華中科技大學

C语言课程设计报告

手机备忘录

院　　系　人工智能与自动化学院

专业班级　自动化类 2204 班

姓　　名　胡天齐、汪耀武

学　　号　U202215016、U202215029

指导教师　全体老师

2023 年 4 月 22 日

目　　录

1 前言

1.1 编写背景

手机备忘录app是一款实用的应用程序，可以实现快速创建、编辑、存储和共享备忘录。 它可以帮助用户在日常生活中轻松管理自己的事务和进行日程安排，从而提高效率并节省时间。

随着现代生活节奏加快，人们的工作和生活越来越繁忙， 需要一种方便快捷的方法来记录和管理重要的信息和事情。 而传统的笔记本和便签因为容易被遗失或损坏逐渐被人们淘汰。

因此，开发一款备忘录app势在必行。 这个app可以帮助用户在计算机上创建备忘录， 随时随地保存和查找信息，无论是文字记录、插入图片还是分级任务等等，都可以轻松的完成。

尽管现代操作系统和应用程序 **已经取代了DOS环境**， 但在 **某些场景下**，DOS环境仍然是必要的。 例如，使用旧版本的电脑或者 **出于特殊需要**，DOS仍然有着用武之地。 因此，开发这个DOS环境下的备忘录应用仍然是有一定的实际需求的。

总而言之，DOS环境下备忘录app的开发是源于某些场景下的需求，帮助特定用户更好地管理自己的信息，以提高效率。

1.2 用户需求分析

在此大背景下，我们对用户的需求进行了详细分析。具体如下。

1.2.1 基本的文本记录功能需求

在忙碌的生活中，为避免忘记某些重要信息或想法，人们需要一个能够随时记录文字的工具。

备忘录的文本记录功能需要具备简洁明了的界面和方便快捷的操作方式，以确保用户可以快速记录信息。

1.2.2 图片插入功能需求

有时候，用户可能需要通过图片来记录某些信息，如产品外观、食物照片等。

此功能可以为用户提供更全面、直观的信息展示方式，有助于更好地记录信息。此外，本功能还可以帮助用户更好地组织信息，比如将多张照片合并到同一个备忘录中。这将大大提高用户使用效率。

1.2.3 任务清单功能需求

人们在日常生活中需要完成的任务很多，有时候难以一下子全部记住，因此需要一个能够清晰列出所有任务的工具。

任务清单可以帮助用户将任务有条理地列出来，更好地管理时间和精力，还可以帮助用户追踪任务的完成情况，避免遗漏。

1.2.4 对备忘录分级处理功能需求

用户在记录信息时，可能需要将信息进行分类、归档，以便更好地管理。

对备忘录分级处理功能可以让用户将备忘录分成不同的级别，比如私人备忘录、工作备忘录等。

1.2.5 多用户共享、协同处理功能需求

生产生活中，有些用户可能需要将记录的信息与其他人共享，如团队成员之间共享某个项目的进展、家庭成员之间共享家务事情等。

多用户共享、协同处理功能可以让多个用户编辑同一个备忘录，提高了团队协作效率。多用户共享、协同处理功能还可以减少因为信息不对称而导致的沟通障碍，提高了信息交流的效率。

1.2.6 用户编辑记录功能需求

许多生产场景中，我们需要记录文最后修改用户以保证内容的可追查性。

1.3 参考资料

1.程序设计教程：用C/C++编程 周纯杰等编著.—北京：机械工业出版社，2016.4

2.数据结构：C语言版 严蔚敏，吴伟民编著.—北京：清华大学出版社，1997.4

1.4 编者的话

感谢您使用我们开发的备忘录App。 我们深知备忘录对当代人们的工作和生活是非常重要的，因此我们致力于为用户提供一款简单、方便、易用的应用程序，来帮助用户管理自己的信息和任务，从而提高工作效率与生活质量。

我们在开发和研究的过程中获得丰富的应用程序开发经验， 并致力于将经验化为实际以为用户带来更好的服务。 我们深入了解了用户的需求和习惯，针对于用户的实际需求，开发了这款备忘录App。

这款备忘录软件的核心功能包括创建、编辑、存储备忘录信息。 用户可以便利地使用该软件来记录备忘录信息，提高工作和生活效率。

最后，我想感谢所有参与这个项目的人，包括开发人员、测试人员和用户。正是你们的支持和鼓励，让我们能够顺利完成这个项目。我们将继续努力，提供更好的作品。

2 任务概述

2.1 目标功能

基于以上需求，我们设计了一款备忘录app。以下是其目标实现的功能。

2.1.1 基本的文本记录

文本记录与读取功能是该备忘录app的核心功能之一。 用户可以使用该功能记录自己的想法、笔记和计划。 在编写和保存备忘录时，用户可以自行拟定标题和内容。

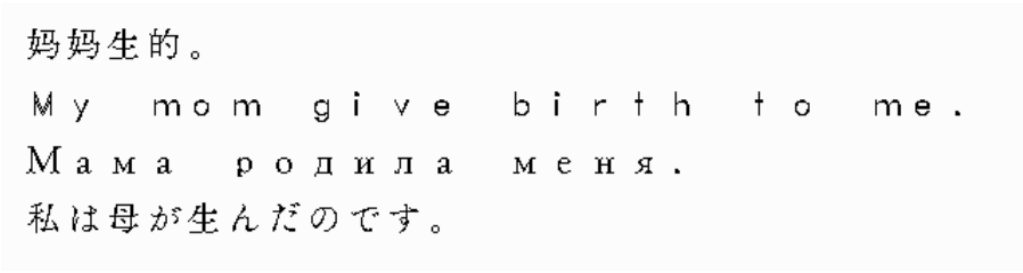

图 2-1 基本的文本记录示意图

2.1.2 图片插入

用户可以在备忘录中插入图片来丰富备忘录的内容和信息。 通过这个功能，用户可以更好地记录和分享自己的经历和想法等。

图 2-2　图片插入示意图

2.1.3　任务清单功能

用户可以以此写下自己的任务和待办事项从而起到备忘的作用。 用户可以创建、编辑、标记和完成任务，以便工作和生活更好的有序进行。

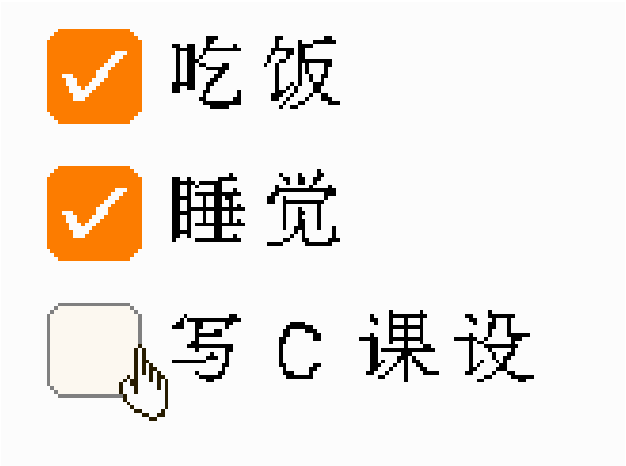

图 2-3　任务清单功能示意图

2.1.4 对备忘录分级处理

为了让备忘录更加有序和易于管理，我们开发了一个对备忘录进行分级处理的功能。 用户可以按照备忘录的重要性或者是紧急程度，将备忘录分为高、中、低三个级别。 从而可以让用户更加有效地管理和处理自己的备忘录，提高工作和生活效率。

2.1.5 多用户共享、协同处理

为了支持团队合作的场景，我们支持多用户访问，实现多个用户共享、协同处理备忘录。 这可以极大程度上提高团队协作的效率。

2.1.6 用户编辑记录

为适应多用户写作的实际场景，app还添加了一个用户编辑记录的功能。 当前用户可以在备忘录中看到上一位编辑者的名称。 这个功能可以让多个用户在共享一个备忘录的情况下，来区分和记录每个用户的修改和贡献。

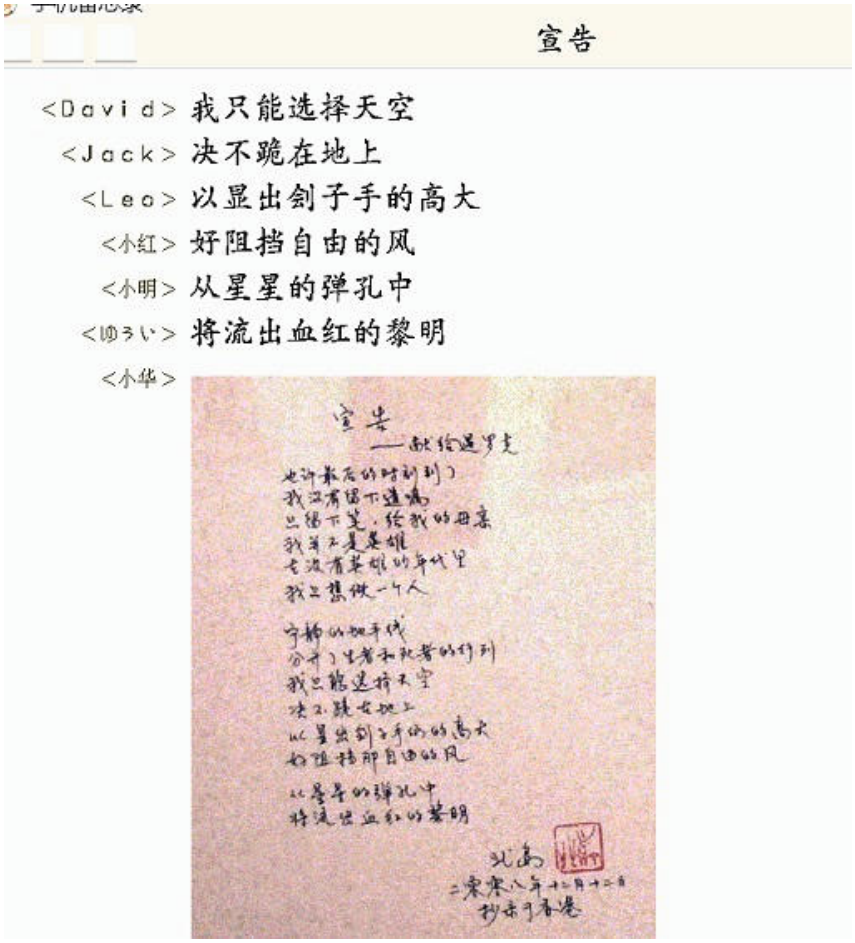

图 2-4 用户编辑记录示意图

2.1.7 用户友好的图形界面

在总体设计上，我们研发的“备忘录”软件旨在提供简洁、易用和功能丰富的界面。 用户可以轻松地浏览和管理自己编写的备忘录、任务和图片。 同时，我们的软件还支持通过快捷键操作，用户可以通过键盘快速完成操作来提高工作效率。

2.2 开发规范

2.2.1 命名规范

本程序使用模块化编程。 对于每个模块中的公有方法，以 <模块名>[_<方法名>] 进行命名； 私有方法以 _<模块名>_<方法名> 进行命名。 其中，模块名与方法名使用驼峰命名法命名。

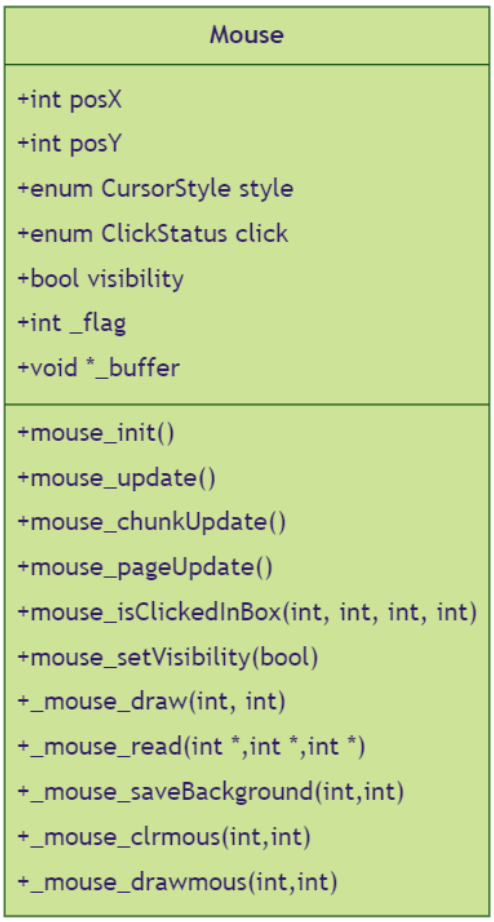

图 2-5 模块命名示例

本程序源码中严格避免使用全局变量。

2.2.2 注释规范

为了增强可读性，本程序源码使用doxygen语法对函数和结构体等进行注释， 以便于后续维护、生成开发文档。

2.3 运行环境和配置

2.3.1 最低硬件需求

- 处理器：要求 Intel Pentium 166 MX 或以上。
- 硬盘：要求空间500MB以上。
- 显示适配器：要求最低支持256色的SVGA显示适配器。
- 系统运行内存：要求32MB以上。

2.3.2 软件需求

- 开发调试软件工具：Borland C 3.1。
- 数据库：纯文本文档。
- 运行操作系统：MSDOS 或 Windows 9X/ME/2000/XP 并启用 Virtual DOS Machine。

3 基本框架与流程图

3.1 模块化设计

3.1.1 页面组件

在BC开发中，前后端是统一的，前端页面也是硬编码形成的。不过为了让结构更加清晰，我们也可以借用一般C/S架构开发的经验。

C/S（Client/Server）开发模式的优点如下：

1. 灵活性：C/S模式允许在客户端和服务器之间进行更灵活的交互，因为客户端和服务器可以独立发展、独立部署。
2. 高安全性：C/S模式通过给服务器分配访问和权限控制，从而提高了应用程序的安全性。这种模式还可以使用各种身份验证和加密技术来保护数据和通信。
3. 可靠性：由于C/S架构将大部分处理任务交给服务器完成，因此减少了客户端上产生的错误，增强了整个应用程序的可靠性和稳定性。
4. 可扩展性：C/S模式允许对服务器进行升级，以满足不断增长的用户需求，同时允许添加新的客户端功能。
5. 良好的用户体验：C/S模式可以使客户端应用程序集成用户友好的图形用户界面，从而增强用户体验。

在前端语言中，展示与交互是同时在页面上体现，而相互关联的。显然，在BC中监听鼠标、键盘等输入时，需要一个循环体，持续对输入进行监测、判断和处理。让我们使用前端开发的思路写一个页面组件。每个组件内可以分为固定界面渲染、事件监听和事件处理三部分。

将固定页面的渲染和事件监听分成两个部分，页面渲染放在组件被创建的最初位置，将事件监听放入循环体内，并与组件通信，判断是否需要处理，并分两种处理方式处理。具体流程如下。

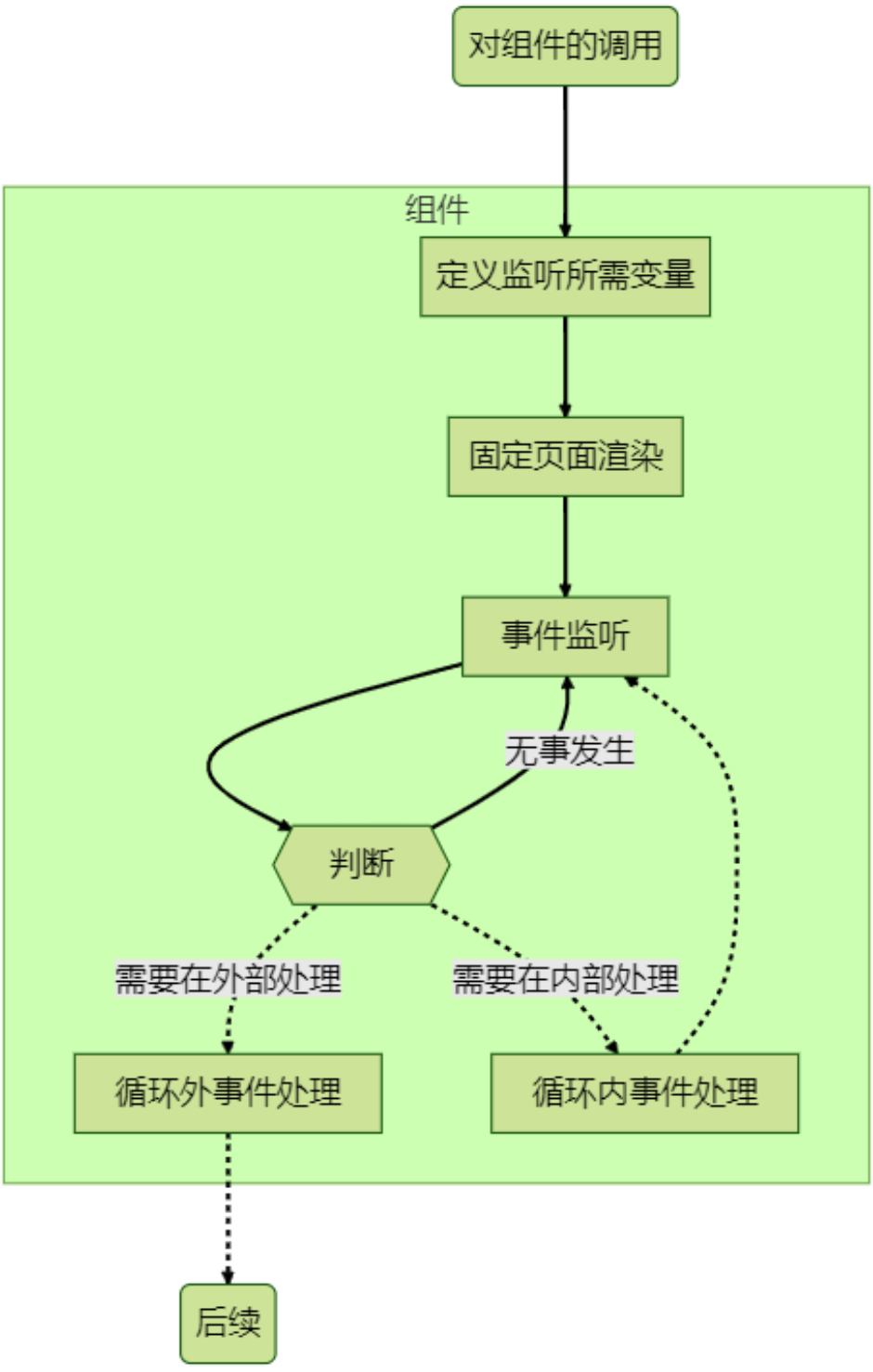

图 3-1　页面组件示意流程图

通过这种方式，实现了类似“前后端分离”的效果。

3.1.2　页面跳转

为了清晰描述页面显示的相关流程，我们将其抽象成如下栈结构。

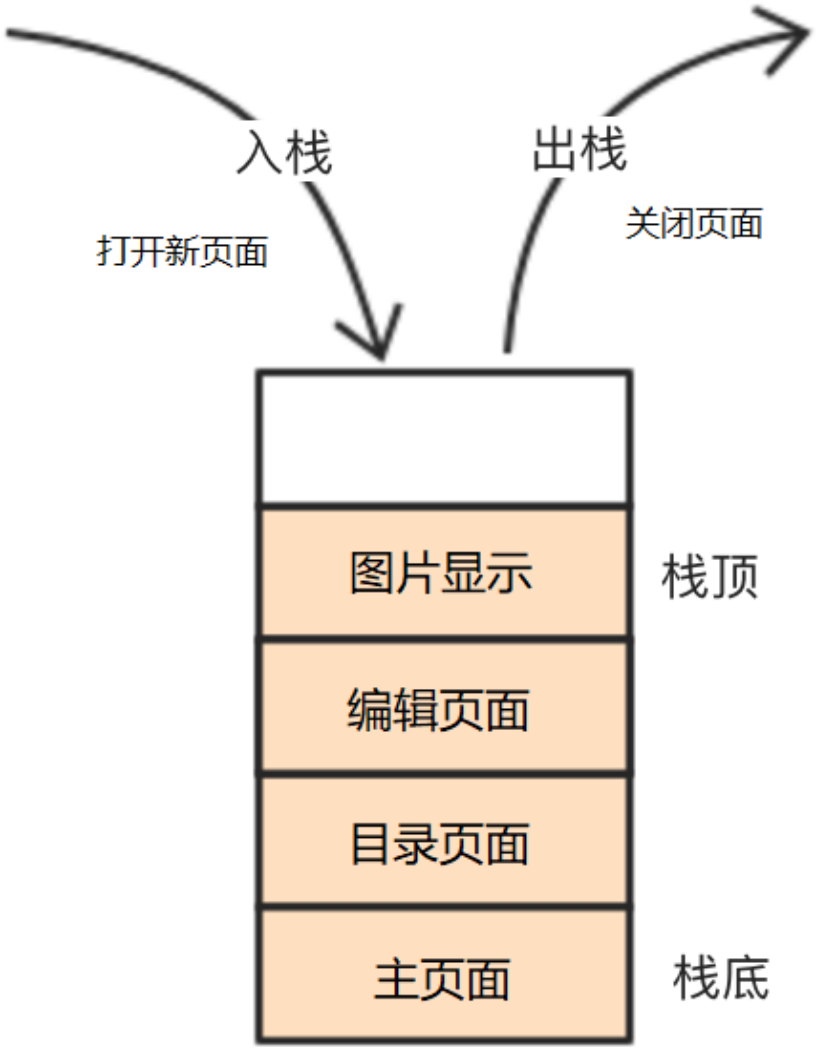

图 3-2　页面栈

3.2　页面框架流程

3.2.1　主页面

主页面需要处理三个用户输入。主要有如下功能。

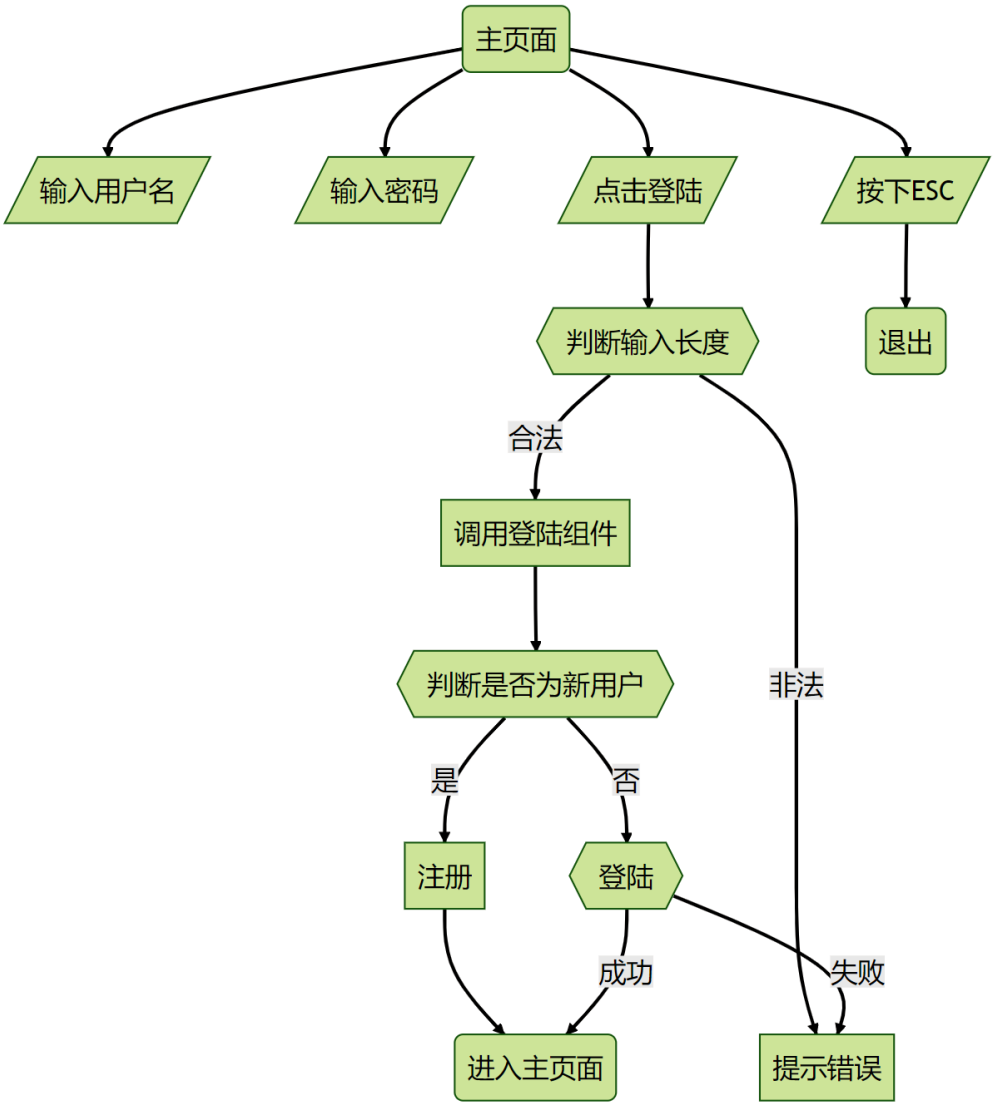

图 3-3　主页面设计流程图

3.2.2　功能页面

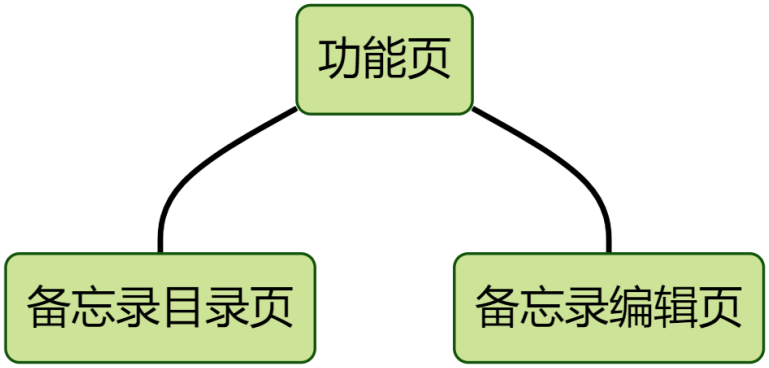

图 3-4　功能页结构图

3.2.2.1 备忘录列表页

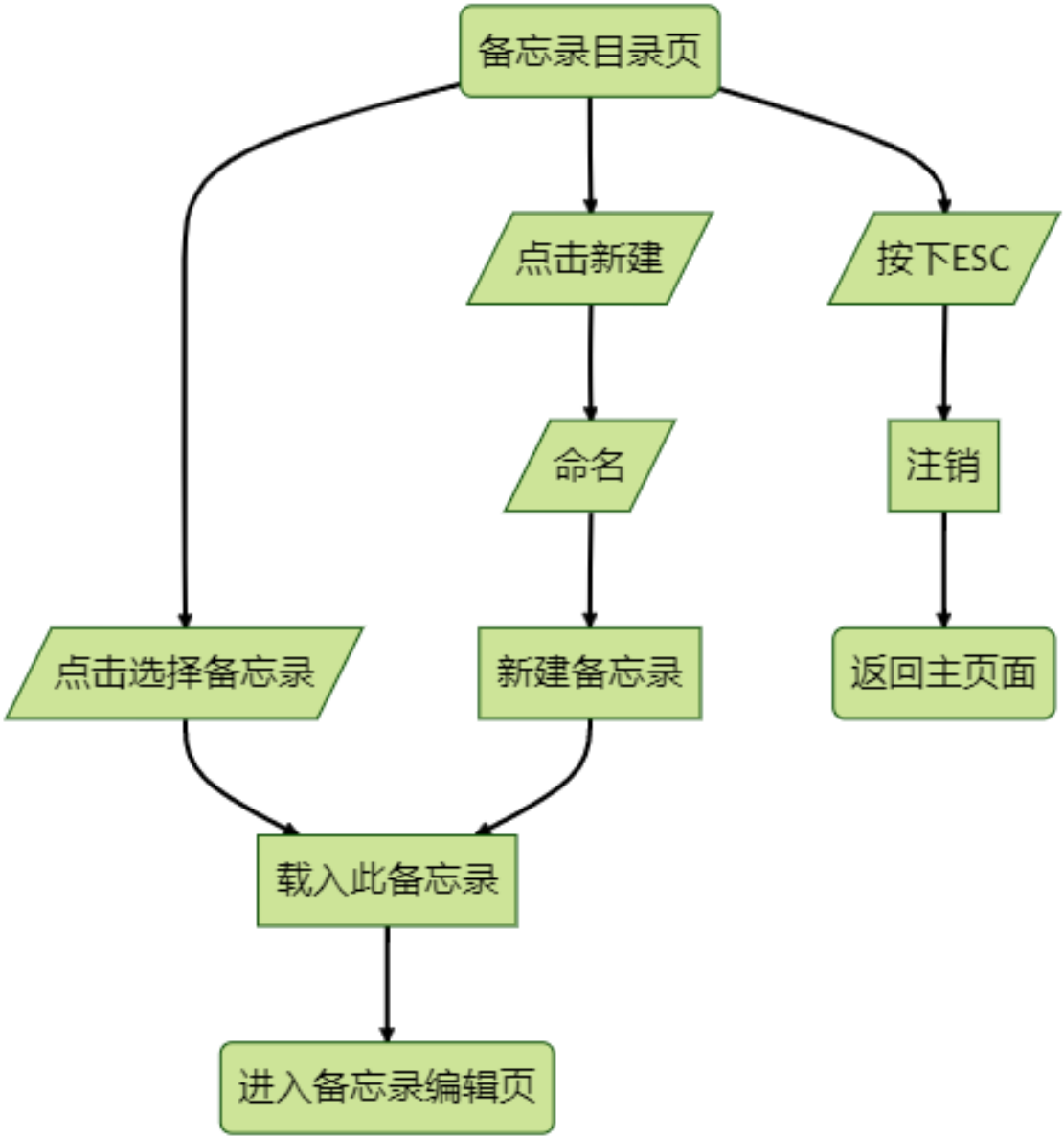

图 3-5　备忘录列表页流程图

3.2.2.2 备忘录编辑页

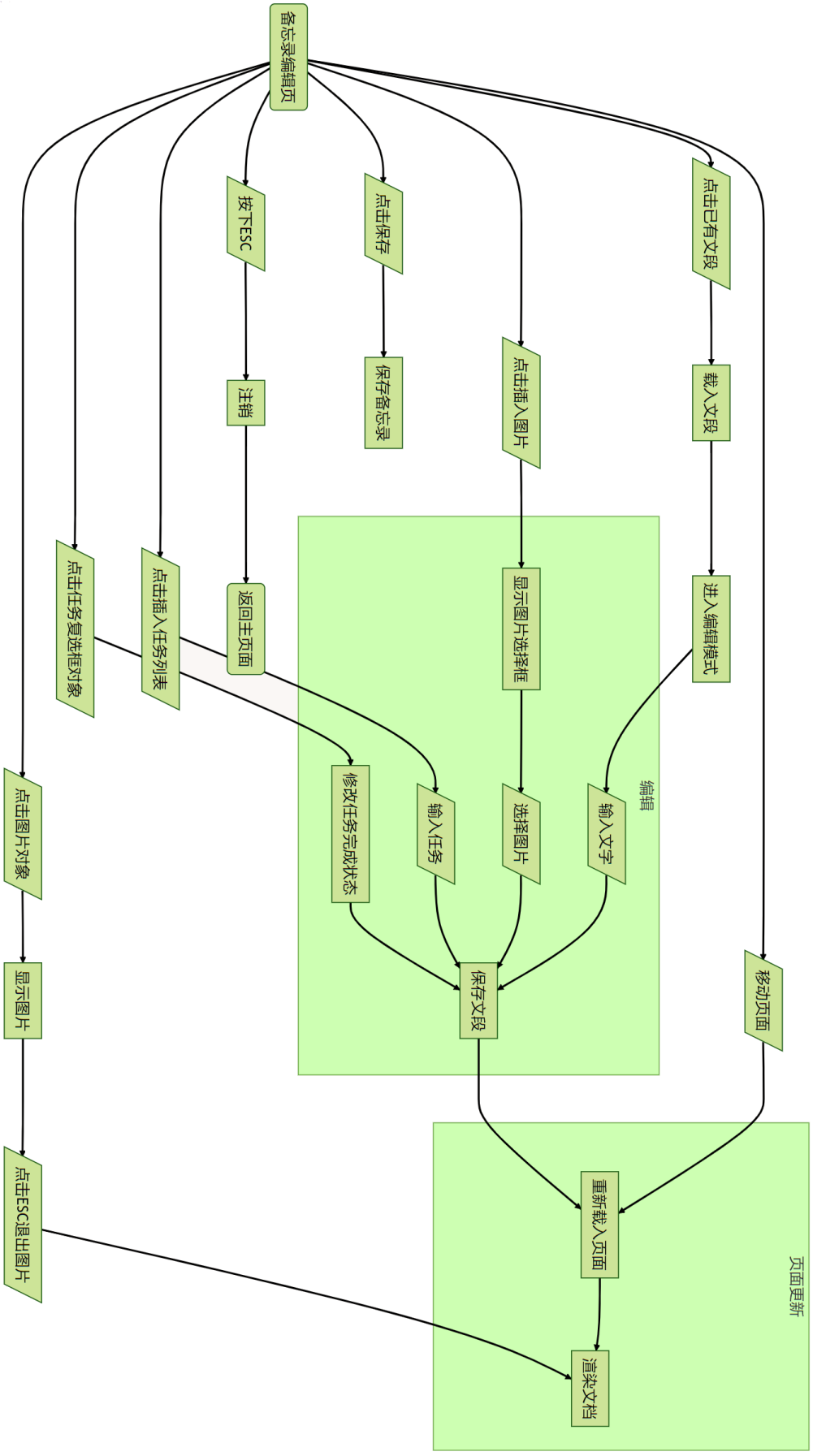

图 3-6　备忘录编辑页流程图

3.2.2.3 画板页

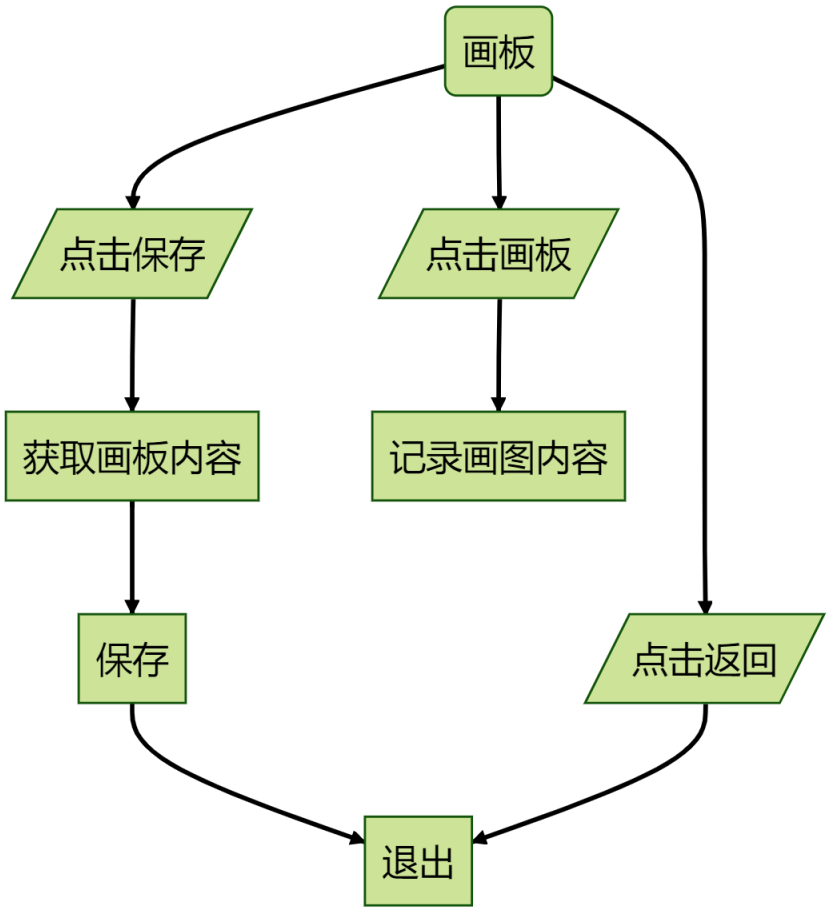

图 3-7　画板页流程图

3.2.2.4 插图页

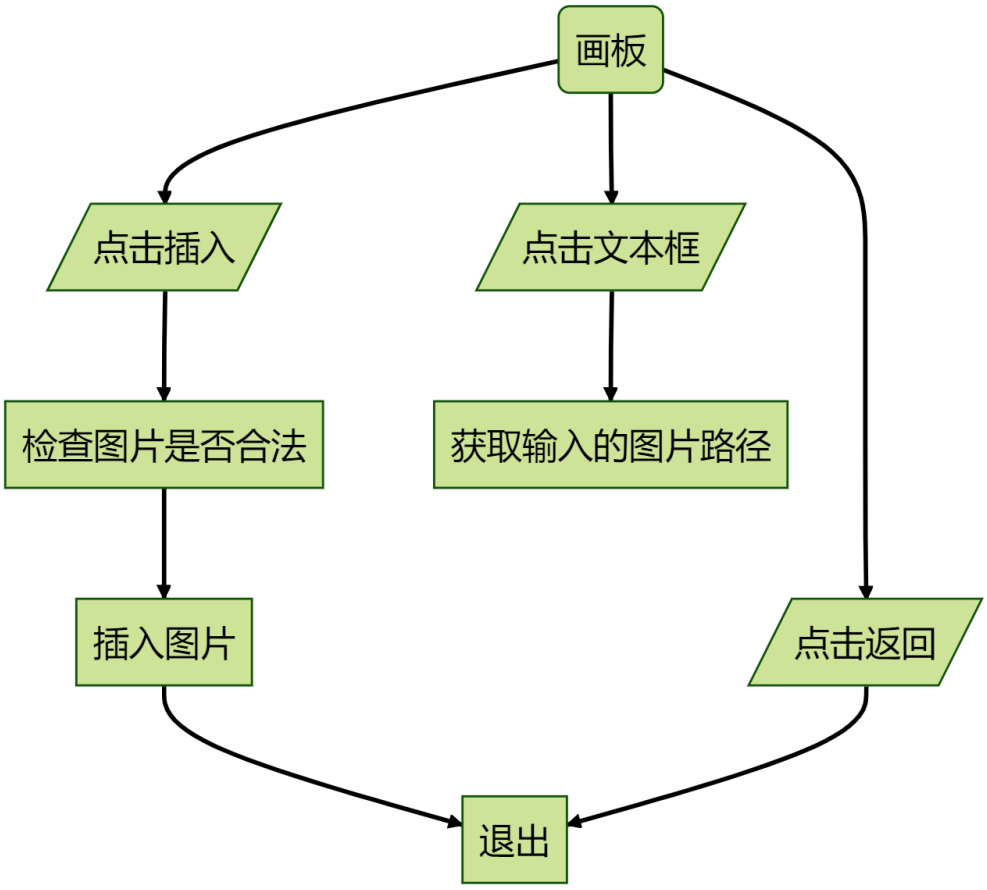

图 3-8　插图页流程图

3.3　后端数据结构

3.3.1　备忘录块

为支持备忘录对文字、图片、任务列表的支持，以及方便相关信息的存储，我们将备忘录分块储存，每块存放一种类型的内容。并通过链表将同一篇备忘录内的备忘录块链接起来，用于存储与读取。

```
enum{IMAGE, PARAGRAPH, CHECKBOX} MemoBlocktype;

typedef struct MemoBlock{
    enum MemoBlocktype type;
    int checkBoxisChecked;
    char content[500];

    struct MemoBlock *next;
}MemoBlock;
```

功能函数原型：

```
MemoBlock *memoBlock_add(MemoBlock *a);
MemoBlock *memoBlock_insert(MemoBlock *p,MemoBlock *a);
MemoBlock *memoBlock_delete(MemoBlock *p);
```

3.3.2 备忘录本体

类似的，我们将整合好的备忘录本体抽象成一个结构体，其中存储了每篇备忘录的相关元数据，也包含了备忘录块链表的头结点数据。 同样使用链表链接。

```
typedef struct Memo{
    char filePath[200];
    char title[100];
    char createUser[200];
    char lastEditUser[200];
    time_t lastEditTime;
    time_t createTime;

    struct MemoBlock *head;
    struct Memo *next;
}Memo;
```

功能函数原型:

```
Memo *memo();
Memo *memo_insert(Memo *p,Memo *a);
Memo *memo_delete(Memo *p);
Memo *memo_makeTop(Memo *p);
```

3.3.3 备忘录列表

所有备忘录的集合，存储了一些基本数据和备忘录本体链表的头结点。

```
typedef struct{
    int count;
    struct Memo *head;
}Memos;
```

功能函数原型:

```
//备忘录列表本体
Memos *memos();
```

3.3.4 文件读写

函数原型:

```
FILE **memofile_current();
void memofile_write(char* filePath, Memo *m);
void memofile_read(Memo *m,char *filePath);
void memofile_writeBlock(MemoBlock *p);
void memofile_readBlock(MemoBlock *p);
```

3.4 基础工具

3.4.1 鼠标

DOS 系统中，可以通过中断51以访问当前鼠标状态。并通过接收到的状态绘制鼠标。

我们参考了往届学长的代码，兼容并蓄，进行了可观的修改，去除了对全局变量的依赖，并添加了一些新功能。

结构体定义：

```
enum CursorStyle{
    CURSORPOINTER = 0,
    CURSORSELECT,
    CURSORTEXT,
    CURSORCROSS
};

enum ClickStatus{
    ClickUnclick,
    ClickLeft,
    ClickRight
};

typedef struct{
    int posX, posY;
    enum CursorStyle style;
    enum ClickStatus click;
    bool visibility;

    void *_buffer;
    int _flag;
}Mouse;
```

函数原型：

```
Mouse *mouse();
void mouse_init();
void mouse_update();
void mouse_chunkUpdate();
void mouse_pageUpdate();
int mouse_isClickedInBox(int, int, int, int);
void mouse_setVisibility(bool);

void _mouse_draw(int, int);
void _mouse_read(int *,int *,int *);
void _mouse_saveBackground(int,int);
void _mouse_clrmous(int,int);
void _mouse_drawmous(int,int);
```

3.4.2 汉字显示

我们对原有学长 `hz.h` GB2312 解码器源码进行了部分修改，使其满足本工程代码命名规范。 最终使用 `graphics.h` 中的 `outtextxy`，并借用了学长 `hz.h` 汉字库，实现了混合文字输出。

GB2312 对汉字进行了分区处理，每个区含有 94 个汉字或者字符，总共有 94 个区，每个汉字或者字符都对应一个分区编号和分区内的位置编号，称为区位码。 区位码对应字符含义如下:

- 01-09区为特殊符号。
- 16-55区为一级汉字，按拼音排序。
- 56-87区为二级汉字，按部首/笔画排序。
- 10-15区及88-94区则未有编码。

GB2312 使用了与上文相同的两字节存储方式，字符的两个字节分别存储了区码和位码。

而区位码对应的字节内容并不是从 `0x80` 开始的，而是 `0xA0`。这二者之间存在一次国标码的转换，在此不再赘述。

以“中”字为例，其区码为54，位码为48。

$$54 + (\mathrm{A0})_{16} = (\mathrm{D6})_{16}$$

$$48 + (\mathrm{A0})_{16} = (\mathrm{D6})_{16}$$

则其 GB2312 编码为 `0xD6 0xD0`。

综上，我们对文字输出及相关字体参数进行封装。 调用函数时，对一段字符串内字符数值进行判断，若转换为 `unsigned char` 类型之后的字符大于255，即说明接下来的两个字符使用GB2312编码，调用 `hz_puthz` 进行输出两个字节，反之，使用 `outtextxy` 输出这个 ascii 字符。

结构定义与函数原型:

```
typedef struct{
    int fontSize;
    int fontColor;
    int spacing;
    int rowSpacing;
}FontFamily;
```

```
typedef struct
{
    char *content;
    int posX;
    int posY;
    int width;
    int hight;
    FontFamily font;
}Text;


void text_display(Text);
int text_getLength(char *s);
Text text_newDefault(char *s, int x1, int y1, int x2, int y2);
char *text_getNthChar(char *s, int n);
int text_getHeight(Text t);
```

3.4.3 按钮

按钮是一个基本模块，我们对其支持高度客制化，可以有多种多样的“按钮”实现。

```
enum ButtonStatus{
    ButtonDefault,
    ButtonFocused,
    ButtonSelected
};

typedef struct Button{
    int posX1;
    int posY1;
    int posX2;
    int posY2;
    char *content;
    enum ButtonStatus status;
    void (*draw)();
}Button;
```

函数原型：

```
Button button_new(int x1, int y1, int x2, int y2, char *s, void (*f)());
void button_draw(Button *);
void button_drawDefault(Button *);
void button_drawWithText(Button *);
int button_event(Button *);
```

`button_event`包含状态检测，可以在鼠标移入时修改状态并切换颜色实现较强视觉反馈。

此外，可以通过修改函数指针`(*draw)()`，指定当前按钮的绘图函数，从而实现文字按钮、图片按钮等多态。

3.4.4 文本框

```
enum TextboxStatus{
    TextboxDefault,
    TextboxFocused,
    TextboxSelected
};

enum TextboxMouseStatus{
    TextboxMouseDefault,
    TextboxMouseFocused,
    TextboxMouseClicked
};

typedef struct{
    char *defaultContent;
    int posX1;
    int posY1;
    int posX2;
    int posY2;
    char *content;
    int maxLength;
    FontFamily font;
    enum TextboxStatus status;
    enum TextboxMouseStatus mstatus;
    int cursorLocation;
    int cursorStatus;
    clock_t cursorLastBlink;
}Textbox;
```

函数原型:

```
void textbox_draw(Textbox *tb);
int textbox_event(Textbox *tb);
void textbox_determinState(Textbox *tb);
Textbox textbox_newDefault(char *ds, int x1, int y1, int x2, int y2,
char *buffer);
Text textbox_convert2text(Textbox tb);
int textbox_getCursorPositionX(Textbox t);
int textbox_getCursorPositionY(Textbox t);
```

我们实现了较为完备的文本框，使用起来与现代应用别无二致。具体特性如下。

3.4.4.1 多进程支持

本项目文本框的设计完全符合上文中“组件化”规范，可以在选中文本框、输入文字的同时进行其他事件的处理。

相见函数 `textbox_event`。

3.4.4.2 选择光标

我们对文本框加入了选择光标，以表示当前输入的操作位置。这个光标可以对键盘 `left`、`right` 按键作出反应，也可通过鼠标点击切换位置。

为明显显示光标位置，选择光标将以 T=1.2S 的周期闪烁。

在实现中，我们使用 `clock_t clock()` 函数记录当前CPU时钟，将上一次闪烁时间保存为 `cursorLastBlink`，并判断当前时间与上一次闪烁时间的差值是否达到切换闪烁状态的标准，若达到，则重新绘制文本框，以切换当前画面上显示的状态。

光标的状态处理位于 `textbox_event` 中，绘制相关代码位于 `textbox_draw` 中。

3.4.4.3 对输入法的支持

本文本框支持使用输入法输入。并且输入法文本框可以跟随光标移动。有关输入法的事项详见下文。

3.4.5 输入法

专有类型和结构体定义：

```
enum imeStatus{
    IMEOFF,
    IMEPINYIN,
    IMEEMOJI
};

typedef struct
{
    enum imeStatus status;
    Button button;
}Ime;
```

结构体记录了其当前输入模式：关闭（即英文键盘输入）、汉字拼音输入法、emoji表情输入法，以及其切换按钮。

函数原型：

```
void ime_init();
int ime_input(char *s, int x, int y);
int ime_en(char *s);
void ime_check();
void ime_next();
void ime_draw();
```

此外，在 ime_check 中，我们通过 bioskey(2) 监测 ctrl 键状态，实现 ctrl+space 的切换输入法快捷键。

3.4.5.1 英文输入

在 ime_en 中我们使用 bioskey(1) 与 bioskey(0) 获取输入的内容。支持键盘可以输入的全部 ascii 值。

3.4.5.2 汉字输入

我们在前辈原有 hz_input 的基础上进行了修改，并使之兼容本项目新的输入法模块。其代码保留在 hzinput.h 中。

另外，我们添加了如下标点符号映射表，以完成对全角符号输入的支持。

特别的，我们支持成对标点符号的自动补全。如输入 < 时返回 《》，并将光标置于成对的标点符号之间。

4　界面设计

4.1　登陆界面

图示为备忘录系统的登陆界面，开头进入时会采用颜色渐变的方法来营造登录的加载过程，然后是全屏的贴图作为背景，紧接着是一个非线性的变换出现一个过渡的背景色，对此我们在256色中采用了最贴近的暖色调。在非线性动画结束后是右侧的贴图。

登陆界面设计如下。

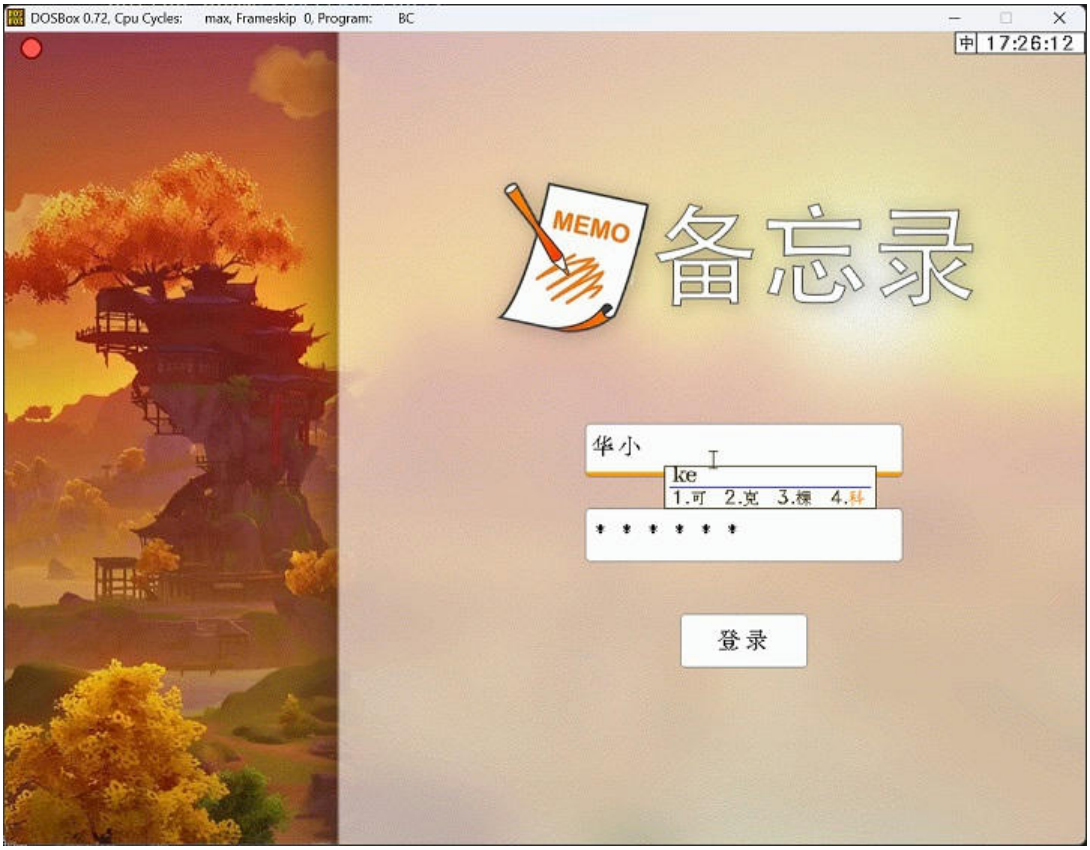

图 4-1　登录界面

动画结束后，会出现两个文本框和一个用来登录的按钮。其中登录的按钮在点击时呈现出不同的颜色以此优化用户的点击效果体验。两个文本框为了美观效果采用了圆角设计，在点击选中时，文本框会出现如图的下划线阴影效果，同时输入时会存在着光标显示输入的位置。

用于输入登录信息的两个文本框在未被选中操作时会显示如图效果。

在输入密码时为了保护用户隐私，我们采用以“*”的形式显示密码。在确定登录时，如若账号已注册过则会根据密码输入是否正确，分别以文本的形式显示登录成功或是密码错误， 若是没有注册过进行登录操作时会自行注册，并以文本的形式显示注册成功。

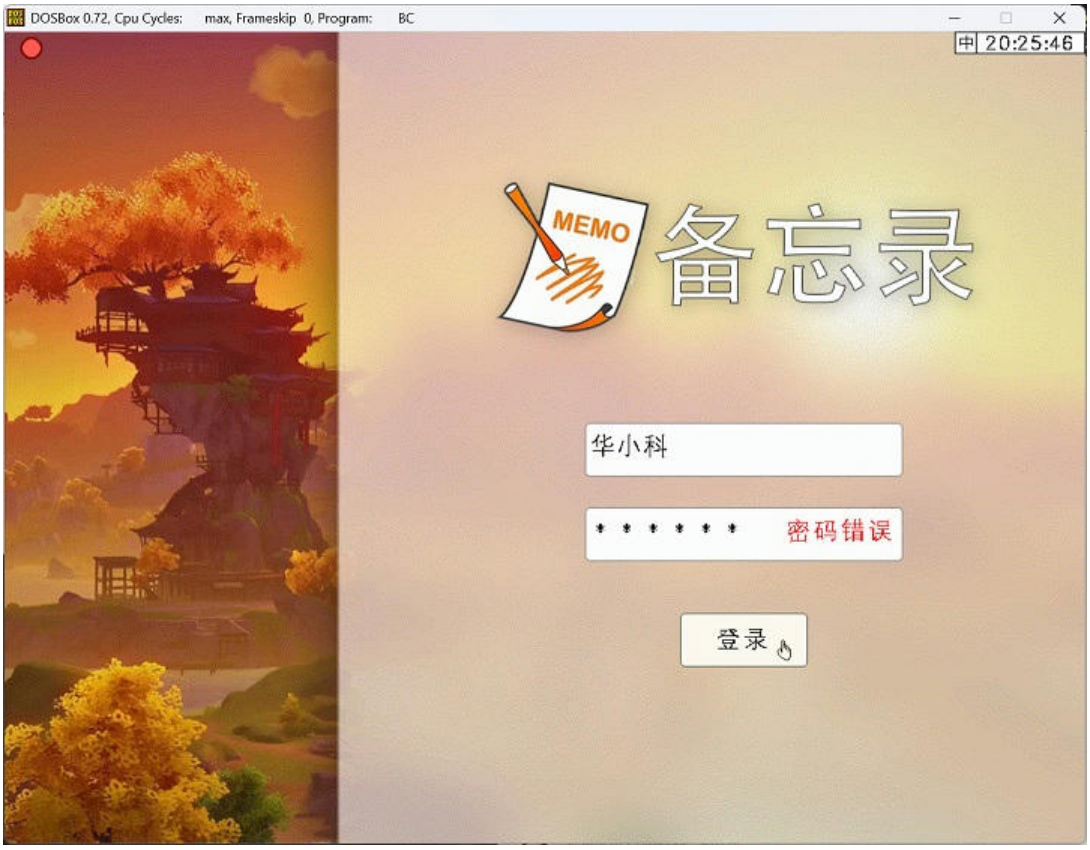

图 4-2　登录界面异常提示界面

另外，画面右上角显示当前时间及输入法状态。输入法支持中英文的切换，可点击或者按住 ctrl+空格 来快捷键切换。 在左上角我们设置了一个用来点击退出的红色圆形按钮。

为了方便用户的使用，我们除了切换输入法的快捷键ctrl+空格，还有用于直接返回的“ESC”键，同时在输入账号完毕时，可以直接按“ENTER”键转到输入密码的文本框进行操作，在输入密码完毕后可以按“ENTER”快捷键或者直接点击“登录”按钮进行登陆操作。

4.2　主界面

在登录后会进入App界面，左边栏上方为红色圆形按钮用于退出，同时也可以用“ESC”快捷键退出。

主界面的左侧为边栏，右侧为编辑器。

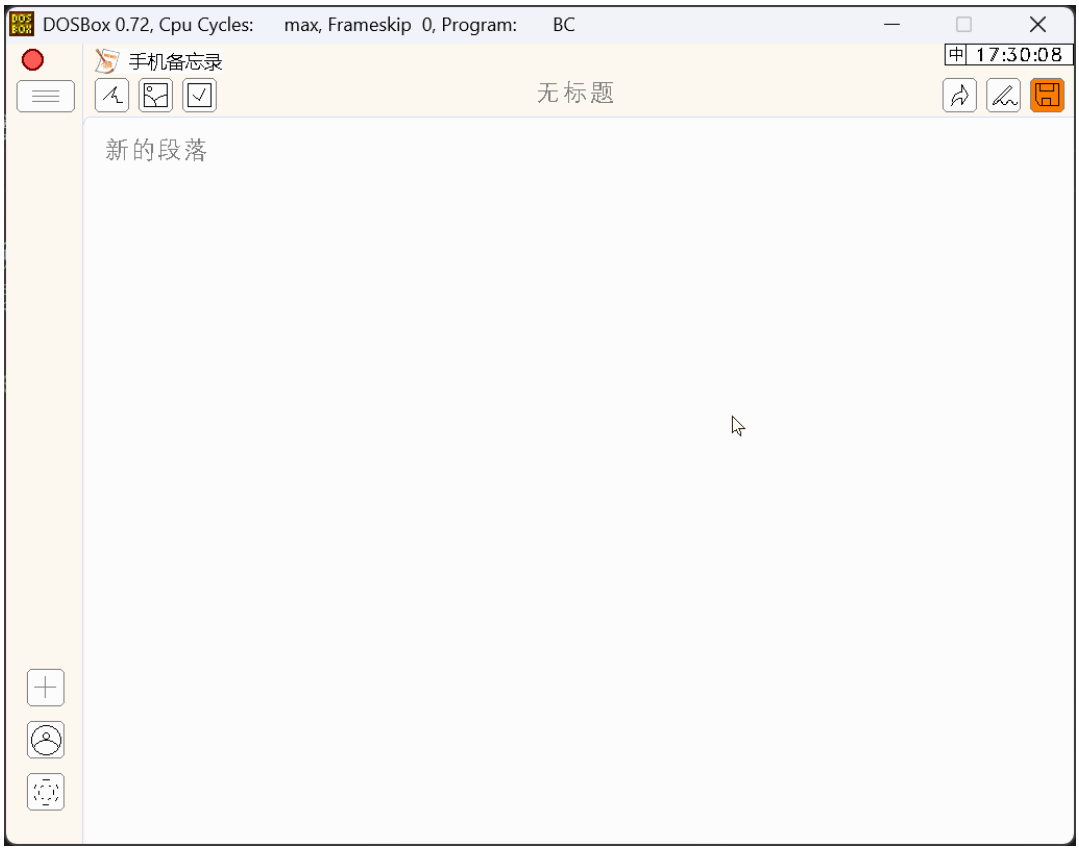

图 4-3　备忘录主界面

默认情况下，当用户进入时，会自动创建一篇新的备忘录供用户记录信息。

4.2.1　侧边栏

打开侧边栏，下方是用来打开备忘录的列表，如图所示可以通过点击来切换到不同的备忘录来进行操作。 备忘录的靠左侧会显示不同的颜色来标识该条备忘录的重要程度。

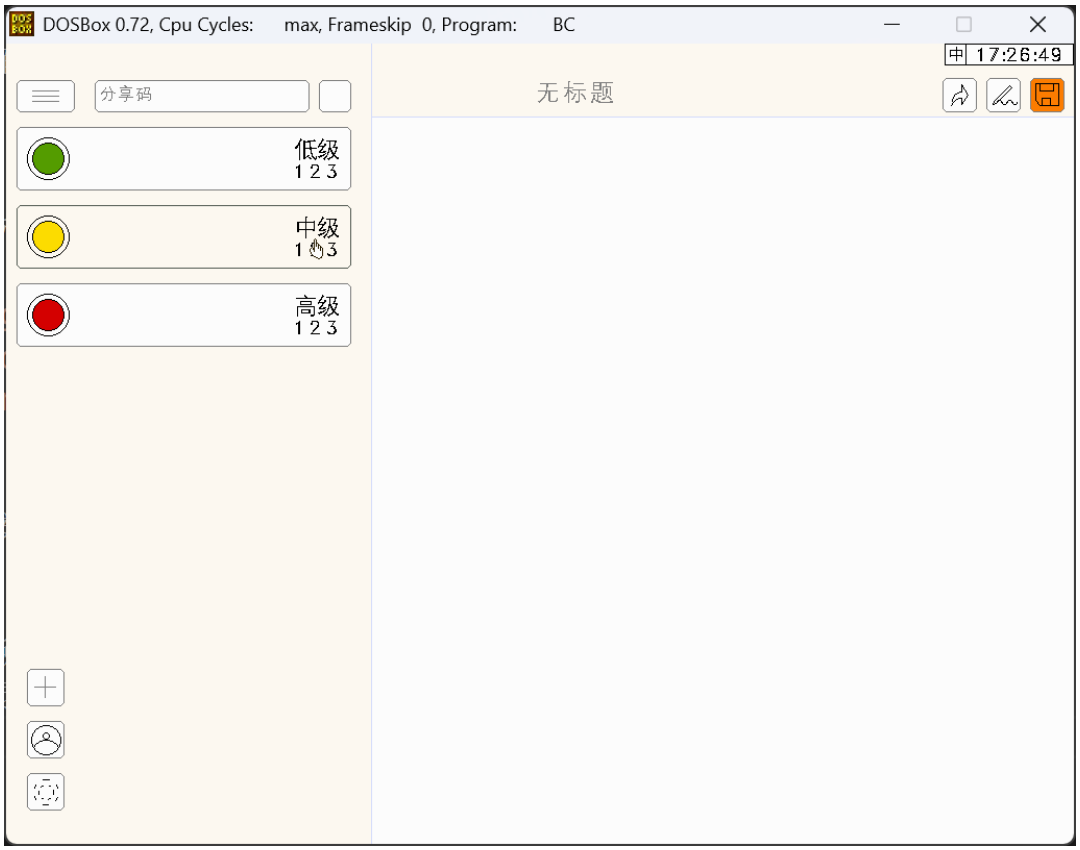

图 4-4　备忘录侧边栏列表及重要程度显示界面

点击左下方加号可以新建一条备忘录。

点击上方文本框与按钮，即可通过分享码导入他人的文章。

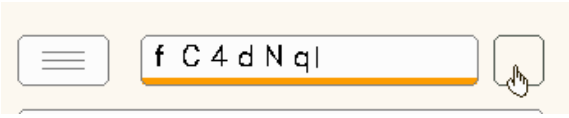

图 4-5　通过分享码导入文章

此外，点击下方的按钮，可以切换显示模式，选择显示/隐藏上一位编辑用户。

图 4-6　最后编辑用户显示

4.2.2　备忘录编辑器

在上方的边栏，由左向右的三个按钮分别是画板功能、添加图片功能、待办功能。

1. 画板功能：在点击后会出现如图的弹窗，可以操作鼠标在白板上按住鼠标左键自由绘画，用右键擦除，完毕后可以点击插入的按钮插入到备忘录内容中。

图 4-7　画板功能界面

2. 插图功能：

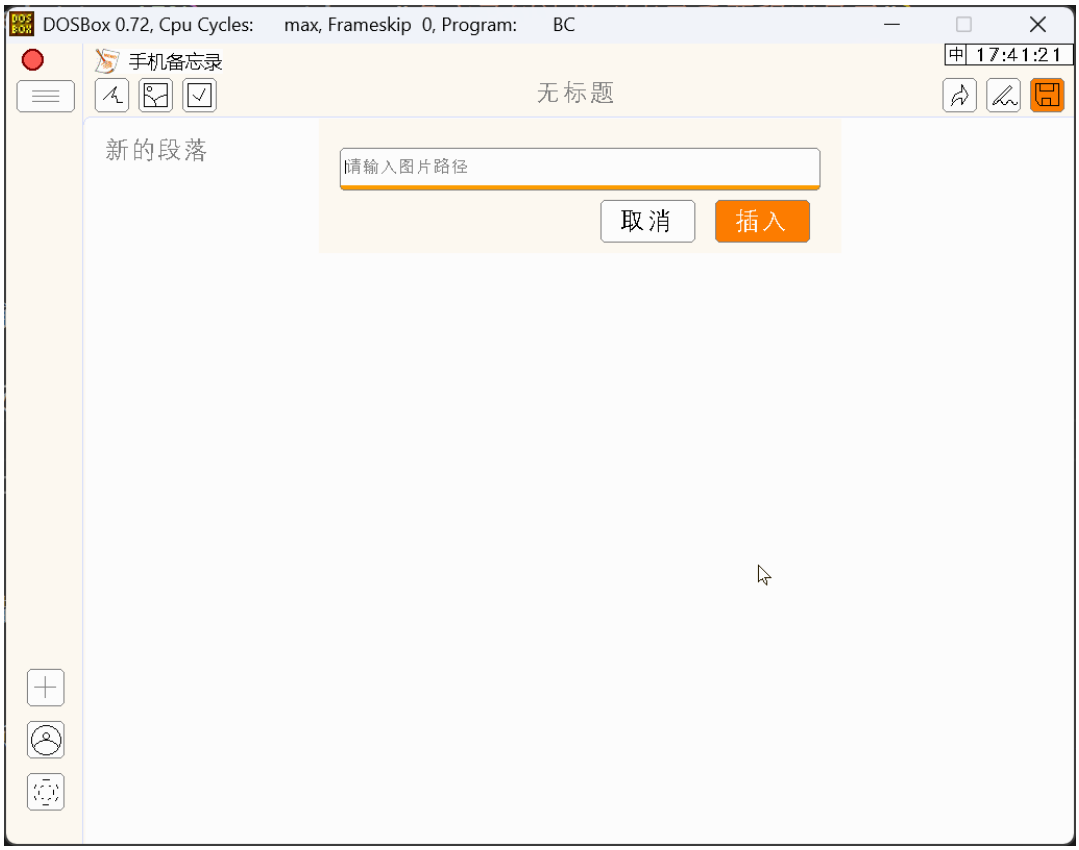

图 4-8　插图功能界面

在文本框里面输入图片地址即可插入，若是输入错误则会显示路径不合法。

图 4-9　插图失败界面

图 4-10　插图功能成功界面

3. 待办功能： 会为该行内容生成一个新的任务，如果完成可以点击圆角方框表示已完成。

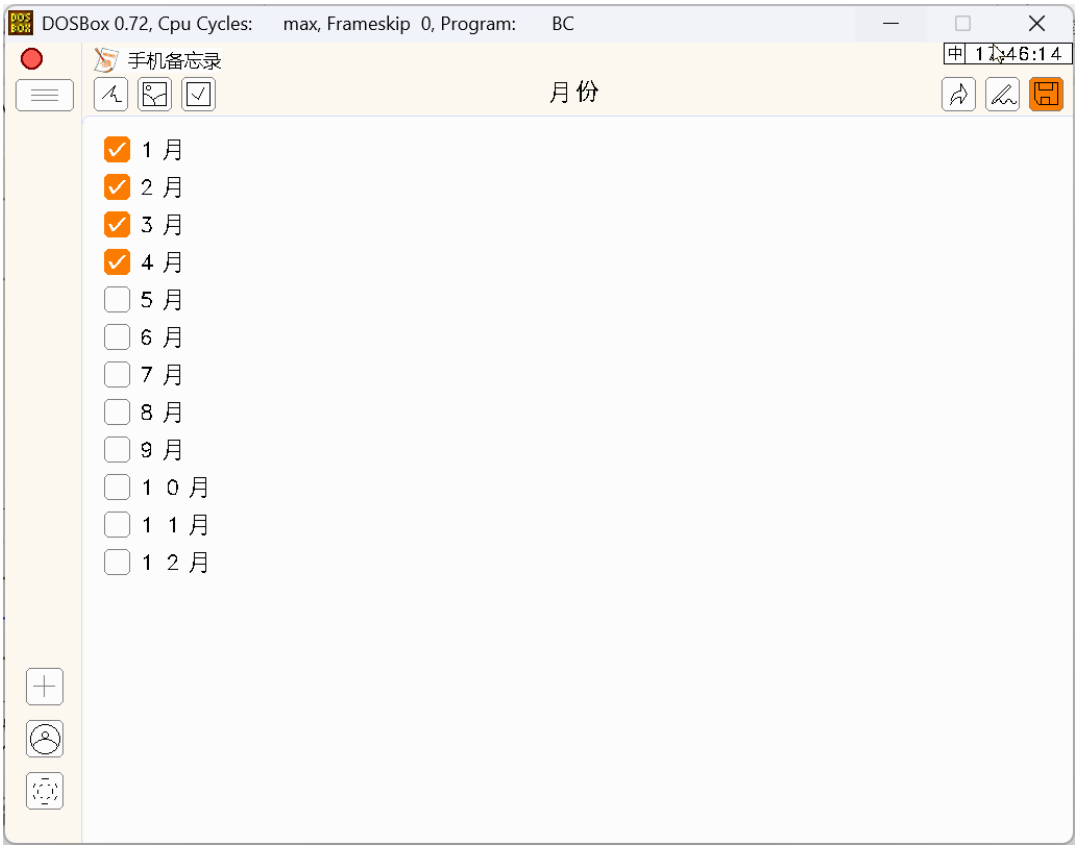

图 4-11　待办功能界面

上方边栏中间为标题，点击后可以在文本框中自由输入标题内容，支持中英文。

右方三个圆角方框依次代表分享功能、备忘录设置功能、保存功能。

1. 分享功能： 分享码的有效期分为永远有效，1天和自定义时间等形式。 在自定义输入时间时，如果输入的非数字则会显示输入形式错误。 输入时间后可以点击获取按钮获取一个随机的分享码，完毕后可点击关闭按钮退出。

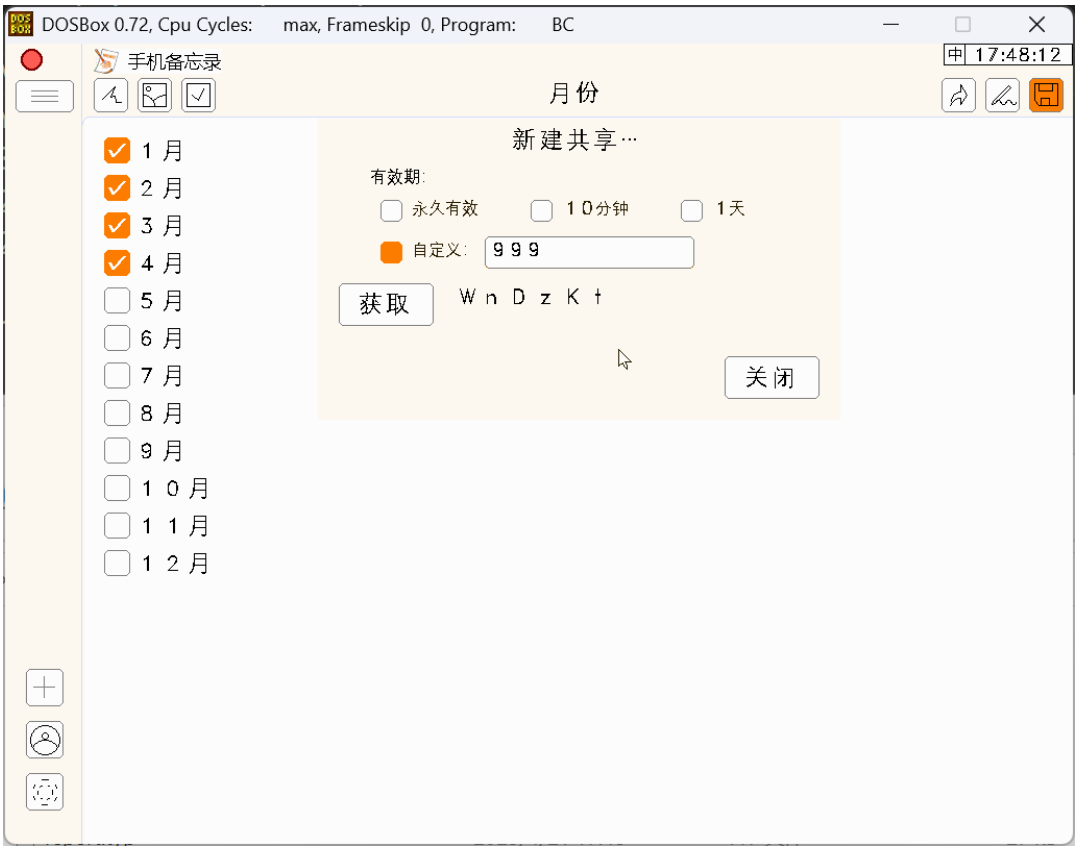

图 4-12　分享功能界面

2. 备忘录设置功能： 可以设置备忘录中事件的重要程度，分为三级，从重要到不重要依次分别用红色，黄色，绿色表示。还可以设置备忘录的权限属性，是属于共有还是私有还是白名单类型。 如果是白名单类型则需要分享码才可以成功打开操作。

图 4-13　备忘录设置功能界面

3. 保存功能： 显示为一个橙色圆角方框按钮，点击后即可保存备忘录的内容和标题。

4.2.3　图片预览界面

双击编辑页面中的图片，即可进入该图片的预览界面。

在该界面中，图片将默认以100%的缩放模式展示。界面右下角会显示当前的缩放比例。

点击左上角的+/- 号，可以得到图片不同比例的缩放预览。

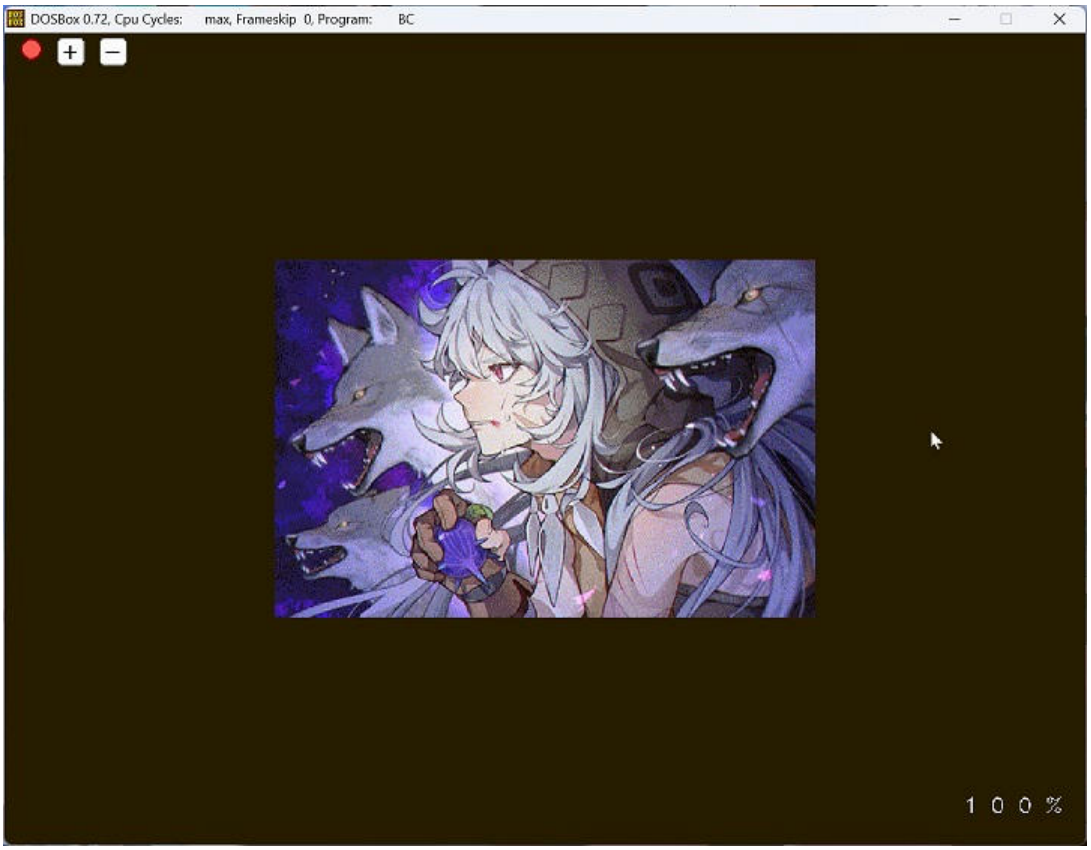

图 4-14　图片预览页面

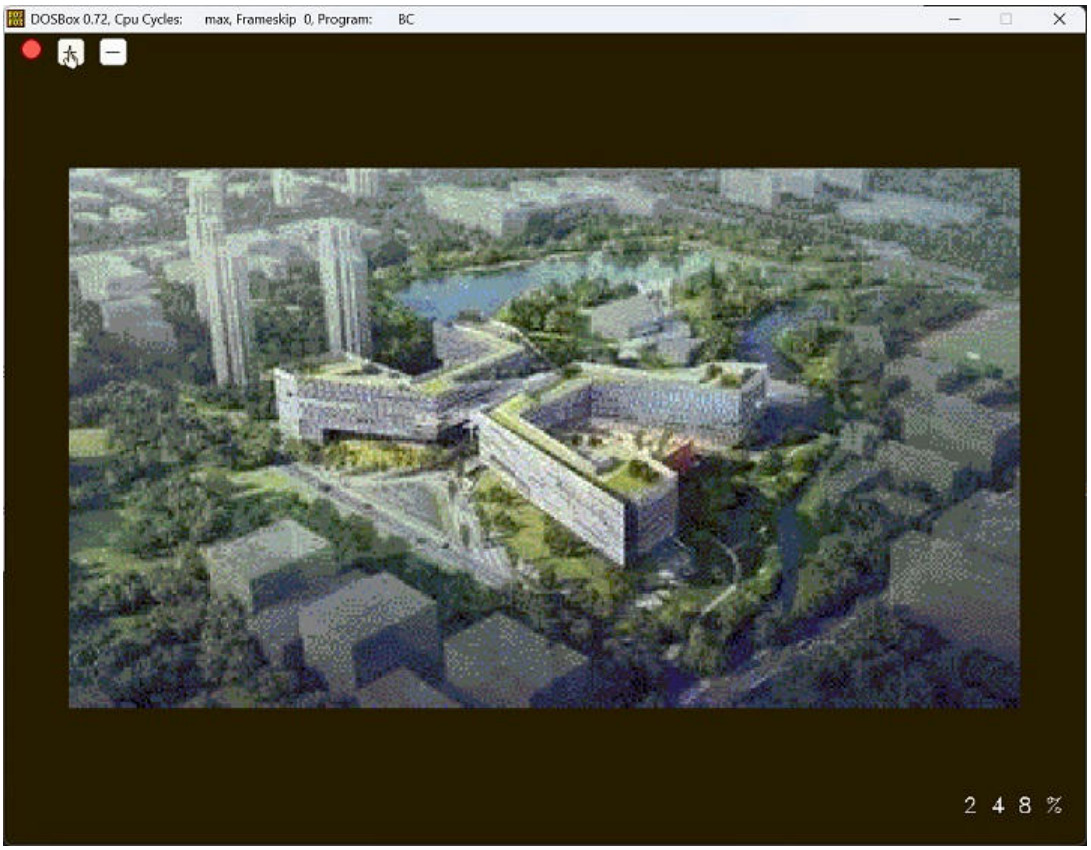

图 4-15　图片预览页面放大展示界面

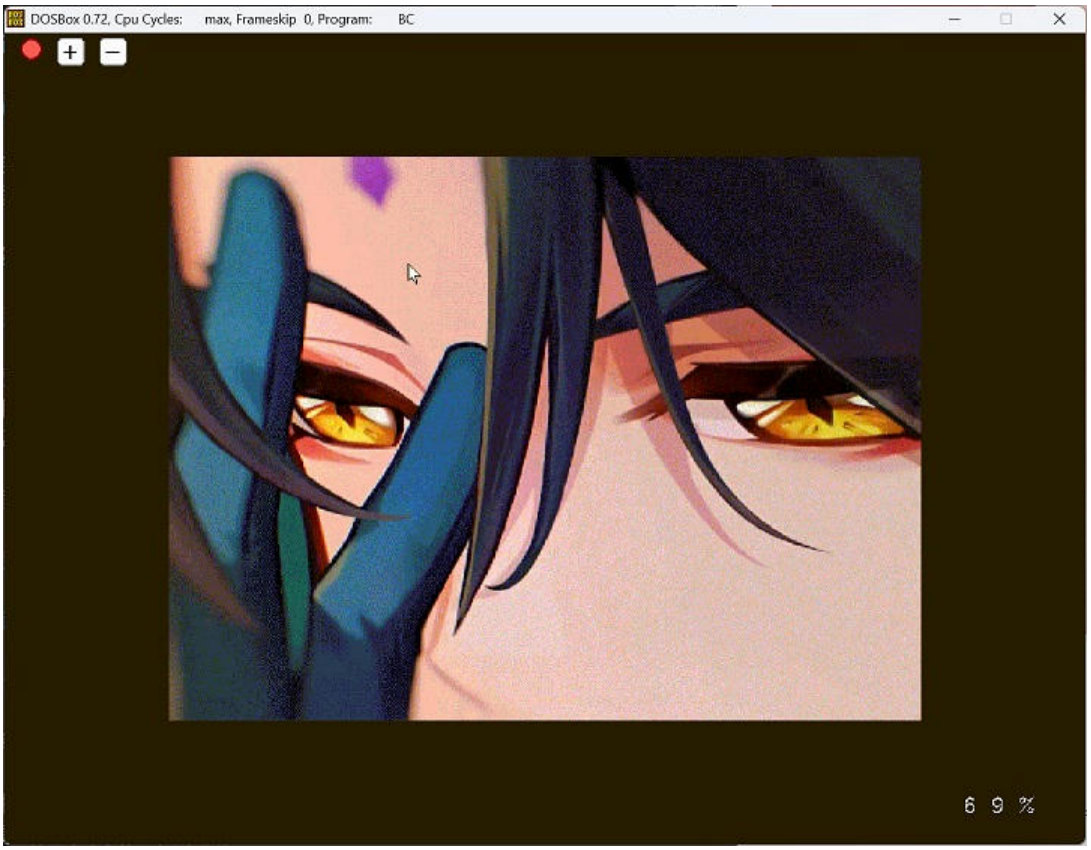

图 4-16　图片预览页面缩小展示界面

4.2.4　未保存退出确认页面

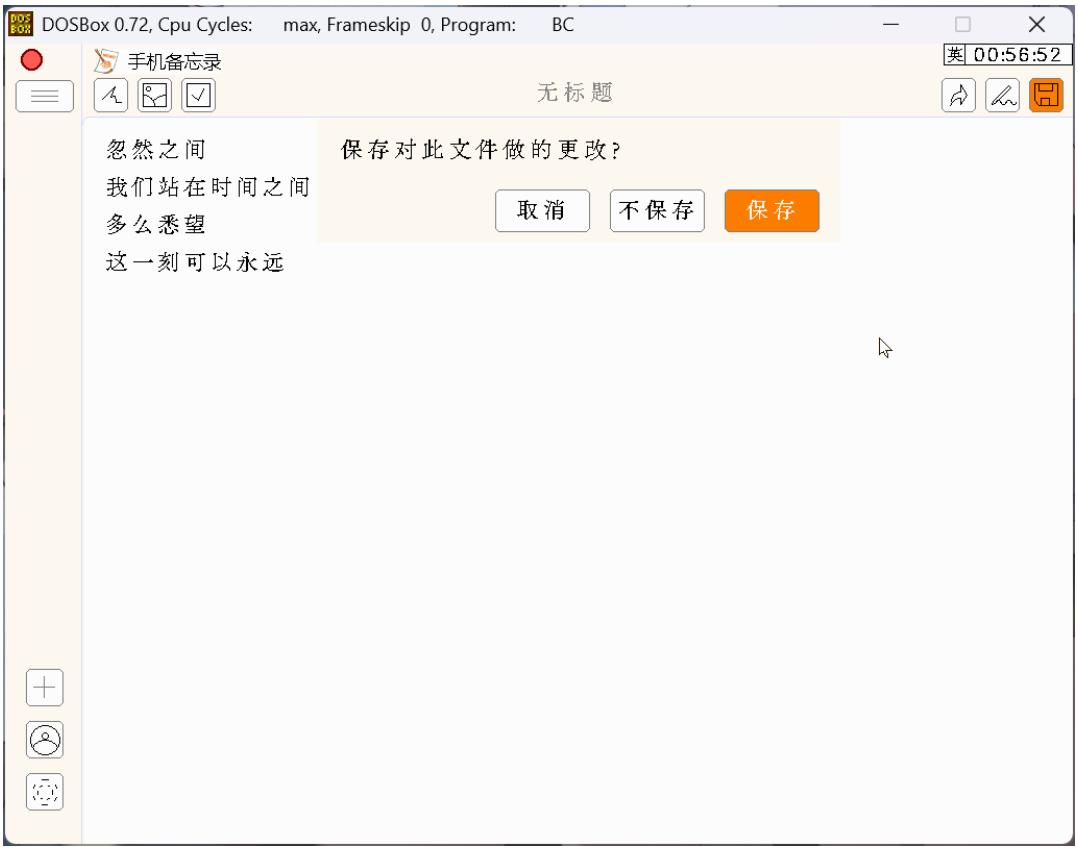

图 4-17　图片预览页面

4.2.5　用户中心

图 4-18　用户中心界面

画面中的圆形图案代表着用户，下方相应着显示先前在主页面登录的用户的账号。此外下方还有一个用来添加的按钮，每次点击都会跳转到输入用户的账号和密码的界面，然后在用户中心的界面显示添加的用户的账号。由于最大登录人数为五人，当存在了五名用户时，用于添加用户的按钮会随之消失。

而对于跳转到的输入界面，会存在两个文本框用来输入账号和密码。当文本框留空时，则会有提示语用来解释该文本框的用途。为保证用户隐私，输入密码将以*的形式显示。下方有一个登录的按钮，按下后会根据用户数据库来判断是账号已存在还是新注册用户，如若是账号已存在会弹出文本来提示用户重新输入，如果是新用户会显示注册成功并返回到用户中心界面。

在用户中心还存在着删除、退出登录的功能，在鼠标移到用户账号右方特定位置时，会显示一个x的图标，若是点击则会退出该用户的登陆状态。旁边显示的是切换账号的按钮， 点击后会将该账号替换到首位来作为当前登录的用户。

5 源代码

5.1 头文件

5.1.1 addimage.h

```
/**
 * @file addimage.h
 * @author Hibanaw Hu (hibanaw@qq.com)
 * @brief
 * @date 2023-04-19
 *
 * @copyright Copyright (c) 2023
 *
 */

#ifndef __ADDIMAGE_H__
#define __ADDIMAGE_H__

int addImage(char *path);

#include <graphics.h>
#include "svga.h"
#include "mouse.h"
#include "button.h"
#include "textipt.h"
#include "image.h"
#include "digclock.h"
#endif
```

5.1.2 anim.h

```
/**
 * @file anim.h
 * @author wywgwt (2504133124@qq.com)
 * @brief
 * @date 2023-04-03
 *
 * @copyright Copyright (c) 2023
 *
 */
#ifndef __ANIMATION_H__
#define __ANIMATION_H__

#include<graphics.h>
#include<time.h>
#include"svga.h"
#include<math.h>

void animation_login();
```

```
void animation_homepage1();

void animation_homepage2();

#endif
```

5.1.3 app.h

```
/**
 * @file app.h
 * @author Hibanaw Hu (hibanaw@qq.com)
 * @brief
 * @date 2023-02-27
 *
 * @copyright Copyright (c) 2023
 *
 */

#ifndef __APP_H__
#define __APP_H__


enum{
    AppRouterExpand = 1,
    AppExit,
    AppRedraw,
    AppChangeMemo,
    AppUserPage
};

typedef struct
{
    char uid[6][20];
    char *currentUser;
    char displayLastEditUser;
    int userCount;
}AppData;

AppData *appData();

void app();

#include <graphics.h>
#include "svga.h"
#include "keyboard.h"
#include "mouse.h"
#include "global.h"
#include "textbox.h"
#include "image.h"
#include "anim.h"
#include "meditor.h"
#include "router.h"
#include "drawpad.h"
#include "userpage.h"
#include "exitsave.h"
#endif
```

5.1.4 auth.h

```c
/**
 * @file auth.h
 * @author Hibanaw Hu (hibanaw@qq.com)
 * @brief
 * @date 2023-04-19
 *
 * @copyright Copyright (c) 2023
 *
 */

#ifndef __AUTH_H__
#define __AUTH_H__

enum AuthType{
    AUTHPRIVATE,
    AUTHPUBLIC,
    AUTHWHITELIST
};

typedef char AuthUser[30];

typedef struct
{
    AuthUser owner;
    enum AuthType type;
}Auth;

int auth_check(char *filename, char *uid);
int auth_set(char *filename, char * uid, enum AuthType type);
int auth_addWhiteList(char *filename, char *uid);
Auth auth_get(char *filename);
#include <stdio.h>
#include <string.h>

#endif
```

5.1.5 button.h

```c
/**
 * @file button.h
 * @author Hibanaw Hu (hibanaw@qq.com)
 * @brief
 * @date 2023-02-28
 *
 * @copyright Copyright (c) 2023
 *
 */

#ifndef __BUTTON_H__
#define __BUTTON_H__

#include <time.h>
```

```
enum ButtonStatus{
    ButtonDefault,
    ButtonFocused,
    ButtonSelected
};

typedef struct Button{
    int posX1;
    int posY1;
    int posX2;
    int posY2;
    char *content;
    enum ButtonStatus status;
    clock_t animationStartTime;
    clock_t animationTick;
    clock_t animationDrawedTick;
    void (*draw)();
}Button;

Button button_new(int x1, int y1, int x2, int y2, char *s, void (*f)());
void button_draw(Button *);
void button_drawDefault(Button *);
void button_drawWithText(Button *);
int button_event(Button *);
void button_drawWINUIAccent(Button *b);
void button_drawWINUI(Button *b);
void button_checkboxAnimation(Button *cb);
int button_checkboxEvent(Button *cb);
void button_checkboxDraw(Button *cb);
void button_drawExitButton(Button *b);
Button button_newExitButton();
#include <graphics.h>
#include "global.h"
#include "svga.h"
#include "mouse.h"
#include "button.h"
#include "text.h"
#endif
```

5.1.6 digclock.h

```
/**
 * @file digclock.h
 * @author Hibanaw Hu (hibanaw@qq.com)
 * @brief
 * @date 2023-04-03
 *
 * @copyright Copyright (c) 2023
 *
 */

#ifndef __DIGCLOCK_H__
#define __DIGCLOCK_H__


void digitalClock_getTime();
```

```
#include <time.h>
#include <graphics.h>
#include "svga.h"
#include "text.h"
#include "mouse.h"

#endif
```

5.1.7 drawpad.h

```
/**
 * @file drawpad.h
 * @author Hibanaw Hu (hibanaw@qq.com)
 * @brief
 * @date 2023-04-16
 *
 * @copyright Copyright (c) 2023
 *
 */

#ifndef __DRAWPAD_H__
#define __DRAWPAD_H__

int drawPad(char *saveFilePath);

#include <time.h>
#include <svga.h>
#include <button.h>
#include <text.h>
#include <bios.h>
#include "keyboard.h"
#include "dir.h"
#endif
```

5.1.8 exitsave.h

```
/**
 * @file exitsave.h
 * @author Hibanaw Hu (hibanaw@qq.com)
 * @brief
 * @date 2023-04-21
 *
 * @copyright Copyright (c) 2023
 *
 */

#ifndef __EXITSAVE_H__
#define __EXITSAVE_H__

#include "keyboard.h"
#include "mouse.h"
#include "button.h"
#include "meditor.h"
```

```
#endif
```

5.1.9 global.h

```
/**
 * @file global.h
 * @author Hibanaw Hu (hibanaw@qq.com)
 * @brief
 * @date 2023-02-07
 *
 * @copyright Copyright (c) 2023
 *
 */

#ifndef __GLOBAL_H__
#define __GLOBAL_H__


#define MAXHEIGHT 768
#define MAXWIDTH 1024

#define MAX(x, y) ((x) > (y) ? (x) : (y))
#define MIN(x, y) ((x) < (y) ? (x) : (y))
#define ABS(x) ((x) > 0 ? (x) : -(x))
#define CEILING(f) ((int)(f) + (((f) - (int)(f) > 0) ? 1 : 0))

typedef enum{false, true} bool;

#include "debug.h"

#endif
```

5.1.10 homepage.h

```
/**
 * @file homepage.h
 * @author wywgwt (2504133124@qq.com), Hibanaw Hu (hibanaw@qq.com)
 * @brief
 * @date 2023-02-12
 *
 * @copyright Copyright (c) 2023
 *
 */

#ifndef __HOMEPAGE_H__
#define __HOMEPAGE_H__


void homepage();

#include "image.h"
#include "keyboard.h"
#include "mouse.h"
```

```
#include "app.h"
#include "button.h"
#include "textbox.h"
#include "text.h"
#include "textipt.h"
#include"userpage.h"
#include<string.h>
#endif
```

5.1.11 hz.h

```
#ifndef __HZ_H__
#define __HZ_H__

#include <graphics.h>
#include<conio.h>
#include<stdio.h>
#include<stdlib.h>
#include<string.h>

#include"global.h"

void hz_puthz(char *s, int x, int y, int flag,int part,int color);

void hz_puthzold(int x, int y, char *s, int flag,int part,int color);

#endif
```

5.1.12 hzinput.h

```
#ifndef _HZINPUT_H_
#define _HZINPUT_H_

//以下头文件如总头文件中已包含则不用，在input.c中包括头文件即可
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <graphics.h>
#include <bios.h>
#include <conio.h>
#include <dos.h>
#include "hz.h"
#include "mouse.h"
#include "keyboard.h"
#include "text.h"
#include "global.h"
#include "ime.h"
#include "digclock.h"

#define FAIL 0
#define SUCCESS 1

int hzinput(int x,int y, char *s);
//汉字输入法
```

```
int input_method(int x,int y,char *str,int value,char *py) ;
//汉字输入法调入

char *itostr(int a,char *s);
//数字标号

void pyFrm(int x1,int y1,int x2,int y2);
//输入法小框

int xouttextxy(int x,int y,char *s,int color);
//字符输入法

#endif
```

5.1.13 image.h

```
/**
 * @file image.h
 * @author Hibanaw Hu (hibanaw@qq.com)
 * @brief
 * @date 2023-02-07
 *
 * @copyright Copyright (c) 2023
 *
 */

#ifndef __IMAGE_H__
#define __IMAGE_H__


void image_render(char *, int, int);
void image_renderZoom(char * filePath, int x, int y, float scale);
void image_getSize(char *, int*, int *);
int image_illegal(char *path);


#include <stdio.h>
#include <stdlib.h>
#include <graphics.h>
#include "global.h"

#endif
```

5.1.14 imagebox.h

```
/**
 * @file imagebox.h
 * @author Hibanaw Hu (hibanaw@qq.com)
 * @brief
 * @date 2023-04-06
 *
 * @copyright Copyright (c) 2023
 *
```

```
 */

#ifndef __IMAGEBOX_H__
#define __IMAGEBOX_H__

#include "button.h"


typedef Button ImageBox;

void imageBox_draw(ImageBox *ib);
int imageBox_event(ImageBox *ib);
ImageBox imageBox_new(char *filePath, int x, int y);
void imageBox_fullScreen(char *filePath);

#include <graphics.h>
#include <svga.h>
#include "image.h"
#include "keyboard.h"
#endif
```

5.1.15 ime.h

```
/**
 * @file ime.h
 * @author Hibanaw Hu (hibanaw@qq.com)
 * @brief
 * @date 2023-03-22
 *
 * @copyright Copyright (c) 2023
 *
 */

#ifndef __IME_H__
#define __IME_H__

#include "button.h"

enum imeStatus{
    IMEOFF,
    IMEPINYIN,
    IMEEMOJI
};

typedef struct
{
    enum imeStatus status;
    int pw;
    char buffer[8540];
    Button button;
}Ime;

Ime *ime();
void ime_init();
int ime_input(char *s, int x, int y);
int ime_en(char *s);
```

```
void ime_check();
void ime_next();
void ime_draw();

#include <bios.h>
#include <time.h>
#include "text.h"
#include "hzinput.h"
#endif
```

5.1.16 init.h

```
/**
 * @file init.h
 * @author Hibanaw Hu (hibanaw@qq.com)
 * @brief
 * @date 2023-02-12
 *
 * @copyright Copyright (c) 2023
 *
 */

#ifndef __INIT_H__
#define __INIT_H__


void init();
void destruct();

#include "svga.h"
#include "mouse.h"
#include "ime.h"

#endif
```

5.1.17 keyboard.h

```
/**
 * @file keyboard.h
 * @author Hibanaw Hu (hibanaw@qq.com)
 * @brief
 * @date 2023-02-12
 *
 * @copyright Copyright (c) 2023
 *
 */
#ifndef __KEYBOARD_H__
#define __KEYBOARD_H__

enum spicalkey{
    KEYESCAPE = 0x011b,
    KEYBACKSPACE = 0xe08,
    KEYSPACE = 0x3920,
    KEYENTER = 0x1c0d,
```

```
    KEYUP = 0x4800,
    KEYLEFT = 0x4b00,
    KEYRIGHT = 0x4d00,
    KEYDOWN = 0x5000,
    KEYONE = 0x0231,
    KEYTWO = 0x0332,
    KEYTHREE = 0x0433,
    KEYFOUR = 0x0534,
    KEYHOME = 0x4700,
    KEYEND = 0x4f00
};

char keybord_bios2ascii(unsigned);
int keybord_isAlphabet(unsigned);
int keybord_isCharacter(unsigned);
int keybord_isESCAPE(unsigned);
int keybord_isBACKSPACE(unsigned);
int keybord_isControlOn();
void keybord_eat();


#include<bios.h>


#endif
```

5.1.18 main.h

```
/**
 * @file main.h
 * @author Hibanaw Hu (hibanaw@qq.com)
 * @brief
 * @date 2023-02-12
 *
 * @copyright Copyright (c) 2023
 *
 */

#ifndef __MAIN_H__
#define __MAIN_H__

#include "init.h"
#include "homepage.h"

#endif
```

5.1.19 meditor.h

```
/**
 * @file meditor.h
 * @author Hibanaw Hu (hibanaw@qq.com)
 * @brief
 * @date 2023-04-02
 *
```

```
 * @copyright Copyright (c) 2023
 *
 */

#ifndef __MEMOEDITOR_H__
#define __MEMOEDITOR_H__

#include "memo.h"
#include "button.h"
#include "scroll.h"
#include "textipt.h"
#include "auth.h"

#define EDITORHEIGHT 680
#define EDITORWIDTH 800

#define BLOCKMAX 30

typedef struct{
    void *widget[30];
    enum Memotype type[30];
    MemoBlock *memoBlock[30];
    int count;
} MemoEditorWidgetList;

typedef struct{
    char *uid;
    char *fileName;
    MemoBlock *beginMemoBlock;
    MemoBlock *focusedBlock;
    int focusedBlockCursorLocation;
    int posX;
    int posY;
    MemoEditorWidgetList list;
    Button drawButton;
    Button imageButton;
    Button checkboxButton;
    Button shareButton;
    Button settingsButton;
    Button saveButton;
    TextInput titleBar;
    ScrollBar scrollBar;
    int unSaved;
    enum AuthType authType;
} MemoEditor;

typedef struct{
    Button checkbox;
    Textbox textbox;
} MemoEditor_Checkbox;

MemoEditor memoEditor_new(char *fileName, char *uid);
void memoEditor_draw(MemoEditor *e);
int memoEditor_event(MemoEditor *me);
void memoEditor_updateList(MemoEditor *e);
void memoEditor_distruct(MemoEditor *me);
void memoEditor_save(MemoEditor *me);

void memoEditor_button_drawAddPicture(Button *b);
```

```
void memoEditor_button_drawAddCheckbox(Button *b);
void memoEditor_button_drawDrawpad(Button *b);
void memoEditor_button_drawSharebutton(Button *b);
void memoEditor_button_drawSavebutton(Button *b);
void memoEditor_button_drawEditbutton(Button *b);
#include "textbox.h"
#include "imagebox.h"
#include "mfile.h"
#include "app.h"
#include "drawpad.h"
#include "addimage.h"
#include "share.h"
#include "mset.h"
#endif
```

5.1.20 memo.h

```
/**
 * @file memo.h
 * @author wywgwt (2504133124@qq.com)
 * @brief
 * @date 2023-04-03
 *
 * @copyright Copyright (c) 2023
 *
 */
#ifndef _MEMO_H_
#define _MEMO_H_

#include <stdio.h>
#include <malloc.h>
#include <stdlib.h>
#include <string.h>

typedef enum
{
    PARAGRAPH,
    IMAGE,
    CHECKBOX
} Memotype;

typedef struct MemoBlock
{
    Memotype type;
    int checkBoxisChecked;
    char content[160];
    struct MemoBlock *next;
    char lastEditUser[20];
} MemoBlock; // 备忘录的信息块

typedef struct Memo
{
    char owner[20];
    MemoBlock *head;
    struct Memo *next;
    char title[30];
```

```
    char fileName[20];
    int level;
} Memo; // 备忘录的节点

Memo *memo();

void *memo_addBlock(MemoBlock *a);

MemoBlock *memo_insertBlock(MemoBlock *p, MemoBlock *a);

// 在第一个参数的位置之后插入新Node，新node的内容是第二个参数，返回插入的node的地
址。
MemoBlock *memo_deleteBlock(MemoBlock *p);

// 删除参数所在的节点，返回该节点的next。

MemoBlock *memo_newBlock(Memotype type, int checkBoxisChecked, char
*content);

MemoBlock *memo_preBlock(MemoBlock *p);

int memo_getBlockSum();

int memo_getBlockNum(MemoBlock *mb);
#endif
```

5.1.21 memos.h

```
#ifndef _MEMOS_H_
#define _MEMOS_H_
#include <stdio.h>
#include "memo.h"
typedef struct
{
    Memo *head;
    unsigned int count;
} Memos; // 所有备忘录，备忘录的链表结构，记录了头结点和总个数。
Memos *memos();
void *memos_add(Memo *a);
Memo *memos_insertMemo(Memo *p, Memo *a);
Memo *memos_deleteMemo(Memo *p);
Memo *memos_makeTopMemo(Memo *p);
void memos_reset();
Memos *memos_getList(char *uid);

Memo *memos_preMemo(Memo *p);
int memos_getSum();
int memos_getNum(Memo *mb);

#include "auth.h"
#include <dir.h>
#endif
```

5.1.22 mfile.h

```
/**
 * @file mfile.h
 * @author wywgwt (2504133124@qq.com)
 * @brief
 * @date 2023-04-06
 *
 * @copyright Copyright (c) 2023
 *
 */

#ifndef __FILE_H__
#define __FILE_H__
#include"memos.h"
#include"memo.h"

FILE **memofile_current();

void memofile_write(char* filePath, Memo *m);

void memofile_writeBlock(MemoBlock *p);

MemoBlock *memofile_readBlock();

Memo memofile_read(char *filePath);


#include<stdio.h>
#include<string.h>
#include "app.h"

#endif
```

5.1.23 mouse.h

```
/**
 * @file mouse.h
 * @author dengshuumin, Hibanaw Hu (hibanaw@qq.com)
 * @brief Header file of mouse under DOS with no global variables.
 * @date 2023-02-26
 *
 * @copyright Copyright (c) 2023
 *
 */

#ifndef __MOUSE_H__
#define __MOUSE_H__

#include <conio.h>
#include <graphics.h>
#include <dos.h>
#include <stdio.h>
#include <stdlib.h>
#include "svga.h"
#include "global.h"

/**
```

```
 * @brief Cursor styles.
 *
 */
enum CursorStyle{
    CURSORPOINTER = 0,
    CURSORSELECT,
    CURSORTEXT,
    CURSORCROSS
};

/**
 * @brief Mouse Click status.
 *
 */
enum ClickStatus{
    ClickUnclick,
    ClickLeft,
    ClickRight
};

/**
 * @brief Struct of mouse.
 *
 */
typedef struct{
    int posX, posY;
    enum CursorStyle style;
    void *buffer;
    enum ClickStatus click;
    bool visibility;

    int _flag;
}Mouse;

Mouse *mouse();
void mouse_init();
void mouse_update();
void mouse_show();
void mouse_hide();
int mouse_isClickedInBox(int, int, int, int);

void _mouse_draw(int, int);
void _mouse_read(int *,int *,int *);
void _mouse_saveBackground(int,int);
void _mouse_clrmous(int, int);
void _mouse_drawmous(int,int);

#endif
```

5.1.24 mset.h

```
/**
 * @file mset.h
 * @author Hibanaw Hu (hibanaw@qq.com), wywgwt (2504133124@qq.com)
 * @brief
 * @date 2023-04-21
```

```
 *
 * @copyright Copyright (c) 2023
 *
 */
#ifndef __MSET_H__
#define __MSET_H__

#include "button.h"
#include "meditor.h"

typedef struct{
    Button radioBox[4];
}MsetRadioBoxArea;
int mset_page(MemoEditor *me);
int mset_newRadioBoxArea(MsetRadioBoxArea *ra, int y, int defualt);
void mset_drawRadioBoxArea(MsetRadioBoxArea *sr);
int mset_radioBoxAreaEvent(MsetRadioBoxArea *sr);

#include <graphics.h>
#include "svga.h"
#include "mouse.h"
#include "keyboard.h"
#include "textipt.h"
#include "memo.h"
#include "meditor.h"
#endif
```

5.1.25 mshare.h

```
/**
 * @file share.h
 * @author Hibanaw Hu (hibanaw@qq.com)
 * @brief
 * @date 2023-04-20
 *
 * @copyright Copyright (c) 2023
 *
 */

#ifndef __MSHARE_H__
#define __MSHARE_H__

#include <time.h>
#include "button.h"
#include "textipt.h"

typedef char shareCode[10];

typedef struct
{
    shareCode code;
    char memoName[20];
    time_t expiryTime;
}Share;

typedef struct
```

```
{
    Button radioBox[5];
    TextInput textInput;
}ShareRadioBoxArea;

void share_page(char *memoName);
int share_add(char *memoName, char *shareCodeBuffer, time_t expireTime);
int share_determine(char *shareCode, char *uid);

share_newRadioBoxArea(ShareRadioBoxArea *ra, char *buffer);
void share_drawRadioBoxArea(ShareRadioBoxArea *sr);
int share_radioBoxAreaEvent(ShareRadioBoxArea *sr);

#include "svga.h"
#include "mouse.h"
#include "keyboard.h"
#include "digclock.h"
#include "auth.h"
#endif
```

5.1.26 router.h

```
/**
 * @file router.h
 * @author Hibanaw Hu (hibanaw@qq.com)
 * @brief
 * @date 2023-04-11
 *
 * @copyright Copyright (c) 2023
 *
 */

#ifndef __ROUTER_H__
#define __ROUTER_H__

#include "memos.h"
#include "meditor.h"
#include "button.h"

typedef struct{
    Memo *topMemo;
    Button expandButton;
    Button newMemoButton;
    Button userButton;
    Button setButton;
    Button memoButton[10];
    char memoName[20];
}Router;

enum {
    RouterExpand = 1,
    RouterChangeMemo,
    RouterUserPage,
    RouterSettingsPage
};
```

```
Router router_new();
void router_draw(Router *r);
int router_expand(Router *r);
int router_event(Router *r);
void router_button_drawMemoList(Button *b);
void router_distrcut();
void router_refresh(Router *r);
void router_button_drawExpandButton(Button *b);
void router_button_drawNewMemoButton(Button *b);
void router_button_drawUserButton(Button *b);
void router_button_drawSetButton(Button *b);
#include <bios.h>
#include "keyboard.h"
#include "app.h"
#include "digclock.h"
#include "scroll.h"
#include "textipt.h"
#include "mshare.h"
#endif
```

5.1.27 scroll.h

```
/**
 * @file scroll.h
 * @author Hibanaw Hu (hibanaw@qq.com)
 * @brief
 * @date 2023-04-11
 *
 * @copyright Copyright (c) 2023
 *
 */

#ifndef __SCROLL_H__
#define __SCROLL_H__


enum ScrollBarStatus{
    ScrollBarDefault,
    ScrollBarFocused,
    ScrollBarSelected
};

typedef struct
{
    int posX;
    int posY;
    int height;
    int width;
    int length;
    enum ScrollBarStatus status;
    int barPosY1;
    int barPosY2;
    int mouseLastPosY;
    int sumItem;
    int inScreenItem;
    int ithItem;
```

```
    int lastDrawBarPosY1;
    int lastDrawBarPosY2;
    int bgColor;
}ScrollBar;

ScrollBar scrollBar_new(int x, int y, int h);
int scrollBar_event(ScrollBar *sb);
void scrollBar_draw(ScrollBar *sb);


#include <graphics.h>
#include "stdlib.h"
#include "global.h"
#include "mouse.h"
#include "svga.h"
#endif
```

5.1.28 svga.h

```
/**
 * @file svga.h
 * @author Hibanaw Hu (hibanaw@qq.com)
 * @brief
 * @date 2023-02-12
 *
 * @copyright Copyright (c) 2023
 *
 */

#ifndef __SVGA_H__
#define __SVGA_H__

#include<graphics.h>
#include<dos.h>
#include<stdio.h>


#define SVGA320x200x256 0    /* 320x200x256 Standard VGA */
#define SVGA640x400x256 1    /* 640x400x256 Svga/VESA */
#define SVGA640x480x256 2    /* 640x480x256 Svga/VESA */
#define SVGA800x600x256 3    /* 800x600x256 Svga/VESA */
#define SVGA1024x768x256 4    /* 1024x768x256 Svga/VESA */
#define SVGA640x350x256 5    /* 640x350x256 Svga/VESA */
#define SVGA1280x1024x256 6 /* 1280x1024x256 Svga/VESA */

typedef unsigned char DacPalette256[256][3];

enum HexColor
{
    hex000000,
    hex800000,
    hex008000,
    hex808000,
    hex000080,
    hex800080,
    hex008080,
```

```
    hexc0c0c0,
    hexc0dcc0,
    hexa6caf0,
    hex2a3faa,
    hex2a3fff,
    hex2a5f00,
    hex2a5f55,
    hex2a5faa,
    hex2a5fff,
    hex2a7f00,
    hex2a7f55,
    hex2a7faa,
    hex2a7fff,
    hex2a9f00,
    hex2a9f55,
    hex2a9faa,
    hex2a9fff,
    hex2abf00,
    hex2abf55,
    hex2abfaa,
    hex2abfff,
    hex2adf00,
    hex2adf55,
    hex2adfaa,
    hex2adfff,
    hex2aff00,
    hex2aff55,
    hex2affaa,
    hex2affff,
    hex550000,
    hex550055,
    hex5500aa,
    hex5500ff,
    hex551f00,
    hex551f55,
    hex551faa,
    hex551fff,
    hex553f00,
    hex553f55,
    hex553faa,
    hex553fff,
    hex555f00,
    hex555f55,
    hex555faa,
    hex555fff,
    hex557f00,
    hex557f55,
    hex557faa,
    hex557fff,
    hex559f00,
    hex559f55,
    hex559faa,
    hex559fff,
    hex55bf00,
    hex55bf55,
    hex55bfaa,
    hex55bfff,
    hex55df00,
    hex55df55,
```

```
    hex55dfaa,
    hex55dfff,
    hex55ff00,
    hex55ff55,
    hex55ffaa,
    hex55ffff,
    hex7f0000,
    hex7f0055,
    hex7f00aa,
    hex7f00ff,
    hex7f1f00,
    hex7f1f55,
    hex7f1faa,
    hex7f1fff,
    hex7f3f00,
    hex7f3f55,
    hex7f3faa,
    hex7f3fff,
    hex7f5f00,
    hex7f5f55,
    hex7f5faa,
    hex7f5fff,
    hex7f7f00,
    hex7f7f55,
    hex7f7faa,
    hex7f7fff,
    hex7f9f00,
    hex7f9f55,
    hex7f9faa,
    hex7f9fff,
    hex7fbf00,
    hex7fbf55,
    hex7fbfaa,
    hex7fbfff,
    hex7fdf00,
    hex7fdf55,
    hex7fdfaa,
    hex7fdfff,
    hex7fff00,
    hex7fff55,
    hex7fffaa,
    hex7fffff,
    hexaa0000,
    hexaa0055,
    hexaa00aa,
    hexaa00ff,
    hexaa1f00,
    hexaa1f55,
    hexaa1faa,
    hexaa1fff,
    hexaa3f00,
    hexaa3f55,
    hexaa3faa,
    hexaa3fff,
    hexaa5f00,
    hexaa5f55,
    hexaa5faa,
    hexaa5fff,
    hexaa7f00,
```

```
    hexaa7f55,
    hexaa7faa,
    hexaa7fff,
    hexaa9f00,
    hexaa9f55,
    hexaa9faa,
    hexaa9fff,
    hexaabf00,
    hexaabf55,
    hexaabfaa,
    hexaabfff,
    hexaadf00,
    hexaadf55,
    hexaadfaa,
    hexaadfff,
    hexaaff00,
    hexaaff55,
    hexaaffaa,
    hexaaffff,
    hexd40000,
    hexd40055,
    hexd400aa,
    hexd400ff,
    hexd41f00,
    hexd41f55,
    hexd41faa,
    hexd41fff,
    hexd43f00,
    hexd43f55,
    hexd43faa,
    hexd43fff,
    hexd45f00,
    hexd45f55,
    hexd45faa,
    hexd45fff,
    hexd47f00,
    hexd47f55,
    hexd47faa,
    hexd47fff,
    hexd49f00,
    hexd49f55,
    hexd49faa,
    hexd49fff,
    hexd4bf00,
    hexd4bf55,
    hexd4bfaa,
    hexd4bfff,
    hexd4df00,
    hexd4df55,
    hexd4dfaa,
    hexd4dfff,
    hexd4ff00,
    hexd4ff55,
    hexd4ffaa,
    hexd4ffff,
    hexff0055,
    hexff00aa,
    hexff1f00,
    hexff1f55,
```

```
    hexff1faa,
    hexff1fff,
    hexff3f00,
    hexff3f55,
    hexff3faa,
    hexff3fff,
    hexff5f00,
    hexff5f55,
    hexff5faa,
    hexff5fff,
    hexff7f00,
    hexff7f55,
    hexff7faa,
    hexff7fff,
    hexff9f00,
    hexff9f55,
    hexff9faa,
    hexff9fff,
    hexffbf00,
    hexffbf55,
    hexffbfaa,
    hexffbfff,
    hexffdf00,
    hexffdf55,
    hexffdfaa,
    hexffdfff,
    hexffff55,
    hexffffaa,
    hexccccff,
    hexffccff,
    hex33ffff,
    hex66ffff,
    hex99ffff,
    hexccffff,
    hex007f00,
    hex007f55,
    hex007faa,
    hex007fff,
    hex009f00,
    hex009f55,
    hex009faa,
    hex009fff,
    hex00bf00,
    hex00bf55,
    hex00bfaa,
    hex00bfff,
    hex00df00,
    hex00df55,
    hex00dfaa,
    hex00dfff,
    hex00ff55,
    hex00ffaa,
    hex2a0000,
    hex2a0055,
    hex2a00aa,
    hex2a00ff,
    hex2a1f00,
    hex2a1f55,
    hex2a1faa,
```

```
    hex2a1fff,
    hex2a3f00,
    hex2a3f55,
    hexfffbf0,
    hexa0a0a4,
    hex808080,
    hexff0000,
    hex00ff00,
    hexffff00,
    hex0000ff,
    hexff00ff,
    hex00ffff,
    hexffffff
};


enum MyColors{
    _WHITE = hexffffff,
    _BLACK = hex000000,
    _DARKGRAY = hex2a1f00,
    _GRAY = hex808080,
    _BLUE = hex0000ff,
    _RED = hexff0000,
    _CYAN = hex2adfff,
    _LIGHTGRAY = hexc0c0c0,
    _LIGHTYELLOW = hexffdf00,
};

int huge detectSVGA256(void);
int initsvga256(void);
int closesvga256(void);

void setvgapalette256(DacPalette256 *);
void useRGB256Colors();

#endif
```

5.1.29 text.h

```
/**
 * @file text.h
 * @author Hibanaw Hu (hibanaw@qq.com)
 * @brief
 * @date 2023-03-08
 *
 * @copyright Copyright (c) 2023
 *
 */

#ifndef __TEXT_H__
#define __TEXT_H__


typedef struct{
    int fontSize;
    int fontColor;
```

```
    int spacing;
    int rowSpacing;
}FontFamily;

typedef struct
{
    char *content;
    int posX;
    int posY;
    int width;
    int hight;
    FontFamily font;
}Text;


void text_display(Text);
int text_getLength(char *s);
Text text_newDefault(char *s, int x1, int y1, int x2, int y2);
char *text_getNthChar(char *s, int n);
int text_getHeight(Text t);
Text text_newSmall(char *s, int x, int y);

#include <graphics.h>
#include "svga.h"
#include "hz.h"
#include "debug.h"
#endif
```

5.1.30 textbox.h

```
/**
 * @file textbox.h
 * @author Hibanaw Hu (hibanaw@qq.com)
 * @brief
 * @date 2023-03-08
 *
 * @copyright Copyright (c) 2023
 *
 */
#ifndef __TEXTBOX_H__
#define __TEXTBOX_H__

#include "global.h"
#include "text.h"

enum TextboxStatus{
    TextboxDefault,
    TextboxFocused,
    TextboxSelected
};

enum TextboxMouseStatus{
    TextboxMouseDefault,
    TextboxMouseFocused,
    TextboxMouseClicked
};
```

```
enum TextboxType{
    TextboxText,
    TextboxPassword
};

typedef struct{
    char *defaultContent;
    int posX1;
    int posY1;
    int posX2;
    int posY2;
    char *content;
    int maxLength;
    FontFamily font;
    enum TextboxStatus status;
    enum TextboxMouseStatus mstatus;
    int cursorLocation;
    int cursorStatus;
    clock_t cursorLastBlink;
    enum TextboxType type;
    char hint;
    char bgColor;
}Textbox;

void textbox_draw(Textbox *tb);
int textbox_event(Textbox *tb);
void textbox_determinState(Textbox *tb);
Textbox textbox_newDefault(char *ds, int x1, int y1, int x2, int y2,
char *buffer);
Text textbox_convert2text(Textbox tb);
int textbox_getCursorPositionX(Textbox t);
int textbox_getCursorPositionY(Textbox t);

#include <string.h>
#include <time.h>
#include "ime.h"
#endif
```

5.1.31 textipt.h

```
/**
 * @file textipt.h
 * @author Hibanaw Hu (hibanaw@qq.com)
 * @brief
 * @date 2023-04-01
 *
 * @copyright Copyright (c) 2023
 *
 */

#ifndef __TEXTBOX__
#define __TEXTBOX__

#include "textbox.h"
```

```
typedef struct{
    int posX1;
    int posY1;
    int posX2;
    int posY2;
    Textbox textbox;
    int align;
    void (*draw)();
}TextInput;

void textinput_draw(TextInput *tb);
void textinput_drawDefault(TextInput *tb);
int textinput_event(TextInput *ti);
TextInput textinput_newDefault(char *ds, int x1, int y1, int x2, int y2,
char *buffer);
TextInput textinput_newTitle(char *ds, int x1, int y1, int x2, int y2,
char *buffer);

#endif
```

5.1.32 user.h

```
/**
 * @file user.h
 * @author wywgwt (2504133124@qq.com)
 * @brief
 * @date 2023-04-20
 *
 * @copyright Copyright (c) 2023
 *
 */
#ifndef __USER_H__
#define __USER_H__
#include<stdio.h>
#include<string.h>
typedef struct
{
  char account[20];
  char password[20];
}Userdata;

int user_login(char *p,char *q,int k);

// void user_Datainput(char *p,char *q);

int length_judge(char *p,char *q);

#endif
```

5.1.33 userpage.h

```
/**
 * @file userpage.h
 * @author wywgwt (2504133124@qq.com)
```

```
 * @brief
 * @date 2023-04-20
 *
 * @copyright Copyright (c) 2023
 *
 */

#ifndef __USERPAGE_H__
#define __USERPAGE_H__

#include <graphics.h>
#include "svga.h"
#include "keyboard.h"
#include "mouse.h"
#include "global.h"
#include "textbox.h"
#include "image.h"
#include "anim.h"
#include "meditor.h"
#include"text.h"
#include"textipt.h"
#include"textbox.h"
#include"app.h"
#include"malloc.h"
#include"button.h"
#include<string.h>
#include"user.h"
#include"button.h"
int userpage();

int userpage_login(int j);

int userpage_deleteuser(int j,int k);
void userpage_button_draw1(Button *b);
void userpage_button_draw2(Button *b);
void userpage_button_draw3(Button *b);
//userpage_addUser()

//userpage_deleteUser()
#endif
```

5.2 源文件

5.2.1 addimage.c

```
/**
 * @file addimage.c
 * @author Hibanaw Hu (hibanaw@qq.com)
 * @brief
 * @date 2023-04-19
 *
 * @copyright Copyright (c) 2023
 *
 */
```

```
#include "addimage.h"

int addImage(char *path){
    while(1){
        int signal = 0;
        Button b = button_new(680, 150, 770, 190, "插入",
button_drawWINUIAccent);
        Button bb = button_new(570, 150, 660, 190, "取消",
button_drawWINUI);
        TextInput t = textinput_newDefault(
                    "请输入图片路径",
                    320, 100,
                    780, 140,
                    path);

        Text terror = text_newSmall("路径不合法",
            690, 110
        );
        terror.font.fontColor = _RED;
        t.textbox.font.fontSize = 16;
        t.textbox.maxLength = 20;
        mouse_hide();
        setfillstyle(1, hexfffbf0);
        bar(300, 72, 800, 200);
        button_draw(&b);
        button_draw(&bb);
        textinput_drawDefault(&t);
        mouse_show();
        while(!signal){
            int k = bioskey(1);
            int bs = 0, bbs = 0, ts = 0;
            keybord_eat();
            ime_check();
            digitalClock_getTime();
            mouse_update();
            if(keybord_isESCAPE(k)){
                bioskey(0);
                signal = -1;
            }
            ts = textinput_event(&t);
            bs = button_event(&b);
            bbs = button_event(&bb);
            if(bbs){
                signal = -1;
            }
            if(bs || ts==1){
                if(image_illegal(path)){
                    text_display(terror);
                    continue;
                }
                else
                signal = 1;
            }
        }
        switch (signal)
        {
        case -1:
            return 1;
```

```
            break;

        case 1:

                return 0;
            break;
        }
    }

}
```

5.2.2 anim.c

```
/**
 * @file anim.c
 * @author wywgwt (2504133124@qq.com)
 * @brief
 * @date 2023-04-03
 *
 * @copyright Copyright (c) 2023
 *
 */
#include "anim.h"

void animation_homepage1()
{
    float k = 0.5;
    clock_t t0 = clock();
    clock_t t = t0;
    while ((t - t0) < k * CLK_TCK)
    {
        t = clock();
        if ((t - t0) < k * 1 * CLK_TCK / 6)
        {
            setfillstyle(1, hex000000);
            bar(0, 0, 1024, 768);
        }
        if (((t - t0) > k * 1 * CLK_TCK / 6) && ((t - t0) < k * 2 *
CLK_TCK / 6))
        {
            setfillstyle(1, hex2a0000);
            bar(0, 0, 1024, 768);
        }
        if (((t - t0) > k * 2 * CLK_TCK / 6) && ((t - t0) < k * 3 *
CLK_TCK / 6))
        {
            setfillstyle(1, hex550000);
            bar(0, 0, 1024, 768);
        }
        if (((t - t0) > k * 3 * CLK_TCK / 6) && ((t - t0) < k * 4 *
CLK_TCK / 6))
        {
            setfillstyle(1, hex551f00);
            bar(0, 0, 1024, 768);
        }
        if (((t - t0) > k * 4 * CLK_TCK / 6) && ((t - t0) < k * 5 *
```

```
CLK_TCK / 6))
        {
            setfillstyle(1, hex7f1f00);
            bar(0, 0, 1024, 768);
        }
        if (((t - t0) > k * 5 * CLK_TCK / 6) && ((t - t0) < k * 6 *
CLK_TCK / 6))
        {
            setfillstyle(1, hexaa3f00);
            bar(0, 0, 1024, 768);
        }
    }
}
void animation_homepage2()
{
    clock_t t0 = clock();
    clock_t t = t0;
    float k = 1;
    while ((t - t0) < (k * CLK_TCK))
    {
        t = clock();
        if ((t - t0) < k * 1 * CLK_TCK / 4)
        {
            setfillstyle(1, hexd4bfaa);
            bar(709 * pow(((t - t0) / (k * CLK_TCK) - 1), 4) + 315, 0,
1024, 768);
        }
        if (((t - t0) > k * 1 * CLK_TCK / 4) && ((t - t0) < k * 1 *
CLK_TCK))
        {
            setfillstyle(1, hexd4bfaa);
            bar(315 + (1 - ((float)(t - t0) / (k * CLK_TCK))) * 299.109,
0, 1024, 768);
        }
    }
}
void animation_login()
{
    clock_t t0 = clock();
    clock_t t = t0;
    float k = 0.4;
    float l = 2;
    clock_t t1;
    while ((t - t0) < (k * CLK_TCK))
    {
        t = clock();
        if ((t - t0) < k * 1 * CLK_TCK / 6)
        {
            setfillstyle(1, hexaa3f00);
            bar(0, 0, 315, 768);
        }
        if (((t - t0) > k * 1 * CLK_TCK / 6) && ((t - t0) < k * 2 *
CLK_TCK / 6))
        {
            setfillstyle(1, hex7f1f00);
            bar(0, 0, 315, 768);
        }
        if (((t - t0) > k * 2 * CLK_TCK / 6) && ((t - t0) < k * 3 *
CLK_TCK / 6))
```

```
        {
            setfillstyle(1, hex551f00);
            bar(0, 0, 315, 768);
        }
        if (((t - t0) > k * 3 * CLK_TCK / 6) && ((t - t0) < k * 4 *
CLK_TCK / 6))
        {
            setfillstyle(1, hex550000);
            bar(0, 0, 315, 768);
        }
        if (((t - t0) > k * 4 * CLK_TCK / 6) && ((t - t0) < k * 5 *
CLK_TCK / 6))
        {
            setfillstyle(1, hex2a0000);
            bar(0, 0, 315, 768);
        }
        if (((t - t0) > k * 5 * CLK_TCK / 6) && ((t - t0) < k * 6 *
CLK_TCK / 6))
        {
            setfillstyle(1, hex000000);
            bar(0, 0, 315, 768);
        }
        if ((t - t0) < k * 1 * CLK_TCK / 12)
        {
            setfillstyle(1, hexd4bfaa);
            bar(315, 0, 1024, 768);
        }
        if (((t - t0) > k * 1 * CLK_TCK / 12) && ((t - t0) < k * 2 *
CLK_TCK / 12))
        {
            setfillstyle(1, hexaabfaa);
            bar(315, 0, 1024, 768);
        }
        if (((t - t0) > k * 2 * CLK_TCK / 12) && ((t - t0) < k * 3 *
CLK_TCK / 12))
        {
            setfillstyle(1, hexaa9faa);
            bar(315, 0, 1024, 768);
        }
        if (((t - t0) > k * 3 * CLK_TCK / 12) && ((t - t0) < k * 4 *
CLK_TCK / 12))
        {
            setfillstyle(1, hexa0a0a4);
            bar(315, 0, 1024, 768);
        }
        if (((t - t0) > k * 4 * CLK_TCK / 12) && ((t - t0) < k * 5 *
CLK_TCK / 12))
        {
            setfillstyle(1, hexa0a0a4);
            bar(315, 0, 1024, 768);
        }
        if (((t - t0) > k * 5 * CLK_TCK / 12) && ((t - t0) < k * 6 *
CLK_TCK / 12))
        {
            setfillstyle(1, hex808080);
            bar(315, 0, 1024, 768);
        }
        if (((t - t0) > k * 6 * CLK_TCK / 12) && ((t - t0) < k * 7 *
CLK_TCK / 12))
```

```
        {
            setfillstyle(1, hex7f7f55);
            bar(315, 0, 1024, 768);
        }
        if (((t - t0) > k * 7 * CLK_TCK / 12) && ((t - t0) < k * 8 *
CLK_TCK / 12))
        {
            setfillstyle(1, hex7f5f55);
            bar(315, 0, 1024, 768);
        }
        if (((t - t0) > k * 8 * CLK_TCK / 12) && ((t - t0) < k * 9 *
CLK_TCK / 12))
        {
            setfillstyle(1, hex555f55);
            bar(315, 0, 1024, 768);
        }
        if (((t - t0) > k * 9 * CLK_TCK / 12) && ((t - t0) < k * 10 *
CLK_TCK / 12))
        {
            setfillstyle(1, hex553f55);
            bar(315, 0, 1024, 768);
        }
        if (((t - t0) > k * 10 * CLK_TCK / 12) && ((t - t0) < k * 11 *
CLK_TCK / 12))
        {
            setfillstyle(1, hex2a1f55);
            bar(315, 0, 1024, 768);
        }
        if (((t - t0) > k * 11 * CLK_TCK / 12) && ((t - t0) < k * 12 *
CLK_TCK / 12))
        {
            setfillstyle(1, hex000000);
            bar(315, 0, 1024, 768);
        }
    }
    t0 = clock();
    t = t0;
    setfillstyle(1, _BLACK);
    bar(0, 0, 1024, 768);
    while (((t - t0) < (3 + 1 / l) * CLK_TCK))
    {
        t = clock();
        if ((t - t0) < 3 * CLK_TCK)
        {
            setfillstyle(1, hex000000);
            bar(400, 400, 600, 768);
            setcolor(_WHITE);
            circle(512, 300, 60);
            circle(512, 282, 15);
            arc(512, 352.5, 28, 150, 55.5);
            if ((t - t0) < 1 * CLK_TCK)
            {
                arc(512, 450, 2 * (float)((t - t0) * (t - t0)), 2 *
(float)((t - t0) * (t - t0)) + 270, 30);
            }
            if (((t - t0) > 1 * CLK_TCK) && ((t - t0) < 2 * CLK_TCK))
            {
                arc(512, 450, 2 * (float)(2 * (t - t0)) + 16 * 18, 2 *
(float)(2 * (t - t0)) + 16 * 18 + 270, 30);
```

```
            }
            if (((t - t0) > 2 * CLK_TCK) && ((t - t0) < 3 * CLK_TCK))
            {
                arc(512, 450, 2 * (float)(108 * (t - t0) - (t - t0) * (t
- t0) - 62 * 36), 2 * (float)(108 * (t - t0) - (t - t0) * (t - t0) - 62
* 36) + 270, 30);
            }
            delay(100);
        }

        if (((t - t0) > 3 * CLK_TCK) && ((t - t0) < (3 + 1 / l) *
CLK_TCK))
        {
        setfillstyle(1, hex000000);
        bar(0, 0, 1024, 768);
        setcolor(_WHITE);
        circle(478-l*(512-37.5)*((t-t0-3*CLK_TCK)/
CLK_TCK),28+315+l*(650-320)*((t-t0-3*CLK_TCK)/CLK_TCK),60-(60-20)*l*((t-
t0-3*CLK_TCK)/CLK_TCK));
        circle(478-l*(512-37.5)*((t-t0-3*CLK_TCK)/
CLK_TCK),28+282+l*(644-282)*((t-t0-3*CLK_TCK)/CLK_TCK),15-(15-5)*l*((t-
t0-3*CLK_TCK)/CLK_TCK));
        arc(478-l*(512-37.5)*((t-t0-3*CLK_TCK)/
CLK_TCK),28+352.5+(667.5-352.5)*l*((t-t0-3*CLK_TCK)/
CLK_TCK),28,150,55.5-(55.5-18.5)*l*((t-t0-3*CLK_TCK)/CLK_TCK));
        delay(10);
        }
    }
}
```

5.2.3 app.c

```
/**
 * @file app.c
 * @author Hibanaw Hu (hibanaw@qq.com)
 * @brief
 * @date 2023-02-27
 *
 * @copyright Copyright (c) 2023
 *
 */
#include "app.h"

AppData *appData(){
    static AppData ad;
    return &ad;
}

void app(){
    Router r = router_new();
    MemoEditor me;
    appData()->displayLastEditUser = 0;
    appData()->currentUser=appData()->uid[0];
    me = memoEditor_new(r.memoName, appData()->currentUser);
    appData()->userCount = 1;
    animation_login();
```

```
    while(1){
        int signal = 0;
        Button eb = button_newExitButton();
        mouse_hide();
        debug(LOG, "Main app starts.");
        setfillstyle(1, _WHITE);
        bar(0, 0, MAXWIDTH, MAXHEIGHT);
        setfillstyle(1, hexfffbf0);
        bar(0, 0, MAXWIDTH, 40);
        image_render("res/img/logotxt.bin", 85, 5);
        setfillstyle(1, hexfffbf0);
        bar(0, 0, 75, MAXHEIGHT);
        setcolor(hexd4dfff);
        setlinestyle(0, 1, 2);
        line(73, 0, 73, MAXHEIGHT);
        ime_draw();
        router_draw(&r);
        button_draw(&eb);
        memoEditor_draw(&me);
        mouse_show();
        digitalClock_getTime();
        while(!signal){
            int k = bioskey(1);
            int mes = 0;
            Mouse *m = mouse();
            int rs;
            int ebs;
            keybord_eat();
            mouse_update();
            ime_check();
            digitalClock_getTime();
            mes = memoEditor_event(&me);
            rs = router_event(&r);
            ebs = button_event(&eb);
            if(keybord_isESCAPE(k)){
                bioskey(0);
                signal = AppExit;
            }
            if(ebs){
                signal = AppExit;
            }
            switch (rs)
            {
            case RouterExpand:
                signal = AppRouterExpand;
                break;
            case RouterChangeMemo:{
                int t = exitsave(&me);
                if(t == 1){
                    memoEditor_save(&me);
                }
                if(t==0){
                    break;
                }
                memoEditor_distruct(&me);
                me = memoEditor_new(r.memoName, appData()->currentUser);
                signal = AppRedraw;
            }
                break;
```

```
            case RouterUserPage:
                signal = AppUserPage;
                break;
            case RouterSettingsPage:
                appData()→displayLastEditUser = !appData()-
>displayLastEditUser;
                signal = AppRedraw;
            }
            if(mes == 1){
                signal = AppRedraw;
            }
        }
        switch(signal){
            case AppExit:{
                int t = exitsave(&me);
                if(t == 1){
                    memoEditor_save(&me);
                }
                if(t==0){
                    break;
                }
                memoEditor_distruct(&me);
                router_distrcut();
            }
                return 0;
                break;
            case AppRouterExpand:
                signal = router_expand(&r);
                if(signal == RouterChangeMemo){
                    int t = exitsave(&me);
                    if(t == 1){
                        memoEditor_save(&me);
                    }
                    if(t==0){
                        break;
                    }
                    memoEditor_distruct(&me);
                    me = memoEditor_new(r.memoName, appData()-
>currentUser);
                }
                if(signal == RouterUserPage){
                    if(userpage()){
                        memoEditor_distruct(&me);
                        r = router_new();
                        me = memoEditor_new(r.memoName, appData()-
>currentUser);
                    }
                }
                if(signal == RouterSettingsPage){
                    appData()→displayLastEditUser = !appData()-
>displayLastEditUser;
                }
                break;
            case AppRedraw:
                break;
            case AppUserPage:
                if(userpage()){
                    memoEditor_distruct(&me);
                    r = router_new();
```

```
                    me = memoEditor_new(r.memoName, appData()->currentUser);
                }
                break;
        }
    }
}
```

5.2.4 auth.c

```
/**
 * @file auth.c
 * @author Hibanaw Hu (hibanaw@qq.com)
 * @brief
 * @date 2023-04-19
 *
 * @copyright Copyright (c) 2023
 *
 */

#include "auth.h"

/**
 * @brief Determine auth.
 *
 * @param filename
 * @param uid
 * @return int 1 for can be open, 0 for cannot be open.
 */
int auth_check(char *filename, char *uid){
    char filePath[30];
    FILE *fp;
    Auth a;
    memset(&a, 0, sizeof(a));
    sprintf(filePath, "data\\auth\\%s.AUT", filename);
    fp = fopen(filePath, "rb");
    if (fp == NULL) {
        return -1;
    }
    else {
        fread(&a, sizeof(a), 1, fp);
        if(a.type == AUTHPUBLIC){
            fclose(fp);
            return 1;
        }
        if(a.type == AUTHPRIVATE){
            if(!strcmp(a.owner, uid)){
                fclose(fp);
                return 1;
            }
            else{
                fclose(fp);
                return 0;
            }
        }
        if(a.type == AUTHWHITELIST){
```

```
            if(!strcmp(a.owner, uid)){
                fclose(fp);
                return 1;
            }
            while(!feof(fp)){
                AuthUser user;
                memset(user, 0, sizeof(AuthUser));
                fread(user, 1, sizeof(user), fp);
                if(!strcmp(user, uid)){
                    fclose(fp);
                    return 1;
                }
            }
        }
        fclose(fp);
        return 0;
    }
}

int auth_set(char *filename, char * uid, enum AuthType type){
    char filePath[30];
    FILE *fp;
    Auth a;
    sprintf(filePath, "data\\auth\\%s.AUT", filename);
    strcpy(a.owner, uid);
    a.type = type;
    fp = fopen(filePath, "wb");
    fwrite(&a, sizeof(a), 1, fp);
    fclose(fp);
    return 0;
}

int auth_addWhiteList(char *filename, char *uid){
    char filePath[30];
    FILE *fp;
    Auth a;
    sprintf(filePath, "data\\auth\\%s.AUT", filename);
    fp = fopen(filePath, "rb+");
    if(fp == NULL){
        return -1;
    }
    fread(&a, sizeof(Auth), 1, fp);
    switch(a.type){
        case AUTHPRIVATE:
            return -1;
        case AUTHPUBLIC:
            return -1;
        case AUTHWHITELIST:{
                AuthUser usr;
                while(!feof(fp)){
                    memset(usr, 0, sizeof(AuthUser));
                    fread(&usr, sizeof(AuthUser), 1, fp);
                    if(!strcmp(usr, uid)){
                        fclose(fp);
                        return 1;
                    }
                }
                strcpy(usr, uid);
                fwrite(usr, 1, sizeof(AuthUser), fp);
```

```
            }
            break;
    }
    fclose(fp);
    return 0;
}

Auth auth_get(char *filename){
    char filePath[30];
    FILE *fp;
    Auth a;
    memset(&a, 0, sizeof(a));
    sprintf(filePath, "data\\auth\\%s.AUT", filename);
    fp = fopen(filePath, "rb");
    if (fp == NULL) {
        return a;
    }
    fread(&a, sizeof(a), 1, fp);
    fclose(fp);
    return a;
}
```

5.2.5 button.c

```
/**
 * @file button.c
 * @author Hibanaw Hu (hibanaw@qq.com), wywgwt (2504133124@qq.com)
 * @brief
 * @date 2023-02-28
 *
 * @copyright Copyright (c) 2023
 *
 */

#include "button.h"


void button_draw(Button *b){
    b->draw(b);
}

void button_drawDefault(Button *b){
    int x1 = b->posX1, y1 = b->posY1,
        x2 = b->posX2, y2 = b->posY2;
    setfillstyle(1,_LIGHTGRAY);
    bar(x1, y1, x2, y2);
    switch (b->status)
    {
    case ButtonDefault:
        setfillstyle(1, hexfffbf0);
        bar(x1, y1, x2, y2-2);
        setfillstyle(1, _WHITE);
        bar(x1+1, y1+2, x2-1, y2-2);
        break;
    case ButtonFocused:
        setfillstyle(1, hexffdfaa);
```

```
        bar(x1, y1, x2, y2-2);
        setfillstyle(1, hexfffbf0);
        bar(x1+1, y1+2, x2-1, y2-2);
        break;
    case ButtonSelected:
        setfillstyle(1, hexffdfaa);
        bar(x1, y1, x2, y2-2);
        setfillstyle(1, hexfffbf0);
        bar(x1+1, y1+2, x2-1, y2-2);
        break;
    default:
        break;
    }
}

void button_drawWINUIAccent(Button *b){
    int c0, c1, ct;
    int x1 = b->posX1, y1 = b->posY1,
        x2 = b->posX2, y2 = b->posY2;
    Text t;
    t.content = b->content;
    t.posX = b->posX1+(x2-x1-text_getLength(b->content)*(24+2))/2;
    t.posY = y1 + (y2-y1-24)/2;
    t.width = 0,
    t.hight = 0,
    t.font.fontSize = 24;
    t.font.fontColor = _BLACK,
    t.font.spacing = 2;
    t.font.rowSpacing = 0;
    switch (b->status)
    {
        case ButtonDefault:
            c1 = hexff7f00;
            c0 = hex808080;
            ct = _WHITE;
            break;
        case ButtonFocused:
            c1 = hexff9f00;
            c0 = hex808080;
            ct = _WHITE;
            break;
        case ButtonSelected:
            c0 = hexc0c0c0;
            c1 = hexfffbf0;
            ct = hexfffbf0;
            break;
    }
    setfillstyle(1, c1);
    setcolor(c1);
    bar(x1+5, y1, x2-5, y2);
    bar(x1, y1+5, x2, y2-5);
    pieslice(x1 + 5, y1 + 5, 90, 180, 5);
    pieslice(x1 + 5, y2 - 5, 180, 270, 5);
    pieslice(x2 - 5, y1 + 5, 0, 90, 5);
    pieslice(x2 - 5, y2 - 5, 270, 360, 5);
    setcolor(c0);
    setlinestyle(0, 1, 2);
    line(x1 + 5, y1, x2 - 5, y1);
    line(x1 + 5, y2, x2 - 5, y2);
```

```
    line(x1, y1 + 5, x1, y2 - 5);
    line(x2, y1 + 5, x2, y2 - 3);
    arc(x1 + 5, y1 + 5, 90, 180, 5);
    arc(x1 + 5, y2 - 5, 180, 270, 5);
    arc(x2 - 5, y1 + 5, 0, 90, 5);
    arc(x2 - 5, y2 - 5, 270, 360, 5);
    t.font.fontColor = ct;
    text_display(t);
}

void button_drawWINUI(Button *b){
    int c0, c1, ct;
    int x1 = b->posX1, y1 = b->posY1,
        x2 = b->posX2, y2 = b->posY2;
    Text t;
    t.content = b->content;
    t.posX = b->posX1+(x2-x1-text_getLength(b->content)*(24+2))/2;
    t.posY = y1 + (y2-y1-24)/2;
    t.width = 0,
    t.hight = 0,
    t.font.fontSize = 24;
    t.font.fontColor = _BLACK,
    t.font.spacing = 2;
    t.font.rowSpacing = 0;
    switch (b->status)
    {
        case ButtonDefault:
            c1 = _WHITE;
            c0 = hex808080;
            ct = _BLACK;
            break;
        case ButtonFocused:
            c1 = hexfffbf0;
            c0 = hex555f55;
            ct = _BLACK;
            break;
        case ButtonSelected:
            c0 = hex555f55;
            c1 = hexaa9faa;
            ct = hexfffbf0;
            break;
    }
    setfillstyle(1, c1);
    setcolor(c1);
    bar(x1+5, y1, x2-5, y2);
    bar(x1, y1+5, x2, y2-5);
    pieslice(x1 + 5, y1 + 5, 90, 180, 5);
    pieslice(x1 + 5, y2 - 5, 180, 270, 5);
    pieslice(x2 - 5, y1 + 5, 0, 90, 5);
    pieslice(x2 - 5, y2 - 5, 270, 360, 5);
    setcolor(c0);
    setlinestyle(0, 1, 2);
    line(x1 + 5, y1, x2 - 5, y1);
    line(x1 + 5, y2, x2 - 5, y2);
    line(x1, y1 + 5, x1, y2 - 5);
    line(x2, y1 + 5, x2, y2 - 3);
    arc(x1 + 5, y1 + 5, 90, 180, 5);
    arc(x1 + 5, y2 - 5, 180, 270, 5);
    arc(x2 - 5, y1 + 5, 0, 90, 5);
```

```
    arc(x2 - 5, y2 - 5, 270, 360, 5);
    t.font.fontColor = ct;
    text_display(t);
}

void button_drawWithText(Button *b){
    int x1 = b->posX1, y1 = b->posY1,
        x2 = b->posX2, y2 = b->posY2;
    Text t;
    t.content = b->content;
    t.posX = b->posX1+(x2-x1-text_getLength(b->content)*(24+2))/2;
    t.posY = y1 + (y2-y1-24)/2;
    t.width = 0,
    t.hight = 0,
    t.font.fontSize = 24;
    t.font.fontColor = _BLACK,
    t.font.spacing = 2;
    t.font.rowSpacing = 0;
    if(b->status == ButtonSelected){
        t.font.fontColor = hex555f55;
    }
    button_drawDefault(b);
    text_display(t);
}

int button_event(Button *b){
    int s;
    Mouse *m = mouse();
    int x1 = b -> posX1, y1 = b -> posY1,
        x2 = b -> posX2, y2 = b -> posY2;
    void (*thisdraw)() = button_draw;
    if(thisdraw == NULL){
        thisdraw = button_drawDefault;
    }
    s = mouse_isClickedInBox(x1, y1, x2, y2);
    if(b -> status == ButtonFocused && s == 1){
        debug(DEBUG, "Button clicked.");
        b -> status = ButtonSelected;
        m -> style = CURSORSELECT;
        mouse_hide();
        thisdraw(b);
        mouse_show();
        return 0;
    }
    if(b -> status == ButtonSelected && s == 2){
        debug(DEBUG, "Button released.");
        b -> status = ButtonDefault;
        m -> style = CURSORPOINTER;
        mouse_hide();
        thisdraw(b);
        mouse_show();
        return 1;
    }
    if(b -> status == ButtonDefault && s == 2){
        debug(DEBUG, "Button focused.");
        b -> status = ButtonFocused;
        m -> style = CURSORSELECT;
        mouse_hide();
        thisdraw(b);
```

```
        mouse_show();
        return 0;
    }
    if(b -> status != ButtonDefault && s == 0){
        debug(DEBUG, "Button disfocused.");
        b -> status = ButtonDefault;
        m -> style = CURSORPOINTER;
        mouse_hide();
        thisdraw(b);
        mouse_show();
        return 0;
    }
    return 0;
}

Button button_new(int x1, int y1, int x2, int y2, char *s, void (*f)()){
    Button b;
    memset(&b, 0, sizeof(b));
    b.posX1 = x1;
    b.posX2 = x2;
    b.posY1 = y1;
    b.posY2 = y2;
    b.content = s;
    b.draw = f;
    b.animationDrawedTick = 0;
    return b;
}


void button_checkboxAnimation(Button *cb){
    int x1, y1;
    x1 = cb->posX1;
    y1 = cb->posY1;
    if(cb->content){
        clock_t t = clock() - cb->animationStartTime;
        float k = t / (0.15 * CLK_TCK);
        if(k < 1){
            if(cb->animationDrawedTick == 0 || t - cb-
>animationDrawedTick > 0){
                mouse_hide();
                setcolor(_WHITE);
                setlinestyle(0, 1, 3);
                if(k > 0.1){
                    float p = MIN((k-0.1)/ 0.3, 1);
                    line(x1+5, y1 + 12, x1+5+p*3, y1 + 12+6*p);
                }
                if(k > 0.4)
                {
                    float p = (k-0.4)/ 0.7;
                    line(x1+8, y1 + 18, x1+8 + p*11, y1 + 18-p*12);
                }
                mouse_show();
                cb->animationDrawedTick = t;
            }
        }
        else if(cb->animationDrawedTick < 0.2 * CLK_TCK){
            mouse_hide();
            setlinestyle(0, 1, 3);
            setcolor(_WHITE);
```

```
            line(x1+5, y1 + 12, x1+8, y1 + 18);
            line(x1+8, y1 + 18, x1+19, y1 + 6);
            mouse_show();
            cb->animationDrawedTick = t;
        }
    }
}

int button_checkboxEvent(Button *cb){
    int k = button_event(cb);
    if(k){
        cb->content = !(int)cb->content;
    }
    if(k && cb->content == 1){
        cb->animationDrawedTick = 0;
        cb->animationTick = 0;
        cb->animationStartTime = clock();
    }
    button_checkboxAnimation(cb);
    return k;
}

void button_checkboxDraw(Button *cb){
    int x1, y1, x2, y2;
    int c0, c1, c2;
    x1 = cb->posX1;
    x2 = cb->posX2;
    y1 = cb->posY1;
    y2 = cb->posY2;
    switch ((int)cb->content)
    {
        case 0:
            switch (cb->status)
            {
                case ButtonDefault:
                    c0 = hex808080;
                    c1 = _WHITE;
                    break;
                case ButtonFocused:
                    c0 = hex808080;
                    c1 = hexfffbf0;
                    break;
                case ButtonSelected:
                    c0 = hexc0c0c0;
                    c1 = hexc0c0c0;
                    break;
            }
            setfillstyle(1, c1);
            setcolor(c1);
            bar(x1+5, y1, x2-5, y2);
            bar(x1, y1+5, x2, y2-5);
            pieslice(x1 + 5, y1 + 5, 90, 180, 5);
            pieslice(x1 + 5, y2 - 5, 180, 270, 5);
            pieslice(x2 - 5, y1 + 5, 0, 90, 5);
            pieslice(x2 - 5, y2 - 5, 270, 360, 5);
            setcolor(c0);
            setlinestyle(0, 1, 2);
            line(x1 + 5, y1, x2 - 5, y1);
            line(x1 + 5, y2, x2 - 5, y2);
```

```
            line(x1, y1 + 5, x1, y2 - 5);
            line(x2, y1 + 5, x2, y2 - 3);
            arc(x1 + 5, y1 + 5, 90, 180, 5);
            arc(x1 + 5, y2 - 5, 180, 270, 5);
            arc(x2 - 5, y1 + 5, 0, 90, 5);
            arc(x2 - 5, y2 - 5, 270, 360, 5);
            break;
        case 1:
            switch (cb->status)
            {
                case ButtonDefault:
                    c0 = hexff7f00;
                    c2 = _WHITE;
                    break;
                case ButtonFocused:
                    c0 = hexff9f00;
                    c2 = _WHITE;
                    break;
                case ButtonSelected:
                    c0 = hexd47f00;
                    c2 = hexd4bfaa;
                    break;
            }
            setfillstyle(1, c0);
            setcolor(c0);
            bar(x1+5, y1, x2-5, y2);
            bar(x1, y1+5, x2, y2-5);
            pieslice(x1 + 5, y1 + 5, 90, 180, 5);
            pieslice(x1 + 5, y2 - 5, 180, 270, 5);
            pieslice(x2 - 5, y1 + 5, 0, 90, 5);
            pieslice(x2 - 5, y2 - 5, 270, 360, 5);
            if(cb->animationDrawedTick > 0.15*CLK_TCK){
                setcolor(c2);
                setlinestyle(0, 1, 3);
                line(x1+5, y1 + 12, x1+8, y1 + 18);
                line(x1+8, y1 + 18, x1+19, y1 + 6);
            }
    }
}

void button_drawExitButton(Button *b){
    int px = b->posX1 + (b->posX2 - b->posX1)/2;
    int py = b->posY1 + (b->posY2 - b->posY1)/2;
    int r = (b->posX2 - b->posX1)/2;

    setfillstyle(1, hex7f0000);
    setcolor(hex7f0000);
    pieslice(px, py, 0, 360, r);
    switch (b->status)
    {
    case ButtonDefault:
        setfillstyle(1, hexff5f55);
        setcolor(hexff5f55);
        pieslice(px, py, 0, 360, r-2);
        break;

    case ButtonFocused:
        setfillstyle(1, hexd43f55);
        setcolor(hexd43f55);
```

```
        pieslice(px, py, 0, 360, r-2);
        break;
    case ButtonSelected:
        setfillstyle(1, hexaa1f00);
        setcolor(hexaa1f00);
        pieslice(px, py, 0, 360, r-2);
        break;
    }
}
Button button_newExitButton(){
    Button b = button_new(15, 5, 35, 25, "", button_drawExitButton);
    return b;
}
```

5.2.6 digclock.c

```
/**
 * @file digclock.c
 * @author Hibanaw Hu (hibanaw@qq.com)
 * @brief
 * @date 2023-04-03
 *
 * @copyright Copyright (c) 2023
 *
 */

#include "digclock.h"

void digitalClock_getTime(){
    static time_t t;
    time_t tn = time(NULL);
    char times[25];
    if(t != tn){
        t = tn;
        strftime(times, 20, "%H:%M:%S", localtime(&tn));
        mouse_hide();
        setfillstyle(1, _BLACK);
        bar(920, 0, 1024, 20);
        setfillstyle(1, _WHITE);
        bar(921, 1, 1022, 19);
        settextstyle(3, 0, 1);
        setcolor(_BLACK);
        settextjustify(LEFT_TEXT, 2);
        outtextxy(930, -3, times);
        mouse_show();
    }
}
```

5.2.7 drawpad.c

```
/**
 * @file drawpad.c
 * @author Hibanaw Hu (hibanaw@qq.com)
 * @brief
```

```
 * @date 2023-04-16
 *
 * @copyright Copyright (c) 2023
 *
 */

#include "drawpad.h"

int drawPad(char *saveFilePath){
    int px = 310, py = 80;
    int signal = 0;
    int color1 = _BLACK, color0 = _WHITE;
    int isDrawing = 0;
    int x0, y0;
    short w = 480, h = 360;
    Button saveButton = button_new(690, 450, 780, 490, "插入",
button_drawWINUIAccent);
    Button exitButton = button_new(580, 450, 670, 490, "取消",
button_drawWINUI);
    mouse_hide();
    setfillstyle(1, hexfffbf0);
    bar(300, 72, 800, 500);
    setfillstyle(1, hexd4dfff);
    bar(308, 78, 792, 442);
    setfillstyle(1, _WHITE);
    bar(310, 80, 790, 440);
    button_draw(&saveButton);
    button_draw(&exitButton);
    mouse_show();
    while(!signal){
        int k = bioskey(1);
        keybord_eat();
        mouse_update();
        if(keybord_isESCAPE(k)){
            bioskey(0);
            signal = -1;
        }
        if(mouse_isClickedInBox(310, 80, 790, 440) == 1){
            int x = mouse()->posX;
            int y = mouse()->posY;
            if(!isDrawing){
                isDrawing = 1;
                mouse()->visibility = 0;
                mouse_hide();
            }
            else{
                setcolor(color1);
                setlinestyle(0, 0, 3);
                line(x0, y0, x, y);
            }
            setfillstyle(1, color1);
            setcolor(color1);
            x0 = x;
            y0 = y;
        }
        else if(mouse_isClickedInBox(310, 80, 790, 440) == 3){
            int x = mouse()->posX;
            int y = mouse()->posY;
            if(!isDrawing){
```

```
                isDrawing = 1;
                mouse()->visibility = 0;
                mouse_hide();
            }
            else{
                setcolor(color0);
                setlinestyle(0, 0, 3);
                line(x0, y0, x, y);
            }
            setfillstyle(1, color0);
            setcolor(color0);
            x0 = x;
            y0 = y;
        }
        else if(isDrawing){
            isDrawing = 0;
            mouse()->visibility = 1;
            mouse_show();
        }
        if(button_event(&exitButton)){
            signal = -1;
        }
        if(button_event(&saveButton)){
            signal = 1;
        }
    }
    switch (signal)
    {
    case -1:
        return 1;
        break;

    case 1:
        {
            FILE *fp;
            int i, j;
            char p[20];
            mkdir(saveFilePath);
            sprintf(p, "/%06ld.bin", time(NULL)%100000);
            strcat(saveFilePath, p);
            fp = fopen(saveFilePath, "wb");
            fwrite(&w, sizeof(short), 1, fp);
            fwrite(&h, sizeof(short), 1, fp);
            for(i = 0; i < h; i++){
                for(j = 0; j < w; j++){
                    char c = getpixel(px+j, py+i);
                    fwrite(&c, 1, 1, fp);
                }
            }
            fclose(fp);
        }
        return 0;
        break;
    }
}
```

5.2.8 exitsave.c

```c
/**
 * @file exitsave.c
 * @author Hibanaw Hu (hibanaw@qq.com)
 * @brief
 * @date 2023-04-21
 *
 * @copyright Copyright (c) 2023
 *
 */

#include "exitsave.h"

int exitsave(MemoEditor *me){
    Button saveButton = button_new(690, 140, 780, 180, "保存",
button_drawWINUIAccent);
    Button exitButton = button_new(580, 140, 670, 180, "不保存",
button_drawWINUI);
    Button cancelButton = button_new(470, 140, 560, 180, "取消",
button_drawWINUI);
    Text title = text_newDefault("保存对此文件做的更改? ", 320, 90, 0, 0);

    if(!me->unSaved){
        return -1;
    }
    mouse_hide();
    setfillstyle(1, hexfffbf0);
    bar(300, 72, 800, 190);
    button_draw(&exitButton);
    button_draw(&saveButton);
    button_draw(&cancelButton);
    text_display(title);

    mouse_show();
    while(1){
        mouse_update();
        keybord_eat();
        if(button_event(&exitButton))
            return -1;
        if(button_event(&saveButton))
            return 1;
        if(button_event(&cancelButton))
            return 0;

    }
}
```

5.2.9 homepage.c

```c
/**
 * @file homepage.c
 * @author wywgwt (2504133124@qq.com)
 * @brief
 * @date 2023-02-12
 *
 * @copyright Copyright (c) 2023
 *
```

```
 */

#include "homepage.h"

void homepage()
{
    char textInputBuffer1[50];
    char textInputBuffer2[50];
    memset(textInputBuffer1, 0, sizeof(textInputBuffer1));
    memset(textInputBuffer2, 0, sizeof(textInputBuffer2));
    mouse_hide();
    setfillstyle(1, _BLACK);
    bar(0, 0, MAXWIDTH, MAXHEIGHT);
    delay(1000);
    animation_homepage1();
    setfillstyle(1, hexaa3f00);
    bar(0, 0, MAXWIDTH, MAXHEIGHT);
    image_render("res\\img\\hpbg.bin", 0, 0);
    animation_homepage2();
    setfillstyle(1, hexd4bfaa);
    bar(315, 0, MAXWIDTH, MAXHEIGHT);
    mouse_show();
    while (1)
    {
        Text tw;
        Text tw1;
        Text tc1, tc2, tc3;
        int signal = 0;
        Button b = button_new(640, 550,
                              760, 600,
                              "登录",
                              button_drawWINUI);
        TextInput t = textinput_newDefault(
                        "请输入用户名",
                        550, 370,
                        850, 420,
                        textInputBuffer1);
        TextInput t1 = textinput_newDefault(
                        "请输入密码",
                        550, 450,
                        850, 500,
                        textInputBuffer2);
        Button exitButton = button_newExitButton();
        t1.textbox.maxLength = 8;
        t.textbox.maxLength = 8;
        t1.textbox.type = TextboxPassword;
        tw = text_newDefault("用户名过短", 715, 380, 1000, 430);
        tw.font.fontColor = _RED;
        tw1 = text_newDefault("密码过短", 740, 460, 1050, 510);
        tw1.font.fontColor = _RED;
        tc1 = text_newDefault("欢迎登录！", 650, 510, 1050, 560);
        tc1.font.fontColor = _RED;
        tc2 = text_newDefault("密码错误", 740, 460, 1050, 510);
        tc2.font.fontColor = _RED;
        tc3 = text_newDefault("注册成功", 740, 460, 1050, 510);
        tc3.font.fontColor = _RED;
        // draw
        mouse_hide();
        image_render("res\\img\\hpf.bin", 0, 0);
```

```
        button_draw(&b);
        textinput_drawDefault(&t);
        textinput_drawDefault(&t1);
        ime_draw();
        button_draw(&exitButton);
        mouse_show();
        digitalClock_getTime();
        // event
        while (!signal)
        {
            int k, bs, tbs, tbs1, lgs, ebs;
            int ly = 3;
            Mouse *m = mouse();
            mouse_update();
            digitalClock_getTime();
            ime_check();
            keybord_eat();
            k = bioskey(1);
            bs = button_event(&b);
            tbs = textinput_event(&t);
            tbs1 = textinput_event(&t1);
            ebs = button_event(&exitButton);
            if (keybord_isESCAPE(k))
            {
                bioskey(0);
                signal = -1;
                break;
            }
            if(ebs){
                signal = -1;
                break;
            }
            if(tbs==1)
            {
                t1.textbox.status=TextboxSelected;
            }
            if (bs || (tbs1 == 1))
            {
                    ly = length_judge(t.textbox.content,
t1.textbox.content);
                    if (ly == 3)
                    {
                        lgs = user_login(t.textbox.content,
t1.textbox.content,0);
                        switch (lgs)
                        {
                        case 4:
                        //text_display(tc1);
                            delay(100);
                            strcpy(appData()->uid[0],t.textbox.content);
                            appData()->currentUser = appData()->uid[0];
                            signal = 1;
                            break;
                        case 5:
                        if(!(textbox_event(&t))&&!(textbox_event(&t1)))
                        {
                             text_display(tc2);
                        }
                            break;
```

```
                        case 6:
                             text_display(tc3);

user_login(t.textbox.content,t1.textbox.content,1);
                            strcpy(appData()→uid[0],t.textbox.content);
                            appData()→currentUser = appData()→uid[0];
                            delay(100);
                            signal = 1;
                            break;
                        }

                    }
            }
            switch (ly)
            {
            case 0:
                if (!(textbox_event(&t)))
                {
                    text_display(tw);
                }
                if (!(textbox_event(&t1)))
                {
                    text_display(tw1);
                }
                break;
            case 1:
                if (!(textbox_event(&t)))
                {
                    text_display(tw);
                }
                break;
            case 2:
                if (!(textbox_event(&t1)))
                {
                    text_display(tw1);
                }
                break;
            }
        }
        switch (signal)
        {
        case 1:
            app();
            break;
        case -1:
            debug(DEBUG, "EXIT.");
            return;
            break;
        case 2:
            text_display(tc2);
            delay(100);
            break;
        }
    }
}
```

5.2.10 hzinput.c

```
#include "hzinput.h"

/
*************************************************************************
FUNCTION:hzinput
DESCRIPTION: 汉字输入法
INPUT:x1,x2,y1,y2,s(input string),len(string's maxlength),color(input
box background color),color2(font color),size(font size)
RETURN:汉字个数len
IMPROVE:只能输入小写字母,可输出汉字或英文
*************************************************************************/

int hzinput(int x,int y, char *s)
{
    int i;
    int flag=0;
    int ST=-1;//输入法返回方式: 1.安SPACE键返回输入汉字 2.按ENTER键返回输入英文
3.退格键返回不输入
    char *image;
    int value=0;
    int asc;
    int barx1,barx2,bary1,bary2;
    char str[5];//一个汉字装入
    char py[15];//拼音字符串(西文字符串)s
    int color = _WHITE, color2 = _BLACK;
    int size = 16;
    int return_value = 0;
    memset(str, 0, sizeof(str));
    memset(py, 0, sizeof(py));
    x = MIN(x, MAXWIDTH - 220);
    y = MIN(y, MAXHEIGHT - 60);
    settextjustify(LEFT_TEXT,CENTER_TEXT);
    value=bioskey(1);
    /*进入汉字输入法*/
    asc=value&0xff;
    if(asc≥97&&asc≤122)
    {
        bioskey(0);
        image=ime()→buffer;
        barx1=x;        //计算输入法位置   离所输入距离较近且不溢出屏幕
        barx2=x+200;
        bary1=y;
        bary2=y+40;
        mouse_hide();
        getimage(barx1,bary1,barx2,bary2,image);
        pyFrm(barx1,bary1,barx2,bary2);
        setfillstyle(1,color);
        mouse_show();
        ST=input_method(barx1,bary1,str,value,py);
        switch(ST)
        {
            case 0:
                return_value = 0;
                break;
            case 1://由数字键或空格键退出输入法   输入汉字
                strcpy(s, str);
                return_value = 1;
                break;
            case 2://由回车键退出输入法 （键入西文）
```

```
                strcpy(s, py);
                return_value = strlen(py);
                break;
            case 3://西文删除为0自动退出输入法   不输入
                return_value = 0;
                break;
        }
        mouse_hide();
        putimage(barx1,bary1,image,0);
        mouse_show();
        // free(image);
    }
    else{
        char *fc = s;
        return_value = ime_en(s);

        switch(*s){
            case '!':
                fc = "！";
                break;
            case '.':
                fc = "。";
                break;
            case ',':
                fc = "，";
                break;
            case '?':
                fc = "？";
                break;
            case '\\':
                fc = "、";
                break;
            case '/':
                fc = "、";
                break;
            case '$':
                fc = "￥";
                break;
            case '<':
                fc = "《》";
                break;
            case '>':
                fc = "》";
                break;
            case '(':
                fc = "（）";
                break;
            case ')':
                fc = "）";
                break;
            case '_':
                fc = "——";
                return_value = 2;
                break;
            case '^':
                fc = "……";
                return_value = 2;
                break;
            case ';':
```

```
                fc = "; ";
                break;
            case ':':
                fc = ": ";
                break;
            case '`':
                fc = "·";
                break;
            case '"':
                fc = "“”";
                break;
            case '\'':
                fc = "‘’";
                break;
            case '[':
                fc = " 【 ";
                break;
            case ']':
                fc = " 】";
                break;
        }
        strcpy(s, fc);
    }

    return return_value;
}

/
*************************************************************************
FUNCTION:input_method
DESCRIPTION: 汉字输入法调入
INPUT:x,y,str,value,py
RETURN:1:输出汉字; 2: 输出字母; 3: 输出空格
*************************************************************************/

int input_method(int x,int y,char *str,int value,char *py)
{
    FILE *fp=NULL,*oldfp=NULL;
    int fJudge=FAIL;
    char *p=py;
    int trigger=1;//进入时触发输入标志
    char temphz[5][3]={{'\0','\0','\0'},{'\0','\0','\0'},
                       {'\0','\0','\0'},{'\0','\0','\0'},
                       {'\0','\0','\0'}},temp[3];
    int fposition=0;
    int hznow=0,hznum=0;
    int asc,i;
    int PyStartx=x+8,PyStarty=y+4;
    int HzStartx=x+8,HzStarty=y+22;
    char *ABpath="etc\\pinyin\\";//汉语拼音检索标准路径
    char pypath[45];                        //汉语拼音检索相对路径
    settextjustify(LEFT_TEXT,CENTER_TEXT);
    strcpy(pypath,"etc\\pinyin\\");
    while(1)
    {
        mouse_update();
        if(mouse_isClickedInBox(x, y, x+200, y+40) == -1){
            if(oldfp) fclose(oldfp);
            if(fp) fclose(fp);
```

```
            break;
        }
        digitalClock_getTime();
        if(trigger||kbhit())//第一次进入自动触发 以后均需键盘
        {
            trigger=0;
            if(kbhit()) value=bioskey(0);
            asc=value&0xff;
            /*特殊按键处理*/
            switch(value)
            {
                case KEYESCAPE:
                    if(oldfp) fclose(oldfp);
                    if(fp) fclose(fp);
                    return 0;
                    break;
                case KEYBACKSPACE:
                    p--;
                    *p='\0';
                    if(py[0]=='\0')
                    {
                        if(oldfp) fclose(oldfp);
                        if(fp) fclose(fp);
                        return 3;
                    }
                    break;
                case KEYSPACE:
                    strcpy(str,temphz[hznow]);
                    if(oldfp) fclose(oldfp);
                    if(fp) fclose(fp);
                    return 1;
                case KEYENTER:
                    if(oldfp) fclose(oldfp);
                    if(fp) fclose(fp);
                    return 2;
                case KEYUP:
                    if(fposition>=8)//接下来重定位文件指针前八个字节（四个汉
字）
                    {
                        fposition-=8;
                    }
                    break;
                case KEYDOWN:
                    if(!feof(fp))//接下来重定位文件指针后八个字节（四个汉字）
                    {
                        fposition+=8;
                    }
                    break;
                case KEYLEFT://左移动一个
                    if(hznow)
                    {
                        hznow--;
                    }
                    else if(fposition>=8)//需要左换页
                    {
                        fposition-=8;
                        hznow=3;
                    }
                    break;
```

```
                case KEYRIGHT:
                    if(hznow<hznum-1)//同左
                    {
                        hznow++;
                    }
                    else if(!feof(fp))
                    {
                        fposition+=8;
                        hznow=0;
                    }
                    break;
                    /*按数字键选中输入汉字*/
                case KEYONE:
                    strcpy(str,temphz[0]);
                    if(oldfp) fclose(oldfp);
                    if(fp) fclose(fp);
                    return 1;
                case KEYTWO:
                    strcpy(str,temphz[1]);
                    if(oldfp) fclose(oldfp);
                    if(fp) fclose(fp);
                    return 1;
                case KEYTHREE:
                    strcpy(str,temphz[2]);
                    if(oldfp) fclose(oldfp);
                    if(fp) fclose(fp);
                    return 1;
                case KEYFOUR:
                    strcpy(str,temphz[3]);
                    if(oldfp) fclose(oldfp);
                    if(fp) fclose(fp);
                    return 1;
            }
            /*输入字符处理*/
            if(asc>31&&asc<127&&strlen(py)<=6&&asc!='['&&asc!=']')    //
有效输入时则复位
            {
                *p=asc;
                p++;
                fposition=0;
                hznow=0;
            }
            mouse_hide();
            pyFrm(x,y,x+200,y+40);
            setfillstyle(1,_WHITE);
            settextstyle(1,0,2);
            settextjustify(LEFT_TEXT,CENTER_TEXT);
            outtextxy(PyStartx,PyStarty,py);          //拼音字体
            strcat(pypath,py);
            strcat(pypath,".txt");
            if(fJudge) //将当前文件指针保存 同时关闭上一个文件   为输入特殊字符准
备
            {
                if(oldfp)
                {
                    fclose(oldfp);
                }
                oldfp=fp;
            }
```

```
            if((fp=fopen(pypath,"r"))==NULL)//特殊字符存在  保留上一个文件检索结果
            {
                fJudge=FAIL;
                fp=oldfp;
            }
            else
            {
                fJudge=SUCCESS;
            }
            if(fp)
            {
                fseek(fp,fposition,SEEK_SET);
                  for(i=0;i<5;i++)
                {
                    fread(temphz[i],2,1,fp);//读入一个汉字
                    if(feof(fp))//读到文件尾
                    {
                        hznum=i;//按道理此处文件尾多读一次 需要减一  然而此处不减一的效果更好
                        break;
                    }
                }
                if(!feof(fp))//未读到文件尾 全显汉字
                {
                    hznum=4;
                }
                for(i=0;i<hznum;i++)
                {
                    setcolor(_DARKGRAY);
                    settextstyle(3, 0, 1);

outtextxy(HzStartx+i*50,HzStarty+4,itostr(i+1,temp));

hz_puthzold(HzStartx+i*50+16,HzStarty,temphz[i],16,16,_DARKGRAY);
                }

hz_puthzold(HzStartx+hznow*50+16,HzStarty,temphz[hznow],16,16,hexff7f00);//显示选中汉字
            }
        mouse_show();
        }
        strcpy(pypath,ABpath);          //绝对路径复原（不可少）
        value=0;
    }
}

/
*****************************************************************************
FUNCTION:itostr
DESCRIPTION: 数字标号
INPUT:a,s
RETURN:数字s
*****************************************************************************/

char *itostr(int a,char *s)
{
    switch(a)
    {
```

```
        case 1:
            strcpy(s,"1.");
            return s;
        case 2:
            strcpy(s,"2.");
            return s;
        case 3:
            strcpy(s,"3.");
            return s;
        case 4:
            strcpy(s,"4.");
            return s;
    }
    return s;
}

/
*************************************************************************
FUNCTION:pyFrm
DESCRIPTION: 输入法小框
INPUT:x1,y1,x2,y2
RETURN:无
*************************************************************************/

void pyFrm(int x1,int y1,int x2,int y2)
{
    setfillstyle(1,_WHITE);
    bar(x1,y1,x2,y2);
    setcolor(_BLUE);
    setlinestyle(0,0,1);
    line(x1+5,y1+20,x2-5,y1+20);
    setcolor(_DARKGRAY);
    rectangle(x1,y1,x2,y2);
}
```

5.2.11 image.c

```
/**
 * @file image.c
 * @author Hibanaw Hu (hibanaw@qq.com)
 * @brief
 * @date 2023-02-07
 *
 * @copyright Copyright (c) 2023
 *
 */

#include "image.h"

void image_render(char * filePath, int x, int y){
    const char *path = filePath;
    FILE *imgFile = fopen(path, "rb");
    int i, j;
    short w, h;
    debug(LOG, "Start rendering image.");
    if(imgFile == NULL){
```

```
        debug(ERROR, "Image load error!");
        return ;
    }
    fread(&w, sizeof(short), 1, imgFile);
    fread(&h, sizeof(short), 1, imgFile);
    for(i = 0; i < h; i++){
        for(j = 0; j < w; j++){
            char p, cp;
            p = fgetc(imgFile);
            cp = getpixel(x+j, y+i);
            if(p != cp)
                putpixel(x+j, y+i, p);
        }
    }
    fclose(imgFile);
    debug(DEBUG, "Image rendering ends.");
}

void image_renderZoom(char * filePath, int x, int y, float scale){
    const char *path = filePath;
    FILE *imgFile = fopen(path, "rb");
    int i, j;
    int k, l;
    short w, h;
    debug(LOG, "Start rendering image.");
    if(imgFile == NULL){
        debug(ERROR, "Image load error!");
        return ;
    }
    fread(&w, sizeof(short), 1, imgFile);
    fread(&h, sizeof(short), 1, imgFile);
    for(i = 0; i < h; i++){
        for(j = 0; j < w; j++){
            char p, cp;
            p = fgetc(imgFile);
            for(k = i*scale; k <(i+1)*scale; k++){
                for(l = j*scale; l <(j+1)*scale; l++){
                    cp = getpixel(x+l, y+k);
                    if(p != cp)
                        putpixel(x+l, y+k, p);
                }
            }
        }
    }
    fclose(imgFile);
}

void image_getSize(char *filePath, int *width, int *height){
    const char *path = filePath;
    FILE *imgFile = fopen(path, "rb");
    short w, h;
    if(imgFile == NULL){
        debug(ERROR, "Image load error!");
        return ;
    }
    fread(&w, sizeof(short), 1, imgFile);
    fread(&h, sizeof(short), 1, imgFile);
    *width = w;
    *height = h;
```

```
    fclose(imgFile);
}

int image_illegal(char *path){
    FILE *imgFile = fopen(path, "rb");
    if(imgFile == NULL){
        return 1;
    }
    fclose(imgFile);
    return 0;
}
```

5.2.12 imagebox.c

```
/**
 * @file imagebox.c
 * @author Hibanaw Hu (hibanaw@qq.com)
 * @brief
 * @date 2023-04-06
 *
 * @copyright Copyright (c) 2023
 *
 */

#include "imagebox.h"

void imageBox_draw(ImageBox *ib){
    char *path = ib->content;
    int x1 = ib->posX1, y1 = ib->posY1, x2 = ib->posX2, y2 = ib->posY2;
    if(ib->status == -1){
        int w, h;
        float k = 1;
        image_getSize(path, &w, &h);
        if(w>400 || h > 500){
            k = MIN((float) 400 / w, (float) 500 / h);
        }
        image_renderZoom(path, x1, y1, k);
        ib->status = 0;
    }
    switch(ib->status){
        case ButtonDefault:
            setcolor(_WHITE);
            setlinestyle(0, 0, 2);
            line(x1, y1, x1, y2);
            line(x1, y1, x2, y1);
            line(x1, y2, x2, y2);
            line(x2, y1, x2, y2);
            break;
        default:
            setcolor(hexff9f00);
            setlinestyle(0, 0, 2);
            line(x1, y1, x1, y2);
            line(x1, y1, x2, y1);
            line(x1, y2, x2, y2);
            line(x2, y1, x2, y2);
            break;
```

```
    }
}

int imageBox_event(ImageBox *ib){
    int k = button_event(ib);
    if(k == 1){
        if(clock() - ib->animationStartTime < 0.5*CLK_TCK){
            imageBox_fullScreen(ib->content);
            return 2;
        }
        else{
            ib->animationStartTime = clock();
        }
    }
    return k;
}

ImageBox imageBox_new(char *filePath, int x, int y){
    ImageBox ib;
    int w, h;
    float k;
    image_getSize(filePath, &w, &h);
    if(w>400 || h > 500){
        k = MIN((float) 400/w, (float) 500/h);
        w = k*w;
        h = k*h;
    }
    ib = button_new(x, y, x + w, y + h, filePath, imageBox_draw);
    ib.status = -1;
    return ib;
}

void imageBox_fullScreen(char *filePath){
    int x, y;
    int w, h;
    float scale = 1;
    image_getSize(filePath, &w, &h);
    while(1){
        char percentage[5];
        Button eb = button_newExitButton();
        Button greater = button_new(50, 5, 75, 30, "+",
button_drawWINUI);
        Button smaller = button_new(90, 5, 115, 30, "-",
button_drawWINUI);
        Text pt = text_newDefault(percentage, MAXWIDTH - 120, MAXHEIGHT
- 50, MAXWIDTH, MAXHEIGHT);
        int signal = 0;
        sprintf(percentage, "%d%%", (int)(scale*100));
        pt.font.fontColor = _WHITE;
        x = (MAXWIDTH - scale*w) /2;
        y = (MAXHEIGHT - scale*h) /2;
        mouse_hide();
        setfillstyle(1, hex2a1f00);
        bar(0, 0, MAXWIDTH, MAXHEIGHT);
        image_renderZoom(filePath, x, y, scale);
        button_draw(&eb);
        button_draw(&greater);
        button_draw(&smaller);
        text_display(pt);
```

```
        mouse_show();
        while(1){
            int k = bioskey(1);
            keybord_eat();
            mouse_update();
            if(button_event(&eb)){
                return;
            }
            if(keybord_isESCAPE(k)){
                bioskey(0);
                return;
            }
            if(button_event(&greater)){
                if(scale ⩽ 3){
                    scale *= 1.2;
                    break;
                }
            }
            if(button_event(&smaller)){
                if(scale ⩾ 0.3){
                    scale ⊨ 1.2;
                    break;
                }
            }
        }
    }
}
```

5.2.13 ime.c

```
/**
 * @file ime.c
 * @author Hibanaw Hu (hibanaw@qq.com)
 * @brief
 * @date 2023-03-22
 *
 * @copyright Copyright (c) 2023
 *
 */

#include "ime.h"

Ime *ime(){
    static Ime i;
    return &i;
}

/**
 * @brief Input Function
 *
 * @param s Inputed string.
 * @return int as string length. -1 if backspace.
 */
int ime_input(char *s, int x, int y){
    int k = bioskey(1);
    if(keybord_isControlOn()){
```

```
        return;
    }
    if(keybord_isBACKSPACE(k)){
        bioskey(0);
        return -1;
    }
    switch(ime()->status){
        case IMEOFF:
            return ime_en(s);
            break;
        case IMEPINYIN:
            return hzinput(x, y, s);
            break;
        case IMEEMOJI:
            break;
    }
    return 0;
}

void ime_check(){
    if(ime()->pw){
        ime()->status = IMEOFF;
        return;
    }
    // 监测快捷键切换
    if(bioskey(2)&0x04){
        if(bioskey(1) == KEYSPACE){
            bioskey(0);
            ime_next();
        }
    }
    if(button_event(&ime()->button)){
        ime_next();
    }
}

void ime_next(){
    ime() -> status = (ime() -> status + 1) % 2;
    ime_draw();
    debug(DEBUG, "IME change");
}

void ime_draw(){
    int x = 900, y = 0;
    Text t = text_newDefault(NULL, x+3, y+2, 0, 0);
    int status = ime()->status;
    t.font.fontSize = 16;
    t.font.fontColor = _BLACK,
    t.font.spacing = 2;
    t.font.rowSpacing = 0;
    setfillstyle(1, _BLACK);
    bar(x, y, x+20, y+20);
    setfillstyle(1, _WHITE);
    bar(x+1, y+1, x+19, y+19);
    switch (status){
        case IMEOFF:
            t.content = "英";
            break;
        case IMEPINYIN:
```

```
            t.content = "中";
            break;
        case IMEEMOJI:
            t.content = "表";
            break;
    }
    text_display(t);
}

int ime_en(char *s){
    int k = bioskey(1);
    if(keybord_isCharacter(k)){
        bioskey(0);
        *s = keybord_bios2ascii(k);
        return 1;
    }
    return 0;
}

void ime_init(){
    Ime *i = ime();
    memset(i, 0, sizeof(Ime));
    i→button = button_new(
        900, 0,
        920, 20,
        NULL,
        ime_draw
    );
}
```

5.2.14 init.c

```
/**
 * @file init.c
 * @author Hibanaw Hu (hibanaw@qq.com)
 * @brief
 * @date 2023-02-12
 *
 * @copyright Copyright (c) 2023
 *
 */

#include "init.h"

void init(){
    //debug(LOG, "Init.");
    initsvga256();
    useRGB256Colors();
    mouse_init();
    ime_init();
    debug(DEBUG, "Init ends.");
    return ;
}

void destruct(){
    debug(LOG, "Destruct.");
```

```
    closesvga256();
}
```

5.2.15 keyboard.c

```
/**
 * @file keyboard.c
 * @author Hibanaw Hu (hibanaw@qq.com)
 * @brief
 * @date 2023-02-12
 *
 * @copyright Copyright (c) 2023
 *
 */

#include"keyboard.h"

char keybord_bios2ascii(unsigned k){
    return (char)k&0xff;
}

int keybord_isAlphabet(unsigned k){
    char a = keybord_bios2ascii(k);
    return a ⩾ 'a' && a ⩽ 'z' || a ⩾ 'A' && a ⩽ 'Z';
}

int keybord_isCharacter(unsigned k){
    char a = keybord_bios2ascii(k);
    return a ⩾ 32 && a ⩽ 126;
}

int keybord_isESCAPE(unsigned k){
    return k == KEYESCAPE;
}

int keybord_isBACKSPACE(unsigned k){
    return k == KEYBACKSPACE;
}

int keybord_isControlOn(){
    return bioskey(2)&0x04;
}

void keybord_eat(){
    static k;
    int tk = bioskey(1);
    if(tk ≠ 0 && tk == k){
        bioskey(0);
    }
    k = tk;
}
```

5.2.16 main.c

```c
/**
 * @file main.c
 * @author Hibanaw Hu (hibanaw@qq.com)
 * @brief
 * @date 2023-02-12
 *
 * @copyright Copyright (c) 2023
 *
 */

#include "main.h"

int main(){
    init();
    homepage();
    destruct();
    return 0;
}
```

5.2.17 meditor.c

```c
/**
 * @file meditor.c
 * @author Hibanaw Hu (hibanaw@qq.com)
 * @brief
 * @date 2023-04-02
 *
 * @copyright Copyright (c) 2023
 *
 */

#include "meditor.h"

void memoEditor_draw(MemoEditor *e){
    MemoEditor me = *e;
    setfillstyle(1, hexfffbf0);
    bar(me.posX, me.posY, MAXWIDTH, me.posY+40);
    bar(me.posX, me.posY, me.posX+7, me.posY+7);
    setcolor(hexd4dfff);
    setfillstyle(1, _WHITE);
    arc(me.posX+6, me.posY+47, 90, 180, 7);
    setlinestyle(0, 1, 2);
    line(me.posX+6, me.posY+40, MAXWIDTH, me.posY+40);
    button_draw(&me.drawButton);
    button_draw(&me.imageButton);
    button_draw(&me.checkboxButton);
    button_draw(&me.saveButton);
    button_draw(&me.shareButton);
    button_draw(&me.settingsButton);
    mouse_show();
    memoEditor_updateList(e);
    mouse_hide();
    me.titleBar.draw(&me.titleBar);
}

MemoEditor memoEditor_new(char *fileName, char *uid){
```

```
    MemoEditor me;
    Memo *m = memo();
    char filePath[20];
    memset(m, 0, sizeof(m));
    sprintf(filePath, "data\\%s.MEM", fileName);
    *m = memofile_read(filePath);
    memset(&me, 0, sizeof(me));
    me.fileName         = fileName;
    me.beginMemoBlock   = m->head;
    me.posX             = 75;
    me.posY             = 30;
    me.focusedBlock     = NULL;
    me.drawButton       = button_new(me.posX + 10, me.posY+3, me.posX +
42, me.posY + 35, "", memoEditor_button_drawDrawpad);
    me.imageButton      = button_new(me.posX + 52, me.posY+3, me.posX +
84, me.posY + 35, "", memoEditor_button_drawAddPicture);
    me.checkboxButton   = button_new(me.posX + 94, me.posY+3, me.posX +
126, me.posY + 35, "", memoEditor_button_drawAddCheckbox);
    me.shareButton      = button_new(MAXWIDTH - 126, me.posY+3, MAXWIDTH
- 94, me.posY + 35, "", memoEditor_button_drawSharebutton);
    me.settingsButton   = button_new(MAXWIDTH - 84, me.posY+3, MAXWIDTH
- 52, me.posY + 35, "", memoEditor_button_drawEditbutton);
    me.saveButton       = button_new(MAXWIDTH - 42, me.posY+3, MAXWIDTH
- 10, me.posY + 35, "", memoEditor_button_drawSavebutton);
    me.uid              = uid;
    me.scrollBar        = scrollBar_new(MAXWIDTH - 15, me.posY + 50,
MAXHEIGHT - (me.posY + 60));
    me.titleBar         = textinput_newTitle("无标题", 300, 35, 794, 65,
memo()->title);
    me.unSaved          = 0;
    me.titleBar.textbox.bgColor = hexfffbf0;
    me.authType = auth_get(fileName).type;
    strcpy(memo()->fileName, me.fileName);
    return me;
}

/**
 * @brief
 *
 * @param me
 * @return
 */
int memoEditor_event(MemoEditor *me){
    MemoBlock *p = me->beginMemoBlock;
    int k = bioskey(1);
    int i;
    int ss;
    int sum = memo_getBlockSum();
    MemoBlock *mb;
    if(button_event(&me->shareButton)){
        share_page(me->fileName);
        memoEditor_updateList(me);
        return 0;
    }
    if(button_event(&me->settingsButton)){
        if(mset_page(me)){
            memoEditor_save(me);
        }
        memoEditor_updateList(me);
```

```
        return 0;
    }
    if(button_event(&me->saveButton)){
        memoEditor_save(me);
    }
    ss = scrollBar_event(&me->scrollBar);
    textinput_event(&me->titleBar);
    switch (ss)
    {
        case -1:
            if(me->beginMemoBlock != memo()->head){
                me->beginMemoBlock = memo_preBlock(me->beginMemoBlock);
                memoEditor_updateList(me);
            }
            break;
        case 1:
            if(me->list.memoBlock[me->list.count-1]->next){
                me->beginMemoBlock = me->beginMemoBlock->next;
                memoEditor_updateList(me);
            }
            break;
    }
    // button
    if(button_event(&me->drawButton)){
        int tg;
        char path[100];
        if(sum >= BLOCKMAX)
            return 0;
        sprintf(path, "data/%s", memo()->fileName);
        tg = drawPad(path);
        if(tg == 0){
            if(me->focusedBlock == NULL){
                mb = memo_newBlock(IMAGE, 0, path);
                strcpy(mb->lastEditUser, me->uid);
                me->unSaved = 1;
                memo_addBlock(mb);
            }
            else{
                if(*(me->focusedBlock->content) == 0){
                    strcpy(me->focusedBlock->content, path);
                    me->focusedBlock->type = IMAGE;
                    strcpy(me->focusedBlock->lastEditUser, me->uid);
                    me->unSaved = 1;
                }
                else{
                    mb = memo_newBlock(IMAGE, 0, path);
                    strcpy(mb->lastEditUser, me->uid);
                    me->unSaved = 1;
                    memo_insertBlock(me->focusedBlock, mb);
                    me->focusedBlock = mb;
                }
            }
        }
        memoEditor_updateList(me);
        return 0;
    }
    if(button_event(&me->imageButton)){
        char path[50];
        int tg;
```

```
        memset(path, 0, sizeof(path));
        if(sum ⩾ BLOCKMAX)
                return 0;
        tg = addImage(path);
        if(tg == 0){
            if(me→focusedBlock == NULL){
                mb = memo_newBlock(IMAGE, 0, path);
                strcpy(mb→lastEditUser, me→uid);
                me→unSaved = 1;
                memo_addBlock(mb);
            }
            else{
                if(*(me→focusedBlock→content) == 0){
                    strcpy(me→focusedBlock→content, path);
                    me→focusedBlock→type = IMAGE;
                    strcpy(me→focusedBlock→lastEditUser, me→uid);
                    me→unSaved = 1;
                }
                else{
                    mb = memo_newBlock(IMAGE, 0, path);
                    strcpy(mb→lastEditUser, me→uid);
                    memo_insertBlock(me→focusedBlock, mb);
                    me→unSaved = 1;
                    me→focusedBlock = mb;
                }
            }
        }
        memoEditor_updateList(me);
        return 0;
    }
    if(button_event(&me→checkboxButton)){
        if(me→focusedBlock == NULL){
            if(sum ⩾ BLOCKMAX)
                return 0;
            mb = memo_newBlock(CHECKBOX, 0, "");
            strcpy(mb→lastEditUser, me→uid);
            me→unSaved = 1;
            memo_addBlock(mb);
            memoEditor_updateList(me);
            return 0;
        }
        else{
            if(me→focusedBlock→type == CHECKBOX){
                me→focusedBlock→type = PARAGRAPH;
            }
            else{
                me→focusedBlock→type = CHECKBOX;
            }
            strcpy(me→focusedBlock→lastEditUser, me→uid);
            me→unSaved = 1;
            memoEditor_updateList(me);
            return 0;
        }
    }
    //page move
    if(me→focusedBlock == NULL){
        MemoBlock *mb;
        switch (k){
            case KEYUP:
```

```
                bioskey(0);

                if(me->beginMemoBlock != memo()->head){
                    me->beginMemoBlock = memo_preBlock(me-
>beginMemoBlock);
                    memoEditor_updateList(me);
                }
                break;
            case KEYDOWN:
                bioskey(0);
                if(me->beginMemoBlock->next){
                    me->beginMemoBlock = me->beginMemoBlock->next;
                    memoEditor_updateList(me);
                }
        }

    }
    else{
        switch (k){
            case KEYUP:
                bioskey(0);
                if(me->focusedBlock != me->beginMemoBlock){
                    me->focusedBlock = memo_preBlock(me->focusedBlock);
                    me->focusedBlockCursorLocation = MIN(me-
>focusedBlockCursorLocation, text_getLength(me->focusedBlock->content));
                    memoEditor_updateList(me);
                }
                else{
                    if(me->beginMemoBlock != memo()->head){
                        me->beginMemoBlock = memo_preBlock(me-
>beginMemoBlock);
                        me->focusedBlock = memo_preBlock(me-
>focusedBlock);
                        me->focusedBlockCursorLocation = MIN(me-
>focusedBlockCursorLocation, text_getLength(me->focusedBlock->content));
                        memoEditor_updateList(me);
                    }
                }
                break;
            case KEYDOWN:
                bioskey(0);
                if(me->focusedBlock != me->list.memoBlock[me-
>list.count-1]){
                    me->focusedBlock = me->focusedBlock->next;
                    me->focusedBlockCursorLocation = MIN(me-
>focusedBlockCursorLocation, text_getLength(me->focusedBlock->content));
                    memoEditor_updateList(me);
                }
                else{
                    if(me->beginMemoBlock != memo_preBlock(memo()-
>head)){
                        me->beginMemoBlock = me->beginMemoBlock->next;
                        me->focusedBlock = (me->focusedBlock->next !=
NULL ? me->focusedBlock->next : me->focusedBlock);
                        me->focusedBlockCursorLocation = MIN(me-
>focusedBlockCursorLocation, text_getLength(me->focusedBlock->content));
                        memoEditor_updateList(me);
                    }
                }
```

```
        }
    }
    for(i = 0; i < me->list.count; i++){
        Textbox *tb;
        int lh;
        int ls;
        ImageBox *ib;
        MemoEditor_Checkbox *cb;
        int trigger;
        int trigger2;
        char lc[160];
        strcpy(lc, me->list.memoBlock[i]->content);
        switch (me->list.type[i]){
            case PARAGRAPH:
                tb = me->list.widget[i];
                lh = text_getHeight(textbox_convert2text(*tb));
                ls = tb->status;
                trigger = textbox_event(tb);
                if(ls == TextboxSelected && tb->status == TextboxDefault
&& mouse_isClickedInBox(me->posX, me->posY + 3, me->posX + 126, me->posY
+ 35) == 1){
                    tb->status = TextboxSelected;
                }
                if(tb->status == TextboxSelected){
                    me->focusedBlock = me->list.memoBlock[i];
                    me->focusedBlockCursorLocation = tb->cursorLocation;
                }
                if(tb->status == TextboxSelected && lh !=
text_getHeight(textbox_convert2text(*tb))){
                    memoEditor_updateList(me);
                    return 0;
                }
                if(trigger == 1){
                    MemoBlock *mb;
                    char sn[160];
                    if(sum >= BLOCKMAX)
                        return 0;
                    strcpy(sn, text_getNthChar(tb->content, tb-
>cursorLocation));
                    mb = memo_newBlock(PARAGRAPH, 0, sn);
                    *text_getNthChar(tb->content, tb->cursorLocation) =
0;
                    me->focusedBlockCursorLocation = 0;
                    me->focusedBlock = memo_insertBlock(me-
>list.memoBlock[i], mb);
                    strcpy(me->focusedBlock->lastEditUser, me->uid);
                    me->unSaved = 1;
                    if(tb->posY2 > MAXHEIGHT - 100){
                        me->beginMemoBlock = me->beginMemoBlock->next;
                    }
                    memoEditor_updateList(me);
                    return 0;
                }
                if(trigger == -1){
                    if(me->focusedBlock != memo()->head){
                        MemoBlock *mb = memo_preBlock(me-
>list.memoBlock[i]);
                        if(text_getLength(mb->content) +
text_getLength(me->list.memoBlock[i]->content) <= 75){
```

```
                            me->focusedBlockCursorLocation =
text_getLength(mb->content);
                            strcat(mb->content, tb->content);
                            me->focusedBlock = mb;
                            if(me->list.memoBlock[i] == me-
>beginMemoBlock){
                                me->beginMemoBlock = memo_preBlock(me-
>list.memoBlock[i]);
                            }
                            memo_deleteBlock(me->list.memoBlock[i]);
                            memoEditor_updateList(me);
                            return 0;
                        }
                    }
                }
                if(me->list.memoBlock[i] == me->focusedBlock && tb-
>status != TextboxSelected){
                    me->focusedBlock = NULL;
                }
                break;
            case IMAGE:
                ib = me->list.widget[i];
                trigger = imageBox_event(ib);
                if(trigger == 1){
                    me->focusedBlock = me->list.memoBlock[i];
                    me->focusedBlockCursorLocation = 1;
                }
                if(trigger == 2){
                    return 1;
                }
                if(me->focusedBlock == me->list.memoBlock[i] &&
mouse_isClickedInBox(ib->posX1, ib->posY1, ib->posX2, ib->posY2) == -1
&& mouse_isClickedInBox(me->posX, me->posY + 3, me->posX + 126, me->posY
+ 35) != 1){
                    me->focusedBlock = NULL;
                }
                if(me->focusedBlock == me->list.memoBlock[i]){
                    MemoBlock *mb;
                    switch(bioskey(1)){
                        case KEYBACKSPACE:
                            bioskey(0);
                            me->focusedBlock->type = PARAGRAPH;
                            me->focusedBlockCursorLocation =
text_getLength(me->focusedBlock->content);
                            strcpy(me->focusedBlock->lastEditUser, me-
>uid);
                            me->unSaved = 1;
                            memoEditor_updateList(me);
                            return 0;
                            break;
                        case KEYENTER:
                            bioskey(0);
                            mb = memo_newBlock(PARAGRAPH, 0, "");
                            me->focusedBlockCursorLocation = 0;
                            me->focusedBlock = memo_insertBlock(me-
>list.memoBlock[i], mb);
                            strcpy(me->focusedBlock->lastEditUser, me-
>uid);
                            me->unSaved = 1;
```

```
                            memoEditor_updateList(me);
                            return 0;
                            break;
                        }
                    }
                    break;
                case CHECKBOX:
                    cb = me->list.widget[i];
                    ls = cb->textbox.status;
                    tb = &cb->textbox;
                    lh = text_getHeight(textbox_convert2text(*tb));
                    trigger = button_checkboxEvent(&cb->checkbox);
                    trigger2 = textbox_event(&cb->textbox);
                    if(ls == TextboxSelected && tb->status == TextboxDefault
&& mouse_isClickedInBox(me->posX, me->posY + 3, me->posX + 126, me->posY
+ 35) == 1){
                        tb->status = TextboxSelected;
                    }
                    if(tb->status == TextboxSelected){
                        me->focusedBlock = me->list.memoBlock[i];
                        me->focusedBlockCursorLocation = tb->cursorLocation;
                    }
                    if(tb->status == TextboxSelected && lh !=
text_getHeight(textbox_convert2text(*tb))){
                        memoEditor_updateList(me);
                        return 0;
                    }
                    if(trigger){
                        me->list.memoBlock[i]->checkBoxisChecked = (int)cb-
>checkbox.content;
                    }
                    if (trigger2 == 1){
                            MemoBlock *mb;
                            char sn[310];
                            strcpy(sn, text_getNthChar(tb->content, tb-
>cursorLocation));
                            mb = memo_newBlock(CHECKBOX, 0, sn);
                            *text_getNthChar(tb->content, tb-
>cursorLocation) = 0;
                            me->focusedBlockCursorLocation = 0;
                            me->focusedBlock = memo_insertBlock(me-
>list.memoBlock[i], mb);
                            strcpy(me->focusedBlock->lastEditUser, me->uid);
                            me->unSaved = 1;
                            memoEditor_updateList(me);
                            return 0;
                    }
                    if(trigger2 == -1){
                        me->focusedBlock->type = PARAGRAPH;
                        strcpy(me->focusedBlock->lastEditUser, me->uid);
                        me->unSaved = 1;
                        memoEditor_updateList(me);
                        return 0;
                    }
                    break;
            }
            if(strcmp(lc, me->list.memoBlock[i]->content)){
                strcpy(me->list.memoBlock[i]->lastEditUser, me->uid);
                me->unSaved = 1;
```

```
        }
    }
    return 0;
}

void memoEditor_updateList(MemoEditor *e){
    int x;
    int y = e->posY + 60;
    int i;
    int dy;
    int sum = memo_getBlockSum();
    int no = memo_getBlockNum(e->beginMemoBlock);
    MemoBlock *p = e->beginMemoBlock;
    x = e->posX + ((appData()->displayLastEditUser)? 150 : 0);
    if(memo()->head == NULL){
        MemoBlock *mb = memo_newBlock(PARAGRAPH, 0, "");
        strcpy(mb->lastEditUser, e->uid);
        memo_addBlock(mb);
    }
    if(p == NULL){
        p = e->beginMemoBlock = memo()->head;
    }
    for(i = 0; i < e->list.count; i++){
        if(e->list.widget[i] != NULL){
            free(e->list.widget[i]);
            e->list.widget[i] = NULL;
        }
    }
    mouse_hide();
    setfillstyle(1, _WHITE);
    bar(e->posX+5, y-18, MAXWIDTH-20, MAXHEIGHT);
    e->list.count = 0;
    i = 0;
    while(i < 25 && y < MAXHEIGHT && p != NULL){
        Textbox *tb;
        ImageBox *ib;
        MemoEditor_Checkbox *cb;

        if(appData()->displayLastEditUser){
            Text ut;
            char buf[30];
            memset(buf, 0, sizeof(buf));
            strcat(buf, "<");
            strcat(buf, p->lastEditUser);
            strcat(buf, ">");
            ut = text_newSmall(buf, x-text_getLength(buf)*16+10, y+5);
            text_display(ut);
        }
        e->list.type[i] = p->type;
        e->list.memoBlock[i] = p;
        switch (p->type){
        case PARAGRAPH:
            tb = malloc(sizeof(Textbox));
            e->list.widget[i] = tb;
            *tb = textbox_newDefault("新的段落", x + 20, y, MAXWIDTH - 30,
y, p->content);
            dy = text_getHeight(textbox_convert2text(*tb));
            tb->posY2 = y + dy;
            if(p == e->focusedBlock){
```

```
                tb->status = TextboxSelected;
                tb->cursorLocation = e->focusedBlockCursorLocation;
            }
            textbox_draw(tb);
            y += dy;
            break;
        case IMAGE:
            if(image_illegal(p->content)){
                p->type = PARAGRAPH;
                continue;
            }
            y+=10;
            ib = malloc(sizeof(ImageBox));
            e->list.widget[i] = ib;
            *ib = imageBox_new(p->content, x + 20, y);
            ib->status = -1;
            y+=ib->posY2 - ib->posY1;
            y+=10;
            imageBox_draw(ib);
            break;
        case CHECKBOX:
            cb = malloc(sizeof(MemoEditor_Checkbox));
            e->list.widget[i] = cb;
            cb->checkbox = button_new(x+20, y, x+44, y+24, 0,
button_checkboxDraw);
            switch (p->checkBoxisChecked)
            {
                case 0:
                    cb->checkbox.content = 0;
                break;
                case 1:
                    cb->checkbox.content = 1;
                break;
            }
            cb->textbox = textbox_newDefault("新的任务", x + 50, y,
MAXWIDTH - 30, y, p->content);
            dy = text_getHeight(textbox_convert2text(cb->textbox));
            cb->textbox.posY2 = y + dy;
            if(p == e->focusedBlock){
                cb->textbox.status = TextboxSelected;
                cb->textbox.cursorLocation = e-
>focusedBlockCursorLocation;
            }
            textbox_draw(&cb->textbox);
            button_draw(&cb->checkbox);
            y += dy;
            break;
        }
        p = p->next;
        i++;
        y+= 10;
    }
    e->list.count = i;
    e->scrollBar.ithItem = no;
    e->scrollBar.sumItem = sum;
    e->scrollBar.inScreenItem = i;
    scrollBar_event(&e->scrollBar);
    scrollBar_draw(&e->scrollBar);
    mouse_show();
```

```
}

void memoEditor_distruct(MemoEditor *me){
    int i = 0;
    for(i = 0; i < 25; i++){
        if(me->list.widget[i] != NULL){
            free(me->list.widget[i]);
            me->list.widget[i] = NULL;
        }
    }
    while(memo()->head != NULL){
        memo_deleteBlock(memo()->head);
    }
    memset(memo(), 0, sizeof(Memo));
}

void memoEditor_save(MemoEditor *me){
    char filePath[20];
    sprintf(filePath, "data//%s.MEM", me->fileName);
    memofile_write(filePath, memo());
    auth_set(me->fileName, memo()->owner, me->authType);
    me->unSaved = 0;
}


void memoEditor_button_drawAddPicture(Button *b)
{
    int x1 = b->posX1 + 5, y1 = b->posY1 + 5,
        x2 = b->posX2 - 5, y2 = b->posY2 - 5;
    float k = 0.8;
    int x3, x4, y3, y4, m;
    button_drawWINUI(b);
    if (b->status != ButtonSelected)
    {
        setcolor(_BLACK);
        setlinestyle(0, 1, 2);
        // setfillstyle(1,_WHITE);
        // bar(x1,y1,x2,y2);
        line(x1,y1,x2,y1);
        line(x2,y1,x2,y2);
        line(x2,y2,x1,y2);
        line(x1,y2,x1,y1);
        arc(x1, y2, 0, 90, (x2 - x1) / 2);
        arc(x2, y2 + (y2 - y1) / 4, 90, 135, 3 * (y2 - y1) / 4);
        circle((x2 + 3 * x1) / 4, (3 * y1 + y2) / 4, 2);
    }
    else
    {
        x3 = (x1 + x2) / 2 - (x2 - x1) * k / 2;
        x4 = (x1 + x2) / 2 + (x2 - x1) * k / 2;
        y3 = (y1 + y2) / 2 - (y2 - y1) * k / 2;
        y4 = (y1 + y2) / 2 + (y2 - y1) * k / 2;
        setcolor(_BLACK);
        setlinestyle(0,1,2);
        // setfillstyle(1,_WHITE);
        // bar(x3,y3,x4,y4);
        line(x3,y3,x4,y3);
        line(x4,y3,x4,y4);
        line(x4,y4,x3,y4);
```

```
        line(x3,y4,x3,y3);
        arc(x3, y4, 0, 90, (x4 - x3) / 2);
        arc(x4, y4 + (y4 - y3) / 4, 90, 135, 3 * (y4 - y3) / 4);
        circle((x4 + 3 * x3) / 4, (3 * y3 + y4) / 4, 2 * k);
    }
}
void memoEditor_button_drawAddCheckbox(Button *b)
{
    int x1 = b->posX1 + 5, y1 = b->posY1 + 5,
        x2 = b->posX2 - 5, y2 = b->posY2 - 5;
    int x3, x4, y3, y4, m;
    float k;
    button_drawWINUI(b);
    if (b->status != ButtonSelected)
    {
        k = 1;
    }
    else
    {
        k = 0.8;
    }
    x3 = (x1 + x2) / 2 - (x2 - x1) * k / 2;
    x4 = (x1 + x2) / 2 + (x2 - x1) * k / 2;
    y3 = (y1 + y2) / 2 - (y2 - y1) * k / 2;
    y4 = (y1 + y2) / 2 + (y2 - y1) * k / 2;
    setcolor(_BLACK);
    setlinestyle(0,1, 2);
      line(x3,y3,x4,y3);
        line(x4,y3,x4,y4);
        line(x4,y4,x3,y4);
        line(x3,y4,x3,y3);
    line((3 * x3 + x4) / 4, (y3 + y4) / 2, (x3 + x4) / 2, (3 * y4 +
y3) / 4);
    line((x3 + x4) / 2, (3 * y4 + y3) / 4, (3 * x4 + x3) / 4, (3 * y3 +
y4) / 4);
}
void memoEditor_button_drawDrawpad(Button *b)
{
    int x1 = b->posX1+3, y1 = b->posY1+3,
        x2 = b->posX2-3, y2 = b->posY2-3;
    float k=0.8;
    int x3, x4, y3, y4, m;
    button_drawWINUI(b);
     if (b->status != ButtonSelected)
    {
        k = 1;
    }
    else
    {
        k = 0.8;
    }
    x3 = (x1 + x2) / 2 - (x2 - x1) * k / 2;
    x4 = (x1 + x2) / 2 + (x2 - x1) * k / 2;
    y3 = (y1 + y2) / 2 - (y2 - y1) * k / 2;
    y4 = (y1 + y2) / 2 + (y2 - y1) * k / 2;
    setcolor(_BLACK);
    setlinestyle(0,1,2);
    line((4*x3+x4)/5,(y3+2*y4)/3,(x3+2*x4)/3,(4*y3+y4)/5);
    line((x3+2*x4)/3,(4*y3+y4)/5,(x3+x4)/2,(2*y3+3*y4)/5);
```

```
    line((x3+x4)/2,(2*y3+3*y4)/5,(2*x4+x3)/3,(3*y3+4*y4)/7);
    line((2*x4+x3)/3,(3*y3+4*y4)/7,(3*x3+7*x4)/10,(4*y4+1*y3)/5);
    line((3*x3+7*x4)/10,(4*y4+1*y3)/5,(9*x4+x3)/10,(4*y4+1*y3)/5);
}
void memoEditor_button_drawSharebutton(Button *b)
{
   int x1 = b->posX1+2, y1 = b->posY1+3,
       x2 = b->posX2-4, y2 = b->posY2-3;
    float k=0.8;
    int x3,x4,y3,y4;
    button_drawWINUI(b);
      if (b->status != ButtonSelected)
    {
        k = 1;
    }
    else
    {
        k = 0.8;
    }
    x3 = (x1 + x2) / 2 - (x2 - x1) * k / 2;
    x4 = (x1 + x2) / 2 + (x2 - x1) * k / 2;
    y3 = (y1 + y2) / 2 - (y2 - y1) * k / 2;
    y4 = (y1 + y2) / 2 + (y2 - y1) * k / 2;
    setcolor(_BLACK);
    setlinestyle(0,1,2);
    arc((5*x4+3*x3)/8,(6*y4+2*y3)/8,90,180,3*(y4-y3)/8);
    arc((5*x4+3*x3)/8,(33*y4-y1)/32,90,157.38,13*(y4-y3)/32);
    line((5*x4+3*x3)/8,(5*y3+3*y4)/8,(5*x4+3*x3)/8,(7*y3+y4)/8);
    line((5*x4+3*x3)/8,(5*y4+3*y3)/8,(5*x4+3*x3)/8,(7*y4+y3)/8);
    line((5*x4+3*x3)/8,(7*y3+y4)/8,(7*x4+x3)/8,(y3+y4)/2);
    line((5*x4+3*x3)/8,(7*y4+y3)/8,(7*x4+x3)/8,(y3+y4)/2);
}

void memoEditor_button_drawEditbutton(Button *b)
{
     int x1 = b->posX1+3, y1 = b->posY1+3,
        x2 = b->posX2-3, y2 = b->posY2-3;
    float k=0.8;
    int x3, x4, y3, y4, m;
    button_drawWINUI(b);
     if (b->status != ButtonSelected)
    {
        k = 1;
    }
    else
    {
        k = 0.8;
    }
    x3 = (x1 + x2) / 2 - (x2 - x1) * k / 2;
    x4 = (x1 + x2) / 2 + (x2 - x1) * k / 2;
    y3 = (y1 + y2) / 2 - (y2 - y1) * k / 2;
    y4 = (y1 + y2) / 2 + (y2 - y1) * k / 2;
    setcolor(_BLACK);
    setlinestyle(0,1,2);
    line((3*x4+x3)/4,(5*y3+y4)/6,(5*x4+x3)/6,(3*y3+y4)/4);
    line((5*x4+x3)/6,(3*y3+y4)/4,(3*x3+x4)/4,(5*y4+y3)/6);
    line((3*x3+x4)/4,(5*y4+y3)/6,(7*x3+x4)/8,(7*y4+y3)/8);
    line((7*x3+x4)/8,(7*y4+y3)/8,(5*x3+x4)/6,(3*y4+y3)/4);
    line((5*x3+x4)/6,(3*y4+y3)/4,(3*x4+x3)/4,(5*y3+y4)/6);
```

```
    arc((2*x3+x4)/3+(y4-y3)/15,(4*y4+y3)/5,0,180,(y4-y3)/10);
    arc((2*x3+x4)/3+2*(y4-y3)/10+(y4-y3)/15,(4*y4+y3)/5,180,360,(y4-
y3)/10);
    arc((2*x3+x4)/3+2*2*(y4-y3)/10+(y4-y3)/15,(4*y4+y3)/5,0,180,(y4-
y3)/10);
    arc((2*x3+x4)/3+3*2*(y4-y3)/10+(y4-y3)/15,(4*y4+y3)/5,180,360,(y4-
y3)/10);
}
void memoEditor_button_drawSavebutton(Button *b)
{
     int x1 = b->posX1+5, y1 = b->posY1+6,
        x2 = b->posX2-5, y2 = b->posY2-5;
    float k=0.8;
    int x3, x4, y3, y4, m;
    button_drawWINUIAccent(b);
     if (b->status != ButtonSelected)
    {
        k = 1;
    }
    else
    {
        k = 0.8;
    }
    x3 = (x1 + x2) / 2 - (x2 - x1) * k / 2;
    x4 = (x1 + x2) / 2 + (x2 - x1) * k / 2;
    y3 = (y1 + y2) / 2 - (y2 - y1) * k / 2;
    y4 = (y1 + y2) / 2 + (y2 - y1) * k / 2;
    setcolor(_BLACK);
    setlinestyle(0,1,2);
    line(x3,y3,x4,y3);
    line(x4,y3,x4,y4);
    line(x4,y4,(4*x3+x4)/5,y4);
    line((4*x3+x4)/5,y4,x3,(4*y4+y3)/5);
    line(x3,(4*y4+y3)/5,x3,y3);
    line((7*x3+3*x4)/10,y4,(7*x3+3*x4)/10,(2*y4+y3)/3);
    line((7*x3+3*x4)/10,(2*y4+y3)/3,(7*x3+3*x4)/10+2*(x4-x3)/5,
(2*y4+y3)/3);
    line((7*x3+3*x4)/10+2*(x4-x3)/5,(2*y4+y3)/3,(7*x3+3*x4)/10+2*(x4-
x3)/5,y4);
    line((4*x3+x4)/5,y3,(4*x3+x4)/5,(2*y3+y4)/3);
    line((4*x3+x4)/5,(2*y3+y4)/3,(4*x4+x3)/5,(2*y3+y4)/3);
    line((4*x4+x3)/5,(2*y3+y4)/3,(4*x4+x3)/5,y3);
}
```

5.2.18 memo.c

```
/**
 * @file memo.c
 * @author wywgwt (2504133124@qq.com), Hibanaw Hu (hibanaw@qq.com)
 * @brief
 * @date 2023-04-03
 *
 * @copyright Copyright (c) 2023
 *
 */
#include "memo.h"
```

```
Memo *memo()
{
    static Memo m;
    return &m;
}

void *memo_addBlock(MemoBlock *a)
{
    MemoBlock *h = memo()->head;
    if (h == NULL)
    {
        memo()->head = a;
    }
    else
    {
        MemoBlock *p = h;
        while (p->next != NULL)
        {
            p = p->next;
        }
        p->next = a;
        a->next = NULL;
    }
}

MemoBlock *memo_insertBlock(MemoBlock *p, MemoBlock *a)
{
    MemoBlock *q, *t;
    t = p->next;
    p->next = a;
    a->next = t;
    return (a);
}
// 在第一个参数的位置之后插入新Node，新node的内容是第二个参数，返回插入的node的地
址。
MemoBlock *memo_deleteBlock(MemoBlock *p)
{
    MemoBlock *head = memo()->head;
    MemoBlock *q;
    q = head;
    if(q == p){
        memo()->head = p->next;
        free(p);
        return (q->next);
    }
    while (q != NULL && (q->next) != p)
    {
        q = q->next;
    }
    q->next = p->next;
    free(p);
    return (q->next);
}
// 删除参数所在的节点，返回该节点的next。

MemoBlock *memo_newBlock(Memotype type, int checkBoxisChecked, char
*content)
{
```

```
    MemoBlock *q, *t;
    q = (MemoBlock *)malloc(sizeof(MemoBlock));
    if (q == NULL)
    {
        printf("no enough memory");
        delay(5000);
        exit(1);
    }
    else
    {
        memset(q, 0, sizeof(MemoBlock));
        q->type = type;
        q->checkBoxisChecked = checkBoxisChecked;
        strcpy(q->content, content);
        q->next = NULL;
    }
    return q;
}
//新建

MemoBlock *memo_preBlock(MemoBlock *p)
{
    MemoBlock *head = memo()->head;
    MemoBlock *q = head;
    while((q->next!=NULL)&&(q->next)!=p)
    {
        q=q->next;
    }
    return(q);
}

int memo_getBlockSum(){
    int s = 0;
    MemoBlock *p = memo()->head;
    while(p != NULL){
        p = p->next;
        s++;
    }
    return s;
}

int memo_getBlockNum(MemoBlock *mb){
    int s = 0;
    MemoBlock *p = memo()->head;
    while(p != NULL && p != mb){
        p = p->next;
        s++;
    }
    return s;
}
```

5.2.19 memos.c

```
/**
 * @file mfile.c
 * @author wywgwt (2504133124@qq.com), Hibanaw Hu (hibanaw@qq.com)
```

```
 * @brief
 * @date 2023-04-06
 *
 * @copyright Copyright (c) 2023
 *
 */

#include "memos.h"
Memos *memos()
{
    static Memos m;
    return &m;
}

void *memos_add(Memo *a){
    Memo *h = memos()->head;
    if (h == NULL)
    {
        memos()->head = a;
    }
    else
    {
        Memo *p = h;
        while (p->next != NULL)
        {
            p = p->next;
        }
        p->next = a;
        a->next = NULL;
    }
}

Memo *memos_insertMemo(Memo *p, Memo *a)
{
    Memo *t;
    t = p->next;
    p->next = a;
    a->next = t;
    return (a);
}
Memo *memos_deleteMemo(Memo *p)
{
    Memo *h = memos()->head;
    Memo *q = h;
    while (q != NULL && (q->next) != p)
    {
        q = q->next;
    }
    q->next = q->next->next;
    free(p);
    return (q->next);
}
Memo *memos_makeTopMemo(Memo *p)
{
    MemoBlock *t;
    int a;
    Memo *h = memos()->head;
    Memo *q;
    q = h;
```

```
    //     h = p;
    //     h->next = q;
    while (q != NULL && (q->next) != p)
    {
        q = q->next;
    }
    q->next = q->next->next;
    q = h;
    memos_insertMemo(h, p);
    t = p->head;
    p->head = h->head;
    h->head = t;
    memos()->head = p;
}

void memos_reset(){
    Memos *m = memos();
    Memo *p = m->head;
    while(p->next!=NULL){
        Memo *pn = p->next->next;
        free(p->next);
        p->next = pn;
    }
    memset(m, 0, sizeof(Memos));
    return;
}

Memos *memos_getList(char *uid){
    Memos *ms = memos();
    struct ffblk ffblk;
    int done;
    memos_reset();
    // 查找第一个匹配的文件
    done = findfirst("data\\*.mem", &ffblk, 0);
    while (!done)
    {
        FILE *fp;
        Memo *m = malloc(sizeof(Memo));
        char path[100];
        char name[10];
        char *dot = strchr(ffblk.ff_name, '.');
        strncpy(name, ffblk.ff_name, dot - ffblk.ff_name);
        name[dot - ffblk.ff_name] = '\0';
        sprintf(path, "data\\%s",ffblk.ff_name);
        if(auth_check(name, uid) == 1){
            fp = fopen(path, "rb");
            if(fp == NULL) continue;
            fread(m, sizeof(Memo), 1, fp);
            fclose(fp);
            strcpy(m->fileName, name);
            memos_add(m);
        }
        done = findnext(&ffblk);
    }

    return ms;
}
Memo *memos_preMemo(Memo *p){
    Memo *head = memos()->head;
```

```
    Memo *q = head;
    while((q->next!=NULL)&&(q->next)!=p)
    {
        q=q->next;
    }
    return(q);
}
int memos_getSum(){
    int s = 0;
    Memo *p = memos()->head;
    while(p != NULL){
        p = p->next;
        s++;
    }
    return s;
}

int memos_getNum(Memo *mb){
    int s = 0;
    Memo *p = memos()->head;
    while(p != NULL && p != mb){
        p = p->next;
        s++;
    }
    return s;
}
```

5.2.20 mfile.c

```
/**
 * @file mfile.c
 * @author wywgwt (2504133124@qq.com), Hibanaw Hu (hibanaw@qq.com)
 * @brief
 * @date 2023-04-06
 *
 * @copyright Copyright (c) 2023
 *
 */

#include "mfile.h"

FILE **memofile_current(){
    static FILE *f;
    return &f;
}

void memofile_write(char* filePath, Memo *m){
    FILE *fp;
    MemoBlock *p;
    if ((fp=fopen(filePath,"wb"))==NULL)
    {
      printf("the file can not be open!");
      getch();
      exit(1);
    }
    p=m->head;
```

```
    if(p==NULL)
    {
        printf("error!");
        getch();
        exit(1);
    }
    else{
        fwrite(m, 1, sizeof(Memo), fp);
        while(p≠NULL)
        {
            fwrite(p, sizeof(MemoBlock), 1, fp);
            p=p→next;
        }
    }
    fflush(fp);
    fclose(fp);
}

MemoBlock *memofile_readBlock()
{
    FILE **fp=memofile_current();
    char t;
    MemoBlock *p;
    p=(MemoBlock *)malloc(sizeof(MemoBlock));
    fread(p, sizeof(MemoBlock), 1, *fp);
    if(feof(*fp)){
        free(p);
        return NULL;
    }
    return p;
 }

Memo memofile_read(char *filePath)
{
    FILE **fp = memofile_current();
    Memo m;
    MemoBlock *p;
    memset(&m, 0, sizeof(m));
     if ((*fp=fopen(filePath,"rb"))==NULL)
    {
        strcpy(m.owner, appData()→currentUser);
        return m;
    }
    fread(&m, sizeof(m), 1, *fp);
    m.head = p = memofile_readBlock();
    while(p)
    {
        p→next = memofile_readBlock();
        p = p→next;
    }
    fclose(*fp);
    return m;
}
```

5.2.21 mouse.c

```c
/**
 * @file mouse.c
 * @author dengshuumin, Hibanaw Hu (hibanaw@qq.com)
 * @brief Mouse under DOS with no global variables.
 * @date 2023-02-26
 * @version 5.1
 *
 * @copyright Copyright (c) 2023
 *
 */

#include "mouse.h"

/**
 * @brief Mouse "object".
 *
 * @return Mouse*
 */
Mouse *mouse(){
    static Mouse m;
    return &m;
}

/**
 * @brief Mouse init.
 *
 */
void mouse_init()
{
    int retcode;
    int xmin,xmax,ymin,ymax,x_max=MAXWIDTH,y_max=MAXHEIGHT;
    union REGS regs;
    Mouse *m = mouse();
    int size=imagesize(0,0,12,19);
    m -> buffer=malloc(size);
    xmin=2;
    xmax=x_max-1;
    ymin=3;
    ymax=y_max-2;
    regs.x.ax=0;
    int86(51,&regs,&regs);
    retcode=regs.x.ax;
    if(retcode==0)
    {
        printf("Mouse or Mouse Driver Obsent,Please Install!");
        delay(5000);
    }
    else
    {
        regs.x.ax=7;
        regs.x.cx=xmin;
        regs.x.dx=xmax;
        int86(51,&regs,&regs);
        regs.x.ax=8;
        regs.x.cx=ymin;
        regs.x.dx=ymax;
        int86(51,&regs,&regs);
    }
    m->style = 0;
```

```
    m->posX=xmax/2;
    m->posY=ymax/2;
    m->_flag=1;
    m->visibility = 1;
    _mouse_saveBackground(m->posX,m->posY);
}

/**
 * @brief Update mouse "object".
 *
 */
void mouse_update()
{
    int xn,yn,buttonsn;
    Mouse *m = mouse();
    int x0=m->posX,y0=m->posY,buttons0=m->click;
    _mouse_read(&xn,&yn,&buttonsn);
    m->posX = xn;
    m->posY = yn;
    m->click = buttonsn;
    if(xn == x0 && yn == y0 && buttonsn == buttons0)
        return;
    if(m->visibility){
        _mouse_clrmous(x0,y0);
        _mouse_saveBackground(xn, yn);
        _mouse_drawmous(xn, yn);
    }

}

void mouse_show(){
    Mouse *m = mouse();
    int x = m->posX, y = m->posY;
    _mouse_saveBackground(x, y);
    _mouse_drawmous(x, y);
}
void mouse_hide(){
    Mouse *m = mouse();
    int x = m->posX, y = m->posY;
    _mouse_clrmous(x, y);
}

/**
 * @brief Whether mouse is clicked in box.
 *
 * @param x1
 * @param y1
 * @param x2
 * @param y2
 * @return int 1 if clicked in box; 2 if in box but not clicked; 0 if
not in box.
 */
int mouse_isClickedInBox(int x1, int y1, int x2, int y2)
{
    int MouseX = mouse()->posX;
    int MouseY = mouse()->posY;
    int press = mouse()->click;

    if(MouseX > x1
```

```
    &&MouseX < x2
    &&MouseY > y1
    &&MouseY < y2
    &&press == 1)
    {
        return 1;
    }

    else if(MouseX > x1
    &&MouseX < x2
    &&MouseY > y1
    &&MouseY < y2
    &&press == 0)
    {
        return 2;
    }

    else if(MouseX > x1
    &&MouseX < x2
    &&MouseY > y1
    &&MouseY < y2
    &&press == 2)
    {
        return 3;
    }
    else if(press){
        return -1;
    }
    else
    {

        return 0;
    }
}

/**
 * @brief Draw mouse at x, y.
 *
 * @param x Position x.
 * @param y Position Y.
 *
 * @private
 */
void _mouse_draw(int x,int y)
{
    switch(mouse()->style)
    {
        case 1:
        {
            setcolor(_WHITE);
            setlinestyle(0,0,1);
            line(x-1,y+9,x-1,y+8);
            line(x,y+7,x,y+11);
            line(x+1,y+6,x+1,y+13);
            line(x+2,y+8,x+2,y+14);
            line(x+3,y-1,x+3,y+15);
            arc(x+4,y-1,0,180,1);
            line(x+4,y-2,x+4,y+15);
            line(x+5,y-1,x+5,y+16);
```

```
            arc(x+6,y+3,0,180,1);
            line(x+6,y+2,x+6,y+16);
            line(x+7,y+3,x+7,y+17);
            arc(x+8,y+5,0,180,1);
            line(x+8,y+4,x+8,y+17);
            line(x+9,y+5,x+9,y+16);
            arc(x+10,y+7,0,180,1);
            line(x+10,y+6,x+10,y+16);
            line(x+11,y+7,x+11,y+13);

            setcolor(_DARKGRAY);
            line(x-1,y+9,x-1,y+8);
            line(x-1,y+8,x+1,y+6);
            line(x+1,y+6,x+3,y+10);
            line(x+3,y+10,x+3,y-1);
            arc(x+4,y-1,0,180,1);
            line(x+5,y-1,x+5,y+5);
            arc(x+6,y+3,0,180,1);
            line(x+7,y+3,x+7,y+7);
            arc(x+8,y+5,0,180,1);
            line(x+9,y+5,x+9,y+9);
            arc(x+10,y+7,0,180,1);
            line(x+11,y+7,x+11,y+13);
            arc(x+7,y+13,-90,0,4);
            line(x+7,y+17,x+3,y+15);
            line(x+3,y+15,x+1,y+13);
            line(x+1,y+13,x-1,y+9);
        }
            break;
        case 2:
        {
            setcolor(_DARKGRAY);
            setlinestyle(0,0,1);
            line(x+1,y-1,x+9,y-1);
            line(x+1,y+15,x+9,y+15);
            line(x+5,y-1,x+5,y+15);
        }
            break;
        case 3:
        {
            setcolor(_WHITE);
            setlinestyle(0,0,1);
            line(x-1,y+7,x+11,y+7);
            line(x+5,y-1,x+5,y+15);
        }
            break;
        default:
        {
            setlinestyle(0,0,1);
            setcolor(_WHITE);
            line(x,y,x,y+13);
            line(x+1,y+1,x+1,y+12);
            line(x+2,y+2,x+2,y+11);
            line(x+3,y+3,x+3,y+10);
            line(x+4,y+4,x+4,y+12);
            line(x+5,y+5,x+5,y+9);
            line(x+5,y+11,x+5,y+14);
            line(x+6,y+6,x+6,y+9);
            line(x+6,y+13,x+6,y+15);
```

```
            line(x+7,y+7,x+7,y+9);
            line(x+8,y+8,x+8,y+9);
            line(x+9,y+9,x+9,y+9);
            setcolor(_DARKGRAY);
            line(x-1,y-1,x-1,y+14);
            line(x-1,y+14,x+3,y+11);
            line(x+3,y+11,x+3,y+12);
            line(x+3,y+12,x+4,y+13);
            line(x+4,y+13,x+4,y+14);
            line(x+4,y+14,x+7,y+17);
            line(x+7,y+17,x+7,y+13);
            line(x+7,y+13,x+6,y+12);
            line(x+6,y+12,x+6,y+11);
            line(x+6,y+11,x+5,y+10);
            line(x+5,y+10,x+11,y+10);
            line(x+11,y+10,x-1,y-2);
        }
        break;
    }
}


/**
 * @brief Read mouse status though DOS interruption.
 *
 * @param nx Mouse position x.
 * @param ny Mouse position y.
 * @param nbuttons Mouse button status.
 *
 * @private
 */
void _mouse_read(int *nx,int *ny,int *nbuttons)
{
    union REGS regs;
    regs.x.ax=3;
    int86(51,&regs,&regs);
    *nx = regs.x.cx;
    *ny = regs.x.dx;
    *nbuttons = regs.x.bx;
}

/**
 * @brief Save mouse background.
 *
 * @param nx Position x.
 * @param ny Position Y.
 *
 * @private
 */
void _mouse_saveBackground(int nx,int ny)
{
    void *buffer = mouse()->buffer;
    getimage(nx-1,ny-2,MIN(nx+11, MAXWIDTH-1),MIN(ny+17,
MAXHEIGHT-1),buffer);
}

/**
 * @brief Clear mouse.
 *
```

```
 * @param nx
 * @param ny
 *
 */
void _mouse_clrmous(int nx, int ny)//锟斤拷锟斤拷斤拷锟斤拷锟斤拷
{
    Mouse *m = mouse();
    if(m->_flag==1)
    {
        setwritemode(XOR_PUT);
        _mouse_draw(nx,ny);
        putimage(nx-1,ny-2,m->buffer,COPY_PUT);
        m->_flag=0;
        setwritemode(COPY_PUT);
    }
}

/**
 * @brief Draw mouse.
 *
 * @param nx
 * @param ny
 *
 */
void _mouse_drawmous(int nx, int ny)
{
    Mouse *m = mouse();
    if(m->_flag==0)
    {
        setwritemode(COPY_PUT);
        _mouse_draw(nx,ny);
        m->_flag=1;
    }
}
```

5.2.22 mset.c

```
/**
 * @file mset.c
 * @author Hibanaw Hu (hibanaw@qq.com), wywgwt (2504133124@qq.com)
 * @brief
 * @date 2023-04-21
 *
 * @copyright Copyright (c) 2023
 *
 */

#include "mset.h"

int mset_page(MemoEditor *me){
    int ol = memo()->level;
    int oa = me->authType;
    while(1){
        int fl = ol, fa = oa;
        Button saveButton = button_new(690, 450, 780, 490, "保存",
button_drawWINUIAccent);
```

```
        Button exitButton = button_new(580, 450, 670, 490, "取消",
button_drawWINUI);
        Text title = text_newDefault("备忘录设置", 486, 80, 0, 0);
        Text levelT = text_newSmall("等级: ", 350, 150);
        Text authT = text_newSmall("权限: ", 350, 300);
        Text levelTS = text_newSmall("低级       中级       高级", 390,
190);
        Text authTS = text_newSmall("私有       共有       白名单", 390,
340);
        MsetRadioBoxArea levelR;
        MsetRadioBoxArea authR;
        int signal = 0;
        mset_newRadioBoxArea(&levelR, 180, ol);
        mset_newRadioBoxArea(&authR, 330, oa);
        mouse_hide();
        setfillstyle(1, hexfffbf0);
        bar(300, 72, 800, 500);
        button_draw(&exitButton);
        button_draw(&saveButton);
        text_display(title);
        text_display(levelT);
        text_display(authT);
        text_display(levelTS);
        text_display(authTS);
        mset_drawRadioBoxArea(&levelR);
        mset_drawRadioBoxArea(&authR);

        mouse_show();
        while(!signal){
            int k = bioskey(1);
            int bs = button_event(&exitButton), sbs =
button_event(&saveButton);
            int lras = 0, aras = 0;
            digitalClock_getTime();
            keybord_eat();
            mouse_update();
            lras = mset_radioBoxAreaEvent(&levelR);
            aras = mset_radioBoxAreaEvent(&authR);
            if(keybord_isESCAPE(k)){
                bioskey(0);
                signal = -1;
            }
            if(bs){
                signal = -1;
            }
            if(sbs){
                signal = 1;
            }
            if(lras){
                fl = lras - 1;
            }
            if(aras){
                fa = aras - 1;
            }
        }
        switch (signal)
        {
        case -1:
            return -1;
```

```
            break;

        case 1:
            memo()->level = fl;
            me->authType = fa;
            return 1;
            break;
        }
    }
}


int mset_newRadioBoxArea(MsetRadioBoxArea *ra, int y, int defualt){
    int i;
    for(i = 0; i < 3; i++){
        ra->radioBox[i] = button_new(360+i*144, y, 380+i*144, y+20, 0,
button_checkboxDraw);
    }
    ra->radioBox[defualt%3].content = 1;
    return 0;
}

void mset_drawRadioBoxArea(MsetRadioBoxArea *sr){
    int i;
    for(i = 0; i < 3; i++){
        button_draw(sr->radioBox+i);
    }
}

int mset_radioBoxAreaEvent(MsetRadioBoxArea *sr){
    int k = 0;
    int lf, nf = -1;
    int i;
    for(i = 0; i < 3; i++){
        if((int)(sr->radioBox[i].content) == 1){
            lf = i;
        }
        if(button_event(sr->radioBox+i) == 1){
            k = 1;
            nf = i;
        }
    }
    if(k && nf != lf){
        sr->radioBox[lf].content = 0;
        sr->radioBox[nf].content = 1;
        mouse_hide();
        mset_drawRadioBoxArea(sr);
        mouse_show();
    }
    return nf+1;
}
```

5.2.23 mshare.c

```
/**
 * @file share.c
```

```
 * @author Hibanaw Hu (hibanaw@qq.com), wywgwt (2504133124@qq.com)
 * @brief
 * @date 2023-04-20
 *
 * @copyright Copyright (c) 2023
 *
 */

#include "mshare.h"


void share_page(char *memoName){
    char strBuffer[30];
    char shareCodeBuffer[20];
    ShareRadioBoxArea sr;
    memset(strBuffer, 0, sizeof(strBuffer));
    share_newRadioBoxArea(&sr, strBuffer);
    while(1){
        Button exitButton = button_new(690, 300, 780, 340, "关闭",
button_drawWINUI);
        Button shareButton = button_new(320, 230, 410, 270, "获取",
button_drawWINUI);
        Text title = text_newDefault("新建共享…", 486, 80, 0, 0);
        Text expire = text_newSmall("有效期: ", 350, 120);
        Text expireTime1 = text_newSmall("永久有效     10分钟     1天",
390, 150);
        Text expireTime2 = text_newSmall("自定义: ", 390, 190);
        int signal = 0;
        int selecetMode = 1;
        expire.font.fontColor = _BLACK;
        mouse_hide();
        setfillstyle(1, hexfffbf0);
        bar(300, 72, 800, 360);
        button_draw(&exitButton);
        button_draw(&shareButton);
        text_display(expireTime1);
        text_display(expireTime2);
        text_display(title);
        text_display(expire);
        share_drawRadioBoxArea(&sr);
        mouse_show();
        while(!signal){
            int k = bioskey(1);
            int bs = button_event(&exitButton), sbs =
button_event(&shareButton);
            int ras = 0;
            digitalClock_getTime();
            keybord_eat();
            mouse_update();
            ras = share_radioBoxAreaEvent(&sr);
            if(ras){
                selecetMode = ras;
            }
            if(keybord_isESCAPE(k)){
                bioskey(0);
                signal = -1;
            }
            if(bs){
                signal = -1;
```

```
            }
            if(sbs){
                time_t nowTime = time(NULL);
                time_t expireTime = -1;
                Text shareCodeText = text_newDefault(shareCodeBuffer,
430, 230, 650, 270);
                char *p;
                switch (selecetMode)
                {
                case 1:
                    expireTime = 0;
                    break;
                case 2:
                    expireTime = nowTime + 10*60;
                    break;
                case 3:
                    expireTime = nowTime + 24*60*60;
                    break;
                case 4:
                    for(p = strBuffer; *p != 0; p++){
                        if(*p < '0' || *p > '9'){
                            Text errorT = text_newSmall(
                                "内容不合法",
                                575, 190
                            );
                            errorT.font.fontColor = _RED;
                            text_display(errorT);
                            break;
                        }
                    }
                    if(strlen(strBuffer) == 0){
                        Text errorT = text_newSmall(
                            "内容不合法",
                            575, 190
                        );
                        errorT.font.fontColor = _RED;
                        text_display(errorT);
                        break;
                    }
                    expireTime = nowTime + 60*atoi(strBuffer);
                    break;
                }
                if(expireTime != -1)
                    if(!share_add(memoName, shareCodeBuffer,
expireTime)){
                        setfillstyle(1, hexfffbf0);
                        bar(430, 230, 650, 270);
                        text_display(shareCodeText);
                    }
            }
        }
        switch (signal)
        {
        case -1:
            return;
            break;

        default:
            break;
```

```
        }
    }
}

int share_add(char *memoName, char *shareCodeBuffer, time_t expireTime){
    Share sd;
    int i;
    FILE *fp;
    memset(&sd, 0, sizeof(Share));
    strcpy(sd.memoName, memoName);
    srand(time(NULL));
    for(i = 0; i < 6; i++){
        int r = rand()%62;
        if(r < 26){
            sd.code[i] = 'a'+r;
        }
        else if(r < 52){
            sd.code[i] = 'A'+r-26;
        }
        else{
            sd.code[i] = '0'+r-52;
        }
    }
    sd.code[i] = 0;
    sd.expiryTime = expireTime;
    fp = fopen("data/share.dat", "ab+");
    if(fp == NULL){
        return 1;
    }
    fwrite(&sd, sizeof(Share), 1, fp);
    fclose(fp);
    strcpy(shareCodeBuffer, sd.code);
    return 0;
}

int share_determine(char *shareCode, char *uid){
    FILE *fp;
    fp = fopen("data/share.dat", "rb");
    if(fp == NULL){
        return 1;
    }
    while(!feof(fp)){
        Share s;
        fread(&s, sizeof(Share), 1, fp);
        if(!strcmp(s.code, shareCode)){
            int t = auth_addWhiteList(s.memoName, uid);
            if(t == 0){
                fclose(fp);
                return 0;
            }
            else{
                fclose(fp);
                return 1;
            }
        }
    }
    fclose(fp);
    return 1;
```

```
}

share_newRadioBoxArea(ShareRadioBoxArea *ra, char *buffer){
    int i;
    for(i = 0; i < 3; i++){
        ra->radioBox[i] = button_new(360+i*144, 150, 380+i*144, 170, 0,
button_checkboxDraw);
    }
    ra->radioBox[3] = button_new(360, 190, 380, 210, 0,
button_checkboxDraw);
    ra->radioBox[0].content = 1;
    buffer[0] = 0;
    ra->textInput = textinput_newDefault("自定义（分钟）", 460, 185, 660,
215, buffer);
    ra->textInput.textbox.maxLength = 4;
    ra->textInput.textbox.font.fontSize = 16;
    ra->textInput.textbox.posY1 = 190;
    ra->textInput.textbox.posY2 = 210;
    return 0;
}

void share_drawRadioBoxArea(ShareRadioBoxArea *sr){
    int i;
    for(i = 0; i < 4; i++){
        button_draw(sr->radioBox+i);
    }
    textinput_draw(&sr->textInput);
}

int share_radioBoxAreaEvent(ShareRadioBoxArea *sr){
    int k = 0;
    int lf, nf = -1;
    int i;
    int tls = sr->textInput.textbox.status;
    for(i = 0; i < 4; i++){
        if((int)(sr->radioBox[i].content) == 1){
            lf = i;
        }
        if(button_event(sr->radioBox+i) == 1){
            k = 1;
            nf = i;
        }
    }
    textinput_event(&sr->textInput);
    if(tls != TextboxSelected && sr->textInput.textbox.status ==
TextboxSelected){
        k = 1;
        nf = 3;
    }
    if(k && nf != lf){
        sr->radioBox[lf].content = 0;
        sr->radioBox[nf].content = 1;
        mouse_hide();
        share_drawRadioBoxArea(sr);
        mouse_show();
    }
    return nf+1;
}
```

5.2.24 router.c

```
/**
 * @file scroll.c
 * @author Hibanaw Hu (hibanaw@qq.com), wywgwt (2504133124@qq.com)
 * @brief
 * @date 2023-04-11
 *
 * @copyright Copyright (c) 2023
 *
 */

#include "scroll.h"

ScrollBar scrollBar_new(int x, int y, int h){
    ScrollBar sb;
    memset(&sb, 0, sizeof(sb));
    sb.posX = x;
    sb.posY = y;
    sb.status = 0;
    sb.width = 15;
    sb.height = h;
    sb.length = 0;
    sb.inScreenItem = sb.sumItem = 0;
    sb.barPosY1 = sb.barPosY2 = 0;
    sb.mouseLastPosY = 0;
    sb.bgColor = _WHITE;
    return sb;
}

int scrollBar_event(ScrollBar *sb){
    int dy;
    int h;
    int ls = sb->status;
    int y1, y2, l;
    if(ls != ScrollBarDefault && mouse_isClickedInBox(sb->posX, sb-
>posY, sb->posX+sb->width, sb->posY + sb->height) <= 0){
        sb->status = ScrollBarDefault;
        mouse()->style = CURSORPOINTER;
        mouse_hide();
        scrollBar_draw(sb);
        mouse_show();
    }
    if(ls != ScrollBarFocused && mouse_isClickedInBox(sb->posX, sb-
>posY, sb->posX+sb->width, sb->posY + sb->height) == 2){
        sb->status = ScrollBarFocused;
        mouse()->style = CURSORPOINTER;
        mouse_hide();
        scrollBar_draw(sb);
        mouse_show();
    }
    if(ls != ScrollBarSelected && mouse_isClickedInBox(sb->posX, sb-
>posY, sb->posX+sb->width, sb->posY + sb->height) == 1){
        sb->status = ScrollBarSelected;
        mouse()->style = CURSORPOINTER;
        sb->mouseLastPosY = mouse()->posY;
        mouse_hide();
```

```
        scrollBar_draw(sb);
        mouse_show();
    }
    if(!sb->sumItem){
        return 0;
    }
    h = sb->height * sb->inScreenItem / sb->sumItem;
    if(sb->status == ScrollBarSelected){
        dy = mouse()->posY - sb->mouseLastPosY;
        sb->barPosY1 += dy;
        if(sb->barPosY1 > sb->posY+sb->height-h){
            sb->barPosY1 = sb->posY+sb->height-h;
        }
        if(sb->barPosY1 < sb->posY){
            sb->barPosY1 = sb->posY;
        }
        sb->barPosY2 = sb->barPosY1 + h;
        if(ABS(sb->lastDrawBarPosY1 - sb->barPosY1) > 5 || ABS(sb-
>lastDrawBarPosY2 - sb->barPosY2) > 5){
            mouse_hide();
            scrollBar_draw(sb);
            mouse_show();
            sb->lastDrawBarPosY1 = sb->barPosY1;
            sb->lastDrawBarPosY2 = sb->barPosY2;
        }
        if((sb->barPosY1-sb->posY) / (sb->height/sb->sumItem) < sb-
>ithItem){
            return -1;
        }
        if((sb->barPosY1-sb->posY) / (sb->height/sb->sumItem) > sb-
>ithItem){
            return 1;
        }
        sb->mouseLastPosY = mouse()->posY;
    }
    else{
        sb->barPosY1 = sb->posY + sb->height * sb->ithItem / sb-
>sumItem;
        sb->barPosY2 = sb->barPosY1 + h;
        return 0;
    }

}

void scrollBar_draw(ScrollBar *sb){
    int x;
    setfillstyle(1, sb->bgColor);
    bar(sb->posX, sb->posY, sb->posX+sb->width, sb->posY+sb->height);
    if(sb->inScreenItem == sb->sumItem){
        return;
    }
    x = sb->posX + 5;
    switch (sb->status)
    {
    case ScrollBarDefault:
        setfillstyle(1, hex808080);
        bar(x+4, sb->barPosY1, x+5, sb->barPosY2);
        break;
    case ScrollBarFocused:
```

```
        setfillstyle(1, hex555f55);
        setcolor(hex555f55);
        pieslice(x+3, sb->barPosY1+3, 0, 180, 2);
        pieslice(x+3, sb->barPosY2-3, 180, 360, 2);
        bar(x+1, sb->barPosY1+3, x+5, sb->barPosY2-3);
        break;
    case ScrollBarSelected:
        setfillstyle(1, hex555f55);
        setcolor(hex555f55);
        pieslice(x+3, sb->barPosY1+3, 0, 180, 2);
        pieslice(x+3, sb->barPosY2-3, 180, 360, 2);
        bar(x+1, sb->barPosY1+3, x+5, sb->barPosY2-3);
        break;
    }
    return;
}
```

5.2.25 scroll.c

```
/**
 * @file scroll.c
 * @author Hibanaw Hu (hibanaw@qq.com)
 * @brief
 * @date 2023-04-11
 *
 * @copyright Copyright (c) 2023
 *
 */

#include "scroll.h"

ScrollBar scrollBar_new(int x, int y, int h){
    ScrollBar sb;
    memset(&sb, 0, sizeof(sb));
    sb.posX = x;
    sb.posY = y;
    sb.status = 0;
    sb.width = 15;
    sb.height = h;
    sb.length = 0;
    sb.inScreenItem = sb.sumItem = 0;
    sb.barPosY1 = sb.barPosY2 = 0;
    sb.mouseLastPosY = 0;
    sb.bgColor = _WHITE;
    return sb;
}

int scrollBar_event(ScrollBar *sb){
    int dy;
    int h;
    int ls = sb->status;
    int y1, y2, l;
    if(ls != ScrollBarDefault && mouse_isClickedInBox(sb->posX, sb-
>posY, sb->posX+sb->width, sb->posY + sb->height) <= 0){
        sb->status = ScrollBarDefault;
        mouse()->style = CURSORPOINTER;
```

```
        mouse_hide();
        scrollBar_draw(sb);
        mouse_show();
    }
    if(ls != ScrollBarFocused && mouse_isClickedInBox(sb->posX, sb-
>posY, sb->posX+sb->width, sb->posY + sb->height) == 2){
        sb->status = ScrollBarFocused;
        mouse()->style = CURSORPOINTER;
        mouse_hide();
        scrollBar_draw(sb);
        mouse_show();
    }
    if(ls != ScrollBarSelected && mouse_isClickedInBox(sb->posX, sb-
>posY, sb->posX+sb->width, sb->posY + sb->height) == 1){
        sb->status = ScrollBarSelected;
        mouse()->style = CURSORPOINTER;
        sb->mouseLastPosY = mouse()->posY;
        mouse_hide();
        scrollBar_draw(sb);
        mouse_show();
    }
    if(!sb->sumItem){
        return 0;
    }
    h = sb->height * sb->inScreenItem / sb->sumItem;
    if(sb->status == ScrollBarSelected){
        dy = mouse()->posY - sb->mouseLastPosY;
        sb->barPosY1 += dy;
        if(sb->barPosY1 > sb->posY+sb->height-h){
            sb->barPosY1 = sb->posY+sb->height-h;
        }
        if(sb->barPosY1 < sb->posY){
            sb->barPosY1 = sb->posY;
        }
        sb->barPosY2 = sb->barPosY1 + h;
        if(ABS(sb->lastDrawBarPosY1 - sb->barPosY1) > 5 || ABS(sb-
>lastDrawBarPosY2 - sb->barPosY2) > 5){
            mouse_hide();
            scrollBar_draw(sb);
            mouse_show();
            sb->lastDrawBarPosY1 = sb->barPosY1;
            sb->lastDrawBarPosY2 = sb->barPosY2;
        }
        if((sb->barPosY1-sb->posY) / (sb->height/sb->sumItem) < sb-
>ithItem){
            return -1;
        }
        if((sb->barPosY1-sb->posY) / (sb->height/sb->sumItem) > sb-
>ithItem){
            return 1;
        }
        sb->mouseLastPosY = mouse()->posY;
    }
    else{
        sb->barPosY1 = sb->posY + sb->height * sb->ithItem / sb-
>sumItem;
        sb->barPosY2 = sb->barPosY1 + h;
        return 0;
    }
```

```
}

void scrollBar_draw(ScrollBar *sb){
    int x;
    setfillstyle(1, sb->bgColor);
    bar(sb->posX, sb->posY, sb->posX+sb->width, sb->posY+sb->height);
    if(sb->inScreenItem == sb->sumItem){
        return;
    }
    x = sb->posX + 5;
    switch (sb->status)
    {
    case ScrollBarDefault:
        setfillstyle(1, hex808080);
        bar(x+4, sb->barPosY1, x+5, sb->barPosY2);
        break;
    case ScrollBarFocused:
        setfillstyle(1, hex555f55);
        setcolor(hex555f55);
        pieslice(x+3, sb->barPosY1+3, 0, 180, 2);
        pieslice(x+3, sb->barPosY2-3, 180, 360, 2);
        bar(x+1, sb->barPosY1+3, x+5, sb->barPosY2-3);
        break;
    case ScrollBarSelected:
        setfillstyle(1, hex555f55);
        setcolor(hex555f55);
        pieslice(x+3, sb->barPosY1+3, 0, 180, 2);
        pieslice(x+3, sb->barPosY2-3, 180, 360, 2);
        bar(x+1, sb->barPosY1+3, x+5, sb->barPosY2-3);
        break;
    }
    return;
}
```

5.2.26 svga.c

```
/**
 * @file svga.c
 * @author Hibanaw Hu (hibanaw@qq.com)
 * @brief
 * @date 2023-02-12
 *
 * @copyright Copyright (c) 2023
 *
 */

#include "svga.h"

int huge detectSVGA256(void) //设置驱动必须的函数
{
    return SVGA1024x768x256; //高分辨率实现
}

int initsvga256(void) //初始化函数
{
```

```
    int G_driver = DETECT, G_model;
    char Gr_error; //错误信息储存
    installuserdriver("SVGA256", detectSVGA256);
    initgraph(&G_driver, &G_model, "C:\\BORLANDC\\BGI");
    Gr_error = graphresult();
    if (Gr_error != grOk)
    {
        printf("error when open svga, id: %d\n",  Gr_error);
        return 1;
    }
    return 0;
}

int closesvga256(void) //关闭图像模式函数
{
    char Gr_error; //错误信息储存
    closegraph();
    Gr_error = graphresult();
    if (Gr_error != grOk)
    {
        return 1;
    }
    return 0;
}

void setvgapalette256(DacPalette256 *PalBuf)
{
  struct REGPACK reg;

  reg.r_ax = 0x1012;
  reg.r_bx = 0;
  reg.r_cx = 256;
  reg.r_es = FP_SEG(PalBuf);
  reg.r_dx = FP_OFF(PalBuf);
  intr(0x10,&reg);
}

void useRGB256Colors(){
    unsigned long hexColors[]={
        0x000000,
        0x800000,
        0x008000,
        0x808000,
        0x000080,
        0x800080,
        0x008080,
        0xc0c0c0,
        0xc0dcc0,
        0xa6caf0,
        0x2a3faa,
        0x2a3fff,
        0x2a5f00,
        0x2a5f55,
        0x2a5faa,
        0x2a5fff,
        0x2a7f00,
        0x2a7f55,
        0x2a7faa,
        0x2a7fff,
```

```
        0x2a9f00,
        0x2a9f55,
        0x2a9faa,
        0x2a9fff,
        0x2abf00,
        0x2abf55,
        0x2abfaa,
        0x2abfff,
        0x2adf00,
        0x2adf55,
        0x2adfaa,
        0x2adfff,
        0x2aff00,
        0x2aff55,
        0x2affaa,
        0x2affff,
        0x550000,
        0x550055,
        0x5500aa,
        0x5500ff,
        0x551f00,
        0x551f55,
        0x551faa,
        0x551fff,
        0x553f00,
        0x553f55,
        0x553faa,
        0x553fff,
        0x555f00,
        0x555f55,
        0x555faa,
        0x555fff,
        0x557f00,
        0x557f55,
        0x557faa,
        0x557fff,
        0x559f00,
        0x559f55,
        0x559faa,
        0x559fff,
        0x55bf00,
        0x55bf55,
        0x55bfaa,
        0x55bfff,
        0x55df00,
        0x55df55,
        0x55dfaa,
        0x55dfff,
        0x55ff00,
        0x55ff55,
        0x55ffaa,
        0x55ffff,
        0x7f0000,
        0x7f0055,
        0x7f00aa,
        0x7f00ff,
        0x7f1f00,
        0x7f1f55,
        0x7f1faa,
```

```
        0x7f1fff,
        0x7f3f00,
        0x7f3f55,
        0x7f3faa,
        0x7f3fff,
        0x7f5f00,
        0x7f5f55,
        0x7f5faa,
        0x7f5fff,
        0x7f7f00,
        0x7f7f55,
        0x7f7faa,
        0x7f7fff,
        0x7f9f00,
        0x7f9f55,
        0x7f9faa,
        0x7f9fff,
        0x7fbf00,
        0x7fbf55,
        0x7fbfaa,
        0x7fbfff,
        0x7fdf00,
        0x7fdf55,
        0x7fdfaa,
        0x7fdfff,
        0x7fff00,
        0x7fff55,
        0x7fffaa,
        0x7fffff,
        0xaa0000,
        0xaa0055,
        0xaa00aa,
        0xaa00ff,
        0xaa1f00,
        0xaa1f55,
        0xaa1faa,
        0xaa1fff,
        0xaa3f00,
        0xaa3f55,
        0xaa3faa,
        0xaa3fff,
        0xaa5f00,
        0xaa5f55,
        0xaa5faa,
        0xaa5fff,
        0xaa7f00,
        0xaa7f55,
        0xaa7faa,
        0xaa7fff,
        0xaa9f00,
        0xaa9f55,
        0xaa9faa,
        0xaa9fff,
        0xaabf00,
        0xaabf55,
        0xaabfaa,
        0xaabfff,
        0xaadf00,
        0xaadf55,
```

```
        0xaadfaa,
        0xaadfff,
        0xaaff00,
        0xaaff55,
        0xaaffaa,
        0xaaffff,
        0xd40000,
        0xd40055,
        0xd400aa,
        0xd400ff,
        0xd41f00,
        0xd41f55,
        0xd41faa,
        0xd41fff,
        0xd43f00,
        0xd43f55,
        0xd43faa,
        0xd43fff,
        0xd45f00,
        0xd45f55,
        0xd45faa,
        0xd45fff,
        0xd47f00,
        0xd47f55,
        0xd47faa,
        0xd47fff,
        0xd49f00,
        0xd49f55,
        0xd49faa,
        0xd49fff,
        0xd4bf00,
        0xd4bf55,
        0xd4bfaa,
        0xd4bfff,
        0xd4df00,
        0xd4df55,
        0xd4dfaa,
        0xd4dfff,
        0xd4ff00,
        0xd4ff55,
        0xd4ffaa,
        0xd4ffff,
        0xff0055,
        0xff00aa,
        0xff1f00,
        0xff1f55,
        0xff1faa,
        0xff1fff,
        0xff3f00,
        0xff3f55,
        0xff3faa,
        0xff3fff,
        0xff5f00,
        0xff5f55,
        0xff5faa,
        0xff5fff,
        0xff7f00,
        0xff7f55,
        0xff7faa,
```

```
        0xff7fff,
        0xff9f00,
        0xff9f55,
        0xff9faa,
        0xff9fff,
        0xffbf00,
        0xffbf55,
        0xffbfaa,
        0xffbfff,
        0xffdf00,
        0xffdf55,
        0xffdfaa,
        0xffdfff,
        0xffff55,
        0xffffaa,
        0xccccff,
        0xffccff,
        0x33ffff,
        0x66ffff,
        0x99ffff,
        0xccffff,
        0x007f00,
        0x007f55,
        0x007faa,
        0x007fff,
        0x009f00,
        0x009f55,
        0x009faa,
        0x009fff,
        0x00bf00,
        0x00bf55,
        0x00bfaa,
        0x00bfff,
        0x00df00,
        0x00df55,
        0x00dfaa,
        0x00dfff,
        0x00ff55,
        0x00ffaa,
        0x2a0000,
        0x2a0055,
        0x2a00aa,
        0x2a00ff,
        0x2a1f00,
        0x2a1f55,
        0x2a1faa,
        0x2a1fff,
        0x2a3f00,
        0x2a3f55,
        0xfffbf0,
        0xa0a0a4,
        0x808080,
        0xff0000,
        0x00ff00,
        0xffff00,
        0x0000ff,
        0xff00ff,
        0x00ffff,
        0xffffff
```

```
    };
    DacPalette256 win_color;
    int i;
    for (i = 0; i < 256; i++)
    {
        win_color[i][0] = (hexColors[i] >> 16) / 4;
        win_color[i][1] = ((hexColors[i] & 0x00ff00) >> 8) / 4;
        win_color[i][2] = (hexColors[i] & 0x0000ff) / 4;
    }
    setvgapalette256(&win_color);
    return;
}
```

5.2.27 text.c

```
/**
 * @file text.c
 * @author Hibanaw Hu (hibanaw@qq.com)
 * @brief
 * @date 2023-03-08
 *
 * @copyright Copyright (c) 2023
 *
 */

#include "text.h"

void text_display(Text t){
    int x = t.posX, y = t.posY,
        w = t.width, h = t.hight;
    int c = t.font.fontColor;
    int tx = x, ty = y;
    int fs = t.font.fontSize;
    int d = t.font.fontSize + t.font.spacing;
    int rd = t.font.rowSpacing;
    char *p = t.content;
    debug(DEBUG, "Start display text");
    while(*p != '\0'){
        char s[3] = {0, 0, 0};
        if((unsigned)*p < 128){
            //ascii
            s[0] = *p;
            p++;
            if(w && tx + fs > x+w){
                if(h && ty + fs + rd > y + h){
                    return;
                }
                tx = x;
                ty += fs + rd;
            }
            setcolor(c);
            settextstyle(SANS_SERIF_FONT, HORIZ_DIR, fs/9);
            settextjustify(LEFT_TEXT,2);
            setwritemode(COPY_PUT);
            switch (fs)
            {
```

```
            case 16:
                outtextxy(tx+fs/4, ty-6, s);
                break;
            case 24:
                outtextxy(tx+fs/4, ty-2, s);
                break;
            default:
                outtextxy(tx+fs/4, ty-fs/7, s);
                break;
            }
            tx+=d;
        }
        if((unsigned)*p > 0xA0){
            //gb2312
            s[0] = *p;
            p++;
            s[1] = *p;
            p++;
            if(w && tx+fs > x+w){
                if(h && ty + fs + rd > y + h){
                    return;
                }
                tx = x;
                ty += fs + rd;
            }
            hz_puthz(s, tx, ty, fs, d, c);
            tx+=d;
        }
        if((unsigned)*p == 0xA0){
            //TODO: emoji
            s[0] = *p;
            p++;
            s[1] = *p;
            p++;
            if(w && tx+d > x+w){
                if(h && ty + fs + rd > y){
                    return;
                }
                tx = x;
                ty += rd;
            }
            hz_puthz("表", tx, ty, fs, d, c);
            tx+=d;
        }
    }
}

int text_getLength(char *s){
    char *p = s;
    int l = 0;
    while(*p != 0){
        if((unsigned char) *p < 128){
            p++;
            l++;
        }
        else{
            p+=2;
            l++;
        }
```

```
    }
    return l;
}

int text_getHeight(Text t){
    int l = text_getLength(t.content);
    int w = t.width;
    int rm = (w % (t.font.fontSize + t.font.spacing) / t.font.fontSize)
+ w / (t.font.fontSize + t.font.spacing);
    int c = CEILING((float) l / rm);
    if(!strlen(t.content)) c = 1;
    return c*(t.font.fontSize + t.font.rowSpacing);
}

char *text_getNthChar(char *s, int n){
    char *p = s;
    int i;
    for(i = 0; i < n; i++){
        if(*p >= 0) p++;
        else p+=2;
    }
    return p;
}

Text text_newDefault(char *s, int x1, int y1, int x2, int y2){
    Text t;
    t.content = s;
    t.posX = x1;
    t.posY = y1;
    if(x2!=0)
        t.width = x2-x1;
    else t.width = 0;
    if(y2!=0)
        t.hight = y2-y1;
    else t.hight = 0;
    t.font.fontSize = 24;
    t.font.fontColor = _BLACK;
    t.font.spacing = 2;
    t.font.rowSpacing = 0;
    return t;
}

Text text_newSmall(char *s, int x, int y){
    Text t;
    t.content = s;
    t.posX = x;
    t.posY = y;
    t.width = 0;
    t.hight = 0;
    t.font.fontSize = 16;
    t.font.spacing = 0;
    t.font.fontColor = _DARKGRAY;
    return t;
}
```

5.2.28 textbox.c

```c
/**
 * @file textbox.c
 * @author Hibanaw Hu (hibanaw@qq.com)
 * @brief
 * @date 2023-03-08
 *
 * @copyright Copyright (c) 2023
 *
 */
#include "textbox.h"

void textbox_draw(Textbox *tb)
{
    int x1 = tb->posX1, y1 = tb->posY1,
        x2 = tb->posX2, y2 = tb->posY2;
    setfillstyle(1, tb->bgColor);
    bar(x1, y1, x2, y2);
    if(strlen(tb->content)){
        Text t;
        char s[200];
        memset(s, tb->hint, sizeof(s));
        if(tb->type == TextboxPassword){
            int l = text_getLength(tb->content);
            s[l] = 0;
            t = text_newDefault(s, x1, y1, x2, y2);
        }
        else{
            t = text_newDefault(tb->content, x1, y1, x2, y2);
        }
        t.font = tb->font;
        text_display(t);
    }
    else{
        Text t = text_newDefault(tb->defaultContent, x1, y1, x2, y2);
        t.font = tb->font;
        t.font.fontColor = _GRAY;
        text_display(t);
    }
    if(tb->status == TextboxSelected){
        int c, r;
        int lx , ly;
        int loc = tb->cursorLocation;
        int w = tb->posX2 - tb->posX1;
        lx = textbox_getCursorPositionX(*tb);
        ly = textbox_getCursorPositionY(*tb);
        setcolor(tb->cursorStatus ? tb->font.fontColor : tb->bgColor);
        setlinestyle(0, 1, 5);
        line(lx, ly+2, lx, ly+tb->font.fontSize-2);
    }
}

int textbox_event(Textbox *tb)
{
    int otbs = tb->status;
    char *s = tb->content;
    Mouse *m = mouse();
    int mx = m->posX, my = m->posY;
    int change = 0;
    int r = 0;
```

```
    textbox_determinState(tb);
    if(otbs != tb->status){
        mouse_hide();
        textbox_draw(tb);
        mouse_show();
    }
    if(tb->type == TextboxPassword){
        if(tb->status == TextboxSelected)
            ime()->pw = 1;
        else
            ime()->pw = 0;
    }
    //cursor blink
    if (tb->status == TextboxSelected){
        clock_t lt, nt, dt;
        nt = clock();
        lt = tb->cursorLastBlink;
        dt = nt - lt;
        if(dt / CLK_TCK >= 0.6){
            tb->cursorLastBlink = nt;
            tb->cursorStatus = !tb->cursorStatus;
            mouse_hide();
            textbox_draw(tb);
            mouse_show();
        }
    }
    //cursor position key
    if (tb->status == TextboxSelected){
        int k;
        if(k = bioskey(1)){
            switch(k){
                case KEYLEFT:
                    bioskey(0);
                    tb->cursorLocation--;
                    tb->cursorLocation = MAX(0, tb->cursorLocation);
                    break;
                case KEYRIGHT:
                    bioskey(0);
                    tb->cursorLocation++;
                    tb->cursorLocation = MIN(tb->cursorLocation,
text_getLength(tb->content));
                    break;
                case KEYHOME:
                    bioskey(0);
                    tb->cursorLocation = 0;
                    break;
                case KEYEND:
                    bioskey(0);
                    tb->cursorLocation = text_getLength(tb->content);
                    break;
            }
            tb->cursorLastBlink = 0;
            tb->cursorStatus = 0;
            mouse_hide();
            textbox_draw(tb);
            mouse_show();
        }
    }
    if (tb->status == TextboxSelected){
```

```
        int k = bioskey(1);
        if(k == KEYENTER){
            bioskey(0);
            return 1;
        }
    }
    // input
    if (tb->status == TextboxSelected)
    {
        char olds[500], inss[20], olds1[500], olds2[500], finals[500];
        int cl = tb->cursorLocation;
        int i;
        int is;
        int clp;
        memset(inss, 0, sizeof(inss));
        memset(olds1, 0, sizeof(olds1));
        memset(olds2, 0, sizeof(olds2));
        memset(finals, 0, sizeof(finals));
        is = ime_input(inss, textbox_getCursorPositionX(*tb) + 20,
textbox_getCursorPositionY(*tb) + 30);
        if(is == 0){
            return 0;
        }
        debug(DEBUG, inss);
        strcpy(olds, tb->content);
        clp = text_getNthChar(olds, cl) - olds;
        for(i = 0; i < strlen(olds); i++){
            if(i < clp ){
                olds1[i] = olds[i];
            }
            else{
                olds2[i-clp] = olds[i];
            }
        }
        if(text_getLength(olds) + text_getLength(inss) > tb->maxLength){
            return 0;
        }
        if(is == -1){
            if(cl != 0){
                *text_getNthChar(olds1, MAX(0, cl-1)) = 0;
            }
            else{
                r = 1;
            }
            strcat(finals, olds1);
            strcat(finals, olds2);
        }
        else{
            strcat(finals, olds1);
            strcat(finals, inss);
            strcat(finals, olds2);
        }
        if(strcmp(olds, finals)){
            strcpy(tb->content, finals);
            tb->cursorLocation += is;
            mouse_hide();
            textbox_draw(tb);
            mouse_show();
        }
```

```
    }
    if(r) return -1;
    return 0;
}

void textbox_determinState(Textbox *tb)
{
    Textbox ltb = *tb;
    Mouse *m = mouse();
    int mx = m->posX, my = m->posY, mc = m->click;

    if (ltb.mstatus == TextboxMouseDefault && mouse_isClickedInBox(tb-
>posX1, tb->posY1, tb->posX2, tb->posY2) > 0)
    {
        tb->mstatus = TextboxMouseFocused;
        m->style = CURSORTEXT;
    }
    if (ltb.mstatus != TextboxMouseDefault && mouse_isClickedInBox(tb-
>posX1, tb->posY1, tb->posX2, tb->posY2) == 0)
    {
        tb->mstatus = TextboxMouseDefault;
        m->style = CURSORPOINTER;
    }
    if(ltb.mstatus == TextboxMouseClicked && mouse_isClickedInBox(tb-
>posX1, tb->posY1, tb->posX2, tb->posY2) != 1){
        tb->mstatus = TextboxMouseFocused;
    }
    // 闆﹀劍锟藉椢鍣烽悳鐗哥礆闔団偍鐐曚焦鑰奸娄妯肩椽鏃堝垵娅捏崗澶嬬娑娴ｅ楽鏀
    if (mouse_isClickedInBox(tb->posX1, tb->posY1, tb->posX2, tb->posY2)
== 1 && tb->mstatus != TextboxMouseClicked)
    {
        int x = tb->posX1, y = tb -> posY1;
        int w = tb->posX2 - tb->posX1, h = tb->posY2 - tb->posY1;
        int row = CEILING(1.0 * (my - y) / (tb->font.fontSize + tb-
>font.rowSpacing))-1;
        int colum = (int)(1.0 * (mx - x) / (tb->font.fontSize + tb-
>font.spacing)*2.0) - ((mx - x) / (tb->font.fontSize + tb-
>font.spacing));
        int words_per_line = (w % (ltb.font.fontSize +
ltb.font.spacing) / ltb.font.fontSize) + w / (ltb.font.fontSize +
ltb.font.spacing);
        tb->status = TextboxSelected;
        tb->cursorLocation = MIN(text_getLength(tb->content), row *
words_per_line + colum);
        tb->cursorLastBlink = 0;
        tb->cursorStatus = 0;
        tb->mstatus = TextboxMouseClicked;
    }
    if (tb->status == TextboxSelected &&mouse_isClickedInBox(tb->posX1,
tb->posY1, tb->posX2, tb->posY2) == -1){
        tb->status = TextboxDefault;
    }
}

Textbox textbox_newDefault(char *ds, int x1, int y1, int x2, int y2,
char *buffer){
    Textbox tb;
    memset(&tb, 0, sizeof(tb));
    tb.status = TextboxDefault;
```

```
    tb.mstatus = TextboxMouseDefault;
    tb.cursorStatus = 0;
    tb.defaultContent = ds;
    tb.content = buffer;
    tb.maxLength = 75;
    tb.font.fontSize = 24;
    tb.font.fontColor = _BLACK,
    tb.font.spacing = 1;
    tb.font.rowSpacing = 2;
    tb.posX1 = x1;
    tb.posX2 = x2;
    tb.posY1 = y1;
    tb.posY2 = y2;
    tb.hint = '*';
    tb.type = TextboxText;
    tb.bgColor = _WHITE;
    return tb;
}

int textbox_getCursorPositionX(Textbox t){
    int p = t.cursorLocation;
    int w = t.posX2 - t.posX1;
    int rm = (w % (t.font.fontSize + t.font.spacing) / t.font.fontSize)
+ w / (t.font.fontSize + t.font.spacing);
    int r = p % rm ? p % rm : rm;
    if(p==0) r = 0;
    return t.posX1 + r * (t.font.fontSize + t.font.spacing);
}

int textbox_getCursorPositionY(Textbox t){
    int p = t.cursorLocation;
    int w = t.posX2 - t.posX1;
    int rm = (w % (t.font.fontSize + t.font.spacing) / t.font.fontSize)
+ w / (t.font.fontSize + t.font.spacing);
    int c = CEILING((float) p / rm)-1;
    if(p==0) c = 0;
    return t.posY1 +  c*(t.font.fontSize + t.font.rowSpacing);
}

Text textbox_convert2text(Textbox tb){
    Text t;
    t.hight = tb.posY2 - tb.posY1;
    t.width = tb.posX2 - tb.posX1;
    t.font = tb.font;
    t.content = tb.content;
    return t;
}
```

5.2.29 textipt.c

```
/**
 * @file textinput.c
 * @author Hibanaw Hu (hibanaw@qq.com)
 * @brief
 * @date 2023-04-01
 *
```

```
 * @copyright Copyright (c) 2023
 *
 */

#include "textipt.h"

void textinput_draw(TextInput *tb){
    tb->draw(tb);
}

void textinput_drawDefault(TextInput *ti)
{
    int x1 = ti->posX1, y1 = ti->posY1,
        x2 = ti->posX2, y2 = ti->posY2;
    int c0 = _GRAY, c1 = _WHITE;

    setfillstyle(1, c1);
    setcolor(c1);
    bar(x1+5, y1, x2-5, y2);
    bar(x1, y1+5, x2, y2-5);
    pieslice(x1 + 5, y1 + 5, 90, 180, 5);
    pieslice(x1 + 5, y2 - 5, 180, 270, 5);
    pieslice(x2 - 5, y1 + 5, 0, 90, 5);
    pieslice(x2 - 5, y2 - 5, 270, 360, 5);
    if(ti->textbox.status == TextboxSelected){
        setcolor(hexff9f00);
        setfillstyle(1, hexff9f00);
        pieslice(x1 + 4, y2 - 4, 180, 270, 4);
        pieslice(x2 - 4, y2 - 4, 270, 360, 4);
        bar(x1 + 4, y2 - 4, x2 - 4, y2);
    }
    setcolor(c0);
    setlinestyle(0, 1, 2);
    line(x1 + 5, y1, x2 - 5, y1);
    line(x1 + 5, y2, x2 - 5, y2);
    line(x1, y1 + 5, x1, y2 - 5);
    line(x2, y1 + 5, x2, y2 - 3);
    arc(x1 + 5, y1 + 5, 90, 180, 5);
    arc(x1 + 5, y2 - 5, 180, 270, 5);
    arc(x2 - 5, y1 + 5, 0, 90, 5);
    arc(x2 - 5, y2 - 5, 270, 360, 5);
    textbox_draw(&ti->textbox);
}

void textinput_drawTransparent(TextInput *ti){
    int x1 = ti->posX1, y1 = ti->posY1,
        x2 = ti->posX2, y2 = ti->posY2;
    setfillstyle(1, ti->textbox.bgColor);
    bar(x1, y1, x2, y2);
    textbox_draw(&ti->textbox);
}

int textinput_event(TextInput *ti){
    int k;
    int tl, l, ltl;
    int ls = ti->textbox.status;
    if(ti->align == 1){
        ltl = text_getLength(ti->textbox.content);
        ltl = (ltl == 0)?text_getLength(ti->textbox.defaultContent) :
```

```
ltl;
        l = ltl*(ti->textbox.font.fontSize + ti->textbox.font.spacing);
        if(ti->textbox.posX1 != ti->posX1 + (ti->posX2-ti->posX1 - l)/2)
{
            ti->textbox.posX1 = ti->posX1 + (ti->posX2-ti->posX1 - l)/2;
            // ti->textbox.posX2 = ti->textbox.posX1 + l + ti-
>textbox.font.fontSize + ti->textbox.font.spacing;
            mouse_hide();
            ti->draw(ti);
            mouse_show();
        }
        k = textbox_event(&ti->textbox);
        tl = text_getLength(ti->textbox.content);
        tl = (tl == 0)?text_getLength(ti->textbox.defaultContent) : tl;
        if(tl != ltl){
            l = tl*(ti->textbox.font.fontSize + ti-
>textbox.font.spacing);
            ti->textbox.posX1 = ti->posX1 + (ti->posX2-ti->posX1 - l)/2;
            // ti->textbox.posX2 = ti->textbox.posX1 + l + ti-
>textbox.font.fontSize + ti->textbox.font.spacing;
            mouse_hide();
            ti->draw(ti);
            mouse_show();
        }
    }
    else{
        k = textbox_event(&ti->textbox);
    }
    if(ls != ti->textbox.status){
        mouse_hide();
        textinput_draw(ti);
        textbox_draw(&ti->textbox);
        mouse_show();
    }
    if(k==1){
        ti->textbox.status = TextboxDefault;
        mouse_hide();
        textinput_draw(ti);
        textbox_draw(&ti->textbox);
        mouse_show();
    }
    return k;
}

TextInput textinput_newDefault(char *ds, int x1, int y1, int x2, int y2,
char *buffer){
    Textbox tb;
    TextInput ti;
    memset(&tb, 0, sizeof(tb));
    tb = textbox_newDefault(ds, x1, y1, x2, y2, buffer);
    tb.status = TextboxDefault;
    tb.mstatus = TextboxMouseDefault;
    tb.cursorStatus = 0;
    tb.defaultContent = ds;
    tb.content = buffer;
    tb.maxLength = 10;
    tb.font.fontSize = 24;
    tb.font.fontColor = _BLACK,
    tb.font.spacing = 1;
```

```
    tb.font.rowSpacing = 0;
    tb.hint = '*';
    tb.type = TextboxText;
    tb.posX1 = x1+5;
    tb.posX2 = x2-5;
    tb.posY1 = y1+10;
    tb.posY2 = y2-10;
    ti.textbox = tb;
    ti.posX1 = x1;
    ti.posX2 = x2;
    ti.posY1 = y1;
    ti.posY2 = y2;
    ti.draw = textinput_drawDefault;
    ti.align = 0;
    return ti;
}

TextInput textinput_newTitle(char *ds, int x1, int y1, int x2, int y2,
char *buffer){
    Textbox tb;
    TextInput ti;
    memset(&tb, 0, sizeof(tb));
    tb = textbox_newDefault(ds, x1, y1, x2, y2, buffer);
    tb.status = TextboxDefault;
    tb.mstatus = TextboxMouseDefault;
    tb.cursorStatus = 0;
    tb.defaultContent = ds;
    tb.content = buffer;
    tb.maxLength = 14;
    tb.font.fontSize = 24;
    tb.font.fontColor = _BLACK,
    tb.font.spacing = 2;
    tb.font.rowSpacing = 0;
    tb.hint = '*';
    tb.type = TextboxText;
    ti.textbox = tb;
    ti.posX1 = x1;
    ti.posX2 = x2;
    ti.posY1 = y1;
    ti.posY2 = y2;
    ti.draw = textinput_drawTransparent;
    ti.align = 1;
    return ti;
}
```

5.2.30 user.c

```
/**
 * @file user.c
 * @author wywgwt (2504133124@qq.com)
 * @brief
 * @date 2023-04-20
 *
 * @copyright Copyright (c) 2023
 *
 */
```

```c
#include "user.h"

int length_judge(char *p, char *q)
{
    if ((strlen(p) < 3) && (strlen(q) < 3))
    {
        return 0;
        // account wrong,password wrong
    }
    if ((strlen(p) < 3) && !(strlen(q) < 3))
    {
        return 1;
        // account wrong,password right
    }
    if (!(strlen(p) < 3) && (strlen(q) < 3))
    {
        return 2;
        // account right,password wrong
    }
    if (!(strlen(p) < 3) && !(strlen(q) < 3))
    {
        return 3;
    }
}
int user_login(char *p, char *q, int k)
{

    if (k == 0)
    {
        FILE *fp;
        int m = 0;
        int k;
        int signal = 0;
        Userdata a;
        if ((fp = fopen("data\\user.dat", "rb+")) == NULL)
        {
            fp = fopen("data\\user.dat", "wb+");
        }
        rewind(fp);
        while (1)
        {
            fread(&a, sizeof(Userdata), 1, fp);
            if (strcmp(a.account, p) == 0)
            {
                if (strcmp(a.password, q) == 0)
                {
                    m++;
                    k=4;
                    fclose(fp);
                    break;
                    // account right,password right
                }
                else
                {
                    m++;
                    k=5;
                    fclose(fp);
                    break;
                    // account right,password wrong
```

```
                }
            }
            if (feof(fp) != 0)
            {
                k= 6;
                fclose(fp);
                break;
            }
        }
        return k;
    }
    if (k == 1)
    {
        FILE *fp;
        Userdata a;
        if ((fp = fopen("data\\user.dat", "ab+")) == NULL)
        {
            fp=fopen("data\\user.dat","wb+");
        }
        fseek(fp, 0, 2);
        strcpy(a.account, p);
        strcpy(a.password, q);
        fwrite(&a, sizeof(Userdata), 1, fp);
        fclose(fp);
    }
}
```

5.2.31 userpage.c

```
/**
 * @file userpage.c
 * @author wywgwt (2504133124@qq.com)
 * @brief
 * @date 2023-04-20
 *
 * @copyright Copyright (c) 2023
 *
 */
#include "userpage.h"
void userpage_button_draw1(Button *b)
{
    int x1 = b->posX1, y1 = b->posY1,
        x2 = b->posX2, y2 = b->posY2;
    Text t;
    t.content = b->content;
    button_drawWINUI(b);
    setcolor(_BLACK);
    setlinestyle(0, 0, THICK_WIDTH);
    circle(x1 + 15, (y1 + y2) / 2, 15);
}
void userpage_button_draw2(Button *b)
{
    int x1 = b->posX1, y1 = b->posY1,
        x2 = b->posX2, y2 = b->posY2;
    Text t = text_newDefault("添加用户", x1 + 100, y1, x2, y2);
    if (b->status == ButtonSelected)
```

```
    {
        t.font.fontColor = hex555f55;
    }
    button_drawWINUI(b);
    text_display(t);
    setcolor(_BLACK);
    setlinestyle(0, 0, THICK_WIDTH);
    circle(x1 + 15 + 5, (y1 + y2) / 2, 15);
    line(x1 + 15 + 5, y1, x1 + 15 + 5, y2);
    line(x1 + 5, y1 + 15, x1 + 30 + 5, y1 + 15);
}
void userpage_button_draw3(Button *b)
{
    int x1 = b->posX1, y1 = b->posY1,
        x2 = b->posX2, y2 = b->posY2;
    setcolor(_BLACK);
    setlinestyle(0, 1, 2);
    button_drawWINUI(b);
    line(x1, y1, x2, y2);
    line(x2, y1, x1, y2);
}
void userpage_button_draw5(Button *b)
{
    int x1 = b->posX1, y1 = b->posY1,
        x2 = b->posX2, y2 = b->posY2;
    setcolor(_BLACK);
    setlinestyle(0,1,2);
    // arc((5*x4+3*x3)/8,(6*y4+2*y3)/8,90,180,3*(y4-y3)/8);
    // arc((5*x4+3*x3)/8,(33*y4-y3)/32,90,157.38,13*(y4-y3)/32);
    // line((5*x4+3*x3)/8,(5*y3+3*y4)/8,(5*x4+3*x3)/8,(7*y3+y4)/8);
    // line((5*x4+3*x3)/8,(5*y4+3*y3)/8,(5*x4+3*x3)/8,(7*y4+y3)/8);
    // line((5*x4+3*x3)/8,(7*y3+y4)/8,(7*x4+x3)/8,(y3+y4)/2);
    // line((5*x4+3*x3)/8,(7*y4+y3)/8,(7*x4+x3)/8,(y3+y4)/2);
    line((5*x1+x2)/6,(y1+y2)/2,(5*x2+x1)/6,(y1+y2)/2);
    line((5*x1+x2)/6,(y1+y2)/2,(5*x1+x2)/6+(x2-x1)/4,(y1+y2)/2-(y2-
y1)/4);
    line((5*x1+x2)/6,(y1+y2)/2,(5*x1+x2)/6+(x2-x1)/4,(y1+y2)/2+(y2-
y1)/4);

}
void userpage_button_draw4(Button *b)
{
    int x1 = b->posX1, y1 = b->posY1,
        x2 = b->posX2, y2 = b->posY2;
    setcolor(_BLACK);
    setlinestyle(0, 1, 2);
    button_drawWINUI(b);
    line((x2 + 6 * x1) / 7, (2 * y1 + y2) / 3, (x1 + 6 * x2) / 7, (2 *
y1 + y2) / 3);
    line((x2 + 6 * x1) / 7, (2 * y2 + y1) / 3, (x1 + 6 * x2) / 7, (2 *
y2 + y1) / 3);
    line((x1 + 6 * x2) / 7, (2 * y1 + y2) / 3, (x1 + 6 * x2) / 7 - (x2 -
x1) / 7, (2 * y1 + y2) / 3 - (y2 - y1) / 4);
    line((x2 + 6 * x1) / 7, (2 * y2 + y1) / 3, (x2 + 6 * x1) / 7 + (x2 -
x1) / 7, (2 * y2 + y1) / 3 + (y2 - y1) / 4);
}
int userpage_deleteuser(int j, int k)
{
    int i;
```

```
    for (i = k; i < j; i++)
    {
        strcpy(appData()->uid[i], appData()->uid[i + 1]);
    }
    appData()->userCount--;
    return (j - 1);
} // 删除uid[k],j为有几个用户;
void userpage_changeuser(int k)
{
    char a[20] = {0};
    strcpy(a, appData()->uid[0]);
    strcpy(appData()->uid[0], appData()->uid[k]);
    strcpy(appData()->uid[k], a);
} // 将uid【k】切换到首位
int userpage_login(int j)
{
    char textInputBuffer1[50];
    char textInputBuffer2[50];
    Button c = button_new(340,150,372,182,"",userpage_button_draw5);
    Button b = button_new(350, 400,
                          650, 450,
                          "登 录",
                          button_drawWithText);
    TextInput t = textinput_newDefault(
        "请输入用户名",
        350, 200,
        650, 250,
        textInputBuffer1);
    TextInput t1 = textinput_newDefault(
        "请输入密码",
        350, 300,
        650, 350,
        textInputBuffer2);
    Text tc1, tc2, tc3, tc4, tw, tw1;
    memset(textInputBuffer1, 0, sizeof(textInputBuffer1));
    memset(textInputBuffer2, 0, sizeof(textInputBuffer2));
    while (1)
    {

        int signal = 0;
        int x1, x2, y1, y2, c0, c10;

        t1.textbox.type = TextboxPassword;
        t1.textbox.maxLength = 8;
        t.textbox.maxLength = 8;
        tw = text_newDefault("用户名过短！", 500, 210, 800, 260);
        tw.font.fontColor = _RED;
        tw1 = text_newDefault("密码过短！", 500, 310, 800, 360);
        tw1.font.fontColor = _RED;
        tc1 = text_newDefault("成功登录！", 400, 360, 800, 410);
        tc1.font.fontColor = _RED;
        tc2 = text_newDefault("账号已存在！", 400, 360, 800, 410);
        tc2.font.fontColor = _RED;
        tc3 = text_newDefault("注册成功！", 400, 360, 800, 410);
        tc3.font.fontColor = _RED;
        tc4 = text_newDefault("用户已存在！", 400, 360, 800, 410);
        tc4.font.fontColor = _RED;
        // Button b=button_new()
        mouse_hide();
```

```
        x1 = 412 - 60 - 20;
        x2 = 612 + 60 + 20;
        y1 = 234 - 90;
        y2 = 384 + 240;
        c0 = hex808080;
        c10 = hexd4bfaa;
        setfillstyle(1, c10);
        setcolor(c10);
        bar(x1 + 6, y1, x2 - 6, y2);
        bar(x1, y1 + 6, x2, y2 - 6);
        pieslice(x1 + 6, y1 + 6, 90, 180, 6);
        pieslice(x1 + 6, y2 - 6, 180, 270, 6);
        pieslice(x2 - 6, y1 + 6, 0, 90, 6);
        pieslice(x2 - 6, y2 - 6, 270, 360, 6);
        setcolor(c0);
        setlinestyle(0, 1, 3);
        line(x1 + 6, y1, x2 - 6, y1);
        line(x1 + 6, y2, x2 - 6, y2);
        line(x1, y1 + 6, x1, y2 - 6);
        line(x2, y1 + 6, x2, y2 - 6);
        arc(x1 + 6, y1 + 6, 90, 180, 6);
        arc(x1 + 6, y2 - 6, 180, 270, 6);
        arc(x2 - 6, y1 + 6, 0, 90, 6);
        arc(x2 - 6, y2 - 6, 270, 360, 6);
        button_draw(&b);
        button_draw(&c);
        textinput_draw(&t);
        textinput_draw(&t1);
        mouse_show();
        while (!signal)
        {
            int k, cs,bs, tbs, tbs1, lgs, ly1, ly2, ly3, ly4;
            int ly = 3;
            int s = 0;
            int p = 0;
            Text ta1, ta2, ta3, ta4;
            Mouse *m = mouse();
            mouse_update();
            digitalClock_getTime();
            ime_check();
            keybord_eat();
            ly1 = mouse_isClickedInBox(0, 0, 1024, y1);
            ly2 = mouse_isClickedInBox(0, y1, x1, y2);
            ly3 = mouse_isClickedInBox(x2, y1, 1024, y2);
            ly4 = mouse_isClickedInBox(0, y2, 1024, 768);
            k = bioskey(1);
            bs = button_event(&b);
            cs = button_event(&c);
            tbs = textinput_event(&t);
            tbs1 = textinput_event(&t1);
            if (keybord_isESCAPE(k))
            {
                bioskey(0);
                signal = -1;
                break;
            }
            if (ly1 == 1)
            {
                return;
```

```
            }
            if (ly2 == 1)
            {
                return;
            }
            if (ly3 == 1)
            {
                return;
            }
            if (ly4 == 1)
            {
                return;
            }
            if (tbs == 1)
            {
                t1.textbox.status = TextboxSelected;
            }
            if(cs==1)
            {
                signal = -1;
                break;
            }
            if (bs || (tbs1 == 1))
            {
                ly = length_judge(t.textbox.content,
t1.textbox.content);
                if (ly == 3)
                {
                    lgs = user_login(t.textbox.content,
t1.textbox.content, 0);
                    switch (lgs)
                    {
                    case 4:

                        for (p = 0; p < j; p++)
                        {
                            if (strcmp(appData()->uid[p],
t.textbox.content) == 0)
                            {
                                // text_display(tc4);
                                // delay(1000);
                                // signal = 2;
                                // break;
                                s++;
                            }
                        }
                        if(s!=0)
                        {
                            text_display(tc4);
                                delay(1000);
                                signal = 2;
                                break;
                        }
                        else{
                        strcpy(appData()->uid[j], t.textbox.content);
                        text_display(tc1);
                        delay(1000);
                        return 1;
                        break;
```

```
                        }

                    case 5:
                        text_display(tc2);
                        delay(1000);
                        break;
                    case 6:
                        text_display(tc3);
                        user_login(t.textbox.content,
t1.textbox.content, 1);
                        strcpy(appData()→uid[j], t.textbox.content);
                        delay(1000);
                        signal = 1;
                        return 1;
                        break;
                    }
                    break;
                }
            }
            switch (ly)
            {
            case 0:
                if (!(textbox_event(&t)))
                {
                    text_display(tw);
                }
                if (!(textbox_event(&t1)))
                {
                    text_display(tw1);
                }
                break;
            case 1:
                if (!(textbox_event(&t)))
                {
                    text_display(tw);
                }
                break;
            case 2:
                if (!(textbox_event(&t1)))
                {
                    text_display(tw1);
                }
                break;
            }
        }
        switch (signal)
        {
        case 1:
            return 1;
            break;
        case -1:
            debug(DEBUG, "EXIT.");
            return 0;
            break;
        }
    }
}

/// @brief
```

```
/// @return
int userpage()
{
    char textInputBuffer[50];
    int n = 1;
    int s = 0;
    memset(textInputBuffer, 0, sizeof(textInputBuffer));
    while (1)
    {
        int j = appData()->userCount;
        int buttonCount = 1;
        int signal = 0;
        int p;
        int j1 = 0;
        int p1 = -1;
        int x1, x2, y1, y2, c0, c10;
        int y3 = 410;
        int m;
        Text b[6];
        Button d[6];
        Button e[6];
        Button c;
        Text t1 = text_newDefault(appData()->uid[0], 350, 380, 640,
580);
        b[0] = text_newDefault(appData()->uid[0], 350, 380, 640, 580);
        mouse_hide();
        c0 = hex808080;
        c10 = _LIGHTGRAY;
        x1 = 412 - 60 - 20;
        x2 = 612 + 60 + 20;
        y1 = 234 - 90;
        y2 = 384 + 240;
        setfillstyle(1, c10);
        setcolor(c10);
        bar(x1 + 6, y1, x2 - 6, y2);
        bar(x1, y1 + 6, x2, y2 - 6);
        pieslice(x1 + 6, y1 + 6, 90, 180, 6);
        pieslice(x1 + 6, y2 - 6, 180, 270, 6);
        pieslice(x2 - 6, y1 + 6, 0, 90, 6);
        pieslice(x2 - 6, y2 - 6, 270, 360, 6);
        setcolor(c0);
        setlinestyle(0, 1, 3);
        line(x1 + 6, y1, x2 - 6, y1);
        line(x1 + 6, y2, x2 - 6, y2);
        line(x1, y1 + 6, x1, y2 - 6);
        line(x2, y1 + 6, x2, y2 - 6);
        arc(x1 + 6, y1 + 6, 90, 180, 6);
        arc(x1 + 6, y2 - 6, 180, 270, 6);
        arc(x2 - 6, y1 + 6, 0, 90, 6);
        arc(x2 - 6, y2 - 6, 270, 360, 6);
        setcolor(_WHITE);
        circle(512, 300, 60);
        circle(512, 282, 15);
        arc(512, 352.5, 27, 153, 55.5);
        if (j < 5)
        {
            y3 += 30 * (j - 1);
            c = button_new(340, y3, 690, y3 + 30, " ",
userpage_button_draw2);
```

```
            button_draw(&c);
        }
        for (p = 0; p < 5; p++)
        {
            b[p] = text_newDefault(appData()→uid[p], 350, 380 + 30 * p,
640, 580 + 30 * p);
            d[p] = button_new(650, 385 + 30 * p, 670, 405 + 30 * p, "",
userpage_button_draw3);
            e[p] = button_new(610, 385 + 30 * p, 630, 405 + 30 * p, "",
userpage_button_draw4);
        }
        for(p=0;p<j;p++)
        {
            text_display(b[p]);
        }
        for (p = 0; p < j; p++)
        {
            setfillstyle(1, _LIGHTGRAY);
            bar(650, 385 + 30 * p, 670, 405 + 30 * p);
            bar(610, 385 + 30 * p, 630, 405 + 30 * p);
        }
        mouse_show();
        digitalClock_getTime();
        while (!signal)
        {
            int k, a = 0;
            int i = 0;
            int be[5] = {0};
            int ly1, ly2, ly3, ly4;
            int tbs;
            Mouse *m = mouse();
            mouse_update();
            keybord_eat();
            ly1 = mouse_isClickedInBox(0, 0, 1024, y1);
            ly2 = mouse_isClickedInBox(0, y1, x1, y2);
            ly3 = mouse_isClickedInBox(x2, y1, 1024, y2);
            ly4 = mouse_isClickedInBox(0, y2, 1024, 768);
            if (j == 1)
            {
                text_display(t1);
            }
            if (j < 5)
            {
                a = button_event(&c);
            }
            if (a == 1)
            {
                if (userpage_login(j))
                {
                    j++;
                    appData()→userCount++;
                    break;
                }
                break;
            }
            for (p = 0; (p < j) && (j != 1); p++)
            {
                if (button_event(d + p))
                {
```

```
                p1 = p;
                j1 = userpage_deleteuser(j, p1);
                j = j1;
                signal = 2;
            }
            if (button_event(e + p))
            {
                p1 = p;
                userpage_changeuser(p1);
                s = 1;
                signal = 2;
            }
        }
        digitalClock_getTime();
        k = bioskey(1);
        if (keybord_isESCAPE(k))
        {
            bioskey(0);
            signal = -1;
            break;
        }
        if (ly1 == 1)
        {
            signal = 1;
            if (s == 0)
            {
                return 0;
            }
            if (s == 1)
            {
                return 1;
            }
        }
        if (ly2 == 1)
        {
            signal = 1;
            if (s == 0)
            {
                return 0;
            }
            if (s == 1)
            {
                return 1;
            }
        }
        if (ly3 == 1)
        {
            signal = 1;
            if (s == 0)
            {
                return 0;
            }
            if (s == 1)
            {
                return 1;
            }
        }
        if (ly4 == 1)
        {
```

```
                signal = 1;
                if (s == 0)
                {
                    return 0;
                }
                if (s == 1)
                {
                    return 1;
                }
            }
        }
        switch (signal)
        {
        case -1:
            if (s == 0)
            {
                return 0;
            }
            if (s == 1)
            {
                return 1;
            }
            break;
        case 1:
            if (s == 0)
            {
                return 0;
            }
            if (s == 1)
            {
                return 1;
            }
            break;
        case 2:
            break;
        }
    }
}
```

5.2.32 小工具

为了便于生成本项目特有的二进制格式图片，我们写了如下 python 脚本。

```
import struct
import cv2
import platform

m = {
    0x000000:0,
    0x800000:1,
    0x008000:2,
    0x808000:3,
    0x000080:4,
    0x800080:5,
    0x008080:6,
    0xc0c0c0:7,
```

```
    0xc0dcc0:8,
    0xa6caf0:9,
    0x2a3faa:10,
    0x2a3fff:11,
    0x2a5f00:12,
    0x2a5f55:13,
    0x2a5faa:14,
    0x2a5fff:15,
    0x2a7f00:16,
    0x2a7f55:17,
    0x2a7faa:18,
    0x2a7fff:19,
    0x2a9f00:20,
    0x2a9f55:21,
    0x2a9faa:22,
    0x2a9fff:23,
    0x2abf00:24,
    0x2abf55:25,
    0x2abfaa:26,
    0x2abfff:27,
    0x2adf00:28,
    0x2adf55:29,
    0x2adfaa:30,
    0x2adfff:31,
    0x2aff00:32,
    0x2aff55:33,
    0x2affaa:34,
    0x2affff:35,
    0x550000:36,
    0x550055:37,
    0x5500aa:38,
    0x5500ff:39,
    0x551f00:40,
    0x551f55:41,
    0x551faa:42,
    0x551fff:43,
    0x553f00:44,
    0x553f55:45,
    0x553faa:46,
    0x553fff:47,
    0x555f00:48,
    0x555f55:49,
    0x555faa:50,
    0x555fff:51,
    0x557f00:52,
    0x557f55:53,
    0x557faa:54,
    0x557fff:55,
    0x559f00:56,
    0x559f55:57,
    0x559faa:58,
    0x559fff:59,
    0x55bf00:60,
    0x55bf55:61,
    0x55bfaa:62,
    0x55bfff:63,
    0x55df00:64,
    0x55df55:65,
    0x55dfaa:66,
```

```
    0x55dfff:67,
    0x55ff00:68,
    0x55ff55:69,
    0x55ffaa:70,
    0x55ffff:71,
    0x7f0000:72,
    0x7f0055:73,
    0x7f00aa:74,
    0x7f00ff:75,
    0x7f1f00:76,
    0x7f1f55:77,
    0x7f1faa:78,
    0x7f1fff:79,
    0x7f3f00:80,
    0x7f3f55:81,
    0x7f3faa:82,
    0x7f3fff:83,
    0x7f5f00:84,
    0x7f5f55:85,
    0x7f5faa:86,
    0x7f5fff:87,
    0x7f7f00:88,
    0x7f7f55:89,
    0x7f7faa:90,
    0x7f7fff:91,
    0x7f9f00:92,
    0x7f9f55:93,
    0x7f9faa:94,
    0x7f9fff:95,
    0x7fbf00:96,
    0x7fbf55:97,
    0x7fbfaa:98,
    0x7fbfff:99,
    0x7fdf00:100,
    0x7fdf55:101,
    0x7fdfaa:102,
    0x7fdfff:103,
    0x7fff00:104,
    0x7fff55:105,
    0x7fffaa:106,
    0x7fffff:107,
    0xaa0000:108,
    0xaa0055:109,
    0xaa00aa:110,
    0xaa00ff:111,
    0xaa1f00:112,
    0xaa1f55:113,
    0xaa1faa:114,
    0xaa1fff:115,
    0xaa3f00:116,
    0xaa3f55:117,
    0xaa3faa:118,
    0xaa3fff:119,
    0xaa5f00:120,
    0xaa5f55:121,
    0xaa5faa:122,
    0xaa5fff:123,
    0xaa7f00:124,
    0xaa7f55:125,
```

```
    0xaa7faa:126,
    0xaa7fff:127,
    0xaa9f00:128,
    0xaa9f55:129,
    0xaa9faa:130,
    0xaa9fff:131,
    0xaabf00:132,
    0xaabf55:133,
    0xaabfaa:134,
    0xaabfff:135,
    0xaadf00:136,
    0xaadf55:137,
    0xaadfaa:138,
    0xaadfff:139,
    0xaaff00:140,
    0xaaff55:141,
    0xaaffaa:142,
    0xaaffff:143,
    0xd40000:144,
    0xd40055:145,
    0xd400aa:146,
    0xd400ff:147,
    0xd41f00:148,
    0xd41f55:149,
    0xd41faa:150,
    0xd41fff:151,
    0xd43f00:152,
    0xd43f55:153,
    0xd43faa:154,
    0xd43fff:155,
    0xd45f00:156,
    0xd45f55:157,
    0xd45faa:158,
    0xd45fff:159,
    0xd47f00:160,
    0xd47f55:161,
    0xd47faa:162,
    0xd47fff:163,
    0xd49f00:164,
    0xd49f55:165,
    0xd49faa:166,
    0xd49fff:167,
    0xd4bf00:168,
    0xd4bf55:169,
    0xd4bfaa:170,
    0xd4bfff:171,
    0xd4df00:172,
    0xd4df55:173,
    0xd4dfaa:174,
    0xd4dfff:175,
    0xd4ff00:176,
    0xd4ff55:177,
    0xd4ffaa:178,
    0xd4ffff:179,
    0xff0055:180,
    0xff00aa:181,
    0xff1f00:182,
    0xff1f55:183,
    0xff1faa:184,
```

```
    0xff1fff:185,
    0xff3f00:186,
    0xff3f55:187,
    0xff3faa:188,
    0xff3fff:189,
    0xff5f00:190,
    0xff5f55:191,
    0xff5faa:192,
    0xff5fff:193,
    0xff7f00:194,
    0xff7f55:195,
    0xff7faa:196,
    0xff7fff:197,
    0xff9f00:198,
    0xff9f55:199,
    0xff9faa:200,
    0xff9fff:201,
    0xffbf00:202,
    0xffbf55:203,
    0xffbfaa:204,
    0xffbfff:205,
    0xffdf00:206,
    0xffdf55:207,
    0xffdfaa:208,
    0xffdfff:209,
    0xffff55:210,
    0xffffaa:211,
    0xccccff:212,
    0xffccff:213,
    0x33ffff:214,
    0x66ffff:215,
    0x99ffff:216,
    0xccffff:217,
    0x007f00:218,
    0x007f55:219,
    0x007faa:220,
    0x007fff:221,
    0x009f00:222,
    0x009f55:223,
    0x009faa:224,
    0x009fff:225,
    0x00bf00:226,
    0x00bf55:227,
    0x00bfaa:228,
    0x00bfff:229,
    0x00df00:230,
    0x00df55:231,
    0x00dfaa:232,
    0x00dfff:233,
    0x00ff55:234,
    0x00ffaa:235,
    0x2a0000:236,
    0x2a0055:237,
    0x2a00aa:238,
    0x2a00ff:239,
    0x2a1f00:240,
    0x2a1f55:241,
    0x2a1faa:242,
    0x2a1fff:243,
```

```
    0x2a3f00:244,
    0x2a3f55:245,
    0xfffbf0:246,
    0xa0a0a4:247,
    0x808080:248,
    0xff0000:249,
    0x00ff00:250,
    0xffff00:251,
    0x0000ff:252,
    0xff00ff:253,
    0x00ffff:254,
    0xffffff:255,
}

def RGB2Hex(r, g, b):
    h = 0
    h += r << 16
    h += g << 8
    h += b
    return h

def Hex2Code(h):
    return m[h]

if __name__ == '__main__':
    infilename = input()
    outfilename = infilename+'.bin'
    image = cv2.imread(infilename)
    size = image.shape
    #print(size)
    h = size[0]
    w = size[1]
    with open(outfilename, mode='wb') as outfile:
        bt = struct.pack("2h", w, h)
        outfile.write(bt)
        for i in range(h):
            for j in range(w):
                pixelcolor = image[i, j]

                r = pixelcolor[2]
                g = pixelcolor[1]
                b = pixelcolor[0]
                #print(r,g,b)
                h = RGB2Hex(r, g, b)
                #print(h)
                try:
                    c = Hex2Code(h)
                    bt = struct.pack('B', c)
                except:
                    print("[error] not a Windows standard 8 bit
bitmap:")
                    print(f"color rgb={[r, g, b]} do not exist in the
palette")
                    print("try ms paint to convert the image once")
                    exit(1)
                #print(c)

                outfile.write(bt)
```

```
    outfile.close()
    print("OK")
```

6 小组工作

6.1 代码量统计

由 VS Code 插件 VS Code Counter，排除注释、空行，得到本项目有效代码量约在6500行以上。

剔除鼠标、汉字显示、汉字输入法等借用学长代码的部分，以及图形驱动初始化等数据部分，保守估计，本项目自主构建的代码量在5500行以上。

Details

Date : 2023-04-22 01:42:12

Directory e:\dev\bc\newmemo\src

Total : 67 files, 6555 codes, 808 comments, 466 blanks, all 7829 lines

Summary / Details / Diff Summary / Diff Details

Files

filename	language	code	comment	blank	total
src/addimage.c	C	61	9	6	76
src/addimage.h	C	11	9	3	23
src/anim.c	C	203	9	3	215
src/anim.h	C++	10	9	5	24
src/app.c	C	133	9	2	144
src/app.h	C	32	9	7	48
src/auth.c	C	105	16	5	126
src/auth.h	C++	20	9	6	35
src/button.c	C	373	9	16	398
src/button.h	C	39	9	5	53
src/debug.c	C	27	9	2	38
src/debug.h	C++	15	9	6	30

图 6-1 代码量统计

src/digclock.c	C	20	9	2	31
src/digclock.h	C++	9	9	5	23
src/drawpad.c	C	104	9	3	116
src/drawpad.h	C	11	9	3	23
src/exitsave.c	C	28	9	5	42
src/exitsave.h	C++	7	9	3	19
src/global.h	C	11	9	7	27
src/homepage.c	C	163	12	3	178
src/homepage.h	C	14	9	4	27
src/hz.c	C	188	17	38	243
src/hz.h	C	11	0	8	19
src/hzinput.c	C	333	30	11	374
src/hzinput.h	C	24	6	9	39
src/image.c	C	75	9	5	89
src/image.h	C	11	9	6	26
src/imagebox.c	C	111	9	5	125
src/imagebox.h	C++	13	9	6	28
src/ime.c	C	90	16	8	114
src/ime.h	C	27	9	6	42
src/init.c	C	13	10	3	26
src/init.h	C	8	9	5	22
src/keyboard.c	C	29	9	8	46
src/keyboard.h	C	27	9	6	42

图 6-2　代码量统计（续）

src/main.c	C	7	9	2	18
src/main.h	C++	5	9	3	17
src/meditor.c	C	697	21	13	731
src/meditor.h	C++	61	9	9	79
src/memo.c	C	98	12	7	117
src/memo.h	C	38	11	14	63
src/memos.c	C	128	12	7	147
src/memos.h	C++	22	0	3	25
src/mfile.c	C	66	9	5	80
src/mfile.h	C++	13	9	10	32
src/mouse.c	C	238	74	20	332
src/mouse.h	C	40	21	10	71
src/mset.c	C	106	9	8	123
src/mset.h	C	19	9	3	31
src/mshare.c	C	213	9	10	232
src/mshare.h	C	29	9	8	46
src/router.c	C	410	12	14	436
src/router.h	C++	39	9	5	53
src/scroll.c	C	109	9	5	123
src/scroll.h	C++	34	9	7	50
src/svga.c	C	310	9	7	326
src/svga.h	C	289	9	11	309
src/text.c	C	134	12	7	153
src/text.h	C	28	9	7	44

图 6-3　代码量统计（续）

src/textbox.c	C	246	13	8	267
src/textbox.h	C	47	9	7	63
src/textipt.c	C	142	11	8	161
src/textipt.h	C	18	9	5	32
src/user.c	C	78	29	2	109
src/user.h	C++	12	10	4	26
src/userpage.c	C	494	56	6	556
src/userpage.h	C	29	11	6	46

Summary / Details / Diff Summary / Diff Details

图 6-4　代码量统计（续）

6.2　小组合作

本项目中，胡天齐同学完成架构搭建、控件设计、页面功能和部分界面绘制、组件动画绘制等； 汪耀武同学完成数据结构、部分页面功能、大部分界面绘制、跳转动画绘制等。

经统计，两人代码比例胡天齐:汪耀武约为6:4。