

**《数据结构课程设计》**

**实验报告**

**实验1： 线性表的实现和操作**

**姓 名： 徐壮**

**学 号： 21334129**

**专 业： 网络工程**

**实验时间：**

**指导老师： 薛梅婷**

**杭州电子科技大学**

**网络空间安全学院**

**一、实验目的**

1. 线性表顺序存储结构的操作与案例的实现

2. 线性表链式存储结构的操作与应用

3. 栈与队列的操作与应用

4. 串的操作与应用

1. **实验内容与实验结果**

实验1-1

1. 实验内容

顺序表存储结构，定义程序所需要的一些符号常量、线性表元素类型和顺序表存储结构，以及所需的库函数。根据定义的顺序表结构

（1）InitList(SqList &L)——构造一个空的顺序表表L，L的初始大小为MAXSIZE。

（2）ClearList(SqList &L)——将顺序表的数据清空。

（3）int Listlength(SqList L)——返回L中数据元素个数。

（4）GetElem(SqList L, int i, ElemType &e)——用e返回L中第i个数据元素的值，

（5）int LocateList(SqList L, ElemType e)——返回L中第1个与e相等的数据元素位序，

（6）ListInsert(SqList &L, int i, ElemType e)——在L中的第i个位置之前插入新的数据元素e

（7）ListDelete(SqList &L, int i, ElemType &e )——删除L中的第i个数据元素，用e返回其值，1≤i≤L.length。

（8）ListTraverse(SqList L)——利用printf函数依次输出L的每个数据元素的值。

线性表元素的集合运算将顺序表La和Lb看成2个集合，利用上述基本操作函数，分别编写出实现集合并、交、差运算的3个函数源码：

两个有序表的合并设La和Lb中的元素是非递减有序的，编写一个有序表合并，即Lc=La+Lb的函数源码

去掉线性表中的重复多余的元素设Lc是一个有重复元素的顺序表，编写一个去掉顺序表中重复多余元素的函数源码

 编写一个主函数main()，来检验上述应用函数是否正确，

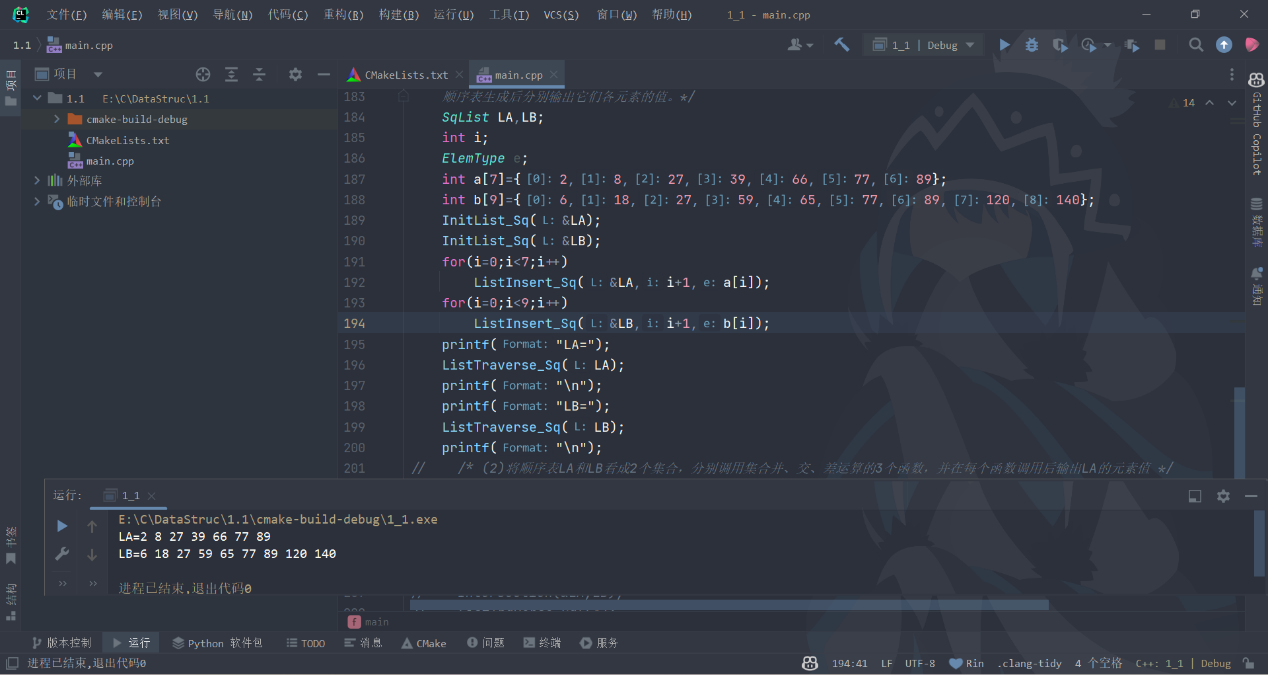
2. 源程序代码（\*.CPP）

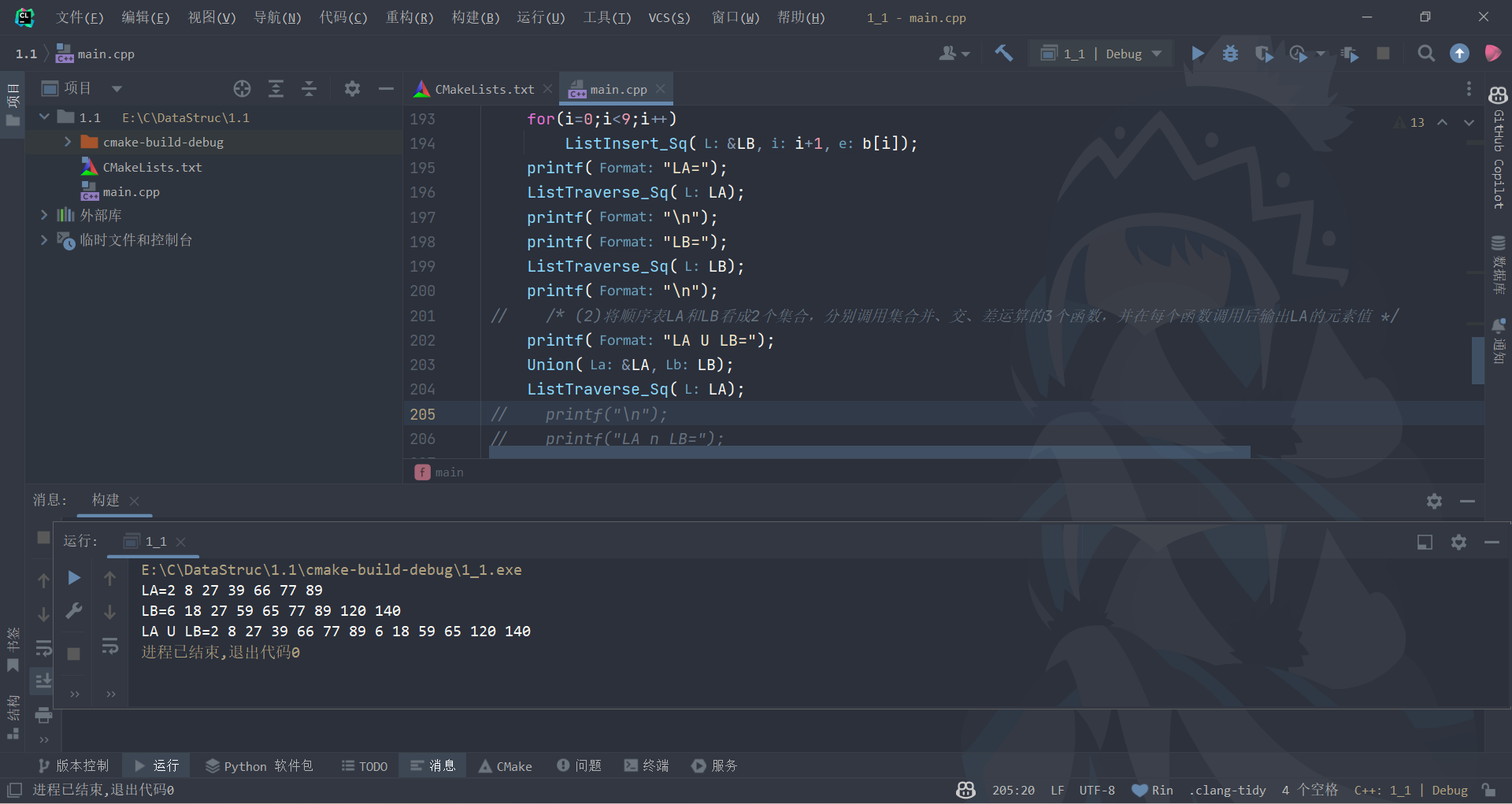
见附件 1.1-main.cpp

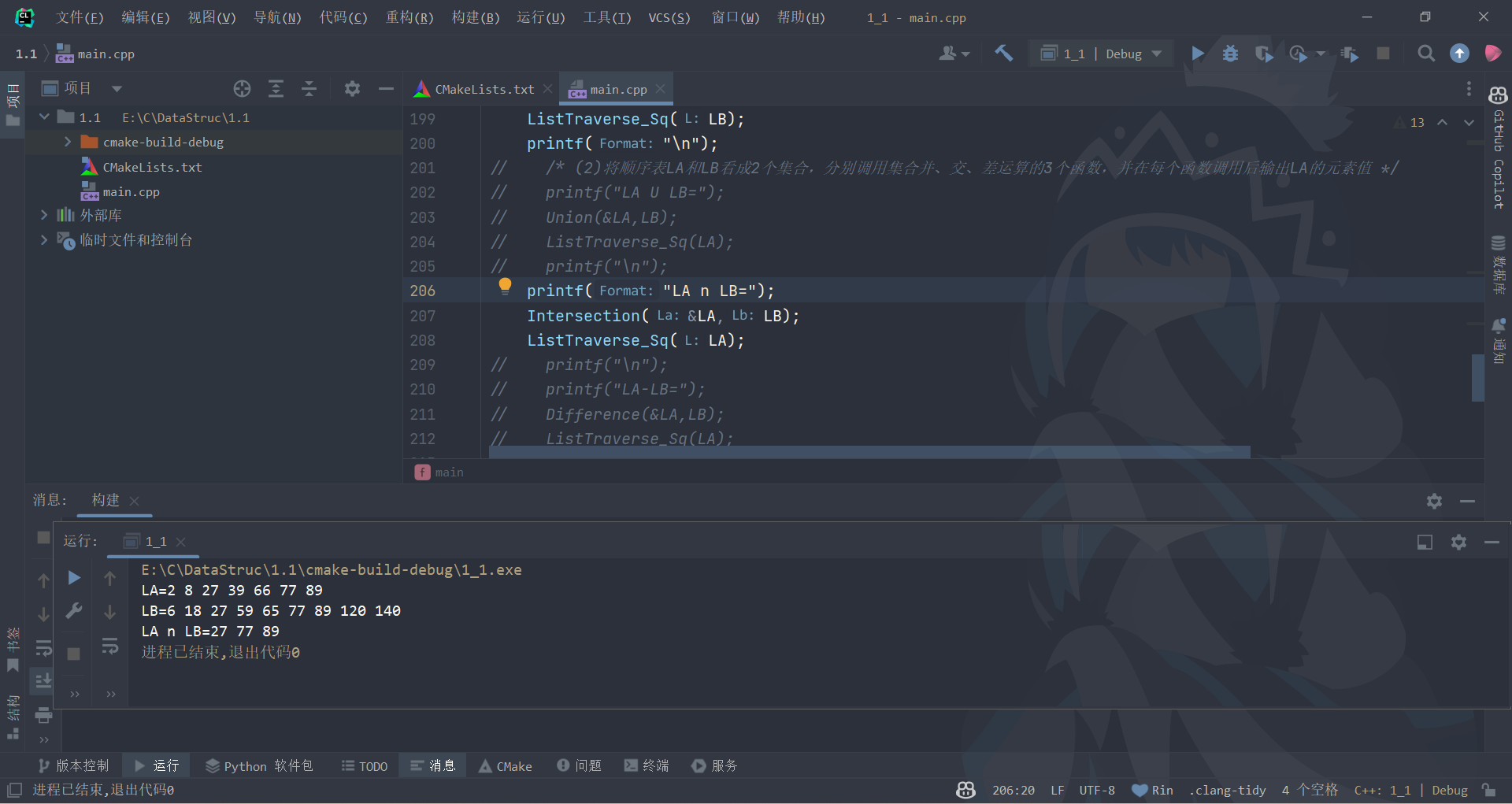
（见附件）

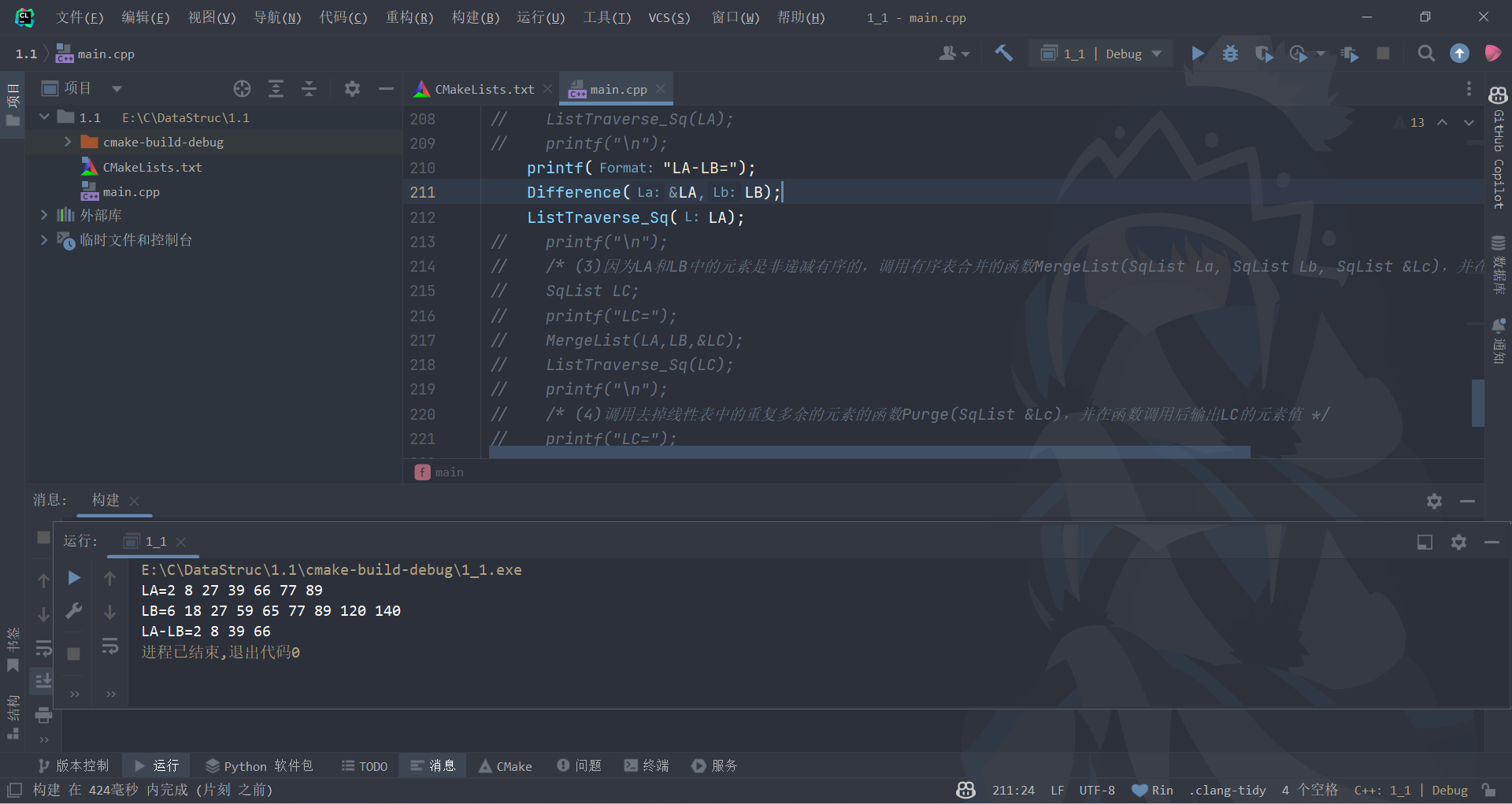
3. 运行结果

(1).数据准备生成两个顺序表LA,LB并输出

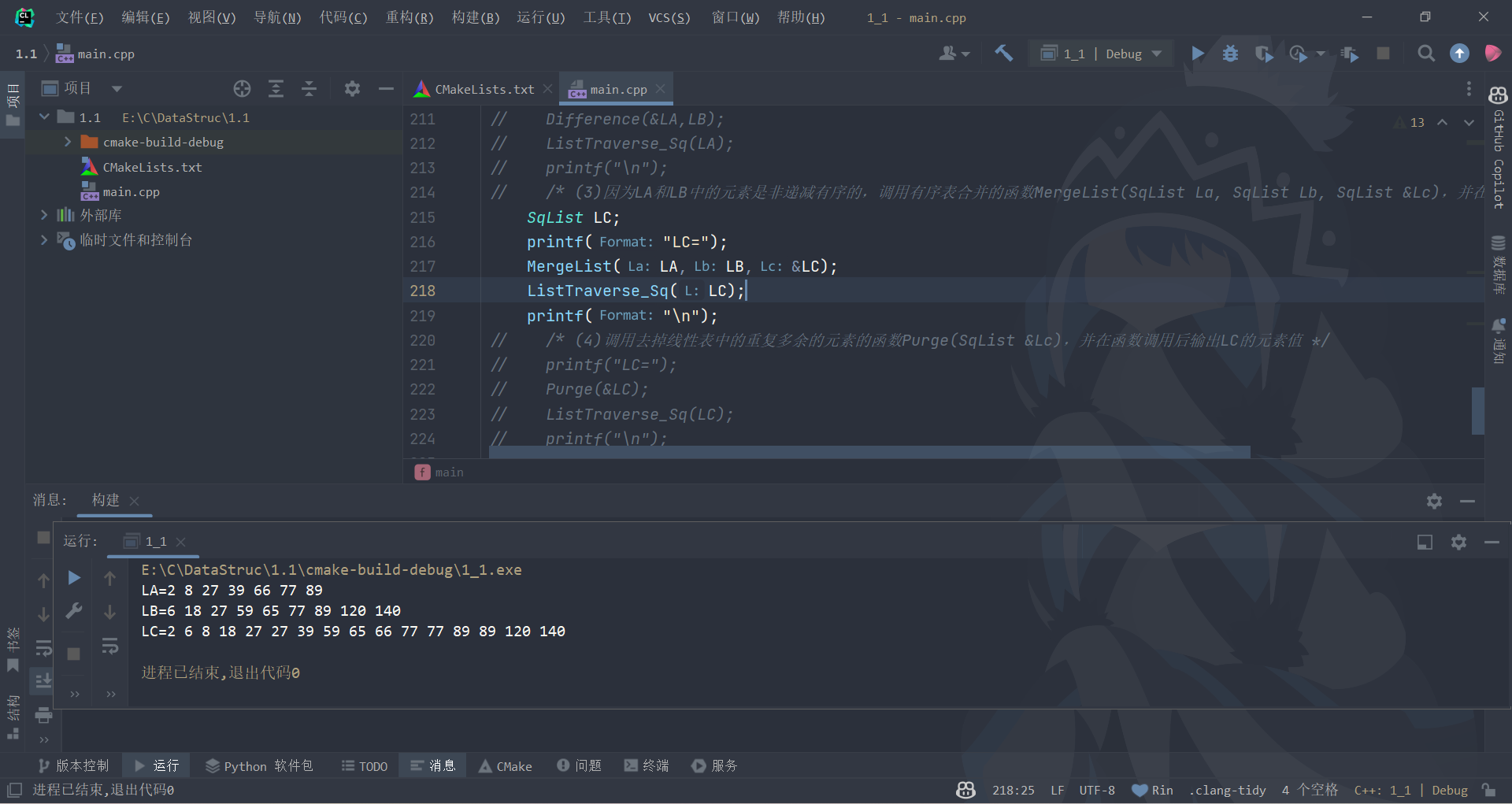
(2) 将顺序表LA和LB看成2个集合，分别调用集合并、交、差运算的3个函数，并在每个函数调用后输出LA的元素值，观察函数执行结果是否正确。



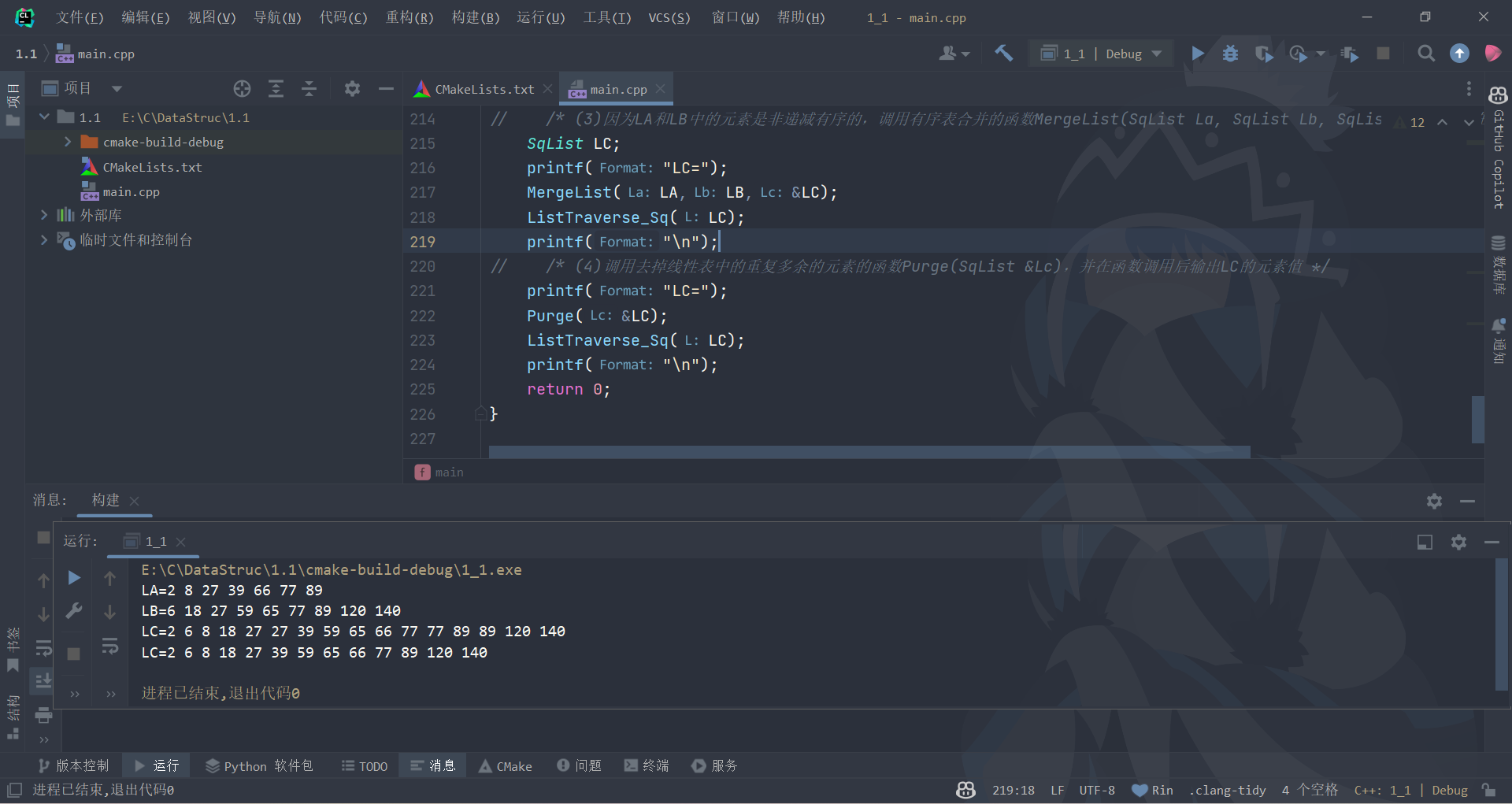




（3）因为LA和LB中的元素是非递减有序的，调用有序表合并的函数MergeList(SqList La, SqList Lb, SqList &Lc)，并在函数调用后输出LC的元素值。



（4）MergeList函数合并后的有序表LC存在重复多余的元素，调用Purge(SqList &Lc)去掉重复多余的元素，并在函数调用后输出LC的元素值。



实验1-2

1. 实验内容

单链表存储结构的定义

单链表基本操作函数实现

（1）InitList(LinkList &L)——初始化单链表L。

（2）ClearList(LinkList &L)——清空单链表。

（3）int Listlength(LinkList L)——求单链表的长度，即返回L中数据结点个数。

（4）GetElem(LinkList L, int i, ElemType &e)——用e返回L中第i个数据结点的值

（5）LNode \*LocateList(LinkList L, ElemType e)——返回L中第1个与e相等的数据结点的地址（指针），若e不存在，则返回NULL。

（6）ListInsert(LinkList &L, int i, ElemType e)——在L中的第i个数据结点之前插入新的数据结点e，1≤i≤Listlength(L)+1。

（7）ListDelete(LinkList &L, int i, ElemType &e )——删除L中的第i个数据j结点，用e返回删除结点的值，1≤i≤Listlength(L)。

（8）ListTraverse(LinkList L)——利用printf函数依次输出L的每个数据结点的值。

将单链表La和Lb看成2个集合，利用上述基本操作函数，分别编写出实现集合并、交、差运算的3个函数源码

两个有序表的合并

去掉线性表中的重复多余的元素

编写一个主函数main()，来检验上述应用函数是否正确

2. 源程序代码（\*.CPP）

见附件 1.2-main.cpp

（见附件）

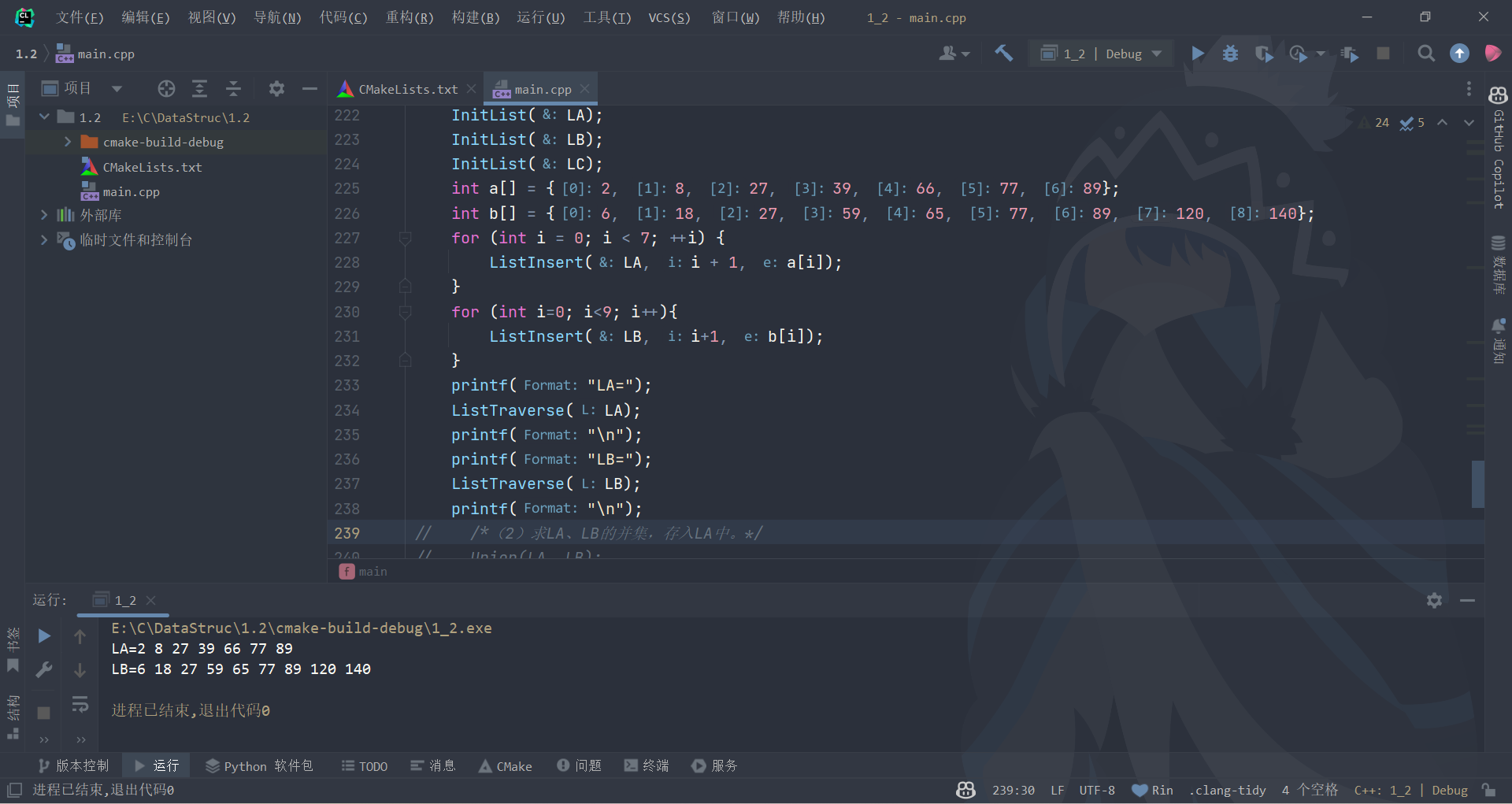
3. 运行结果

（1）数据准备。生成2个顺序表LA、LB，其数据元素分别为：

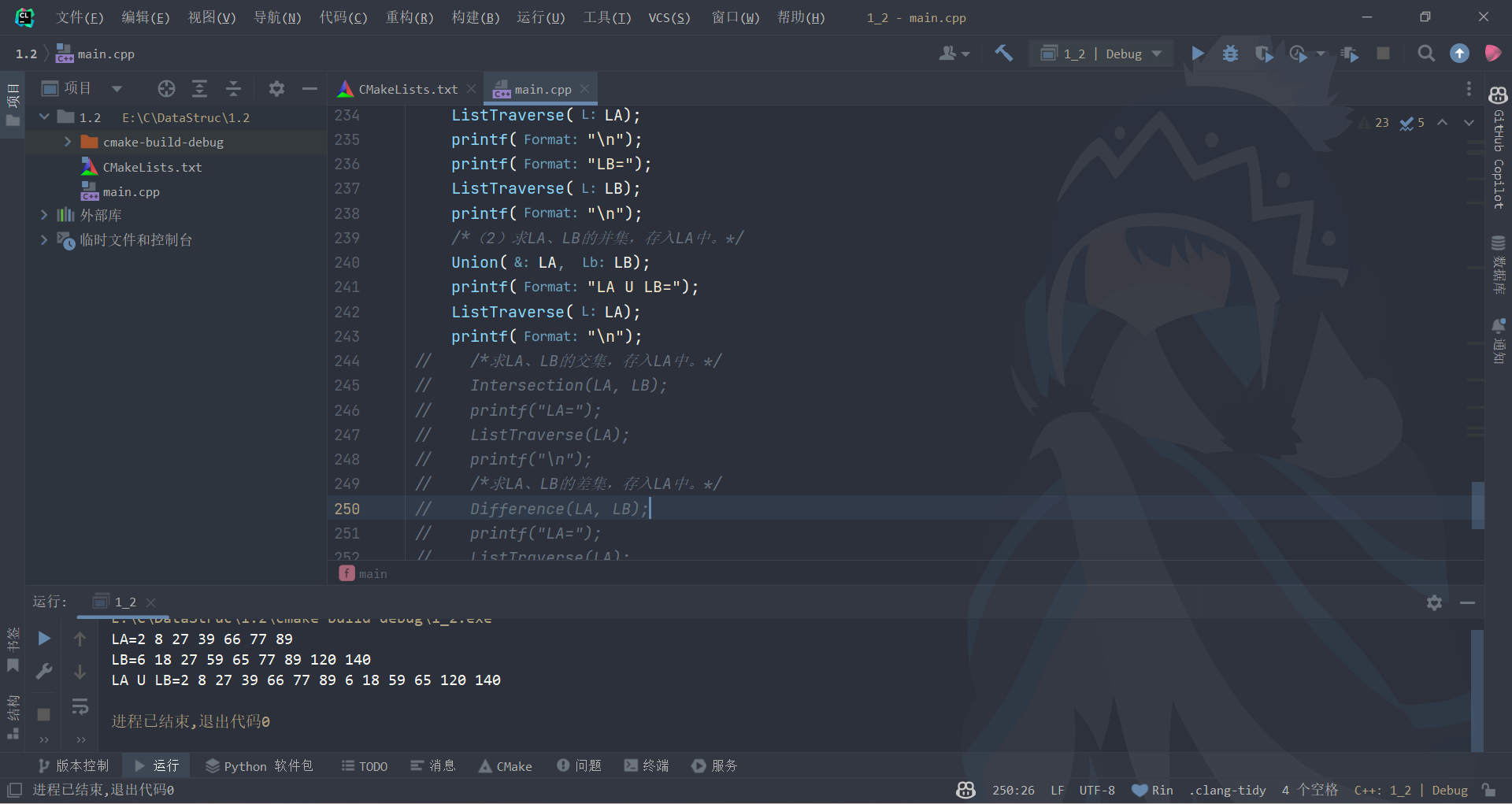
LA=2→8→27→39→66→77→89→∧

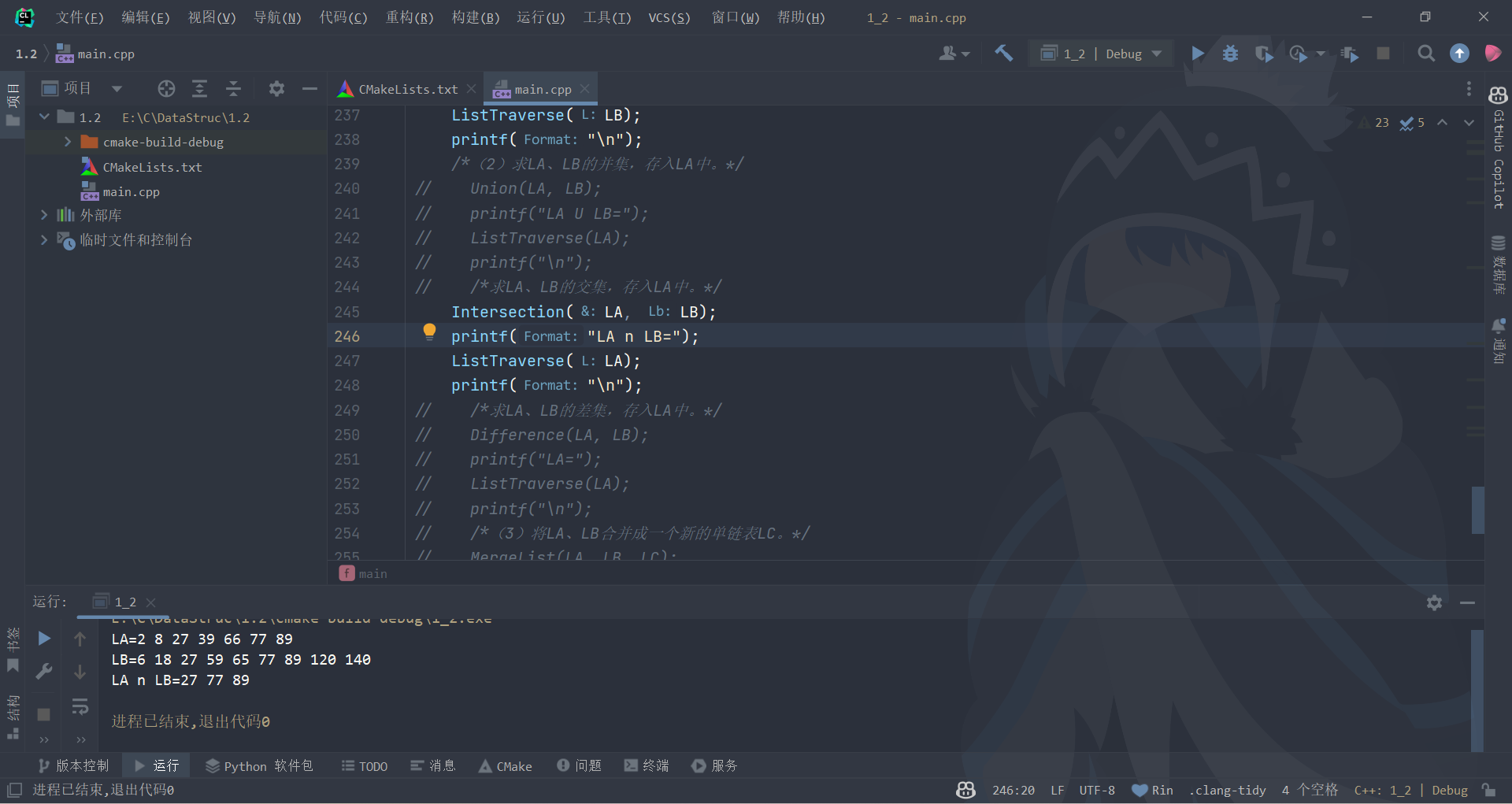
LB=6→18→27→59→65→77→89→120→140→∧

单链表生成后，以上述格式输出各元素值。



（2）将单链表LA和LB看成2个集合，分别调用集合并、交、差运算的3个函数，并在每个函数调用后输出LA的元素值，观察函数执行结果是否正确。



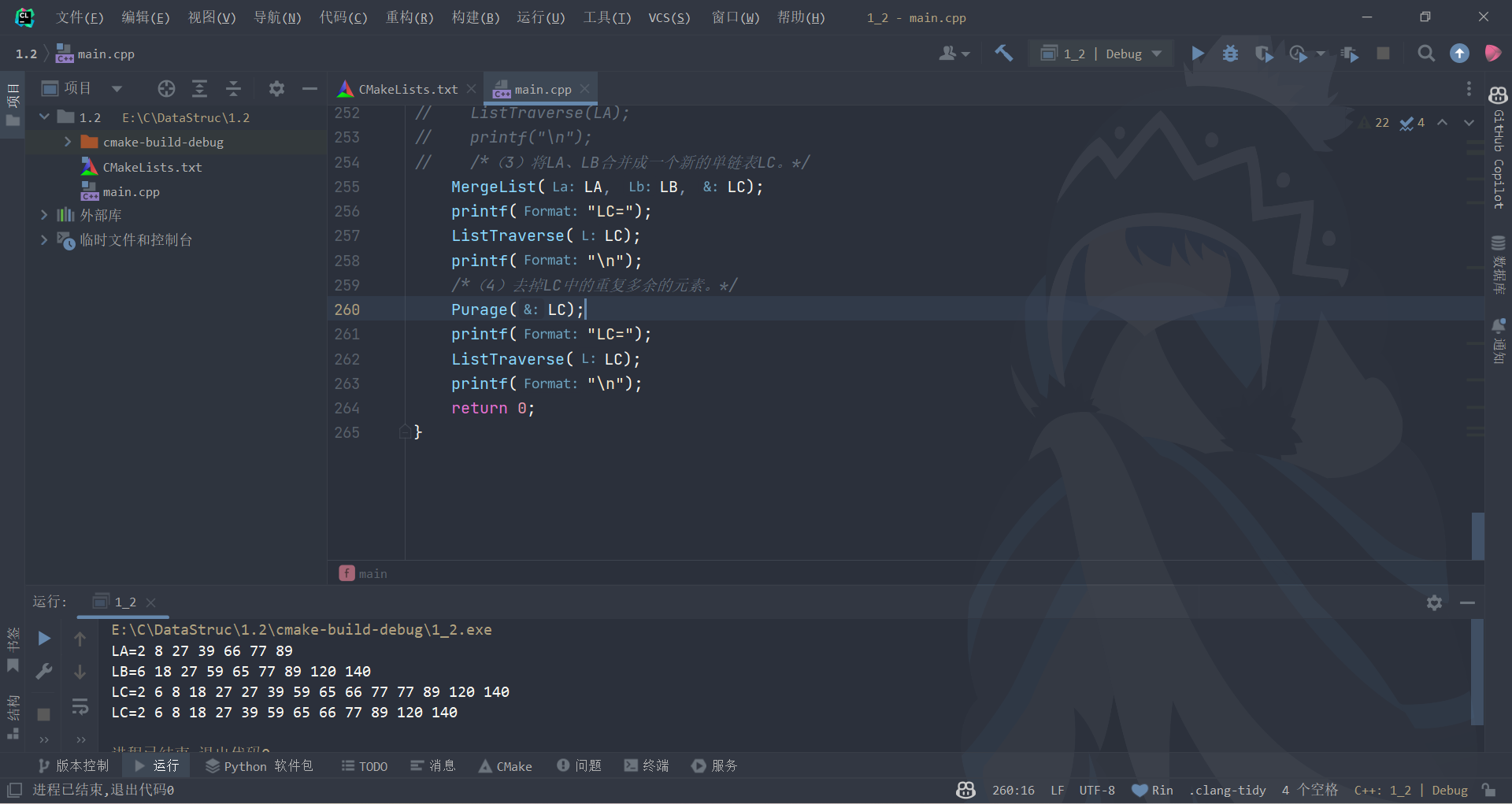




（3）因为LA和LB中的元素是非递减有序的，调用有序表合并的函数MergeList(LA, LB, LC)，并在函数调用后输出LC的元素值。



（4）MergeList函数合并后的有序表LC存在重复多余的元素，调用Purge(LC)去掉重复多余的元素，并在函数调用后输出LC的元素值。



实验1-3

1. 实验内容

**顺序栈和循环队列的存储结构和相关内容的定义**

**顺序栈基本操作函数的实现**

根据定义的顺序栈结构，编写下列基本操作函数的C/C++源代码：

（1）InitStack(SqStack &S)——利用malloc函数申请存储空间，构造一个空的顺序栈S，S的初始大小为STACK\_INIT\_SIZE。

（2）ClearStack(SqStack &S)——将顺序栈数据清空。只需做S.top=S.base赋值操作即可。

（3）StackLength(SqStack S)——返回L中数据元素个数。直接返回S.top-S.base的值即可。

（4）GetTop(SqStack S, SElemType &e)——用e返回栈顶元素的值。

（5）Push(SqStack &S, SElemType e)——将数据e进栈。

（6）Pop(SqStack &S, SElemType &e)——将栈顶元素出栈，用e返回其值。

**循环队列基本操作函数的实现**

根据定义的循环队列结构，编写下列基本操作函数的C/C++源代码：

（1）InitQueue(SqQueue &Q)——利用malloc函数申请存储空间，构造一个空的循环队列Q，Q的大小为MAXQSIZE。

（2）ClearQueue(SqQueue &Q)——将循环队列数据清空。

（3）QueueLength(SqQueue Q)——返回Q中数据元素个数。

（4）EnQueue(SqQueue &Q, QElemType e)——数据e入队。

（5）DeQueue(SqStack &Q, QElemType &e)——数据出队，用e返回其值。

（6）QueueEmpty(SqQueue Q)——判对空。

（7）QueueFull(SqQueue Q)——判队满。

应用案例

1. 利用顺序栈的基本操作函数，编制一个将十进制整数m转换成n进制(n=2或8)整数的数制转换函数Conversion(m,n)
2. 编一个函数QueueReverse(Q)，利用栈将将队列中的元素逆序重排，即队头元素变成队尾元素。

5. 主函数main()

（1）数制转换

输入一个十进制数m，调用Conversion(m,n) 函数将m转换成二进制数和八进制数。

（2）循环队列操作

<1>连续入队9个整数后，显示队列Q的front和rear的值；

<2>若再入队1个整数，显示此时队列的状态信息；

<3>连续出队5个整数，再入队2个整数，显示此时的队列Q的长度与front和rear的值；

<4>调用函数QueueReverse(Q)，将队列元素逆序重排，然后将队列元素出队并输出。

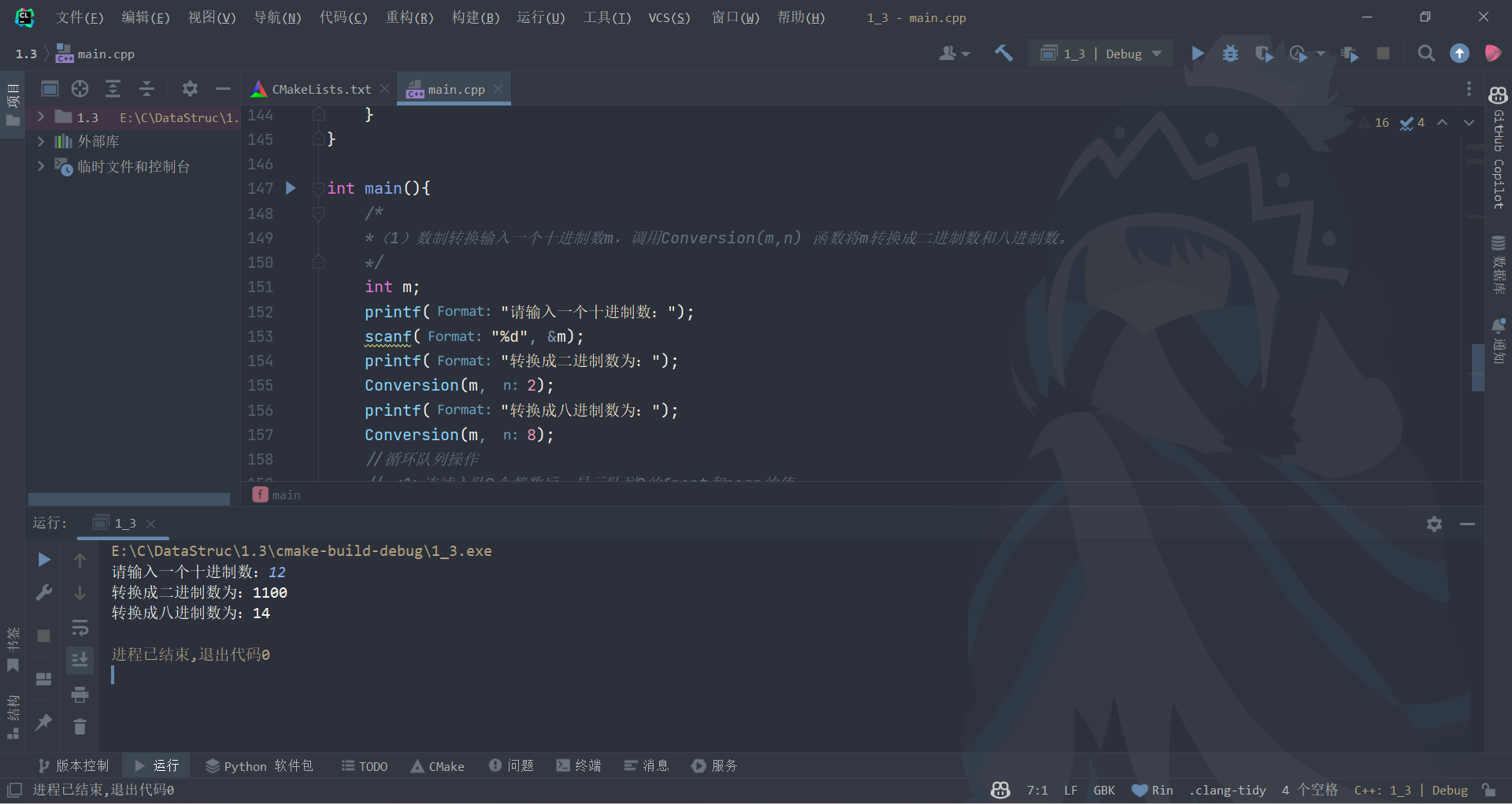
2. 源程序代码（\*.CPP）

见附件 1.3-main.cpp

（见附件）

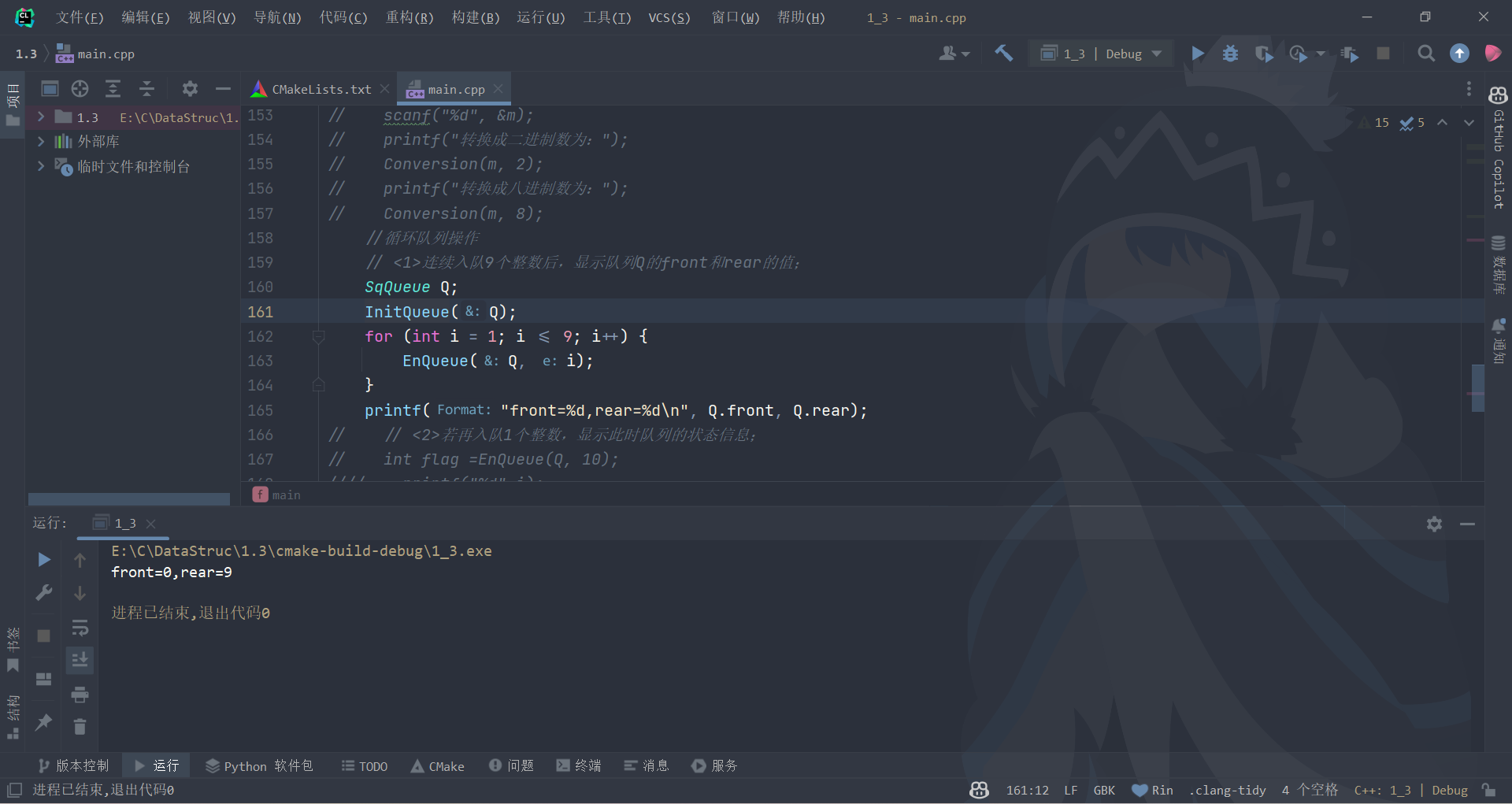
3. 运行结果

（1）数制转换输入一个十进制数m，调用Conversion(m,n) 函数将m转换成二进制数和八进制数。

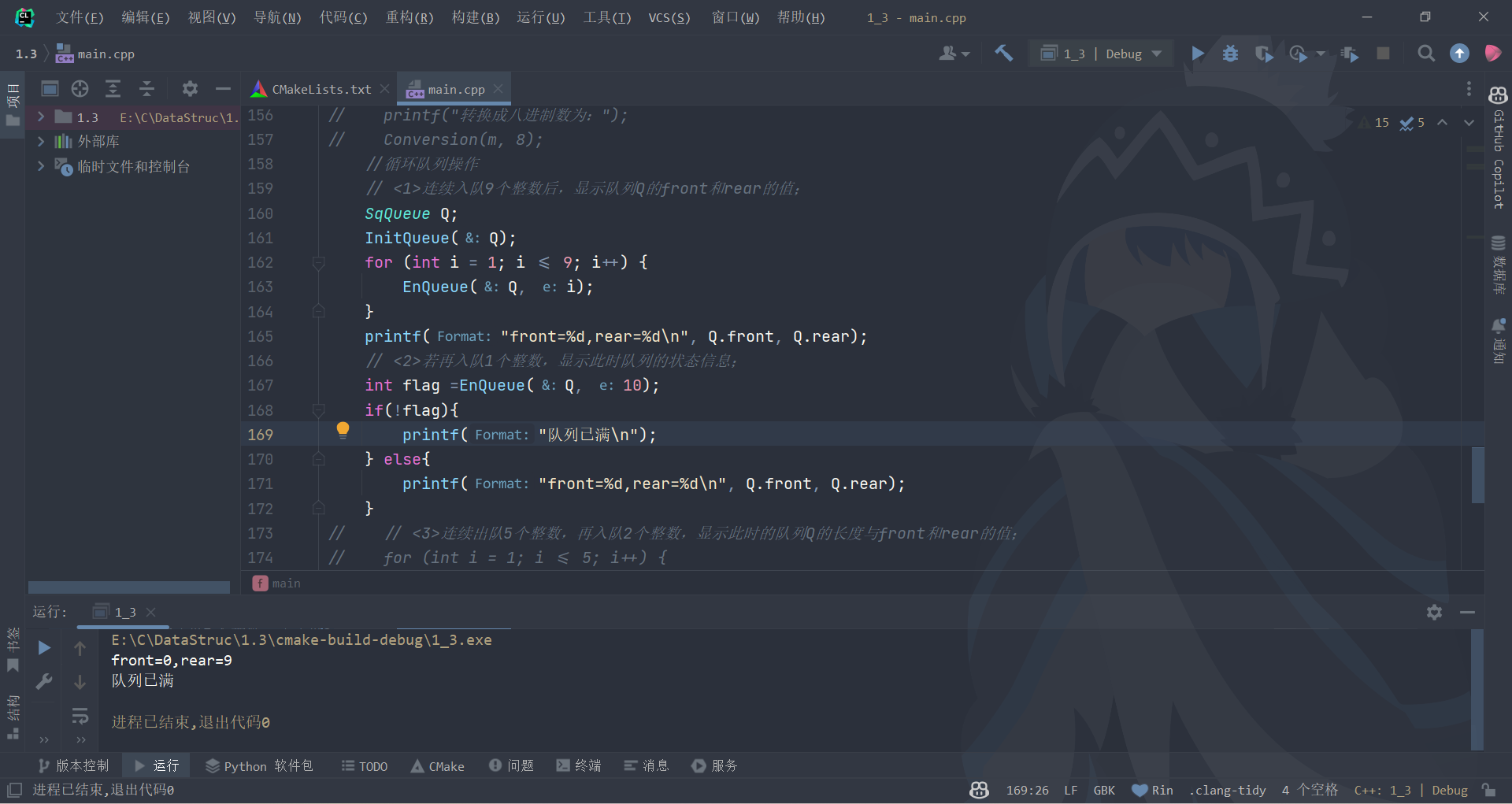


（2）循环队列操作

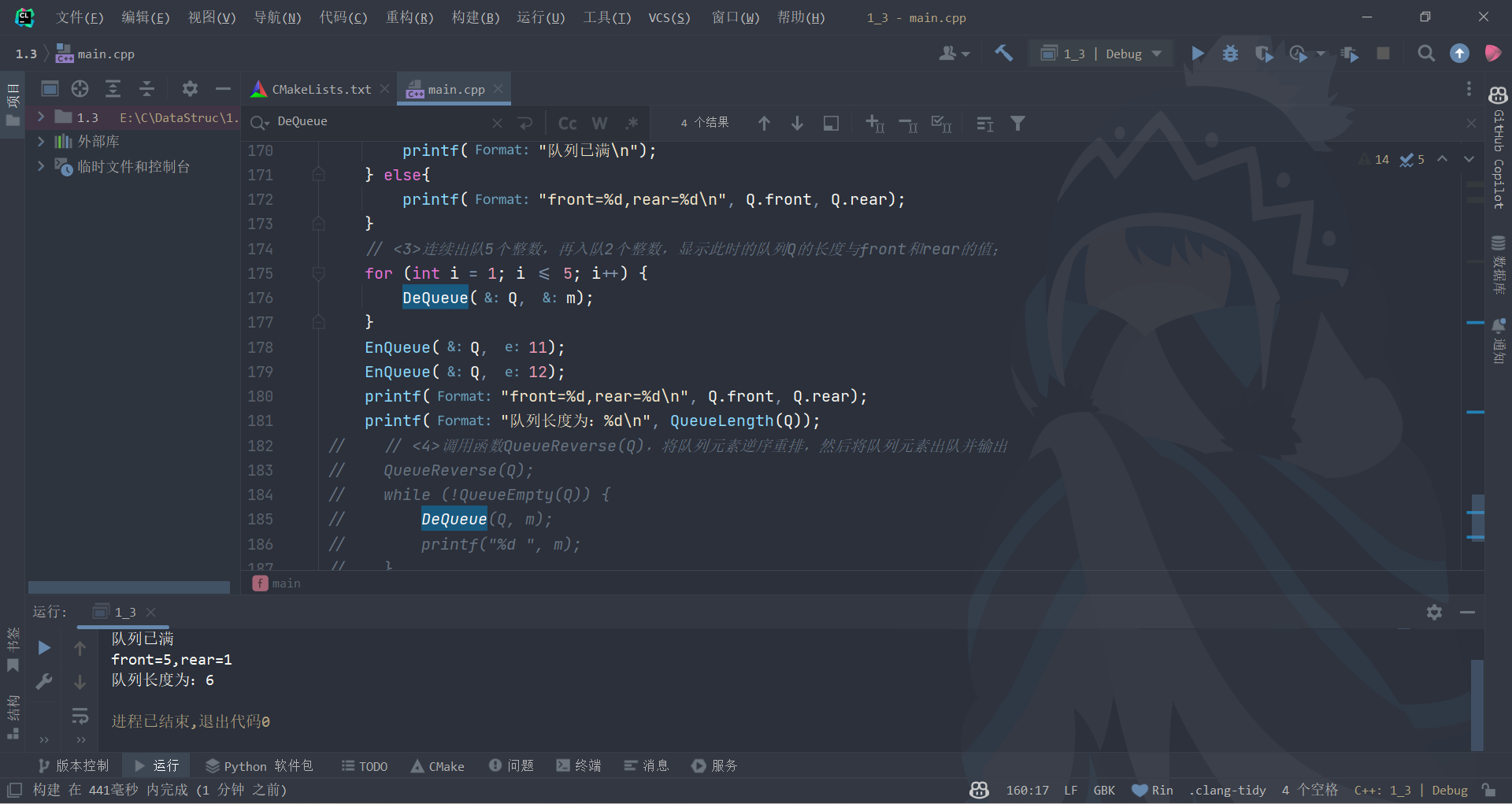
<1>连续入队9个整数后，显示队列Q的front和rear的值；



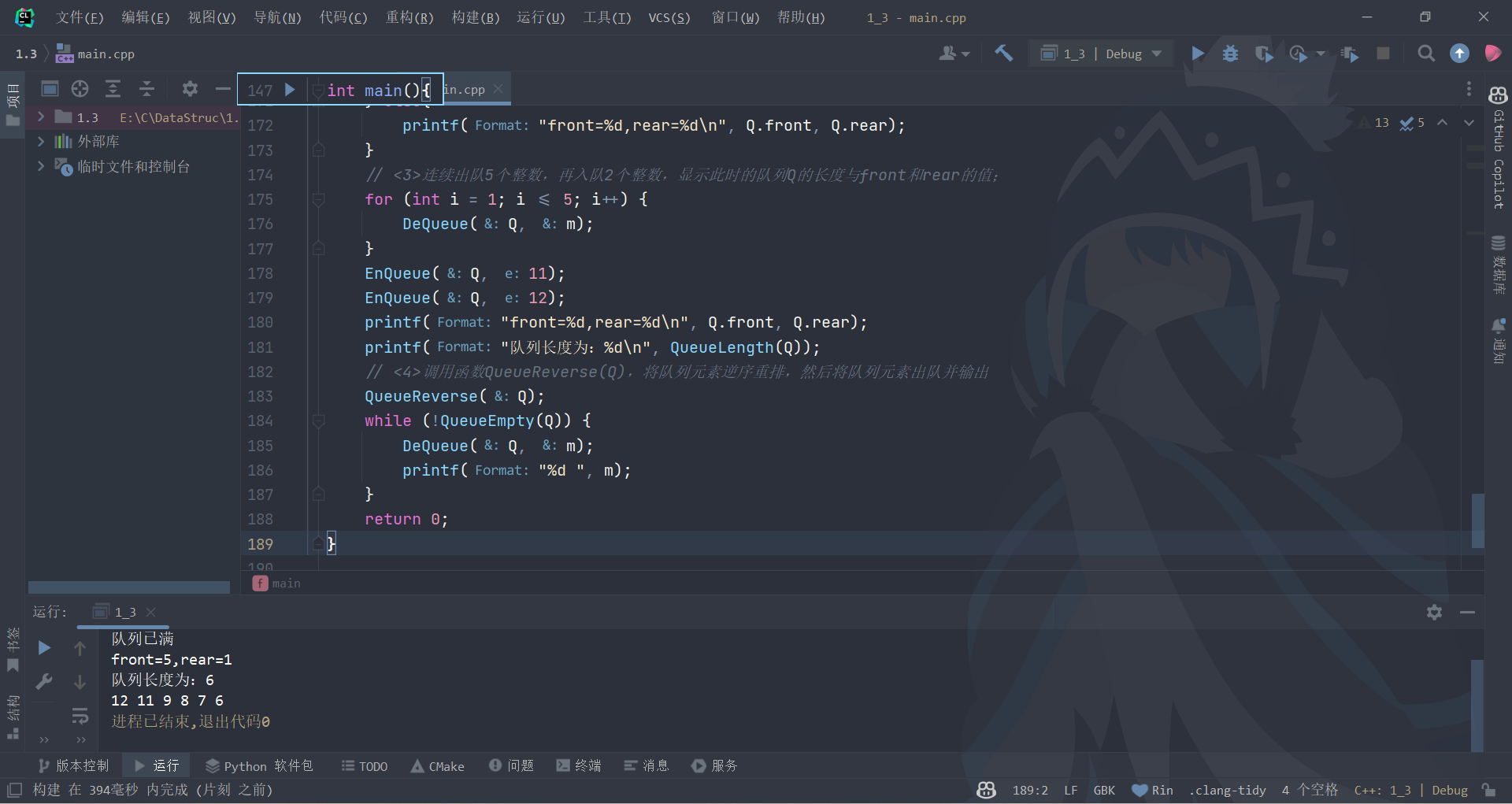
<2>若再入队1个整数，显示此时队列的状态信息；



<3>连续出队5个整数，再入队2个整数，显示此时的队列Q的长度与front和rear的值；



<4>调用函数QueueReverse(Q)，将队列元素逆序重排，然后将队列元素出队并输出。



实验1-4

1. 实验内容

串的堆分配存储结构定义

串基本操作函数的实现

编写下列基本操作函数的C/C++源代码。

（1）StrAssign(HString &S,chars)——将字符串常量chars赋给串变量S；

（2）StrCompare(HSTring S, HString T)——比较串S与T的值 ，若S>T, 则返回1；若S=T,则返回0；否则返回-1。

（3）StrLength(HString S)——返回串S的长度。

（4）Concat(HString &S, HString T1, HString T2)——将串T1和T2连接在一起赋给串变量S。

（5）SubString(HString &sub, HString S, int pos, int len)——用Sub返回串S的第pos个字符起长度为len的子串，1≤ pos ≤StrLength(S)，1≤len≤StrLength(S)-pos+1

（6）StrTraverse(HString S)——遍历（输出）字符串S中的各个字符。

串其他操作函数的实现

利用上述串的基本操作函数，编制下列串的操作函数。

（1）StrCopy(HString &S, HString T)——将串T赋给串S。若使用堆分配存储结构，可利用Concat()函数实现。若使用堆分配存储结构，则需针对存储结构操作。

（2）Index(HString S, HString T, int pos)——返回串T在串S中第pos个字符开始出现的位置，若串T不是串S的子串，则返回0。1≤ pos ≤StrLength(S)。利用SubString()和StrCompare()实现。

（3）Replace(HString &S, HString T1, HString T2)——将S串中所有的子串T1用串T2代替。利用SubString()、StrCompare()和Concat()实现。

（4）StrInsert (HString &S, int pos, HString T)——在串S的第pos个字符之前插入串T。1≤ pos ≤StrLength(S)+1。利用SubString()和Concat()实现。

（5）StrDelete(HString &S, int pos, int len)——删除串S中从第pos个字符开始连续的len个字符。1≤ pos ≤StrLength(S)-len+1，利用SubString()和Concat()实现。

编写一个主函数main()，检验上述操作函数是否正确，实现以下操作：

（1）数据准备。利用StrAssign()将设定的字符串常量（自行设定）赋给3个字符串变量T1,T2,T3，然后利用StrTraverse()输出字符串字符。

（2）利用StrCopy()将串T1赋给串S，并输出S的字符。

（3）利用StrInsert()将串T2插入到S的尾部，并输出S的字符。

（4）利用StrReplace()将串S中的T1替换成T3，并输出S的字符。

（5）利用Index()查找串T2在S中的位置，并输出位置结果。

（6）利用StrDelete()删除S中子串T2，并输出删除后S的结果。

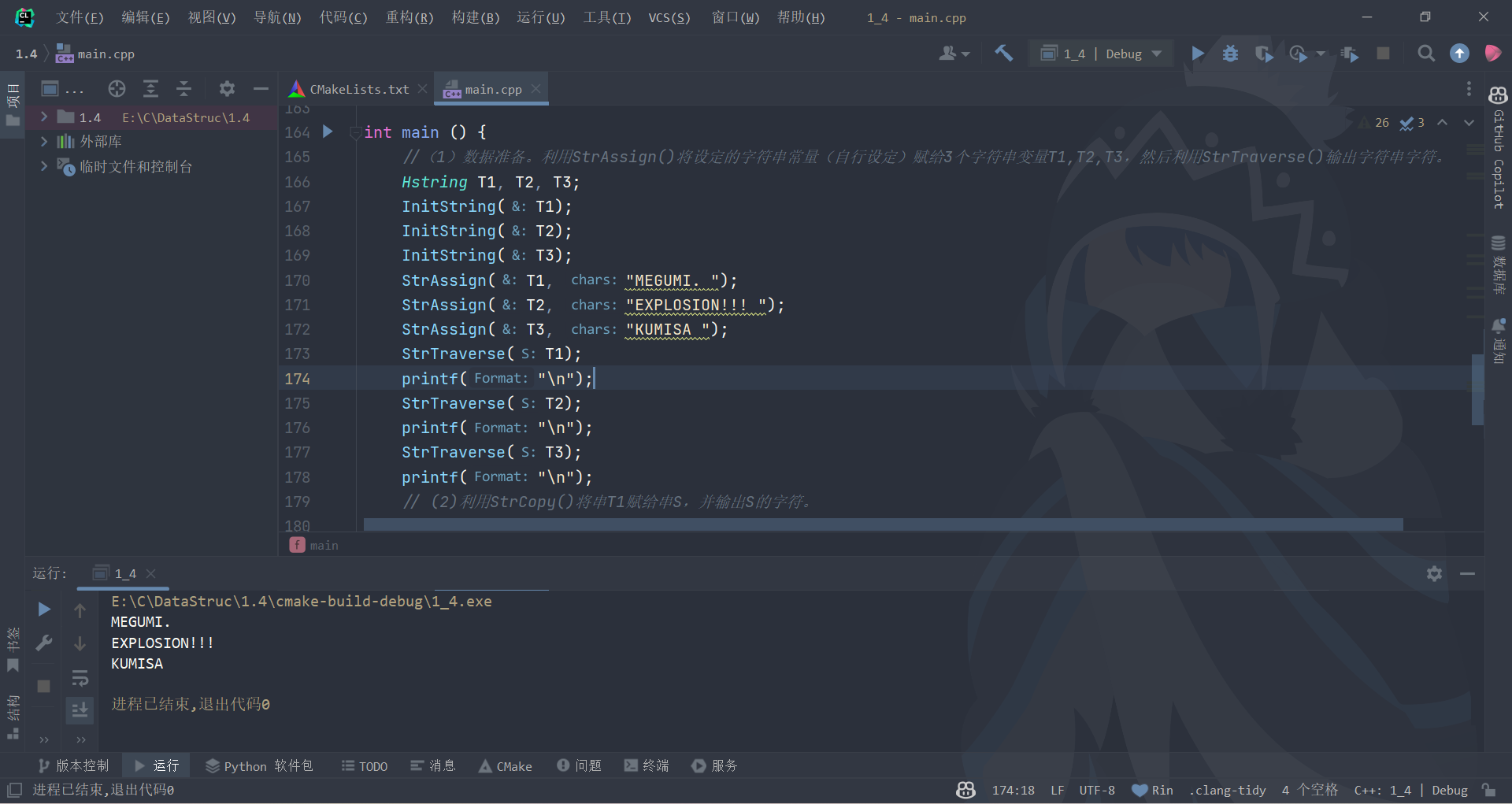
2. 源程序代码（\*.CPP）

见附件 1.4-main.cpp

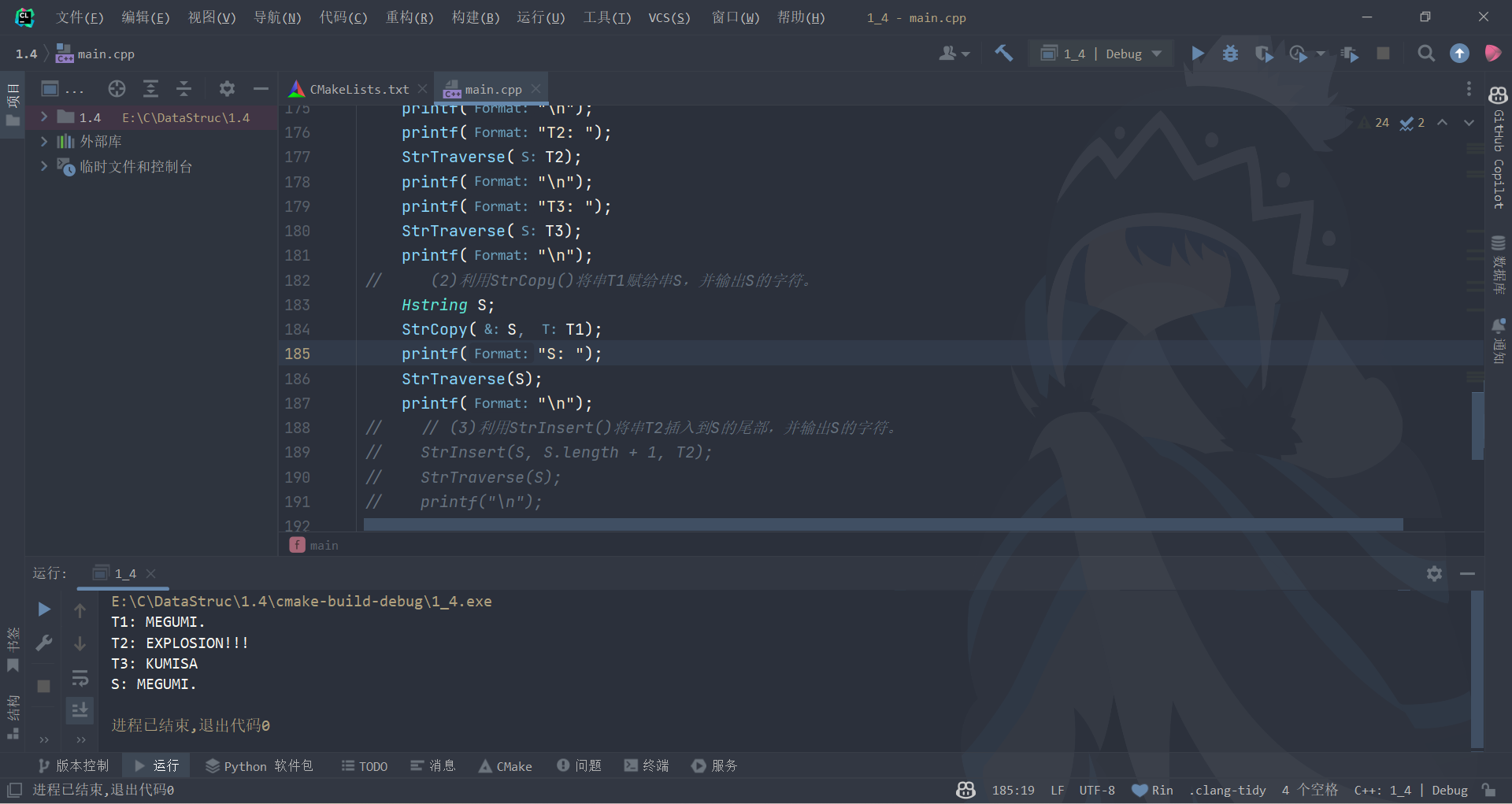
（见附件）

3. 运行结果

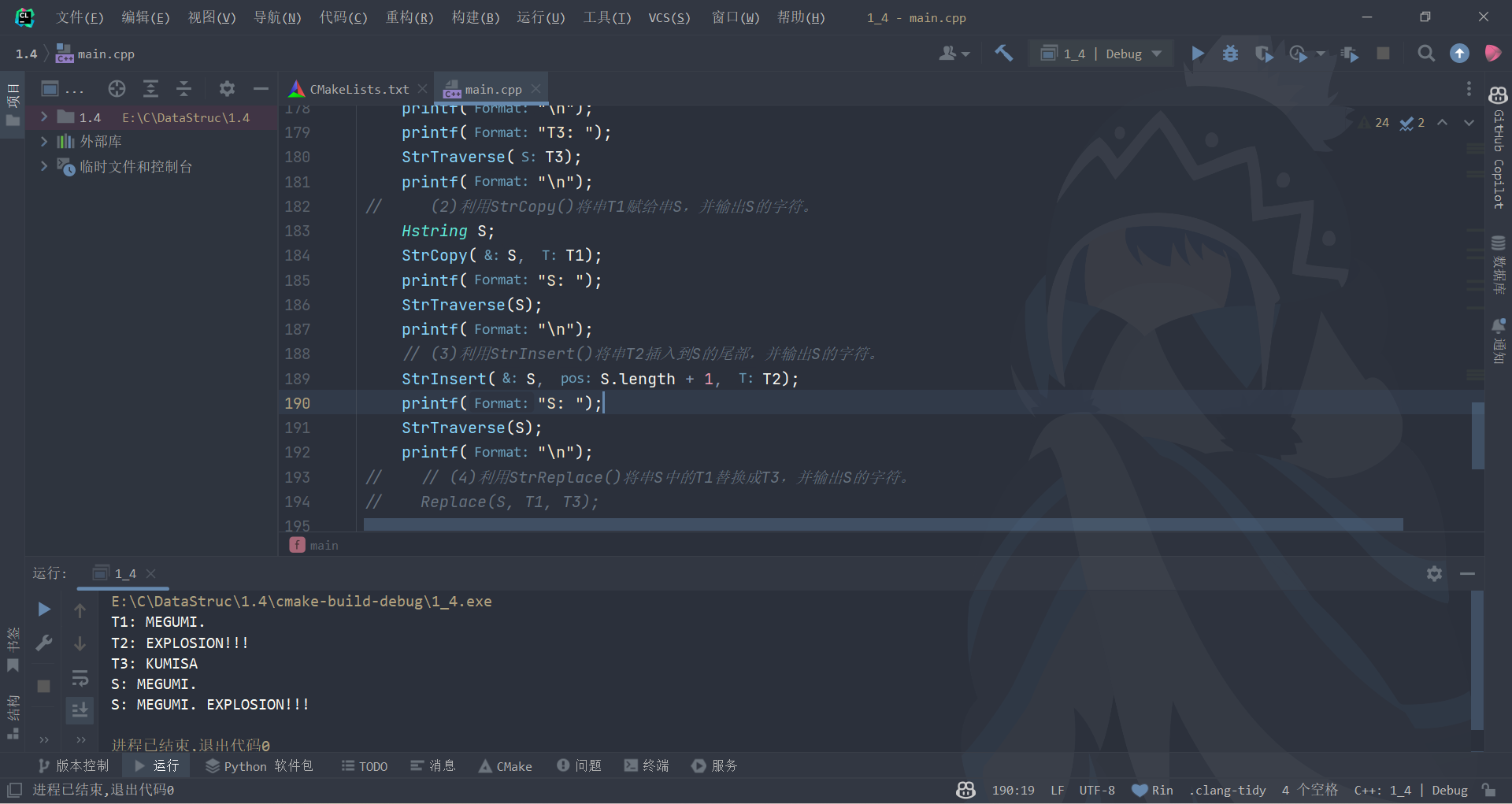
（1）数据准备。利用StrAssign()将设定的字符串常量（自行设定）赋给3个字符串变量T1,T2,T3，然后利用StrTraverse()输出字符串字符。



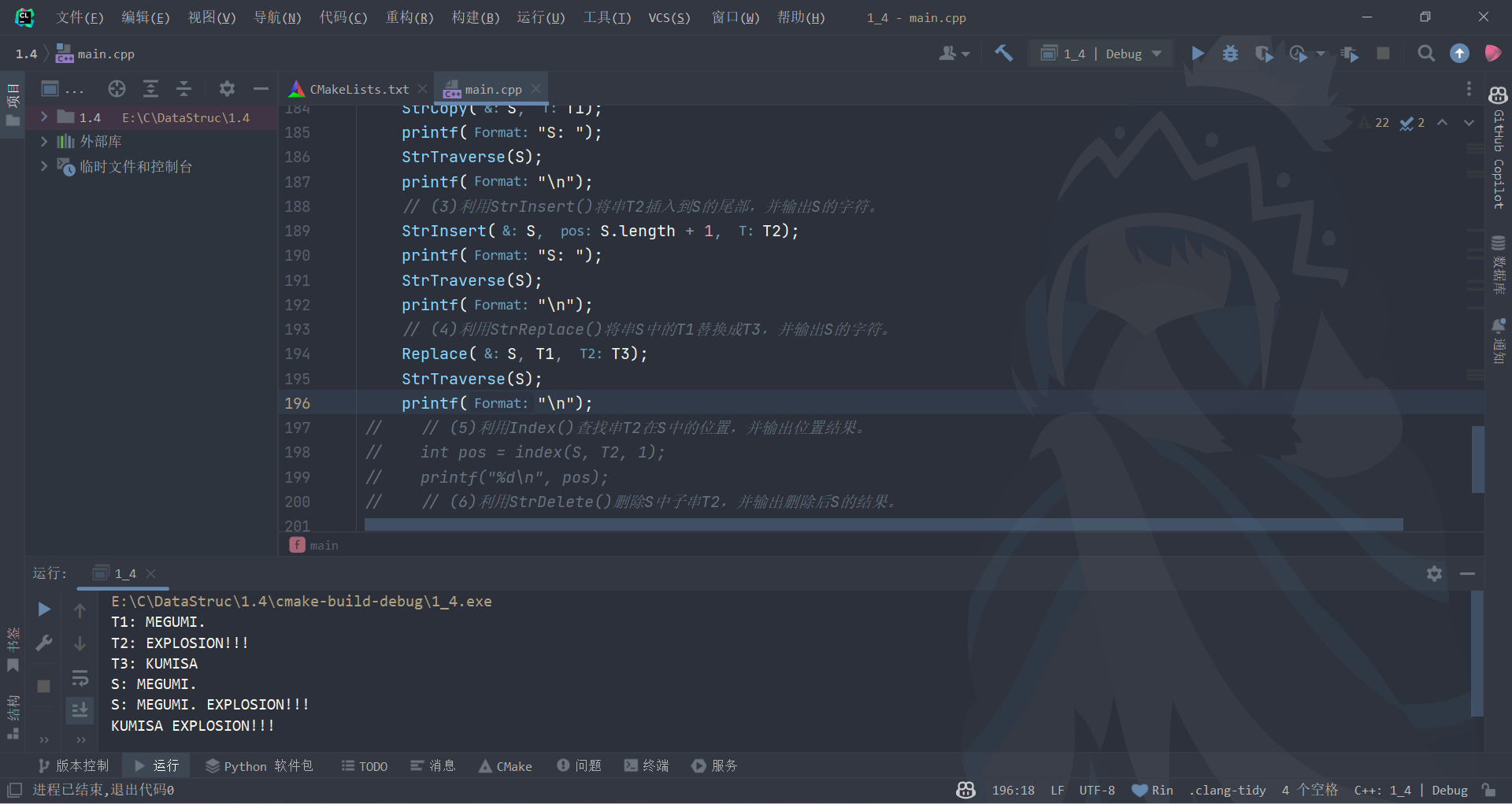
1. 利用StrCopy()将串T1赋给串S，并输出S的字符。



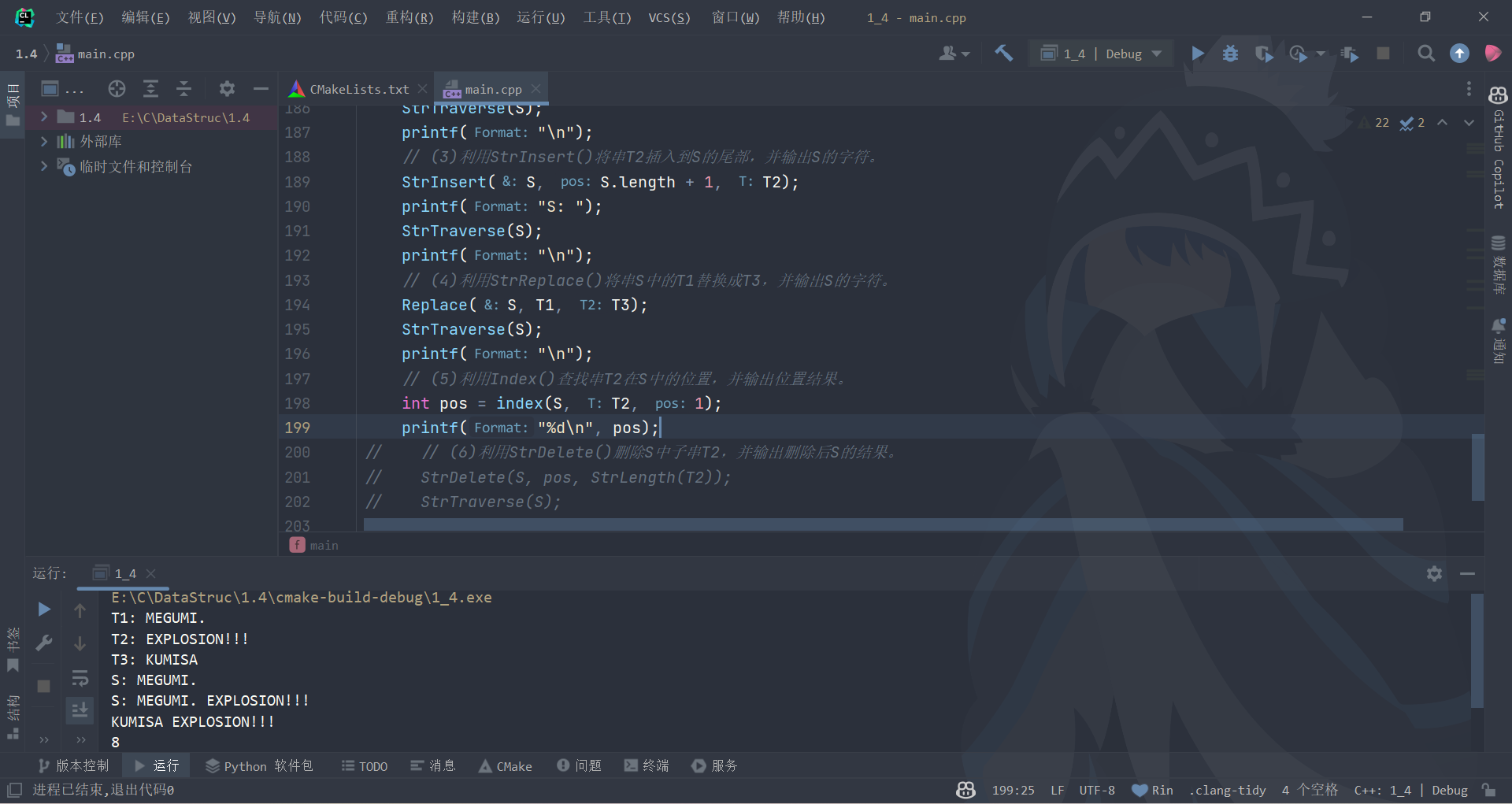
（3）利用StrInsert()将串T2插入到S的尾部，并输出S的字符。



1. 利用StrReplace()将串S中的T1替换成T3，并输出S的字符。



1. 利用Index()查找串T2在S中的位置，并输出位置结果。



1. 利用StrDelete()删除S中子串T2，并输出删除后S的结果

