DETAIL
LIST IN C++ STL

(Training materials for students)

Lists are sequence containers that allow non-contiguous memory allocation. As compared to vector, list has slow traversal, but once a position has been found, insertion and deletion are quick. Normally, when we say a List, we talk about doubly linked list. For implementing a singly linked list, we use forward list.Certain functions associated with the vector are:

1. Functions used with List:

- **front**() – Returns the value of the first element in the list.
- **back**() – Returns the value of the last element in the list .
- **push_front**(g) – Adds a new element 'g' at the beginning of the list .
- **push_back**(g) – Adds a new element 'g' at the end of the list.
- **pop_front**() – Removes the first element of the list, and reduces size of the list by 1.
- **pop_back**() – Removes the last element of the list, and reduces size of the list by 1
- **list::begin**() and **list::end**() in C++ STL– begin() function returns an iterator pointing to the first element of the list
- **end**()– **end**() function returns an iterator pointing to the theoretical last element which follows the last element.
- list **rbegin**() and rend() function in C++ STL– **rbegin**() returns a reverse iterator which points to the last element of the list. **rend**() returns a reverse iterator which points to the position before the beginning of the list.
- list **cbegin**() and **cend**() function in C++ STL– **cbegin**() returns a constant random access iterator which points to the beginning of the list. **cend**() returns a constant random access iterator which points to the end of the list.
- list **crbegin**() and crend() function in C++ STL– **crbegin**() returns a constant reverse iterator which points to the last element of the list i.e reversed beginning of container. **crend**() returns a constant reverse iterator which points to the theoretical element preceding the first element in the list i.e. the reverse end of the list.
- **empty**() – Returns whether the list is empty(1) or not(0).
- **insert**() – Inserts new elements in the list before the element at a specified position.
- **erase**() – Removes a single element or a range of elements from the list.
- **assign**() – Assigns new elements to list by replacing current elements and resizes the list.

- **remove**() – Removes all the elements from the list, which are equal to given element.
- **list::remove_if**() in C++ STL– Used to remove all the values from the list that correspond true to the predicate or condition given as parameter to the function.
- **reverse**() – Reverses the list.
- **size**() – Returns the number of elements in the list.
- list **resize**()function in C++ STL– Used to resize a list container.
- **sort**() – Sorts the list in increasing order.
- list **max_size**() function in C++ STL– Returns the maximum number of elements a list container can hold.
- list **unique**() in C++ STL– Removes all duplicate consecutive elements from the list.
- **list::emplace_front**() and **list::emplace_back**() in C++ STL– **emplace_front**() function is used to insert a new element into the list container, the new element is added to the beginning of the list. **emplace_back**() function is used to insert a new element into the list container, the new element is added to the end of the list.
- list::**clear**() in C++ STL– **clear**() function is used to remove all the elements of the list container, thus making it size 0.
- list::**operator**= in C++ STL– This operator is used to assign new contents to the container by replacing the existing contents.
- list::**swap**() in C++ STL– This function is used to swap the contents of one list with another list of same type and size.
- list **splice**() function in C++ STL– Used to transfer elements from one list to another.
- list **merge**() function in C++ STL– Merges two sorted lists into one
- list **emplace**() function in C++ STL– Extends list by inserting new element at a given position.

2. Some Useful Functions:

- **emplace(position, value)**:- This function is used to insert an element at the position specified.
- **emplace_back(value)**:- This function adds value at end of list. It is different from **push_back**() by the fact that it directly creates element at position whereas **push_back**() first makes a temporary copy and copies from there. **emplace_back**() is faster in implementation than **push_back**() in most situations.
- **emplace_front** :- This function adds value at beginning of list. It is different from **push_front**() by the fact that it directly creates element at position whereas **push_front**()

first makes a temporary copy and copies from there. **emplace_front**() is faster in implementation than **push_front**() in most situations.

- **merge(list2)**:- This function is used to merge list2 with list1. If both the lists are in sorted order, then the resulting list is also sorted.
- **remove_if(condition)**:- This function removes the element from list on the basis of condition given in its argument.
- **unique**() :- This function is used to delete the repeated occurrences of the number. List has to be sorted for this function to get executed.
- **splice(position, list2)** :- This function is used to transfer elements from one list into another.
- **swap(list2)** :- This function is used to swap one list element with other.

3. Operations on Forward List:

- **assign**() :- This function is used to assign values to forward list, its another variant is used to assign repeated elements.
- **push_front**() :- This function is used to insert the element at the first position on forward list. The value from this function is copied to the space before first element in the container. The size of forward list increases by 1.
- **emplace_front**() :- This function is similar to the previous function but in this no copying operation occurs, the element is created directly at the memory before the first element of the forward list.
- **pop_front**() :- This function is used to delete the first element of list.
- **insert_after**() This function gives us a choice to insert elements at any position in forward list. The arguments in this function are copied at the desired position.
- **emplace_after**() This function also does the same operation as above function but the elements are directly made without any copy operation.
- **erase_after**() This function is used to erase elements from a particular position in the forward list.
- **remove**() :- This function removes the particular element from the forward list mentioned in its argument.
- **remove_if**() :- This function removes according to the condition in its argument.
- **splice_after**() :- This function transfers elements from one forward list to other.

--- The End ---