**哈尔滨工业大学计算机科学与技术学院**

**2016年秋季学期《软件工程》**

**Lab 1：结对编程**

|  |  |  |
| --- | --- | --- |
| **姓名** | **班级/学号** | **联系方式** |
| 金策 | 3班/1140310301 | 1204125485@qq.com/13936355857 |
| 杜冠宏 | 3班/1143510120 | 523678660@qq.com/18627978736 |

**目 录**

[1 实验要求 1](#_Toc6686)

[2 待求解问题描述与数学模型 1](#_Toc15393)

[3 算法与数据结构设计 2](#_Toc32446)

[3.1 设计思路 2](#_Toc6373)

[3.2 算法流程图 3](#_Toc20649)

[3.3 核心数据结构 3](#_Toc21028)

[3.4 算法时间复杂度分析 3](#_Toc24845)

[4 实验结果 4](#_Toc4407)

[4.1 输入/期望输出/实际输出 4](#_Toc25914)

[4.2程序运行界面截图 5](#_Toc2591)

[5 编程语言与开发环境 6](#_Toc3414)

[6 结对编程 6](#_Toc28649)

[6.1 分组依据 6](#_Toc32182)

[6.2 角色切换与任务分工 7](#_Toc1328)

[6.3 工作照片 7](#_Toc10197)

[6.4 工作日志 8](#_Toc29086)

[7 计划与实际进度 8](#_Toc10313)

[8 小结 8](#_Toc3237)

# 实验要求

**一.表达式输入**

对输入的表达式做如下限定：  
1.满足多项式的基本规则；  
2.不包含任何非数字、非字母、非+和\*的符号 (如括号、小数点、减号等)；  
3.多项式中的变量用单字母表示 (即不允许多字母的变量，意味着多个变量之间必须用\*号相连)；

4.“ \*”可以忽略；  
5.如果不满足上述条件，则展示错误信息。

**二. 命令输入**

1.一条命令(Command)以“ !”开头；  
2.针对求值命令，其格式为“ ! simplify var1=num1 ... varn=numn”  
• 如果simplify后面未指定var及其取值，仍为合法，直接展示当前表达式；  
• 如果后面指定了var的取值，则将当前表达式中var替换为具体的取值，并作适当简化运算，展示结果；例如，针对3\*x+2\*x\*y执行simplify x=2，输出结果可以为3\*2+2\*2\*y，但更理想的情形是输出6+4\*y；  
• 可以指定当前表达式中的全部变量取值，也可以只指定其中一部分的值；  
• 多个var及其取值之间用空格分隔  
• 变量的取值为正整数 (扩展的可选要求：可以为带小数点，也可以为负数)  
3.针对求导命令，其格式为“ !d/d var”  
• d/d后面为空格和变量名，且该变量名一定在当前表达式中出现；  
• 有且只能有一个变量名；

**三．实验要求**

提交一个.java文件，其中至少包含以下函数：  
– main()：主程序入口，接收用户输入，返回给用户计算结果  
– expression()：处理表达式，形成自定义数据结构  
– simplify()：求值、简化  
– derivative()：求导数

# 待求解问题描述与数学模型

输入数据：一个符合规范和特点条件的表达式，再次输入一个符合规范的求值或者求导命令，系统输出求值结果或者求导结果；

约束条件如实验要求所示；

输入的数学表达式用字符串存储存储起来，然后转化为数组类型；然后依次执行以下步骤：  
1. 判断表达式是否合法，按照+ \*位置判断表达式正误

2.将所有digit×digit类型的子串合并为一个数字，并且删除所有乘号：找到这样的乘号，乘号两边是数字，依据离乘号左右最近的非数字内分离数字相乘，其余的元素照原样复制

3. 预处理分割+号，将+两边的式子存入数组；

4.合并：表达式形式转为：digit （\*）character

5. 读入命令，判断是否是求导？是则求导

6. 判断是否是求值？是则求值

7.化简：合并同类项

8.输出新的表达式

# 算法与数据结构设计

## 设计思路

用字符串型接受输入，然后转化为数组型方便处理；

检查这个输入是否正确的方法很简单：因为有条件的约束，\*号两边必定不会出现+，\*，+两边必定不会出现\*，+，且+，\*不能出现在首尾，由于程序允许忽略乘号，所以其他的情况都包含在算法中了；

判断指令是否正确也很简单，按照指令的规律判断相应的字符是否合理即可；

先化简表达式：找到这样的乘号，乘号两边是数字，依据离乘号左右最近的非数字内分离数字相乘，其余的元素照原样复制；然后将+两边的式子存入数组；方便使用；表达式形式转为：数字 （乘以）字母；即提取出一个子式的所有数字相乘然后复制字符即可；

求导：找到欲求元素，统计次数，乘上该次数，其余照搬复制，并且计数求导元素使其次数少一；

求值：截取数字，找到字符相等的替换，并且数字相乘；

最后化简：把所有常数项合并；因为之前用+分割所以也很简单判断出常数项；

输出：常数项照搬，含有未知数的项目，统计未知数的次数，用^输出；

## 算法流程图

## C:\Users\lenovo\Desktop\无标题.png核心数据结构

本项目总共两个数据结构：  
第一个就是存储表达式和指令的字符串；

第二个是一个结构体，这个结构体是为了存储被拆分的子式；如下：  
class node {

int now;

String str;

node(int now, String str) {

this.now = now;

this.str = str;

}

}

记录了子式的内容和长度；

## 算法时间复杂度分析

使用大二期间算法与数据结构课程的相关知识，简要分析你设计的算法的时间复杂度。

可针对每一个模块分别分析（处理表达式、简化、求导）。

处理表达式：

preStr：表达式从头处理到尾，O(n)

Combination：表达式从头扫到尾，O(n)

Pre：表达式从头扫到尾，O(n)

简化：

Simplify：二层嵌套循环，O(n2)  
求导：

Derivative：表达式从头处理到尾，O(n)

总共：O(n2)

# 实验结果

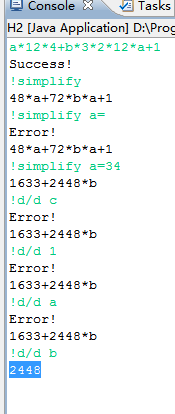
设计一组不少于20个覆盖各种约束条件的输入数据（多项式），在你开发的程序中分别输入，记录程序的输出结果，判断输出结果是否与期望一致。

## 输入/期望输出/实际输出

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 输入 | 期望输出 | 实际输出 | 程序运行是否正确 |
| 1 | a\*12\*4+b\*3\*2\*12\*a+1 | Success! | Success! | 是 |
| 2 | !simplify | 48\*a+72\*b\*a+1 | 48\*a+72\*b\*a+1 | 是 |
| 3 | !simplify a= | Error!  48\*a+72\*b\*a+1 | Error!  48\*a+72\*b\*a+1 | 是 |
| 4 | !simplify a=34 | 1633+2448\*b | 1633+2448\*b | 是 |
| 5 | !d/d c | Error!  1633+2448\*b | Error!  1633+2448\*b | 是 |
| 6 | !d/d 1 | Error!  1633+2448\*b | Error!  1633+2448\*b | 是 |
| 7 | !d/d a | Error!  1633+2448\*b | Error!  1633+2448\*b | 是 |
| 8 | !d/d b | 2448 | 2448 | 是 |
| 9 | 1+23\*4++5 | Error! | Error! | 是 |
| 10 | 12\*a\*b+4+ | Error! | Error! | 是 |
| 11 | 12\*a\*b+4 | Success! | Success! | 是 |
| 12 | !d/d | Error!  12\*a\*b+4 | Error!  12\*a\*b+4 | 是 |
| 13 | !simplify a=3 b=51 | 1840 | 1840 | 是 |
| 14 | aaa\*a+ba\*b+c | Success! | Success! | 是 |
| 15 | !d/d ca | Error!  1\*a^4+1\*b^2\*a+1\*c | Error!  1\*a^4+1\*b^2\*a+1\*c | 是 |
| 16 | !simplifyb=34 c=51 a=10 | 21611 | 21611 | 是 |
| 17 | !simp | Error!  21611 | Error!  21611 | 是 |
| 18 | !d/d a | Error!  21611 | Error!  21611 | 是 |
| 19 | 3aabd+cd4c5\*4\*a+ab | Success! | Success! | 是 |
| 20 | !d/d a | 6\*a\*b\*d+80\*c^2\*d+1\*b | 6\*a\*b\*d+80\*c^2\*d+1\*b | 是 |
| 21 | !simplify a=4 b=23 c=10 d=123 | 1051919 | 1051919 | 是 |

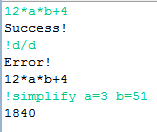
## 4.2程序运行界面截图

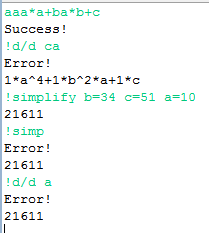
给出针对上表各表达式输入程序之后实际运行得到结果的界面截图。

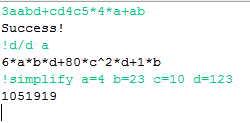












# 编程语言与开发环境

JavaDK版本

Eclipse IDE版本

# 结对编程

## 分组依据

杜冠宏善于处理细节和铺垫基础；比较有耐心，能够负责检错和改善的环节；

金策善于谋划大局，决定算法；思维活跃，能够大体上建立框架；

在整体的组对编程上，杜冠宏负责实现了基础的读入和检测功能，金策负责了求值和化简功能，然后二人协商后决定再加入一个可以省略乘号的功能，将程序前后又改了下；

## 角色切换与任务分工

该表格可自行增加更多的行。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 日期 | 工作时间 | “驾驶员” | “领航员” | 本段时间的任务 |
| 9.13 | 5h | 杜冠宏 | 金策 | 输入和检查错误 |
| 9.13 | 3h | 金策 | 杜冠宏 | 化简求值 |
| 9.14 | 3h | 金策 | 杜冠宏 | 省略乘号 |

## 工作照片





## 工作日志

由领航员负责记录，记录结对编程期间的遇到的问题、两人如何通过交流合作解决每个问题的。可增加表格的行。

|  |  |  |  |
| --- | --- | --- | --- |
| 日期/时间 | 问题描述 | 最终解决方法 | 两人如何通过交流找到解决方法 |
| 9.13 | 如何能高效的排除表达式错误 | 拆分+左右的式子 | 参考了下以前编写的程序 |
| 9.13 | 怎么分配求值时的代入和化简步骤 | 拆分为三个函数，化简，整合，输出 | 考虑到可以顺便把求导的问题一同解决，不如把功能拆开看 |
| 9.14 | 乘号该怎么省略 | 输入的时候就处理表达式 | 因为要重新组织一个数据结构，不然没法完成功能 |

# 计划与实际进度

|  |  |  |  |
| --- | --- | --- | --- |
| 任务编号 | 计划时间长度（分钟） | 实际耗费时间（分钟） | 提前或延期的原因分析 |
| 表达式和命令检错 | 120 | 200 | 因为在编程中屡次报错浪费了时间 |
| 求导和求值 | 150 | 120 | 负责人编程水平很高 |
| 加省略乘号功能 | 60 | 120 | 从输入开始就要修改，然后花了更多的时间 |

# 小结

两人编程确实效率比单人编程效率要高，质量也提高了，因为毕竟是两个人在思考，有错误能很快发现，有不对的地方有partner可以立马指出来，为后期处理和优化节省了不少的时间，比起一个人来说，两个人能够先统合各自的思路，然后折中出最优的方案，这个方案不仅仅能排除错误的思路，还能节省代码量；不合意的地方在于只有两个人有空的时候才能进行，一旦时间任务很赶而时间又错开了，就很难实现，不得不一个人编程；况且有时候会发现对方的代码和自己的代码可能接合的不是很好，这个也是需要时间去适应的；敏捷开发强调快速满足用户的需求，并且长期听取用户的需求，这样结对编程的效率会大大提升，比起单人来说，思路更广，更容易符合用户的需求，更容易按时交工；在尝试了此次结对编程后，今后的学习和工作中我们可能会考虑这样进行互动；