

**课 程 设 计 报 告**

年 级 专 业： 电子信息工程6班

学 生 姓 名： 雷子豪

学 号： 20223004514

课 程 名 称： 数据结构课程设计

实 验 名 称： 利用哈希表进行存储

任 课 老 师： 吴泽晖

|  |  |
| --- | --- |
| 实验报告成绩 |  |
| 任课教师签名 |  |

海南大学·信息与通信工程学院

School of Information and Communication Engineering, Hainan University

**目 录**

1. 需求分析 1
2. 概要设计 2
3. 详细设计 3
4. 调试分析 4
5. 运行结果 5
6. 参考文献 6
7. 附录 7

题目：利用哈希表进行存储。

【任务要求】

任务要求:针对一组数据进行初始化哈希表,可以进行显示哈希表,查找元素,插入元素,删除元素,退出程序操作

设计思想:哈希函数用除留余数法构造,用线性探测再散列处理冲突。

设计目的:实现哈希表的综合操作

简体中文控制台界面:用户可以进行创建哈希表,显示哈希表,查找元素,插入元素,删除元素。

显示元素:显示已经创建的哈希表

查找元素:查找哈希表中的元素,分为查找成功和查找不成功

插入元素:在哈希表中,插入一个元素,分为插入成功和失败。

删除元素:在已有的数据中,删除一个元素退出系统:退出程序。

【测试数据】

自行设定,注意边界等特殊情况。

**一.需求分析**

以无歧义的陈述说明程序设计的任务,强调的是程序要做什么?明确规定:

1. 输入的形式和输入值的范围;

* 用户通过控制台界面进行交互操作。
* 输入的值范围为整数。

1. 输出的形式;

* 控制台输出相应的操作结果或错误提示信息。

1. 程序所能达到的功能;
2. 创建哈希表
3. 显示哈希表
4. 查找元素
5. 插入元素
6. 删除元素
7. 退出程序
8. 测试数据:包括正确的输入及其输出结果和含有错误的输入及其输出结果

* 正确的输入及其输出结果
* 含有错误的输入及其输出结果

1. **概要设计**

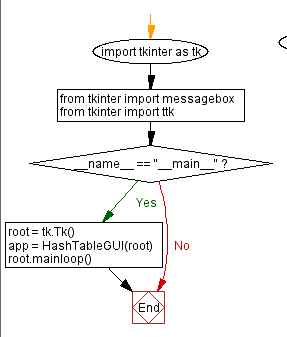
**总体流程图**

1. 创建哈希表

2. 执行用户操作（插入、查找、删除）

3. 更新并显示哈希表

4. 重复用户操作直到用户选择退出



**三．详细设计**

**哈希表设计**

1. 哈希函数（Hash Function）

- 算法：除留余数法

- 作用：将键值映射到哈希表的索引

- 函数：`hash\_function(key)`

2. 线性探测（Linear Probing）

- 算法：线性探测法处理冲突

- 作用：在发生冲突时，探测下一个位置

- 函数：`linear\_probing(key, index)`

3. 懒惰删除（Lazy Deletion）

- 特殊标记：`DELETED`，表示元素被删除的位置

- 作用：在删除元素时标记位置而不是直接置空，以保持探测路径完整

**哈希表操作**

1. 创建哈希表

- 结构：使用大小为 `size` 的列表 `self.table`

- 特殊标记：`self.DELETED = "DELETED"`

2. 插入元素

- 算法：先用哈希函数找到初始位置，如果该位置已被占用则使用线性探测找下一个空位置

- 成功：在空位置插入元素

- 失败：哈希表已满

- 函数：`insert(key)`

3. 查找元素

- 算法：用哈希函数找到初始位置，如果该位置不是目标元素则使用线性探测继续找

- 成功：找到目标元素

- 失败：遍历一圈未找到目标元素

- 函数：`search(key)`

4. 删除元素

- 算法：用哈希函数找到初始位置，如果该位置是目标元素则标记为 `DELETED`，否则使用线性探测继续找

- 成功：找到并标记删除目标元素

- 失败：遍历一圈未找到目标元素

- 函数：`delete(key)`

5. 显示哈希表

- 功能：将哈希表中每个位置及其对应的值显示在图形化界面中

- 函数：`update\_display()`

**存储结构设计**

- 使用Python中的列表 `self.table` 作为哈希表的存储结构

- 列表大小由用户指定，初始化时为 `None`

- 使用特殊标记 `DELETED` 表示被删除的位置

**四.调试分析**

1. 调试过程中遇到的问题是如何解决的以及对设计与实现的回顾讨论和分析

问题一：在哈希表中使用线性探测法处理冲突时，删除元素时会导致查找路径中断，导致无法正确查找到后续的元素。

案例：当我设置哈希表长度为4，并依次插入哈希表数值5，9，13，1.但是当我删除数值9后，再删除1时，其显示元素未找到，删除失败。

分析：这是因为删除元素后并没有正确处理探测路径中的空位。

解决方案

1. 删除标志：引入一个特殊标记`DELETED`，表示元素被删除，而不是直接设为`None`。

2. 哈希表操作：在插入时，遇到`DELETED`标记的位置也可以插入新元素。

3. 删除操作：删除时，将对应位置标记为`DELETED`，而不是设为`None`。

4. 实时显示：在`update\_display`函数中处理`DELETED`标记，显示为“已删除”。

这样处理后，即使删除一个元素，后续的元素也不会受影响，保证哈希表的查找路径完整。

问题二：只有终端输出，无图形化界面，交互性行界面。

分析：寻找可图形化界面的库think。

解决：引入think库。

1. 算法的时空分析(包括基本操作和其他算法的时间复杂度和空间复杂度的分析)和改进设想;

**时间复杂度分析**

1. 哈希函数

- 时间复杂度：`O(1)`

- 原因：计算 `key % size` 仅需常数时间

2. 线性探测

- 平均时间复杂度：`O(1)`

- 最坏时间复杂度：`O(n)`

- 原因：在最坏情况下，可能需要遍历整个哈希表才能找到空位置或确定元素不存在

3. 插入、查找和删除

- 平均时间复杂度：`O(1)`

- 最坏时间复杂度：`O(n)`

- 原因：在最坏情况下，可能需要遍历整个哈希表

**空间复杂度分析**

- 空间复杂度：`O(n)`

- 原因：哈希表使用大小为 `n` 的列表 `self.table`

**改进设想**

1. 再散列（Rehashing）

- 在哈希表接近满时进行再散列，扩大哈希表的大小并重新分配所有元素

- 这样可以减少冲突，提高操作效率

- 平均时间复杂度仍为 `O(1)`，但每次再散列操作的时间复杂度为 `O(n)`

2. 使用更高级的探测方法

- 如二次探测法或双重散列法，可以减少冲突，提高查找效率

3. 优化删除

- 维护一个被删除位置的列表，在插入新元素时优先考虑这些位置，进一步提高效率

1. **经验和体会等**。

在完成本次数据结构课程设计——哈希表的实现过程中，我学到了很多有价值的知识和技能。以下是我的一些经验和体会：

1. 理论与实践结合的重要性

在课堂上学习数据结构的理论知识时，我们掌握了哈希表的基本概念和操作方法，包括哈希函数、冲突处理方法等。然而，真正动手编写哈希表的代码，并实现插入、查找和删除操作，使我对这些概念有了更深刻的理解。在实践中遇到的问题和挑战，帮助我更好地掌握了如何将理论知识应用到实际编程中。

2. 处理冲突的技巧

哈希表最大的挑战之一是如何有效地处理冲突。本次设计中，我采用了线性探测法来处理冲突。通过实现和调试这个过程，我发现线性探测虽然简单，但在哈希表负载因子较高时，性能可能会下降。未来，我会考虑学习和尝试更多高级的冲突处理方法，如二次探测法和双重散列法，以提高哈希表的效率。

3. 删除操作的复杂性

在实现删除操作时，我遇到了如何保证查找路径不被破坏的问题。通过引入“懒惰删除”标记，我成功解决了这个问题，并确保删除操作后哈希表仍能正常工作。这一经验让我认识到，删除操作的实现往往比插入和查找更复杂，需要特别注意数据结构的一致性和完整性。

4. 图形化界面的实现

为了提高程序的用户体验，我使用了Tkinter库实现了哈希表的图形化界面。这是我第一次将数据结构与图形化界面结合在一起，尽管遇到了一些困难，但最终的结果令人满意。通过这种方式，用户可以直观地看到哈希表的变化，使程序更加易用和友好。

5. 调试和测试的重要性

在整个开发过程中，我进行了大量的调试和测试。特别是针对哈希表的边界情况和特殊情况（如哈希表已满、删除非存在元素等），我编写了相应的测试用例。这些测试帮助我发现并修复了不少潜在的错误，确保了程序的稳定性和可靠性。

6. 未来的改进方向

尽管本次设计基本实现了哈希表的基本功能，但仍有很多可以改进的地方。例如，哈希表的再散列（rehashing）功能，更多种类的哈希函数，以及更复杂的冲突处理策略等。未来，我希望能在这些方面进行更多的探索和优化，提高哈希表的性能和实用性。

总的来说，通过本次课程设计，我不仅巩固了数据结构的基础知识，还锻炼了编程和问题解决能力。更加深刻地体会到了从设计到实现、从理论到实践的全过程。这些经验和体会，将对我未来的学习和工作产生积极的影响。

**五.运行结果**

说明如何使用你编写的程序,详细列出每一步的操作步骤，列出你的测试结果,包括输入和输出，抓图说明。

1.各个功能截图

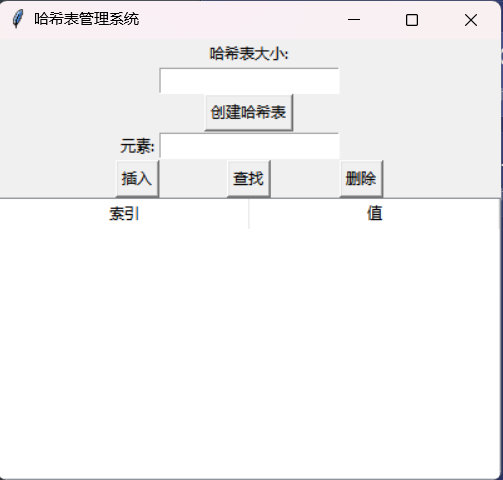
2. 界面说明

程序启动后，会显示一个图形化界面，包括以下几个部分：

- 哈希表大小输入框和创建按钮

- 元素输入框及插入、查找、删除按钮

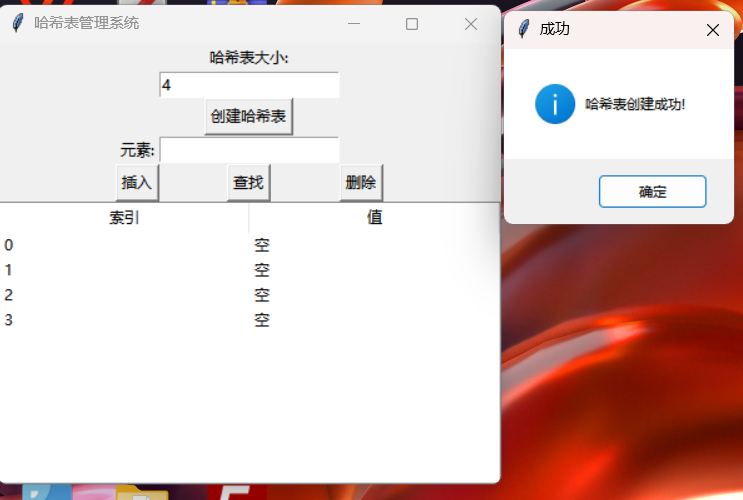
- 哈希表的图形化显示区域



3. 创建哈希表

- 在“哈希表大小”输入框中输入一个正整数（例如`4`），表示哈希表的大小。

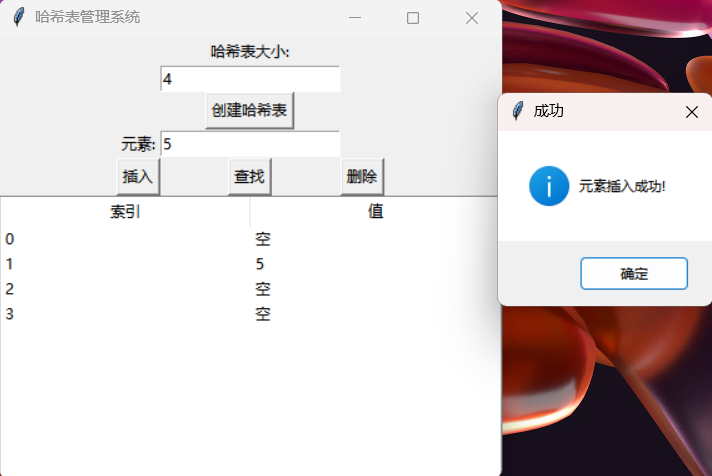
- 点击“创建哈希表”按钮，程序会根据输入的大小初始化一个空的哈希表。



4. 插入元素

- 在“元素”输入框中输入一个整数（例如`5`）。

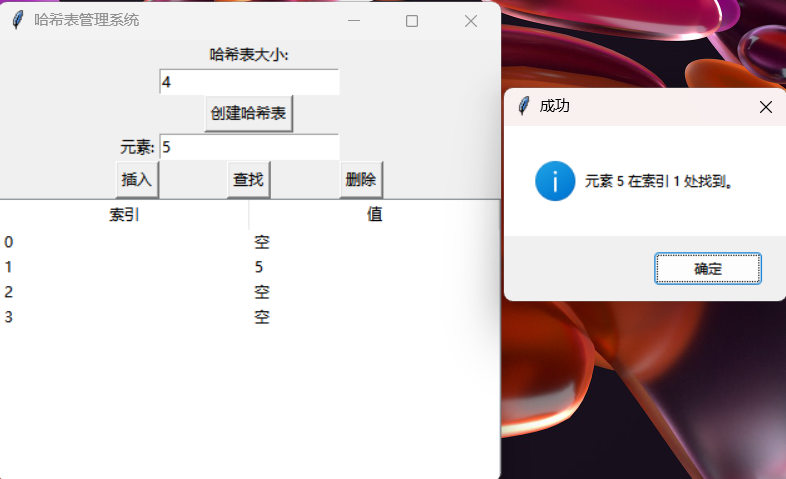
- 点击“插入”按钮，程序会将该元素插入到哈希表中，并更新显示区域。



5. 查找元素

- 在“元素”输入框中输入要查找的整数（例如`5`）。

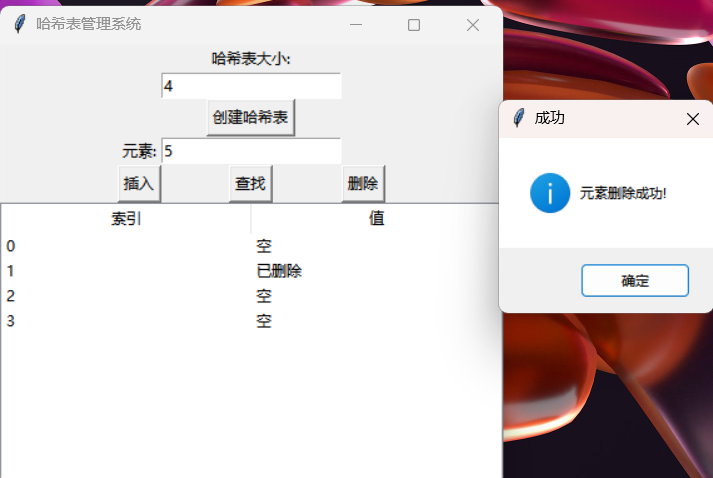
- 点击“查找”按钮，程序会在哈希表中查找该元素，并弹出对话框显示查找结果。



6. 删除元素

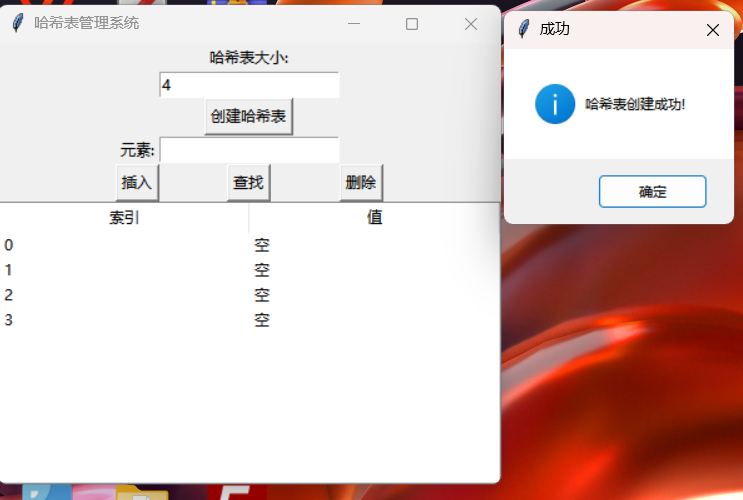
- 在“元素”输入框中输入要删除的整数（例如`5`）。删除元素9，删除元素1

- 点击“删除”按钮，程序会在哈希表中删除该元素，并更新显示区域。

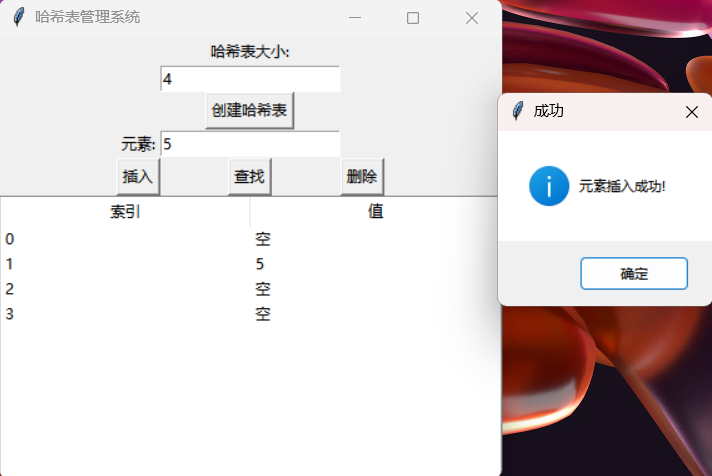


**以下是详细的使用示例：**

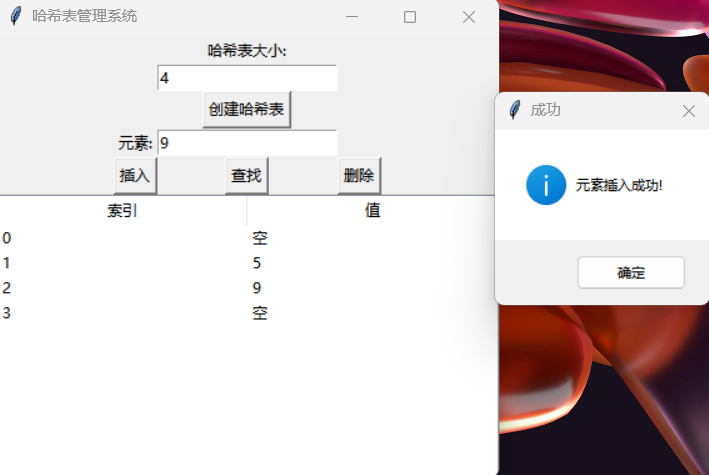
1. 创建哈希表大小为4



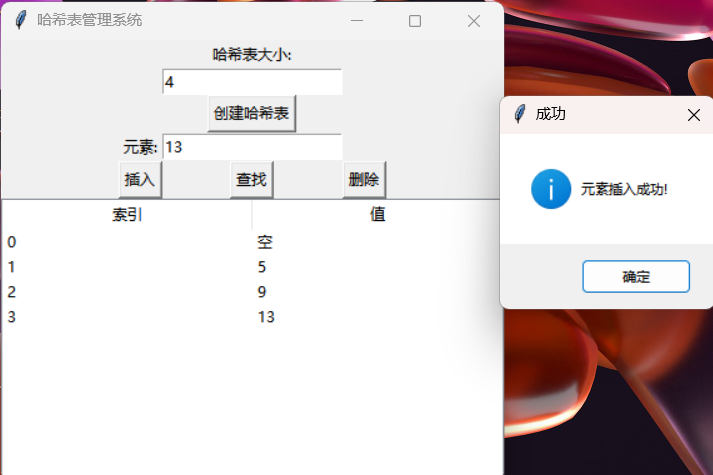
1. 插入元素5



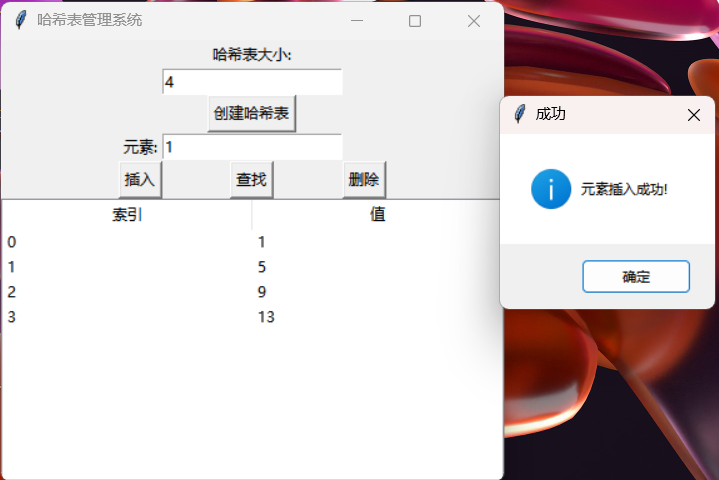
1. 插入元素9



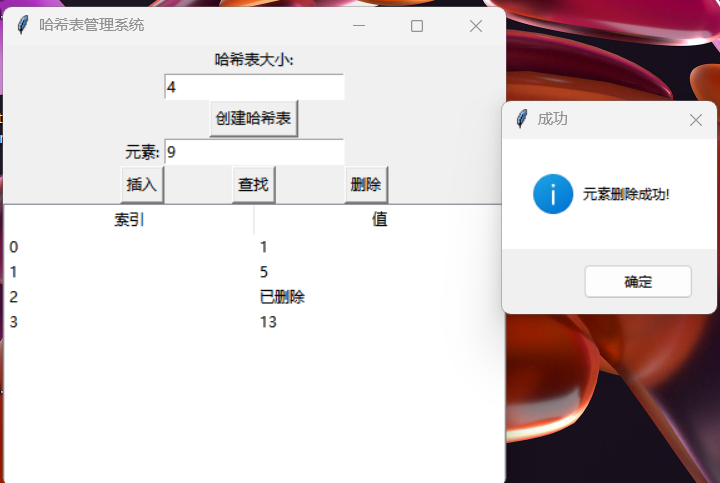
1. 插入元素13



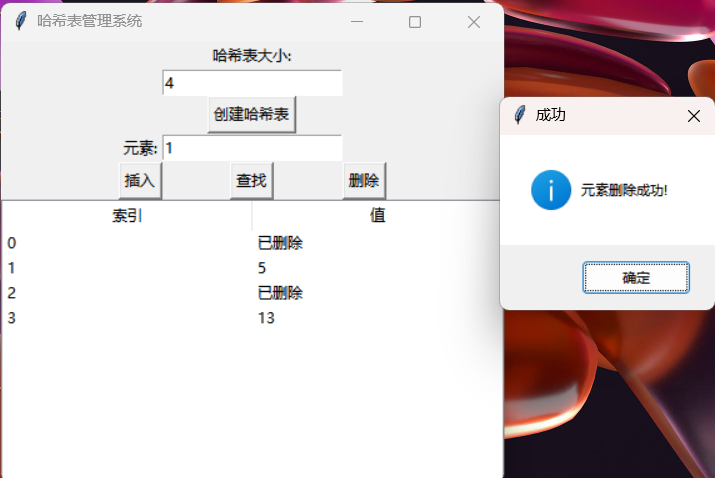
5.插入元素1



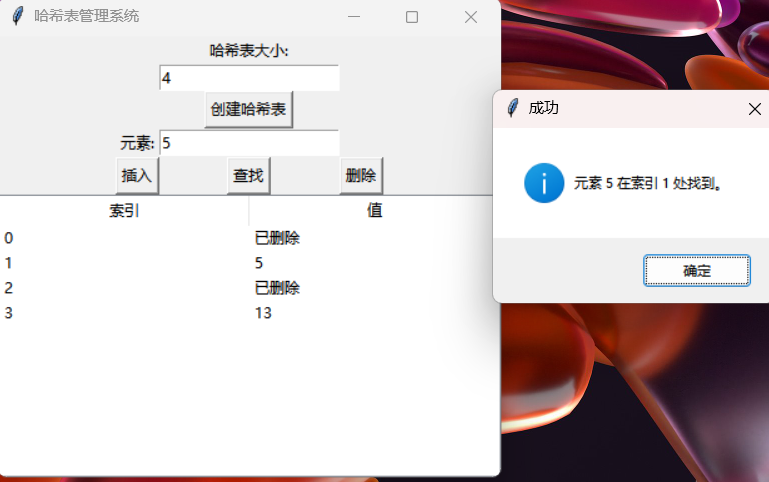
6. 删除元素9



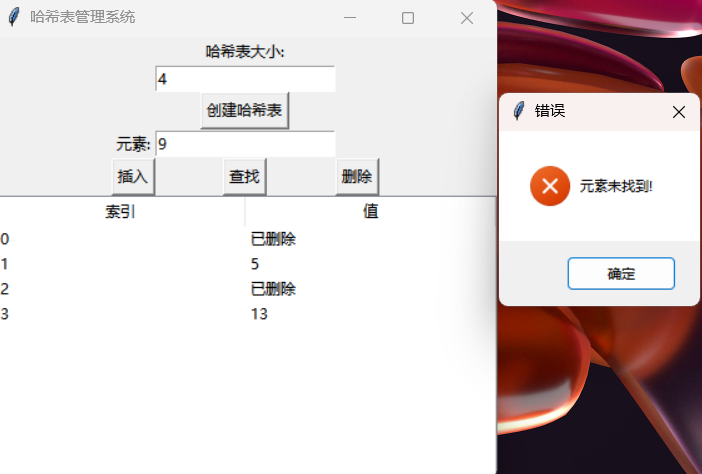
7. 删除元素1



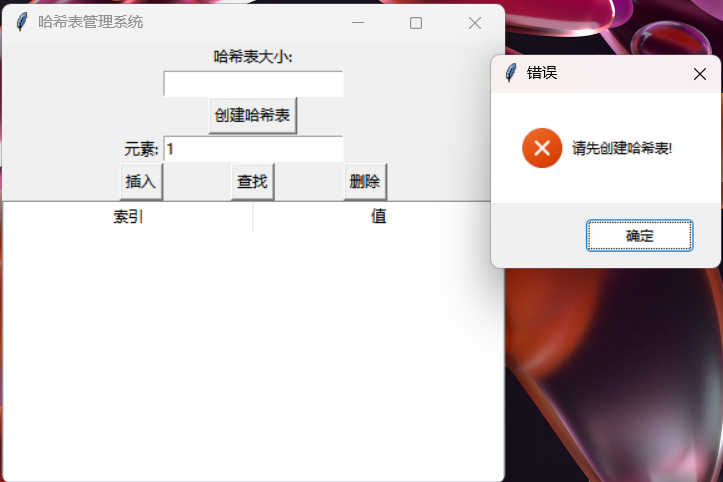
8. 查找元素5

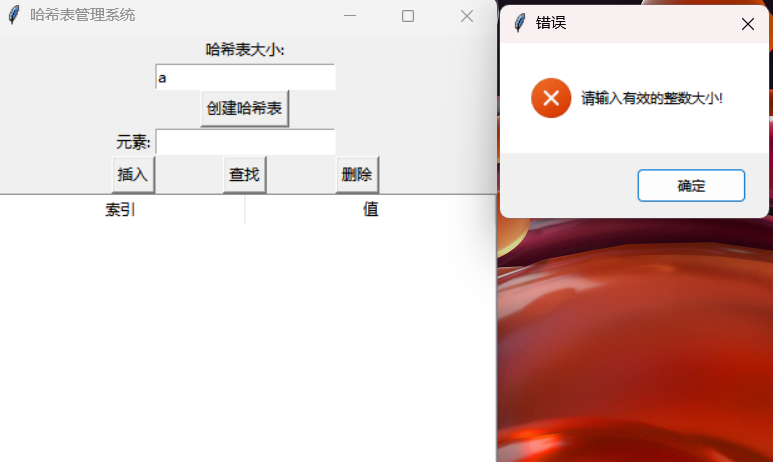


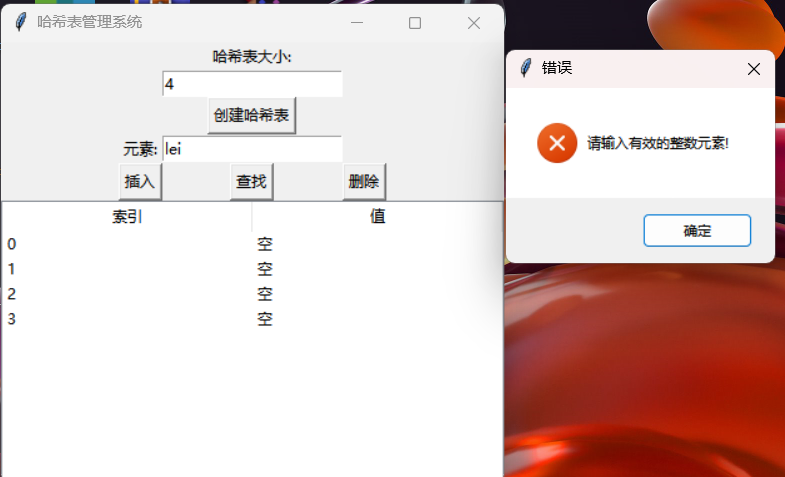
9. 查找元素9

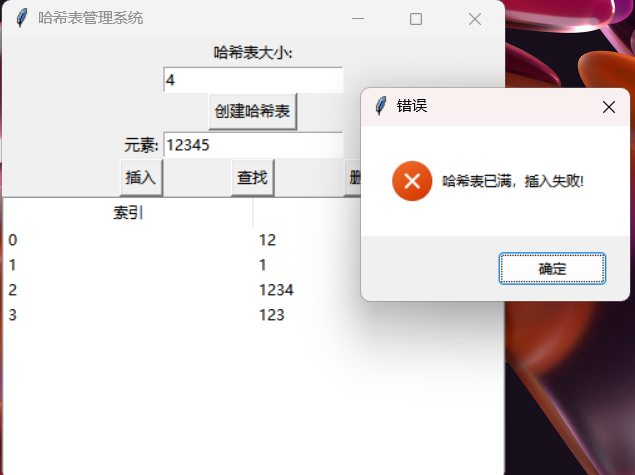


**异常值处理：**









1. **参考文献**

1. Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). Introduction to Algorithms (3rd ed.). MIT Press.

2. Sedgewick, R., & Wayne, K. (2011). Algorithms, Part I. Addison-Wesley.

3. Weiss, M. A. (2014). Data Structures and Algorithm Analysis in C++ (4th ed.). Pearson.

4. Knuth, D. E. (1998). The Art of Computer Programming, Volume 3: Sorting and Searching (2nd ed.). Addison-Wesley.

5. Pagh, R., & Rodler, F. F. (2001). Performance Analysis of Hashing Algorithms for Network Applications. Proceedings of the 10th Annual ACM-SIAM Symposium on Discrete Algorithms.

1. **附录**

import tkinter as tk

from tkinter import messagebox

from tkinter import ttk

class HashTable:

def \_\_init\_\_(self, size):

self.size = size

self.table = [None] \* size 初始化哈希表

self.DELETED = "DELETED" 特殊标记，表示已删除的位置

def hash\_function(self, key):

return key % self.size 除留余数法构造哈希函数

def linear\_probing(self, key, index):

return (index + 1) % self.size 线性探测再散列处理冲突

def display(self):

return [(i, self.table[i]) for i in range(self.size)]

def search(self, key):

index = self.hash\_function(key)

start\_index = index

while self.table[index] is not None:

if self.table[index] == key:

return index

index = self.linear\_probing(key, index)

if index == start\_index:

break

return -1

def insert(self, key):

index = self.hash\_function(key)

if self.table[index] is None or self.table[index] == self.DELETED:

self.table[index] = key

return True

else:

start\_index = index

index = self.linear\_probing(key, index)

while index != start\_index and self.table[index] is not None and self.table[index] != self.DELETED:

index = self.linear\_probing(key, index)

if index != start\_index:

self.table[index] = key

return True

else:

return False

def delete(self, key):

index = self.hash\_function(key)

start\_index = index

while self.table[index] is not None:

if self.table[index] == key:

self.table[index] = self.DELETED

return True

index = self.linear\_probing(key, index)

if index == start\_index:

break

return False

GUI部分

class HashTableGUI:

def \_\_init\_\_(self, root):

self.root = root

self.root.title("哈希表管理系统")

self.size\_label = tk.Label(root, text="哈希表大小:")

self.size\_label.pack()

self.size\_entry = tk.Entry(root)

self.size\_entry.pack()

self.create\_button = tk.Button(root, text="创建哈希表", command=self.create\_hash\_table)

self.create\_button.pack()

self.operation\_frame = tk.Frame(root)

self.operation\_frame.pack()

self.key\_label = tk.Label(self.operation\_frame, text="元素:")

self.key\_label.grid(row=0, column=0)

self.key\_entry = tk.Entry(self.operation\_frame)

self.key\_entry.grid(row=0, column=1)

self.insert\_button = tk.Button(self.operation\_frame, text="插入", command=self.insert\_key)

self.insert\_button.grid(row=1, column=0)

self.search\_button = tk.Button(self.operation\_frame, text="查找", command=self.search\_key)

self.search\_button.grid(row=1, column=1)

self.delete\_button = tk.Button(self.operation\_frame, text="删除", command=self.delete\_key)

self.delete\_button.grid(row=1, column=2)

self.hash\_table\_view = ttk.Treeview(root, columns=("Index", "Value"), show="headings")

self.hash\_table\_view.heading("Index", text="索引")

self.hash\_table\_view.heading("Value", text="值")

self.hash\_table\_view.pack()

self.hash\_table = None

def create\_hash\_table(self):

try:

size = int(self.size\_entry.get())

self.hash\_table = HashTable(size)

self.update\_display()

messagebox.showinfo("成功", "哈希表创建成功!")

except ValueError:

messagebox.showerror("错误", "请输入有效的整数大小!")

def insert\_key(self):

if self.hash\_table is None:

messagebox.showerror("错误", "请先创建哈希表!")

return

try:

key = int(self.key\_entry.get())

if self.hash\_table.insert(key):

self.update\_display()

messagebox.showinfo("成功", "元素插入成功!")

else:

messagebox.showerror("错误", "哈希表已满，插入失败!")

except ValueError:

messagebox.showerror("错误", "请输入有效的整数元素!")

def search\_key(self):

if self.hash\_table is None:

messagebox.showerror("错误", "请先创建哈希表!")

return

try:

key = int(self.key\_entry.get())

index = self.hash\_table.search(key)

if index != -1:

messagebox.showinfo("成功", f"元素 {key} 在索引 {index} 处找到。")

else:

messagebox.showerror("错误", "元素未找到!")

except ValueError:

messagebox.showerror("错误", "请输入有效的整数元素!")

def delete\_key(self):

if self.hash\_table is None:

messagebox.showerror("错误", "请先创建哈希表!")

return

try:

key = int(self.key\_entry.get())

if self.hash\_table.delete(key):

self.update\_display()

messagebox.showinfo("成功", "元素删除成功!")

else:

messagebox.showerror("错误", "元素未找到，删除失败!")

except ValueError:

messagebox.showerror("错误", "请输入有效的整数元素!")

def update\_display(self):

if self.hash\_table is not None:

for i in self.hash\_table\_view.get\_children():

self.hash\_table\_view.delete(i)

for index, value in self.hash\_table.display():

if value is None:

value\_str = "空"

elif value == self.hash\_table.DELETED:

value\_str = "已删除"

else:

value\_str = str(value)

self.hash\_table\_view.insert("", "end", values=(index, value\_str))

if \_\_name\_\_ == "\_\_main\_\_":

root = tk.Tk()

app = HashTableGUI(root)

root.mainloop()

一、上交要求：

　　上交的成果的内容必须由以下四个部分组成，缺一不可

　　1． 上交源程序：学生按照课程设计的具体要求所开发的所有源程序（应该放到一个文件夹中）；

　　2． 上交程序的说明文件：（保存在.txt中）在说明文档中应该写明上交程序所在的目录，上交程序的主程序文件名，如果需要安装，要有程序的安装使用说明；

　　3． 课程设计报告：（保存在word 文档中，文件名要求 按照"姓名-学号-课程设计报告"起名，如文件名为"张三-001-课程设计报告".doc ）按照课程设计的具体要求建立的功能模块，每个模块要求按照如下几个内容认真完成；

　　其中包括: