

정규식(C++)

아티클 • 2023. 10. 13.

C++ 표준 라이브러리는 여러 정규식 문법을 지원합니다. 이 항목에서는 정규식을 사용할 때 사용할 수 있는 문법 변형에 대해 설명합니다.

정규식 문법

사용할 정규식 문법은 열거형 값 중 `std::regex_constants::syntax_option_type` 하나를 사용하여 지정합니다. 이러한 정규식 문법은 다음에서 정의됩니다. `std::regex_constants`

- `ECMAScript`: JavaScript 및 .NET 언어에서 사용하는 문법과 가장 가깝습니다.
- `basic`: POSIX basic 정규식 또는 BRE입니다.
- `extended`: POSIX extended 정규식 또는 ERE입니다.
- `awk`: 이 `extended` 경우 인쇄되지 않는 문자에 대한 이스케이프가 더 많이 있습니다.
- `grep`: 이 `basic` 경우 줄 바꿈(`\n`) 문자가 교대로 구분됩니다.
- `egrep`: 이 `extended` 경우 줄 바꿈 문자가 교대로 구분할 수도 있습니다.

기본적으로 문법을 지정 `ECMAScript` 하지 않으면 가정합니다. 하나의 문법만 지정할 수 있습니다.

다음과 같은 여러 플래그를 적용할 수도 있습니다.

- `icase`: 일치 시 대/소문자를 무시합니다.
- `nosubs`: 표시된 일치 항목(즉, 괄호 안의 식)을 무시합니다. 대체 항목이 저장되지 않습니다.
- `optimize`: 더 큰 건설 시간의 가능한 비용으로, 더 빨리 일치합니다.
- `collate`: 로컬 구분 데이터 정렬 시퀀스(예: 양식 `[a-z]` 범위)를 사용합니다.

0개 이상의 플래그를 문법과 결합하여 정규식 엔진 동작을 지정할 수 있습니다. 플래그만 지정 `ECMAScript` 하면 문법으로 간주됩니다.

요소

요소는 다음 중 하나일 수 있습니다.

- 대상 시퀀스에서 동일한 문자와 일치하는 *일반 문자*.
- 줄 바꿈을 제외한 대상 시퀀스의 모든 문자와 일치하는 와일드카드 문자 `.` 입니다.

- 식 $[expr]$ 에 정의된 집합에 있는 대상 시퀀스의 문자 또는 데이터 정렬 요소와 일치하거나, 식 $expr$ 에 정의된 집합에 없는 $expr$ 대상 시퀀스의 문자 또는 데이터 정렬 요소와 일치하는 폼 $[^expr]$ 의 대괄호 식입니다.

식 $expr$ 에는 다음 조합이 포함될 수 있습니다.

- 개별 문자. 에 정의된 $expr$ 집합에 문자를 추가합니다.
- 폼 $ch1-ch2$ 의 문자 범위입니다. 닫힌 범위 $[ch1, ch2]$ 의 값으로 표현되는 문자를 정의된 집합에 $expr$ 추가합니다.
- 폼 $[:name:]$ 의 문자 클래스입니다. 명명된 클래스의 문자를 $expr$ 로 정의된 집합에 추가합니다.
- 양식 $[=elt=]$ 의 동등 클래스입니다. elt 와 동급의 데이터 정렬 요소를 $expr$ 로 정의된 집합에 추가합니다.
- 양식 $[.elt.]$ 의 정렬 기호입니다. elt 로 정의된 집합에 데이터 정렬 요소 $expr$ 를 추가합니다.
- 앵커. 앵커 $^$ 는 대상 시퀀스의 시작 부분과 일치합니다. 앵커 $$$ 는 대상 시퀀스의 끝과 일치합니다.

구분 기호 사이의 패턴과 $grep$ 일치하는 대상 시퀀스의 문자 시퀀스와 일치하는 형식(하위 식) 또는 $\backslash(subexpression)\backslash basic$ 의 캡처 그룹입니다.

- 대상 시퀀스의 문자 k 와 일치하는 양식 $\backslash k$ 의 ID 이스케이프입니다.

예:

- a 대상 시퀀스를 일치하지만 대상 시 "a" 퀀스 "B" "b" 또는 "c".
- $.$ 는 모든 대상 시퀀스, "B" 및 "b" "c".와 일치합니다 "a".
- $[b-z]$ 대상 시퀀스와 "b" "c" 일치하지만 대상 시퀀스 "a" 또는 "B" 와 일치하지 않습니다.
- $[:lower:]$ 는 대상 시퀀스와 "a" "b" "c" 일치하지만 대상 시퀀스와 "B" 일치하지 않습니다.
- (a) 는 대상 시퀀스와 "a" 일치하고 캡처 그룹 1을 하위 시 "a" 퀀스와 연결하지만 대상 시퀀스 "B" "b" 또는 "c" 와 일치하지 않습니다.

에서 ECMAScript, basic 및 $grep$, 요소는 N번째 캡처 그룹과 일치하는 문자 시퀀스와 동일한 대상 시퀀스의 문자 시퀀스와 일치하는 10진수 값 N을 나타내는 폼 $\backslash dd dd$ 의 백 참조

일 수도 있습니다.

예를 들어 (a)\1 첫 번째(및 유일한) 캡처 그룹이 초기 시퀀스와 "aa" 일치한 다음 최종 시퀀스와 일치하기 \1 때문에 대상 시퀀 "a" 스와 "a" 일치합니다.

에서 ECMAScript요소는 다음 중 하나일 수도 있습니다.

- *양식의 캡처가 아닌 그룹*(?: *subexpression*)입니다. 대상 시퀀스에서 구분 기호 간 패턴과 일치하는 문자 시퀀스와 일치합니다.
- 폼 \f, , \r \t \n 또는 \v.의 제한된 *파일 형식 이스케이프*입니다. 대상 시퀀스에서 각각 폼 피드, 줄 바꿈, 캐리지 리턴, 가로 탭, 수직 탭과 일치합니다.
- *양식의 긍정 어설션*(= *하위 식*)입니다. 구분 기호 간의 패턴과 일치하는 대상 시퀀스의 문자 시퀀스를 일치하지만 대상 시퀀스의 일치 위치는 변경되지 않습니다.
- *형식의 부정 어설션*(! *subexpression*). 구분 기호 간의 패턴과 일치하지 않고 대상 시퀀스의 일치 위치를 변경하지 않는 대상 시퀀스의 모든 문자 시퀀스와 일치합니다.
- 양식 \xhh의 *16진수 이스케이프 시퀀스*입니다. 대상 시퀀스에서 두 자릿수 16진수 hh로 표현된 문자와 일치합니다.
- 양식 \uhhhh의 *유니코드 이스케이프 시퀀스*입니다. 대상 시퀀스에서 네 자릿수 16진수 hhhh로 표현된 문자와 일치합니다.
- 폼 \ck의 *컨트롤 이스케이프 시퀀스*입니다. 문자 k라는 이름의 컨트롤 문자와 일치합니다.
- 양식 \b의 *단어 경계 어설션*입니다. 대상 시퀀스에서 현재 위치가 *단어 경계* 바로 다음일 경우 일치합니다.
- 형식 \B의 *음수 단어 경계 어설션*입니다. 대상 시퀀스의 현재 위치가 단어 경계 *바로 뒤*가 아닌 경우 일치합니다.
- , , \w \s \S, \W 형식 \D \d의 *dsw 문자 이스케이프*입니다. 문자 클래스의 약식 이름을 제공합니다.

예:

- (?:a) 는 대상 시퀀스 "a"와 일치하지만 "(?:a)\1" 캡처 그룹 1이 없으므로 유효하지 않습니다.
- (=a)a 는 대상 시퀀스 "a"와 일치합니다. 양수 어설션은 대상 시퀀스의 초기 시퀀스와 "a" 일치하고 정규식의 마지막 "a" 은 대상 시퀀스의 초기 시퀀스와 "a" 일치합니다.

- `(!a)a` 는 대상 시퀀스 "a"와 일치하지 않습니다.
- `a\b.` 는 대상 시퀀스를 일치하지만 대상 시 "a~" 퀀스 "ab"와 일치하지 않습니다.
- `a\B.` 는 대상 시퀀스를 일치하지만 대상 시 "ab" 퀀스 "a~"와 일치하지 않습니다.

에서 awk요소는 다음 중 하나일 수도 있습니다.

- 형식 `\[, \a, \n \b \f \r \t`, 또는 `\v.`의 파일 형식 이스케이프입니다. 대상 시퀀스에서 각각 백슬래시, 경고, 백스페이스, 폼 피드, 줄 바꿈, 캐리지 리턴, 가로 탭, 수직 탭과 일치합니다.
- 폼 `\ooo`의 8진수 이스케이프 시퀀스입니다. 대상 시퀀스에서 표현이 한 자리, 두 자리 또는 세 자리 8진수 `ooo`로 표현된 값인 문자와 일치합니다.

반복

긍정 어설선, 부정 어설선 또는 앵커 이외의 모든 요소 다음에 반복 횟수가 올 수 있습니다. 가장 일반적인 종류의 반복 횟수는 `{min,max}`, 또는 `\{min,max\}` basic 형식을 grep사 용합니다. 이 형식의 반복 횟수가 뒤에 오는 요소는 최소 최소 연속 항목과 일치하며 요소와 일치하는 시퀀스의 최대 연속 발생을 초과하지 않습니다.

예를 들어 `a{2,3}` 대상 시퀀스 및 대상 시퀀스와 "aaa" "aa" 일치하지만 대상 시퀀스 또는 대상 시퀀스는 "a" "aaaa" 일치하지 않습니다.

또한 반복 횟수는 다음 형식 중 하나를 사용할 수 있습니다.

- `{min}` 또는 `\{min\}` in basic and grep. `{min,min}`에 해당합니다.
- `{min,}` 또는 `\{min,\}` in basic and grep. `{min,unbounded}`에 해당합니다.
- `*`는 `{0,unbounded}`에 해당합니다.

예:

- `a{2}` 대상 시퀀스나 대상 시퀀스가 "aa" 아닌 대상 시퀀스 "a" "aaa"와 일치합니다.
- `a{2,}` 는 대상 시퀀스 "aa", 대상 시퀀스 "aaa" 등과 일치하지만 대상 시퀀스와 "a" 일치하지 않습니다.
- `a*` 는 대상 시퀀스 "", 대상 시퀀스 "a", 대상 시퀀스 "aa" 등과 일치합니다.

basic 및 grep를 제외한 모든 문법의 경우 반복 횟수는 다음 중 하나의 형식이 될 수도 있습니다.

- $?$ 는 $\{0,1\}$ 와 같습니다.
- $+$ 는 $\{1, \text{unbounded}\}$ 에 해당합니다.

예:

- $a?$ 대상 시퀀스 및 대상 시퀀스와 "" 일치하지만 대상 시퀀 "a" 스는 "aa" 일치하지 않습니다.
- $a+$ 는 대상 시퀀스 "a", 대상 시퀀스 "aa" 등과 일치하지만 대상 시퀀스는 "" 일치하지 않습니다.

에서 ECMAScript반복 횟수의 모든 형태 뒤에 욕심이 없는 반복을 지정하는 문자 $?$ 가 뒤따를 수 있습니다.

Concatenation

정규식 요소는 *반복 횟수* 유무와 상관없이 연결하여 긴 정규식을 형성할 수 있습니다. 결과 식은 개별 요소와 일치하는 시퀀스 연결인 대상 시퀀스와 일치합니다.

예를 들어 $a\{2,3\}b$ 대상 시퀀스 및 대상 시퀀스와 "aaab" "aab" 일치하지만 대상 시퀀스 또는 대상 시퀀스와 "ab" "aaaab" 일치하지 않습니다.

교체

을 제외한 basic grep모든 정규식 문법에서 연결된 정규식 뒤에 문자 $|$ (파이프) 및 연결된 다른 정규식이 뒤따를 수 있습니다. 이러한 방식으로 연결된 정규식을 무한히 결합할 수 있습니다. 결과 식은 하나 이상의 연결된 정규식과 일치하는 대상 시퀀스와 일치합니다.

연결된 정규식 중 하나 이상이 대상 시퀀스와 일치하는 경우 시퀀 ECMAScript 스와 일치하는 첫 번째 정규식을 일치 항목으로 선택합니다. 이 정규식을 첫 번째 일치 *항목이라고* 합니다. 다른 정규식 문법은 가장 긴 일치를 *달성하는 문법*을 선택합니다.

예를 들어 $ab|cd$ 대상 시퀀스 및 대상 시퀀스와 "cd" "ab" 일치하지만 대상 시퀀스 또는 대상 시퀀스와 "abd" "acd" 일치하지 않습니다.


In grep 및 egrep, 줄 바꿈 문자($\backslash n$)를 사용하여 번갈아 구분할 수 있습니다.

하위 식

basic 및 grep에서 하위 식은 연결입니다. 다른 정규식 문법에서 하위 식은 교체입니다.

문법 요약

다음 표에 다양한 정규식 문법에서 사용할 수 있는 기능이 요약되어 있습니다.

 테이블 확장

요소	basic	extended	ECMAScript	grep	egrep	awk
을 사용하여 변경		+	+		+	+
을 사용하여 변경 \n				+	+	
anchor	+	+	+	+	+	+
역참조	+		+	+		
대괄호 식	+	+	+	+	+	+
를 사용하여 그룹 캡처 ()		+	+		+	+
를 사용하여 그룹 캡처 \(\)	+			+		
컨트롤 이스케이프 시퀀스			+			
dsw 문자 이스케이프			+			
파일 형식 이스케이프			+			+
16진수 이스케이프 시퀀스			+			
ID 이스케이프	+	+	+	+	+	+
부정 어설선			+			
부정 단어 경계 어설선			+			
비캡처 그룹			+			
non-greedy 반복			+			
8진수 이스케이프 시퀀스						+
일반 문자	+	+	+	+	+	+
긍정 어설선			+			
를 사용하여 반복 {}		+	+		+	+
를 사용하여 반복 \{\}	+			+		
를 사용하여 반복 *	+	+	+	+	+	+

요소	basic	extended	ECMAScript	grep	egrep	awk
반복 사용 ? 및 +		+	+		+	+
유니코드 이스케이프 시퀀스(unicode escape sequence)			+			
와일드카드 문자	+	+	+	+	+	+
단어 경계 어설션(word boundary assert)			+			

의미 체계 세부 정보

기준 위치

앵커는 대상 문자열에서 문자가 아닌 위치와 일치합니다. 대상 `^` 문자열의 시작 부분과 일치하고 `$` 대상 문자열의 끝과 일치합니다.

뒤로 참조

역참조는 백슬래시 뒤에 10진수 값 `N`이 붙는 형식이며, `N`번째 *캡처 그룹*의 내용과 일치합니다. `N` 값은 역참조 앞의 캡처 그룹 수보다 많을 수 없습니다. basic 및 grep에서 `N` 값은 백슬래시 다음의 10진수에 의해 결정됩니다. ECMAScript에서 `N` 값은 백슬래시 바로 다음의 모든 10진수에 의해 결정됩니다. 따라서 basic 및 grep에서 정규식의 캡처 그룹이 9개를 초과하는 경우에도 `N` 값은 절대 9를 초과하지 않습니다. ECMAScript에서 `N` 값은 제한이 없습니다.

예:

- `((a+)(b+))(c+)\3` 는 대상 시퀀스 `"aabbbcbbb"` 와 일치합니다. 뒤로 참조 `\3` 는 세 번째 캡처 그룹의 텍스트, 즉 `"(b+)"`. 대상 시퀀스 `"aabbbcbb"` 와 일치하지 않습니다.
- `(a)\2` 가 잘못되었습니다.
- `(b((((((((((a))))))))))\10` 의 의미와 ECMAScript의 의미는 basic 다릅니다. 에서 basic 백 참조는 .입니다 `\1`. 뒤로 참조는 첫 번째 캡처 그룹의 내용(즉, 최종으로 시작하고 (b 종료) 되고 백 참조 앞에 오는 항목)과 일치하며, 마지막 `0` 은 일반 문자 `0` 와 일치합니다. 에서 ECMAScript 백 참조는 .입니다 `\10`. 10번째 캡처 그룹, 즉 가장 안쪽의 캡처 그룹과 일치합니다.

대괄호 식

대괄호 식은 문자 집합 및 *데이터 정렬* 요소를 정의합니다. 대괄호 식이 문자로 시작되면 집합에 대상 시퀀스의 현재 문자 \wedge 와 일치하는 요소가 없으면 일치가 성공합니다. 그렇지 않을 경우, 집합의 요소 하나라도 대상 시퀀스의 현재 문자와 일치하면 일치가 성공합니다.

문자 집합은 *개별 요소*, *문자 범위*, *문자 클래스*, *동등 클래스* 및 *데이터 정렬 기호*를 임의로 결합하여 정의할 수 있습니다.

캡처 그룹

캡처 그룹은 정규식 문법에서 내용을 단일 단위로 표시하며 내용과 일치하는 대상 텍스트에 레이블을 지정합니다. 각 캡처 그룹과 연결된 레이블은 캡처 그룹을 표시하는 여는 괄호와 현재 캡처 그룹을 표시하는 여는 괄호까지 포함해 계산하여 결정되는 숫자입니다. 이 구현에서 캡처 그룹의 최대 수는 31입니다.

예:

- ab^+ 는 대상 시퀀스를 일치하지만 대상 시 "abb" 퀀스 "abab" 와 일치하지 않습니다.
- $(ab)^+$ 는 대상 시퀀스와 일치하지 않지만 대상 시 "abb" 퀀스 "abab" 와 일치합니다.
- $((a^+)(b^+))(c^+)$ 는 대상 시퀀스를 "aabbbc" 일치시키고 캡처 그룹 1을 하위 시퀀스와 "aabbbb" 연결하고, 하위 시퀀 "aa" 스를 사용하여 그룹 2를 캡처하고, 그룹 3을 "bbb" 캡처하고, 그룹 4를 하위 시퀀스와 "c" 연결합니다.

문자 클래스

대괄호 식의 문자 클래스는 명명된 클래스의 모든 문자를 대괄호 식으로 정의된 문자 집합에 추가합니다. 문자 클래스를 만들려면 클래스 이름 `:` 뒤에 다음을 사용합니다 `[`:

내부적으로 문자 클래스의 이름은 `id = traits.lookup_classname` 을 호출하여 인식됩니다. `ch` 에서 `true` 를 반환할 경우 문자 `traits.isctype(ch, id)` 는 해당 클래스에 속합니다. 기본 `regex_traits` 템플릿은 다음 표의 클래스 이름을 지원합니다.

[\[\]](#) 테이블 확장

클래스 이름	설명
alnum	소문자, 대문자 및 숫자
alpha	소문자 및 대문자
blank	공백 또는 탭

클래스 이름	설명
<code>cntrl</code>	파일 서식 이스케이프 문자
<code>digit</code>	<code>digits</code>
<code>graph</code>	소문자, 대문자, 숫자 및 문장 부호
<code>lower</code>	소문자
<code>print</code>	소문자, 대문자, 숫자, 문장 부호 및 공백
<code>punct</code>	문장 부호
<code>space</code>	<code>space</code>
<code>upper</code>	대문자
<code>xdigit</code>	<code>digits</code> , <code>,</code> , <code>a</code> , <code>b</code> , <code>c</code> , <code>e</code> , <code>d</code> , <code>f</code> , <code>A</code> , <code>B</code> , <code>C</code> , <code>D</code> , <code>E</code> , <code>F</code>
<code>d</code>	<code>digit</code> 과 같습니다.
<code>s</code>	<code>space</code> 과 같습니다.
<code>w</code>	<code>alnum</code> 과 같습니다.

문자 범위

대괄호 식의 문자 범위는 범위의 모든 문자를 대괄호 식으로 정의된 문자 집합에 추가합니다. 문자 범위를 만들려면 범위의 첫 번째 문자와 마지막 문자 사이에 문자를 '-' 배치합니다. 문자 범위는 첫 번째 문자의 숫자 값보다 크거나 같고 마지막 문자의 숫자 값보다 작거나 같은 숫자 값을 가진 모든 문자를 집합에 넣습니다. 이 추가된 문자 집합은 플랫폼 별 문자 표현에 따라 달라집니다. 문자 '-' 가 대괄호 식의 시작 또는 끝에서 발생하거나 문자 범위의 첫 번째 또는 마지막 문자로 발생하는 경우 문자 자체를 나타냅니다.

예:

- `[0-7]` 는 { `0`, `,`, `2`, `1`, `4`, `3`, `5`, `6` } 7 문자 집합을 나타냅니다. 대상 시퀀스 등과 `"0"` `"1"` 일치하지만 일치하지는 않습니다 `"a"`.
- ASCII 문자 인코딩을 사용하는 시스템에서 { `h`, `[h-k]`, `i`, `j`, `k` }의 문자 집합을 나타냅니다. 대상 시퀀스 등과 `"h"` 일치하지만 그렇지 않습니다 `"\x8A"` `"0"`. `"i"`
- EBCDIC 문자 인코딩을 사용하는 시스템에서 { `,`, `[h-k]`, `i`, `'\x8B'` `'\x8D'` `'\x8C'` `'\x8F'` `'\x90'` `'\x8E'` `k` `j` `'\x8A'` } (`h` 로 인코딩되고 `k`)로

0x92 0x88 인코딩되는 문자 집합을 나타냅니다. h 대상 시퀀스 "h", "i", "\x8A" 등과 일치하지만 일치하지는 않습니다 "0".

- [-0-24] 는 { -, , 0 1, 2 4 } 문자 집합을 나타냅니다.
- [0-2-] 는 { 0, , 1 2 } 문자 집합을 - 나타냅니다.
- ASCII 문자 인코딩을 [+--] 사용하는 시스템에서 { + , - } 문자 집합을 나타냅니다.

하지만 로캘에 민감한 범위를 사용할 경우 범위의 문자는 로캘에 대한 데이터 정렬 규칙으로 결정됩니다. 범위 정의의 첫 번째 문자 다음과 범위 정의의 마지막 문자 앞에서 정렬되는 문자가 집합에 있습니다. 두 개의 끝 문자도 집합에 있습니다.

Collating 요소

데이터 정렬 요소는 단일 문자로 취급되는 복수 문자 시퀀스입니다.

기호 정렬

대괄호 식의 데이터 정렬 기호는 대괄호 식으로 정의된 집합에 *데이터 정렬 요소*를 추가합니다. 정렬 기호를 만들려면 정렬 요소 뒤에 다음을 사용합니다 [. . .]

컨트롤 이스케이프 시퀀스

컨트롤 이스케이프 시퀀스는 백슬래시 뒤에 문자 'c' 가 뒤에 오거나 'A' 'z' 흐 'z' 른 문자 'a' 중 하나입니다. 해당 문자로 이름이 지정되는 ASCII 컨트롤 문자와 일치합니다. 예를 들어 "\ci" Ctrl+I 값이 있으므로 대상 시퀀스를 "\x09" 일치합니다 0x09.

DSW 문자 이스케이프

dsw 문자 이스케이프는 다음 표와 같이 문자 클래스에 대한 짧은 이름입니다.

[\[\] 테이블 확장](#)

이스케이프 시퀀스	동일한 명명 클래스	기본 명명 클래스
\d	[[[:d:]]	[[[:digit:]]
\D	[^[:d:]]	[^[:digit:]]
\s	[[[:s:]]	[[[:space:]]
\S	[^[:s:]]	[^[:space:]]

이스케이프 시퀀스	동일한 명명 클래스	기본 명명 클래스
\w	[[:w:]]	[a-zA-Z0-9_]*
\W	[^ :w:]	[^a-zA-Z0-9_]*

*ASCII 문자 집합

동등 클래스

대괄호 식의 동등 클래스는 동급 클래스 정의의 데이터 정렬 요소와 동일한 모든 문자와 *데이터 정렬 요소*를 대괄호 식으로 정의된 집합에 추가합니다.

등가 클래스를 만들려면 그 뒤에 정렬 요소와 다음 =] 을 사용합니다 [= . elt1 일 경우, 내부적으로 두 개의 데이터 정렬 요소 elt2 및 traits.transform_primary(elt1.begin(), elt1.end()) == traits.transform_primary(elt2.begin(), elt2.end()) 는 동일합니다.

파일 형식 이스케이프

파일 형식 이스케이프는 일반적인 C 언어 문자 이스케이프 시퀀스, \\, \a, \b, \n \f, \r, \t \v. 백슬래시, 경고, 백스페이스, 양식 피드, 줄 바꿈, 캐리지 리턴, 가로 탭, 세로 탭 등 일반적인 의미가 있습니다. 에서 ECMAScript 허용되지 \a \b 않습니다. (\\ 허용되지만 파일 형식 이스케이프가 아니라 ID 이스케이프입니다).

16진수 이스케이프 시퀀스

16진수 이스케이프 시퀀스는 백슬래시 뒤에 두 개의 16진수 숫자(0-9a-fA-F)가 뒤에 있는 문자 x 입니다. 이는 2자리로 지정된 값을 갖는 대상 시퀀스의 문자와 일치합니다.

예를 들어 "\x41" ASCII 문자 인코딩을 사용하는 경우 대상 시퀀스를 "a" 일치합니다.

ID 이스케이프

ID 이스케이프는 백슬래시와 단일 문자의 순서로 구성되어 있습니다. 해당 문자와 일치합니다. 문자에 특별한 의미가 있는 경우 필요합니다. ID 이스케이프를 사용하면 특별한 의미가 제거됩니다. 예시:

- a* 는 대상 시퀀스를 일치하지만 대상 시 "aaa" 퀀스 "a*" 와 일치하지 않습니다.
- a* 는 대상 시퀀스와 일치하지 않지만 대상 시 "aaa" 퀀스 "a*" 와 일치합니다.

ID 이스케이프에 허용되는 문자 집합은 다음 표와 같이 정규식 문법에 따라 다릅니다.

문법	ID 이스케이프 문자에 허용됩니다.
basic, grep	{ (\$ ^ * [] \) { . }
extended, egrep	{ (? + \$ ^ \ . *) { [] }
awk, extended	plus { " / }
ECMAScript	식별자에 포함될 수 있는 문자를 제외한 모든 문자. 일반적으로 문자, 숫자, \$ _ 유니코드 이스케이프 시퀀스가 포함됩니다. 자세한 내용은 언어 사양을 ECMAScript 참조하세요.

개별 문자

대괄호 식의 개별 문자는 해당 문자를 대괄호 식으로 정의된 문자 집합에 추가합니다. 시작 ^ 위치를 제외한 대괄호 식의 아무 곳이나 자신을 나타냅니다.

예:

- [abc] 는 대상 시퀀스와 일치하지만 "c" 시퀀스는 "d" "a" "b" 일치하지 않습니다.
- [^abc] 대상 시퀀스가 아닌 대상 시퀀 "d" 스 "a" "b" 또는 "c".
- [a^bc] 는 대상 시퀀스, "b" 및 "c" "^" 대상 시퀀스와 "a" 일치하지만 대상 시퀀스는 "d" 일치하지 않습니다.

을 제외한 ECMAScript 모든 정규식 문법에서 a] 가 여 [는 문자 뒤에 있는 첫 번째 문자이거나 이니셜 ^ 뒤에 있는 첫 번째 문자인 경우 자체를 나타냅니다.

예:

- []a 는 대괄호 식을 종료할 수 없으므로 유효하지 않습니다] .
- []abc 는 대상 시퀀스, "b" 및 "c" "]" 대상 시퀀스와 "a" 일치하지만 대상 시퀀스는 "d" 일치하지 않습니다.
- [^]abc 대상 시퀀스가 아닌 대상 시퀀 "d" 스 "a", "b" "c" 또는 "]" .

에서 ECMAScript 대괄호 식의 문자를] 나타내는 데 사용합니다 \ .

예:

- `[]a` 는 대괄호 식이 비어 있기 때문에 대상 시퀀스 "a" 와 일치합니다.
- `[\\]abc` 는 대상 시퀀스와 일치하지만 대상 시퀀스는 "a" "d" "b" "c" 일치하지 않습니다. "]"

부정 어설선

부정 어설선은 해당 내용 이외의 모든 항목과 일치합니다. 대상 시퀀스의 문자는 소비하지 않습니다.

예를 들어 `(!aa)(a*)` 대상 시퀀스를 "a" 일치시키고 캡처 그룹 1을 하위 시퀀스와 "a" 연결합니다. 대상 시퀀스 또는 대상 시퀀스 "aa" "aaa" 와 일치하지 않습니다.

음수 단어 경계 어설선

음수 단어 경계 어설선은 대상 문자열의 현재 위치가 단어 경계 바로 뒤가 아닌 경우 일치합니다.

비 캡처 그룹

캡처되지 않은 그룹은 해당 내용을 정규식 문법의 단일 단위로 표시하지만 대상 텍스트에 레이블을 지정하지는 않습니다.

예를 들어 `(a)(?:b)*(c)` 대상 텍스트를 "abbc" 일치시키고 캡처 그룹 1을 하위 시퀀스와 "a" 연결하고 그룹 2를 하위 시퀀스와 "c" 연결합니다.

욕심이 없는 반복

non-greedy 반복은 패턴과 일치하는 대상 시퀀스의 가장 짧은 하위 시퀀스를 사용합니다. greedy 반복은 가장 긴 항목을 사용합니다. 예를 들어 `(a+)(a*b)` 대상 시퀀스를 "aaab" 일치합니다.

비욕심 반복을 사용하면 대상 시퀀스의 시작 부분에 있는 하위 시퀀 "a" 스와 캡처 그룹 1을 연결하고, 캡처 그룹 2를 대상 시퀀스의 끝에 있는 하위 시퀀스와 "aab" 연결합니다.

greedy 일치를 사용하면 캡처 그룹 1을 하위 시퀀스와 "aaa" 연결하고 캡처 그룹 2를 하위 시퀀스와 "b" 연결합니다.

8진수 이스케이프 시퀀스

8진수 이스케이프 시퀀스는 백슬래시와 한 자리, 두 자리 또는 세 자리 8진수(0-7)로 구성됩니다. 이러한 자릿수로 지정된 값을 갖는 대상 시퀀스의 문자와 일치합니다. 모든 숫자가 0 있으면 시퀀스가 유효하지 않습니다.

예를 들어 `\t` ASCII 문자 인코딩을 사용하는 경우 대상 시퀀스를 "a" 일치합니다.

일반 문자

일반 문자는 현재 문법에 특별한 의미가 없는 유효한 문자입니다.

ECMAScript에서 다음 문자는 특별한 의미가 있습니다.

- `^ $ \ . * + ? () [] { } |`

basic 및 grep에서 다음 문자는 특별한 의미가 있습니다.

- `. [\`

basic grep 또한 다음 문자는 특정 컨텍스트에서 사용될 때 특별한 의미를 갖습니다.

- `*` 정규식의 첫 번째 문자이거나 정규식의 이니셜 `^` 뒤에 있는 첫 번째 문자이거나 캡처 그룹의 첫 번째 문자이거나 캡처 그룹의 초기 `^` 뒤에 있는 첫 번째 문자인 경우를 제외하고 모든 경우에 특별한 의미가 있습니다.
- `^` 정규식의 첫 번째 문자인 경우 특별한 의미가 있습니다.
- `$` 정규식의 마지막 문자인 경우 특별한 의미가 있습니다.

extended, egrep 및 awk에서 다음 문자는 특별한 의미가 있습니다.

- `. [\ (* + ? { |`

extended 또한 , egrep 및 awk 다음 문자는 특정 컨텍스트에서 사용될 때 특별한 의미를 갖습니다.

- `)` 에는 이전과 일치하는 경우 특별한 의미가 있습니다. `(`
- `^` 정규식의 첫 번째 문자인 경우 특별한 의미가 있습니다.
- `$` 정규식의 마지막 문자인 경우 특별한 의미가 있습니다.

대상 시퀀스에서 동일한 문자와 일치하는 일반 문자. 기본적으로 두 문자가 동일한 값으로 표현될 경우 일치 성공함을 의미합니다. 대/소문자를 구분하는 일치에서 `ch0` 일 경우 두 문자 `ch1` 및 `traits.translate_nocase(ch0) == traits.translate_nocase(ch1)` 이 일

치합니다. 로컬을 구분하는 일치에서 `ch0` 일 경우 두 문자 `ch1` 및 `traits.translate(ch0) == traits.translate(ch1)` 이 일치합니다.

긍정 어설선

긍정 어설선은 해당 내용과 일치하지만 대상 시퀀스에서 문자를 소비하지는 않습니다.

예:

- `(=aa)(a*)` 는 대상 시퀀스와 "aaaa" 일치하고 캡처 그룹 1을 하위 시퀀스와 "aaaa" 연결합니다.
- `(aa)(a*)` 는 대상 시퀀스를 "aaaa" 일치시키고 캡처 그룹 1을 대상 시퀀스의 시작 부분에 있는 하위 시 "aa" 퀀스와 연결하고, 캡처 그룹 2를 대상 시퀀스의 끝에 있는 하위 시퀀스와 "aa" 연결합니다.
- `(=aa)(a)|(a)` 는 대상 시퀀스와 "a" 일치하고 캡처 그룹 1을 빈 시퀀스(긍정 어설선 실패로 인해)와 연결하고 그룹 2를 하위 시퀀스와 "a" 연결합니다. 또한 대상 시퀀스를 "aa" 일치시키고 캡처 그룹 1을 하위 시퀀스와 "aa" 연결하고 빈 시퀀스를 사용하여 그룹 2를 캡처합니다.

유니코드 이스케이프 시퀀스

유니코드 이스케이프 시퀀스는 백슬래시 뒤에 4개의 16진수 숫자(`0-9a-fA-F`)가 뒤에 있는 문자 'u' 입니다. 이는 4자리로 지정된 값을 갖는 대상 시퀀스의 문자와 일치합니다. 예를 들어 `\u0041` ASCII 문자 인코딩을 사용하는 경우 대상 시퀀스를 "a" 일치합니다.

와일드카드 문자

와일드카드 문자는 줄 바꿈을 제외하고 대상 식에서 모든 문자와 일치합니다.

단어 경계(word boundary)

단어 경계는 다음과 같은 경우에 발생합니다.

- 현재 문자가 대상 시퀀스의 시작 부분에 있고 단어 문자 `A-Za-z0-9_` 중 하나인 경우
- 현재 문자 위치가 대상 시퀀스의 끝 다음에 있으며 대상 시퀀스의 마지막 문자가 단어 문자 중 하나인 경우.
- 현재 문자는 단어 문자 중 하나이며 이전 문자는 그렇지 않습니다.

- 현재 문자는 단어 문자 중 하나가 아니며 앞의 문자는 단어 문자입니다.

Word 경계 어설선

대상 문자열에서 현재 위치가 *단어 경계* 바로 다음에 있을 경우 일치하는 단어 경계 어설선입니다.

일치 및 검색

대상 시퀀스와 일치하는 정규식의 경우 전체 정규식이 전체 대상 시퀀스와 일치해야 합니다. 예를 들어 정규식 `bcd` 은 대상 시퀀스와 일치하지만 대상 시퀀스 `"bcd"` 쿼스나 대상 시퀀스 `"abcd"` `"bcde"` 와 일치하지 않습니다.

정규식 검색이 성공하려면 대상 시퀀스에 정규식과 일치하는 하위 시퀀스가 있어야 합니다. 검색은 일반적으로 가장 왼쪽에서 일치하는 하위 시퀀스를 찾습니다.

예:

- 대상 시퀀스의 정규식 `bcd` 검색이 성공하고 전체 시퀀스 `"bcd"` 쿼스와 일치합니다. 대상 시퀀스에서 `"abcd"` 동일한 검색도 성공하고 마지막 세 문자와 일치합니다. 대상 시퀀스에서 `"bcde"` 동일한 검색도 성공하고 처음 세 문자와 일치합니다.
- 대상 시퀀스에서 `"bcdabcd"` 정규식 `bcd` 검색이 성공하고 처음 세 문자와 일치합니다.

대상 시퀀스의 일부 위치에서 일치하는 하위 시퀀스가 두 개 이상 있는 경우 일치하는 패턴을 선택하는 두 가지 방법이 있습니다.

*처음 일치*는 정규식이 일치할 경우 첫 번째로 검색된 하위 시퀀스를 선택합니다.

*최대 길이 일치*는 해당 위치에서 일치하는 하위 시퀀스에서 가장 긴 하위 시퀀스를 선택합니다. 최대 길이가 두 개 이상인 하위 시퀀스가 있는 경우 가장 긴 일치 항목이 먼저 발견된 하위 시퀀스를 선택합니다.

예를 들어 첫 번째 일치를 사용하는 경우 대체의 왼쪽 용어가 해당 하위 시퀀스 `"b"` 스와 일치하므로 대상 시퀀스에서 정규식을 `b|bc` 검색하면 하위 시퀀스 `"abcd"` 스가 일치하므로 첫 번째 일치하는 대체의 오른쪽 용어를 시도하지 않습니다. 가장 긴 일치 항목이 사용되는 경우 동일한 검색이 `"b"` 일치 `"bc"` `"bc"` 합니다.

정규식의 끝에 도달하지 않은 경우에도 일치 항목이 실패하지 않고 대상 시퀀스의 끝에 도달하면 부분 일치가 성공합니다. 따라서 부분 일치가 성공한 다음 대상 시퀀스에 문자를 추가하면 나중에 부분 일치가 실패할 수 있습니다. 그러나 부분 일치가 실패하면 대상

시퀀스에 문자를 추가해도 나중에 부분 일치가 성공할 수 없습니다. 예를 들어 부분 일치 ab 항목은 대상 시퀀스와 "a" 일치하지만 일치하지는 않습니다 "ac".

서식 플래그

[🔗](#) 테이블 확장

ECMAScript 형식 규칙	sed 형식 규칙	대체 텍스트
\$&	&	전체 정규식과 일치하는 문자 시퀀스입니다. [match[0].first, match[0].second)
\$		\$
	\&	&
\$` (달러 기호 뒤에 따옴표가 있는 경우)		정규식과 일치하는 하위 시퀀스 앞에 오는 문자 시퀀스입니다. [match.prefix().first, match.prefix().second)
\$' (달러 기호 뒤에 따옴표가 있는 경우)		정규식과 일치하는 하위 시퀀스를 따르는 문자 시퀀스입니다. [match.suffix().first, match.suffix().second)
\$n	\n	위치 n의 캡처 그룹과 일치하는 문자 시퀀스입니다. 여기서 n 0에서 9 사이의 숫자입니다. [match[n].first, match[n].second)
	\\n	\\n
\$nn		10에서 99 사이의 숫자인 위치에 nn 있는 nn 캡처 그룹과 일치하는 문자 시퀀스입니다. [match[nn].first, match[nn].second)

참고 항목

[C++ 표준 라이브러리 개요](#)

피드백

이 페이지가 도움이 되었나요?

👍 Yes

👎 No

[제품 사용자 의견 제공](#) | [Microsoft Q&A에서 도움말 보기](#)