Ruby master - Feature #9557

Enumerator#next and Enumerator#peek with argument

02/23/2014 09:14 AM - sawa (Tsuyoshi Sawada)

Status:	Open
Priority:	Normal
Assignee:	
Target version:	

Description

It often happens that I want to move the current index of an enumerator by some arbitrary number n. Enumerator#feed takes the element as the argument, but that cannot be used if the enumerator has duplicate elements, or when I do not have information of a particular element to choose but just want to increment the index by some number. Enumerator#next, on the other hand, has a fixed value 1 to be incremented. It would be convenient if Enumerator#next takes an optional argument that represents the difference of the index to be incremented. The argument can be understood to be defaulted to 1 when absent.

Also, I often want to look not necessarily the current position, but some position away. It would be good if Enumerator#peek takes an optional argument that represents the positional difference to be peeked. The argument can be understood to be defaulted to 0 when absent.

```
enum = [0, 1, 2, 3, 4, 5, 6, 7, 8].to_enum
enum.peek    # => 0
enum.peek(0) # => 0
enum.peek(1) # => 1
enum.peek    # => 0
enum.next    # => 0
enum.next(1) # => 1
enum.next(2) # => 2
enum.peek    # => 4
enum.peek(0) # => 4
enum.peek(1) # => 5
enum.peek    # => 4
enum.next    # => 4
enum.next(1) # => 5
enum.next(2) # => 6
peek         # => 8
```

History

#1 - 02/23/2014 05:13 PM - matz (Yukihiro Matsumoto)

Do you mean peek and next to take *index* as an optional argument?
Some enumerators cannot be rewinded thus it's impossible to index.
If you mean skip not index, some enumerators cannot be peek further elements without modifying the position.

Besides that your description does not fully explain the behavior in your example.
Maybe the example need to be fixed.

Matz.

#2 - 02/23/2014 06:41 PM - sawa (Tsuyoshi Sawada)

> Do you mean peek and next to take index as an optional argument?

No, for peek, I mean the difference between the current position. So if the current position's index is i, then peek(d) looks at the element at index i + d.

And for take, my understanding is that, supposing the current index is i, then take reads whatever at position i and then increments the index to i + 1. My proposal is that when take(d) is given, increment the index to i + d instead of i + 1 after it reads at i.

> Some enumerators cannot be rewinded thus it's impossible to index.
> If you mean skip not index, some enumerators cannot be peek further elements without modifying the position.

I see. Then I would like to ask this feature just for subset of enumerators, those that respond to rewind.

#3 - 04/11/2014 07:00 PM - trans (Thomas Sawyer)

Such features might benefit from an Indexable mixin.