一. 翻译程序第一步

哪怕是**在预处理之前**，我们的编译器也要对文本进行一些翻译处理，这个主要有三步：

1. 把源代码中的字符映射到源字符集。（这个过程主要处理多字节字符和三字节序列，这些字符的产生是由于有些国家不使用西文键盘所导致的，所以说我们**可以忽略**这一点）

2. 定位每个反斜杠后面跟着换行符的实例（所谓的换行符实例，就是指你在文本编辑器里按下回车产生的\n）并删除他们。（将多行语句或指令转成一行）

3. 划分文本成：

①预处理**记号**（记号这个名词很重要，后面会提到）

②空白序列（就是将多个空白符换成一个空格的形式）

③注释（同时用一个空格替换注释）

（**注意由于在预处理之前编译器还要做一点工作，所以在宏定义后面加注释是没有任何问题的**）

二. 一些名词的解释

1. 基本概念

1. 预处理指令：类似#define, #include的指令就叫做预处理指令。

2. 宏（marco）：被替换的缩写就叫做宏（注意：**宏的名称中不能有空格，而且必须遵守C的命名规则**）

3. 替换体：用于替换marco的内容就叫做替换体（**注意在替换体中不能包含新的预处理指令**）。

4. 展开（expansion）：正文中对宏内容的替换的过程就叫做expansion**【可以这么说，preprocesser把宏展开成它们的替换体】**

5. 逻辑行和物理行：由于每条预处理指令都没有分号作为结束的标准，所以一行就代表一条预处理指令，这一行我们把它叫做逻辑行；有时候一个逻辑行太长了，我们有\把它分成多行，这个多行就是物理行。（总而言之，多个物理行组成一个逻辑行）

2. 记号(token)型字符串和字符型字符串

这两个概念的区别主要在于对于替换体中空格的处理，如#define SIX 2 * 3 // 注意中间有两个空格

1. 记号型：用空格分开的词算作不同的记号，在以上的例子中我们的宏SIX共有三个记号。

2. 字符型：用空格分开的词也算作一个相同的字符串，比如以上的例子我们的宏SIX就只有一个字符串。

原书并没有直接指明记号型和字符型的区别，只是模糊地说“在更复杂的情况下，两者的区别才有意义”。一般而言，**编译器都是默认使用记号型字符串。**

3.类对象宏(object-like)和类函数(function-like)宏

这两者的区别我们下面会细讲。

4. 重定义常量

虽然不清楚为什么在书中会单独出现重定义常量的概念。

如果出现了多个#define，那么就有两种情况：

①警告（GCC）②报错；

不管怎么样避免就对了（如何优雅地避免？请看下面的内容）

三. 预处理指令()

1. #include

这个用的也太多了，讲一下“”和<>的区别吧：“”框起来的header表示在当前目录中寻找，找不到再到系统目录下寻找，<>表示直接在系统目录下找。

（注：事实上，你在“”中**可以包含绝对路径**）

2. #define

定义object-like marco和function-like marco

1. object-like marco:这个没什么好说。
2. function-like marco:

①实例：#define SQUARE(X) ((X)*(X))

②作用：类似于函数，在函数体中使用，给定参数，转换为特定的表达式。（个人认为这里不应该使用返回值的概念）

③为什么选择function-like marco？他有三个优点：节省空间，时间（算两点），不用担心类型。　　④为什么要有这么多括号？

我们来考虑几个特殊的情景，就用上面的例子好了

- 每个X都要用括号，考虑y = SQUARE(1+x)，没有括号的话，就是1+x*1+x
- 最后的结果也要用括号，考虑y = 100/ SQUARE(5)，没有括号的话，就是100/5*5

3. #undef

作用：取消相应定义，哪怕对象的定义没有，也是有效的

实例：#undef SIZE

4. #if，#elif，#else和#endif

作用：条件编译，在#if和#elif后面跟上整型常量表达式，下一行跟上相应的预处理指令，#endif就是结束#if的语句块，示例：

```
#if defined(IBMPC)
    #include"ibmpc.h"
#elif defined (VAX)
    #include"VAX.h"
#else
    #include"MAC.h"
#endif
```

(在上面的示例中，defined是关键字，用来判断相应的宏是否已经定义)

5. #ifdef和#ifndef

作用：都是判断宏是否已经存在，#ifdef在存在时返回1，不存在返回0，#ifndef(可以理解为if not defined)恰恰相反

这两个中间，#ifndef的用处最大，因为重定义常量会导致warning和error，所以在写头文件的时候，可以使用#ifndef,见下：

```
#ifndef SIZE
    #define SIZE
    #define LEN
#endif
```

（注意，在很多头文件中，许多宏都是一起定义的，只要有一个定义，其他的就都定义了，本例中就采用了这种思想）

6. 其他的预定义指令

1. #line: 重置行数和文件名，似乎没什么用处。

例: #line 10 "cool.h"　　// 更改行数为10，文件名为cool.h

2. #error:让预处理器发出一条错误消息，用来**中断编译**（stderr用来**中断程序的执行**）

3. #pragma: 将编译器指令放在源代码中（这个也没必要，现在我们有IDE，不用IDE也用命令行）

四. 特别的宏

1. 预定义宏

代表这些宏都是程序内置的。

这个有好多，我就先写自己认为重要的，其他的请在C Primer Plus上P539页查看

1. __DATE__:日期，格式“Mmm dd yyyy”
2. __TIME__:时间， “hh:mm:ss”
3. __FILE__:文件名字符串
4. __LINE__:当前行数
5. __STDC_VERSION__:所支持的C标准，C99——199901L， C11——201112L（我现在用gcc编译器显示的是201710，查了一下，发现是c++17的标准）
6. __func__: 在函数内使用，返回对应函数的名字。（唯一一个小写的）

(注意前后都是两个下划线！！！)

2. 变参宏

1. ...

使用：在宏定义中的宏中使用，用来指代后面数量可变的参数。

2. __VA_ARGS__

使用：在可替换列表中使用，用来指代传入的数量可变的参数。

这两个往往结合使用

如：#define PR(...) printf(__VA_ARGS__)

PR("Howdy");

（注：省略号（其实不是省略号，而是那三个点）只能代表最后的参数）

五. 泛型选择

1. 定义：泛型编程指那些没有特定类型，但是一旦指定一种**类型**，就可以转换成特定类型的代码
2. 怎么使用：使用C11的关键字_Generic(*参数， 类型和返回值的列表*)　　// 用逗号分隔
3. 举例：#define MYTYPE(X) _Generic((X), int:0, float:1, default:3)
4. 说明：这个有点类似于switch，不过switch是对于给定表达式的值，而_Generic是对于**类型，**分清楚传入的是参数，返回值和传入的参数不一样。
5. 技巧：利用“\”将一个逻辑行分成多个物理行

六. 头文件怎么写

之前写过一点头文件，在头文件（即.h为扩展名的文件）中应该包含

1. #include必要的库文件
2. #define对应的**常量**
3. #define对应的**宏函数**
4. struct相应的结构模板
5. typedef相应的类型
6. 声明对应的函数

（由此可见，在头函数中要干的东西真TM多啊！）

七. 其他

1. 双引号中的字符串不做宏替换。
2. **字符串化**(stringizing)

顾名思义，将什么东西（其实是function-like marco中的参数）转化为字符串的形式。

例：#define PR(X) printf(#X)

PR(5)　　// 输出字符串5

3. **预处理器黏合剂**：##

之前有提到过token的概念，在使用记号型字符串的前提下，一个宏可能会有多个tokens

有些时候我们会想将我们的参数融合到前一个token中，但这样写会被当成一个token，于是　　黏合剂就这样诞生了

例：　#define XNAME(n)　X ## n

这样我们就可以用XNAME(1)来代表X1了，如果没有这种粘合剂，我们可能会写成Xn，根据之前所说的，这样做是没用的。