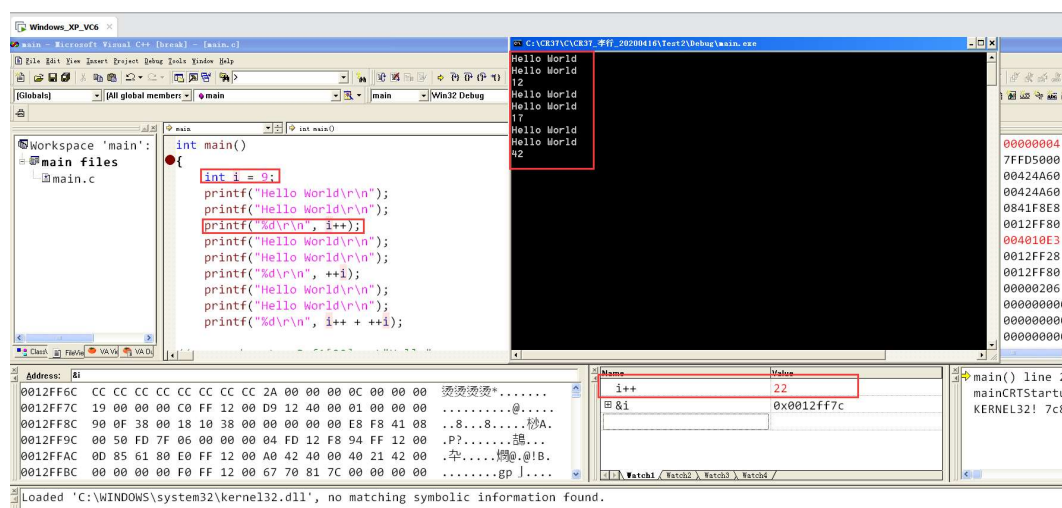


2020/04/16_第12课_预处理、条件编译

笔记本: C
创建时间: 2020/4/16 星期四 16:10
作者: ileemi
标签: ASCII和Unicode, 条件编译, 预处理

- [宏](#)
- [宏使用的场合](#)
 - [宏参数的链接](#)
- [条件编译](#)
- [字节编码](#)

VC++ 6.0的watch窗口支持表达式求值



在Debug调试程序时在watch窗口对变量++，其程序中的每条语句都会对表达式进行求值。

宏

预处理指令，用于程序员和编译器进行交互。

预处理指令是写给编译器看的，而不是写给处理器看的。

宏需要大写，每个宏参数记得加括号

宏使用的场合

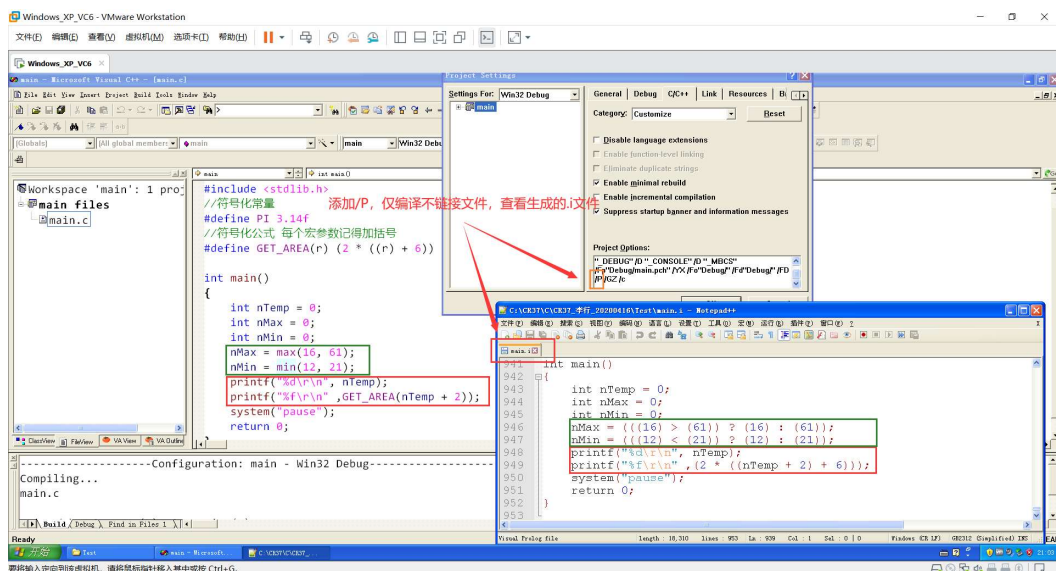
1、符号化常量（常量宏），让数值或者其它类型的常量更具有说明性意义。

```
#define PI 3.14f //符号化常量
```

2、符号化公式（表达式宏）

```
#include <stdio.h>
#include <stdlib.h>
//符号化常量
#define PI 3.14f
//符号化公式
#define GET_AREA(r) 2*r

int main()
{
    int n = 6;
    printf("%f\\r\\n", GET_AREA(n + 2));
    system("pause");
    return 0;
}
```



宏的机制：对代码文本的查找替换

max宏返回两个数的最大值

3、代码块宏

用来完成一块代码段

禁忌：

- 后面不加分号
- #define 后的参数部分不要加 {} 号，会引发其它问题
- {}交给编写代码者，编写的时候记得添加块

```
#define OUTPUT(s) printf(s);printf("\\r\\n")
```

```
#define DEF_INT(n) int_n
```

```

#include <stdio.h>
#include <stdlib.h>
// 1、符号化常量
#define PI 3.14f
// 2、符号化公式 每个宏参数记得加括号
#define GET_AREA(r) (2 * ((r) + 6))
// 3、代码块宏，参数末尾不要加分号以及{ }
#define OUTPUT(s) printf(s);
#define OUTPUT(s) printf(s);printf("World\r\n")
int main()
{
    int i = 0;
    for (i = 0; i < 2; i++)
    {
        OUTPUT("Hello");
    }
    system("pause");
    return 0;
}

```

4、说明性宏

没有宏的实现部分，只有符号部分

```

#define IN
#define OUT
int Fun(IN int n, OUT int ary[])
{
    //TODO
    return 0;
}
int main()
{
    _CRTIMP
    afx_msg;    消息宏，起到消息声明
    OnClick;
    return 0;
}

```

5、兼容性宏，牵一发动全身（为了兼容不同版本的的开发环境的）

```

#define INT int //32位
#define INT long int //16位
//为了让以前的代码不加修改进行编译，可定义空宏

```

```

#define near
#define far
int main()
{
    INT nTemp1;
    INT nTemp2;
    INT nTemp3;
    INT nTemp4;

    //16位下, 关键字: near far
    //指针分两类: 段内指针、段外指针
    int near *p = NULL; //2byte 段内指针, 短指针, 近指针

    int far *lp = NULL; //4byte 段外指针, 长指针, 远指针(可以跨段)

    //短指针到32位能兼容吗?
    //不兼容, 也无法定义

    return 0;
}

```

VC++ 6.0 .cpp下for语句控制循环语句的变量Bug

```

#include <stdio.h>
#include <stdlib.h>
#define for if(1)for //将for循环至于if 块内, 该语句不产生判定
代码
#define for if(0){}else for //将for循环至于if 块内
int main()
{
    //编译不通过 n重复定义
    for(int i = 0; i < 3; i++)
    {
        printf("Hello\r\n");
    }
    for(int i = 0; i < 3; i++)
    {
        printf("world\r\n");
    }
    //VS 2019就可以

    system("pause");
    return 0;
}

```

6、编译器的内置宏

宏不支持注释，字符串内相关宏名的替换

全字匹配

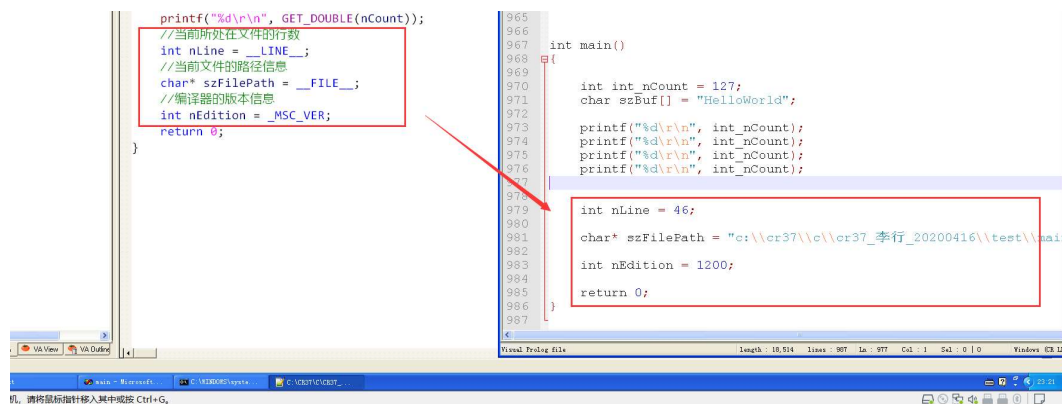
内置宏，例如：

判断当前文件的是.cpp文件还是.c文件

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
    int isC++ = __cplusplus;
    //当前所在文件的行数
    int nLine = __LINE__;
    //当前文件的路径信息
    char* szFilePath = __FILE__;
    //编译器的版本信息
    int nEdition = _MSC_VER;

    return 0;
}
```



宏参数的链接

```
#include <stdio.h>
#include <stdlib.h>

// # 号修饰的宏参数，表示将宏参数设置成字符串
#define MYSTRING(s) #s

// ## 宏参数连接符
#define DEF_INT(n) int int_##n
#define GET_INT(n) int_##n

#define DEF_FLOAT(n) int int_##n
```

```

#define GET_FLOAT(n) int_##n

#define DEF_DOUBLE(n) int int_##n
#define GET_DOUBLE(n) int_##n

int main()
{

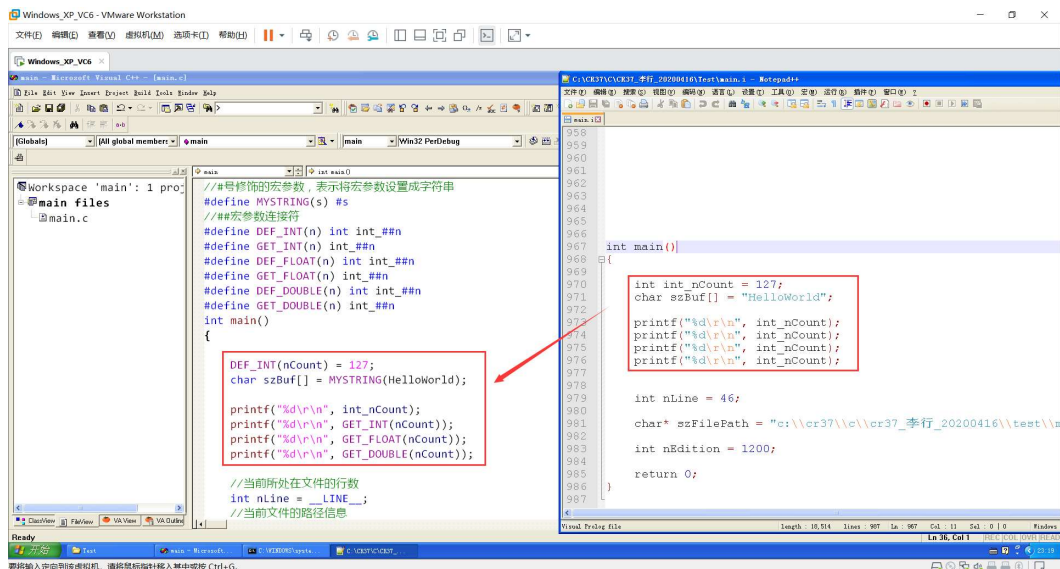
    DEF_INT(nCount) = 999;

    char szBuf[] = MYSTRING>HelloWorld);

    printf("%d\r\n", int_nCount);
    printf("%d\r\n", GET_INT(nCount));
    printf("%d\r\n", GET_FLOAT(nCount));
    printf("%d\r\n", GET_DOUBLE(nCount));

    return 0;
}

```



条件编译

关键字:

#define

#if

#ifdef

#ifndef

#else

#endif

```

int main()
{
    //不产生流程控制代码，不属于分支语句，是对代码文件的预处理
}

```

```

    // #define IS_DEBUG 1
    // #define _DEBUG
    for (int i = 0; i < 6; i++)
    {
#ifdef _DEBUG
        printf("%s:%d i = %d\r\n", __FILE__, __LINE__, i);
#else
        printf("i = %d\r\n", i);
#endif

    }

    system("pause");
    return 0;
}

```

编译器内置宏可以自定义

cl /c /D"xxxx" xxxx为需要自定义的宏

字节编码

Unicode通常用两个字节表示一个字符，原有的英文编码从单字节变成双字节，只需要把高字节全部填为0就可以。

简单使用

```

#include <stdio.h>
#include <stdlib.h>
#include <locale.h> //包含setlocale
int main()
{
    wchar_t wszBuf1[32] = L"Hello";
    wchar_t wszBuf2[32] = L"World\r\n";

    setlocale(LC_ALL, "chinese");
    wprintf(wszBuf1);    //需要设置地区
    wprintf(wszBuf2);    //需要设置地区

    _wsystem(L"pause");
    return 0;
}

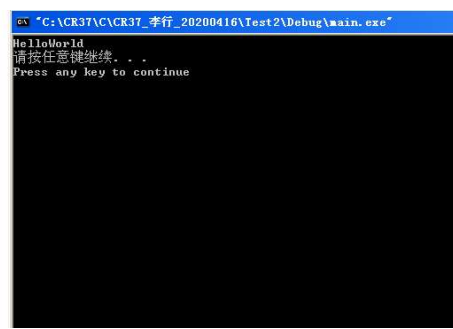
```

```
//      printf( "Hello World\r\n" );
//      printf( "%d\r\n", i++ + ++i);

wchar_t wszBuf1[32] = L"Hello";
wchar_t wszBuf2[32] = L"World\r\n";

setlocale(LC_ALL, "chinese");
wprintf(wszBuf1); //需要设置地区
wprintf(wszBuf2); //需要设置地区

_wsystem(L"pause");
return 0;
}
```



将下面的ASCII编码转换成Unicode编码

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <locale.h>

int main()
{
    /*
    char szBuf1[32] = "Hello";
    char szBuf2[64] = { 0 };
    strcpy(szBuf2, szBuf1);
    strcat(szBuf2, "World\r\n");

    printf(szBuf2);
    system("pause");
    */

    wchar_t wszBuf1[32] = L"Hello";
    wchar_t wszBuf2[64] = { 0 };
    wcsncpy(wszBuf2, wszBuf1);
    wcscat(wszBuf2, L"World\r\n");
    /*
    字符串前需要添加 L, 否则会报一下错误
    incompatible types - from 'char [6]' to 'const unsigned short *'
    不兼容的类型-从 "char[6]" 到 "const unsigned short*"
    */
    wprintf(wszBuf2);
    _wsystem(L"pause");
    return 0;
}
```

如何把ASCII字符集和Unicode宽字符集在一个文件内都存在，可以修改相应的编译选项即可编译对应的版本
使用条件编译即可。

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
```



```

#include <locale.h>

//是否定义UNICODE
#ifdef UNICODE
//定义UNICODE, 执行这块
#define _tmain      wmain
#define TCHAR       wchar_t
#define _tcscpy      wcscpy
#define _tscat       wscat
#define _tprintf     wprintf
#define _tsetlocale  wsetlocale
#define _tsystem     _wsystem
#define __T(x)       L##x
//解决L问题

#else
//未定义UNICODE, 执行这块
#define _tmain      main
#define TCHAR       char
#define _tcscpy      strcpy
#define _tscat       strcat
#define _tprintf     printf
#define _tsetlocale  setlocale
#define _tsystem     system
#define __T(x)       x

#endif

int _tmain()
{
    TCHAR wszBuf1[32] = __T("Hello");
    TCHAR wszBuf2[64] = { 0 };
    _tcscpy(wszBuf2, wszBuf1);
    _tscat(wszBuf2, __T("World\r\n"));
    /*
    字符串前需要添加 L, 不然会报一下错误
    incompatible types - from 'char [6]' to 'const unsigned short *'
    不兼容的类型-从 "char[6]" 到 "const unsigned short*"
    */
    _tprintf(wszBuf2);
    /*
    使用unicode编码输出中文时需要添加setlocale(), 设置时区
    */
    _tsetlocale(LC_ALL, __T("chinese"));
    TCHAR wszbuf3[64] = __T("Hello你好World世界\r\n");
    _tsystem(__T("pause"));
}

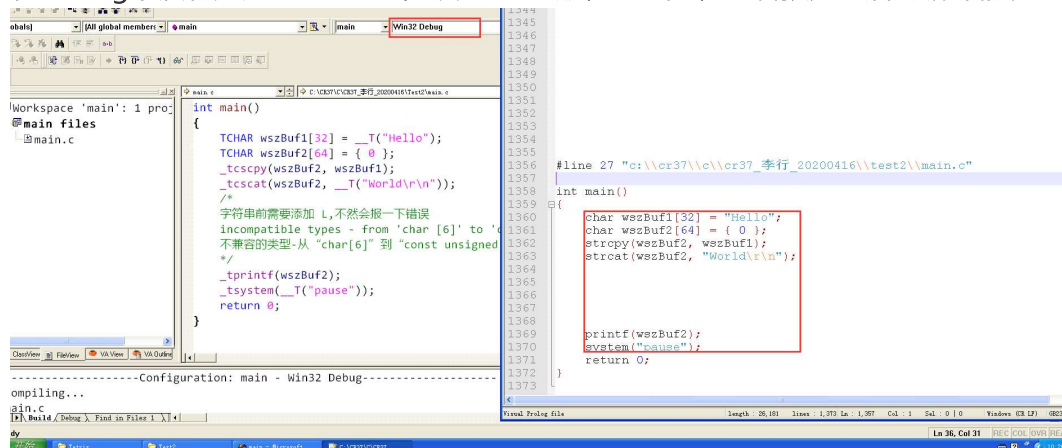
```

```

return 0;
}

```

在Debug下没有定义UNICODE，就是ASCII版，通过/P，查看预处理后的编译情况



在Debugunicode下定义UNICODE，就是unicode版，通过/P，查看预处理后的编译情况

