Document Number: 95-0140/N0740
Date: 7/11/95
Project: Programming Language C++
Reply To: David Dodgson
dsd@tr.unisys.com

Small Changes for Sections 24 & 25

WP Changes

Section	Changes
24.1.4	In table 60 first row, right column, 5th line down, --r == --r implies r should read --r == --s implies r
24.1.6	Move note in paragraph 2 after example in paragraph 3 & 4
24.1.6-5	Change “far” to “__far” Add “__far” to produce the following: inline T* value_type(const T __far*) { return(T __far*)(0); } ... inline long* distance_type(const T __far *) {return (long __far*)(0);}
24.2	Typo: “def inable” should be “definable”
24.3.1.1 box 116	Add “const” to: BidirectionalIterator base() const; Reference operator*() const;
24.3.1.2.5	Returns: should be Returns: *this
24.3.1.2.6	return x.current == y.current;
24.3.1.3	put “};” after : Reference operator[ ](Distance n); remove from previous location at end of section
24.3.1.3 box 117	Reference operator[ ](Distance n); should be Reference operator[ ](Distance n) const;
24.3.1.3-1	Move note to end of section 24.3.1.1
24.3.1.4.5	Returns: should be Returns: *this
24.3.1.4.5+	Add description of the operators: operator+ reverse_iterator<RandomAccessIterator,T,Reference,Distance> operator+ (Distance n) const;

Section	Changes

	Returns: reverse_iterator(current-n)
	
	operator+=
	reverse_iterator<RandomAccessIterator,T,Reference,Distance>
	  operator+= (Distance n);
	Effects: current -= n;
	  return *this;
	
	operator-
	reverse_iterator<RandomAccessIterator,T,Reference,Distance>
	  operator- (Distance n) const;
	Returns: reverse_iterator(current+n)
	
	operator-=
	reverse_iterator<RandomAccessIterator,T,Reference,Distance>
	  operator-= (Distance n);
	Effects: current += n;
	  return *this;

24.3.1.4.6+	Add description of the operators:
	
	template <class RandomAccessIterator, class T, class Reference, class Distance>
	bool operator<(
	  const reverse_iterator<RandomAccessIterator,T,
	  Reference,Distance>& x,
	  const reverse_iterator<RandomAccessIterator,T,
	  Reference,Distance>& y);
	Returns: y.current < x.current;
	
	template <class RandomAccessIterator, class T, class Reference, class Distance>
	Distance operator-(
	  const reverse_iterator<RandomAccessIterator,T,
	  Reference,Distance>& x,
	  const reverse_iterator<RandomAccessIterator,T,
	  Reference,Distance>& y);
	Returns: y.current - x.current;
	
	template <class RandomAccessIterator, class T, class Reference, class Distance>
	reverse_iterator<RandomAccessIterator,T,
	  Reference,Distance> operator+(
	  Distance n,

Section	Changes
	const reverse_iterator<RandomAccessIterator,T, Reference,Distance>& x); Returns: reverse_iterator<RandomAccessIterator,T, Reference, Distance> (x.current - n);
24.4.3	Closing braces should be in normal font
24.4.3.5	Typo: “iterator over” should be “iterate over”
Section 25 beginning, par 5 & 6	Predicate and BinaryPredicate should be described as parameters, not classes (they may be classes, but they shouldn’t be described as such).
25.1.4	pred(i, first2+n) should be pred(*i, *(first2+n))
25.1.9	Fourth version of search( ) should read: ForwardIterator search(ForwardIterator first, ForwardIterator last, Size count, const T& value, BinaryPredicate pred);
25.2.9 25.2.10	Code line after “effects” description should be indented to follow style of rest of document
25.3.2-1	Change comp(*i, *j) to comp(*j, *i) On that same line, the beginning of the line should read: ator in the range [nth, last) it holds...
25.3.3	for binary search, the container must be in order for the binary search to work. Add the assumption that the contents are sorted.
25.3.3.3	Add “without violating the ordering” to first sentence of the “effects” section
25.3.5.2	Set-union Effects: Constructs a sorted union of the elements from the two ranges; that is, the set of elements that are present in one or both of the ranges.
25.3.5.3	Set-Intersection Effects: Constructs a sorted intersection of the elements from the two ranges; that is, the set of elements that are present in both of the ranges.
25.3.5.4	Set-difference Effects: Constructs a sorted difference of the elements from the two ranges; that is, the set of elements that are present in the first range and not present in the second range.
25.3.5.5	Set-symmetric Effects: Constructs a sorted symmetric difference of the elements from the two ranges; that is, the set of elements that are present in one of

Section	Changes
	the ranges but not in both.

25.3.8	Note: One sequence of elements is lexicographically less than another sequence of elements if, in the first pair of elements that compare not equal, the element from the first sequence compares less than the corresponding element from the second sequence. If the sequences have a different number of elements, and all the elements compare equal, the shorter sequence is lexicographically less than the longer sequence.
	`for(i = first1, j = first2;`
	`  i != last1 && j != last2 && !(*i < *j) && !(*j < *i);`
	`  ++i, ++j);`
	`return j == last2 ? false : i == last1 || *i < *j;`