程序设计方法与实践

——编程方法

编程方法简介

- 程序安装
- 创建项目
- 程序调试

程序安装

- 安装vs编程工具，安装语言选择中选择安装vc++。vc6/vs2010/vs2012/vs2015/vs2022...
  - 向下兼容，低版本建立的工程高版本可以打开升级，但是高版本的工程低版本打不开
  - 一台电脑可安装多个版本的vs
- 安装辅助工具 Visual Assist X。带有一些自动代码补全、变量/函数改名，变量引用查找等功能。
- UltraEdit编辑各种文本文件/UltraCompare比较整个程序目录
- 代码管理工具GitHub

创建项目

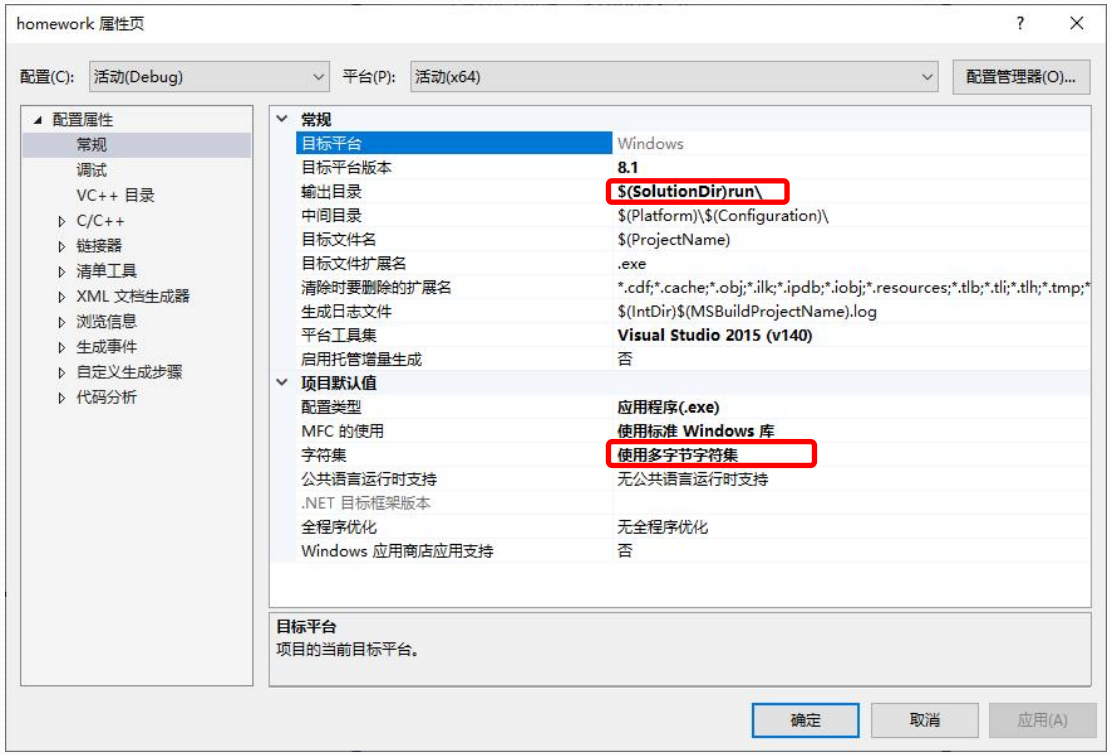
- 1、文件->新建->项目->VC++->Win32->Win32控制台应用程序->（附加选项中选）空项目
- 2、鼠标右键选择“解决方案视图”下的源文件目录，添加->新建项->c++文件（.cpp）
- 3、解决方案目录下建立目录run
- 4、写程序

```
#include <stdlib.h>
int main()
{
        //写内容
        return 0;
}
```

环境设置

- 1、项目右键点击属性打开环境设置
- 常规属性
  - 输出目录

- 默认是.\Debug\，改为run
- 可以把以后生成的exe直接放到run目录下，使用更方便



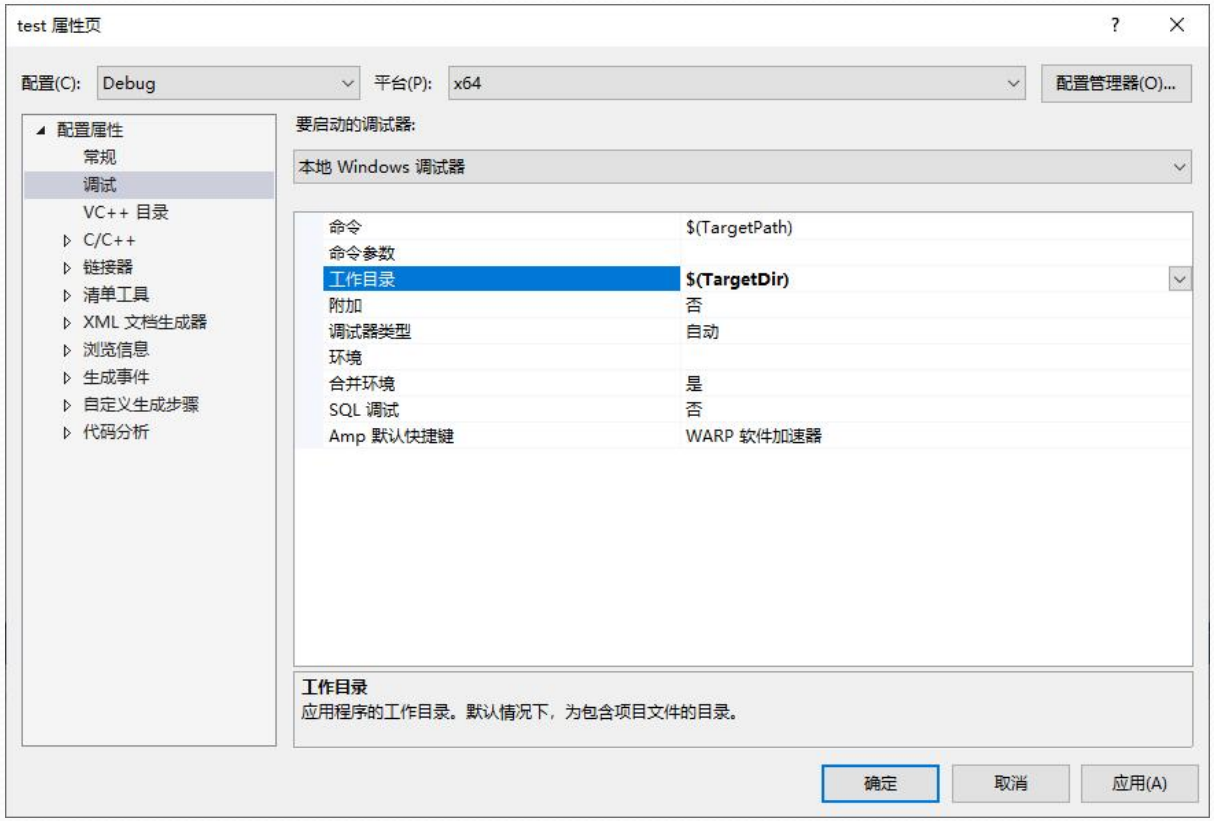
  - 字符集

- 默认是Unicode，改为多字节字符集
- 处理字符串更方便

环境设置

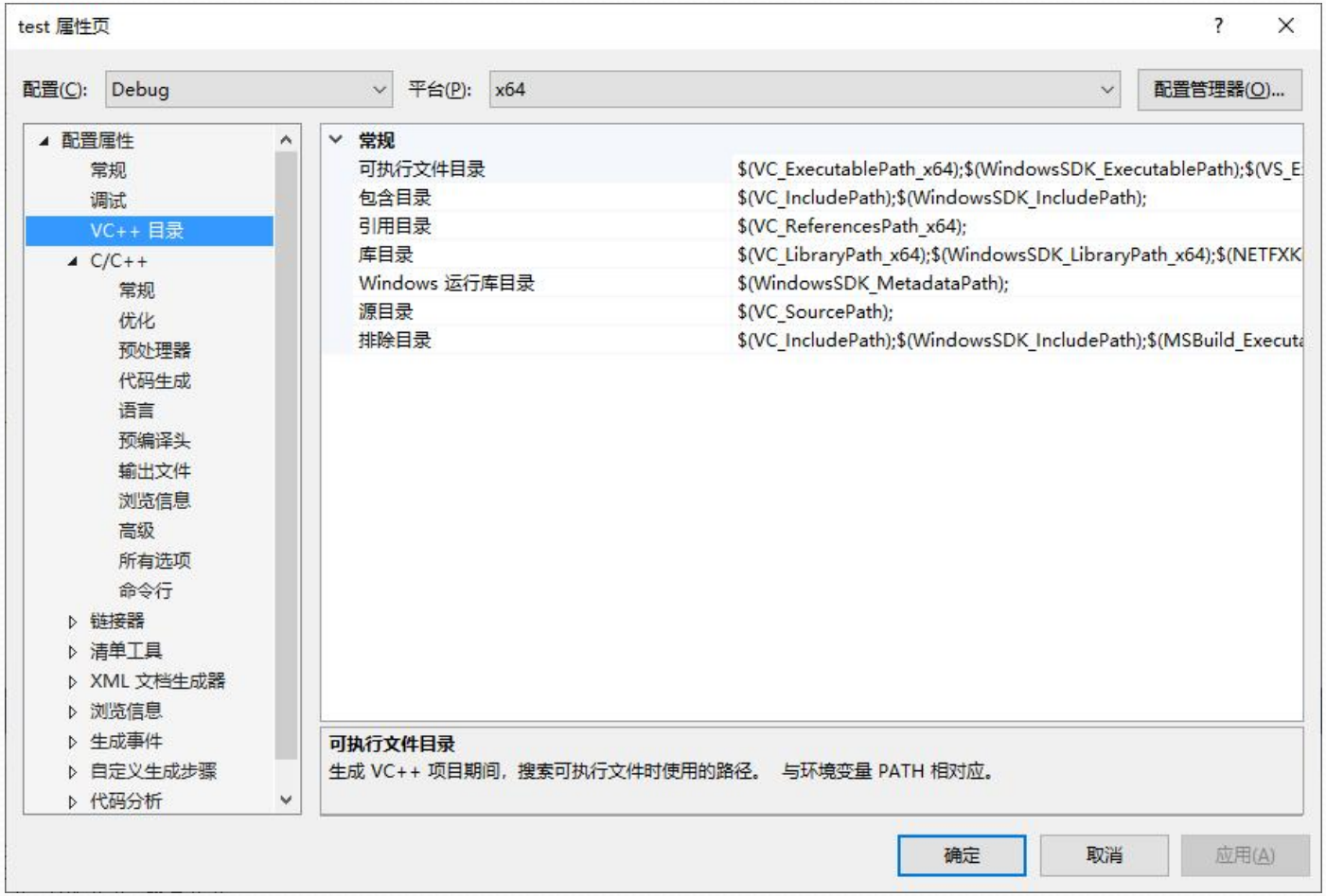
- 1、项目右键点击属性打开环境设置
- 调试操作
  - 命令
  - 命令参数
  - 工作目录

- **默认是$(ProjectDir)**
- **改为$(TargetDir)**



环境设置

- VC++目录——设置所有工程搜索路径
  - 包含目录
    - .h文件
  - 库目录
    - .lib文件

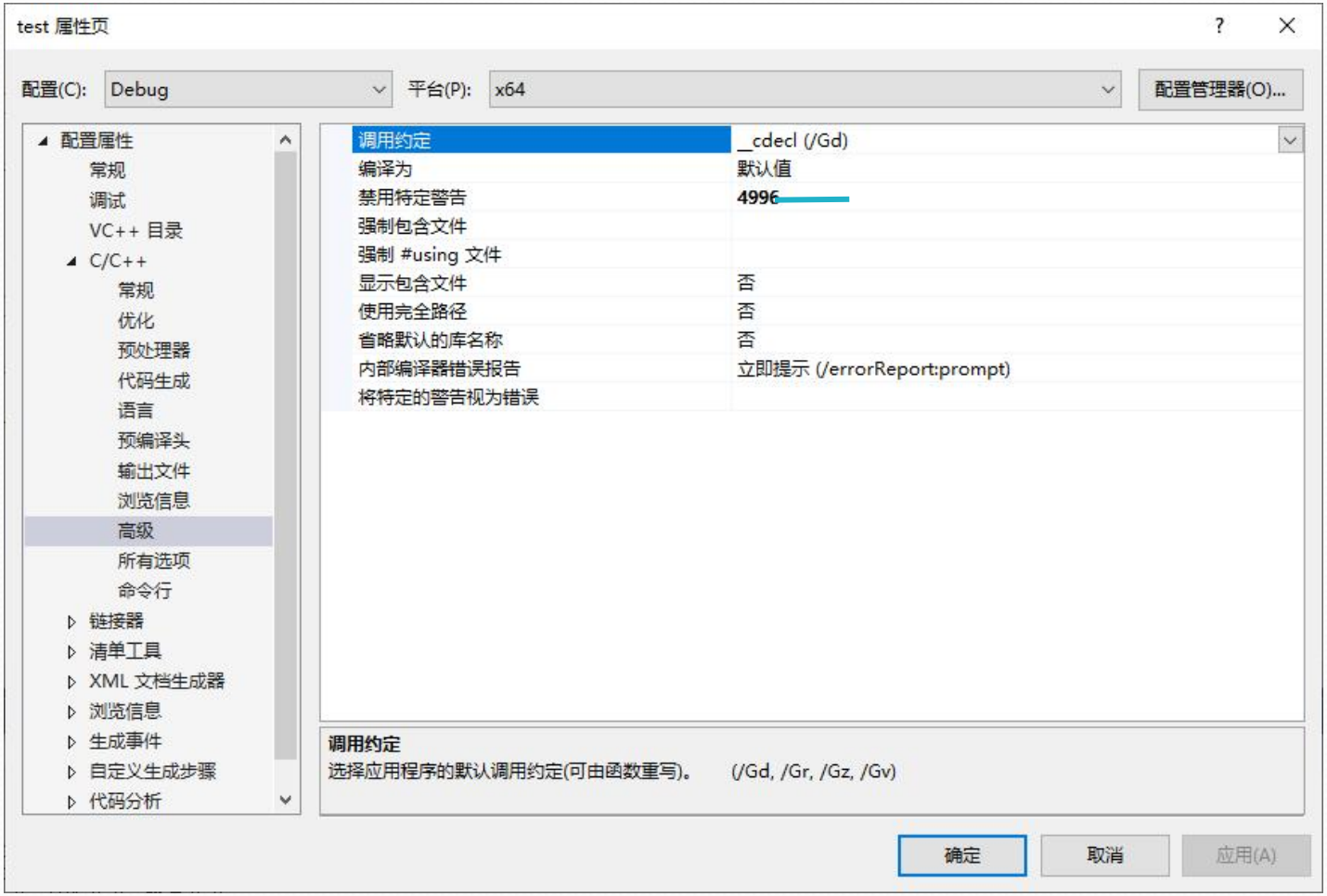


环境设置

- C/C++

编译相关

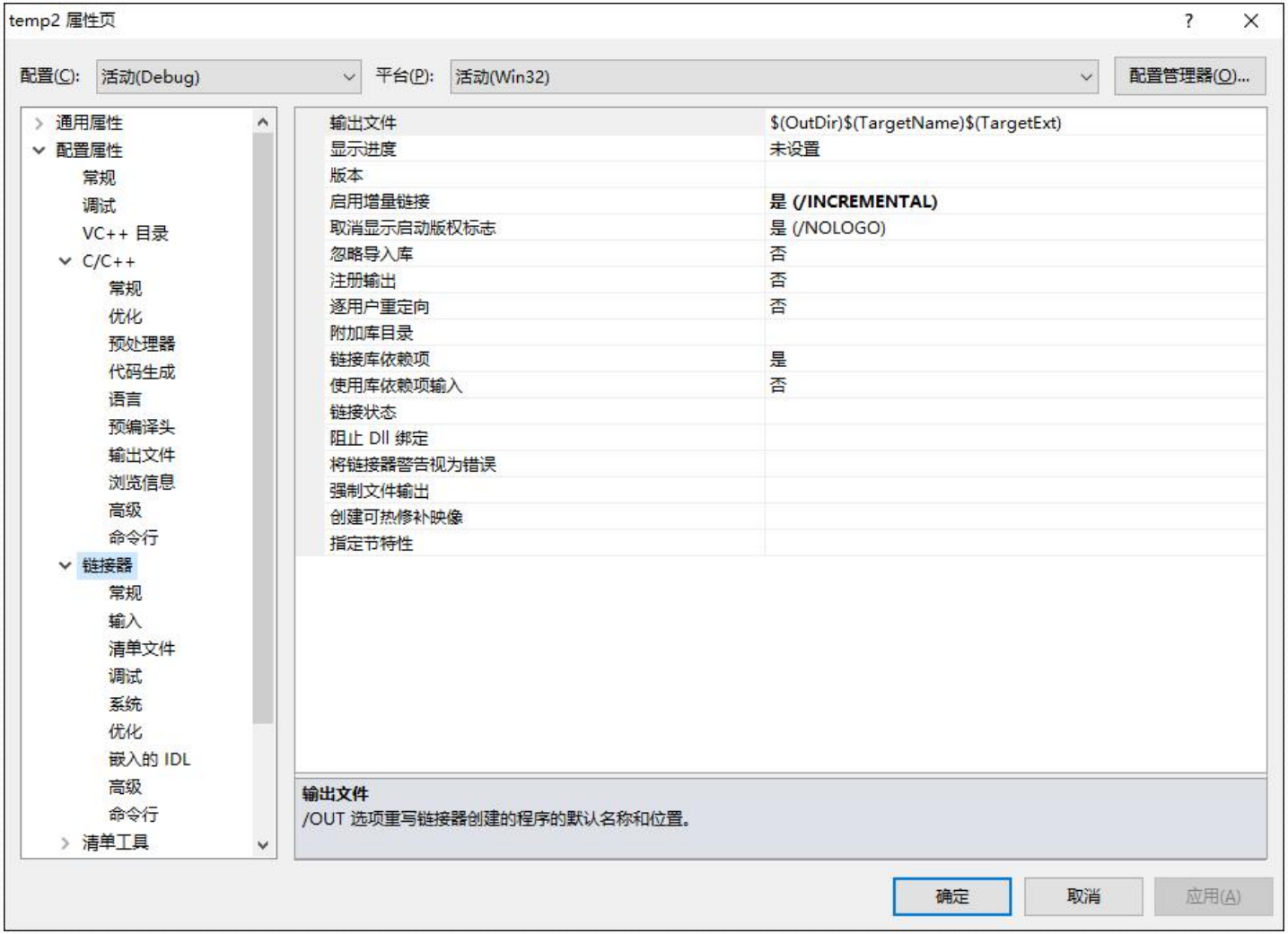
禁止特定警告



环境设置

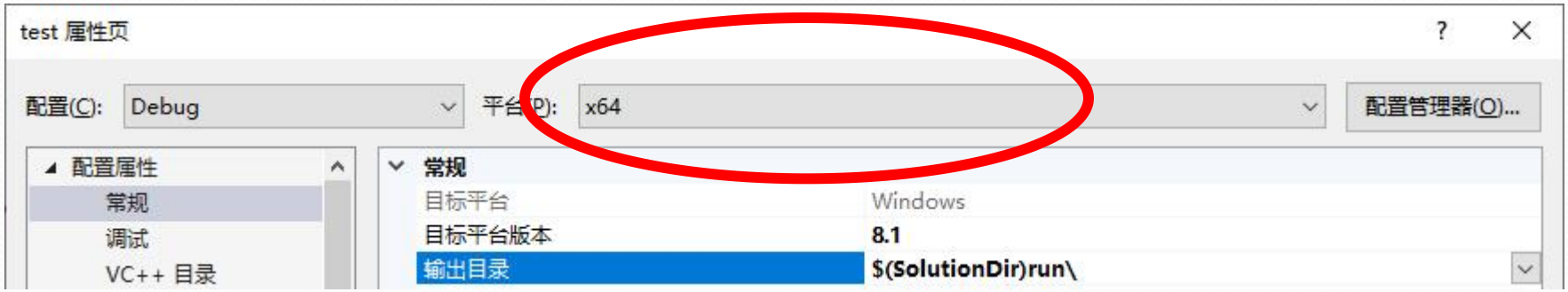
- 链接器

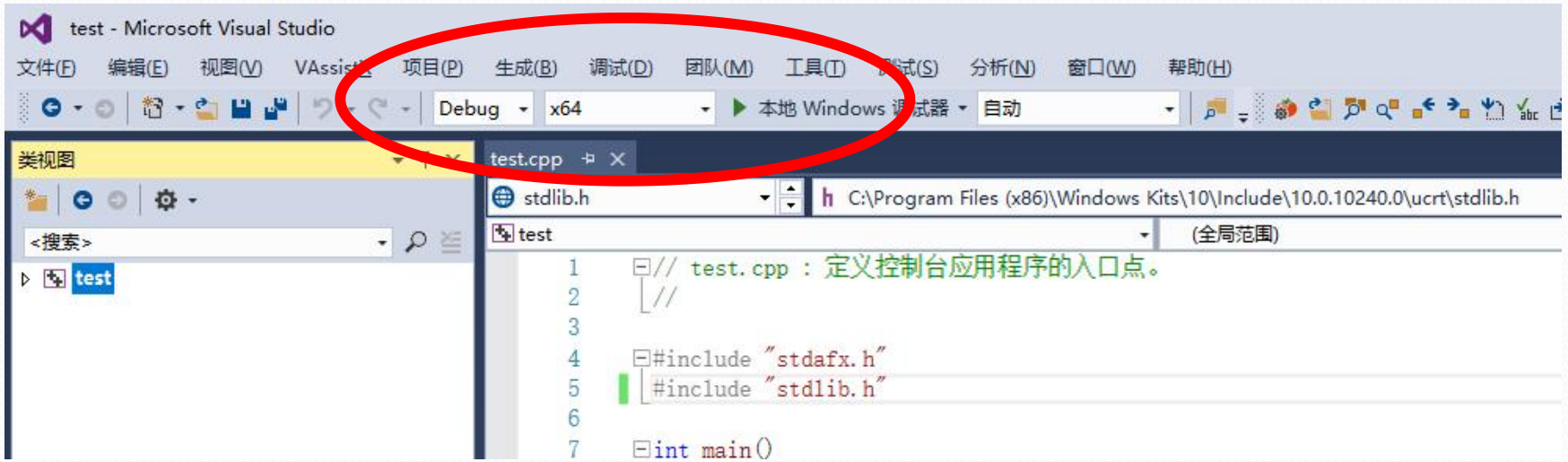
链接相关



环境设置

- 注意：配置属性选择平台与程序执行的平台一致。





调试程序的方便设置：

- 1、增加简单计时功能
- 2、设置输入重定向
- 3、*system*(“pause”);程序结尾加这个可以暂停

```
freopen(filename, "r", stdin); //输入重定向到文件
int t1 = GetTickCount();  //计时功能

中间写自己的程序

int t2 = GetTickCount() - t1;
printf("\n---------------\n总计时：%d\n", t2);  //计时功能
freopen("con", "r", stdin);   //恢复输入定向到键盘
system("pause");  //程序退出时候暂停
return 0;
```

调试程序的方便设置：

- 4、多次作业统一编写调试
  - 课程作业都比较小，大部分用2个函数以内就可以完成。
  - 多次作业有多个main函数，放在一个工程会出错。
  - 每次作业都类似的函数名，要不能互相干扰。
  - **使用命名空间**

```
namespace zy1
{
    char *filename = "data1.txt";
#include "stdio.h"
    int AddTwoNumber(int a, int b)
    {
        int result = a + b;
        return result;
    }
    int main()
    {
        int c = AddTwoNumber(2,3);
        printf("result = %d\n", c);
        return 0;
    }
}
#define CURZY zy1
int main()
{
    freopen(CURZY::filename, "r", stdin);
    int t1 = GetTickCount();
    CURZY::main();
    int t2 = GetTickCount() - t1;
    printf("\n---------\n总计时：%d\n", t2);
    freopen("con", "r", stdin);
    system("pause");
    return 0;
}
```

常用的调试快捷键

- F5执行
- F9加断点
- F10单步执行
- F11执行到函数中
- Ctrl+Tab文件切换显示
- Ctrl+F2切换书签，F2跳转下一书签

（vs2012以后默认是Ctrl+K+K,Ctrl+K+N）

可以设置修改快捷方式：工具->选项->环境->键盘->可以编辑快捷方式

- 编辑.切换书签　　改成Ctrl+F2
- 编辑.下一书签　　改成F2,

调试过程

- 1、写好程序，实现全部或者部分功能，可编译通过
- 2、F5执行看看是否达到需要效果
- 3、可能出错的地方加断点F9
- 4、F5执行到断点处，监视窗口查看变量值
- 5、单步执行F10
- 6、进入函数执行F11
- 7、修改代码或修改变量值，按F5继续执行。
  - “编辑并继续”
- 8、增加断点/编辑断点F9,按F5执行

调试过程

- 9、在某个循环内部满足条件时候进入断点：
  - 在需要加入断点的地方写判断条件

  进入断点后使用F10进行单步调试。

```
for(int i=0;i<10000;i++)
{
        if(i==23) //条件按需要写
        {
            int z=0; //在这里加断点。
        }
          ...    //后面是需要调试的代码
}
```

- 10、代码某位置建立标签，以后可以按F2在多个标签内跳转。进行代码对比很方便
- 11、调试信息输出

调试过程

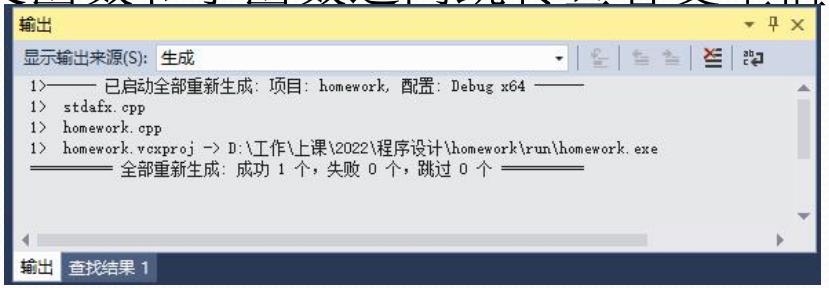
- 11、调试信息输出
  - 1)单步执行时候查看监视窗口——最常用
  - 2)控制台程序可直接打印到屏幕——常用（不需断点）
  - 3)如果是windows程序可以使用MessageBox命令
  - 4)用TRACE命令输出到调试窗口——常用（不需断点）
  - 5)自己写log文件输出
  - 6)assert、throw/try/catch异常检测…

调试过程中主要观察的对象

- 1、监视窗口
  - 看增加多个。调试->窗口->监视
  - 把要查看的变量拖动到监视窗口
  - 可手动修改监视窗口的变量名，变量类型，甚至修改变量值
  - 鼠标右键可以选择16进制观察

- 2、显示堆栈窗口
  - 查看当前代码执行的位置，
  - 鼠标选择堆栈不同层次，在父函数和子函数之间跳转查看变量值

- 3、输出窗口
  - 编译的错误信息在这里
  - TRACE输出的内容到这里



- 4、内存窗口，查看具体内存数据的变化

一些程序的文本编辑功能

- 1、文件中查找：可以在整个工程中查找变量或者字符串，查找结果放入“查找结果窗口”，可通过鼠标点击在多个结果之间跳转，常用。
- 列选择模式，按下Alt键并用鼠标选择一个区域，可以按块选择代码进行批量修改拷贝等，很方便。
- 选中一个区域按F8可以对这个区域代码缩进自动整理(vs2012之后Ctrl+K+D)
- 手动修改代码缩进：按Tab实现对选中区域右移，Shift Tab是选中区域左移
- Ctrl Tab是在打开的几个文件中切换。