科研开源软件创意大赛

算法精灵用户手册

参赛团队/个人名称

2019年7月15日

目录

[一．安装部署手册 3](#_Toc8891504)

[1.1 环境配置 3](#_Toc8891505)

[1.2 安装包使用 3](#_Toc8891506)

[1.2.1 注意事项 3](#_Toc8891507)

[1.2.2 安装详解 3](#_Toc8891508)

[二．操作说明 5](#_Toc8891509)

[三．系统及数据维护 11](#_Toc8891510)

[四．附录 11](#_Toc8891511)

# 一．安装部署手册

## 环境配置

支持的平台

----------

算法精灵主要是使用脚本语言 Python 开发，Python 支持的平台本产

品也全部都支持。

下列操作系统已经进行过测试：

- Windows XP, 2000, 2003

- Linux

## 安装包使用

### 安装详解

1. **Windows系列平台**

**安装包**

如果你没有安装包，可以在

<https://github.com/dashingsoft/algorithm-elf/releases>

选择 Windows 平台的最新安装包下载，一般安装包的名字如下。

[algorithm-elf-1.2.2.exe](https://github.com/dashingsoft/algorithm-elf/releases/download/v1.2.2/algorithm-elf-1.2.2.exe)

其中 X.Y.M.N 是版本号。

**安装**

直接双击安装包 [algorithm-elf-1.2.2.exe](https://github.com/dashingsoft/algorithm-elf/releases/download/v1.2.2/algorithm-elf-1.2.2.exe)，按照提示完成安装。

安装完成会有桌面图标和开始菜单程序组：算法精灵。双击桌面图标

算法精灵即可启动。

**卸载**

直接双击安装包，如果已经安装过算法精灵，那么会提示修复还是卸

载，选择卸载，按照提示完成即可卸载。

或者在开始菜单中，找到算法精灵程序组，点击里面的卸载也可以卸

载算法精灵。

1. **Linux系列平台**

**安装包**

如果你没有安装包，可以在

<https://github.com/dashingsoft/algorithm-elf/releases>

选择 Linux 平台下的对应的安装包

[algorithm-elf-1.2.2-1.i686-linux.rpm](https://github.com/dashingsoft/algorithm-elf/releases/download/v1.2.2/algorithm-elf-1.2.2-1.i686-linux.rpm)

[algorithm-elf-1.2.2.tar.gz](https://github.com/dashingsoft/algorithm-elf/archive/v1.2.2.tar.gz)

**安装**

安装算法精灵使用命令：

$ rpm -ivh algorithm-elf-1.2.2-1.i686-linux.rpm

如果权限不够的话，请切换到超级用户执行安装。

安装默认目录为 /usr/algorithm-elf

安装完成之后在终端执行下列命令启动算法精灵

$ cd /usr/algorithm-elf

$ ./algorithm-elf.bin

**卸载**

执行下面的命令即可卸载

$ rpm -e algorithm-elf

1. **其他安装方式**

如果你使用 tar 包安装，那么运行下列命令：

$ tar xzf algorithm-elf-1.2.2.tar.gz

在当前目录下会创建一个 algorithm-elf 的子目录，运行下面的命令

启动算法精灵：

$ cd algorithm-elf

$ ./algorithm-elf.bin

# 二．操作说明

**Chapter 1. 概述**

算法精灵™是一款辅助学生理解算法实现过程的教学软件。它支持的主要功能包括：

经典算法过程的动画演示；

使用类 Pascal 语言编写算法，然后动画演示算法执行过程；

使用图形化的方式设置算法的传入参数，譬如二叉树、需要排序的队列等；

下面我们是两个典型的使用算法精灵™的场景：

老师在课堂上讲解算法

张教授正在给学生讲解快速排序算法，他启动算法精灵™，打开快速排序算法，输入一组需要排序的数据，点击算法演示，算法精灵™使用动画的方式显示出这组数据的排序过程：每一个排序步骤数据的比较和交换等等。

学生在设计自己的算法程序

王小丫是大二学生，正在完成数据结构的作业：编写一个二叉树先序遍历的算法。她启动算法精灵™，使用类 Pascal 语言输入二叉树先序遍历算法，然后从算法精灵™数据池中选择一颗二叉树，使用算法精灵™的单步演示模式，算法精灵™会一步步的显示二叉树的结点被访问的顺序。如果算法存在问题，算法精灵™会提示正确的信息或者使用图片显示出来，譬如访问的顺序不正确等。

**Chapter 2. 安装和启动**

你可以从 http://zh-cn.dashingsoft.com 下载最新的安装包， 算法精灵目前可以运行于 32 位 Windows 平台和 Linux 兼容的平台：

Windows 平台

在 Windows 平台下的安装包是一个可执行文件: algorithm-elf-1.2.1.exe ，直接双击安装包，按照提示完成安装；安装成功完成会有算法精灵™的开始菜单程序组；

点击开始菜单 -> Algorithm-Elf 中的 algorithm-elf 即可启动算法精灵™。

Linux 平台

Linux 平台下使用 RPM 格式的安装包：algorithm-elf-1.2.1-1.i686-linux.rpm，使用下面的命令安装算法精灵™：

$ rpm -ivh algorithm-elf-1.2.1-1.i686-linux.rpm

Note

如果权限不够的话，请切换到超级用户执行安装。

安装默认目录为 /usr/algorithm-elf，安装完成之后在终端执行下列命令启动算法精灵™

$ cd /usr/algorithm-elf

$ ./algorithm-elf.bin

如果你使用 tar 包安装，首先下载 algorithm-elf.1.2.1.tar.gz，然后运行命令：

$ tar xzf algorithm-elf-1.2.1.tar.gz

在当前目录下会创建一个 algorithm-elf 的子目录，运行下面的命令启动算法精灵：

$ cd algorithm-elf

$ ./algorithm-elf.bin

Chapter 3. 使用示例算法

算法精灵™内嵌了一些经典算法，主要包括递归算法汉诺塔，二叉树遍历和线索化算法，快速排序、归并排序等。下面我们以汉诺塔为例说明如何使用这些算法

动画演示算法执行过程

点击菜单文件中的打开示例，选中汉诺塔

点击工具栏按钮演示，弹出算法演示窗口，下面的操作都是在算法演示窗口中执行

点击工具栏按钮启动，启动算法演示

点击工具栏按钮运行，等待算法演示完成

Tip

可以设定动画演示的速度，在工具栏右边的延迟时间中输入一个数值，表示算法每执行一步延迟的毫秒数，可以根据电脑的速度调整，使得演示过程能够直观清晰。

单步执行算法

在上面的演示中，启动算法之后，点击单步，而不是运行，进入单步执行模式；

点击单步，每点击一次，执行一行算法代码

在单步模式，点击运行，进入动画演示模式

在动画演示过程，点击暂停，进入单步模式

单步模式主要是为了查看数据和详细的了解算法执行过程

改变算法传入参数

算法精灵™允许任意改变算法的传入参数，假设我们要使用5个盘子来演示汉诺塔，在算法演示窗口，操作如下

点击窗口右边的算法参数，这时候会显示出算法使用的参数：n, x, y, z

选中n，点击原来的值3，输入新值: 5

选中x，在下面的属性列表中，设置尺寸为 5

对参数 y, z 进行和 x 相同的操作

点击应用

点击启动，重新开始算法演示，这时候你可以注意到新设置的参数已经生效

查看其他算法和汉诺塔类似。

**Chapter 4. 编写自己的算法**

使用算法精灵™编写自己的算法程序，对于数组、链表、队列、树、堆等数据结构，算法精灵™都可以动画演示算法的执行过程。

算法精灵™目前支持的算法语言是 Pascal 的一个子集，你可以使用 Pascal 语言来编写自己的算法，下面是编写一个新算法的操作步骤

编写算法

启动算法精灵™

点击工具栏按钮新建

在编辑区输入算法，

program locate\_sequence\_list;

type

sqlist: array [ 1..3 ] of integer;

var

va: sqlist;

k: integer;

index: integer;

function loc\_sqlist(var va: sqlist; x : integer) : integer;

var

i : integer;

begin

i := 1;

while i <= 3 and va[i] <> x do

i := i + 1;

if i > 3 then

result := 0

else

result := i

end;

begin

va[1] := 5;

va[2] := 8;

va[3] := 9;

k := 8;

index := loc\_sqlist(va, k);

end.

算法语言和标准 Pascal 类似，但是函数的返回值使用关键字 result 而不是函数名称

点击保存，选择要保存的路径，输入文件名称，点击确定

编译算法

点击编译，如果算法编译失败，那么右边的编译消息窗口会显示具出错的行号和原因，修改这些问题，直到提示编译成功

演示算法

算法编译成功之后才可以进行算法演示，演示方式和内置算法类似，点击演示即可

关于算法语言的细节，譬如支持的语法、可以使用的 Pascal 库函数请参考附录中算法语言。

Chapter 5. 使用数据池

在上例中，我们的算法使用的参数是固定的，如果在算法内部修改算法的参数需要重新编译之后才能演示，那么如何可以动态的修改算法参数呢？答案就是数据池。

算法精灵内置的算法大部分都使用了数据池，数据池的作用就是定义大量的常用数据，譬如队列、树、排序数组等，以供算法使用；如果算法使用数据池的数据，那么不需要修改算法，只要改变数据池数据，就可以改变算法的传入参数，这就解决了我们上面的问题。

打开汉诺塔算法，我们可以看到，那些使用数据池的变量在声明的时候前面都有一个 '@' 符号，事实上，这就是使用数据池的基本方法：使用 '@' 为前缀来声明变量。下面我们将上例的传入参数使用数据池来实现

启动算法精灵™

点击工具栏按钮打开，选择刚才保存的算法文件

在编辑区修改算法内容如下，

program locate\_sequence\_list;

type

sqlist: array of integer;

var

@va: sqlist;

@k: integer;

index: integer;

function loc\_sqlist(var va: sqlist; x : integer) : integer;

var

i : integer;

begin

i := Low(va);

while i <= High(va) and va[i] <> x do

i := i + 1;

if i > High(va) then

result := 0

else

result := i

end;

begin

index := loc\_sqlist(va, k);

end.

点击保存

点击编译

点击演示，进入算法演示窗口；下面的操作都是在算法演示窗口

点击工具栏按钮算法参数，可以看到下面列出两个参数: va, k

选择va

选择参数下面的属性类型的值为：数组

设置参数下面的属性尺寸的值为：5，指定数组的大小

点击数组的任何一个结点，可以修改对应元素的值

选择k

点击图形结点，输入新值: 15

点击应用，参数设置完毕

这时候可以使用设置的参数进行算法演示，演示完成之后可以修改参数，然后重新演示

Chapter 6. 设置断点和监控表达式

设置断点允许指定算法执行到某一行暂停演示，监控表达式允许设定当表达式的值发生变化之后，暂停演示，进入单步模式；这是算法精灵™的高级用法，用于发现算法中比较隐蔽的数据错误；

设置断点和监控表达式都是在进入算法演示的时候才可用，下面的操作都是在算法演示窗口内执行

设置断点

点击需要设置断点的行，然后按键b；或者直接点击代码行的行首

点击工具栏按钮断点，显示断点列表，这里中会看到新增加的断点

设置监控表达式

在代码窗口选中需要监控的表达式，然后按键w；或者在断点列表窗口的下方输入需要监控的表达式，点击按钮数据断点

清除断点或者监控表达式

点击工具栏按钮断点，显示出断点列表窗口，点击需要删除的项，选中的行首会有标识：\*

点击列表下面的按钮移除，删除选中的项

设置断点或者监控表达式之后，可以启动算法演示；在演示的过程中，遇到断点或者监控表达式的值发生了变化，就暂停演示，进入单步模式；这时候可以查看执行堆栈、变量数据等，以发现问题所在。

# 三．附录

Appendix A. 算法语言

Table of Contents

A.1. 算法文件结构

A.2. 数据类型

A.3. 控制语句和表达式

A.4. 函数和过程

A.5. 扩展功能：数据池

A.6. 系统函数库

算法语言基本和 Pascal 类似，但是目前版本不支持 with语句，不支持 子范围类型（Subrange），并且库函数也只实现了部分。

A.1. 算法文件结构

一个典型的算法文件结构如下：

program 算法名称;

const

常量定义; { 可选 }

type

类型定义; { 可选 }

var

变量声明; { 可选 }

函数或者过程定义; { 函数或者过程定义必须在声明的最后部分 }

begin { 算法入口 }

赋值和控制语句序列

end.

A.2. 数据类型

简单数据类型

Integer, Real, Char, Boolean, String

示例：

var

r: Real;

i: Integer;

c: Char;

b: Boolean;

复合数据类型

Set, Enum, Record, Array

示例：

var

set1: set of Integer;

arr1: array of Integer; { 动态数组 }

arr2: array [0..10] of String;

r1: record of

name: String;

age: integer;

end;

e: (apple, pear, banana, orange, lemon);

自定义数据类型和指针数据类型

type

x = Integer;

y = x;

...

type

a = Array [1..10] of Integer;

b = record

x: Integer;

y: Char

end;

type

a = ^b;

b = record

x: Integer;

y: Char;

z: a

end;

var

pointertob: a;

pInt : ^Integer;

new(pointertob);

pointertob^.x := 10;

pointertob^.y := 'A';

pointertob^.z := nil;

A.3. 控制语句和表达式

控制语句. if, while, repeat, case, for

while a<>b do

writeln('Waiting');

if a > b then

writeln('Condition met')

else

writeln('Condition not met');

for i := 1 to 10 do

writeln('Iteration: ', i:1);

repeat

a := a + 1

until a = 10;

case i of

0: write('zero');

1: write('one');

2: write('two');

end;

表达式.

数值表达式： +, -, \*, /

1 + a, 2 \* 4, 3.5 / 6

布尔表达式: and, or, not, <, >, <=, >=, <>

a and b

a = b

字符串表达式: +

'abc' + 'efg'

集合运算: in

a in [1,2,3]

指针运算: ^ , @

p^ := s;

p := @i;

p := addr(x);

访问记录体: .

s.name := 'jondy';

赋值语句：:=

s := 2;

A.4. 函数和过程

函数和过程的定义示例

function next(k: integer): integer;

begin

result := k + 1

end;

procedure print(var j: integer);

begin

writeln('The total is: ', j);

j := next(j)

end;

Note

函数的返回值使用关键字 result 返回，而不是函数名称，这是和标准 Pascal 不兼容的地方之一。

Note

不支持在函数定义中再次定义其他函数。

A.5. 扩展功能：数据池

数据池功能是 Pascal 不支持的，访问数据池的方式是在变量声明的时候使用 '@' 作为变量前缀，那么该变量在执行过程中会被自动使用数据池中的同名变量来赋值。

var

@x: Integer;

begin

writeln('Get the value from datapool: ', x);

end.

A.6. 系统函数库

这里实现了常用了的 Pascal 的库函数，这里定义的函数可以直接在算法中使用

abs

function Abs(X);

addr

function Addr(X): Pointer;

chr

function Chr(X: Byte): Char;

copy

function Copy(S; Index, Count: Integer): string;

cos

function Cos(X: Extended): Extended;

delete

procedure Delete(var S: string; Index, Count:Integer);

dispose

procedure Dispose(var P: Pointer);

exp

function Exp(X: Real): Real;

frac

function Frac(X: Extended): Extended;

high

function High(X);

insert

procedure Insert(Source: string; var S: string; Index: Integer);

int

function Int\_(X: Extended): Extended;

length

function Length(S): Integer;

ln

function Ln(X: Real): Real;

low

function Low(X);

move

procedure Move(const Source; var Dest; Count: Integer);

new

procedure New(var P: Pointer);

odd

function Odd(X: Longint): Boolean;

ord

function Ord(X);

ptr

function Ptr(Address: Integer): Pointer;

pos

function Pos(Substr: string; S: string): Integer;

round

function Round(X: Extended): Int64;

setlength

procedure SetLength(var S; NewLength: Integer);

sin

function Sin(X: Extended): Extended;

sqr

function Sqr(X: Extended): Extended;

sqrt

function Sqrt(X: Extended): Extended;

trunc

function Trunc(X: Extended): Int64;

write

procedure Write([ var F: Text; ] P1 [, P2, ...,Pn ] );

writeln

procedure WriteLn([ var F: Text; ] P1 [, P2, ...,Pn ] );

Appendix B. 常见问题

算法精灵™使用过程中的常见问题的解答和说明。

B.1. 基本问题

B.1.1. 算法精灵™是什么？

算法精灵™是一个辅助教学的工具软件，可以帮助学生更好的掌握算法的实现原理和了解算法的实现过程。