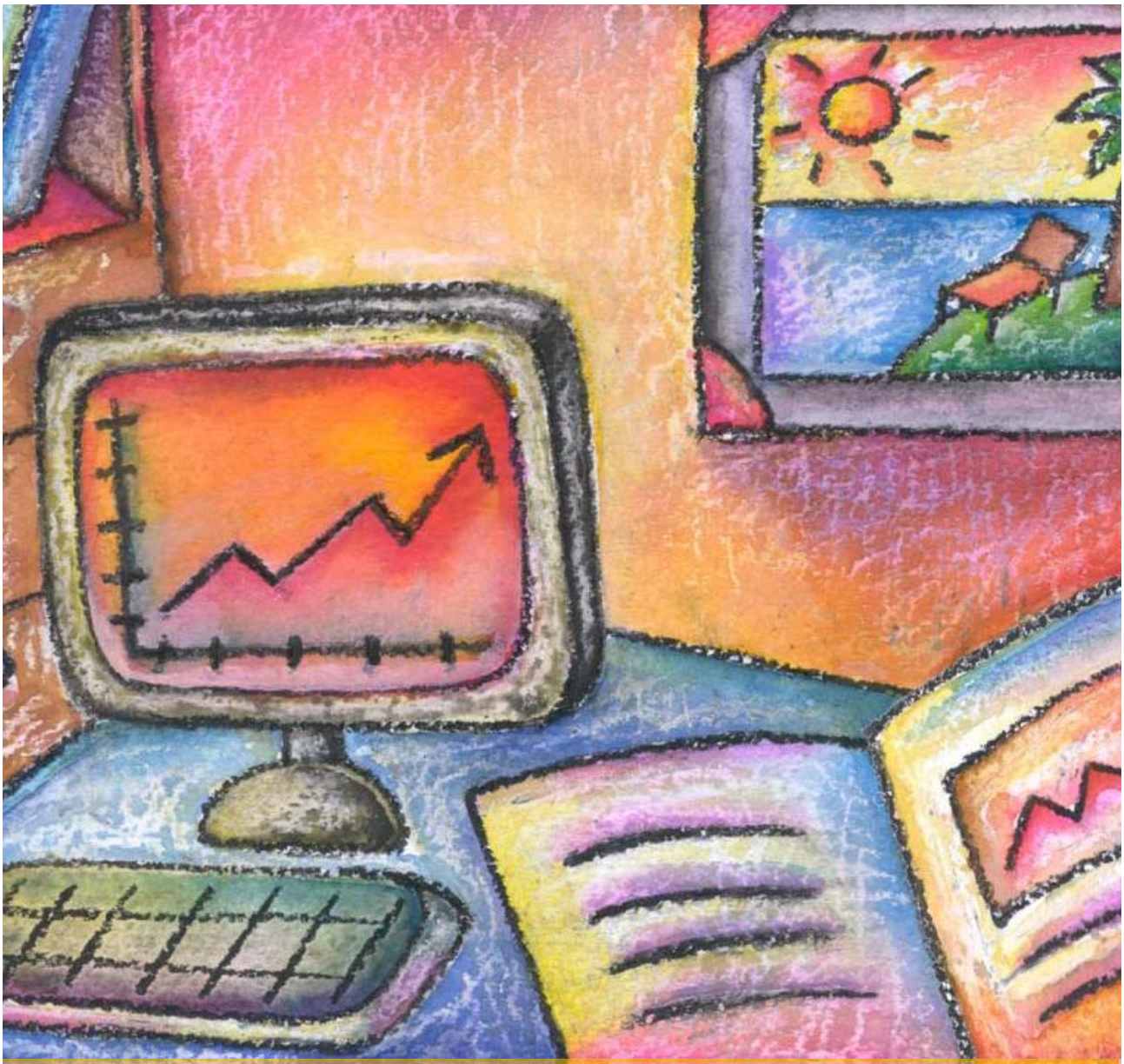


فصل هشتم

برنامه‌نویسی 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.

عناوین 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. محاسبه‌ی بیشینه و کمینه (بزرگ‌ترین و کوچک‌ترین)

یکی از پایه‌ای‌ترین الگوریتم‌ها، 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 ببینیم و مشخص کنیم که آیا عدد فعلی از تمام اعداد قبلی بزرگ‌تر (کوچک‌تر) هست یا نه. برای این کار از یک متغیر به عنوان نگه‌دارنده‌ی عدد بیشینه (کمینه) تا این مرحله استفاده می‌کنیم.

مثلاً 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 این‌جا 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)

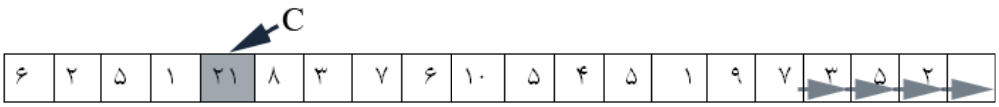
در بسیاری از وقت‌ها، نیاز است که یک مقدار را در یک خانه از آرایه قرار دهیم، بدون آن‌که محتوای قبلی آن خانه از بین برود. برای این کار باید تمام خانه‌های بعدی را یک خانه به جلو ببریم و مقدار مورد نظر را در خانه‌ی مربوطه قرار دهیم. به این عمل که حتماً در هنگام نوشتن متن در Word یا بیسیک، هنگامی که حرفی 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 آرایه‌ی۲۰ 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)

۳. 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 می‌گردند. (برنامه ابتدا مقدار K را از کاربر می‌پرسد)

۸. ۲. ۳. 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.

۸. ۲. ۴. جست‌وجو کردن (SEARCH)

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

در ادامه به بررسی مهم‌ترین الگوریتم‌های جست‌وجو 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. ساده-ترین روش این است که تمام خانه‌های آرایه را در یک حلقه بررسی کنیم. برنامه‌ی زیر این کار را انجام می‌دهد: (فرض کنید که آرایه قبلا 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 جست‌وجو، در اولین خانه‌ی آرایه باشد، جست‌وجوی بقیه‌ی آرایه بیهوده است. در حالی که اگر مقدار مورد نظر در خانه‌ی آخر آرایه باشد، جست‌وجوی تمام آرایه لازم است. اصطلاحاً به این روش جست‌وجو، جست‌وجوی 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 جست‌وجوی 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 جست‌وجو وجود دارد. نام این الگوریتم، *جست‌وجوی دودویی* (Binary Search) است. این جست‌وجو 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 جست‌وجو را با عضو وسط آرایه مقایسه می‌کنیم، 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 جست‌وجو با عضو وسط قسمت باقی‌مانده برابر شود. جست‌وجو 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 صرف‌نظر 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 رشتهٔ "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 همدیگر 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" ، "aLi" و "ALI" از نظر بیسیک مقادیر مختلفی دارند.

- کوچکتری / بزرگتری : بیسیک قابلیت انجام مقایسه بین دو رشته بر حسب حروف الفبا را داراست. 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.

۸. ۳. ۴. جداسازی از چپ و راست

به کمکم 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.

۸. 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:

```
MID$ ( رشته‌ی مورد نظر , شروع جداسازی , طول جداسازی )
```

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.

۸. ۳. ۶. جستجو

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

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

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

۸. ۳. ۷. تبدیلات

به کمک دستور STR$ می‌توان معادل رشته‌ای یک عدد را به دست آورد. دستور VAL برعکس 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 LCASE$ و UCASE$ نیز به ترتیب تمامی حروف بزرگ یک رشته را به حروف کوچک و برعکس 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 مسأله مهم نباشد.

۸. ۳. ۸. تولید رشته

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

```
STRING$( 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 اَسکی 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 دست‌آوردن 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.

```
DIM A( تعداد ستون , تعداد سطر )
```

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 نوشتن کلیه‌ی عناصر یک آرایه‌ی دوبعدی 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 اعضای یک مربع N x N را گرفته و سپس بگوید که آیا این مربع وفقی (جادویی) است یا خیر. مربع وفقی 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 وفقی نمونه‌ای از یک مربع وفقی ۳×۳ به شکل زیر است:

۸	۱۳	۶
۷	۹	۱۱
۱۲	۵	۱۰