Ruby master - Feature #10770

chr and ord behavior for ill-formed byte sequences and surrogate code points

01/22/2015 01:09 AM - masakielastic (Masaki Kagaya)

Status:	Open
Priority:	Normal
Assignee:	
Target version:	

Description

ord raises error when meeting ill-formed byte sequences, thus the difference of atttiute exists between each_char and each_codepoint.

```
str = "a\x80bc"
str.each_char {|c| puts c }
 # no error
str.each_codepoint {|c| puts c }
 # invalid byte sequence in UTF-8 (ArgumentError)
```

The one way of keeping consistency is change ord to return substitute code point such as 0xFFFD adopted by scrub.

Another problem about consitency is surrogate code points. Althouh CRuby allows to use surrogate code points in unicode literal, ord and chr don't allow them.

```
"\uD800".ord
 # invalid byte sequence in UTF-8 (ArgumentError)

0xD800.chr('UTF-8')
 # invalid codepoint 0xD800 in UTF-8 (RangeError)
```

How about remove the restriction? The one example of using surrogate code points is converting a 4-byte character to a pair of 3-byte characters for MySQL/MariaDB's utf8mb3.

```
str = "\u{1F436}" # DOG FACE
cp = str.ord

if cp > 0x10000 then
  # http://unicode.org/faq/utf_bom.html#utf16-4
  lead = 0xD800 - (0x10000 >> 10) + (cp >> 10)
  trail = 0xDC00 + (cp & 0x3FF)
  ret = lead.chr('UTF-8') + trail.chr('UTF-8')
end
```

History

#1 - 01/22/2015 01:12 AM - masakielastic (Masaki Kagaya)

This issue comes from discussion about mruby's behavior (https://github.com/mruby/mruby/issues/2708).

#2 - 01/22/2015 10:19 AM - nobu (Nobuyoshi Nakada)

- Description updated

Masaki Kagaya wrote:

```
str = "a\x80bc"
str.each_char {|c| puts c }
 # no error
```

Sounds like a bug of String#each_char, but maybe intensional.

> The one way of keeping consistency is change ord to return substitute code point such as 0xFFFD adopted by scrub.

Implicit substitution doesn't feel a nice idea to me.

> How about remove the restriction? The one example of using surrogate code points is converting a 4-byte character to a pair of 3-byte characters for MySQL/MariaDB's utf8mb3.

Primarily, it's a responsibility of those bindings.

```
str.encode("UTF-16BE").unpack("v*").pack("U*")
```