云 南 大 学 信 息 学 院

面向对象技术导论

实验指导书

王津 编写

2024 年 3 月

实验要求

一、可读性

一个好的程序要具备可读性，可方便自己也可方便别人。所以，要培养一个良好的编程习惯，可注意以下几方面：

1. 代码的缩进；
2. 有效使用空格；
3. 简明的注释；
4. 意义明确的命名；
5. 着重表示的常量。

二、亲手编写源程序

在编写Java 程序过程中，还可以利用一些可视化的开发工具，它们可以综合使用Java 的编译器和调试器等，例如Intellij IDEA及Eclipse等，使用编程开发工具可以加快编程的速度。但在初始学习时还是最好亲手编写源程序，以便理解类和编程思想。

三、立即运行程序

编写的源程序要立即上机编译运行来检验程序中存在的问题。通过运行的结果验证程序的功能是否实现。有一些系统类的方法、变量也需要上机实验去了解它们的含义。

Java 还有很多相关的技术，JavaScript 是具有Java 子集的脚本语言，可用于编写ASP 文件；JSP是比ASP 更优越的Web 动态开发技术，以Java 作为脚本语言，通过JDBC 能够利用SQL 与数据库进行通信，使Java 得到更大范围的应用，如企业信息系统开发，电子商务网站开发。Java 在2D 图形、3D动画、移动电话方面也有很好的应用，学习Java 才是刚刚开始。

四、遇到不懂的问题时，怎么办？

首先，任何人编程时都会遇到各种各样的问题，遇到问题时，要自己先主动想办法解决，然后再寻求外援，例如：

1、 课本及工具书上查找；

2、 搜索引擎Google，Baidu上查找；

3、 自己试着找到问题根源所在；

4、 向他人求教。

实验一 Java 基本语法练习

【开发语言及实现平台或实验环境】

Windows10 或 11，JDK 21 与 Intellij IDEA

【实验目的】

1. 了解 Java 的数据类型
2. 掌握各种变量的声明方式
3. 理解运算符的优先级
4. 掌握 java 基本数据类型。运算符与表达式、数组的使用方法
5. 理解 Java 程序语法结构，掌握顺序结构、选择结构和循环结构语法的程序设计方法

【实验要求】

1. 编写一个声明 java 不同数据类型变量的程序
2. 编写一个使用运算符、表达式、变量的程序
3. 编写一个使用 java 数据的程序
4. 编写表达式语句、复合语句的程序
5. 编写使用不同选择结构的程序
6. 编写使用不同循环结构的程序

【实验内容】

一、 编写程序，打印下列九九乘法表。

```
1X1=1
1X2=2  2X2=4
1X3=3  2X3=6  3X3=9
1X4=4  2X4=8  3X4=12  4X4=16
1X5=5  2X5=10  3X5=15  4X5=20  5X5=25
1X6=6  2X6=12  3X6=18  4X6=24  5X6=30  6X6=36
1X7=7  2X7=14  3X7=21  4X7=28  5X7=35  6X7=42  7X7=49
1X8=8  2X8=16  3X8=24  4X8=32  5X8=40  6X8=48  7X8=56  8X8=64
1X9=9  2X9=18  3X9=27  4X9=36  5X9=45  6X9=54  7X9=63  8X9=72  9X9=81
```

二、 编程输出以下图案:

要求：*行数 n 小于 40，在程序开头直接指定，不必从键盘输入。

三、 **给定某人这一月的工资收入为 salary，该变量在程序开始处赋值，不需要从键盘输入，计算他这一月应该缴纳的个人所得税。按 2011 年 9 月执行的新标准计算，免征额为 3500 元。**

全月应纳税所得额	税率	速算扣除数(元)
全月应纳税额不超过 1500 元	税率为 3%	0
全月应纳税额超过 1500 元至 4500 元	税率为 10%	105
全月应纳税额超过 4500 元至 9000 元	税率为 20%	555
全月应纳税额超过 9000 元至 35000 元	税率为 25%	1005
全月应纳税额超过 35000 元至 55000 元	税率为 30%	2755
全月应纳税额超过 55000 元至 80000 元	税率为 35%	5505
全月应纳税额超过 80000 元	税率为 45%	13505

例如：**salary=2000，　个税=0;**

salary=3500，　个税=0;

salary=5000，　个税=(5000-3500)*0.03-0=45;

salary=8000，　个税=(8000-3500)*0.1-105=345;

salary=20000，　个税=(20000-3500)*0.25-1005=3120。

四、 **编写程序，将 2~10000 之间的素数保存到数组，输出这些素数的平均值，输出小于 8000 的最大素数、输出大于 1000 的最小素数。**

五、 **有一个 5 行 6 列的二维数组，元素值 array[i][j]=i^2-$(j-i)^2$+10，其中**

$0 \le i \le 4, 0 \le j \le 5$,输出该数组，并求出每一行的最大值和每一列的最小值。

六、 编写求阶乘 n! 的函数。求从 P 个不同的数中选出 Q 个的组合数（P>=Q）是 $\frac{P!}{Q!(P-Q)!}$,输出从 30 个不同的数中选出 7 个的组合数。（注意：求阶乘 n！的函数需要在返回类型 double 之前加关键字 static 才能调用）

七、 圆周率

数学中重要的数字 π（3.1415926…）可以使用很多种办法计算得出，最简单的公式之一就是莱布尼茨公式。它假定以下无穷技术收敛于 $\frac{\pi}{4}$，其中 $n \ge 0$。

$$\frac{\pi}{4} = \sum_{n=0} \frac{(-1)^n}{2n+1} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \cdots$$

那么，

$$\pi = \sum_{n=0} \frac{4 \cdot (-1)^n}{2n+1}$$

试计算，当 *n*=100 时，圆周率的结果是多少？当 *n*=10000 时，圆周率的结果又是多少？

八、微分计算

微分是非常重要的运算，目前人工智能中神经网络的优化方法都是建立在微分求解的基础上的。深度学习的神经网络本质上是一个最优化问题，深度学习训练的核心就是在于通过优化算法不断地调整模型参数。当然，这个参数量可能很大，自然语言处理最近流行的大模型动不动就是百亿、千亿参数的大模型。但不管怎么样，本质上还是调整模型参数。而模型参数的调整其实是通过梯度累加来实现，对于给定的参数 w，需要求解损失函数对于参数 *w* 的偏导数，然后将计算的梯度累加到 *w* 上。

计算机中求解 *f*(*x*)函数在某点的微分，有很多种办法，包括数值微分、符号微分和自动微分等等。对于一些比较基础的函数，可以直接指导 *f*'(*x*)，例如 *sin*(*x*)

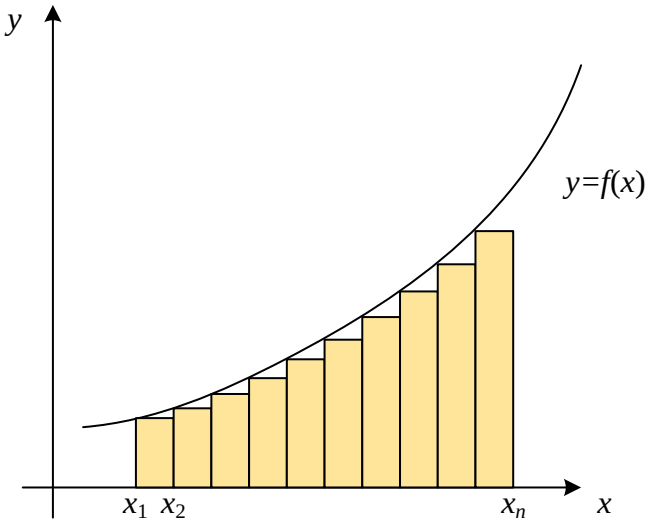
的微分是 ***cos(x)***。那么只需要计算 **cos(x)**即可。但不是所有的函数都可以轻易地指导 **f'(x)**。这个时候就可以使用数值微分，也就是通过微分的定义进行计算:

$$f'(x)=\lim_{\triangle x\to 0}\frac{f(x+\triangle x)-f(x-\triangle x)}{2\triangle x}$$

这里，如果给定 $f(x)=\dfrac{1}{1+e^{-0.085x}}$，试求解 ***x*=1** 位置的微分值。

九、定积分计算

计算机中计算积分的本质就是将积分区域划分为多个区间段，求解每一个区间段的矩形区域面积进行累加，如下图所示。



如果给定下列函数，试计算其定积分:

(1) $\int_0^1 4/(1+x^2)dx$

(2) $\int_0^1(\sin x-\cos x)dx$

(3) $\int_{-1}^1\int_{-1}^1(x^2+y^2)dxdy$

十、简单的压缩算法

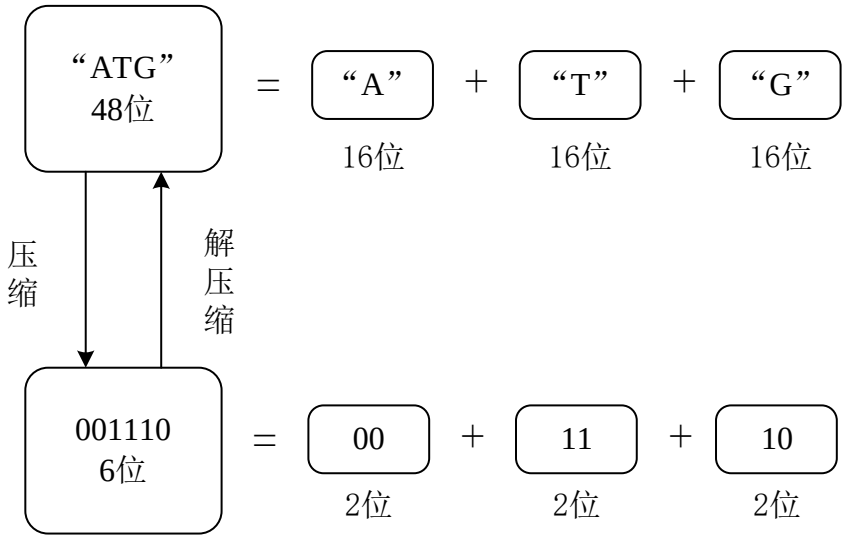
无论是在虚拟世界还是现实环境，节省空间十分重要。空间占用越少，利用率就越高，也会更节省成本。如果所租的公寓大小超过了家中人和物品所需要的空间，就可以“缩”到更便宜的小公寓去。如果数据按字节付费的方式存储在服务器上，那么压缩一下数据就可以降低存储成本。压缩就是读取数据并对其进行编码（修改格式）操作，以便减少数据所占用的空间。解压缩则是反

向的过程，即将数据恢复到原始格式。

既然压缩数据的存储效率更高，那么为什么不把所有数据全都压缩了呢？这是因为需要对实践和空间进行权衡。压缩一段数据并将其解压回原始格式需要耗费一定的时间。因此，只有在对数据大小的要求优先于数据传输速度的情况下，数据压缩才有意义。想想正在通过互联网传输的大文件。压缩它们是有道理的，因为传输文件所需的时间比收到后解压缩文本所需的时间更长。此外，压缩文件以将其存储在原始服务器上所花费的时间只需计算一次。

如果能够意识到数据类型占用的二进制位数要比其内容实际需要的多，就可以想到一个最简单的数据压缩方案。例如，从底层考虑，如果一个永远不会超过 32767 的有符号整数在内存中存储为 64 位的 long 类型，其存储效率就很低。所以他可以存储为 16 位的 short 类型。将 64 位转换为 16 位就可以让该整数实际占用的空间减少 75%。如果数以百万计的此类数字被低效存储，则可能会浪费多达数兆字节的空间。

在 Java 编程中，有时为了简单起见，开发人员无法换位思考。绝大多数 Java 代码都使用 32 位 int 类型来存储整数。对于绝大多数应用程序来说，这确定没有错。但是，如果要存储数百万个整数，或者需要特定精度的证书，那么可能需要考虑适合它们的类型是什么。



如果某个类型可以表示的数字数量少于用来存储它的位数可以表示的值的数量，或许存储效率就能得以提高。可以试想一个 DNA 中组成基因的核苷酸。每个核苷酸只能是以下 4 个值之一：A、C、G 或 T。如果将基因存储为 String 类型，每个核苷酸由一个字符表示，在 Java 中就需要 16 位存储空间。二进制

则只需要 2 位来存储这种具有 4 个值的类型，例如 00、01、10、11。如果考虑用 00 来表示 A，01 表示 C，10 表示 G，11 表示 T，那么一个核苷酸字符串所需要的存储空间可以减少 87.5%（每个核苷酸从 16 位减少到 2 位）。

如果给定了核苷酸序列 TAGGGATTAACCGTTATATATATATAGCCATGGATGGATCGATTATATAACCGTTATATATATATAGCCATGGATCGATTATA。试着按上述方法，编写 compress(String s) 函数进行压缩，以及 decompress(String s)函数进行解压缩，并试计算压缩前后的压缩比是多少？(注意：在位操作时，有两种办法，譬如给定了 ACGT，第一种是使用移位操作，缺点时移位操作只能应用于 int 数据类型)

```
int a = 0;
int c[] = {3, 2, 1, 0}
for(int i; i < a.length; i++){
    a += c[i];
    a = a << 2;
}
System.out.println(a);
```

(另一种是使用乘以 4 操作)

```
int a = 0;
int c[] = {3, 2, 1, 0}
for(int i; i < a.length; i++){
    a += c[i];
    a *= 4;
}
System.out.println(a);
```