// KaiserCode.cpp : 定义控制台应用程序的入口点。

//

/\* 第一周 第一题

题目：凯撒密码

编写一个能将单词进行加密和解密的程序。

加密原理：

凯撒加密其实是通过将明文中每个字符，

按照字符表中顺序，

全部后移K(K为密钥，K<字符表总数)位，

得到新的字符，即密文。

输入单词及偏移量，输出加密后的字符

developing 4

输出

hizipstmrk

编译环境：Visual Studio 2015

作者：杨政权

\*/

//实验代码

#include "stdafx.h"

#include <iostream>

using namespace std;

int main()

{

char array[1024]; //字符串数组

int choice; //选择加密或者解密

int key; //秘钥

cout << "请输入密匙 key ：" << endl; //提示用户输入秘钥

cin >> key; //读取用户输入的秘钥

do

{

cout << "请选择:1(加密) 0(解密)："; //选择加密或者解密

cin >> choice;

if (choice == 1)

{

cout << "请输入要加密的字符串：";

cin >> array;

cout << "加密后的密文是：" << endl;

for (auto n :array) //基于范围for循环 auto C++11新语法 自动推断变量类型

{

if (n <= 'z'&&n >= 'a') //用户输入的是小写字母

{

cout << (char)((n - 'a' + key) % 26 + 'a'); //输出加密之后的字符

}

else if (n <= 'Z'&&n >= 'A') //用户输入的是大写字母

{

cout << (char)((n - 'A' + key) % 26 + 'A'); //输入加密后的字符

}

}

}

else if (choice == 0)

{

cout << "请输入你的密文字符串：";

cin >> array;

cout << "加密以前的明文是：" << endl;

for (auto n:array)

{

if (n <= 'z'&&n>= 'a') //如果用户输入的该字符是小写

{

cout << (char)((n - 'a' + 26 - key) % 26 + 'a'); //输出解密后的字符

}

else if (n<= 'Z'&&n >= 'A')//如果用户输入的该字符是大写

{

cout << (char)((n - 'A' + 26 - key) % 26 + 'A'); //输出解密后的字符

}

}

}

else

cout << "你输入的命令有误，请重新输入："<< endl;

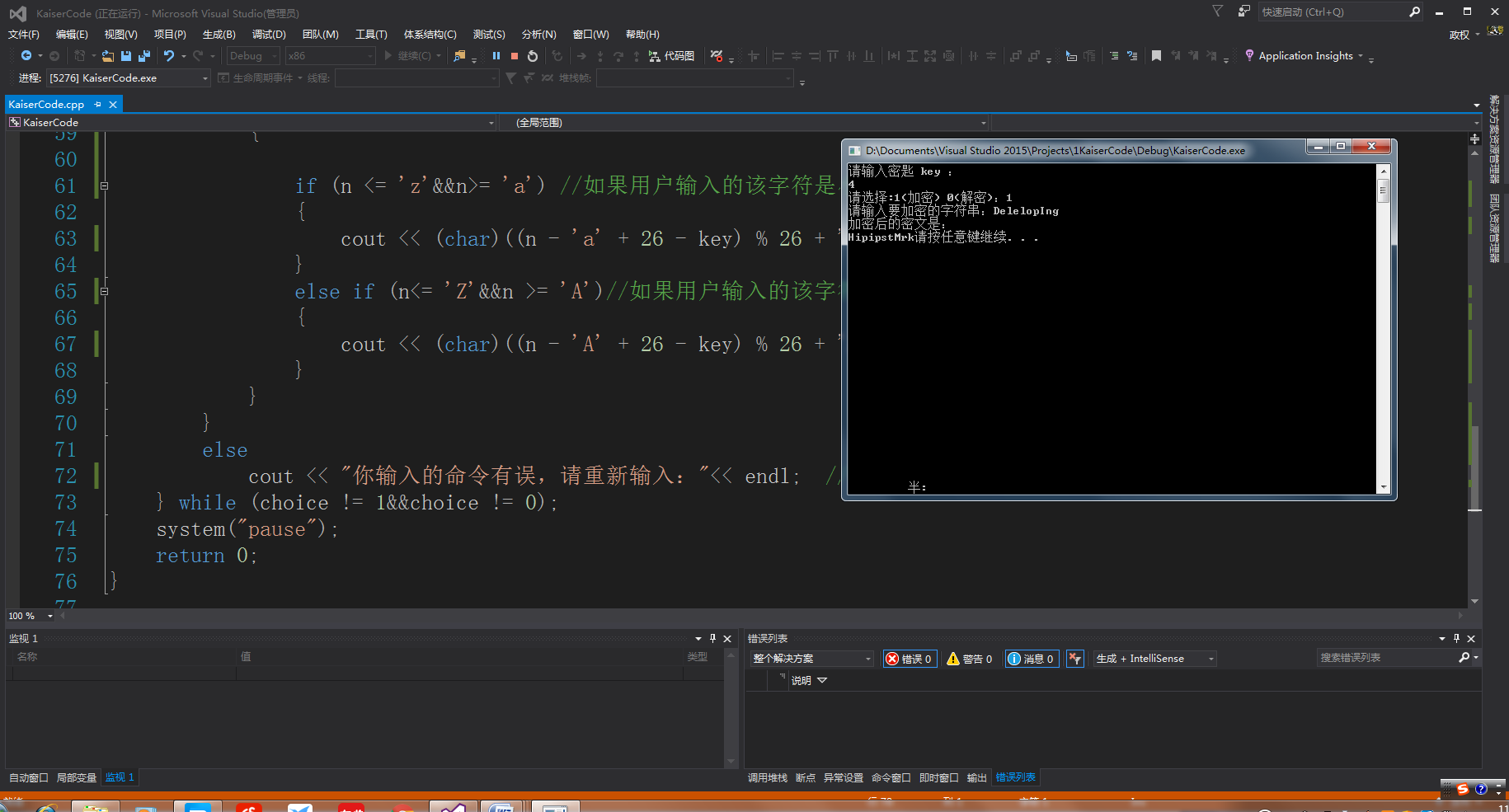
//命令不符合要求，重新输入命令

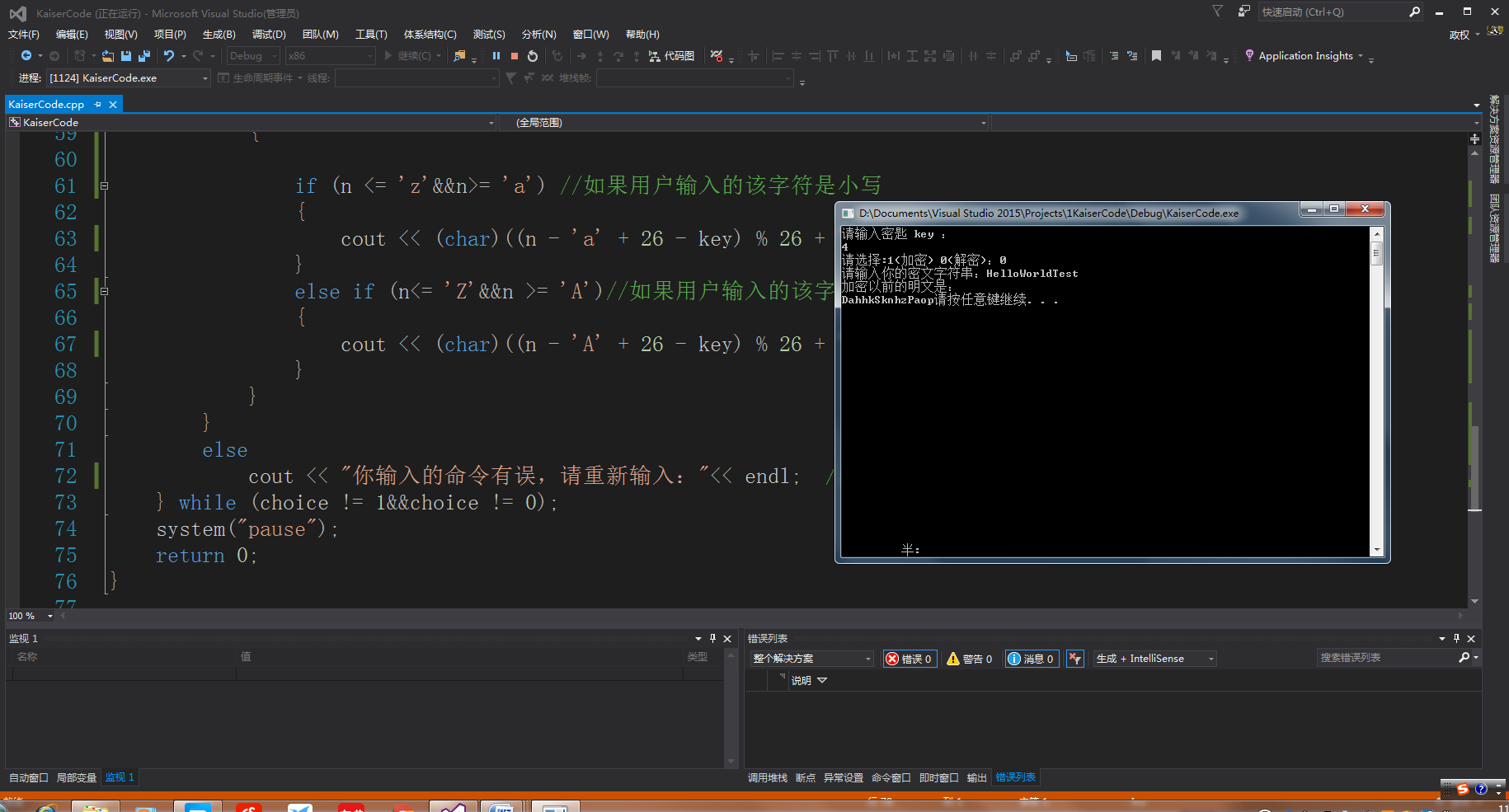
} while (choice != 1&&choice != 0);

system("pause");

return 0;

} //实验运行截图 ：加密截图

//实验运行截图 ：解密截图



//实验总结：

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

这个题目难度不大，算法挺简单的。

主要复习了一些基础语法。

写代码时有些基础的知识点遗忘了，

赶紧查资料学习。

好在之前有基础，捡起来很快。用C++11的新语法写起来代码简洁许多，很友好。

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/