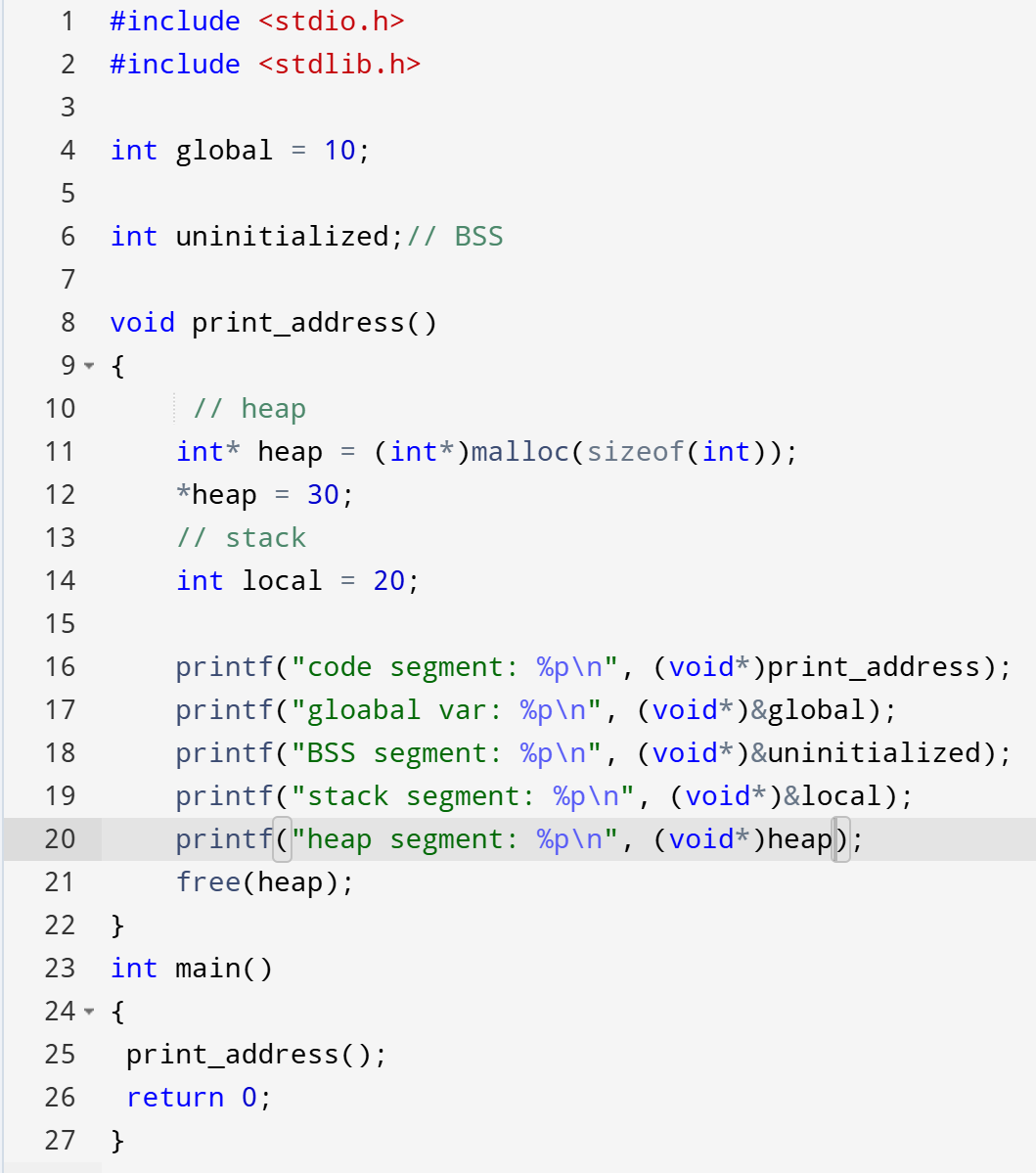
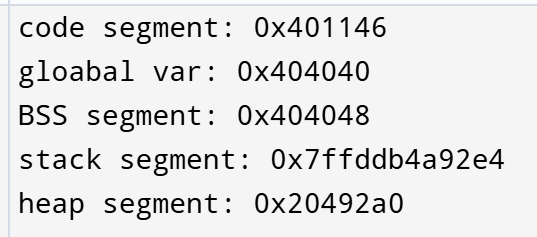
# 编译器对栈的使用分析

实验环境：Visual Studio 2022，MASM

问题1：系统程序内存空间分布情况，如代码部分、全局变量部分、堆区、栈区等排列顺序；

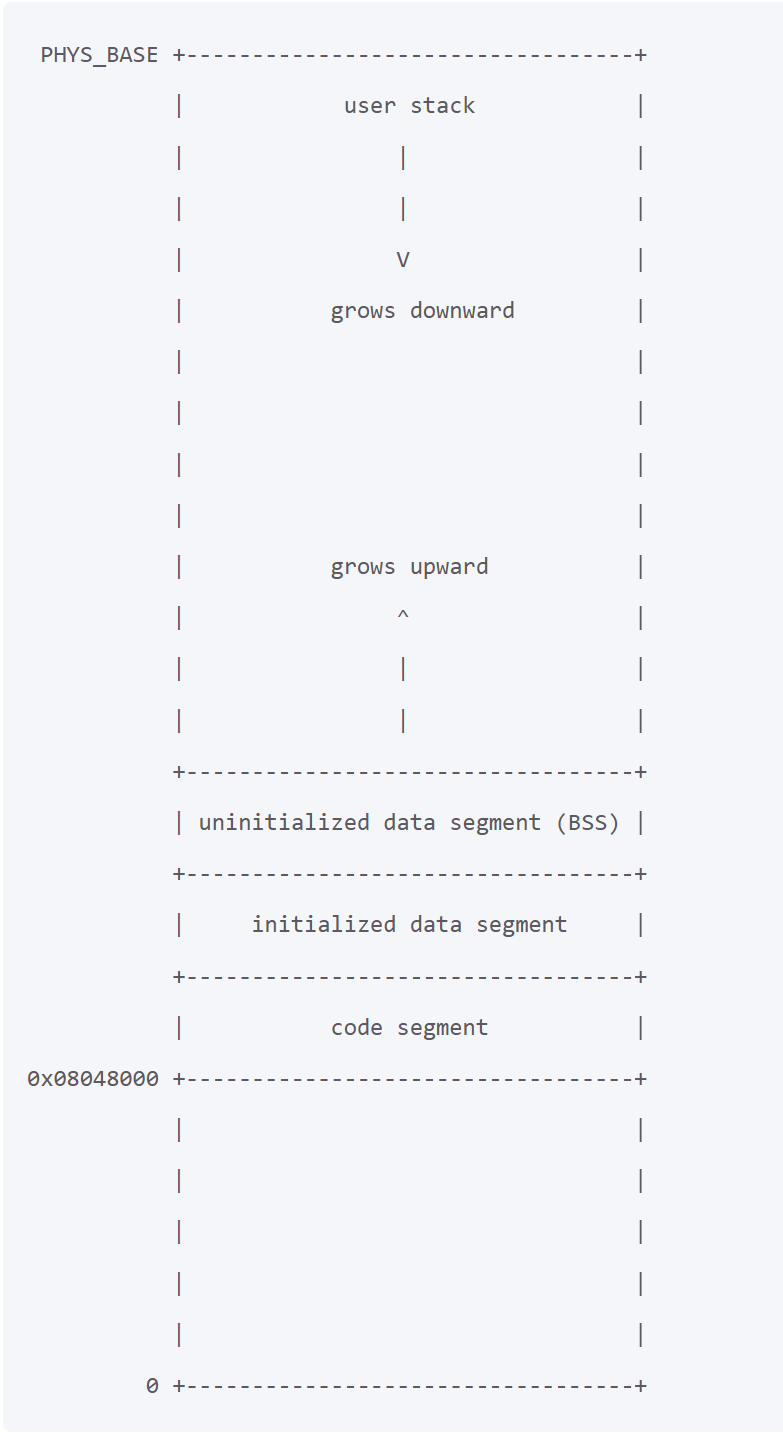


运行结果（除栈区地址，其他地址打印时默认省略了前面的0）：



分析如下：

从操作系统课程中我们学习了Virtual Memory Space的相关知识，因此此处打印的不是真实的物理地址，而是虚拟内存地址，每一个process之间互不干扰。而在这段内存空间中，各部分的位置从低位开始是process的代码段空间，全局变量空间，未初始化变量空间（BSS），堆（向上增长，并留出增长空间），最后是栈（从PHYS\_BASE向下增长）。PHYS\_BASE一般默认是0xC0000000，其上方空间为kernel memory。

这个是示意图：

而运行结果的显示也和我们的分析相吻合（将结果的打印句以从高位到低位的方式重新排列）：

栈区地址: 0x7ffddb4a92e4

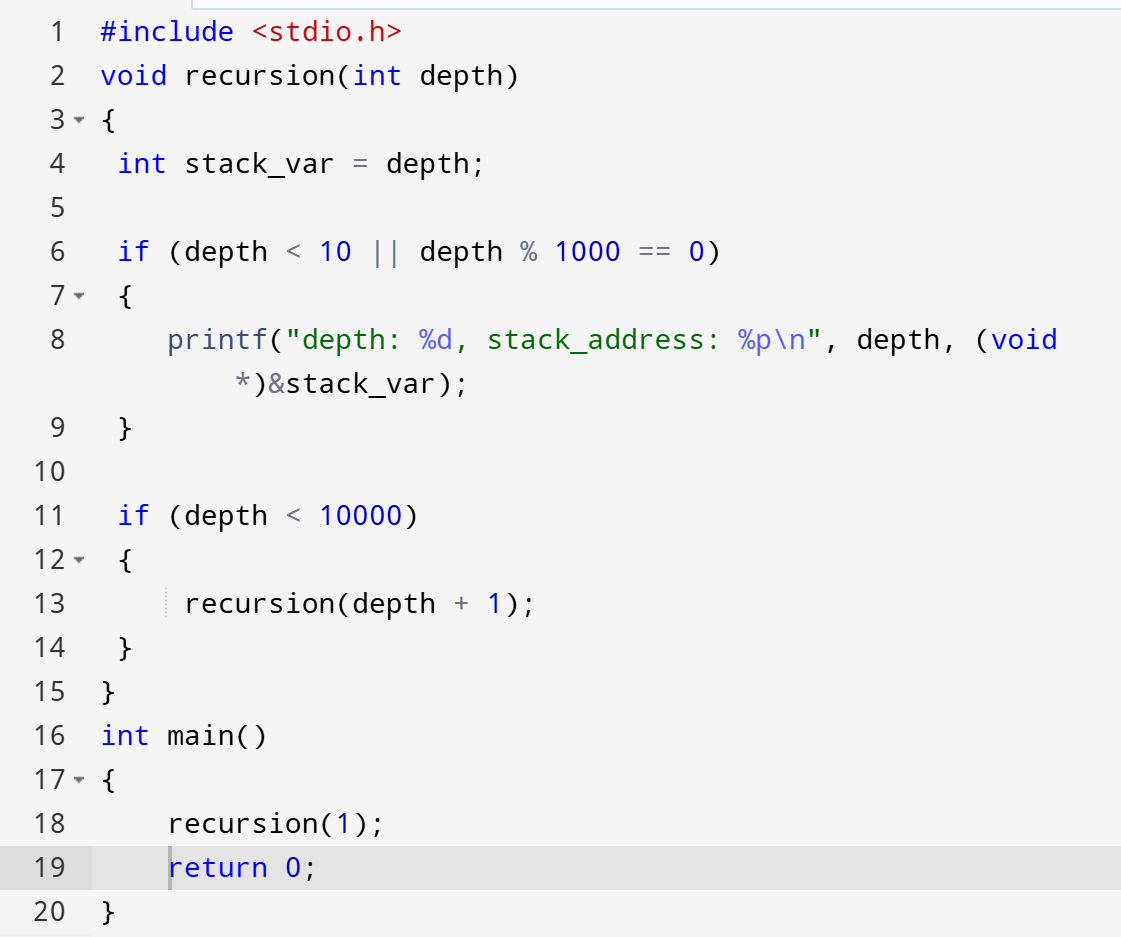
堆区地址: 0x0000020492a0（补足前面5位的0）

未初始化全局变量地址: 0x000000404048（补足前面6位的0）

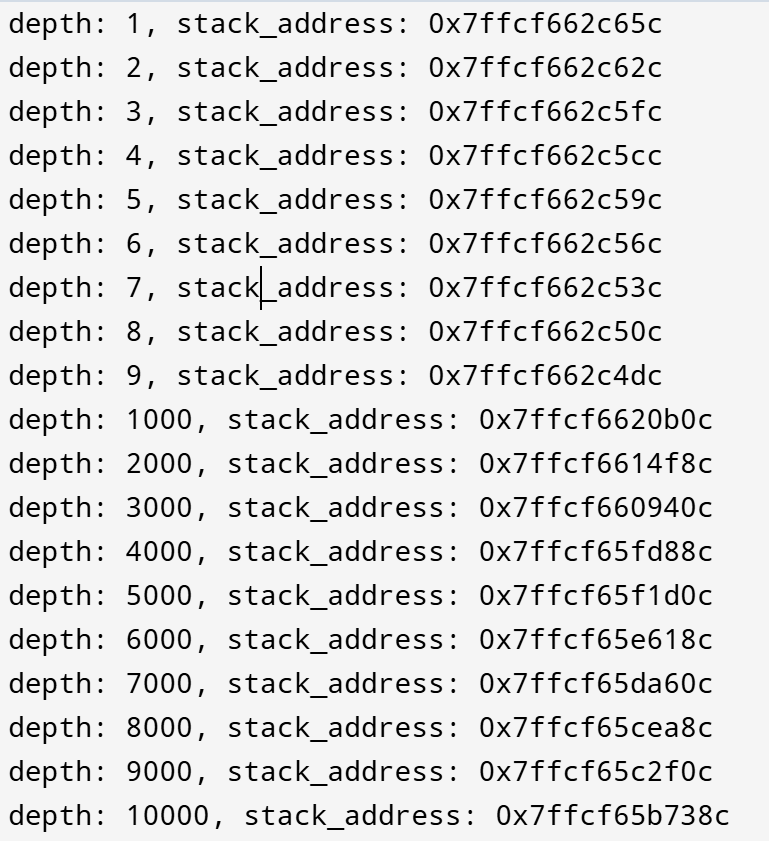
全局变量地址: 0x000000404040（补足前面6位的0）

代码段地址: 0x000000401146（补足前面6位的0）

问题2：系统栈空间默认多大？如何修改？



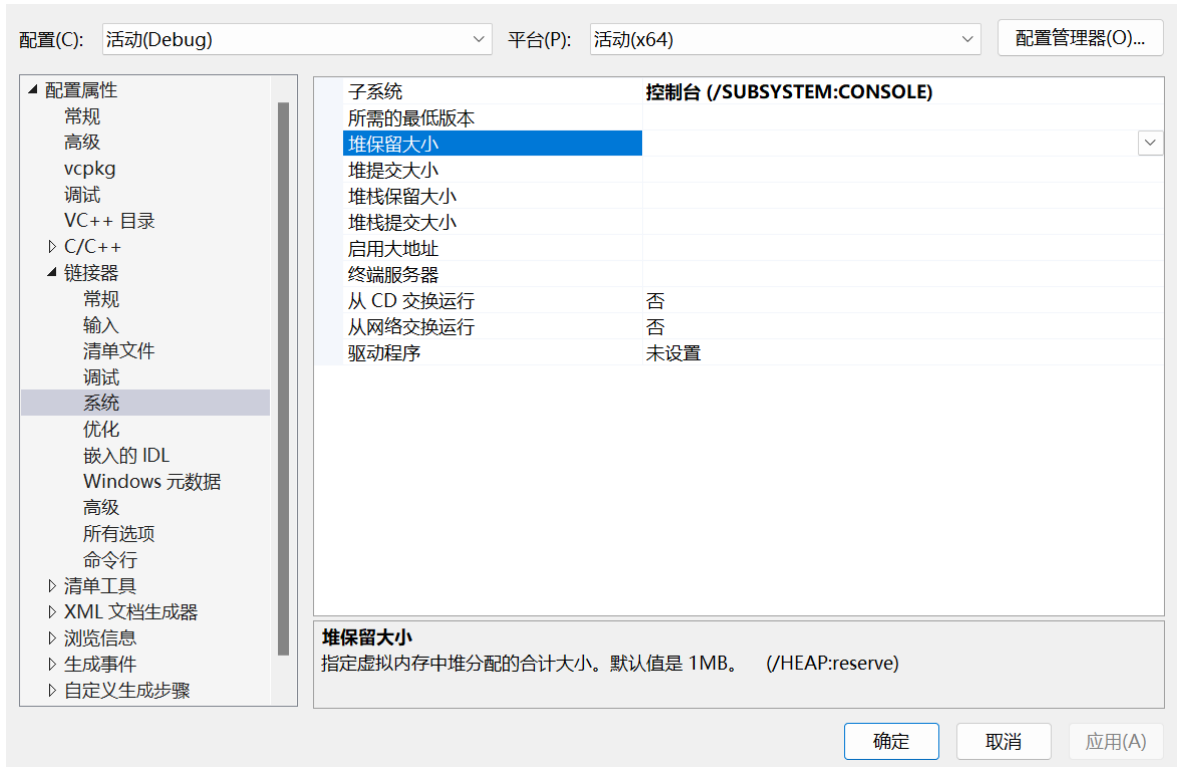
我通过递归函数不断压栈来探索系统给程序默认分配的栈空间大小，并设置了打印条件筛选，每1000深度进行打印，以及最大深度限制：



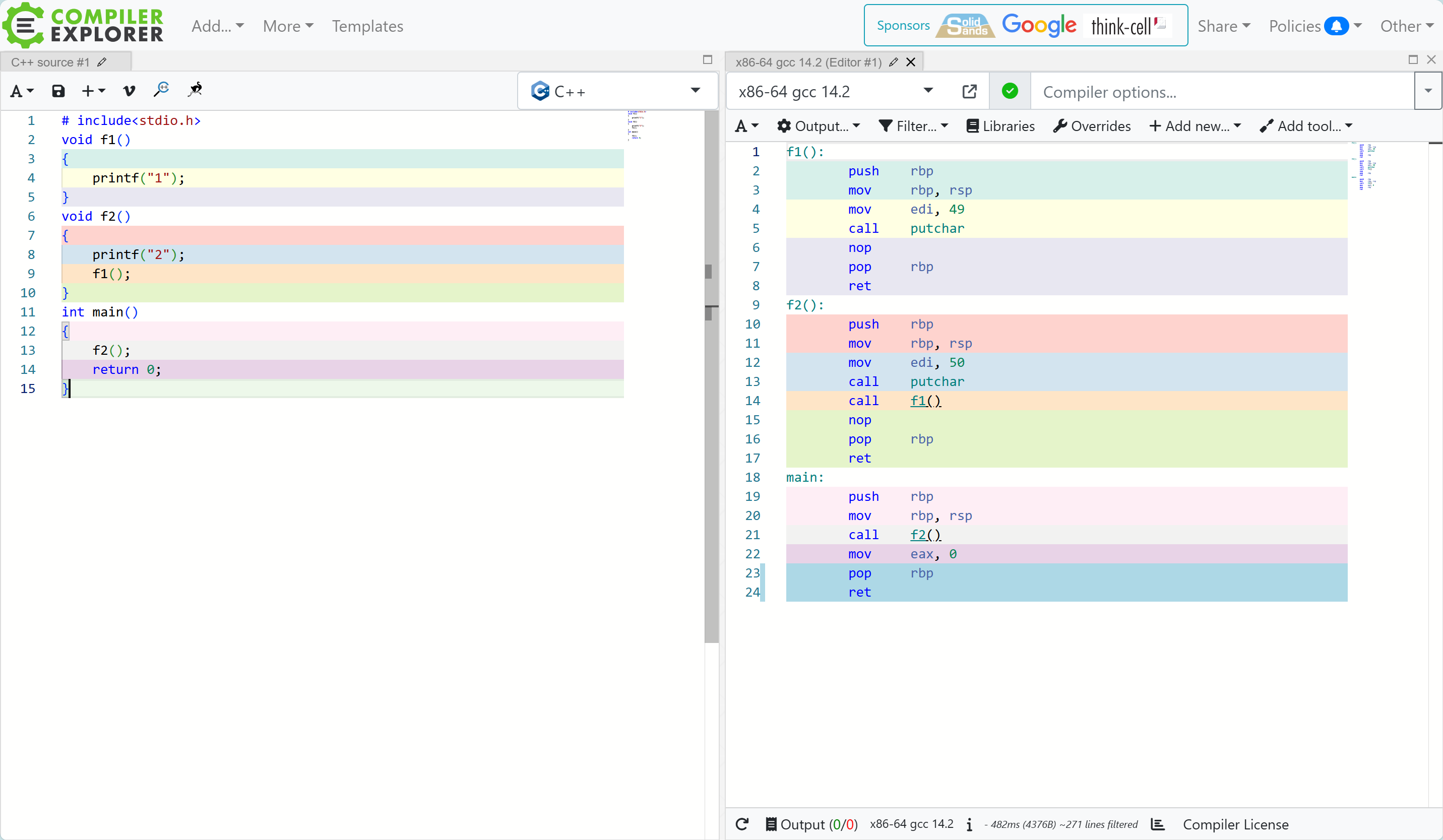
我发现可以达到最大深度而栈没有溢出，因此增加最大深度。最终在打印深度为21000后由于栈溢出报错。

分析：可以看出每次递归所占的栈空间大小为0x65c-0x62c = 0x30 = 48字节。而系统默认的程序栈空间，我们可以通过计算 21000 \* 48 = 1008000 字节，22000\*48 = 1056000字节，而得到其约为1MB（1024\*1024 = 1048576字节）

修改栈空间可以在VS Studio的项目属性中，选择配置属性，再选择链接器的系统，里面有对应的设置选项。



问题3：课堂作业内容。



如图所示，左边为源程序代码，右边为源程序代码的汇编结果。

分析：

1. 函数调用关系：
   1. Main函数调用f1
   2. f1调用f2
2. 栈的使用规则

在汇编结果中，可以看到函数 main、f1 和 f2 都有进入函数和结束函数时的栈操作：

push rbp：将rbp推入栈中，保存调用者的基址寄存器（即其前者的栈帧）。

Mov rbp, rsp: 将rsp（栈指针）赋值给rbp，设置新的栈帧（即自己的栈帧）

pop rbp: 将栈上的第一个元素取出放入rbp，恢复调用者的基址寄存器

ret：返回调用点，退出函数。

实际过程中，还有更详细的栈操作过程（举例假设）：

sub rsp, 16：为栈帧分配空间

lea rsp, ... : 释放栈帧空间

1. 调用过程的栈帧变化：每一个函数都对应一个frame，里面含有该函数的相关信息，如参数，局部变量和返回地址等。而frame会在调用其他函数时保存，返回到原函数后恢复原来的frame。当一个函数返回后（ret），其frame会销毁。
2. 栈的增长方向： 与第一问相同，栈从高位向低位增长。