List

• `Create`	List<T> LIST
• `Insert`	add a element into list
• `Assign`	assign all of the list element
• `Erase`	delete a element in the list

Example: Insert, create

```
#include <list>
#include <iostream>
using namespace std ;
typedef list<int> LISTINT;
int main()
{
    LISTINT listInt;
    LISTINT::iterator i;

    // Insert one at a time
    listInt.insert (listInt.begin(), 2);
    listInt.insert (listInt.begin(), 1);
    listInt.insert (listInt.end(), 3);

    // 1 2 3
    cout << "lintInt:";
    for (i = listInt.begin(); i != listInt.end(); i++)
    {
        if(*i == 2)
        // Insert 3 fives before value of 2
        listInt.insert(i, 3, 5);
        cout << " " << *i;
    }
    cout << endl;

    //1 5 5 5 2 3
    // Insert 3 fours
    listInt.insert (listInt.end(), 3, 4);

    // 1 5 5 5 2 3 4 4 4
    cout << "lintInt:";
    for (i = listInt.begin(); i != listInt.end(); ++i)
        cout << " " << *i;
    cout << endl;
}
```

assign empty erase

```
#include <list>
#include <iostream>
using namespace std ;
typedef list<int> LISTINT;
int main()
{
    LISTINT listOne;
    LISTINT listAnother;
    LISTINT::iterator i;

    // Add some data
    listOne.push_front (2);
    listOne.push_front (1);
    listOne.push_back (3);
    listAnother.push_front(4);
    listAnother.assign(listOne.begin(), listOne.end());

    // 1 2 3
    for (i = listAnother.begin();
         i != listAnother.end(); ++i)
        cout << *i << " ";
    cout << endl;
    listAnother.assign(4, 1);

    // 1 1 1 1
    for (i = listAnother.begin();
         i != listAnother.end(); ++i)
        cout << *i << " ";
    cout << endl;
    listAnother.erase(listAnother.begin());

    // 1 1 1
    for (i = listAnother.begin();
         i != listAnother.end(); ++i)
        cout << *i << " ";
    cout << endl;

    listAnother.erase(listAnother.begin(),
                      listAnother.end());
    if (listAnother.empty())
        cout << "All gone\n";
}
```

Queue

• `Push`	append a element into the queue
• `Front`	return the first element in the queue
• `Pop`	remove the first element in the queue
• `Back`	return the last element in the dequeue
• `Size`	return the length of the queue

Example

```
#include <iostream>
#include <queue>
#include <deque>
using namespace std ;

// Using queue with list
typedef queue<int>  INTQUEUE;

// Using queue with deque
typedef deque<char*> CHARDEQUE;
typedef queue<char*> CHARQUEUE;

int main(void)
{
    size_t size_q;
    INTQUEUE q;
    CHARQUEUE p;

    // Insert items in the queue(uses list)
    q.push(42);
    q.push(100);
    q.push(49);
    q.push(201);

    // Output the size of queue
    size_q = q.size();
    cout << "size of q is:" << size_q << endl;

    // Output items in queue using front()
    // and use pop() to get to next item until
    // queue is empty
    while (!q.empty())
    {
        cout << q.front() << endl;
        q.pop();
    }

    // Insert items in the queue(uses deque)
    p.push("cat");
    p.push("ape");
    p.push("dog");
    p.push("mouse");
    p.push("horse");

    // Output the item inserted last using back()
    cout << p.back() << endl;

    // Output the size of queue
    size_q = p.size();
    cout << "size of p is:" << size_q << endl;

    // Output items in queue using front()
    // and use pop() to get to next item until
    // queue is empty
    while (!p.empty())
    {
        cout << p.front() << endl;
        p.pop();
    }
}
```

Stack

- `top` returns the top element of the stack.
- `empty` returns true if the stack has 0 elements.
- `push` add a element in the top
- `Pop` get out of the top of stack

```
#include <stack>
#include <iostream>
using namespace std ;
typedef stack<int> STACK_INT;
int main()
{
   STACK_INT stack1;
   cout << "stack1.empty() returned " <<
      (stack1.empty()? "true": "false") << endl;
   cout << "stack1.push(2)" << endl;
   stack1.push(2);
   if (!stack1.empty())
      cout << "stack1.top() returned " <<
      stack1.top() << endl;
   stack1.push(5);
   stack1.push(11);
   if (!stack1.empty())
      cout << "stack1.top() returned " <<
      stack1.top() << endl;
   // Modify the top item. Set it to 6.
   // Repeat until stack is empty
   while (!stack1.empty()) {
      const int& t=stack1.top();
      cout << "stack1.top() returned " << t << endl;
      cout << "stack1.pop()" << endl;
      stack1.pop();
   }
}
```

Reference

- Google
- Microsoft
  http://msdn.microsoft.com/en-us/library/f1dtts6s.aspx