابع پیاده‌سازی کنید و دو بار آن را برای اعداد صحیح و اعشاری فراخوانی نمایید. این تابع با استفاده از template پیاده‌سازی شود.

۱۱. برنامه‌ای بنویسید که عدد صحیح مثبتی را خوانده، مجموع ارقام آن را محاسبه کند. اگر حاصل مجموع ارقام، عدد یک رقمی نبود، این روند را ادامه داده تا نهایتاً عدد یک رقمی حاصل شود (برای انجام این کار از یک تابع بازگشتی استفاده شود).

۱۲. برنامه‌ای بنویسید که اضلاع مثلثی را خوانده، مساحت آن را به کمک تابعی محاسبه کند و به خروجی ببرد. اگر اضلاع مثلثی a، b و c باشند، مساحت آن با فرمول زیر محاسبه می‌شود:

p = (a + b + c) / 2

مساحت مثلث = $\sqrt{p(p-a)(p-b)(p-c)}$

۱۳. برنامه‌ای بنویسید که یک عدد بزرگ (حتی ۱۰۰۰ رقمی) را گرفته، تعیین می‌کند بر ۱۵ بخش پذیر است یا خیر. عددی بر ۱۵ بخش پذیر است که بر ۳ و ۵ بخش پذیر باشد. عددی بر سه بخش پذیر است که مجموع ارقام آن بر سه بخش پذیر باشد و عددی بر ۵ بخش پذیر است که رقم یکان آن صفر یا ۵ باشد (رقم یکان آن بر ۵ بخش پذیر باشد).

۱۴. برنامه‌ای بنویسید که یک رقم را از ورودی خوانده، تمام اعداد صحیح بین ۱ و ۱۰۰ را چاپ کند، به طوری که رقم دریافتی در اعداد ۱ تا ۱۰۰، مجذور و مکعب این اعداد وجود داشته باشد. به عنوان مثال، رقم (۱) را از ورودی بخوانیم، عدد ۱۳ یکی از اعدادی است که رقم یک در آن شرط صدق می‌کند، زیرا در ۱۳، ۱۶۹ و ۲۱۹۷ وجود دارد (برای تشخیص وجود رقم در عدد، مجذور و مکعب عدد از یک تابع استفاده کنید).

۱۵. برنامه‌ای بنویسید که دو عدد را از ورودی خواند به تابعی ارسال کند. تابع تفاضل حاصل ضرب و حاصل تقسیم آن‌ها را محاسبه کرده، به برنامه برگرداند و برنامه اصلی این حاصل را نمایش دهد.

۱۶. برنامه‌ای بنویسید که سه عدد را از ورودی خوانده و توسط تابعی میانگین آن‌ها را محاسبه کند و نمایش دهد.

۱۷. برنامه‌ای بنویسید که یک عدد را از ورودی خوانده و توسط یک تابع بازگشتی، شمارش معکوس از آن عدد به یک را انجام دهد (شماره‌ها را در خروجی چاپ کند).

۱۸. تابعی که دو آرگومان را می‌پذیرد و آرگومان اول را به توان آرگومان دوم می‌رساند. آرگومان اول یک مقدار double و آرگومان دوم یک مقدار صحیح مثبت یا منفی است. برنامه‌ای بنویسید که از این تابع استفاده کند.

۱۹. برنامه‌ای بنویسید که نمایش زمان را از حالت ۲۴ ساعت به ۱۲ ساعت تبدیل کند. مثلاً باید 14:25 را به صورت 2:25pm تبدیل نماید.

۲۰. قیمت سهام معمولاً به صورت کسری بیان می‌شود، مثلاً $2,\frac{7}{8}$ یا $8,\frac{1}{2}$. برنامه‌ای بنویسید که ارزش سهام را به صورت دو عدد صحیح و کسری از کاربر بگیرد. قسمت کسری را نیز به صورت دو عدد صحیح (عدد اول صورت و عدد دوم مخرج) دریافت نماید (برای تبدیل سهام از یک تابع استفاده کند که سه پارامتر را گرفته، ارزش سهام را به صورت double برگرداند).

۲۱. پارکینگی برای توقف تا سه ساعت، حداقل ۲ دلار و برای هر ساعت اضافه یا بخشی از ساعت اضافه بر ۳ ساعت ۰/۵ دلار می‌گیرد. حداکثر مبلغ قابل پرداخت برای هر دوره ۲۴ ساعتی ۱۰ دلار است. فرض کنید که هیچ خودرویی بیش از ۲۴ ساعت توقف نمی‌کند. برنامه‌ای بنویسید که مبلغ پرداختی را برای مشتریان دریافت کند و مبلغ پرداختی مشتری، مبلغ کل پرداختی را چاپ نماید. کاربر برای هر مشتری ساعات و دقیقه توقف را وارد کند. تابعی مبلغ پرداختی را محاسبه و برگرداند.

۲۲. برنامه‌ای بنویسید که با استفاده از چند تابع اعمال زیر را انجام دهد:

۱. تمام اعداد دو رقمی که رقم اول و دوم آن‌ها برابر باشند را چاپ کند.

۲. تمام اعداد سه رقمی که رقم وسط آن‌ها صفر باشد را چاپ کند.

۳. تمام اعداد سه رقمی که مجموع رقم اول و دوم کوچک‌تر از رقم سوم باشد را نمایش دهد.

۴. تمام اعداد چهار رقمی که قرینه یکدیگرند، را نمایش دهد (مثلاً ۲۳۳۲). یعنی رقم اول برابر رقم چهارم و رقم دوم برابر رقم سوم باشد.

۲۳. برنامه‌ای بنویسید که با استفاده از یک تابع بازگشتی بزرگ‌ترین مقسوم علیه مشترک دو عدد را محاسبه و چاپ کند.

۲٤. برنامه‌ای بنویسید که اولین روز سال را گرفته و تقویم سالانه را نمایش دهد.

۲۵. برنامه‌ای بنویسید که یک عدد زوج را گرفته، تمام دوتایی‌های عدد اول (دو عدد فرد اولی که مجموع آن‌ها برابر با آن عدد زوج باشد) را نمایش دهد. به عنوان مثال، اگر کاربر عدد ۲۲ را وارد کند، خروجی به صورت زیر نمایش داده شود.

```
3  +  19  = 22
5  +  17  = 22
11 +  11  = 22
```

۲٦. یونانیان باستان اعداد را به صورت هندسی دسته‌بندی می‌کردند. به عنوان مثال، آن‌ها یک عدد را مثلثی می‌نامیدند که می‌توانستند با آن تعدادی ریگ (به اندازه عدد ریگ)، در یک تقارن مثلثی بچینند. تابعی به نام isTrain که یک عدد را گرفته تشخیص می‌دهد، عدد مثلثی است یا خیر. اگر عدد مثلثی باشد، مقدار ۱، وگرنه مقدار صفر را برمی‌گرداند. دوازده نمونه عدد مثلثی در زیر آمده‌اند:

۰ ۱ ۳ ۶ ۱۰ ۱۵ ۲۱ ۲۸ ۳۶ ۴۵ ۵۵ ۶۶

برنامه‌ای بنویسید که از این تابع استفاده نماید.

۲۷. تابعی به نام isSquare که تشخیص می‌دهد، یک عدد مربعی است یا خیر. چند عدد مربعی عبارت‌اند از:

0 1 4 9 16 25 36 49 64 81 100 121 …

برنامه‌ای بنویسید که از این تابع استفاده نماید.

فصل ۵

آرایه‌ها، رشته‌ها و اشاره‌گرها

تاکنون برنامه‌هایی که نوشته شده‌اند، هر متغیر حداکثر یک مقدار را در یک لحظه نگهداری می‌کرد. یعنی، داده جدید جایگزین داده فعلی می‌شود. به عنوان مثال، دستورات زیر را در نظر بگیرید:

```
int sum = 0, num;
for(int i=1; i <= 5; i++)
{
  cout << "enter a number:"
  cin  >> num;
  sum += num;
}
cout << (float) sum / 5;
```

این دستورات ۵ عدد را خوانده، میانگین آن‌ها را نمایش می‌دهند. حال، اگر بخواهید اعدادی که بزرگ‌تر از میانگین هستند را نمایش دهید، به داده‌های خوانده شده نیاز دارید. در این برنامه داده‌های خوانده شده در num قرار می‌گیرند. بنابراین، پس از خواندن ۵ عدد، فقط پنجمین عدد در num قرار دارد. برای حل این مسئله باید داده‌های خوانده شده را نگهداری نمود. برای این منظور می‌توان به دو طریق عمل کرد:

۱. می‌توان پنج متغیر (با نام‌های متفاوت) در نظر گرفت و هر داده را در یک متغیر ذخیره کرد. این روش دو مشکل عمده دارد که عبارت‌اند از:

- اگر تعداد مقادیر زیاد شود، تعداد متغیرها نیز زیاد خواهد شد(همه این متغیرها را باید معرفی کرد).
- از حلقه تکرار نمی‌توان برای پردازش مقادیر استفاده نمود.

لذا، این روش معقول نمی‌باشد.

۲. می‌توان از ساختار داده جدیدی به نام **آرایه**[۱۲] استفاده کرد. آرایه مجموعه‌ای از عناصر است که دارای ویژگی‌های زیر باشند:

- چند خانه از حافظه که دارای **یک نام** باشند. اگر چند نام برای خانه‌های حافظه در نظر گرفته شود، همان مشکل تعریف متغیر و غیر قابل پردازش بودن توسط حلقه‌های تکرار وجود خواهد داشت.
- دارای **یک نوع** باشند و به **صورت پشت سرهم** در حافظه ذخیره شوند. چون، اگر عناصر آرایه از یک نوع نباشند و به صوت پشت سرهم ذخیره نشوند، پیمایش عناصر مشکل خواهد شد. بنابراین، عناصر آرایه باید

[12] .Array

دارای یک نوع باشند و به صورت پشت سرهم ذخیره گردند تا با داشتن آدرس شروع آرایه (همان نام آرایه)، بتوان آدرس هر خانه را به سادگی حساب کرد. این عمل به صورت زیر انجام می‌شود:

آدرس خانه‌iام = آدرس شروع آرایه + i * sizeof (نوع آرایه) ;

به عنوان مثال، اگر آدرس شروع آرایه a از نوع int، ۱۰۰۰باشد، آدرس شروع خانه سوم به صورت زیر محاسبه می‌گردد:

آدرس شروع خانه سوم = 1000 + 3 * sizeof (int) = 1000 + 6 =1006

برای دسترسی به عناصر آرایه از **اندیس**[۱۳] (شماره خانه) استفاده می‌شود. به همین دلیل نام دیگر آرایه‌ها، **متغیرهای اندیس‌دار** است. آرایه‌ها می‌توانند با توجه به تعداد اندیس آن‌ها چند نوع باشند که عبارت‌اند از:

۱. **آرایه‌های یک بعدی**، دارای یک اندیس هستند.

۲. **آرایه‌های دو بعدی**، دارای دو اندیس هستند.

۳. **آرایه‌های چند بعدی**، دارای چند اندیس هستند.

در این کتاب آرایه‌های یک بعدی و دو بعدی را می‌آموزیم.

۱ ـ ۵. آرایه‌های یک بعدی

همان‌طور که بیان گردید، آرایه‌های یک بعدی، یک اندیس دارند. برای استفاده از آرایه‌ها باید دو عمل زیر انجام شود:

۱. **تعریف آرایه**، آرایه‌ها نیز مانند متغیرهای معمولی باید قبل از استفاده تعریف گردند. تعریف آرایه به صورت زیر می‌باشد:

نوع آرایه نام آرایه [تعداد عناصر] ;

نوع آرایه، نوع داده‌ای را تعیین می‌کند که در خانه‌های آرایه قرار می‌گیرد و یکی از انواع شناخته شده در C++ یا تعریف شده توسط کاربر می‌باشد، نام آرایه، از قانون نام‌گذاری شناسه‌ها پیروی می‌کند و **تعداد عناصر**، تعداد خانه‌های آرایه را تعیین می‌کند. به عنوان مثال، دستور زیر را در نظر بگیرید:

```
float   a[5];
```

این دستور، آرایه‌ای به نام a با ۵ خانه تعریف می‌کند که خانه‌های آن داده‌های عددی اعشاری هستند.

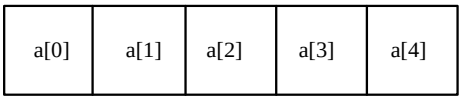
۲. **دسترسی به عناصر آرایه**، همان‌طور که بیان گردید، برای دسترسی به عناصر آرایه از اندیس آن به صورت زیر استفاده می‌شود:

[اندیس] نام آرایه

اندیس، شماره خانه آرایه را تعیین می‌کند. اندیس آرایه در C++ از صفر شروع می‌شود. بنابراین، حداکثر مقداری که اندیس می‌تواند بپذیرد، برابر با **[۱- تعداد عناصر آرایه]** است. یعنی، آرایه‌ای با ۵ عنصر، به صورت زیر نمایش داده می‌شود:

---

[13] .Index



همان‌طور که در شکل می‌بینید، آخرین خانه آرایه دارای اندیس ۴ است. اکنون دستورات زیر را ببینید:

```
cin  >> a[2];
cout << a[2];
```

دستور اول، عنصر سوم آرایه (a[2]) را می‌خواند و دستور دوم، مقدار عنصر سوم آرایه (a[2]) را نمایش می‌دهد.

۲- ۵. مقداردهی به عناصر آرایه

به سه روش می‌توان به عناصر آرایه مقدار داد. این روش‌ها عبارت‌اند از:

۱ ـ ۲ ـ ۵. مقداردهی به عناصر آرایه به صورت خانه‌های مجزا

در این روش به خانه‌های آرایه به صورت مجزا و مستقل مقدار تخصیص می‌یابد. به عنوان مثال، دستورات زیر را ببینید:

```
int a[4];
a[0] = 10; a[1] = 12; a[2] = 3; a[3] = 15;
```

این دستورات، آرایه a را با ۴ عنصر تعریف کرده، مقادیر ۱۰، ۱۲، ۳ و ۱۵ را به ترتیب در خانه‌های ۰، ۱، ۲ و ۳ آن قرار می‌دهند (شکل زیر):

a[0]	a[1]	a[2]	a[3]
10	12	3	15

۲ ـ ۲ ـ ۵. مقداردهی اولیه به آرایه در هنگام تعریف آن

اگر به آرایه‌ای در هنگام تعریف مقدار اولیه تخصیص ندهید، دو حالت اتفاق می‌افتد:

۱. آرایه قبل از تابع اصلی ()main سراسری تعریف شود. در این صورت آرایه به صورت سراسری تعریف می‌گردد، مقادیر عناصر آن صفر در نظر گرفته می‌شود. به عنوان مثال، اگر دستور زیر قبل از تابع ()main قرار گیرد، مقادیر عناصر آن صفر خواهند شد:

	a[0]	a[1]	a[2]	a[3]
int a[4]; →	0	0	0	0

۲. آرایه در داخل یکی از توابع به صورت محلی تعریف شود. در این صورت مقدار اولیه خانه‌های آن تعریف نشده است. به عنوان مثال، اگر با دستور زیر آرایه a در یکی از توابع تعریف شود، مقادیر عناصر آن نامشخص است:

	a[0]	a[1]	a[2]	a[3]
int a[4]; →	?	?	?	?

در هنگام تعریف آرایه می‌توان به عناصر آن مقدار اولیه تخصیص داد. این عمل به صورت زیر انجام می‌شود:

نوع آرایه نام آرایه[تعداد عناصر] = { مقادیر آرایه} ;

در هنگام تخصیص مقادیر به آرایه باید به نکات زیر دقت کرد:

۱. اگر مقادیر بیش از یکی باشد، با کاما از یکدیگر جدا می‌شوند. به عنوان مثال، دستور زیر را ببینید:

```
int   a[5] = {1, 12, 5, 7, 13};
```

این دستور آرایه a با ۵ عنصر را تعریف کرده، مقادیر ۱، ۱۲، ۵، ۷ و ۱۳ را به عناصر آن تخصیص می‌دهد (شکل زیر):

a[0]	a[1]	a[2]	a[3]	a[4]
1	12	5	7	13

۲. اگر تعداد مقادیری که به آرایه تخصیص می‌دهید، از تعداد عناصر آرایه کمتر باشد، بقیه عناصر صفر منظور خواهند شد. به عنوان مثال، دستور زیر را ببینید:

```
int   a[ 5] = {1, 5, 12};
```

این دستور آرایه ۵ عنصری به نام a را تعریف کرده، مقادیر ۱، ۵، ۱۲ و صفر را به عناصر آن تخصیص می‌دهد(شکل زیر):

a[0]	a[1]	a[2]	a[3]	a[4]
1	5	12	0	0

۳. اگر در هنگام تعریف آرایه تعداد عناصر آن مشخص نگردد، طول آرایه برابر با تعداد عناصری که به آن تخصیص می‌یابد، خواهد شد. به عنوان مثال ، دستور زیر را ببینید:

```
int   a[] = {1, 7, 13, 14};
```

این دستور آرایه‌ای به نام a با ۴ عنصر را تعریف کرده، مقادیر ۱، ۷، ۱۳ و ۱۴ را به عناصر آن تخصیص می‌دهد(شکل زیر):

a[0]	a[1]	a[2]	a[3]
1	7	13	14

۳ ـ ۲ ـ۵. مقداردهی به عناصر آرایه با حلقه تکرار و شیء cin

مقادیر عناصر آرایه را می‌توان با حلقه‌های تکرار نظیر for، while و شیء cin تعیین کرد. در این صورت باید یک حلقه for ایجاد نمود که بدنه آن شامل دستور یا دستوراتی برای تخصیص مقادیر به عناصر آرایه باشد. به عنوان مثال، دستورات زیر را در نظر بگیرید:

```
int a[4];
for(int i= 0; i <4; i++)
{
cout << "Enter a number:";
cin >> a[i];
}
```

این دستورات چهار مقدار را خوانده و در خانه‌های تعریف شده، دستور اول قرار می‌دهند (در

آرایه قرار می‌دهند).

۳ ـ ۵. نمایش عناصر آرایه

برای نمایش عناصر آرایه دو روش وجود دارد که عبارت‌اند از:

۱ ـ ۳ ـ ۵. نمایش مقادیر هر عنصر به صورت مجزا

در این روش، با شیء cout می‌توان مقادیر عناصر آرایه را نمایش داد. به عنوان مثال، دستورات زیر را ببینید:

```
int   a[] = {10, 5, 7};
cout << a[2] << a[1] << a[0];
```

دستور اول، آرایه‌ای با سه عنصر تعریف کرده، مقادیر ۱۰، ۵ و ۷ را به ترتیب به خانه‌های ۰، ۱ و ۲ آن تخصیص می‌دهد و دستور دوم، عناصر آرایه را به صورت معکوس (از آخرین به اولین خانه) نمایش می‌دهد.

۲ ـ ۳ ـ ۵. نمایش عناصر آرایه با حلقه تکرار

عناصر آرایه را با حلقه‌های تکرار نظیر for، while و شیء cout می‌توان نمایش داد. به عنوان مثال، دستورات زیر را مشاهده کنید:

```
int   a[5] = {10, 5, 7, 8};
 for(int i = 0; i< 5; i ++)
      cout << "\t" << a[i];
```

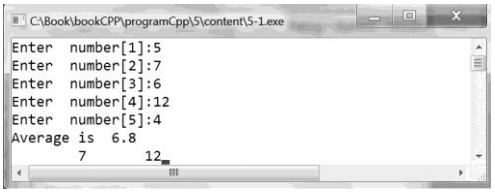
دستور اول، آرایه‌ای با ۵ عنصر به نام a را تعریف کرده، مقادیر ۱۰، ۵، ۷، ۸ و ۰ را به خانه‌های ۰، ۱، ۲، ۳ و ۴ آن نسبت می‌دهد و دستور دوم، با استفاده از حلقه for و شیء cout مقادیر خانه‌های آرایه را به صورت زیر نمایش می‌دهد:

10 5 7 8 0

مثال ۱ ـ ۵. برنامه‌ای که ۵ عدد را خوانده، اعدادی که بزرگ‌تر از میانگین ۵ عدد باشند را نمایش می‌دهد (هدف این برنامه، آشنایی با چگونگی تعریف آرایه، مقداردهی، پیمایش و نمایش عناصر آن می‌باشد).

```
#include<iostream.h>
   #include<conio.h>
    const int n = 5;
         void main()
                {
             float a[n];
for(int i = 0; i < n; i++)
                {
cout <<"Enter number[" << i+1 << "]:";
             cin >> a[i];
                }
           float sum = 0;
for (int i = 0; i < n; i++)   sum += a[i];
          float aver = sum / n;
 cout   << "Average is  " << aver << endl;
          for(int j = 0; j < n; j++)
   if (a[j]> aver)  cout << "\t" << a[j];
```

```
getch();
}
```



توضیح: این برنامه، ابتدا آرایه‌ای با ۵ عنصر را تعریف کرده، با استفاده از یک حلقه تکرار، ۵ عدد را می‌خواند و در خانه‌های آن قرار می‌دهد. سپس، با استفاده از حلقه for دوم، مجموع عناصر آرایه را محاسبه نموده، در متغیر sum قرار می‌دهد و در پایان، با استفاده از حلقه for سوم عناصر آرایه که بیشتر از میانگین (avg) مقادیر آرایه هستند، را نمایش می‌دهد.

۴ – ۵. تولید اعداد تصادفی

برای تولید اعداد تصادفی دو تابع وجود دارند. این دو تابع عبارت‌اند از:

۱. **تابع ()rand**، یک عدد تصادفی بین 0 تا ۳۲۷۶۷ را تولید کرده، برمی‌گرداند. برای تولید اعداد تصادفی توسط این تابع باید دستور زیر را قبل از اجرای تابع ()rand قرار داد:

```
randomize;
```

۲. **تابع ()random**، برای تولید عدد تصادفی بین 0 تا n – 1 به کار می‌رود و به صورت زیر استفاده می‌شود:

```
random(n);
```

این دو تابع در فایل سرآیند stdlib.h قرار دارند. بنابراین، قبل از استفاده از این توابع باید فایل stdlib.h را به صورت زیر به برنامه اضافه کرد:

```
#include <stdlib.h>
```

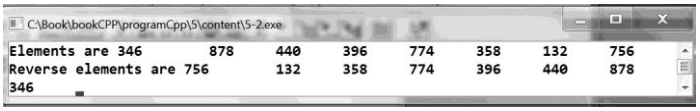
مثال ۲ – ۵. برنامه‌ای که ۸ عدد تصادفی بین ۰ تا ۲۰۰۰ را تولید کرده، در آرایه‌ای قرار می‌دهد و عناصر آرایه را از آخرین عنصر به اولین عنصر نمایش می‌دهد.

این برنامه یک تابع برای تولید اعداد تصادفی و قرار دادن آن در عناصر آرایه دارد (تابع ()createRand)، یک تابع برای نمایش عناصر آرایه دارد (تابع ()printArray) و تابع دیگری برای نمایش عناصر آرایه از آخرین عنصر به اولین عنصر دارد (تابع ()printReverse).

متغیر	هدف
پارامتر a	آرایه‌ای با n عنصر (در تمام توابع)
پارامتر n	تعداد عناصر (در تمام توابع)
i	شمارنده حلقه

```
#include<iostream.h>
#include<conio.h>
#include<stdlib.h>
void createRand(int a[], int n)
{
for(int i = 0; i < n; i++)  a[i] = random(2000);
}void printArray(int a[], int n)
{
for(int i = 0; i < n; i++)  cout << a[i] << "\t";
}void printReverse(int a[], int n)
```

```
{
for(int i = n - 1; i >= 0; i--)  cout << a[i] << "\t";
}void main()
{
int a[8], n =8;
createRand(a, n);
cout << "Elements are " ;
printArray(a, n);
cout << "Reverse elements are " ;
printReverse(a, n);
getch();
}
```



تمام توابع دو پارامتر را دریافت کرده، اولین پارامتر، آرایه مورد نظر است و به صورت int a[] تعریف شده است و پارامتر دوم تعداد عناصر آرایه را تعیین می‌کند. در هنگام تعریف آرایه به صورت پارامتر مشاهده می‌شود که تعداد عناصر آرایه می‌تواند در پارامتر تعیین شود یا ذکر نشود.

۵ – ۵. مرتب سازی حبابی

آرایه دارای n عنصر است و بخواهیم عناصر آرایه را مرتب کنیم. در این روش برای مرتب کردن عناصر آرایه، عنصر اول را با عنصر دوم مقایسه می‌کنیم، در صورتی که اولی بزرگ‌تر از دومی باشد، جایشان را عوض می‌کنیم. عنصر اول را با عنصر سوم مقایسه می‌کنیم، اگر اولی بزرگ‌تر باشد، جایشان را عوض می‌کنیم و این روند را تا عنصر آخر ادامه می‌دهیم. اکنون، کوچک‌ترین عنصر در ابتدای آرایه قرار می‌گیرد. در ادامه عنصر دوم را با عنصر سوم مقایسه می‌نماییم، اگر عنصر دوم بزرگ‌تر از عنصر سوم باشد، جایشان را عوض می‌کنیم، عنصر دوم را با عنصر چهارم، عنصر پنجم، ... و عنصر آخر مقایسه می‌کنیم، اگر عنصر دوم بزرگ‌تر از هر یک از عناصر باشد، جایشان را تعویض خواهیم کرد. اکنون، عناصر اول و دوم مرتب شده‌اند. این روند را برای مرتب‌سازی بقیه عناصر ادامه خواهیم داد. به عنوان مثال، مقادیر زیر را در نظر بگیرید:

5 2 4 3 1

برای مرتب‌سازی این عناصر گام‌های زیر انجام می‌شوند:

گام ۱ – ۱: عنصر اول (۵) با عنصر دوم (۲) مقایسه می‌گردد. چون عنصر اول، بزرگ‌تر از عنصر دوم است، جایشان تعویض می‌شود:

2 5 4 3 1

گام ۲ – ۱. عنصر اول (۲) با عنصر سوم (۴) مقایسه می‌شود. چون عنصر اول کوچک‌تر از عنصر سوم است، جایشان تعویض نخواهد شد:

2 5 4 3 1

گام ۳ – ۱. عنصر اول (۲) با عنصر چهارم (۳) مقایسه می‌گردد و جایشان تعویض نمی‌شود.

2 5 4 3 1

گام ٤ – ۱. عنصر اول (۲) با عنصر پنجم (۱) مقایسه می‌گردد. چون عنصر اول بزرگ‌تر از عنصر پنجم است، جایشان عوض می‌شود:

1 5 4 3 2

اکنون، کوچک‌ترین عنصر به ابتدای لیست انتقال یافته است.

گام ۱ – ۲: عنصر دوم (۵) با عنصر سوم (۴) مقایسه می‌شود و جایشان عوض می‌شود:

1 4 5 3 2

گام ۲ – ۲: عنصر دوم (۴) با عنصر چهارم (۳) مقایسه می‌گردد و جایشان عوض می‌شود:

1 3 5 4 2

گام ۳ – ۲: عنصر دوم (۳) با عنصر پنجم (۲) مقایسه می‌گردد و جایشان عوض می‌شود:

1 2 5 4 3

اکنون دو عدد اول لیست (۲ و ۱) مرتب گردیدند.

گام ۱ – ۳: عنصر سوم (۵) با عنصر چهارم(۴) مقایسه می‌گردد و جایشان تعویض می‌گردد:

1 2 4 5 3

گام ۲ – ۳: عنصر سوم (۴) با عنصر پنجم (۳) مقایسه می‌گردد و جایشان عوض می‌شود:

1 2 3 5 4

اکنون سه عدد اول لیست (۳ و ۲ و ۱) مرتب شده‌اند.

گام ۱ – ٤: عنصر چهارم (۵) با عنصر پنجم (۴) مقایسه می‌گردد و جایشان عوض می‌شود:

1 2 3 4 5

اکنون به لیست دقت کنید، مقادیر آن مرتب شده‌اند.

۶ – ۵. جست‌وجوی مقدار در آرایه

یکی از مباحثی که در کامپیوتر بسیار مهم است، جست‌وجوی مقدار خاصی در داده‌های ذخیره شده آرایه است. برای این منظور، دو روش جست‌وجو وجود دارد که عبارت‌اند از:

۱. جست‌وجوی خطی (ترتیبی)
۲. جست‌وجوی دودویی

مثال ۵ – ۵. برنامه‌ای که اعمال زیر را انجام می‌دهد:

۱. اطلاعاتی را خوانده، در یک ماتریس ۳ × ۳ قرار می‌دهد.
۲. اطلاعات ماتریس را نمایش می‌دهد.
۳. اطلاعات قطر اصلی ماتریس را نمایش می‌دهد.
۴. اطلاعات قطر فرعی ماتریس را نمایش می‌دهد.
۵. اطلاعات بالای قطر اصلی ماتریس را نمایش می‌دهد.
۶. اطلاعات پایین قطر اصلی ماتریس را نمایش می‌دهد.

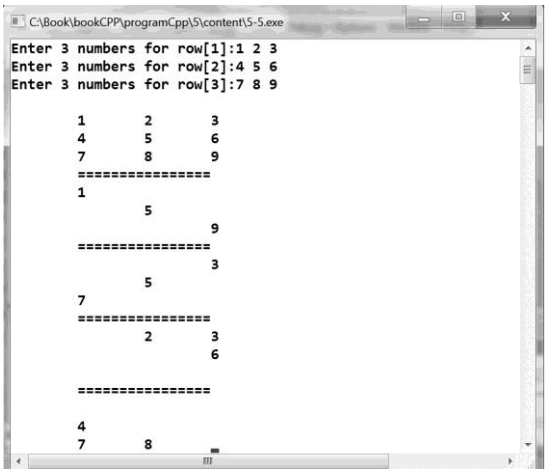
(همه این کارها با استفاده از توابع انجام می‌گردند).

توابع	متغیر	هدف
تمام توابع	پارامتر arr	آرایه دوبعدی n × m (در اینجا ۳ × ۳ است)
	پارامتر n	تعداد سطرهای آرایه arr
	پارامتر m	تعداد ستون‌های آرایه arr

	i	شمارنده سطرهای آرایه
	j	شمارنده ستون‌های آرایه
تابع main	a	آرایه ۳ × ۳
	n	تعداد سطرهای آرایه برابر ۳ است
	m	تعداد ستون‌های آرایه برابر ۳ است

```
#include<iostream.h>
#include<conio.h>
void readArray(int arr[][3], int n,int m)
{
 for(int i = 0; i < n; i++)
 {
       cout << "Enter"<< m <<" numbers for row["<< (i+1)<<"]:";
      for (int j=0; j < m; j ++)   cin >> arr[i][j];
 }
}
void printArray(int arr[][3], int n, int m)
{
 for(int i = 0; i < n; i++)
 {
  for(int j = 0; j < m; j++)  cout << "\t" <<  arr[i][j];
     cout << endl;
  }
}
void printArray1(int arr[][3], int n, int m)
{
 for(int i = 0; i < n; i++)
 {
  cout << endl;
  for(int j = 0; j < m; j++)
     if ( i == j ) cout << "\t" <<  arr[i][j];
     else cout << "\t";
  }
}
void printArray2(int arr[][3], int n, int m)
{
 for(int i = 0; i < n; i++)
 {
  cout << endl;
  for(int j = 0; j < m; j++)
     if ( (i + j )== (m-1) ) cout << "\t" <<  arr[i][j];
     else cout << "\t";
  }
}
void printArray3(int arr[][3], int n, int m)
{
 for(int i = 0; i < n; i++)
 {
  cout << endl;
  for(int j = 0; j < m; j++)
     if ( i < j  ) cout << "\t" <<  arr[i][j];
     else cout << "\t";
  }
}
void printArray4(int arr[][3], int n, int m)
{
 for(int i = 0; i < n; i++)
 {
  cout << endl;
  for(int j = 0; j < m; j++)
     if ( i > j ) cout << "\t" <<  arr[i][j];
     else cout << "\t";
  }
}
void main()
{
```

```
    int a[3][3];
    int n = 3 , m =3 ;
    readArray(a, n ,m );
    printArray(a, n, m);
    cout <<"\n\t================";
    printArray1(a, n, m);
    cout <<"\n\t================";
    printArray2(a, n, m);
    cout <<"\n\t================";
    printArray3(a, n, m);
    cout <<"\n\t================";
    printArray4(a, n, m);
    getch();
}
```



عملکرد توابع

☒ **تابع ()main**، آرایه ۳ × ۳ به نام a را تعریف می‌کند، تعداد سطرها و ستون‌های آن را به ترتیب متغیرهای n و m در نظر گرفته، آن‌ها را تعریف کرده، مقدار اولیه ۳ را به آن‌ها تخصیص می‌دهد. با فراخوانی تابع readArray()، ۹ عدد را خوانده، در آرایه ۳ × ۳ قرار می‌دهد. سپس، با فراخوانی تابع printArray()، اطلاعات آرایه را نمایش می‌دهد. با فراخوانی تابع printArray1()، عناصر قطر اصلی آرایه را نمایش می‌دهد. با فراخوانی تابع printArray2() عناصر قطر فرعی آرایه را نمایش خواهد داد. با فراخوانی printArray3()، عناصر زیر قطر اصلی و با فراخوانی تابع printArray4() عناصر بالای قطر اصلی را نمایش می‌دهد.

☒ **تابع readArray()**، پارامترهای arr (آرایه دوبعدی)، n (تعداد سطرها) و m (تعداد ستون‌ها) را دریافت می‌کند. یک حلقه تکرار با شمارنده i (حلقه تکرار خارجی) از 0 تا n – 1 (تعداد سطرها) ایجاد می‌کند و در داخل این حلقه تکرار یک پیغام را نمایش می‌دهد که به اندازه تعداد ستون‌ها از کاربر برای هر سطر داده درخواست می‌کند. سپس، یک حلقه تکرار با شمارنده j (حلقه تکرار داخلی) ایجاد می‌کند تا داده‌های وارد شده را از کاربر دریافت کند و در خانه‌های مورد نظر (arr [i][j]) قرار دهد.

☒ **تابع printArray()**، پارامترهای arr (آرایه)، n (تعداد سطرها) و m (تعداد ستون‌ها) را دریافت کرده، اطلاعات آرایه n × m را نمایش می‌دهد. برای این منظور، از حلقه تودرتو استفاده می‌کند. حلقه خارجی که شمارنده i آن از 0 تا n-1 تغییر می‌کند و در داخل این حلقه ابتدا به خط بعد می‌رود و عناصر آن سطر (سطر

نام) را با استفاده از یک حلقه تکرار که شمارنده j آن از 0 تا m – 1 تغییر می‌کند، عنصر arr [i][j] را نمایش می‌دهد (i از 0 تا n – 1 و j از 0 تا m – 1 تغییر می‌کند).

☒ **تابع printArray1()**، عناصر قطر اصلی ماتریس را نمایش می‌دهد (مانند تابع printArray() است با این تفاوت که اگر i برابر j باشد، عنصر arr [i][j] را نمایش می‌دهد، وگرنه جای خالی را نمایش خواهد داد).

☒ **تابع printArray2()**، عناصر قطر فرعی ماتریس را نمایش می‌دهد (مانند تابع printArray1() عمل می‌کند با این تفاوت که اگر i + j برابر m – 1 باشد، مقادیر خانه arr [i][j] را نمایش می‌دهد).

☒ **تابع printArray3()**، عناصر بالای قطر اصلی ماتریس را نمایش می‌دهد (مانند تابع printArray1() عمل می‌کند با این تفاوت که اگر i کوچک‌تر از j باشد، مقدار خانه arr [j][i] را نمایش می‌دهد).

☒ **تابع printArray4()**، عناصر زیر قطر اصلی ماتریس را نمایش می‌دهد (مانند تابع printArray() عمل می‌کند با این تفاوت که اگر i بزرگ‌تر از j باشد، مقدار خانه arr [i][j] را نمایش می‌دهد).

مثال ۶ – ۵. برنامه‌ای که یک آرایه ۳ × ۳ تعریف کرده، آن را مقداردهی اولیه می‌کند و سپس، توسط دو تابع مجموع عناصر آرایه (تابع sumElement()) و حاصل ضرب عناصر قطر اصلی (تابع multiply()) را محاسبه کرده، نمایش می‌دهد).

نام تابع	نام متغیر	هدف
تمام توابع	پارامتر arr	آرایه مورد نظر به عنوان پارامتر
	پارامتر n	تعداد سطرهای آرایه arr
	پارامتر m	تعداد ستون‌های آرایه arr
	i، j	شمارنده‌های سطرها و ستون‌های آرایه
mian	a	آرایه‌ی که باید مجموع عناصر و حاصل ضرب عناصر قطر اصلی آن نمایش داده شود.
sumElement	sum	مجموع عناصر آرایه
multiply	mul	حاصل ضرب عناصر قطر اصلی آرایه

```
#include<iostream.h>
#include<conio.h>
long sumElement(int [][3], int, int);
long multiply(int [][3],int , int);
void printArray(int [][3],int, int);
void main()
{
 int a[3][3]={{11, 22, 33}, {44, 55, 66}, {77, 88, 99}};
 printArray(a, 3, 3);
 cout << "\nSum is " << sumElement( a, 3, 3);
 cout << "\nMultiply is " << multiply( a, 3, 3);
 getch();
}
void printArray(int arr[][3],int n, int m)
{
 for(int i =0; i < n; i ++)
 {
  cout << endl;
  for(int j =0; j < m; j ++) cout << "\t" << arr[i][j];
 }
```

```
}
long sumElement(int arr[][3],int n, int m)
{
 long sum =0;
 for(int i =0; i < n; i ++)
  for(int j =0; j < m; j ++) sum += arr[i][j];
 return sum;
}
long multiply(int arr[][3],int n, int m)
{
 long mul =1;
 for(int i =0; i < n; i ++)
  for(int j =0; j < m; j ++)
    if (i == j) mul *=  arr[i][j];
 return mul;
}
```

عملکرد توابع

☒ **تابع main()**، آرایه‌ای به نام a با سه سطر و سه ستون تعریف می‌کند، مقدار اولیه به عناصر آن تخصیص می‌دهد. سپس، با فراخوانی تابع printArray()، مقادیر آرایه را نمایش می‌دهد. در ادامه، با نمایش پیغام مناسب و فراخوانی تابع sumElement()، مجموع عناصر آرایه را نمایش می‌دهد و در پایان، با یک پیغام مناسب و فراخوانی تابع multiply()، حاصل ضرب عناصر قطر اصلی آرایه را نمایش می‌دهد.

☒ **تابع sumElement()**، آرایه (arr)، تعداد سطرها (n) و تعداد ستون‌های (m) را به عنوان پارامتر دریافت کرده، ابتدا مقدار صفر را در مجموع عناصر آرایه (sum) قرار می‌دهد و با استفاده از حلقه تودرتو مجموع عناصر آرایه را حساب کرده و برمی‌گرداند.

☒ **تابع multiply()**، آرایه (arr)، تعداد سطرها (n) و تعداد ستون‌های (m) را به عنوان پارامتر دریافت کرده، ابتدا مقدار یک (1) را در حاصل ضرب عناصر قطر اصلی (mul) قرار می‌دهد و با استفاده از حلقه تودرتو حاصل‌ضرب عناصر قطر اصلی (اگر i برابر j باشد، arr[i][j] روی قطر اصلی است) را در mul قرار می‌دهد و mul را برمی‌گرداند.

۱۱ ـ ۵. رشته‌ها

رشته‌ها، مجموعه‌ای از کاراکترها هستند که به صورت پشت سرهم ذخیره می‌گردند و انتهای آن‌ها با '\0' مشخص می‌شود. در زبان C و C++، نوع رشته‌ای وجود ندارد. اما، می‌توان آن را به صورت آرایه‌ای از کاراکترها پیاده‌سازی نمود. بنابراین، پیاده‌سازی رشته به صورت زیر انجام می‌شود:

char نام رشته [طول رشته];

نام رشته از قانون نام‌گذاری متغیرها پیروی می‌کند و طول رشته تعداد کاراکترهای رشته را تعیین می‌نماید به عنوان مثال، دستورات زیر را ببینید:

```
char   fname[ 16];
char   address[ 256];
```

دستور اول، رشته‌ای به نام fname تعریف می‌کند که حداکثر ۱۵ کاراکتر را می‌توان در آن ذخیره کرد و دستور دوم، رشته‌ای به نام address تعریف کرده که حداکثر ۲۵۵ کاراکتر را می‌تواند نگهداری کند (چون انتهای رشته کاراکتر '\0' قرار می‌گیرد، بنابراین، یک کاراکتر کمتر ذخیره می‌کند).

۱ ـ ۱۱ ـ ۵. مقداردهی به رشته‌ها

پس از تعریف رشته‌ها باید بتوان مقدار را به آن‌ها نسبت داد. سه روش برای مقداردهی به رشته‌ها وجود دارد که عبارت‌اند از:

۱. مقداردهی اولیه به رشته
۲. مقداردهی رشته از طریق دستورات ورودی
۳. مقداردهی به رشته از طریق تابع strcpy

مقداردهی اولیه به رشته‌ها

به سه روش می‌توان به رشته‌ها مقدار اولیه تخصیص داد. این روش‌ها عبارت‌اند از:

☒ **مقداردهی به رشته‌ها به صورت کاراکتر به کاراکتر**، این روش به صورت زیر انجام می‌شود:

char نام رشته [طول] = {مقادیر کاراکتری};

به عنوان مثال، دستورات زیر را ببینید:

```
char lang[20] = {'C', '+', '+', ' ', 'L', 'a', 'n', 'g', 'u', 'a', 'g', 'e', '\0'};
```

این دستور، رشته lang را به صورت زیر مقداردهی می‌کند:

lage	C	+	+		L	a	n	g	u	a	g	e	\0	?	?	?	?	?	?	?
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19

همان‌طور که در این تعریف و خروجی می‌بینید، انتهای رشته باید کاراکتر '\0' قرار گیرد.

☒ **مقداردهی اولیه به رشته به صورت یکجا**، این روش مقدار پیوسته را در رشته قرار می‌دهد:

char نام رشته [طول] = "مقدار رشته‌ای";

به عنوان مثال، دستور زیر را ببینید:

```
char url[20] = "fanavarienovin.net";
```

این دستور متغیر url را به صورت رشته‌ای تعریف کرده، مقدار آن را به صورت زیر تخصیص می‌دهد:

url	f	a	n	a	v	a	r	i	e	n	o	v	i	n	.	n	e	t	\0	?
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19

☒ **مقداردهی اولیه به رشته به صورت پویا**، در این روش در هنگام تعریف رشته، تعداد بایت‌های (طول) آن تعیین نمی‌شود (تعداد بایت‌هایی که به رشته تخصیص می‌یابد، به عنوان طول رشته در نظر گرفته می‌شود). این روش تخصیص مقدار به رشته به صورت‌های زیر انجام می‌شود:

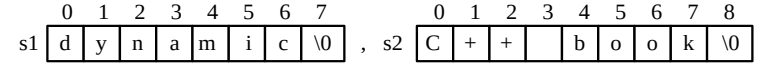
char نام رشته [ ] = {مقادیر};

char نام رشته [ ] = "مقدار رشته‌ای";

به عنوان مثال، دستورات زیر را در نظر بگیرید:

```
char s1 [ ] = {'d', 'y', 'n', 'a', 'm', 'i', 'c', '\0'};
char s2 [ ] = "C++  book";
```

این دستورات رشته‌های s1 و s2 را تعریف کرده، مقادیر آن‌ها را به صورت زیر تخصیص می‌دهند:

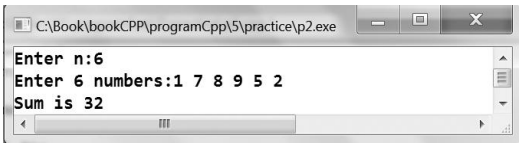


مثال ۲. برنامه‌ای که n عدد از نوع float را دریافت کرده، اعداد خوانده شده را در اشاره‌گری از نوع float قرار می‌دهد و مجموع اعداد خوانده شده را نمایش می‌دهد (برنامه تابعی، برای ورود و تابعی برای محاسبه مجموع دارد. در این توابع، آرایه به صورت اشاره‌گر تعریف شده است).

نام تابع	متغیر	هدف
main	اشاره‌گر p*	اشاره‌گری است که به آرایه پویا اشاره می‌کند
	n	تعداد عناصر آرایه به صورت پویا تعیین می‌گردد.
	s	مجموع عناصر آرایه
توابع readArray و sum	پارامتر p*	اشاره‌گری است که به ابتدای آرایه اشاره می‌کند.
	پارامتر n	تعداد عناصر آرایه
	i	شمارنده‌ای برای پیمایش آرایه
sum	s	مجموع عناصر آرایه

```
#include <iostream.h>
#include <conio.h>
void readArray(float *p, int n)
{
 cout << "Enter " << n << " numbers:";
 for (int i =0; i < n; i++)
   cin >> *(p+i);
}
float sum(float *p,int n)
{
 float s = 0;
 for (int i =0; i < n; i++ )
   s += p[i];
 return s;
}
void main()
{
 float *p;
 int n;
 cout <<"Enter n:";
 cin >> n;
 p = new float[n];
 readArray(p, n);
 float s = sum(p, n);
 cout << "Sum is " << s << endl;
 getch();
```

```
}
```



عملکرد توابع

☒ **تابع ()main**، اشاره‌گر p و متغیر n را تعریف کرده، با یک پیغام n را از ورودی دریافت می‌کند. سپس، n عنصر از نوع float به صورت پویا از حافظه اخذ می‌کند و آدرس شروع آن‌ها را در p قرار می‌دهد. در ادامه با فراخوانی تابع ()readArray، مقدار عناصر آرایه را خوانده، در p قرار می‌دهد و با فراخوانی تابع ()sum، مجموع عناصر آرایه را در s قرار می‌دهد. در پایان، مقدار s را نمایش می‌دهد.

☒ **تابع ()readArray**، اشاره‌گر p (آدرس شروع آرایه) و مقدار n (تعداد عناصر آرایه) را به عنوان پارامتر دریافت کرده، n مقدار را با استفاده از یک حلقه تکرار می‌خواند و در آرایه‌ی که p به آدرس شروع آن اشاره می‌کند، قرار می‌دهد.

☒ **تابع ()sum**، اشاره‌گر p و تعداد عناصر آرایه n را به عنوان پارامتر دریافت کرده، مجموع عناصر آرایه را محاسبه کرده، برمی‌گرداند.

مثال ۳. برنامه‌ای که اعمال زیر را انجام می‌دهد:

۱. تابعی به نام lower دارد که کلیه حروف بزرگ رشته را به حروف کوچک تبدیل می‌کند.

۲. تابعی به نام upper دارد که کلیه حروف کوچک رشته را به حروف بزرگ تبدیل می‌کند.

۳. تابعی به نام strLen دارد که طول رشته را برمی‌گرداند.

۴. تابعی به نام strCpy دارد که رشته‌ای را در رشته دیگر کپی می‌کند.

۵. تابعی به نام strCmp دارد که دو رشته را با هم مقایسه می‌کند.

۶. تابعی به نام isDigit دارد که تعیین می‌کند آیا کلیه کاراکترهای رشته رقم است یا خیر.

۷. تابعی به نام isAlpha دارد که تعیین می‌کند آیا کلیه کاراکترها رشته حروف الفبا هستند یا خیر.

۸. تابعی به نام right دارد که چند کاراکتر سمت راست رشته را برمی‌گرداند.

۹. تابعی به نام left دارد که چند کاراکتر ابتدای رشته را برمی‌گرداند.

۱۰. تابعی به نام substr دارد که بخشی از رشته را از نقطه خاص به طول n کاراکتر برمی‌گرداند.

۱۱. تابعی به نام convertToNnumber دارد که یک رشته عددی را به عدد صحیح تبدیل می‌کند و برمی‌گرداند.

۱۲. تابعی به نام strCat دارد که رشته‌ای را به انتهای رشته دیگر اضافه می‌کند.

۱۳. تابعی به نام insert دارد که رشته‌ای را در درون یا انتهای رشته دیگر درج می‌کند.

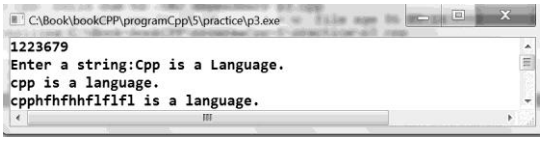
هدف این برنامه آشنایی با رشته‌ها، اشاره‌گرها و ترکیب رشته‌ها و اشاره‌گرها می‌باشد.

```
#include<iostream.h>
#include<conio.h>
```

```
void lower(char *s)
{
  while (*s)
  {
    if (*s >= 'A' && *s <= 'Z')   *s +=32;
    s++;
  }
}
void upper(char s[])
{
  for(int i= 0; s[i] ; i++)
    if (s[i] >= 'a' && s[i] <= 'z')   s[i] -=32;
}
int  strLen(char s[])
{
  int i ;
  for( i= 0; s[i]!= '\0' ; i++);
 return i;
}
void  strCpy(char s1[], char s2[])
{
  int i = 0;
  while (s1[i]) s2[i] = s1[i++];
  s2[i] = s1[i];
}
int  strCmp(char s1[], char s2[])
{
  int i = 0;
  while (s1[i])
  {
    if ( s1[i] >  s2[i] ) return 1;
    if  (s1[i] < s2[i] ) return -1;
    i++;
  }
  if (s2[i] ) return 1;
  else return 0;
}
bool isDigit(char *s)
{
  for (; *s; s++)
    if ( *s < '0' || *s  >'9' ) return false;
 return true;
}
bool isAlpha(char s[])
{
  for (int i= 0; s[i]; i++)
    if ( !((s[i] >= 'a' &&  s[i] <= 'z' ) ||(s[i] >= 'A' &&
s[i] <= 'Z' )) ) return false;
 return true;
}
void right(char s[], int n, char s1[])
{
  int index = strLen(s) - n;
  if (index < 0) index = 0;
  int i =0;
  while ( s[index] )
  {
   s1[i++]= s[index++];
  }
  s1[i] = '\0';
}
```

```
char*  left(char *s, int n)
{
  char* s1;
  int i = 0;
  while ( *s && i < n )
  {
   *s1 ++  = *s ++;
   i++;
  }
  *s1 = '\0';
  s1 -= i;
  return s1;
}
char* substr(char *s,int index,int n)
{
  int i =0;
  char *s1="";
  if (index < strLen(s)) s +=index;
 {
  while ( *s && i < n )
  {
   *s1 ++ = *s ++;
   i++;
  }
  *s1 = '\0';
 }
 s1 -= i;
 return s1;
}
void strCat(char s[], char* s1)
{
  int i = strLen(s) ;
  while (*s1) s[i++] = *s1 ++;
  s[i] = '\0';
}
void insert(char s[], int index, char* s1)
{
  if(index < strLen(s))
  {
   int curIndex = index;
   while (*s1)
   {
    for (int i= strLen(s); i >= curIndex; i--) s[i+1] = s[i];
    s[curIndex ++] = *s1 ++;
   }
  }
  else
  strCat(s, s1);
}
long convertToNumber(char s[])
{
  long number =0;
  int i =0 ;
  while (s[i] )
  {
    number = number * 10 + s[i++] - 48;
  }
  return number;
}
void main()
{
 char s[50]="1223679";
```

```
    char *s1;
    cout << convertToNumber(s);
    cout << "\nEnter a string:";
    cin.getline(s, 50);
    lower(s);
    cout <<"\n" <<  s;
    insert(s,3, "hfhfhhflflfl");
    cout << s;
    getch();
}
```



عملکرد توابع

☒ **تابع lower()**، اشاره‌گری از نوع کاراکتر به نام s (رشته) را به عنوان پارامتر دریافت کرده، کاراکترهای بزرگ رشته را به حروف کوچک تبدیل می‌کند و از طریق پارامتر s (پارامتر از نوع ارجاع) برمی‌گرداند. برای این منظور، با استفاده از یک حلقه while تا انتهای رشته جلو می‌رود (شرط حلقه به صورت s* است). بر اساس این شرط تا انتهای رشته کاراکترهایی از s را که بین 'A' تا 'Z' باشند، **۳۲** واحد به آن‌ها اضافه می‌کند و در مکان فعلی s قرار می‌دهد (وقتی به کاراکترهای بزرگ **۳۲** واحد اضافه نمایید، کاراکتر کوچک معادل آن بدست می‌آید).

☒ **تابع upper()**، آرایه رشته‌ای s را به عنوان پارامتر دریافت کرده، کلیه کاراکترهای کوچک آن را به حروف بزرگ تبدیل می‌کند و در خودش قرار می‌دهد و از طریق s به برنامه فراخوان برمی‌گرداند. برای این منظور، از یک حلقه تکرار for استفاده می‌کند. در این حلقه، کاراکترهای رشته s را از اولین کاراکتر (i = 0) تا زمانی که به '\0' نرسد (شرط s[i] داخل for) یکی، یکی بررسی می‌کند و هر کاراکتری از s که بین 'a' تا 'z' باشد، **۳۲** واحد از آن کم می‌کند و در همان مکان s قرار می‌دهد تا حروف کوچک رشته به حروف بزرگ تبدیل شوند.

☒ **تابع strLen()**، آرایه رشته‌ای به نام s را گرفته طول آن را برمی‌گرداند. برای این منظور، با استفاده از یک حلقه for تا زمانی که به کاراکتر '\0' نرسید ('\0' =! s[i]، معادل همان s[i] است)، به i یک واحد اضافه می‌کند (چون، انتهای for، کاراکتر ; قرار دارد) و در پایان i (همان طول رشته) را برمی‌گرداند.

☒ **تابع strCpy()**، دو رشته به نام‌های s1 و s2 را دریافت کرده، رشته s1 را در s2 کپی می‌کند. برای این منظور، از یک حلقه while استفاده می‌کند و تا زمانی که به انتهای رشته نرسید، کاراکتر به کاراکتر از s1 در s2 قرار می‌دهد (s2[i] = s1[i++]، ابتدا s1[i] را در s2[i] قرار می‌دهد، سپس به i یک واحد اضافه می‌کند). پس از پایان حلقه، s1[i] را در s2[i] قرار می‌دهد تا کاراکتر '\0' (انتهای رشته) s1 در انتهای s2 قرار گیرد.

☒ **تابع strCmp()**، دو رشته s1 و s2 را به عنوان پارامتر دریافت کرده، اگر رشته s1 کوچک‌تر از رشته s2 باشد، ۱- را برمی‌گرداند، اما، اگر رشته s1 بزرگ‌تراز رشته s2 باشد، ۱، وگرنه 0 را برمی‌گرداند. برای این

منظور، کاراکتر به کاراکتر رشته‌های s1 و s2 را با هم مقایسه می‌کند. اگر s1[i] > s2[i] باشد، ۱+ را برمی‌گرداند. اگر s1[i] < s2[i] باشد، ۱- را برگشت خواهد داد. اما، اگر برای هیچ کاراکتری از s1 و s2 دو شرط بیان شده صدق نکنند، تابع 0 (دو رشته برابرند) را برمی‌گرداند.

☒ **تابع isDigit()**، اشاره‌گر s از نوع کاراکتری (رشته s) را دریافت کرده، اگر تمام کاراکترهای آن از ارقام تشکیل شده باشند، true، وگرنه، false را برمی‌گرداند. برای این منظور، از یک حلقه for به صورت for(; *s; s++) استفاده می‌کند. این حلقه تا زمانی که به انتهای رشته نرسد، اگر محتوی s (*s) کوچک‌تر از کاراکتر '0' یا بزرگ‌تر از کاراکتر '9' باشد (شامل کاراکتر غیر عددی است)، تابع false را برمی‌گرداند. ولی، اگر به انتهای رشته برسد و هیچ یک از کاراکترهای رشته شامل این شرط نباشند، تابع true را برگشت خواهد داد.

☒ **تابع isAlpha()**، آرایه s (رشته‌ای) را به عنوان پارامتر دریافت کرده، اگر رشته شامل حداقل یک کاراکتر غیر الفبایی (غیر 'a' تا 'z' یا 'A' تا 'Z')باشد، تابع false، وگرنه true را برمی‌گرداند.

☒ **تابع right()**، آرایه رشته‌ای s، تعداد کاراکترهایی که باید از سمت راست رشته برگردانده شود (n) و پارامتر رشته‌ای s1 را گرفته، n کاراکتر از سمت راست رشته s جدا کرده، در s1 قرار می‌دهد. برای این منظور، ابتدا طول رشته s را از n کم کرده، در مکان شروع رشته s(index) که باید اطلاعات از آن مکان تا انتهای رشته در s1 کپی شوند، قرار می‌دهد. اگر index کوچک‌تر از صفر شود (یعنی، n بیشتر از طول رشته است)، index را برابر صفر قرار می‌دهد تا کپی از ابتدای رشته s به s1 انجام شود. برای این منظور، از دو شمارنده استفاده می‌کند. شمارنده i (کاراکتر فعلی رشته در s1) و شمارنده index (کاراکتر فعلی رشته در s) را تعیین می‌کند.

☒ **تابع left()**، اشاره‌گر رشته‌ای s(*s) و تعداد کاراکترهایی که باید از سمت چپ رشته جدا شود (n) را به عنوان پارامتر گرفته و n کاراکتر سمت چپ رشته را با دستور return برمی‌گرداند. (نوع تابع * char تعریف شده است. بنابراین، می‌تواند یک آدرس را برگرداند). برای این منظور، ابتدا، رشته s1 را به صورت اشاره‌گر تعریف می‌کند و کاراکتر به کاراکتر (تا n کاراکتر از ابتدای رشته به طرف انتهای رشته s) از رشته s در s1 قرار می‌دهد. در ادامه، در انتهای رشته s1، کاراکتر '\0' قرار می‌دهد و مکان فعلی که s1 اشاره می‌کند را از i کم می‌کند، در اشاره‌گر s1 قرار می‌دهد. این عمل اشاره‌گر s1 را به ابتدای متغیر s1 برمی‌گرداند. اکنون با دستور return، s1 را برمی‌گرداند (یعنی، آدرس شروع s1 برگشت داده می‌شود).

☒ **تابع substr()**، رشته s (از نوع اشاره‌گر)، آدرس مکانی در داخل رشته s (index) و تعداد کاراکترها (n) را به عنوان پارامتر دریافت کرده، n کاراکتر از مکان index رشته s را جدا کرده، به عنوان بخشی از رشته s با دستور return برمی‌گرداند. برای این منظور، متغیر i شمارنده برای تعیین n کاراکتر و رشته s1 (از نوع اشاره‌گر) را تعریف می‌کند تا بخش جدا شده رشته در آن قرار گیرد. اکنون، اگر index کوچک‌تر از طول

رشته s باشد، s را به اندازه index جلو می‌برد (s += index;) و کاراکتر به کاراکتر از مکان فعلی رشته s تا n کاراکتر یا به انتهای رشته s برسد، در s1 قرار می‌دهد. در پایان، s1 را از i کم کرده، در s1 قرار می‌دهد تا ابتدای رشته s1 را به دست آورد. اکنون، آدرس رشته s1 را برمی‌گرداند (return s1).

☒ **تابع strCat()**، آرایه‌ای از رشته (s) و اشاره‌گر رشته‌ای s1 را به عنوان پارامتر دریافت کرده، s1 را به انتهای رشته s اضافه می‌کند. برای این منظور، ابتدا، طول رشته s را در i قرار می‌دهد. اکنون کاراکتر به کاراکتر از رشته s1 در مکان iام رشته s قرار می‌دهد (این عمل تا زمانی انجام می‌شود که رشته s1 به کاراکتر '\0' نرسد). در هر زمان که کاراکتر s1 را در s قرار داد، s1 و s را یک واحد جلو می‌برد (دستور s[i++] = *s1++;). s[i++]، اشاره‌گر s را یک واحد جلو می‌برد و *s1++ اشاره‌گر s1 را یک واحد جلو می‌برد. یعنی، این دستور معادل سه دستور زیر است:

```
s[i] =  *s1;
i++;
s1 ++;
```

☒ **تابع insert()**، رشته s، مکان درج index و رشته s1 (به صورت اشاره‌گر) را به عنوان پارامتر دریافت کرده، رشته s1 را از مکان index در رشته s درج می‌کند. برای این منظور، ابتدا index را با طول رشته s مقایسه می‌کند تا index خارج از بازه رشته s نباشد، اگر index در داخل رشته s بود، برای هر کاراکتر s1 ((*s1) while) ابتدا، طول رشته s را پیدا کرده، و از انتهای رشته s، آن را یک کاراکتر به سمت راست منتقل می‌کند تا مکان فعلی که کاراکتر s1 باید در آن درج شود را به دست آورد (دستور زیر):

```
for (int i = strlen (s); i >= curIndex; i--) s[i+1] = s[i];
```

بعد از این که رشته s را یک کاراکتر به سمت راست منتقل کرد، کاراکتر فعلی رشته s1 را در مکان فعلی رشته s قرار می‌دهد (دستور زیر):

```
s [ curIndex ++] = *s1++;
```

ولی، اگر طول رشته s1 از s از طول رشته s بیشتر باشد، با فراخوانی تابع strCat، رشته s1 را به انتهای رشته s اضافه می‌کند.

☒ **تابع convertToNumber()**، رشته‌ای شامل کاراکترهای عددی s را به عنوان پارامتر دریافت کرده، به معادل عددی تبدیل می‌کند و برمی‌گرداند. برای این منظور، ابتدا، number را به صورت عددی تعریف کرده، در آن صفر قرار می‌دهد. سپس، از ابتدای رشته s شروع می‌کند و number را ضرب در ۱۰ کرده، به علاوه معادل مقدار عددی s[i] را در number قرار می‌دهد و این کار را تا خاتمه رشته ادامه می‌دهد. این عمل با دستور زیر انجام می‌شود:

```
number = number * 10 + s[i ++] – 48;
```

این دستور معادل دستورات زیر می‌باشد:

```
number = number * 10 + s[i] – 48 ; i ++;
```

s[i] – 48، معادل عددی کاراکتر s[i] را برمی‌گرداند .زیرا، کد اسکی کاراکترهای عددی بین ۴۸ تا ۵۷ است. بنابراین، اگر رقم صفر باشد، s[i] – 48 برابر با صفر (یعنی ۴۸ منهای ۴۸) است. این روند برای ارقام دیگر انجام خواهد شد. حال فرض کنید، رشته عددی به صورت زیر باشد:

s = "789"

این رشته به صورت مقابل در آرایه قرار می‌گیرد:

s[0]	s[1]	s[2]	
7	8	9	\0

اکنون با اجرای این تابع i و number برابر صفر خواهند شد. چون s[0]، کاراکتر ۷ است. پس دستور زیر انجام می‌شود:

number = 0 × 10 + 7 = 7

اکنون، یک واحد به i اضافه خواهد شد و s[1] برابر ۸ می‌شود. بنابراین، گام بعدی حلقه while دستور زیر را اجرا می‌کند:

number = 7 × 10 + 8 = 78

سپس یک واحد به i اضافه می‌شود و s[2] برابر با ۹ می‌شود. گام بعدی حلقه while به صورت زیر اجرا می‌شود:

number = 78 × 10 + 9 = 789

در ادامه یک واحد به i اضافه می‌گردد. چون، s[3] برابر با کاراکتر '\0' است، پس حلقه خاتمه می‌یابد و مقدار ۷۸۹ برگشت داده می‌شود.

۱۷ ـ ۵. مسائل حل شده در سایت

۱. برنامه‌ای که تعداد خانه‌های هر سطر شطرنج را گرفته، تعداد حالت‌های که n وزیر را می‌توان بدون برخورد با یکدیگر در این خانه‌های شطرنج قرار داد را نمایش می‌دهد.

۲. برنامه‌ای که دو آرایه ۴×۲ و ۳×۴ را دریافت کرده، حاصل ضرب آن‌ها را محاسبه نموده، نمایش می‌دهد.

۳. برنامه‌ای که دو آرایه ۴ × ۳، x و y را خوانده، حاصل عبارت زیر را در آرایه z قرار می‌دهد:

Z = 2x – 3y

۴. برنامه‌ای که a، b و c (سه عدد) را خوانده، با استفاده تابعی نوع مثلث را تعیین کند (تابع توسط اشاره‌گر یک پیغام را برمی‌گرداند).

۵. برنامه‌ای که بین تعدادی اعداد که در آرایه قرار دارند، با استفاده از تابعی بزرگ‌ترین، کوچک‌ترین مقدار و مکان آن‌ها را پیدا کرده، برمی‌گرداند (این تابع بزرگ‌ترین، کوچک‌ترین مقدار و مکان آن‌ها را از طریق پارامتر ارجاع برمی‌گرداند).

۶. برنامه‌ای که ۱۰ عدد تصادفی تولید کرده، در آرایه‌ای قرار می‌دهد (توسط یک تابع)، سپس دو عدد بزرگ‌تر را پیدا می‌کند (توسط تابع دیگری). در پایان، عناصر آرایه را نمایش داده (توسط تابع سوم) و دو عدد بزرگ‌تر را نیز نمایش می‌دهد.

۷. برنامه‌ای که ۱۰ عدد تصادفی بین صفر تا ۲۰ تولید می‌کند، در آرایه‌ای قرار می‌دهد. سپس عناصر آرایه را نمایش می‌دهد و عددی را دریافت کرده، تمام تکرار عدد در آرایه را حذف و نتیجه را نمایش می‌دهد.

۸. برنامه‌ای که ۲۰ عدد تصادفی بین صفر تا ۲۰ تولید کرده، در دو آرایه قرار می‌دهد. سپس، یک عدد را خوانده، تمام عناصر آرایه دوم را در آرایه اول از آن مکان دریافت شده درج می‌نماید.

۹. برنامه‌ای که ۲۰ عدد تصادفی را تولید کرده، در دو آرایه قرار می‌دهد و سپس عناصر آرایه اول و دوم را با هم جمع کرده، نمایش می‌دهد.

۱۰. برنامه‌ای که بخشی از رشته را حذف می‌کند (این برنامه دارای دو تابع به نام‌های lenStr() (برای محاسبه طول رشته) و delStr (جهت حذف بخشی از رشته) می‌باشد.

۱۱. برنامه‌ای که رشته‌ای را خوانده، سپس، دو رشته دیگر را می‌خواند و رشته سوم را جایگزین رشته دوم در رشته اول می‌نماید.

۱۲. برنامه‌ای که مرتب‌سازی ادغامی (merge) را پیاده‌سازی می‌کند.

۱۳. برنامه‌ای که تعدادی عدد تصادفی تولید کرده، در آرایه ۴ × ۴ قرار می‌دهد. سپس، سطرهای آرایه را جداگانه مرتب کرده، نمایش می‌دهد.

۱۴. برنامه‌ای که ۷ عدد را خوانده، اعدادی که کمتر از تفاوت بین بزرگ‌ترین و کوچک‌ترین عدد هستند را نمایش می‌دهد.

۱۵. برنامه‌ای که آرایه پنج عنصری را خوانده، توسط علامت * میله افقی برای اعداد خانه‌های آن آرایه رسم می‌کند.

۱۶. برنامه‌ای که ۷ عدد را خوانده، سپس دو عدد دیگر خوانده و عدد دوم را جایگزین عدد اول در ۷ می‌نماید.

۱۷. برنامه‌ای که دو عدد صحیح و مثبت m و n که $m > n > 3$ و $m - n < 10$ باشد، را از ورودی خوانده، بر اساس جدول زیر ماتریس را تولید و چاپ می‌نماید:

به عنوان مثال، m = 7 و n = 4 ماتریس زیر را تولید می‌کند:

+	n	n+1	n+2	…	m
m					
m-1					
m-2					
…					
n					

	4	5	6	7
7	11	12	13	14
6	10	11	12	13
5	9	10	11	12
4	8	9	10	11

۱۸. برنامه‌ای که عدد صحیح n (کوچک‌تر از ۹) را از ورودی خوانده، خروجی زیر را با استفاده از آرایه دو بعدی (ماتریس)، تولید و چاپ می‌کند:

```
1      2      3      …   n
2      3      4      …   n +1
…
N+1    n+2    n+ 3   …   n + n
```

۱۹. برنامه‌ای که رشته‌ای را خوانده، با استفاده از یک تابع فضای خالی سمت چپ (ابتدای رشته) را حذف می‌کند (کد اسکی کاراکتر جای خالی ۳۲ است).

۲۰. برنامه‌ای که رشته‌ای را خوانده، با استفاده از اشاره‌گر و تابع بازگشتی، طول رشته را حساب کرده، نمایش می‌دهد.

۲۱. برنامه‌ای که رشته‌ای را خوانده، با استفاده از اشاره‌گر و تابع بازگشتی، معکوس رشته را نمایش می‌دهد.

۲۲. برنامه‌ای که دو آرایه 5 و 10 عنصری را دریافت کرده، در آرایه سوم ادغام کند.

۲۳. برنامه‌ای که ماتریسی ۳ × ۳ را خوانده، میانگین حاصل جمع ستون‌های آن را نمایش می‌دهد.

۲٤. برنامه‌ای که ماتریسی ۳ × ۳ را خوانده، حاصل جمع عناصر قطر اصلی را نمایش می‌دهد.

۲۵. برنامه‌ای که حاصل جمع عناصر قطر فرعی یک ماتریس n × n (حداکثر ۱۰۰× ۱۰) را نمایش می‌دهد.

۲٦. برنامه‌ای که دو نام را خوانده، تعیین می‌کند برابر هستند یا نه؟

۲۷. برنامه‌ای که تعدادی نام را دریافت کرده، سپس یک نام جدید را خوانده و تعیین می‌کند این نام در لیست نام‌های دریافتی قبلی وجود دارد یا خیر؟

۲۸. برنامه‌ای که دو ماتریس حداکثر ۱۰۰ × ۱۰۰ را دریافت کرده، تعداد سطرهای نظیر که با هم برابر هستند را نمایش می‌دهد.

۲۹. برنامه‌ای که لیستی از اسامی افراد را دریافت کرده، نام‌های مشترک بین دو لیست را نمایش می‌دهد.

۱۸ ـ ۵. تمرین‌ها

۱. برنامه ای بنویسید که عددی صحیح را از ورودی خوانده، تمام اعداد اول قبل از آن را با استفاده از تعریف بیان شده تعیین کرده، به خروجی ببرد. عددی اول است که بر هیچ عدد اول قبل از خودش قابل قسمت نباشد.

۲. برنامه‌ای بنویسید که شماره دانشجویی تعدادی از دانشجویان را از ورودی خوانده، در آرایه قرار دهد. سپس عناصر آرایه را به روش انتخابی مرتب کند. مرتب سازی آرایه به روش انتخابی به این صورت انجام می‌شود: کوچک‌ترین عنصر آرایه پیدا شده، جای آن با عنصر اول آرایه عوض می‌شود. در مرحله بعد بقیه عناصر آرایه برای یافتن کوچک‌ترین عنصر آرایه جست‌وجو می‌شود و جای آن با عنصر دوم آرایه عوض می‌شود. این روند تا مرتب سازی کامل آرایه ادامه می‌یابد. پس از مرتب سازی نتیجه را در خروجی چاپ می‌کند. این برنامه باید دارای سه تابع باشد که عبارت‌اند از: تابعی برای خواندن عناصر آرایه، تابعی برای مرتب سازی و تابعی برای چاپ عناصر آرایه.

۳. برنامه‌ای بنویسید که تعدادی عدد را از ورودی خوانده، آن‌ها را به طور مرتب در آرایه‌ای قرار دهد (دقت داشته باشید که اعداد در موقع ورود در آرایه، به طور صعودی مرتب شوند). سپس، آرایه مرتب شده، را به خروجی ببرد.

٤. برنامه‌ای بنویسید که دو رشته S2 و S1 را از ورودی خوانده، رشته S1 را در رشته S2 جست‌وجو کند. در این برنامه، خواندن رشته‌ها توسط تابع اصلی و جست‌وجوی رشته توسط تابعی دیگر انجام شود.

۵. برنامه‌ای بنویسید که با خواندن تعدادی عدد از ورودی، آن‌ها را در آرایه‌ای قرار دهد. سپس، کلیه عناصر آرایه را بر عنصر وسط تقسیم کند. اگر عنصر وسط صفر باشد، بر عنصر بعد از عنصر وسط تقسیم نماید. اگر این عنصر صفر باشد، بر عنصر قبل از عنصر وسط تقسیم کند. اگر این عنصر نیز صفر باشد، برای پیدا کردن عنصری غیر صفر و انجام تقسیم، به روند قبلی ادامه دهد. اگر همه عناصر آرایه صفر باشند، پیام مناسبی صادر کند.

٦. کارخانه‌ای دارای ۵ ردیف شغلی است که به کارمندان در مقابل ۴۰ ساعت کار در هفته، ماهانه مطابق جدول زیر حقوق ثابتی پرداخت می‌شود. در صورتی که کارمندی بیش از ۱۶۰ ساعت در ماه کار کند به او اضافه‌کاری به ازای هر ساعت مطابق جدول زیر، پرداخت می‌گردد. اگر کارمندی کمتر از ۱۶۰ ساعت در ماه کار کند، به ازای هر ساعت مبلغی مطابق جدول از حقوق وی کسر می‌شود.

ردیف شغلی	نوع تخصص	حقوق ثابت	هر ساعت اضافه‌کاری	هر ساعت کم‌کاری
۰	مهندس ارشد	1,000,000	5,000	6,000
۱	مهندس ساده	800,000	4,000	5,000
۲	تکنسین	600,000	3,000	4,000
۳	کارگر ماهر	500,00	2,500	3,000
٤	کارگر ساده	460,000	2,000	2,500

برنامه‌ای بنویسید که شماره کارمندی، ردیف شغلی و ساعت کارکرد در ماه برای هر کارمند را خوانده، دریافتی آن‌ها را چاپ کند. برای خاتمه برنامه به جای شماره کارمندی **۹۹۹**- وارد می‌شود.

۷. برنامه‌ای بنویسید که n عدد را خوانده، در آرایه‌ای قرار دهد و توسط تابعی عناصر آن را معکوس (از آخرین عدد به اولین عدد) کرده، به برنامه برگرداند و برنامه آن را نمایش دهد (برنامه، برای دریافت داده، معکوس کردن و چاپ عناصر آرایه باید از توابع جداگانه استفاده کند).

۸. فرض کنید که در قسمت تخلفات ماشین‌ها در اداره راهنمایی رانندگی، ۱۰ نوع تخلف منظور گردید و کد تخلف از ۰ تا ۹ در نظر گرفته شد و هر تخلف جریمه خاصی دارد. برای هر ماشین اطلاعاتی مثل شماره ماشین، تعداد تخلفات و کد هر تخلف موجود است. برنامه‌ای بنویسید که مبلغ جریمه را برای هر ماشین محاسبه کند. برای خروج از برنامه کاربر به جای شماره ماشین **۹۹۹**- را وارد می‌کند.

۹. برنامه‌ای بنویسید که دو عدد مبنای ۲ را خوانده، جمع کرده، به خروجی ببرد (طول عدد مبنای دو ۳۲ رقم است).

۱۰. برنامه‌ای بنویسید که عناصر آرایه‌ای ۴×۴ را خوانده، بزرگ‌ترین عنصر هر سطر را پیدا کرده، به همراه شماره سطر در خروجی چاپ کند.

۱۱. برنامه‌ای بنویسید که رشته‌ای را در رشته دیگر کپی کند (بدون استفاده از تابع strcpy).

۱۲. ماتریس جادویی یک ماتریس $N \times N$ است که هر عنصر آن یک عدد صحیح ۱ تا N^2 است. حاصل جمع هر سطر، هر ستون و همچنین عناصر قطر همگی با یکدیگر مساوی‌اند. برای تولید این ماتریس به صورت زیر عمل می‌شود:

۱۵	۸	۱	۲٤	۱۷
۱٦	۱٤	۷	۵	۲۳
۲۲	۲۰	۱۳	٦	٤
۳	۲۱	۱۹	۱۲	۱۰
۹	۲	۲۵	۱۸	۱۱

الف: عدد یک در ستون میانی اولین سطر قرار می‌گیرد.

ب: به طور مورب به سمت چپ و بالا حرکت می‌کنیم.

ج: اگر این خانه پر باشد، یک خانه به طرف پایین حرکت می‌کنیم و عدد بعدی را در آن مکان می‌نویسیم.

د: اگر در حرکت به صورت مورب، از محدوده ماتریس خارج شویم، باید به آخرین عنصر در آن سطر یا ستون برگردیم.

ح: اگر در محدوده سطر و ستون از هر دو خارج شویم، یک خانه به سمت پایین حرکت می‌کنیم.

برنامه‌ای بنویسید که ماتریس جادویی را تولید کرده و چاپ نماید.

۱۳. برنامه‌ای بنویسید که رشته‌ای را به انتهای رشته دیگر اضافه کند (بدون استفاده از تابع strcat).

۱٤. برنامه‌ای بنویسید که رشته‌ای را از ورودی خوانده، به تابعی ارسال کند و تابع آن را به طور معکوس به خروجی ببرد.

۱۵. برنامه‌ای بنویسید که رشته عددی که حاوی نقطه اعشار است را از ورودی خوانده، آن را به عدد اعشاری تبدیل کند. به عنوان مثال، رشته "72.352" را به 72.352 تبدیل نماید. این برنامه، تابعی برای خواندن رشته، تابعی برای تبدیل و تابعی برای نوشتن عدد در خروجی دارد. پارامترها از طریق فراخوانی با ارجاع به توابع ارسال می‌شوند.

۱٦. برنامه‌ای بنویسید که بازی دوز را شبیه‌سازی کند. در این بازی یک ماتریس ۳×۳ داریم که دو بازیکن ۱ و ۲ با هم بازی می‌کنند که هر یک از این بازیکنان سه مهره دارند (هر سه مهره شماره همان بازیکن را دارند). بازیکنی برنده است که بتواند سریع‌تر مهره‌های خود را به صورت سطری یا ستونی و یا قطری ردیف کند. به عنوان مثال، در ماتریس زیر بازیکن شماره ۲ برنده است.

		2
	2	1
2	1	1

۱۷. هر عدد زوج بزرگ‌تر از ۶ برابر با مجموع دو عدد اول کوچک‌تر از خودش است. برنامه‌ای بنویسید که چند عدد زوج بزرگ‌تر از ۶ را خوانده، مجموع دو عدد اولی که برابر با اعداد خوانده شده باشند را پیدا کرده، چاپ نماید (برنامه تا زمانی که کاربر بخواهد ادامه می‌یابد).

۱۸. برنامه‌ای بنویسید که رشته عددی را که حاوی نقطه اعشار است را از ورودی خوانده، آن را به صورتی که گفته شد به عدد اعشاری تبدیل کند. به عنوان مثال، رشته "7245.465" را به عدد 465.7245 تبدیل کند. برنامه، تابعی برای خواندن رشته، تابعی برای تبدیل و تابعی برای نوشتن عدد در خروجی داشته باشد. پارامترها از طریق فراخوانی با ارجاع به توابع ارسال شوند.

۱۹. تابعی بنویسید که طول رشته را برمی‌گرداند (با استفاده از اشاره‌گر). سپس، برنامه‌ای بنویسید که از این تابع استفاده کند. برنامه، تابعی برای خواندن رشته، تابعی برای محاسبه طول رشته و تابعی برای چاپ طول رشته داشته باشد.

۲۰. تابعی بنویسید که توان n یک عدد را برمی‌گرداند. سپس، برنامه‌ای بنویسید که از این تابع استفاده کند.

۲۱. برنامه‌ای بنویسید که سه رشته را خوانده، رشته دوم و سوم را به انتهای رشته اول اضافه کند.

۲۲. برنامه‌ای بنویسید که رشته‌ای را خوانده اعمال زیر را انجام دهد:

الف. کلیه حروف بزرگ رشته را به حروف کوچک تبدیل کند.

ب. کلیه ارقام رشته را شمارش کند و مجموع ارقام موجود در رشته را چاپ کند.

ج: تعداد حروف رشته را شمارش کند.

برنامه، برای هر یک از این درخواست‌ها تابعی داشته باشد و کلیه پارامترهای تابع را با ارجاع ارسال نماید.

۲۳. برنامه‌ای بنویسید که میزان میانگین بارش باران را در سال قبل بگیرد و سپس میزان واقعی بارش باران را در ۱۲ ماه گرفته، در آرایه‌ای قرار دهد (برای خواندن میزان واقعی بارش باران در ۱۲ ماه از یک تابع استفاده کند). سپس، با یک تابع دیگر اختلاف بارش باران در هر ماه را با میانگین بارش باران در سال قبل محاسبه کرده، نمایش دهد.

۲۴. برنامه‌ای بنویسید که ۵ عدد را از ورودی خوانده، در آرایه‌ای به نام S قرار دهد. برنامه ابتدا میانگین مقادیر را محاسبه کند (با یک تابع). سپس، با استفاده از فرمول زیر انحراف معیار را محاسبه کرده، چاپ نماید (برای محاسبه انحراف معیار نیز از یک تابع استفاده کند):

$$\text{انحراف معیار} = \frac{\sqrt{\sum_{i=1}^{N} (Si - a)^2}}{N}$$

a، میانگین اعداد و N، تعداد اعداد است.

۲۵. برنامه‌ای بنویسید که رشته‌ای را از ورودی خوانده، تمام کلمات چهار حرفی آن را با کلمه "Love" جایگزین کند. مثلاً رشته "hate you, you doer" به رشته love you, you love تبدیل گردد.

۲۶. در یک دوره مسابقات ورزشی تعدادی تیم (حداکثر ۱۰۰ تیم) شرکت کرده‌اند، می‌خواهیم در انتهای مسابقه گزارشی از مجموع امتیازات هر تیم را چاپ کنیم. برنامه‌ای بنویسید که برای هر مسابقه در یک خط ورودی شامل آیتم‌های برنده و امتیاز کسب شده را خوانده، نتیجه را نمایش دهد. اگر به جای تیم برنده ۱- وارد شد، برنامه خاتمه یابد.

۲۷. برنامه‌ای بنویسید که یک رشته را خوانده، تعداد تکرار هر حرف را نمایش دهد. به عنوان مثال، ACCURENCE را در نظر بگیرید که حرف C سه بار تکرار شده است.

۲۸. برنامه‌ای که یک رشته را خوانده، تشخیص دهد آیا متجانس است یا خیر؟ رشته‌ای متجانس است که از دو طرف یکی خوانده شود.

۲۹. برنامه‌ای که حداکثر ۱۰ عدد را از ورودی خوانده، بگوید هر عدد چند بار تکرار شده است. به عنوان مثال، اگر ورودی به صورت زیر باشد:

70 100 143 100 52 143 72 100 143 70

خروجی زیر را چاپ کند:

143 OCCURS 3 TIMES, AT POSITIONS 3 6 9
70 OCCURS 2 TIMES, AT POSITIONS 1 10

...

۳۰. برنامه‌ای بنویسید که ماتریس A[۳×۴] را از ورودی خوانده و ترانهاده آن را در B ذخیره کند. در ماتریس ترانهاده باید برای هر j , i رابطه زیر برقرار باشد:

$$B_{i,j} = A_{j,i}$$

۳۱. برنامه‌ای بنویسید که انداختن دو تاس را شبیه‌سازی نماید. این برنامه برای انداختن تاس‌ها از تابع rand استفاده کند که عدد بین ۱ تا ۶ را تولید نماید. سپس مجموع این دو مقدار محاسبه گردد. از آنجایی که تاس‌ها مقادیر بین ۱ تا ۶ را نشان می‌دهند، پس مجموع این دو مقدار بین ۲ تا۱۲ است که مجموع ۷، بیشترین دفعات و مجموع ۱۲ کم‌ترین دفعات پیش می‌آیند. این جدول ترکیبات ممکن برای این دو تاس را نشان می‌دهد. برنامه، ۳۶۰۰۰ بار تاس می‌ریزد و با استفاده از یک آرایه، تک بعدی تعداد دفعاتی را که هر یک از این مجموع‌های ممکن پیش می‌آیند، را ثبت می‌کند و سپس تحقیق می‌کند که آیا تعداد مجموع‌های حاصل معقول است (مثلاً به شش طریق، مجموع ۷ حاصل می‌شود. بنابراین تقریباً 1/6 همه پیشامدها باید ۷ باشد).

	۱	۲	۳	۴	۵	۶
۱	۲	۳	۴	۵	۶	۷
۲	۳	۴	۵	۶	۷	۸
۳	۴	۵	۶	۷	۸	۹
۴	۵	۶	۷	۸	۹	۱۰
۵	۶	۷	۸	۹	۱۰	۱۱
۶	۷	۸	۹	۱۰	۱۱	۱۲

۳۲. در هر سطر ورودی برای هر کالا دو عدد وجود دارد. عدد اول شماره کالا و عدد دوم موجودی آن در انبار است.

الف: برنامه‌ای که اطلاعات اولیه را می‌خواند و در یک آرایه ذخیره می‌کند. انتهای اطلاعات با دو عدد ۱- مشخص شده است.

ب: پس از ورود اطلاعات اولیه تعدادی ورودی دیگر وجود دارد. این ورودی‌ها عبارت‌اند از: شماره کالا، تعداد و کاراکتری که مشخص کننده این است که تعداد ورودی سفارش (کاراکتر s یا S) یا خرید (کاراکترp یا P) است. برنامه‌ای بنویسید که اطلاعات را خوانده، مقدار خروجی انبار هر کالا را نمایش دهد. اگر سفارش بیش از موجودی انبار باشد، آن را انجام نمی‌دهد و پیام مناسبی چاپ کند.

۳۳. برنامه‌ای بنویسید که تعدادی کلمه را از ورودی خوانده، تعداد کلمات ۱ تا ۵ کاراکتری، ۶ تا ۱۲ کاراکتری و بیشتر از ۱۲ کاراکتر را نمایش دهد (تعداد کلمات را از ورودی می‌خواند).

۳٤. برنامه‌ای بنویسید که چند جمله‌ای زیر را در یک آرایه نمایش دهد (سپس، x را می‌خواند) و عبارت P را محاسبه کرده، چاپ نماید (n حداکثر ۱۰۰ باشد).

$$P = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x^1 + a_0$$

۳۵. برنامه‌ای بنویسید که یک رشته حداکثر ۱۰۰ کلمه‌ای را خوانده تمام کلمات آن را وارونه کند و چاپ نماید (بین کلمات کاراکتر blank (فاصله) قرار می‌گیرد). به عنوان مثال، رشته Ali and Reza به رشته ilA dna azeR تبدیل شود.

۳٦. برنامه‌ای که یک متن را از ورودی خوانده، اعمال زیر را انجام دهد:

الف: درصد تکرار هر حرف را در متن پیدا کند (حروف بین A تا Z و حروف کوچک را بزرگ در نظر بگیرید).

ب: درصد تکرار هر کلمه را در متن پیدا می‌کند (کلمات با فاصله از یکدیگر جدا می‌شوند (حداکثر ۱۰۰ کلمه داریم).

۳۷. برنامه‌ای بنویسید که تعدادی عدد را خوانده، تشخیص دهد مرتب شده هستند یا خیر.

۳۸. برنامه‌ای بنویسید که n عدد را خوانده، در آرایه قرار دهد. سپس، توسط تابعی عنصر kام را حذف کند.

۳۹. برنامه‌ای بنویسید که دو آرایه m و n عنصری را خوانده، عناصر آرایه دوم را به انتهای آرایه اول اضافه کند.

☒برنامه‌ای بنویسید که عناصر یک آرایه و مقدار x را خوانده، تعداد تکرار مقدار x در آرایه a را بشمرد. این برنامه توابعی برای خواندن عناصر آرایه a و برای شمارش تعداد تکرار x در آرایه a دارد.

۴۰. برنامه‌ای بنویسید که دو آرایه n عنصری یک بعدی را خوانده، حاصل ضرب بیرونی n عنصر اول آرایه a با n عنصر اول آرایه b را برگرداند (حاصل ضرب این دو آرایه، در آرایه دیگری به نام c قرار بگیرد). به عنوان مثال، فرض کنید آرایه‌های زیر را داشته باشیم.

a

۲/۲
۳/۳
۴/۴

b

۲/۰
−۱/۰
۰,۰

۰/۰	−۲/۲	۴/۴
۰/۰	−۳/۳	۶/۶
۰/۰	−۴/۴	۸/۸

☒ تابعی برای ضرب بیرونی(outer product) داشته باشید.

☒ تابعی برای چاپ حاصل ضرب داشته باشید.

$$c[i][j] = a[i] \times b[j]$$

۴۱. برنامه‌ای بنویسید که عناصر یک آرایه دو بعدی را ۹۰ درجه چرخش دهد. به عنوان مثال، اگر آرایه زیر را داشته باشیم.

11	22	33
44	55	66
77	88	99

با چرخش ۹۰ درجه، آرایه زیر را خواهیم داشت:

77	44	11
88	55	22
99	66	33

پیوست ب

پروژه برنامه‌نویسی

برنامه‌ای که بازی مین روب را پیاده‌سازی می‌کند. این برنامه دارای امکانات زیر می‌باشد:

۱. کاربر می‌تواند تعداد مین‌های صفحه را وارد کند.

۲. طراحی صفحه بازی مین روب به صورت گرافیکی است.

۳. ماوس در بازی فعال است و کاربر می‌تواند خانه‌های بدون مین را کلیک کند یا با کلیک راست بر روی خانه‌ها آن‌ها را به علامت ? یا پرچم تغییر دهد.

٤. مدت زمان صرف شده بازی و تعداد مین‌ها در صفحه، نمایش داده می‌شود.

```
#include<stdlib.h>
#include<conio.h>
#include<stdio.h>
#include<time.h>
#include<math.h>
#include<graphics.h>
#include<dos.h>
#include<iostream.h>
void free(int, int );
void minSweeper();
int  minrub[14][24], helpMin[14][24], leftClick[14][24];
int op = 0;
char m[2], timeString1[3], timeString2[3], timeString3[3];
struct time t, t2;
void initArray();
void fillMin(int);
void displayMemo();
void countMinRound();
void drawScreen();
void drawSymbol();
void drawSymbol1();
void drawMin(int, int, int );
void drawFillRectangle();
void drawFillRectangle(int, int , int);
void findEL(int, int, union REGS, int &, int &);
void findEL(union REGS, int &, int &);
void initMouse();
void showMinCount(int);
void showInitTime();
void showTime();
int main()
{
  int gdriver = DETECT, gmode, errorCode;
  int left, top, right, bottom;
  initgraph(&gdriver, &gmode, "../BGI");
```

```
  errorCode = graphresult();
  if (errorCode != grOk)
  {
   cout << "Graphics error: " << grapherrormsg(errorCode) << endl;
   cout << "Press any key to halt:";
    getch();
    exit(1);
  }
  minSweeper();
  restorecrtmode();
  return 0;
}
void minSweeper()
{
    int x0, y0, d, c, i, j, b, count = 0, num;
    int a, k, e = 0, l = 0, y, endTime = 0, u = 0, g, win = 0;
    union REGS inreg, outreg;
    static begin = 1;
    int  driver = DETECT,gdriver = DETECT, mode;
    mode = 0;
    cout << "Enter the number of bombs between 1 to 200 > ";
    cin >> num;
    initArray();
    fillMin(num);
    countMinRound();
    clrscr();
    if(begin == 1)
    {
      displayMemo();
    }
    begin = 0;
    drawScreen();
    drawSymbol();
    initMouse();
    showMinCount(num);
    showInitTime();
    do
    {
      inreg.x.ax = 3;
      int86(0x33, &inreg, &outreg);
      if(outreg.x.bx == 1) //if Left Click
      {
       if(endTime == 0)
       {
        gettime(&t);
        endTime = 100;
       }
       if(outreg.x.cx > 297 && outreg.x.cx < 331 &&
          outreg.x.dx > 21 && outreg.x.dx < 53)  minSweeper();
       if(minrub[a][b] != 11)
       {
        findEL(outreg, a, b);
       }//of if
      if(helpMin[a][b] % 3 == 1)
      {
```

```
    inreg.x.ax = 2;
    int86(0x33, &inreg, &outreg);
    setcolor(BLUE);
    rectangle(9 + 31 * (b - 1), 58 + 31 * (a - 1), 40 + 31
              *(b - 1), 89 + 31 *( a-1));
    setfillstyle(1,7);
    floodfill(9 + 31 * (b - 1) + 2,58 + 31 * (a - 1) +2, 1);
    itoa(minrub[a][b], m, 10);
    if(minrub[a][b] == 0) free(a,b);
    for(i=1; i <= 2; i++)
     if(minrub[a][b] == i) setcolor(i);
    if(minrub[a][b] == 3) setcolor(4);
    if(minrub[a][b] == 4) setcolor(13);
    for(i=5; i <= 8; i++)
   if(minrub[a][b] == i) setcolor(i);
    leftClick[a][b]++;
    if(minrub[a][b] == 11)
    {
   drawSymbol1();
   for(i = 1; i < 11; i++)
    for(j = 1; j < 21; j++)
    if(helpMin[i][j] %3 == 2 && minrub[i][j] != 11)
    {
     setcolor(BLUE);
     rectangle(9+31*(j-1),58+31*(i-1),40+31*(j-1),89+31*(i-1));
     setfillstyle(1, 7);
     floodfill(9 + 31 *(j-1) + 2, 58 + 31 *(i - 1) + 2, 1);
     setcolor(RED);
     line(9+31*(j-1)+3,58+31*(i-1)+3,40+31*(j-1)- 3, 89 +31
              * (i - 1) -3);
     line(40 + 31 * (j - 1) - 3, 58 + 31 * (i - 1) + 3, 9 +
           31 * (j - 1) + 3, 89 + 31 * (i - 1) - 3);
   }
   for(i = 1; i < 11; i++)
     for(j = 1; j < 21; j++)
      if(minrub[i][j] == 11)
      {
       if(helpMin[i][j] % 3 != 2)
       {
        x0 =((9 + 31 * (j - 1)) + (40 + 31 * (j - 1))) / 2;
        y0 = (58 + 31 * (i - 1) + 89 + 31 * (i - 1)) / 2;
        drawFillRectangle(i, j, 7);
        if(i == a && j == b)
        {
          drawFillRectangle(i, j, RED);
        }
         drawMin(x0, y0, 1);
    }
   }//of if helpMin
 } //of if minrub=11
 if(minrub[a][b]!=0&&minrub[a][b]!=11&& minrub[a][b] != 12)
   outtextxy((9 + 31 * (b - 1) + 40 + 31 * (b - 1)) / 2 -
         2,(58 + 31 * (a - 1) + 89 + 31 * (a -1))/2-3,m);
if(minrub[a][b] != 11)
{
```

```
   setcolor(14);
   arc(314, 37, 220, 320, 6);
   setcolor(RED);
   circle(314, 43, 3);
  }
  delay(250);
 }
 inreg.x.ax = 1;
 int86(0x33, &inreg, &outreg);
}//if click left.
if(outreg.x.bx != 1 && minrub[a][b] != 11)
{
  setcolor(14);
  circle(314, 43, 3);
  setcolor(RED);
  arc(314, 37, 220, 320, 6);
}
inreg.x.ax = 3;
int86(0x33, &inreg, &outreg);
if(outreg.x.bx == 2)
{
  if(endTime == 0)
  {
    gettime(&t);
    endTime = 100;
  }
  if(outreg.x.cx > 9 && outreg.x.cx < 629 && outreg.x.dx > 58
       && outreg.x.dx < 368)
  {
    if(minrub[a][b] != 11)
    {
       findEL(outreg, e, l);
       if(leftClick[e][l] == 0 && g != count && win != 200)
       {      //44
         inreg.x.ax = 2;
         int86(0x33, &inreg, &outreg);
         if(helpMin[e][l] %3 ==1)
         {
           helpMin[e][l]++;
           setcolor(7);
           outtextxy(90, 35, timeString1);
           num--;
           drawFillRectangle();
           itoa(num, timeString1, 10);
           outtextxy(90, 35, timeString1);
           for(i = 1; i < 11; i++)
            for(j = 1; j < 21; j++)
            {
             findEL(i, j, outreg, e, l);
             a = (9 + 31*(l-1) + 40 + 31 * (l - 1)) / 2 - 5;
             b = (58 + 31*(e-1) + 89 + 31 * (e - 1)) / 2 - 3;
             setcolor(7);
             outtextxy(a + 4, b, "?");
             setcolor(RED);
             line(a + 2, b, a + 7, b + 2);
```

```
    line(a + 2, b, a + 7, b - 2);
    line(a + 7, b + 2, a + 7, b - 2);
    setfillstyle(1, RED);
    floodfill(a + 5, b, RED);
    setcolor(8);
    line(a + 7, b + 2, a + 7, b + 7);
    setcolor(RED);
    delay(2);
    outtextxy(90, 35, timeString1);
  } //of for
}
else if(helpMin[e][l]%3 == 2)
{
   findEL(outreg, e, l);
   helpMin[e][l]++;
   setcolor(7);
   outtextxy(90, 35, timeString1);
   num++;
   drawFillRectangle();
   itoa(num, timeString1, 10);
   outtextxy(90, 35, timeString1);
   for(i = 1; i < 11; i++)
    for(j = 1; j < 21; j++)
    {
     findEL(i, j, outreg, e, l);
     setcolor(RED);
     a = (9 + 31*(l-1) + 40 + 31 * (l - 1)) / 2 - 5;
     b = (58 + 31*(e- 1)+89 + 31 * (e - 1)) / 2 - 3;
     setcolor(7);
     line(a + 2, b, a + 7, b + 2);
     line(a + 2, b, a + 7, b - 2);
     line(a + 7, b + 2, a + 7, b - 2);
     setfillstyle(1, 7);
     floodfill(a + 5, b, 7);
     setcolor(7);
     line(a + 7, b + 2, a + 7, b + 7);
     setcolor(1);
     outtextxy(a + 4, b, "?");
     setcolor(RED);
     delay(2);
     outtextxy(90, 35, timeString1);
    }
 }
 else
 {
   findEL(outreg, e, l);
   helpMin[e][l]++;
    for(i = 1; i < 11; i++)
     for(j = 1; j < 21; j++)
     {
      findEL(i, j, outreg, e, l);
      setcolor(7);
      a = (9 + 31*(l-1)+40 + 31 *(l - 1)) / 2 - 5;
      b=(58+31 * (e - 1) + 89 + 31 * (e - 1))/2 - 3;
      delay(2);
```

```
                setcolor(7);
                outtextxy(a + 4, b, "?");
               } //of for
             }
            }
            inreg.x.ax = 1;
            int86(0x33, &inreg, &outreg);
         }
       }
    }
    delay(200);
    gettime(&t2);
    if(minrub[a][b] != 11)
    {
     if(endTime > 0 && g != count && win != 200)
     {
       showTime();
     }
   }
   win = 0;
   for(i = 1; i <= 10; i++)
     for(j = 1; j <= 20; j++)
      if(helpMin[i][j] % 3 == 2 && minrub[i][j] == 11) win++;
    for(i = 1; i <= 10; i++)
     for(j = 1; j <= 20; j++)
      if(minrub[i][j] != 11 && leftClick[i][j] >= 1) win++;
    if(win == 200)
    {
      setcolor(1);
      circle(309, 32, 4);
      setfillstyle(1, 8);
      floodfill(310, 33, 1);
      circle(320, 32, 4);
      floodfill(318, 33, 1);
      line(312, 32, 317, 32);
     }
 }
 while(!kbhit());
 return;
}
void initArray()
{
  for(int j = 0; j <= 21; j++)
   for(int i = 0; i <= 12; i++)
   {
    minrub[i][j] = 0;
    helpMin[i][j] = 1;
    leftClick[i][j] = 0;
  }
}
void fillMin(int num)
{
  randomize();
  int  count = 0;
  while(count < num)
```

```
  {
    int i = random(10);
    int j = random(20);
    if(minrub[i + 1][j + 1] != 11)
    {
      count++;
      minrub[i + 1][j + 1] = 11;
    }
  }
}
void displayMemo()
{
  int  driver = DETECT,   gdriver = DETECT,  gmode;
  initgraph(&gdriver,  &gmode,  "c:\\tc\\bgi");
  setbkcolor(LIGHTGRAY);
  setcolor(RED);
  settextstyle(DEFAULT_FONT, HORIZ_DIR, 10);
  outtextxy(270, 180, "3");
  delay(3000);
  clrscr();
  outtextxy(270, 180, "2");
  delay(1000);
  clrscr();
  outtextxy(270, 180, "1");
  delay(1000);
  clrscr();
}
void countMinRound()
{
   for(int j = 1; j <= 20; j++)
    for(int i = 1; i <= 10; i++)
     if(minrub[i][j] == 11)
     {
      if(minrub[i - 1][j - 1] != 11) minrub[i - 1][j - 1]++;
      if(minrub[i - 1][j] != 11)   minrub[i-1][j]++;
      if(minrub[i - 1][j + 1] != 11) minrub[i - 1][j + 1]++;
      if(minrub[i][j - 1] != 11)   minrub[i][j - 1]++;
      if(minrub[i][j + 1] != 11)   minrub[i][j + 1]++;
      if(minrub[i + 1][j - 1] != 11) minrub[i + 1][j - 1]++;
      if(minrub[i + 1][j] != 11)   minrub[i + 1][j]++;
      if(minrub[i + 1][j + 1] != 11) minrub[i + 1][j + 1]++;
     }
}
void drawScreen()
{
  int  driver = DETECT,  mode;
  initgraph(&driver,&mode,"c:\\tc\\bgi");
  setbkcolor(BLUE);
  setcolor(8);
  for(int q = 0; q < 5; q++)
   rectangle(9 - q, 20 - q, 629 + q, 368 + q);
  setfillstyle(1, 8);
  floodfill(30, 50, 8);
  for(int p= 0; p < 5; p++)
   rectangle(9 - p, 58 - p, 629 + p, 368 + p);
```

```
  for(int i = 1; i < 11; i++)
   for(int j = 0; j < 20; j++)
   {
    setcolor(LIGHTGRAY);
    rectangle(9+j * 31, 27 + i * 31, 40 + j * 31, 58 + i * 31);
    line(9 + j*31,58+(i-1)*31, 14 + 31 * j, 63 + (i - 1) * 31);
    line(36 + 31*j,84+31*(i-1),40 + 31 * j, 89 + 31 * (i - 1));
    line(14 + 31*j,84+(i-1)*31, 9 + j * 31, 89 + (i - 1) * 31);
    line(36 + j * 31, 63+31*(i-1),40+j* 31, 58 + 31 * (i - 1));
    setcolor(LIGHTGRAY);
    rectangle(14+31*j,63+31*(i-1),36+31* j, 84 + 31 * (i - 1));
    setfillstyle(1, 15);
    floodfill(12 + 31 * j, 59 + 31 * (i - 1), 7);
    setfillstyle(1, 15);
    floodfill(10 + 31 * j, 70 + 31 * (i - 1), 7);
    setfillstyle(1, 7);
    floodfill(20 + 31 * j, 80 + 31 * (i - 1), 7);
    setfillstyle(1, 8);
    floodfill(14 + 31 * j, 85 + 31 * (i - 1), 7);
    setfillstyle(1, 8);
    floodfill(38 + 31 * j, 70 + 31 * (i - 1), 7);
   }
}
void drawSymbol()
{
  setcolor(7);
  rectangle(297,21, 331, 53);
  rectangle(300, 24, 328, 50);
  line(297, 21, 300, 24);
  line(331, 53, 328, 50);
  line(331, 21, 328, 24);
  line(297, 53, 300, 50);
  setfillstyle(1, 7);
  floodfill(302, 26, 7);
  setfillstyle(1, 15);
  floodfill(299, 22, 7);
  floodfill(298, 25, 7);
  setfillstyle(1, 8);
  floodfill(330, 24, 7);
  floodfill(300, 51, 7);
  setcolor(14);
  circle(314, 37, 10);
  setfillstyle(1, 14);
  floodfill(315, 38, 14);
  setcolor(RED);
  for(int p = 0; p < 10; p += 7)
  {
   putpixel(310 + p, 33, 16);
   putpixel(311 + p, 33, 16);
   putpixel(310 + p, 32, 16);
   putpixel(311 + p, 32, 16);
  }
  arc(314, 37, 220, 320, 6);
}
void drawSymbol1()
```

```
{
  setcolor(15);
  circle(314, 37, 10);
  setfillstyle(1, 14);
  floodfill(314, 37, 15);
  setcolor(YELLOW);
  arc(314, 37, 220, 320, 6);
  circle(314, 43, 3);
  setcolor(RED);
  arc(314, 47, 40, 130, 6);
  for(int p = 0; p < 10; p += 7)
  {
   line(309 + p, 30, 312 + p, 35);
   line(309 + p, 35, 312 + p, 30);
  }
}
void drawMin(int x0, int y0, int lineColor)
{
  setcolor(lineColor);
  line(x0, y0 - 8, x0, y0 + 8);
  line(x0 - 8, y0, x0 + 8, y0);
  line(x0 - 5, y0 - 5, x0 + 5, y0 + 5);
  line(x0 + 5, y0 - 5, x0 - 5, y0 + 5);
  setcolor(8);
  circle(x0, y0, 5);
  setfillstyle(1, 8);
  floodfill(x0, y0, 8);
}
void drawFillRectangle()
{
  setcolor(7);
  rectangle(70, 24, 130, 50);
  setfillstyle(1, 7);
  floodfill(79, 27, 7);
}
void drawFillRectangle(int i, int j, int fillColor)
{
  setcolor(BLUE);
  rectangle(9 + 31 * (j - 1), 58 + 31 * (i - 1), 40 + 31 * (j -
    1), 89 + 31 * (i - 1));
  setfillstyle(1, fillColor);
  floodfill(9 + 31 * (j - 1) + 2, 58 + 31 * (i - 1) + 2, 1);
}
void findEL(union REGS outreg, int &e, int &l)
{
 for(int i = 1; i < 11; i++)
  for(int j = 1; j < 21; j++)
  {
    if(9+31*(j-1) <outreg.x.cx && outreg.x.cx < 40+31 *(j - 1))
     if(58+31*(i-1)<outreg.x.dx && outreg.x.dx < 89+31*(i - 1))
     {
      e = i;
      l = j;
     }
   }
```

```
}
void findEL(int i, int j, union REGS outreg, int &e, int &l)
{
  if(9+31*(j - 1) <outreg.x.cx && outreg.x.cx < 40+31 *(j - 1))
   if(58+31*(i-1) < outreg.x.dx && outreg.x.dx < 89+31*(i - 1))
   {
      e = i;
      l = j;
   }
}
void initMouse()
{
  union REGS inreg,outreg;
  inreg.x.ax = 0;
  int86(0x33, &inreg, &outreg);
  inreg.x.ax = 1;
  int86(0x33, &inreg, &outreg);
}
void showMinCount(int num)
{
  setcolor(7);
  rectangle(70, 24, 130, 50);
  setfillstyle(1, 7);
  floodfill(79, 27, 7);
  setcolor(RED);
  itoa(num, timeString1, 10);
  outtextxy(90, 35, timeString1);
}
void showInitTime()
{
  setcolor(7);
  rectangle(500, 24, 560, 50);
  setfillstyle(1, 7);
  floodfill(509, 27, 7);
  setcolor(RED);
  outtextxy(519, 35, "000");
}
void showTime()
{
  setcolor(7);
  outtextxy(519, 35, "000");
  if((t2.ti_min - t.ti_min) * 60 + t2.ti_sec - t.ti_sec >= 100)
  outtextxy(519, 35, timeString3);
  if((t2.ti_min - t.ti_min) * 60 + t2.ti_sec - t.ti_sec <= 10)
   outtextxy(535, 35, timeString3);
  if((t2.ti_min - t.ti_min) * 60 + t2.ti_sec - t.ti_sec >= 10)
   outtextxy(529, 35, timeString3);
  itoa((t2.ti_min - t.ti_min) * 60 + t2.ti_sec - t.ti_sec,
         timeString3, 10);
  setcolor(8);
  for(int q = 0; q < 30; q++)
   line(561, 24 + q, 600, 24 + q);
  setcolor(RED);
  if((t2.ti_min - t.ti_min) * 60 + t2.ti_sec - t.ti_sec < 10)
   outtextxy(519, 35, "00");
```

```
  else if((t2.ti_min-t.ti_min)*60 + t2.ti_sec - t.ti_sec < 100)
  {
   setcolor(7);
   outtextxy(519, 35, "00");
   setcolor(RED);
   outtextxy(519, 35, "0");
  }
  if((t2.ti_min - t.ti_min) * 60 + t2.ti_sec - t.ti_sec >= 100)
  {
   setcolor(7);
   outtextxy(519, 35, "000");
  }
  setcolor(RED);
  if((t2.ti_min - t.ti_min) * 60 + t2.ti_sec - t.ti_sec >= 100)
   outtextxy(519, 35, timeString3);
  else if((t2.ti_min - t.ti_min)*60+ t2.ti_sec - t.ti_sec < 10)
   outtextxy(535, 35, timeString3);
  else
   outtextxy(529, 35, timeString3);
}
void free(int a, int b)
{
  int i;
  if(a < 1 || a > 10 || b < 1 || b > 20)   return;
  char mmm[3], mm[3];
  if(helpMin[a][b] %3 == 1)
  {
    leftClick[a][b]++;
    setcolor(BLUE);
    rectangle(9+31*(b-1),58+31*(a-1),40+31*(b -1),89+31*(a-1));
    setfillstyle(1, 7);
    floodfill(9 + 31 * (b - 1) + 2, 58 + 31 *(a - 1) + 2, 1);
    if(minrub[a][b] != 0)
    {
      itoa(minrub[a][b], mmm, 10);
      for(i=1; i <= 2; i++)
       if(minrub[a][b] == i) setcolor(i);
      if(minrub[a][b] == 3) setcolor(4);
      if(minrub[a][b] == 4) setcolor(13);
      for(i=5; i <= 8; i++)
       if(minrub[a][b] == i) setcolor(i);
      if(minrub[a][b] != 12)
        outtextxy((9 + 31 * (b - 1) + 40 + 31 * (b - 1)) / 2 -
          2,(58+31 *(a - 1) + 89 + 31 * (a - 1)) / 2 - 3, mmm);
      return;
    }
 }   //of if helpMin%3==1
 if(minrub[a][b] != 0 || helpMin[a][b] % 3 !=1 ) return;
 minrub[a][b] = 12;
 free(a, b - 1);
 free(a, b + 1);
 free(a - 1, b - 1);
 free(a - 1, b);
 free(a - 1, b + 1);
 free(a + 1, b - 1);
```

```
    free(a + 1, b);
    free(a + 1, b + 1);
}
```

توابع برنامه و عملکرد آن‌ها

این برنامه از توابع زیر تشکیل شده است:

- **تابع main()**، محیط گرافیکی را بررسی می‌کند و تابع mineSweeper() را فراخوانی می‌نماید.
- **تابع mineSweeper()**، تعداد مین‌ها را از کاربر دریافت کرده، آرایه‌هایی که برای بازی مین روب در نظر گرفته را مقداردهی اولیه می‌نماید (با فراخوانی تابع initArray()، خانه‌های آرایه را با تعداد مین‌ها پر می‌کند (با فراخوانی تابع fillMin()، تعداد مین‌های اطراف هر خانه را محاسبه می‌کند (با فراخوانی تابع countMinRound()، صفحه نمایش را پاک کرده، اگر begin (دقت کنید که متغیر begin از کلاس حافظه static است. چون، باید فقط یک بار تعریف شده، یک بار مقدار اولیه بگیرد و آخر مقدارش را حفظ نماید) برابر یک باشد، برای اولین بار که بازی شروع می‌شود، اعداد ۳، ۲ و ۱ را با جلو ویژه نمایش خواهد داد(با فراخوانی تابع displayMemo()، صفحه مین روب را رسم می‌کند (با فراخوانی تابع drawScreen()، عکس یک آدمک را بر روی صفحه، نمایش می‌دهد (با فراخوانی تابع drawSymbol()، تابع initMouse() را فراخوانی می‌کند تا ماوس را راه‌اندازی کند. با فراخوانی تابع showMinCount(num)، تعداد مین‌ها را بر روی صفحه، نمایش می‌دهد. با فراخوانی تابع showInitTime()، زمان 000 را بر روی صفحه، نمایش خواهد داد. در ادامه با استفاده از یک حلقه تکرار تا زمانی که کاربر کلیدی از صفحه کلید را وارد نکند، بازی ادامه می‌یابد. اگر کاربر بر روی خانه‌ای کلیک کند (کلید سمت چپ ماوس را فشار دهد (outreg.x.bx == 1))، زمان پایان نیافته باشد و اگر خانه فشار داده شده مین نباشد، اطراف آن خالی خواهد شد. اما، اگر خانه کلیک شده مین باشد، تمام خانه‌هایی که مین باشند، با نماد مین نمایش داده می‌شوند و خانه‌ی که بر روی آن کلیک گردید، با زمینه قرمز و نماد مین نمایش داده می‌شود و برای بقیه خانه‌ها تعداد مین‌های اطراف آن نمایش داده خواهد شد. کاربر می‌تواند با کلیک راست خانه‌هایی را علامت‌دار کند. علامت می‌تواند ? یا پرچم باشد.
- **تابع initArray()**، آرایه‌های minrub، helpMin و leftClick را مقداردهی اولیه می‌کند. عناصر آرایه minrub، مقادیر مین‌های موجود در صفحه را تعیین می‌کنند که در حالت اولیه مینی در صفحه وجود ندارد، بنابراین صفر در خانه‌های آرایه قرار می‌گیرد. آرایه leftClick، کلیک‌های انجام شده بر روی خانه‌ها را نگهداری می‌کند. چون، در ابتدا کلیکی انجام نشده است، پس مقادیر خانه‌های این آرایه نیز صفر خواهند بود و آرایه helpClick تعیین می‌کند، خانه مربوطه کلیک راست شده است یا خیر. با کلیک راست بر روی خانه‌ها علامت ? یا پرچم بر روی آن نمایش داده می‌شود. تعداد کلیک راست‌ها، حالت معمولی، علامت سوال(?) یا علامت پرچم تعیین می‌کنند.