实验 02：数据类型

一、实验目的

1、了解基本数据类型的使用；

2、了解内置数据类型的使用；

3、掌握 String、 StringBuffer、 StringBuilder 类各自的特点及用法；

4、掌握 Date 类和 SimpleDateFormat 类特点及用法。

5、掌握 LocalDate 和 LocalDateTime 类特点及用法

二、实验学时

2 学时

三、实验类型

验证性

四、实验需求

1、硬件

每人配备计算机 1 台，建议优先使用个人计算机开展实验。

2、软件

IntelliJ IDEA，以及 Java 运行所需要的相关基础环境。

3、网络

本地主机能够访问互联网和实验中心网络。

4、工具

无。

五、实验任务

1、编写程序,打印出 Java 八种数据类型的各自占多少个二进制位,最大值和最小值；

2、编写程序,定义并赋值一个整型数组,遍历这个数据,然后用 Debug 调试每次循环结果；

3、编写程序,定义一个集合 ArrayList 用来存储数据,遍历这个集合,然后用 Debug 调试每次循环结果。

4、编写程序,展示 String、 StringBuffer、 StringBuilder 类各自的特点及用法。

5、编写程序,展示使用 SimpleDateFormat 将日期(Date) 字符串与对象形式相互转换。

6、编写程序,使用 LocalDate 和 LocalDateTime 打印当前日期。

六、实验内容及步骤

1、编写程序,打印出 Java 八种数据类型的各自占多少个二进制位,最大值和最小值

```
public class DataTypeProperties {
  public static void main(String[] args) {
    System.out.println("Java 基本数据类型的大小、最大值和最小值:");

    // byte
    System.out.println("byte: " +
      "8 位, " +
      "最大值: " + Byte.MAX_VALUE + ", " +
      "最小值: " + Byte.MIN_VALUE);

    / / short
    System.out.println("short: " +
      "16 位, " +
      "最大值: " + Short.MAX_VALUE + ", " +
      "最小值: " + Short.MIN_VALUE);

    // int
    System.out.println("int: " +
      "32 位, " +
      "最大值: " + Integer.MAX_VALUE + ", " +
      "最小值: " + Integer.MIN_VALUE);

    // long
    System.out.println("long: " +
      "64 位, " +
      "最大值: " + Long.MAX_VALUE + ", " +
      "最小值: " + Long.MIN_VALUE);

    // float
    System.out.println("float: " +
      "32 位(IEEE 754), " +
      "最大值: " + Float.MAX_VALUE + ", " +
      "最小值: " + Float.MIN_VALUE + " (注意：这是正的最小非零值，负
的最小值为-" + (-Float.MIN_VALUE) + ")");

    // double
    System.out.println("double: " +
      "64 位(IEEE 754), " +
      "最大值: " + Double.MAX_VALUE + ", " +
      "最小值: " + Double.MIN_VALUE + " (注意：这是正的最小非零值，
负的最小值为-" + (-Double.MIN_VALUE) + ")");

    // char
    System.out.println("char: " +
      "16 位, " +
      "最大值: " + (int) Character.MAX_VALUE + ", " +
      "最小值: " + (int) Character.MIN_VALUE);

    // boolean（特殊说明）
```

```
        System.out.println("boolean: " +
            "不是以二进制位数直接表示, " +
            "值只能是 true 或 false");
    }
}
```

2、编写程序,定义并赋值一个整型数组,遍历这个数据,然后用 Debug 调试每次循环结果

```
public class Main {
    public static void main(String[] args) {
        // 定义并赋值一个整型数组
        int[] numbers = {1, 2, 3, 4, 5};

        // 遍历这个数组
        for (int i = 0; i < numbers.length; i++) {
            // 使用 System.out.println 来模拟调试输出
            System.out.println("当前循环的索引: " + i + ", 当前值: " + numb
ers[i]);

            // 这里是你可以设置断点的位置
            // 在大多数 IDE 中，你可以在行号旁边点击来设置断点
            // 当程序运行到这里时，它将暂停，你可以检查变量的当前值
        }
    }
}
```

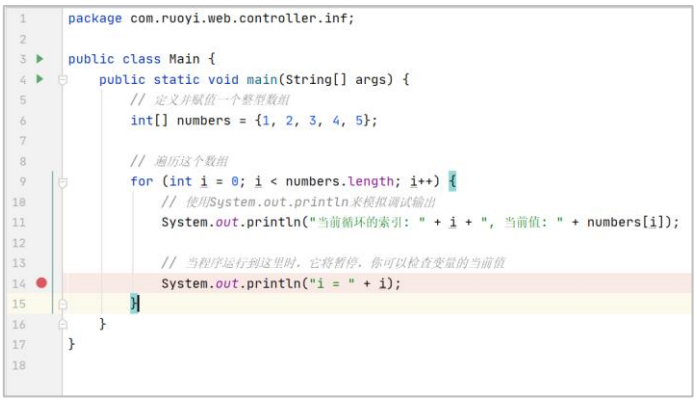

图 1

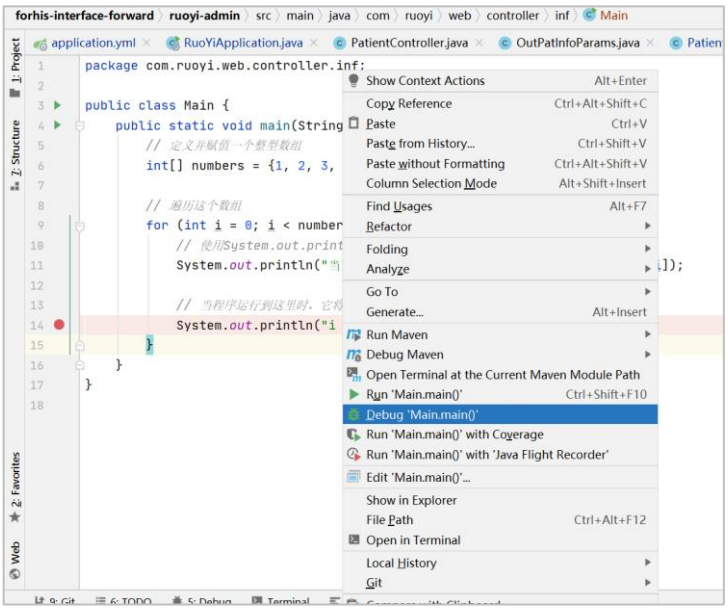
forhis-interface-forward ruoyi-admin src main java com ruoyi web controller inf Main
application.yml RuoYiApplication.java PatientController.java OutPatInfoParams.java Patien
package com.ruoyi.web.controller.inf;
public class Main {
public static void main(String
// 定义并赋值一个整型数组
int[] numbers = {1, 2, 3,
// 遍历这个数组
for (int i = 0; i < number
// 使用System.out.print
System.out.println("当
// 当程序运行到这里时，它将
System.out.println("i
Show Context Actions Alt+Enter
Copy Reference Ctrl+Alt+Shift+C
Paste Ctrl+V
Paste from History... Ctrl+Shift+V
Paste without Formatting Ctrl+Alt+Shift+V
Column Selection Mode Alt+Shift+Insert
Find Usages Alt+F7
Refactor
Folding
Analyze
Go To
Generate... Alt+Insert
Run Maven
Debug Maven
Open Terminal at the Current Maven Module Path
Run 'Main.main()' Ctrl+Shift+F10
Debug 'Main.main()'
Run 'Main.main()' with Coverage
Run 'Main.main()' with 'Java Flight Recorder'
Edit 'Main.main()'...
Show in Explorer
File Path Ctrl+Alt+F12
Open in Terminal
Local History
Git

图 2

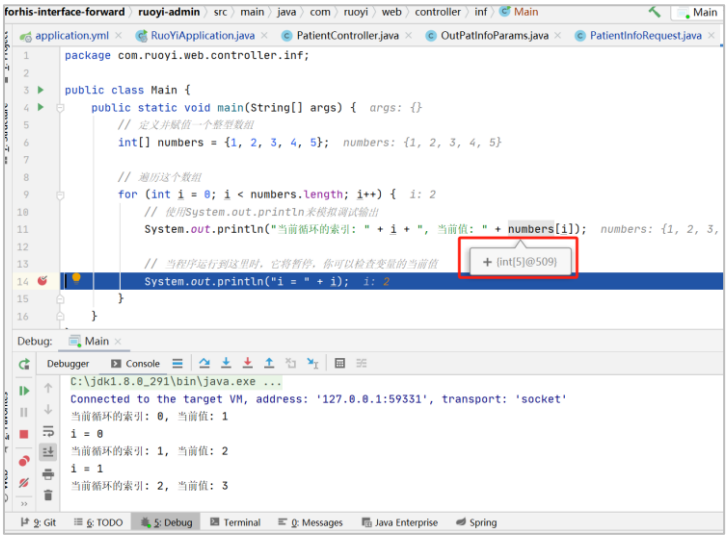
forhis-interface-forward ruoyi-admin src main java com ruoyi web controller inf Main
application.yml RuoYiApplication.java PatientController.java OutPatInfoParams.java PatientInfoRequest.java
package com.ruoyi.web.controller.inf;
public class Main {
public static void main(String[] args) { args: {}
// 定义并赋值一个整型数组
int[] numbers = {1, 2, 3, 4, 5}; numbers: {1, 2, 3, 4, 5}
// 遍历这个数组
for (int i = 0; i < numbers.length; i++) { i: 2
// 使用System.out.println来模拟调试输出
System.out.println("当前循环的索引: " + i + ", 当前值: " + numbers[i]); numbers: {1, 2, 3,
// 当程序运行到这里时，它将暂停，你可以检查变量的当前值
System.out.println("i = " + i); i: 2
{int[5]@509}
Debug: Main
Debugger Console
C:\jdk1.8.0_291\bin\java.exe ...
Connected to the target VM, address: '127.0.0.1:59331', transport: 'socket'
当前循环的索引: 0, 当前值: 1
i = 0
当前循环的索引: 1, 当前值: 2
i = 1
当前循环的索引: 2, 当前值: 3
9: Git 6: TODO 5: Debug Terminal 0: Messages Java Enterprise Spring

图 3

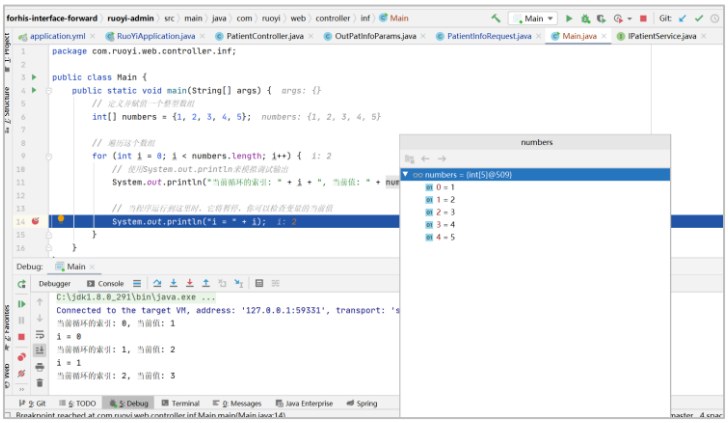
numbers
0 = 1
1 = 2
2 = 3
3 = 4
4 = 5

图 4

3 编写程序,定义一个集合 ArrayList 用来存储数据,遍历这个集合,然后用 Debug 调试每次循环结果

```
import java.util.ArrayList;

public class Main {
    public static void main(String[] args) {
        // 定义一个 ArrayList 来存储数据
        ArrayList<Integer> numbers = new ArrayList<>();

        // 向 ArrayList 中添加一些数据
        numbers.add(1);
        numbers.add(2);
        numbers.add(3);
        numbers.add(4);
        numbers.add(5);

        // 遍历这个 ArrayList
        for (int i = 0; i < numbers.size(); i++) {
            // 在这里，我们可以通过 System.out.println 来模拟调试输出
            // 实际上，在真实的调试环境中，我们会设置断点并使用 IDE 的调试功能
            System.out.println("当前循环的索引: " + i + ", 当前值: " + numbers.get(i));

            // 注意：这里是我们可能想要设置断点的位置
            // 在 IDE 中，我们可以在这一行设置断点，并在调试模式下运行程序
            // 当程序执行到这里时，它将暂停，允许我们检查变量的当前值和程序的执行状态
        }
    }
}
```

图 5

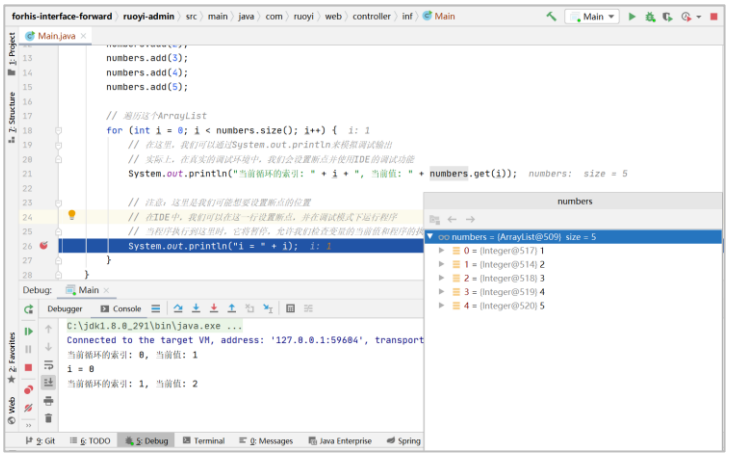

图 6

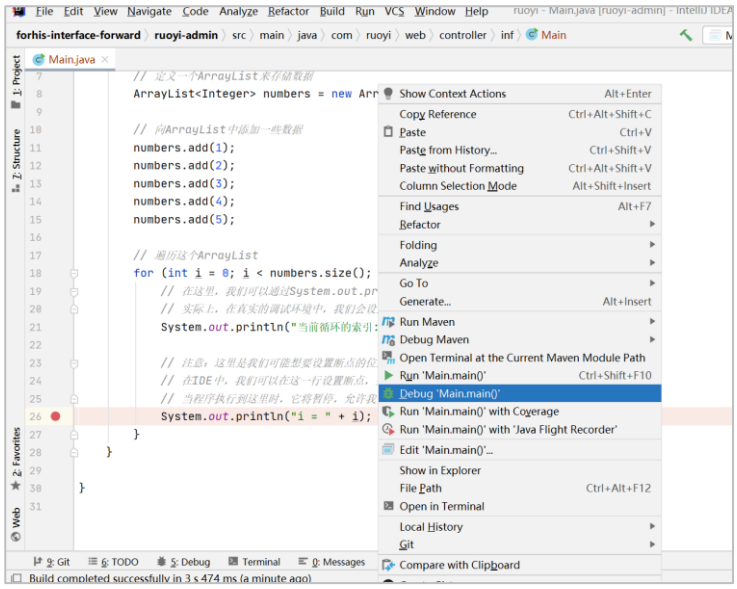

图 7

4、编写程序,展示 String、 StringBuffer、 StringBuilder 类各自的特点及用法

```
public class StringDemo {
public static void main(String[] args) {
//String 特性
    String str = "Hello, World!";
System.out.println(str); // 输出: Hello, World!

    // 字符串连接（注意：这会导致创建一个新的 String 对象）
    String newStr = str + " Again!";
System.out.println(newStr); // 输出: Hello, World! Again!

    //StringBuffer 特性
    StringBuffer sb = new StringBuffer("StringBuffer Initial");
sb.append(" Append");
System.out.println(sb.toString()); // 输出: StringBuffer Initial Append

    // 在指定位置插入字符串
```

```
        sb.insert(0, "Insert at beginning: ");
System.out.println(sb.toString()); // 输出: Insert at beginning: StringBu
ffer Initial Append
        //StringBuilder 特性
        StringBuilder sbBuilder = new StringBuilder("StringBuilder Initial
");

sbBuilder.append(" Append");
System.out.println(sbBuilder.toString()); // 输出: StringBuilder Initial A
ppend
        // 删除字符串的一部分
        sbBuilder.delete(0, 5); // 删除前 5 个字符
        System.out.println(sbBuilder.toString()); // 输出: Initial Append
    }
}
```

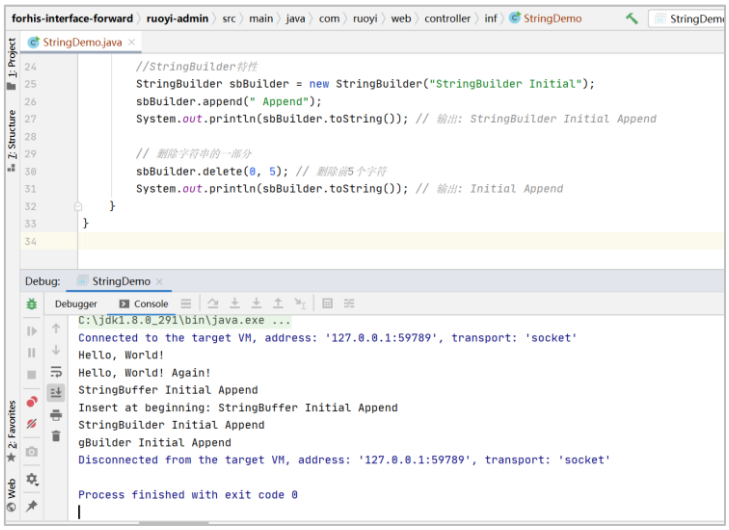

图 8

5 编写程序,展示使用 SimpleDateFormat 将日期(Date) 字符串与对象形式相互转换

```
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;

public class DateFormatDemo {
    public static void main(String[] args) {
    // 定义日期格式
    SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");

    // Date 对象转字符串
    Date now = new Date(); // 获取当前日期
    String dateStr = sdf.format(now); // 将 Date 对象转换为字符串
    System.out.println("当前日期（字符串格式）: " + dateStr);

    // 字符串转 Date 对象
    try {
```

```
    String specificDateStr = "2024-10-01"; // 假设我们有一个特定格式的日期字符串
        Date specificDate = sdf.parse(specificDateStr); // 将字符串转换为 Date 对象
        System.out.println("特定日期（Date 对象）：" + specificDate);
    // 注意：直接打印 Date 对象时，它并不会以我们指定的格式显示
        // 如果需要以特定格式显示 Date 对象，我们仍然需要使用 SimpleDateFormat 的 format 方法
        System.out.println("特定日期（重新格式化为字符串）：" + sdf.format(specificDate));
        } catch (ParseException e) {
        e.printStackTrace();
        System.out.println("日期字符串解析失败");
        }
    }
}
```

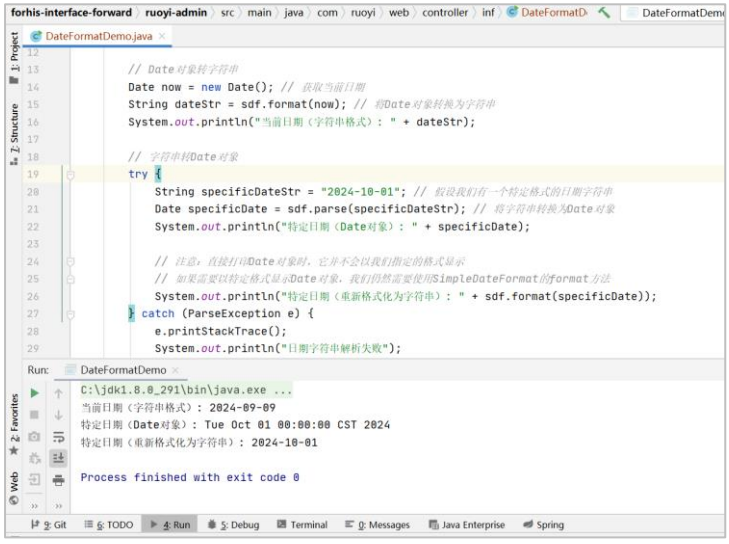

图 9

6、编写程序,使用 LocalDate 和 LocalDateTime 打印当前日期

```
import java.time.LocalDate;

public class LocalDateDemo {
public static void main(String[] args) {
// 获取当前日期
    LocalDate today = LocalDate.now();
System.out.println("今天的日期是: " + today);

// 创建特定日期
    LocalDate specificDate = LocalDate.of(2023, 10, 1);
System.out.println("特定日期是: " + specificDate);

// 日期加减
    LocalDate tomorrow = today.plusDays(1);
LocalDate lastMonth = today.minusMonths(1);
```

```
System.out.println("明天是: " + tomorrow);
System.out.println("上个月同期是: " + lastMonth);

// 日期比较
    if (today.isAfter(specificDate)) {
System.out.println("今天晚于特定日期");
    }
  }
}

import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;

public class LocalDateTimeDemo {
public static void main(String[] args) {
// 获取当前日期和时间
    LocalDateTime now = LocalDateTime.now();
System.out.println("现在的日期和时间是: " + now);

// 创建特定日期和时间
    LocalDateTime specificDateTime = LocalDateTime.of(2024, 10, 1,
12, 0, 0);
System.out.println("特定日期和时间是: " + specificDateTime);

// 日期时间加减
    LocalDateTime later = now.plusHours(2);
LocalDateTime earlier = now.minusMinutes(30);
System.out.println("两小时后是: " + later);
System.out.println("半小时前是: " + earlier);

// 日期时间比较
    if (now.isAfter(specificDateTime)) {
System.out.println("现在晚于特定日期和时间");
    }

// 使用自定义的格式化模式
    DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yy
yy-MM-dd HH:mm:ss");

String specificDateTimeStr = specificDateTime.format(formatter);

System.out.println("自定义格式日期时间: " + specificDateTimeStr);
String laterStr = later.format(formatter);
System.out.println("自定义格式日期时间: " + laterStr);
String earlierStr = earlier.format(formatter);
System.out.println("自定义格式日期时间: " + earlierStr);
  }
}
```

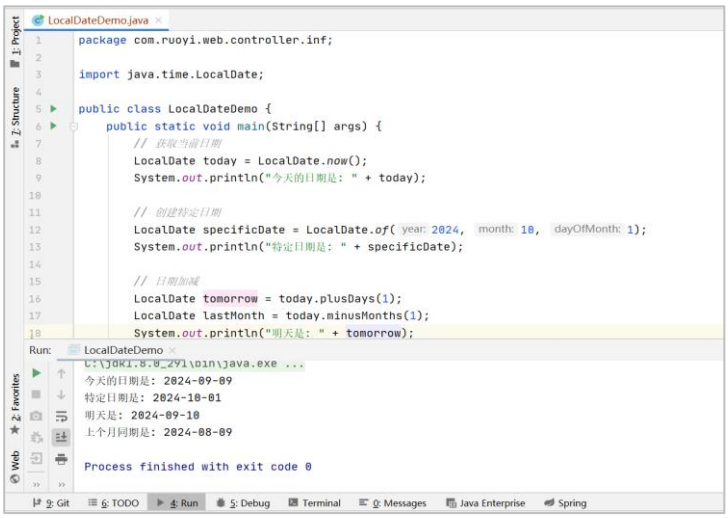

图 10

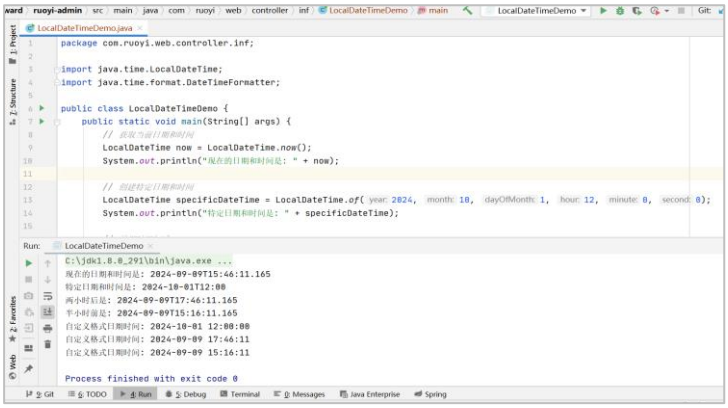

图 11

七、实验考核

本实验考核采用【实验随堂查】方式开展。

每个实验完成后，在实验课上通过现场演示的方式向实验指导教师进行汇报，并完成现场问答交流。

每个实验考核满分 100 分，其中实验成果汇报 60 分，现场提问交流 40 分。

实验考核流程：

（1）学生演示汇报实验内容的完成情况，实验指导老师现场打分。

（2）指导老师结合实验内容进行提问，每位学生提问 2-3 个问题，根据回答的情况现场打分。

（3）实验考核结束后，进行公布成绩。

八、创作说明

本实验指导书由河南中医药大学信息技术学院互联网技术教学团队与河南方和信息科技有限公司联合创作。

作者：黄子杰（河南方和信息科技有限公司）

审核：阮晓龙（河南中医药大学信息技术学院）

排版：冯冰（河南中医药大学智能医学工程专业 2023 级）