Subject: Re: Request: please make Range classes compatible with Vector!
Posted by piotr5 on Thu, 12 Oct 2017 21:22:36 GMT
View Forum Message <> Reply to Message

thanks for the reminder I'm on a 64-bit system now with pointers taking up 8 bytes each. you're absolutely right it requires some caution, people aren't used to the idea of something like pointers or iterators losing their meaning when the container changes. I too experienced problems in past that got solved by switching from Vector to Array because the data I used wasn't truely moveable... :lol:

I admire the range classes because I tried to implement them myself for a file-management program: the string stores the full pathname, while the range alike class just stores the filename and maybe the directory-name they're in. so for example I'd have a directory with files named [0-9]*.pdf each representing pages scanned, and the directory-name contains the whole book-title and author-name and isbn and so on. the two together are definitely longer than 16 characters, and storing them in some Range class could help in calling some algorithms on them. another use-case I found was to store a 2D object in a Vector<SubRangeClass>, with each element pointing at a row of an image, thereby describing an outline of a (non-rectangular) area selected by the user or an algorithm. again such ranges are definitely not really small, and even if some more compressed method for storing that info would exist, it's still quicker to alter the colours of some container of other container-alike objects than having to navigate in the actual image. (I hate drawing programs which insist that no image can ever be larger than 4GB, what am I supposed to use for high-resolution scans?...)

I agree that the speed-up isn't always noticable. if I'd store a substring in a range-alike class, the cpu-cache must load another 16 bytes in addition to the actual string, quite some waste when the same string is referred to from many places. so even a 32 byte string would see a speed-up in search compared to the SubString class -- unless string and SubString are stored within the same cache-line. but I must emphasize that strings aren't the only use-case that come into mind. I've used various objects as Keys into a Map object. floating-point numbers (or big rational numbers) for example take up quite some space, an std::array of them might determine some coordinates in an n-dimensional space, and I could take a look at a smaller sub-space by selecting a SubRangeClass or ViewRangeClass as a key mapping to further information of that particular place. the idea is, when working with the same vectors over and over again, in different sub-spaces, it might save on cache-consumption if I avoid copying those values...