Класс ***list*** содержит следующие функции-члены:

**template <class InIter> void assign(InIter start, InIter end);**

Помещает в список последовательность, определяемую параметрами start и end.

**void assign(size\_type num, const T &val);**

Помещает в список num элементов со значением val.

**reference back();**

**const\_reference back() const;**

Возвращает ссылку на последний элемент в списке.

**iterator begin();**

**const\_iterator begin() const;**

Возвращает итератор для первого элемента в списке.

**void clear();**

Удаляет все элементы из списка.

**bool empty() const;**

Возвращает значение истины, если используемый список пуст, и значение лжи в противном случае.

**const\_iterator end() const;**

**iterator end();**

Возвращает итератор для конца списка.

**iterator erase(iterator i);**

Удаляет элемент, адресуемый итератором i, возвращает итератор для элемента, расположенного после удаленного.

**iterator erase(iterator start, iterator end);**

Удаляет элементы в диапазоне, задаваемом параметрами start и end, возвращает итератор для элемента, расположенного за последним удалённым элементом.

**reference front();**

**const\_reference front() const;**

Возвращает ссылку на первый элемент в списке.

**allocator\_type get\_allocator() const;**

Возвращает распределитель списка.

**iterator insert(iterator i, const T &val = T());**

Вставляет значение val непосредственно перед элементом, заданным параметром i, возвращает итератор для этого элемента.

**void insert(iterator i, size\_type num, const T &val);**

Вставляет num копий значения val непосредственно перед элементом, заданным параметром i.

**template <class InIter> void insert(iterator i, InIter start, InIter end);**

Вставляет в список последовательность, определяемую параметрами start и end, непосредственно перед элементом, заданным параметром i.

**size\_type max\_size() const;**

Возвращает максимальное число элементов, которое может содержать список.

**void merge(list<T,Allocator> &ob);**

**template <class Comp> void merge(list<T,Allocator> &ob, Comp cmpfn);**

Объединяет упорядоченный список, содержащийся в объекте ob, с данным упорядоченным списком Результат также упорядочивается. После объединения список, содержащийся в объекте ob, остается пустым. Во второй форме может быть задана функция сравнения, которая определяет, когда один элемент меньше другого.

**void pop\_back();**

Удаляет последний элемент в списке.

**void pop\_front();**

Удаляет первый элемент в списке.

**void push\_back(const T &val);**

Добавляет в конец списка элемент со значением, заданным параметром val.

**void push\_front(const T &val);**

Добавляет в начало списка элемент со значением, заданным параметром val.

**reverse\_iterator rbegin();**

**const\_reverse\_iterator rbegin() const;**

Возвращает реверсивный итератор для конца списка.

**reverse\_iterator rend();**

**const\_reverse\_iterator rend() const;**

Возвращает реверсивный итератор для начала списка.

**void remove(const T &val);**

Удаляет из списка элементы со значением, заданным параметром val.

**template <class UnPred> void remove\_if(UnPred pr);**

Удаляет элементы, для которых унарный предикат pr равен значению true.

**void resize(size\_type num, const T &val = T());**

Устанавливает емкость списка равной не менее заданного значения num, если вектор для этого нужно удлинить, то в его конец добавляются элементы со значением, заданным параметром val.

**void reverse();**

Реверсирует список.

**size\_type size() const;**

Возвращает текущее количество элементов в списке.

**void sort();**

**template <class Comp> void sort(Comp cmpfn);**

Сортирует список. Вторая форма сортирует список с помощью функции сравнения cmpfn, чтобы определять, когда один элемент меньше другого.

**void splice(iterator i, list<T,Allocator> &ob);**

Вставляет содержимое списка ob в данный список в позиции, указанной итератором i, После выполнения этой операции список ob остается пустым.

**void splice(iterator i, list<T,Allocator> &ob, iterator el);**

удаляет из списка ob элемент, адресуемый итератором el, и сохраняет его в позиции, адресуемой итератором i.

**void splice(iterator i, list<T,Allocator> &ob, iterator start, iterator end);**

удаляет из списка ob диапазон, определяемый параметрами start и end, и сохраняет его в данном списке, начиная с позиции, адресуемой итератором i.

**void swap(list<T,Allocator> &ob);**

Выполняет обмен элементами данного списка и списка ob.

**void unique();**

**template <class BinPred> void unique(BinPred pr);**

Удаляет из списка элементы-дубликаты. Вторая форма для определения уникальности использует предикат pr.