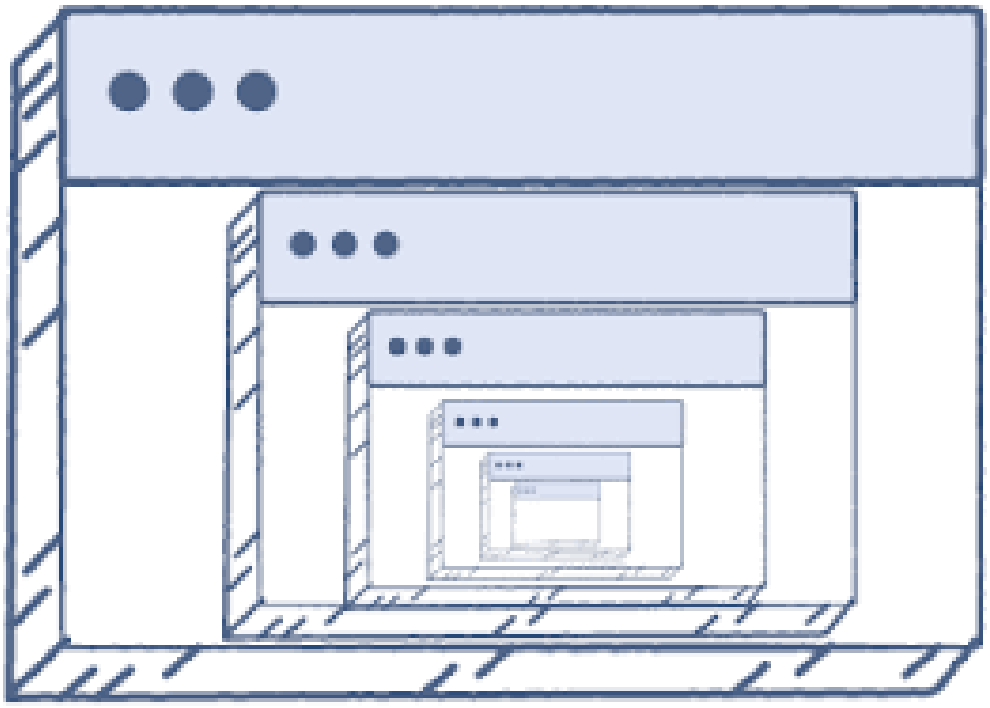
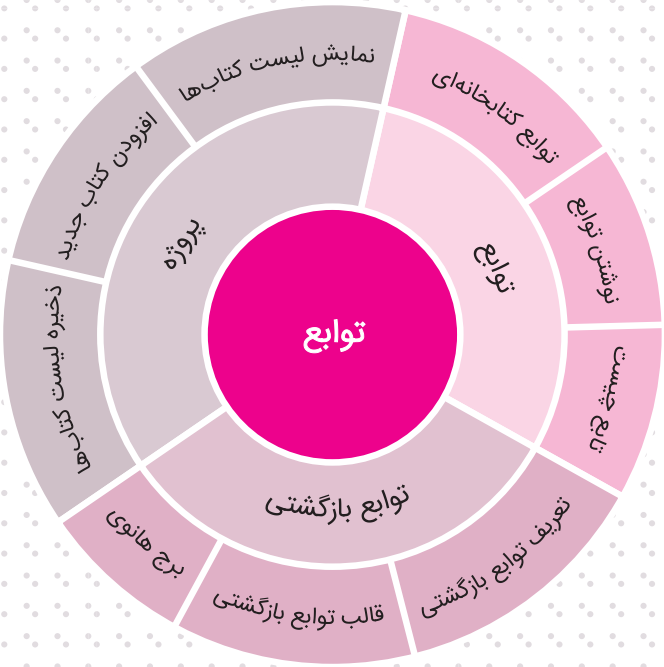


فصل دوم تابع و تابع بازگشتی



اگر این فصل را به خوبی مطالعه کنی و کارهای خواسته شده را به‌دقت انجام دهی:

- با تعریف تابع آشنا می‌شوی و می‌توانی یک تابع بنویسی.
- یاد می‌گیری فراخوانی تابع چگونه انجام می‌شود.
- با توابع کتاب‌خانه‌ای آشنا می‌شوی.
- یاد می‌گیری از تابع بازگشتی استفاده کنی.
- توابع موردنیاز را به پروژه اضافه می‌کنی.

اهداف رفتاری

تابع چیست؟

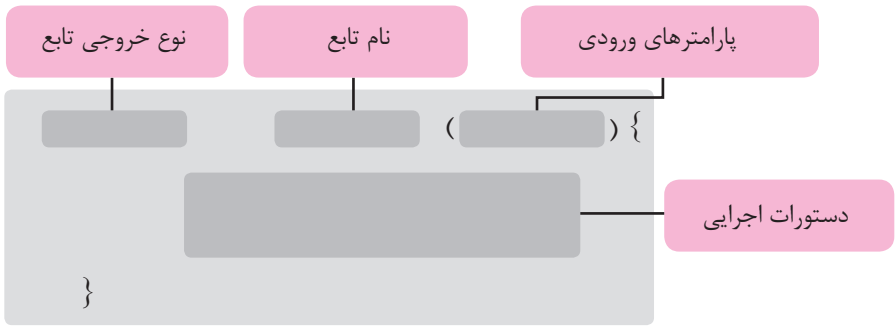
تابع، یک ابزار قدرتمند و مفید در برنامه‌نویسی است که به ما کمک می‌کند برنامه‌های خود را مرتب‌تر و در برخی موارد، با تعداد کمتری دستور بنویسیم. هر تابع، مانند یک دستگاه است که تعدادی ورودی را دریافت می‌کند، روی آن‌ها عملیات خاصی انجام می‌دهد و ورودی‌ها را به خروجی‌های موردنظر تبدیل می‌کند.

استفاده از توابع چطور می‌تواند برنامه ما را مرتب و خوانا کند و یا منجربه کوتاه و مختصر شدن دستورات آن شود؟

چطور یک تابع بنویسیم؟

اولین و مهم‌ترین نکته‌ای که وجود دارد، این است که تابع باید خارج از main نوشته شود. (می‌تواند بالای main باشد یا پایین آن)، همچنین برای نوشتن یک تابع، باید به نام تابع، نوع خروجی و ورودی‌ها و دستورالعمل تابع توجه کنیم و همه این موارد را به‌صورت زیر، در برنامه خود به کار ببریم:

کار با توابع در C++



کلمه return خروجی تابع را به بیرون از تابع؛ یعنی جایی که آن را فراخوانی کرده‌ایم ارسال می‌کند.

تابعی بنویسید که ورودی آن یک عدد صحیح و خروجی، فاکتوریل آن عدد است.

چطور تابع را فراخوانی کنیم؟

برای اینکه از تابع استفاده کنیم، تنها کافی‌است جایی که به آن نیاز داریم، اسمش را صدا کنیم. مثلاً بگوییم برای عدد یا متغیرمان فاکتوریل را حساب کن! باید توجه داشت که وقتی در برنامه یک تابع را صدا بزنیم تعداد ورودی‌هایی که به آن می‌دهیم حتماً برابر تعداد ورودی‌هایی که تابع دریافت می‌کند باشد وگرنه با خطا روبه‌رو خواهیم شد.


```
#include <iostream>
using namespace std;

bool mazrab_5 (int a)
{
    if (a % 5 == 0)
    {
       return 1 ;
    }
       return 0 ;
}

int main()
{
    int x;
    cin >> x;
    if(mazrab_5(x))
        cout << "yes";
    else
        cout << "no";
}
```

تابعی بنویسید که دو عدد را به‌عنوان ورودی بگیرد و حاصل عدد اول به توان عدد دوم را خروجی دهد.

توابع کتابخانه‌ای

توابع کتابخانه‌ای، توابعی هستند که از قبل نوشته شده‌اند و به‌دلیل کارآیی زیادشان، در اختیار همه قرار گرفته‌اند. توابع کتابخانه‌ای زیادی وجود دارند، مثل توابع ریاضی، رندوم، زمان و ... برای اینکه بتوانیم از این توابع استفاده کنیم، باید در ابتدای برنامه، سربرگ موردنیاز را اضافه کنیم تا مجوز استفاده از دستورات برای ما صادر شود!

توابع کتابخانه‌ای کاربردی

در این بخش، برخی از توابع کاربردی را به شما معرفی می‌کنیم:

تابع swap

این تابع دو متغیر از یک نوع را دریافت و مقدار داخلی این دو متغیر را جابه‌جا می‌کند.

```
#include<iostream>
using namespace std;

int main()
{
   int a = 4, b = 5;
   swap(a, b);
   cout << a << ' ' << b;
   return 0;
}
```

خروجی این برنامه به این صورت خواهد بود:

```
5 4
```

پیش از این، به کمک یک متغیر کمکی، برنامه‌ای نوشتیم که این کار را انجام می‌داد. این برنامه حدود ۳ الی ۴ خط اصلی داشت؛ اما با استفاده از تابع swap، می‌توانیم خیلی راحت و سریع به هدفمان برسیم.

تابع min

این تابع دو متغیر از یک نوع را دریافت می‌کند و مقدار متغیر کوچکتر را به‌عنوان خروجی برمی‌گرداند.

```
#include<iostream>
using namespace std;

int main()
{
   double a = 9.32, b = 5.81;
   double c = min(a, b);
   cout << c;
   return 0;
}
```

خروجی این برنامه به این صورت خواهد بود:

```
5.81
```

مجموعه کتاب‌های علامه حلی

برنامه‌نویسی سی پلاس پلاس ۳

• لادن جاماسبی
• عطیه پوردرخشان

شناسنامه کتاب

عنوان کتاب	برنامه‌نویسی سی‌پلاس‌پلاس ۳
ناشر	انتشارات حلی
ناشر همکار	انتشارات دانش‌پژوهان جوان
مؤلفان	عطیه پوردرخشان، لادن جاماسبی پویان علی‌پناهی
ویراستار علمی	محمدرضا معتبر
مسئول هماهنگی	سمیه‌سادات فاطمی
صفحه‌آرا	راضیه فرهانیان
طراح جلد	زهره شیروانی‌هرندی
تصویرساز	محمدحسین صفدریان
سال چاپ	۱۴۰۲
نوبت چاپ	اول
چاپ و صحافی	واژه‌پرداز اندیشه
شمارگان	۲۰۰۰ جلد
قیمت	۱۰۴۰۰۰ تومان
شماره شابک	۹۷۸-۶۰۰-۴۹۶-۲۸۸-۹

بهترین روش برای یادگیری عمیق و کاربردی برنامه‌نویسی، دست به کد شدن و حل تمرین‌های متنوع است. مخصوصا اگر تمرین‌ها در کمترین لحظه ممکن صحیح شوند و بتوانیم درستی و نادرستی آن را در کوتاه‌ترین زمان متوجه شویم و سعی کنیم اشکالات خود را برطرف کنیم. سامانه آموزش برنامه‌نویسی کوئرا، مجهز به داوری خودکار سوالات و تمرین‌هاست و به کمک آن می‌توانید از صحت سوالات حل شده خود مطلع شوید.

برای شروع به سایت Quera.org بروید، یک حساب کاربری برای خود باز کنید و با اسکن این بارکد، به صفحه اصلی کتاب وارد شوید و با انتخاب فصل مورد نظر، تمرین خود را انتخاب و پاسخ آن را ارسال کنید.

تهران، خیابان انقلاب، میدان فردوسی، ابتدای کوچه براتی، پلاک ۱۶ واحد ۱۴

تلفن دفتر مرکزی: ۹-۶۶۷۴۴۳۸۴

جالب است براتی

فهرست مطالب

قبل از شروع به مطالعهٔ کتاب، این قسمت را بخوانید:

وقتی شروع به خواندن این کتاب کنید با بخش‌های مختلفی مواجه می‌شوید که غالباً یک لاک‌پشت متفاوت در اول هرکدام وجود دارد. برای هرکدام از این بخش‌ها از شما انتظار داریم کار متفاوتی انجام دهید. این قسمت‌ها بر اساس تئوری‌های نوین آموزش و تجارب موفق تدریس برای آموزش دانش‌آموزان مستعد طراحی شده است. این بخش‌ها شامل:

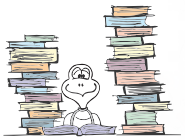
درخت دانش: در صفحه دوم هر فصل، نمودار دایره‌ای شکلی کشیده شده که به ما کمک می‌کند بفهمیم در آن فصل مطالب علمی چطوری تقسیم‌بندی شده و ارتباط آن‌ها با هم چیست. درواقع این بخش نقشه‌ای است برای گم نشدن در موضوعات علمی.

اهداف رفتاری: زیر هر درخت دانش، چند جمله نوشته شده که از اول کار معلوم کند که این فصل را می‌خوانیم که چه بشود. خوب است در آخر فصل هم برگردیم و ببینیم که می‌توانیم کارهایی را که در این بخش گفته انجام دهیم یا نه.

پاسخگو باش: در این قسمت باید پاسخگو باشیم. پاسخگوی سؤالی که پرسیده شده و انتظار می‌رود بعد از خواندن درس تا آن قسمت، بتوانیم باکمی فکر کردن به آن جواب دهیم.

فسفر بسوزان: شاید لازم باشد مقدار بیشتری از مغز خودمان استفاده کنیم و قدری از فسفرهای ذخیره‌شده را بسوزانیم! سؤالاتی که در بخش فسفر بسوزان مطرح می‌شود فقط با خواندن مطالب درسی قابل پاسخگویی نیست و باید کمی بیش از معمول درباره آن‌ها فکر کنیم.

جالب است بدانی: برای افرادی که دوست دارند بیشتر از سطح استاندارد با موضوعات آشنا شوند این قسمت توصیه می‌شود. در این قسمت مطالبی آورده شده که خواندن و یادگرفتن آن الزامی نیست ولی آن‌قدر جذاب است که نشود به‌راحتی بی‌خیال خواندن آن شد.



لغت‌نامه: ما دانش‌آموزان مستعد و متفاوت (!) دوست داریم بتوانیم علاوه‌بر مطالب درسی، جست‌وجویی هم بکنیم و ببینیم در دنیا درباره موضوع درسی ما چه چیزی وجود دارد. برای همین در پایان هر فصل لغات مهم فصل با معادل انگلیسی آن آورده شده است.

تمرین‌ها: در آخر هر فصل تمرین‌های مرتبط با آن آورده شده است. از آنجایی‌که مؤلفان کتاب از دبیران باسابقه هستند پس تعداد تمرین‌ها، وقت لازم برای انجام آن‌ها، تعداد سؤالات سخت و آسان و نوع سؤالات با برنامه و محاسبه تعیین شده است. پس خیالتان راحت باشد که همه تمرین‌ها را در طول سال می‌شود انجام داد. تمرین‌ها بر اساس موضوعات هر فصل بخش‌بندی شده، بنابراین لازم نیست برای تمرین منتظر پایان فصل باشید؛ در پایان هر مبحث می‌توانید به بخش تمرین‌ها مراجعه کنید و تمرین‌های همان مبحث را حل کنید.

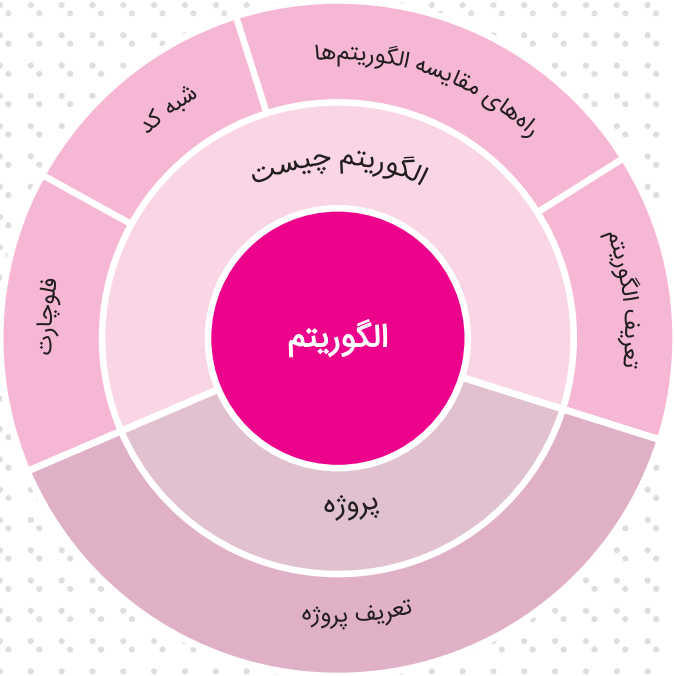
دست‌به‌کد شو: برنامه‌نویسی درسی کاربردی است که در حین آموزش آن لازم است شما هم دست به کد بشوید. در بخش دست به کد شو از شما خواسته شده تا سعی کنید خودتان برنامه را بنویسید. حواستان باشد این بخش، قسمت مهمی از روند درسی است و نمی‌شود بدون دست‌به‌کد شدن برنامه نویسی یاد گرفت.

اشتباه رایج: همان‌طور که از اسمش مشخص است، در این قسمت اشتباهاتی که ممکن است برای هرکسی پیش آید را برای شما توضیح داده‌ایم تا شما دیگر آن‌ها را تکرار نکنید! می‌توان گفت، اگر قرار باشد در بخش‌های دیگر راه برنامه‌نویسی را یاد بگیرید، در این قسمت با چاه‌های آن آشنا می‌شوید.

چه می‌کنه: در این قسمت، یک برنامه کامل برای شما نوشته‌ایم و از شما انتظار داریم بگویید این برنامه برای چه هدفی نوشته شده، چه کاری انجام می‌دهد و برای ورودی‌های مختلف، چه خروجی تولید می‌کند.

در ضمن شما هم می‌توانید برای ما مطالب و مسئله ارسال کنید! مطالب و مسئله‌هایی که خودتان از آن‌ها لذت برده‌اید به آدرس: ketab.helli@gmail.com

فصل صفر الگوریتم و پروژه



اگر این فصل را به خوبی مطالعه کنی و کارهای خواسته شده را به‌دقت انجام دهی:

- با الگوریتم آشنا می‌شوی.
- یاد می‌گیری چگونه الگوریتم‌ها را به صورت شبه‌کد بنویسی.
- یاد می‌گیری چگونه الگوریتم‌ها را با استفاده از فلوچارت بنویسی.
- با پروژه‌ای که قرار است در طول این کتاب بنویسی آشنا می‌شوی.

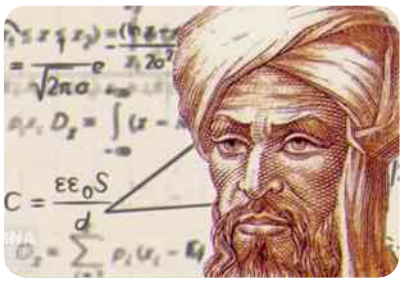
الگوریتم و پروژه

تاکنون با دستورات مختلف برنامه‌نویسی آشنا شده‌اید و برنامه‌های متنوعی نوشته‌اید پس طبیعی است که بارها و بارها کلمه «الگوریتم» را شنیده و حتی از آن استفاده کرده باشید! اما این «الگوریتم» دقیقاً چیست و چه کاربردهایی دارد؟ چه ویژگی‌هایی دارد و چه زمانی می‌گوییم یک الگوریتم، مفید است؟ در این فصل می‌خواهیم کمی از الگوریتم بگوییم و با فرایند حل مسئله بیشتر آشنا شویم.

الگوریتم چیست؟

الگوریتم، مجموعه‌ای از دستورات و اعمالی است که به‌ترتیب و پشت‌سرهم انجام می‌شوند تا مسئله را به‌درستی حل کنند. حالا این مسئله می‌تواند یکی از مسائل ساده و روزمره در زندگی مثل نوشیدن آب، سفارش غذا از یک اپلیکیشن یا روشن کردن یک دستگاه باشد یا مسئله پیچیده‌ای در علوم مختلف مثل هوش مصنوعی و علوم داده‌ها!

ابوجعفر محمد بن موسی خوارزمی، دانشمند و ریاضی‌دان قرن دوم هجری شمسی در دوره مامون عباسی بوده است. در آن زمان، زبان علمی، عربی بود و به همین علت، او را به‌صورت الخوارزمی صدا می‌زدند.



خوارزمی، در حل مسائل و استفاده از عملیات جمع و ضرب و تقسیم و تفریق، روش متفاوتی داشت و سعی می‌کرد مسئله‌ها را به‌صورت مرحله به مرحله و دقیق حل کند. اروپاییان از این شیوه حل مسئله بسیار شگفت‌زده شدند و این روش بسیار موردتوجه آن‌ها قرار گرفت. به همین خاطر، هر روشی که حل مسائل را به‌صورت مرحله به مرحله و دقیق و با جزئیات کافی بیان کند و در پایان به جواب درست برسد را روش الگوریتمی نام‌گذاری کردند. تلفظ لاتینی الخوارزمی به‌صورت Algorism یا الگوریتمی Algorithmic است.

برای آماده کردن صبحانه، چه الگوریتمی به ذهنتان می‌رسد؟

از آنجایی که کتاب پیش روی شما، کتاب آموزش برنامه‌نویسی و رایانه است، به مثال‌های روزمره نمی‌پردازیم و کم‌کم وارد فضای حل مسئله به کمک رایانه می‌شویم! یکی از مهم‌ترین دلایلی که رایانه اختراع شد، سرعت و دقت محاسبات رایانه در مقایسه با انسان بود. از طرفی انسان تمایل دارد کارهای تکراری و پیچیده را انجام ندهد و به دنبال پیدا کردن راهکاری است که این عملیات، خودبه‌خودی و توسط دستگاه‌ها و رایانه‌ها انجام شود؛ اما نکته مهم و اساسی که وجود دارد این است که رایانه بر خلاف انسان، نمی‌تواند هر دستوری را بفهمد. چه برسد به اینکه بتواند آن را انجام دهد! برای مثال، اگر به رایانه بگوییم «برو از یخچال آب بیار»، می‌توانیم مطمئن باشیم که آبی به دست ما نمی‌رسد! چون دستوراتی که داریم به آن می‌دهیم، مثل «رفتن» و «آب آوردن» برایش معنا و مفهوم ندارد و تعریف نشده است. درواقع باید سعی کنیم با دستوراتی که برای رایانه قابل فهم و درک است، به او بگوییم کاری را برایمان انجام دهد. از طرفی می‌دانیم رایانه‌ها ورودی‌ها را دریافت می‌کنند، روی آن‌ها پردازش می‌کنند و اطلاعات پردازش‌شده را به شکل خروجی به ما می‌دهند. پس بیایید یک‌بار دیگر به تعریف الگوریتم نگاه کنیم و آن را به تعریفی که برای رایانه مناسب است تبدیل کنیم:

الگوریتم، مجموعه‌ای از دستورات قابل‌فهم برای رایانه هستند که به‌ترتیب و پشت‌سرهم اجرا می‌شوند تا ورودی‌ها را به خروجی موردنظر ما تبدیل کنند.

با خواندن و مواجه‌شدن با هر مسئله، ممکن است یک‌سری راه‌حل به ذهنمان برسد. مثلاً می‌دانیم برای اینکه بفهمیم سه عدد، می‌توانند تشکیل مثلث دهند، باید قضیه حمار استفاده کنیم. (در قضیه حمار، هر ضلع از

مجموع دو ضلع دیگر کوچک‌تر است)؛ حالا بیایید با سلام و صلوات قضیه حمار را به رایانه بفهمانید! پس چاره چیست؟ چاره کار این است که سعی کنیم تمام چیزهایی که در مغزمان می‌گذرد را به متغیرها، نمادها و عملگرها، به شکل قابل‌فهم برای رایانه تبدیل کنیم. به این کار مدل‌سازی ریاضی هم می‌گویند! مثلاً برای اینکه بگوییم طول هر ضلع از جمع دو ضلع دیگر کوچک‌تر است، به این شکل عمل می‌کنیم:

```
a + b > c & a + c > b & b + c > a
```

به قول یکی از معلم‌های ریاضی، به تعداد آدم‌ها راه برای رسیدن به خدا وجود دارد پس، اگر برای حل یک مسئله چند راه وجود داشته باشد، اتفاق عجیبی نیفتاده است! شما هم حتماً با این مورد مواجه شده‌اید که یک سوال را با چندین راه‌حل دیده باشید. برای برنامه‌ها هم ممکن است چندین الگوریتم وجود داشته باشد؛ اما همه ما می‌دانیم بعضی راه‌حل‌ها یا الگوریتم‌ها از بعضی دیگر بهتر هستند. به‌نظر شما چه فاکتورها و مواردی وجود دارد که باعث می‌شود یک الگوریتم از الگوریتم دیگر بهتر باشد؟

۱. درست بودن: یکی از مهم‌ترین مواردی که وجود دارد، این است که راهکار و الگوریتم، ما را به برنامه و پاسخ درست برساند.

۲. زمان اجرا: هر برنامه یا الگوریتم، در یک زمان مشخصی اجرا می‌شود. میزان زمان اجرای الگوریتم‌ها به عواملی مانند تعداد دستورات نوشته‌شده و پیچیدگی دستورات بستگی دارد. منطقی هم هست! یک نامه ۱۰ خطی زمان کمتری برای مطالعه از ما می‌گیرد تا یک نامه ۲۰ صفحه‌ای! درواقع می‌توان گفت الگوریتمی سریع‌تر است که تعداد عملیات کمتر و پیچیدگی کمتری داشته باشد.

۳. حافظه استفاده شده: می‌دانیم متغیرها، بخش‌هایی از حافظه هستند که اطلاعات موردنیاز ما را در خود ذخیره می‌کنند پس منطقی است هر متغیری که در برنامه و الگوریتم خود استفاده می‌کنیم، قسمتی از حافظه را اشغال کند. هر چقدر تعداد متغیرهای ما بیشتر باشد، برنامه حافظه بیشتری را مصرف می‌کند و برعکس، هر چقدر تعداد متغیر کمتری را استفاده کنیم، حافظه کمتر اشغال می‌شود.


کنکاش کن

الگوریتم‌ها به‌جز ریاضی و برنامه‌نویسی، در خیلی از علوم دیگر مثل روانشناسی، اقتصاد و ... کاربرد دارند. با جست‌وجو در اینترنت، کاربردهای آن در علوم مختلف را بررسی کنید و یک نمونه الگوریتم کاربردی بیان کنید.

نوشتن الگوریتم

برای نوشتن الگوریتم، روش‌های مختلفی وجود دارد. رسم فلوچارت یا روندنما و نوشتن شبه کد، از جمله روش‌های پرکاربرد و قابل استفاده در دنیای برنامه‌نویسان است:

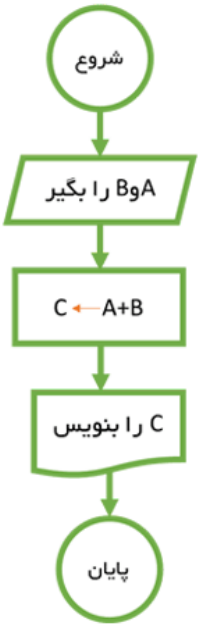
۱. شبه کد: در این روش، ابتدا اهداف نوشتن الگوریتم نوشته می‌شود سپس زیر آن خط کشیده می‌شود و در ادامه، با دستورات شماره‌گذاری شده که شامل گرفتن ورودی، استفاده از شرط و حلقه و ... است، الگوریتم نوشته می‌شود. برای مثال، در شبه کد زیر، دو عدد از کاربر گرفته می‌شود و حاصل‌جمع آن دو در خروجی نمایش داده می‌شود:

```
get three numbers 'a' , and 'b'
and output their sum
-----------------------------------
1. input 'a' and 'b'
2. d <- a + b
3. output 'd'
```

شبه کد زیر را بررسی کنید و بگویید خروجی آن چیست؟

```
Input 'n' and then output the character 'a' n-times
-----------------------------------------------------
1. counter <- 0
2. while counter < n do
3.      output 'a'
4.      counter <- counter + 1
```

۲. فلوچارت یا روندنما: روشی است که در آن از یک‌سری نماد استفاده می‌کنیم تا به کمک آن‌ها بتوانیم الگوریتم خود را به‌صورت تصویری نمایش دهیم. برای مثال برای نمایش جمع دو عدد در خروجی، به شکل زیر عمل می‌کنیم:



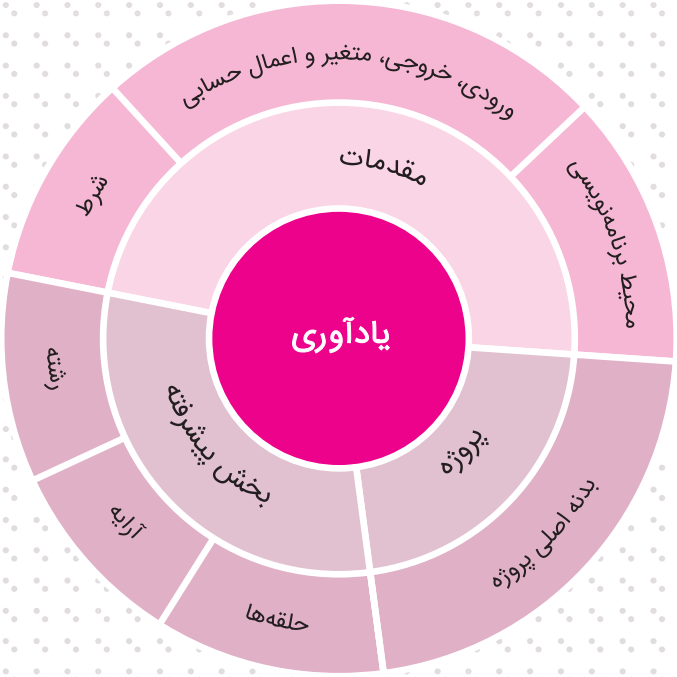
در مورد فلوچارت و نحوه رسم کردن آن تحقیق کنید.

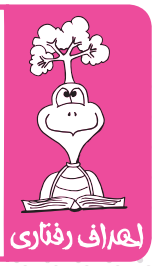
در دنیای امروز، برنامه‌های متنوعی توسط برنامه‌نویسان ارائه می‌شوند و ما هم از آن‌ها به شکل گسترده استفاده می‌کنیم. برنامه‌های سفارش غذا، دستیار معلم برای جمع‌بندی و دسته‌بندی نمرات و اعلام آن به دانش‌آموزان، پیام‌رسان‌ها، سامانه‌های پرداخت قبض‌ها و ... همه از برنامه‌هایی هستند که روزانه به‌صورت گسترده از آن‌ها استفاده می‌شود. برای نوشتن هرکدام از این برنامه‌ها، ممکن است صدها خط کد نوشته شود و تیم‌های مختلفی روی آن کار کنند. چیزی که خیلی مهم است، این است که بتوانیم ابتدا بفهمیم دنیای امروز به برنامه ما چه نیازی دارد یا برعکس، چه نیازی وجود دارد و برنامه ما قرار است آن را برطرف کند. در گام بعدی، باید ببینیم برای رفع آن نیاز، برنامه ما باید به چه صورتی نوشته شود. درواقع باید مسئله را به چند زیر مسئله تبدیل کنیم و برای هر زیر مسئله، راهکار و الگوریتم و کد مربوط را بنویسیم و درنهایت بررسی کنیم ببینیم آیا برنامه و الگوریتم ما در مرحله اول درست و در مرحله بعدی بهینه است یا نه! در این کتاب، می‌خواهیم نحوه نوشتن یک پروژه واقعی را با هم تمرین کنیم! این پروژه که کمی جلوتر آن را معرفی می‌کنیم، ابتدا به چند زیر پروژه تقسیم می‌شود و در فصل‌های پیش‌رو، زیرمسئله‌ها و زیرپروژه‌ها بررسی می‌شوند و مفاهیم موردنیاز آموزش داده می‌شود. با مطالعه این مفاهیم و بررسی دقیق مثال‌ها، می‌توانید اطلاعات موردنیاز برای نوشتن پروژه را دریافت کنید سپس به بررسی پروژه بپردازید. در پایان این کتاب، شما یک پروژه کاملاً کاربردی را اجرا کرده‌اید و تفکر خلاقانه، تفکر رایانشی و الگوریتمی را تجربه خواهید کرد.

و اما پروژه این کتاب!

در این کتاب می‌خواهیم برنامه سیستم مدیریت یک کتاب‌فروشی را پیاده‌سازی کنیم که می‌تواند اطلاعات کتاب‌های کتاب‌فروشی را ذخیره کند، آن‌ها را بر اساس حروف الفبا یا قیمت مرتب کند و در صورت جست‌وجوی نام یک کتاب، اطلاعاتی مانند تعداد موجودی و قیمت آن را نمایش دهد، همچنین زمانی که کتابی خریداری می‌شود از موجودی آن کم کند. در هر فصل پس از یادگیری مطالب آن، بخشی از این پروژه را تکمیل خواهیم کرد تا در پایان کتاب به یک پروژه کامل سیستم کتاب‌فروشی تبدیل شود.

فصل اول مفاهیم پایه





اگر این فصل را به خوبی مطالعه کنی و کارهای خواسته شده را به‌دقت انجام دهی:

- مباحث گذشته شامل آغاز برنامه‌نویسی به زبان C++، ورودی، خروجی، متغیر، اعمال حسابی، شرط، حلقه‌ها، آرایه و رشته را مرور می‌کنی.
- نوشتن پروژه سیستم کتاب‌فروشی را آغاز خواهی کرد.

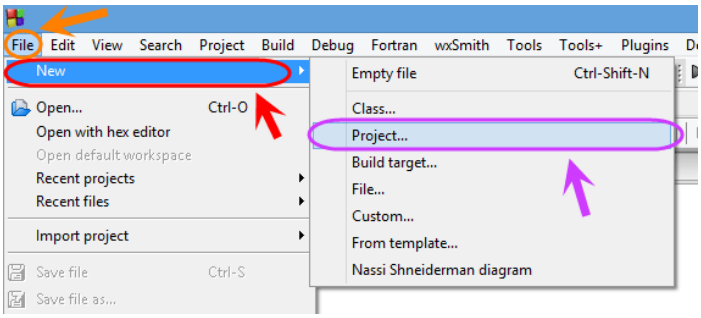
محیط برنامه‌نویسی، ایجاد پروژه و آغاز برنامه به زبان C++

برای نوشتن یک برنامه به زبان C++ به یک IDE قابل‌استفاده برای این زبان نیاز داریم که شامل ویرایشگر کد برای نوشتن دستورات، کامپایلر یا مفسر برای ترجمه دستورات نوشته شده به زبان کامپیوتر (زبان صفر و یک) و اصلاح‌کننده کد که در رفع اشکالات برنامه به ما کمک می‌کند و خطاهای موجود در برنامه را برایمان مشخص خواهد کرد. در این کتاب از Code::Blocks به‌عنوان IDE استفاده می‌کنیم و به شما یاد می‌دهیم چطور در این فضا برنامه بنویسید. برای شروع هم به شما یاد می‌دهیم چطور در این محیط یک پروژه ایجاد کنید.

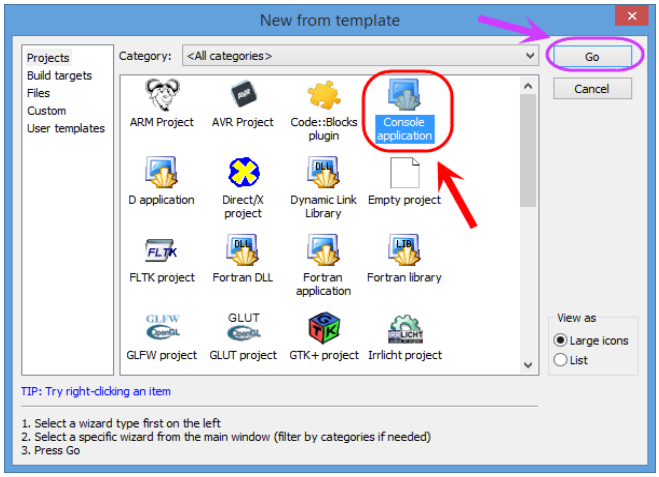
مراحل ایجاد پروژه:

۱. ابتدا نرم افزار Code::Blocks را اجرا می‌کنیم.

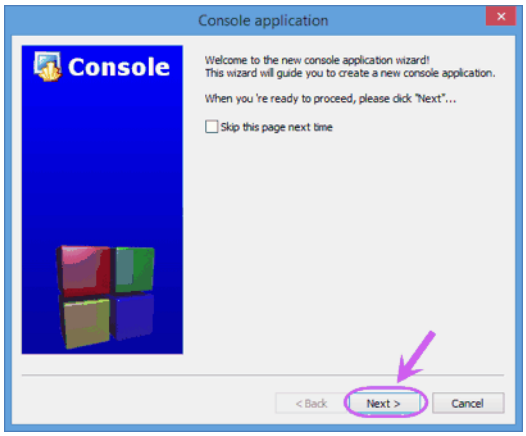
۲. برای ساخت یک پروژه جدید، ابتدا از منوی File، گزینه New سپس گزینه Project... را انتخاب می‌کنیم:



۳. با این کار، با پنجره زیر مواجه می‌شویم. گزینه console application را انتخاب کرده و Go را می‌زنیم:

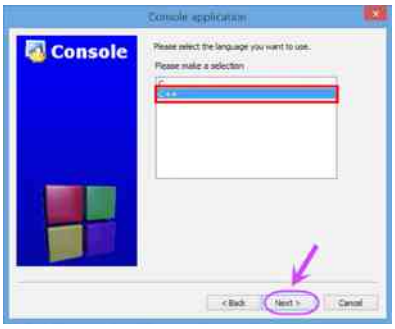


۴. با انجام کارهای گفته شده، به این مرحله می‌رسیم. روی گزینه Next کلیک کنید:

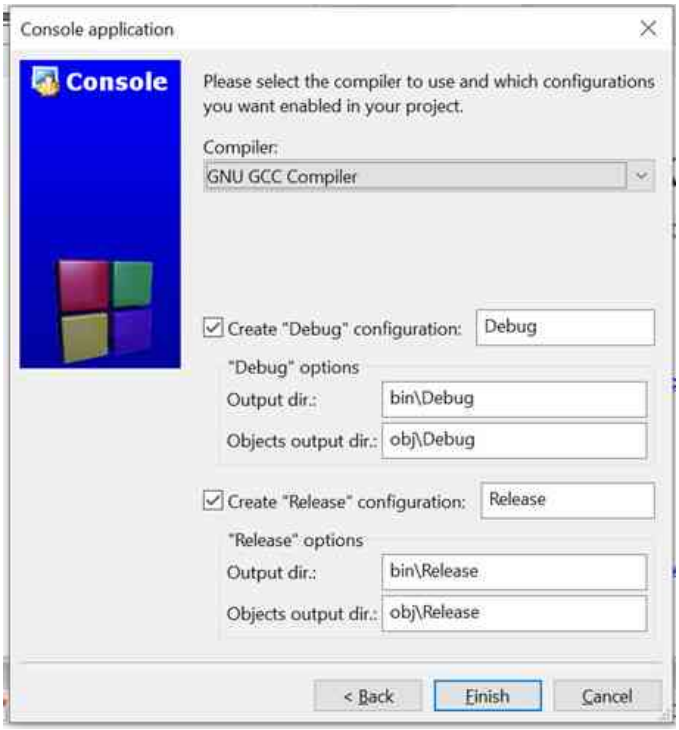


۵. مشخص است که باید زبان را C++ انتخاب کنید و گزینه Next را بزنید!

۶. حال باید یک نام برای پروژه انتخاب کنیم و محل ذخیره‌سازی آن را مشخص کنیم. توصیه می‌کنیم نامی برای پروژه خود انتخاب کنید که مفهوم و معنا داشته باشد و هم‌راستا با کار اصلی پروژه باشد.



۷. در مرحله بعد باید کامپایلر را انتخاب کنیم تا مراحل پیاده‌سازی تمام شود و پروژه جدید را بسازیم.



حالا باید دست به کد شویم و در این پروژه ایجاد شده، کد بنویسیم. برای شروع کدنویسی به زبان C++ قطعه کد زیر را ببینید:

```
#include <iostream>
using namespace std;

int main()
{
        دستورات
 }
```

در خطوط ابتدایی هر برنامه به زبان C++ یک یا چند include نوشته شده است. این خطوط نشان‌دهنده نام کتابخانه‌هایی هستند که ما می‌خواهیم از دستورات آن‌ها استفاده کنیم. پس از نوشتن تمام کتابخانه‌های موردنظر برای برنامه نوبت نوشتن عبارت using namespace std است که باید آن را در تمام برنامه‌هایمان داشته باشیم.

مهم‌ترین بخش برنامه، در int main اتفاق می‌افتد و باید اصل دستوراتمان را در این قسمت بنویسیم.

ورودی، خروجی، متغیر و اعمال حسابی

خروجی، نتیجه و حاصل برنامه است که توسط ابزارهای خروجی مثل نمایشگر، چاپگر و ... نمایش داده می‌شود. برای اینکه بتوانیم یک عبارت را در خروجی نمایش دهیم، کافی است به شکل زیر عمل کنیم.

```
cout << عبارت موردنظر;
```

متغیرها ظرف‌هایی از حافظه هستند که اطلاعاتی مانند اعداد، حروف و ... را در خود ذخیره می‌کنند. این ظروف، ظرفیت محدود و مشخصی دارند و به کمک نامی که برای آن‌ها انتخاب می‌کنیم، می‌توانیم از آن‌ها استفاده کنیم.

با چند نوع مهم از متغیرها در سی‌پلاس‌پلاس آشنا شدیم:

- اعداد صحیح (int)
- اعداد اعشاری (double)
- کاراکتر (char)
- رشته (string)

به اطلاعاتی که توسط کاربر دریافت شده و از آن در برنامه خود استفاده می‌کنیم، ورودی می‌گوییم. برای اینکه بتوانیم یک عدد، حرف یا کلمه را از کاربر بگیریم، ابتدا باید یک متغیر از آن جنس تعریف کنیم و بعد با کمک دستور زیر، از کاربر ورودی موردنظر را دریافت کنیم:

```
cin >> نام متغیر ;
```

به‌طور کلی به عملیات اصلی ریاضی؛ یعنی جمع، تفریق، تقسیم و ضرب، چهار عمل اصلی می‌گوییم. عملگرهای ریاضی در C++ به‌صورت زیر استفاده می‌شوند:

عملیات	جمع	تفریق	ضرب	تقسیم
نحوه نوشتن با متغیرهای a و b	a+b	a-b	a*b	a/b

پیدا کردن باقی‌مانده تقسیم، به کمک یک عملگر خاص انجام می‌شود. برای به‌دست آوردن باقی‌مانده تقسیم دو عدد صحیح، کافی‌است از نماد ٪ استفاده کنیم.

عملیات باقی‌مانده	برای دو متغیر صحیح a و b
٪	a٪b

برنامه‌ای بنویسید که دو عدد در ورودی دریافت کرده و حاصل ضرب، حاصل جمع، تفریق و تقسیم آن‌ها را در خروجی چاپ کند.

شرط

گاهی نیاز است دستوری را در حالت خاص و در صورت برقرار بودن یک شرط اجرا کنیم. در این صورت از دستور زیر استفاده می‌کنیم:

```
if (شرط۱)
  {
      دستور۱;
      دستور۲;
      دستور۳;
      ...
  }
  else if (شرط۲)
  {
      دستور۱;
      دستور۲;
      دستور۳;
      ...
  }
  else
  {
      دستور۱;
      دستور۲;
      دستور۳;
      ...
  }
```

اگر شرط‌های بیشتری داشتیم، می‌توانیم برای هر if به تعداد موردنیاز else if داشته باشیم.

برای نوشتن شرط‌ها به آشنایی با عملگرهای مقایسه‌ای نیاز داریم:

در ریاضی	در C++
>	>
<	<
=	==
	=!
	=>
	=<

برنامه‌ای بنویسید که یک عدد را به‌عنوان نمره نهایی درس ریاضی از کاربر بگیرد، اگر این عدد بیشتر یا مساوی ۱۷ بود، عبارت aali، اگر کمتر از ۱۷ و بیشتر یا مساوی ۱۰ بود، عبارت ghabool و در غیر این صورت عبارت mardood را در خروجی نمایش دهد.


```
#include <iostream>
using namespace std;

int main()
{
    char ch;
    int a, b;
    cin >> a >> ch >> b;
    if (ch == '+')
    {
        cout << a + b << endl;
    }
    else if (ch == '-')
    {
        cout << a - b << endl;
    }
    else if (ch == '*')
    {
        cout << a * b << endl;
    }
    else
    {
       cout << a / b << endl;
    }
}
```

به کمک عملگرهای منطقی، می‌توانیم به‌راحتی شرط‌ها را با هم ترکیب کنیم. سه عملگر منطقی (و)، (یا) و (نقیض یا نه) را معرفی می‌کنیم:

وَ:

فرض کنید P و Q دو شرط باشند، مقدار P && Q در صورتی درست است که هر دو شرط درست باشند؛ یعنی هم P و هم Q درست باشد، وگرنه درست نیست.


```
#include <iostream>
using namespace std;

int main()
{
    int a, b, c;
    cin >> a >> b >> c;
    if ((a > 0) && (b > 0) && (c > 0) && (a + b + c == 180))
    {
        cout << "yes" << endl;
    }
    else
    {
        cout << "no" << endl;
    }
}
```

یا:

فرض کنید P و Q دو شرط باشند، مقدار P || Q در صورتی درست است که حداقل یکی از دو شرط درست باشند؛ یعنی یا P یا Q و یا هر دو درست باشند، وگرنه درست نیست.


```
#include <iostream>
using namespace std;

int main()
{
    int a, b, c;
  cin >> a >> b >> c;
  if ((a * a + b * b == c * c) || (b * b + c * c == a * a) || (a * a + c * c == b * b))
        cout << "Yes" << endl;
  else
        cout << "No" << endl;
}
```

نه:

این عملگر منطقی، اگر پشت هر شرطی بیاید، آن را برعکس می‌کند، اگر درست باشد، آن را نادرست، اگر نادرست باشد، درستش می‌کند. درواقع، اگر P درست باشد، نادرست است، اگر P نادرست باشد، درست خواهد بود.


```
#include <iostream>
using namespace std;

int main()
{
      int a, b;
      cin >> a >> b;
      if (!(a  == b))
           cout << "No" << endl;
     else
           cout << "Yes" << endl;
}
```

حلقه‌ها

انجام کارهای تکراری برای انسان طاقت‌فرسا، خسته‌کننده و همراه با خطا خواهد بود؛ اما کامپیوترها این کار را درست و سریع انجام می‌دهند. برای اینکه بتوانیم کارهای تکراری را به کامپیوتر بسپاریم، از حلقه‌های تکرار استفاده می‌کنیم. دو حلقه while (حلقه شرطی) و for (حلقه شمارشی) داریم که در ادامه آن‌ها را توضیح می‌دهیم:

While:

حلقه while، دستوری است که می‌توان به کمک آن، دستورات را تا زمانی‌که شرطی برقرار باشد اجرا کرد. برای اینکه بتوانیم از این دستور استفاده کنیم، کافی‌است مانند نمونه، آن را در برنامه خود به‌کار ببریم تا بتوانیم دستوراتمان را با شرط دلخواه، تکرار کنیم.

```
While (شرط)
{
      ۱ دستور;
      ۲ دستور;
      ۳ دستور;
        ...
}
```

برنامه‌ای بنویسید که یک عدد طبیعی از کاربر بگیرد و مقلوب آن را در خروجی نمایش دهد. مقلوب عدد ۱۲۳، عدد ۳۲۱ است.

for:

به کمک حلقه for می‌توانیم یک‌سری دستورات را به تعداد معین تکرار کنیم. برای این‌کار به گام، شمارنده و مقدار اولیه نیاز داریم. با استفاده مناسب از این ابزارها و دستور زیر، می‌توانیم برنامه خود را بهینه‌تر کنیم:

دست به کار شو

```
for (گام حرکت ; شرط ادامه حلقه ; شمارنده با مقدار اولیه)
{
    دستور تکرار شونده;
}
```

دست به کار شو

برنامه‌ای بنویسید که ابتدا n سپس n عدد از کاربر می‌گیرد و میانگین آن اعداد را در خروجی نمایش می‌دهد.

چند نکته:

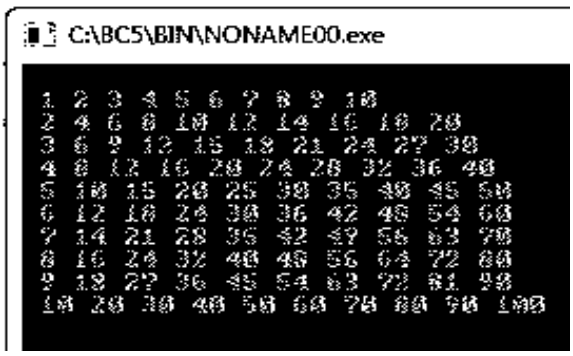
- به کمک جدولی به اسم جدول تعقیب، می‌توانیم فرایند پیشرفت حلقه و دستوراتش را رصد کنیم. در این جدول، برای هر متغیری که در حلقه استفاده شده، شمارنده‌ها و خروجی (در صورتی که دستور خروجی داخل حلقه باشد) یک ستون ایجاد می‌کنیم و در هر مرحله، تغییراتش را ثبت می‌کنیم.
- اگر در یک حلقه تکرار به دستور break برسیم کار حلقه همان‌جا تمام شده و برنامه از حلقه خارج می‌شود، همچنین اگر در یک حلقه تکرار به دستور continue برسیم، ادامه دستورات پس از آنکه در حلقه موجود هستند اجرا نشده و اجرای حلقه به مرحله یا دفعه بعدی خود می‌رود و شرط حلقه چک می‌شود.
- می‌توانیم یک حلقه را به‌عنوان دستور حلقه دیگر بنویسیم و به اصطلاح حلقه تودرتو بسازیم. در این صورت، هرکدام از دستورات حلقه داخلی، به تعداد دفعات تکرار حلقه بیرونی تکرار می‌شود.

پاسخگو باش

اگر در یک حلقه تودرتو، حلقه بیرونی ۵ بار و حلقه درونی ۸ بار تکرار شوند، دستورات حلقه درونی چند بار تکرار می‌شوند؟

دست به کار شو

برنامه‌ای بنویسید که جدول ضرب اعداد ۱ تا ۱۰ را به‌صورت زیر در خروجی نمایش دهد.



تمرین‌ها

۱. برنامه‌ای بنویسید که یک کاراکتر از کاربر دریافت کند، اگر حرف کوچک وارد شده بود، معادل حرف بزرگ آن را نمایش دهد، اگر حرف بزرگ وارد شده بود، معادل حرف کوچکش را در خروجی نمایش دهد.

۲. برنامه‌ای بنویسید که یک عدد از کاربر دریافت کند و آن رقمی که بیشترین تکرار را دارد در خروجی نمایش دهد.

۳. برنامه‌ای بنویسید که دو آرایه دوبعدی ۵*۵ از کاربر دریافت کند و عناصر متناظر را با هم جمع کرده و به شکل یک جدول در خروجی نمایش دهد.

۴. برنامه‌ای بنویسید که عدد n و سپس n کلمه از کاربر دریافت کند و کلمه‌هایی که بیشترین و کمترین تعداد حرف را دارند در خروجی نمایش دهد.

۵. برنامه‌ای بنویسید که آرایه‌ای از رشته‌ها دریافت کند و رشته‌هایی که با حرف Q شروع شده‌اند را در خروجی نمایش دهد.

پروژه:

می‌خواهیم نوشتن برنامه سیستم کتاب‌فروشی را آغاز کنیم. بیایید مرحله به مرحله برای عملکرد برنامه تصمیم بگیریم و سعی کنیم تصمیمات‌مان را پیاده‌سازی کنیم. پس بزن بریم!

- سیستم قرار است چه قابلیت‌هایی داشته باشد؟

در ابتدا بررسی کنیم که این سیستم قرار است چه کارهایی انجام دهد. احتمالاً باید لیستی از کتاب‌ها را ذخیره داشته باشیم سپس بتوانیم عملیات زیر را انجام دهیم:

- کتابی را به موجودی لیست اضافه کنیم،
- لیست کتاب‌ها را نمایش دهیم،
- کتاب بفروشیم،
- اطلاعات و تعداد موجودی یک کتاب را بیابیم،
- اطلاعات ذخیره شده یک کتاب را ویرایش کنیم.

این عملیات می‌تواند میزان خوبی از فعالیت‌های موردنیاز کتاب‌فروشی را پوشش دهد.

- طراحی برنامه به چه صورتی باشد که کاربر تا قبل از خروج از آن در هر مرحله بتواند عملیات موردنیاز خود را انتخاب و اجرا کند؟

برای این قسمت می‌توانیم به ایجاد یک فهرست از عملیات فکر کنیم. می‌توانیم یک فهرست از عملیات امکان‌پذیر در سیستم را به کاربر نمایش دهیم تا گزینه موردنیاز خود را انتخاب کند و عملیات مورد نظرش را انجام دهد پس از اجرای کامل آن، مجدد به فهرست باز گردد تا امکان انتخاب عملیات بعدی فراهم باشد.

از آنجایی که لیست قابلیت‌های سیستم را طراحی کردیم، می‌توانیم بر همان اساس فهرست، سیستم را طراحی کنیم. فقط توجه کنیم که در فهرست علاوه بر گزینه‌های اصلی، یک گزینه خروج از سیستم هم قرار دهیم تا کاربر بتواند در صورتی که کارش با برنامه تمام شد از آن خارج شود.

```
فهرست:
۱) افزودن کتاب جدید
۲) نمایش لیست کتاب‌ها
۳) خرید کتاب
۴) اطلاعات و موجودی کتاب
۵) ویرایش اطلاعات کتاب
۶) خروج
```

پس از نمایش فهرست باید گزینه انتخابی کاربر را بپرسیم،

```
شماره عملیات موردنظر را وارد کنید:
```

و منتظر وارد شدن شماره عملیات موردنظر باشیم.

- با وارد شدن اعداد ۱ تا ۵ چه می‌شود؟

بدیهی است که با وارد شدن هرکدام از شماره‌های ۱ تا ۵ باید دستورات مربوط به آن عملیات اجرا شوند که نتیجه اجرای آن دستورات عملکرد متناظر با آن گزینه خواهد بود؛ اما فعلا در این فصل از نوشتن این دستورات عبور می‌کنیم و با انتخاب این گزینه‌ها اعلام می‌کنیم:

```
سیستم در حال به‌روزرسانی می‌باشد.
در حال حاضر عملیات موردنظر شما در دسترس نیست.
```

و پس از آن مجدد به قسمت نمایش فهرست باز می‌گردیم و منتظر ورود شماره عملیات بعدی خواهیم بود.

- با وارد شدن عدد ۶ چه اتفاقی می‌افتد؟

وارد شدن عدد ۶ به معنی پایان کار و خروج از سیستم است پس زمانی که این گزینه انتخاب شد، می‌توانیم با نمایش پیغام مناسب برنامه را به پایان برسانیم.

```
خسته نباشید :)
```

- اگر اعداد و مواردی خارج از بازه ۱ تا ۶ وارد شود چه کنیم؟

در این صورت، عدد وارد شده اشتباه است و جزو گزینه‌های معتبر نخواهد بود. درنتیجه می‌توانیم اعلام کنیم:

```
انتخاب شما معتبر نیست.
```

سپس مجدد به قسمت نمایش فهرست بازگردیم و منتظر ورود شماره عملیات بعدی باشیم.

- در آغاز اجرای برنامه چه اتفاقی بیفتد؟

به‌نظر می‌رسد خوب باشد، اگر در ابتدا یک پیغام خوش‌آمد به کاربر نمایش دهیم که نشان‌دهنده آغاز اجرای برنامه و شروع کار سیستم باشد.

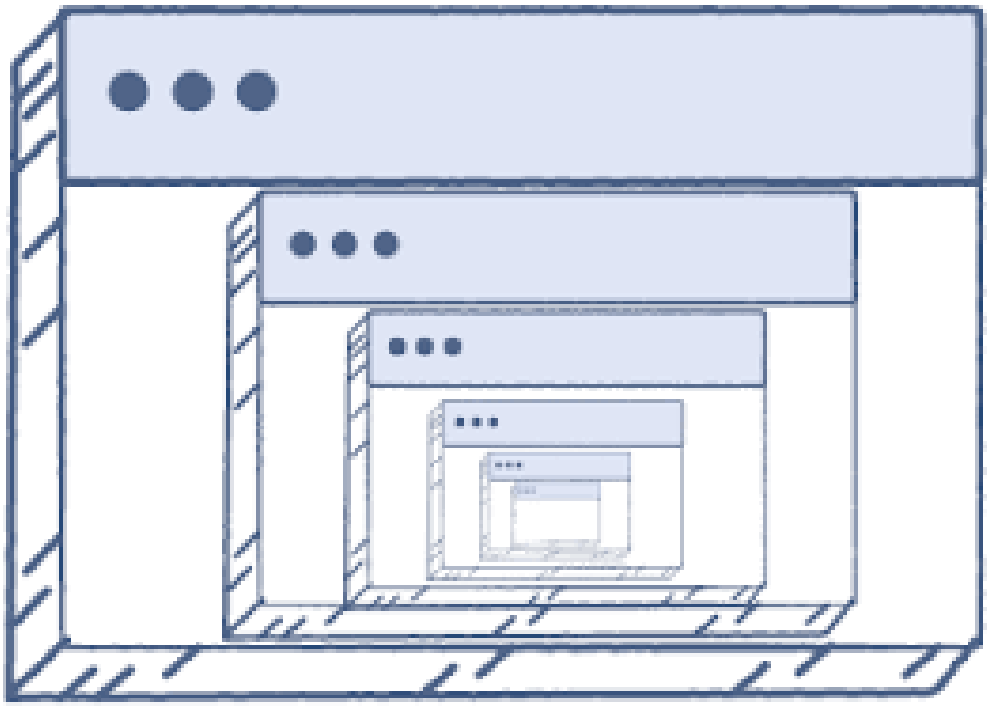
```
به سیستم کتاب‌فروشی خوش آمدید :)
```

و بعد از آن چرخه اصلی سیستم با نمایش فهرست آغاز شود.

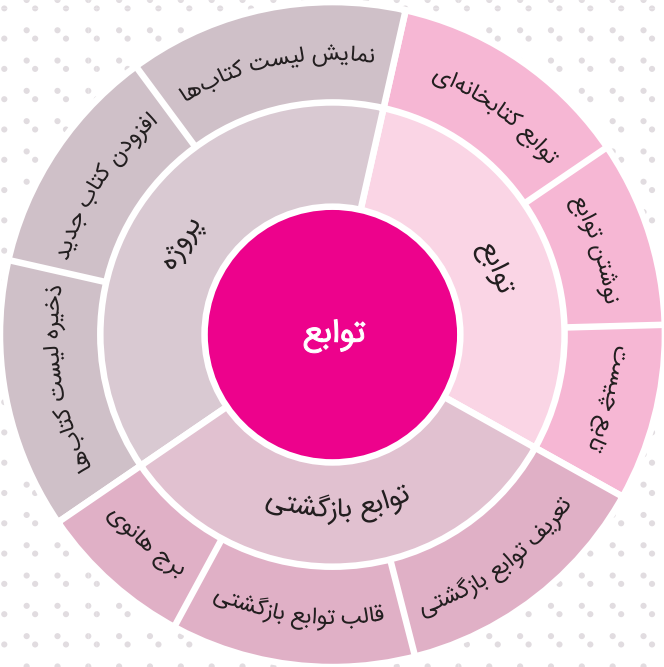
کد پروژه:

```
#include<iostream>
using namespace std;

int main()
}
    cout” >> Be Sistem Ketab Frooshi Khosh Amadid >> ”(: endl>>
    endl;
    bool Exit = false;
    while!)Exit(
}
          cout” >> Menu >> ”:endl;
          cout .1” >> Afzoodan Ketab Jadid >> ”endl;
          cout .2” >> Namayesh List Ketabha >> ”endl;
          cout .3” >> Kharid Ketab >> ”endl;
          cout .4” >> Etelaat va Mojoodi Ketab >> ”endl;
          cout .5” >> Virayesh Etelaat Ketab >> ”endl;
          cout .6” >> Khorooj >> ”endl;
          cout >> endl;
          int input;
          cout” >> Shomare Amaliat Mored Nazar ra Vared Konid;”:
          cin << input;
          cout >> endl;
          if => 1)input && input(5 =>
         }
                cout” >> System Dar Hal Berooz Resani Mibashad>> ”.
                      endl;
                cout” >> Dar Hal Hazer Amaliat Mored Nazar Shoma Dar
                                        Dastres Nist >> ”.endl;
                cout >> endl;
         {
          else if)input(6 ==
         }
                cout” >> Khaste Nabashid >> ”(: endl;
                Exit = true;
         {
          else
                cout” >> Entekhab Shoma Mootabar Nist >> ”.endl>>
                      endl;
    }
}
```



فصل دوم تابع و تابع بازگشتی



اگر این فصل را به خوبی مطالعه کنی و کارهای خواسته شده را به‌دقت انجام دهی:

- با تعریف تابع آشنا می‌شوی و می‌توانی یک تابع بنویسی.
- یاد می‌گیری فراخوانی تابع چگونه انجام می‌شود.
- با توابع کتاب‌خانه‌ای آشنا می‌شوی.
- یاد می‌گیری از تابع بازگشتی استفاده کنی.
- توابع موردنیاز را به پروژه اضافه می‌کنی.

اهداف رفتاری

تابع چیست؟

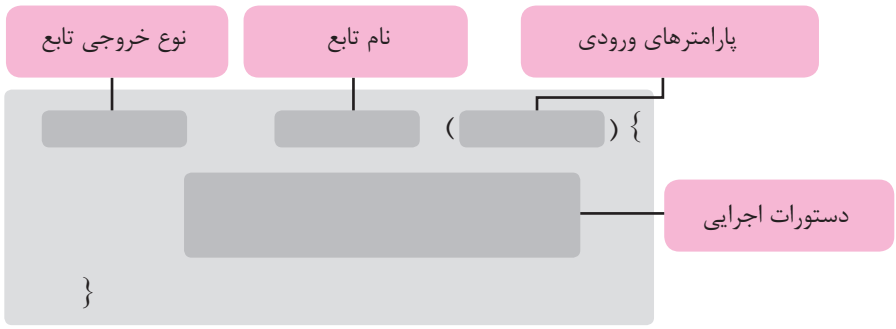
تابع، یک ابزار قدرتمند و مفید در برنامه‌نویسی است که به ما کمک می‌کند برنامه‌های خود را مرتب‌تر و در برخی موارد، با تعداد کمتری دستور بنویسیم. هر تابع، مانند یک دستگاه است که تعدادی ورودی را دریافت می‌کند، روی آن‌ها عملیات خاصی انجام می‌دهد و ورودی‌ها را به خروجی‌های موردنظر تبدیل می‌کند.

استفاده از توابع چطور می‌تواند برنامه ما را مرتب و خوانا کند و یا منجربه کوتاه و مختصر شدن دستورات آن شود؟

چطور یک تابع بنویسیم؟

اولین و مهم‌ترین نکته‌ای که وجود دارد، این است که تابع باید خارج از main نوشته شود. (می‌تواند بالای main باشد یا پایین آن)، همچنین برای نوشتن یک تابع، باید به نام تابع، نوع خروجی و ورودی‌ها و دستورالعمل تابع توجه کنیم و همه این موارد را به‌صورت زیر، در برنامه خود به کار ببریم:

کار با توابع در C++



کلمه return خروجی تابع را به بیرون از تابع؛ یعنی جایی که آن را فراخوانی کرده‌ایم ارسال می‌کند.

تابعی بنویسید که ورودی آن یک عدد صحیح و خروجی، فاکتوریل آن عدد است.

چطور تابع را فراخوانی کنیم؟

برای اینکه از تابع استفاده کنیم، تنها کافی‌است جایی که به آن نیاز داریم، اسمش را صدا کنیم. مثلاً بگوییم برای عدد یا متغیرمان فاکتوریل را حساب کن! باید توجه داشت که وقتی در برنامه یک تابع را صدا بزنیم تعداد ورودی‌هایی که به آن می‌دهیم حتماً برابر تعداد ورودی‌هایی که تابع دریافت می‌کند باشد وگرنه با خطا روبه‌رو خواهیم شد.


```
#include <iostream>
using namespace std;

bool mazrab_5 (int a)
{
    if (a % 5 == 0)
    {
       return 1 ;
    }
       return 0 ;
}

int main()
{
    int x;
    cin >> x;
    if(mazrab_5(x))
        cout << "yes";
    else
        cout << "no";
}
```

تابعی بنویسید که دو عدد را به‌عنوان ورودی بگیرد و حاصل عدد اول به توان عدد دوم را خروجی دهد.

توابع کتابخانه‌ای

توابع کتابخانه‌ای، توابعی هستند که از قبل نوشته شده‌اند و به‌دلیل کارآیی زیادشان، در اختیار همه قرار گرفته‌اند. توابع کتابخانه‌ای زیادی وجود دارند، مثل توابع ریاضی، رندوم، زمان و ... برای اینکه بتوانیم از این توابع استفاده کنیم، باید در ابتدای برنامه، سربرگ موردنیاز را اضافه کنیم تا مجوز استفاده از دستورات برای ما صادر شود!

توابع کتابخانه‌ای کاربردی

در این بخش، برخی از توابع کاربردی را به شما معرفی می‌کنیم:

تابع swap

این تابع دو متغیر از یک نوع را دریافت و مقدار داخلی این دو متغیر را جابه‌جا می‌کند.

```
#include<iostream>
using namespace std;

int main()
{
   int a = 4, b = 5;
   swap(a, b);
   cout << a << ' ' << b;
   return 0;
}
```

خروجی این برنامه به این صورت خواهد بود:

```
5 4
```

پیش از این، به کمک یک متغیر کمکی، برنامه‌ای نوشتیم که این کار را انجام می‌داد. این برنامه حدود ۳ الی ۴ خط اصلی داشت؛ اما با استفاده از تابع swap، می‌توانیم خیلی راحت و سریع به هدفمان برسیم.

تابع min

این تابع دو متغیر از یک نوع را دریافت می‌کند و مقدار متغیر کوچکتر را به‌عنوان خروجی برمی‌گرداند.

```
#include<iostream>
using namespace std;

int main()
{
   double a = 9.32, b = 5.81;
   double c = min(a, b);
   cout << c;
   return 0;
}
```

خروجی این برنامه به این صورت خواهد بود:

```
5.81
```