短学期 c 语言项目设计报告书

小组编号：26

项目名称：猫咪喂食器

学号：2335060506

姓名：杨紫雯

学院：基础学院

专业：工科试验班

小组成员：申佳依（2335060506），郑世婷（2335060508），杨紫雯（2335060507）

任课教师：林剑

2024 年 7 月 8 日

目录

简介

1.1 作品创意/项目背景

随着现代家庭对宠物生活质量的日益重视，如何科学、合理地照顾宠物成为了许多宠物主人关注的焦点。猫咪作为家庭中最常见的宠物之一，其饮食习惯和营养需求对它们的健康至关重要。然而，由于工作繁忙或生活节奏加快，许多宠物主人难以保证每天按时、按量地为猫咪喂食，这可能导致猫咪饮食不规律，进而影响其健康状况。

在此背景下，我们萌生了设计一款基于 C 语言的猫咪喂食器的创意。这款喂食器旨在通过智能化、自动化的方式，帮助宠物主人解决猫咪喂食的难题，确保猫咪能够按时、按量地进食，从而保持健康的体魄和愉悦的心情。

该项目的主要功能包括时间和喂食量预设功能，自动喂食功能，手动喂食功能，喂食提醒功能，数据持久化与加载，友好的用户界面等。

本项目的最终目标是成为一款实用、可靠的猫咪喂食器，为宠物主人提供智能化、自动化的喂食解决方案。通过该项目的应用，我们可以帮助更多的宠物主人解决猫咪喂食的难题，提升猫咪的生活质量，同时也为宠物市场带来一款创新的产品。未来，我们还将继续优化和完善该作品的功能和性能，以满足更多用户的需求和期望

1.2 项目实施计划

1.2.1 人员分工：

- 杨紫雯：项目总负责人，负责开题报告的撰写，用户交互界面，预设数据管理模块，喂食执行模块和主控模块的总编程，项目报告的修订。
- 郑世婷：负责开题报告的修订，用户交互界面的修订，喂食执行模块的设计，数据持久化的总编程，项目报告的撰写。
- 申佳依：负责开题报告的设计修订，时间管理和喂食提醒模块的总编程，项目报告的撰写。

1.2.2 设计制作进度安排

- 需求分析（第 1 天）：明确项目需求，包括喂食时间预设、喂食量控制、数据持久化及用户界面设计等。
- 系统设计（第 2-3 天）：完成系统总体设计，划分功能模块，确定关键技术点，设计数据结构和用户界面。
- 编码实现（第 4-8 天）：根据设计文档，分模块进行编码实现，包括用户界面、时间管理、预设管理、喂食执行、喂食提醒和数据持久化等模块。
- 测试与优化（第 9-10 天）：进行系统测试，包括单元测试、集成测试和用户测试，根据反馈进行功能优化和性能调优。
- 文档编写与总结（第 11 天）：整理项目文档，撰写项目总结报告。

2 总体设计

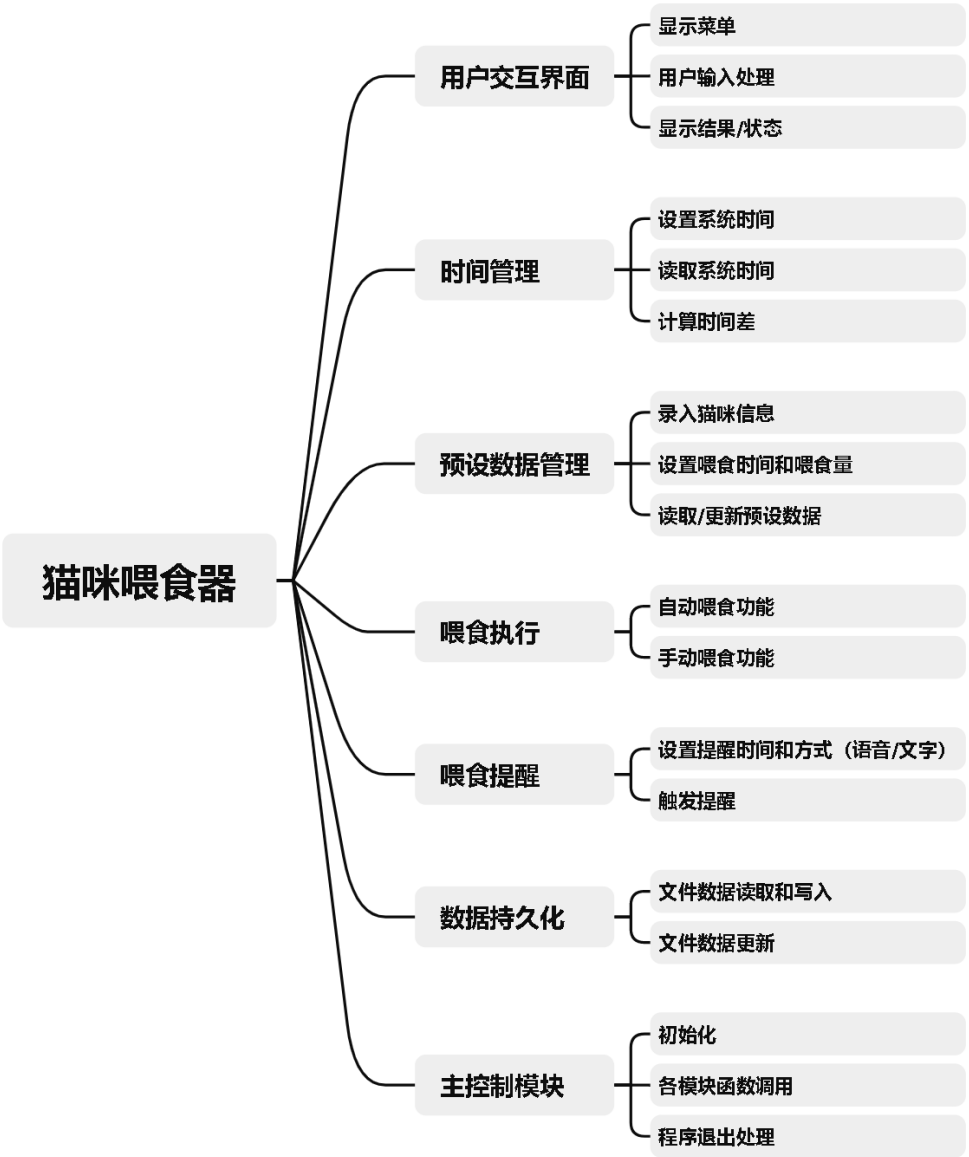
2.1 系统功能

2.1.1 功能概述

猫咪喂食器主要实现以下功能：

- 预设喂食时间和喂食量，支持多只猫咪的不同设置。
- 自动喂食功能，根据预设时间自动执行。
- 手动喂食功能，用户可随时触发。
- 喂食前提醒功能，通过语音或文字方式提醒用户。
- 数据持久化，保存预设数据和喂食记录到文件。
- 用户界面设计，提供友好的菜单和交互界面。

系统功能的框架结构图：



2.1.2 功能说明

- 预设管理：允许用户为每只猫咪设置不同的喂食时间和喂食量，并将这些设置保存到文件中。
- 自动喂食：系统定时检查当前时间，若到达预设的喂食时间，则自动执行喂食操作，并记录喂食日志。
- 手动喂食：用户可通过用户界面触发手动喂食，输入喂食量后执行，并记录喂食日志。
- 喂食提醒：在喂食前几分钟，通过显示文字的方式提醒用户。
- 数据持久化：使用文件存储预设数据和喂食记录，确保数据在系统重启后不会丢失。
- 用户界面：提供简洁明了的菜单和交互界面，方便用户进行各项操作。

2.2 系统软硬件平台

2.2.1 系统开发平台

开发语言：C 语言
开发工具：GCC 编译器，Windows 开发环境
辅助工具：文本编辑器 Visual Studio Code，调试器 Visual Studio Debugger

2.2.2 系统运行平台

硬件平台：x86 架构的 PC
软件平台：Windows11 操作系统，命令行界面（CMD）

2.3 关键技术

项目设计：使用 Xmind 辅助进行流程图与架构图设计
程序设计：

模块划分：使用引入头文件方式，声明各函数，实现模块化设计与管理。
时间管理：使用定时器或时钟中断实现时间管理和定时任务调度。
文件操作：文本文件信息读取与日志文件数据写入。
用户界面设计：设计简洁明了的菜单和交互界面，提高用户体验。

程序编译：在 CMD 中编译目录下所有文件，生成可执行文件并运行。

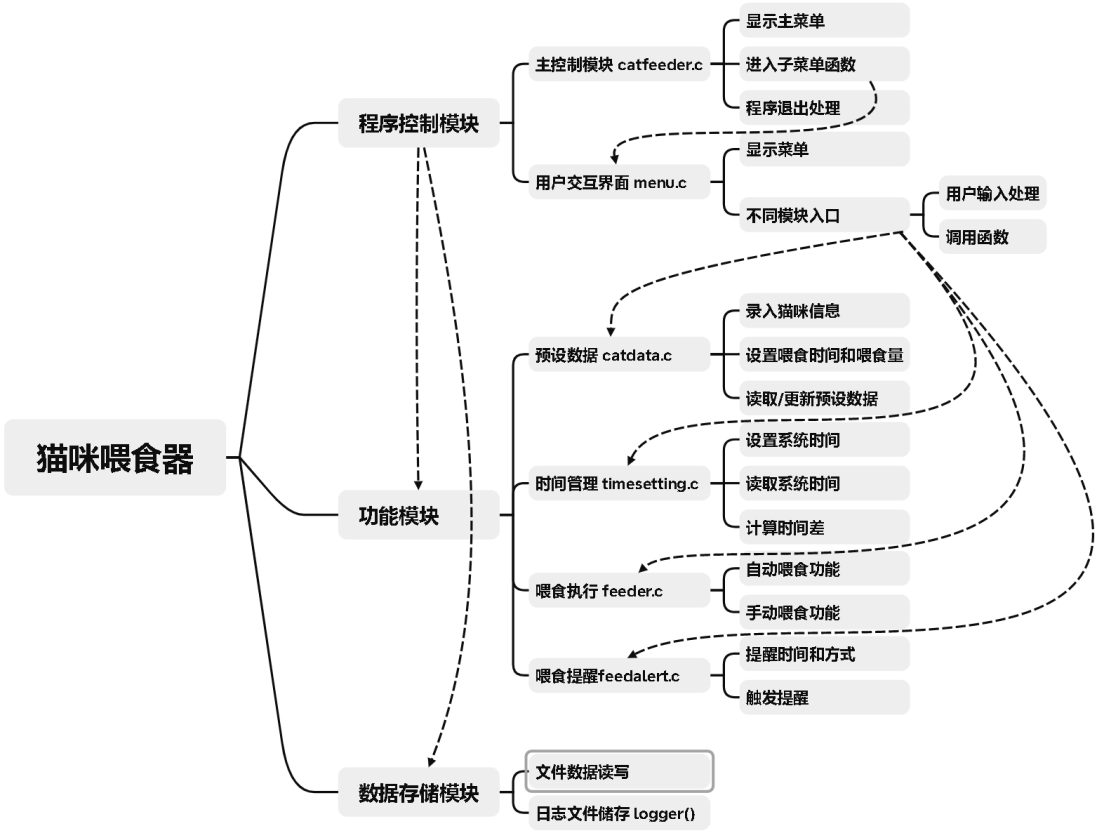
2.4 作品特色

多猫支持：支持为不同猫咪设置不同的喂食时间和喂食量。
智能化管理：自动喂食、喂食提醒等功能实现猫咪饮食的智能化管理。
语音提示：支持语音输出功能，可在喂食前与喂食时实时提醒。

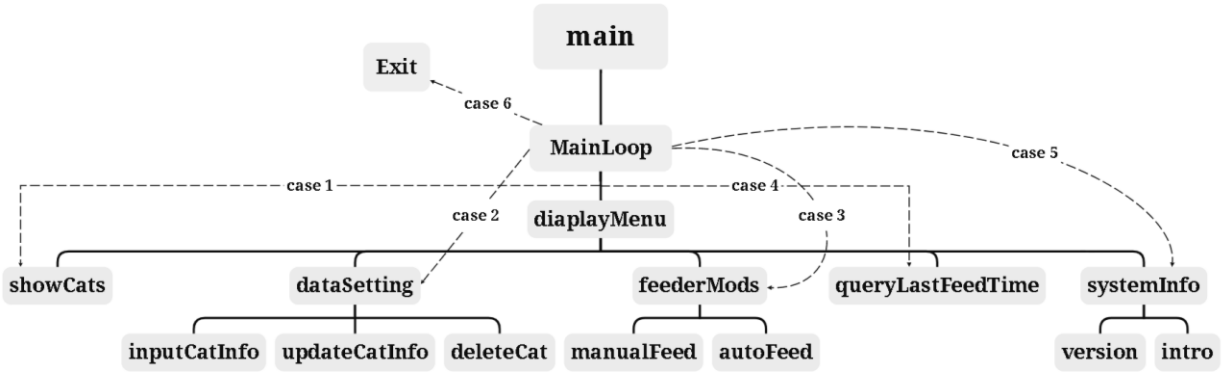
3 详细设计

3.1 系统结构设计

3.1.1 功能模块设计



3.1.2 关键功能/算法设计



3.2 数据结构设计

3.2.1 存储数据

本程序采用文件读取与输出的方式进行数据存储和处理。具体而言，程序通过标准 C 库中的文件操作函数实现数据的读写，确保数据的持久化和正确处理。

数据读取

程序首先使用“fopen”函数以读取模式打开指定的输入文件。然后通过“fscanf”函数逐行读取文件中的数据，并将其存储到内存中的相应数据结构中。在读取过程中，程序会进行必要的错误检查，确保文件存在且格式正确。读取完成后，使用“fclose”函数关闭文件。

数据输出

在数据处理完成后，程序使用“fopen”函数以写入模式打开输出文件。然后通过“fprintf”函数将处理后的数据逐行写入文件。为了保证数据的完整性，程序在写入过程中同样进行错误检查，确保文件能够正确写入并保存。数据写入完成后，使用“fclose”函数关闭文件。

3.2.2 关键数据结构

1. 结构体 Cat，定义喂食时间和喂食量

```
typedef struct Cat
{
    char name[MAX_NAME_LENGTH];
    int feedHour;
    int feedMinute;
    float feedAmount;
    struct Cat *next;点
} Cat;
```

2. 结构体 CatList，定义链表头指针

```
typedef struct
{
    Cat *head;
} CatList;
```

3.3 系统界面设计

3.3.1 界面设计风格

在 CMD 中启动程序，进入任意菜单时，使用 system("cls")清屏，以”=”符等构成表格菜单，并显示选项。每执行一个命令，输出文字换行，使页面更加清晰。

3.3.2 主要功能页面

1. 主菜单：

```
==================================
         Main menu
==================================
1. Show the catlist
2. Set cats' data
3. Set feeder mods
4. Query last feeding time
5. System Info
6. Exit
==================================
Please input your choice:
```

选项 1：显示猫咪信息
选项 2：设置猫咪信息
选项 3：设置喂食模式
选项 4：查看喂食日志
选项 5：查看系统信息
选项 6：退出程序

选项 1：显示当前猫咪信息：

```
=========================================
CatName         FeedTime        FeedAmount
=========================================
Leogao          15:30           77.00
Legao           6:55            80.00
Laoli           7:30            60.00
Nianqi          22:8            45.00
请按任意键继续. . .
```

分为三类显示：猫咪名字，喂食时间和喂食量

选项 2：设置猫咪信息

```
==================================
         Data Setting
==================================
1. Add a new cat to feed
2. Update a cat's feeding data
3. Delete a cat's feeding data
4. Return
==================================
Please input your choice:
```

选项功能：增加猫咪信息；更新猫咪信息；删除猫咪信息以及返回

选项 3：设置喂食模式

```
==================================
       Set Feeder Mods
==================================
1. Manual feeding
2. Automatic feeding
3. Return
==================================
Please input your choice:
```

选项功能：设置手动或自动喂食以及返回

4 总结

4.1 作品的创意

本作品的创意源自于对现代宠物饲养方式智能化的需求。随着人们生活节奏的加快，如何确保宠物猫咪能够按时、按量进食成为了一个亟待解决的问题。因此，我们设计了一款基于 C 语言的猫咪喂食器，旨在通过预设喂食时间、喂食量以及提供语音提醒等功能，实现猫咪饮食的智能化管理。这款喂食器不仅能够减轻宠物主人的负担，还能确保猫咪的饮食健康，提升宠物的生活质量。

4.2 开发实现过程

在开发过程中，我们首先将项目划分为多个模块，包括用户界面模块、时间管理模块、预设管理模块、喂食执行模块、喂食提醒模块和数据持久化模块。每个模块都承担了特定的功能，并通过主控制模块进行协调和调度。

- 用户界面模块：我们设计了简洁明了的菜单和交互界面，方便用户进行操作和查询。通过文本方式显示菜单，并接收用户的输入指令，执行相应的功能。
- 时间管理模块：实现了系统时间的读取、设置以及时间差的计算。这是实现自动喂食功能的基础，确保喂食器能够准确判断当前时间是否到达预设的喂食时间。
- 预设管理模块：允许用户为每只猫咪设置不同的喂食时间和喂食量，并将这些设置保存到文件中。同时，该模块还负责从文件中读取预设数据，以便在程序启动时恢复用户的设置。
- 喂食执行模块：根据预设的喂食时间和喂食量自动执行喂食操作。在喂食结束后，记录喂食时间和喂食量到文件中，以便下次使用。此外，该模块还提供了手动喂食功能，以应对特殊情况。
- 喂食提醒模块：在喂食前 5 分钟通过语音或文字方式提醒用户。我们采用了简单的文本提醒方式，但在未来版本中可以考虑集成语音模块，实现更直观的提醒效果。
- 数据持久化模块：负责将预设喂食时间和喂食量以及喂食记录保存到文件中，并能够从文件中加载和更新数据。这是确保数据不丢失的关键。
- 在主控制模块的协调下，各个模块协同工作，共同实现了猫咪喂食器的各项功能。

4.3 改进方向

尽管我们已经完成了猫咪喂食器的初步开发，但仍有许多地方可以进一步改进和优化：

- 优化用户界面：虽然目前的用户界面已经足够简洁明了，但仍有提升空间。我们可以考虑引入图形界面（GUI），使操作更加直观和友好。
- 增强数据安全性：在数据持久化方面，我们目前采用了简单的文件存储方式。为了保障数据的安全性，未来可以考虑引入数据库管理系统，对数据进行加密和备份。
- 提升系统稳定性：在长时间运行过程中，系统可能会遇到各种异常情况。为了提升系统的稳定性，我们需要加强异常处理机制，确保系统能够在遇到错误时及时恢复并继续运行。
- 增加多猫识别功能：目前我们的喂食器支持为每只猫咪设置不同的喂食时间和喂食量，但并未实现多猫识别功能。未来可以考虑引入 RFID 等技术，实现喂食器的自动识别和多猫管理功能。

通过不断的改进和优化，我们相信猫咪喂食器将能够成为更多宠物家庭的得力助手，为宠物猫咪提供更加科学、便捷的饮食管理服务。

5 附录

5.1 名词定义

名词/缩写	说明
GUI	全称是 Graphical User Interface，中文意思为“图形用户界面”。是指采用图形方式显示的计算机操作用户界面，是一种人与计算机通信的界面显示格式，允许用户使用鼠标等输入设备操纵屏幕上的图标或菜单选项。
RFID	全称是 Radio Frequency Identification，中文意思为射频识别技术。RFID 是一种通信技术，它通过无线电讯号识别特定目标并读写相关数据，而无需识别系统与特定目标之间建立机械或光学接触。
CMD	全称是 Command，意思是命令提示符（Command Prompt）。它是在 Windows 操作系统中的一个基本的系统组件，允许用户通过键入文本命令来与计算机进行交互和执行各种任务。CMD 是一个文本界面的命令行工具，它提供了一种通过命令行界面来控制和管理计算机的方法，这与图形用户界面（GUI）相对，后者使用图形元素和可视化控件来进行交互。
Xmind	Xmind 是一款功能全面的思维导图和头脑风暴软件

5.2 参考资料

[1] 计算机软件产品开发文件编制指南. 中华人民共和国国家标准 GB8567-88. 国家标准局，1988 年 1 月 7 日.

[2] 谢希仁.计算机网络（第四版）. 大连理工大学出版社，2006.8.

[3] 严霄凤,高炽扬. 美国联邦信息安全风险管理框架及其相关标准研究. 信息安全与通信保密, 2(2009): 40-44.

[4] Salakhutdinov and Geoff Hinton, Training a deep autoencoder or a classifier on MNIST digits, http://www.cs.toronto.edu/~hinton/MatlabForSciencePaper.html, 2006.

5.3 源代码清单

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <unistd.h>
#include <conio.h>

#include "headers/catfeeder.h"
#include "headers/menu.h"
#include "headers/timesetting.h"
#include "headers/catdata.h"
#include "headers/feeder.h"
```

```
#include "headers/feedalert.h"
//***模块一：用户交互页面
/**
 * display the main menu 显示菜单
 */
void displayMenu(CatList *list)
{
    // clrscr();
    system("cls");
    printf("==================================\n");
    printf("              Main menu\n");
    printf("==================================\n");
    printf("1. Show the catlist\n");
    printf("2. Set cats' data\n");
    printf("3. Set feeder mods\n");
    printf("4. Query last feeding time\n");
    printf("5. System Info\n");
    printf("6. Exit\n");
    printf("==================================\n");
}

/**
 * display the main menu
 */
void showCats(CatList *list)
{
    system("cls");
    printf("=========================================\n");
    printf("CatName\t\tFeedTime\tFeedAmount\n");
    printf("=========================================\n");
    showData(list);
}

void dataSetting(CatList *list)
{
    system("cls");
    printf("==================================\n");
    printf("              Data Setting\n");
    printf("==================================\n");
    printf("1. Add a new cat to feed\n");
    printf("2. Update a cat's feeding data\n");
    printf("3. Delete a cat's feeding data\n");
    printf("4. Return\n");
    printf("==================================\n");
```

```
    int choice;
    while (1)
    {
        printf("Please input your choice:\n");
        scanf("%d", &choice);
        switch (choice)
        {
        case 1:
            inputCatInfo(list); // add a new cat info
            break;
        case 2:
            updateCatInfo(list); // update a cat's info
            break;
        case 3:
            deleteCat(list); // delete a cat's info
            break;
        case 4:
            saveData(list); // recieve and return
            return;
        default:
            printf("no such choice");
            system("pause");
            break;
        }
    }
}

void feederMods(CatList *list)
{

    int choice;
    FeedTime time;
    while (1)
    {
        system("cls");
        printf("==================================\n");
        printf("            Set Feeder Mods\n");
        printf("==================================\n");
        printf("1. Manual feeding\n");
        printf("2. Automatic feeding\n");
        printf("3. Return\n");
        printf("==================================\n");
```

```
            printf("Please input your choice:\n");
            scanf("%d", &choice);
            switch (choice)
            {
            case 1:
                manualFeed(list);
                system("pause");
                break;
            case 2:
                autoFeed(list);
                system("pause");
                break;
            case 3:
                return;
            default:
                printf("no such choice");
                system("pause");
                break;
            }
        }
}

void systemInfo()
{
    int choice;
    while (1)
    {
        system("cls");
        printf("==================================\n");
        printf("            System Info\n");
        printf("==================================\n");
        printf("1. Version Info\n");
        printf("2. System introduction\n");
        printf("3. Return\n");
        printf("==================================\n");

        printf("Please input your choice:\n");
        scanf("%d", &choice);
        switch (choice)
        {
        case 1:
            printf("CatFeeder version 1.0.1\n");
            system("pause");
            break;
```

```
            case 2:
                systmeIntroduction();
                system("pause");
                break;
            case 3:
                return;
            default:
                printf("no such choice");
                system("pause");
                break;
            }
        }
    }
    /**
     * Display the system introduction
     */
    void systmeIntroduction()
    {
        printf("This is a Cat Feeder\n\n");
        printf("It based on C language, realizing smart feeding through functions like presetting feeding time, feed amount, and sound alert.\n\n");
        printf("Below are Feeder's main functions:\n\n");
        printf("1. Presetting time and feeding amount: The cat feeder can preset the feeding time and feeding amount at any time, \n");
        printf("\tand each cat can preset different feeding amount and time.\n");
        printf("2. Automatic feeding: The cat feeder can automatically feed according to the preset feeding time and feeding amount.\n");
        printf("3. Manual feeding: In special circumstances, users can manually trigger the feeder to feed, such as when they forget to set \n");
        printf("\tthe feeding time for the cat, users can feed the cat manually.\n");
        printf("4. Feeding reminder: The cat feeder can make voice and text reminders 5 minutes before feeding\n");
        printf("5. Feeding log will be saved in the catfeeder.log\n");
        printf("6. Master can find out lastest feeding time in feeding log.\n\n");
    }
    //**模块二：时间管理模块
    /**
     * 设置系统时间
     * */
    void set_time(struct tm *time_a) // tm 这是一个结构体类型，定义在 <time.h> 中，用于表示日期和时间。
    {
        printf("set system time:");
        int sec = (*time_a).tm_sec;                // 秒（0-59）
```

```
        int min = (*time_a).tm_min;                // 分（0-59）
        int hour = (*time_a).tm_hour;              // 时（0-23）
        int day = (*time_a).tm_mday;               // 月中的第几天（1-31）
        int month = (*time_a).tm_mon + 1;          // 月份(0-11,其中 0 表示 1 月)
        int year = (*time_a).tm_year + 1900; // 年份(自 1900 起)
        printf("%d-%d-%d %d:%d:%d\n", year, month, day, hour, min, sec);
    }

    /**
     * 读取系统时间
     */
    void read_current_time(struct tm *current_time)
    {
        time_t now = time(NULL);                   // time_t 是数据类型，是 1970 年 1 月 1 日以来的秒
数
        *current_time = *localtime(&now); // localtime 函数，将 time_t 类型的值转换为本地时间
表示
    }

    /**
     * 计算时间差
     */
    int time_difference(struct tm *time_a, struct tm *current_time)
    {
        int td;
        td = (time_a->tm_hour - current_time->tm_hour) * 60 + (time_a->tm_min -
current_time->tm_min);
        return (td);
    }
    //***模块三 ：预设数据管理
    void initCatList(CatList *list) // 初始化链表
    {
        list->head = NULL;
    }

    void destroyCatList(CatList *list) // 销毁链表
    {
        Cat *current = list->head;
        Cat *next;
        while (current != NULL)
        {
            next = current->next;
            free(current);
            current = next;
```

```
        }
        // 遍历列表，挨个释放内存
        list->head = NULL;
    }

    /**
     * showCats(): go through the catlist, and show all cats' data
     */
    void showData(CatList *list)
    {
        Cat *current = list->head; // 定义 current 作为链表头指针
        if (current == NULL)
        {
            printf("Dear Master, there is no cat yet! Please add cats' info");
            return;
        }
        while (current != NULL)
        {
            printf("%s\t\t%d:%d\t\t%.2f\n", current->name, current->feedHour,
current->feedMinute, current->feedAmount);
            current = current->next; // 将指针 current 指向下一个节点
        }
    }

    /**
     * addCat(): add a new cat to the catlist
     */
    void addCat(CatList *list, const char *name, int hour, int minute, float amount) // 添加猫咪信息
    {
        Cat *newCat = (Cat *)malloc(sizeof(Cat)); // 为指向结构体 Cat 的指针 newCat 设置内存空
间

        strcpy(newCat->name, name); // 将结构体 Cat 中的变量赋值给定义的变量
        newCat->feedHour = hour;
        newCat->feedMinute = minute;
        newCat->feedAmount = amount;
        newCat->next = list->head;
        // printf("%s %d %d %.2f\n", newCat->name, newCat->feedHour, newCat->feedMinute,
newCat->feedAmount);
        list->head = newCat;
    }

    /**
     * removeCat(): remove a cat from catlist
```

```
 */
void removeCat(CatList *list, const char *name) // 删除猫咪信息
{
    Cat *current = list->head; // 定义 current 作为链表头指针
    Cat *prev = NULL;          // 用于跟踪当前节点的前一个结点

    while (current != NULL && strcmp(current->name, name) != 0)
    {
        prev = current;          // 将当前 current 要删除的目标猫咪信息赋给 prev
        current = current->next; // 链表传递向前覆盖
    } // 遍历匹配到猫咪信息

    if (current == NULL)
    {
        printf("Cat not found!\n");
        return;
    }
    else if (prev == NULL)
        list->head = current->next;
    else
        prev->next = current->next;

    free(current); // 释放当前节点占用的内存，删除该节点
}

/**
 * saveData(): write and save the cats' data to feeding_schedule.txt
 */
void saveData(CatList *list)
{
    FILE *file = fopen(CAT_INFO_FILE, "w"); // （结尾可以用.conf(配置)或者.ini(初始化)文件）
    if (file == NULL)
    {
        printf("Error opening file!\n");
        return;
    }
    Cat *current = list->head; // 设置链表头指针位置
    while (current != NULL)
    {
        // fwrite(current, sizeof(Cat), 1, file); // 写入当前节点 current 中的信息【只能以二进制保存】
        fprintf(file, "%s\t%d:%d\t%.2f\n", current->name, current->feedHour, current->feedMinute, current->feedAmount);
```

```
            current = current->next; // 将指针 current 指向下一个节点
        }
        fclose(file);
}

/**
 * loadData(): read the cat info from the document feeding_schedule.txt
 */
void loadData(CatList *list)
{
        FILE *file = fopen(CAT_INFO_FILE, "r");
        if (file == NULL)
        {
            printf("No existing data found.\n");
            return;
        }

        char name[MAX_NAME_LENGTH];
        int feedHour;
        int feedMinute;
        float feedAmount;
        while (fscanf(file, "%s\t%d:%d\t%f", name, &feedHour, &feedMinute, &feedAmount) == 4)
        {
            addCat(list, name, feedHour, feedMinute, feedAmount);
            printf("%s %d %d %.2f\n", name, feedHour, feedMinute, feedAmount);
        }

        fclose(file);
}

/**
 * inputCatInfo(): setting a cat's feeding time and feed amount
 */
void inputCatInfo(CatList *list)
{
        char name[MAX_NAME_LENGTH];
        int hour, minute;
        float amount;

        printf("Enter cat's name: ");
        scanf("%s", name);
        printf("Set pre-feed time-->\n");
        printf("Set pre-feed hour (0-23): ");
        scanf("%d", &hour);
```

```
        printf("Set pre-feed minute (0-59): ");
        scanf("%d", &minute);
        printf("Set feeding amount in grams: ");
        scanf("%f", &amount);

        addCat(list, name, hour, minute, amount); // 调用 addCat()函数，将设定的某猫喂食时间和喂食量添加到 CatList 链表
        saveData(list);

        char logMessage[MAX_NAME_LENGTH + 20];
        snprintf(logMessage, sizeof(logMessage), "Data of %s has been added", name);
        logger(logMessage);
    }

    /**
     * updateCatInfo()： update certain cat's feeding time and feed amount
     */
    void updateCatInfo(CatList *list)
    {
        char name[MAX_NAME_LENGTH];
        int hour, minute, amount;

        printf("Enter cat's name to update: ");
        scanf("%s", name);

        // Invoke removeCat() to remove the cat info
        removeCat(list, name);

        printf("Set new feeding hour (0-23): ");
        scanf("%d", &hour);
        printf("Set new feeding minute (0-59): ");
        scanf("%d", &minute);
        printf("Set new feeding amount in grams: ");
        scanf("%d", &amount);

        // 调用 addCat()增加新的喂食信息
        addCat(list, name, hour, minute, amount);
        // 调用 saveData()函数保存信息
        saveData(list);

        char logMessage[MAX_NAME_LENGTH + 20];
        snprintf(logMessage, sizeof(logMessage), "Data of %s has been updated", name);
        logger(logMessage);
    }
```

```
/**
 * deleteCat(): delete a cat's data from the catlist
 */
void deleteCat(CatList *list)
{
    char name[MAX_NAME_LENGTH];
    printf("Enter cat's name to delete: ");
    scanf("%s", name);

    removeCat(list, name);

    char logMessage[MAX_NAME_LENGTH + 20];
    snprintf(logMessage, sizeof(logMessage), "Data of %s has been deleted", name);
    logger(logMessage);
}

/**
 * retrieve feeding time from a line of log
 */
void extract_time(const char *line, char *time)
{
    strncpy(time, line + 1, 19);
    time[19] = '\0';
}

/**
 * Search log file to find out when the latest time the specified cat being fed.
 */
void queryLastFeedTimeInLog()
{
    char catName[MAX_NAME_LENGTH];
    printf("Please input cat name: ");
    scanf("%s", catName);
    char strFedTime[20] = "";

    FILE *file = fopen(LOG_FILE, "r");
    if (!file)
    {
        perror("Error opening file");
        return;
    }

    char line[200];
```

```
    char temp_time[20] = "";
    // Initialize last_time to empty string
    strFedTime[0] = '\0';

    while (fgets(line, sizeof(line), file))
    {
        if (strstr(line, catName))
        {
            extract_time(line, temp_time);
            // Update strFedTime to the current found time
            strcpy(strFedTime, temp_time);
        }
    }
    if (strFedTime[0] == '\0')
        printf("no %s feeding record in log file!\n", catName);
    else
        printf("%s was fed at %s \n", catName, strFedTime);

    system("pause");
    fclose(file);
}
//***模块四：喂食执行
void feedAction(char *name)
{
    printf("Start feeding %s ", name);
    for (int i = 0; i < 3; i++)
    {
        printf(".");
        sleep(1);
    }
    printf("Done!\n");
}

/**
 * The auto feeding function. In this function, the feeder do:
 * 1. build a endless loop to check the difference between current time and pre-set feeding time
 * 2. if the difference time is 5 minutes, alert master the feeding will start soon
 * 3. if the difference time is 0 minutes, start the feeding
 * 4. anytime, hit "esc" key to quit
 */
void autoFeed(CatList *list)
{
    FeedTime feedTime;
    struct tm time;
```

```
        struct tm currentTime;
        int autoFeeding = 1;
        int diffTime = 0;
        printf("Auto-feeding has been started, press 'esc' key to exit the auto-feeding\n");
        while (autoFeeding)
        {
            Cat *current = list->head;
            while (current != NULL)
            {
                // trigger to remind master to feed before 5 minute ahead the preset feeding time
                time.tm_hour = current->feedHour;
                time.tm_min = current->feedMinute;

                read_current_time(&currentTime);                                    //
read_current_time(&currentTime); //get current time
                diffTime = time_difference(&time, &currentTime); // caculate the difftime between
current time and pre-set time
                if (diffTime == 5)
                {
                    // printf("Wait %d minutes to feed %s\n",diffTime, current->name);
                    playSound(SOUND_ALERT); // when difftime is 5 minutes, play sound to
remind feeding
                }

                // trigger to feed the cats when the preset time is current time
                feedTime.hour = current->feedHour;
                feedTime.minute = current->feedMinute;
                triggerFeedReminder(current->name, feedTime);

                current = current->next;
            }

            // check if there is "esc" being hit every 1 second. After 1 minute, start next check loop.
            for (int i = 0; i < 60; i++)
            {
                // sleep 1 second
                sleep(1);
                if (_kbhit())
                {
                    char ch = _getch(); // 获取按下的键
                    // ASCII 27 是 'esc' 键
                    if (ch == 27)
                    {
                        printf("Detected 'esc' be pressed! Auto-feeding has been terminated\n");
```

```
                        autoFeeding = 0;
                        break;
                    }
                }
            }
        }
    }

    void manualFeed(CatList *list)
    {
        char name[MAX_NAME_LENGTH];
        int flag = 0;
        printf("Input the cat you want to feed manually: ");
        scanf("%s", name);

        Cat *current = list->head;
        while (current != NULL)
        {
            if (strcmp(current->name, name) == 0)
            {
                flag = 1;
                break;
            }
            current = current->next; // go through the cat list
        }

        if (flag == 0)
        {
            printf("Dear Master, the catlist is empty now! Please add cats' info:\n");
            inputCatInfo(list);
            saveData(list);
        }
        playSound(SOUND_CALLCAT);
        feedAction(name);

        char logMessage[MAX_NAME_LENGTH + 20]; // Buffer for log message
        snprintf(logMessage, sizeof(logMessage), "%s has been fed\n", name);
        logger(logMessage);
    }
    //***喂食提醒模块
    // 判断是否提醒
    int checkReminder(FeedTime *presetTime, const struct tm *currentTime)
    {
        if (presetTime->hour == currentTime->tm_hour && presetTime->minute ==
```

```
currentTime->tm_min) // 简化处理：仅比较小时和分钟
        {
            return 1; // 需要提醒
        }
        return 0; // 不需要提醒
    }

    // 触发提醒
    void triggerFeedReminder(char *catName, FeedTime presetTime)
    {
        time_t rawtime;
        struct tm *currentTime;
        time(&rawtime); // 获取当前时间
        currentTime = localtime(&rawtime);
        if (checkReminder(&presetTime, currentTime))
        {
            //printf("Reminder for %s: It's time to feed!\n", catName);
            playSound(SOUND_CALLCAT);
            printf("Start feeding %s ", catName);
            for (int i = 0; i < 3; i++)
            {
                printf(".");
                sleep(1);
            }
            printf("Done!\n");
            char logMessage[MAX_NAME_LENGTH + 20]; // Buffer for log message
            snprintf(logMessage, sizeof(logMessage), "%s has been fed\n", catName);
            logger(logMessage);
        }
    }
    //***数据持久化模块
    /**
     * this function is to output the message to log file with timestamp
     * every operation, such as adding a cat info, manu-feeding, to catfeeder should be logged into
log file.
     */
    void logger(char *message)
    {
        // 1. open the log file(already defined as LOG_FILE in catfeeder.h) in append mode
        // 2. get current time and convert to char*, like "[2024-07-06 15:08:55]"
        // 3. using fprintf to output the current time and meesage to log file
        // 4. clost the log file
        FILE *file = fopen(LOG_FILE, "a");
        if (file == NULL)
```

```
        {
            printf("Error opening log file!\n");
            return;
        }

        int year, month, day, hour, minute, second;
        get_current_datetime(&year, &month, &day, &hour, &minute, &second);
        fprintf(file, "[%4d-%02d-%02d %02d:%02d:%02d] %s\n", year, month, day, hour, minute, second, message);

        fclose(file);
    }

    /**
     * play specified sound to alert
     */
    int playSound(char *wavFilePath)
    {
        // play the video 'wav'
        if (!PlaySound(wavFilePath, NULL, SND_FILENAME | SND_SYNC))
        {
            printf("fail to play a sound!\n");
        }

        return 0;
    }

    void get_current_datetime(int *year, int *month, int *day, int *hour, int *minute, int *second)
    {
        // get current time
        time_t now;
        time(&now);

        // change the time into local time
        struct tm *local = localtime(&now);

        // get the year, month, day, hour, minute and second
        *year = local->tm_year + 1900;
        *month = local->tm_mon + 1;
        *day = local->tm_mday;
        *hour = local->tm_hour;
        *minute = local->tm_min;
        *second = local->tm_sec;
    }
```

```
//***主控模块
int main()
{
    CatList catList;

    initCatList(&catList);
    loadData(&catList);

    int choice;
    while (1)
    {
        displayMenu();
        printf("Please input your choice:\n");
        scanf("%d", &choice);

        switch (choice)
        {
            case 1:
                showCats(&catList);
                system("pause");
                break;
            case 2:
                dataSetting(&catList);
                break;
            case 3:
                feederMods(&catList);
                break;
            case 4:
                queryLastFeedTimeInLog();
                break;
            case 5:
                systemInfo();
                break;
            case 6:
                saveData(&catList);
                destroyCatList(&catList);
                return 0;
            default:
                printf("Invalid choice!\n");
                system("pause");
                break;
        }
    }
```

```
        return 0;
    }
```