**NAME**

w_list_t – generic list structures

**SYNOPSIS**

```
#include <w.h>
#include <w_list.h>
// for definition of offsetof(x,y):
#include <stddef.h>

class w_link_t;
class T {
    // your type
    ...
    w_link_t    _link;
    ...
};

// unsorted lists:
template <class T> class w_list_t;

// iterator over a w_list_t:
template <class T> class w_list_i;
template <class T> class w_list_const_i;

// sorted lists:
template <class T, class K> w_descend_list_t;
template <class T, class K> w_ascend_list_t;
```

**DESCRIPTION**

This is a set templates for managing doubly-linked lists of objects of type **T** (for user-defined types T).  The double-linking of items into lists is accomplished with the class *w_link_t*, which must be a member of the type T.  The template methods operate on the *w_link_t* member.  The name of the member can be anything; the methods locate the member by information given when the list is constructed:

```
w_list_t<mytype> l(offsetof(mytype, mylink));
```

where the template parameter is the type:

```
class mytype {
    int             a;
    w_link_t        mylink;
    ...
    int             b;
};
```

The lists managed by these templates are of two general kinds: unsorted and sorted.

**Unsorted lists**

Unsorted lists ( **w_list_t<T>** are constructed as in the example given above.

Items are put into the list with the any of the following methods:

```
w_list_t<T>&    push(T* t);
T*              pop();    // reverse of push
w_list_t<T>&    append(T* t) ;
T*              chop();    //reverse of append
```

These methods return the objects at the front and rear of the lists:

```
T*      top();
T*      bottom();
```

A list can be printed with

```
friend ostream&                operator<<(
    ostream&                       o,
    const w_list_t<T>&             l);
```

Unsorted lists are traversed with iterators (instances of **w_list_i<T>** which have methods **next(), curr(),** and **reset().**

```
{
    mytype     *p;
    w_list_i<mytype> iter(l);
    for (int i = 0; i < 10; i++)  {
        p = iter.next();
        if ( p->a == ....  // whatever you wish
    }
}
```

**Sorted lists**

Sorted lists ( **w_descend_list_t<T,K>** and **w_ascend_list_t<T,K>** are lists of objects containing **keys**, which are members of the template parameter class T, whose type is the template parameter type K.

Sorted lists are traversed by calling their methods

```
virtual T *search(const K &);
```

In order for the method **search** to work, the template has to find a member of the key type **K** in each instance of type T that is in the list. For this reason, each ordered list is constructed with the location of the key as well as the location of the **w_link_t:**

```
// order this list on the value of a in descending order
w_descend_list_t<mytype, int> l(
        offsetof(mytype, a),
        offsetof(mytype, mylink) // offset of link
);

// order this list on the value of b in ascending order
w_ascend_list_t<mytype, int> l(
        offsetof(mytype, b),
        offsetof(mytype, mylink) // offset of link
);
```

Objects are inserted into sorted lists with the method **put_in_order:**

```
virtual void   put_in_order(T* t);
```

Sorted lists can be traversed with iterators ( **w_list_i** and with these methods:

```
T*     first();
T*     last();
```

In addition, methods derived from **w_list_i** such as **pop** can be used to remove items from the head list (e.g, for destroying the entire list).

### DERIVING NEW LISTS
The methods **search** and **put_in_order** are declared virtual so that other list types can be derived from these templates.

## BUGS
There are no methods for removing items from the middle of a list.

## VERSION
This manual page applies to Version 2.0 of the Shore Storage Manager.

## SPONSORSHIP
The Shore project is sponsored by the Advanced Research Project Agency, ARPA order number 018 (formerly 8230), monitored by the U.S. Army Research Laboratory under contract DAAB07-91-C-Q518. Further funding for this work was provided by DARPA through Rome Research Laboratory Contract No. F30602-97-2-0247.

## COPYRIGHT
Copyright (c) 1994-1999, Computer Sciences Department, University of Wisconsin -- Madison. All Rights Reserved.

## SEE ALSO
**rc(fc), intro(fc), statistics(oc), statistics(svas),** and **statistics(ssm).**