شماره مستند: ۱۹۰/۲۵۳۷/۱/۷

جمهوری اسلامی ایران

دبیرخانه شورای عالی اطلاع رسانی

تعیین چارچوب سیستم نرم‌افزاری تشخیص خودکار اعداد در متون زبان فارسی

نسخه ۱.۰

دانشگاه علم و صنعت ایران

فروردین ۸۸

فهرست

مقدمه

یکی از بخش‌های افزونه‌ی زبان فارسی ویراستیار، سامانه‌ی تبدیل اعداد می‌باشد. عنوان بهتری که برای این سامانه می‌توان بیان کرد «سامانه‌ی تشخیص و تبدیل صورت نوشتاری اعداد فارسی» می‌باشد. صورت‌های نوشتاری ممکن برای اعداد، عبارتند از *صورت نوشتاری رقمی* و *صورت نوشتاری حرفی*. صورت نوشتاری رقمی به معنی بیان اعداد فارسی با استفاده از ارقام می‌باشد. صورت‌های رقمی قابل تشخیص توسط این سامانه در **جدول ۱** خلاصه شده‌اند.

شرح	مثال
اعداد با ارقام فارسی	۱۶۴۸۱۹۵
اعداد با ارقام انگلیسی	1648195
اعداد با جدا کننده‌ی هزارکان با ارقام فارسی	۱٬۶۴۸٬۱۹۵
اعداد با جدا کننده‌ی هزارگان با ارقام انگلیسی	1,648,195
اعداد اعشاری با ارقام فارسی	۲۰٬۸۷۳٫۲۵
اعداد اعشاری با ارقام انگلیسی	20,873.25
اعداد با ارقام عربی	١٦٤٨١٩٥

جدول ۱. صورت نوشتاری رقمی قابل تشخیص توسط سامانه‌ی اعداد

تبدیل و یکسان‌سازی انواع نوشتارهای رقمی

در این بخش به بررسی صورت‌های مختلف نمایش ارقام خواهیم پرداخت، سپس راهکارهای تبدیل ارقام از صورت‌های نوشتاری مختلف به یکدیگر را بررسی خواهیم کرد.

۲.۱ انواع صورت‌های نمایش ارقام

ارقام فارسی عبارتند از ۰۱۲۳۴۵۶۷۸۹. در جدول یونی‌کد این ارقام از کد 6F0 هگزادسیمال تا کد 6F9 هگزادسیمال قرار گرفته‌اند. ارقام عربی عبارتند از ۰١٢٣٤٥٦٧٨٩. در جدول یونی‌کد این ارقام از کد 660 هگزادسیمال تا کد 669 هگزادسیمال قرار گرفته‌اند. ارقام انگلیسی عبارتند از 0123456789. در جدول یونی‌کد این ارقام از کد 30 تا 39 هگزادسیمال قرار گرفته‌اند.

بنابراین برای تشخیص این‌که آیا نویسه‌ی دلخواهی، نمایانگر یک رقم است، کافی است کد آن نویسه را با سه بازه‌ی ذکر شده در فوق مقایسه کنیم. مقدار عددی که هر نویسه‌ی رقم، دارا می‌باشد را می‌توان از تفریق کد آن نویسه با کد رقم صفر در

همان بازه بدست آورد. مثلا کد رقم ۴ فارسی 6F4 می‌باشد و کد رقم صفر فارسی نیز 6F0 می‌باشد که تفریق این دو، عدد ۴ را بدست می‌دهد. عکس قضیه‌ی فوق نیز برقرار می‌باشد. بدین معنی که برای بدست آوردن نویسه‌ی ۴ فارسی کافی است عدد ۴ را با کد نویسه‌ی صفر فارسی جمع کنیم. این دو قانون مبنای تبدیل ارقام فارسی، عربی، و انگلیسی به یکدیگر را تشکیل می‌دهند. در **لیست ۱** الگوریتم تبدیل ارقام دلخواه به ارقام انگلیسی آمده است.

```
Function ConvertDigitToEnglish

input    in_digit: character
output   out_digit: character
locals   value: integer

if (6F0 <= code_of(in_digit) <= 6F9) then
   value = code_of(in_digit) - 6F0
   out_digit = char_of(30 + value)
else if (660 <= code_of(in_digit) <= 669) then
   value = code_of(in_digit) - 660
   out_digit = char_of(30 + value)
else
   out_digit = in_digit
```

لیست ۱. الگوریتم تبدیل ارقام به انگلیسی

۲.۲ دیگر نویسه‌های مورد استفاده در نوشتار رقمی اعداد

در نوشتارهای رقمی اعداد، به غیر از نویسه‌های ارقام، نویسه‌های دیگری نیز استفاده می‌شوند. به عنوان نمونه می‌توان از جداکننده‌های هزارگان، ممیز، و علائم مثبت و منفی (+ و -) نام برد. همچنین در نماد علمی (بالاخص در نوشتار انگلیسی اعداد) از نویسه‌ی e به عنوان نمای ۱۰ استفاده می‌شود. مثلا 2e+3 به معنای عدد ۲۰۰۰ می‌باشد.

نویسه‌های به‌کارگرفته شده برای جداکننده‌های هزارگان و ممیز در نوشتارهای فارسی و انگلیسی متفاوتند. با استفاده از یک نگاشت ساده می‌توان این دو را به یکدیگر تبدیل کرد. **جدول ۲** این نویسه‌ها را با یکدیگر مقایسه می‌کند.

شرح	نویسه‌ی فارسی/ عربی	نویسه‌ی انگلیسی
جداکننده‌ی هزارگان	، (0x66C)	, (0x2C)
ممیز اعشاری	٫ (0x66B)	. (0x2E)

جدول ۲. مقایسه‌ی نویسه‌های به کار رفته در نمایش اعداد

لازم به ذکر است که در مورد زبان‌های فارسی و عربی نویسه‌های یاد شده در بالا منحصر بفرد نیستند. در بسیاری مواقع در میان ارقام فارسی از نویسه‌های انگلیسی استفاده می‌شود و همین‌طور به جای ممیز اعشاری، استفاده از نویسه‌ی slash (/) در فارسی بسیار رایج است که در روال‌های تبدیل نوشتارها این موضوع را نیز باید مدنظر قرار داد.

۲.۳ یکسان‌سازی نوشتارهای رقمی اعداد

در سرتاسر سامانه‌ی تبدیل اعداد، با نوشتار عددی اعداد و روال‌های مرتبط با آن سر و کار خواهیم داشت و چون همانگونه که ذکر شد، نوشتار رقمی می‌تواند به سه صورت فارسی، عربی، و انگلیسی باشد؛ بنابراین بهتر است که یک صورت نوشتاری از سه صورت نوشتاری فوق را به عنوان صورت نرمال نوشتار رقمی برگزینیم، و تمامی الگوریتم‌ها را بر پایه‌ی آن نوشتار

پیاده‌سازی کنیم و در نهایت نتیجه را به هر نوشتار دلخواه تبدیل کنیم. از آن‌جا که در زبان‌های برنامه نویسی متداول، توابع کتابخانه‌ای فراوانی بر پایه‌ی زبان انگلیسی وجود دارد، بنابراین نوشتار انگلیسی را به عنوان صورت نرمال نمایش اعداد برمی‌گزینیم.

تبدیل اعداد از نوشتار رقمی به نوشتار فارسی

۳.۱ تشکیل نگاشتی از مقادیر به واژگان

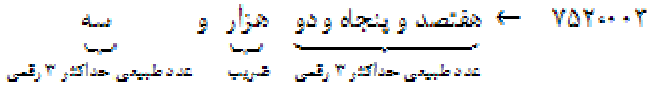
به عنوان یکی از ابتدایی‌ترین مراحل تبدیل اعداد از نوشتار رقمی به نوشتار فارسی یک نگاشت از مقادیر عددی کلیدی به واژگانی که در زبان فارسی برای آن‌ها استفاده می‌شود، می‌سازیم. منظور از مقادیر عددی کلیدی مقادیری است که تمامی اعداد با ترکیبی از آن‌ها ساخته می‌شود. این اعداد عبارتند از:

- اعداد ۰ تا ۹
- اعداد ۱۰ تا ۱۹
- ضرایب ۱۰ از ۲۰ تا ۹۰
- ضرایب ۱۰۰ از ۱۰۰ تا ۹۰۰
- اعداد ۱۰۰۰ (هزار)، ۱٬۰۰۰٬۰۰۰ (میلیون)، ۱٬۰۰۰٬۰۰۰٬۰۰۰ (میلیارد)، و ۱٬۰۰۰٬۰۰۰٬۰۰۰٬۰۰۰ (تریلیارد)

زین‌پس در شبه‌کدهای این مستند، این نگاشت را با تابعی به نام MapNum2Str خواهیم شناخت. مثلا MapNum2Str(200) رشته‌ی «دویست» را بدست خواهد داد.

۳.۲ تبدیل اعداد حداکثر سه رقمی به نوشتار فارسی

اعداد طبیعی حداکثر ۳ رقمی یکی از اجزای اصلی تشکیل دهنده‌ی اعداد بزرگتر هستند، به طوری که اعداد بزرگتر به نحوی ترکیب این اعداد با یک سری ضرایب هستند. به طور مثال تبدیل عدد زیر را در نظر بگیرید.



الگوریتم تبدیل با استفاده از نگاشت یاد شده در بخش ۳.۱ در **لیست ۲** آمده است.

```
Function ConvertUpTo3Digits

inputs    n: integer
outputs   s: string
locals    c: integer

if (n > 99) then
   c = n / 100;
   s = MapNum2Str(c * 100);
   n = n - (c*100);
   if(n < = 0) return;
```

```
   else s = s. " و ";
if (n > 20) then
   c = n / 10;
   s = s.MapNum2Str(c * 10);
   n = n - c*10;
   if (n <= 0) return;
   else s = s. " و ";
if (n > 0) then
   s = s.MapNum2Str(n);
```

لیست ۲. الگوریتم تبدیل اعداد حداکثر ۳ رقمی از حالت رقمی به حارت حرفی

۳.۳ تبدیل اعداد طبیعی در حالت کلی

اکنون با در دست داشتن الگوریتم تبدیل اعداد طبیعی حداکثر ۳ رقمی، می‌توان مبدل اعداد طبیعی در حالت کلی‌تر را طراحی کرد. این الگوریتم نیز از لحاظ ساختار، بسیار شبیه الگوریتم تبدیل اعداد طبیعی حداکثر ۳ رقمی می‌باشد. این الگوریتم در **لیست ۳** ارائه شده است.

```
Function ConvertIntegers

inputs    n: integer
outputs   s: string
locals    c: integer

if (n < 0) then
   s = "منهای ";
   n = -n;
if (n == 0) then
   s = MapNum2Str(n);
   return;
if (n > 999999999999)then
   c = n / 1000000000000;
   s = s.ConvertUpTo3Digits(c). " ".MapNum2Str(1000000000000);
   n = n - (c * 1000000000000);
   if(n < = 0) return;
   else s = s. " و ";
if (n > 999999999)then
   c = n / 1000000000;
   s = s.ConvertUpTo3Digits(c). " ".MapNum2Str(1000000000);
   n = n - (c * 1000000000);
   if(n < = 0) return;
   else s = s. " و ";
if (n > 999999)then
   c = n / 1000000;
   s = s.ConvertUpTo3Digits(c). " ".MapNum2Str(1000000);
   n = n - (c * 1000000);
   if(n < = 0) return;
   else s = s. " و ";
if (n > 999)then
   c = n / 1000;
   s = s.ConvertUpTo3Digits(c). " ".MapNum2Str(1000);
   n = n - (c * 1000);
   if(n < = 0) return;
   else s = s. " و ";
if (n > 0)then
   s = s.ConvertUpTo3Digits(n);
```

لیست ۳. الگوریتم تبدیل اعداد طبیعی در حالت کلی

۳.۴ تبدیل اعداد حقیقی از نوشتار رقمی به نوشتار فارسی

در این قسمت هدف تعمیم الگوریتم تبدیل اعداد طبیعی به حالت حقیقی است، در واقع ما بدنبال آن هستیم که چنان‌چه عدد ورودی، یک عدد با ممیز اعشاری بود، بتوانیم آن را با موفقیت به حالت نوشتار فارسی تبدیل کنیم. برای این منظور ابتدا

باید بررسی کنیم که آیا عدد ورودی یک عدد طبیعی است یا اعشاری. چنان‌چه ورودی یک عدد طبیعی بود، از روال ConvertIntegers که در بخش ۳.۳ توضیح داده شد، می‌توان استفاده کرد، در غیر این صورت با انجام چند عملیات ساده می‌توان عدد اعشاری را به اجزای طبیعی خرد کرد . کار را به روال آشنای ConvertIntegers سپرد.

برای این منظور ابتدا عدد را به رشته‌ی انگلیسی (نرمال) تبدیل می‌کنیم و از مکان ممیز (نویسه‌ی نقطه در انگلیسی) رشته را به دو بخش تقسیم می‌کنیم. این الگوریتم در **لیست ۴** آمده است.

```
Function ConvertRealNumbers

inputs     r: real
outputs    s: string
locals     r_str: string
           dot_index: integer
           n1: integer
           n2: integer
           order: integer

r_str = String-of-Floating-Point-Representation-of(r);
dot_index = index-of-point-in(r_str);
if (dot_index > 0) then
   n1 = the-number-before-dot
   n2 = the-number-after-dot
   if (n1 != 0) then
       s = ConvertIntegers(n1). " ممیز ";
   s = s.ConvertIntegers(n2);
   order = 10 ^ number-of-digits(n2);
   s = s. " ".CreateOrdinalNumber(order);
else
   s = ConvertIntegers(r as integer);
```

لیست ۴. الگوریتم تبدیل اعداد حقیقی از نوشتار رقمی به نوشتار فارسی

در الگوریتم فوق از تابعی به نام CreateOrdinalNumber استفاده شده. این تابع، رشته‌ی نوشتار عدد را می‌گیرد و رشته‌ی نوشتار ترتیبی معادل را بدست می‌دهد. به عنوان مثال «سه» را به «سوم» و «میلیون» را به «میلیونیم» تبدیل می‌کند.

تبدیل اعداد از نوشتار فارسی به نوشتار رقمی

۴.۱ ساختن نگاشت از رشته‌های بسیط اعداد به مقادیرشان

همانند آن‌چه در مورد تبدیل اعداد از نوشتار رقمی به فارسی در بخش ۳.۱ بحث شد، در مورد عکس این تبدیل نیز به عنوان ماده‌ی خام تمامی الگوریتم‌ها، نیاز به نگاشتی از رشته‌های بسیط اعدادی که در متن‌های فارسی احتمال مشاهده‌ی آن‌ها وجود دارد به مقادیر عددی‌شان وجود دارد. برای این منظور رشته‌ی اعداد، به همراه فرم ترتیبی‌شان به نگاشت افزوده خواهند شد. رشته‌هایی که باید در نگاشت مورد بحث قرار بگیرند، از قرار زیرند:

- رشته‌های «صفر» و «صفرم»
- رشته‌های «یک» تا «نه» به همراه حالت ترتیبی «یکم»، «اول» تا «نهم»
- رشته‌های «ده» تا «نوزده» به همراه حالت ترتیبی‌شان
- ضرایب ۱۰ از «بیست» تا «نود» به همراه حالت ترتیبی‌شان

- ضرایب ۱۰۰ از «صد» تا «نهصد» به همراه حالت ترتیبی‌شان
- رشته‌های «هزار»، «میلیون»، «میلیارد»، «تریلیارد»
- برخی پیشوندهای خاص، مانند «پان» که به عدد ۵ نگاشته شده.

دامنه‌ی نگاشت فوق، تنها محدود به موارد بالا نمی‌باشد، بلکه سعی شده، تمامی شیوه‌های مختلف بیان یک عدد در نظر گرفته شود، چه شیوه‌های مختلف رسمی، چه شیوه‌های محاوره‌ای. مثلا در نگاشت فوق علاوه بر «شش» رشته‌ی «شیش» نیز افزوده شده. همچنین علاوه بر «پانصد» رشته‌ی «پونصد» نیز در نگاشت وجود دارد.

زین‌پس در شبه‌کدهای این مستند، این نگاشت را با تابعی به نام MapStr2Num خواهیم شناخت.

۴.۲ یافتن توده‌های متن شامل رشته‌های اعداد

فرض کنید، متن نسبتا بزرگی در اختیار دارید که در نقاطی از آن چند عدد به حالت نوشتاری وجود دارند، و هدف تشخیص و به دست آوردن مقدار آن‌ها باشد. رشته‌هایی که این اعداد را تشکیل می‌دهند همان رشته‌های موجود در دامنه‌ی نگاشت فوق الذکر است، به انضمام رشته‌های زیر:

- حرف «و» که حرف ربط بین اجزای اعداد است. مانند: «صد و ده»
- حرف «م» در انتهای کلمه. این حرف عدد ترتیبی و همین‌طور کسری می‌سازد. مانند «پنجم»، یا، «سه هفتم»
- کلمه‌ی «ممیز» که گاهی اوقات از آن برای بیان ممیز استفاده می‌شود، مانند «صد و بیست ممیز بیست و پنج صدم»
- کلمه‌های «منفی» و «منهای» که نشان دهنده‌ی اعداد منفی هستند
- کلمه‌ی «نیم» به معنای «پنج دهم»

توجه کنید که در میان رشته‌های فوق حرف «و» هم می‌تواند یک کلمه‌ی جدا باشد، و هم این که در انتهای کلمه ظاهر شده باشد (گرچه دومی از لحاظ املایی خطا به حساب می‌آید، اما چون خطای بسیار رایجی است، این وضعیت نیز تشخیص داده می‌شود). همچنین حرف «م» نیز باید در انتهای کلمه ظاهر شود، به شرطی که کلمه‌ی پیش از آن در دامنه‌ی نگاشت موجود باشد. با این تفاسیر در داخل متن محدوده‌ای که در آن *احتمالا* عدد مجازی ظاهر شده باشد تشخیص داده می‌شود، و چون هنوز بر روی محتویات این محدوده هیچ تحلیلی صورت نگرفته، آن را توده[1] می‌نامیم و به کلماتی که آن توده را می‌سازند عنصر توده[2] می‌گوییم. هر عنصر توده می‌تواند در بر دارنده‌ی یک یا چند ثابت عددی باشد. مثلا کلمه‌ی «نیم» به تنهایی شامل سه ثابت (۵، ۱۰، م)، به معنای «پنج دهم» است[3]. به دنباله‌ی ثابت‌های عددی که هر عنصر توده در بر دارد، لیست مقادیر آن عنصر توده می‌گوییم. به عنوان مثال توده‌ی «هفت‌صد و هشتادو شش و پنج صدم» از عناصر توده‌ی «هفت»،

[1] Chunk
[2] Chunk-Element
[3] در پیاده‌سازی به جای کلماتی چون «م» و «و» و «منفی» و ... می‌توان از اعداد منفی استفاده کرد. بنابراین فرض بر ثابت عددی بودن این کلمات، چندان فرض اشتباهی نیست.

«صد»، «و»، «هشتادو»، «شش»، «و»، «پنج»، «صدم» تشکیل شده، که با کنار هم گذاردن لیست مقادیر تمامی این عناصر توده خواهیم داشت (۷، ۱۰۰، و، ۸۰، و، ۶، و، ۵، ۱۰۰، م). حال صورت مسئله به یافتن عددی که از کنارهم قرار دادن اعضای این لیست ساخته خواهد شد، کاهش می‌یابد.

در نظر گرفتن چند نکته به هنگام استخراج توده‌ها می‌تواند توده‌های بهینه‌تری را بدست دهد. یکی آن که هیچ‌گاه عددی با عنصر توده‌ی «و» شروع نمی‌شود و با آن خاتمه نمی‌یابد. دیگر آن‌که «م» نشان‌دهنده‌ی انتهای یک عدد است، بنابراین می‌توان به راحتی آن را انتهای یک توده دانست و توده‌ی جدید را (در صورت وجود) از بعد از آن آغاز کرد. همین‌طور عنصر توده‌ی «منفی» یا «منها» همواره نشانگر ابتدای یک عددند، بنابراین می‌توان در صورت مشاهده‌ی آن در وسط یک توده، آن توده را از آن نقطه به دو توده تقسیم کرد، به طوری که «منفی» ابتدای توده‌ی دوم قرار بگیرد.

۴.۳ استخراج اعداد صحیح

فرض کنید که توده‌ای از اعداد داریم که تنها دربر دارنده‌ی اعداد صحیح است و فاقد اعداد اعشاری است. در این قسمت نشان می‌دهیم که چگونه می‌توان عدد مورد نظر را از این توده استخراج کرد. به این نکته توجه داشته باشید، که یک توده ممکن است دربر گیرنده‌ی بیش از یک عدد باشد. مثلا توده‌ی «هفتصد و شصت و پنج» در برگیرنده‌ی یک عدد است، توده‌ی «هفت و شصت و پنج» در برگیرنده‌ی دو عدد، و توده‌ی «هفت و شش و پنج» در برگیرنده‌ی سه عدد است. در ابتدا احتیاج به یک الگوریتم داریم که لیست مقادیر یک توده را دریافت کند و عددی که در آن لیست به آن اشاره شده را برگرداند، اما در صورتیکه موفق به بدست آوردن این عدد نشد، به ما بگوید که این لیست را از کجا باید به دو قسمت تقسیم کنیم. مثلا این الگوریتم باید به ازای ورودی لیست متناظر با «هفتصد و شصت و پنج» به ما عدد ۷۶۵ را برگرداند و به ازای ورودی «هفت و شصت و پنج» لیست را به دو قمست «هفت» و «و شصت و پنج» تقسیم کند. فراخوانی این الگوریتم به طور بازگشتی به ازای لیست‌های جدید ادامه خواهد یافت تا جایی که یا یک‌سری عدد بدست بیاید، یا یک سری زیر لیست که قابل تبدیل به عدد نیستند. اگر مثال قبل را ادامه بدهیم در انتها ۳ زیر لیست ساخته خواهد شد که متناظرند با «هفت»، «و»، «شصت و پنج» که در آن زیرلیست «و» نمایانگر هیچ عددی نیست، و زیرلیست‌های «هفت» و «شصت و پنج» به ترتیب نمایانگر اعداد ۷ و ۶۵ می‌باشند. در این متن الگوریتمی را که چنین کاری برای ما انجام دهد، ExtractIntegerFromList می‌نامیم، که الگوی فراخوانی آن در **لیست ۵** آمده است. به علت بزرگی و پیچیدگی مراحل انجام این الگوریتم از ارائه جزئیات آن در این‌جا خودداری می‌کنیم. یک نمونه پیاده‌سازی این الگوریتم به زبان C# در کلاس `PersianRealNumberParser` و تابع `GenerateIntegralPartFrom` قابل دسترسی است، توجه کنید که در پیاده‌سازی ذکر شده، به جای برگرداندن موقعیت برش عدد، از تکنیک‌های Exception Handling استفاده شده است.

```
Function ExtractIntegerFromList

inputs    list_of_values: list of integers
          start_index: integer
          end_index:  integer
outputs   v: integer
          c: cut information
```

لیست ۵. الگوی فراخوانی و مقادیر بازگشتی الگوریتم تشکیل عدد صحیح از لیستی از اعداد

اکنون با در دست داشتن الگوریتم ExtractIntegerFromList می‌توانیم الگوریتم کلی ExtractIntegers را ارائه دهیم که در آن از تکنیک‌های بازگشتی توضیح داده شده استفاده می‌شود، تا یک توده‌ی عددی به *مجموعه‌ای* از اعداد صحیح که ممکن است داخل آن باشد تبدیل شود. این الگوریتم در **لیست ۶** ارائه شده است، که در آن ماهیت بازگشتی این الگوریتم قابل مشاهده است.

```
Function ExtractIntegers

inputs     chk: Chunk
outputs    s: list of integers
locals     l: list of integers
           n: integer
           c: cut information
           chk1: Chunk
           chk2: Chunk

l = list of values for all chunk-elements in chk;
s = emplty list of integers

[n, c] = ExtractIntegerFromList(l);
if (c is empty) then
   s.append(n);
else if l.length == 1 and c is not empty then
   s = empty list of integers;
else
   chk1 = sub-chunk of chk before cut index in c;
   chk2 = sub-chunk of chk after cut index in c;
   s.append(ExtractIntegers(chk1));
   s.append(ExtractIntegers(chk2));
```

لیست ۶. الگوریتم استخراج مجموعه‌ی اعداد صحیح داخل یک توده‌ی عددی

۴.۴ تشخیص اعداد اعشاری که در آن کلمه‌ی «ممیز» ذکر شده

صورت نوشتاری حرفی ساختار عدد اعشاری‌ای که با ذکر کلمه‌ی «ممیز» بیان شده باشد به شکل زیر است:

عدد صحیح - ممیز - عدد طبیعی - عدد طبیعی- م

به عنوان مثال عدد زیر را در نظر بگیرید:

صد و هفده ممیز پنجاه ده هزارم

که در آن عدد صحیح ابتدایی «صد و هفده» است که بعد از آن کلمه‌ی «ممیز» ذکر شده، و پس از آن عدد طبیعی «پنجاه» و بلافاصله بعد از آن نیز عدد طبیعی «ده هزار» قرار گرفته که کل عبارت با حرف «م» به پایان می‌رسد. بنابراین کافی است الگوریتمی ارائه دهیم، که الگوی فوق را در مورد اعداد اعشاری «ممیز» دار بررسی کند و اجزای اصلی این الگو را استخراج کند. همان‌گونه که مشاهده می‌شود، اجزای اصلی این الگو، همگی اعداد طبیعی هستند که در مورد چگونگی استخراج آن‌ها در بخش‌های قبل توضیح داده شد. اکنون با ترکیب ساده‌ی همان الگوریتم‌ها الگوی جدید را استخراج می‌کنیم.

در بخش ۴.۳ توضیح داده شد، که الگوریتم ExtractIntegers (که جزئیات آن در **لیست ۶** آورده شده است)، تمامی نمونه‌های اعداد طبیعی واقع در یک توده‌ی کلمات را استخراج می‌کند. این ویژگی الگوریتم، به ما کمک می‌کند که دو عدد طبیعی بعد

از ممیز را با موفقیب استخراج کنیم. سپس باید مطمئن شویم که این اعداد طبیعی استخراج شده، شرایط زیر را دارا می‌باشند:

- تعداد اعداد طبیعی استخراج شده باید دقیقا دو تا باشد
- بلافاصله بعد از عدد طبیعی دوم حرف «م» باشد
- بین عدد طبیعی اول و عدد طبیعی دوم، هیچ حرف یا کلمه‌ای نباید قرار گرفته باشد

با توجه به شرایط یاد شده در مورد بخش اعشاری عدد، الگوریتم ExtractFloatingPart را به صورت **لیست ۷** ارائه می‌دهیم. توجه کنید که در این الگوریتم اندیس ممیز به صورت ورودی به الگوریتم ارسال می‌شود. همچنین خروجی این الگوریتم یک عدد اعشاری نیست، بلکه دو عدد است که یک صورت، و دیگر مخرج بخش اعشاری است. این خصوصیات به ما کمک خواهند کرد تا در وضعیت‌های عمومی‌تری که در آن‌ها محل ممیز به درستی مشخص نیست و تشخیص اعداد کسری از این الگوریتم استفاده کنیم.

```
Function ExtractFloatingPart

inputs     chk: Chunk
           dot_index: integer
outputs    numerator: integer
           denominator: integer
locals     l: list of integers
           sub_chk: Chunk

if that chk does not end with "م" then
   denominator = 0;
   return;
sub_chk = sub-chunk of chk from dot_index to final "م" exclusive
l = ExtractIntegers(sub_chk);
if l.length == 2 and
   there are no chunk-elements in chk between sub_chunks for l[1] and l[2] and
   there are no chunk-elements in chk between l[2] and final "م" then

   numerator = l[1];
   denominator = l[2];
else
   denominator = 0;
```

لیست ۷. الگوریتم تشخیص و استخراج قسمت اعشاری عدد

اکنون با در دست داشتن الگوریتم استخراج قسمت اعشاری اعداد حقیقی، می‌توانیم الگوریتم تشخیص و استخراج اعداد حقیقی که در آن‌ها کلمه‌ی «ممیز» ذکر شده را تدوین کینم. جزئیات این الگوریتم در **لیست ۸** مشاهده می‌شود.

```
Function ExtractRealNumberWithMomayez

inputs     chk: Chunk
outputs    mantissa: integer
           numerator: integer
           denominator: integer
locals     chk1: Chunk
           dot_index: integer
           m: integer
           num: integer
           denom: integer

if that chk does not end with "م" then
    denominator = 0;
```

```
    return;
dot_index = find chunk-element in chk that equals "ممیز"
chk1 = sub-chunk of chk from beginning up to dot_index exclusive
mantissa = ExtractIntegers(chk1)[1];
[numerator, denominator] = ExtractFloatingPart(chk, dot_index);
if denominator != 0 then
    value = mantissa + numerator / denominator;
```

لیست ۸. الگوریتم استخراج اعداد حقیقی با ذکر کلمه‌ی «ممیز»

۴.۵ تشخیص اعداد اعشاری در حالت کلی

در بخش قبل حالت ساده‌ای را بررسی کردیم که در آن هنگام بیان اعداد اعشاری، کلمه‌ی «ممیز» صریحا بیان شده باشد. اما در اکثر حالات هنگام بیان اعداد اعشاری، کلمه‌ی ممیز بیان نمی‌شود. این موضوع دو مشکل می‌تواند ایجاد کند. یکی آن که محل ممیز باید یافت شود و دوم آن که ممکن است محل منحصر بفردی برای ممیز وجود نداشته باشد، که در این صورت الگوریتم ما چند خروجی باید داشته باشد. مثلا عدد «هجده و بیست و پنج صدم» تنها یک خروجی ماکسیمال دارد که آن عدد ۱۸٫۲۵ است. اما عدد «صد و بیست و پنج صدم» می‌تواند سه خروجی داشته باشد «۱۲۰٫۰۵» و « ۱۰۰٫۲۵» و $\frac{۱۲۵}{۱۰۰}$.

آن‌چه که برما مشخص است این است که یک یا چندتا از «و» ها نقش ممیز را بازی می‌کند که در صورت جایگزینی آن «و» با کلمه‌ی «ممیز» صورت مسئله‌ی تشخیص اعداد اعشاری در حالت کلی به صورت مسئله‌ی تشخیص اعداد اعشاری با «ممیز» کاهش می‌یابد. همچنین یک حالت اضافی ممکن است وجود داشته باشد، که عدد ورودی کسری باشد. بنابراین برای تشخیص اعداد اعشاری در حالت کلی، فرض می‌کنیم «و» ها، ممیز هستند. با شروع از آخرین «و» و جایگزینی آن با ممیز و حرکت به سمت ابتدای عدد، و اعمال الگوریتمی شبیه **لیست ۸** می‌توان اعداد اعشاری در این حالت را تشخیص داد. الگوریتم تشخیص اعداد اعشاری در حالت کلی، در **لیست ۹** ارائه شده است.

```
Function ExtractFloatingPointNumbers

inputs     chk: Chunk
outputs    values: list of real numbers
locals     vaavs: list of integer numbers
           value: real
           mantissa: integer
           numerator: integer
           denominator: integer
           chk1: Chunk

values = empty list of real numbers;
if chk does not end with "م" then
    return;
vaavs = find all indices of "و" in chk
reverse for each dot_index in vaavs do
    chk1 = sub-chunk of chk from beginning up to dot_index exclusive
    [numerator, denominator] = ExtractFloatingPart(chk, dot_index);
    if denominator != 0 then
        mantissa = ExtractIntegers(chk1)[1];
        value = mantissa + numerator / denominator;
        values.append(value);
```

لیست ۹. الگوریتم تشخیص اعداد اعشاری در حالت کلی