characterp

Character protocol.

author:
`Paulo Moura`
version:
`1.2`
date:
`2011/2/19`

compilation:
`static, context_switching_calls`

(no dependencies on other files)

Public interface

is_ascii/1

True if the argument is an ASCII character.

compilation:
`static`

template:
`is_ascii(Char)`

mode - number of solutions:
`is_ascii(+char) - zero_or_one`

is_alphanumeric/1

True if the argument is an alphanumeric character.

compilation:
`static`

template:
`is_alphanumeric(Char)`

mode - number of solutions:
`is_alphanumeric(+char) - zero_or_one`

is_alpha/1

True if the argument is a letter or an underscore.

compilation:
`static`

template:
`is_alpha(Char)`

mode - number of solutions:
`is_alpha(+char) - zero_or_one`

is_letter/1

True if the argument is a letter.

compilation:
`static`

template:
```
is_letter(Char)
```
mode - number of solutions:
```
is_letter(+char) - zero_or_one
```

is_bin_digit/1

True if the argument is a binary digit.

compilation:
```
static
```
template:
```
is_bin_digit(Char)
```
mode - number of solutions:
```
is_bin_digit(+char) - zero_or_one
```

is_octal_digit/1

True if the argument is an octal digit.

compilation:
```
static
```
template:
```
is_octal_digit(Char)
```
mode - number of solutions:
```
is_octal_digit(+char) - zero_or_one
```

is_dec_digit/1

True if the argument is a decimal digit.

compilation:
```
static
```
template:
```
is_dec_digit(Char)
```
mode - number of solutions:
```
is_dec_digit(+char) - zero_or_one
```

is_hex_digit/1

True if the argument is an hexadecimal digit.

compilation:
```
static
```
template:
```
is_hex_digit(Char)
```
mode - number of solutions:
```
is_hex_digit(+char) - zero_or_one
```

is_lower_case/1

True if the argument is a lower case letter.

compilation:
```
static
```
template:
```
is_lower_case(Char)
```

mode - number of solutions:
```
is_lower_case(+char) - zero_or_one
```

is_upper_case/1

True if the argument is a upper case letter.

compilation:
```
static
```

template:
```
is_upper_case(Char)
```

mode - number of solutions:
```
is_upper_case(+char) - zero_or_one
```

is_vowel/1

True if the argument is a vowel.

compilation:
```
static
```

template:
```
is_vowel(Char)
```

mode - number of solutions:
```
is_vowel(+char) - zero_or_one
```

is_white_space/1

True if the argument is a white space character (a space or a tab) inside a line of characters.

compilation:
```
static
```

template:
```
is_white_space(Char)
```

mode - number of solutions:
```
is_white_space(+char) - zero_or_one
```

is_layout/1

True if the argument is a layout character.

compilation:
```
static
```

template:
```
is_layout(Char)
```

mode - number of solutions:
```
is_layout(+char) - zero_or_one
```

is_quote/1

True if the argument is a quote character.

compilation:
```
static
```

template:
```
is_quote(Char)
```

mode - number of solutions:
```
is_quote(+char) - zero_or_one
```

is_punctation/1

True if the argument is a sentence punctation character.

compilation:
`static`

template:
`is_punctation(Char)`

mode - number of solutions:
`is_punctation(+char) - zero_or_one`

is_period/1

True if the argument is a character that ends a sentence.

compilation:
`static`

template:
`is_period(Char)`

mode - number of solutions:
`is_period(+char) - zero_or_one`

is_control/1

True if the argument is an ASCII control character.

compilation:
`static`

template:
`is_control(Char)`

mode - number of solutions:
`is_control(+char) - zero_or_one`

is_newline/1

True if the argument is the ASCII newline character.

compilation:
`static`

template:
`is_newline(Char)`

mode - number of solutions:
`is_newline(+char) - zero_or_one`

is_end_of_line/1

True if the argument is the ASCII end-of-line character (either a carriage return or a line feed).

compilation:
`static`

template:
`is_end_of_line(Char)`

mode - number of solutions:
`is_end_of_line(+char) - zero_or_one`

parenthesis/2

Recognises and converts between open and close parenthesis.

compilation:
`static`

template:
`parenthesis(Char1,Char2)`

mode - number of solutions:
```
parenthesis(?char,?char) - zero_or_more
parenthesis(+char,?char) - zero_or_one
parenthesis(?char,+char) - zero_or_one
```

lower_upper/2

Recognises and converts between lower and upper case letters.

compilation:
`static`

template:
`lower_upper(Char1,Char2)`

mode - number of solutions:
```
lower_upper(?char,?char) - zero_or_more
lower_upper(+char,?char) - zero_or_one
lower_upper(?char,+char) - zero_or_one
```

Protected interface

(none)

Private predicates

(none)