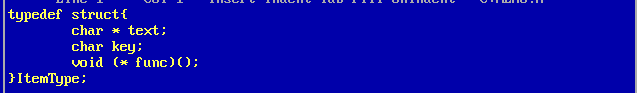
综合研究13

1. 学习过程

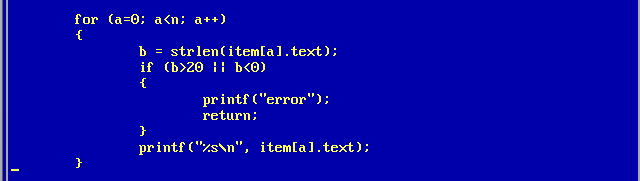
⑴ 首先对menu.h程序进行分析，它的功能分为两个，一是显示三种选项，二是根据用户输入的不同选择执行相应的操作。首先定义了一个名为ItemType的结构体，其中有三个数据项：text（每个选项中显示的字符串）、key（选项中供用户选择的编号‘1’、‘2’、‘3’）、func（选择选项后对应要执行的函数）。



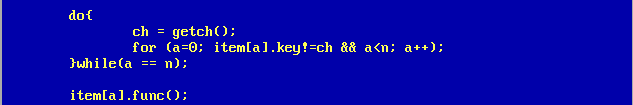
在定义了结构体后，编写menu（）函数，首先定义了int型变量a（作为结构体数组的下标实现for循环中将每个选项输出在屏幕上）、b（获取每个选项的长度，判断长度是否符合要求）以及char型变量ch（用来获取用户按键，并判断该按键是否正确）。



在定义所需的变量后，通过for循环来实现将菜单项显示在屏幕上：item[0]表示一个结构体数组的第一个元素，数组中元素为结构体类型数据，一个元素中有三个对应的数据项，通过item[0]来表示第一个结构体数据，item[0].text可以表示第一个结构体数据的text数据项，在本次循环中表示选项中的第一个选项，将下标使用变量a来表示，a的初值为0，通过a=0，打印出第一个选项，但是我们本程序中存储一个选项的长度为20B，因此需要通过变量b获取到输入字符串的字节数，保证其长度是有效地，有效时将字符串输出，显示在屏幕上，否则就会提示出错信息并结束menu（）函数。通过循环，打印出所有的选项信息，执行到for循环结束，我们就可以看到基本的选项信息。



在实现菜单显示后，需要接收用户的输入并执行所对应的函数，通过do…while语句实现，首先要接收到用户的键盘输入，接收到的数据保存在变量ch中，之后为变量a赋值0，从item[0]开始判断其key数据项是否与用户输入的数据相同，相同的话就执行相应的函数，不相同a就进行加一操作，与item[1]的key数据项进行对比，执行上述判断，直到找到与之相同的key值，如果用户输入的值不正确，就不会执行任何语句并等待用户正确输入。假设与item[1]的key值一致，do…while就会结束循环，结束循环后，通过a可以表示调用的函数是哪一个结构体中的，通过该结构体对应的func数据项调用要执行的函数。



⑵ 对use.c函数进行分析，use.c程序调用emnu.h程序显示出选项信息，在接受了用户的键盘输入后在屏幕中显示不同的字符串，第一个选项是（1） hello，用户选择该选项会在屏幕中显示出彩色的字符串“hello world”，第二个选项是（2） atr，用户选择该选项后会在屏幕中显示出彩色的字符串“the art of c”，第三个选项是（3） welcome，用户选择该选项会在屏幕中显示出彩色的字符串“welcome to c”。

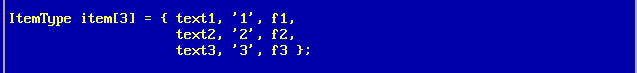
首先添加了menu.h头文件，因为在use.c程序中要调用emnu.h中的内容



定义三个名为text1、text2、text3的字符串，就是菜单中要显示的三个选项，用户通过选择这三个选项中的其中一个，实现上面所说的不同字符串的输出。



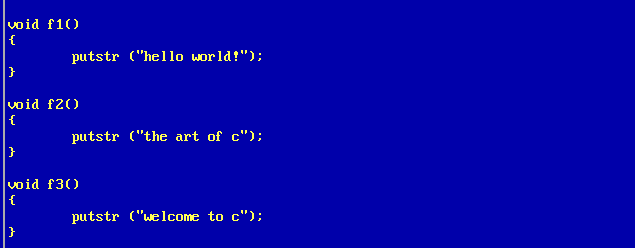
之后，定义了结构体类型的数组item，共有三个元素，第一个元素的数据为text1、‘1’、f1对应于提到的text、key、func数据项。text1对应上述定义的字符串数据，‘1’表示每个选项对应的编号，f1即函数名，表示对应要执行的函数的地址。定义的后两个元素的数据与此相同。



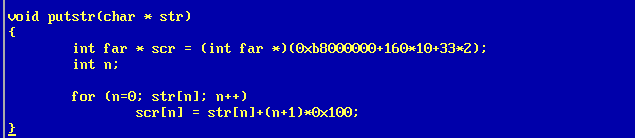
main函数中只有一条调用menu（）函数语句：调用该函数，就可以实现显示选项信息以及接收用户输入执行相应操作。



以下三个程序是三个不同选项分别要执行的函数f1、f2、f3，每个函数中也仅有一条调用putstr函数的语句，通过该函数可以将输入的字符串参数进行输出。



f1、f2、f3函数中都调用了putstr函数，我们具体来看看该函数的内部实现：



该函数的参数为char型的指针变量，即字符串。定义int型指针变量scr，表示要显示的字符串的位置，通过下标n来进行偏移，n为0时，对应屏幕第10行第33列的位置，将字符串的第一个字符显示在该位置。

⑶ 我们在分析完程序后，运行程序，观察其执行结果：在显示出选项后，输入‘1’。



⑷ 现在我们要基于程序menu.h与use.c中的数据组织方式，编写程序，实现加减乘除运算的功能的程序，我们也通过两个程序来实现，13menu0.h程序中显示加减乘除四种运算对应的四个选项，并可以接收用户的输入，来进行相应的运算操作。首先我们编写13menu0.h程序：定义结构体变量，有三个数据项：text（选项中显示的字符串）、key（选项中供用户选择的编号‘1’、‘2’、‘3’、‘4’），func（选项中不同运算执行的函数）。选项对应的要执行的是返回值为int，参数为int、int型的函数，因此此程序中定义的变量func与menu.h中的有一些不同，menu.h中不同选项要执行的函数是无返回值、无参的。

**typedef struct{**

**char \* text;**

**char key;**

**int (\* func)(int, int);**

**}ItemType;**

在menu（）函数中，定义了变量a、b、ch，这三个变量与menu.h中的menu（）函数中的三个变量是相同的，变量a1[20]、b1[20]用来存放进行运算的两个操作数，code表示程序支持的所有运算对应的运算符号。

**void menu(ItemType \* item, int n)**

**{**

**int a, b;**

**char ch;**

**char a1[20];**

**char b1[20];**

**char \* code = "+-\*/";**

接下来依旧使用for循环来显示所有的选项信息在屏幕上，与menu.h中实现的完全相同，也还是通过do…while函数来调用不同选项的函数，首先要接收到用户的键盘输入，获取到要执行哪个选项的对应的函数。结果依旧存储在a中。

**for (a=0; a<n; a++)**

**{**

**b = strlen(item[a].text);**

**if (b>10 || b<0)**

**{**

**printf("error");**

**return;**

**}**

**printf("%s\n", item[a].text);**

**}**

**do{**

**ch = getch();**

**for (a=0; item[a].key!=ch && a<n; a++);**

**}while(a == n);**

menu.h中要调用的函数是无参数的，因此可以直接调用，但我们要调用的函数有两个参数，所以我们首先要获取到这两个参数，依旧通过项目12中的方式通过gets（）函数来获取到两个操作数。与项目12中不同的是，如果我们已经确定了要进行的运算，就不在需要用户输入该运算符号，我们可以通过code[a]直接将选择的运算符号进行输出，相对于项目12中此程序中的实现方式是简便的，获取到参数后，就可以直接调用函数来进行运算。至此menu（）函数的功能（一是显示支持的运算选项，二是接收用户的选择进行相应的运算）都已实现。

**gets(a1);**

**printf("%c\n", code[a]);**

**gets(b1);**

**printf("---------------\n");**

**printf("%d", item[a].func(atoi(a1), atoi(b1)));**

⑸ 我们已经编写好我们需要的13menu0.h文件，现在我们编写13.c程序，在这个程序中调用13menu0.h中的内容显示出支持的运算选项：第一个选项是：⑴ add（加法），第二个选项是：⑵ sub（减法），第三个选项是：⑶ mul（乘法），第四个选项是：⑷ div（除法），用户选择1就会等待用户输入进行加法运算的第一个操作数，加号‘+’会自动显示出来，用户接着输入第二个操作数，进行回车就可以看到进行加法后的结果。

定义四个名为text1、text2、text3、text4的字符串，就是要显示的四个选项，用户通过选择这四个选项中的其中一个，实现上面所说的不同运算。

**# include "13menu0.h"**

**char text1[10] = "(1) add";**

**char text2[10] = "(2) sub";**

**char text3[10] = "(3) mul";**

**char text4[10] = "(4) div";**

之后，定义了结构体类型的数组item，共有四个元素，第一个元素的数据为text1、‘1’、add对应于提到的text、key、func数据项。text1对应上述定义的字符串数据，‘1’表示每个选项对应的编号，add即函数名，表示对应要执行的函数的地址。定义的后三个元素的数据与此相同。

**ItemType item[4] = { text1, '1', add,**

**text2, '2', sub,**

**text3, '3', mul,**

**text4, '4', div };**

在main函数中同样只有一条调用menu（）函数的指令，add、sub、mul、div为加减乘除运算对应的四个函数。

**main()**

**{**

**menu(item, 4);**

**}**

**int add(int a, int b)**

**{**

**return a+b;**

**}**

**int sub(int a, int b)**

**{**

**return a-b;**

**}**

**int mul(int a, int b)**

**{**

**return a\*b;**

**}**

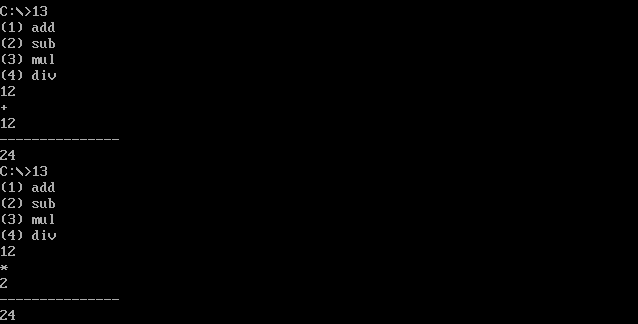
**int div(int a, int b)**

**{**

**return a/b;**

**}**

至此，程序编写完成，我们通过执行程序验证程序的正确性：



经过试验，程序可以实现与项目12中的a.c相同的功能。区别是有一个选项提示用户可以进行的运算操作，用户选择后输入参数可以进行相应的运算。

⑹ 在编写完程序后，重新思考了综合研究13想让我们通过研究获得到的知识，应该不仅仅是要做一个类似于menu.h和use.c相类似的程序，重新分析该研究的目的，发现自己理解错误该研究本来要达到的研究目的。应该不是要将a.c程序做成选项信息的形式，供用户选择实现不同的运算，而是单纯的理解这两个程序（一个头文件，一个主程序）中分别编写实现的是什么内容，进而理解头文件存在的意义以及头文件中所包含的内容是什么。本着这样的出发点，重新编写程序，实现与项目12中a.c功能相同的程序。本次编写的程序将全局变量的定义，接收用户输入以及调用相应功能的函数指令写入.h头文件，将具体实现加减运算的函数add、sub写入.c源文件中。

实现a.c功能程序的头文件13a.h：

**char \* code;**

**int (\* func[2]) (int, int);**

**void ys()**

**{**

**char a[20];**

**char b[20];**

**char ch;**

**int n;**

**gets(a);**

**printf("%c\n", ch = getch());**

**gets(b);**

**for (n=0; code[n] && code[n]!=ch; n++);**

**if (!code[n])**

**{**

**printf("error!");**

**return;**

**}**

**printf("---------------\n");**

**printf("%d", func[n](atoi(a), atoi(b)));**

**}**

实现a.c功能程序的主程序13a0.c：

**# include "13a.h"**

**int add (int, int);**

**int sub (int, int);**

**char \* code = "+-";**

**int (\* func[2])(int, int) = {add, sub};**

**main()**

**{**

**ys();**

**}**

**int add (int a, int b)**

**{**

**return a+b;**

**}**

**int sub (int a, int b)**

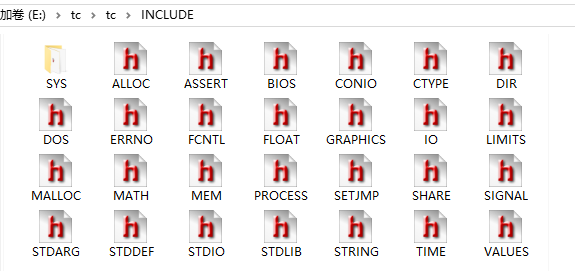
**{**

**return a-b;**

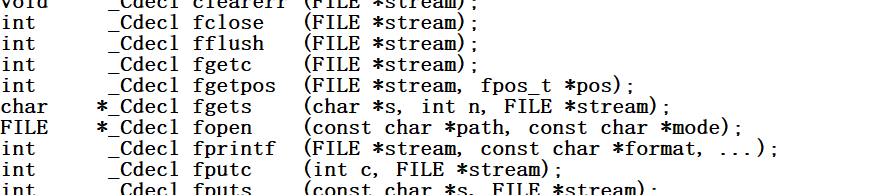
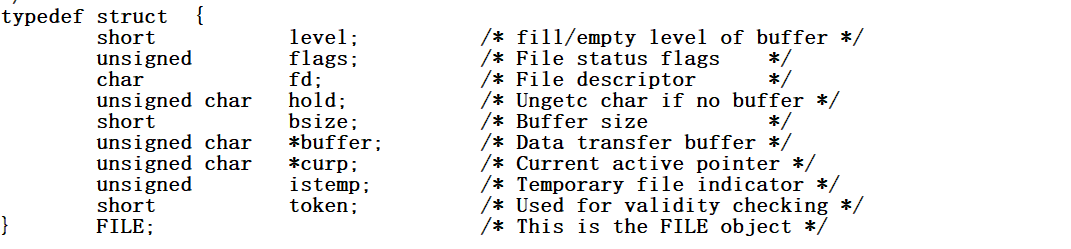
**}**

程序中的代码都是之前分析过的，在此处不再做过多的说明，通过这种方式编写自己的头文件，我们可以理解系统直接给我们提供的头文件的作用，头文件其本身是不需要包含具体函数的实现，只是起描述性的作用，我们头文件中可以定义结构体数据类型、定义常量、定义函数、函数声明等之类的内容以及调用要使用的函数的语句，但是变量的赋值、函数实现的内容都是存放在主程序中的。

总结：.h文件是头文件，内含函数声明、宏定义、结构体定义等内容，.c文件是程序文件，内含函数实现，变量定义等内容。在tc2.0目录中有include目录与lib目录，观察include目录中存放的文件：



其中有很多的头文件，打开其中一个文件stdio.h文件查看其中的内容：



其中有定义的结构体数据类型以及很多函数的原型声明，从中我们能了解到fprintf函数是在stdio.h头文件中的，但是其中并不知道该函数的内容，所以头文件主要就是定义、声明的作用，因为系统提供给我们的函数我们可以使用，其内部具体实现我们又是不清楚的，但是我们既然可以使用，我们实际上就有函数的真正的实现代码，经过之前的研究，我们可以知道具体的实现代码可以在cs.lib中找到，里面是包含fprintf函数的主要实现代码的，只不过该文件是经过编译生成的目标文件，可以在连接时保证我们的程序可以运行。



通过对综合研究13内容的研究，我觉得是让我们通过研究获得头文件以及源文件的关系，这样我们也可以更好的理解include目录与lib目录中的内容，更深入的了解tc2.0目录中各个目录、文件的作用。了解头文件与源文件中分别要编写的内容，就可以更好的理解我们在编写C语言程序中开头使用系统提供的头文件的作用。

总结：基于程序menu.h与use.c中的数据组织方式，编写出的与项目12中的a.c功能相同的程序，同样有两个文件，一个头文件13a.h，一个实现具体功能的文件13a0.c。观察两个文件中所写的内容，13a.h中所编写的程序内容正是项目12中理解的程序的共性实现。13a0.c中所编写的程序内容是项目12中理解的程序的个性实现，将一个a.c通过两个文件来实现，头文件中编写程序的共性，实现功能的文件里编写程序的个性。这样的数据组织方式将一个程序中的共性与个性分开在两个文件中实现，一个程序的结构是更加清晰地。

1. 思考解决的问题

⑴　之前调用库文件时通常使用＜＞，本程序中使用的“”，两者有什么区别？

分析：＜＞是从系统环境变量指定的目录中去搜索头文件，主要是系统提供的头文件，而我们使用的是自己编写的头文件，通过“”可以从当前目录中搜索到自己编写的头文件，当找不到的时候才会去系统环境变量指定的目录中搜索。

1. 思考未解决的问题
2. 研究体会

通过对综合研究13的研究，对经常使用的头文件有了一定的了解，我们可以看出Ｃ语言的设计中，函数只提供给我们头文件，而没有具体实现的源代码，这样作为用户有了接口，可以使用函数，作为提供方，不想公开的源代码也起到了保护的作用，这样做既防止了函数被修改，又保证了公司的一些源代码不泄露。而我们自己编写的头文件的主要作用就是我们直接查看头文件就可以知道程序实现所定义的一些函数，以及变量，以及定义的数据类型都可以获取到，不必在源程序中一个个查找，对于复杂的项目，通过分开编写源文件和头文件的方法可以优化代码，也方便后期的维护，这样的数据组织方式有助于构造良好的程序设计风格。