前言：

本周的上机中，似乎很多同学遇到了这样的场景：
眼：我看懂了！
手：不就是敲代码吗！
脑子：我可以！
编译器：你似乎又写了很多errors 

你们可能很想微笑三连 
但其实不需要太激动喔~
恭喜你们，你们已经踏上了bug修复者的修炼之路，你们写的每一个bug，都终将成就你们的编程大业！（前提是要把bug搞懂记住哈~）

当然记不住也是很正常的，我已经无数次的看到很多同学激愤的发朋友圈：如果我再忘记等于是两个等号/不切换英文输入法/失误在大小写/分号/括号上，我就是狗！然后过两天：汪。
甚至每年到了期末，我都还能看到试卷上有几十处这种“低级”错误。
因此十分建议大家在学习编程之初就一定要养成**注意细节**的好习惯。
当然，**及时复习**也是一个好习惯。

我们非常繁杂无趣的第二章终于即将学完了，它是编程学习的基石，所以希望这个文档指引你们完成：
初阶任务：认识每一个字并记住
进阶任务：根据文档内容回忆起各种细节，建构起自己的知识框架
高阶任务：部分内容可以联想到如何在代码中应用
举例：
初阶：八进制的常量以0开头
进阶：整型常量有3种表示方法：十进制、十六进制、八进制，它们分别是……
高阶：所以我不能在代码里写x=09
Get到了吗？
无法顺利进阶到高阶也没关系，试试看倒着来。
回忆下自己写代码时遇到的问题，追溯它们相应的知识点或语法点。
这样再次加深印象也很棒！
总之，一定要学会把知识点映射到代码中，这对后面的学习很重要！

1、程序的一般组成：

程序注释
预编译命令
使用名字空间（常见，但非必须）
函数们（一定**有且只有一个main函数，也叫主函数**）
【关于“主函数是不是有且只有一个”，前几年期末考过一次，还是有同学做错了，可能觉得题目描述的语气过于绝对，就一定有陷阱？还是不确定main函数是不是叫主函数？不管怎么样，请记住，main函数/主函数有且只有一个】

2、注释的两种方式：

第一种（C++风格的注释）：
//每一行都要写前面的两条斜线
//就像这样
第二种（C风格的注释）：
/*可以直接换行
不用重新写前面的符号*/

3、预编译命令：

库包含：把库的接口文件中的代码放进当前的源文件中
#include <iostream> //每次作业都要用的，不然没办法输入输出
#include <cmath> //不太常用，有复杂的数学运算记得写
#include <ctime> //后面会用到（为了根据当前时间生成随机数）

宏定义：执行文本替换
不带参数的宏：定义符号常量，例如
#define PI 3.14159
带参数的宏：一般没什么人用，万一考试考到要能读懂，例如
#define CIRCLE_AREA(x) (PI*(x)*(x))
当程序中出现语句 area = CIRCLE_AREA(4)
就会被替换成 area = (3.14159*(4)*(4))

4、名字空间：

多人合作开发大型程序时，最好每人都写上自己的名字空间，例如：

```
// File: zhang.cpp
// 程序员Zhang写的代码
namespace Zhang
{
   //代码写在这个大括号里
   int num = 26;
}
```

```
// File: wang.cpp
// 程序员Wang写的代码
namespace Wang
{
   //代码写在这个大括号里
   int num = 27;
}
```

连接上面的代码时，编译器会将程序员Zhang定义的变量num识别为Zhang::num，而将程序员Wang定义的变量num识别为Wang::num，避免了同名冲突。

（我们这门课平时的作业都是个人写的小型程序，只要不自己跟自己冲突，就可以不用特意写自己的名字空间，只使用标准库名字空间std即可）

using namespace std; //考试时一定记得写，尤其在看到编程题就大脑一片空白的时候，至少写上这句，别交白卷，还能有渡劫的可能

【为什么加上使用名字空间的语句会使编程更方便？】

5、C++风格的符号常量：

用const也可以定义符号常量，且比用define好【原因是什么？】

【使用符号常量有什么好处？】

6、main函数（也叫主函数）

一个程序可以由多个函数组成，main函数是整个程序的入口。

一个程序必须要有一个main函数，而且只能有一个main函数，不然不能编译。

【main函数的函数头怎么写？】

一个main函数里必须至少有一个return语句，一般是“return 0; ”。

（理论上main函数也可以返回别的值，不过一般用return 0来表示程序正常结束，如果返回别的值，一般表示异常结束）

7、程序的基本组成

程序中通常有变量定义、输入阶段、计算阶段和输出阶段，它们是可以不按顺序交替出现、反复穿插的，应用时要灵活编排。

8、变量与初始化

在一行定义语句中，可以定义同个类型的多个变量。
如 int a, b, c; //不要忘记逗号，以及一定要记得加分号啊
一个变量完成定义后，其类型永不改变。
定义变量的同时可以给它赋初值，也称初始化。
【赋初值和赋值的区别是什么？】

9、变量命名

变量名只能以字母或下划线开头，不能以数字开头。
变量名应“见名知意”。
真的不要用 x1、x2、x3、x4、x5 或 a、b、c、d、e、f、g 系列去命名啊， 去查查英文怎么说，正好多学俩英文单词，难道它不赚嘛~
用了奇怪的命名， debug 会痛苦，改作业会痛苦，改试卷也会痛苦。
请珍爱自己、 助教和老师，谢谢大家。

10、整型

整型数在计算机内用补码表示。
对于有符号整型数，正数的最高位必为0，负数的最高位必为1。
【怎样把一个十进制整型数转换为其补码？】
【十进制、八进制和十六进制的常量分别怎样写？】
【为什么 x=09 会报错？】
【十六进制数中 10 以上的数字怎样表示？】

11、实型（浮点型）

浮点数在计算机内用尾数和指数来表示。【具体是怎么表示的？】
浮点数是不精确的， 在比较**是否相等**和**转换成整数**时， 务必小心，很可能有意想不到的结果。

12、字符型

a 和‘a’毫无关系。
a 是一个变量名，可以是整型、浮点型、字符型等任何类型。
‘a’是一个字符常量，用 cout 输出的话会显示小写字母 a。
【大家写代码一定要注意，想明白自己到底是要用哪个】

字符参与计算时实际上是用其 ASCII 码的数值来计算。
【数字字符和整型数字之间如何转换？】

13、转义字符

【直接 cout<<“ “abc” ”; 为什么无法显示 “abc” ？】
【想输出引号要怎么办？】

14、枚举类型

定义枚举类型的语句： enum 类型名{元素表};
注意该语句定义的是一个类型，不是一个变量。
例如 enum element {Earth, Water, Fire, Air}; 定义了类型element。
被定义好的类型可以用来定义该类型的变量，例如element e;
【怎样给某种枚举类型的变量赋值？】
【元素表中各个元素对应的机内编码是怎样确定的？】
输出枚举类型的变量时，显示的是其内码值，例如：
e = Fire; cout << e; // 显示2，因为Fire对应的机内编码为2

15、sizeof

sizeof()的括号里可以放数据类型、变量或表达式。
【int、 float、 double、 char、 bool 分别各占多少字节？】

16、算术运算

乘法运算符是*，除法运算符是/，取余数运算符是%
只有整型数才能做取余数的运算，浮点数不行。
【整型数、浮点数、字符一起运算时，遵循怎样的类型转换规则？】
C++语言中没有乘方运算符，a的3次方要用a*a*a来表示。
【为什么不能用cmath库的pow函数来计算整型数的n次幂？】

17、强制类型转换

加上目标类型和括号可以进行强制类型转换。
(类型名)(表达式)的意思是对表达式的计算结果进行强制类型转换，而不是先转换表达式中的各个变量后再进行计算。
注意：(int) ‘0’的结果是48而不是0，(char) 0的结果是‘\0’而不是‘0’。

强制类型转换（以及自动类型转换）改变的是寄存器中的临时值，且数值可能会发生相应的变化，但没有改变内存中的变量。
被转换的变量仍保持原来的类型和数值不变。

18、赋值运算

赋值表达式会把赋值运算符右边的量转换为左边的数据类型的量。
赋值运算符左边必须是个变量，不能是常量或某个运算的表达式。
在同一个语句中， 先计算，后赋值。
多重赋值的顺序是从右到左，每次赋值都转换为赋值运算符左边的数据类型。
复合赋值表达式i+=1和自增表达式i++、++i在用作其它表达式的一部分的时候有所不同，i+=1对外而言是个赋值表达式，而i++、++i对外而言是个变量。
例如cout<<++i;或cout<<i++;都会输出变量i的值（尽管是两个不同的值），
但由于cout<<i+=1;等价于cout<<i=i+1;，而cout<<i的结果是cout（<<的优先级高于=），因此该语句是无法编译的（不能用=给cout赋值）。

19、输入输出

在连续输入或输出多个变量（或常量、字符串和表达式的计算结果）时，要用>>或<<来分隔各个变量。
不能用逗号来分隔，如 cin >> a, b; 或 cout<< a, b; 都是**错误**的！
输入多个数据可以用一个 cin 连续输入， 也可以分成多次 cin 输入。
在两个输入语句之间也可以输出。
【cin.get()相比 cin>>有什么优势？】
如果不想在输出信息之后换行，就不要输出 endl。
cout << a << endl; cout << b; 等价于 cout << a << endl << b;（但一般不建议把endl放在一行代码的中间，因为endl表示换行，放在中间有点反直觉。）

20、程序设计的风格

要注意写注释、缩进对齐、变量命名等等。