



نشر سپهاد



کتاب دوم

مجموعه کتاب‌های
نکمیلی سپهاد

بُلْغَاتُ الْجَنَانِ

بُلْغَاتُ الْجَنَانِ

دوم راهنمایی

محمد رضا جهانگیر
محمد آزین
محمد نبی‌زاده

به نام پژوهشگار

برنامه‌نویسی مقدماتی با بیسیک

دوم راهنمایی

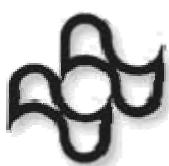
محمد نبی‌زاده

محمد آزین

محمد رضا جهانگیر

مجموعه کتاب‌های تکمیلی سمپاد

کتاب دوم



نشر سمپاد



نشر سمپاد

برنامه‌نویسی مقدماتی با بیسیک

پایه اول راهنمایی

مجموعه کتاب‌های تکمیلی سمپاد، کتاب اول

نویسنده‌ان: محمد نبی‌زاده، محمد آزین، محمدرضا جهانگیر

طرح جلد، صفحه آرایی و تصویرسازی: محمد سیاحت‌گر

حروف چینی: نشر سمپاد

شمارگان: ۱۰/۰۰۰ جلد

چاپ اول: ۱۳۸۶

چاپ و صحافی: طیف نگار

قیمت: ۱۰۰۰ تومان

کلیه حقوق برای نشر سمپاد محفوظ است.

شابک

ISBN

فهرست مطالب

۵.	مقدمه
۶.	سخنی با دانش آموز
۷.	فصل اول - مقدمات رایانه
۸.	ساختار رایانه
۹.	چگونه با رایانه ارتباط برقرار کنیم
۱۰.	زبان های برنامه نویسی
۱۱.	الگوریتم
۱۲.	برنامه
۱۳.	هوش مصنوعی
۱۴.	فصل دوم - آشنایی با محیط Quick BASIC
۱۵.	مطالب کلی
۱۶.	اجرای اولین برنامه
۱۷.	منوها
۱۸.	منوی فایل
۱۹.	منوی Edit
۲۰.	منوی Search
۲۱.	منوی Debug
۲۲.	خطاهای
۲۳.	فصل سوم - دستورات ورودی، خروجی و متغیرها
۲۴.	دستور خروجی
۲۵.	متغیرها (خانه های حافظه)
۲۶.	دستور ورودی
۲۷.	دستور خروجی، کمی بیشتر
۲۸.	تمرین
۲۹.	فصل چهارم - جای گزینی، شرط، پرش و توابع
۳۰.	دستور جای گزینی (LET).
۳۱.	متغیرها (خانه های حافظه)
۳۲.	دستور شرط
۳۳.	دستور پرش
۳۴.	توابع پر کاربرد
۳۵.	تابع قسمت صحیح
۳۶.	تابع تولید عدد تصادفی
۳۷.	عمل گر باقیمانده
۳۸.	عمل گر خارج قسمت
۳۹.	تمرین

۵۴	فصل پنجم - جایگزینی، شرط، پرس و توابع
۵۵	حلقه‌ی FOR
۵۹	گام حلقه
۶۱	تمرین
۶۵	فصل ششم - آرایه‌ها
۶۶	آرایه‌ی یک‌بعدی
۷۲	تمرین
۷۵	فصل هفتم - گرافیک
۷۶	مختصات گرافیکی
۷۸	دستور رسم نقطه
۸۰	دستور رسم خط
۸۱	دستور رسم دایره
۸۳	دستور رنگ‌آمیزی
۸۴	تمرین
۸۶	فصل هشتم - برنامه‌نویسی پیشرفته
۸۷	الگوریتم‌های پایه
۸۸	محاسبه‌ی بیشینه و کمینه (بزرگ‌ترین و کوچک‌ترین)
۸۸	انتقال دادن (Shift) و درج کردن (Insert)
۹۰	حذف کردن (Delete)
۹۰	جستجو کردن (Search)
۸۸	مرتب‌سازی (Sort)
۹۴	پردازش رشته
۹۴	الحاق دو رشته (Concatenation)
۹۵	مقایسه
۹۵	طول رشته
۹۶	جداسازی از چپ و راست
۹۶	زیررشته
۹۸	جستجو
۹۸	تبدیلات
۹۸	تولید رشته
۹۹	جدول اسکی (ASCII)
۹۹	آرایه‌های دوبعدی
۱۰۱	پیوست ۱ - نهرست پژوهشها
۱۰۵	پیوست ۲ - جدول اسکی

مقدمه

امروزه رایانه برای اکثر دانشمندان و متخصصین، نقش یک قلم را پیدا کرده است. این افراد با جدیت و پشتکار و با سلطی که از پیش بر علوم پایه‌ی رایانه به دست آورده‌اند، همگام با دانش روز دنیا، در مسیر تکامل علمی خویش گام برپی‌دارند. از سوی بسیاری از دانشمندان علوم رایانه، تکنولوژی آموزشی و حتی متخصصین روان‌شناسی، از برنامه‌نویسی و برنامه‌سازی در بین علوم پایه‌ی رایانه، به عنوان یکی از ابزارهای مفید و کلیدی، برای آموزش اصول تفکر منطقی و منظم و همچنین توسعه‌ی خلاقیت‌های فردی و تحصیلی یاد می‌شود.

یک برنامه‌نویس با دید وسیع‌تر و بهتری به وقایع و مسایل اطراف خود می‌نگردد. او مجهز به مهارت‌ها و فنونی است که به او یاد می‌دهند که چگونه از مسایل بزرگ و پیچیده نهراسد، چگونه با دید منطقی و منظم خود، مسایل بزرگ را به مسایل کوچک‌تر بشکاند و سپس نسبت به حل این مسایل کوچک‌تر و ساده‌تر اقدام نماید. او به خوبی یاد گرفته است که چگونه از رایانه به عنوان یک قلم استفاده کند.

یک برنامه‌نویس لزوماً متخصص رایانه نیست. حافظه‌ی ماء، بسیاری از دانش‌آموختگان قدیمی و جدید سمپاد را به یاد دارد، که اگر چه در رشته‌ها و گرایش‌های علوم رایانه تحصیل نکرده‌اند، اما به واسطه‌ی تسلط بر برنامه‌نویسی، همواره از دیگر هم‌کلاسی‌ها و همدره‌ای‌های خود، چه در مراکز علمی و آکادمیک و چه در محیط‌های کاری، پیش بوده و هستند؛ چرا که برنامه‌نویسی به آنها آموخته بود، هیچ‌گاه و در هیچ موقعیتی، از برخورد با مسایل ناشناخته و بزرگ، ترسی نداشته باشند و خلاقیت و تجربه قابلی خود را در حل مسایل جدید، به کار بینند.

تیم نویسنده‌ی این کتاب، حاصل تجربیات طولانی و گران‌ستگ خود و اساتیدشان، در گروه‌های کامپیوتر مراکز راهنمایی و دبیرستان علامه‌حلى تهران، را در قالب این کتاب جمع‌آوری و تدوین نموده‌اند. کتاب حاضر حاصل گردآوری تلاش‌ها و تجربیات بیش از ۱۶ سال تدریس درس برنامه‌نویسی در طی جلسات مختلف گروهی، برای دانش‌آموزانی است که هم‌اکنون در اقصی نقاط دنیا، مهارت‌ها و فنون اکتسابی خود را در شاخه‌های مختلف علم و فناوری به کار می‌بینند. نویسنده‌گان کتاب وظیفه‌ی خود می‌دانند از همه‌ی پیش‌کسوتان و اساتید خود در گروه کامپیوتر مرکز علامه‌حلى تهران، به ویژه آقایان رضا صدیق، فرید رزازی و سعید سرکاری و دیگر همکاران محترم گروه از سال ۱۳۷۰ تا کنون تشکر و قدردانی نمایند. همچنین از آقای محمد سیاحت‌گر که وظیفه‌ی ویراستاری و سرکار خانم ریحانه کبیری که زحمت تایپ مطالب کتاب را بر عهده داشتند، تشکر و قدردانی می‌گردد.

در خاتمه جا دارد یاد شهدای سمپاد، معلمین و اساتید خود، بویژه آن‌هایی که اعتقاد داشتند - تیزهوشان کسانی هستند که توان‌مندی بیشتری برای خدمت به خلق خدا دارند - را گرامی بدارم.

به امید توفیق روز افزون در سایه‌ی الطاف الهی

محمد رضا چهانگیر

دانش‌آموخته و معلم رایانه‌ی سمپاد

سخنی با دانشآموز

همیشه به یاد داشته باشید که یادگیری علوم رایانه، مانند یادگیری فنون شنا است. آیا شما تا به حال سراغ دارید، که کسی بدون تمرين در استخر و تنها با خرید یک کتاب آموزش شنا، توانسته باشد شناگر ماهری شود؟! به همین خاطر، خیلی جدی به شما توصیه می‌کنیم که صرفاً به حل مثال‌ها و تمرين‌های این کتاب بر روی کاغذ، اکتفا نکنید و حتماً برنامه‌های خود را بر روی رایانه اجرا کنید.

کسب مهارت‌ها و شگردهای برنامه‌نویسی، بدون سر و کله زدن با رایانه و گرفتن جواب دلخواهتان از رایانه، ممکن نیست. اگر بخواهیم خیلی راحت و بی‌پرده سخن بگوییم، یک برنامه‌نویس خوب دو ویژگی مهم دارد: در درجه‌ی اول از هوش و استعداد خوب و در درجه‌ی دوم از صبر و تحمل بالایی بهره‌مند است؛ چرا که جواب گرفتن از یک مجری غیر خلاق مانند رایانه، احتیاج به صبر و تحمل قابل توجهی دارد و لذا یادگیری برنامه‌نویسی با عجله و مثلاً در ظرف یک هفته و یا چند روز، به طور فشرده و تضمینی!، اصلاً عاقلانه و شدنی نیست.

پس از مایی که سالیان سال است این راه را رفته‌ایم، به شما نوگل نوپا در این عرصه، چند نصیحت:

حتماً مثال‌ها و تمرين‌های کتاب را بر روی رایانه اجرا کنید.

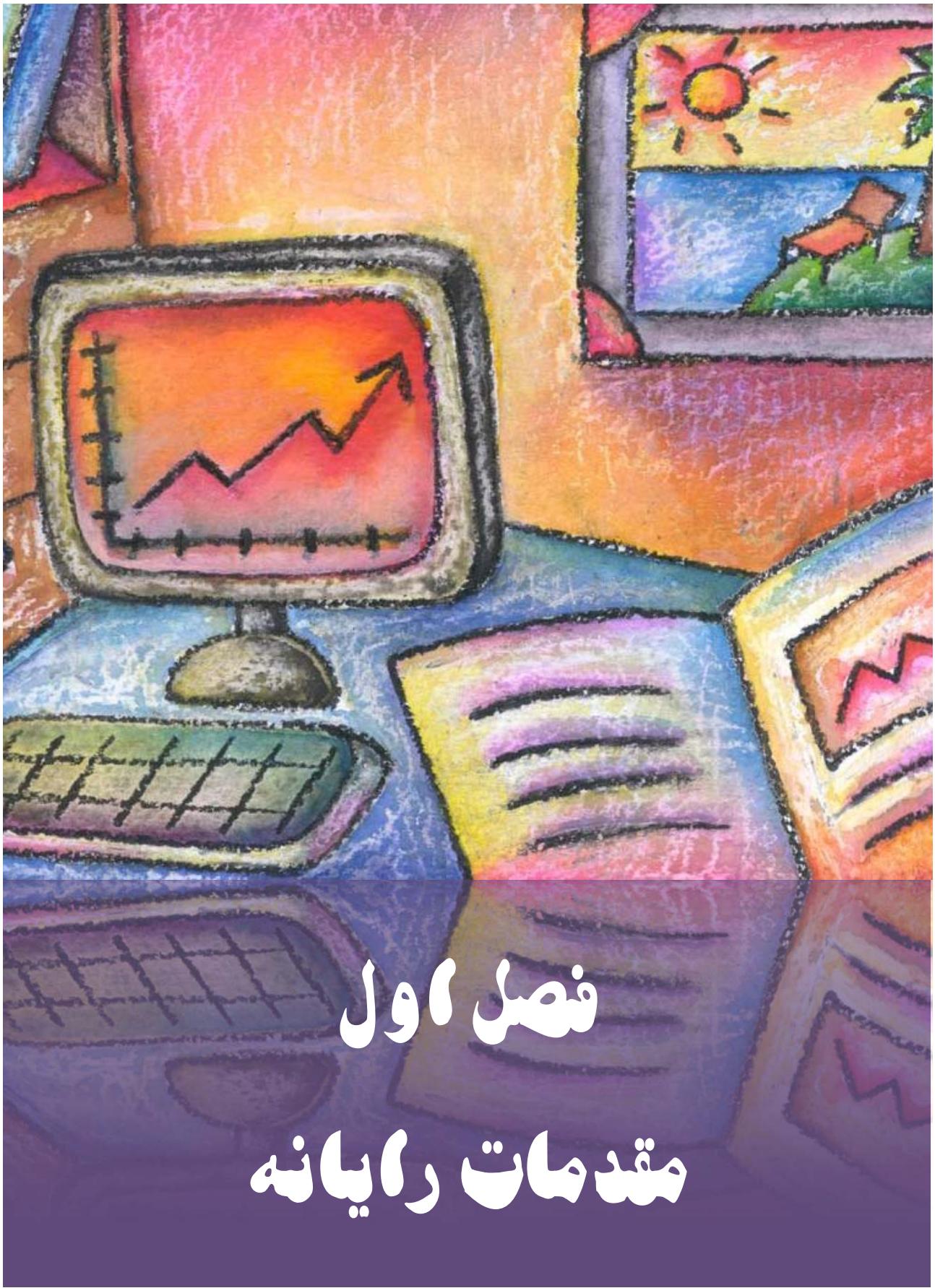
سعی کنید قسمت‌هایی که با عنوان «برای مطالعه‌ی بیشتر و یا تمرينات ویژه‌ی دانشآموزان علاقه‌مند» مشخص شده است را در ساعات خارج از کلاس دنبال کنید.

بهترین تمرين‌ها، تمرين‌هایی هستند که خودتان طرح می‌کنید. سعی کنید به همراه معلم و دوستان خود، تمرين‌ها و مسایل جدیدی را برای برنامه‌نویسی پیشنهاد کرده و آن‌ها را پای رایانه حل کنید. برنامه‌نویسی رایانه، درس شیرینی است که به مرور زمان اثرات مثبت خود را بر ذهن، فکر و حتی برنامه‌ی زندگی شما به جا می‌گذارد. نقش این درس در توسعه‌ی خلاقیت‌ها و آموزش تفکر منطقی و منظم، امروزه بر همه ثابت شده است.

حتی شما دانشآموز عزیز، که نمی‌خواهید در دوره‌ی دبیرستان رشته‌ی ریاضی فیریک را انتخاب کنید، بدان که استفاده‌ی شما از برنامه‌نویسی در سایر دروس، کمک شایانی به پیشرفت شما در دروس مورد علاقه‌یتان می‌کند. پس قدر این درس را بیشتر بدانید.

موفق باشید!

تیم نویسنده‌ی کتاب



فصل اول

مقدمات رایانه

۱.۱. مقدمه

امروزه رایانه به صورت یک ابزار جدایی ناپذیر از زندگی انسان‌ها درآمده است که استفاده‌ی درست از آن نیازمند شناخت دقیق ساختار، توانایی‌ها و نحوه‌ی تعامل با رایانه می‌باشد. در درس رایانه‌ی سال گذشته با ساختار رایانه و نحوه‌ی کار کردن با برخی از مهمترین نرم‌افزارهای کاربردی آن آشنا شدیم. امسال پس از یک مرور سریع، با جنبه‌ی دیگری از توانایی‌های رایانه یعنی قابلیت برنامه‌ریزی، آشنا می‌شویم.

۱.۲. ساختار رایانه

همانطور که می‌دانید به هر ماشین محاسبه‌گر، کامپیوتر یا رایانه گفته می‌شود. صرف نظر از ماشین‌های محاسبه‌ی مکانیکی که امروزه استفاده‌ی عمومی ندارند، می‌توانیم هر سیستم محاسبه‌گری که قابل برنامه‌ریزی باشد را رایانه بنامیم. این سیستم‌ها می‌توانند طیف گسترده‌ای از ماشین‌های لباس‌شویی تا ابررایانه‌های پیش‌بینی کننده‌ی وضع هوا در بر بگیرند.



هر رایانه از یک واحد پردازشگر مرکزی (یا CPU) برای اجرای دستورات و انجام محاسبات استفاده می‌کند. در کنار این واحد، ابزارهای ورودی، خروجی و واحد حافظه قرار دارند. داده‌ها از جهان خارج به وسیله‌ی ورودی‌ها وارد رایانه شده و پاسخ محاسبه شده توسط خروجی‌ها به کاربر ارائه می‌شود. واحد حافظه به ذخیره‌ی نتایج و اطلاعات مورد نیاز برای انجام محاسبات می‌پردازد. مطمئناً بدون وجود حافظه‌ای که اطلاعات اولیه را در اختیار داشته باشد و پاسخ‌های نهایی را در خود ذخیره کند، محاسبات بی معنی و بی استفاده خواهد بود.

در رایانه سه نوع حافظه وجود دارد:

۱. حافظه‌ی پنهان
۲. حافظه‌ی موقت
۳. حافظه‌ی دائم

برای روشن شدن جایگاه هر یک از این سه نوع حافظه، از یک مثال استفاده می‌کنیم و بعد به شرح هر یک از این انواع می‌پردازیم. فرض کنید دانش‌آموز تیزهوشی در آزمون ریاضی شرکت کرده است. این دانش‌آموز برای پاسخ به آزمون، مسیر زیر را طی می‌کند:



ابتدا با استفاده از ابزار ورودی یعنی چشم، صورت مسئله را مطالعه می‌کند. سپس داده‌های مسئله در حافظه‌ی مغز قرار می‌گیرند و مغز شروع به محاسبه می‌کند. تا جایی که محاسبات کوتاه هستند، مغز می‌تواند از حافظه‌ی کوتاه مدت استفاده کرده و جواب این محاسبات را به دست آورد. حافظه‌ی کوتاه مدت مغز شبیه حافظه‌ی پنهان پردازنده است. برای انجام محاسبات پیچیده‌تر، مغز به یک حافظه‌ی

کمکی نیازمند است، این حافظه همان صفحه‌ی چرکنویس است که گنجایش بیشتری برای ذخیره‌سازی محاسبات دارد. این حافظه معادل حافظه‌ی موقت در رایانه می‌باشد. دانش‌آموز در پایان، پاسخ‌هایی را که مغز آن‌ها را محاسبه کرده و در برگه‌ی چرکنویس ذخیره شده‌اند را به برگه‌ی پاسخ‌نامه منتقل می‌کند؛ برگه‌ی پاسخ‌نامه مثالی از حافظه‌ی دائم می‌باشد.

با توجه به مثال فوق، تعریف دقیق‌تری از این سه نوع حافظه ارائه می‌دهیم :

۱. حافظه‌ی پنهانی : این حافظه گنجایش بسیار کم و سرعت انتقال اطلاعات بسیار بالایی دارد، بنابرین برای انجام محاسبات سریع و کوچک استفاده می‌شود.
 ۲. حافظه‌ی موقت : اطلاعات مورد نیاز برای حل یک مسئله و یا محاسبات لازم برای حل یک مسئله بر روی این حافظه ذخیره می‌شوند. از ویژگی‌های این حافظه، سریع بودن آن و پاک شدن اطلاعات آن در اثر قطع جریان برق می‌باشد.
 ۳. حافظه‌ی دائم : برنامه‌ها و نتایج محاسبات در نهایت بر روی این حافظه ذخیره می‌شوند. می‌توان اطلاعات را بر روی این حافظه به طور دائم نگهداری کرد، لذا محتوای این حافظه در اثر قطع جریان برق پاک نمی‌شود.
-

۱. ۳. چگونه با رایانه ارتباط برقرار کنیم؟

حتماً شنیده‌اید که زبان رایانه صفر و یک است. زبان عجیبی است! نه؟ در مدارهای الکترونیکی رایانه، محاسبات تنها با ارقام صفر و یک انجام می‌شوند، به این روش محاسبه، محاسبه در مبنای ۲ می‌گوییم. ما انسان‌ها محاسباتمان را در مبنای ۱۰ انجام می‌دهیم (چرا؟) به همین خاطر است که از ارقام ۰، ۱، ۲، ۳، ...، ۸ و ۹ استفاده می‌کنیم؛ حتماً رایانه هم از این روش تعجب خواهد کرد ! (شما روش محاسبه در مبناهای مختلف را امسال در درس ریاضی خواهید آموخت.)

در واقع رایانه تمام اعداد و دستورات را به صورت دنباله‌هایی از صفر و یک می‌فهمد. بنابراین برای ارتباط برقرار کردن با آن نیز باید به همین روش عمل کرد، یعنی باید با زبان صفر و یک با رایانه

صحبت کرد! این کار بسیار مشکل است و عملاً انجام کارهای کمی پیچیده با این روش غیر ممکن خواهد بود. اما این مشکل راه حلی دارد: زبان‌های برنامه نویسی.

۱. ۴. زبان‌های برنامه نویسی

زبان‌های برنامه نویسی، برقراری ارتباط ما با رایانه را راحت‌تر می‌کنند. این زبان‌ها ساختاری شبیه به زبان انسان دارند تا کاربر بتواند به راحتی منظور خود را بیان کند، سپس این زبان، به زبان صفر و یک ترجمه می‌شود تا برای رایانه قابل فهم باشد. به این روش می‌توانیم بسیار راحت از رایانه بخواهیم که کارهای مورد نظر ما را انجام دهد.

طبق آنچه گفتیم زبان‌های برنامه نویسی در سطوح مختلفی طبقه‌بندی می‌شوند، به این ترتیب که هر چه زبان برنامه نویسی به زبان انسان نزدیک‌تر باشد به آن زبان، سطح بالاتر می‌گوییم و هر چه به زبان رایانه (صفر و یک) نزدیک‌تر باشد، آن زبان را سطح پایین‌تر می‌گوییم.

بر اساس این تقسیم بندی، نگاهی به معروف‌ترین زبان‌های برنامه نویسی موجود (از سطح پایین به بالا) می‌اندازیم:

- **زبان اسembly (Assembly)**: این زبان نزدیک ترین زبان به زبان صفر و یک رایانه است. در واقع یک رابطه‌ی ساده‌ی یک به یک بین دستورات صفر و یکی رایانه و دستورات زبان اسembly وجود دارد. برنامه نویسی با زبان اسembly، هر چند از کارکردن با صفر و یک‌ها ساده‌تر است، ولی هنوز فاصله‌ی زیادی با زبان انسان دارد. از طرف دیگر زبان اسembly زبان بسیار قدرتمندی است، زیرا به طور مستقیم با پردازنده‌ی رایانه در ارتباط است و پردازنده را کاملاً در کنترل خود می‌گیرد.

```
.model small
.stack
.data
message db "Hello World!", "$"
.code

main proc
    mov ax,seg message
    mov ds,ax

    mov ah,09
    lea dx,message
    int 21h

    mov ax,4c00h
    int 21h
main endp
end main
```

برنامه‌ای برای نوشتن یک پیغام ساده در زبان اسembلی!

- **زبان‌های C و C++ :** زبان C ، زبان سطح بالاتر از زبان اسembلی است، پس می‌توانید حدس بزنید که برنامه‌نویسی با آن باید ساده‌تر از اسembلی باشد. همچنین این زبان، زبان قدرتمندی نیز هست. خوب است بدانید که زبان C را با استفاده از زبان اسembلی ساخته‌اند. نسخه جدیدتر زبان C به نام C++ شناخته می‌شود.

```
#include <iostream.h>

int main()
{
    cout << "Hello World!" << endl;
    return 0;
}
```

همان برنامه در زبان C++

- **زبان پاسکال** : دستورات زبان پاسکال بسیار شبیه به کلمات زبان انگلیسی هستند، بنابرین یادگیری و برنامه‌نویسی با آن، راحت‌تر از زبان C است و در نتیجه این زبان از زبان C سطح بالاتر است.

```
program HelloWorld(output);
begin
    WriteLn('Hello World!');
end.
```

باز هم همان برنامه در زبان پاسکال

- **زبان بیسیک (BASIC)** : زبان بیسیک یکی دیگر از زبان‌های سطح بالا است. یادگیری زبان بیسیک و برنامه‌نویسی با آن بسیار ساده است. ما نیز به همین دلیل آن را برای آموزش در این کتاب انتخاب کردایم. شما نیز به زودی لذت برنامه‌نویسی در بیسیک را حس خواهید کرد ! زبان بیسیک نسخه‌های مختلفی دارد که ما معروف‌ترین آن‌ها یعنی Quick BASIC (بیسیک سریع!) را انتخاب کردایم.

```
PRINT "Hello World!"
```

و در آخر همان برنامه به زبان بیسیک.

آیا می‌توانید حدس بزنید که این برنامه چه کاری انجام می‌دهد؟!

- زبان‌های بصری (Visual)** : زبان‌های بصری مانند Visual C++ و Visual Basic قابلیت‌های جدیدی را به زبان‌های پیشین اضافه کرده‌اند. شما می‌توانید در این زبان‌ها علاوه بر نوشتن برنامه، از امکانات آماده‌ای که به صورت تصویری در محیط برنامه‌نویسی گنجانده شده‌اند، استفاده کنید. این کار برنامه‌نویسی در محیط سیستم‌عامل ویندوز را بسیار راحت‌تر می‌کند. سال آینده شما با زبان آشنا خواهید شد.
- سایر زبان‌ها** : در دنیای برنامه‌نویسی از زبان‌های برنامه‌نویسی بسیاری استفاده می‌شود که هر کدام ویژگی‌ها و توانایی‌های مخصوص به خود را دارند. از جمله آن‌ها می‌توان به زبان‌های جاوا، C#، دلفی، پل، فرترن، PHP و ... اشاره کرد.

۱. ۵. الگوریتم

شما در خانه نشسته‌اید و مشغول درس خواندن هستید. دوستتان با شما تماس می‌گیرد و مشکلی را با شما در میان می‌گذارد. مهمانی سرزده برای او از راه رسیده است؛ ولی او نمی‌داند چگونه باید چای درست کند ! دوستتان از شما درخواست کمک می‌کند، چگونه به او کمک خواهید کرد؟



حتماً روش انجام کار را مرحله به مرحله برای او شرح خواهید داد. مثلاً خواهید گفت : اول آب را در قوری بریز، بعد قوری را روی شعله قرار بده تا آب به جوش آید، سپس یک قاشق چای خوری چای خشک در قوری بریز و شعله را کم کن، در نهایت پنج دقیقه صبر کن تا چای دم بکشد.

پرسش : می‌توانید بگویید این دستورالعمل باید چه ویژگی‌هایی داشته باشد؟

پاسخ : هر دستورالعملی باید دارای این ویژگی‌ها باشد:



۱۰۱

۱. زبان قابل فهم برای اجرا کننده داشته باشد.
۲. مراحل باید تا حدی ساده باشند که اجرا کننده بتواند تک‌تک آن‌ها را اجرا کند.
۳. مراحل باید ترتیب صحیح داشته باشند.

۱۰۲

به روش انجام کار که طبق قوانین بالا نوشته شده باشد، الگوریتم می‌گویند. محمد بن موسی خوارزمی دانشمند بزرگ ایرانی، اولین کسی است که این راه را برای بیان روش انجام کارها پیشنهاد داد. لفظ الگوریتم نیز از نام الخوارزمی گرفته شده است.



در این کتاب یاد می‌گیریم که چگونه راه حل مسئله را در قالب الگوریتم بیان کنیم و چگونه این الگوریتم را به برنامه‌ی رایانه‌ای تبدیل کنیم.

۱. ۶. برنامه

رایانه دستگاهی است که توانایی اجرای برنامه‌های مختلف را دارد. اگر به این مسئله توجه داشته باشیم که رایانه یک ابزار است و مانند سایر ابزارها وظیفه‌ی آسان‌تر کردن کارهای ما را دارد، متوجه می‌شویم که برنامه‌ها باید کاربرد خاصی برای ما داشته باشند. مثلاً بتوانند تمرینات ریاضی ما را حل کنند(!) یا اطلاعات کارکنان یک شرکت را به سرعت و راحتی در اختیار ما بگذارند و یا به پیش‌بینی وضعیت هوا در ساعت‌های آینده بپردازند. شما می‌توانید مثال‌های گوناگونی را به این مجموعه اضافه کنید. در تمام این مثال‌ها برنامه، اطلاعاتی را از کاربر می‌گیرد که به آن ورودی می‌گوییم، سپس این اطلاعات را پردازش می‌کند تا به جواب خواسته شده‌ی کاربر برسد و در نهایت پاسخ به عنوان خروجی برنامه به کاربر ارائه می‌شود. (ورودی، خروجی، پردازش)

یک برنامه از دستوراتی شکل گرفته که پشت سر هم اجرا می‌شوند. این دستورات، در هر زبان برنامه‌نویسی شکل و کارکرد خاصی دارند. از پشت سر هم قرار گرفتن دستورات یک برنامه و اجرای آن توسط رایانه، کار خواسته شده برای ما انجام می‌شود. کسی که می‌خواهد با زبان خاصی برنامه بنویسد، باید ابتدا دستورات مورد نیازش را بشناسد و شکل و کاربرد آن‌ها را بداند. در مرحله بعد برنامه نویس، با پشت سر هم قرار دادن این دستورات، الگوریتم مورد نظرش را به برنامه تبدیل می‌کند.

```
def add5(x):
    return x+5

def dotwrite(ast):
    nodename = getNodename()
    label=symbol.sym_name.get(int(ast[0]),ast[0])
    print ' %s [%s]' % (nodename, label),
    if isinstance(ast[1], str):
        if ast[1].strip():
            print '= %s';' % ast[1]
        else:
            print ''
    else:
        print '['
        children = []
        for n, child in enumerate(ast[1:]):
            children.append(dotwrite(child))
        print ' %s -> {' % nodename,
        for name in children:
            print '%s' % name,
```

۱.۷. هوش مصنوعی

شاید این خبر را شنیده باشید: «رایانه توانست گری کاسپاروف، مرد اول شطرنج جهان را شکست دهد». شطرنج، یکی از پیچیده‌ترین بازی‌های فکری است و برنده شدن در آن، نتیجه‌ی مستقیم هوش فرد بازیکن می‌باشد. وقتی به دو عبارت بالا نگاه می‌کنیم، طبیعتاً نتیجه‌ی می‌گیریم که رایانه یک موجود هوشمند است. آیا این نکته به این معناست که انسان رقیبی سرسخت پیدا کرده است؟



گری کاسپاروف در حال مسابقه با دیپ جونیور در سال ۲۰۰۳

این مسابقه منجر به تساوی ۳-۳ گشت

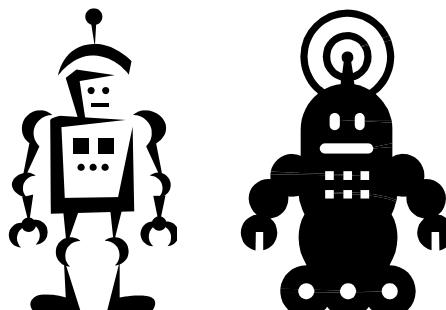
بهتر است نگاهی به الگوریتم بازی شطرنج بیاندازیم. هنگام بازی، حریف با هر حرکت خود، ما را در موقعیتی جدید قرار می‌دهد. در این موقعیت جدید، ما امکان حرکات مختلفی را داریم که می‌توان تمام احتمالات ممکن را بر روی کاغذ نوشت. تعداد این حالات ممکن است زیاد باشد، ولی وقتی کمی دقت می‌کنیم متوجه می‌شویم که حریف هم به ازای هر یک از این حالات، امکان حرکات مختلفی را دارد. پس تعداد کل حرکات احتمالی در این دو مرحله بسیار زیاد می‌شود. حال شما این بحث را در مورد تمام مراحل یک بازی کامل شطرنج گسترش دهید. خواهید دید احتمالات ممکن را باید با ارقام نجومی بیان

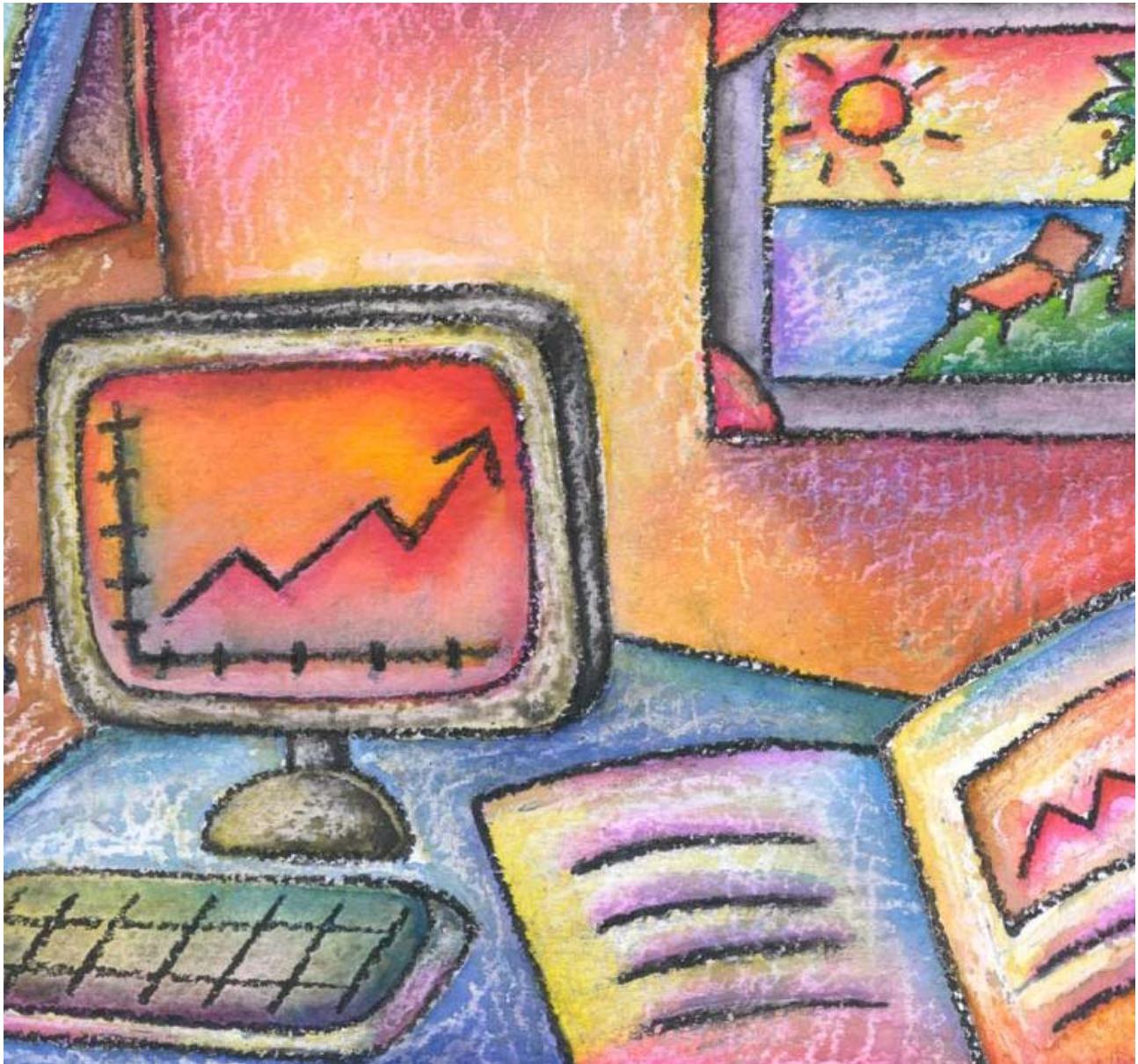
کرد! در واقع راه صحیح شطرنج بازی کردن این است که ما در هر مرحله به ازای هر حرکت خود، بتوانیم حرکات حریف را در مراحل بعدی بستجیم و بهترین حالت را انتخاب کنیم. وقتی بهترین حالت حاصل خواهد شد که ما بتوانیم تمام احتمالات ممکن را تا انتهای یک بازی پیش‌بینی کنیم. واضح است که این کار عملاً غیر ممکن است.

روش بالا روشی است که هر شطرنج بازی در ذهن خود اجرا می‌کند. بازیکنان مبتدی می‌توانند یک یا دو مرحله بعد را پیش‌بینی کنند و هرچه بازیکن حرفه‌ای‌تر باشد، می‌تواند مراحل بیشتری را پیش‌بینی کند. رایانه هم با همین روش شطرنج بازی می‌کند. البته به علت سرعت بالای محاسبات، رایانه می‌تواند در زمان کوتاهی مراحل بعدی بازی را پیش‌بینی کند.

می‌بینیم که رایانه برای انجام هر کاری، اگرچه به ظاهر هوشمندانه باشد، از یک الگوریتم مشخص پیروی می‌کند. به این الگوریتم‌ها که باعث می‌شوند رایانه رفتاری (به ظاهر) هوشمندانه داشته باشد، الگوریتم‌های هوش مصنوعی گفته می‌شود. هوش مصنوعی یکی از جالب‌ترین و پر کاربردترین حوزه‌های علم نرم‌افزار است.

الگوریتم‌های هوش مصنوعی، قابلیت حل مسائل پیچیده در زمان کوتاه را برای ما فراهم می‌کنند. هم‌چنین ما می‌توانیم با استفاده از این الگوریتم‌ها، قابلیت یادگیری را به نرم‌افزار و سخت‌افزارمان اضافه کنیم، برای مثال رباتی بسازیم که دستورات صاحبش را یاد بگیرد!





فصل دوم

آشنایی با Quick BASIC

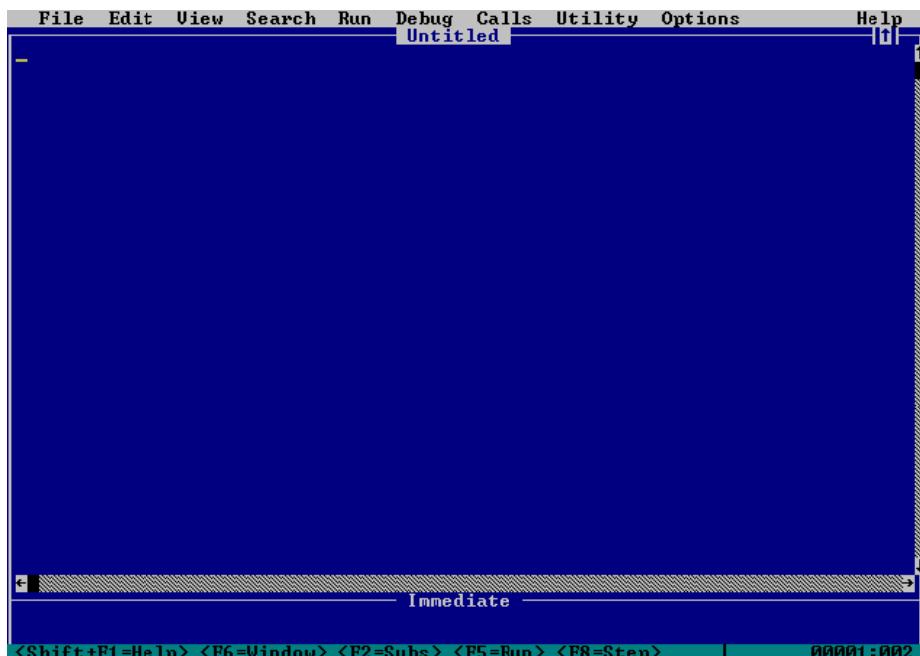
۱.۲. مقدمه

زبان‌های برنامه‌نویسی دارای یک محیط ویرایش‌گر برنامه هستند. ویرایش‌گر، در واقع محیطی است که برنامه در آن نوشته شده و تغییرات لازم در آن داده می‌شود. با استفاده از امکانات ویرایش‌گر می‌توانیم برنامه را ذخیره و بازیابی کنیم. همچنین می‌توانیم برنامه‌ی خود را عیب‌یابی کنیم. ویرایش‌گر امکانات دیگری چون راهنمای بیسیک را نیز در اختیارمان می‌گذارد.

در این فصل با مهمترین بخش‌های QBASIC آشنا می‌شویم و در ادامه‌ی کتاب، کاربرد هر یک از این دستورات را می‌بینیم.

۲. مطالب کلی

وقتی وارد محیط برنامه می‌شوید، چیزی شبیه به تصویر زیر را خواهید دید. شما برنامه‌ی خود را در قسمت آبی رنگ بزرگ می‌نویسید و سپس آن را اجرا می‌کنید.



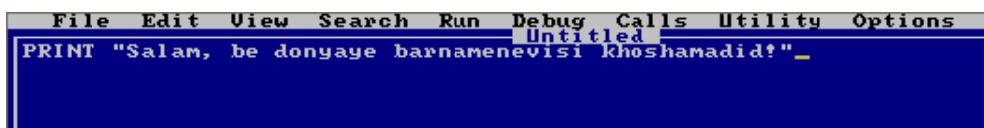
منوهای بالای صفحه امکانات بیسیک را در اختیار ما می‌گذارند. نام برنامه‌ی ما در بالای قسمت آبی رنگ نوشته می‌شود. از آنجایی که در ابتدا هنوز نامی برای برنامه‌ی خود انتخاب نکرده‌ایم، عبارت Untitled (بدون عنوان) در بالای صفحه دیده می‌شود. در نوار پایین صفحه، سمت راست، شماره سطر و ستونی که مکان نما در آن قرار دارد، نمایش داده می‌شود.

۲.۳. اجرای اولین برنامه

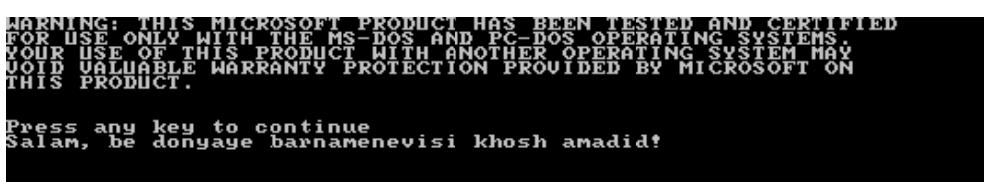
اولین کاری که بعد از وارد شدن به محیط بیسیک انجام می‌دهیم، نوشتن یک برنامه‌ی ساده و اجرا کردن آن است. برنامه‌ی ما روی صفحه می‌نویسد:

Salam, be donyaye barnamenevisi khoshamadid!

برای این کار از دستور PRINT استفاده می‌کنیم. (بعداً با دستورات مهم بیسیک آشنا خواهید شد)



پس از نوشتن این برنامه‌ی یک خطی، با فشردن کلید F5 برنامه‌ی شما اجرا خواهد شد و شما خروجی برنامه را روی صفحه خواهید دید:



همانطور که مشاهده می‌کنید، خروجی برنامه بر روی صفحه‌ی سیاه رنگی ظاهر می‌شود. نوشهای بالای صفحه، مربوط به متونی می‌باشد که قبل از اجرای برنامه بر روی صفحه وجود داشته‌اند.

با زدن هر کلیدی می‌توانید از این صفحه خارج شده و به محیط بیسیک بازگردید. در قسمت آبی رنگ پایین صفحه (Immediate) تنها می‌توان یک خط برنامه نوشت که با فشردن کلید Enter به اجرا درخواهد آمد.

۲.۴. منوها (Menu)

۲.۴.۱. منوی File

با ورود به منوی فایل، گزینه‌های مختلفی مشاهده می‌شود که مهم‌ترین آن‌ها را در زیر مشاهده می‌کنید :

- New Program
- Open Program
- Save
- Save As
- Exit

گزینه New Program را زمانی انتخاب می‌کنیم که می‌خواهیم یک برنامه‌ی جدید بنویسیم. این گزینه برنامه‌ی قبلی را از محیط بیسیک پاک کرده و امکان نوشتن برنامه‌ی جدید را فراهم می‌آورد.

گزینه Open Program زمانی استفاده می‌شود که بخواهیم برنامه‌ای را که قبلاً ذخیره کردایم، باز کنیم. پس از بازکردن برنامه، می‌توانیم آن را اجرا کنیم و یا تغییرات دلخواهمان را در آن بدھیم.

از گزینه‌های Save و Save As برای ذخیره کردن برنامه روی دیسک استفاده می‌کنیم. تفاوت Save و Save As در این است که Save As همواره نام فایلی را که می‌خواهیم برنامه را در آن ذخیره کنیم، می‌پرسد و ما می‌توانیم تغییرات برنامه را در فایل‌های مختلفی ذخیره کنیم. این در حالی است که گزینه Save تغییرات برنامه را همواره روی همان فایلی که در حال حاضر باز است، ذخیره می‌کند که باعث از بین رفتن فایل قبلی می‌شود. در این حالت مراقب از دست رفتن برنامه‌ی قبلی خود باشید! علاوه بر آن سعی کنید همواره عادت کنید که بعد از هر تغییری در برنامه، آن را ذخیره کنید. برنامه‌نویسان بسیاری در دنیا به دلیل قطع برق دچار مصیبت شده‌اند!

گزینه Exit باعث خروج کامل از محیط زبان برنامه‌نویسی بیسیک می‌شود.

Edit .۲.۴. منوی

منوی Edit یا ویرایش، دستوراتی را در خود جای داده که برای ویرایش کردن برنامه استفاده می‌شوند. مهمترین گزینه‌های این منو، Undo، Cut، Copy و Paste می‌باشند.

گزینه Undo باعث بازگرداندن تغییرات ایجاد شده در برنامه به حالت قبلی می‌شود. اگر در نوشتن برنامه اشتباهی صورت بگیرد، با استفاده از فرمان Undo می‌توانیم آن را اصلاح کنیم.

فرمان‌های Paste، Cut و Copy برای کپی کردن و انتقال بخش‌هایی از برنامه به قسمت‌های دیگر به کار می‌روند. دقیقاً شبیه به نرمافزار Word شما باید ابتدا متن مورد نظرتان را انتخاب کنید. سپس با استفاده از فرمان‌های Copy یا Cut آن را به حافظه منتقل کنید. حالا می‌توانید با استفاده از فرمان Paste، متن منتقل شده به حافظه را در محل مورد نظر انتقال دهید.

برای اجرای برنامه می‌توانیم کلید F5 را بزنیم و یا از منوی Run فرمان Run را انتخاب کنیم.

Search .۳.۴. منوی

با انتخاب فرمان Find از این منو می‌توانیم عبارت مورد نظرمان را در متن برنامه بیابیم. فرمان Change نیز زمانی بکار می‌رود که می‌خواهیم عبارتی را در برنامه، به عبارت دیگری تغییر دهیم.

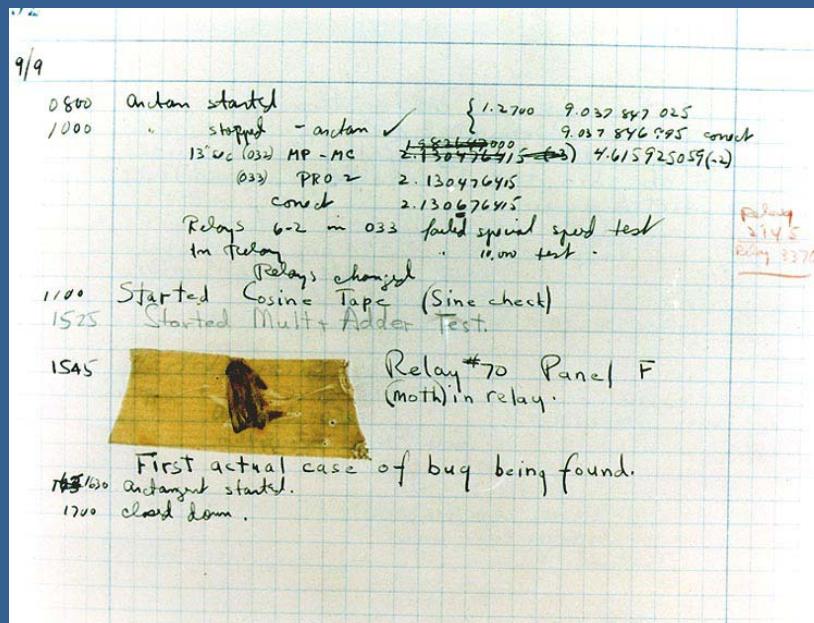
Debug .۴.۴. منوی

این منو ابزارهای اشکال‌بازی برنامه را در خود جای داده است. به علت اهمیت این ابزارها، حتماً پس از شروع برنامه‌نویسی، این بخش را به دقت مطالعه کرده و سعی نمایید به خوبی از آن در کارтан استفاده کنید.

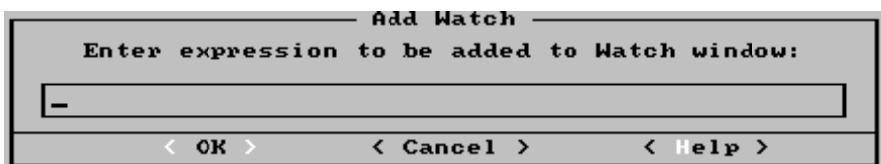


به معنای اشکال زدایی است. بسیاری اوقات در هنگام برنامه‌نویسی، برنامه خروجی مورد نظر ما را ارائه نمی‌دهد و یا بیسیک از برنامه خطای گیرد. به ویژه هنگامی که برنامه کمی بزرگ و پیچیده شود، فهمیدن اشکال کار، کار سخت و زمان بری است. ابزارهای اشکال زدایی کمک زیادی به رفع این اشکال‌ها می‌کنند.

کلمه Bug در زبان انگلیسی به معنای حشره است. ارتباط حشرات با دنیای برنامه‌نویسی از آنجا آغاز شد که در حدود شصت سال قبل، یک بید سبب از کار افتادن یک رایانه شد! تصویر بید مذکور را در زیر مشاهده می‌کنید.



اولین فرمان این منو Add Watch است. با انتخاب این فرمان، پنجره‌ی زیر ظاهر می‌شود.



اگر نام یک متغیر و یا عبارت را در این پنجره بنویسیم، مقدار آن در قسمت بالای صفحه بیسیک، در حین اجرای برنامه، به نمایش در می‌آید. ما می‌توانیم در هر مرحله از اجرای برنامه که بخواهیم، برنامه را متوقف کرده و مقدار متغیرها را در آن مرحله مشاهده نماییم. یکی از راه‌های متوقف کردن برنامه در حین اجرا، فشردن همزمان کلیدهای Ctrl و Break است. در این حالت خطی که برنامه تا قبل از آن خط اجرا شده، به صورت روشن به نمایش در می‌آید.

ابزار قدرتمند دیگر اشکال‌زدایی، اجرای مرحله به مرحلهٔ برنامه است. با فشردن کلید F8 می‌توانیم برنامه را خط به خط اجرا کنیم و تغییرات پدید آمده را زیر نظر داشته باشیم.

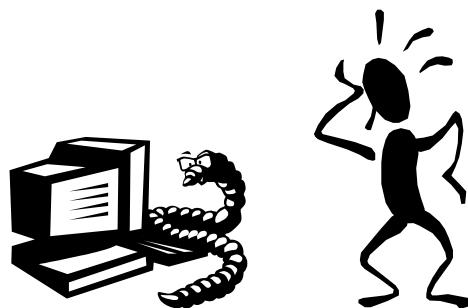
منوی Options نیز تنظیمات بیسیک را در خود جای داده است.

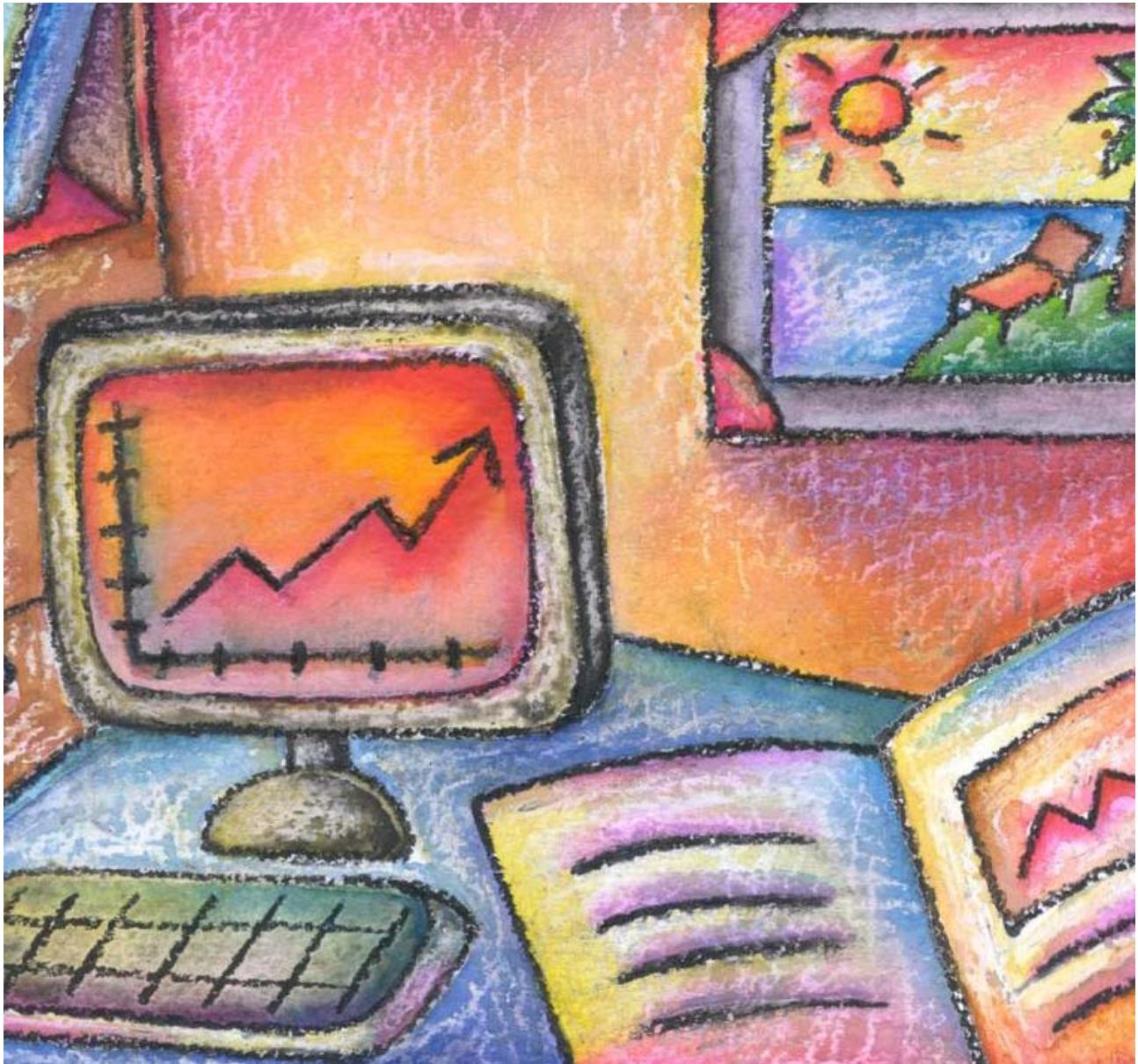


۲.۵. خطاهای برنامه

در برنامه‌نویسی چند نوع خطا وجود دارد که آشنایی با آن‌ها، کمک زیادی به صحیح نوشتگری برنامه و تصحیح برنامه می‌کند.

- خطای شکلی (Syntax Error) زمانی به وجود می‌آید که در نحوه نوشتگری یک دستور اشتباه کرده باشیم. در این موقع باید با مراجعه به مرجع برنامه‌نویسی یا راهنمای بیسیک، برنامه‌ی خود را اصلاح کنیم.
- خطای زمان اجرا (Runtime Error) به خطایی گفته می‌شود که تنها در زمان اجرای برنامه، امکان روی دادن دارد. برای مثال هنگامی که در برنامه عددی را بر صفر تقسیم کنیم و یا کاربر اطلاعات نامربوطی به برنامه بدهد، زبان بیسیک برنامه را متوقف کرده و خطای زمان اجرا می‌دهد. در این حالت باید بینیم که چه چیزی موجب ایجاد این خطا شده و آن را اصلاح کنیم.
- خطای منطقی (Logical Error) زبان بیسیک نمی‌تواند این نوع خطا را تشخیص دهد! در این حالت برنامه اجرا می‌شود ولی کار مطلوب را انجام نمی‌دهد. مثلاً اگر قرار است دو عدد را در هم ضرب کند، آن‌ها را با هم جمع می‌کند. روشن است که از نظر بیسیک خطای روی نداده، بلکه ما در نوشتگری برنامه اشتباه کرده‌ایم. در این شرایط باید برنامه را به دقت مطالعه کنیم و بینیم کدام قسمت از آن را اشتباه نوشته‌ایم. حتی ممکن است جایی از الگوریتم‌مان ایراد داشته باشد.





فصل سوم

دستورات ورودی، خروجی

و متغیرها

۳.۱. مقدمه

معمولاً هر برنامه رایانه، اطلاعاتی را از ما می‌گیرد، آن‌ها را پردازش می‌کند و سپس نتایج را به ما برمی‌گرداند. مثلاً یک برنامه‌ی صدور کارنامه مدرسه را در نظر بگیرید. این برنامه نمرات تک‌تک درس‌های تمام دانش‌آموزان را از ما می‌گیرد، (ورودی) سپس میانگین آن‌ها را حساب کرده، رتبه‌ی هر دانش‌آموز را به دست می‌آورد (پردازش) و در پایان کارنامه‌ی هر دانش‌آموز (شامل تمام نمره‌ها، میانگین هر نمره، رتبه در هر درس و رتبه کل) را به ما برمی‌گرداند. (خروجی)

در این فصل، با دستورات ورودی که اطلاعات را از ما می‌گیرند و به رایانه می‌دهند و همچنین دستورات خروجی که اطلاعات پردازش شده را برای ما نمایش می‌دهند، آشنا می‌شویم.

۳.۲. دستور خروجی

برای نمایش هر عبارت یا متنی بر روی صفحه‌ی نمایش‌گر از دستور PRINT استفاده می‌کنیم.
دستور PRINT دو حالت مختلف دارد که عبارتند از:

الف) چاپ حاصل عبارات ریاضی. برای نمایش حاصل یک عبارت ریاضی، آن را جلوی دستور PRINT می‌نویسیم.

 **نکته** : در یک عبارت ریاضی می‌توانید از اعمال جمع (+)، تفریق (-)، ضرب (*)، تقسیم (/) و توان (^) و علامت پرانتز استفاده کنید.



نمونه: هر کدام از خطاهای زیر را در محیط بیسیک تایپ کرده و اجرا کنید. خروجی هر خط را بینید.

- PRINT 27
- PRINT 2 + 14
- PRINT 3 - 7
- PRINT 20 / 5
- PRINT 3 * 72
- PRINT 5 ^ 2
- PRINT (3 + 2) * 5 - 4
- PRINT 2 ^ 4 - 3 * 5 + 8 / 4

پاسخ: خروجی دستورات بالا به صورت زیر است:

- 27
- 16
- -4
- 4
- 216
- 25
- 21
- 3

اولویت(حق تقدم) اعمال ریاضی در بیسیک

زبان بیسیک برای محاسبه عبارت‌های ریاضی، اعمال ریاضی را به ترتیب اولویت آن‌ها انجام می‌دهد. اولویت اعمال ریاضی در زبان بیسیک به ترتیب زیر است:

پرانتز

توان

ضرب و تقسیم

جمع و تفریق

یعنی بیسیک ابتدا همه عبارات داخل پرانتز را محاسبه می‌کند. سپس به سراغ توان‌ها رفته و آن‌ها را حساب می‌کند، سپس ضرب و تقسیم‌ها را از سمت چپ انجام می‌دهد و نهایتاً به سراغ جمع و تفریق می‌رود.

 **نکته** : در زبان بیسیک، کروشه و آکولاد نداریم. به جای آن‌ها می‌توانیم هر چند تا پرانتز که دلمان خواست، استفاده کنیم.

 **نمونه** : برنامه زیر را اجرا کرده و خروجی آن را ببینید.

PRINT $10 + 15 * 3 / 3 ^ 2 - 4$

: پاسخ

$$3^2 = 9 \longrightarrow 15 \times 3 = 45 \longrightarrow 45 \div 9 = 5 \longrightarrow 10 + 5 = 15 \longrightarrow 15 - 4 = 11$$

 **نمونه** : برنامه‌ای بنویسید که رایانه حاصل عبارت زیر را محاسبه کند. از کمترین تعداد پرانتز استفاده کنید.

$$\frac{(3+4)^2 - \frac{9}{3}}{2 + \frac{1}{3}} - \frac{4-3}{9}$$

: پاسخ

PRINT $((3 + 4) ^ 2 - 9 / 3) / (2 + 1 / 3) - (4 - 3) / 9$

ب) چاپ متن. برای نمایش یک متن روی صفحه نمایش‌گر، آن متن را داخل گیومه جلوی دستور PRINT می‌نویسیم.

 **نمونه** : برنامه‌ای بنویسید که بنویسد SALAM .

: پاسخ

PRINT "SALAM "



نمونه: خروجی این دو خط دستور را بنویسید.

PRINT 12 + 13

PRINT "12 + 13"

پاسخ: خط اول حاصل عبارت $12+13^3$ یعنی 25 را نمایش می‌دهد. اما خط دوم عبارت $12+13$ را عیناً نمایش می‌دهد. پس هر موقع رایانه به علامت گیومه رسید، هر چه داخل آن نوشته شده را عیناً چاپ می‌کند و هیچ محاسبه‌ای انجام نمی‌دهد.

خروجی:

25

12+13

۳.۳. متغیرها (خانه‌های حافظه)

وقتی ما اطلاعاتمان را به رایانه می‌دهیم تا آن‌ها را پردازش کرده و نتایج را نمایش دهد، این اطلاعات باید جایی ذخیره شود تا رایانه بتواند بر روی آن‌ها پردازش انجام دهد. برای ذخیره اطلاعات، از خانه‌های حافظه (متغیرها) استفاده می‌کنیم. حافظه کامپیوتر از تعدادی خانه‌ی حافظه تشکیل شده که در هر کدام می‌توان یک عدد یا یک کلمه ذخیره کرد. برای استفاده از خانه‌های حافظه (متغیرها) باید اول برای آن‌ها اسمی انتخاب کرد. اسم‌گذاری یک خانه حافظه قواعدی دارد که به شرح زیر است:



- اسم یک خانه حافظه حتماً باید با یکی از حروف الفبا شروع شود.
- اسم خانه حافظه فقط می‌تواند شامل حروف و اعداد باشد.
- اسم خانه حافظه نباید جزء دستورات بیسیک باشد.

اگر بخواهیم در داخل خانه حافظه، به جای عدد، کلمه (یا مجموعه‌ای از حروف الفبا) ذخیره کنیم، باید در انتهای اسم آن یک علامت \$ بگذاریم.



نمونه: اسامی زیر نمی‌توانند اسم خانه‌های حافظه در زبان بیسیک باشند:

1Ali

با عدد شروع شده است

Ali.Asad

نقطه دارد

Ali+Asad

علامت + دارد



Print

جزو دستورات بیسیک است



نمونه: اسامی زیر همگی به عنوان اسم خانه حافظه در زبان بیسیک قابل قبولند:

Ali



Ali23

R

Shalgham

Ali\$

۴. دستور ورودی

اکنون می‌خواهیم دستوری را یاد بگیریم که با اجرای آن، رایانه از ما اطلاعات گرفته و در داخل یک خانه حافظه ذخیره کند. این دستور INPUT است و جلوی آن اسم خانه حافظه نوشته می‌شود. مثلاً اگر بنویسیم:

INPUT A

و آن را اجرا کنیم، رایانه از ما یک عدد می‌گیرد و آن را در خانه حافظه A ذخیره می‌کند. هم چنین اگر بخواهیم رایانه یک کلمه از ما بگیرد و در داخل خانه حافظه A\$ ذخیره کند، کافی است بنویسیم:

INPUT A\$

فصل سوم

دستورات ورودی و خروجی و متغیرها

نکته : در آغاز اجرای هر برنامه، داخل تمام خانه‌های حافظه‌ی عددی که در آن برنامه استفاده شده‌اند مقدار صفر قرار دارد.

زبان بیسیک این قابلیت را دارد که جلوی یک دستور INPUT، چند خانه حافظه را بنویسیم. مثلاً

دستور:

INPUT A, B

هر گاه اجرا شود، دو عدد از ما می‌گیرد (که باید با کاما از هم جدا شوند) و آن‌ها را به ترتیب در خانه‌های حافظه A و B ذخیره می‌کند.

۳.۵. دستور خروجی، کمی بیشتر

حالا وقت آن است که کمی بیشتر قابلیت‌های دستور PRINT را بشناسیم. دستور PRINT این قابلیت را دارد که مقداری که درون هر خانه حافظه ذخیره شده را نمایش دهد. برای این کار کافی است اسم آن خانه حافظه را جلوی دستور PRINT بنویسید. مثلاً دستور A PRINT عددی که درون خانه حافظه A ذخیره شده را نمایش می‌دهد. البته اگر قبلًاً توسط یک دستور INPUT عددی را درون این متغیر ذخیره نکرده باشیم، عدد صفر نمایش داده می‌شود.

قابلیت دستور PRINT به همین جا ختم نمی‌شود. شما می‌توانید مقدار هر خانه حافظه‌ی عددی را در عبارات ریاضی هم استفاده کنید.

مثلاً دستور $A = 5 * 2 + 5$ PRINT عددی که درون خانه حافظه A هست را دو برابر کرده، سپس آن را با ۵ جمع کرده و نتیجه را نمایش می‌دهد.

نمونه : برنامه‌ای بنویسید که با اجرای آن، رایانه شعاع یک دایره را از ما بگیرد و محیط و مساحت آن را چاپ کند.

پاسخ : ابتدا دستوری می‌نویسیم که شعاع را از کاربر بگیرد و آن را درون خانه حافظه Shoa ذخیره کند (اسم خانه حافظه کاملاً اختیاری است، اما ما اسم آن را با ربط انتخاب کرده‌ایم). سپس از فرمول محیط و مساحت دایره استفاده کرده و نتایج را نمایش می‌دهیم. سپس برنامه به صورت زیر در می‌آید:

```
INPUT Shoa
PRINT Shoa * 2 * 3.14
PRINT Shoa * Shoa * 3.14
```

نمونه : کاربر پسند کردن برنامه

می‌خواهیم برنامه مثال قبل را جوری تغییر دهیم که کسی که با آن کار می‌کند راحت‌تر باشد. برای این کار می‌توانیم از پیغام‌های ورودی و خروجی استفاده کنیم. برنامه زیر را اجرا کنید و تفاوت آن را احساس کنید!

```
PRINT "Shoa ra vared konid"
INPUT Shoa
PRINT "mohit =", Shoa * 2 * 3.14
PRINT "masahat =", Shoa * Shoa * 3.14
```

همان طور که در کادر بالا دیدید، در دستور PRINT هم مثل دستور INPUT می‌توانیم چند عبارت را با یک دستور نمایش دهیم. برای این کار کافی است بین عبارتها در جلوی دستور PRINT، از علامت ویرگول (,) یا نقطه ویرگول (;) استفاده کنیم. فرق این دو علامت این است که اگر از علامت ویرگول بین عبارتها استفاده کنیم، عبارت دوم در ناحیه بعدی خط نمایش داده می‌شود (هر خط روی صفحه نمایش گر به چند ناحیه تقسیم می‌شود) ولی اگر از نقطه ویرگول استفاده کنیم، عبارت دوم چسبیده به عبارت اول چاپ می‌شود.

فصل سوم

دستورات ورودی و خروجی و متغیرها

 نمونه : خروجی برنامه زیر را بینید و تحلیل کنید.

```
PRINT 2 * 3 - 4 , 4 ^ 2 ; 3  
PRINT 2 * 3 - 4 ; 4 ^ 2 , 3
```

پاسخ :

```
2           16   3  
2 16       3  
↑
```

محل شروع ناحیه دوم

۳.۶. تمرین‌های برنامه نویسی

۱. خروجی برنامه زیر را بنویسید.

```
PRINT 11 ^ 2 + 3 * 4 + 9 / 3 ^ 2
```

۲. برنامه‌ای بنویسید که طول و عرض یک مستطیل را گرفته، محیط و مساحت آن را نمایش دهد.

۳. برنامه‌ای بنویسید که شعاع یک کره را گرفته، حجم و مساحت آن را نمایش دهد.

$$r^2 \pi = \text{مساحت کره با شعاع } r$$

$$\frac{4}{3} \pi r^3 = \text{حجم کره با شعاع } r$$

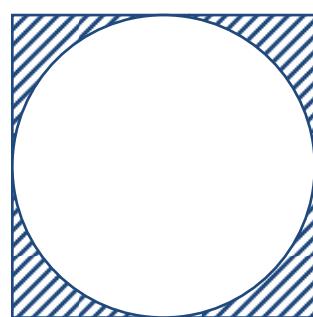
۴. برنامه‌ای بنویسید که ۳ عدد از ما بگیرد و میانگین آن‌ها را نمایش دهد.

۵. برنامه‌ای بنویسید که جرم و حجم یک ماده را از ما بگیرد و چگالی آن را حساب کرده، نمایش دهد.

$$\frac{\text{حجم}}{\text{چگالی}} = \text{چگالی (جرم حجمی)} \text{ یک ماده}$$

۴. برنامه‌ای بنویسید که وزن یک غذا و درصد پروتئین آن را از ما بپرسد و وزن پروتئین آن غذا را نمایش دهد.

۵. برنامه‌ای بنویسید که شعاع دایره‌ی شکل زیر را بگیرد و مساحت قسمت هاشور خورده را نمایش دهد.



۶. برنامه تمرین ۴ را تا جایی که می‌توانید کاربر پسند کنید.





فصل چهارم

جایگزینی، شرط، پرسش، توابع

۴.۱. مقدمه

در این فصل با چند دستور پرکاربرد دیگر در زبان بیسیک آشنا می‌شویم. دستور جایگزینی برای قرار دادن یک مقدار درون یک خانه حافظه (بدون پرسیدن از کاربر) مورد استفاده قرار می‌گیرد. هم چنین دستورات شرطی را برای گذاشتن شرط در برنامه‌هایمان یاد می‌گیریم. در آخر فصل هم با تعدادی از توابع پرکاربرد در زبان بیسیک آشنا می‌شویم.

.....

۴.۲. دستور جایگزینی (LET)

قبل‌اً در دستور INPUT با گرفتن یک مقدار از کاربر و قرار دادن آن درون یک خانه حافظه آشنا شده‌اید. اکنون این سوال پیش می‌آید که اگر بخواهیم یک مقدار معلوم (که آن را از کاربر نمی‌پرسیم) درون یک خانه حافظه برویزیم، چه باید بکنیم.

برای این منظور از دستور LET استفاده می‌کنیم. شکل کلی این دستور به صورت زیر است:

عبارت = نام خانه حافظه LET

این دستور حاصل عبارت سمت راست را حساب کرده و آن را درون خانه حافظه (متغیر) مشخص شده می‌ریزد.

توجه: در دستور LET همواره نام خانه حافظه در سمت چپ و عبارت در سمت راست نوشته می‌شود. یعنی اگر بخواهیم عدد 2 را درون خانه حافظه A برویزیم، باید بنویسیم: $LET A = 2$ ، نه این که بنویسیم: $LET 2 = A$.

 نمونه: چند کاربرد دستور LET :

LET B = 34

LET Salam = 12 * 5 - 6 * 3

LET Ali = (2 + 5) * 3 / 7

LET A = B * 2 + 5

بعد از اجرای دستورات فوق، محتوای خانه‌های حافظه B، Salam و Ali به ترتیب برابر 34 ، 42 و 3 می‌شود. همچنین در خانه حافظه A ، 2 برابر مقدار خانه حافظه B به علاوه‌ی 5 قرار می‌گیرد.

توجه:

. ۱. دستور $LET A+B = C+D$ در زبان بیسیک بی معنی است، چون در سمت چپ مساوی حتماً باید اسم یک خانه حافظه بیاید، اما A + B اسم خانه حافظه نیست.

. ۲. در زبان بیسیک $LET B = A$ و $LET A = B$ با هم تفاوت دارند، دستور اول محتوای خانه حافظه B را درون خانه حافظه A می‌ریزد و دستور دوم محتوای خانه حافظه A را درون B می‌ریزد.

. ۳. وقتی دستور $LET A = B$ اجرا می‌شود، محتوای خانه حافظه B درون خانه حافظه A ریخته می‌شود، اما محتوای خانه حافظه B خودش دست نخورده باقی می‌ماند. یعنی مثلاً اگر قبل از اجرای این دستور، درون خانه حافظه A عدد 3 و درون خانه حافظه B عدد 5 قرار گرفته باشد، بعد از اجرای دستور، درون هر دو خانه حافظه، عدد 5 قرار دارد.

. ۴. می‌توانید دستور LET را در زبان بیسیک نویسید، یعنی دو خط زیر در زبان بیسیک یک معنی دارند:

LET A = 2 * 3

A = 2 * 3

. ۵. در هر خانه حافظه تنها یک عدد می‌توانیم ذخیره کنیم. یعنی وقتی مقداری را با دستور LET درون یک خانه حافظه می‌ریزیم، مقدار قبلی آن از بین می‌رود.

 **نمونه:** برنامه‌ای بنویسید که محتوای خانه حافظه A را از کاربر بپرسد. سپس آن را ۵ برابر کرده، درون خانه حافظه B بربزد و نهایتاً محتوای خانه حافظه را نمایش دهد.

: پاسخ

```
INPUT A
LET B = A * 5
PRINT B
```

 **نمونه:** برنامه‌ای بنویسید که مقدار x را بگیرد و حاصل عبارت $u^2 + 2u + 4$ را با شرط $u = \frac{1+x}{1-x}$ نمایش دهد.

: پاسخ

```
INPUT x
LET u = (1 + x) / (1 - x)
PRINT u ^ 2 + 2 * u + 4
```

 **نمونه:** برنامه‌ای بنویسید که دو عدد گرفته و در خانه‌های حافظه A و B قرار دهد. سپس محتوای این دو خانه حافظه را با هم عوض کند.

پاسخ: شاید اولین ایده‌ای که به ذهن مان برسد به صورت زیر باشد:

```
INPUT A, B
LET A = B
LET B = A
```

اما این برنامه اشتباه است. بباید خط به خط آن را دنبال کرده و اشکالش را بباییم. فرض کنید کاربر محتوای خانه‌های حافظه A و B را به ترتیب 2 و 3 وارد کند. پس بعد از اجرای خط اول، محتوای A و

B به صورت زیر است:

A	B
2	3

بعد از اجرای خط دوم، محتوای B درون A ریخته می‌شود، یعنی داریم:

A	B
3	3

بعد از اجرای خط سوم، محتوای A درون B ریخته می‌شود، یعنی داریم:

A	B
3	3

می‌بینید که محتوای خانه حافظه A در خط دوم از بین می‌رود و دیگر به آن دسترسی نداریم تا در B بریزیم. پس برای رفع این مشکل، قبل از خط دوم محتوای A را در جایی دیگر (مثلاً خانه حافظه C) ذخیره می‌کنیم و در خط آخر آن را درون B می‌بریزیم. یعنی برنامه درست به این شکل است:

```
INPUT A, B
LET C = A
LET A = B
LET B = C
```

این برنامه را خط به خط تعقیب کنید و درستی آن را تحقیق کنید.

آیا می‌دانید؟

زبان بیسیک برای عوض کردن محتوای دو خانه حافظه با هم دیگر یک دستور دارد. این دستور

SWAP A, B

محتوای دو خانه حافظه A و B را با هم عوض می‌کند.

برای ریختن یک کلمه یا جمله در خانه حافظه از نوع کلمه‌ای، از دستور LET استفاده می‌کنیم و کلمه یا جمله را درون گیومه می‌گذاریم. مثلاً دستور:

LET A\$ = "SALAM"

کلمه‌ی SALAM را درون خانه حافظه $\$A$ ذخیره می‌کند. باز هم دقت کنید که عبارت داخل گیومه بدون هیچ تغییری در متغیر ذخیره می‌گردد.

۴. ۳. دستور شرط

می‌خواهیم برنامه‌ای بنویسیم که با گرفتن طول و عرض یک چهار ضلعی بگوید آن چهار ضلعی مربع است یا مستطیل. مسلماً با دستوراتی که تا به حال خوانده‌ایم این کار امکان ندارد. زبان بیسیک برای مقایسه و شرط هم دستوراتی دارد که در این قسمت آن‌ها را یاد می‌گیریم.

زبان بیسیک دستوری دارد که در صورت درست بودن یک شرط، کاری را انجام می‌دهد و در صورت غلط بودن آن به خط بعدی می‌رود. این دستور به شکل زیر است:

دستور IF شرط THEN

وقتی رایانه به خط بالا می‌رسد، شرط بعد از کلمه IF را بررسی می‌کند. اگر این شرط درست بود، دستور بعد از کلمه‌ی THEN را اجرا می‌کند، و گزنه هیچ کاری انجام نمی‌دهد و به خط بعدی می‌رود.

مثالاً هر گاه رایانه به دستور:

IF A < 2 THEN PRINT "SALAM"

بررسد، بررسی می‌کند که آیا محتوای خانه حافظه A کمتر از ۲ است یا خیر. اگر کمتر بود، کلمه SALAM را نمایش می‌دهد و در غیر این صورت کاری نمی‌کند و به خط بعدی می‌رود.

توجه:

شرطی که جلوی دستور IF وجود دارد، غالباً مقایسه بین دو مقدار است، مثلاً:

IF A = B THEN دستور

IF A * 2 < B THEN دستور

IF C * 2 + B > K THEN دستور

علامت‌های مقایسه دیگر در زبان بیسیک به صورت زیر هستند:

شرط	علامت در بیسیک		
\leq	$<=$	یا	$=<$
\geq	$>=$	یا	$=>$
\neq	$<>$	یا	$><$

یک دستور جدید :

هر جای برنامه دستور END را بنویسید، اجرای برنامه همان‌جا خاتمه می‌یابد.



حال می‌خواهیم برنامه‌ای که در ابتدای این بخش گفتم (تشخیص مستطیل یا مربع بودن) را بنویسیم:

INPUT Tool , Arz

IF Tool = Arz THEN PRINT "MORABBAE"

IF Tool <> Arz THEN PRINT "MOSTATIL"

مطلوب دیگر این است که بخواهیم در صورت درست بودن شرط، به جای یک دستور، چند دستور

اجرا شود، باید چه کنیم؟

چاره‌ی کار به شکل زیر است:

IF شرط THEN

 _ دستور ۱

 _ دستور ۲

 ...

END IF

یعنی تمام دستوراتی که می‌خواهیم در صورت درست بودن شرط اجرا شوند را بعد از THEN (هر کدام در یک خط) می‌نویسیم و در آخر کلمه IF و END IF را می‌نویسیم. رایانه با دستوراتی که بعد از THEN و قبل از END IF نوشته شده‌اند به صورت یک مجموعه رفتار می‌کند، یعنی یا همه آن‌ها را اجرا می‌کند یا هیچ کدام را.

دستورات شرطی زبان بیسیک قابلیت دیگری نیز دارند. فرض کنید می‌خواهید رایانه شرطی را بررسی کند. اگر شرط درست بود دستور شماره ۱ و اگر شرط غلط بود دستور شماره ۲ را اجرا می‌کند. با این که با دو دستور IF می‌توان چنین کاری انجام داد، اما زبان بیسیک دستور ویژه‌ای هم برای این کار دارد که به شکل زیر است:

دستور IF دستور THEN شرط دستور ELSE

کلمات IF و THEN و ELSE به همین ترتیب نوشته می‌شوند. اگر شرط جلوی IF درست باشد، دستور ۱ و اگر غلط باشد دستور ۲ اجرا می‌شود.

 **نمونه:** برنامه‌ای بنویسید که دو عدد از کاربر بگیرد و سپس آن‌ها را به ترتیب بزرگ به کوچک نمایش دهد.

: پاسخ

```
INPUT A , B
IF A > B THEN PRINT A , B ELSE PRINT B , A
```

اگر بخواهیم در صورت درست بودن شرط، چند دستور خاص و در صورت غلط بودن آن چند دستور خاص دیگر اجرا شوند، می‌توانیم از شکل زیر استفاده کنیم:

IF شرط THEN

مجموعه دستورات ۱

ELSE

مجموعه دستورات ۲

END IF

در این حالت در صورت درست بودن شرط، مجموعه دستورات ۱ و در صورت غلط بودن آن مجموعه دستورات ۲ اجرا می‌شود.

 نمونه: برنامه‌ای بنویسید که نمره‌ی یک دانش‌آموز را بگیرد و بگوید قبول است یا تجدید.

: پاسخ

INPUT A

IF A >= 10 THEN PRINT "GHABOOL" ELSE PRINT "TAJDID"

 آیا می‌دانید؟

برای ترکیب چند شرط می‌توان بین آن دو از دو کلمه **AND** به معنی « و » و **OR** به معنی « یا » استفاده کرد. وقت کنید که در بیسیک بر خلاف ریاضی، شرط‌های دوتایی (مثلًاً به صورت 10 < A < 20) نداریم.

 نمونه: برنامه‌ای بنویسید که سن کاربر را بپرسد و فقط اگر کاربر نوجوانی بین ۱۰ تا ۱۵ سال بود

به او سلام کند!

پاسخ :

```
INPUT sen
IF sen > 10 AND sen < 15 THEN PRINT "SALAM NOJAVOON!"
```

 نمونه: در مثال قبل اگر کاربر نوجوان نباشد از او خدا حافظی کنید!

پاسخ :

```
INPUT sen
IF sen < 10 OR sen > 15 THEN PRINT "khodahafez !!"
```

۴. دستور پرش



دستوری که می‌خواهیم در این بخش آن را به شما یاد بدهیم، دستور خوبی نیست (!) و خیلی از برنامه نویسان بزرگ از آن استفاده نمی‌کنند. دلیل استفاده نکردن از آن هم این است که این دستور، دنبال کردن اجرای برنامه را برای ما سخت می‌کند. فرض کنید بنا به دلایلی می‌خواهید از یک خط برنامه به خط دیگری بپرید. برای این کار هم زبان بیسیک دستور ویژه‌ای به شکل زیر دارد:

GOTO شماره خط

وقتی رایانه به دستور بالا برسد، می‌رود به سراغ خطی از برنامه که شماره آن را جلوی دستور GOTO نوشته‌ایم.

 **توجه:** برای هر خط از برنامه که خواستید شماره بگذارید، کافی است در ابتدای آن خط از برنامه، شماره‌اش را بنویسید و بعد از یک فاصله، دستور آن خط را بنویسید.

 **نمونه:** برنامه‌ای بنویسید که از کاربر دو عدد به همراه حاصل ضرب آن‌ها را بپرسد. اگر حاصل ضرب درست بود بنویسد AFARIN و برنامه تمام شود. در غیر این صورت دوباره حاصل ضرب را بپرسد و آن قدر این کار را تکرار کند تا حاصل ضرب درست وارد شود.

: پاسخ

```

INPUT A , B
10 INPUT hasel_zarb
IF hasel_zarb = A * B THEN
    PRINT " AFARIN !"
    END
ELSE
    PRINT " ESHTEBAH BOOD !"
    GOTO 10
END IF

```

۴.۵. توابع پر کاربرد

برای انجام بعضی از اعمال ریاضی در زبان بیسیک دستوراتی وجود دارد که در این بخش مهم‌ترین آن‌ها را می‌گیریم.

۴.۵.۱. تابع قسمت صحیح

اگر بخواهید رایانه قسمت صحیح (عدد صحیح قبل از) هر عدد را برای شما حساب کند، کافی است از عبارت زیر استفاده کنید:

INT (عبارت)

دقّت کنید که خط بالا دستور نیست، بلکه یک عدد است که آن را می‌توان نمایش داد (با دستور PRINT) یا در عبارت‌ها استفاده کرد.

 نمونه:

```
PRINT INT (7.4)
PRINT INT (3)
PRINT INT (-1.5)
LET A = INT (7.4)
LET A = INT (B)
```

پاسخ : خطوط اول تا سوم به ترتیب اعداد 7 ، 3 و -2 را نمایش می‌دهند(توجه کنید که عدد صحیح قبل از -1.5 ، برابر -2 است). خط چهارم مقدار 7 را در A ذخیره می‌کند و خط پنجم قسمت صحیح عدد ذخیره شده در خانه حافظه B را در A ذخیره می‌کند.

 نمونه:

پاسخ : هر عددی قسمت صحیح‌اش با خودش برابر باشد، صحیح است و در غیر این صورت اعشاری است.

```
INPUT A
IF A = INT (A) THEN PRINT "SAHIH" ELSE PRINT "ASHARI"
```

۴.۵.۲. تابع تولید عدد تصادفی

برای تولید یک عدد تصادفی بین ۰ و ۱ از دستور RND استفاده می‌کنیم. رایانه هر وقت به کلمه RND بر بخورد، به جای آن یک عدد اتفاقی اعشاری بین ۰ و ۱ قرار می‌دهد. (امکان دارد خود صفر تولید شود، اما یک نه)

 نمونه:

```
PRINT RND
A = RND
```

خط اول، یک عدد اتفاقی نمایش داده و خط دوم یک عدد اتفاقی را در خانه حافظه A ذخیره می‌کند.

برای تولید یک عدد اتفاقی صحیح بین A و B (با امکان تولید خود A و B) می‌توان از دستور زیر استفاده کرد:

INT(RND * (B - A + 1)) + A

 **نمونه:** برنامه‌ای بنویسید که با اجرای آن رایانه مثل تاس، یک عدد اتفاقی صحیح بین ۱ تا ۶ (شامل خود ۱ و ۶) بنویسد.

پاسخ:

PRINT INT (RND * 6) + 1

 **توجه:** اگر برنامه بالا را چند دفعه پشت سر هم اجرا کنید، ممکن است همان جواب قبلی را می‌نویسد. این به این دلیل است که شما تصادفی بودن عددها را فعال نکرده‌اید. برای فعال کردن دستور RND، در ابتدای هر برنامه‌ای که در آن از دستور RND استفاده شده، دستور زیر را عیناً بنویسید:

Randomize timer

۴.۵.۳. عملگر باقی‌مانده

برای بدست آوردن باقی‌مانده‌ی تقسیم A بر B می‌توانید از دستور زیر استفاده کنید:

A MOD B

 **نمونه:**

PRINT 24 MOD 7
PRINT 6 MOD 11

خط اول عدد 3 و خط دوم عدد 6 را نمایش می‌دهد.

۴.۵. عملگر خارج قسمت

برای به دست آوردن خارج قسمت صحیح تقسیم A بر B، ضمن استفاده از تابع INT، می‌توانید از دستور زیر هم استفاده کنید:

A \ B

 نمونه:

```
PRINT INT (34 / 7)
PRINT 34 \ 7
LET A = 32 \ 5
```

خطهای اول و دوم، هر دو عدد 4 را نمایش می‌دهند و خط دوم عدد 6 را درون A ذخیره می‌کند.

۴.۶. تمرین‌های برنامه نویسی

برنامه‌ای بنویسید که رایانه ولتاژ دو سر یک لامپ و حداکثر ولتاژ قابل تحمل آن لامپ را بپرسد. سپس بنویسد لامپ می‌سوزد یا نه.

برنامه‌ای بنویسید که با اجرای آن رایانه سه عدد از ما گرفته و میانگین آن‌ها را بنویسد. سپس روی صفحه بنویسد:

aya edame midahid?

1. bale
2. na

آنگاه یک عدد از ما بگیرد. اگر 1 بود به ابتدای برنامه برود و اگر 2 بود برنامه پایان پذیرد.

.**۳** برنامه‌ای بنویسید که رایانه یک عدد از ما بگیرد. اگر عدد برابر ۱۳۸۶ نبود، برنامه پایان پذیرد.
اگر بود، چهار عدد بگیرد و میانگین آن‌ها را بنویسد.

.**۴** برنامه‌ای بنویسید که یک کلمه پرسد. اگر کلمه SALAM نبود، برنامه پایان پذیرد. در غیر این صورت دو عدد گرفته، باقی‌مانده تقسیم اولی بر دومی را بنویسد.

.**۵** برنامه‌ای بنویسید که رایانه با گرفتن ساعت به ما صبح به خیر، ظهر به خیر، عصر به خیر و شب به خیر بگوید.

تا ۱۰: صبح به خیر

۱۰ تا ۱۴: ظهر به خیر

۱۴ تا ۱۹: عصر به خیر

۱۹ به بعد: شب به خیر

.**۶** برنامه‌ای بنویسید که قابلیت انجام کارهای زیر را داشته باشد:

(۱) محاسبه محیط و مساحت مستطیل

(۲) محاسبه محیط و مساحت دایره

(۳) محاسبه میانگین سه عدد

پس از اجرای برنامه باید ابتدا نوشته شود:

Kodam Ra Mikhahid?

1. Mostatil
2. Dayereh
3. Miangin

سپس رایانه یک عدد از کاربر بگیرد و بر مبنای آن به قسمت انتخاب شده رفته و آن را اجرا کند و بعد برنامه تمام شود.

.**۷** برنامه‌ای بنویسید که دو عدد گرفته و خارج قسمت صحیح و باقی‌مانده تقسیم اولی بر دومی را بدون استفاده از دستورات MOD و \ بنویسد.

.**۷.** برنامه‌ای بنویسید که دو عدد گرفته و بگوید عدد اول بر عدد دوم بخش‌پذیر است یا نه.

.**۸.** برنامه‌ای بنویسید که یک عدد گرفته و آن را گرد کرده و حاصل را نمایش دهد.

.**۹.** برنامه‌ای بنویسید که یک عدد بپرسد و بگوید زوج است یا فرد.

.**۱۰.** برنامه‌ای بنویسید که یک عدد صحیح اتفاقی بین ۷ تا ۵۴۰ بنویسد. (امکان نوشتن خود ۷ و ۵۴۰ هم وجود داشته باشد)

.**۱۱.** برنامه‌ای بنویسید که یک دانش‌آموز به کمک آن بتواند ضرب را تمرین کند. به این شکل که رایانه ابتدا دو عدد صحیح اتفاقی بین ۱۰ تا ۹۹ بنویسد. سپس حاصل ضرب آن دو را بپرسد. اگر حاصل ضرب درست گفته شده بود، بنویسد AFARIN. اگر هم غلط بود، حاصل ضرب صحیح را بنویسد.

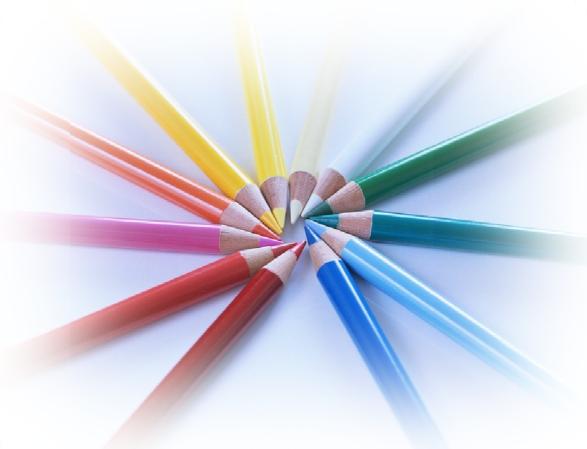
.**۱۲.** کنار دستورات درست "ص" و کنار دستورات غلط "غ" بگذارید.

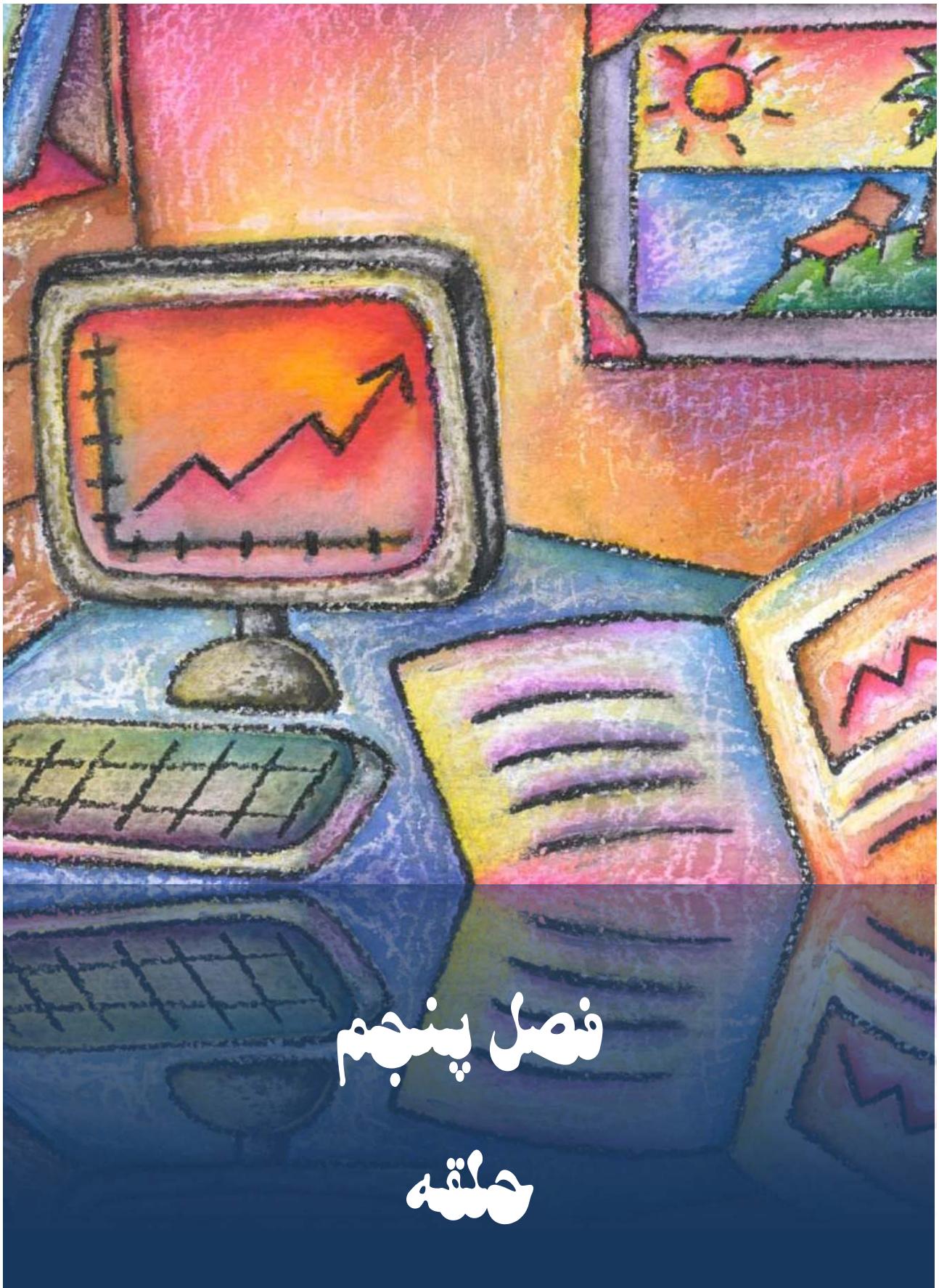
- INPUT Komput
- INPUT PRINT
- PRINT Shalgham
- IF A THEN PRINT B
- PRINT 2A , Salam
- IF A = 1 THEN INPUT B4
- IF A > B * 2 PRINT Salam
- PRINT A * 2 ; B + 4 , Salam
- IF A = "3" THEN PRINT "Salam"
- INPUT A * 2 , B + 4
- PRINT Sal ; am

- PRINT "GOTO IF "
- IF A + B = 2 ^ 3 THEN PRINT 2 ^ 3
- PRINT {s * 2 ^ (8 + 4)}
- IF A = 5 THEN A + 1
- IF Folan = Folani THEN IF A = 2 THEN END

۱۴. خروجی برنامه زیر را بنویسید.

```
10 LET A = 3
20 LET B = A - 1
30 LET C = B * A
40 PRINT C
50 LET A = A + 1
60 LET B = B + 1
70 IF C < 40 THEN GOTO 20
```





فصل پنجم

حلقه

۱.۵. مقدمه

یکی از دلایل استفاده از رایانه‌ها، انجام اعمال تکراری با دقّت و سرعت بالا است. بسیار پیش می‌آید که ما برنامه‌ای برای انجام کارهای تکراری نیاز داشته باشیم. مثلاً فرض کنید می‌خواهیم برنامه‌ای بنویسیم که تمام اعداد اول کوچکتر از ۱۰۰۰ را نمایش دهد. باید این اعداد را بررسی کنیم و هر کدام را که اول هستند نمایش دهیم. پس باید عمل بررسی اول یا مرکب بودن اعداد را ۱۰۰۰ بار تکرار کنیم. در زبان بیسیک برای انجام کارهای تکراری از حلقه‌ها استفاده می‌کنیم. در این فصل دو نمونه از حلقه‌های پر کاربرد زبان بیسیک و طرز ساخت آن‌ها را یاد می‌گیریم.

.....

۲. حلقه‌ی FOR

می‌خواهیم برنامه‌ای بنویسیم که اعداد ۱ تا ۱۰۰۰ را نمایش دهد. با توجه به دستوراتی که در فصل‌های گذشته خوانده‌ایم، می‌توانیم یک متغیر به اسم A انتخاب کنیم و عدد ۱ را در آن بگذاریم. سپس آن را چاپ کرده و یکی به مقدارش اضافه کنیم و این کار تکراری را ۱۰۰۰ بار انجام دهیم. با این کار به برنامه زیر می‌رسیم:

```
LET A = 1
PRINT A
LET A = A + 1
IF A <= 1000 THEN GOTO 10
```

به خط سوم برنامه بالا دقّت کنید. این خط به محتوای خانه حافظه A یک واحد اضافه می‌کند. هم چنین به علامت $=$ در خط چهارم دقّت کنید. اگر به جای آن از علامت $<$ استفاده کنیم، عدد 1000 نمایش داده نمی‌شود.

توجه: برای افزودن X واحد به محتوای خانه حافظه A می‌توانیم از این دستور استفاده کنیم:

```
LET A = A + X
```

چون تکرار یک کار در زبان بیسیک زیاد مورد استفاده دارد، در زبان بیسیک دستور خاصی هم برای تکرار داریم. این دستور به شکل زیر است:

مقدار نهایی TO مقدار اولیه = نام متغیر FOR
دستورات
نام متغیر NEXT

رایانه به این شکل با سه خط بالا رفتار می‌کند: ابتدا مقدار اولیه را درون متغیر ذخیره می‌کند. سپس دستوراتی که بین خط FOR و خط NEXT نوشته شده را انجام می‌دهد. بعد به خط NEXT می‌رسد. کار این خط این است که یک واحد به محتوای متغیر مورد استفاده اضافه می‌کند. سپس رایانه بررسی می‌کند که آیا محتوای فعلی متغیر از مقدار نهایی اش که در خط FOR نوشته‌ایم، بیشتر شده یا نه. اگر نه، دوباره تمام دستورات بین FOR و NEXT را انجام می‌دهد. بعد دوباره به خط NEXT بیشتر نبود، دوباره تمام دستورات بین FOR و NEXT را انجام می‌دهد. آن قدر این رویه تکرار نهایی بیشتر شده یا نه. اگر بیشتر نشده بود، دوباره این کارها را انجام می‌دهد و آن قدر این رویه تکرار می‌شود تا در خط NEXT، محتوای متغیر از مقدار نهایی که برایش مشخص کردہ‌ایم، بیشتر شود. هر موقع چنین اتفاقی افتاد دیگر رایانه به خطهای بالای NEXT برنمی‌گردد و به سراغ خط بعد از NEXT می‌رود و ادامه‌ی برنامه را اجرا می‌کند.

 **نمونه** : برنامه‌ای بنویسید که طول و عرض یک مستطیل را گرفته، محیط و مساحت آن را نمایش دهد و این کار را ۱۰۰ بار تکرار کند.

پاسخ :

```
FOR A = 1 TO 100
  INPUT X, Y
  PRINT (X+Y) * 2, X * Y
NEXT A
```

مقدار اولیه متغیر A برابر ۱ و مقدار نهایی اش برابر ۱۰۰ است. پس دستورات بین FOR و NEXT ۱۰۰ بار تکرار می‌شوند.



نمونه : برنامه‌ای بنویسید که اعداد ۱ تا ۱۰۰۰ را نمایش می‌دهد.

پاسخ :

```
FOR A = 1 TO 1000
    PRINT A
NEXT A
```

سعی کنید با توجه به توضیحاتی که در مورد روش کار دستور FOR گفتیم، این برنامه را خط به خط دنبال و اجرا کنید.



نمونه : برنامه‌ای بنویسید که اعداد زیر را بنویسد:

$1^2, 2^2, 3^2, \dots, 20^2$

پاسخ :

```
FOR A = 1 TO 20
    PRINT A ^ 2
NEXT A
```



نمونه : برنامه‌ای بنویسید که یک عدد گرفته، تمام مقسوم علیه‌های آن را بنویسد.

پاسخ :

```
INPUT A
FOR X = 1 TO A
    IF A MOD X = 0 THEN PRINT X
NEXT X
```

در این برنامه از عدد ۱ تا عدد گرفته شده (یعنی A) را بررسی می‌کیم. عدد گرفته شده بر هر کدام از این اعداد بخش‌پذیر بود، آن عدد مقسوم علیه‌اش است. برای بخش‌پذیر بودن یک عدد بر یک عدد دیگر کافی است باقی‌مانده‌ی تقسیم اولی بر دومی صفر شود.

 نمونه : برنامه‌ای بنویسید که یک عدد گرفته، تعداد مقسوم علیه‌های آن را بنویسد.

پاسخ : از همان ابتدای برنامه مثال قبل استفاده می‌کنیم. تنها فرق برنامه جدید با قبلی این است که به جای نمایش مقسوم علیه‌ها، باید تعدادشان را بشماریم. برای شمردن تعداد مقسوم علیه‌ها از این روش استفاده می‌کنیم؛ یک متغیر (مثلاً به اسم C) انتخاب می‌کنیم. هر وقت یک مقسوم علیه از عددمان پیدا کردیم، یک واحد به محتوای C اضافه می‌کنیم. در این صورت کار متغیر C می‌شود شمردن تعداد مقسوم علیه‌ها. به این روش، تکنیک شمارش می‌گویند و معمولاً برای شمردن تعداد یک چیز از آن استفاده می‌کنیم. پس برنامه به صورت زیر در می‌آید:

```
INPUT A
FOR X = 1 TO A
    IF A MOD X = 0 THEN LET C = C + 1
NEXT X
PRINT C
```

توجه کنید که بعد از خاتمه‌ی کار حلقه و شمارش کل مقسوم علیه‌ها، C را نمایش داده‌ایم.
(در خط آخر)

 نمونه : برنامه‌ای بنویسید که ۱۰۰ عدد گرفته، در پایان حاصل جمع تمام آن‌ها را بنویسد.

پاسخ : برای حل این مثال، یک خانه حافظه برای ذخیره حاصل جمع (مثلاً به اسم Bank) انتخاب می‌کنیم. هر کدام از ۱۰۰ عدد را که گرفتیم، آن را به محتوای این خانه حافظه اضافه می‌کنیم. در مثال‌های قبلی دیدیم که اگر بخواهیم x واحد به محتوای خانه حافظه Bank اضافه کنیم، کافی است بنویسیم:

```
LET Bank = Bank + x
```

برنامه مورد نظر به صورت زیر در می‌آید:

```
FOR A = 1 TO 100
    INPUT x
    LET Bank = Bank + x
NEXT A
PRINT Bank
```


نکته عملی :

اگر بخواهید درستی برنامه نوشته شده در بالا را روی رایانه امتحان کنید، به مشکل بر می خورید. چون وارد کردن ۱۰۰ عدد و بررسی صحت حاصل جمع آنها کار سختی است. برای امتحان برنامه بالا بهترین کار این است که در خط اول به جای ۱۰۰، مثلاً ۵ بنویسید و برای ۵ عدد برنامه را امتحان کنید.

۳. گام حلقه

یک نکته دیگر در مورد دستور FOR باقی می ماند و آن این است که فرض کنید ما بخواهیم در خط NEXT به جای این که هر دفعه ۱ واحد به محتوای متغیرمان اضافه شود، یک مقدار دیگر (مثلاً ۲ واحد یا ۳ واحد یا حتی ۱.۵ واحد یا ۰.۵ واحد) اضافه شود. در زبان بیسیک قابلیت این کار پیش بینی شده است. برای عوض کردن گام حلقه به عددی غیر از ۱، کافی است جلوی خط FOR، کلمهی STEP و سپس گام مورد نظر را بنویسید.


نمونه : برنامه‌ای بنویسید که اعداد زوج ۲ تا ۱۰۰ را بنویسد.

پاسخ :

```
FOR A = 2 TO 100 STEP 2
PRINT A
NEXT A
```

مشاهده می کنید که برای ساختن اعداد زوج و نمایش آنها، از یک حلقه با گام ۲ استفاده کردایم.


نمونه : برنامه‌ای بنویسید که اعداد زیر را به ترتیب از سمت چپ بنویسد:

۳۰۰ , ۲۹۷ , ۲۹۴ , ۲۹۱ , ... , ۳ , ۰

پاسخ :

```
FOR A = 300 TO 0 STEP -3
PRINT A
NEXT A
```

توجه:

۱. موقعی که گام حلقه منفی باشد، زمانی کار حلقه خاتمه می‌یابد که مقدار متغیر از مقدار نهایی که برای آن مشخص کرده‌ایم، کمتر شود.
۲. گام حلقه، مقدار نهایی و مقدار اولیه می‌توانند هر عدد مثبت یا منفی یا اعشاری باشند.
۳. اگر در حلقه‌ای با گام مثبت (گام منفی) مقدار اولیه از مقدار نهایی بزرگ‌تر (کوچک‌تر) باشد، دستورات داخل حلقه اصلاً اجرا نمی‌شوند و رایانه به سراغ دستورات بعد از خط NEXT می‌رود.
۴. اگر مقدار اولیه و نهایی یک حلقه با هم مساوی باشند، دستورات داخل حلقه فقط یکبار اجرا می‌شوند.

نمونه: برنامه زیر چه کار می‌کند?

```
FOR A = 1 TO 5 STEP 0
    PRINT A
NEXT A
```

پاسخ: چون گام حلقه صفر است، هر بار در خط NEXT، صفر واحد به محتوای A اضافه می‌شود، یعنی محتوای آن در خط NEXT اصلاً تغییری نمی‌کند. پس مقدار آن هم هیچ‌گاه از مقدار نهایی بیشتر نمی‌شود. پس این حلقه اصلاً تمام نمی‌شود، یعنی یک حلقه پایان ناپذیر داریم که بی‌نهایت ۱ می‌نویسد.

نکته:

اگر خواستید در حین اجرای برنامه، آن را متوقف کنید، کلیدهای Ctrl و Break را با هم فشار دهید. در برنامه بالا اگر این کلیدها را بگیرید، تا موقعی که برق نرود، رایانه به نوشتن عدد ۱ ادامه می‌دهد!

 **نمونه:** برنامه‌ای بنویسید که ۱۰۰ عدد گرفته، بزرگ‌ترین آن‌ها را بنویسد.

پاسخ: روش کار بدین صورت است که یک خانه حافظه (مثلاً به اسم Max) برای ذخیره بزرگ‌ترین عدد انتخاب می‌کنیم. ابتدا اولین عدد را در آن قرار می‌دهیم. سپس اعداد بعدی را دانه دانه با محتوای این خانه حافظه مقایسه می‌کنیم. هر وقت یک عدد بزرگ‌تر پیدا کردیم، همان را درون خانه حافظه Max ذخیره می‌کنیم. با انجام این مقایسه برای هر ۱۰۰ عدد، نهایتاً بزرگ‌ترین عدد از بین اعداد گرفته شده درون خانه حافظه Max ذخیره شده است که آن را نمایش می‌دهیم.

پس برنامه به این شکل در می‌آید:

```
INPUT A
LET Max = A
FOR X = 1 TO 99
    INPUT A
    IF A > Max THEN Max = A
NEXT X
PRINT Max
```

تمرین‌های برنامه نویسی

۱. برنامه‌ای بنویسید که شعاع ۱۰۰ دایره را گرفته، محیط و مساحت آن‌ها را بنویسد.

۲. برنامه‌ای بنویسید که ۷۰ عدد صحیح اتفاقی بین ۲ و ۲۷ هم امکان تولید داشته باشند)

۳. برنامه‌ای بنویسید که اعداد زیر را بنویسد:

۳, ۶, ۹, ..., ۳۰

۴. برنامه‌ای بنویسید که حاصل کسرهای زیر را بنویسد: (هر کدام در یک خط)

$$1, \frac{1}{2}, \frac{1}{3}, \frac{1}{4}, \dots, \frac{1}{100}$$

۵. برنامه‌ای بنویسید که یک عدد گرفته، بگوید اول است یا مرکب.

۶. برنامه‌ای بنویسید که ۱۰۰ عدد پرورد و در آخر تعداد اعداد بزرگ‌تر از ۵ را بنویسد.

۷. برنامه‌ای بنویسید که خروجی آن به شکل زیر باشد:

$$\begin{array}{ll} 1 & 2 \\ 2 & 4 \\ 3 & 6 \\ 4 & 8 \\ \vdots & \vdots \\ \vdots & \vdots \\ 95 & 190 \end{array}$$

۸. برنامه‌ای بنویسید که N را بگیرد و $N!$ را حساب کرده و نمایش دهد.

راهنمایی:

$$N! = 1 \times 2 \times 3 \times \dots \times N$$

۹. برنامه‌ای بنویسید که حاصل b^a را بدون استفاده از توان و تنها با استفاده از

ضربهای متوالی حساب کند.

$$a^b = \overbrace{a \times a \times \dots \times a}^{\text{بار}} \quad \text{با}$$

۱۰. برنامه‌ای بنویسید که با گرفتن n (که یک عدد فرد بزرگ‌تر از ۷ است)، حاصل جمع دنباله‌های

زیر را حساب کند:

$$1 + \frac{1}{3} + \frac{1}{5} + \frac{1}{7} + \dots + \frac{1}{n}$$

II. در ریاضیات، عددی را که مساوی مجموع همه مقسوم علیه‌هایش (به جز خودش) باشد، عدد کامل می‌گویند. مثلاً $28 = 1 + 2 + 4 + 7 + 14$ عدد کامل است، چون $1 + 2 + 4 + 7 + 14 = 28$. برنامه‌ای بنویسید که یک عدد از ما گرفته، بگوید کامل است یا نه.

III. در ریاضیات ثابت می‌شود که :

$$1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \dots = \frac{\pi}{4}$$

حال شما برنامه‌ای بنویسید که با استفاده از دنباله‌ی فوق و با گرفتن مخرج آخرين کسر مورد نظر، عدد π را حساب کند و نمایش دهد.

IV. در ریاضیات دو عدد را که حاصل جمع مقسوم علیه‌های هر یک برابر دیگری باشد، «همدوست» می‌نامند. مثلاً 220 و 284 همدوست هستند، زیرا :

$$\begin{aligned} 284 &= 1+2+4+8+10+11+20+22+44+55+110 &= 220 \\ 220 &= 142+71+4+2+1 &= 284 \end{aligned}$$

حال شما برنامه‌ای بنویسید که دو عدد گرفته، بگوید هم دوست هستند یا نه.

V. برنامه‌ای بنویسید که تمام اعداد دنباله فیبوناچی را که کمتر از 1000 هستند، نمایش دهد.

$$\text{دنباله‌ی فیبوناچی} = 1, 1, 2, 3, 5, 8, 13, 21, 34, \dots$$

VI. برنامه‌ای بنویسید که دو عدد گرفته، و ب.م. آنها را به روش نرdbانی حساب کرده و نمایش دهد.

تمرین‌های مخصوص دانش آموزان علاقه‌مند:

۱. برنامه‌ای بنویسید که یک عدد گرفته، ارقام آن را برعکس بنویسد. مثلاً اگر عدد ۶۷۴۱ را وارد کنیم، بنویسد:

1

4

7

6

آیا می‌توانید خروجی را به صورت یک عدد بنویسید؟

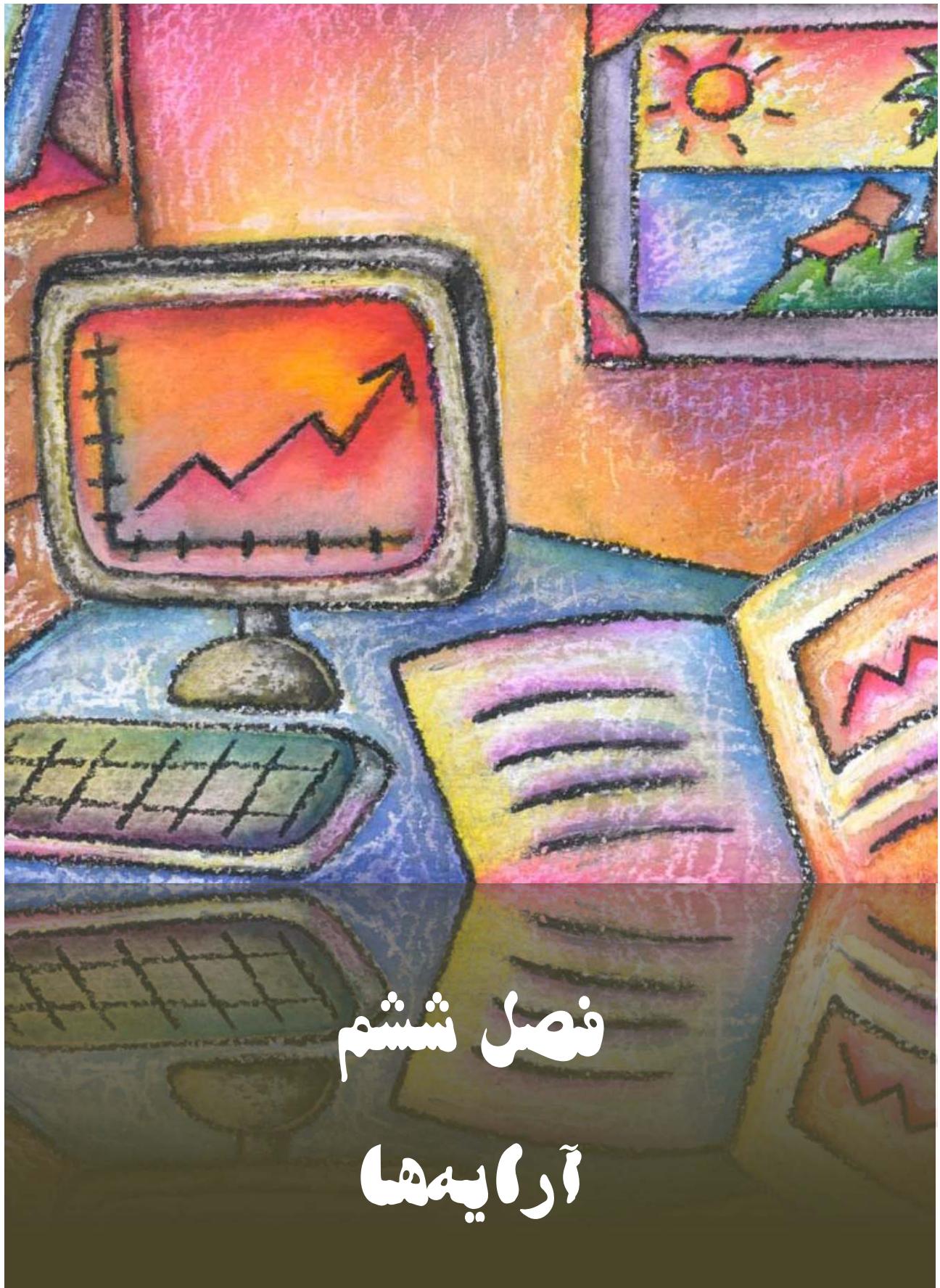
۲. برنامه‌ای بنویسید که اعداد ۱ تا ۱۰۰ را ۵۰ بار بنویسد.

۳. برنامه‌ای بنویسید که تمام اعداد اول بین ۲ تا ۱۰۰۰ را بنویسد.

۴. برنامه‌ای بنویسید که جدول ضرب اعداد ۱ تا ۱۰ را به طور مرتب زیر هم بنویسد.

۵. برنامه‌ای بنویسید که تمام اعداد ۳ رقمی را که می‌توان با ارقام ۱ تا ۵ نوشت بنویسد.





فصل ششم

آرایهها

۶.۱. مقدمه

در این فصل با آرایه‌ها آشنا می‌شویم. آرایه‌ها، خانه‌های حافظه‌ی پلاک دار هستند و در بسیاری موارد، کار ما را آسان می‌کنند. کاربرد آرایه‌ها به قدری وسیع است که نمی‌توان تصور کرد یک زبان برنامه‌نویسی، آرایه نداشته باشد. تمام زبان‌های برنامه‌نویسی سطح بالا، خانه‌های حافظه‌ی پلاک دار یا آرایه را پشتیبانی می‌کنند.

۶.۲. آرایه‌ی یک بعدی

می‌خواهیم برنامه‌ای بنویسیم که ۵ عدد از ما بگیرد، سپس آن‌ها را به ترتیب عکس نمایش دهد. برای نوشتن این برنامه نمی‌توانیم از یک حلقه‌ی FOR پنج تایی استفاده کنیم، چون باید هر ۵ عدد را جایی نگه داریم و بعد از تمام شدن پرسیدن، آن‌ها را نمایش دهیم. تنها چاره‌ای که برایمان می‌ماند، نوشتن برنامه به صورت زیر است:

```
INPUT A, B, C, D, E
PRINT E, D, C, B, A
```

حال فرض کنید می‌خواهیم به جای ۵ عدد، ۱۰۰۰ عدد داشته باشیم. مسلماً روش بالا دیگر به درد نمی‌خورد. خوب‌بختانه زبان بیسیک مشکل ما را حل کرده است. به این صورت که می‌توانیم از حلقه FOR استفاده کنیم و در ضمن تمام اعداد گرفته شده هم در خانه‌های حافظه‌ی مختلف ذخیره شوند. بدین منظور از خانه‌های حافظه‌ی پلاک‌دار استفاده می‌کنیم. تمام مشخصات و ویژگی‌های خانه‌های حافظه‌ی پلاک‌دار مثل خانه‌های حافظه معمولی است، فقط با این تفاوت که این خانه‌ها جلوی اسمشان در داخل پرانتز یک پلاک هم دارند. بنابراین یک خانه حافظه‌ی پلاک‌دار به صورت زیر شناخته می‌شود:

(پلاک) نام خانه حافظه

به تعدادی خانه حافظه‌ی پلاک دار که اسمشان یک چیز است، اما پلاکشان فرق می‌کند، آرایه می‌گوییم.

حال می‌خواهیم بینیم چگونه به رایانه بفهمانیم برایمان یک آرایه درست کند. روش کار خیلی ساده است. ابتدا اسم خانه‌های حافظه‌ی آرایه را انتخاب می‌کنیم. سپس تعداد پلاک‌ها را معلوم می‌کنیم و بعد در ابتدای برنامه خط زیر را می‌نویسیم:

(تعداد پلاک) نام خانه‌ی حافظه DIM

این خط، تعدادی خانه حافظه‌ی پلاک دار با اسم مشابه و پلاک‌های مختلف (از ۱ تا تعدادی که مشخص کردایم) برای ما کنار می‌گذارد و ما در طول برنامه می‌توانیم از هر کدام آن‌ها که خواستیم استفاده کنیم.

: نمونه

DIM A(100)

DIM B2(73)

DIM Ali\$(15)

خط اول یک آرایه ۱۰۰ تایی به نام A درست می‌کند که خانه‌های آن به شکل زیر هستند:

A(1)	A(2)	A(3)	A(4)	...	A(99)	A(100)

در هر کدام از این ۱۰۰ خانه می‌توانیم یک عدد ذخیره کنیم.

خط دوم یک آرایه ۷۳ تایی عددی به نام B2 درست می‌کند که خانه‌های آن به شکل زیر هستند:

B2(1)	B2(2)	B2(3)	B2(4)	...	B2(72)	B2(73)

و بالاخره خط سوم یک آرایه ۱۵ تایی به نام Ali درست می‌کند که به خاطر وجود علامت \$ در اسم

آن، در هر خانه‌اش می‌توان یک کلمه ذخیره کرد.

Ali\$(1)	Ali\$(2)	Ali\$(3)	Ali\$(4)	...	Ali\$(14)	Ali\$(15)

توجه: تمام خانه‌های حافظه یک آرایه، از یک نوع و کاملاً یکسان (مثلاً همه عددی یا همه کلمه‌ای) هستند، اما محتواشان با هم فرق می‌کند.

حال برمی‌گردیم که به مسئله‌ای که با آن کار می‌کردیم. می‌خواهیم ۱۰۰۰ عدد بگیریم و به ترتیب عکس نمایش دهیم. اکنون روش کار مشخص است. یک آرایه ۱۰۰۰ تایی می‌سازیم. یک حلقه‌ی ۱۰۰۰ تایی هم تشکیل می‌دهیم و در هر بار اجرای حلقه، یک خانه از خانه‌های آرایه را پر می‌کنیم. حلقه را روی شماره پلاک خانه‌ها (از ۱ تا ۱۰۰۰) می‌بندیم. برنامه به شکل زیر در می‌آید:

```
DIM A(1000)
FOR I = 1 TO 1000
    INPUT A(I)
NEXT I
FOR I=1000 TO 1 STEP -1
    PRINT A(I)
NEXT I
```

کار حلقه اول این است که I را از ۱ تا ۱۰۰۰ تغییر می‌دهد و تمام خانه‌های حافظه آرایه را پر می‌کند، می‌بینید که به جای شماره پلاک، یک متغیر (I) گذاشته‌ایم. شماره پلاک یک خانه از آرایه یک عدد است و هر عبارت عددی را به جای آن می‌توان قرار داد. کار حلقه دوم هم این است که از آخر به اول محتوای تمام خانه‌ها را نمایش می‌دهد.

دقّت کنید که چون کار حلقه دوم بعد از اتمام کار حلقه اول آغاز می‌شود، از همان متغیر A برای حلقه دوم استفاده کردہ‌ایم. این کار اشکالی ندارد، اما هر موقع احساس کردید ممکن است برنامه‌تان به هم بریزد، ریسک نکنید و از متغیرهای مختلف برای حلقه‌های مختلف استفاده کنید.

 **نمونه:** برنامه‌ی زیر پس از اجرا چه چیزی بر روی صفحه چاپ می‌کند؟

```
DIM A(20)
FOR I = 1 TO 20
    A(I) = I * 2
NEXT I
PRINT A(2), A(3), A(19)
```

پاسخ: برای به دست آوردن خروجی به سراغ حلقه M رویم، A را از ۱ تا ۲۰ تغییر می‌دهیم و هر دفعه ۲ برابر A را در خانه‌های حافظه‌ی A با پلاک A می‌ریزیم، یعنی داریم:

$$\begin{aligned}A(1) &= 1 * 2 \\A(2) &= 2 * 2 \\A(3) &= 3 * 2 \\\vdots \\A(20) &= 20 * 2\end{aligned}$$

پس در نهایت در هر خانه، عددی معادل ۲ برابر پلاکش ذخیره می‌شود، یعنی داریم:

$A(1)$	$A(2)$	$A(3)$	$A(4)$...	$A(19)$	$A(20)$
2	4	6	8		38	40

در خط آخر محتوای سه خانه‌ی $A(2)$ ، $A(3)$ و $A(19)$ نمایش داده می‌شود، پس خروجی به شکل زیر است:

4 6 38

 **توجه:** هرگاه خواستید برنامه‌ای را خط به خط دنبال و خودتان اجرا کنید، یک جدول بکشید و محتوای هر خانه حافظه را در آن وارد کنید. با عوض شدن محتوای هر کدام از خانه‌ها در طول برنامه، در جدول هم آن را عوض کنید. به این جدول، جدول تعقیب می‌گویند و دنبال کردن برنامه را آسان تر می‌کند.

 نمونه : برنامه‌ای بنویسید که ۲۰ عدد را بپرسد. سپس اعداد بزرگ‌تر از میانگین را نمایش دهد.

پاسخ : برنامه را به ۳ قسمت تقسیم می‌کنیم و هر کدام را جداگانه می‌نویسیم:

۱. پرسیدن ۲۰ عدد

۲. محاسبه میانگین

۳. نمایش اعداد بزرگ‌تر از میانگین

برنامه به شکل زیر در می‌آید:

```
DIM A(20)
FOR i = 1 TO 20
    INPUT A(i)           ► قسمت اول
NEXT i
```

```
FOR i = 1 TO 20
    Bank = Bank + A(i)      ► قسمت دوم
NEXT i
Miangin = Bank / 20
```

```
FOR i = 1 TO 20
    IF A(i) > Miangin THEN PRINT A(i)  ► قسمت سوم
NEXT i
```

در قسمت ۲ با پیمایش آرایه، حاصل جمع تمام خانه‌ها را در متغیر Bank ریخته‌ایم و در آخر حاصل جمع را تقسیم بر تعداد کرده‌ایم تا میانگین به دست آید. می‌شد قسمت ۱ و ۲ را ادغام کرد، ولی نوشتن برنامه به این صورت راحت‌تر است.

 توجه: برای نوشتن برنامه‌های بزرگ و طولانی، آن‌ها را به چند برنامه کوچک تقسیم کنید و هر کدام را جداگانه با خیال راحت بنویسید.

 نمونه : برنامه ای بنویسید که ۲۰ عدد بپرسد و در خانه های یک آرایه قرار دهد. سپس محتوای خانه های با پلاک زوج را نمایش دهد.

پاسخ :

باز هم طبق معمول حلقه را روی شماره پلاک ها می بندیم.

```
DIM A(20)
FOR i = 1 TO 20
    INPUT A(i)
NEXT i
FOR I = 2 TO 20 STEP 2
    PRINT A(i)
NEXT i
```

 نمونه : برنامه ای بنویسید که پس از پرسیدن تعداد اعداد، به آن تعداد عدد گرفته، سپس میانگین آن ها را نمایش دهد.

پاسخ :

```
INPUT Tedad
DIM A(Tedad)
FOR i = 1 TO Tedad
    INPUT A(i)
NEXT i
FOR I = 1 TO Tedad
    Majmoo = Majmoo + A(i)
NEXT i
PRINT Majmoo / Tedad
```



۶. ۳. تمرین‌های برنامه نویسی

I. برنامه‌ای بنویسید که ۱۰۰ عدد گرفته، داخل آرایه A قرار دهد. سپس :

(الف) به محتوای هر خانه، ۲۵ واحد اضافه کند.

(ب) به محتوای هر خانه، به اندازه‌ی شماره پلاک آن خانه اضافه کند.

II. برنامه‌ای بنویسید که یک آرایه ۱۰۰ تایی را با اعداد اتفاقی بین ۱ تا ۱۰۰ پر کند. (در هر خانه

یک عدد اتفاقی قرار دهد. اعداد ممکن است تکراری هم باشند)

III. برنامه‌ای بنویسید که ۱۰۰ عدد گرفته، آن‌ها را به ترتیب عکس دریافت در آرایه A قرار دهد.

(یعنی آخرین عدد گرفته شده را در خانه‌ی اول آرایه، نود و نهمین عدد گرفته شده را در خانه‌ی

دوم آرایه و ... بریزد)

IV. جمع دو آرایه عددی با تعداد خانه‌های مساوی، آرایه سومی است با همان تعداد خانه که هر

خانه‌ی آن از جمع دو خانه‌ی هم پلاک در آرایه‌های اولیه به دست می‌آید. برای مثال اگر A و

B دو آرایه‌ی ۳ تایی به شکل زیر باشند، آرایه‌ی C حاصل جمع این دو آرایه است.

$$\begin{array}{|c|c|c|c|} \hline A & 1 & 2 & 3 \\ \hline \end{array} + \begin{array}{|c|c|c|c|} \hline B & -1 & 3 & 4 \\ \hline \end{array} = \begin{array}{|c|c|c|c|} \hline C & 0 & 5 & 7 \\ \hline \end{array}$$

حال شما برنامه‌ای بنویسید که پس از پرسیدن n، اعضای دو آرایه‌ی n تایی A و B را بپرسد و

در نهایت حاصل جمع این دو آرایه را در آرایه C ذخیره کند.

V. برنامه‌ای بنویسید که ۱۰۰ عدد گرفته و داخل آرایه A قرار دهد. سپس ۱۰۰ عدد دیگر گرفته و

داخل آرایه B قرار دهد. سپس آرایه C را به نحوی پر کند که اعضای آرایه A در

خانه‌های پلاک فرد آن و اعضای آرایه B در خانه‌های پلاک زوج آن قرار گیرند.

۶. برنامه‌ای بنویسید که اسامی ۱۰۰ نامزد انتخاباتی را گرفته، داخل آرایه $A\$$ قرار داده، سپس آرای ۱۰۰۰ نفر افراد شرکت کننده در انتخابات را بپرسد و در پایان تعداد آرای هر نامزد را با ذکر نام او بنویسد. (هر نفر به ۸ اسم رأی می‌دهد)

۷. برنامه‌ای بنویسید که اعضای دو مجموعه‌ی عددی A و B را گرفته، اشتراک آن‌ها را بنویسد.
(مجموعه A، ۸ عضوی و مجموعه B، ۱۵ عضوی است)

۸. برنامه‌ای بنویسید که ۱۰۰ اسم گرفته، داخل آرایه $A\$$ بگذارد. سپس یک نام را به طور تصادفی روی صفحه بنویسد.

تمرینهای مخصوص دانشآموزان علاقه‌مند

۹. برنامه‌ای بنویسید که اعضای دو مجموعه‌ی عددی ۸ عضوی A و ۱۵ عضوی B را گرفته، اجتماع آن‌ها را در آرایه‌ی C ذخیره کند و در نهایت اعضای مجموعه‌ی C را نمایش دهد.
(دقّت کنید به هیچ عنوان عضو تکراری چاپ نکنید)

۱۰. فرض کنید دانشآموزان مدرسه را با شماره‌های ۱ تا ۱۰۰ نام گذاری کردایم. برنامه‌ای بنویسید که این دانشآموزان را برای سال بعد در ۴ کلاس ۲۵ نفره، به طور اتفاقی کلاس بندي کند.

راهنمایی : ۴ آرایه‌ی عددی ۲۵ خانه‌ای به نام Class1 ، Class2 ، Class3 و Class4 باشند و هر آرایه را با ۲۵ عدد اتفاقی بین ۱ تا ۱۰۰ پر کنید. دقّت کنید هر عدد فقط در یک آرایه ظاهر شود. (یعنی هر نفر فقط می‌تواند در یک کلاس قرار بگیرد)

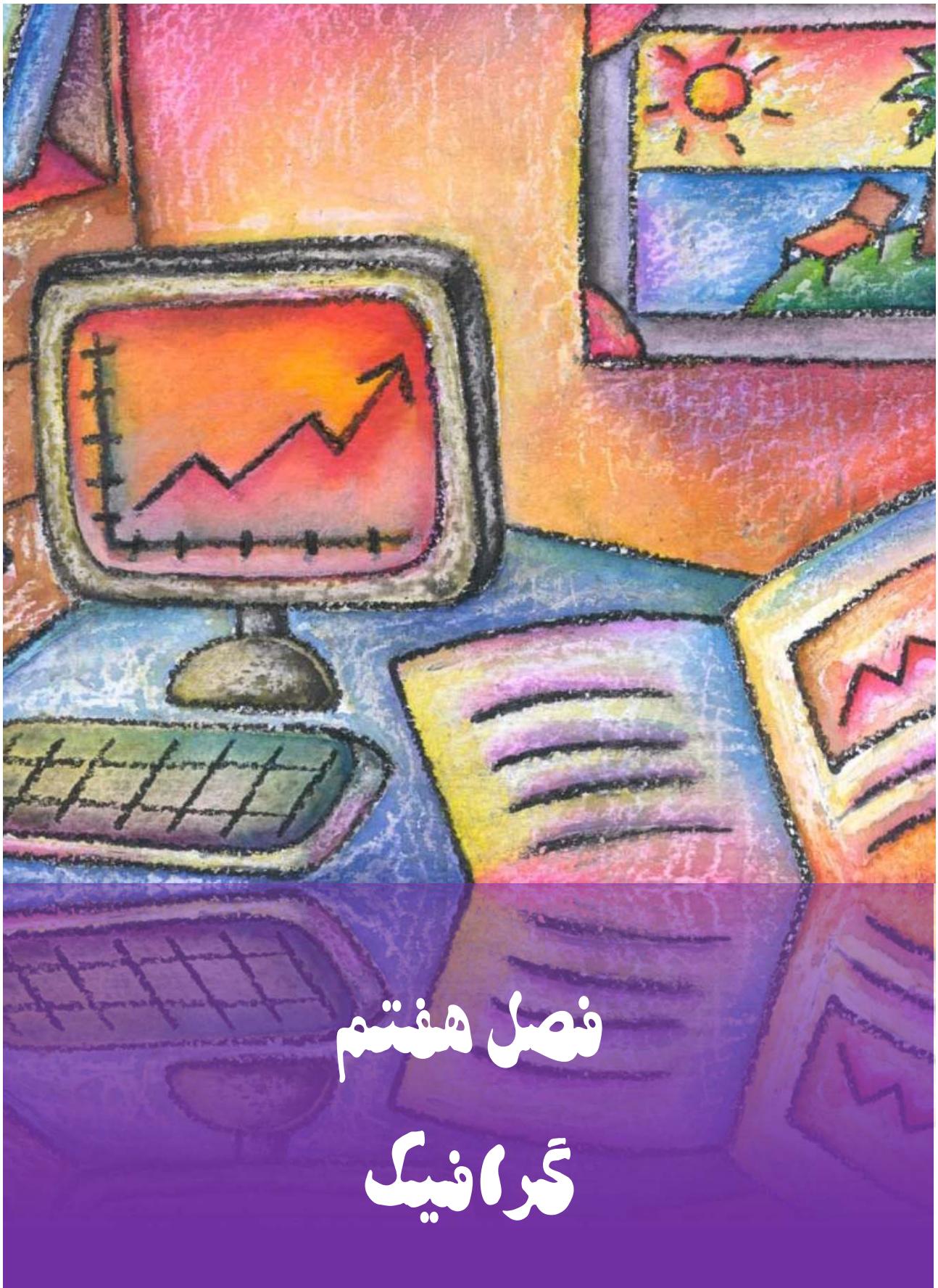
۳۰. در روزگار قدیم زندانیان را به صورت دایره‌وار می‌نشانندند و آن‌ها را یک در میان گردن می‌زدند و این کار را آن قدر ادامه می‌دادند تا فقط یک نفر باقی بماند. سپس آن یک نفر را آزاد می‌کردند. تعیین اولین قربانی همیشه به صورت اتفاقی انجام می‌شد.

حال شما برنامه‌ای بنویسید که :

- (الف) پس از گرفتن n (تعداد زندانی‌ها)، شماره‌ی زندانی خوش شанс را بنویسد.
- (ب) پس از گرفتن n و m (به جای یکی در میان، m تا m گردن می‌زنیم)، شماره‌ی زندانی خوش شанс را بنویسد.

۴۰. برنامه‌ای بنویسید که ۱۰۰ اسم گرفته، سپس ۵ نام را به طور تصادفی بنویسد. (اسم تکراری نباید بنویسد)





فصل هفتم

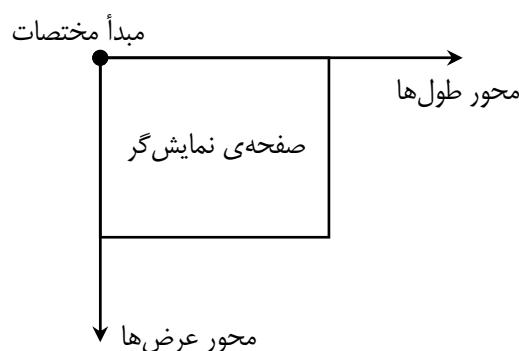
گرافیک

۷.۱. مقدمه

تا این فصل، برنامه‌هایی که نوشته‌ایم، همه با اعداد و ارقام و متون سر و کار داشتند. در این فصل می‌خواهیم با برنامه‌هایی که خروجی آن‌ها تصاویر گرافیکی است، آشنا شویم. در زبان بیسیک دستورات مختلفی برای کشیدن انواع و اقسام شکل‌های هندسی (نقطه، خط، مستطیل، دایره، بیضی) و همچنین رنگ کردن آن‌ها وجود دارد. در این فصل چند تا از مهمترین این دستورات را یاد می‌گیریم.

۷.۲. مختصات گرافیکی

صفحه‌ی نمایش گر رایانه مثل صفحه تلویزیون از تعداد زیادی نقطه تشکیل شده است که با روشن و خاموش کردن (یا بهتر بگوییم، با تنظیم کردن رنگ) آن‌ها می‌توانیم هر متن یا تصویری را روی صفحه نمایش دهیم. به این نقاط پیکسل (Pixel) می‌گوییم. کاری که در این قسمت می‌خواهیم یاد بگیریم این است که چگونه جای نقاط مختلف در صفحه را به رایانه نشان دهیم و بگوییم آن‌ها را روشن یا خاموش کند. برای این کار به سراغ ریاضیات می‌رویم. در ریاضیات خوانده‌اید که برای مشخص کردن جای نقاط هر صفحه کافی است دو محور طول و عرض رسم کنیم و مختصات هر نقطه را با طول و عرض آن مشخص کنیم. در زبان بیسیک هم برای مشخص کردن جای نقاط، از یک دستگاه مختصات دو بعدی استفاده می‌کنیم. شکل این دستگاه در صفحه‌ی نمایش گر به صورت زیر است:



همان‌طور که می‌بینید تنها تفاوت این دستگاه با دستگاه‌های معمول در ریاضیات، این است که جهت مثبت محور عرض‌ها به سمت پایین است. واحد درجه‌بندی محورها هم یک پیکسل است، یعنی در جهت طول‌ها، هر یک پیکسل، یک واحد به طول اضافه می‌کند و هم چنین برای جهت عرض‌ها.

در محیط گرافیکی زبان بیسیک، 640×480 نقطه در طول صفحه و 480 نقطه در عرض صفحه داریم. یعنی کلًا $= 307200$ نقطه درون صفحه داریم. مختصات نقاط در بیسیک نمی‌تواند منفی باشد و مختصات نقطه‌ی مبدأ ($0, 0$) است. برای بیان مختصات هر نقطه در بیسیک از شکل کلی زیر استفاده می‌کنیم:

(عرض نقطه، طول نقطه)

یعنی ابتدا طول نقطه و سپس با قرار دادن یک کاما، عرض نقطه را می‌نویسیم و هر دو را درون یک پرانتز قرار می‌دهیم. با این قرارداد، مختصات تعدادی از نقاط صفحه را روی شکل زیر مشخص کرده‌ایم:



برای فعال کردن محیط گرافیکی بیسیک از دستور SCREEN 12 در ابتدای برنامه استفاده می‌کنیم. یعنی هر گاه خواستیم در برنامه‌ای از دستورات گرافیکی استفاده کنیم. این دستور را در خط ابتدای آن می‌نویسیم.

آیا می‌دانید؟

زبان بیسیک چندین محیط گرافیکی دارد که از لحاظ تعداد نقاط صفحه و تعداد رنگها با هم فرق دارد. در جدول زیر چند تا از پرکاربردترین محیط‌های گرافیکی زبان بیسیک را با شماره‌شان مشخص کرده‌ایم:

شماره محیط	تعداد نقاط	تعداد رنگ‌ها
۱۶	۳۲۰×۲۰۰	۱
۱۶	۶۴۰×۲۰۰	۲
۲۵۶	۶۴۰×۴۸۰	۱۲
۲۵۶	۳۲۰×۲۰۰	۱۳

برای وارد شدن به هر محیط گرافیکی، کافی است شماره‌ی آن را جلوی دستور SCREEN بنویسید. مثلاً هر جای برنامه بنویسید 2 SCREEN وارد محیط گرافیکی شماره ۲ می‌شود.

برای بازگشت به محیط متنی در هر جای برنامه، کافی است دستور 0 SCREEN را بنویسید.

۷.۳. دستور رسم نقطه

دستور رسم نقطه در زبان بیسیک به صورت زیر است:

شماره رنگ نقطه ، (عرض نقطه ، طول نقطه) PSET

این دستور، نقطه‌ی با مختصات مشخص شده را با رنگ نوشته شده مشخص می‌کند. شماره تعدادی از رنگ‌های پر کاربرد به این صورت است:

رنگ	شماره رنگ
سیاه	۰
آبی	۱
سبز	۲
فیروزه‌ای	۳
قرمز	۴
بنفش	۵
قهوه‌ای	۶
سفید	۷
حاکستری	۸
زرد	۱۴

به جای طول و عرض نقطه و رنگ، می‌توان هر عبارت ریاضی را استفاده کرد.

◀ **نمونه** : برنامه زیر نقطه‌ی وسط صفحه را با رنگ آبی و نقطه‌ی با مختصات (100,100) را با رنگ قرمز روشن می‌کند:
پاسخ :

SCREEN 12
PSET (319 , 239) , 1
PSET (100 , 100) , 4

 نمونه : برنامه‌ای بنویسید که با استفاده از دستور PSET، خط افقی زیر را روی صفحه رسم کند:

(100, 100) (200, 100)

پاسخ : یک خط از تعدادی نقطه که به دنبال هم قرار گرفته‌اند، ساخته می‌شود. در شکل بالا طول نقاط خط از ۱۰۰ تا ۲۰۰ تغییر می‌کند، ولی عرضشان ثابت است. پس یک حلقه FOR روی طول نقاط می‌بندیم.

```
SCREEN 12
FOR i = 100 TO 200
    PSET (i, 100), 1
NEXT i
```

۷.۴. دستور رسم خط

واضح است که اگر بخواهیم هر خط را مانند نمونه‌ی قبل با رسم نقاط رسم کنیم، کار سختی است. زبان بیسیک برای کشیدن خط یک دستور مجزا دارد که به شکل زیر با گرفتن مختصات دو نقطه‌ی ابتدایی و انتهایی خط آن را رسم می‌کند.

رنگ خط ، (عرض نقطه انتهای ، طول نقطه انتهای) – (عرض نقطه ابتدای ، طول نقطه ابتدای)

 نمونه : برنامه‌ای بنویسید که خط زیر را به رنگ زرد رسم کند:

(10, 10) (100, 80)

پاسخ :

```
SCREEN 12
LINE (10 , 10) – (100 , 80) , 14
```

 **نمونه** : برنامه‌ای بنویسید که با گرفتن مختصات گوشه بالا سمت چپ و پایین سمت راست یک مستطیل، آن را رسم کند.



پاسخ : مستطیل را به شکل زیر در نظر می‌گیریم. باید x_1 و y_1 و x_2 و y_2 را از کاربر بگیریم. برای رسم این مستطیل باید ۴ خط رسم کنیم. با کمی دقت در شکل می‌توانیم مختصات دو گوشی دیگر را هم به صورت زیر بر حسب مختصات گرفته شده حساب کنیم:



حال برنامه را می‌نویسیم:

SCREEN 12

```
INPUT x1 , y1 , x2 , y2
LINE (x1 , y1) – (x2 , y1) , 1
LINE (x2 , y1) – (x2 , y2) , 1
LINE (x2 , y2) – (x1 , y2) , 1
LINE (x1 , y2) – (x1 , y1) , 1
```

۷. ۵. دستور رسم دایره

شکل این دستور به صورت زیر است:

شماره رنگ ، شعاع ، (عرض مرکز ، طول مرکز) CIRCLE

 نمونه : برنامه‌ای بنویسید که در مرکز صفحه، دایره‌ای با شعاع 100 و رنگ بنفش رسم کند.

پاسخ:

```
SCREEN 12
CIRCLE (319 , 239) , 100 , 5
```

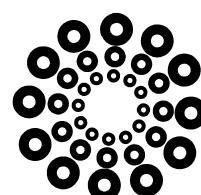
 نمونه : برنامه‌ای بنویسید که دایره نمونه‌ی قبل را به صورت توپر رسم کند.

پاسخ: یک راه برای توپر کردن دایره این است که در داخل آن دایره‌هایی با شعاع‌های کمتر رسم کنیم. در این مثال با همان مرکز وسط صفحه، دایره‌هایی با شعاع 0 تا 100 رسم می‌کنیم تا یک دایره توپر به دست آید (دایره با شعاع صفر، یک نقطه است). حلقه را روی شعاع می‌بندیم.

```
SCREEN 12
FOR i = 0 TO 100
  CIRCLE (319 , 239) , i , 5
NEXT i
```

 آیا می‌دانید؟

با همین دستور CIRCLE می‌توان کمان و بیضی هم کشید. برای این که با قابلیت‌های این دستور آشنا شوید کلمه‌ی CIRCLE را بنویسید. مکان نما را روی آن برد و Shift + F1 را فشار دهید. با این کار صفحه‌ای از راهنمای بیسیک که مربوط به دستور CIRCLE است باز می‌شود. در این صفحه تمام قابلیت‌های دستور CIRCLE با مثال توضیح داده شده‌اند. هم چنین با دستور LINE هم می‌توان مستطیل افقی کشید. آن را هم امتحان کنید.



۷. ۶. دستور رنگ آمیزی

برای رنگ کردن یک شکل رسم شده، می‌توان از دستور زیر استفاده کرد:

شماره رنگ مرز ، شماره رنگ آمیزی ، (عرض نقطه شروع ، طول نقطه شروع) PAINT

کار این دستور این است که از نقطه‌ی مشخص شده با رنگ مورد نظر شروع به رنگ آمیزی می‌کند و آن قدر پیشروی می‌کند تا به نقطه‌ای با رنگ مرز برسد.

 **نمونه** : می‌خواهیم داخل دایره‌ی با رنگ بنفسن دو مثلث قبل را با رنگ سبز رنگ بزنیم. از برنامه زیر می‌توانیم استفاده کنیم:

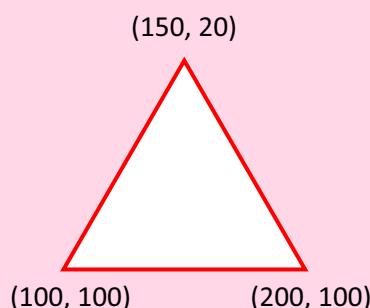
پاسخ:

SCREEN 12
CIRCLE (319 , 239) , 100 , 5
PAINT (319 , 239) , 2 , 5

برای رنگ آمیزی به یک نقطه شروع در داخل دایره احتیاج داشته‌ایم که از همان مرکز دایره استفاده کرده‌ایم.

با این دستور می‌توانیم داخل هر شکل بسته را رنگ آمیزی کنیم. اگر شکل بسته نباشد، رنگ داخل آن به بیرون راه پیدا می‌کند و بیرون شکل هم رنگ می‌شود.

 **نمونه** : برنامه‌ای بنویسید که مثلث زیر را با رنگ قرمز رسم کرده و داخل آن را رنگ سفید بزند.



پاسخ:

SCREEN 13

```
LINE (150 , 20) – (200 , 100) , 4  
LINE (200 , 100) – (100 , 100) , 4  
LINE (100 , 100) – (150 , 20) , 4  
PAINT (150 , 50) , 15 , 4
```

۷. تمرین‌های برنامه‌نویسی

۱. برنامه‌ای بنویسید که هزار نقطه اتفاقی با رنگ آبی در صفحه رسم کند. (طول نقاط بین ۰ و ۶۳۹ و عرض آن‌ها بین ۰ و ۴۷۹ باشد)

۲. برنامه‌ای بنویسید که رنگ نقاط تمرین بالا هم به طور اتفاقی بین ۱ تا ۱۵ انتخاب کند.

۳. برنامه‌ای بنویسید که مستطیل توپر زیر را فقط با دستور LINE رسم کند.
(100, 10)

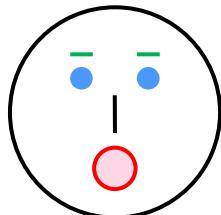


(200, 100)

۴. برنامه‌ای بنویسید که ابتدا یک نقطه با مختصات اتفاقی در صفحه رسم کند. سپس ۱۰۰ نقطه اتفاقی دیگر تولید کرده، همه آن‌ها را با خط به نقطه اول رسم کند.

۵. برنامه‌ای بنویسید که اثر هنری زیر را به طور تقریبی در وسط صفحه رسم کند. (محاسبه مختصات

نقاط مورد نیاز به طور تقریبی با خودتان)

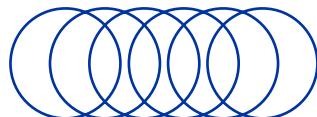


۶. برنامه‌ای بنویسید که ابتدا یک نقطه اتفاقی در صفحه رسم کند. سپس هزار نقطه اتفاقی دیگر تولید

کرده، هر کدام را به آخرین نقطه‌ی رسم شده وصل کند. (شبیه سازی حرکت براونی)

۷. برنامه‌ای بنویسید که با گرفتن مرکز اولین دایره، شعاع دایره‌ها (که همه با هم مساوی است)،

فاصله‌ی مراکز آن‌ها و تعداد آن‌ها، شکل زیر را رسم کند:



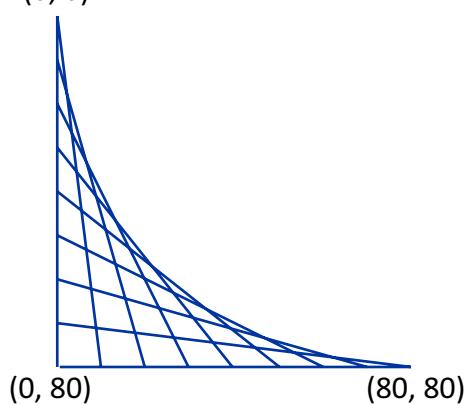
۸. برنامه‌ای بنویسید که با گرفتن مرکز و شعاع دایره‌ی اول و میزان افزایش شعاع آن‌ها در هر مرحله،

شکل زیر را رسم کند:



$(0, 0)$

۹. برنامه‌ای بنویسید که شکل زیر را رسم کند:





فصل هشتم

برنامه نویسی پیشرفته

۸.۱. مقدمه

در فصل‌های قبل با دستورات و ساختارهای مهم زبان بیسیک آشنا شدید. اکنون وقت آن رسیده است که با هم پا به دنیای برنامه‌نویسی پیشرفته بگذاریم.

همان‌طور که یک مهندس عمران برای طراحی و ساخت یک بنای باشکوه، نیاز به دانستن مطالب پایه و کسب مهارت‌های ویژه دارد، یک برنامه‌نویس نیز برای اینکه به دنیای حرفه‌ای‌ها وارد شود، می‌بایست در ابتدا با الگوریتم‌های پایه آشنا شده و در خلال انجام تمرین‌ها و نوشتن برنامه‌های متعدد، بر این مفاهیم پایه و زیربنایی، تسلط لازم را پیدا کند.

۸.۲. الگوریتم‌های پایه

در نوشتن برنامه‌های پیچیده و بزرگ معمولاً از برنامه‌های کوچک‌تر استفاده می‌شود. بخش عمده‌ای از این برنامه‌های کوچک، برنامه‌هایی هستند که کاربرد زیادی دارند و بنابرین هر برنامه‌نویس باید با آن‌ها آشنایی کامل داشته باشد. بخشی از این برنامه‌ها تحت عنوان الگوریتم‌های پایه در این فصل معرفی می‌شوند.

عنوانین مباحثی که در این فصل مطرح می‌شوند به شرح زیر است:

- محاسبه‌ی بیشینه و کمینه
- انتقال دادن و درج کردن
- حذف کردن
- جست‌وجو کردن
- مرتب سازی

۸.۲.۱. محاسبه بیشینه و کمینه (بزرگترین و کوچکترین)

یکی از پایه‌ای ترین الگوریتم‌های یافتن کمینه و بیشینه است. برای نوشتن این الگوریتم، قدم به قدم پیش می‌رویم. در این الگوریتم لازم است تا عددها را تک تک بیینیم و مشخص کنیم که آیا عدد فعلی از تمام اعداد قبلی بزرگ‌تر (کوچک‌تر) هست یا نه. برای این کار از یک متغیر به عنوان نگهدارندهٔ عدد بیشینه (کمینه) تا این مرحله استفاده می‌کنیم.

مثالاً فرض کنید می‌خواهیم بیشینه را بین ۱۰ عدد داده شده به دست آوریم. از عدد اول شروع می‌کنیم. چون تنها یک عدد داریم، پس عدد اول تا این جا بیشینه است و آن را در متغیری که اشاره شد می‌ریزیم. سپس با اعداد بعدی ادامه می‌دهیم. در هر مرحله به شرط اینکه عدد فعلی از عدد درون متغیر بیشتر (کمتر) باشد، آن را در متغیر قرار می‌دهیم. به این ترتیب تا هر مرحله‌ای که پیش برویم عدد درون متغیر، بیشینه (کمینه) اعداد تا آن نقطه خواهد بود. بدین ترتیب توانستیم بیشینه و کمینه یک سری عدد را پیدا کنیم.

تمرین: متن برنامه‌ی محاسبه و چاپ بیشینه در بین ۱۰۰ عدد در یکی از فصل‌های قبل آمده است، سعی کنید قبل از مراجعه به آن، برنامه‌ی محاسبه و چاپ بیشینه و کمینه در بین ۱۰ عدد را نوشته، بر روی رایانه اجرا کنید.

۸.۲.۲. انتقال دادن (Shift) و درج کردن (Insert)

در بسیاری از وقت‌ها، نیاز است که یک مقدار را در یک خانه از آرایه قرار دهیم، بدون آن که محتوای قبلی آن خانه از بین برود. برای این کار باید تمام خانه‌های بعدی را یک خانه به جلو ببریم و مقدار مورد نظر را در خانه‌ی مربوطه قرار دهیم. به این عمل که حتماً در هنگام نوشتن متن در Word یا بیسیک، هنگامی که حرفی را بین دو حرفی که از پیش تایپ شده‌اند قرار می‌دهید، با آن برخورد کرده‌اید، درج کردن یا Insert می‌گویند. درج کردن از دو عمل انتقال (Shift) و مقداردهی خانه‌ی مربوطه تشکیل شده است. دقت کنید که در حین عمل انتقال نباید از مرزهای آرایه بیرون روید و گرنه با یکی از

پیام‌های خطای بیسیک (Subscript Out of Range) روبرو می‌شوید که مفهوم آن این است که شما پای خود را از گلیم خود (مرز آرایه) بیرون گذاشته‌اید.

برای انتقال کافی است محتوای هر خانه از آرایه را به خانه‌ی بعدی انتقال دهیم. قطعه برنامه‌ی زیر عدد C را در خانه‌ی پنجم از آرایه‌ی ۲۰ خانه‌ای A قرار می‌دهد. در این مثال فرض می‌کنیم پیش از این ۱۹ خانه‌ی اول آرایه A پر شده‌اند.

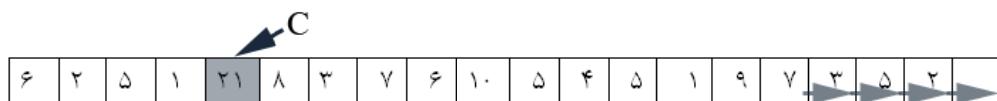
```
FOR I = 19 TO 5 step -1
```

```
    A(I+1) = A(I)
```

```
NEXT I
```

```
A(5) = C
```

به شکل زیر دقت کنید؛ ابتدا محتوای خانه‌ها یک خانه به راست منتقل شده و سپس عدد C در خانه‌ی پنجم آرایه درج می‌گردد.



دقت کنید که عمل انتقال اعضای آرایه به راست باید از آخرین عضو آرایه شروع شود. چرا؟

تمرین: برای تمرین بیشتر، برنامه‌های زیر را بنویسید:

.۱ برنامه‌ای بنویسید که اعضای یک آرایه‌ی عددی ۲۰ عضوی را پرسیده، سپس اعضای آرایه را یک خانه به سمت چپ انتقال دهد و در پایان عدد صفر را در خانه‌ی انتهایی (بیستم) آرایه قرار دهد.

.۲ چرخش یک خانه‌ای آرایه: برنامه‌ای بنویسید که اعضای یک آرایه‌ی ۱۰۰ عضوی را یک خانه به راست ببرد. آخرین عضو آرایه باید به اولین خانه منتقل شود. این عمل بر روی یک آرایه‌ی کاملاً پر صورت می‌گیرد. (برنامه را برای یک آرایه‌ی ۱۰ عضوی اجرا کنید)

۳۴. چرخش K تایی: برنامه‌ای بنویسید که آرایه‌ی سوال قبل را K خانه چرخش دهد. در این حالت K عضو انتهای آرایه به K خانه‌ی ابتدای آرایه منتقل می‌گردد. (برنامه ابتدا مقدار K را از کاربر می‌پرسد)

۳.۲.۸. حذف کردن (Delete)

حتماً تا بهحال کلید delete را در نرمافزار Microsoft Word فشار داده و اثر آن را دیده‌اید. delete کردن به معنای حذف کردن یک خانه و انتقال خانه‌های بعدی به صورتی است که خانه‌ی حذف شده خالی نماند. برای انجام این عمل بر روی یک آرایه، باید عملی عکس درج کردن انجام شود. یعنی ابتدا محتوای خانه‌ی مورد نظر پاک شده و سپس خانه‌های بعدی یک خانه به سمت عقب انتقال یابند. البته واضح است که با انتقال خانه‌ها به عقب، محتوای خانه‌ی مورد نظر، خود به خود پاک می‌شود و عملاً delete کردن، تنها انتقال بخشی از آرایه به عقب است.

تمرین: برنامه‌ای بنویسید که پس از گرفتن اعضای یک آرایه‌ی عددی ۱۰ عضوی، با استفاده از الگوریتم بالا، عضو چهارم از این آرایه را حذف کند.

۴. جستجو کردن (SEARCH)

یکی از پرکاربردترین الگوریتم‌ها، الگوریتم‌های جستجو هستند. هدف این است که مقدار خاصی را در میان تعدادی داده پیدا کنیم. برای مثال فرض کنید می‌خواهیم هنگامی که کاربر نام یک دانش‌آموز را وارد کرد، برنامه به دنبال این نام گشته و پس از پیدا کردن نام او، نمره‌ی آن دانش‌آموز را چاپ کند. این مثال کوچک را در کنار مثال دیگری قرار دهیم؛ سایت سازمان سنجش کشور. می‌دانید که سالانه بیش از یک میلیون نفر در آزمون سراسری دانشگاه‌های کشور شرکت می‌کنند و در زمان اعلام نتایج، این تعداد برای دیدن نتیجه‌ی کنکور خود به سایت اینترنتی سازمان سنجش مراجعه می‌کنند. این بار هم از

الگوریتم‌های جستجو استفاده می‌شود، ولی مطمئناً باید این الگوریتم‌ها از سرعت و دقت بسیار بالاتری برخوردار باشند.

در ادامه به بررسی مهم‌ترین الگوریتم‌های جستجو می‌پردازیم.

برای شروع می‌خواهیم برنامه‌ای بنویسیم که یک اسم را در یک آرایه‌ی ۳۰ عضوی از اسم‌ها که قبلاً پر شده است، پیدا کرده و شماره‌ی خانه‌ای که اسم مورد نظر در آن قرار داشته را چاپ کند. ساده‌ترین روش این است که تمام خانه‌های آرایه را در یک حلقه بررسی کنیم. برنامه‌ی زیر این کار را انجام می‌دهد: (فرض کنید که آرایه قبلاً پر شده است)

DIM Names\$(30)

...

```
INPUT StudentName$  
FOR i = 1 TO 30  
  IF StudentName$ = Names$(i) THEN C = i  
NEXT i  
PRINT "Shomareye Khaneh: "; C
```

در مورد سرعت این الگوریتم چه می‌توانید بگویید؟ سریع‌ترین و کندترین حالت یافتن یک عنصر چیست؟

اگر مقدار مورد جستجو، در اولین خانه‌ی آرایه باشد، جستجوی بقیه‌ی آرایه بیهوده است. در حالی که اگر مقدار مورد نظر در خانه‌ی آخر آرایه باشد، جستجوی تمام آرایه لازم است. اصطلاحاً به این روش جستجو، جستجوی خطی (Linear Search) می‌گویند.

برای افزایش سرعت این الگوریتم، می‌توانیم به جای استفاده از حلقه‌ی FOR، از حلقه‌ای استفاده کنیم که به محض پیدا کردن مقدار مورد نظر، حلقه پایان پیدا کند. با این روش از جستجوی ادامه‌ی آرایه جلوگیری می‌شود و بر سرعت الگوریتم در حالات ذکر شده افزوده می‌شود.

تمرین:

- ۱.** برنامه‌ای بنویسید که اعضای آرایه‌ای ۲۴ تایی شامل نمره‌ی ۲۴ دانش‌آموز را بگیرد. سپس یک نمره گرفته و شماره‌ی خانه‌ی اولین دانش‌آموزی که آن نمره را گرفته، چاپ کند.
- ۲.** فرض کنید اسامی این ۲۴ دانش‌آموز در آرایه‌ی دیگری قرار گرفته است، به گونه‌ای که اسم هر دانش‌آموز در همان شماره خانه‌ای قرار گرفته که نمره‌ی او در آرایه‌ی نمرات در آن شماره خانه ذخیره شده است. برنامه‌ی فوق را به گونه‌ای تکمیل کنید، که پس از گرفتن نمره، نام دانش‌آموز مربوطه را چاپ کند.

روش سریع‌تری برای جستجو وجود دارد. نام این الگوریتم، جستجوی دودویی (Binary Search) است. این جستجو بر روی یک آرایه‌ی از پیش مرتب شده اعمال می‌شود. فرض کنیم اعضای آرایه از کوچک به بزرگ مرتب شده‌اند. روش کار به این صورت است که ابتدا مقدار مورد جستجو را با عضو وسط آرایه مقایسه می‌کیم، اگر از آن بزرگ‌تر بود، از نیمه‌ی اول آرایه صرف‌نظر می‌کنیم و اگر کوچک‌تر بود، از نیمه‌ی دوم آن صرف‌نظر می‌کنیم. در مرحله‌ی بعد همین عمل را با نیمه‌ی باقی‌مانده‌ی آرایه انجام می‌دهیم و به همین صورت ادامه می‌دهیم تا مقدار مورد جستجو با عضو وسط قسمت باقی‌مانده برابر شود. جستجو در اینجا به پایان می‌رسد.

برنامه‌ی این الگوریتم در زیر آمده است: (فرض کنید اعداد به ترتیب در آرایه قرار گرفته‌اند).

```
DIM A(100)
...
INPUT C
M = 1
N = 100
10 L = INT((N+M)/2)
IF A(L) = C THEN PRINT L : END
IF C > A(L) THEN M = L ELSE N = L
GOTO 10
```



۸.۲.۵. مرتب سازی (SORT)

یک دسته‌ی مهم از الگوریتم‌های پایه، الگوریتم‌های مرتب سازی هستند. هدف این دسته از الگوریتم‌ها، این است که مقادیر یک آرایه را از کوچک به بزرگ یا از بزرگ به کوچک مرتب کنند. برای این کار هم روش‌های مختلفی وجود دارد که به مهم‌ترین آن‌ها اشاره می‌کنیم.

الف) مرتب سازی انتخابی (Selection Sort)

یکی از معروف‌ترین روش‌های مرتب‌سازی، مرتب‌سازی انتخابی است. در این روش ابتدا با استفاده از الگوریتم یافتن بیشینه، بزرگ‌ترین عضو آرایه را پیدا می‌کنیم و سپس جای آن عضو را با عضو اول آرایه عوض می‌کنیم. سپس با صرف نظر کردن از عضو اول، دومین عدد بزرگ را در آرایه پیدا می‌کنیم و جای آن را با خانه‌ی دوم عوض می‌کنیم. این کار را تا انتهای آرایه انجام می‌دهیم. پس از انجام این عمل بر روی تک‌تک اعضای آرایه، آرایه مرتب شده خواهد بود.

تمرین: برنامه‌ای بنویسید که با استفاده از روش مرتب سازی انتخابی، نمره‌های کلاس رایانه را از کم‌ترین به بیشترین مرتب کند.

ب) مرتب سازی حبابی (Bubble Sort)

روش دیگر که یکی از محبوب‌ترین روش‌های مرتب‌سازی است، مرتب‌سازی حبابی است. الگوریتم کار به این صورت است که ابتدا خانه‌های اول و دوم آرایه با هم مقایسه می‌شوند. اگر ترتیب‌شان مناسب بود تغییری نمی‌کنند، ولی اگر ترتیب‌شان نادرست بود، جای مقادیر آن‌ها باهم عوض می‌شود. سپس همین عمل روی خانه‌های دوم و سوم صورت می‌گیرد تا آخر آرایه. به این ترتیب پس از یک بار حرکت تا انتهای آرایه و مقایسه‌ی دو به دوی اعضای آرایه، آخرین خانه‌ی آرایه مرتب می‌شود، بار دیگر همین عمل از خانه‌ی اول شروع شده و تکرار می‌شود تا خانه‌ی یکی مانده به آخر مرتب شود و الی آخر. سادگی و سرعت این روش از روش مرتب سازی انتخابی بیشتر است.

تمرین: برنامه‌ای بنویسید که با استفاده از روش مرتب سازی حبابی، نام اجناس یک بقالی را از گران‌ترین به ارزان‌ترین مرتب کند (قیمت‌های اجناس و نام اجناس در دو آرایه از کاربر پرسیده می‌شوند)

تمرین امتیازی : برنامه‌ی بالا را طوری تغییر دهید که اگر در ابتدا و یا در اواسط کار الگوریتم، آرایه مرتب شد، رایانه متوجه شده و از حلقه‌ی مرتب سازی خارج گردد.

۸.۳. پردازش رشته

رشته‌ها یکی از مهم‌ترین عناصر مربوط به هر زبان برنامه‌نویسی هستند. به علت پیوند رشته‌ها با اطلاعات مهم روزمره مانند انواع متن‌هایی که هر روزه با آن‌ها مواجه هستیم، پردازش آن‌ها از اهمیت زیادی برخوردار است. بیسیک نیز دارای دستوراتی برای ساختن و یا تغییر دادن رشته‌ها است که در ادامه به آن‌ها خواهیم پرداخت.

۸.۳.۱. الحاق دو رشته (Concatenation)

یکی از قابلیت‌های زبان بیسیک استفاده از عمل‌گر جمع (+) به منظور ترکیب رشته‌ها با هم‌دیگر است. مجموع دو رشته در بیسیک به معنای الحاق دو رشته به هم‌دیگر می‌باشد. برای مثال اگر در خانه‌ی حافظه‌ی A\$ مقدار "salam" و در خانه‌ی B\$ مقدار "sosis!" ذخیره شده باشد، حاصل جمع این دو رشته یعنی A\$ + B\$ برابر رشتة "salamsosis!" خواهد بود. توجه کنید که هیچ نماد دیگری بین دو رشته اضافه نخواهد شد.

توجه: سایر عمل‌گرهای محاسباتی در دنیای رشته‌های حرفی جایی ندارند.

۸.۳.۲. مقایسه

همانند حالت عددی، رشته‌ها نیز می‌توانند با هم دیگر مقایسه شوند. در هنگام مقایسه‌ی دو رشته ممکن است دو حالت اتفاق بیفتد:

- **تساوی**: دو رشته تنها در حالتی با هم برابر هستند که تمامی حروفشان یکسان باشد. دقت کنید که حروف کوچک و بزرگ زبان انگلیسی برابر نیستند. برای مثال رشته‌های "ali" ، "ali" که حروف کوچک و بزرگ زبان انگلیسی برابر نیستند. برای مثال رشته‌های "Ali" ، "aLi" و "ALI" از نظر بیسیک مقادیر مختلفی دارند.
- **کوچکتری / بزرگتری**: بیسیک قابلیت انجام مقایسه بین دو رشته بر حسب حروف الفبا را دارد. برای نمونه رشته‌ی "ali" از رشته‌ی "bahman" مقدار کمتری داشته و رشته‌ی "zahra" مقدار بزرگتری نسبت به هر دوی آن‌ها دارد. بیسیک برای مقایسه‌ی بین دو رشته ابتدا اولین حرف هر دو را با هم مقایسه می‌کند. توجه داشته باشید که از نظر بیسیک حروف کوچک مقدار بزرگتری نسبت به حروف بزرگ دارند. (مثلاً رشته‌ی "BZ" از رشته‌ی "Ba" کوچک‌تر است) در صورت برابر بودن دو حرف از دو رشته، بیسیک به حروف بعدی مراجعه کرده و مقایسه را بین آن‌ها انجام می‌دهد. اگر دو رشته دارای حروف یکسانی باشند ولی یکی از رشته‌ها از دیگری طولانی‌تر باشد، رشته‌ی کوتاه‌تر مقدار کمتری نسبت به رشته‌ی بزرگ‌تر خواهد داشت.

 **تمرین:** برنامه‌ای بنویسید که با گرفتن نام دانش‌آموزان یک کلاس، فهرست اسامی مرتب‌شده‌ی آن‌ها را چاپ کند.

۸.۳.۳. طول رشته

به کمک دستور LEN می‌توان تعداد حروف داخل یک رشته را به دست آورد. برای مثال دستور PRINT LEN("Salam!") عدد 6 را بر روی صفحه به نمایش در خواهد آورد.

۴.۳.۸. جداسازی از چپ و راست

به کمک دو دستور LEFT\$ و RIGHT\$ می‌توان بخشی از سمت چپ و یا سمت راست یک رشته را جدا کرد. ورودی‌های این دو دستور رشته‌ی مورد نظر و تعداد حروف مورد نیاز برای جدا کردن می‌باشد.

 نمونه : خروجی برنامه‌ی زیر را بنویسید.

```
PRINT LEFT$("Internet", 5)
D$ = RIGHT$("Download", 4)
PRINT D$
```

پاسخ: خط اول از سمت چپ رشته‌ی "Internet" پنج حرف جدا کرده و حاصل آن یعنی "Inter" را بر روی صفحه چاپ می‌کند. خط دوم نیز از سمت راست رشته‌ی "Download" چهار حرف یعنی "load" را جدا و آن را در خط سوم چاپ می‌کند.

 **تمرین:** برنامه‌ای بنویسید که به کمک دستورهای LEFT\$ و RIGHT\$، یک رشته از کاربر گرفته و از داخل آن چند حرف جدا کند. از کاربر علاوه بر رشته مورد نظر، مکان شروع و طول جداسازی را نیز سوال کنید.

۴.۳.۵. زیررشته

بیسیک دارای دستوری به نام MID\$ است که به کمک آن می‌توان یک زیررشته را از داخل رشته‌ی دیگری جدا کرده و یا آن را تغییر داد. شکل این دستور به صورت زیر است:

(طول جداسازی ، شروع جداسازی ، رشته‌ی مورد نظر) MID\$

به کمک این دستور می‌توان زیررشته‌ای از رشته‌ی مورد نظر را که از محل مورد نظر و با طول مشخص جدا شده به دست آورده و یا جایگزین کرد. اگر طول جداسازی در این دستور مشخص نشود، بیسیک جداسازی را تا انتهای جمله انجام خواهد داد.



نمونه: خروجی برنامه‌ی زیر را بنویسید.

```
name$ = "Mohammad Hasan Akbari"
PRINT MID$(name$, 10, 5)
MID$(name$, 10, 5) = "Majid"
PRINT name$
```

پاسخ: دستور دوم، زیر رشته‌ی شروع شده از حرف دهم و به طول پنج حرف از خانه‌ی name\$ را که برابر "Hasan" است چاپ می‌کند. دستور بعدی همین زیر رشته را به مقدار "Majid" تغییر می‌دهد. بنابراین حاصل چاپ خانه‌ی name\$ مقدار "Mohammad Majid Akbari" خواهد بود.



توجه: با استفاده از دستور MID\$ نمی‌توان طول رشته را تغییر داد. اگر رشته‌ای را که می‌خواهیم در رشته‌ی اصلی جایگزین کنیم، طول بیشتری از زیررشته مورد نظر داشته باشد، جایگزینی تنها با طول زیر رشته‌ی جدا شده با MID\$ انجام می‌پذیرد. برای گنجاندن رشته‌ی جدید و یا حذف یک زیر رشته بهتر است رشته‌ی اصلی را به قطعات کوچکتر شکست و آن را مجدداً ترکیب کرد. مانند:

```
A$ = "This is a simple string."
A$ = LEFT$(A$, 10) + "good" + MID$(A$, 17)
PRINT A$
```

خروجی برنامه‌ی بالا "This is a good string." خواهد بود.



تمرین: برنامه‌ای بنویسید که دو رشته از کاربر گرفته و مکان رشته‌ی اول را درون رشته‌ی دوم پیدا کند. برای این کار بهتر است در یک حلقه، تمامی زیررشته‌هایی از رشته‌ی اصلی که دارای طولی یکسان با رشته‌ی اول دارند را با رشته‌ی اول مقایسه کنید.



۶.۳. جستجو

به کمک دستور `INSTR$` می‌توان مکان یک زیرشته را در یک رشته جستجو کرد. شکل کلی این دستور به صورت زیر است:

`(رشته‌ای که می‌خواهیم پیدا کنیم , رشته اصلی , مکان شروع جستجو)`

این دستور موقعیت زیرشته‌ی مورد نظر را در رشته‌ی اصلی برمی‌گرداند. در صورت پیدا نشدن زیرشته مقدار صفر برگردانده خواهد شد. اگر مکان شروع برای این دستور ذکر نشود، بیسیک از ابتدای رشته‌ی اصلی شروع به جستجو خواهد کرد.

۷. تبدیلات

به کمک دستور `STR$` می‌توان معادل رشته‌ای یک عدد را به دست آورد. دستور `VAL` بر عکس این عمل را انجام داده و مقدار عددی یک رشته را (در صورت امکان) برمی‌گرداند. توجه داشته باشید که اگر هنگام کار با دستور `VAL` رشته‌ی شما شامل حروف غیر عددی باشد (همچنین عملگرهای ریاضی) مقدار صفر برگردانده می‌شود. این دستور توانایی انجام عملیات ریاضی را ندارد!

دو دستور `UCASE$` و `LCASE$` نیز به ترتیب تمامی حروف بزرگ یک رشته را به حروف کوچک و بر عکس تبدیل می‌کنند. این کار در هنگام مقایسه رشته‌ها مهم خواهد بود چون همان‌طور که قبلاً ذکر شد، حروف کوچک و بزرگ برای بیسیک مقادیر متفاوتی دارند، اما ممکن است برای ما این مسئله مهم نباشد.

۸. تولید رشته

به کمک تابع `STRING$` می‌توان یک رشته متشكل از تکرار یک حرف با طول مشخص تولید کرد. شکل استفاده از این دستور به صورت زیر است.

`(تعداد , طول)`

دستور `SPACE$` نیز رشته‌ای به طول مشخص از جای خالی (" ") تولید می‌کند.

۸.۳.۹. جدول اسکی (ASCII)

در اصل رشته‌ها مجموعه‌ای از نمادها هستند. این نمادها شامل حروف کوچک و بزرگ، اعداد، عملگرهای ریاضی، نقطه، کاما و ... می‌باشند. تمامی این نمادها در جدولی به نام آسکی فهرست شده‌اند. در این جدول به هر نماد یا به اصطلاح دقیق‌تر به هر کاراکتر یک عدد (کد اسکی) نسبت داده شده است. برای نمونه کد اسکی حرف A در این جدول برابر ۶۵ است. جدول اسکی دارای ۲۵۶ کاراکتر مختلف بوده (چرا ۲۵۶ تا؟) که شامل تقریباً تمامی نمادهای مربوط به نوشتار می‌باشد. به کمک تابع ASC می‌توان کد اسکی یک کاراکتر را به دست آورد. همچنین دستور CHR\$ برای به دست آوردن کاراکتر مربوط به یک کد اسکی خاص مورد استفاده قرار می‌گیرد.

در انتهای کتاب جدول کاراکترهای اسکی پیوست شده است. در این جدول ستون Char دارای شکل کاراکتر، ستون Dec دارای کد اسکی کاراکتر در مبنای ده و ستون Hex دارای کد اسکی کاراکتر در مبنای شانزده است.

۸.۴. آرایه‌های دوبعدی

ما در زندگی روزمره از جدول‌های زیادی استفاده می‌کنیم. برای نمونه جدول رده‌بندی تیم‌ها در لیگ برتر شامل تعدادی ستون برای نگهداری تعداد بازی‌های انجام شده‌ی هر تیم، تعداد بردها، باختها، مساوی‌ها و تعداد گل‌های خورده و زده شده و امتیاز تیم‌هاست. به طول مشابه در بیسیک هم می‌توان آرایه‌ای از آرایه‌ها تعریف کرد. (مانند مثال فوق که جدول شامل تعدادی ستون برای تعدادی تیم است). برای تعریف چنین ابرآرایه‌ای که آن را آرایه‌ی دوبعدی می‌نامیم، کافیست به شکل زیر عمل کنیم.

(تعداد ستون ، تعداد سطر) DIM A

این دستور جدولی با تعداد سطر و ستون مشخص شده برای ما فراهم می‌کند. به طور مشابه برای حالت یکبعدی، می‌توان به خانه‌های یک آرایه‌ی دوبعدی نمونه‌ی تعریف شده با دستور DIM X(3,5) به صورت زیر دسترسی داشت:

X(1,1)	X(1,2)	X(1,3)	X(1,4)	X(1,5)
X(2,1)	X(2,2)	X(2,3)	X(2,4)	X(2,5)
X(3,1)	X(3,2)	X(3,3)	X(3,4)	X(3,5)

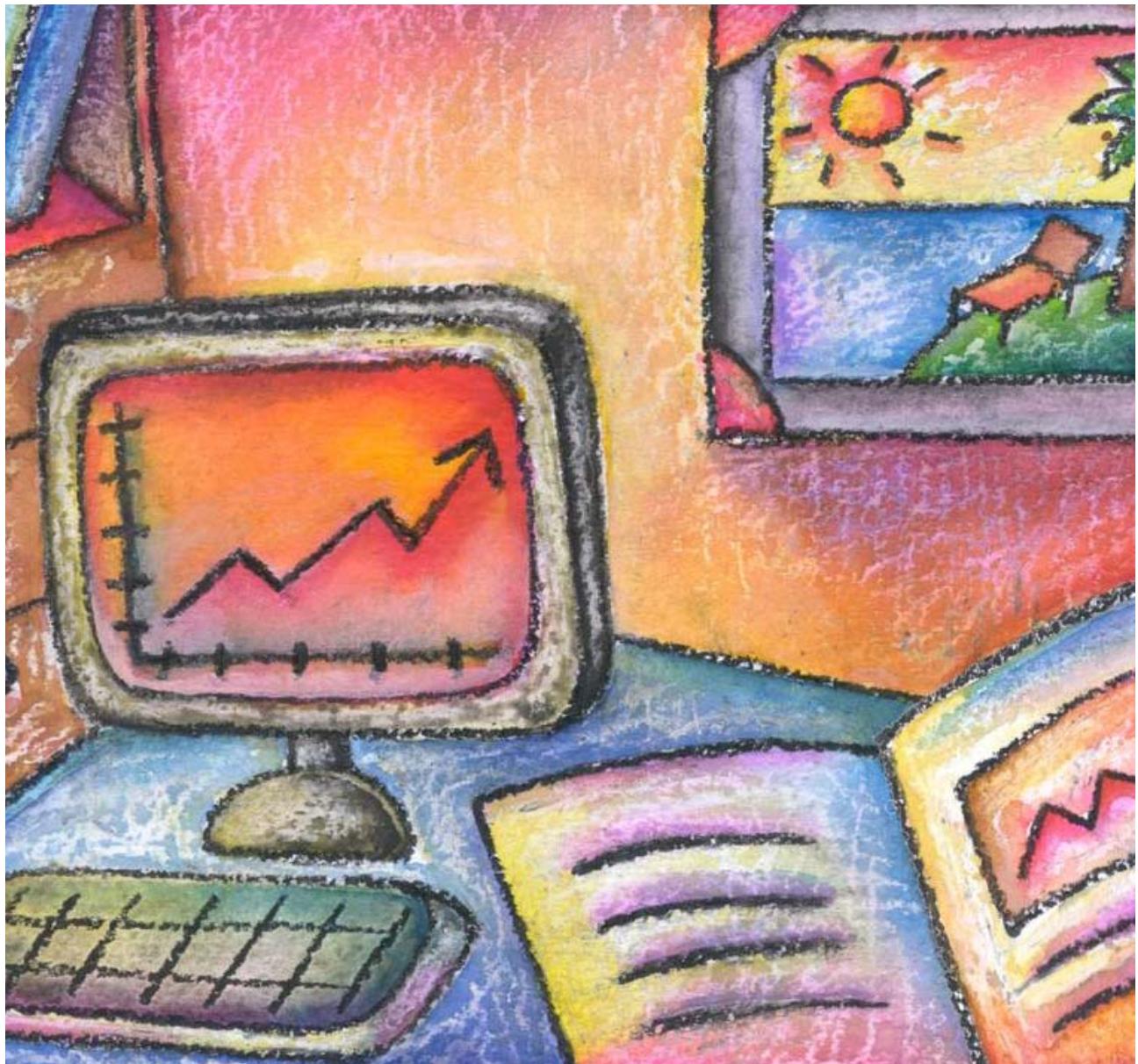
توجه داشته باشید که در زبان بیسیک انتخاب اولین عدد به عنوان سطر دلخواه است. همچنین برای خواندن و یا نوشتן کلیه‌ی عناصر یک آرایه‌ی دو بعدی به دو حلقه‌ی تودرتو احتیاج داریم.

در بیسیک امکان تعریف آرایه‌هایی با ابعاد بالاتر نیز وجود دارد. (برای نمونه برای ثبت درجه حرارت تمام نقاط درون یک استخر در یک آرایه‌ی سه بعدی برای ذخیره کردن دمای نقاط دارای طول، عرض و عمق مشخص)

تمرین: برنامه‌ای بنویسید که پس از گرفتن $N \times N$ اعضای یک مربع $N \times N$ را گرفته و سپس بگوید که آیا این مربع ورقی (جادویی) است یا خیر. مربع ورقی به مربعی گفته می‌شود که در آن حاصل جمع تمامی اعداد واقع در هر سطر، هر ستون، قطر اصلی و قطر فرعی برابر است. نمونه‌ای از یک مربع ورقی نمونه‌ای از یک مربع ورقی 3×3 به شکل زیر است:

۸	۱۳	۶
۷	۹	۱۱
۱۲	۵	۱۰





پیوست ۱

فهرست پژوهشها

فهرست پروژه‌ها برای دانش آموزان علاقه‌مند

این فهرست شامل پروژه‌هایی است که انجام آن‌ها به شدت برای دانش آموزانی که خواهان یادگیری بیشتر برنامه‌نویسی هستند توصیه می‌گردد.

هشت وزیر

برنامه‌ای بنویسید که هشت وزیر را در صفحه‌ی شطرنج طوری قرار دهد که هیچ‌کدام از آن‌ها هم دیگر را تهدید نکنند.

عملیات ریاضی بر روی اعداد بزرگ

برنامه‌ای بنویسید که بر روی دو عدد صد رقمی اعمال جمع، تفریق و ضرب را انجام دهد.

فاکتوریل بزرگ

برنامه‌ای بنویسید که $100!$ را حساب کند.

بازی X-O

برای دو بازیکن بازی X-O را شبیه‌سازی کنید. آیا می‌توانید بازی خود را طوری هوشمند کنید که کاربر با کامپیوتر بازی کند؟


جدول میوه‌ها

برنامه‌ای بنویسید که ابتدا شش کلمه شش حرفی را از کاربر پرسیده و در جدولی 6×6 ذخیره کند. سپس تعدادی کلمه از کاربر پرسیده و این کلمات را به صورت افقی و یا عمودی در جدول پیدا کرده و آن‌ها را خط بزنند. (اگر کلمه در جدول یافت شد جهت آن را با یکی از نمادهای \leftarrow , \rightarrow , \uparrow و \downarrow نمایش داده و اگر کلمه پیدا نشد علامت تعجب (!) چاپ کند) سپس در پایان با وارد کردن کلمه‌ی END از سوی کاربر، رمز جدول که شامل حروف باقی‌مانده است را چاپ کند. برای مثال در جدول زیر:

A	O	L	I	V	E
P	R	I	C	O	T
R	F	I	G	T	A
A	P	P	L	E	D
E	C	N	I	U	Q
P	E	A	C	H	S

با وارد کردن کلمات زیر:

PEACH (هلو)	→
PEAR (گلابی)	↑
APPLE (سیب)	→
DATE (خرما)	↑
OLIVE (زیتون)	→
FIG (انجیر)	←
QUINCE (به)	→
MELON (خربزه)	!
END	

رمز عبور برابر APRICOTS (زردآلوها) خواهد بود.

سعی کنید این پروژه را به صورت گرافیکی در آورید و همچنین راستاهای مورب را به آن اضافه کنید.

--	--

حدس گلدباخ

(الف) برنامه‌ای بنویسید که یک عدد زوج گرفته و دو عدد اول را که مجموع آن‌ها برابر آن عدد زوج باشد پیدا کند.

(ب) برنامه‌ای بنویسید که یک عدد فرد گرفته و سه عدد اول را که مجموع آن‌ها برابر آن عدد فرد باشد پیدا کند.

تجزیه به عوامل اول

برنامه‌ای بنویسید که یک عدد گرفته و آن را به عوامل اول آن تجزیه کند و آن‌ها رو بنویسد.

زیرمجموعه‌ها

برنامه‌ای بنویسید که عدد N را گرفته و تمام زیرمجموعه‌های مجموعه‌ی $\{1, 2, 3, \dots, N\}$ را بنویسد. مثلاً اگر N برابر ۲ داده شود، خروجی برنامه باید به صورت زیر باشد:

{ }
{1}
{2}
{1, 2}

راهنمایی: می‌توان ثابت کرد که تعداد زیرمجموعه‌های یک مجموعه‌ی N عضوی برابر 2^N است.

نمودار ستونی

برنامه‌ای بنویسید که نمودار ستونی نمرات درس رایانه‌ی دانش‌آموزان را رسم کند.

تبديل مبنای

برنامه‌ای بنویسید که یک عدد را از یک مبنای دلخواه به مبنای دیگری ببرد.

رسم اشکال منتظم

برنامه‌ای بنویسید که با گرفتن N یک N ضلعی منتظم رسم کند.



پیوست ۲

جدول اسکی

Ctrl	Dec	Hex	Char	Code	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
^@	0	00		NUL	32	20	!	64	40	@	96	60	'
^A	1	01		SOH	33	21	"	65	41	A	97	61	a
^B	2	02		STX	34	22	#	66	42	B	98	62	b
^C	3	03		ETX	35	23	\$	67	43	C	99	63	c
^D	4	04		EOT	36	24	%	68	44	D	100	64	d
^E	5	05		ENQ	37	25	&	69	45	E	101	65	e
^F	6	06		ACK	38	26	(70	46	F	102	66	f
^G	7	07		BEL	39	27)	71	47	G	103	67	g
^H	8	08		BS	40	28	,	72	48	H	104	68	h
^I	9	09		HT	41	29	*	73	49	I	105	69	i
^J	10	0A		LF	42	2A	+	74	4A	J	106	6A	j
^K	11	0B		VT	43	2B	-	75	4B	K	107	6B	k
^L	12	0C		FF	44	2C	.	76	4C	L	108	6C	l
^M	13	0D		CR	45	2D	/	77	4D	M	109	6D	m
^N	14	0E		SO	46	2E	0	78	4E	N	110	6E	n
^O	15	0F		SI	47	2F	1	79	4F	O	111	6F	o
^P	16	10		DLE	48	30	2	80	50	P	112	70	p
^Q	17	11		DC1	49	31	3	81	51	Q	113	71	q
^R	18	12		DC2	50	32	4	82	52	R	114	72	r
^S	19	13		DC3	51	33	5	83	53	S	115	73	s
^T	20	14		DC4	52	34	6	84	54	T	116	74	t
^U	21	15		NAK	53	35	7	85	55	U	117	75	u
^V	22	16		SYN	54	36	8	86	56	V	118	76	v
^W	23	17		ETB	55	37	9	87	57	W	119	77	w
^X	24	18		CAN	56	38	:	88	58	X	120	78	x
^Y	25	19		EM	57	39	;	89	59	Y	121	79	y
^Z	26	1A		SUB	58	3A	<	90	5A	Z	122	7A	z
^[_	27	1B		ESC	59	3B	=	91	5B	[123	7B	{
^`]	28	1C		FS	60	3C	>	92	5C	\	124	7C	
^_]	29	1D		GS	61	3D	?	93	5D]	125	7D	}
^_	30	1E	▲	RS	62	3E		94	5E	~	126	7E	*
^_	31	1F	▼	US	63	3F		95	5F	Ø	127	7F	

* ASCII code 127 has the code DEL. Under MS-DOS, this code has the same effect as ASCII 8 (BS). The DEL code can be generated by the CTRL + BKSP key.

Dec	Hex	Char									
128	80	Ç	160	A0	đ	192	C0	Ł	224	E0	ꝑ
129	81	Ü	161	A1	í	193	C1	Ł	225	E1	ꝑ
130	82	é	162	A2	ó	194	C2	ꝑ	226	E2	ꝑ
131	83	â	163	A3	ú	195	C3	ꝑ	227	E3	ꝑ
132	84	ã	164	A4	ñ	196	C4	ꝑ	228	E4	ꝑ
133	85	à	165	A5	ń	197	C5	ꝑ	229	E5	ꝑ
134	86	ä	166	A6	ő	198	C6	ꝑ	230	E6	ꝑ
135	87	ç	167	A7	ő	199	C7	ꝑ	231	E7	ꝑ
136	88	ê	168	A8	ڦ	200	C8	ڦ	232	E8	ڦ
137	89	ë	169	A9	ڻ	201	C9	ڦ	233	E9	ڦ
138	8A	î	170	AA	ڻ	202	CA	ڦ	234	EA	ڦ
139	8B	î	171	AB	ڻ	203	CB	ڦ	235	EB	ڦ
140	8C	î	172	AC	ڻ	204	CC	ڦ	236	EC	ڦ
141	8D	î	173	AD	ڦ	205	CD	ڦ	237	ED	ڦ
142	8E	ää	174	AE	«	206	CE	ڦ	238	EE	ڦ
143	8F	å	175	AF	»	207	CF	ڦ	239	EF	ڦ
144	90	É	176	B0	ڦ	208	D0	ڦ	240	F0	ڦ
145	91	æ	177	B1	ڦ	209	D1	ڦ	241	F1	ڦ
146	92	œ	178	B2	ڦ	210	D2	ڦ	242	F2	ڦ
147	93	ô	179	B3	ڦ	211	D3	ڦ	243	F3	ڦ
148	94	ö	180	B4	ڦ	212	D4	ڦ	244	F4	ڦ
149	95	ö	181	B5	ڦ	213	D5	ڦ	245	F5	ڦ
150	96	û	182	B6	ڦ	214	D6	ڦ	246	F6	ڦ
151	97	ü	183	B7	ڦ	215	D7	ڦ	247	F7	ڦ
152	98	ÿ	184	B8	ڦ	216	D8	ڦ	248	F8	ڦ
153	99	ö	185	B9	ڦ	217	D9	ڦ	249	F9	ڦ
154	9A	ü	186	BA	ڦ	218	DA	ڦ	250	FA	ڦ
155	9B	¢	187	BB	ڦ	219	DB	ڦ	251	FB	ڦ
156	9C	£	188	BC	ڦ	220	DC	ڦ	252	FC	ڦ
157	9D	¥	189	BD	ڦ	221	DD	ڦ	253	FD	ڦ
158	9E	₩	190	BE	ڦ	222	DE	ڦ	254	FE	ڦ
159	9F	f	191	BF	ڦ	223	DF	ڦ	255	FF	ڦ

یادداشت

1-1