第二章节c语言的相关概述如下：

对于一个程序细节而言 各个步骤的细节如下操作:

一、#include的指令和头文件

#include<stdio.h>

这是系统的第一行 相当于把 stdio.h的全部内容都输入这个 所在的位置

并且 对于#include而言 起是一条c预处理器指令

输入输出头文件 这个 <stdio.h>这个东西包含了好多所涉及的输入输出函数的一个声明阶段

告诉编译器我接下来需要用到什么样子的函数

对于头文件而言：

1 定义一些常量 告诉函数如何用他们 并且告知后面的编译系统如何去定义这些函数

2 注意 #include<stdio.h>中stdio.h是不需要被特殊记住的

这个stdio.h是众多c语言函数库中的一种类型而已

#include 中的#意味着 我的 c语言独特的系统告诉c语言我的执行代码是在是预处理器在编译器前面完成的工作

介绍工作的先后顺序

1. 对于main（）函数而言

Int main（void）

函数名字是main 识别是个main函数

c程序中一定是从main函数开始执行的（除了特殊情况）

而且在vsstudio2022中只允许出现一个main函数也就是说一个项目只能有一个main函数 其他的函数需要人为进行自定义说明的函数。

请注意 c语言的函数 编译器认可的常用标准表达方式是int main(void)正常情况下是void代表一个函数名称 名词解释

1. 注释

注释只采用两种方式 一种是/\* \*/

另外一种是//其中 //只能被用在单行中 而/\* \*/是可以用在多行当中的注释 注意 /\* \*/ 的运用

1. 花括号函数体和块

花括号可用于把函数中多条语句合并成一个单元和块

1. 声明

Int num;

这个代表完成了两件事情

1 函数中有一个名为num 变量

2 int告诉函数名字是一个整数的形式存在的

并且int 是一个数据类型的形式

Int 是一个关键字 这个关键字意味着在c语言中这个大框架下面无法把int座位函数名和变量的名字来存在。 仅仅是在c语言中 不是说你不能把你身边的宠物或者某个东西叫做int！！

3 所有的变量就跟数学的设 函数x代表谁一样都需要事先声明才可以使用

4 数据类型 暂且不展开说明 等后续咋展开

命名 命名 要用合适 并且有一定意义的英文名字 比如数羊= sheep\_count 而不是 x3如果非要用后面这个 一定记住千万记住得在后面加注释的形式说明这个x3代表啥东西啥行为这是一个编程人员必须做到的事情

5 c语言为何要提前声明变量呢？

方便读者查找变量 就跟做数学题为啥要在开头把所有的需要的 变量先设定好并且表示出他要对应出的意思呢

声明变量有利于咱们做计划 并且就跟解一道数学题类似的东西

尽量在程序头文件运行后在进行变量的声明

6 赋值

Num=1 注意该表达式是把 右侧的值付给左侧的值

并且在之前的int=num中 相当于程序已经提前声明并且定义了变量的数据类型 然后给她预留了一个空间存放自己的值 让后续这个值能把这个变量名字塞满

7 printf（）

Printf(“woshiyige dashabi ”)

这个信息被称为参数 也可叫做实际参数

实际参数 是传递给函数的特定值：是有实际意义的值比如1 2 3 4 566 2 等等 或者一段文字等等

形式参数是函数中用于存储值的变量 ：你不知道这个值x 等于什么他可能等于2 3 4 21 等等但是他现在存在状态是一个不定的值

当程序执行到这一行这时候进行printf（）并且执行他们 执行完printf函数后会返回到他们的主调函数 也就是最开始定义的int main 中去。

Printf中的f题型用户这是一种给格式化打印的函数

八 返回值

Return 0；

Int main中的int表示 main函数最终会返回一个整数值

注意 ：提高程序可读性的技巧（让你在以后工作中避免被老板骂避免被客户喷 人 或者以后尽量减少屎山代码书写的技巧）

技巧一：选择有意义的函数名

技巧二 写注释

技巧三 函数中用空行分割二等概念上的多个部分

技巧四 每条语句各占一行

程序中有如下例子：

Fathom=6\*feet

Printf(“there are %dfeet %dfathome\n”,feet,fathom);

Printf(“aaaaabbbb%d”,6\*feet);

这两个printf分别代表两个打印语句的不同的表达形式 其中第一个语句 中第一个%d代表的是feet这个变量替换了这个%d

第二个这个变量中的%d则是代表着 fathom变量打印的值不一定是变量还可能是个交换式

2.6 多个函数 如何融合进来：

#include<stdio.h>

void butler(void);

int main(void)

{

printf("i will summon the butler function.\n");

butler();

printf("yes.bring me some tea and writeable DVDS.\n");

return 0;

}

void butler(void)

{

printf("You rang sir?\n");

printf("You rang sir?\n");

}

其中官方语言 中 函数 bulter（void）被使用了三次

第一次：官话 函数原型 ：我告诉这个程序 我要用这个函数啦 我要用的这个函数叫bulter

第二次 官话 函数调用的形式出现在main中 ：我之前喊出来的在这用 在这里用 图省事 等同于 那个做数学题 A=blabla 但是正式用的时候 只写一个A

第三次 官话 函数的定义：就是我上面这个函数得说一下他究竟是干嘛的 解释清楚了不然电脑太笨看不到

总结： 先喊一嗓子 我要用这个函数快从库里拿出来

再 直接使用就行不用管那么多

最后你得告诉这个老六 这个b函数是干嘛的不然他傻der认不出来就坏事了

以下是较为严肃的官方语言

函数原型 ：也称作函数声明告诉编译器正在使用某个函数并且在这个其中还指明了函数的属性是啥比如在上述

Void Butler（void）中

第一个void代表的含义是butler 没返回值 （通常被调用的函数会向主调函数返回一个数值但是这个没有）

第二个void的意思是 这个butler不带任何的参数 就是说不带有任何的x y a b之类的blalba参数

注意:void在这里代表的是空的 不是无效的意思

这时候编译器在运行到这里时候会检查 butler （）是否使用的到位

调用过程中： 先一行一行执行到 butler（）后当执行完butler后再继续butler 的后面的内容执行

记住：这个执行buthler（）函数取决于他在main的位置 及时我把void buther的定义扔到main前面 也是先执行main而不是先执行buther

因为所有的c语言语句都是从main函数开始的

关于如何纠正bug的问题

不管人会不会出现语法错误 编译器中也会出现语法错误 比如 一 某处隐藏的语法错误会导致编译器的误判

二 报错位置比真正的位置往后面错开一行

语义错误

要把自己想象成计算机 一步一步的执行代码

跟着他的逻辑走一遍即可

程序状态：

程序状态是指在程序执行的过程中某给顶点上所有变量的集合 的一张快照

就是某个点的位置上突然拍一张照片 此时此刻收集所有变量的值的集合作为程序的一张快照此时 就是程序的状态

跟踪程序状态的方法：

1 自己模拟计算机逐步执行

2 定义语义错误的另一种方法是在程序的某个地方插入额外的printf语句以监视这个前面的变量是否按照你自己想的运行下去

3 调试器 进行断点调试步骤：

具体步骤详情请见如何驯服vsstudio篇

2.8 关键字和保留标志符

总是有那么一些关键字 他不能用作变量的描述

课后复习题的坑：

\n是表示 换行符但是/n就是个符号没任何意义

Fuction是函数的意思 代表的是函数的意思