高级程序设计训练（80L016Q）

实验#7

实验报告提交时间要求

- 见《高级程序设计训练 2016-2017-2 学期课程安排表—韩升老师课堂》

相关知识点

- 基本程序设计技术
- 多模块程序设计

需自学的相关技术

- 屏幕绘制函数库 EasyX
- 排序相关知识

实验目的

- 训练学生掌握使用 C 语言程序设计知识解决中大型问题的能力；
- 训练学生熟练掌握文件的操作方法；
- 学习并掌握屏幕绘制程序；
- 训练强化对程序设计开发过程和多模块程序设计思想的理解。

实验内容

- 程序设计：

  请根据《高级程序设计训练实验 7 程序设计说明书》中相关要求，实现设计说明书中描述的软件。

  【提示】：程序设计说明书见附录 1

- 程序功能扩展设计（选作）：

  1. 对于本系统是否存在设计不合理和有待改进的地方，如果让你来做设计，你可以做哪些改进。
  2. 如果将联动电梯改为三部电梯，那么电梯响应逻辑应该做怎样的调整，给出调整后的电梯响应逻辑图。对于设计文档其他部分，还要做哪些方面的调整，给出调整方案。

结果提交

- 实验完毕后需提交项目源代码、可执行程序，以电子文档形式用邮件提交给任课教师；

- 结合你在程序开发过程中实际体会，重新审视实验提供软件设计说明书，是否存在设计不合理的地方，如果存在，说明你的理由及建议如何调整，并以实验报告的形式提交给任课教师；
- 针对程序扩展部分的要求，对设计文档进行修改后的新的设计文档（如果不选做可以不提交）；
- 你须使用本课程所要求的命名规范对实验报告文档进行命名，详见《高级程序设计训练实验文档命名要求》；
- 由于课堂时间有限，本实验要求同学们充分利用课余时间进行文献资料的收集和学习。

成绩评定

- 程序采分点：
  - 程序是否独立调试通过并运行正常；
  - 源代码格式是否清晰，代码是否易于阅读；
  - **代码注释是否充分；**
  - 从程序工作中反映出的学生工作量；

附录 1：

高级程序设计训练实验 7

—电梯仿真程序设计说明书

学期：2016-2017-2 学期

编制人：韩升

编制日期：2017 年 1 月 20 日

修订记录

版本号	修订状态	修订记录	修订日期	修订人
V1.0	A	实验 7 设计说明书初稿完成	2017.3.10	韩升
V1.1	M	实验 7 设计说明书正式稿第一版，发给各位课题组老师	2017.4.9	韩升
V1.2	M	根据梁伯涵和吕亿林同学反馈的问题修改了系统参数设计部分、静默仿真函数及子函数的部分流程	2017.4.20	韩升
V1.3	M	根据李文浩同学反馈的问题修改了 4.2.1 节部分表述性错误，根据梁博涵同学反馈的问题修改了 4.2.4 节相关流程图错误	2017.4.26	韩升

【注】修订状态说明：A-增加、M-修改、D-删除

目录

1. 概述

1.1. 标识

文档名称：高级程序设计训练实验 7—电梯仿真程序设计规格说明书

文档编号：APT-Lab7-DESIGN

1.2. 范围

本文档适用于《高级程序设计训练》课程，为课程实验 7 环节的基础材料。文档阅读对象为课程授课教师及高级程序设计训练选课同学。

1.3. 文档说明

本文档主要描述了《电梯仿真》程序的实现逻辑，对软件的功能、架构、模块划分、数据存储、业务流程等进行了说明。本文档主要用于指导学生进行高级程序设计训练实验 7 环节的开发工作。

2. 软件需求说明

2.1. 软件功能需求

电梯仿真系统是高级程序设计训练课程中实验 7 环节程序设计题目，题目要求设计并实现一个针对多部电梯联动运行过程的仿真程序，并以动画的方式对电梯联动运行过程进行仿真演示。软件功能要求如下：

1. 用户可以以一定的方式输入电梯请求指令，程序根据电梯联动规则确定电梯的运行路径；
2. 软件应可以仿真多部电梯联动运行，有多个电梯同时服务，根据乘梯指令自动选择电梯提供服务；
3. 软件应可以以直观的方式将电梯服务响应用户请求后的服务过程展示出来；

4. 软件应可以以一定的格式将电梯的响应用户请求并驱动电梯运行的过程用文本描述出来，并能将其保存成文件；
5. 软件应该可以将保存好的电梯响应过程文件进行回放；
6. 软件应提供程序菜单供用户选择。

2.2. 软件环境需求

软件开发环境要求如下：

- 开发语言：C 语言；
- 开发环境：不限；
- 操作系统：Window7 及以上版本；

软件运行环境要求：

- CPU：双核 3.0GHz 及以上；
- 内存：大于 4GB；
- 操作系统：Windows7 及以上版本；

2.3. 软件性能需求

无

3. 系统概要设计

3.1. 系统架构

3. 1. 1. 系统约束设计

本系统实现的仿真软件目标是实现第 2 章中提出的各项功能需求，为了更好地开展设计工作，特对系统需要明确的一些问题做以下约定：

- 本系统仅实现两部电梯的联动仿真，不支持 3 部或 3 部以上的电梯联动仿真；

- 仅考虑电梯外部指令，用户对电梯的需求以“[用户当前所在楼层]，[用户目标楼层]，[指令提出的时间点]”的方式给出，忽略用户对电梯内外按钮的操作过程；
- 本仿真程序中，用户需求指令提出的时间点采用相对时间方式，时间点以整型数值表示，单位为分钟，数值越大表示用户提出请求的时间越晚，时间点数值必须大于等于零；
- 两部电梯均可以到达所有楼层，每次仿真开始时两部电梯的初始状态均为停靠在第 1 层，仿真时间轴从时间点 0 时刻开始仿真；

3.1.2. 系统架构

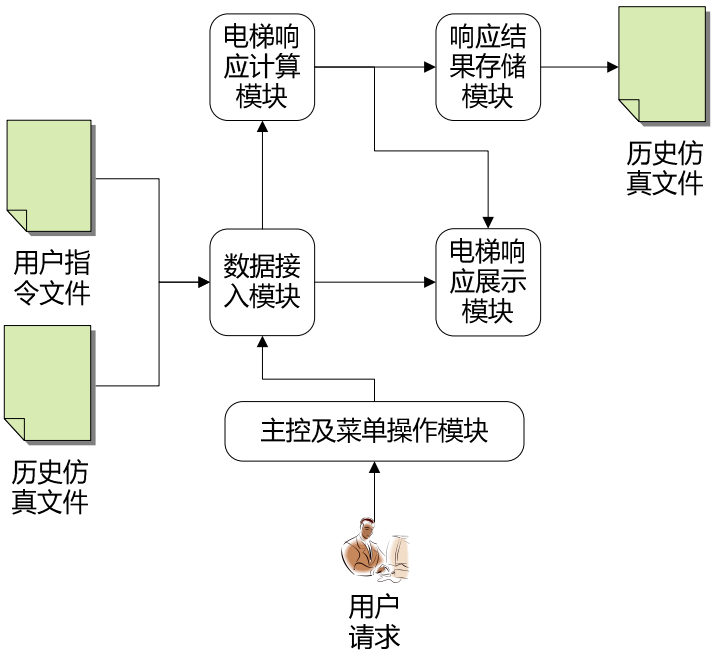

图 3-1 电梯仿真程序系统架构图

如图 3-1 所示，系统总体分为 5 大模块，其中：

- 主控模块：负责整个程序的调度，生成程序菜单，接收用户请求；
- 数据接入模块：负责根据主控程序的调度需要，读取程序需要的文件，并对文件数据进行解读；
- 电梯响应计算模块：负责根据用户指令文件中的用户指令以及电梯的初始状态制定电梯服务计划，即电梯的运动方案；
- 响应结果存储模块：负责将电梯服务计划以一定的格式存储到外部文件中去；
- 电梯响应展示模块：负责将电梯响应结果以图形化的方式展示出来。

3.1.3. 功能架构

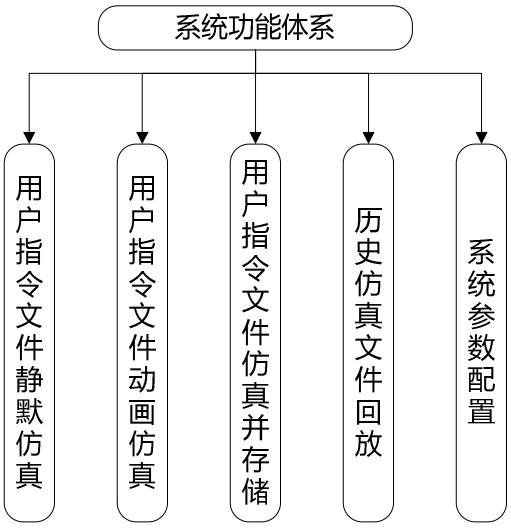

图 3-2 系统功能体系架构图

根据软件需求，如图 3-2 所示，系统主要向用户提供五大功能：

- 用户指令文件静默仿真：使用本功能，用户指定要仿真的用户指令文件，系统根据电梯联动规则计算出电梯服务计划，并将电梯服务计划输出保存为历史仿真文件；
- 用户指令文件动画仿真：使用本功能，用户指定要仿真的用户指令文件，系统根据电梯联动规则计算出电梯服务计划，并在屏幕上以动画的方式将电梯服务过程仿真展示出来；
- 用户指令文件仿真并存储：使用本功能，用户指定要仿真的用户指令文件，系统根据电梯联动规则计算出电梯服务计划，将电梯服务计划输出保存为历史仿真文件，并在屏幕上以动画的方式将电梯服务过程仿真展示出来，本模块也可简称为全面仿真模块；
- 历史仿真文件回放：使用本功能，用户指定要回放的历史仿真文件，系统将文件数据读取出来并根据文件数据，以动画的方式将电梯仿真过程展示出来；
- 系统参数配置：使用本功能，用户可以对系统各个参数进行设置；

3.1.4. 业务流程设计

整个系统业务流程如图 3-3 所示。程序主要由主控模块进行控制，程序启动后，首先执行主控模块，对系统进行自检，自检内容包括各个数据文件夹是否存在，配置文件是否存在等，自检无误后，程序加载系统配置文件，将系统各项参数从文件中读取出来并存入相应变量，然后在屏幕上输出功能菜单等待用户响应。当用户选择了相关的功能后，由主控程序调用相应的功能模块执行。

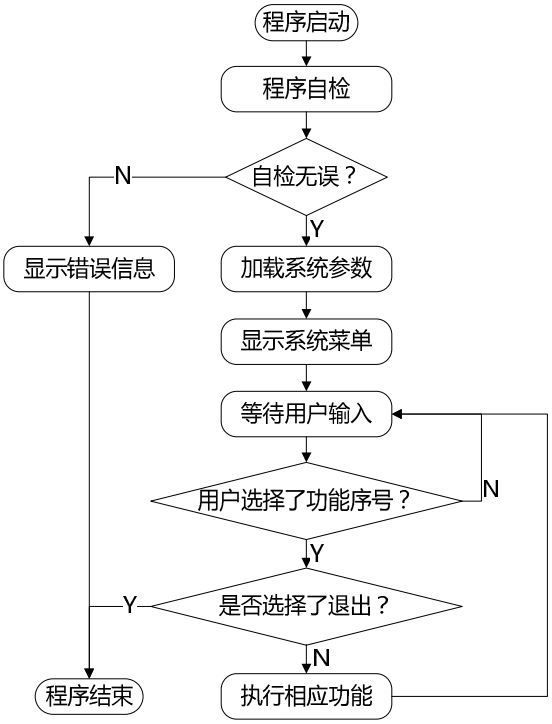

图 3-3 系统业务流程

3.2.模块化设计

3. 2. 1. 程序源文件划分设计

根据系统架构的模块划分，拟采用 11 个程序文件来存储系统程序源代码，各个文件名称及说明如表 3-1 所示。

表 3-1 系统程序文件划分设计表

模块名称	文件名称	文件说明
主控模块	ElevatorSimulation.h	用于存储系统主控模块的相关功能和函数源代码，包括系统主函数、自检函数、菜单逻辑等。
	ElevatorSimulation.cpp	
数据接入模块	DataImport.h	用于存储系统数据接入模块的相关功能和函数源代码，包括用户指令文件的读取，历史仿真文件的读取和系统配置文件的读取等。
	DataImport.cpp	
电梯响应计算模块	ResponseCalculate.h	用于存储电梯响应计算模块的相关功能和函数源代码。
	ResponseCalculate.cpp	
响应结果存储模块	DataExport.h	用于存储响应结果存储模块的相关功能和函数源代码，包括将响应计算结果存储到历史仿真文件以及将系统配置参数写入系统配置文件等。
	DataExport.cpp	
电梯响应展示模块	ScreenSimulation.h	用于存储电梯响应展示模块的相关功能和函数源代码。

	ScreenSimulation.cpp	
全局数据结构	DataModel.h	用于存储系统数据结构体的声明

3.2.2. 程序目录组织结构设计

根据系统架构及系统功能体系，采用如图 3-4 所示的目录结构存储系统最终可执行程序。

图 3-4 系统目录组织结构

其中“[程序安装目录]”为整个程序的根目录，其具体值以用户 PC 机上的实际安装路径为准，系统所有可执行文件和配套文件均放在该目录下；“SysConf”文件夹用于存放系统配置文件，系统配置文件为“SysParam.txt”；“UserRequest”文件夹为默认的用户指令文件存放位置，用于存放用户指令文件；“SimulationFiles”文件夹为默认的历史仿真文件存放位置，用于存放历史仿真文件。

3.2.3. 电梯联动逻辑设计

本系统实现两部电梯的联动仿真，本节对电梯仿真逻辑进行设计和约定，

- 两部电话编号用 A、B 表示；
- 当有用户请求指令到来，并且两部电梯均处于空闲状态时，优先使用 A 号电梯提供服务；
- 电梯运行状态分为三个个状态：
  - 停止状态：表示电梯当前停靠在某一楼层，用 S 表示；
  - 上行状态：表示电梯正在向上运行，用 U 表示；
  - 下行状态：表示电梯正在向下运行，用 D 表示；
- 电梯三个状态之间的转换关系如图 3-5 所示，由停止状态转换为上升或下降状态，需要消耗 1 个单位时间，且楼层不变，而由上升或下降状态转换为停止状态，不需要消耗时间。举例来说假设第 3 时刻 A 号电梯位于 3 层，当电梯需要上行时，在第 4 时刻，电梯由停止状态转换为上行状态，此时电梯仍处于 3 层，然后，在后续时刻中，电梯开始上行。同样假设第 3 时刻 A 号电梯位于 3 层，电梯处于上行状态，并且每一个单位时间，电梯可以运动 1 层，那么第 4 时刻电梯应处于第 4 层，如果电梯目标是运动到第四层接送人员进行服务，则可以在第 4 时刻直接转换为停止状态，而不需要等到第 5 时刻才转换为停止状态；

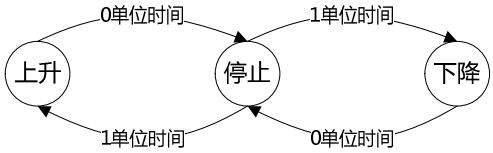

图 3-5 电梯状态转换图

- 当电梯处于上行或下行状态时，表示电梯正在响应用户指令，此时电梯分为两种状态，我们将其称为服务状态，具体服务状态包括：
  - 提供服务前状态：在这一状态下，表示电梯正在向用户所在楼层运动，尚未接到用户并开始提供服务，用 P 表示；
  - 提供服务中状态：在这一状态下，表示电梯已接到用户并在向所在用户目标楼层移动，用 E 表示；
- 系统设置一个用户请求指令队列，用于模拟用户指令依次由用户发出的情景。其保存当前时刻已到来的用户指令，电梯需要不断检查这个队列中是否有用户请求到来，并指派电梯为用户提供服务；
- 每部电梯均设置一个当前服务指令队列，表示当前电梯正在响应哪几个用户指令，当电梯响应用户的某一请求指令时，系统将该指令由用户请求指令队列中取出，加入到具体电梯的服务指令队列中去；
- 在每一个时刻电梯对当前电梯的状态、用户的指令请求进行检查，根据当前电梯状态和用户指令计算下一个时刻电梯处于什么状态，将下一时刻的电梯状态输出；然后将电梯下一时刻的状态作为当前状态，重复上述步骤，一直到电梯对所有用户指令均完成响应为止，如图 3-6 所示；

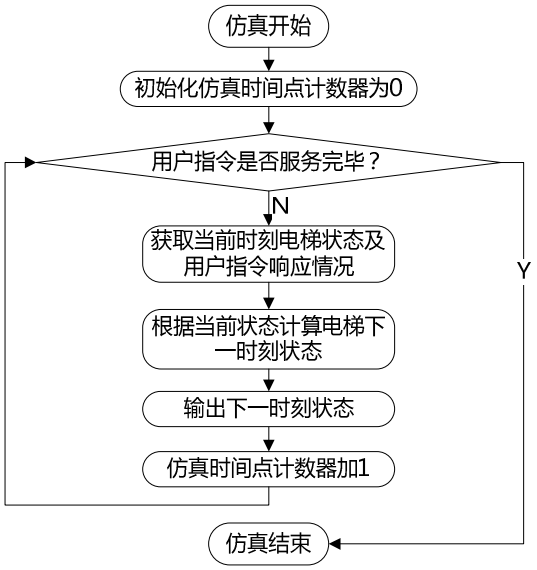

图 3-6 电梯仿真程序仿真逻辑流程图

- 某一时刻电梯根据当前状态确定下一时刻状态的具体判断逻辑如图 3-7、图 3-8、图 3-9 所示。

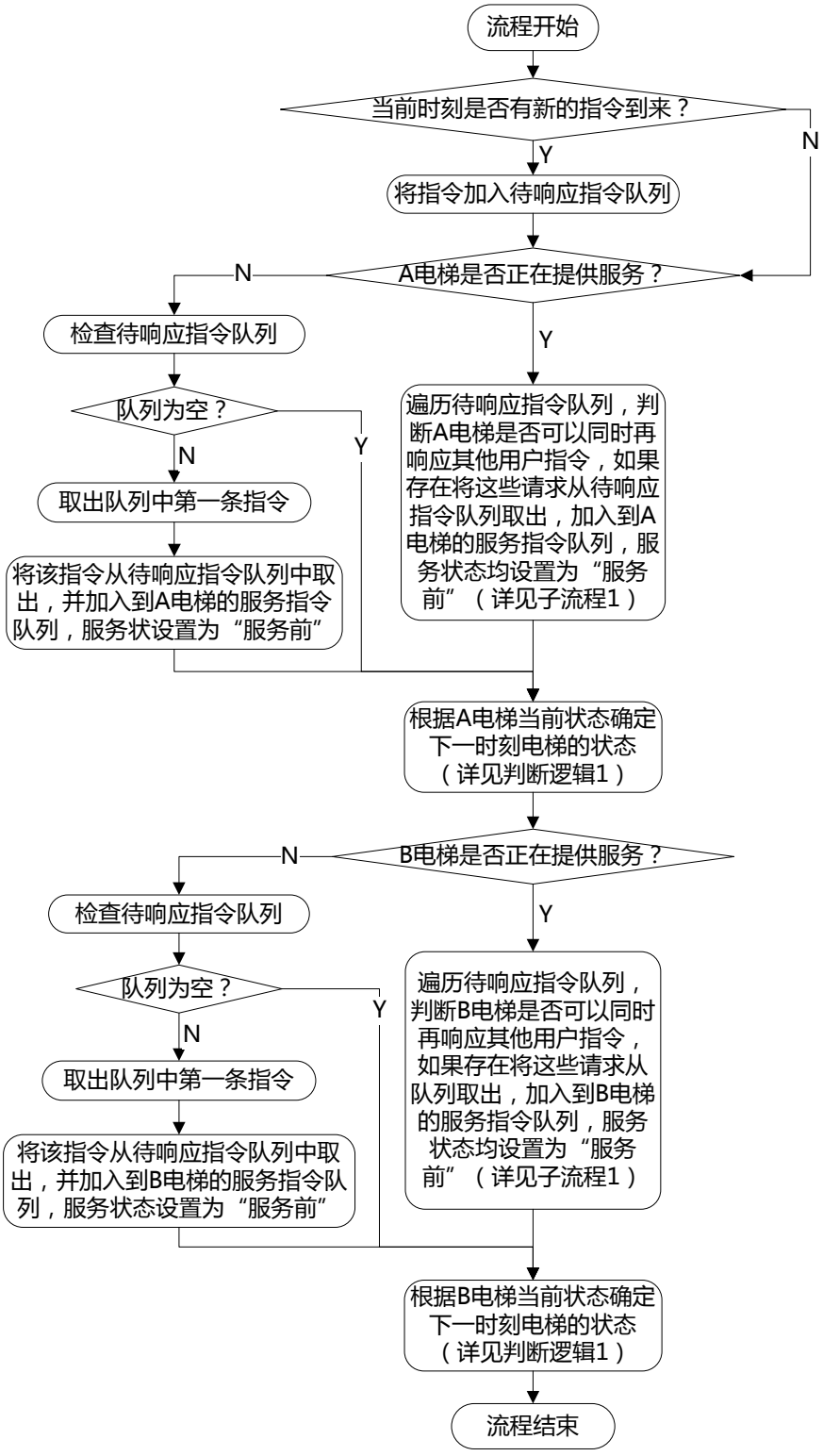

图 3-7 当前时刻电梯处理逻辑流程图

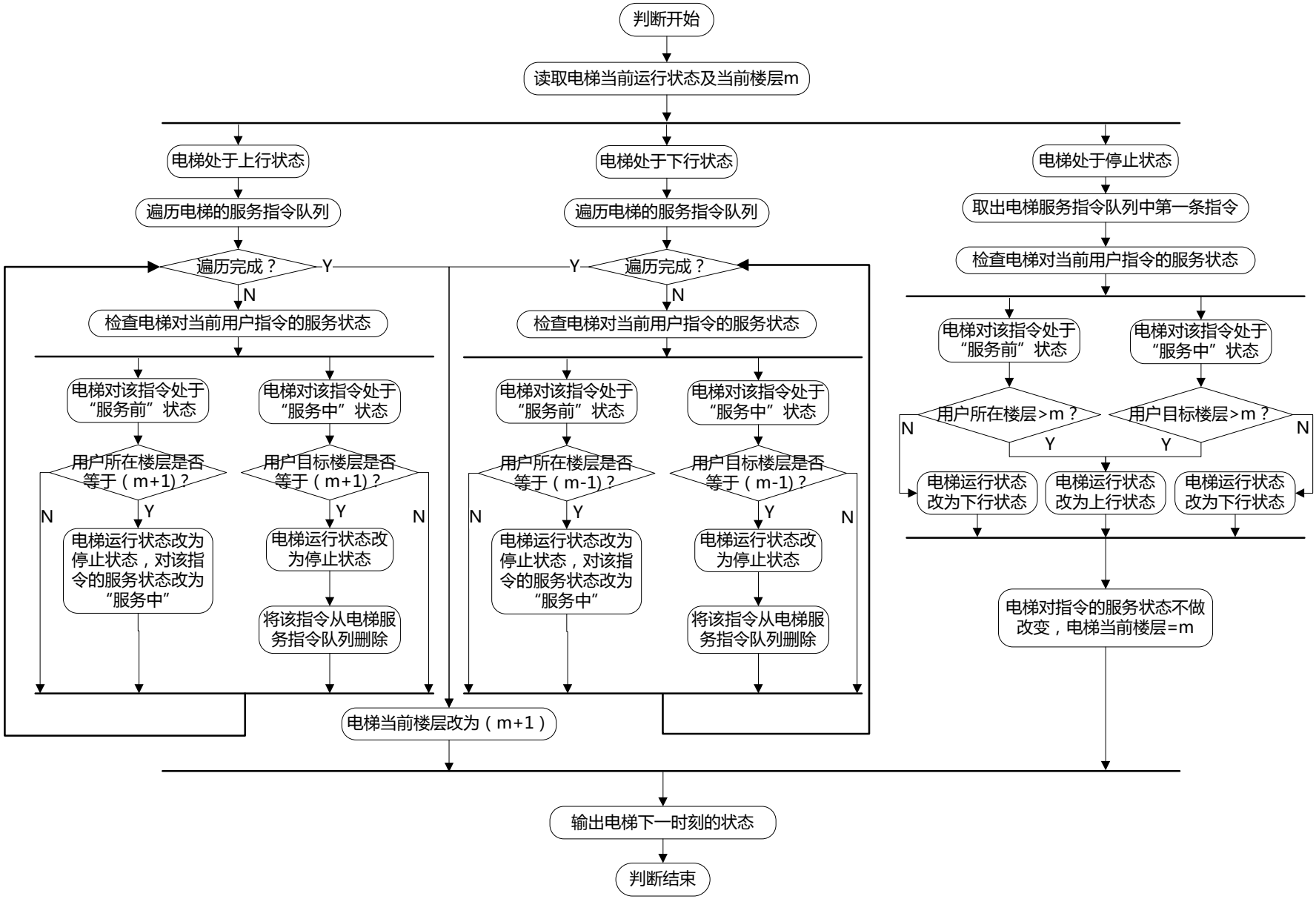

图 3-8 判断逻辑 1—根据电梯当前状态确定下一时刻电梯状态的判断逻辑图

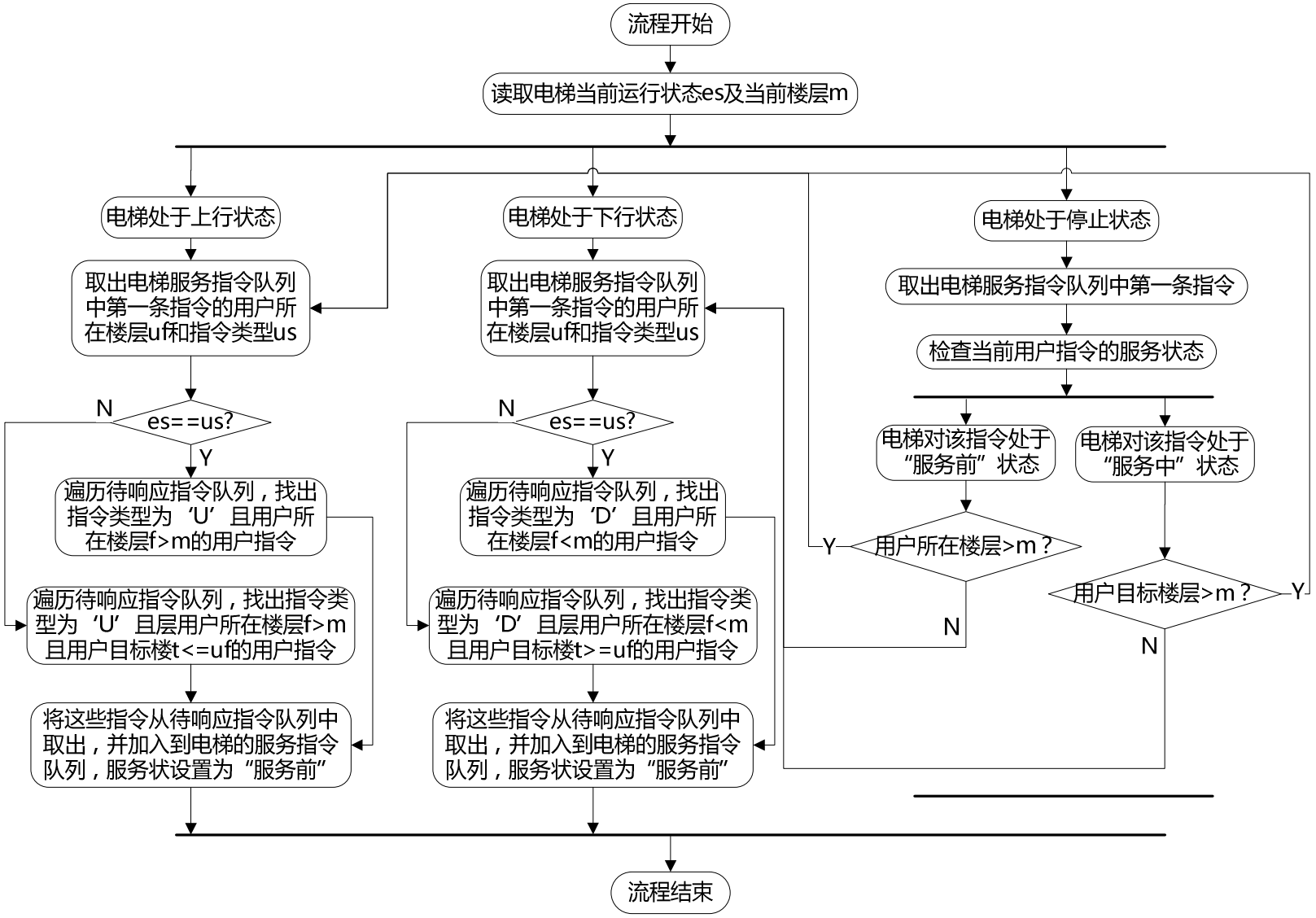

图 3-9 子流程 1—判断电梯是否可以同时响应其它用户指令的流程

- 根据上述业务流程，假设某一时刻电梯处于 m 层，电梯服务的用户指令队列中有 i 条指令，每条指令对应的用户所在楼层为 f_i，用户目标楼层为 t_i，该条指令当前的服务状态为 p_i，则电梯在某时刻其状态与电梯响应的用户指令各参数间必须满足以下关系：
  - 当电梯运行状态为“U”或“D”时，电梯服务指令队列肯定不为空；
  - 当电梯运行状态为“U”时，若 p_i 为“P”，则必有 $m<f_i$，若 p_i 为“E”，则必有 $m<t_i$；
  - 当电梯运行状态为“D”时，若 p_i 为“P”，则必有 $m>f_i$，若 p_i 为“E”，则必有 $m>t_i$；
  - 当电梯服务队列不为空时，对于电梯服务队列中的各个指令，当服务状态为“P”时，不存在 $m==f_i$ 的情况，当服务状态为“E”时，不存在 $m==t_i$ 的情况；
  - 当电梯运行状态为“S”时，若电梯服务指令队列为空，表示此时电梯为空载、静止状态，除非后续有新指令到来，否则不做任何操作，状态也不做改变。若电梯服务指令队列不为空，则表示此时电梯仍在响应用户指令为用户提供服务，电梯临时停止接用户进电梯或完成服务送客户处电梯。

4. 系统详细设计

4.1. 数据对象设计

4. 1. 1. 数据外部存储格式设计

根据系统概要设计所述，系统需要使用三种外部文件：用户指令文件、历史仿真文件和系统配置文件，在本节将对这三种文件的数据组织形式进行设计和说明。

一、用户指令文件

1. 用户指令文件存储格式说明

用户指令文件用于存储用户向电梯发送的指令，根据 3.1.1 节所述，用户指令以“[用户当前所在楼层]，[用户目标楼层]，[指令提出的时间点]”的方式给出，因此用户指令文件将以用户指令三元组的方式存储。用户指令文件内容组织规则如下：

- 文件第一行存储一个整型数字，用于指示文档中一共存储了几条用户指令；
- 从文件第二行开始，依次存储代表用户请求指令的三元组，每一行只允许存储一个三元组；
- 每一个三元组代表一条用户请求指令，元素之间以“,”（英文逗号）进行分隔，第一个元素代表用户当前所在楼层，第二个元素代码用户目标到达楼层，第三个元素代码用户请求的时间点。

- 三元组中的每一个元素必须为整数，前两个元素值必须大于 0（电梯可停靠楼层从 1 开始计数），第三个元素值必须大于等于 0；
- 三元组中，第一个元素和第二个元素不能相等；
- 对于用户指令，需要按照时间点先后顺序进行排序，不可以任意安排用户指令的顺序。

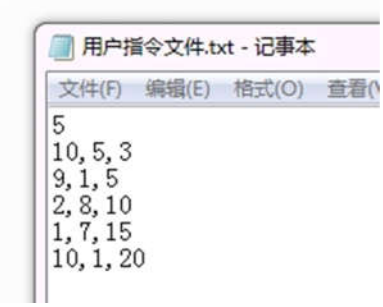

图 4-1 用户指令文件内容组织形式效果图

实际存储效果如图 4-1 所示，效果图中数据表示用户一共发出了 5 条指令（注意，用户指令数据在文件中必须以时间点先后顺序排列）：

- 第一个指令在第 3 分钟发出，用户处于第 10 层，要去第五层；
- 第二个指令在第 5 分钟发出，用户处于第 9 层，要去第 1 层；
- 第三个指令在第 10 分钟发出，用户处于第 2 层，要去第 8 层；
- 第四个指令在第 15 分钟发出，用户处于第 1 层，要去第 7 层；
- 第五个指令在第 20 分钟发出，用户处于第 10 层，要去第 1 层；

四. 用户指令文件命名规则说明

用户指令文件由用户自行创建，故不对其命名做特殊要求，用户可以随意指定文件名称，指令文件名作为系统输入，在系统开始运行时由用户出入给系统。

四. 用户指令文件存储规则说明

用户指令文件必须存放到系统指定的文件夹，系统才能正确识别并进一步操作。

二、系统配置文件

1. 系统配置文件存储格式说明

系统配置文件用于存放系统各类可配置参数，其数据组织规则如下：

- 文件第一行存储一个整型数字，用于指示文档中存储的参数个数；
- 从文件第二行开始，依次存储相应的参数，每行存储一个参数及其参数值；
- 每一个参数均采用“[参数名]=[参数值]”的方式描述，参数及参数值之间用“=”（英文等号）进行分隔；

系统配置文件内容组织方式效果如图 4-2 所示（实际需要存储的参数及参数值见 4.1.3 节所述）。

四. 系统配置文件命名规则说明

系统配置文件命名固定，文件名为“**SysParam.txt**”；

四. 系统配置文件存储规则说明

系统配置文件必须存放在程序安装目录的“**SysConf**”文件夹下。

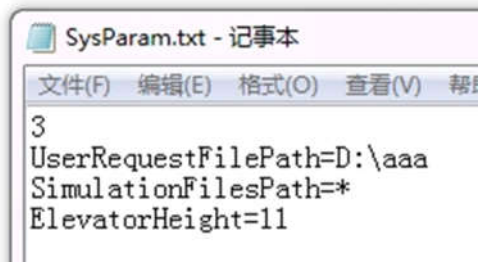

图 4-2 系统配置文件内容组织形式效果图

三、历史仿真文件

1. 历史仿真文件存储格式说明

历史仿真文件也叫仿真结果文件，用于存储电梯对用户指令的仿真结果数据，为了能够实现对数据的回放，文件需要保存 3 部分数据：用户指令数据，仿真参数数据和电梯响应数据，文件组织规则如下：

- 文件存储的三大类型数据，存储顺序为用户指令数据、仿真参数数据、电梯响应数据，三部分数据依次存储，数据块之前用“**********”进行分隔；
- 用户指令数据部分的存储格式与用户指令文件数据组织方式相同，数据块的第一行存放指令条数，后续依次存放用户指令；
- 仿真参数数据部分的存储格式与系统配置文件数据组织方式相同，数据块的第一行存放参数数目，后续依次存放每一个参数及参数值（注意，这里存储的仿真参数与系统配置文件中存储得配置参数可能回有所不同）；
- 电梯响应数据按时间序列顺序存储，每一行记录一个时刻电梯的运行状态和当前电梯响应的用户指令，每一条记录的记录格式如下：

  <当前时刻>,A,<A 所处楼层>,<A 状态>,B,<B 所处楼层>,<B 状态>;[<用户指令序号,用户所在楼层,目标楼层,电梯服务状态,响应电梯标志>,……]
- 每一条电梯响应数据包含两部分内容，前半部分记录当前时刻的电梯状态，后半部分记录当前电梯需要响应的指令集合，两部分数据用英文“;”隔开；
- 对于每一条电梯响应数据，前半部分当前时刻电梯状态数据是必须存在，后半部分数据，根据电梯状态，可以存在，也可以为空；
- 电梯状态信息部分主要记录 5 个数据：当前时刻、第一部电梯编号、第一部电梯当前所在楼层、第一部电梯当前状态、第二部电梯编号、第二部电梯当前所在楼层、第二部电梯当前状态，各个数据项之间用英文“,”分隔；

- 电梯状态信息部分记录的电梯状态主要为电梯运行状态，即取值为 U、S、D，不记录电梯服务状态；
- 电梯响应数据的后半部分为电梯待响应的用户指令集合，由 0 个或多个“<>”表示的五元组组成，多个五元组之间用英文“,”分隔；
- 电梯待响应的用户指令集合中，每个五元组包含五个元素，第一个元素指明当前电梯响应的用户指令在指令数据集中的排序序号(指令从 1 开始排序)，第二个元素指明用户当前发出指令时所在的楼层，第三个元素指明用户的目标楼层，第四个元素指明电梯对当前这条指令的服务状态，第无个元素指明当前哪部电梯正在响应这个用户指令，如果该指令没有电梯为其提供服务，则五元组中第四个和第五个元素均用“N”表示；
- 只有在电梯状态改变或电梯待响应用户指令集有变动是，才记录当前时刻的电梯运行数据，其余时刻数据不做记录；

历史仿真文件内容组织方式效果如图 4-3 所示。

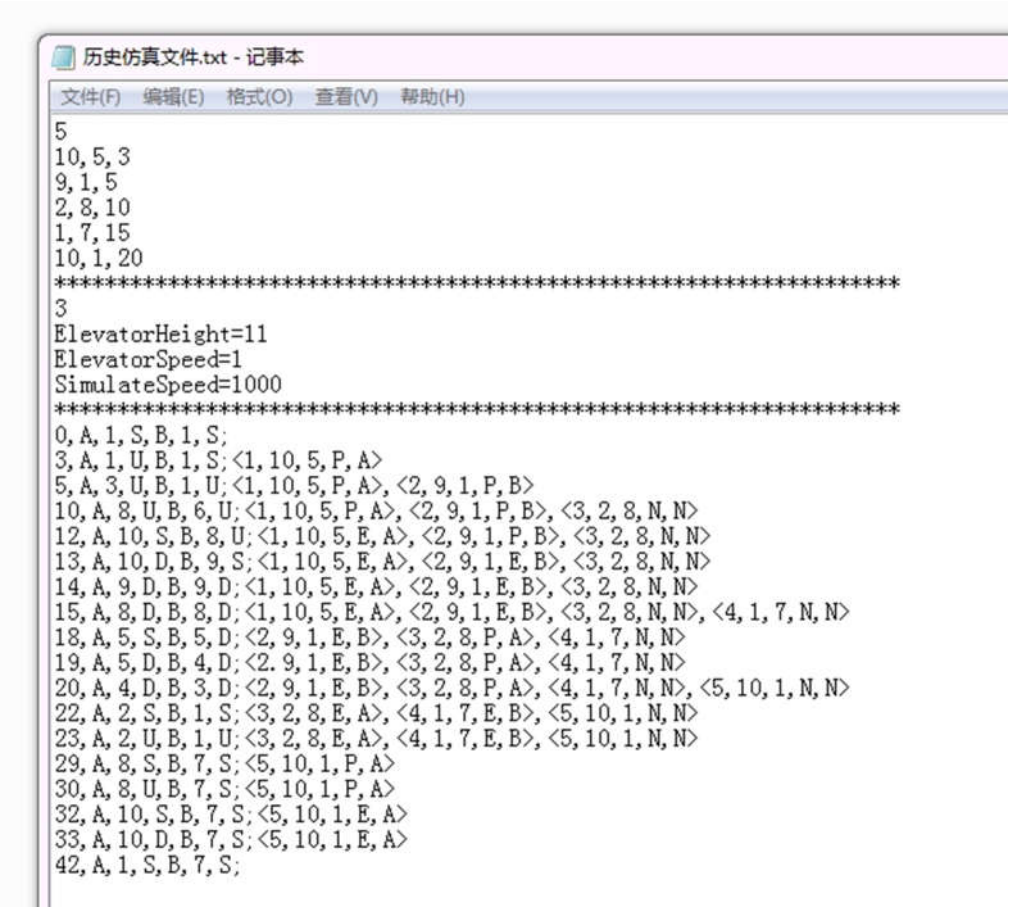

图 4-3 历史仿真文件内容组织形式效果图

从图 3-12 可看出，本次仿真包含 5 个用户指令，系统仿真过程可描述如下（电梯响应数据集部分）：

1) 0 时刻，A 电梯位于 1 层，处于停止状态，B 电梯位于 1 层，处于停止状态；

2) 3 时刻，第一个用户请求到来，A 电梯响应。此时 A 电梯位于 1 层，处于上行状态，B 电梯位于 1 层，处于停止状态。电梯响应数据为：“<1,10,5,P,A>”，即由 A 电梯响应第一条指令，此时刻电梯处于“服务前”状态；
3) 5 时刻，第二个用户请求到来，B 电梯响应。此时 A 电梯位于 3 层（一个时间间隔电梯走一层），处于上行状态，B 电梯位于 1 层，处于上行状态。电梯待响应用户指令集合为：“<1,10,5,P,A>,<2,9,1,P,B>”；
4) 10 时刻，第三个用户请求到来，由于两部电梯均处于运行状态，所以无响应该指令的电梯。此时 A 电梯位于 8 层，处于上行状态，B 电梯位于 6 层，处于上行状态。电梯待响应指令集合为：“<1,10,5,P,A>,<2,9,1,P,B>,<3,2,8,N,N>”；
5) 12 时刻，A 电梯到达用户楼层，等待用户进电梯，电梯状态改变。此时 A 电梯位于 10 层，处于停止状态，B 电梯位于 8 层，处于上行状态。电梯待响应指令集合为：“<1,10,5,E,A>,<2,9,1,P,B>,<3,2,8,NN>”；
6) 13 时刻，A 电梯开始下行，B 电梯到达用户楼层，电梯状态改变。此时 A 电梯位于 10 层，处于下行状态，B 电梯位于 9 层，处于停止状态。电梯待响应指令集合为：“<1,10,5,E,A>,<2,9,1,E,B>,<3,2,8,N,N>”；
7) 14 时刻，B 电梯开始下行，电梯状态改变。此时 A 电梯位于 9 层，处于下行状态，B 电梯位于 9 层，处于下行状态。电梯待响应指令集合为：“<1,10,5,E,A>,<2,9,1,E,B>,<3,2,8,N,N>”；
8) 15 时刻，第四个用户请求到来。此时 A 电梯位于 8 层，处于下行状态，B 电梯位于 8 层，处于下行状态。电梯待响应指令集合为：“<1,10,5,E,A>,<2,9,1,E,B>,<3,2,8,N,N>,<4,1,7,N,N>”；
9) 18 时刻，A 电梯到达 5 层，完成用户指令响应，电梯状态变更，A 电梯开始响应新的指令。此时 A 电梯位于 5 层，处于停止状态，B 电梯位于 5 层，处于下行状态。电梯待响应指令集合为：“<2,9,1,E,B>,<3,2,8,P,A>,<4,1,7,N,N>”；
10) 19 时刻，A 电梯响应新用户指令，A 电梯状态改变。此时 A 电梯位于 5 层，处于下行状态，B 电梯位于 4 层，处于下行状态。电梯待响应指令集合为：“<2,9,1,E,B>,<3,2,8,P,A>,<3,1,7,N,N>”；
11) 20 时刻，第五个用户指令到来。此时 A 电梯位于 4 层，处于下行状态，B 电梯位于 3 层，处于下行状态。电梯待响应指令集合为：“<2,9,1,E,B>,<3,2,8,P,A>,<4,1,7,N,N>,<5,10,1,N,N>”；
12) 22 时刻，B 电梯到达指定位置，电梯状态变更，B 电梯开始响应新指令，A 电梯到达用户楼层，电梯状态变更。此时 A 电梯位于 2 层，处于停止状态，B 电梯位于 1 层，处于停止状态。电梯待响应指令集合为：“<3,2,8,E,A>,<4,1,7,E,B>,<5,10,1,N,N>”；

13) 23 时刻，A、B 电梯为了响应用户指令，均转换状态。此时 A 电梯位于 2 层，处于上行状态，B 电梯位于 2 层，处于上行状态。电梯待响应指令集合为：“<3,2,8,E,A>,<4,1,7,E,B>,<5,10,1,N,N>”；
14) 29 时刻，A、B 电梯均到达目标楼层，转换状态，A 电梯优先响应用户指令。此时 A 电梯位于 8 层，处于停止状态，B 电梯位于 7 层，处于停止状态。电梯待响应指令集合为：“<5,10,1,P,A>”；
15) 30 时刻，A 电梯开始运行，转换状态。此时 A 电梯位于 8 层，处于上行状态，B 电梯位于 7 层，处于停止状态。电梯待响应指令集合为：“<5,10,1,P,A>”；
16) 32 时刻，A 电梯到达用户楼层，转换状态。此时 A 电梯位于 10 层，处于停止状态，B 电梯位于 7 层，处于停止状态。电梯待响应指令集合为：“<5,10,1,E,A>”；
17) 33 时刻，A 电梯开始运行，转换状态。此时 A 电梯位于 10 层，处于下行状态，B 电梯位于 7 层，处于停止状态。电梯待响应指令集合为：“<5,10,1,E,A>”；
18) 42 时刻，A 电梯到达目标楼层，转换状态，整个仿真结束。此时 A 电梯位于 1 层，处于停止状态，B 电梯位于 7 层，处于停止状态。电梯待响应指令集合为空；

四. 历史仿真文件命名规则说明

历史仿真文件的命名与该仿真文件对应的用户指令文件的文件名相关，以用户指令文件的文件名为基础，在其后添加“_SimResult”后缀。举例来说，假设用户指令文件名为“UserFile001.txt”则对这个指令文件进行仿真后，输出保存的历史仿真文件名称为“UserFile001_ SimResult.txt”。

四. 历史仿真文件存储规则说明

历史仿真文件必须存放到系统指定的文件夹，存放路径有系统参数指定，如果用户未指定系统参数，则默认存放在系统 exe 文件相同的目录下。

4.1.2. 数据内部存储模型设计

为了便于在程序中操作数据，对系统数据内部存储做以下设计。

一、用户指令数据

用户指令数据采用变长结构体数组存储，数组长度为用户指令个数，根据用户指令文件中给出的指令个数动态申请空间，数组元素为一个结构体，描述了用户指令的三个必要信息。此外，根据用户指令的特点，在结构体中增加了一个类型分量，用于区分用户指令的类型，当用户所在楼层大于用户目标楼层时，为“下行”类型的用户指令，反之为“上行”类型的用户指令，分别用“U”和“D”表示。用户指令结构体声明如下：

```
typedef  struct  UserCall{
```

```
    int  user_floor ;      //用户所在楼层
    int  user_target ;     //用户目标楼层
    int  call_time ;       //用户请求时刻
    char  call_type ;      //用户指令类型，´U´表示上行指令，´D´表示下行指令
}USERCALL ;
```

用户指令数组首地址用指针变量“**usercall_list**”存储。用户指令数组长度用变量“**usercall_list_len**”存储。

二、系统参数数据

用户参数数据采用定长结构体数组存储，数组长度为系统参数个数（具体参数个数见 4.1.3 节），用数值常量“SYS_PARAM_ARRAY_LEN”表示。数组元素为一个结构体，描述了用户参数及参数值，用户参数结构体声明如下：

```
typedef  struct  SysParam{
    char  param_name[MAX_PARAM_NAME_LEN] ;     //参数名
    char  param_value[MAX_PARAM_VALUE_LEN] ;   //参数值
}SYSPARAM ;
```

其中“MAX_PARAM_NAME_LEN”和“MAX_PARAM_VALUE_LEN”为两个常量，表示存放参数名和参数值的字符数组最大存储字符个数，其值需保证能够存储下系统所需的参数和参数值，其值由同学们根据 4.1.3 节所述自行拟定。

三、电梯状态数据

电梯状态数据采用结构体变量存储，主要存储内容包括电梯状态及电梯服务的指令队列，电梯服务指令队列采用无头结点的链表形式存储，其链表结点结构体声明如下：

```
typedef  struct  ServeListNode{
    char  serve_state ;      //电梯服务状态
    USERCALL  *user_call ;//电梯当前响应用户指令时，指向指令数组的某一个元素
    struct ServeListNode  *next_node ;//存储下一个结点的地址
}SERVELISTNODE
```

电梯状态数据结构体声明如下：

```
typedef  struct  elevatorstate{
    int  current_floor ;      //电梯当前所处楼层
    char  run_state ;         //电梯运行状态
    SERVELISTNODE  *serve_list  ; //电梯响应的用户指令队列
}ELEVATORSTATE ;
```

由于本系统模拟两部电梯的联动运行，因此需要声明两个上述结构体变量，分别存储 A 号电梯和 B 号电梯的状态数据，两个结构体变量声明如下：

```
ELEVATORSTATE  elevator_a , elevator_b ; //用于存放两部电梯状态的结构体变量
```

四、电梯待响应指令数据

由于用户指令数据已经在用户指令数组中存放，且用户指令数据严格按照时间先后顺序排列，因此，对电梯待响应指令数据的存储形式采用一个带头结点的链表表示，其中链表的头结点保存队列中的结点个数及指向第一个和最后一个数据结点的指针，数据结点的数据域部分存放指令在用户指令数组中的序号。链表数据结点声明如下：

```
typedef  struct  ResponseListNode{
    int  usercall_index;      //用户指令在指令数组中对应的序号
    struct  ResponseListNode  *next_node;//存储下一个结点的地址
}RESPONSELISTNODE ;
```

链表表头结点声明如下：

```
typedef  struct  ResponseListHeadNode{
    int  list_num ;      //待响应用户指令链表中数据结点的个数
    RESPONSELISTNODE  *head ;//链表中第一个数据结点的指针
    RESPONSELISTNODE  *tail ;//链表中最后一个数据结点的指针
}RESPONSELISTHEADNODE ;
```

电梯待响应指令链表声明如下：

RESPONSELISTNODE *response_list ;

以图 4-3 所示的仿真过程为例，在 18 时刻，电梯待响应指令数组及其他内存数据结构中数据值状态如图 4-4 所示。此时可以看出，A 电梯正在响应第 2 条用户指令，B 电梯正在响应第 3 条用户指令，而此时还有一条用户指令等待响应，因此 response_list 数组的头指针和为指针均指向第 4 条指令。

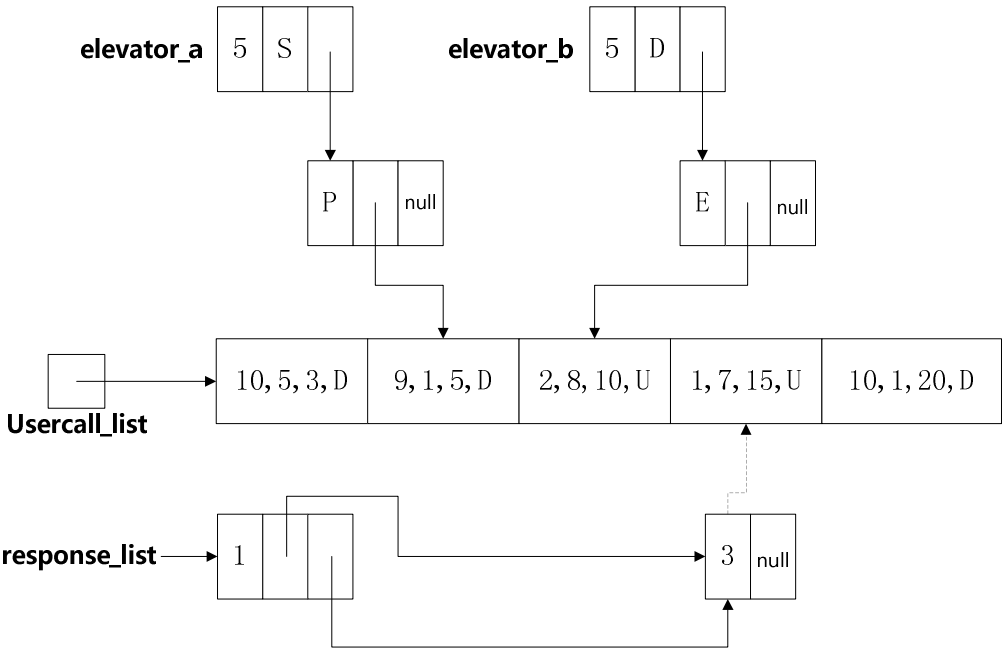

图 4-4 在 18 时刻内存数据结构中的数据值

4.1.3. 系统参数设计

整个系统共设计参数 4 个，参数名称及参数值如表 4-1 所示：

表 4-1 系统参数一览表

参数名	默认值	参数说明
UserRequestFilePath	*	存放用户指令文件的目录，默认值为“*”时，表示与本系统 exe 文件存放在同一目录下
SimulationFilesPath	*	存放用户历史仿真文件（历史仿真文件），默认值为“*”时，表示与本系统 exe 文件存放在同一目录下
ElevatorHeight	20	电梯总层数
DelayTime	1000	动画展示仿真过程时，每个时刻将电梯状态绘制完毕后，延迟时间，单位为毫秒

4.1.4. 系统全局变量

程序中需要声明的全局系统变量如表 4-2 所示。

表 4-2 系统数据变量表

变量名称	变量类型	变量说明

*usercall_list	USERCALL	用户指令数组首地址，数组根据用户指令多少动态生成
usercall_list_len	Int	用户指令数组长度，从用户指令文件第一行读出
param_array[]	SYSPARAM	系统参数数组
elevator_a	ELEVATORSTATE	A 电梯状态变量
elevator_b	ELEVATORSTATE	B 电梯状态变量
*response_list	RESPONSELISTHEADNODE	电梯响应指令队列，用于指示当前响应的电梯指令
user_file_name	char[FILE_NAME_LEN]	用户存放用户指令文件名称的数组，FILE_NAME_LEN 为常量
finish_call_num	int	用于记录当前已响应完成的用户指令个数
time	int	仿真时间计数器，用于记录当前的仿真时刻
status_change_flag	int	用于记录下一时刻相较于当前时刻电梯状态是否发生了改变，其值为 1 表示电梯状态发生了改变，为 0 表示未发生改变

4.2. 函数设计

4.2.1. 系统主函数 main

系统主函数流程如图 4-5 所示。主函数无返回值，无参数输入。各子函数说明如下：

- SystemInit 函数：负责将系统参数配置文件读入系统，并在读入过程中校验文件目录、配置文件、参数及参数值的正确性；
- ShowMenu 函数：主要负责将系统菜单使用 printf 函数打印输出到界面上，展示给用户，使得用户可通过系统菜单了解到系统功能及调用相应功能的方法；
- SilenceSimulate 函数：用于实现静默仿真模块功能，函数为用户打印模块菜单，接收用户输入，计算仿真结果并直接将结果输出为历史仿真文件；
- MovieSimulate 函数：用于实现动画仿真模块功能，采用动画的方式将电梯服务过程展示出来。
- FullSimulate 函数：用户实现全面仿真模块功能，采用动画进行仿真过程展示的同时将仿真结果保存到结果文件中去；
- HistorySimulate 函数：用于实现历史仿真回放模块功能，读入用户历史仿真文件，用动画的方式将电梯服务过程展示出来；
- ParamConf 函数：用于实现系统配置模块功能。

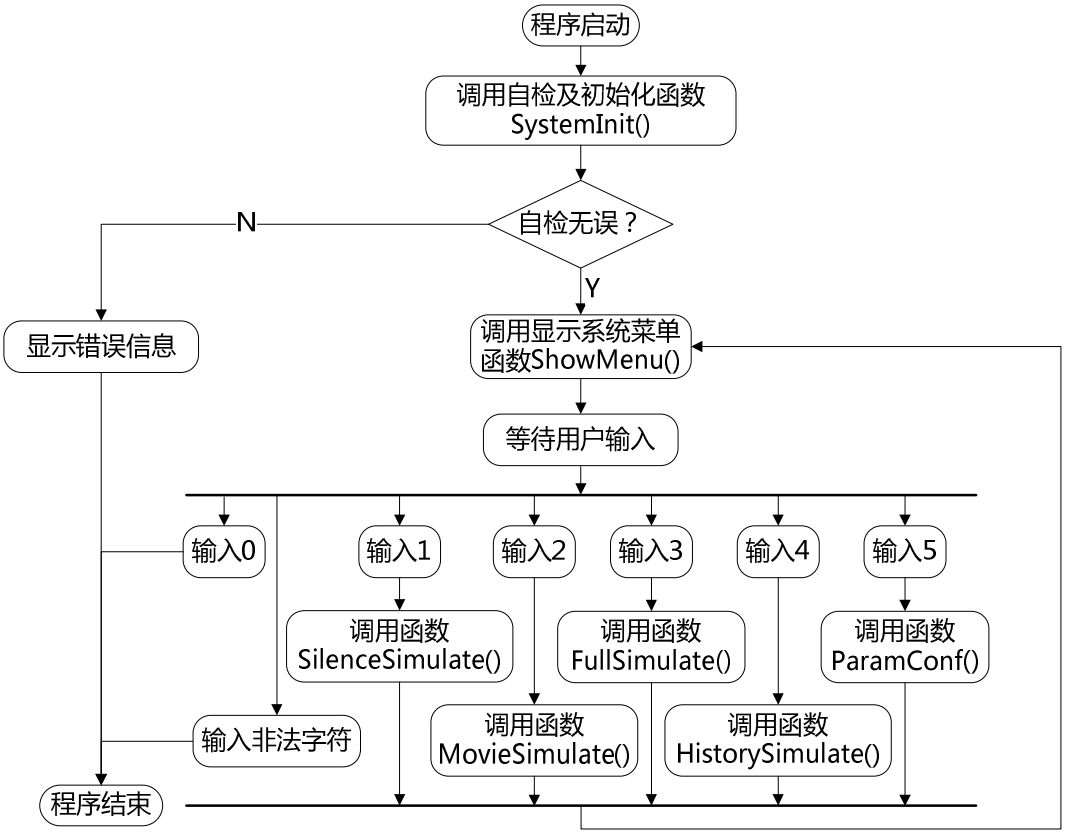

图 4-5 系统主函数流程图

4.2.2. 系统自检及初始化函数 SystemInit

系统自检及初始化函数负责将系统参数配置文件读入系统，并在读入过程中校验文件目录、配置文件、参数及参数值的正确性。函数流程如图 4-6、4-7 所示，无输入参数。返回值为 int 型，返回值取值及其意义如表 4-3 所示。

表 4-3 系统自检及初始化函数返回值含义一览表

函数返回值	返回值含义
1	函数正确执行，系统自检无问题，参数正确加载。
0	系统自检有误找不到 SysConf 目录
-1	系统自检有误，找不到 SysConf 目录下的 SysParam.txt 文件
-2	SysParam.txt 文件打开失败
-3	用户自定义的存放用户指令文件的目录不存在
-4	用户自定义的存放用户历史仿真文件的目录不存在
-5	电梯楼层参数有误
-6	电梯动画仿真延迟参数有误
-7	实际读出的系统参数个数与文件第一行声明的参数个数不符
-8	从参数文件中没有找到任何系统定义的参数

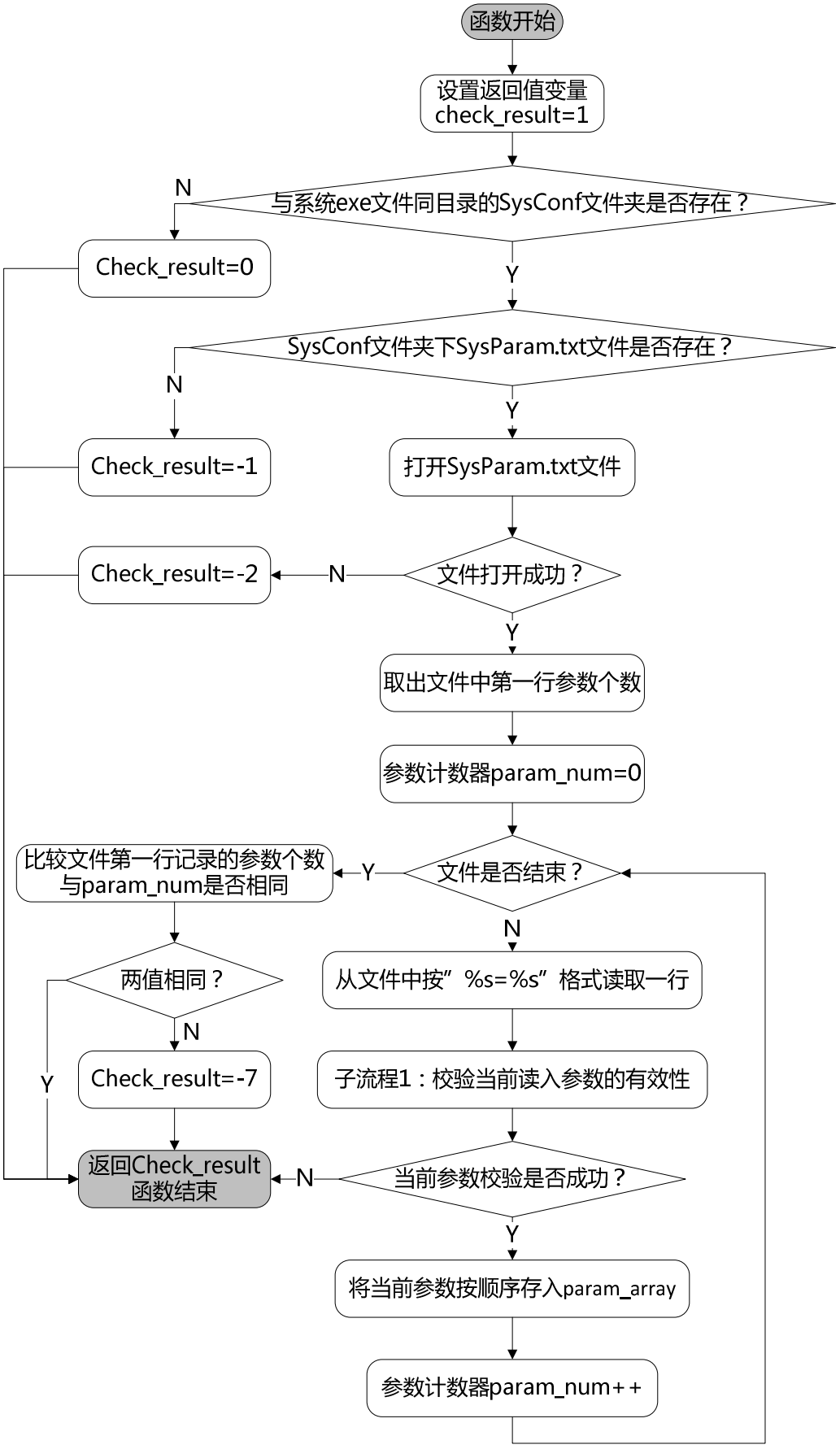

图 4-6 系统自检及初始化函数流程图

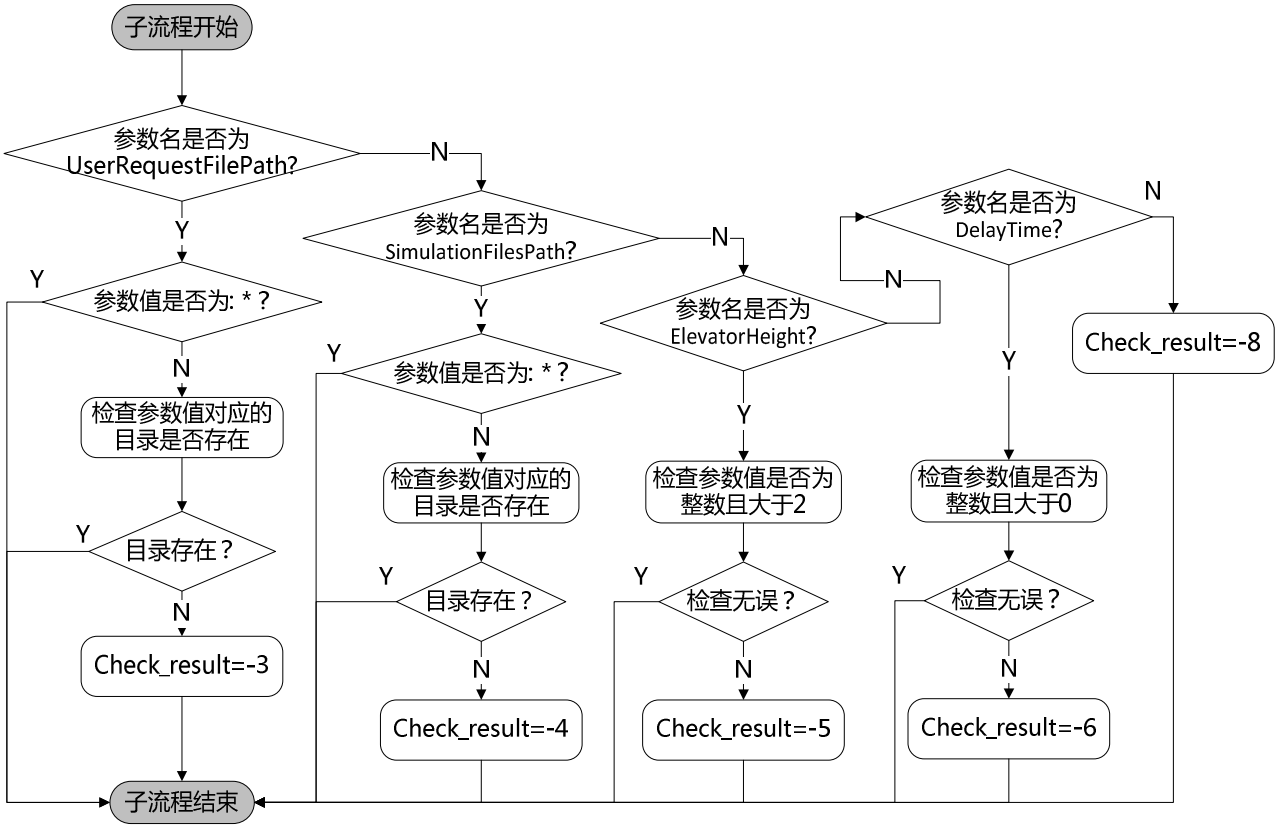

图 4-7 系统自检及初始化函数子流程：参数有效性校验流程图

系统自检及初始化函数声明如下：

int SystemInit() ;//系统自检函数

4.2.3. 显示系统菜单函数 ShowMenu

显示系统菜单函数主要负责将系统菜单使用 printf 函数打印输出到界面上，展示给用户，使得用户可通过系统菜单了解到系统功能及调用相应功能的方法。根据系统功能架构，系统共需要展示五六个菜单：第一个为系统主功能菜单，称之为“功能菜单 1”；第二个为静默仿真模块菜单，该菜单主要提供返回上级菜单功能，称之为“功能菜单 2”；第三个为动画仿真模块菜单，称之为“功能菜单 3”；第四个为全面仿真模块功能菜单，称之为“功能菜单 4”；第五个为历史仿真文件回放模块菜单，称之为“功能菜单 5”；第六个为系统参数配置模块菜单，该菜单提示用户选择要配置的具体系统参数，称之为“功能菜单 6”。菜单具体展示效果如下

【功能菜单 1】

本程序为一个电梯仿真程序，仿真两部电梯的联动运行，提供如下功能：

1. 指定用户指令文件，静默仿真（只输出仿真结果文件）

2. 指定用户指令文件，动画仿真（不输出仿真结果文件）

3. 指定用户指令文件，全面仿真（即显示动画，又输出结果文件）

4.指定历史仿真文件，动画形式回放

5.修改系统参数

0. 退出

请输入您要执行的功能序号：

【功能菜单 2】

您当前处于“静默仿真”模块，请选择要进行的操作：

1. 输入用户指令文件名称，开始静默仿真

0. 返回上级菜单

请输入您需要进行的操作序号：

【功能菜单 3】

您当前处于“动画仿真”模块，请选择要进行的操作：

1. 输入用户指令文件名称，开始动画仿真

0. 返回上级菜单

请输入您需要进行的操作序号：

【功能菜单 4】

您当前处于“全面仿真”模块，请选择要进行的操作：

1. 输入用户指令文件名称，开始全面仿真

0. 返回上级菜单

请输入您需要进行的操作序号：

【功能菜单 3】

您当前处于“历史仿真文件回放”模块，请选择要进行的操作：

1. 输入历史仿真文件名称，开始仿真回放

0. 返回上级菜单

请输入您需要进行的操作序号：

【功能菜单 6】

您当前处于“系统参数配置”模块，请选择要配置的参数：

1. 配置用户指令文件存放目录

2. 配置历史仿真文件存放目录

3. 配置电梯总层数

4. 配置电梯动画仿真时的延迟系数

0. 返回上级菜单

请输入您需要重新配置的参数序号：

显示系统菜单函数无返回参数，输入一个 int 型参数，当参数值为 1 时，打印输出功能菜单 1；当参数值为 2 时，打印输出功能菜单 2，以此类推。当参数值不在 1-6 范围内时，提示错误。函数声明如下：

```
void  ShowMenu(int  menu_flag) ;//显示系统菜单函数
```

4.2.4. 静默仿真函数 SilenceSimulate

静默仿真函数用于实现静默仿真模块功能，函数为用户打印模块菜单，接收用户输入，计算仿真结果并直接将结果输出为历史仿真文件。静默仿真函数涉及 12 个函数及子函数，相关函数声明如下：

```
void  SilenceSimulate() ;//静默仿真模块函数

void  InitSimulation() ;//初始化仿真变量函数

void  InitElevator(ELEVATORSTATE  *elevator) ;//初始化电梯状态变量函数

int  LoadUserCallArray(File  *fp) ;//用户指令数组加载数据函数

int  OutputSimulationResult() ;//计算仿真结果并输出为结果文件的函数

void  ImportUserCall(File  *fp) ;//将用户指令写入仿真结果文件的函数

void  ImportSimulateParam(File  *fp) ;//将仿真参数写入仿真结果文件的函数

void  GetNextTimeStatus(ELEVATORSTATE  *elevator) ;//计算电梯下一时刻状态
```

的函数

void ImportSimulateResult(File *fp) ;//将电梯当前时刻状态写入仿真结果文件的函数

void FindUserCallCanServe(ELEVATORSTATE *elevator , char r_state) ;//处理当前电梯是否可以同时响应其它用户指令的函数

char GetElevatorDirection(ELEVATORSTATE *elevator) ;//电梯在服务过程中处于停止状态时判断其上下行状态的函数

void SetElevatorState(ELEVATORSTATE *elevator) ;//将电梯状态修改为下一时刻状态的函数

静默仿真函数流程如图 4-8、图 4-9、图 4-10、图 4-11、图 4-12、图 4-13、图 4-14、图 4-15、图 4-16、图 4-17、图 4-18、图 4-19 所示。其中：

- SilenceSimulate 函数为静默仿真模块的主函数无输入参数，无返回值；
- InitSimulation 函数负责在静默仿真前初始化系统各个仿真全局变量包括为各个变量赋初值，清空各链表和队列，释放相关内存空间等，函数无输入参数、无返回值；
- InitElevator 函数负责将电梯状态变量初始化为仿真初始值，包括将电梯当前楼层赋值为 1，将电梯状态赋值为“S”及将电梯服务队列赋值为空，函数无返回值，输入为电梯状态变量；
- LoadUserCallArray 函数负责将用户指令文件内的用户指令数据加载到用户指令数组中去，函数输入为用户指令文件的文件指针，返回值为 int 型，当返回值为 0 时表示加载用户指令数组失败，当返回值为 1 时表示加载成功；
- OutputSimulationResult 函数用户将静默仿真结果输出到仿真结果文件中去，函数无输入参数，返回值为 int 型，返回为 0 表示执行失败，返回为 1 表示执行成功；
- ImportUserCall 函数负责将用户指令数据中的用户指令信息写入到仿真结果文件，函数输入为仿真结果文件的文件指针，返回值为空；
- ImportSimulateParam 函数负责将仿真参数写入到仿真结果文件，主要输出的仿真参数有 3 个，分别为：电梯总高度 ElevatorHeight、电梯运行速度参数 ElevatorSpeed、仿真速度参数 SimulateSpeed；
- GetNextTimeStatus 函数负责根据电梯当前时刻的状态信息计算下一时刻的状态信息，函数主要操作对象为系统全局变量，如果下一时刻电梯状态会发生改变，函数会直接将电梯状态信息修改为最新的状态信息，并将 status_change_flag 变量置为 1；

- ImportSimulateResult 函数负责将当前时刻电梯状态写入仿真结果文件，即向仿真结果文件输出一行仿真结果数据；
- FindUserCallCanServe 函数负责检查当前运行中的电梯是否可以同时响应其它待响应的用户指令，如果有，将待响应的用户指令从待响应指令队列中移出并加入电梯的服务队列；
- GetElevatorDirection 函数负责检查当电梯处于响应用户指令状态下，当电梯运行状态为“停止”状态时，根据电梯正在响应的用户指令，判断电梯后续状态；
- SetElevatorState 函数负责根据电梯当前时刻状态，计算出电梯下一时刻状态并将电梯状态修改为下一时刻状态。

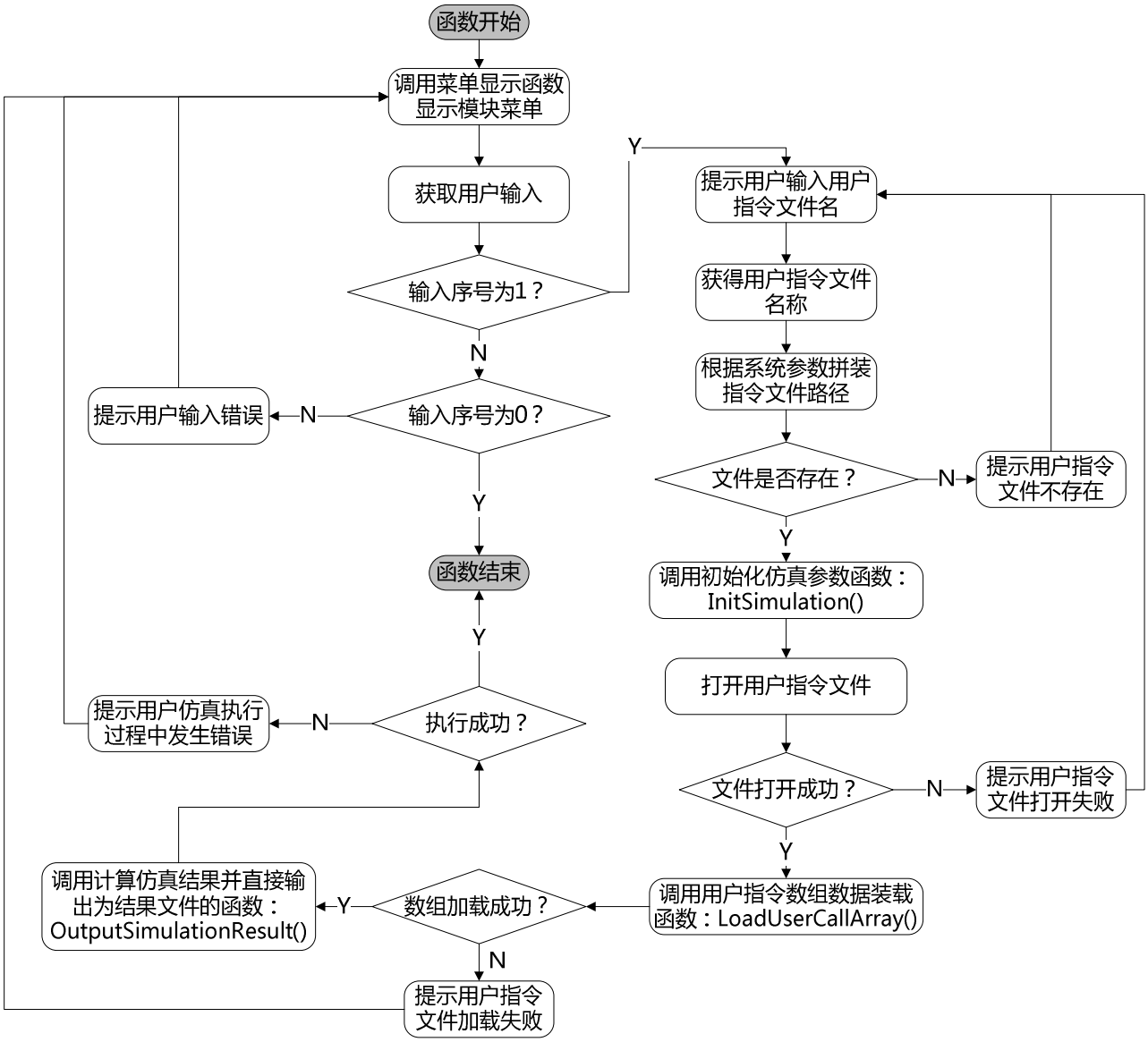

图 4-8 SilenceSimulate 函数流程图

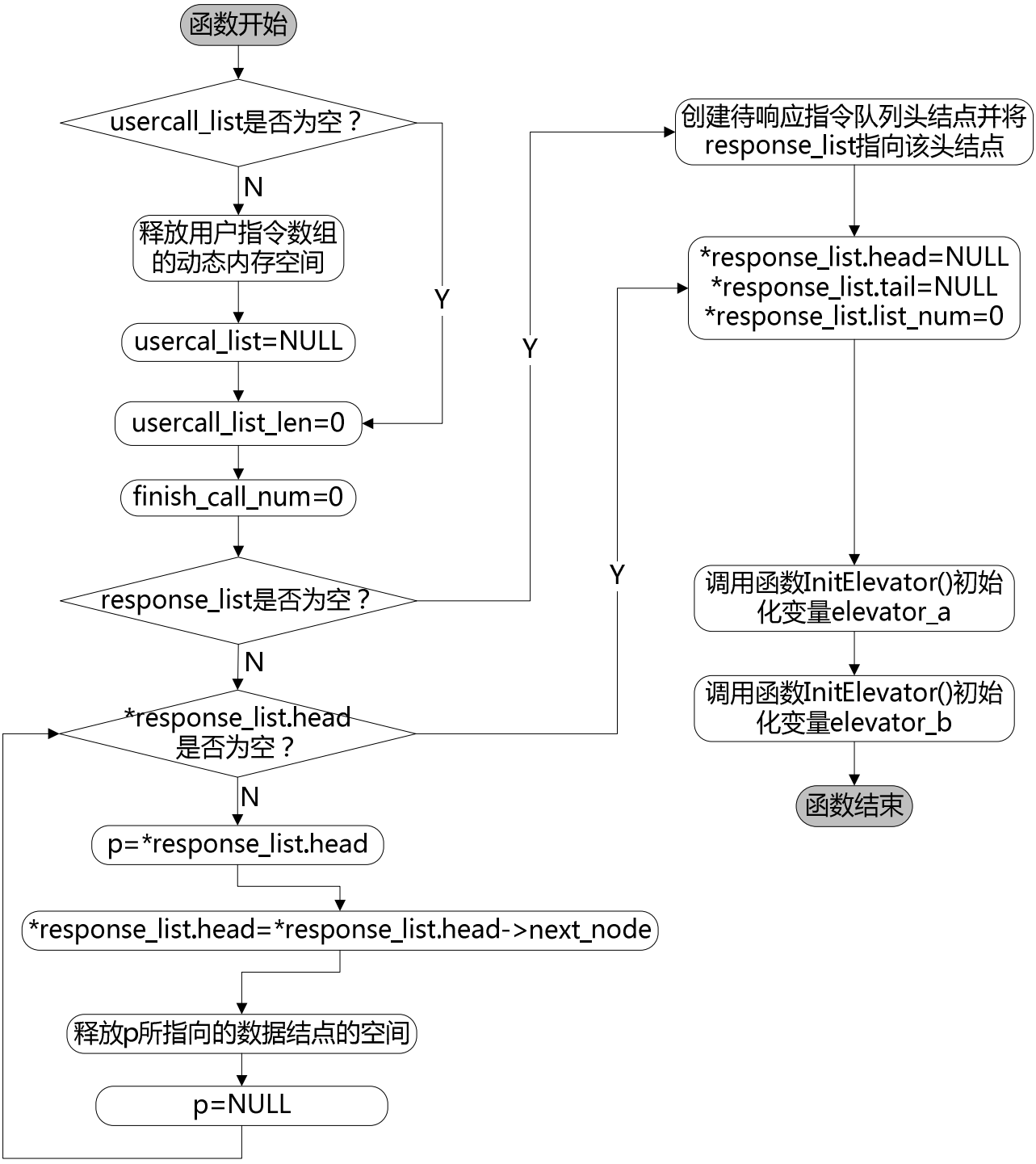

图 4-9 InitSimulation 函数流程图

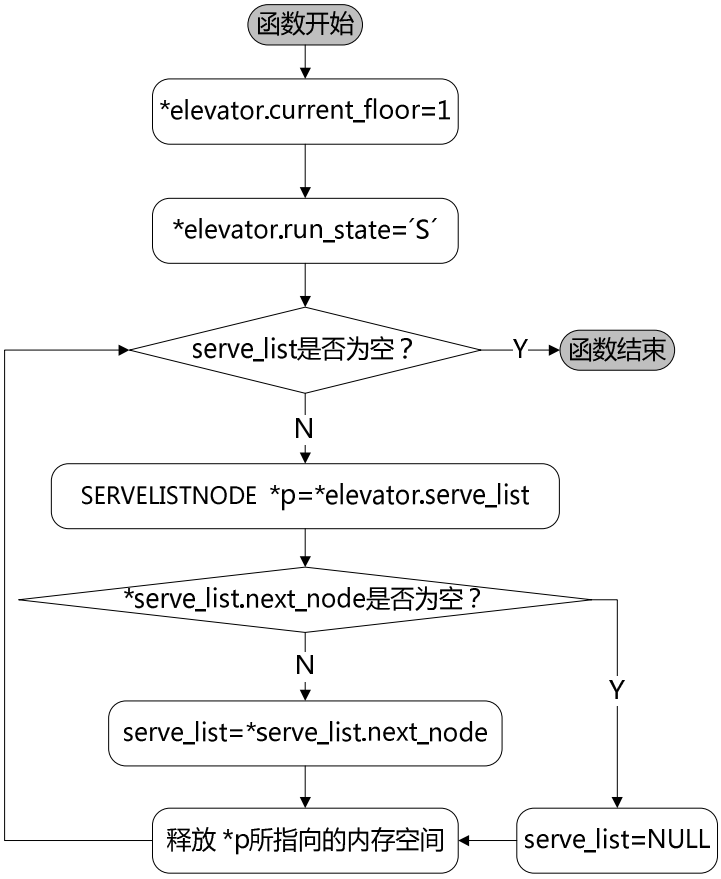

图 4-10 InitElevator 函数流程图

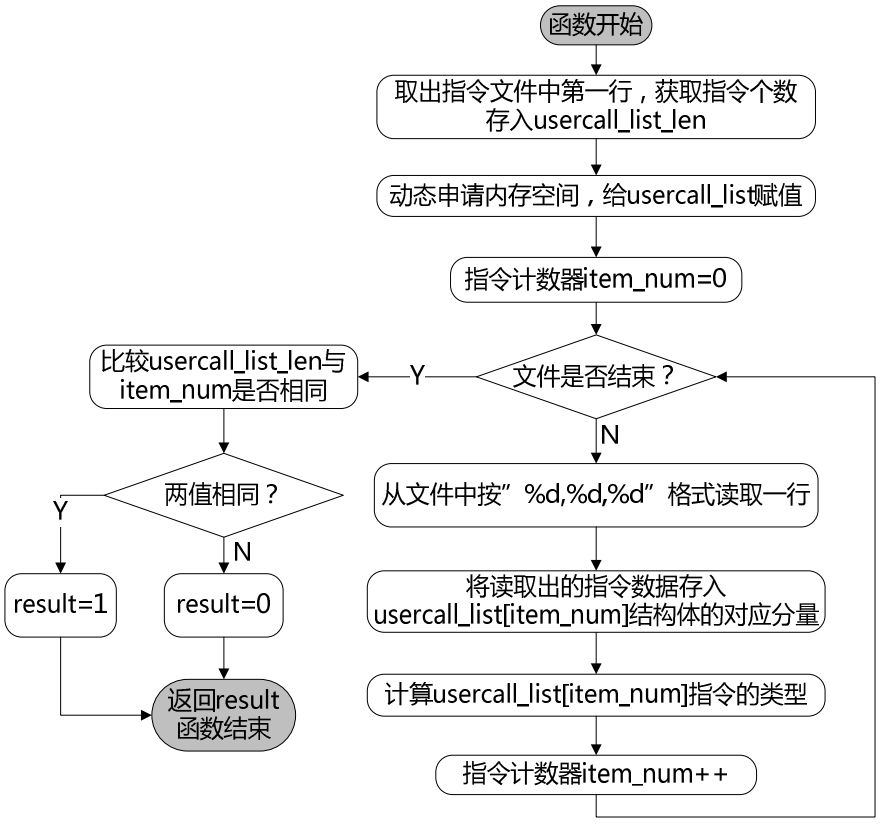

图 4-11 LoadUserCallArray 函数流程图

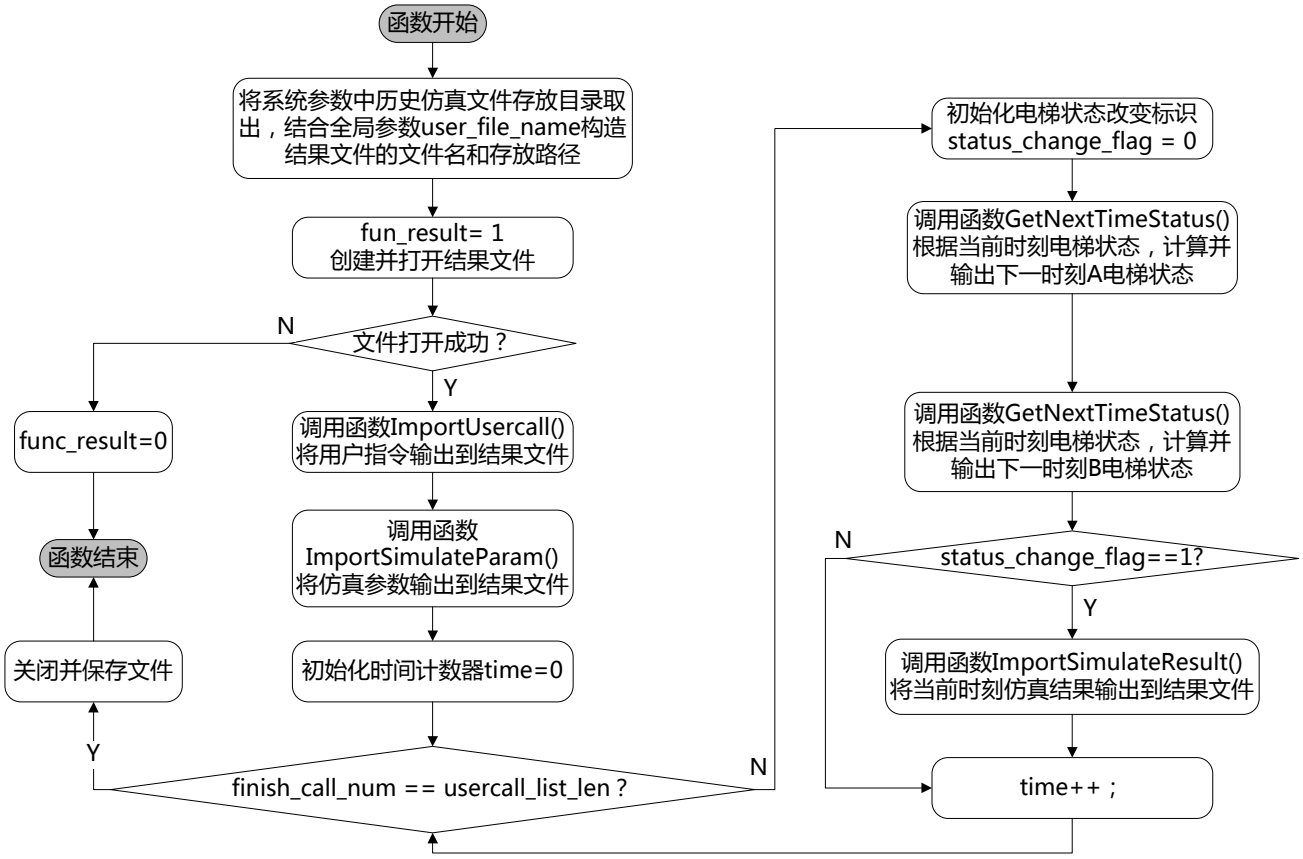

图 4-12 OutputSimulationResult 函数流程图

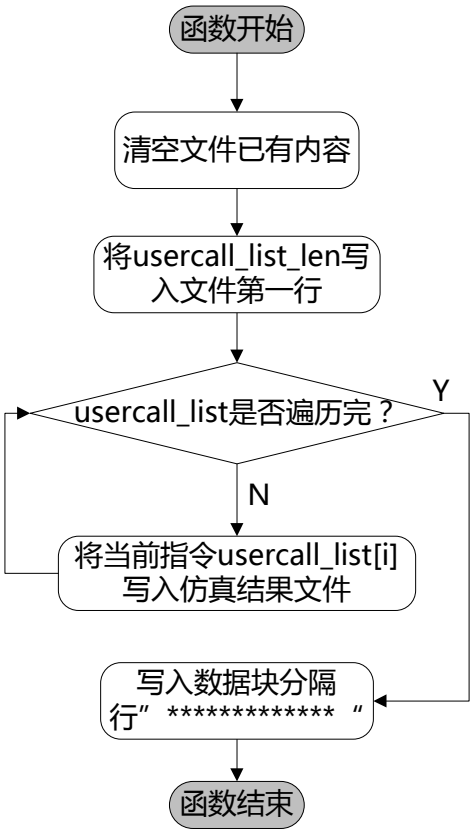

图 4-13 ImportUserCall 函数流程图

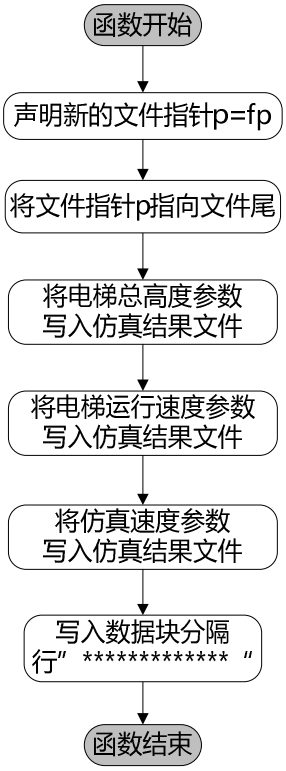

图 4-14 ImportSimulateParam 函数流程图

图 4-15 ImportSimulateResult 函数流程图

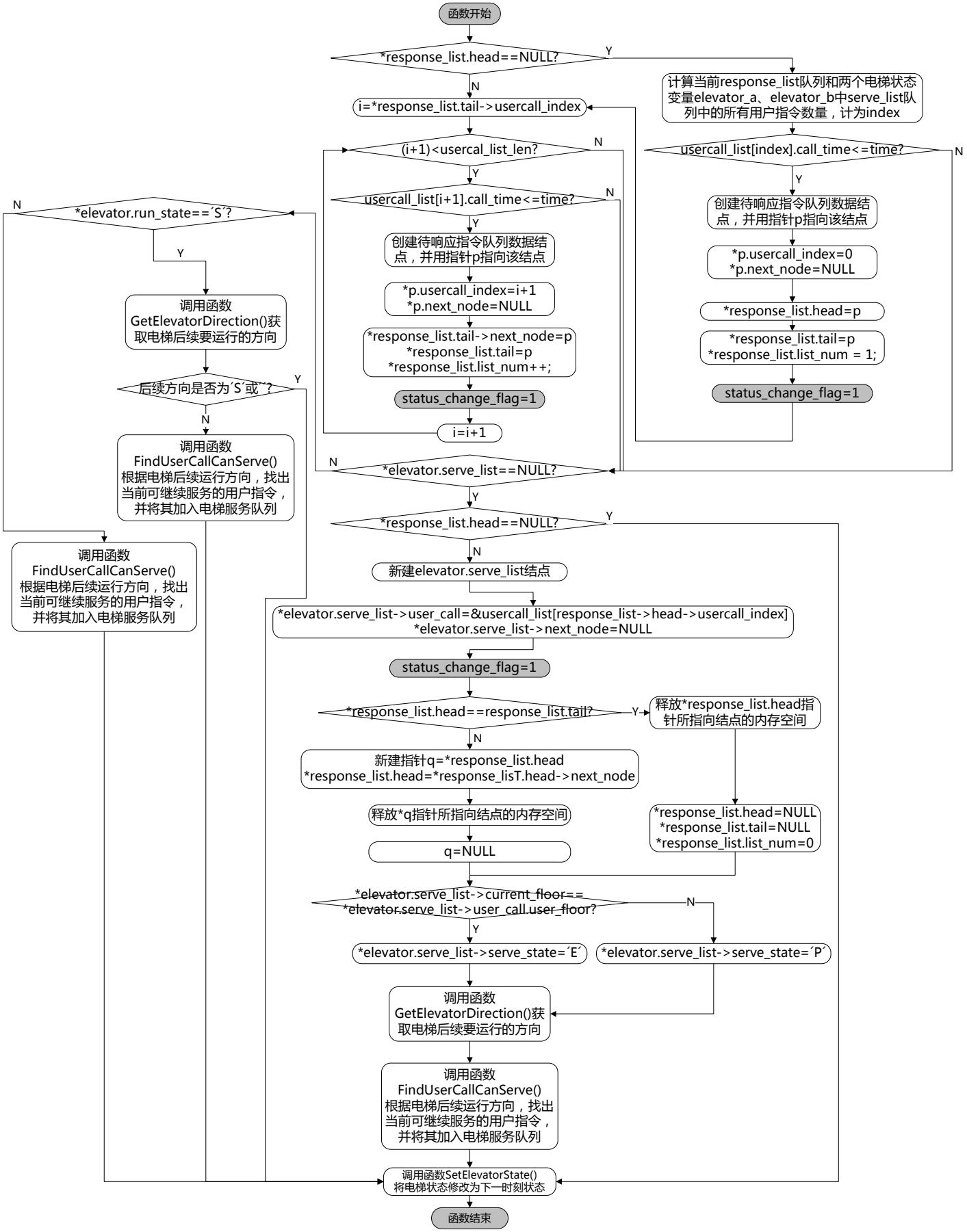

图 4-16 GetNextTimeStatus 函数流程图

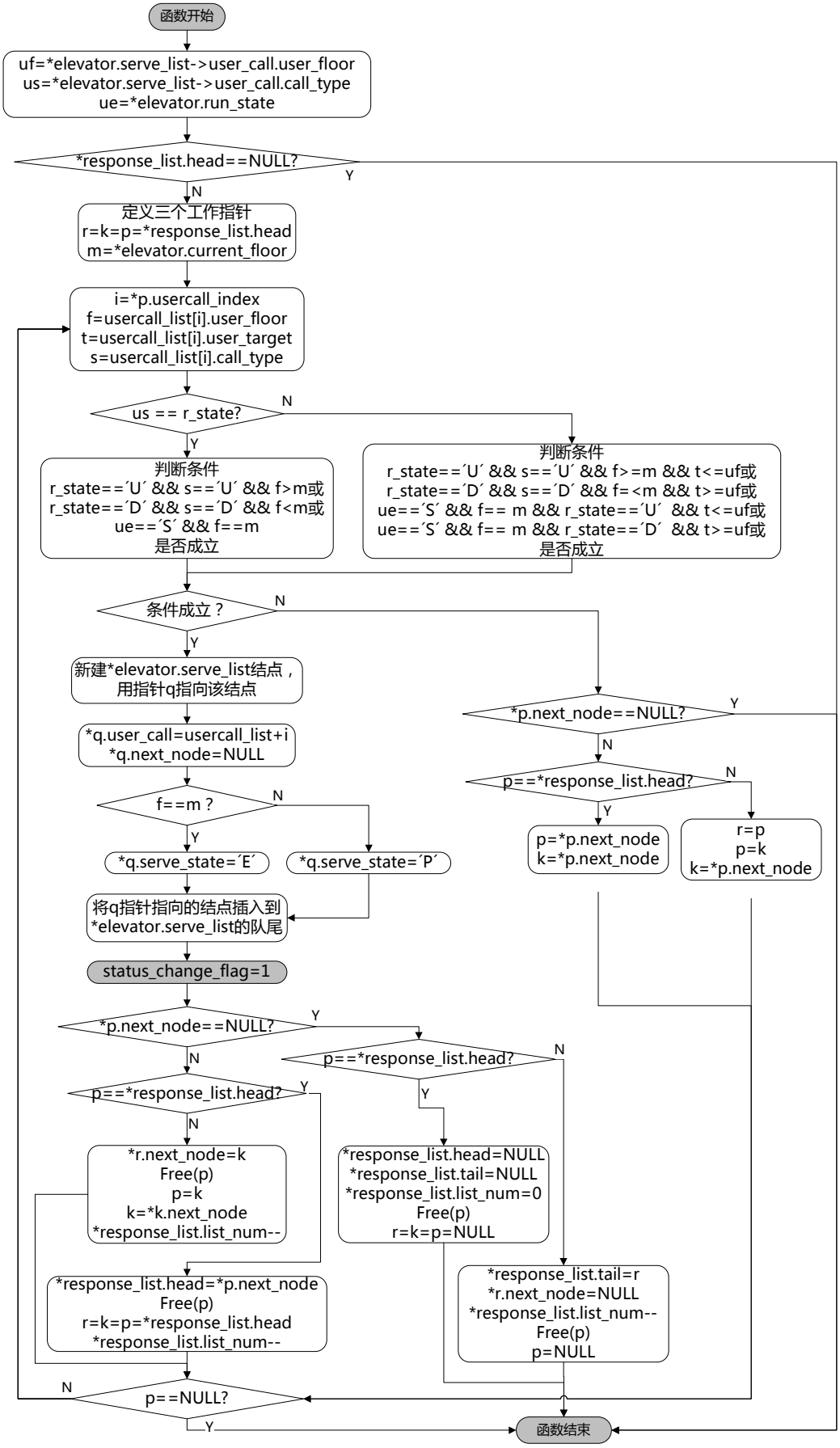

图 4-17 FindUserCallCanServe 函数流程图

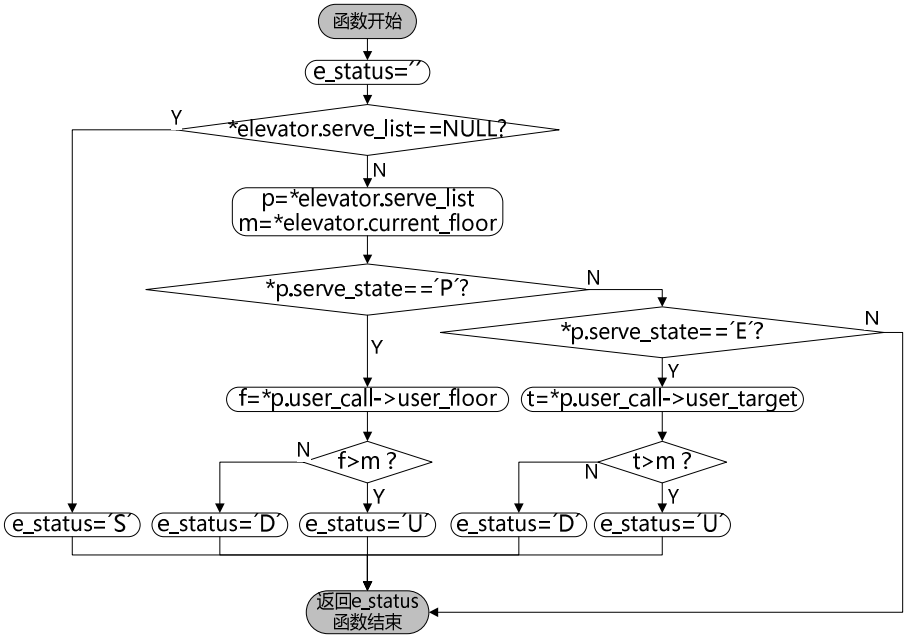

图 4-18 GetElevatorDirection 函数流程图

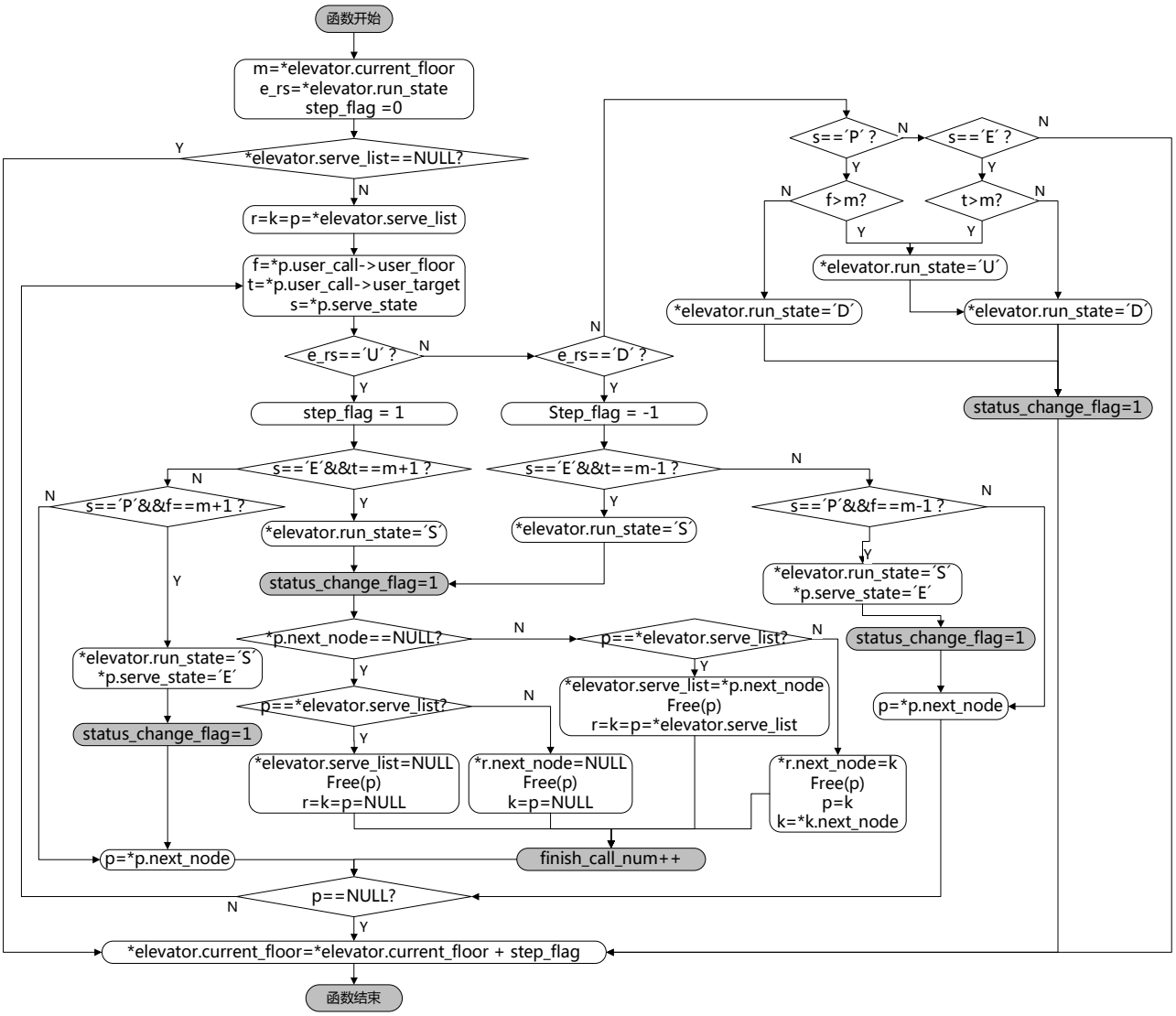

图 4-19 SetElevatorState 函数流程图

4.2.5.动画仿真函数 MovieSimulate

动画仿真函数用于采用动画的方式展现电梯仿真过程。动画仿真函数涉及 3 个函数及子函数，相关函数声明如下：

void MovieSimulate() ;//动画仿真模块函数

void ShowMovie() ;//显示动画仿真过程函数

void PaintPicture() ;//将当前时刻电梯状态绘制成图形的函数

其中 MovieSimulate 函数为模块主函数，无输入参数和返回值，主要实现对模块菜单逻辑的控制和对用户指令文件的读取、参数初始化；ShowMovie 函数主要实现仿真过程的动画展示，无输入参数，无返回值；PaintPicture 函数负责根据当前时刻电梯状态，在屏幕上将当前的电梯状态绘制成图像，函数无输入参数，无返回值。MovieSimulate 函数和 ShowMovie 函数实现过程中，调用了 4.2.4 节的部分子函数。本模块函数流程图如图 4-20、图 4-21、图 4-22 所示。实际动画展示效果见 4.3 节。

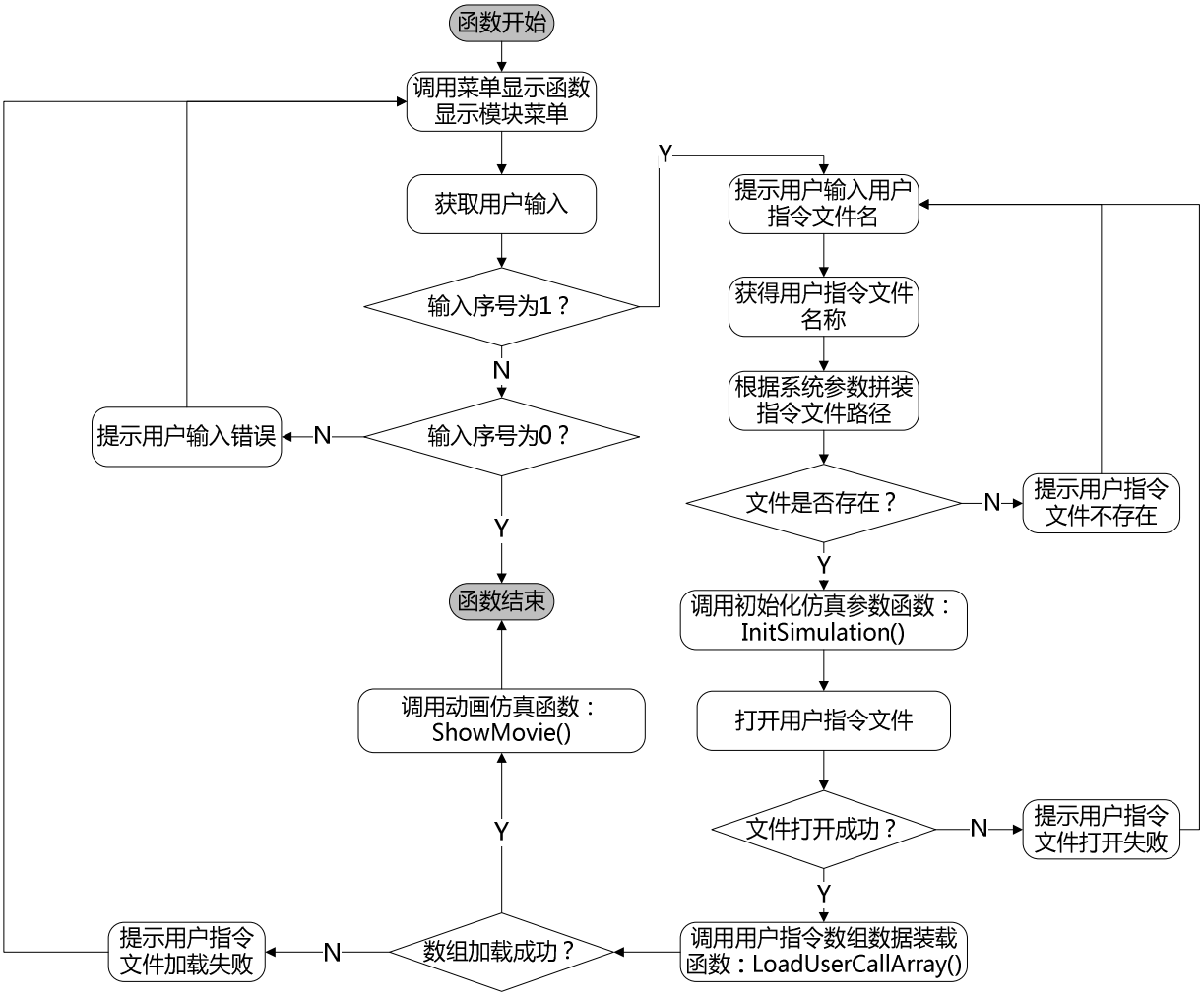

图 4-20 MovieSimulate 函数流程图

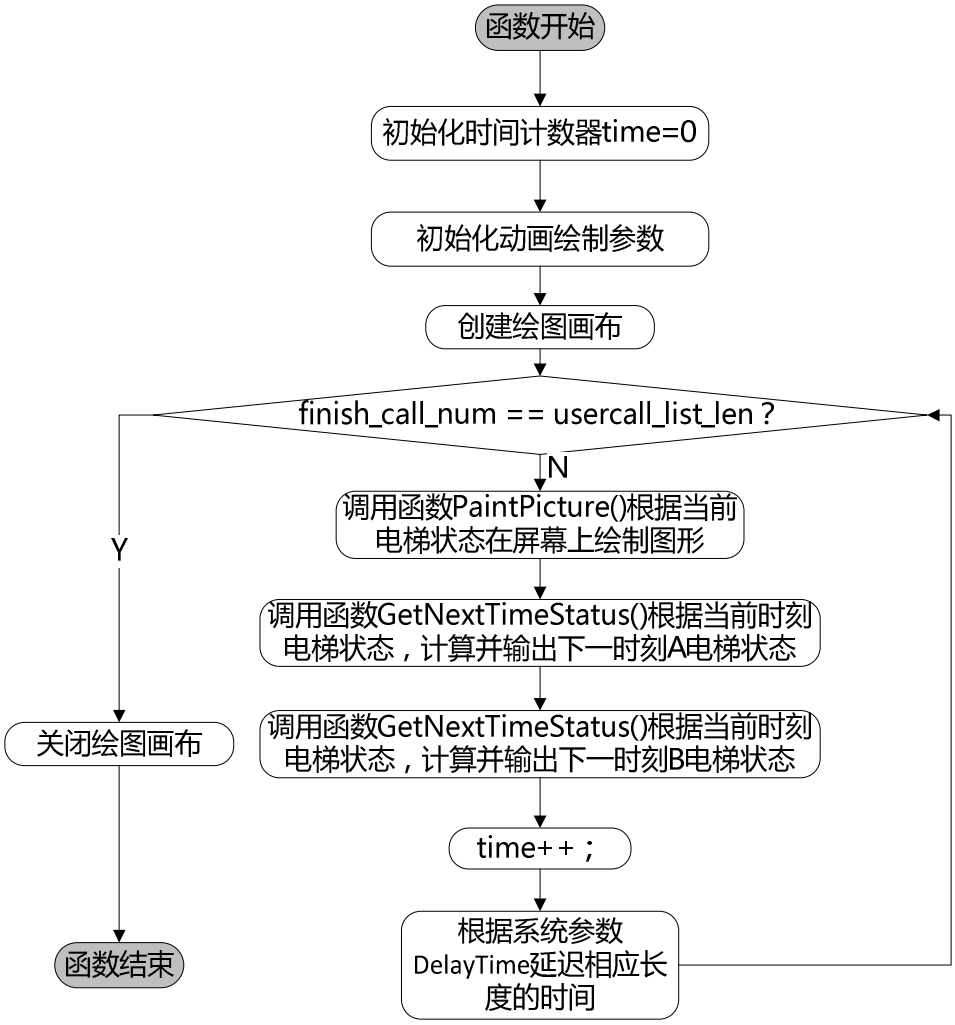

图 4-21 ShowMovie 函数流程图

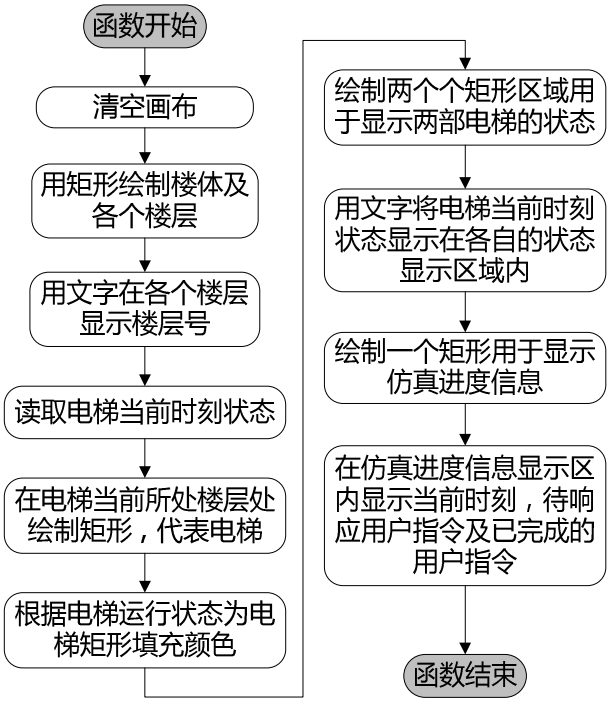

图 4-22 PaintPicture 函数流程图

4.2.6. 全面仿真函数 FullSimulate

全面仿真函数相当于静默仿真和动画仿真的结合体，采用动画的方式展现电梯仿真过程的同时，将仿真结果输出为仿真结果文件。全面仿真函数涉及 2 个函数及子函数，相关函数声明如下：

void FullSimulate() ;//动画仿真模块函数

void ShowMovieAndImport() ;//显示动画仿真过程并输出结果文件的函数

其中 FullSimulate 函数为模块主函数，无输入参数和返回值，主要实现对模块菜单逻辑的控制和对用户指令文件的读取、参数初始化；ShowMovieAndImport 函数主要实现仿真过程的动画展示，并在动画展示过程中将仿真结果输出为结果文件，函数无输入参数，无返回值。FullSimulate 函数和 ShowMovieAndImport 函数实现过程中，调用了 4.2.4 节、4.2.5 节的部分子函数。本模块函数流程图如图 4-23、图 4-24 所示。

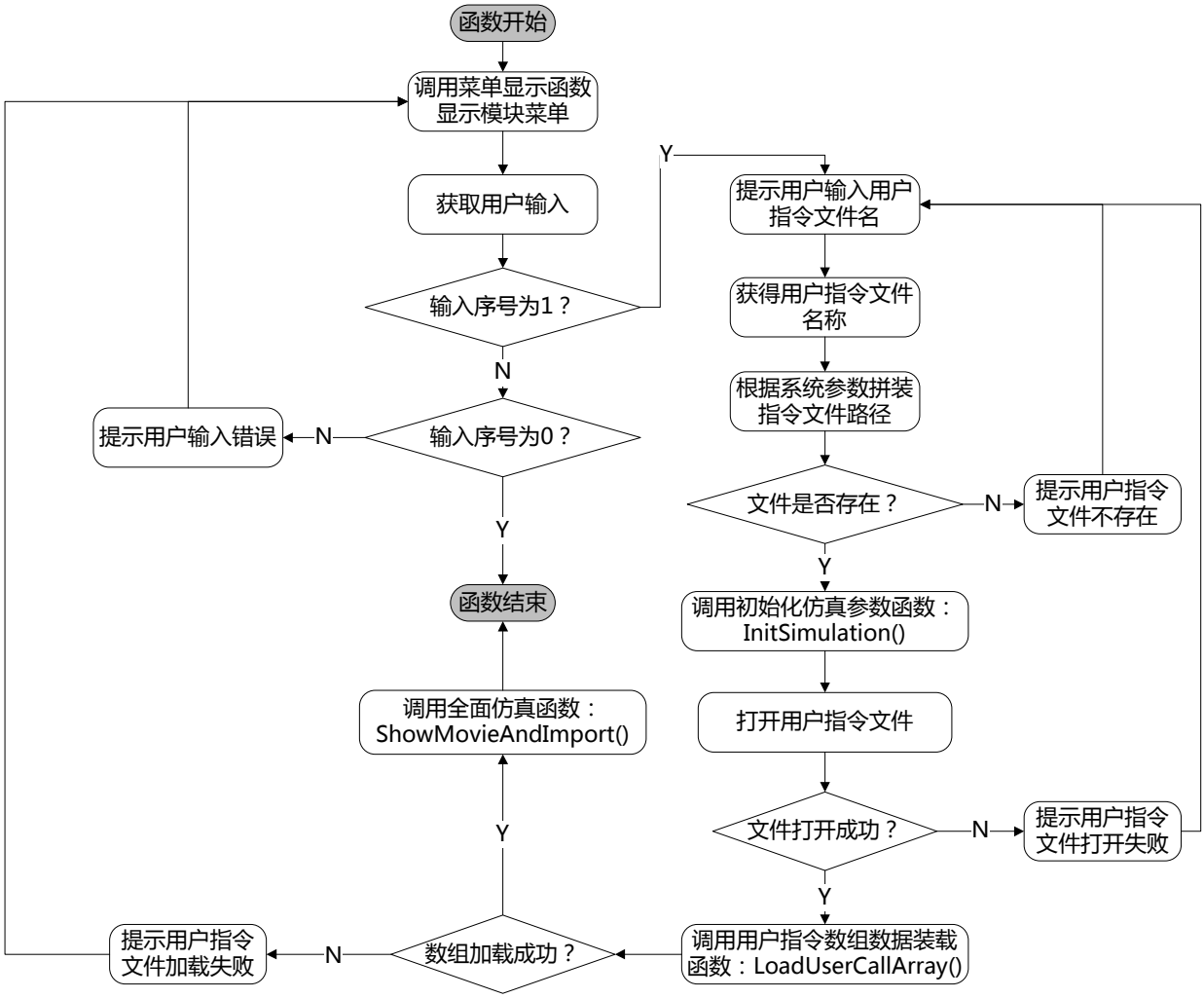

图 4-23 FullSimulate 函数流程图

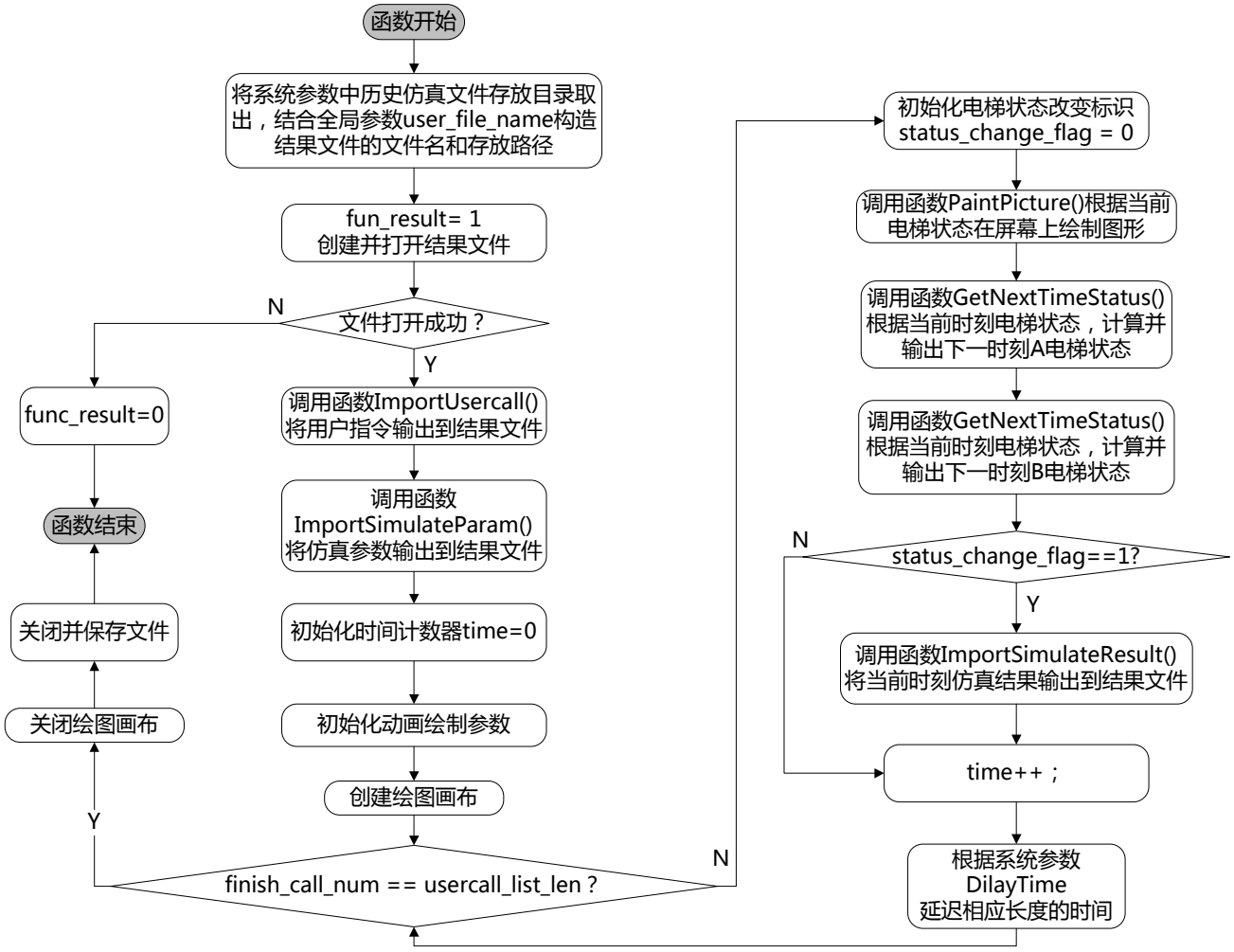

图 4-24 ShowMovieAndImport 函数流程图

4.2.7. 历史仿真文件回放函数 HistorySimulate

历史仿真文件回放函数实现重新加载仿真结果文件，并采用动画的方式展现电梯仿真过程。历史仿真文件回放函数涉及 2 个函数及子函数，相关函数声明如下：

void HistorySimulate() ;//历史仿真文件回放函数

void ShowMovieAndImport() ;//显示动画仿真过程并输出结果文件的函数

其中 FullSimulate 函数为模块主函数，无输入参数和返回值，主要实现对模块菜单逻辑的控制和对用户指令文件的读取、参数初始化；ShowMovieAndImport 函数主要实现仿真过程的动画展示，并在动画展示过程中将仿真结果输出为结果文件，函数无输入参数，无返回值。HistorySimulate 函数实现过程中，调用了 4.2.4 节、4.2.5 节的部分子函数。本模块函数流程图如图 4-25、图 4-26、图 4-27、图 4-28 所示。

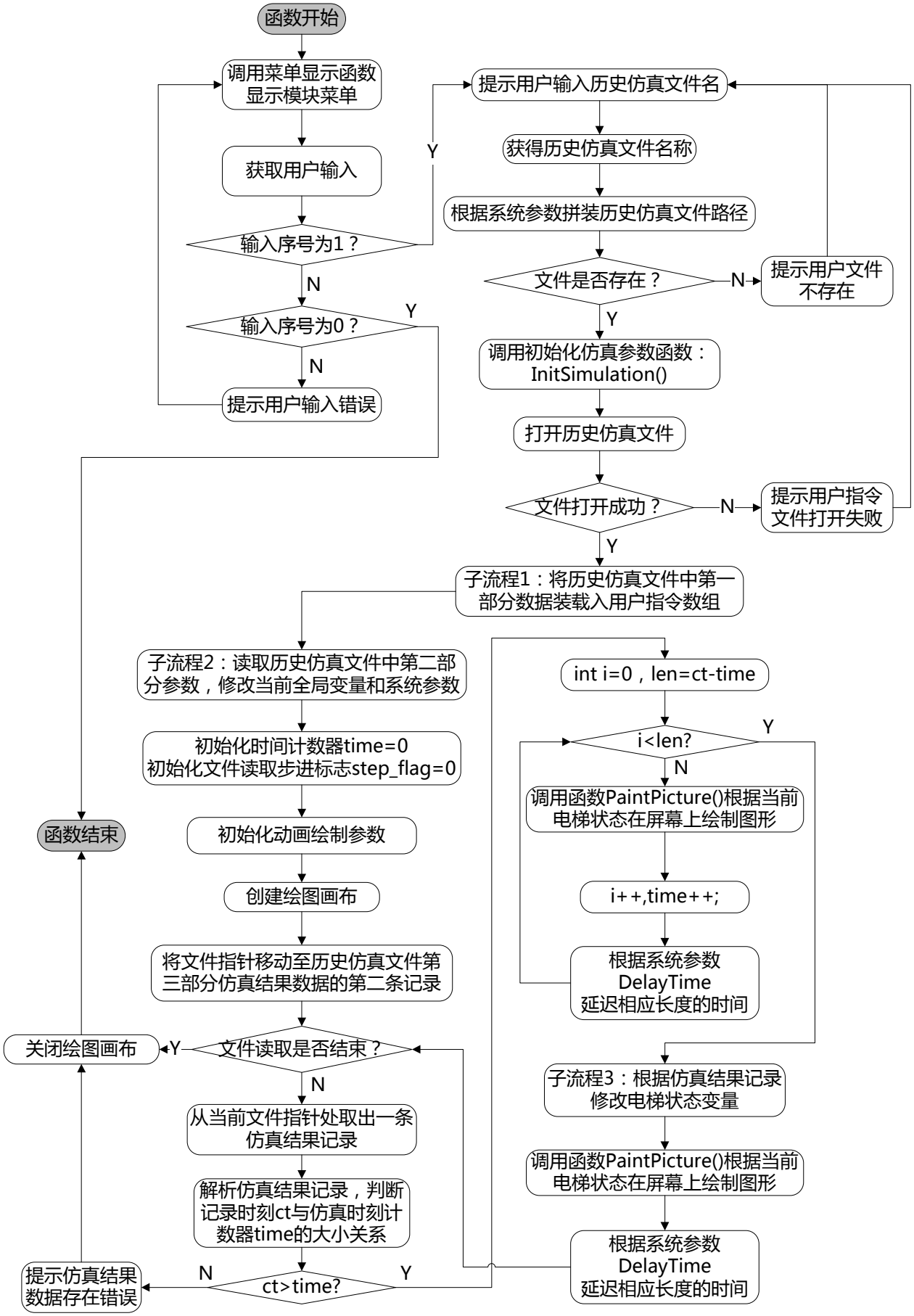

图 4-25 HistorySimulate 函数流程图

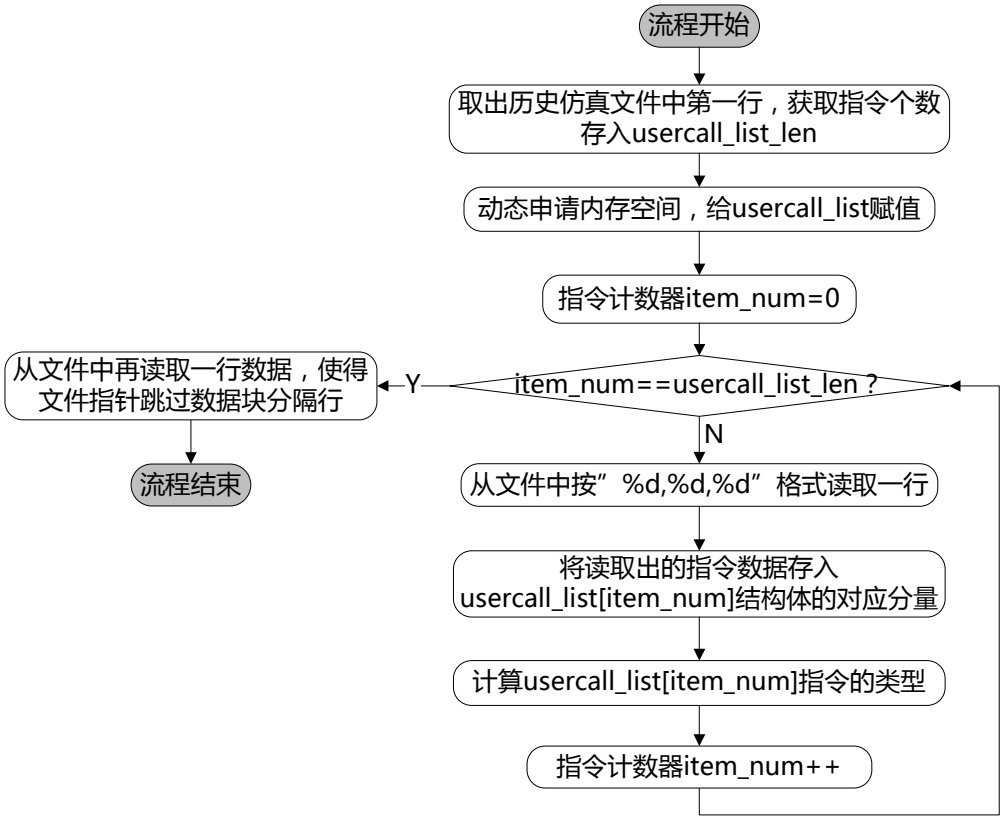

图 4-26 HistorySimulate 函数子流程 1：将历史仿真文件中的用户指令装载至用户指令数组

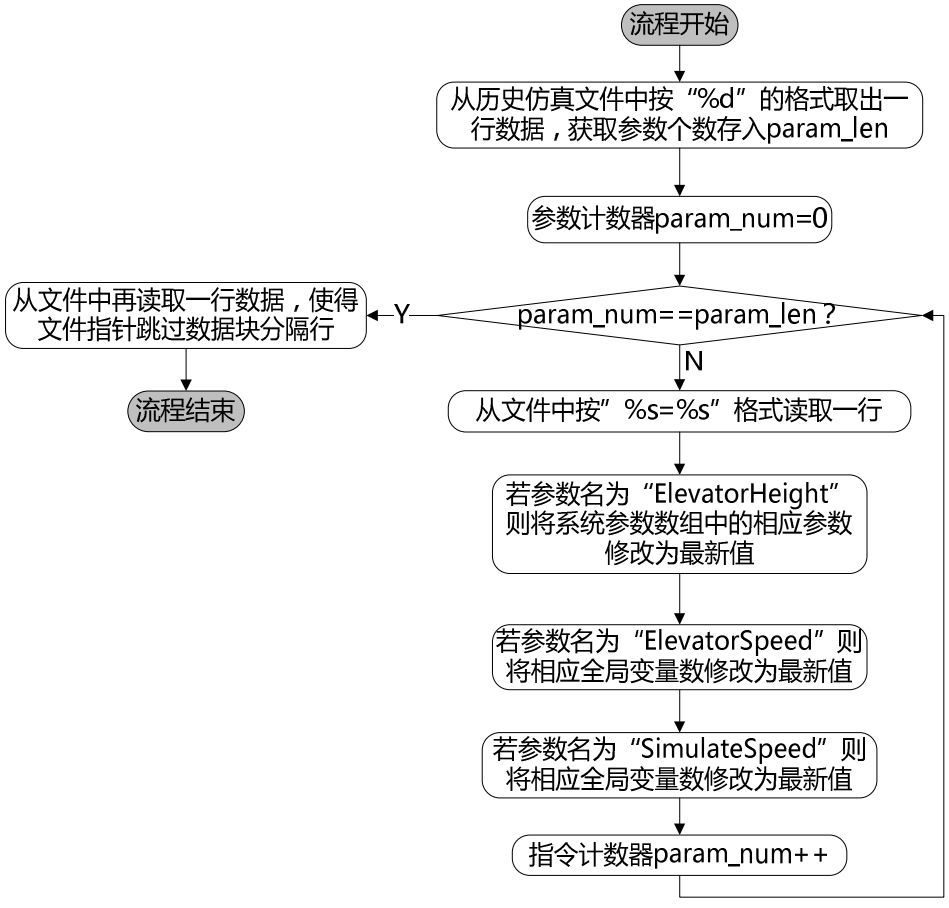

图 4-27 HistorySimulate 函数子流程 2：将历史仿真文件中的仿真参数更新到相关变量

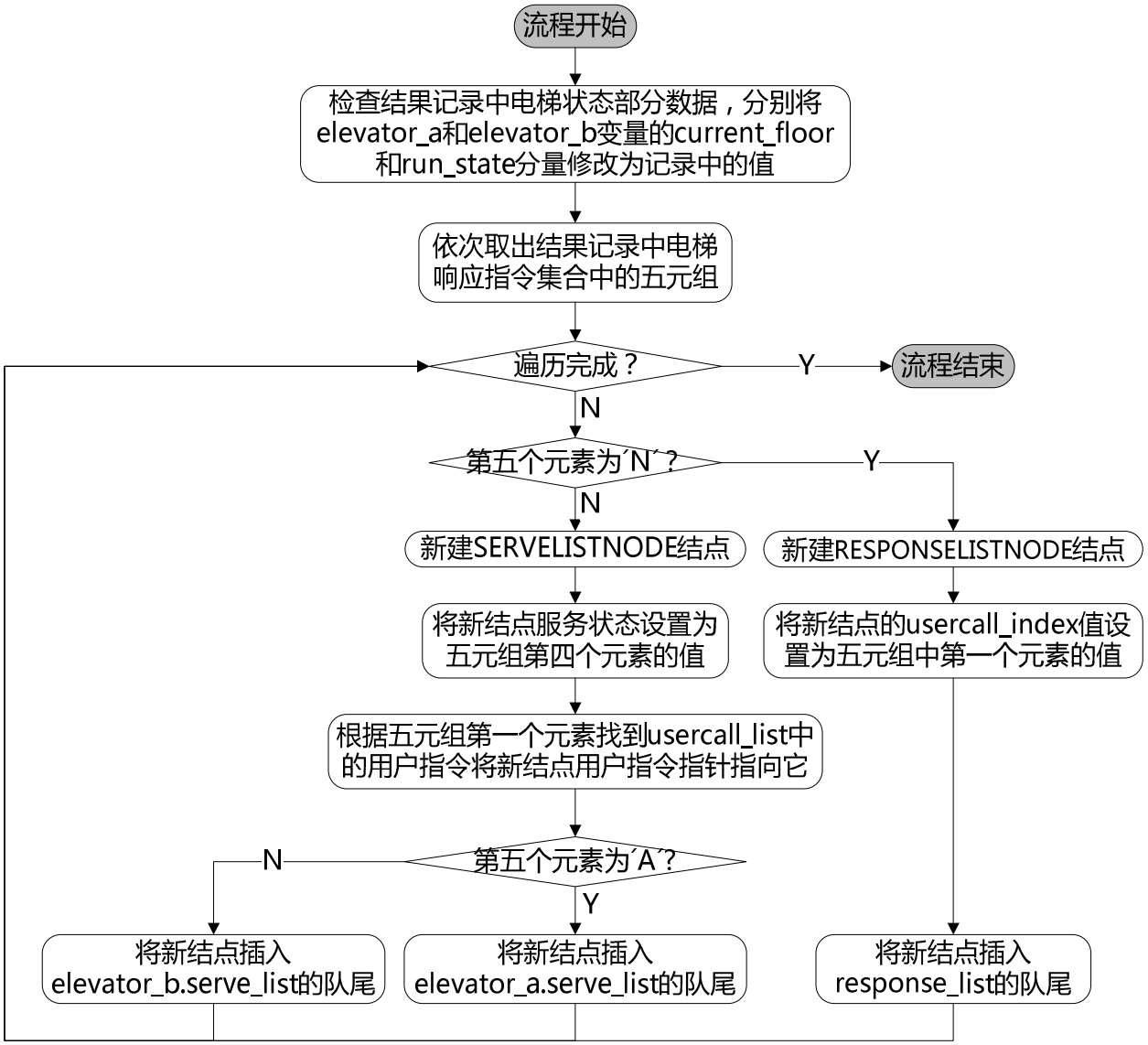

图 4-28 HistorySimulate 函数子流程 3：将历史仿真文件中的仿真参数更新到相关变量

4.2.8. 参数配置函数 ParamConf

参数配置函数实现参数配置模块功能，涉及 1 个函数，相关函数声明如下：

void ParamConf () ;//参数配置函数

ParamConf 函数为模块主函数，无输入参数和返回值，主要实现对模块菜单逻辑的控制和对用户参数的配置，函数流程图如图 4-29 所示。

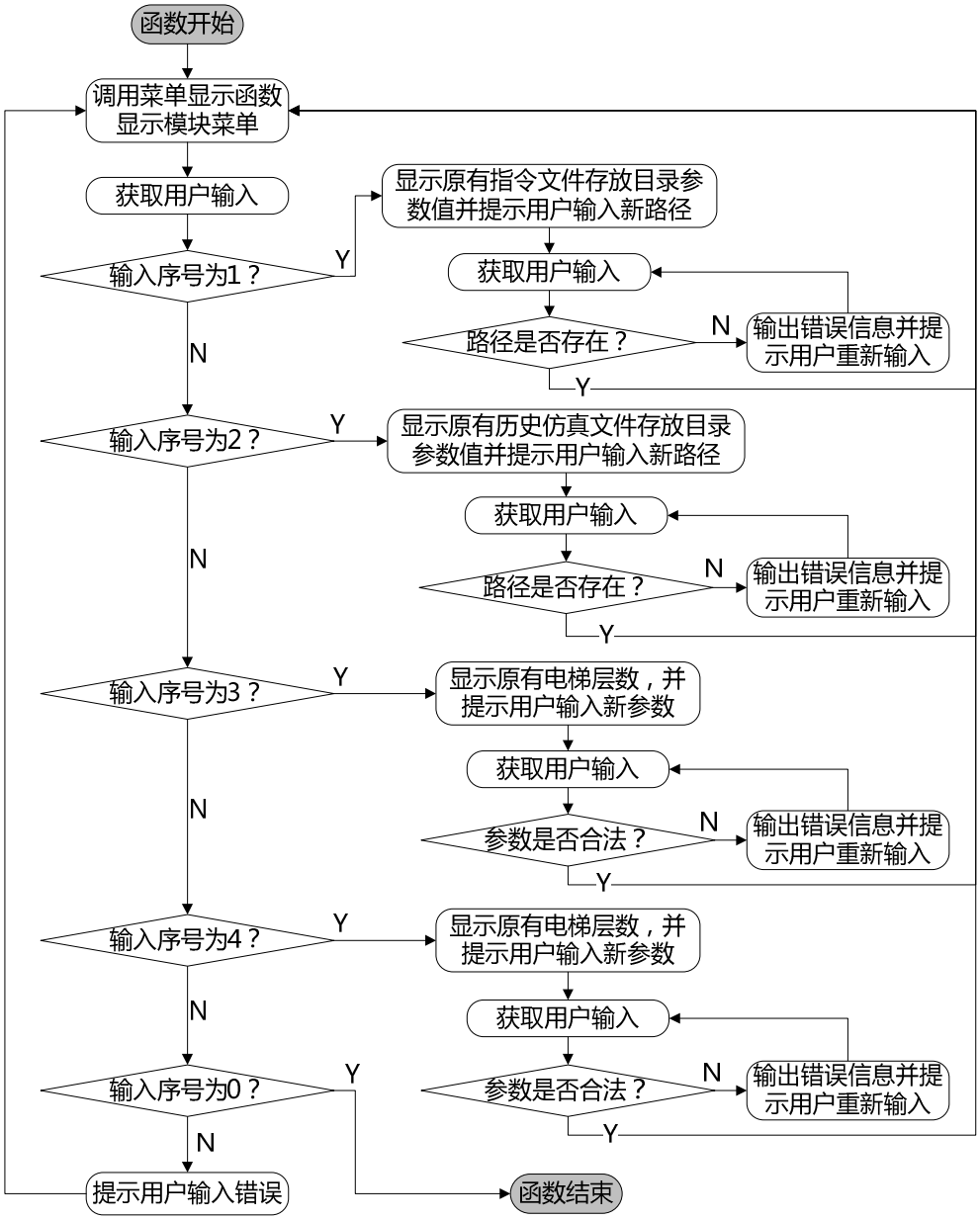

图 4-29 ParamConf 函数流程图

4.3. 用户视图设计

在动画仿真模块、全面仿真模块和历史仿真数据回放模块，均涉及到使用动画方式展现电梯服务过程。因此需要设计一个界面来展现电梯的移动效果，整个界面分为左右两个部分，界面左侧为模拟建筑物及模拟电梯图形区域，主要展示电梯在建筑物内的移动过程；界面右侧为状态信息展示区，分为上下两个部分，右上部分展示电梯的状态，包括电梯当前所处楼层、运行状态、正在服务中的用户指令信息及指令服务状态，右下部分为仿真进度信息，显示仿真当前的时刻和待响应用户指令队列中的用户指令信息。界面效果如图 4-30 所示。

图 4-30 动画仿真界面示意图