**c语言上机学习报告**

姓名：吾拉孜别克·塔斯肯

学号： 20040100010

通过一个多月的学习，我在4次上机学到了很多，这些是我不上机的话是不太可能知道自己还没学会怎么写代码，我知道了学c语言的同时而且自己也要会敲代码，我的收获满满，在4次上机学习中学到了很多。

第一次上机学习时，第一次去机房敲代码，第一次体会到自己以为会的代码到了机房就写不出来了，不知道写什么，通过请教学长，学习同学终于敲出了我的第一个代码，但也有很多错误，通过观察我发现我没有在每个结束语句后面打' ;' 琢磨很长时间改正了这个错误，然后发现有出错了，然后我旁边的同学跟我讲了我的printf('' %d'', a );写错了，应该是printf里面的输出运算符%d应该写成%f才行，我才发现我把双精度数的输出运算符写错了，通过这个错误我知道了这个，就这样我把我的第一个代码写出来了，然后突然有了敲别的代码的热情，我连续敲了好几个，这些提中我没有在犯前面的错，写的都挺好的，但做到后后几道题发现没那么简单，然后通过在慢慢琢磨也做出了这些题。我在这个星期上机课学到了很多如：

printf() --将变量的内容输出到显示器上

四中用法

1.printf(''字符串'');

2.printf(''输出运算符'',输出参数);

3. 1.printf(''输出运算符1 输出运算符2 .....'', 输出参数1, 输出参数2, ……);

printf(''输出运算符 非输出运算符'', 输出参数);

输出控制符

|  |  |
| --- | --- |
| %d — | int |
| %ld — | long int |
| %c — | char |
| %f — | float |
| %lf — | double |
| %x（或者%X后者舗X）— | int 或 long int 或 short int |
| %o — | 同上 |
| %S — | 字符串 |

scanf()[通过键盘将数据输入到变量中]

两种用法

用法一:

scanf(''输入控制符''，输入参数)；

功能:将从键盘输入的字符转化 为输入控制符所规定格式的数据，然后存入 参数的值为地址的变量中

用法二:

scanf(''非输入控制符 输入控制符''，输入参数)；

功能:将从键盘输入的字符转化为输入控制符所规定格式的数据，然后存入 参数的值为地址的变量中

非输入控制符必须原样输入

用scanf编写是的注意事项

1.为了让用户方便，使用scanf之前最好先使用printf提示用户以什么样的方式输入

2.scanf中尽量不要使用非输入控制符，尤其是\n。

第二次上机学习我也学到了许多，能把班里老师讲的内容用到题目当中，比第一次敲代码时犯的错误减少了许多，通过选择和循环语句来做练习题，做完这些题我在里面体会到的自己犯的错误，及老师讲过的一些内容都用到了这些题目当中，通过第二次上机我能很好的用循环语句和选择语句，从中总结出了一下内容:

if的最简单用法

格式：

if(表达式)

语句；

功能:

如果表达式为真, 执行语句

如果表达式为假，语句不执行

if的范围问题

1. if（表达式）

语句A;

语句B;

解释:if默认只能控制语句A的执行或不执行if无法控制语句B的执行或不执行，或者讲：语句B一定执行

2. if（表达式）

{

语句A;

语句B;

}

此时if可以控制语句A和语句B

3.if.. else... 的用法

4.if.. else if.. else.. 的用法

格式：

if (1)

A;

else if (2)

B;

else if（3）

C；

else

D；

if的常见问题解析

1>.空语句的问题

if （3 > 2）;

等价于

if（3 > 2）

；〃这是一个空语句

2>. if （表达式1）

A；

else

B；

是正确的 if （表达式1）;

A;

else

B；

是错误的

3>.

if （表达式1）

A；

else if （表达式2）

B；

else if （表达式3）

C；

else

D；

即便表达式1和2都成立，也只会执行A语句

4>.

if （表达式1）

A;

else if （表达式2）

B；

else if （表达式3）

C；

这样写语法不会出错，但逻辑上有漏洞

5>.

if （表达式D

A;

else if （表达式2）

B；

else if （表达式3）

C；

else （表达式4） 〃7行

D；

这样写是不对的，正确的写法是： 要么去掉7行的（表达式4） 要么在7行的else后面加if 6>.

if （表达式1）

A;

else if （表达式2）

B；

else if （表达式1）

C；

else （表达式4）;

D；

这样写语法不会出错，但逻辑上是错误的 else (表达式4);

D；

等价于

else

(表达式4);

D；

for

1. 格式：

for (1; 2; 3) 语句A;

1. 执行的流程【重点】

单个for循环的使用 多个for循环的嵌套使用

1.

for (1; 2; 3) //I

for (4; 5; 6) //2

A; //3

B; 〃4

整体是两个语句，12 3是第一个语句

4是第二个语句

2.

for (1; 2; 3) for (4; 5; 6) (

A；

B；

}

整体是一个语句

3.

for(7; 8; 9)

for (1; 2; 3)

A；

B；

for (4; 5; 6) C；

}

整体是一个语句

通过第三次上机我加强了对对继续分支与循环语句的使用，尝试了比第二次上机难的题，也通过上机知道了自己对循环语句的了解还不够深，也了解到流程控制对学c语言的重要性，在加固基础的同时也学到了很多内容，这些是我学到的内容：

while

1. 执行顺序

格式:

while (表达式)

语句；

1. 与for的相互比较

for和while可以相互转换

for (1; 2; 3)

A;

等价于

1；

while (2)

(

A;

3；

}

while和for可以相互转化

但for的逻辑性更强，更不容易出错，推荐多使用for

1. 举例

从键盘输入一个数字，如果该数字是回文数， 则返回yes,否则返回no

回文数：正着写和倒着写都一样

比如：121 12321都是回文数

1. 什么时候使用while,什么时候使用for

没法说，用多了自然而然就知道了

do... while

格式

do

} while (表达式);

do... while.并不等价于for,当然也不等价于while

主要用于人机交互 一元二次方程

break 和 cont inue

break

break如果用于循环是用来终止循环

break如果用于switch,则是用于终止switch

break不能直接用于if,除非if属于循环内部的一个子句 例子：

for (i=0; i<3; ++i )

(

if (3 > 2)

break; //breakM然是if内部的语句，

//{fibre ak终止的确是外部的for循环  
printfC嘿嘿!\n〃)； 〃永远不会输出

}

在多层循环中，break只能终止最里面包裹它的那个循环

例子：

for (i=0; i<3; ++i)

(

for (j=l; j<4; ++j)

break; //break只能终止距离它最近的循环 printf 同志们好! \n”);

)

在多层switch嵌套中，break只能终止距离它最近的switch

例子：

int x=l, y=0, a=0, b=0;

switch(x) // 第一个switch

{

case 1:

switch(y) // 第二个switch

(

case 0:

a++;

break; //终止的是第二个switch

case 1:

b++；

break;

}

b = 100;

break; 〃终止的是第一个switch

case 2:

a++；

b++；

break;

)

printf ("％d %d\n”, a, b): 〃26行

最终输出结果是：1 100

continue

用于跳过本次循环余下的语句，

转去判断是否需要执行下次循环

例子：

1.

for (1; 2; 3)

(

A;

B；

continue; 〃如果执行该语句，则执行完该语句后，会执行语句3, C和D都会被跳过去,C 和D不会被执行

C；

D；

}

2.

while (表达式)

{

A;

B；

continue; 〃如果执行该语句，则执行完该语句后，

〃会执行表达式，C和D都会被跳过去,C和D不会被执行

C；

D；

}

第四次上机继续加固了输入语句，循环，数组的简单使用，通过上机学习知道了自己对数组的了解还不够，所以得继续努力，继续加固对前面内容和对最新内容的了解，这样才能学好c语言，以下是我对数组的理解，跟我总结的内容：

为什么需要数组

为了解决大量同类型数据的存储和使用问题

为了模拟现实世界

数组的分类

一维数组

怎样定义一维数组

为n个变量连续分配存储空间

所有的变量数据类型必须相同

所有变量所占的字节大小必须相等

例子：

int a [5];

一维数组名不代表数组中所有的元素，

一維数组名代表数组第一个元素的地址

有关一维数组的操作

初始化

完全初始化

int a[5J = (1,2,3,4,5);

不完全初始化，未被初始化的元素自动为零

int a [5] = {1,2,3};

不初始化，所有元素是垃圾值

int a [5];

清零

int a [5] = {0};

错误写法：

int a [5];

a[5] = {1,2, 3, 4, 5}; 〃错误

只有在定义数组的同时才可以整体赋值， 其他情况下整体赋值都是错误的

int a [5] = (1,2, 3,4, 5}:

a[5] = 100; //error因为没有a[5]这个元素，最大只有a[4] int a [5] = (1,2, 3,4,5);

int b [5];

如果要把a数组中的值全部复制给b数组 错误的写法：

b = a;// error

正确的写法

for (i=0; i<5; ++i)

b[i] = a[i];

二维数组

int a[3][4];

总共是12个元素，可以当做3行4列看待，这12个元素的名字依次是a[0][0] a[0][l] a[0] [2] a[0]⑶

a[l] [0] a[l][2] a[l] [3]

a⑵[0] a[2][l] M2] [2] a[2]⑶

表示第i+1行第j+1列的元素

int a[m] [n];该二维数组右下角位置的元素只能是a[mT] [n-1] 初始化

int a[3] [4] = {1,2,3,4,5,6,7,8,9,10,11,12};

int a [3] [4] = {

{1, 2, 3, 4},

{5, 6, 7, 8}

};

操作

输出二维数组内容：

int a [3] [4] = (

(1, 2, 3, 4},

{5, 6, 7, 8), {9, 10,11,12}

}；

int i, j;

〃输出数组内容

for (i=0; i<3; ++i)

(

for (j=0; j<4; ++j) printfC^d ”, a[i] [j]);

printf ("\n");

}

对二维数组排序

求每一行的最大值

判断矩阵是否对称 矩阵的相乘

多维数组

是否存在多维数组

不存在

因为内存是线性一维的 n维数组可以当做每个元素是比1维数组的一维数组 比如：

int a[3][4];

该数组是含有3个元素的一维数组 只不过每个元素都可以再分成4个小元素 int a [3] [4] [5];

该数组是含有3个元素的一维数组 只不过每个元素都是4行5列的二维数组