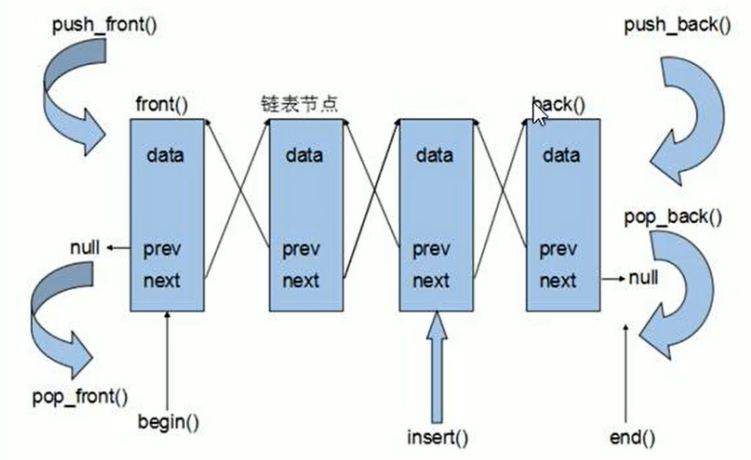
list链表基本概念

STL中的链表是一个双向循化链表

优点：可以对任意位置进行插入和删除数据

缺点：

- 容器遍历速度慢，没有数组快（不便于查找）
- 占用空间比数组大（还有指针域）



注意：由于链表的存储方式并不是连续的内存空间，因此list中的迭代器只支持前移和后移，属于双向迭代器。(不能跳跃式的访问)

基本操作：

创建容器：

```
list<int> L1; //普通构造
list<int> L2(L1.begin(),L1.end()); //区间构造
```

添加数据：

```
L.push_back(x);
L1.swap(L2); //交换L1和L2
L1.size(); //容器大小
L1.empty(); //list判空
```

插入和删除：

- push_back(elem);//在容器尾部加入一个元素
- pop_back();//删除容器中最后一个元素
- push_front(elem);//在容器开头插入一个元素
- pop_front();//从容器开头移除第一个元素
- insert(pos,elem);//在pos位置插elem元素的拷贝，返回新数据的位置。
- insert(pos,n,elem);//在pos位置插入n个elem数据，无返回值。
- insert(pos,beg,end);//在pos位置插入[beg,end)区间的数据，无返回值。
- clear();//移除容器的所有数据
- erase(beg,end);//删除[beg,end)区间的数据，返回下一个数据的位置。
- erase(pos);//删除pos位置的数据，返回下一个数据的位置。
- remove(elem);//删除容器中所有与elem值匹配的元素。

注意：

- 插入的是迭代器

```
//insert插入
list<int>::iterator it = L.begin();
L.insert(++it, 1000);
//  200 1000 100 10 20
```

- 删除的也是迭代器

```
//删除
it = L.begin();
L.erase(++it);
// 200 100 10 20
```

- 移除（删除所有值为1000的数据）

```
//移除
L.push_back(10000);
L.push_back(10000);
L.push_back(10000);
L.push_back(10000);
printList(L);
L.remove(10000);
printList(L);
```

- 排序

下面是默认升序排序：

```
//排序
cout << "排序前：  " << endl;
printList(L1);

//所有不支持随机访问迭代器的容器，不可以用标准算法
//不支持随机访问迭代器的容器，内部会提供对应一些算法
//sort(L1.begin(), L1.end());

L1.sort();
cout << "排序后：  " << endl;
printList(L1);
```

下面是降序排序操作：

1. 先写一个仿函数

```
bool myCompare(int v1 ,int v2)
{
    //降序  就让第一个数 > 第二个数
    return v1 > v2;
}
```

2.再将仿函数名放入sort中

```
L1.sort(myCompare);
printList(L1);
```

这样就是降序排序了。

自定义数据类型的排序

```
//指定排序规则
bool comparePerson(Person &p1,Person &p2)
{
    //按照年龄 升序
    if (p1.m_Age == p2.m_Age)
    {
        //年龄相同  按照身高降序
        return p1.m_Height > p2.m_Height;
    }
    else
    {
        return  p1.m_Age < p2.m_Age;
    }

}
```