目录

[修改日志 2](#_Toc133649416)

[可视化规则引擎简介 2](#_Toc133649417)

[什么是规则引擎？ 2](#_Toc133649418)

[规则引擎适用场景 2](#_Toc133649419)

[规则引擎优势 3](#_Toc133649420)

[可视化规则引擎优势 3](#_Toc133649421)

[一、压缩包名称说明 5](#_Toc133649422)

[二、快速上手 6](#_Toc133649423)

[三、程序执行步骤 7](#_Toc133649424)

[四、公式说明 7](#_Toc133649425)

[4.1、注意事项 7](#_Toc133649426)

[4.2、计算符号顺序 8](#_Toc133649427)

[五、后台管理设计 8](#_Toc133649428)

[5.1、项目字典 8](#_Toc133649429)

[5.2、项目公式 8](#_Toc133649430)

[5.3、厂区信息 9](#_Toc133649431)

[5.4、厂区机械 9](#_Toc133649432)

[5.5、厂区工艺 10](#_Toc133649433)

[5.6、流程信息 10](#_Toc133649434)

[5.7、输入检测 11](#_Toc133649435)

[5.8、初始值 11](#_Toc133649436)

[六、技巧与注意事项 12](#_Toc133649437)

[6.1、流程图快捷键： 12](#_Toc133649438)

[6.2、流程设计注意事项 12](#_Toc133649439)

[七、API使用说明 13](#_Toc133649440)

[7.1、计算状态码 13](#_Toc133649441)

[7.2、计算公式 14](#_Toc133649442)

[7.3、获取流程工序 14](#_Toc133649443)

[7.4、更新缓存 16](#_Toc133649444)

[八、跳转至调试页面代码使用说明 16](#_Toc133649445)

[九、FlowWork部署 17](#_Toc133649446)

[9.1、appsettings.json配置 17](#_Toc133649447)

[9.2、注册windows服务 17](#_Toc133649448)

[9.3、注册Linux自动启动 18](#_Toc133649449)

[9.4、API说明 18](#_Toc133649450)

[9.5、FlowVision交流群 18](#_Toc133649451)

[附件EXCEL函数 19](#_Toc133649452)

[逻辑函数 19](#_Toc133649453)

[数学与三角函数 20](#_Toc133649454)

[文本函数 23](#_Toc133649455)

[日期与时间函数 24](#_Toc133649456)

[统计函数 25](#_Toc133649457)

[其他函数 28](#_Toc133649458)

## 修改日志

|  |  |  |  |
| --- | --- | --- | --- |
| 版本 | 日期 | 编辑人 | 内容说明 |
| v0.11 | 2022-12-01 | ToolGood(林知君) | 初稿 |
|  |  |  |  |
|  |  |  |  |

## 可视化规则引擎简介

### 什么是规则引擎？

规则引擎，全称为业务规则管理系统，英文名为BRMS(即Business Rule Management System)。**规则引擎的主要思想是将应用程序中的业务决策部分分离出来，并使用预定义的语义模块编写业务决策（业务规则），由用户或开发者在需要时进行配置、管理。**

在企业项目中，关键或核心部分总是业务逻辑或业务规则，也就是 CRUD，这些系统都有一个共同的特征是，某个模块中的一些或许多规则或策略总会发生变化，例如购物网站的顾客折扣、物流企业的运价计算等。随着这些变化而来的是大量的重复工作，如果系统没有足够的抽象，那么每当增加一种规则时，开发者需要在规则、回归测试、性能测试等方面的变化中编写代码。

**规则引擎实现了将业务决策从应用程序代码中分离出来，接收数据输入，解释业务规则，并根据业务规则做出业务决策。**规则引擎其实就是一个输入输出平台。

系统中引入规则引擎后，业务规则不再以程序代码的形式驻留在系统中，取而代之的是处理规则的规则引擎，业务规则存储在规则库中，完全独立于程序。业务人员可以像管理数据一样对业务规则进行管理，比如查询、添加、更新、统计、提交业务规则等。业务规则被加载到规则引擎中供应用系统调用。

### 规则引擎适用场景

1、商品价格计算

2、任务分配规则

贷后催收业务中的应用：<https://www.cnblogs.com/vivotech/p/16921060.html>

3、申请规则

信用卡申请：<https://www.cnblogs.com/gitBook/p/16937613.html>

保险产品准入规则：<https://www.cnblogs.com/gitBook/p/16940388.html>

### 规则引擎优势

1、业务规则与系统代码分离，避免升级的风险；

2、加强业务处理的透明度，实现业务规则的集中管理；

3、提高业务灵活性，在不重启服务的情况下可随时对业务规则进行扩展和维护；

4、规则引擎是相对独立的，业务分析人员可以参与编辑、维护系统的业务规则，减少对IT人员的依赖程度；

5、简化系统架构，优化应用，方便系统的整合；

6、减少了程序使用“硬编码“业务规则的成本和风险；

缺点：学习成本高，学习语法规则，还有交流成本高，业务人员使用中文术语，开发使用英文术语。

### 可视化规则引擎优势

**1、可视化编辑是一张思维导图，每个节点都能有下一个节点，并且节点上都有条件公式，这样就成了一个完整规则**。普通人都看懂各个节点的关系，学习成本低，对新人培训非常友好。程序还可以导出图片，方便与业务人员进行沟通交流。

图示

描述已自动生成

**2、复原规则引擎的计算过程**

当价格计算特别复杂，客服过来让开发人员来解释这件商品为何这么低的价格时，开发人员是很苦恼的，开发人员研究的是代码，又不研究价格，那只能进行调试，一步步调试，查到原因来跟客服沟通。如果这种情况过多，开发就完全成为客服的工具人了。

有些项目会使用计算引擎，这种情况下，调试复原计算逻辑中，老难了。其中让人头痛的是无效断点。价格计算逻辑特别复杂时，通常公式套另一个公式，或者一个公式套着三四个公式，又因加了缓存数据加快运行速度这个逻辑，我们查了一个公式，用了一个断点，结果跑到另一个公式的条件检测上去了。当我们又想查另一个公式时，结果发现值已缓存，跳不进去了，被迫得进行第二次调试。

所以**可视化规则引擎很重要！！！**

有了可视化规则引擎，查异常价格变得简单，过程步骤如下：

**a、查流程是否正确；**

图片包含 图示

描述已自动生成

红色标记了不符合条件的原因，点击红色字，下方可显示相关计算来源

表格

描述已自动生成

点击节点会显示匹配规则，如下图：

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

**b、查价格是否出错；**

文本, 应用程序, 电子邮件

描述已自动生成

点击蓝色字，会隐藏无关计算逻辑，如下图

图形用户界面, 文本, 应用程序

描述已自动生成

再次点击还原计算逻辑。

**c、查条件是否出错；**

点击左侧加粗字，会显示匹配规则

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

红色标记了不符合的条件

**d、查公式是否出错；**

如果前几步都未找到原因，查一下配置的公式是否出错。

**e、查输入值是否有问题。**

**3、倒推生产数量**

在制造业内，如纸箱制造，纸很容易损坏，要生产1000只，就要倒推工序，每个工序需要多少材料。而市面上的规则引擎都不能满足。

非制造业，可忽略此条。

**4、使用EXCEL函数**，降低财务人员学习成本。

如果是财务人员使用，将索引设置为EXCEL索引，从1开始。

## 一、压缩包名称说明

下载地址：https://pan.baidu.com/s/138R4d172-drb731GoTI6Vg?pwd=92i2

复制这段内容后打开百度网盘手机App，操作更方便哦

压缩包名称格式如下：

程序名称-操作系统-CPU类型-打包日期.zip

FlowVision：设计主程

FlowWork：部署程序

FlowWork-Consul：部署程序支持Consul服务注册

Windows：Windows操作系统

Linux：Linux操作系统

X64：cpu类型为x64，若无此参数，默认为x64

## 二、快速上手

1、使用管理员账号登录管理后台：

网址http://localhost:5000/Members/login

管理员账号：admin 密码：admin

开发账号：developer 密码：developer

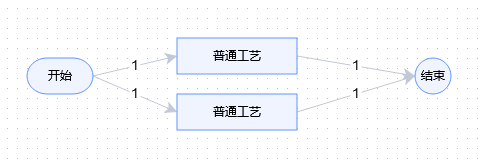
只读账号：reader密码：reader

2、在顶部【项目管理】->左侧菜单栏【流程设计】->【流程信息】页面点击【新增流程】

编码：Flow

名称：流程一

3、【流程信息】页面双击【流程一】



双击顶部的【普通工艺】，编辑

标签：数量<100总价

检测条件：数量<100

配置公式：

变量名称：总价

默认公式：数量\*10

双击底部的【普通工艺】，编辑

标签：数量>=100总价

检测条件（可不写）：数量>=100

配置公式：

变量名称：总价

默认公式：数量\*8

注：计算顺序，从左到右，从上到下。必须有【开始】【结束】，【项目列表】可取消【数量】必填。

4、在顶部【项目管理】->左侧菜单栏【流程设计】->【流程信息】页面点击【生成项目】

选择【生成项目文件并更新缓存】，生成完毕后关闭

5、进入前台【计算公式】

厂区：项目一

流程：流程一

参数：{"数量":"1000"}

公式：总价

点击【提交】

返回：{"code":1,"data":{"result":"8000"},"state":"SUCCESS","message":"SUCCESS","status":["END"]}{"code":1,"data":{"result":"8000"},"state":"SUCCESS","message":"SUCCESS","status":["END"]}

6、进入前台【流程调试】，**（订阅服务）**

厂区：项目一

流程：流程一

参数：{"数量":"1000"}

公式：总价

点击【提交】

图形用户界面

描述已自动生成

## 三、程序执行步骤

第一步：校验【厂区编码】、【流程编码】；

第二步：校验【数量】，注：【项目列表】可取消【数量】必填；

第三步：校验【输入值】，去单位字符，转化单位，系统会自动转长度、平方、立方、重量单位到指定单位；

第四步：校验【初始值】，可以替换【输入值】；

第五步：进入流程，流程从左到右，从上到下，流程结束后第一个状态码为END，如遇到执行节点，匹配后返回的第一个状态码为节点的状态码。开始节点可以预设变量，其他节点的变量名不能重复；

第六步：进行公式计算。

注：如果节点为工序节点，会先匹配节点的判断公式、入量公式，再匹配工序的判断公式，最后检测机器的判断公式。

注2：错误节点也可以有判断公式。

## 四、公式说明

支持四则运算、Excel公式,并支持自定义参数。

### 4.1、注意事项

**1、函数名不分大小写，自定义变量名区分大小写；**

2、数字转字符串后，**小数位数最长保留10位；**

3、字符串拼接使用&；

4、find为Excel公式，find(要查找的字符串,被查找的字符串[,开始位置])；

5、支持常量pi,e,true,false；

6、数值转bool，非零为真,零为假；

7、字符串转bool：0、FALSE、无、否、没有、假为FALSE，1、TRUE、有、是、真为TRUE。不区分大小写；

8、bool转数值：true为1，false为0。公式9\*true+100\*false 结果为9；

9、长度单位默认为m，支持长度单位有M、DM、CM、MM、KM，只支持英文字母，1DM=0.1；

10、平方单位默认为m2，支持长度单位有M2、DM2、CM2、MM2、KM2，只支持英文字母；

11、立方单位默认为m3，支持长度单位有M3、DM3、CM3、MM3、KM3、L、ML，只支持英文字母；

12、平方单位默认为kg，支持长度单位有G、KG、T，只支持英文字母；

13、中文符号自动转成英文符号：括号,方括号,逗号,引号,双引号；

14、财务数据、统计数据之中的算法，(注:本项目使用double类型，建议使用分为计价单位)；

15、EXCEL函数见附件。

### 4.2、计算符号顺序

|  |  |  |
| --- | --- | --- |
| 序列 | 名称 | 说明 |
| 1 | . (点) | 如：XX.has(“STEP”) |
| 2 | ( ) | 如：(1+1) |
| 3 | ! | 如：!(A=A) |
| 4 | % | 如：1% |
| 5 | \*、/、% | 乘除模 |
| 6 | +、-、& | 加减 （字符串）合并 |
| **7** | **>、>=、<、<=** | **大于，大于等于、小于、小于等于** |
| **8** | **=、==、===、!=、!==、<>** | **等于、不等于** |
| **9** | **&&、and** | **并且** |
| **10** | **||、or** | **或者** |
| 11 | A?B:C | 三元运算符 |

## 五、后台管理设计

### 5.1、项目字典

路径：后台->顶部【项目管理】->项目名称->【参考文档】->【项目字典】

项目字典记录着业务名称、单位、名称解释，不影响业务流程。

### 5.2、项目公式

路径：后台->顶部【项目管理】->项目名称->【参考文档】->【项目公式】

项目公式记录基础公式，当项目有多外流程时，可减少编辑量，影响业务流程。

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

选中代码替换，在流程设计编辑代码时，如果选中该公式名称，就会自动被替换成指定公式

### 5.3、厂区信息

路径：后台->顶部【项目管理】->项目名称->【厂区管理】->【厂区信息】

厂区信息必需有一条。

图形用户界面, 文本, 应用程序

描述已自动生成

### 5.4、厂区机械

路径：后台->顶部【项目管理】->项目名称->【厂区管理】->【厂区机械】

厂区机械可为空。与厂区绑定。

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

名称必填，流程图会绑定此名称，所以不要随意修改名称。

### 5.5、厂区工艺

路径：后台->顶部【项目管理】->项目名称->【厂区管理】->【厂区工艺】

厂区工艺可为空。与厂区绑定。

图形用户界面, 应用程序

描述已自动生成

编码必填，流程图会绑定此编码，所以不要随意修改编码。

### 5.6、流程信息

路径：后台->顶部【项目管理】->项目名称->【流程设计】->【流程信息】

流程信息必须要有一条。

图形用户界面, 文本, 应用程序

描述已自动生成

编码为接口调用的AppCode，不要随意修改。

### 5.7、输入检测

路径：后台->顶部【项目管理】->项目名称->【流程设计】->【输入检测】

输入检测可空，规范输入值。

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

如果是枚举类型，推荐写法：{"单黑","彩色"}.has(数码颜色)

输入值去除单位符号功能，如输入项为颜色，类型为数字，单位为色，传入{“颜色”:”1色”}，系统会自动去除色字，再转成1这个数字。如果转入{“颜色”:”1”} ，系统会正常记录1这个数字。

如果这个单位为长度、平方、立方、重量单位，系统会自动转化。如输入项为长主，单位为M，传入{“长”:“23cm”}，系统会自动转成0.23这个数字。

### 5.8、初始值

路径：后台->顶部【项目管理】->项目名称->【流程设计】->【初始值】

初始值可空。

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

## 六、技巧与注意事项

### 6.1、流程图快捷键：

Ctrl+z 还复

Ctrl+y 重做

Ctrl+a 全选

Delete 删除

Ctrl+x 剪切

Ctrl+c 复制

Ctrl+v 粘贴

Ctrl+h 粘贴+覆盖

Ctrl+s 保存

### 6.2、流程设计注意事项

1、每个流程必须包括开始与结束。

2、条件公式优先使用输入值、初始值，不要使用数量相关公式。

3、节点后续添加错误节点

图示

描述已自动生成

4、项目设计初期建议使用EXCEL导入。EXCEL模板与导入在【基础设置】->【项目列表】->【上传导入】。

图形用户界面, 应用程序

描述已自动生成

## 七、API使用说明

API全部使用POST请求方式

请求头如下：

Content-Type: application/x-www-form-urlencoded;

返回为JSON格式

### 7.1、计算状态码

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **接口：**/api/EvalStatus | | | | | |
| **入参：**POST | | | | | |
| 名称 | | | 类型 | 必填 | 说明 |
| appCode | | | string | 是 | 流程CODE |
| factoryCode | | | string | 是 | 厂区CODE |
| json | | | jsonString | 是 | 参数 |
|  | | 数量 | number | 是 | 数量 |
| previous | | | jsonString | 否 | 前一个流程的参数， |
| attachData | | | JsonString/String | 否 | 附带信息，用于js脚本计算 |
| machineInfos | | | jsonString | 否 | 机器信息，维修时可以暂停计算 |
|  | categoryCode | | string | 否 | 分类CODE |
|  | code | | string | 是 | 机器CODE |
|  | isStop | | bool | 是 | 是否停用 |
| projectCode | | | string | 否 | 项目CODE |
| **出参** | | | | | |
| code | | | number | 是 | 成功为1，失败为0 |
| data | | | list | 否 | 状态列表，格式如：["END","STEP2","STEP1"] |
| state | | | string | 是 | 成功SUCCESS，失败ERROR |
| message | | | string | 是 | 信息提示 |

### 7.2、计算公式

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **接口：**/api/EvalFormulas | | | | | |
| **入参：**POST | | | | | |
| 名称 | | | 类型 | 必填 | 说明 |
| appCode | | | string | 是 | 流程CODE |
| factoryCode | | | string | 是 | 厂区CODE |
| json | | | jsonString | 是 | 参数 |
|  | | 数量 | number | 是 | 数量 |
| previous | | | jsonString | 否 | 前一个流程的参数， |
| attachData | | | JsonString/String | 否 | 附带信息，用于js脚本计算 |
| machineInfos | | | jsonString | 否 | 机器信息，维修时可以暂停计算 |
|  | categoryCode | | string | 否 | 分类CODE |
|  | code | | string | 是 | 机器CODE |
|  | isStop | | bool | 是 | 是否停用 |
| formulas | | | JsonString/String | 是 | 要被计算的公式，符合以下几种 |
|  | | string |  |  | 支持字符串公式 |
|  | | {key:string} |  |  | 支持json对象 |
|  | | [{...},{...}] |  |  | 支持json数组 |
| projectCode | | | string | 否 | 项目CODE |
| **出参** | | | | | |
| code | | | number | 是 | 成功为1，失败为0 |
| data | | | json | 否 |  |
|  | | result | string | 否 | formulas为string格式，默认result返回 |
|  | | ... | string | 否 | 自定义名称 |
| state | | | string | 是 | 成功SUCCESS，失败ERROR |
| message | | | string | 是 | 信息提示 |
| status | | | list | 否 | 状态列表，格式如：["END","STEP2","STEP1"] |

### 7.3、获取流程工序

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **接口：**/api/EvalProcedureDetails | | | | | | | |
| **入参：**POST | | | | | | | |
| 名称 | | | | | 类型 | 必填 | 说明 |
| appCode | | | | | string | 是 | 流程CODE |
| factoryCode | | | | | string | 是 | 厂区CODE |
| json | | | | | jsonString | 是 | 参数 |
|  | | 数量 | | | number | 是 | 数量 |
| previous | | | | | jsonString | 否 | 前一个流程的参数， |
| attachData | | | | | JsonString/String | 否 | 附带信息，用于js脚本计算 |
| machineInfos | | | | | jsonString | 否 | 机器信息，维修时可以暂停计算 |
|  | categoryCode | | | | string | 否 | 分类CODE |
|  | code | | | | string | 是 | 机器CODE |
|  | isStop | | | | bool | 是 | 是否停用 |
| matches | | | | | list | 是 | 匹配公式 |
|  | | name | | | string | 是 | 类型名称 |
|  | | text | | | string | 否 | 单一匹配 |
|  | | match | | | list | 否 | 多种匹配 |
|  | |  | type | | string | 否 | 类型：start,end,in,regex |
|  | |  | text | | string | 是 | 匹配文本 |
| all | | | | | bool | 否 | 是否返回相关计算规则 |
| projectCode | | | | | string | 否 | 项目CODE |
| **出参** | | | | | | | |
| code | | | | | number | 是 | 成功为1，失败为0 |
| data | | | | | json | 否 |  |
|  | | procedures | | | list | 否 | 工序信息 |
|  | |  | id | | string | 是 |  |
|  | |  | type | | string | 是 |  |
|  | |  | procedureLabel | | string | 否 |  |
|  | |  | procedure | | itemObject | 否 |  |
|  | |  | machine | | itemObject | 否 |  |
|  | |  | inputNum | | number | 否 |  |
|  | |  | outputNum | | number | 否 |  |
|  | |  | exps | | dict | 否 |  |
|  | |  |  | key | string | 是 |  |
|  | |  |  | value | formulaObject | 是 |  |
|  | | formulaItems | | | formulaObject | 否 |  |
| state | | | | | string | 是 | 成功SUCCESS，失败ERROR |
| message | | | | | string | 是 | 信息提示 |

itemObject对象

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 类型 | 必填 | 说明 |
| categoryCode | string | 否 | 分类CODE |
| category | string | 否 | 分类名 |
| code | string | 是 | CODE |
| name | string | 是 | 名称 |

formulaObject对象

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 名称 | | 类型 | 必填 | 说明 |
| name | | string | 是 | 名称 |
| formulas | | list | 否 | 子公式 |
|  | text | string | 否 | 文件 |
|  | name | string | 否 | 名称 |
|  | value | string | 否 | 值 |
| value | | string | 是 | 值 |
| type | | string | 是 | 类型 |
| nodeId | | string | 是 | 节点ID |

### 7.4、更新缓存

接口：/api/UpdateFile

请求方式：GET、POST

## 八、跳转至调试页面代码使用说明

**页面地址：**

公式步骤：/Analysis/FlowFormulaDetailsResult

流程路径：/Analysis/FlowPathResult

流程调试：/Analysis/FlowDebugResult

流程工艺：/Analysis/FlowProcedureResult

**入参：**

| 名称 | 类型 | 必填 | 备注 |
| --- | --- | --- | --- |
| factoryCode | string | 是 | 厂区编号 |
| appCode | string | 是 | 流程编号 |
| json | string | 是 | 参数，【数量】必填 |
| attachData | string | 否 | 附加参数，用于脚本计算 |
| machineInfos | string | 否 | 机械状态：[{categoryCode:"分类编码",code:"机械编码",isStop:true},,,] |
| previous | string | 否 | 上个流程的参数 |
| formulas | string | 是  （流程路径:否） | 公式string，对象{name:名称,formula:公式} 或 集合[{name:名称,formula:公式},...] |
| projectCode | string | 否 | 项目Code |

**例：**

<html><body onload="javascript:document.forms[0].submit()">

<form action="/Analysis/FlowFormulaDetailsResult" method="post" style="display:none">

<input type="hidden" name="factoryCode" value="厂区编号" />

<input type="hidden" name="appCode" value="流程编号" />

<input type="hidden" name="json" value='{"数量":123,参数,,,}' />

<input type="hidden" name="attachData" value='' />

<input type="hidden" name="machineInfos" value='' />

<input type="hidden" name="formulas" value="公式" />

<input type="hidden" name="previous" value="" />

<button type="submit" >submit</button>

</form>

</body></html>

## 九、FlowWork部署

### 9.1、appsettings.json配置

{

  "Logging": {

    "LogLevel": {

      "Default": "Information",

      "Microsoft.AspNetCore": "Warning"

    }

  },

  "AllowedHosts": "\*",

  "FlowSetting": {

    "PorjectName": "项目一",

    "PorjectCode": "02048942-EACB-4B78-BCAE-0F39FDC72AE2",

    "license": ""

  },

  "Kestrel": {

    "EndPoints": {

      "Http": {

        "Url": "http://localhost:5123"

      }

    }

  }

}

PorjectName：项目名称

PorjectCode：项目code

licese：订阅注册码

Url：网址端口

### 9.2、注册windows服务

1、下载nssm

官网 http://www.nssm.cc/

下载地址 http://www.nssm.cc/download

2、搭建windows 服务

解压下载包， 将 nssm.exe 复制到 安装目录，如D:\ FlowWork

以管理员模式进入CMD，执行以下命令：

d:

cd D:\FlowWork

nssm install FlowWork D:\FlowWork\FlowWork.exe

net start FlowWork

### 9.3、注册Linux自动启动

略

### 9.4、API说明

请参考FlowVision的API，其中projectCode不传。

### 9.5、FlowVision交流群

QR 代码

描述已自动生成

# 附件EXCEL函数

EXCEL函数：逻辑函数、数学与三角函数、文本函数、统计函数、日期与时间函数

注：

1、函数名不分大小写,带方括号的参数可省略,示例的返回值,为近似值。

2、函数名带★，表示第一个参数可以前置，如(-1).ISTEXT()

3、函数名带▲，表示可以使用Excel索引从1开始。

### 逻辑函数

|  |  |  |
| --- | --- | --- |
| 函数名 | 说明 | 示例 |
| IF | if(测试条件,真值[,假值]) 执行真假值判断,根据逻辑计算的真假值,返回不同结果。 | if(1=1,1,2) >>1 |
| ifError | ifError(测试条件,真值[,假值]) 如果公式计算出错误则返回您指定的值；否则返回公式结果。 | ifError(1/0,1,2) >>1 |
| isError ★ | isError(值) 判断是否出错,返回 TRUE 或 FALSE  isError(值,替换值) 如果出错,返回替换值，否则返回原值 | isError(1) >>false |
| isNull ★ | isNull(值) 判断是否为空,返回 TRUE 或 FALSE  isNull(值,替换值) 如果为空,返回替换值，否则返回原值 | isNull(null) >>true |
| isNullOrError ★ | isNullOrError(值) 判断是否为空或错误,返回 TRUE 或 FALSE  isNullOrError(值,替换值) 如果为空或错误,返回替换值，否则返回原值 | isNullOrError(null) >>true |
| isNumber ★ | isNumber(值) 判断是否数值,返回 TRUE 或 FALSE | isNumber(1) >>true |
| isText ★ | isText(值) 判断是否文字,返回 TRUE 或 FALSE | isText('1') >>true |
| IsNonText ★ | IsNonText(值) 判断是否为非文字,返回 TRUE 或 FALSE | IsNonText('1') >>false |
| IsLogical ★ | IsLogical(值) 判断是否为逻辑值,返回 TRUE 或 FALSE | IsLogical('1') >>false |
| IsEven ★ | IsEven(值) 如果数值是偶数,返回 TRUE 或 FALSE | IsEven('1') >>false |
| IsOdd ★ | IsOdd(值) 如果数值是奇数,返回 TRUE 或 FALSE | IsOdd('1') >>true |
| AND | and(逻辑值1,...) 如果所有参数均为TRUE,则返回TRUE,如有错误先报错 | and(1,2=2) >>true |
| OR | or(逻辑值1,...) 如果任一参数为TRUE,则返回TRUE,如有错误先报错 | or(1,2=3) >>true |
| NOT | not(逻辑值) 对参数的逻辑值求反 | NOT(true()) >>false |
| TRUE | true() 返回逻辑值TRUE | true() >>true |
| FALSE | false() 返回逻辑值FALSE | false() >>false |

### 数学与三角函数

|  |  |  |  |
| --- | --- | --- | --- |
| 分类 | 函数名 | 说明 | 示例 |
| 基  础  数  学 | E | e() 返回 e 值 | E() >> |
| PI | pi() 返回 PI 值 | pi() >>3.141592654 |
| abs | abs(数值) 返回数值的绝对值 | abs(-1) >>1 |
| QUOTIENT | quotient(除数,被除数) 返回商的整数部分,该函数可用于舍掉商的小数部分。 | QUOTIENT(7,3) >>2 |
| mod | mod(除数,被除数) 返回两数相除的余数 | MOD(7,3) >>1 |
| SIGN | sign(数值) 返回数值的符号。当数值为正数时返回 1,为零时返回 0,为负数时返回 -1。 | SIGN(-9) >>-1 |
| SQRT | sqrt(数值) 返回正平方根 | SQRT(9) >>3 |
| TRUNC | trunc(数值) 将数值截尾取整 | TRUNC(9.222) >>9 |
| int ★ | int(数值) 将数值向下舍入到最接近的整数。 | int(9.222) >>9 |
| gcd | gcd(数值,...) 返回最大公约数 | GCD(3,5,7) >>1 |
| LCM | lcm(数值,...) 返回整数参数的最小公倍数 | LCM(3,5,7) >>105 |
| combin | combin(总数,排列数) 计算从给定数目的对象集合中提取若干对象的组合数 | combin(10,2) >>45 |
| PERMUT | permut(总数,排列数) 返回从给定数目的对象集合中选取的若干对象的排列数 | PERMUT(10,2) >>990 |
| FIXED | fixed(数值[,小数位数[,有无逗号分隔符]]) 将数值设置为具有固定小数位的文本格式 | FIXED(4567.89,1) >>4,567.9 |
| 三  角  函  数 | degrees | degrees(弧度) 将弧度转换为度 | degrees(pi()) >>180 |
| RADIANS | radians(度) 将度转换为弧度 | RADIANS(180) >>3.141592654 |
| cos | cos(弧度) 返回数值的余弦值 | cos(1) >>0.540302305868 |
| cosh | cosh(弧度) 返回数值的双曲余弦值 | cosh(1) >>1.54308063481 |
| SIN | sin(弧度) 返回给定角度的正弦值 | sin(1) >>0.84147098480 |
| SINH | sinh(弧度) 返回数值的双曲正弦值 | sinh(1) >>1.1752011936 |
| TAN | tan(弧度) 返回数值的正切值 | tan(1) >>1.55740772465 |
| TANH | tanh(弧度) 返回数值的双曲正切值 | tanh(1) >>0.761594155955 |
| acos | acos(数值) 返回数值的反余弦值 | acos(0.5) >>1.04719755119 |
| acosh | acosh(数值) 返回数值的反双曲余弦值 | acosh(1.5) >>0.962423650119 |
| asin | asin(数值) 返回数值的反正弦值 | asin(0.5) >>0.523598775598 |
| asinh | asinh(数值) 返回数值的反双曲正弦值。 | asinh(1.5) >>1.1947632172 |
| atan | atan(数值) 返回数值的反正切值 | atan(1) >>0.785398163397 |
| atanh | atanh(数值) 返回参数的反双曲正切值 | atanh(1) >>0.549306144334 |
| atan2 | atan2(数值,数值) 从X和Y坐标返回反正切 | atan2(1,2) >>1.10714871779 |
| 四  舍  五  入 | ROUND | round(数值,小数位数) 返回某个数值按指定位数取整后的数值。 | ROUND(4.333,2) >>4.33 |
| roundDown | roundDown(数值,小数位数) 靠近零值,向下（绝对值减小的方向）舍入数值。 | roundDown(4.333,2) >>4.33 |
| roundUp | roundUp(数值,小数位数) 远离零值,向上（绝对值增长的方向）舍入数值。 | roundUp(4.333,2) >>4.34 |
| CEILING | ceiling(数值,舍入基数) 向上舍入（沿绝对值增大的方向）为最接近的 舍入基数 的倍数。 | CEILING(4.333,0.1) >>4.4 |
| floor | floor(数值,舍入基数) 向下舍入,使其等于最接近的 Significance 的倍数。 | FLOOR(4.333,0.1) >>4.3 |
| even | even(数值) 返回沿绝对值增大方向取整后最接近的偶数。 | EVEN(3) >>4 |
| ODD | odd(数值) 将数值向上舍入为最接近的奇型整数 | ODD(3.1) >>5 |
| MROUND | mround(数值,舍入基数) 返回一个舍入到所需倍数的数值 | MROUND(13,5) >>15 |
| 随  机  数 | RAND | rand() 返回 0 到 1 之间的随机数 | RAND() >>0.2 |
| randBetween | randBetween(最小整数,最大整数) 返回大于等于指定的最小值,小于指定最大值之间的一个随机整数。 | randBetween(2,44) >>9 |
| 幂  /  对  数  /  阶  乘 | fact | fact(数值) 返回数的阶乘,一个数的阶乘等于 1\*2\*3\*…\* 该数。 | FACT(3) >>6 |
| factdouble | factDouble(数值) 返回数值的双倍阶乘 | factDouble(10) >>3840 |
| POWER | power(数值,幂) 返回数的乘幂结果 | POWER(10,2) >>100 |
| exp | exp(幂) 返回e的指定数乘幂 | exp(2) >>7.389056099 |
| ln | ln(数值) 返回数值的自然对数 | LN(4) >>1.386294361 |
| log | log(数值[,底数]) 返回数值的常用对数,如省略底数,默认为10 | LOG(100,10) >>2 |
| LOG10 | log10(数值) 返回数值的10对数 | LOG10(100) >>2 |
| MULTINOMIAL | multinomial(数值,...) 返回参数和的阶乘与各参数阶乘乘积的比值 | MULTINOMIAL(1,2,3) >>60 |
| PRODUCT | product(数值,...) 将所有以参数形式给出的数值相乘,并返回乘积值。 | PRODUCT(1,2,3,4) >>24 |
| SqrtPi | SqrtPi(数值) 返回某数与 PI 的乘积的平方根 | SqrtPi(3) >>3.069980124 |
| SUMSQ | sumQq(数值,...) 返回参数的平方和 | SUMSQ(1,2) >>5 |

### 文本函数

|  |  |  |
| --- | --- | --- |
| 函数名 | 说明 | 示例 |
| ASC | asc(字符串) 将字符串内的全角英文字母更改为半角字符 | asc('ａｂｃＡＢＣ１２３') >>abcABC123 |
| JIS WIDECHAR ★ | jis(字符串) 将字符串中的半角英文字符更改为全角字符 | jis('abcABC123') >>ａｂｃＡＢＣ１２３ |
| CHAR | CHAR(数值) 返回由代码数值指定的字符 | char(49) >>1 |
| CLEAN | clean(字符串) 删除文本中所有打印不出的字符 | clean('\r112\t') >>112 |
| CODE | code(字符串) 返回文本字符串中第一个字符的数值代码 | CODE("1") >>49 |
| CONCATENATE | concatenate(字符串1,...) 将若干文本项合并到一个文本项中 | CONCATENATE('tt','11') >>tt11 |
| EXACT | exact(字符串1,字符串2) 检查两个文本值是否完全相同 | EXACT("11","22") >>false |
| FIND ★▲ | find(要查找的字符串,被查找的字符串[,开始位置]) 在一文本值内查找另一文本值（区分大小写） | FIND("11","12221122") >>5 |
| LEFT ★ | left(字符串[,字符个数]) 返回文本值最左边的字符 | LEFT('123222',3) >>123 |
| LEN ★ | len(字符串) 返回文本字符串中的字符个数 | LEN('123222') >>6 |
| MID ★ ▲ | mid(字符串,开始位置,字符个数) 从文本字符串中的指定位置起返回特定个数的字符 | MID('ABCDEF',2,3) >>BCD |
| PROPER ★ | proper(字符串) 将文本值中每一个单词的首字母设置为大写 | PROPER('abc abc') >>Abc Abc |
| REPLACE ★ ▲ | replace(原字符串,开始位置,字符个数,新字符串) replace(原字符串,要替换的字符串, 新字符串) 替换文本内的字符 | REPLACE("abccd",2,3,"2") >>a2d REPLACE("abccd","bc","2") >>a2cd |
| REPT ★ | rept(字符串,重复次数) 按给定次数重复文本 | REPT("q",3) >>qqq |
| RIGHT ★ | right(字符串[,字符个数]) 返回文本值最右边的字符 | RIGHT("123q",3) >>23q |
| RMB ★ | RMB(数值) 将数值转换为大写数值文本 | RMB(12.3) >>壹拾贰元叁角 |
| SEARCH ★ ▲ | search(要找的字符串,被查找的字符串[,开始位置]) 在一文本值中查找另一文本值（不区分大小写） | SEARCH("aa","abbAaddd") >>4 |
| SUBSTITUTE ★ | substitute(字符串,原字符串,新字符串[,替换序号]) 在文本字符串中以新文本替换旧文本 | SUBSTITUTE("ababcc","ab","12") >>1212cc |
| T ★ | t(数值) 将参数转换为文本 | T('123') >>123 |
| TEXT ★ | text(数值,数值格式) 设置数值的格式并将数值转换为文本 | TEXT(123,"0.00") >>123.00 |
| TRIM ★ | trim(字符串) 删除文本中的空格 | TRIM(" 123 123 ") >>123 123 |
| LOWER ★ TOLOWER ★ | lower(字符串) tolower(字符串) 将文本转换为小写形式 | LOWER('ABC') >>abc |
| UPPER ★ TOUPPER ★ | upper(字符串) toupper(字符串) 将文本转换为大写形式 | UPPER("abc") >>ABC |
| VALUE ★ | value(字符串) 将文本参数转换为数值 | VALUE("123") >>123 |

### 日期与时间函数

|  |  |  |
| --- | --- | --- |
| 函数名 | 说明 | 示例 |
| NOW | now() 返回当前日期和时间的序列号 | NOW() >>2017-01-07 11:00:00 |
| TODAY | today() 返回今天日期的序列号 | TODAY() >>2017-01-07 |
| DateValue ★ | DateValue(字符串) 将文本格式的日期转换为序列号 | DateValue("2017-01-02") >>2017-01-02 |
| TimeValue ★ | TimeValue(字符串) 将文本格式的时间转换为序列号 | TimeValue("12:12:12") >>12:12:12 |
| DATE ★ | date(年,月,日[,时[,分[,秒]]]) 返回特定日期的序列号 | DATE(2016,1,1) >>2016-01-01 |
| TIME ★ | time(时,分,秒) 返回特定时间的序列号 | TIME(12,13,14) >>12:13:14 |
| YEAR ★ | year(日期) 将序列号转换为年 | YEAR(NOW()) >>2017 |
| MONTH ★ | month(日期) 将序列号转换为月 | MONTH(NOW()) >>1 |
| DAY ★ | day(日期) 将序列号转换为月份中的日 | DAY(NOW()) >>7 |
| HOUR ★ | hour(日期) 将序列号转换为小时 | HOUR(NOW()) >>11 |
| MINUTE ★ | minute(日期) 将序列号转换为分钟 | MINUTE(NOW()) >>12 |
| SECOND ★ | second(日期) 将序列号转换为秒 | SECOND(NOW()) >>34 |
| WEEKDAY ★ | second(日期) 将序列号转换为星期几 | WEEKDAY(date(2017,1,7)) >>7 |
| dateDIF | dateDif(开始日期,结束日期,类型Y/M/D/YD/MD/YM) 返回两个日期之间的相隔天数 | dateDIF("1975-1-30","2017-1-7","Y") >>41 |
| DAYS360 | days360(开始日期,结束日期[,选项0/1]) 以一年 360 天为基准计算两个日期间的天数 | DAYS360('1975-1-30','2017-1-7') >>15097 |
| EDATE | eDate(开始日期,月数) 返回用于表示开始日期之前或之后月数的日期的序列号 | EDATE("2012-1-31",32) >>2014-09-30 |
| EOMONTH | eoMonth(开始日期,月数) 返回指定月数之前或之后的月份的最后一天的序列号 | EOMONTH("2012-2-1",32) >>2014-10-31 |
| netWorkdays | netWorkdays(开始日期,结束日期[,假日]) 返回两个日期之间的全部工作日数 | netWorkdays("2012-1-1","2013-1-1") >>262 |
| workDay | workday(开始日期,天数[,假日]) 返回指定的若干个工作日之前或之后的日期的序列号 | workDay("2012-1-2",145) >>2012-07-23 |
| WEEKNUM | weekNum(日期[,类型：1/2]) 将序列号转换为一年中相应的周数 | WEEKNUM("2016-1-3") >>2 |

### 统计函数

|  |  |  |
| --- | --- | --- |
| 函数名 | 说明 | 示例 |
| MAX | max(数值,...) 返回参数列表中的最大值 | max(1,2,3,4,2,2,1,4) >>4 |
| MEDIAN | median(数值,...) 返回给定数值的中值 | MEDIAN(1,2,3,4,2,2,1,4) >>2 |
| MIN | min(数值,...) 返回参数列表中的最小值 | MIN(1,2,3,4,2,2,1,4) >>1 |
| QUARTILE | quartile(数值,四分位：0-4) 返回数据集的四分位数 | QUARTILE({1,2,3,4,2,2,1,4},0) >>1 |
| MODE | mode(数值,...) 返回在数组中出现频率最多的数值 | MODE(1,2,3,4,2,2,1,4) >>2 |
| LARGE ▲ | large(数组,K) 返回数据集中第 k 个最大值 | LARGE({1,2,3,4,2,2,1,4},3) >>3 |
| SMALL ▲ | small(数值,K) 返回数据集中第 k 个最小值 | SMALL({1,2,3,4,2,2,1,4},3) >>2 |
| PERCENTILE | percentile(数值,K) 返回区域中的第 k 个百分位值 | PERCENTILE({1,2,3,4,2,2,1,4},0.4) >>2 |
| PERCENTRANK | percentRank(数值,K) 返回数据集中值的百分比排位 | PERCENTRANK({1,2,3,4,2,2,1,4},3) >>0.714 |
| AVERAGE | average(数值,...) 返回参数的平均值 | AVERAGE(1,2,3,4,2,2,1,4) >>2.375 |
| averageIf | averageIf({数值,...},条件[,{值1,...}]) 返回参数的平均值 | averageIf({1,2,3,4,2,2,1,4},'>1') >>2.833333333 |
| GEOMEAN | geoMean(数值,...) 返回正数数组或区域的几何平均值 | GEOMEAN(1,2,3,4) >>2.213363839 |
| HARMEAN | harMean(数值,...) 返回数据集合的调和平均值 | HARMEAN(1,2,3,4) >>1.92 |
| COUNT | count(数值,...) 计算参数列表中数值的个数 | COUNT(1,2,3,4,2,2,1,4) >>8 |
| countIf | countIf({数值,...},条件[,{值1,...}]) 计算参数列表中数值的个数 | countIf({1,2,3,4,2,2,1,4},'>1') >>6 |
| SUM | sum(数值,...) 返回所有数值之和。 | SUM(1,2,3,4) >>10 |
| sumIf | sumIf({数值,...},条件[,{值1,...}]) 返回所有数值之和 | sumIf({1,2,3,4,2,2,1,4},'>1') >>17 |
| AVEDEV | aveDev(数值,...) 返回数据点与其平均值的绝对偏差的平均值 | AVEDEV(1,2,3,4,2,2,1,4) >>0.96875 |
| STDEV | stDev(数值,...) 基于样本估算标准偏差 | STDEV(1,2,3,4,2,2,1,4) >>1.1877349391654208 |
| STDEVP | stDevp(数值,...) 计算基于整个样本总体的标准偏差 | STDEVP(1,2,3,4,2,2,1,4) >>1.1110243021644486 |
| DEVSQ | devSq(数值,...) 返回偏差的平方和 | DEVSQ(1,2,3,4,2,2,1,4) >>9.875 |
| VAR | var(数值,...) 基于样本估算方差 | VAR(1,2,3,4,2,2,1,4) >>1.4107142857142858 |
| VARP | varp(数值,...) 基于整个样本总体计算方差 | VARP(1,2,3,4,2,2,1,4) >>1.234375 |
| normDist | normDist(数值,算术平均值,标准偏差,返回类型：0/1) 返回正态累积分布 | normDist(3,8,4,1) >>0.105649774 |
| normInv | normInv(分布概率,算术平均值,标准偏差) 返回反正态累积分布 | normInv(0.8,8,3) >>10.5248637 |
| NormSDist | normSDist(数值) 返回标准正态累积分布函数,该分布的平均值为 0,标准偏差为 1。 | NORMSDist(1) >>0.841344746 |
| normSInv | normSInv(数值) 返回反标准正态累积分布 | normSInv(0.3) >>-0.524400513 |
| BetaDist | BetaDist(数值,分布参数α,分布参数β) 返回 Beta 累积分布函数 | BetaDist(0.5,11,22) >>0.97494877 |
| BetaInv | BetaInv(数值,分布参数α,分布参数β) 返回指定 Beta 分布的累积分布函数的反函数 | BetaInv(0.5,23,45) >>0.336640759 |
| binomDist | binomDist(试验成功次数,试验次数,成功概率,返回类型：0/1) 返回一元二项式分布概率 | binomDist(12,45,0.5,0) >>0.000817409 |
| exponDist | exponDist(函数值,参数值,返回类型：0/1) 返回指数分布 | exponDist(3,1,0) >>0.049787068 |
| FDist | fDist(数值X,分子自由度,分母自由度) 返回 F 概率分布 | FDist(0.4,2,3) >>0.701465776 |
| FInv | fInv(分布概率,分子自由度,分母自由度) 返回 F 概率分布的反函数 | FInv(0.7,2,3) >>0.402651432 |
| FISHER | fisher(数值) 返回点 x 的 Fisher 变换。该变换生成一个正态分布而非偏斜的函数 | FISHER(0.68) >>0.8291140383 |
| fisherInv | fisherInv(数值) 返回 Fisher 变换的反函数值。 | fisherInv(0.6) >>0.537049567 |
| gammaDist | gammaDist(数值,分布参数α,分布参数β,返回类型：0/1) 返回 γ 分布 | gammaDist(0.5,3,4,0) >>0.001723627 |
| gammaInv | gammaInv(分布概率,分布参数α,分布参数β) 返回 γ 累积分布函数的反函数 | gammaInv(0.2,3,4) >>6.140176811 |
| GAMMALN | gammaLn(数值) 返回 γ 累积分布函数的反函数 | GAMMALN(4) >>1.791759469 |
| hypgeomDist | hypgeomDist(样本成功次数,样本容量,样本总体成功次数,样本总体容量) 返回超几何分布 | hypgeomDist(23,45,45,100) >>0.08715016 |
| logInv | logInv(分布概率,算法平均数,标准偏差) 返回 x 的对数累积分布函数的反函数 | logInv(0.1,45,33) >>15.01122624 |
| LognormDist | lognormDist(数值,算法平均数,标准偏差) 返回反对数正态分布 | lognormDist(15,23,45) >>0.326019201 |
| negbinomDist | negbinomDist(失败次数,成功极限次数,成功概率) 返回负二项式分布 | negbinomDist(23,45,0.7) >>0.053463314 |
| POISSON | poisson(数值,算法平均数,返回类型：0/1) 返回 Poisson 分布 | POISSON(23,23,0) >>0.082884384 |
| TDist | tDist(数值,自由度,返回类型：1/2) 返回学生的 t 分布 | TDist(1.2,24,1) >>0.120925677 |
| TInv | TInv(分布概率,自由度) 返回学生的 t 分布的反分布 | TInv(0.12,23) >>1.614756561 |
| WEIBULL | weibull(数值,分布参数α,分布参数β,返回类型：0/1) 返回 Weibull 分布 | WEIBULL(1,2,3,1) >>0.105160683 |

### 其他函数

|  |  |  |
| --- | --- | --- |
| 函数名 | 说明 | 示例 |
| RegexRepalce ★ | RegexRepalce(文本,匹配文本,替换文本) 匹配替换字符串。 |  |
| IsRegex ★ IsMatch ★ | IsRegex(文本,匹配文本) IsMatch(文本,匹配文本) 判断是否匹配。 |  |
| TrimStart ★ LTrim ★ | TrimStart(文本) LTrim(文本) LTrim(文本[,字符集]) 消空字符串左边。 |  |
| TrimEnd ★ RTrim ★ | TrimEnd(文本) RTrim(文本) RTrim(文本,字符集) 消空字符串右边。 |  |
| IndexOf ★ ▲ | IndexOf(文本,查找文本[,开始位置[,索引]]) 查找字符串位置。 |  |
| LastIndexOf ★ ▲ | LastIndexOf(文本,查找文本[,开始位置[,索引]]) 查找字符串位置。 |  |
| Split ★ | Split(文本,分隔符) 分成数组 |  |
| Join ★ | Join(文本1,文本2....) 合并字符串。 |  |
| Substring ★ ▲ | Substring(文本,位置) Substring(文本,位置,数量) 切割字符串。 |  |
| StartsWith ★ | StartsWith(文本,开始文本[,是否忽略大小写:1/0]) 确定此字符串实例的开头是否与指定的字符串匹配。 |  |
| EndsWith ★ | EndsWith(文本,开始文本[,是否忽略大小写:1/0]) 确定使用指定的比较选项进行比较时此字符串实例的结尾是否与指定的字符串匹配。 |  |
| IsNullOrEmpty ★ | IsNullOrEmpty(文本) 指示指定的字符串是 null 还是 空字符串。 |  |
| IsNullOrWhiteSpace ★ | IsNullOrWhiteSpace(文本) 指示指定的字符串是 null、空还是仅由空白字符组成。 |  |
| RemoveStart ★ | RemoveStart(文本,左边文本[,忽略大小写]) 匹配左边，成功则去除左边字符串。 |  |
| RemoveEnd ★ | RemoveEnd(文本,右边文本[,忽略大小写]) 匹配右边，成功则去除右边字符串。 |  |
| Json ★ | json(文本) 动态json查询。 |  |
| Has | {1,2,3}.has(1) 判断集合是存在1。初始值请使用这个方法，不要使用 in 。 |  |
| In | 1.in({1,2,3}) 判断1是否存在集合中。 |  |
| PARAM | PARAM(变量名)  获取变量 |  |

窗体底端