复旦大学计算机科学技术学院

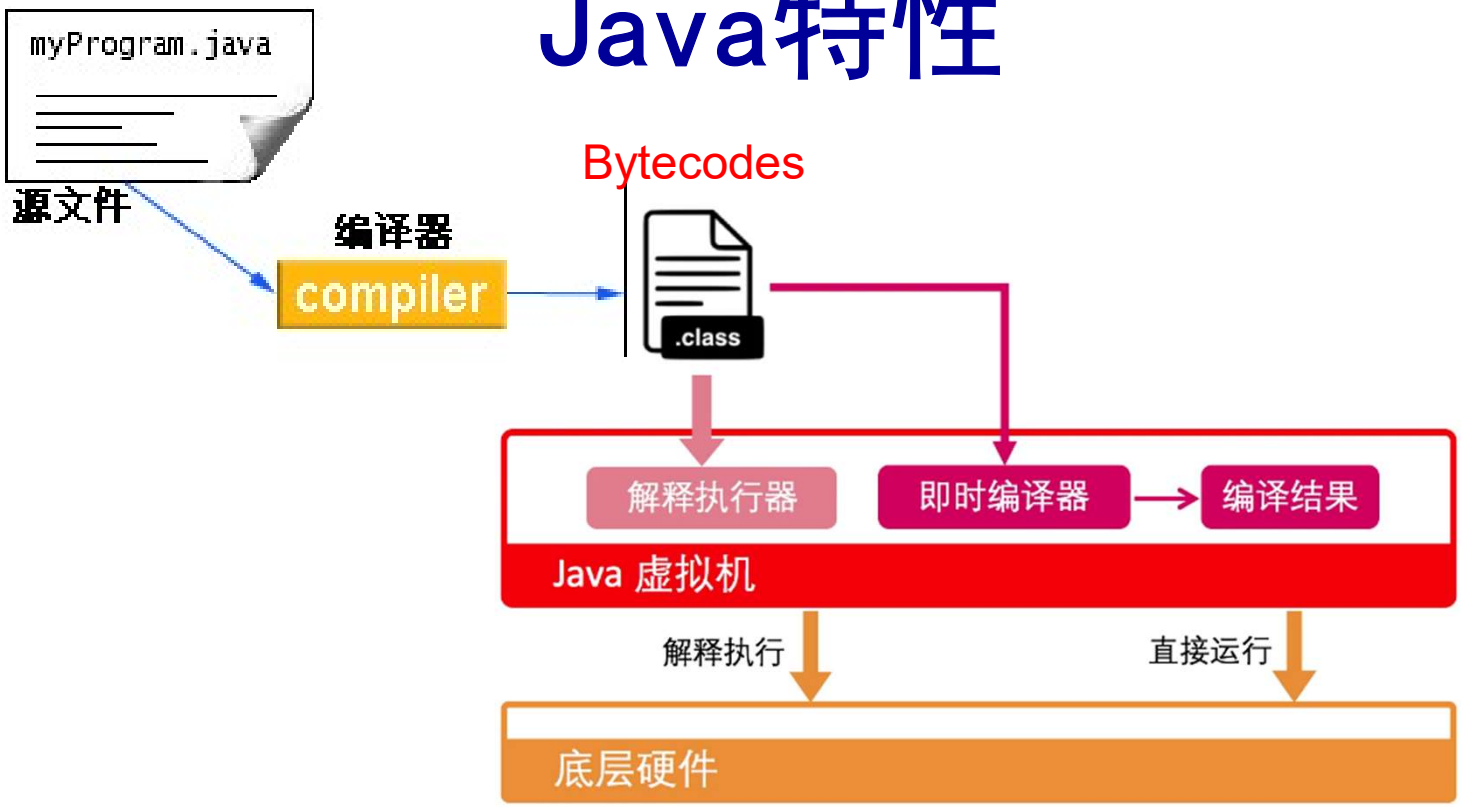
编程方法与技术

2.1. 上次课复习

周扬帆

2021-2022第一学期

Java特性



- 简单、面向对象、易学易用
- 鲁棒、安全
- 结构中立、可移植
- 高性能
- 解释执行、多线程、动态

Java程序的类

```
class Bird
{
    int color;
    float weight;
    int Fly()
    {
        …
    }
}
```

数据：域 → 基本数据类型

short int long
float double
boolean byte char

→ 转变丢不丢精度?

double d = 2/4;
d = ?

double d = 2.0/4;
d = ?

double d = (double)2/4;
d = ?

Java程序的类

```
class Bird
{
    int color;
    float weight;
    int Fly()
    {
        …
    }
}
```

数据：域 → 基本数据类型

short int long
float double
boolean byte char

→ 扩展数据类型： 数组

声明 (不指定大小)

```
int[] a1, a2;
```

创建 (灵活指定大小)

```
a1 = new int [10];

int n = 10;
a2 = new int [n];
```

Java程序的类

```
class Bird
{
    int color;
    float weight;
    int Fly()
    {
        …
    }
}
```

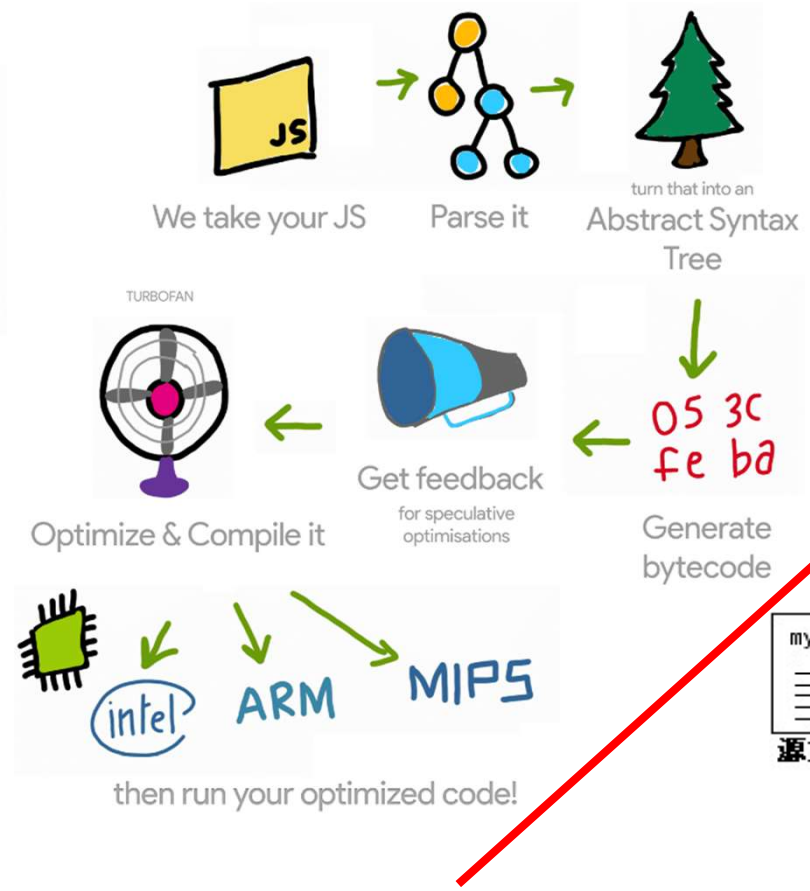
逻辑：方法 →

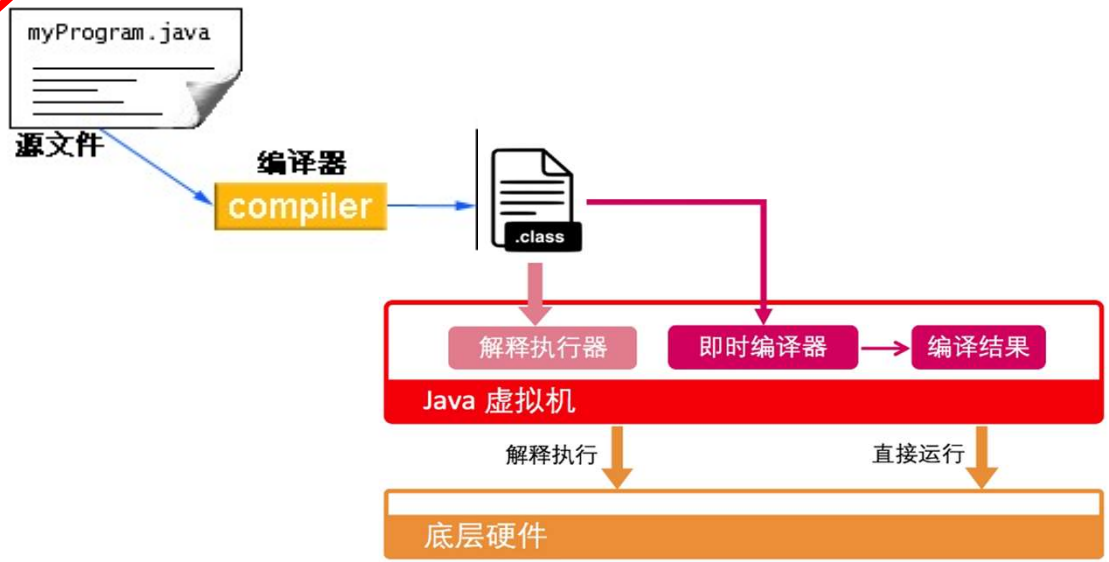
→ 运算符、表达式

→分支
(if/else, switch/case/break/default)

→循环
(for, do/while, while)

JavaScript运行原理





JS基础语法

- 基本数据类型
  - **String(字符)** **'abc'**
  - **Number(数字)** **12**
  - **Boolean (布尔)** **true**
  - **Null (空)** **null**
  - **Undefined (未定义)** **undefined**

JS基础语法

- 复合数据类型
  - **Array (数组)** **[1, 2, 3]**
  - **Object (对象)** **{name: 'YZ'}**

```
var arr1 = [1, 2, 3];
var arr2 = [1, 'Hello World', null];
```

```
var obj1 = {
        key: 'name',
        value: 'YZ'
};
```

```
var obj2 = {
        key: 'information',
        value: [1, 'YZ']
};
```

JavaScript typeof关键字

```
typeof undefined === "undefined"; // true
typeof true === "boolean"; // true
typeof 42 === "number"; // true
typeof "42" === "string"; // true
typeof { life: 42 } === "object"; // true
```

typeof null === “object”
typeof function a() {} === “function”
typeof [1,2,3] === “object”

JS基础语法

- 表达式
  - ===、!==
- 分支
  - if/else if/else
  - switch/case/break
- 循环
  - for
  - do/while
  - while
  - break/continue

函数

- 定义
- 函数也是一种变量

```
function add(a, b) {
        return a + b;
}
alert(add(1, 2));

var add2 = add;
alert(add2(1, 2));
```

```
var add = function (a, b) {
                return a + b;
}
alert(add(1,2));
```

  - 很方便将一个函数作为参数传给另一个函数
  - 方便于实现函数式编程

函数的嵌套

- JS支持函数嵌套定义

```
var dispalysum = function (a, b) {
        var add = function(a, b) {
                return a + b;
        }
        alert(add(1,2));
}
dispalysum(1,2);
```

```
// alert(add(1,2));
```

怎么调add?

练习一：螺旋

1. 输入一个数n
2. 产生一个n*n数组
3. 输入一个角的位置 （左上、左下、右上、右下）
4. 从这个位置开始，顺时针，填入数字1到n*n

7	8	1
6	9	2
5	4	3

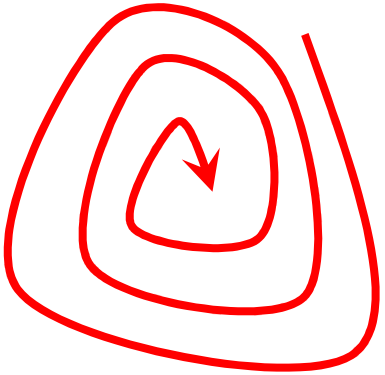
注： 不要用通式a[i][j] = f(i, j)

上次课的练习：最朴素的思路

- a[i, j]
- //用一个变量direction代表移动方向

7	8	1
6	9	2
5	4	3

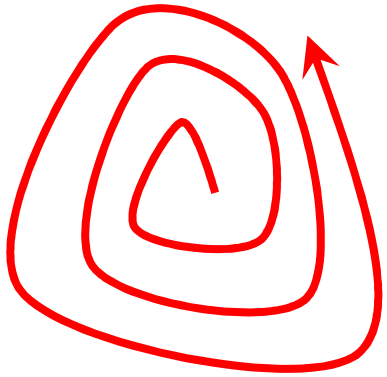
- //默认向某方向（根据输入的角）
- //向下(j++)遇到阻碍(j>=n||a[i,j]!=0)，因为顺时针，所以方向改向左
- //向左(i--)遇到阻碍 (j>=n||a[i,j]!=0)， 因为顺时针，所以方向改向上
- //向上遇到阻碍，因为顺时针，所以方向改向右
- //向右遇到阻碍，因为顺时针，所方向改以向下
- 都走不通 → 结束



练习二： 螺旋

1. 输入一个奇数n
2. 产生一个n*n数组
3. 从中心开始，逆时针（首先向上），填入数字1到n*n

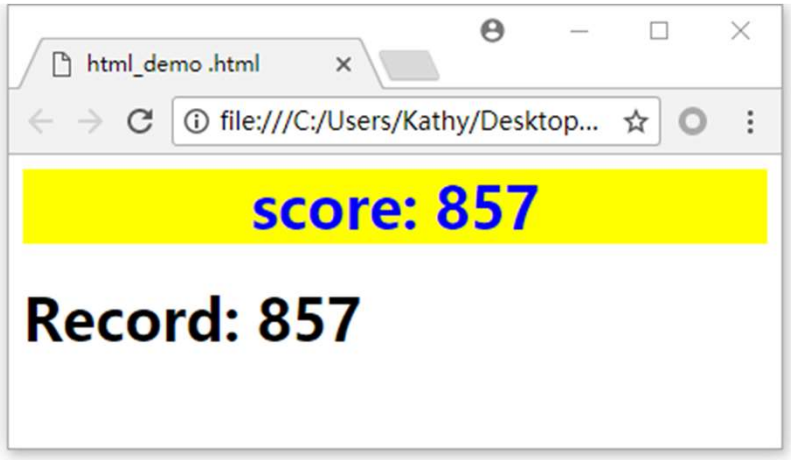
3	2	9
4	1	8
5	6	7



注： 不要用通式a[i][j] = f(i, j)

练习三：网页小游戏

- 写一个小游戏网页
  - 计算双击之间的时间差
  - 必须小于1000ms
  - 越接近1000ms分数越高



  - 参考: http://y-droid.com/Prog-01.html

练习三：网页小游戏

- **写一个小游戏网页**

状态

定义一个变量记录是奇数点击还是偶数点击
定义一个变量记录上次点击事件

被点击之后

如果是奇数点击，更新点击时间
如果是偶数点击，计算点击时间差

```
<HTML>
<HEAD>
<SCRIPT>
function letsgo() {
    alert(getTime());
}
function getTime() {      // 得到当前离盘古开天地的时间差（毫秒）
    myDate=new Date();
    return Number(myDate.getTime());
}
</SCRIPT>
</HEAD>
```

网页

```
<BODY>
<button onclick="letsgo()">CLICK ME</button>
</BODY>
</HTML>
```

练习四：冒泡排序

1. 给定n, 创建一个数组 nums[n]
2. 产生n个随机数，范围1 → n*n，分别存到数组nums

```
Random ra = new Random();
循环n次
{
        随机数 = ra.nextInt(n*n)+1
}
```

3. 用冒泡排序对数组进行排序, 结果放回num
4. 输出

```
for(int i: nums)
{
        System.out.println(i);
}
```

复旦大学计算机科学技术学院

编程方法与技术

2.2. JavaScript的var

周扬帆

2021-2022第一学期

关于var

- **var用于变量声明**

```
var a = 'Hello World!';
alert(a);
```

- **var的作用域**
  - **函数作用域**

```
function a() {
    console.log(value);
    var value;
}
a();
```

v.s.

```
function a() {
    console.log(value1);
    var value;
}
a();
```

关于var

- **var定义的变量**
  - **函数作用域**

```
var value = 'local';
var func = function() {
    var value = 'func_local';
    console.log(value);
}
func();
console.log(value);
```

关于var

- **var定义的变量**
  - **链式作用域 – chain scope**
  - **函数一层层往外找，直到全局**

```
var value = 'global';
var func1 = function {
    var value = 'local';
    var func2 = function() {
        console.log(value);
    }
    func2();
}
func1();
```

关于var

- **var定义的变量**
  - **函数作用域**

```
var value = 'local';
var func = function() {
    if (false) {
        var value = 'func_local';
    }
    console.log(value);          undefined
}
func();
console.log(value);
```

更多关于var

```
var a = 1;
function test () {
      var a = 2;
}
test();
alert(a);
```

1

```
var a = 1;
function test () {
       a = 2;
}
test();
alert(a);
```

2

复旦大学计算机科学技术学院

编程方法与技术

2.3. 程序的测试入门

周扬帆

2021-2022第一学期

进阶问题：测试

- **怎么发现程序的bug： 以排序为例**

人来看：排序结果确实是有序的

问题？

- 只能看有限的情况
- 看得慢
- 有可能看错

进阶问题：测试

- **怎么发现程序的bug： 以排序为例**

机器来看：排序结果确实是有序的

问题？ 如何设计这样一个能看程序对不对的程序

Oh my goodness! Shut me down. Machines building machines. How perverse!

软件是否满足设计规格specification

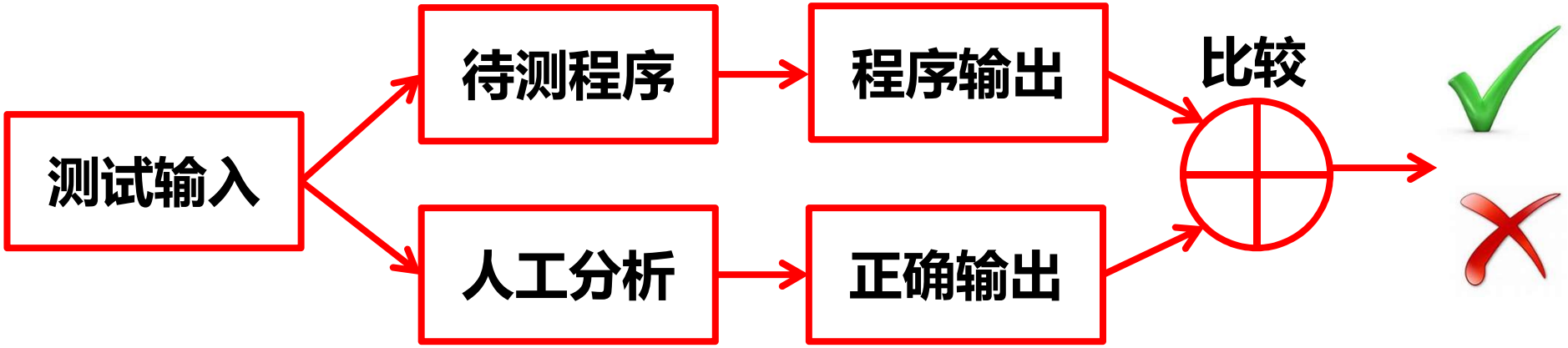
进阶问题：测试

- **怎么发现程序的bug： 以排序为例**

人来看：排序结果确实是有序的

启发：人来看的过程

- 给程序一组输入 (test inputs)
- 程序产生一组输出 (test results)
- 判断输出是不是符合预期



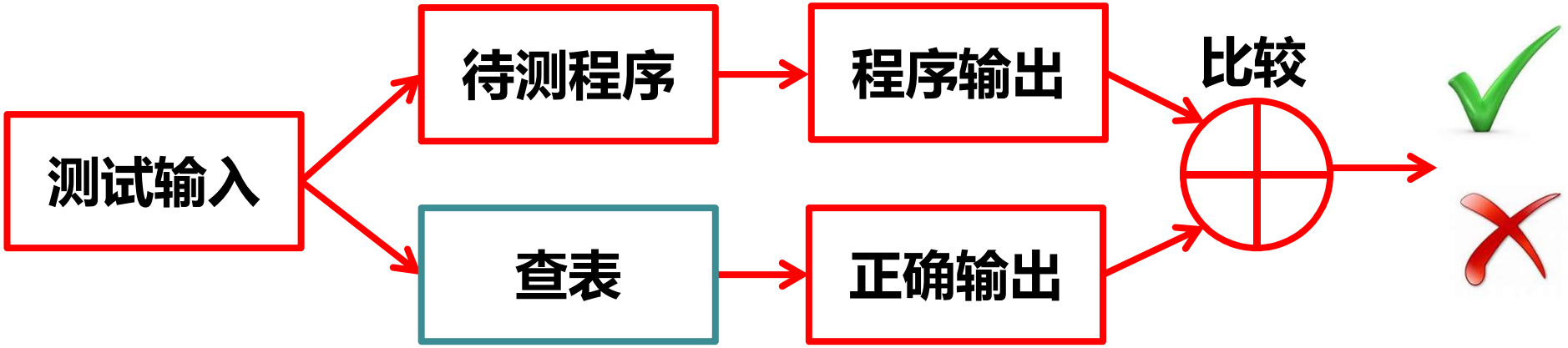
进阶问题：测试

- **怎么发现程序的bug： 以排序为例**

机器来看：排序结果确实是有序的

方法1： 查输入输出对应表?

2 6 7 4 3 → 2 3 4 6 7
1 2 4 7 5 → 1 2 4 5 7
……



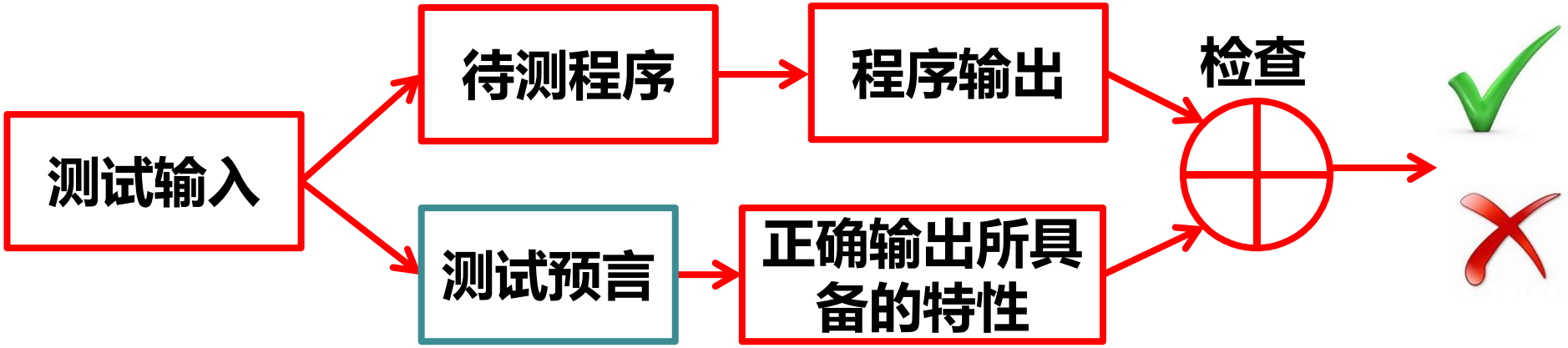
进阶问题：测试

- **怎么发现程序的bug： 以排序为例**

机器来看：排序结果确实是有序的

测试预言 (test oracle)

- 给程序一组输入 (test inputs)
- 程序产生一组输出 (test results)
- 判断输出是不是符合预期



练习三：冒泡排序程序的测试

对于结果nums[0 → n-1] 怎么检验结果

测试用例怎么定（测试输入和预期结果怎么定）

进阶问题：测试

测试输入的产生

1. **随机产生**
2. **人工构造边缘数据 (corner cases)**

如测试sin(x)：x = 0，π/2, π, 2π

进阶问题：测试

检查结果具备的特性，例如：

1. **界: sin(x)必须在[-1, 1]区间**
2. **恒等式: sin(x) = sin(x +2π)**

练习四：冒泡排序程序的测试

对于结果nums[0 → n-1]，怎么检验结果

思考结果特性，写测试程序

练习四：螺旋程序的测试

1. 给定一个数n
2. 产生一个n*n数组
3. 给定一个角的位置 （左上、左下、右上、右下）
4. 从这个位置开始，逆时针，填入数字1到n*n

7	8	1
6	9	2
5	4	3

如何写一个程序，检验螺旋程序结果

思考

1. 测试新的排序算法，我们可以利用一个现有的排序算法作为测试预言。用正确的程序来做测试预言，对于一般的软件来说，可行性如何

2. 测试能否保证程序100%正确

3. 工业上一般如何评估一个程序测试的充分性

4. 除了输出结果正确，测试还需要关注什么

5. 理解“测试驱动开发”

复旦大学计算机科学技术学院

编程方法与技术

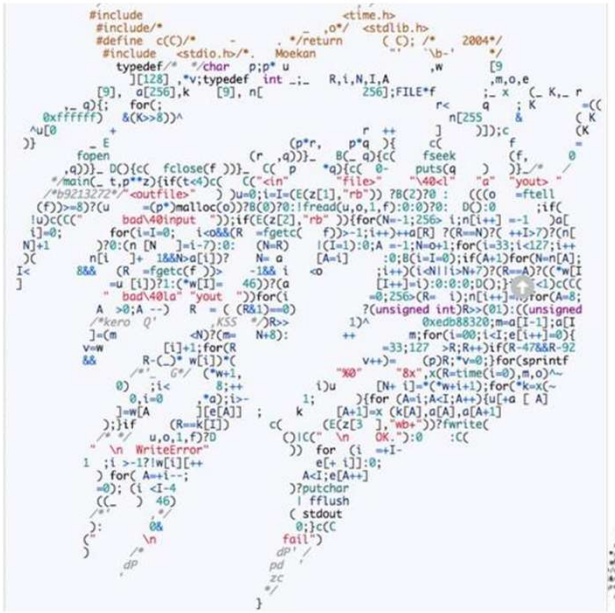
2.4. 代码风格

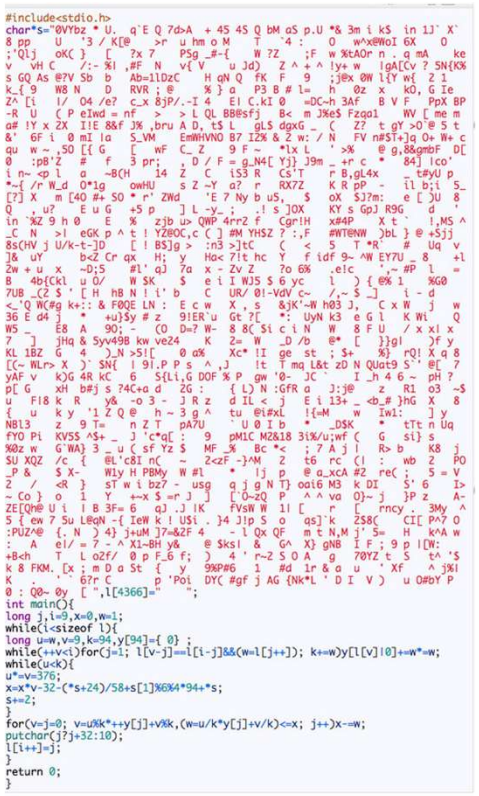
周扬帆

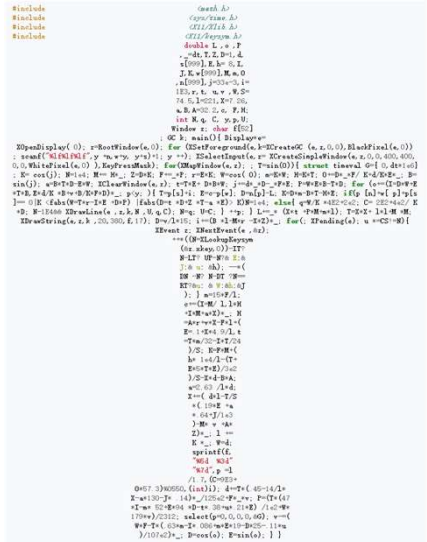
2021-2022第一学期

IOCCC

- The International Obfuscated C Code Contest
  - www.ioccc.org/
  - 一年一次







代码风格

- 为什么要代码风格coding style
  - 人要活得有style, 我有我风格?
  - No!
- 软件生命周期
  - 大部分时候，代码是要维护的：修bug，改进
  - 大部分时候，维护代码的人 != 开发代码的人
- 代码风格
  - 协助别人理解
  - 协助自己理解（遗忘）

内容纲要

1. 命名规范
2. 排版风格
3. 注释
4. 其他

命名规范

- 起名字
  - 人如其名
  - 韩熙惠 金玄哲？

相关书籍

测试你的韩国名字

用生日来测试的哦！1、你出生年的最后一个数字： 1 易 2 金 3 宋 4 徐 5 吴 6 安 7 林 8 圣 9 韩 0 李 2、你出生月份 女：1 媛 2 翠 3 天 4 柔 5 熙 6 希 7 韵 8 芸 9 允 10 甜 11 沅 12 语 男：1天 2子 3太 4泰 5夜 6成 7七 8小 9伊 10玄 11轩 12炫 3、你的出生日期最后一个数字 女：1 亭 2 婷 3 圆 4 纤 5 儿 6 惠 7 慧 8 恬 9 甜 0 美 男：1 元 2 贤 3 宪 4 羽 5 宇 6 浩 7 哲 8 生 9 俊 0 峻

宝宝起名

怎样起个好名字

中国起名宝典

命名规范

- 起名字
  - 人如其名
  - 金熙惠　金玄哲？
  - 好名字?
    - → 朱月坡
    - → 秦寿生
    - → 范统
    - → 沈京兵

命名规范

- **命名规范: 有利于理解代码**

文件名: StudentRecord.java		MyClass.java
函数名: getRecordbyStudentID	v.s.	naJilu
变量名: studentName		stringA

- **c/c++命名规范**
  - **匈牙利命名法则**
    - → **微软**
    - → **nStudentNumber, pName, m_pName, g_pName**
    - → **[作用域+下划线]+变量类型+变量描述**
    - → **变量描述：词组或词，首字母大写**

命名规范

- c/c++命名规范
  - 匈牙利命名法则
    - 微软
    - nStudentNumber, pName, m_pName, g_pName
    - [作用域+下划线]+变量类型+变量描述
    - 变量描述：词组或词，首字母大写
  - 某些批评：太麻烦，废话太多
    - strcpy(pstrFoo,pcstrFoo2) 谁不知道是指针参数
    - c++太复杂，static const unsigned long … scul?

命名规范

- 蛇形命名法：snack_case
  - like_this，常见于Linux内核，C++标准库等
- 驼峰命名法：CamelCase
  - 小驼峰：likeThis， 第一个字母小写
  - 大驼峰（帕斯卡命名法）：LikeThis，第一个字母大写

命名规范

- Java/JavaScript风格惯例
  - 变量和方法: 小驼峰
    - → getRoomNumber
    - → roomNumber
  - 常量: 蛇形全大写 MAX_VALUE
    - → static final int MIN_WIDTH = 14;
- Java风格惯例
  - 类和接口名: 大驼峰
    - → StudentHall
  - 包名:全部使用英文小写字母
    - → 项目： <域名反转>.<团队名>.<父项目名>.<子项目名>

命名规范

- 遵从一般人的风格
  - 类名: 大驼峰
  - 变量和方法: 小驼峰
- 名字取容易理解的词

文件名: StudentRecord.java
函数名: getRecordbyStudentID
变量名: studentName

- 尽量不要用拼音
  - String xueShengXingming
  - String studentName

命名规范

- 变量和常量: 名词词组
  - studentName
- 方法: 动词词组
  - getStudentName
- 类和接口: 名词词组
- 对象名: 对象属于变量，名词词组

命名规范

- **除非是作用域很小的变量，否则避免单字母随便拎过来就用**

```
for(int i = 0; i < studentNum; i++)
```

```
int a[] = new int [n];
//此处省去30行

//看到这里就费事去理解a是什么了
caculateValueofP(a);
a[m] = 30;
```

内容纲要

1. 命名规范
2. 排版风格
3. 注释
4. 其他

排版风格

- 排版也有助于理解代码的结构

```
for(;;){int i = j = 10; do{
i --; j++}while(  i != 0)
```

- K&R缩进风格

```
if (<cond>) {
        <body>
}
```

  - 缩进8或者4个空格，或者一个tab
  - { 在上一行末
  - } 独立一行

排版风格

- **BSD缩进风格**

```
if (<cond>)
 {
        <body>
 }
```

  - **缩进8或者4个空格，或者一个tab**
  - **{ 和 }都独立一行**
- **其他缩进风格**

Java排版风格

- Java多采用K&R风格
- 缩进用tab还是用空格?
  - 随便, 保持统一
- 一行不超过一条语句
  - if/else单独一行
  - for/while/do单独一行
  - switch/case/default/单独一行

Java排版风格

- 长行切断换行规则
  - 切在比较好理解的分界点
  - 下一行的缩进要考虑上一行，建议比上一行缩进一个位置 （tab或者4/8个空格）
  - if (((a == b) && (c ==d)) || (a > c)
    || c< 0)

Java排版风格

- 大部分操作符前后留空格
  - int i = 10; 等号前后都有空格
  - 一元操作符除外: !b
- 小括号
  - if ((a == b) && (d == e))
    - → (前面不是括号或者一元操作符，便和前面间隔空格
    - →) 后面不是括号，便和后面间隔空格
  - 函数的(前面不留空格
- 中括号，前后没有空格

……

Java排版风格

- 现代集成开发环境都带风格自动调整
  - Eclipse
    - → Ctrl – Shift – f
    - → Preference →Java→ Code Style → Formatter
  - IntelliJ
    - → Ctrl – Alt – l
    - → Editor (Ctrl – Alt – s) → Code Style → Java
- 课后请尝试一下

内容纲要

1. 命名规范
2. 排版风格
3. 注释
4. 其他

Java注释风格

- 原则
  - 多注释
  - 注释形式统一

    如/*　　*/ 用于类、方法、域的注释

    // 用于程序内部逻辑的注释

    todo: 用于表示未完成功能

    fixme: 用于表示是临时代码，需要更改
  - 注释内容准确简洁

    注释一定要准确，不模棱两可

Java注释风格

- 一般需注释
  - 类（接口）的注释
  - 构造函数的注释
  - 方法的注释
  - 域变量的注释

Java注释风格

- 一般需注释
  - 典型算法必须有注释
  - 在代码不明晰处必须有注释
  - 在代码修改处加上修改标识的注释
  - 在循环和逻辑分支组成的代码中加注释
  - 为他人提供的接口必须加详细注释。

Java注释风格

- 单行(single-line)注释： //……
- 块(block)注释： /*……*/
- 文档(javadoc)注释：/**……*/
  - Javadoc: 从代码中抽取类、方法、成员的规范的注释中自动生成一个API帮助文档

Java注释风格

- Javadoc基本关键字

注释中可以出现的关键字以@开始	意义
@author	作者名
@version	版本标识
@since	最早出现的JDK版本
@deprecated	引起不推荐使用的警告
@see	交叉参考

Java注释风格

- **Javadoc方法关键字**

@return	返回值
@throws	异常类及抛出条件
@param	参数名及其意义

Java注释风格

Javadoc例子

```
public class BusTestJavaDoc {
        public int maxSpeed;
        public int averageSpeed;
        public int waterTemperature;
        public int Temperature;
        BusTestJavaDoc() {
        }

        public int measureAverageSpeed(int start, int end) {
                int aspeed = 12;
                return aspeed;
        }

        public int measureMaxSpeed( ){
                return maxSpeed;
        }
}
```

Java注释风格

- **Javadoc例子：类名**

```
1 /**
2 *汽车类的简介
3 *<p>汽车类具体阐述第一行<br>
4 *汽车类具体阐述第二行
5 *@author man
6 *@author man2
7 *@version 1.0
8 *@see ship
9 *@see aircraft
10 */
11 public class BusTestJavaDoc{
```

Java注释风格

- **Javadoc例子：类的变量**

```
12    /**
13    *用来标识汽车行驶当中最大速度
14    *@see #averageSpeed
15    */
16    public int maxSpeed;
17    /**用来标识汽车行驶当中平均速度*/
18    public int averageSpeed;
19    /**用来标识汽车行驶当中的水温*/
20    public int waterTemperature;
21    /**用来标识天气温度*/
22    public int Temperature;
```

Java注释风格

- **Javadoc例子：函数**

```
26    /**
27     *该方法用来测量一段时间内的平均速度
28     *@param start 起始时间
29     *@param end 截止时间
30     *@return 返回int型变量
31     *@exception java.lang.exceptionthrowwhenswitchis1
32     */
33    public int measureAverageSpeed(int start,int end ){
34        int aspeed=12;
35        return aspeed;
36        }
```

Java注释风格

- **Javadoc例子：函数**

```
37    /**
38     * 该方法用来测量最大速度
39     */
40    public int measureMaxSpeed(){
41        return maxSpeed;
42
43    }
```

Java注释风格

- **Javadoc的输出**
  - **IDE鼠标放过去，就有显示**
  - **IntelliJ**
    - → **Tools → Generate Javadoc**
  - **命令javadoc**

    $\> javadoc XXX.java

y-droid.com/javadoc

Java逻辑代码注释经验

- 多注释永远是个梦，除非...
- 不要在实现好之后注释
  - 一般码农哪有那闲功夫
- 在写代码之前注释
  - 用注释理清代码逻辑
  - 再开始写代码

Java逻辑代码注释经验

- **在写代码之前注释: 例子**
  - //用一个变量direction代表移动方向
  - //得到初始方向（根据输入的角）
  - //向该方向赋值
    - → //向下赋值失败(遇到阻碍)，因为顺时针，所以direction变向左
      - //不可以向左，结束
    - → //向左赋值失败(遇到阻碍)， 因为顺时针，所以direction变向上
      - //不可以向上，结束
    - → //向上赋值失败(遇到阻碍)， 因为顺时针，所以direction变向右
      - //不可以向右，结束
    - → //向右赋值失败(遇到阻碍)， 因为顺时针， 所以direction变向下
      - //不可以向下，结束

Good design takes time

内容纲要

1. 命名规范
2. 排版风格
3. 注释
4. 其他

其他

- **不要在判断的boolean表达式里写语句**

```
if ( number == MAX_NUM )
{
        …
}
```

C语言

```
if ( number = MAX_NUM )
{
        …
}
```

C语言

```
if (MAX_NUM == number)
{
        …
}
```

```
if ( (number = MAX_NUM) == MAX_NUM )
{
        …
}
```

其他

- **避免多变量同时赋值，在其他语句中赋值**

```
int number = 0,  counter = MAX_NUM;
```

```
int number = 0;
int counter = MAX_NUM;
```

```
if ((c++ = d++) != 0)
```

```
d = (a = b + c) + r;
```

```
getStudentName (sID++);
```

其他

- **优先顺序不明显，加括号**

```
int k = i++ / j++;
```

```
int k = (i++) / (j++);
```

- **类型转换不确定，加强制转换**

```
double k = 2/3;
```

```
double k = (double)2/3;
```

不要酷，不赌一把

其他

Google Java Style

https://google.github.io/styleguide/javaguide.html

Google Java Style

https://google.github.io/styleguide/jsguide.html

简单的示例

http://mushiqianmeng.blog.51cto.com/3970029/737120/

思考

- 思考你自己编程的命名规范/排版样式/注释规范，思考是否是否需要改进
- 除了命名规范/排版风格/注释等，思考其他影响代码风格的因素
- 再啰嗦一遍
  - 怕别人看不懂你的代码
  - 不耍酷、不赌一把

复旦大学计算机科学技术学院

编程方法与技术

2.5. 类的方法

周扬帆

2021-2022第一学期

方法（函数）

- **C语言的函数定义**

```
int add(int a, int b)
{
        return a + b;
}

void bubbleSort(int *nums, int length)
```

- **Java语言的方法定义**
  - **类似**
  - **作用域 + 类型 + 返回值 + 函数名 (参数序列)**

```
public static void main (String[] args)
```

以后讲具体意思

引入方法（函数）的目的

- 提高代码可读性
  - 尤其对于复杂的程序过程(procedure)
  - 帮助分块理解
- 实现代码的可复用性
  - 比如c标准库的qsort
  - 简洁、省内存空间
- 便于实现库（library）
- 实现模块化设计
  - Separation-of-concerns

方法（函数）：软件工程视角

- **目的1：提高代码可读性**

```
int add(int a, int b)
{
        return a + b;
}

void bubbleSort(int *nums, int length)
```

螺旋的例子

```
boolean moveLeft()
boolean moveRight()
boolean moveUp()
boolean moveDown()
```

方法（函数）：软件工程视角

- **目的2：实现代码复用**
  - **课堂练习2：排序**

1. 对数组从大到小进行排序
2. 对数组从小到大进行排序

```
for(int i = 0; i < n; i ++)
{
    for(int j = 0; j < n - i - 1; j ++)
    {
        if (nums[j] < nums[j+1])
        {
            int temp = nums[j+1];
            nums[j+1] = nums[j];
            nums[j] = temp;
        }
    }
}
```

方法（函数）：软件工程视角

- **目的2：实现代码复用**
  - **课堂练习2：排序的函数**

```
void sort (boolean isMaxToMin)
{
    for(int i = 0; i < n; i ++)
    {
        for(int j = 0; j < n - i - 1; j ++)
        {
            if ((nums[j] < nums[j+1] && isMaxToMin )
                || (nums[j] > nums[j+1] && !isMaxToMin ))
            {
                int temp = nums[j+1];
                nums[j+1] = nums[j];
                nums[j] = temp;
            }
        }
    }
}
```

方法（函数）：软件工程视角

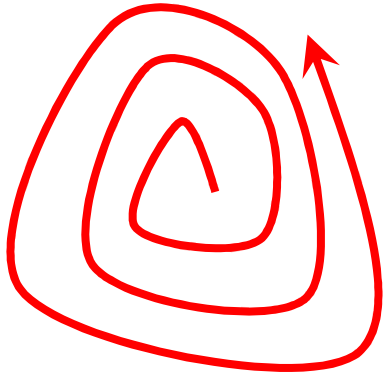
- **目的2：实现代码复用**
  - **课堂练习1：螺旋**

上次课的练习1

7	8	1
6	9	2
5	4	3

上次课的练习2

3	2	9
4	1	8
5	6	7



方法（函数）：软件工程视角

- **目的2：实现代码复用**
  - **课堂练习1：螺旋**

1. 将上次课的对数组的操作视为函数
2. 该函数输出结果为二维数组a[][]
3. 对该数组进行如下操作

```
for(int j = 0; j < n; j++)
{
    for(int i = 0; i < n; i++)
    {
        a[j][i] = n * n + 1 - a[j][i];
        System.out.print(a[j][i] + " ");
    }
    System.out.println();
}
```

方法（函数）：软件工程视角

- **目的3：便于实现库**
  - **先行者实现很多常见的算法**
  - **封装成函数，打包成库**
    - → **C语言标准库**
    - → **C++类库**
    - → **JAVA类库**
  - **提高程序实现效率**
  - **语言被广泛使用的关键**

方法（函数）：软件工程视角

- **目的4：模块化设计**
  - **方便测试**
  - **方便修改**
  - **不同人/团队做不同的事情，松耦合**

思考

- 比较C/C++与Java声明方法的差异
- 思考总结提高代码可读性的方法
- 思考总结提高代码复用的方法

课下

- 微信群
  - 请同学拉你进去
  - 欢迎提问、交流
  - 请修改群昵称为真实姓名

复旦大学计算机科学技术学院

编程方法与技术

2.6. 课堂练习

周扬帆

2021-2022第一学期

练习：随机矩阵

1. 实现方法，输入整数n, 输出n*n矩阵，使得

   (1). 矩阵的元素为整数

   (2). 矩阵的行列式为1

   (3). 矩阵看起来比较随机 ☺

(调用每次生成的矩阵不一样，而且不能都是上三角矩阵)

2. 实现上述方法的测试代码

练习：随机矩阵

1. 实现方法，输入整数n, 输出n*n矩阵，使得

(1). 矩阵的元素为整数

(2). 矩阵的行列式为1

(3). 矩阵看起来比较随机 ☺ (调用每次生成的矩阵不一样，而且不能都是上三角矩阵)

问题1: 行列式是什么? **查!**

问题2: 什么样的矩阵行列式一定为1(行列式性质，**查!**)

问题3: 怎么将上述矩阵转换成比较随机的样子，行列式不变 (行列式性质，**查!**)

问题4: 随机数怎么生成: **查!**

```
java.util.Random random = new …
random.nextInt()
```

练习：随机矩阵

1. 实现方法，输入整数n, 输出n*n矩阵，使得

(1). 矩阵的元素为整数

(2). 矩阵的行列式为1

(3). 矩阵看起来比较随机 ☺

(调用每次生成的矩阵不一样，而且不能都是上三角矩阵)

2. 实现上述方法的测试代码

怎么算行列式？ **查！** **baidu/google: 抄代码**