06 How the C++ Compiler Works

1. Abstract syntax tree

编译器的工作是把代码转化为*constant data*(常数资料)和*instructions*(指令)，构建抽象语法树后开始生产代码

cpp中没有文件的概念，文件只是给编译器提供源码的一个方式

.cpp告诉编译器用c++编译→编译器当成一个*translation unit*→得到一个.obj

常见的preprocessor语句`pragma statement`

```
include
define
if
ifdef
```

常见的Preprocessor statement

2. 1 #include

`#include`就是把想要的文件中的内容复制进来.比如创建了一个EndBrace.h,2其中的内容是：

C++

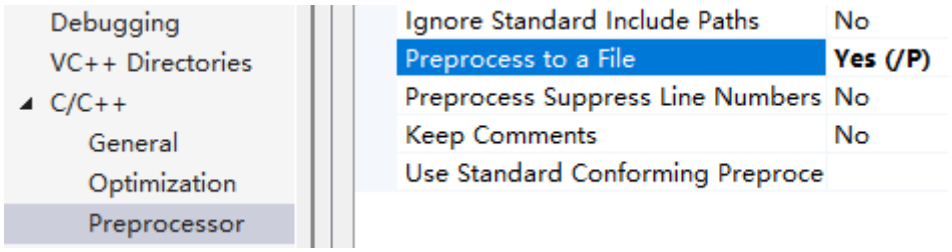
```
}
```

则我的Multiple函数这么写仍可以正常编译

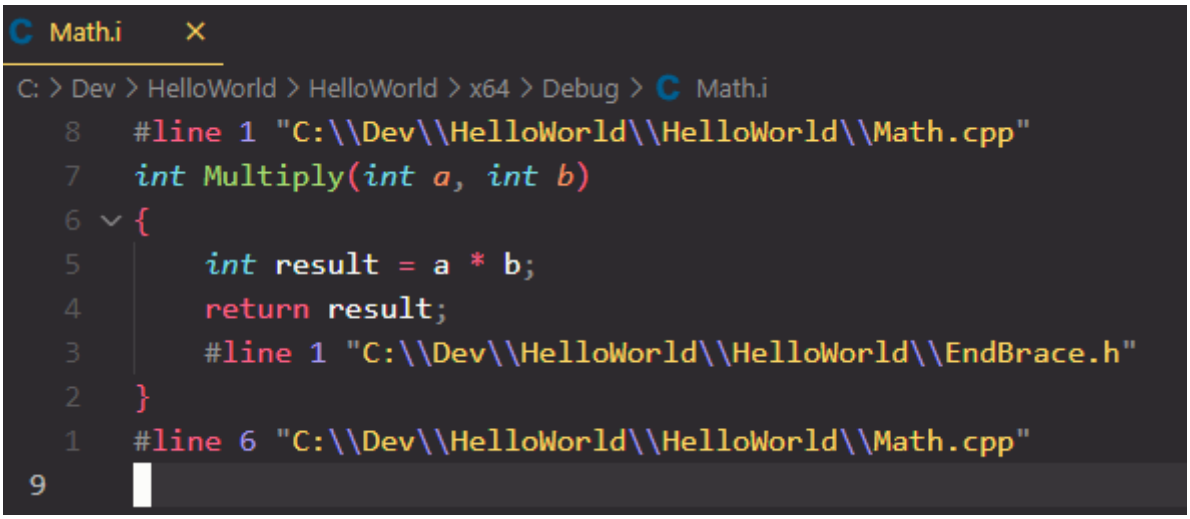
C++

```
int Multiply(int a, int b)
{
    int result = a * b;
    return result;
    #include "EndBrace.h"
```

检查Preprocess的文件，可通过如下操作生成.i文件



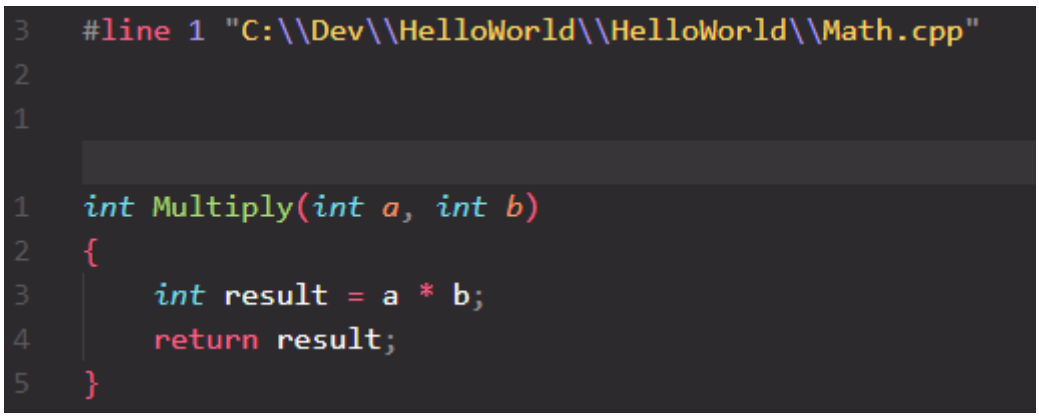
Math.i文件内容如下



2.2 #define

```C++
#define INTEGER int
```

搜索INTEGER这个词，然后替换成后面的东西



2.3 #if

可以让我们依据特定条件包含或剔除代码

```C++
//剔除
#if 0
int Multiply(int a, int b)
{
    int result = a * b;
    return result;
}
#endif
```

```
#line 1 "C:\\Dev\\HelloWorld\\HelloWorld\\Math.cpp"



#line 9 "C:\\Dev\\HelloWorld\\HelloWorld\\Math.cpp"
```

可将.obj转化为可读版本的.asm文件，阅读二进制用了vscode插件Hex Editor

3. 编译器优化

3.1 简单案例

```C++
int result = a * b
return result

==>

return a * b
```

3.2 constant folding

```C++
return 5 * 2

==>

return 10
```

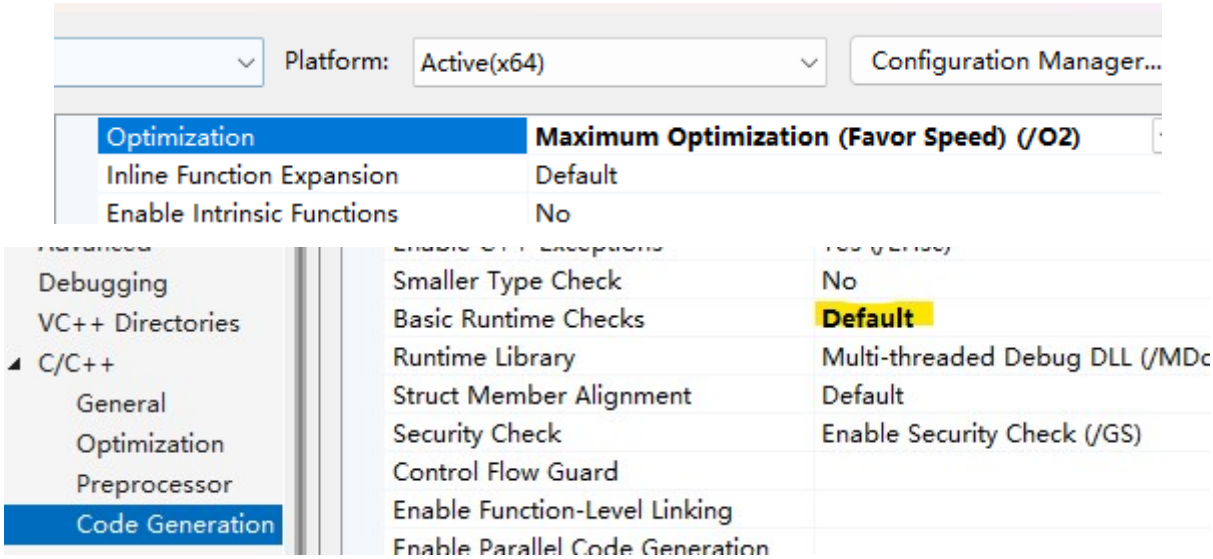
```
call    __CheckForDe
mov eax, 10
Line 4
```

3.3 函数签名和call

call即调用函数，通过图中的符号修饰对函数做唯一签名认真，让linker按函数签名寻找函数

```
lea rcx, OFFSET FLAT:??_C@_
call    ?Log@@YAPEBDPEBD@Z
mov eax, 10
```

修改编译器，可以自动优化汇编指令


Platform:
Active(x64)
Configuration Manager...
Optimization
Maximum Optimization (Favor Speed) (/O2)
Inline Function Expansion
Default
Enable Intrinsic Functions
No
Debugging
VC++ Directories
C/C++
General
Optimization
Preprocessor
Code Generation
Smaller Type Check
No
Basic Runtime Checks
Default
Runtime Library
Multi-threaded Debug DLL (/MDc
Struct Member Alignment
Default
Security Check
Enable Security Check (/GS)
Control Flow Guard
Enable Function-Level Linking
Enable Parallel Code Generation