附件3:

**广东外语外贸大学南国商学院**

**实验报告**

**课程名称： 数据结构**

**实验项目： 顺序表的基本操作**

**实验室： 个人计算机**

**任课教师： 苗磊**

**学生姓名和学号： 曾昱铨 2420601440**

**专业和班级： 计算机科学与技术4班**

**实验时间： 2024 年 10 月 7 日**

**报告提交时间： 2024 年 10 月 7 日**

**广东外语外贸大学南国商学院教务处 印制**

**学生姓名：**   **实验报告成绩：**

**评语：**

**任课教师（签名）**

**年 月 日**

说明：指导教师评分后，实验报告交相关教学单位保存。

**实验项目：顺序表的基本操作**

**一、实验目的**

此实验的主要目的是实现顺序表的基本操作，其中有顺序表的初始化顺序表、创建顺序表、插入数据元素、删除数据元素、查找数据元素（分为按值查找和按元素位置查找）等操作。通过操作去实现顺序表的基本功能，从而来解决现实中的需求。同时也研究线性表的特性。

**二、实验原理**

顺序表是一种常见的数据结构，它的特性是除头结点和尾结点外，其余结点均有且仅有一个前驱结点和后继结点，这样的一对一线性结构是顺序表的特点。在此特点上，我们将对顺序表进行增加（插入）结点、删除结点、查找等操作，同时也研究这些操作对于该表中其他结点的影响。

**三、实验条件（设备与软件等）**

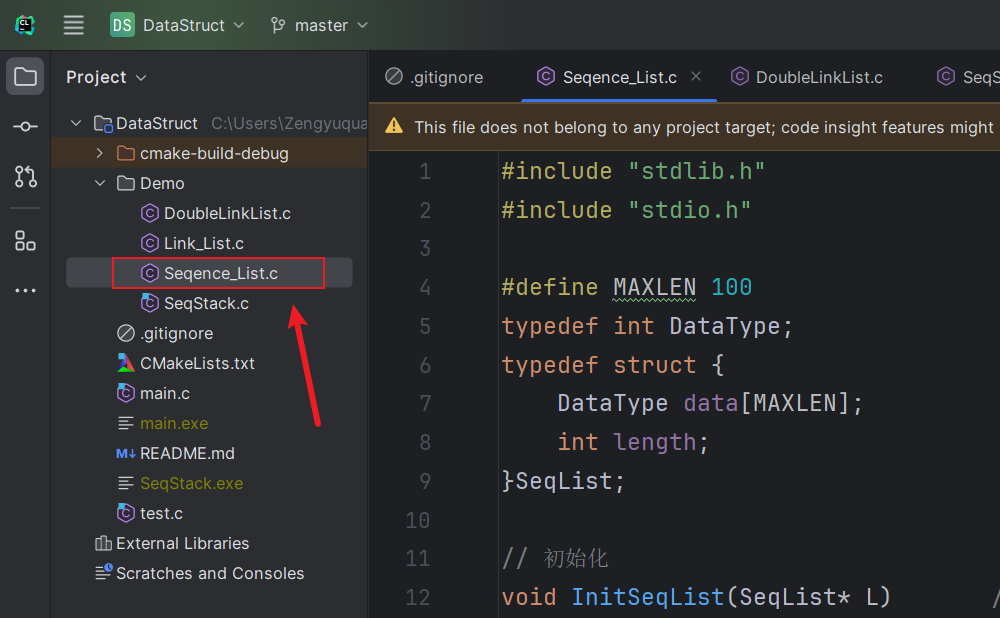
设备：个人计算机

软件：CLion编辑器的C语言坏境（windows）。

**四、实验步骤**

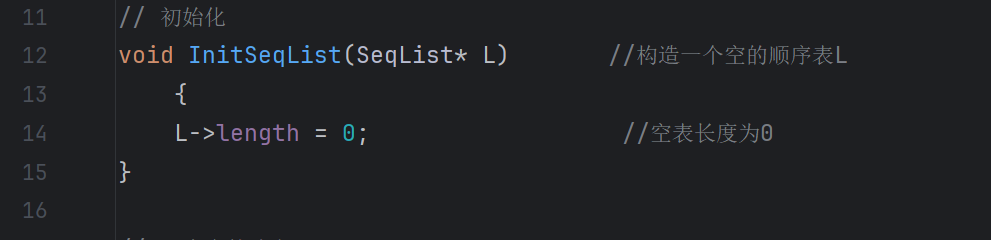
4.1 准备工作。

创建一个DataStruct的项目，并创建一个Sequence\_List.c文件作为实验的主要文件。



4.2 顺序表的初始化。

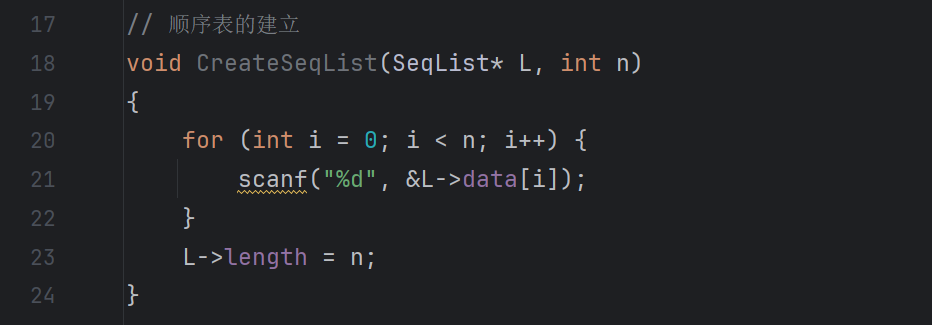
在创建顺序表之前，我们需要初始化顺序表，以方便后续操作。



1、定义一个InitSeqList()函数，形参是SeqList类型的指针L。

2、把顺序表的长度设置为0，表示顺序表存在，但目前没有参数，为空。

4.3 顺序表的建立。



1、定义一个CreateSeqList()函数，形参是SeqList类型的指针L和顺序表的长度n。

2、利用for循环，从键盘输入数据到顺序表；利用i的自增，从0开始到n-1结束，添加到顺序表中。

3、将输入的长度n赋值给length，来记录顺序表实时的长度。

4.4 顺序表的查找。

顺序表的查找分为按元素位置查找和按值查找，实现我们要介绍的是按元素位置查找。



1、定义一个GetElem()函数，形参是SeqList类型的指针L和希望查找的位置i。

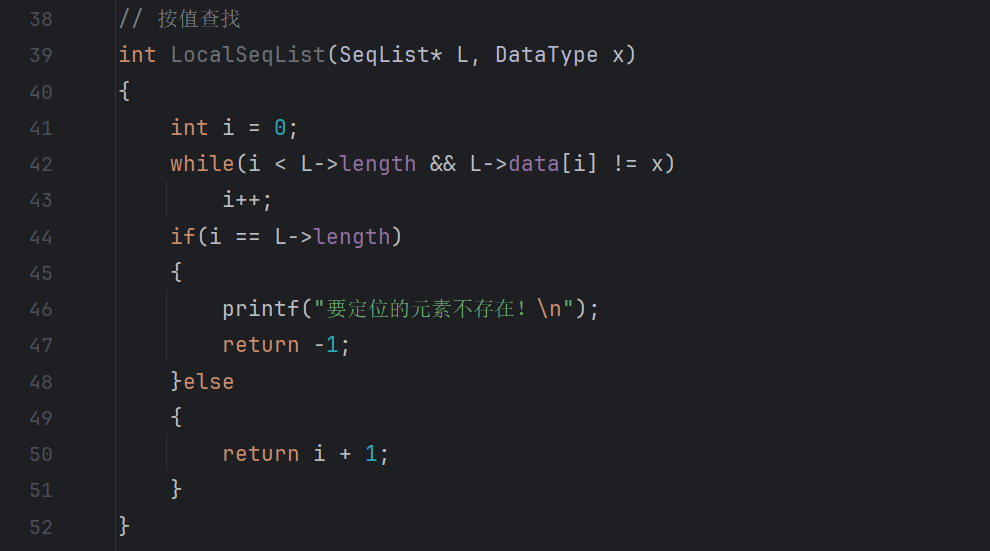
2、利用if判断查找的位置是否合法，其合法范围是i大于或等于1且1小于或等于length，即顺序表实时的长度。所以if的判断条件是



若查找的位置在范围内。则打印“位置错误！”，并执行exit(0)退出程序。

3、如果查找的位置合法，则返回i-1位置的元素；因为下标位置是从0开始的，所以i要减一才能准确表示出该位置的元素。

接下来的是按值查找：



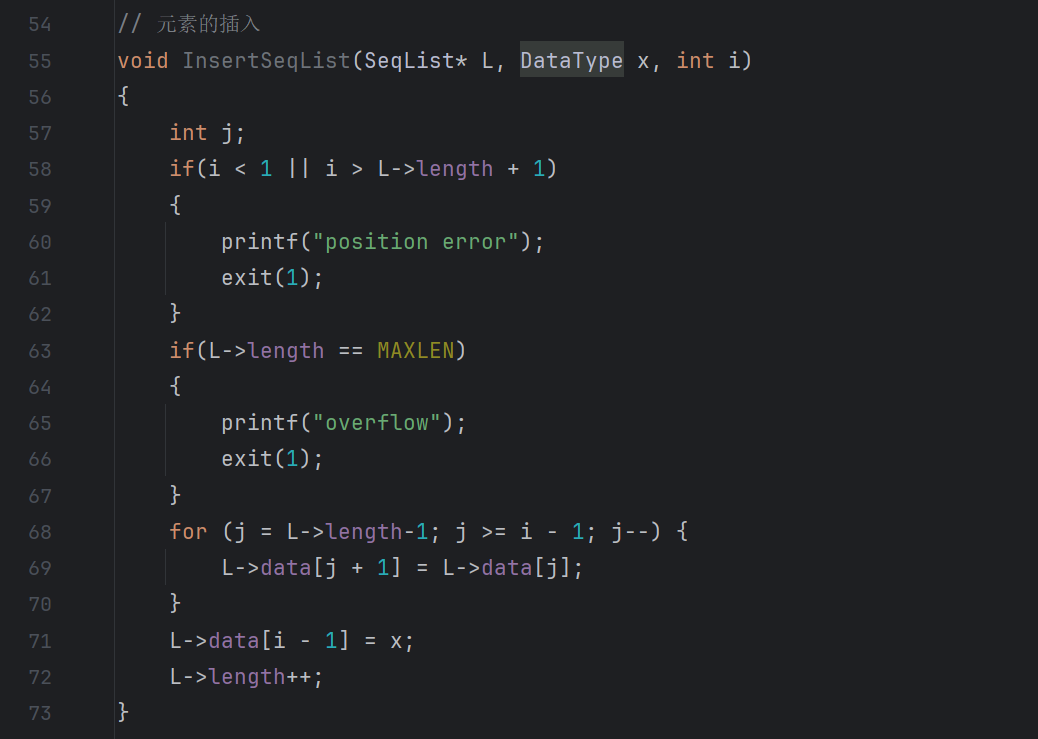
1、定义一个LocalSeqList()函数，形参是SeqList类型的指针L和希望在顺序表中查找的值x。

2、定义一个int型变量i，利用while找出期望的值与顺序表中值的匹配情况；其查找的范围是i小于顺序表实时的长度，且判断顺序表中的元素是否匹配希望查找的值。若不匹配顺序表中的元素，则i++自增。

3、利用if判断i的情况，若i == L->length，则表示while循环到顺序表实时的长度，即顺序表中并没有匹配的值；打印“要定位的元素不存在！”返回-1，表示该元素不存在。

4、若在while循环中匹配到元素，则返回i+1的值；因为i为顺序表中的下标，而下标从0开始，要加一才能表示该数据元素所在的位置。

4.5 顺序表元素的插入。



1、定义一个InsertSeqList()函数，形参是SeqList类型的指针L和希望插入的值x，希望插入的位置i。

2、利用if判断插入的位置是否合法，其合法范围是i大于或等于1且1小于或等于length+1，即顺序表实时的长度。所以if的判断条件是



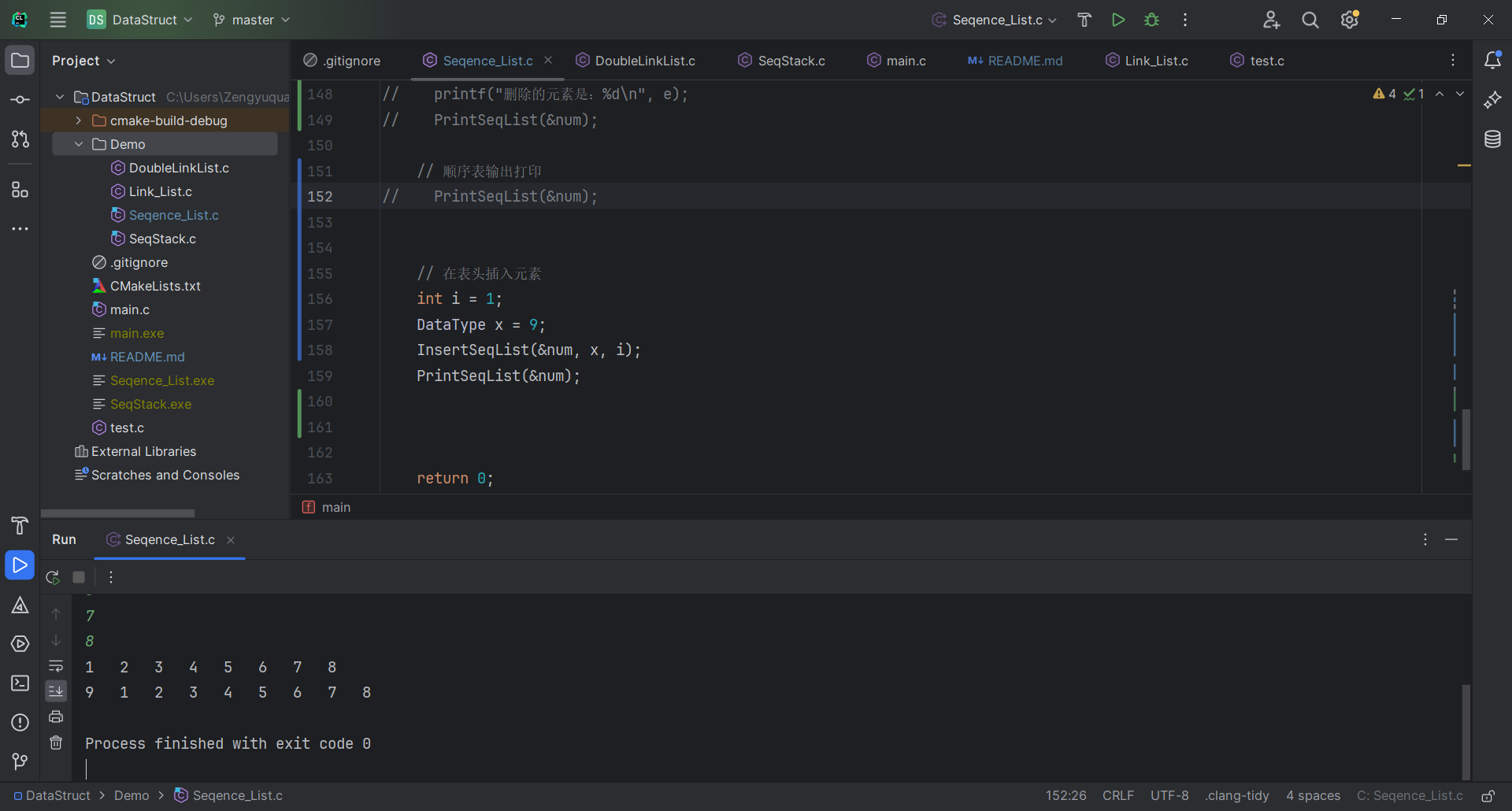
若插入的位置在范围内。则打印“position error”，并执行exit(1)退出程序。

3、第二个if判断顺序表实时的长度是否为MAXLEN顺序表最大长度，若为顺序表最大长度，则无法在插入元素（溢出），打印“overflow”，并执行exit(1)退出程序。

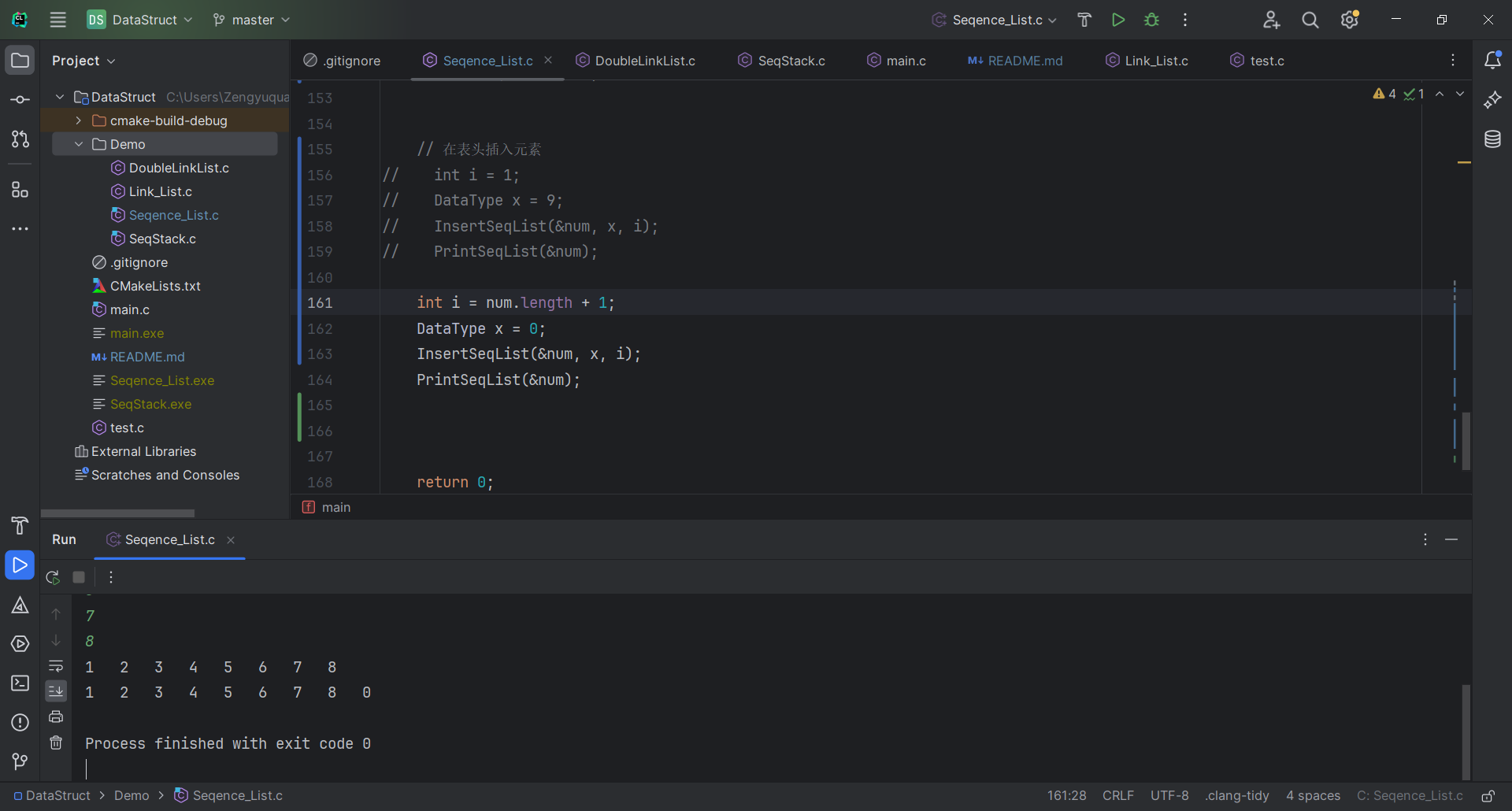
4、定义一个类型为int的变量j，并赋值为L -> length（顺序表实时的长度）- 1,表示最后一个元素的下标；利用for循环，判断条件是j >= i - 1，即到希望插入的位置的下标，j--自减。从顺序表实时的长度最后一个元素到希望插入的位置i的元素，依次往后移动一位：L->data[j + 1] = L->data[j];后，空出i-1的下标为插入元素x： L->data[i - 1] = x;

5、最后顺序表实时的长度自增加一。

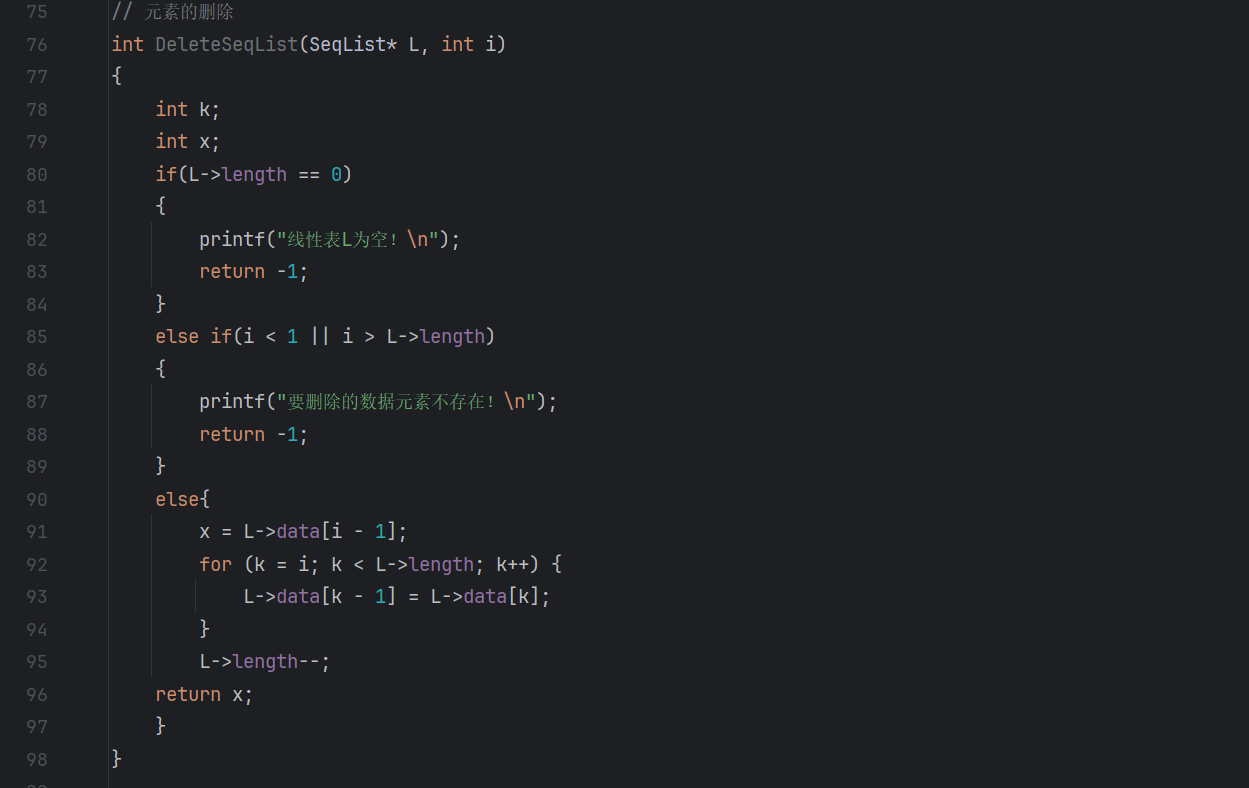
在表头插入元素：



在表尾插入元素：



4.6 顺序表的删除。



1、定义一个DeleteSeqList()函数，形参是SeqList类型的指针L和希望删除的位置i。

2、利用if循环判断顺序表实时的长度是否等于0，若等于0；则表示线性表为空，不能进行删除存操作；打印“线性表L为空！”返回-1，表示删除失败。

3、else if再判断删除的位置是否合法，其合法范围是i大于或等于1且1小于或等于length，即顺序表实时的长度。所以if的判断条件是



若删除的位置在范围内。则打印“要删除的数据元素不存在！”，并返回-1，表示删除失败。

4、定义一个类型为int的变量x来存储删除的元素值。

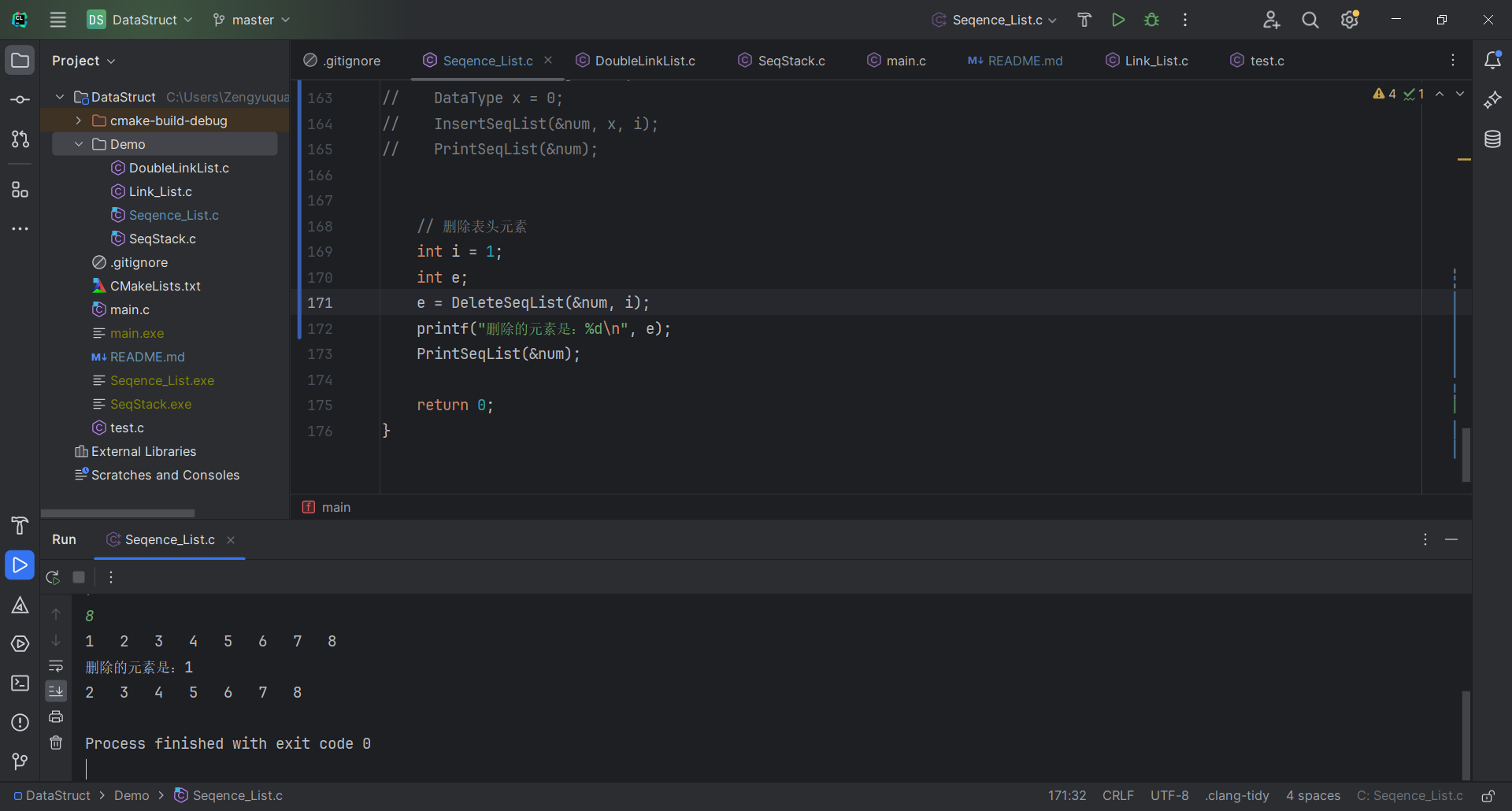
5、如都不为非法条件则，定义一个类型为int的变量k，并赋值为i，即i为删除元素位置的下一个元素，利用for循环，判断条件是：

k < L->length，即到顺序表实时的长度最后一个元素，k++自增。从k开始，即删除元素位置的下一个元素，往前移动，覆盖删除元素的位置，以此类推到顺序表实时的长度最后一个元素：L->data[k - 1] = L->data[k];

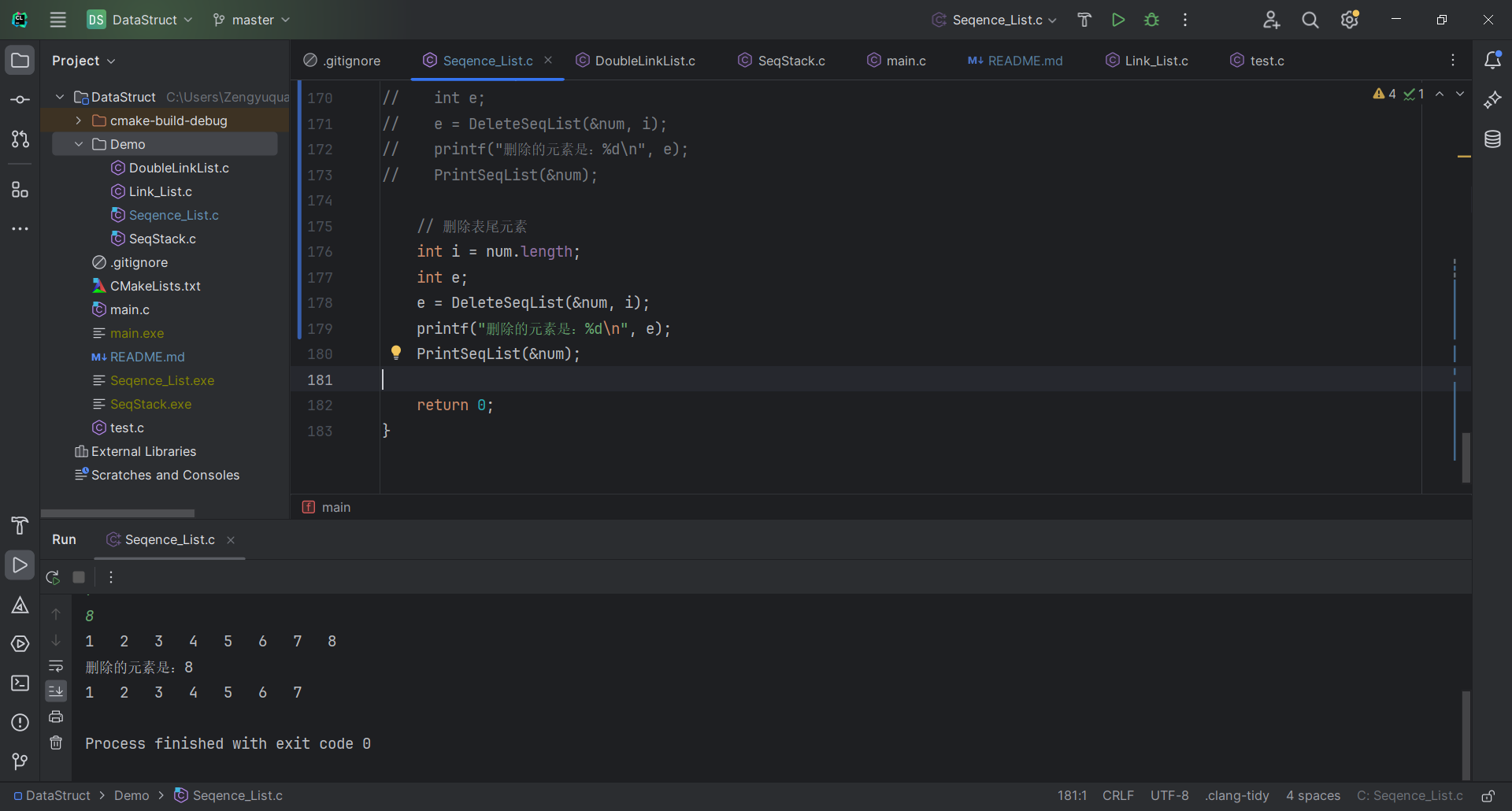
6、5、最后顺序表实时的长度自减减一。

7、返回x，删除的元素值。

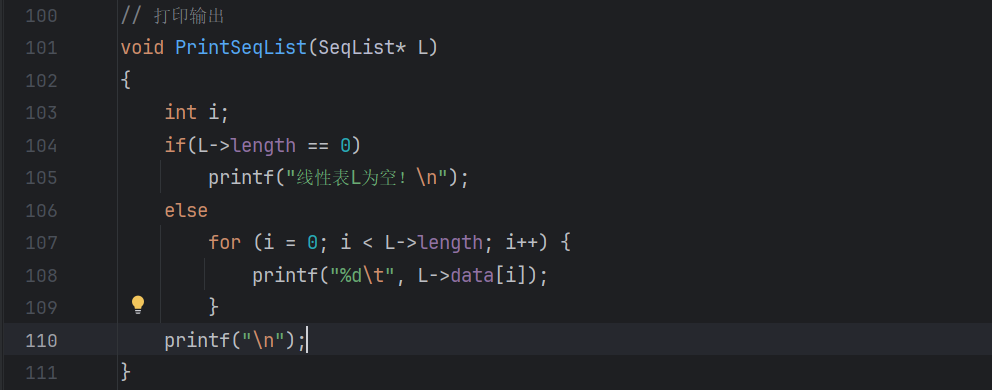
删除表头元素：



删除表尾元素：



4.7 顺序表元素的输出打印



1、定义一个PrintSeqList()函数，形参是SeqList类型的指针L。

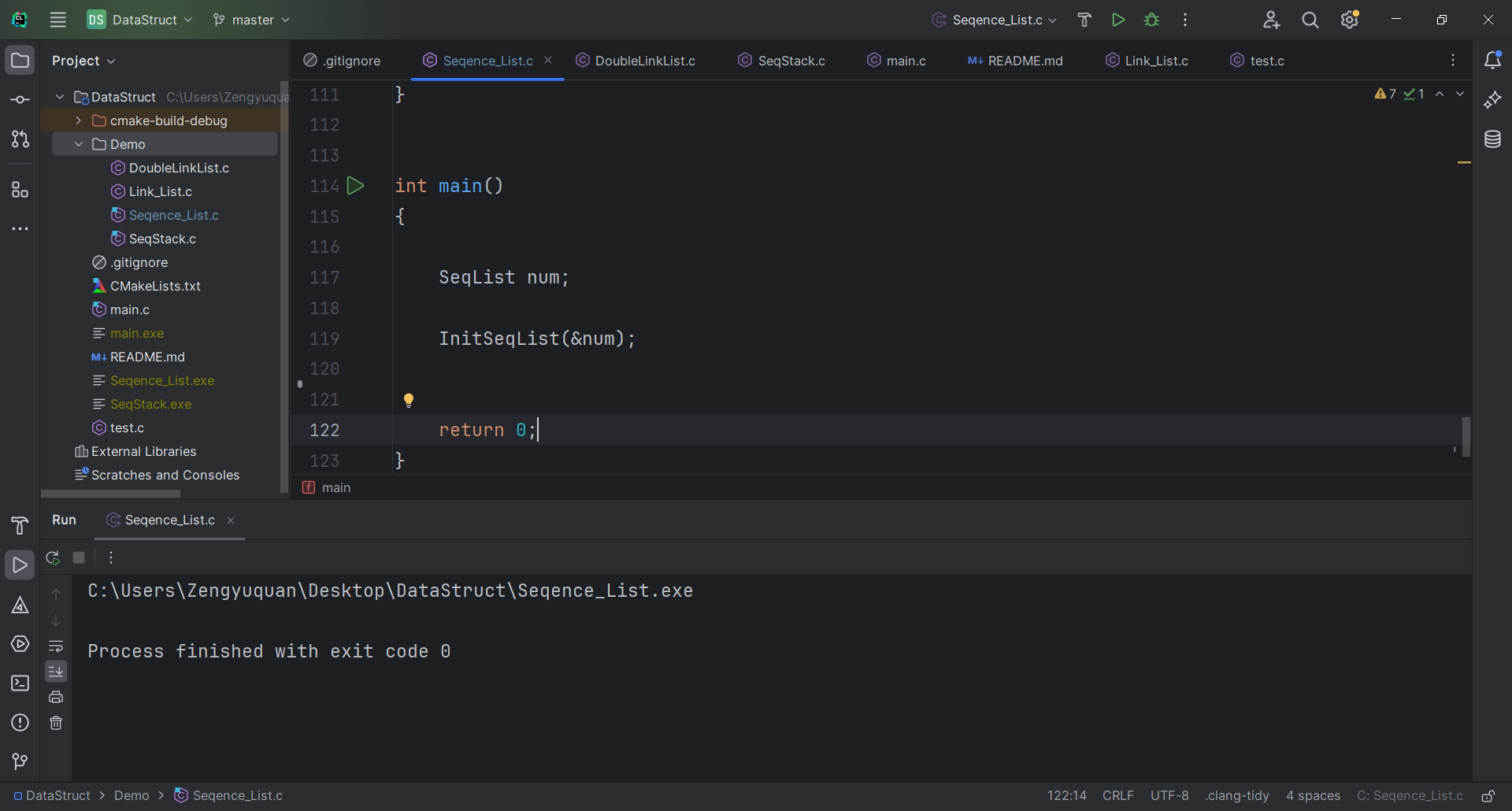
2、利用if判断顺序表是否为空，若为空，则无法输出打印元素，打印”线性表L为空！“。

3、若顺序表不为空，则用for循环遍历出顺序表中的值。

**五、实验结果**

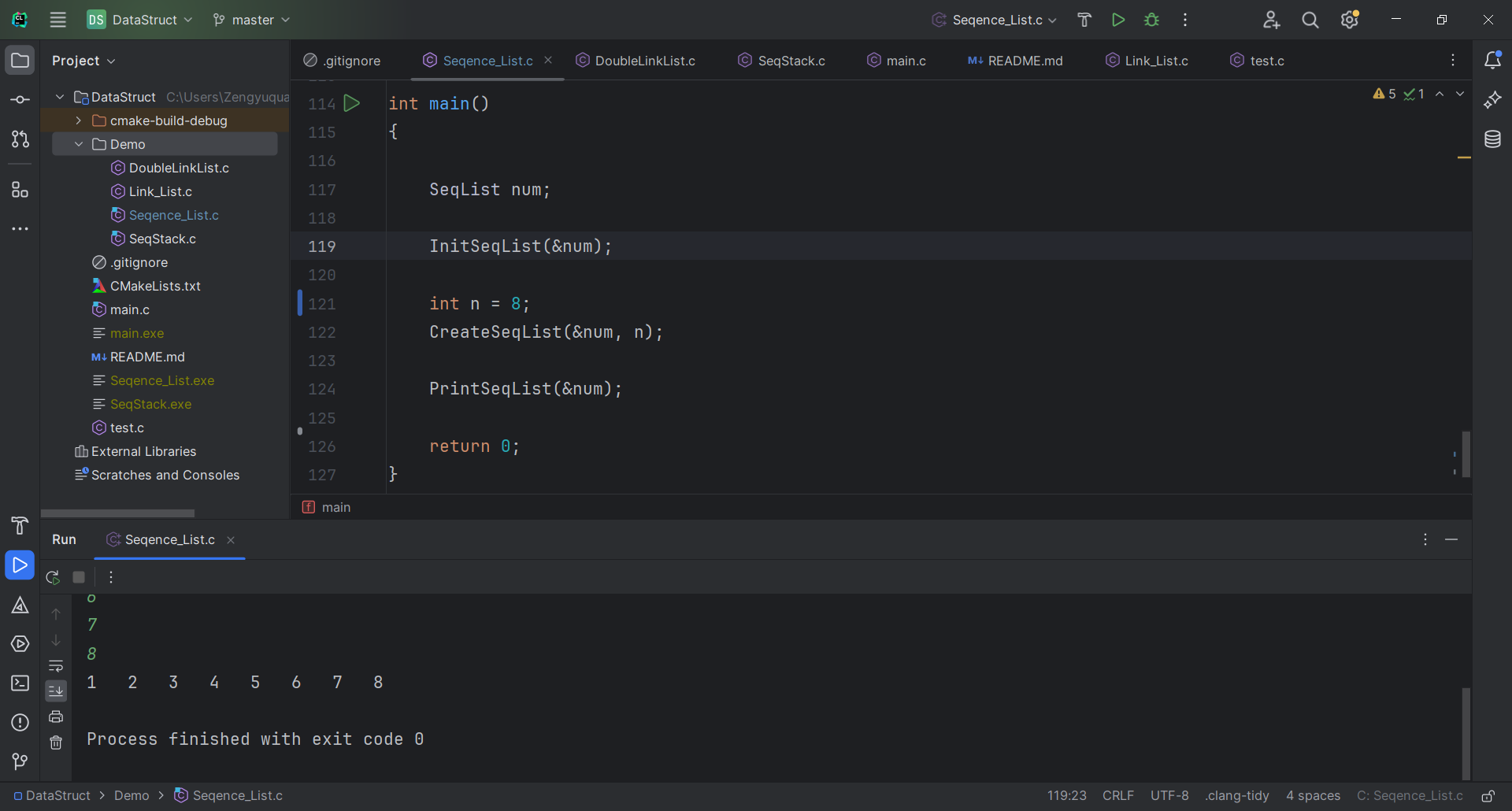
5.1 初始化顺序表操作的实现。

初始化一个名为num的顺序表。



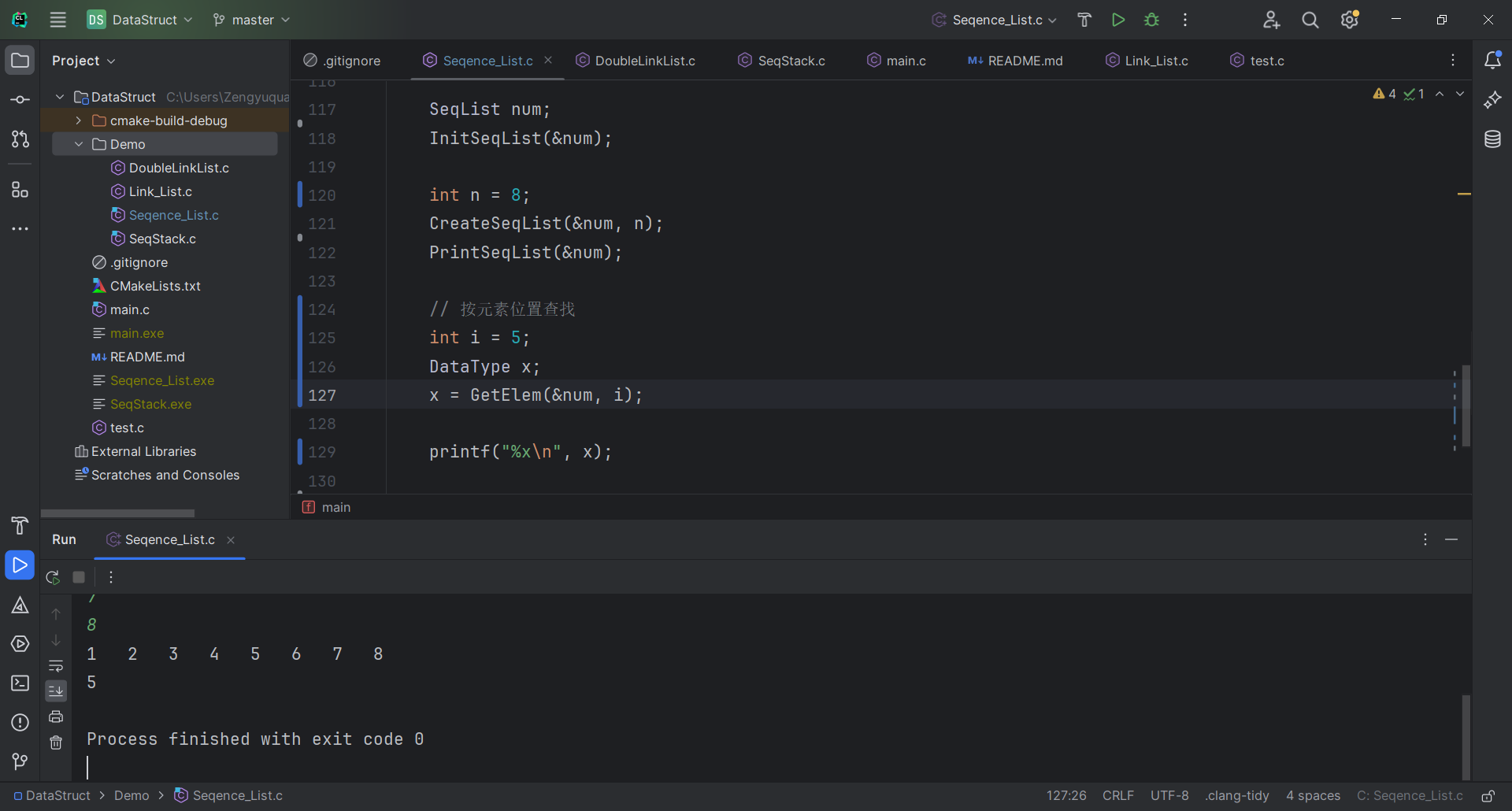
5.2 顺序表的创建操作的实现。

创建一个长度为8的顺序表，将num的地址指针和顺序表长度为8作为实参传递给CreateSeqList()函数。并在键盘输入1到8的数据到线性表中，最后打印线性表。



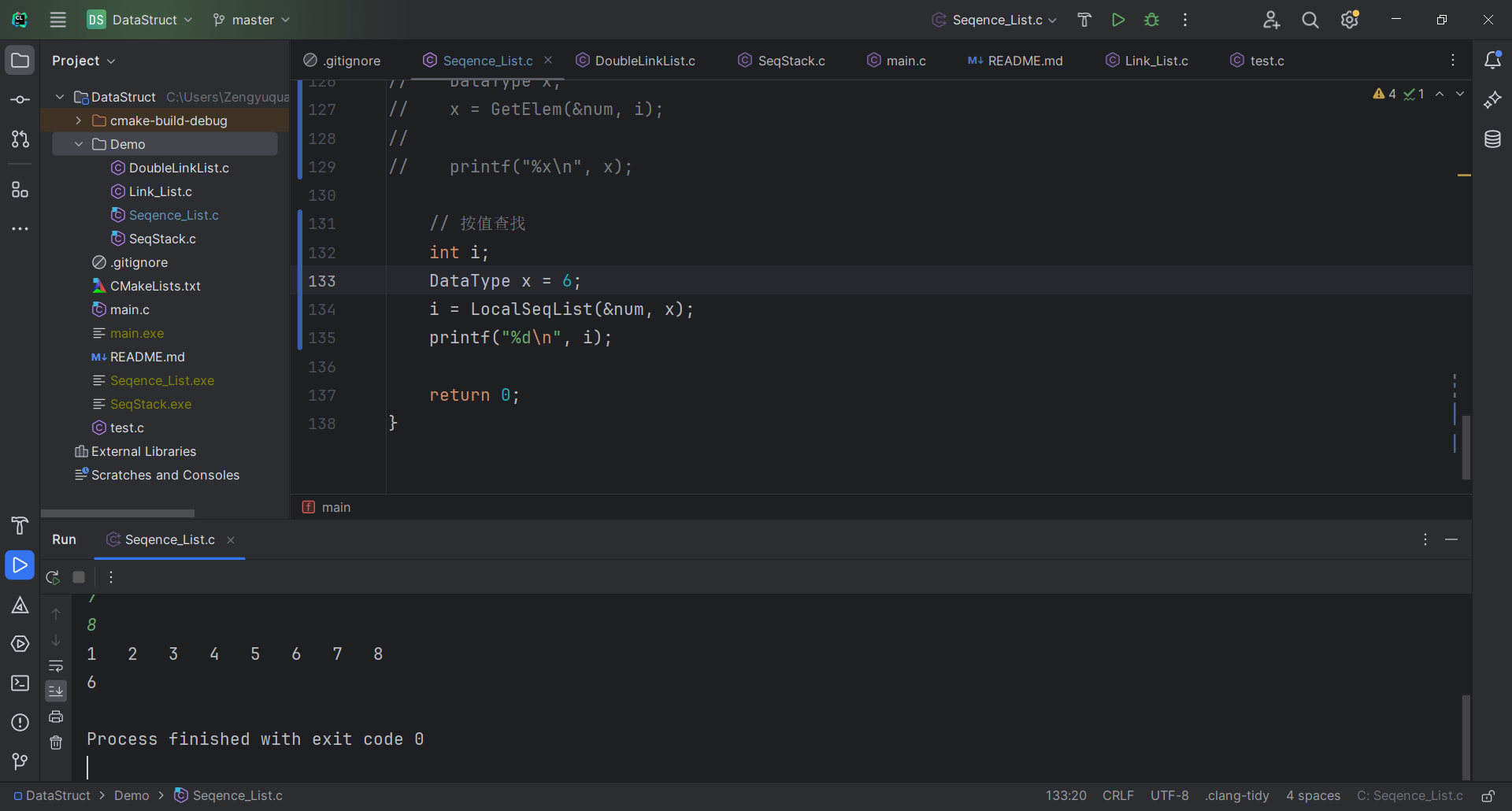
5.3 按元素位置查找顺序表操作的实现。

假设我们要查找第5个位置上的元素，把num的地址指针和查找位置为5的变量i传递给GetElem()函数，最终打印出来“5”这个在顺序表中第5个位置上的元素。

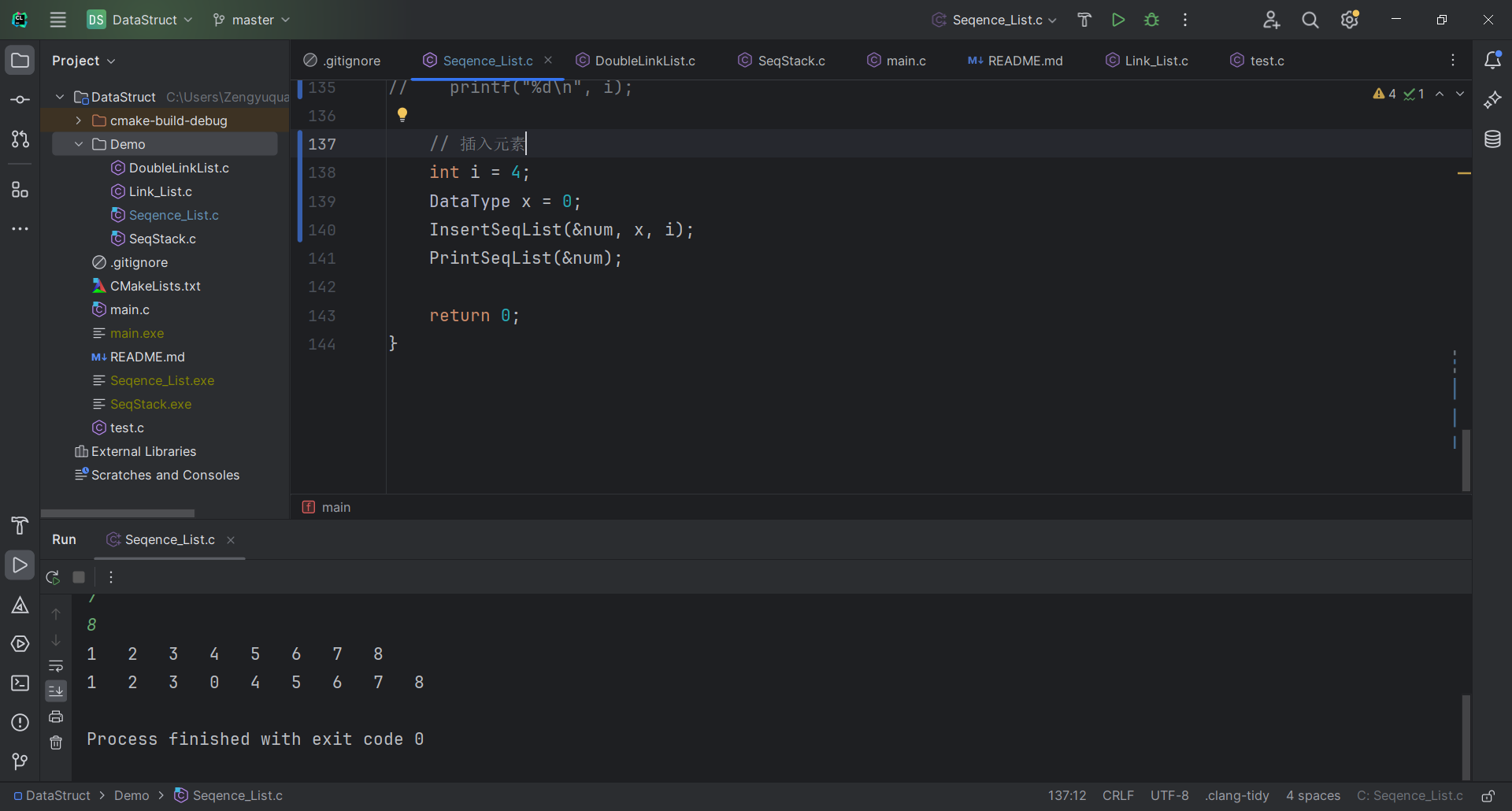


5.4 按值查找顺序表操作的实现。

定义一个名为x的变量，假设要查找值为6的元素在顺序表中的位置。将x变量连同num的地址指针传递给LocalSeqList()函数。最终打印出值为6的元素在顺序表中位置。

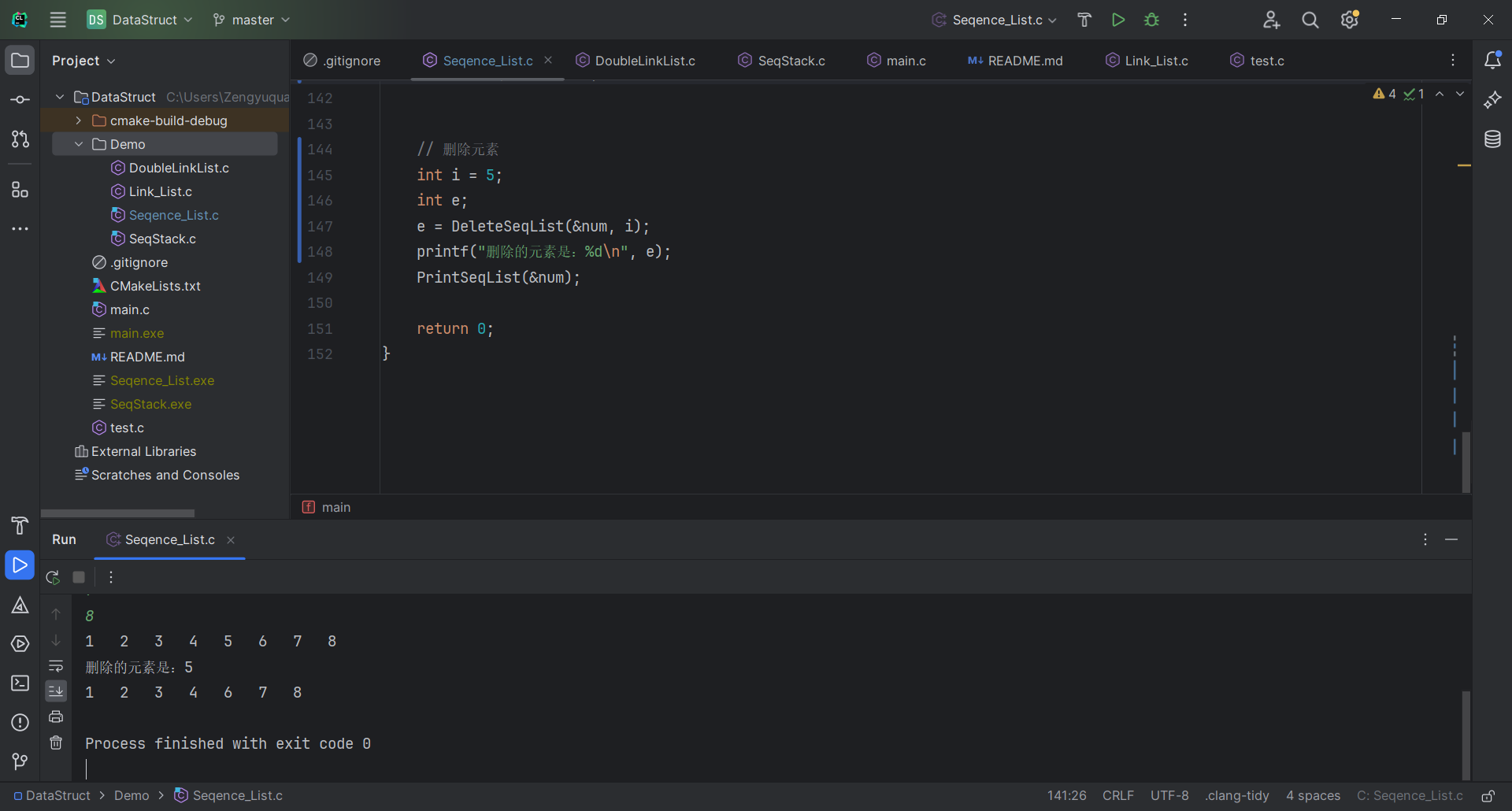


5.5 顺序表的插入操作的实现。

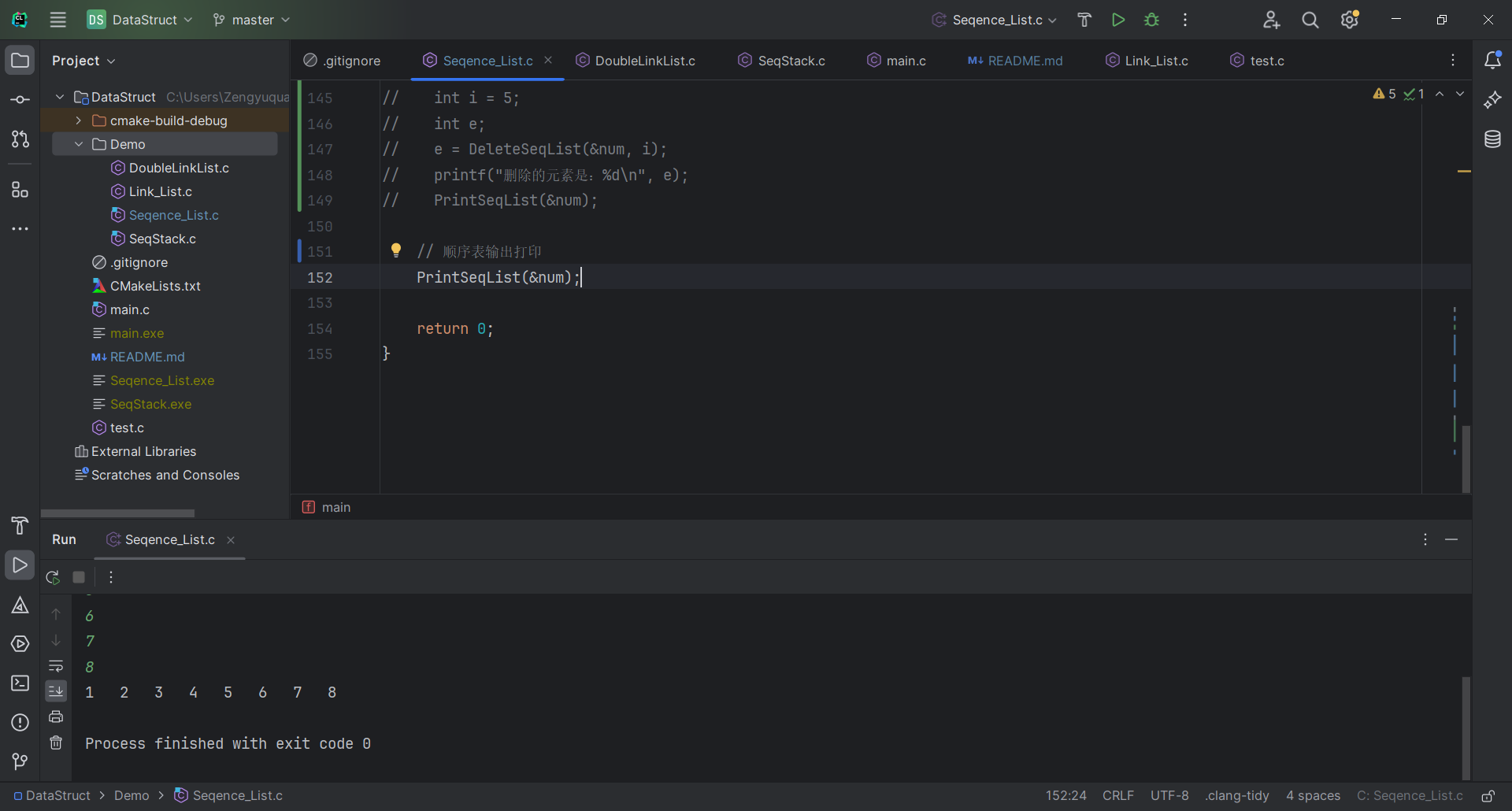
假设在第四个位置上插入一个0元素。定义一个int类型i表示插入的位置，DataType类型的x表示插入的元素值。三者传递给InsertSeqList()函数，最终打印出插入后的顺序表元素。

5.6 顺序表的删除操作的实现。

假设删除的数据元素是“5”，则将i（赋值为元素是“5”）的变量，num的地址指针传递给DeleteSeqList()函数。最终将函数的返回值。



5.7 顺序表的输出打印操作的实现。

调用PrintSeqList()函数，传递num的地址指针，便可实现将顺序表num中的元素输出打印。

**六、实验分析**

线性表是一种随机存取的数据结构，查找顺序表中的某位置i上的元素，就是直接对数据元素data[i-1]进行查找；这样子的操作算法时间复杂度是O(1)。

顺序表的插入操作，由于在某一位置上插入元素，则在此位置往后的所有元素都要往后移动一位。假设在有n个元素的顺序表中，在i位置上插入元素，需要移动n-i+1个元素。

设是第一个元素之前插入移动的元素的概率，则=1/(n+1)；所有顺序表中所有元素的平均移动次数为：

其结果为。可得出插入线性表操作的时间复杂度为O(n)。

顺序表的删除操作，由于在某一位置上删除元素，则在此位置往后的所有元素都要往前移动一位。假设在有n个元素的顺序表中，在i位置上删除元素，需要移动n-i个元素。

设是删除第i个元素的概率，则=1/n；所有顺序表中所有元素的平均移动次数为：

其结果为。可得出删除线性表操作的时间复杂度为O(n)。

**七、思考与总结**

顺序表在实际应用中广泛用于需要快速访问和顺序存储的场景，如数据库索引、缓存管理、文本编辑器等。为优化其操作，可采用动态扩容策略减少插入时的重分配次数，利用分段存储技术提高内存利用率，并通过预读和缓存机制加速数据访问，从而提升整体性能。此外，结合多线程技术，可以实现并行处理，进一步提高顺序表在大规模数据处理中的效率。通过这些优化手段，顺序表不仅能够应对高频访问的需求，还能在复杂应用中保持高效稳定的性能表现。