The Asian Journal of TEX, Volume 1, No. 1, April 2007
Article revision 2007/02/22

연분수 자동 조판

Automatic Typesetting of Continued Fraction

남수진 Soojin Nam
㈜다음커뮤니케이션 sjnam@ktug.or.kr

ABSTRACT TEX은 조건문, 반복문, 매크로, 파일 입출력과 같은 다른 대부분의 컴퓨터 프로그래밍 언어가 가지고 있는 특징을 가지고 있다. 이러한 컴퓨터 프로그래밍 언어로서의 TEX은 이를 워드프로세서와 뚜렷이 구별하게 한다. 그리고 TEX이 가지고 있는 여러가지 프로그래밍 언어의 특성을 가장 잘 표현하는 것이 "매크로"일 것이다. 이 글에서는 주어진 분수를 연분수로 바꾸는 매크로를 작성하면서 매크로 프로그래밍 언어로서의 TEX의 면모를 살펴본다.

1 연분수란 무엇인가?

연분수(連分數; continued fraction)는 다음과 같은 모양의 분수를 뜻한다.

$$x = a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{a_3 + \cfrac{1}{a_4}}}}$$

여기에서 a_0은 임의의 정수를, 그리고 자연수 n에 대해 a_n은 양의 정수를 나타낸다. 일반적으로 연분수는 분자와 분모에 임의의 값을 가질 수 있는데, 위와 같이 분자가 모두 1인 연분수를 특별히 단순(simple) 연분수 또는 정규(regular) 연분수라고 한다. 연분수에 관한 보다 자세한 설명은 [4]를 참조하자. 이 글에서는 길이가 유한한 단순 연분수를 다룬다.

1.1 연분수의 의미

연분수는 수학적으로 가장 순수한 수인 정수만을 이용해서 실수를 표기하려는 동기로 연구하게 되었다. 열 손가락을 가지고 있는 인간은 십진수를 사용하여 실수를 소수점 형태로 표기하는데 익숙하다. 예를 들어, $\frac{415}{93}$는 $4.462365591\cdots$ 로 표기하는데 소수점 이하의 수들은 무한히 계속되므로 적당한 자리에서 반올림하여 표기하기도 한다. 예를 들어 소수점 이하 다섯 번째 자리에서 반올림하면 4.4624가 된다. 이처럼 실수를 소수점 형태로 표기할 때 무한한 수를 모두 표기할 수 없으므로 소수점 이하 특정 자리에서 잘라내야 하고

따라서 원래의 실수 값과 항상 오차가 발생한다. 하지만 연분수를 이용하면 어떻게 될까?

$$4+\cfrac{1}{2+\cfrac{1}{6+\cfrac{1}{7}}}$$

위에서 보듯이 $\dfrac{415}{93}$를 정수만으로 너무나 깔끔하게 표기할 수 있다. 이 연분수는 반올림하여 근사값으로 표기하는 소수점 형태와 달리 그 자체로 오차 없이 정확한 값이다.

연분수는 최적의 유리근사(rational approximation)를 제공해 준다 [4]. 이는 연분수가 다른 어떤 방법보다도 주어진 실수를 근접하게 표현하는 능력을 가지고 있다는 뜻이다. 예를 들어보자. 원주율 π는 십진 소수점 표기로 보면 3, $\frac{31}{10}$, $\frac{314}{1000}$, $\frac{3141}{1000}$과 같은 근사치를 가지는데, π를 연분수로 표기하면 훨씬 더 나은 근사치를 얻을 수 있다. 원주율 π는 $[3; 7, 15, 1, 292, \ldots]$와 같은 연분수로 나타낼 수 있다. 따라서 π의 근사치는 바로 3, $\frac{31}{10}$, $\frac{314}{1000}$, $\frac{3141}{1000}$보다는 $[3]$, $[3; 7]$, $[3; 7, 15]$, $[3; 7, 15, 1]$로 표현되는 3, $\frac{22}{7}$, $\frac{333}{106}$, $\frac{355}{113}$이 더욱 정확하다.

$$\begin{aligned}
/\!/3/\!/ &= 3 && = \mathbf{3}\\
/\!/3,7/\!/ &= 3+\frac{1}{7}=\frac{22}{7} && = \mathbf{3.14}285714286\\
/\!/3,7,15/\!/ &= 3+\frac{1}{7+\frac{1}{15}}=\frac{333}{106} && = \mathbf{3.1415}0943396\\
/\!/3,7,15,1/\!/ &= 3+\frac{1}{7+\frac{1}{15+1}}=\frac{355}{113} && = \mathbf{3.141592}92035
\end{aligned}$$

1.2 연분수 계산

임의의 분수를 연분수로 바꾸는 방법을 살펴보자. 분수 $\dfrac{415}{93}$를 연분수로 바꾸는 첫 번째 단계는 다음과 같다.

$$\frac{415}{93} = 4+\frac{43}{93}$$

두 번째 단계로, 분수 $\dfrac{43}{93}$은 $\dfrac{1}{\frac{93}{43}}$과 같으므로 앞과 동일한 방법으로 $\dfrac{93}{43}=2+\dfrac{7}{43}$ 을 얻을 수 있다. 또한 분수 $\dfrac{7}{43}$은 $\dfrac{1}{\frac{43}{7}}$과 같으므로 $\dfrac{43}{7}=6+\dfrac{1}{7}$ 을 얻을 수 있고, 최종적으로 다음과 같은 연분수를 얻는다.

$$\frac{415}{93} = 4+\cfrac{1}{1+\cfrac{1}{2+\cfrac{1}{6+\cfrac{1}{7}}}}$$

1.3 연분수 표기

연분수는 여러가지 형태로 표기할 수 있는데, 다음과 같은 모양의 표기법을 일반형 표기

법이라고 한다.

$$3+\cfrac{1}{1+\cfrac{1}{4+\cfrac{1}{1+\cfrac{1}{5+\cfrac{1}{9+\cfrac{1}{2}}}}}}$$

연분수의 리스트형 표기법은 분자가 1인 단순 연분수의 경우 사용할 수 있는데 일반형의 축약 형태로 다음과 같이 표기한다.

$$[3;1,4,1,5,9,2] \qquad \text{또는} \qquad /\!/3,1,4,1,5,9,2/\!/$$

리스트 표기법과 관련한 유한 연분수의 재미있는 특성들이 알려져 있다. 먼저 아래의 식은 항상 성립한다.

$$[a_0;a_1,a_2,a_3,\ldots,a_n,1] \quad = \quad [a_0;a_1,a_2,a_3,\ldots,a_n+1]$$

예를 들어 $[5;4,3,2,1]$과 $[5;4,3,3]$은 동일한 수 $\frac{225}{43}$를 나타낸다. 또한 주어진 연분수 $[a_0;a_1,a_2,a_3,\ldots,a_n]$의 역수는 $[0;a_0,a_1,a_2,a_3,\ldots,a_n]$의 형태를 가진다. 즉,

$$[a_0;a_1,a_2,a_3,\ldots,a_n,1]\cdot[0;a_0,a_1,a_2,a_3,\ldots,a_n] \quad = \quad 1$$

예를 들어 $[5;4,3,2,1]=\frac{225}{43}$인 반면 $[0;5,4,3,2,1]$은 그 역수인 $\frac{43}{225}$이다.

2 연분수 매크로

이 절에서는 분수를 연분수로 자동으로 바꾸는 연분수 TEX 매크로에 대해 설명한다. 앞으로 만들어 나갈 매크로는 대부분 정수만을 다루는 연산이 대부분이므로 TEX이 산술 연산을 하는 방법을 미리 알아두자.

2.1 TEX의 산술 연산

TEX은 정수를 저장할 수 있는 카운트 레지스터(count register)라고 불리는 저장 장소를 `\count0`부터 `\count255`까지 모두 256개 가지고 있다. 따라서 산술 연산 도중 값을 저장할 일이 생기면 위의 256개의 레지스터 중에서 골라서 사용하면 된다. 이 때 TEX이 기본으로 사용하는 레지스터 또는 현재 사용하고 있는 레지스터를 고르면 문제가 발생할 수 있다. 하지만 사용하지 말아야할 레지스터를 일일이 기억하는 것은 귀찮은 일이므로 TEX은 `\newcount`라는 명령어를 제공한다. 이 명령어를 통해 사용자는 원하는 이름의 레지스터를 손 쉽게 만들 수 있다. 예를 들어 `\foo`라는 레지스터를 만들고자 한다면, `\newcount\foo`라고 하면 그만이다. `\foo`라는 레지스터가 256개의 레지스터 중의 몇 번째인가는 생각할 필요가 없다.

카운트 레지스터를 이용해 할 수 있는 연산은 더하기(\advance), 곱하기(\multiply), 그리고 나누기(\divide)이다. 여기서 주의할 것은 TEX은 정수만을 다루기 때문에 사칙 연산의 결과도 모두 정수라는 점이다. 현재 \foo의 값이 3일때, 다음의 연산에 대한 결과로 \foo가 갖는 값은 각각 5, 6, 그리고 1이다.

```
\advance\foo by 2     \multiply\foo by 2     \divide\foo by 2
```

마지막 나누기의 연산 결과가 1.5가 아닌 이유는 앞에서 설명했듯이, TEX 연산의 결과는 항상 정수이기 때문이다. 빼기 연산은

```
\advance\foo by -2
```

와 같이 음수를 더해주면 된다.

산술 연산에서 빠질 수 없는 것이 두 값의 비교이다. 두 개의 카운트 레지스터에 저장되어 있는 정수의 값들을 비교하는 TEX 명령어는 \ifnum이고, 비교 연산자는 >, =, < 세 가지를 사용할 수 있다. 예를 들면 다음과 같다.

```
\ifnum\foo > 0 양수 \else 0 또는 음수 \fi
```

카운트 레지스터 \foo가 가지고 있는 값이 홀수인지 짝수인지 알아보는 방법을 살펴보자. 예를 들어 \foo가 가지고 있는 값을 3이라고 하자. \foo를 2로 나누어주는 TEX 연산을 수행하면 정수 결과만을 허용하는 TEX의 특성으로 인해 \foo는 1이 되고 여기에 다시 2를 곱하면 \foo는 2가 된다. 즉 레지스터의 값이 홀수일 때는 2로 나눈 다음에 다시 2를 곱하면 그 값이 원래의 값과 달라지고, 짝수일 때는 변하지 않는다.

```
\newcount\temp     % 임시 저장장소 \temp를 만든다.
\temp=\foo         % 비교를 위해서 \foo의 값을 잠시 저장한다.
\divide\foo by 2   % \foo를 2로 나눈다.
\multiply\foo by 2 % \foo에 2를 곱한다.
\ifnum\foo=\temp 짝수 \else 홀수 \fi % 원래 값과 비교한다.
```

그런데 TEX은 레지스터에 저장된 정수가 짝수인지 홀수인지를 판단하는 명령어 \ifodd를 이미 가지고 있다. 이제 본격적으로 연분수를 만드는 매크로를 작성해보자.

2.2 연분수 표기의 일반형

분수를 연분수로 변환하는 과정은 1.2절에서 살펴보았듯이, 매 단계마다 같은 과정을 반복한다. 즉, 분자를 분모로 나누어 몫을 구한 후 나머지가 0이 아니면 $1/\left(\frac{\text{분자}}{\text{나머지}}\right)$을 계속해서 연분수로 변환한다. 이처럼 연분수 변환 매크로는 반복 또는 재귀적으로 정의할 수 있으며, 아래와 같이 TEX 매크로를 이용해 간단하게 구현할 수 있다.

```
\def\generalcfrac#1#2{
  \mymod{#1}{#2} % 분자를 분모로 나눈 몫(\q)과 나머지(\r)
  \number\q      % 분자를 분모로 나눈 몫을 출력한다.
  \ifnum\r>0     % 나머지가 0이 아니면, 계속...
    +{\strut1\hfill\over\displaystyle\generalcfrac\n\r}\fi}
```

위에서 \mymod는 두 수를 인자로 입력받아 첫 번째 수를 두 번째 수로 나눈 몫(\q)과 나머지(\r)를 구하는 역할을 하며 코드는 아래와 같다.

```
\newcount\k \newcount\m \newcount\n
\newcount\r \newcount\q \newcount\t
\def\mymod#1#2{\m=#1 \n=#2 \t=0
  \ifnum\n=0
  \else\ifnum\n<0 \n=-\n \m=-\m \ifnum\m<0 \t=-1 \fi
    \else\ifnum\m<0 \t=-1\fi\fi \r=\m
    \divide\m by\n \q=\m \multiply\m by\n
    \ifnum\t<0 \ifnum\m=\r \else\advance\q by-1
    \t=\q \multiply\t by\n \m=\t \fi\fi
  \advance\r-\m \fi}
```

위의 매크로 정의에서, 정수를 저장하는 카운트 레지스터인 \m, \n, \r, \q 는 각각 분자(피제수), 분모(제수), 몫, 그리고 나머지를 나타낸다. 그리고 이 레지스터들은 연분수를 변환하는 매크로에서 프로그래밍 언어의 전역 변수처럼 사용된다.

2.3 연분수 표기의 리스트형

분수 $\frac{45}{16}$ 의 연분수는 앞에서와 같이 일반적인 형태뿐만 아니라, 다음과 같은 리스트 형태로도 나타낼 수 있다.

$$[2;1,4,3], \qquad //2,1,4,3//$$

리스트 형태는 위와 같이 두 가지로 표기하기로 하고, 이를 각각 \blcfrac와 \slcfrac 라고 하자. 이 두 매크로 역시 \generalcfrac과 같이 재귀적으로 정의할 수 있다. 이 매크로들은 표현을 리스트 형태로 한다는 것만 다르고, 개념적으로 \generalcfrac이 하는 일과 같은 일을 한다. 그러므로 이 두 매크로 역시 재귀적으로 정의된다.

```
\def\blcfrac#1#2{\advance\k by1 \mymod{#1}{#2}\number\q
  \ifnum\t>0 \ifnum\k=1 ;\else,\fi \blcfrac\n\t\fi}
\def\slcfrac#1#2{\mymod{#1}{#2}\number\q
  \ifnum\t>0 ,\slcfrac\n\t\fi}
```

2.4 옵션 인자를 이용한 매크로

분수를 연분수로 변환할 때, 일반형으로 변환할 것인지 리스트형으로 할 것인지를 매크로의 옵션 인자로 결정하기로 하자. 매크로의 모양은 다음과 같을 것이다.

```
\contfrac{45}{16}      % 일반형 표기
\contfrac[b]{45}{16}   % bracket을 이용한 리스트 표기
\contfrac[s]{45}{16}   % slash를 이용한 리스트 표기
```

즉, 옵션이 없으면 기본으로 일반형태를 사용하고, 옵션에는 대괄호(square bracket)을 나타내는 [b] 또는 사선기호(slash)를 나타내는 [s]를 쓸 수 있다.

옵션 인자를 사용하는 매크로는 TeX의 원시 명령어(primitive) \futurelet을 이용해 만들 수 있다. 자세한 내용은 [1, p. 107]을 참고하자.

```
\def\contfrac{\futurelet\optchar\optargcfrac}
\def\optargcfrac{\ifx[\optchar \let\next\listcfrac
  \else\let\next\generalcfrac\fi \next}
```

여기서 \listcfrac은 옵션에 따라서 \blcfrac 또는 \slcfrac을 호출하도록 다음과 같이 정의한다.

```
\def\listcfrac[#1]#2#3{\k=0 \if#1b [\blcfrac{#2}{#3}]
  \else\bslash\slcfrac{#2}{#3}\bslash\fi}
```

2.5 연분수에서 분수 만들기

지금까지와는 반대로 리스트 형태의 연분수를 원래의 분수로 바꾸는 매크로를 만들어보자. 이 매크로는 다음과 같이 사용되고

```
[3;1,4,15,9,2,6,5,3] = \fraction[3;1,4,15,9,2,6,5,3]
```

결과는 다음과 같다.

$$[3;1,4,15,9,2,6,5,3] \quad = \quad \frac{589291}{154970}$$

매크로 \fraction의 특징은 리스트 형태의 연분수를 입력받아서 처리하는데, 리스트의 원소 수를 미리 알 수 없다는 것이다. 인자의 수가 정해지지 않은 매크로를 다루는 방법은 [2, p. 219]와 [1, p. 105]에 나와있는데, 여기에서는 그 방법을 좀 더 개선시킨 van der Laan[3]의 \lifo 및 \ofil 방법을 이용한다.

```
\def\reverse#1,#2\esrever{\ifx\empty#2\empty\esrever\fi
  \reverse#2\esrever\makefrac#1,}
\def\esrever#1\esrever{\fi}
\def\makefrac#1,{\reciprocal
  \r=\m \multiply\r by#1 \advance\r by\n \n=\r}
\def\fraction[#1;#2]{\n=1 \m=0 \reverse#1,#2,\esrever
  \ifnum\m=1 \number\n \else{\number\n\over\number\m}\fi}
\def\reciprocal{\r=\n \n=\m \m=\r}
```

3 연분수와 유클리드 알고리즘

최대공약수를 구하는 유클리드 알고리즘은 가장 널리 알려진 알고리즘 중의 하나이다. 이 알고리즘은 연분수와 상당히 밀접한 관계가 있다. 45와 16의 최대공약수를 구하는 유클리드 알고리즘을 단계별로 살펴보자.

$$\begin{array}{rclrcrcrl} 45 & = & \mathbf{2} & \times & 16 & + & 13 & & \text{(45를 16으로 나누면 13이 남는다.)} \\ 16 & = & \mathbf{1} & \times & 13 & + & 3 & & \text{(16을 13으로 나누면 3이 남는다.)} \\ 13 & = & \mathbf{4} & \times & 3 & + & 1 & & \text{(13을 3으로 나누면 1이 남는다.)} \\ 3 & = & \mathbf{3} & \times & 1 & + & 0 & & \text{(3은 1의 배수이다.)} \end{array}$$

위의 식에서 굵은 글씨로 표시된 숫자들 $2, 1, 4, 3$은 $\frac{45}{16}$를 리스트 형태로 표시한 연분수 $[2; 1, 4, 3]$에 열거된 수들과 동일하다. 따라서 주어진 두 수의 최대공약수를 구하는 매크로 역시 연분수 매크로와 상당히 유사하게 구현할 수 있다. 연분수를 구하는 매크로 `\generalcfrac`과 비교해 보기 바란다.

```
\newcount\a \newcount\b
\def\euclid#1#2{\a=#1 \b=#2\unskip
  \ifnum\b>\a \k=\a \a=\b \b=\k\fi
  \ifnum\b=0 \ \number\a\else\mymod\a\b \euclid\b\r\fi}
```

예를 들어, 두 수 119와 544의 최대공약수는 `\euclid{119}{544}` 명령으로 구할 수 있고 그 값은 17이다.

4 몇 가지 예제들

분수를 연분수로 변환하는 `\contfrac` 매크로는 순서대로 분자와 분모, 두 개의 인자를 가지는데 각 인자에 양의 정수뿐만 아니라 음의 정수도 사용할 수 있다. 예를 들어 양의 분수 $\frac{45}{16} = \frac{-45}{-16}$와 음의 분수 $-\frac{45}{16}$는 다음과 같이 구할 수 있으며

```
\contfrac{45}{16}, \contfrac{45}{-16},
\contfrac{-45}{-16}, \contfrac{-45}{16} \par
\contfrac[b]{45}{16}, \contfrac[s]{45}{-16},
\contfrac[b]{-45}{-16}, \contfrac[s]{-45}{16}
```

그 결과는 아래와 같다.

$$2+\cfrac{1}{1+\cfrac{1}{4+\cfrac{1}{3}}}, \qquad -3+\cfrac{1}{5+\cfrac{1}{3}}, \qquad -3+\cfrac{1}{5+\cfrac{1}{3}}, \qquad 2+\cfrac{1}{1+\cfrac{1}{4+\cfrac{1}{3}}}$$

$$[2; 1, 4, 3], \qquad /\!/ -3, 5, 3/\!/, \qquad [-3; 5, 3], \qquad /\!/2, 1, 4, 3/\!/$$

이번에는 대표적인 무리수, 원주율 π를 연분수로 변환하고 이를 다시 분수로 바꾸어 보자. π의 연분수 근사치는 다음과 같다.

$$\begin{aligned}\pi &= [3; 7, 15, 1, 292, 1, 1, 1, 2, 1, 3, 1, 14, 2, 1, 1, 2, 2, 2, 2, 1, 84, 2, 1, 1, 15, 3, 13, 1, 4, 2, 6,\\ &\quad 6, 99, 1, 2, 2, 6, 3, 5, 1, 1, 6, 8, 1, 7, 1, 2, 3, 7, 1, 2, 1, 1, 12, 1, 1, 1, 3, 1, 1, 8, 1, 1, 2, 1, 6,\\ &\quad 1, 1, 5, 2, 2, 3, 1, 2, 4, 4, 16, 1, 161, 45, 1, 22, 1, 2, 2, 1, 4, 1, 2, 24, 1, 2, 1, 3, 1, 2, 1]\end{aligned}$$

연분수를 분수로 바꾸는 `\fraction` 매크로에 위 리스트를 입력하면, 어느 순간 분자와 분모의 값이 TEX이 다룰 수 있는 한계인 $2^{31} - 1$을 벗어난다. 이 리스트를 TEX이 다룰 수 있는 크기까지 아래와 같이 입력하면

```
\fraction[3;7,15,1,292,1,1,1,2,1,3,1,14,2,1,1,2]
```

분수

$$\frac{1068966896}{340262731}$$

을 얻는다. 이 분수를 다시 연분수로 변환해서 위의 리스트와 동일한지 확인해보자.

```
\contfrac{1068966896}{340262731}
\contfrac[b]{1068966896}{340262731}
\contfrac[s]{1068966896}{340262731}
```

위 명령어들의 결과는 다음과 같다.

$$3+\cfrac{1}{7+\cfrac{1}{15+\cfrac{1}{1+\cfrac{1}{292+\cfrac{1}{1+\cfrac{1}{1+\cfrac{1}{1+\cfrac{1}{2+\cfrac{1}{1+\cfrac{1}{3+\cfrac{1}{1+\cfrac{1}{14+\cfrac{1}{2+\cfrac{1}{1+\cfrac{1}{1+\frac{1}{2}}}}}}}}}}}}}}}$$

$$[3; 7, 15, 1, 292, 1, 1, 1, 2, 1, 3, 1, 14, 2, 1, 1, 2]$$

$$/\!/3, 7, 15, 1, 292, 1, 1, 1, 2, 1, 3, 1, 14, 2, 1, 1, 2/\!/$$

참고 문헌

1. Victor Eijkhout, *TEX by Topic, A TEXnician's Reference*, Addison-Wesley, 1992. http://www.cs.utk.edu/~eijkhout/texbytopic-a4.pdf
2. Donald E. Knuth, *The TEX book*, Addison-Wesley, 1986.
3. Kees van der Laan, *FIFO and LIFO sing the BLUes*, TUGboat **14** (1993), no. 1, 54–60.
4. Continued fraction — Wikipedia, the free encyclopedia. http://en.wikipedia.org/wiki/Continued_fraction

cfrac.tex 소스 코드

```
%% 연분수 매크로에 사용되는 유틸리티 매크로
\newcount\k \newcount\m \newcount\n
\newcount\r \newcount\q \newcount\t

\def\bslash{/\mkern-4.5mu/}
\def\reciprocal{\r=\n \n=\m \m=\r}
\def\mymod#1#2{\m=#1 \n=#2 \t=0
  \ifnum\n=0 \else\ifnum\n<0 \n=-\n \m=-\m \ifnum\m<0 \t=-1 \fi
    \else\ifnum\m<0 \t=-1\fi\fi \r=\m
    \divide\m by\n \q=\m \multiply\m by\n
    \ifnum\t<0 \ifnum\m=\r \else\advance\q by-1
      \t=\q \multiply\t by\n \m=\t \fi\fi\advance\r-\m \fi}

%% 연분수의 일반형과 리스형 표기법을 구현한 매크로
\def\contfrac{\futurelet\optchar\optargcfrac}
\def\optargcfrac{\ifx[\optchar \let\next\listcfrac
  \else\let\next\generalcfrac\fi \next}
\def\listcfrac[#1]#2#3{\k=0 \if#1b [\blcfrac{#2}{#3}]
  \else\bslash\slcfrac{#2}{#3}\bslash\fi}
\def\blcfrac#1#2{\advance\k by1 \mymod{#1}{#2}\number\q
  \ifnum\r>0 \ifnum\k=1;\else,\fi\blcfrac\n\r\fi}
\def\slcfrac#1#2{\mymod{#1}{#2}\number\q
  \ifnum\r>0,\slcfrac\n\r \fi}
\def\generalcfrac#1#2{\mymod{#1}{#2} \number\q
  \ifnum\r>0+{\strut1\hfill\over\displaystyle\generalcfrac\n\r}\fi}

%% 리스트형 표기법의 연분수에서 분수를 구하는 매크로
\def\reverse#1,#2\esrever{\ifx\empty#2\empty\esrever\fi
  \reverse#2\esrever\makefrac#1,}
\def\esrever#1\esrever{\fi}
\def\makefrac#1,{\reciprocal
  \r=\m \multiply\r by#1 \advance\r by\n \n=\r}
\def\fraction[#1;#2]{\n=1 \m=0 \reverse#1,#2,\esrever
  \ifnum\m=1 \number\n \else{\number\n\over\number\m}\fi}

%% 두 수의 최대공약수를 구하는 유클리드 알고리즘 매크로
\newcount\a \newcount\b
\def\euclid#1#2{\a=#1 \b=#2\unskip
  \ifnum\b>\a \k=\a \a=\b \b=\k\fi
  \ifnum\b=0 \ \number\a\else\mymod\a\b \euclid\b\r\fi}

\endinput
```