练习1.1.1

编译器相对于解释器的优点是什么？解释器相对于编译其的优点是什么?

- 编译器的优点：

    1. 编译器能够将源代码全部转化为机器代码，因此其运行时效率更高。
    2. 编译器编译时能够进行全面的错误检查，因此能够在运行前发现大部分错误。
    3. 经由编译器编译生成的代码难以进行反编译，在一定程度上能够保护源代码的安全。

- 解释器的优点：

    1. 解释器逐行执行代码，其适合交互式编程和开发。
    2. 解释器在调试时更容易找到错误位置，其错误诊断效果会更好。
    3. 解释器不需要特定的操作系统和硬件，拥有良好的平台兼容性。

练习1.1.2

在一个语言处理系统中，编译器产生汇编语言而不是机器语言的好处是什么?

优点：

1. 汇编语言相比于机器语言可读性更好，便于程序员更容易理解和修改生成的代码，从而进行调试和优化。
2. 编译生成的中间表示可以轻松的翻译成目标机器上的语言，因此能够更好地实现跨平台。

练习1.1.3

对下图中的块结构的C代码，指出赋给w、x、y和z的值

(1)

```
{
    int w,x,y,z;
    int i = 21;int j = 8;
    {
        int j = 4;
        i = 7;
        w = i * j ;
    }
    x = i + j;
    {
        int i = j;
        y = i + j;
    }
```

```
    z = i - j;
}
```

赋值给w、x、y和z的值分别为28、15、16和-1。

(2)

```
{
    int w,x,y,z;
    int i = 9;int j = 14;
    {
        int i = 5;
        w = i + j ;
    }
    x = j - i;
    {
        int j = 3;
        i = 2;
        y = i + j;
    }
    z = i + j;
}
```

赋值给w、x、y和z的值分别为19、5、5和16。

练习1.1.4

下面的C代码的打印结果是什么?

```
{
    #include <stdio.h>
    #define a x
    int x = 12;
    void b(){x = a * 2;printf("%d\n",x);}
    void c(){int a = x + 3;b();printf("%d\n",a + 1);}
    int main(){b();c();}
}
```

- LINUX环境下运行结果为：

```
solomon@DESKTOP-23PER74:~/CODES/C/CP$ ./homework
24
48
21932
solomon@DESKTOP-23PER74:~/CODES/C/CP$ ./homework
24
48
21893
```

- WINDOWS环境下的运行结果为：

原因是C函数中声明了一个与x同名的局部变量，在编译后，该变量会覆盖全局变量的值。由于该变量未初始化，在linux系统下其初始值是随机值，而在windows下其初始值为0。

练习1.1.5

有人把程序设计语言分为编译型和解释型两类，例如C是编译型，Python是解释型。这个分类是否合理？能否构建C语言的解释器，或者Python的静态编译器？谈谈你的看法。

这个分类结果在一定程度上是不合理的，编译和解释的本质区别在于目标计算机是否以编译语言直接翻译程序原因，而从这一点上来说，大多语言都可以被编译或者解释，也就不能单独归为编译型或者解释型语言。例如说，CINT和Ch就是C语言中的解释器，而对于python，PyPy和Cython也可以将python进行静态编译。不过这种违反程序设计语言初衷的翻译方式并不一定能够提高程序执行的效率，所以将程序设计语言分为编译型和解释型两类，有一定道理，但两类语言之间执行效率的差异正在逐步减小。