Week2

a.从C++代码到机器代码，再到可执行的二进制文件，需要哪两个主要操作？
从文本形式的C++代码到可执行的二进制文件，基本上需要两个主要操作：一个是编译（compiling），另一个是链接（linking）。

b.预处理语句将在什么时候被评估？我们需要把它们放在代码的开头吗？
预处理语句会在编译过程的第一阶段——预处理阶段——被评估。实际上，文中的例子展示了`#include`语句可以放在函数体的末尾，而`#if`和`#define`语句也可以用在函数内部。

c.解释cpp文件、翻译单元和obj文件之间的关系。

- 通常情况下，项目中的每一个C++源文件（`.cpp`文件）都被视为一个翻译单元（translation unit）。
- 编译器会将每一个翻译单元编译成一个独立的目标文件（object file,`.obj`文件）。
- 因此，一般一个`.cpp`文件会对应一个翻译单元，并生成一个`.obj`文件。不过，一个翻译单元也可以由多个源文件构成（例如在一个`.cpp`文件中包含另一个`.cpp`文件），这种情况下多个源文件只会生成一个`.obj`文件。

d.在实际编译发生之前，编译器会做什么？
在编译器进行实际的编译工作（如词法分析、语法分析和生成机器码）之前，它首先需要对代码进行预处理（preprocess）。这意味着所有的预处理器语句（preprocessor statements）都会在这个阶段被评估和处理。

e.链接器的主要目的是什么？
链接器（linker）的主要目的是找到每一个符号（symbol）和函数在各个目标文件中的位置，然后将它们链接在一起，形成一个单一的程序。如果一个程序被分割到多个C++文件中，链接器就是将这些文件链接成一个程序的方式。

f.解释`#include`、`#define`、`#if`和`#endif`的用法。
这些都是预处理器语句，其作用如下：

- `#include`：当你指定一个要包含的文件时，预处理器会打开该文件，读取其所有内容，然后将其粘贴到你写`#include`语句的地方。
- `#define`：这个语句会在代码中搜索第一个词，并用其后跟随的内容替换掉它。
- `#if`和`#endif`：`#if`语句可以根据给定的条件来包含或排除一段代码。例如，`#if 1`会包含代码块，而`#if 0`则会将其从编译中排除。`#endif`用来标记这个条件块的结束。

g.在微软Visual Studio中，编译（compile）和生成（build）有什么区别？
在Visual Studio中，编译（compile，快捷键Ctrl+F7）只执行编译阶段，不进行链接。而生成（build）则会先执行编译，然后执行链接，最终生成可执行文件。

h.如果我们声明并调用了一个函数，但没有定义该函数，这会导致编译器错误吗？
不，这不会导致编译器错误。如果一个函数被声明了，编译器在编译当前文件时会相信这个函数的定义存在于别处，因此编译会成功。错误会发生在链接阶段，当链接器找不到该函数的实际定义时，会报告一个链接错误，例如“未解析的外部符”（unresolved external symbol）。

i.在文件作用域中，函数前的关键字static有什么用？
在函数前使用`static`关键字意味着该函数只对当前的翻译单元（即它所在的`.cpp`文件）可见。这使得函数的链接是内部的（internal），其他目标文件将无法看到它。这个特性可以用来解决在头文件中定义函数时可能出现的重复定义链接错误。

j.我们可以有两个同名的变量吗？为什么？
提供的文本指出，拥有相同名称和相同签名的函数或变量会导致“重复的符号”（duplicate symbols）链接错误。这是因为链接器不知道应该链接到哪一个，从而产生了歧义。

k.我们可以有两个函数名、返回类型、参数列表都相同，但函数体不同的函数吗？为什么？
不可以。这种情况会导致“重复的符号”（duplicate symbols）或“多重定义的符号”（multiply defined symbols）的链接错误。原因是当代码调用这个函数时，链接器找到了两个定义，它不知道该链接到哪一个，这造成了歧义。

l.我们可以在一个翻译单元中同时声明和定义一个函数吗？
是的。在提供的文本中，像`multiply`或`main`这样的函数被直接写在了`.cpp`文件（即一个翻译单元）中，这本身就是一个定义。一个函数的定义同时也作为它在该文件后续代码中的声明。

m. inline关键字有什么用？

`inline`关键字的作用是建议编译器拿走函数的实际函数体，并用它来替换掉调用该函数的地方。当一个函数被定义在头文件中时，使用`inline`可以解决因头文件被多处包含而导致的“重复的符号”链接错误。

n.一般来说，什么样的函数可以定义在头文件中？
在头文件中定义普通函数，并且该头文件被多个源文件包含时，会导致“重复的符号”链接错误。为了避免这个问题，通常可以定义在头文件中的函数应该是被标记为`static`或`inline`的函数。