

한국어 문장 부호를 조판하는 kopunct 패키지*

강우현

kangwh.2003@gmail.com

2025년 5월 1일

요 약

kopunct 패키지는 L^AT_EX에서 한국어 문장 부호를 올바르게 조판하는 명령어를 제공한다. 이 패키지를 불러오면 한국어 키보드에 등장하지 않는 가운뎃점·낫표·화살괄호·줄임표와 다양한 형태의 괄호를 쉽게 입력할 수 있으며, 레거시 엔진에서는 더 보기 좋은 물결표를 얻을 수 있다.

차 례

1 들어가며	2
2 불러오기	2
3 기본 사용 방법	3
3.1 가운뎃점과 줄임표	3
3.2 낫표와 화살괄호	4
3.3 추가 괄호	4
4 물결표	5
4.1 물결표 세부 조정	6
4.2 다른 기호를 사용하여 물결표 조판	6
5 oblivoir 호환 모드	6
6 패키지 옵션	6
6.1 재정의 방지 옵션	7
6.2 oblivoir 클래스 호환 옵션	7

*이 문서는 2025년 1월 23일에 작성된 kopunct 패키지 버전 1.2 에 대응한다.

7 구현	8
7.1 패키지 정보	8
7.2 옵션 선언	8
7.3 패키지 불러오기	8
7.4 매크로 정의	9
7.4.1 가운데점	9
7.4.2 줄임표	9
7.4.3 낫표와 화살괄호	10
7.4.4 추가 괄호	11
7.4.5 물결표 재정의	12
7.5 oblivoir 호환 매크로 정의	13
7.5.1 가운데점	13
7.5.2 줄임표	13
7.5.3 낫표와 화살괄호	14
7.5.4 줄표	14

1 들어가며

국립국어원의 「한글 맞춤법」 부록에는 여러 가지 문장 부호가 정의되어 있다. 그러나 한글 키보드에 있는 13개¹를 제외하면 입력의 불편함 때문에 잘 사용되지 않는다. 문자 메시지나 온라인 커뮤니티의 게시글 등 비공적인 글에는 문장 부호를 엄격히 지켜 사용할 필요가 없지만, 이러한 습관이 공적인 글을 쓸 때에도 이어져 공문이나 논문, 책 등을 쓸 때조차도 문장 부호를 잘못 사용하는 일이 흔하다.

이 문제를 해결하기 위해서는 문장 부호에 대한 교육과 함께 한글 키보드 배열이 개선되어야 하지만, 사회적 비용을 고려하였을 때 한글 키보드 배열이 바뀔 일은 없으리라 본다. 따라서 필자는 L^AT_EX에서 한글 문장 부호의 입력을 보조하기 위해 이 패키지를 작성하였다.

2 불러오기

다른 패키지들과 마찬가지로 전처리부에

```
\usepackage{kopunct}
```

를 입력하여 패키지를 불러오면 된다.

유의 사항에는 다음과 같은 것들이 있다.

¹마침표, 물음표, 느낌표, 쉼표, 쌍점, 빗금, 큰따옴표, 작은따옴표, 소괄호, 중괄호, 대괄호, 붙임표, 물결표. T_EX 환경에서는 ‘---’라는 합자 기능 덕분에 그나마 줄표까지는 잘 사용된다.

- `ko.TeX`이나 `polyglossia` 등 한국어 조판과 관련된 패키지를 불러온 후에 이를 불러오는 것이 좋다.
- 레거시 엔진에서는 물결표를 조정하는 데 `graphicx` 패키지를 사용하므로, 이 패키지를 불러와야 한다. `kopunct`는 `graphicx`를 자동으로 불러오지 않는다. 패키지를 불러오는 순서는 중요하지 않다.
- 레거시 엔진에서는 T1 인코딩을 사용해야 한다. 이 인코딩을 사용하지 않는 경우 가운데점이 조판되지 않는다.

3 기본 사용 방법

이 패키지에서 제공하는 문장 부호 조판 명령어를 설명한다. 이번 절에서 소개하는 명령어들은 `oblivoironly` 옵션을 지정한 경우 사용할 수 없다.

3.1 가운데점과 줄임표

`\cdot` 가운데점은 `\cdot`으로, 3점 줄임표는 `\cdots`로 입력한다. 현재 모드에 상관없이 `\cdots` 일한 명령어로 동일한 모양의 기호를 조판할 수 있다. 6점 줄임표는 `\cdots`를 두 번 `\Cdots` 입력해도 되지만, 간편하게 `\Cdots`로도 입력할 수 있다.

예시

아침`\cdot` 점심`\cdot` 저녁 → 아침·점심·저녁

이`\cdots` 이럴 수가`\Cdots`. → 이...이럴 수가.....

`\(1 \cdot 2 \cdots 5 = 120\)` → $1 \cdot 2 \cdots 5 = 120$.

`\hellipsis` 유니코드 엔진의 `ko.TeX`에서는 6점 줄임표를 조판하는 명령어로 `\hellipsis`를 제공하는데,² `kopunct`는 이 명령어가 정의되어 있지 않은 경우에 한하여 `\hellipsis`를 정의해 준다.

예시

이럴 수가`\hellipsis`. → 이럴 수가.....

`\dots` 한편, 유니코드 엔진에서 `\dots`와 `\ldots`는 레거시 엔진의 줄임표(...)와 다른 `\ldots` 모양의 줄임표를 조판한다. 이 패키지는 엔진에 관계없이 `\dots`와 `\ldots`가 동일한 모양의 줄임표(영어 등에서 사용하는 모양)를 조판하도록 한다.

예시

My brown fox is `\dots` gone! → My brown fox is ...gone!

²어떤 이유에서인지 `LuaTeX-ko`나 `XgTeX-ko`의 설명서에는 이 사실이 나와 있지 않다.

3.2 낫표와 화살괄호

`\nat` 홑낫표는 `\nat`으로 입력한다. 인자를 입력하면 그 내용이 홑낫표에 싸인 채로 조판
`\Nat` 된다. ‘n’을 대문자 ‘N’으로 바꾸면 겹낫표가 조판된다.

예시

`\nat{홑낫표}` → 「홑낫표」

`\Nat{겹낫표}` → 『겹낫표』

`\hwa` 화살괄호는 `\hwa`와 `\Hwa`로 입력한다. 앞과 마찬가지로 소문자 `\hwa`는 홑화살괄
`\Hwa` 호, 대문자 `\Hwa`는 겹화살괄호를 조판한다.

예시

`\hwa{홑화살괄호}` → 〈홑화살괄호〉

`\Hwa{겹화살괄호}` → 《겹화살괄호》

`\lnat` 여는 쪽만 입력하려면 ‘l’을, 닫는 쪽만 입력하려면 ‘r’를 역슬래시 뒤에 입력하면
`\rnat` 된다. 이때에는 인자를 지정하지 않는다.

`\lNat`

예시

`\nNat`

`\lhwa`

`\rhwa`

`\lHwa`

`\nHwa`

`\lnat` 홑낫표 `\rnat` → 「홑낫표」

`\lNat` 겹낫표 `\rNat` → 『겹낫표』

`\lhwa` 홑화살괄호 `\rhwa` → 〈홑화살괄호〉

`\lHwa` 겹화살괄호 `\rHwa` → 《겹화살괄호》

3.3 추가 괄호

2014년 이전의 문장 부호 규정에서는 대괄호의 자형이 ‘[]’이 아닌 ‘〔 〕’였다. 그러나
‘〔 〕’보다 ‘[]’가 더 자주 쓰인다는 이유로 2014년 개정에서 대괄호의 기본 자형은
‘[]’가 되었다. 그렇다고 해서 ‘〔 〕’를 대괄호로 사용하지 못하는 것은 아니라고 적혀
있으므로, 이 패키지에서 ‘〔 〕’를 입력하는 명령어를 제공하지 않을 이유도 없다. 여
기에 유니코드의 ‘CJK Symbols and Punctuations(한중일 기호 및 구두점)’ 영역에
있는 다양한 괄호들을 입력하는 명령어도 함께 제공한다.

낫표와 화살괄호처럼, 다음에서 설명하는 명령어의 역슬래시 뒤에 ‘l’을 덧붙여 왼
쪽 괄호만을, ‘r’을 덧붙여 오른쪽 괄호만을 조판할 수 있다. 괄호의 이름은 그 모양에
따라 필자가 임의로 붙인 것이다.

`\tort` 사다리꼴 괄호 `[]`는 `\tort`로 입력한다.

`\ltort`

예시

`\rtort`

오늘 `\tort{5.1.(목)}`은 `\cdots` → 오늘(5.1.(목))은...

`\Tort` 첫 글자가 대문자인 `\Tort`를 사용하면 겹사다리꼴 괄호 $\left[\right]$ 가 조판된다. 많은 한글 `\lTort` 글꼴은 이 기호를 표시할 수 없으므로, 이 명령어를 사용하는 경우 기호가 잘 조판되지 있는지 확인하기를 권장한다.³

`\lent` 렌즈꼴 괄호 $\llbracket \rrbracket$ 는 `\lent`로 입력한다.

`\llent` 예시

`\rlent` `\lent{긴급}` 급여 변경 안내 → \llbracket 긴급 \rrbracket 급여 변경 안내

`\Lent` 첫 글자가 대문자인 `\Lent`를 사용하면 겹렌즈꼴 괄호 $\Llbracket \Rlbracket$ 가 조판된다. `\Tort`와 마찬가지로, 글꼴에 따라 이 기호가 조판되지 않을 수 있으므로 조판 후 검토를 권장한다.

`\rLent` 마지막으로, `\Brac`을 사용하면 겹대괄호 $\llbracket \rrbracket$ 가 조판된다. `\Lent`나 `\Tort`와 마찬가지로, 글꼴에 따라 이 기호가 조판되지 않을 수 있다.⁴

`\lBrac`

`\rBrac` 4 물결표

(L^A)T_EX에서 물결표는 `\textasciitilde` 또는 `\~{}`로 조판한다. 유니코드 엔진에서는 우리가 생각하는 모양의 물결표가 그려지지만, 레거시 엔진에서는 악센트로 쓰이는 형태의 물결표가 그려진다. 비슷한 모양의 수학 기호를 조판하는 `\sim`을 대안으로 제시하는 이도 있지만, 이 기호를 사용하면 문장부호만 도드라져 보인다. 이는 ‘실제’ 물결표를 사용한 것도 아니므로 검색이나 복사 기능도 제대로 작동하지 않는다.

예시

`2010\~{}`2015 → 2010~2015

`381\(\sim\)`403쪽 → 381~403쪽

`\~` `kopunct`는 매크로를 재정의함으로써 레거시 엔진에서도 물결표가 바른 모양으로 그려지도록 한다. 패키지를 불러온 뒤 `\~{}`를 입력하면 된다. `\~`의 틸데 악센트를 조판하는 기능도 정상적으로 작동한다. 만약 — 어떤 이유에서든 — 틸데 악센트를 단독으로 출력해야 한다면 `\textasciitilde`를 입력하면 된다.⁵

예시

`제1\~{}`2장 → 제1~2장

`Espa\~no1` → Español

틸데 악센트(`\textasciitilde`) → 틸데 악센트(`\~`)

³레거시 엔진에서는 기본 한글 산세리프·고정폭 글꼴(나눔고딕)이 이 기호의 출력을 지원한다. 유니코드 엔진에서 사용할 수 있는 대표 글꼴에는 본명조·본고딕·본모노 시리즈가 있다. 이 내용은 `\Lent`와 `\Brac`에도 동일하게 적용된다.

⁴이 기호는 레거시 엔진의 나눔고딕도 지원하지 않는다.

⁵OT1 인코딩을 사용하는 경우에는 `\textasciitilde`를 사용하더라도 물결표가 출력된다. 이때에는 `\symbol{“}~`를 입력하면 틸데 악센트를 얻을 수 있다.

4.1 물결표 세부 조정

kopunct 패키지는 T_EX의 기본 로마자 글꼴인 Computer Modern Roman에 맞추어 틸데 악센트가 물결표처럼 보이도록 조정해 두었다. 따라서 기본 글꼴이 바뀐다면 물결표가 어울리지 않을 수 있다. 사용자는 내부 변수를 수정함으로써 현재 글꼴에 어울리도록 물결표의 크기와 위치를 조정할 수 있다.

`\TildeScaleFactor` `\TildeScaleFactor`는 틸데를 확대할 배율로, 기본값은 1.5이다. `\TildeRaise`
`\TildeRaise` `\TildeRaise`는 틸데의 상하 이동 값으로, 기본값은 -1.35ex 이다(음수이면 틸데가 아래로 내려간다). `\TildeHeight`는 물결표 문자의 높이 값으로, 기본값은 1ex 이다. 이때 `\TildeRaise`와 `\TildeHeight`의 단위는 ex 또는 em으로 지정해야 글꼴의 크기 변화에 대응할 수 있다.

4.2 다른 기호를 사용하여 물결표 조판

유니코드 문자표를 찾아 보면 키보드로 바로 입력할 수 있는 물결표(Tilde, U+007E) 외에도, 한중일 문장 부호용 물결표(Wave Dash, U+301C)와 전각 물결표(Fullwidth Tilde, U+FF5E)가 있다.⁶ 패키지를 불러올 때 `cjktilde`를 옵션으로 지정하면 `\~{}`를 입력했을 때 (컴파일 엔진에 관계없이) 한중일 문장 부호용 물결표가 조판되며, `fullwidthtilde`를 옵션으로 지정하면 `\~{}`를 입력했을 때 전각 물결표가 조판된다.⁷

문자 코드 등에 특별히 잘 알고 있거나, 물결표를 조판하는 데 특정 기호를 반드시 사용해야 하는 것이 아니라면 두 옵션은 불필요하다.

5 oblivoir 호환 모드

oblivoir 클래스는 한국어 문장 부호를 조판하는 명령어를 기본 제공한다. 패키지를 불러올 때 `oblivoir` 옵션을 지정하면 기본 명령어와 `oblivoir` 클래스의 명령어를 함께 사용할 수 있으며, `oblivoironly` 옵션을 지정하면 기본 명령어는 사용할 수 없고 `oblivoir`의 명령어만을 사용할 수 있다. 두 종류의 명령어의 대조표는 표 1에서 확인할 수 있다.

6 패키지 옵션

현재 버전의 kopunct는 `nocdot`, `nocdots`, `notilde`, `cjktilde`, `fullwidthtilde`, `oblivoir`, `oblivoironly` 이렇게 일곱 개의 옵션을 제공한다. 앞서 설명한 `cjktilde`와 `fullwidthtilde`를 제외하고, 나머지 옵션 다섯 개의 설명을 아래에 제시한다.

⁶비슷한 모양의 기호로 Tilde Operator(U+223C)도 있지만, 이는 말 그대로 수식에 적합한 기호이다.

⁷현재 사용하는 글꼴이 한중일 문장 부호용 물결표와 전각 물결표에 동일한 글리프를 사용한다면 컴파일 후 생성된 PDF 파일의 문자는 사용한 옵션에 관계없이 어느 한쪽으로도만 검색될 수 있다.

표 1: kopunct 기본 명령어와 oblivoir 호환 명령어 대조표

문장 부호	kopunct 기본 명령어	oblivoir 호환 명령어
가운뎃점	<code>\cdot</code>	<code>\cntrdot</code>
3점 줄임표(한국어)	<code>\cdots</code>	<code>\cntrdots</code>
3점 줄임표(영어)	<code>\ldots</code> 또는 <code>\dots</code>	<code>\obl dots</code>
6점 줄임표	<code>\Cdots</code> 또는 <code>\hellipsis</code>	<code>\obellipsis</code>
홀낫표	<code>\nat{...}</code> 또는 <code>\lnat ...\rnat</code>	<code>\snm{...}</code>
겹낫표	<code>\Nat{...}</code> 또는 <code>\lNat ...\rNat</code>	<code>\bnm{...}</code>
홀화살괄호	<code>\hwa{...}</code> 또는 <code>\lhwa ...\rhwa</code>	<code>\cnm{...}</code>
겹화살괄호	<code>\Hwa{...}</code> 또는 <code>\lHwa ...\rHwa</code>	<code>\ccnm{...}</code>

6.1 재정의 방지 옵션

kopunct는 `\cdot`, `\cdots`, `\textasciitilde`를 재정의함으로써 문장 부호를 조판하는 명령어를 제공한다. 이 과정에서 예기치 못한 문제가 생길 수 있기 때문에, kopunct는 이들을 재정의하지 않도록 지시하는 옵션을 제공한다.

`nocdot`은 `\cdot`을 재정의하지 않도록 한다. 따라서 가운뎃점은 소스 코드에 직접 입력하거나 `\textperiodcentered`를 사용하여 입력해야 한다.

`nocdots`는 `\cdots`를 재정의하지 않도록 한다. 이에 따라 `\Cdots`도 정의되지 않는다. 필요한 경우 레거시 엔진에서는 `\unichar{"2026}`을 입력하여, 유니코드 엔진에서는 `\textellipsis`나 `\dots`, `\ldots`를 사용하여 3점 줄임표를 입력할 수 있다. 유니코드 엔진에서는 줄임표 문자를 소스 파일에 직접 입력해도 된다.

`notilde`는 레거시 엔진에서 `\~`를 재정의하지 않도록 한다. 유니코드 엔진에서는 아무런 효과가 없다.

6.2 oblivoir 클래스 호환 옵션

oblivoir 클래스는 한국어 문장 부호를 조판하는 자체적인 매크로를 제공한다. kopunct는 이 클래스와의 호환 기능을 제공한다.

oblivoir 옵션은 kopunct의 기본 명령어와 함께 oblivoir의 명령어를 사용할 수 있도록 한다. 반면, `oblivoironly` 옵션은 kopunct의 기본 명령어를 정의하지 않고 oblivoir의 문장 부호 명령어만을 정의한다. 호환 명령어는 본래의 클래스와 일부 다르게 동작할 수 있다.

`oblivoironly`를 지정한 경우, `\cdot`, `\cdots`, `\~`가 모두 재정의되지 않으므로 `nocdot`, `nocdots`, `notilde`는 아무 효과를 내지 않는다.

7 구현

7.1 패키지 정보

패키지의 정보를 제공한다.

```
1 \NeedsTeXFormat{LaTeX2e}
2 \ProvidesPackage{kopunct}[2025/05/01 v1.2
3   Package for providing macros to input Korean punctuation marks]
```

7.2 옵션 선언

먼저 옵션을 처리하기 위한 스위치를 정의한다. 정의한 각 스위치는 `false`로 초기화한다.

```
4 \newif\ifkopunct@nocdot\kopunct@nocdotfalse
5 \newif\ifkopunct@nocdots\kopunct@nocdotsfalse
6 \newif\ifkopunct@oblivoir\kopunct@oblivoirfalse
7 \newif\ifkopunct@oblivoironly\kopunct@oblivoironlyfalse
```

물결표 관련 옵션은 0부터 3까지의 정수 값을 가진다. 기본값은 1이다.

```
8 \def\kopunct@tildestatus{1}
```

`\kopunct@tildestatus`가 0으로 설정되면 `\~{}`를 재정의하지 않음을 의미한다. 1로 설정되면 U+007E로 물결표를 조판하며, 레거시 엔진에서는 이 기호의 크기와 위치를 조정하여 보기 좋게 한다. 2로 설정되면 컴파일 엔진에 관계없이 U+301C로 물결표를 조판하며, 3으로 설정되면 컴파일 엔진에 관계없이 U+FF5E로 조판한다. 이후 옵션을 선언한다.

```
9 \DeclareOption{nocdot}{\kopunct@nocdottrue}
10 \DeclareOption{nocdots}{\kopunct@nocdotstrue}
11 \DeclareOption{notilde}{\def\kopunct@tildestatus{0}}
12 \DeclareOption{cjktilde}{\def\kopunct@tildestatus{2}}
13 \DeclareOption{fullwidthtilde}{\def\kopunct@tildestatus{3}}
14 \DeclareOption{oblivoir}{\kopunct@oblivoirtrue}
15 \DeclareOption{oblivoironly}{%
16   \kopunct@oblivoirtrue
17   \kopunct@oblivoironlytrue
18 }
```

옵션 선언이 완료되었으면 옵션을 처리한다.

```
19 \ProcessOptions\relax
```

7.3 패키지 불러오기

레거시 엔진과 유니코드 엔진을 구분하기 위해 `iftex` 패키지를 불러온다.


```
20 \RequirePackage{iftex}
```

이 패키지는 줄임표를 정의할 때와 물결표를 재정의할 때 사용된다.

7.4 매크로 정의

기본 매크로는 `\ifkopunct@oblivioironly` 스위치가 `false`일 때에만 정의된다.

```
21 \ifkopunct@oblivioironly\else
```

7.4.1 가운데점

`\cdot` 텍스트 모드에서는 가운데점을, 수식 모드에서는 내적 기호를 조판하도록 `\cdot`을 재정의한다. 이 작업은 `\ifkopunct@nocdot` 스위치가 `false`일 때에만 진행된다.

```
22 \ifkopunct@nocdot\else
```

`\math@cdot` `\cdot`의 정의에는 `\math@cdot`과 `\text@cdot`이 필요하다. `\math@cdot`은 수식 모드에서 내적 기호를 조판할 때 실제로 사용할 제어 문자열, `\text@cdot`은 텍스트 모드에서 가운데점을 조판할 때 실제로 사용할 제어 문자열이다. 각각 기존의 `\cdot`과 `\textperiodcentered`의 정의를 복사하면 된다.

```
23 \NewCommandCopy{\math@cdot}{\cdot}
```

```
24 \NewCommandCopy{\text@cdot}{\textperiodcentered}
```

두 제어 문자열을 정의했다면 `\cdot`을 재정의한 뒤 스위치를 종료한다.

```
25 \protected\def\cdot{\ifmmode\math@cdot\else\text@cdot\fi}
```

```
26 \fi
```

7.4.2 줄임표

`\cdots` 텍스트 모드에서는 줄임표를, 수식 모드에서는 점이 가운데에 찍히는 줄임표를 조판하도록 `\cdots`를 재정의한다. 이 작업은 `\ifkopunct@nocdots` 스위치가 `true`일 때에만 진행된다.

```
27 \ifkopunct@nocdots\else
```

`\math@cdots` `\cdot`과 마찬가지로, `\cdots`를 정의할 때에도 `\math@cdots`와 `\text@cdots`가 필요하다. `\math@cdots`는 간단히 `\cdots`의 기존 정의를 복사하면 된다.

```
28 \NewCommandCopy{\math@cdots}{\cdots}
```

`\text@cdots` `\text@cdots`는 엔진에 따라 정의 방법이 다르다. 먼저 유니코드 엔진에서는 간단히 줄임표 문자(`⋯`, U+2026)를 `\text@cdots`로 정의하면 된다.

```
29 \iftutex
```

```
30 \def\text@cdots{...}
```

레거시 엔진에서는 `\unichar`과 `\DeclareUnicodeCharacter`를 사용하여 다음과 같이 정의한다. 줄임표 문자를 소스 파일에 직접 입력하더라도 줄임표가 올바른 모양으로 조판된다.

```
31 \else
32 \DeclareTextCommandDefault{\text@cdots}{\unichar{"2026}}
33 \DeclareUnicodeCharacter{2026}{\unichar{"2026}}
34 \fi
```

두 제어 문자열을 정의했다면 `\cdots`를 재정의한다.

```
35 \protected\def\cdots{\ifmmode\math@cdots\else\text@cdots\fi}
```

`\Cdots` 6점 줄임표는 `\Cdots`로 정의한다.

```
36 \protected\def\Cdots{%
37 \ifmmode\math@cdots\math@cdots\else\text@cdots\text@cdots\fi
38 }
```

`\hellipsis` `\hellipsis`가 정의되어 있지 않다면 이를 정의한다. 이는 `XYTeX-ko` 및 `LuaTeX-ko`와의 호환을 위한 것이다.

```
39 \@ifundefined{hellipsis}{%
40 \def\hellipsis{.....}%
41 }
```

`\textellipsis` 유니코드 엔진에서도 `\dots`나 `\ldots`, `\textellipsis`가 레거시 엔진의 줄임표와 동일한 기호를 조판하도록 재정의한다.

```
42 \iftutex
43 \UndeclareTextCommand{\textellipsis}{TU}
44 \fi
```

줄임표와 관련된 모든 매크로 정의가 완료되었으면 스위치를 종료한다.

```
45 \fi
```

7.4.3 낫표와 화살괄호

`\lnat` 왼쪽 홑낫표는 `\lnat`으로, 오른쪽 홑낫표는 `\rnat`으로 조판한다.

```
\rnat 46 \newcommand*\lnat{「}\newcommand*\rnat{」}%
```

`\nat` 이를 사용하여 인자를 홑낫표로 감싸는 `\nat`을 정의한다.

```
47 \newcommand*\nat[1]{\{\lnat{#1}\rnat}}%
```

`\lNat` 겹낫표를 조판하는 `\lNat`, `\rNat`, `\Nat`도 동일한 방법으로 정의한다.

```
\rNat 48 \newcommand*\lNat{『}\newcommand*\rNat{』}%
```

```
\Nat 49 \newcommand*\Nat[1]{\{\lNat{#1}\rNat}}%
```

`\lhwa` 흘화살괄호는 `\lhwa`, `\rhwa`, `\hwa`로 입력한다.

```

\rhwa 50 \newcommand*\lhwa{<}\newcommand*\rhwa{>}%
\hwa 51 \newcommand*\hwa[1]{\{\lhwa{#1}\rhwa}\}%

```

`\lHwa` 겹화살괄호는 `\lHwa`, `\rHwa`, `\Hwa`로 입력한다.

```

\rHwa 52 \newcommand*\lHwa{<>}\newcommand*\rHwa{>}\newcommand*\Hwa[1]{\{\lHwa{#1}\rHwa}\}%
\Hwa 53 \newcommand*\Hwa[1]{\{\lHwa{#1}\rHwa}\}%

```

7.4.4 추가 괄호

괄호류 매크로들도 위와 동일한 방법으로 정의된다. 매크로의 이름은 유니코드 문자 표에 표기된 이름을 바탕으로 정하였다.

`\ltort` 사다리꼴 괄호(tortoise shell bracket, `()`)는 `\ltort`, `\rtort`, `\tort`로 입력한다.

```

\rtort 54 \newcommand*\ltort{({}\newcommand*\rtort{)}}%
\tort 55 \newcommand*\tort[1]{\{\ltort{#1}\rtort}\}%

```

`\lTort` `tort`의 첫 번째 `t`를 대문자로 입력하면 겹사다리꼴 괄호(white tortoise shell bracket, `()`)가 조판된다.

```

\Tort 56 \newcommand*\lTort{({}\newcommand*\rTort{)}}%
57 \newcommand*\Tort[1]{\{\lTort{#1}\rTort}\}%

```

`\llent` 렌즈꼴 괄호(lenticular bracket, `[]`)는 `\llent`, `\rlent`, `\lent`로 입력한다.

```

\rlent 58 \newcommand*\llent{[{}\newcommand*\rlent{]}}%
\lent 59 \newcommand*\lent[1]{\{\llent{#1}\rlent}\}%

```

`\lLent` 겹렌즈꼴 괄호(white lenticular bracket, `[]`)는 `\lLent`, `\rLent`, `\Lent`로 입력한다.

```

\rLent 60 \newcommand*\lLent{[{}\newcommand*\rLent{]}}%
\Lent 61 \newcommand*\Lent[1]{\{\lLent{#1}\rLent}\}%

```

`\lBrac` 겹대괄호(white square bracket, `[]`)는 `\lBrac`, `\rBrac`, `\Brac`으로 입력한다. \LaTeX

`\rBrac` 과 \LaTeX-utf 의 기본 글꼴의 한계로 아래 코드에는 괄호가 공백으로 표시되지만, 소 `\Brac` 스 파일을 열어 보면 괄호를 확인할 수 있을 것이다.

```

62 \newcommand*\lBrac{[}\newcommand*\rBrac{]}\%
63 \newcommand*\Brac[1]{\{\lBrac{#1}\rBrac}\}%

```

매크로 정의가 완료되었으면 스위치를 종료한다.

```

64 \fi

```

7.4.5 물결표 재정의

물결표는 `\kopunct@tildestatus`의 값에 따라 재정의 방식과 여부가 결정된다. 먼저 값이 0일 때에는 재정의를 하지 않는다.

```
65 \ifcase\kopunct@tildestatus
66 \or
```

값이 1인 경우, 레거시 엔진에서는 물결표를 더 보기 좋게 조판하기 위해 `\textasciitilde`를 재정의한다.

```
67 \iftutex\else
```

`\kopunct@tilde` `\kopunct@tilde`는 올바른 형태의 물결표를 조판하는 제어 문자열이다.

`\kopunct@f@size` `\kopunct@tilde`의 정의에는 현재 글꼴 크기를 저장하는 변수 `\kopunct@f@size`와 `\kopunct@f@baselineskip` 줄 높이를 저장하는 변수 `\kopunct@f@baselineskip`이 필요하다.

```
68 \newlength\kopunct@f@size
69 \newlength\kopunct@f@baselineskip
```

`\TildeScaleFactor` 물결표의 모양을 조정하는 변수 세 개를 정의한다.

```
\TildeRaise 70 \newcommand\TildeScaleFactor{1.5}%
\TildeHeight 71 \newcommand\TildeRaise{-1.35ex}%
72 \newcommand\TildeHeight{1ex}%
```

이상의 변수들을 바탕으로 `\kopunct@tilde`를 정의한다. 글꼴 크기를 `\TildeScaleFactor`배 키운 뒤, 이를 `\TildeRaise`에 저장된 값만큼 상하로 이동하여 틸데 악센트를 조판한다.

```
73 \def\kopunct@tilde{%
74 \setlength\kopunct@f@size{\f@size pt}%
75 \setlength\kopunct@f@baselineskip{\f@baselineskip}
76 \raisebox{\TildeRaise}[\TildeHeight][0pt]{%
77 \fontsize{\TildeScaleFactor\kopunct@f@size}%
78 {\baselinestretch\kopunct@f@baselineskip}%
79 \selectfont\char"7E
80 }%
81 }%
```

`\~` 이제 `\~{}`이 `\kopunct@tilde`를 사용하도록 `\~`를 재정의한다.

```
82 \DeclareTextCompositeCommand{\~}{OT1}{\kopunct@tilde}
83 \DeclareTextCompositeCommand{\~}{T1}{\kopunct@tilde}
```

매크로 재정의가 완료되었으면 스위치를 종료한다.

```
84 \fi
85 \or
```

`\kopunct@tildestatus`의 값이 2인 경우, `\~{}`를 입력했을 때 한중일 물결표 (U+301C)를 조판하도록 `\~{}`를 재정의한다. 글꼴 문제로 아래 코드에는 기호가 표시되지 않는다.

```

86     \iftutex
87     \DeclareTextCompositeCommand{\~}{TU}{}{}
88     \else
89     \DeclareTextCompositeCommand{\~}{OT1}{}{}
90     \DeclareTextCompositeCommand{\~}{T1}{}{}
91     \fi
92     \or

```

`\kopunct@tildestatus`의 값이 3인 경우, `\~{}`를 입력했을 때 전각 물결표(U+FF5E)를 조판하도록 `\~{}`를 재정의한다.

```

93     \iftutex
94     \DeclareTextCompositeCommand{\~}{TU}{}{\~}
95     \else
96     \DeclareTextCompositeCommand{\~}{OT1}{}{\~}
97     \DeclareTextCompositeCommand{\~}{T1}{}{\~}
98     \fi

```

매크로 재정의가 완료되었으면 스위치를 종료한다.

```

99 \fi

```

7.5 oblivoir 호환 매크로 정의

oblivoir 호환 매크로는 `\ifkopunct@oblivoir` 스위치가 `true`일 때에만 정의된다.

```

100 \ifkopunct@oblivoir

```

7.5.1 가운데점

`\cntrdot` 가운데점은 `\cntrdot`으로 조판한다.

```

101 \NewCommandCopy{\cntrdot}{\textperiodcentered}

```

`\cntrdot`은 `kopunct`의 `\cdot`과는 달리 수식 모드에서는 사용할 수 없다.

7.5.2 줄임표

`\cntrdots` 줄임표는 `\cntrdots`로 조판한다. 한국어에서 사용하는 형태의 줄임표가 조판된다. 줄임표 문자를 직접 입력하더라도 동일한 결과를 얻는다.

```

102 \iftutex
103 \protected\def\cntrdots{...}
104 \else

```

```

105     \protected\def\cntrdots{\unichar{"2026}}
106     \DeclareUnicodeCharacter{2026}{\unichar{"2026}}
107     \fi

```

`\obldots` 영어의 줄임표는 `\obldots`로 조판한다.

```

108     \def\obldots{%
109         .\kern\fontdimen3\font
110         .\kern\fontdimen3\font
111         .\kern\fontdimen3\font
112     }

```

`\obellipsis` 한국어의 6점 줄임표는 `\obellipsis`로 조판한다. `oblivoir` 클래스의 정의를 그대로 사용하였다.⁸

```

113     \protected\def\obellipsis{\textellipsis\textellipsis}

```

이상의 매크로는 `kopunct`의 `\cdots`와 달리 수식 모드에서는 사용할 수 없다.

7.5.3 낫표와 화살괄호

`\snm` `\snm`은 인자를 하나 입력받아 홑낫표로 감싸 준다.

```

114     \def\snm#1{「#1」}

```

`\bnm` `\bnm`은 인자를 하나 입력받아 겹낫표로 감싸 준다.

```

115     \def\bnm#1{『#1』}

```

`\cnm` `\cnm`은 인자를 하나 입력받아 홑화살괄호로 감싸 준다.

```

116     \def\cnm#1{<#1>}

```

`\ccnm` `\ccnm`은 인자를 하나 입력받아 겹화살괄호로 감싸 준다.

```

117     \def\ccnm#1{《#1》}

```

이상의 매크로는 `\nat`, `\hwa`와 달리 인자들을 별도의 영역(group)으로 묶은 뒤 조판하지 않는다. 따라서 인자에 선언형 명령어를 사용할 때 주의해야 한다.

7.5.4 줄표

`\expldash` `\expldash`는 줄표를 조판한다. `oblivoir`와는 다르게 그냥 \TeX 의 엠 대시를 사용한다.

```

118     \protected\def\expldash{\,---\,}

```

⁸따라서 `oblivoir` 옵션을 사용한 경우에는 모든 엔진에서 한국어 줄임표가 올바르게 조판되지만, `oblivoironly` 옵션을 사용한 경우에는 유니코드 엔진에서만 한국어 줄임표가 올바르게 조판된다. 레거시 엔진에서는 ‘.....’로 나타난다.

`\explpunc` `\explpunc`도 정의되어 있다. `oblivoir`의 명령어와 동일한 방법으로 사용한다.

```
119 \protected\def\explpunc.#1.\ {\,---#1---\,}
```

매크로 정의가 완료되었으면 스위치를 종료한다.

```
120 \fi
```