实验环境

C语言实验环境：Code::Blocks 13.12、GNU GCC Compiler

Java语言实验环境：

java version "1.8.0\_60"

Java(TM) SE Runtime Environment (build 1.8.0\_60-b27)

Java HotSpot(TM) 64-Bit Server VM (build 25.60-b23, mixed mode)

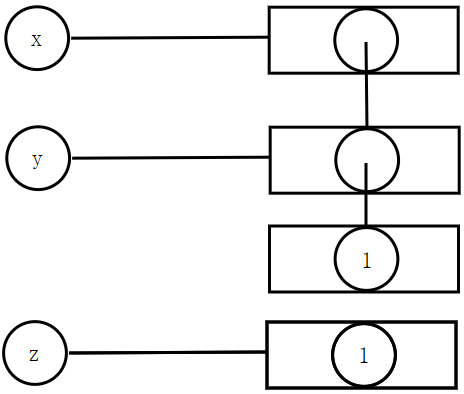
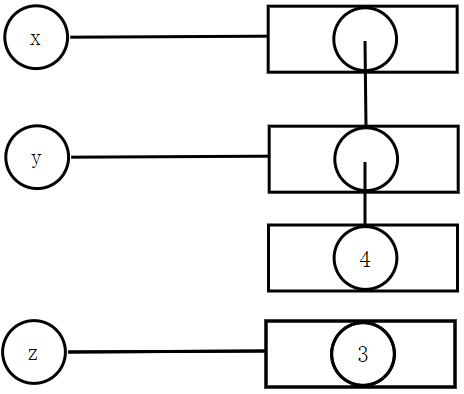
文本语言编写：Notepad++

5.26 给定下面的 C 语言程序，画出每一次对 `\*\*x` 赋值（第 11 行和第 15 行）后的变量的 box-and-circle 图。哪些变量互为别名？程序打印出的值是什么？



解答：

前部分变量box-and-circle图与后半部分图：

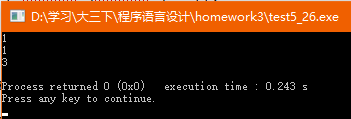
 

变量别名：

\*x 与 y；

\*\*x 与 \*y ；

运行截图：



程序打印：

1

1

3

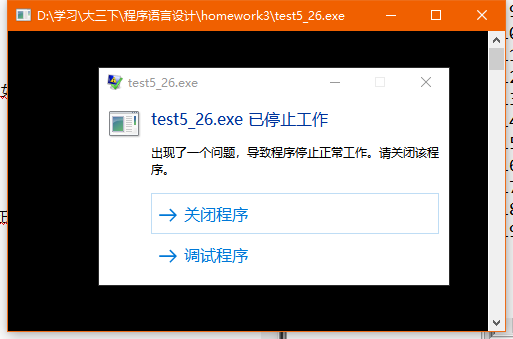
5.27 解释在上面的习题例子中，两次调用 `malloc`（第 6 行和第 7 行）的原因。如果删除第 6 行，会有什么事情发生？删除第 7 行呢？

解答：

两次调用malloc是为了，在使用变量 x和 y 之前，给他们申请空间并赋值。

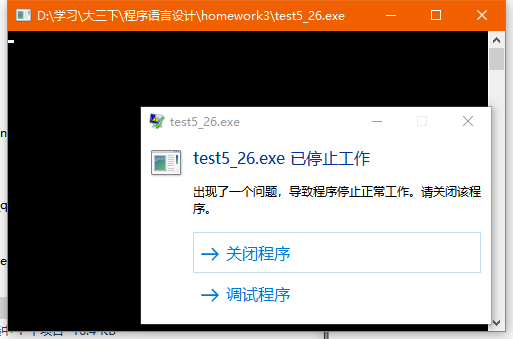
若删掉之后，会导致使用\*运算符对无效地址解引用，程序发生错误：

注释第6行，能够编译通过，但程序运行时发生崩溃，如下图：



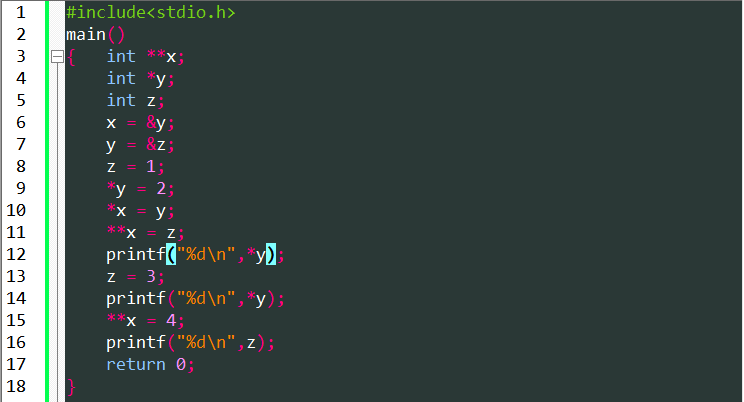
程序第10行 \*x = y 发生失败。

注释第7行，能够编译通过，但程序运行时发生崩溃，如下图：



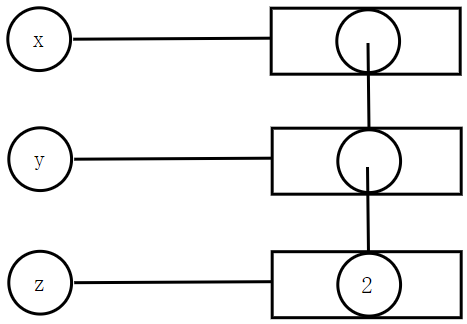
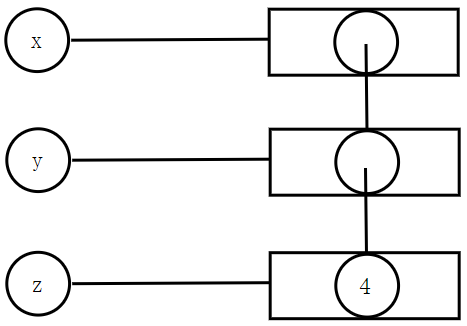
程序第10行 \*y = 2 发生失败。

5.28 针对下面的代码，重新按照习题 5.26 的要求给出解释：



解答：

前部分变量box-and-circle图与后半部分图：

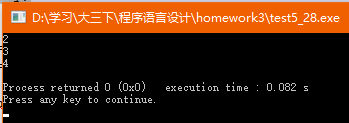
 

变量别名：

\*x 与 y；

\*\*x 与 \*y 与 z；

运行截图：



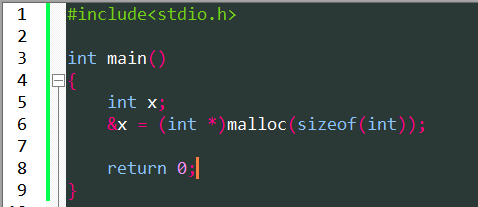
程序打印：

2

3

4

5.31 为什么下面的 C 语言代码是非法的：



解答：

程序编译运行不通过，第六行出现错误：

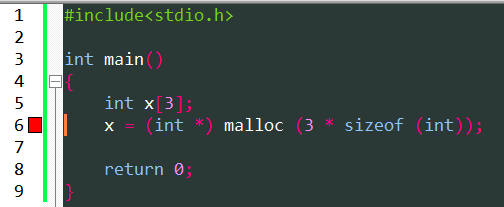


**D:\学习\大三下\程序语言设计\homework3\test5\_31.c|6|error: lvalue required as left operand of assignment**

**结论：**

&x 是右值表达式，不能被赋值。

5.32 为什么下面的 C 语言代码是非法的：



解答：

程序编译运行不通过，第六行出现错误：

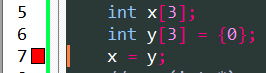


**D:\学习\大三下\程序语言设计\homework3\test5\_32.c|6|error: incompatible types when assigning to type 'int[3]' from type 'int \*'**

**结论：**

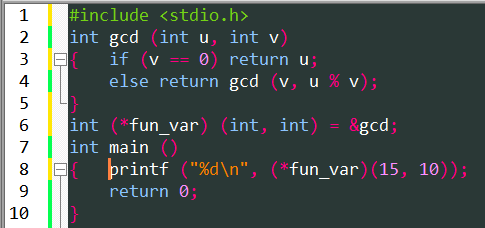
x 类型是 int [3]，不能被赋值成 int \*指针。

另外，若修改成：



仍然编译不通过，报错如上面一样不能完成类型转换。

5.33 下面是一个带有函数变量的合法 C 语言程序：



将代码中的第 6 行与第 8 行和下面的代码相应语句（第 1 行与第 3 行）做比较。为什么这种版本的代码是允许的？为什么它不是必需的？

图5.29中代码：

(1) int (\*fun\_var) (int, int) = gcd;

(2) int main ()

(3) { printf ("%d\n", fun\_var (15, 10));

(4) return 0;

(5) }

解答：

本题中代码运行结果：



图5.29中代码运行结果：



**结论：**

两种版本代码运行结果相同

int (\*fun\_var) (int, int) 定义变量 fun\_var 为函数指针类型

对于一个函数名，取地址、解引用得到的值不变，即gcd，&gcd，\*gcd，\*(&gcd)，&(\*gcd) 均相同

对于一个函数指针，解引用、不解引用都可以调用。