list基本概念

功能：

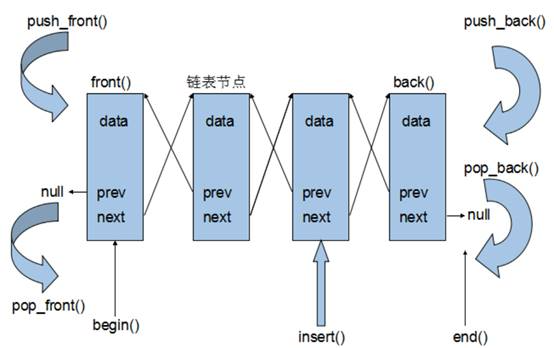
将数据进行链式存储

链表（list）是一种物理存储单元上非连续的存储结构，数据元素的逻辑顺序是通过链表中的指针链接实现的

链表的组成：链表由一系列结点组成

结点的组成：一个是存储数据元素的数据域，另一个是存储下一个结点地址的指针域

STL中的链表是一个双向循环链表



list的优点：

采用动态存储分配，不会造成内存浪费和溢出

链表执行插入和删除操作十分方便，修改指针即可，不需要移动大量元素

list的缺点：

链表灵活，但是空间(指针域) 和 时间（遍历）额外耗费较大

List有一个重要的性质，插入操作和删除操作都不会造成原有list迭代器的失效，这在vector是不成立的。

总结：STL中List和vector是两个最常被使用的容器，各有优缺点

list构造函数

功能描述：

创建list容器

函数原型：

list<T> lst; //list采用采用模板类实现,对象的默认构造形式：

list(beg,end); //构造函数将[beg, end)区间中的元素拷贝给本身。

list(n,elem); //构造函数将n个elem拷贝给本身。

list(const list &lst); //拷贝构造函数。

总结：list构造方式同其他几个STL常用容器，熟练掌握即可。

list 大小操作

功能描述：

对list容器的大小进行操作

函数原型：

size(); //返回容器中元素的个数

empty(); //判断容器是否为空

resize(num); //重新指定容器的长度为num，若容器变长，则以默认值填充新位置。

​ //如果容器变短，则末尾超出容器长度的元素被删除。

resize(num, elem); //重新指定容器的长度为num，若容器变长，则以elem值填充新位置。

总结：

判断是否为空 --- empty

返回元素个数 --- size

重新指定个数 --- resize

list 插入和删除

功能描述：

对list容器进行数据的插入和删除

函数原型：

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。

总结：

尾插 --- push\_back

尾删 --- pop\_back

头插 --- push\_front

头删 --- pop\_front

插入 --- insert

删除 --- erase

移除 --- remove

清空 --- clear

list 反转和排序

功能描述：

将容器中的元素反转，以及将容器中的数据进行排序

函数原型：

reverse(); //反转链表

sort(); //链表排序

总结：

反转 --- reverse

排序 --- sort （成员函数）

**注意：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**

案例描述：

将Person自定义数据类型进行排序，Person中属性有姓名、年龄、身高

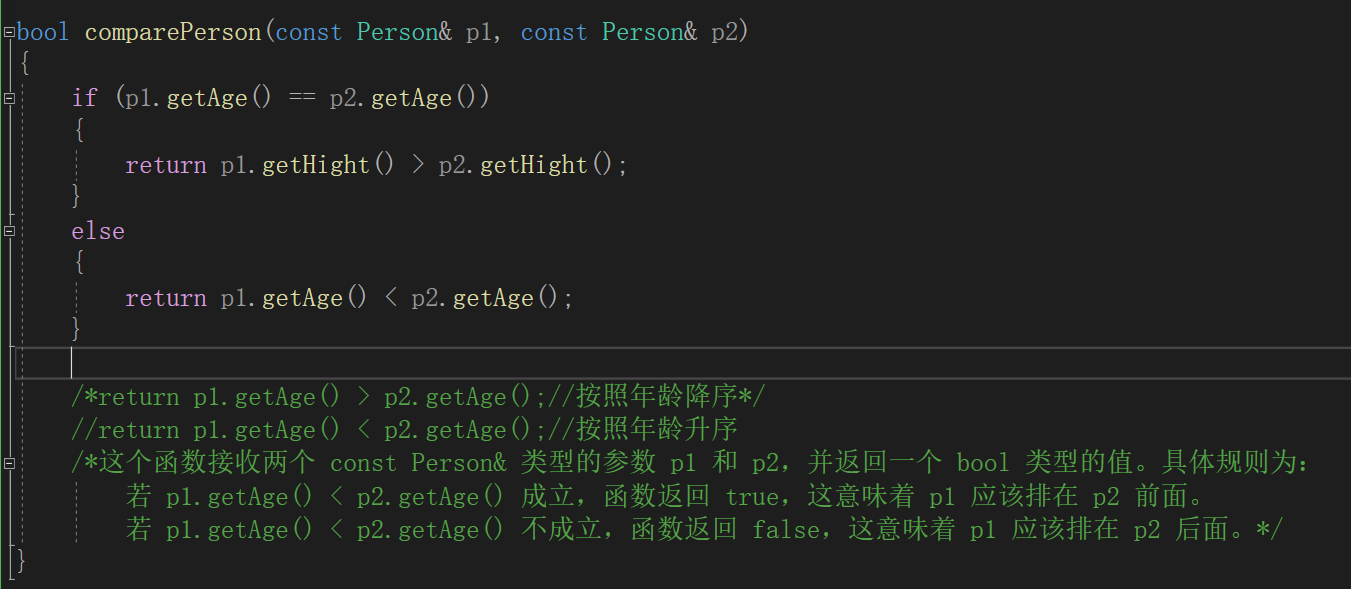
排序规则：

按照年龄进行升序，如果年龄相同按照身高进行降序

总结：

对于自定义数据类型，必须要指定排序规则，否则编译器不知道如何进行排序

高级排序只是在排序规则上再进行一次逻辑规则制定，并不复杂





**重点在于如何自定义排序规则**