

电子科技大学

UNIVERSITY OF ELECTRONIC SCIENCE AND TECHNOLOGY OF CHINA

新生项目结课报告



C 语言新生项目—— GhostTyper 英文打字训练软件

学 院 信息与通信工程学院

专 业 电子信息类

作者姓名 李书扬 王宇杰

学 号 2021010910011/2021010910017

指导教师 蒲恬

姓名	学号	贡献度
李书扬	2021010910011	50%
王宇杰	2021010910017	50%

目录

摘要.....3

Abstract.....4

项目编写前的准备和预测5

 一、 背景介绍.....5

 1.1 编写目的5

 1.2 项目背景5

 二、 项目的实现特点6

 2.1 要求6

 2.2 目标7

 2.3 条件、假定和限制.....7

 2.4 可行性研究方法7

 2.5 决定可行性的主要因素.....7

 2.6 本项目的局限性8

 2.7 确认需要实现的功能8

项目的实现过程.....8

 一、 项目的文件系统8

 1.1 文件和代码状态8

 1.2 项目的编译环境介绍10

 二、 项目主要模块介绍实现11

 1.1 UI 界面的实现11

 1.2 文件管理的实现24

 1.3 打字界面的实现27

 1.4 额外功能的实现33

 三、 项目开发进度设置.....43

评价、经验与总结44

 一、项目的最终完成情况44

 四、 项目的优势44

 三、项目的各方面评价.....44

 3.1 生产率评价.....44

 3.2 技术方案评价.....44

 3.3 产品质量评价.....44

 3.4 产品成品的展示45

摘要

本项目名称为：GhostTyper——英文打字训练软件。界面简单清爽，内置功能和基础商业软件类似，且具有用户名和文章可以自由管理，自由定义这一优势。具体实现了：文章训练、文章管理、排行榜、打字游戏、单词练习等功能。在界面的设计上使用 EasyX 进行编程，同时结合面向对象的编程方法和编程思想，将界面和文件索引系统分开存储，实现本地化保存数据的安全性。该英文打字训练软件具有较高的美观性和完备的功能，可以作为日常打字训练的好帮手长期使用。

关键词:EasyX ,C 语言，C++，文件操作，打字训练

Abstract

The name of this project is GhostTyper , an English typing training software. Its interface is simple, and its built-in functions are similar to the basic commercial software, moreover, during typing practices your user name and articles can be freely manage, it's a function beyond almost commercial software. This software realizes the functions of article training, article management, ranking, typing game and word practice with a visual keyboard. In the design of the interface, EasyX Library is used for programming. Combined with the object-oriented programming method and programming idea, the interface and file management system are stored separately to realize the security of localized data storage. This English typing training software has complete functions. It can be used as a good helper for students' daily typing training.

Key words: EasyX, C language, C + +, file operation, typing training

项目编写前的准备和预测

一、背景介绍

本项目依托 C 语言课程的需要，进行英文打字软件的项目设计开发。

通过比较和借鉴现有产品，综合得出本项目需要的各模块内容和需要实现的目标。来做到和商业性软件有一定可比性和一定优势。

项目的具体内容介绍见下。

1.1 编写目的

本项目旨在编写一个较为完善的打字软件系统，需要做到的目标有：

1. 可以和市面上现有的商业化打字软件有可比性，有相似性或者其可替代功能。
2. 在和商业项目具有可比性的同时，拥有一定自己优势或独特之处。
3. 能够符合 C 语言新生项目课程的需要，在项目的截止日期之前完成打字训练软件功能完善的开发以及报告的写作。将进行迅捷开发和安排合理的开发流程作为本项目开发的目的之一。
4. 旨在提升参与项目学生的编程能力，抽象实现能力和报告编写写作能力。
5. 通过对项目的写作了解 C 语言编程大项目时需要的文件结构架构能力。

1.2 项目背景

1.2.1 项目的发起情况

开发软件的名称：英文打字训练软件——GhostTyper

项目的任务提出者：高级语言程序设计（新生项目课程）

项目的开发者：李书扬、王宇杰

项目的用户：暂定为有打字学习需求的群体，以及本项目报告汇报的对象

实现软件的单位：个人

1.2.2 软件其他系统的关系

本项目与市面上现有的商业打字训练软件处于一种竞争和借鉴的关系。将现有的商业打字软件细分开来的功能如下：

1. 能够存入不同的文章，在打字训练时进行自选文章
2. 个人用户排行榜
3. 在打字训练过程中，显示字每分钟数
4. 通过对键盘上键位的指示开展的新手引导
5. 一些打字趣味游戏

.....

我们的打字训练软件系统也应该同样拥有以上功能，对以上几大小系统，应该为一种包含、借鉴、竞争的关系。

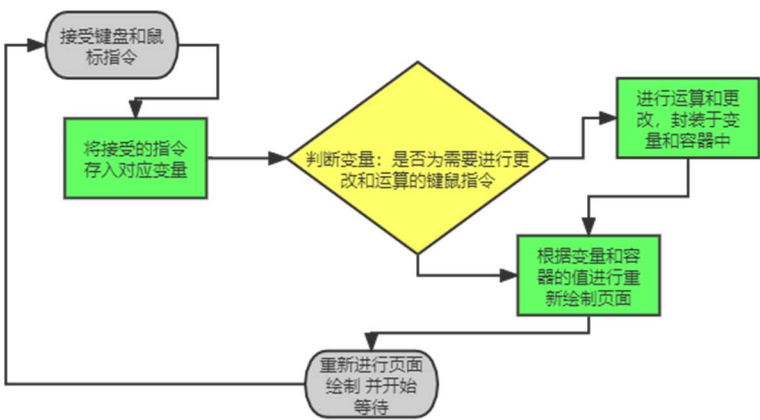
二、项目的实现特点

本项目的实现应该有着：模块化、面向对象和广阔借鉴素材，快速开发，简化复杂原理，差异化项目开发的特点。

具体总结为：英文打字训练软件——幽灵打字通，是一款通过 c++ 与 easyx 编写的简单的打字软件，可以实现简单的打字的练习。打字时可以手动选择想要训练的文章。文章为 txt 格式，用户可以自行增加或删除文章，从而使打字训练可以有更多的文章选择。同时本款软件还可以记录不同用户的训练准确率，所用时间等生成一个排行榜，供用户进行参考。除了基本的打字之外，本软件还植入了一款简单的打字游戏，使用户的使用体验更丰富。

2.1 要求

1. 功能：打字训练、文章增减管理、打字游戏、新手引导联系
2. 性能：在 windows11 平台 x86/64，一般商用处理器，运行内存 2g 以上运行时，占用运行内存 24 - 40MB 左右且较为稳定 占用 CPU 可忽略不计
3. 输入：通过键盘和鼠标共同接受指令
4. 基本的数据流程和处理流程：



图表 1 数据流程和处理流程

5. 安全与保密要求：无
6. 与软件相关的其他系统：EasyX 库函数支持
7. 完成期限：2022 年 1 月 1 日前

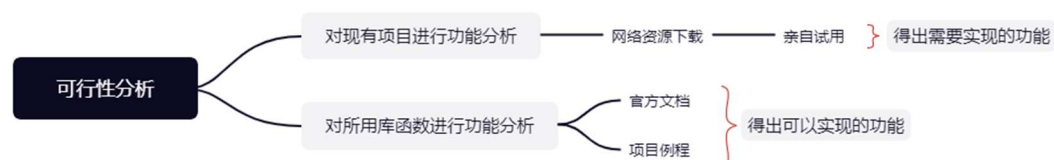
2.2 目标

1. 人力与设备费用的节省：采用现有个人电脑和社区版本的 Visual Studio，开发成本为零
2. 管理信息服务的改进：在文章管理系统中能够本地添加文章，而非商业运作软件中只能练习制定文章或者从云端下载文章，缺乏自选性
3. 决策系统的改进：让单个用户拥有多个用户名成为可能，而不是在单一机器、单一软件的使用时，只能使用在打字软件中登录的用户名
4. 人员工作效率的提高：进行合理的分工，模块化开发该项目

2.3 条件、假定和限制

1. 建议开发软件运行的最短寿命：本地化运行，开发完成后可用至删除
2. 进行系统方案选择比较的期限：0 天
3. 经费来源和使用限制：无经费预算，自主开发。
4. 法律和政策方面的限制：使用开源库和开源社区中的代码、素材，无法律责任。
5. 硬件、软件、运行环境和开发环境的条件和限制：适用于 Windows 平台，且在 win10 及更高版本中运行。
6. 可利用的信息和资源：开源社区资源，如 CSDN 和 GitHub，以及 EasyX 官方文档。
7. 建议开发软件投入使用的最迟时间：2022 年 1 月 1 日

2.4 可行性研究方法



图表 2 可行性分析

在对竞争产品和其他系统的研究中，采用在网络上下载相关产品和进行产品的直接试用，发掘现有产品的功能特点，并参照现有产品的功能特点对于本产品功能进行改进和完善。

通过对于现有库函数文档的阅读、以及对于案例代码的重新再发现，发掘现有库函数的可实现功能范围和不足之处，进行功能的增删查改。

2.5 决定可行性的主要因素

1. 大量同质化的优质商业化竞争产品，有着大量可供借鉴的界面布局。
2. 大量同质化的优质商业化竞争产品，有着可以参考的功能设置。

3. 本项目能够作为新生项目提出，说明该项目有开源代码的可能性较高。
4. 库函数功能较为精简强大，在一定时限内进行开发成功的可能性较高。

2.6 本项目的局限性

1. 现有库函数的功能较为简单，同时项目开发的时间限制让一些高级功能的开发具有一定局限性而难以实现。
2. 现有功能较完善，难以做出差异化
3. 将学生作品和商业软件比较，使学生自学内容较多

2.7 确认需要实现的功能

1. 自由的文章管理功能：包括自主增添删除文章
2. 较为人性化的打字界设置：包括输入进度、打字进行时间、正确率、每分钟字数的提醒
3. 较为个人定制化的排行榜设置：可以做到在文章训练时用户指定特定用户名进行训练，然后进行文章上榜
4. 具有趣味性的打字游戏
5. 具有新手引导性的功能：确定为，通过在屏幕上显示虚拟键盘、能够在虚拟键盘的相应位置找到自己所需要打的字符按钮。

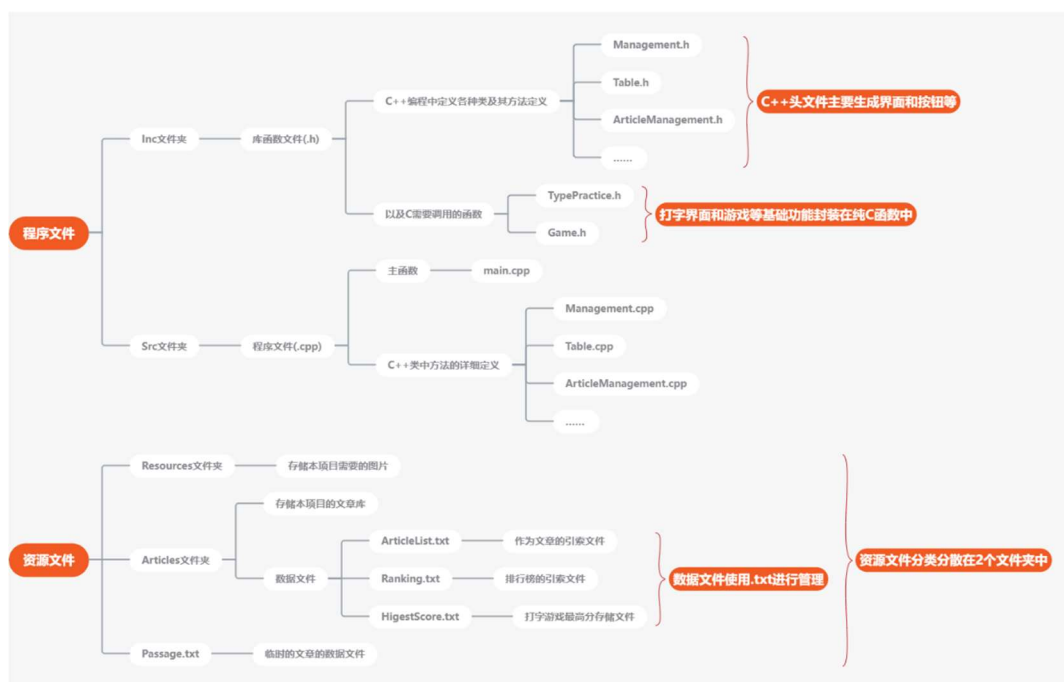
项目的实现过程

本项目采用模块化的编程方法。下文将从各个模块的角度进行详细的流程和程序实现的分析。

一、项目的文件系统

1.1 文件和代码状态

本项目的文件结构较为清晰，使用思维导图的方式展现部分分支完整的文件结构树，库文件采用英文单词含义命名的方法，且在下文中有详细的分模块介绍，所以不过多赘述，有部分文件由于思维导图容量有限，没有进行展示。



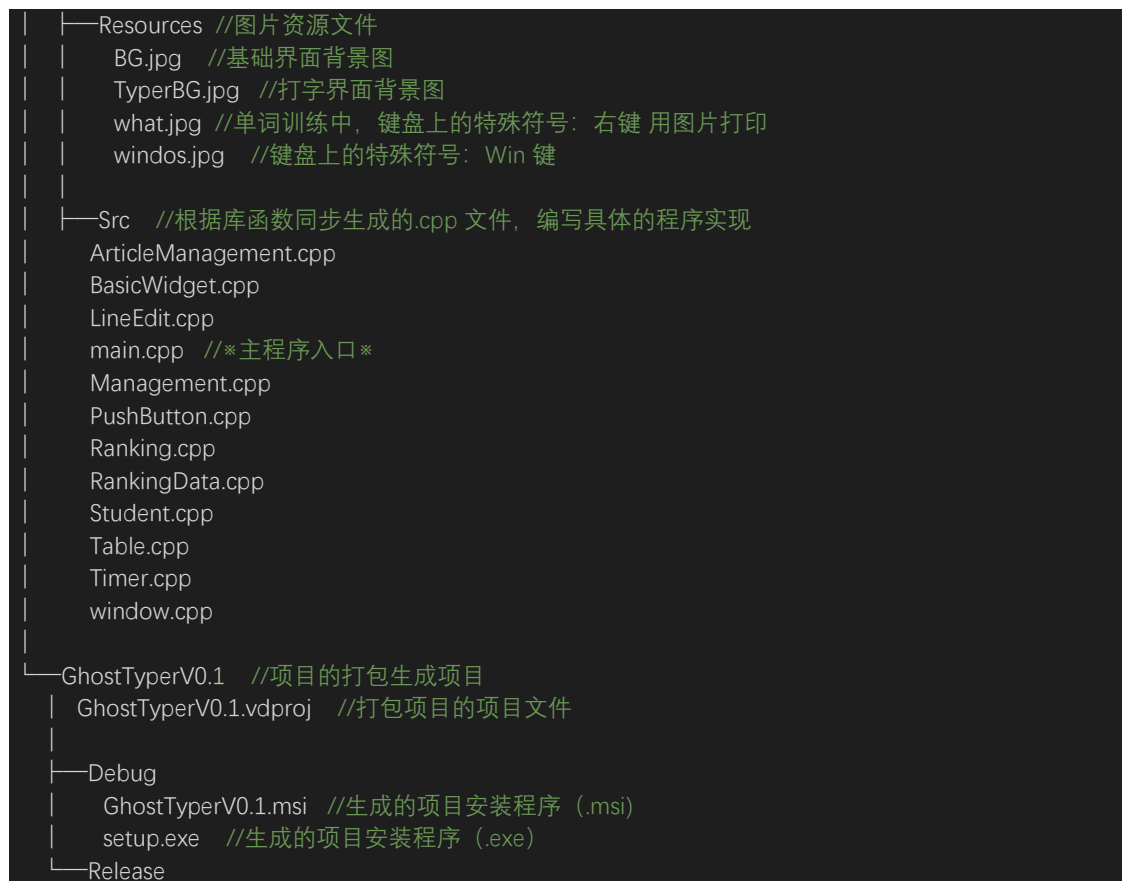
图表 3 文件系统

在 VisualStudio2022 中，项目的文件结构显示如下。

```

|—GhostTyper //项目的主文件目录
| | GhostTyper.sln //工程项目解决方案的名称
| | GhostTyper.vcxproj //生成项目的名称
| | ..... //※省略 Debug/.vs/Release 等调试文件夹※
| | passage.txt //临时文件：本次练习的文章
|
|—Articles //文章和索引文件目录
| | ArticleList.txt //文章目录的索引
| | Darkness.txt //以下为具体的文章篇目
| | HighestScore.txt //打字游戏最高成绩的存储文件
| | word.txt //单词练习的单词词库
| | Ranking.txt //排行榜的存储文件
| | ShortLife.txt //其他文章
| | SunShine.txt
| | .....
|
|—Inc //库文件目录
| | ArticleManagement.h //基础管理界面的库文件
| | BasicWidget.h //计算打表基础宽度的库文件
| | Configure.h //easyX.h 的功能扩展配置
| | GAME.h //打字游戏的程序库
| | LineEdit.h //文章管理界面打印库
| | Management.h //主界面即管理系统库
| | PushButton.h //按钮功能实现库
| | Ranking.h //排行榜打印库
| | RankingData.h //读取排行榜信息的库
| | Student.h //存储用户（学生）打字成绩需要的库
| | Table.h //打印文章列表的库
| | Timer.h //计时器库
| | typeparactise.h //※文章练习的实现库※
| | window.h //初始化窗口的库

```



图表 4 库文件结构

项目的整体代码行数在 4000 行左右，使用正则表达式查询结果为：4271 行



图表 5

1.2 项目的编译环境介绍

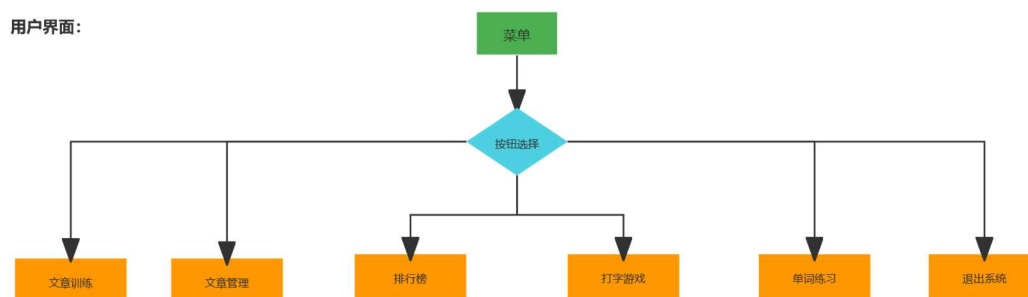
- VS 版本：最低为 **VS2019**，并且安装最新工具集，即：SDK 版本为 **10.0**，平台工具集为：**v143**
- 也可以通过对于项目 SDK 和工具集的重新定向，以此通过低版本的 VS 进行运行（已经在 **VS2017** 尝试过）
- C++ 标准：**ISO C++17 标准 (/std:c++17)** 及以上才能通过编译
- 同时 C/C++ 语言选项中：符合模式应该设置为否
- 在项目使用的编码中设置：**多字节字符集**，而非 UTF-8 字符集
- 如果无法正确调用库函数，请在 VC++ 选项中添加包含路径：
\$(SolutionDir)\Inc 否则无法读取对应库文件

- 使用的第三方库仅为中文最新版本的：EasyX.h，官网地址为 easyx.cn

二、项目主要模块介绍实现

1.1 UI 界面的实现

如下是用户界面的基本逻辑：



图表 6 用户界面

开始的时候我们看到了菜单界面。

通过菜单上的几个按钮我们可以选择对应的功能，这些按钮是与对应功能的值绑定的，通过点击它们，会生成传到主函数的返回值，再有 Management 类中的 run () 函数进行处理，就可以做到通过这些按钮来进入不同的功能了。

文章训练：本软件的主要功能，实现简单的打字训练。

文章管理：可以增添新的文章，或者对已有的文章进行删除。

排行榜：查看不同用户的排名。

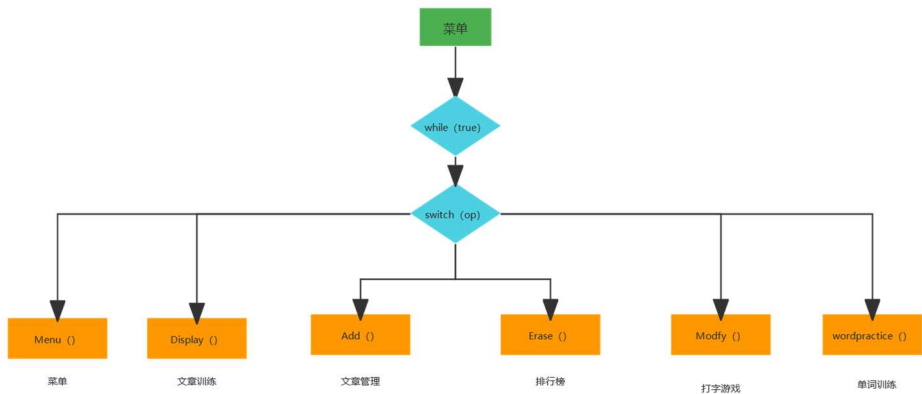
打字游戏：一款从界面上方不断下降字符的打字游戏。

单词练习：针对词汇的专项练习。

1.1.1 菜单



图表 7 菜单



图表 8 菜单代码逻辑

1.包含内容

菜单包括了：文章训练、文章管理、排行榜、打字游戏、单词训练、退出。

具体介绍：

文章训练：本软件的主要功能，实现简单的打字训练。

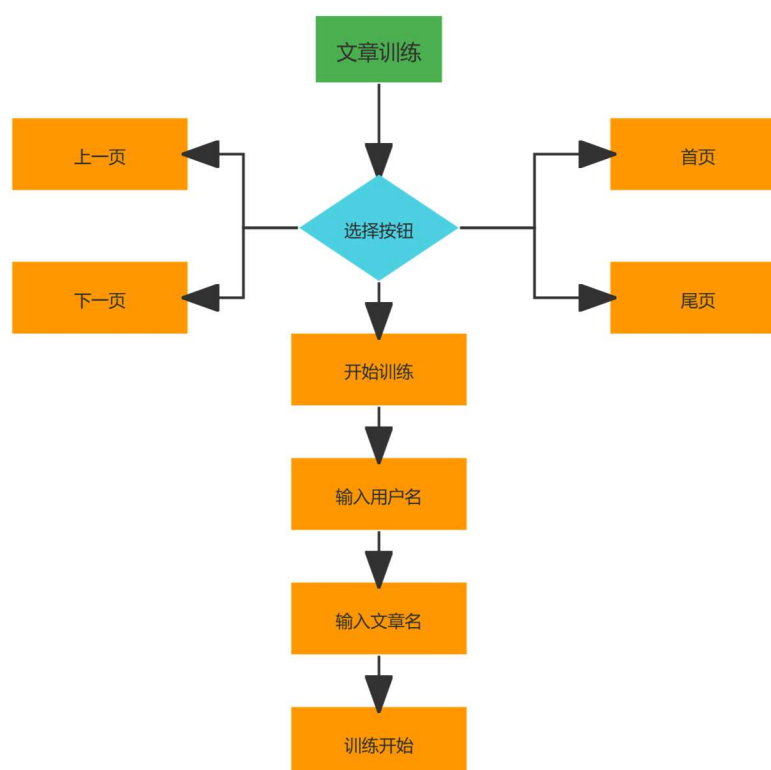
文章管理：可以增添新的文章，或者对已有的文章进行删除。

排行榜：查看不同用户的排名。

打字游戏：一款从界面上方不断下降字符的打字游戏。

单词练习：针对词汇的专项练习。

l) 文章训练：



图表 9 文章训练逻辑

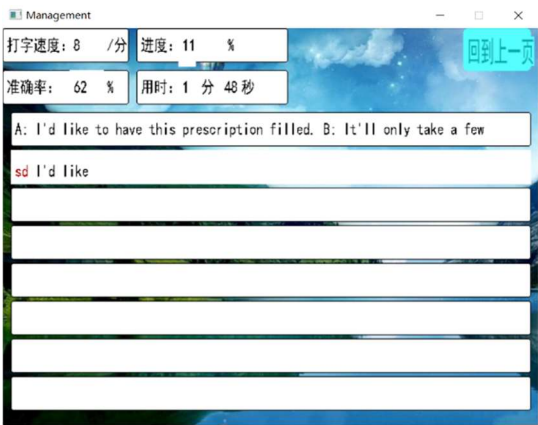
The screenshot shows a web application window titled 'ghosttyper'. It features a table with three columns: '文章名称' (Article Name), '文章字数' (Article Word Count), and '录入时间' (Entry Time). The table contains six rows of data. To the left of the table is a vertical sidebar with five buttons: '开始训练' (Start Training), '上一页' (Previous Page), '下一页' (Next Page), '首页' (Home Page), and '尾页' (End Page). At the bottom left, it says '第1页 / 共1页' (Page 1 / Total 1 page). The background of the interface is a map of Japan.

文章名称	文章字数	录入时间
ForeverTw	283	2021年12月19日
DrugHouse	316	2021年12月19日
Talks	102	2021年12月19日
Me	41	2021年12月19日
abd	91	2021年12月19日
LOL	192	2021年12月19日

图表 10 文章训练界面

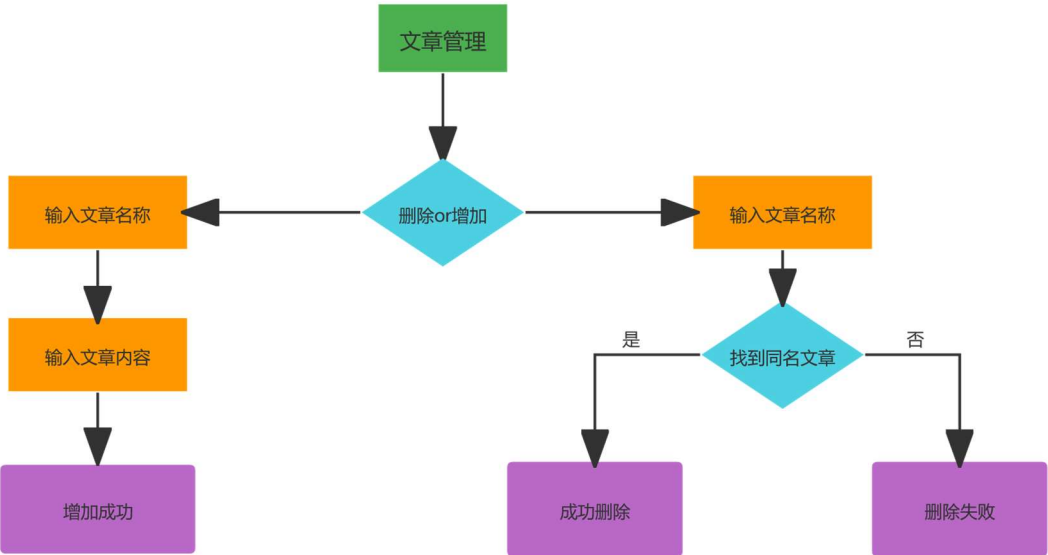
进入文章训练后会出现上图的表格，表格中有文章名称、文章字数、录入时间，这三个文章的基本信息来供用户参考。左方还有五个按钮，后四个是切换表格的内容，本项目中添加的文章较少，故表格只有一页。

正式训练的内容：



图表 11 正式训练界面

II) 文章管理



图表 12 文章管理逻辑



图表 13 添加文章界面

III) 排行榜



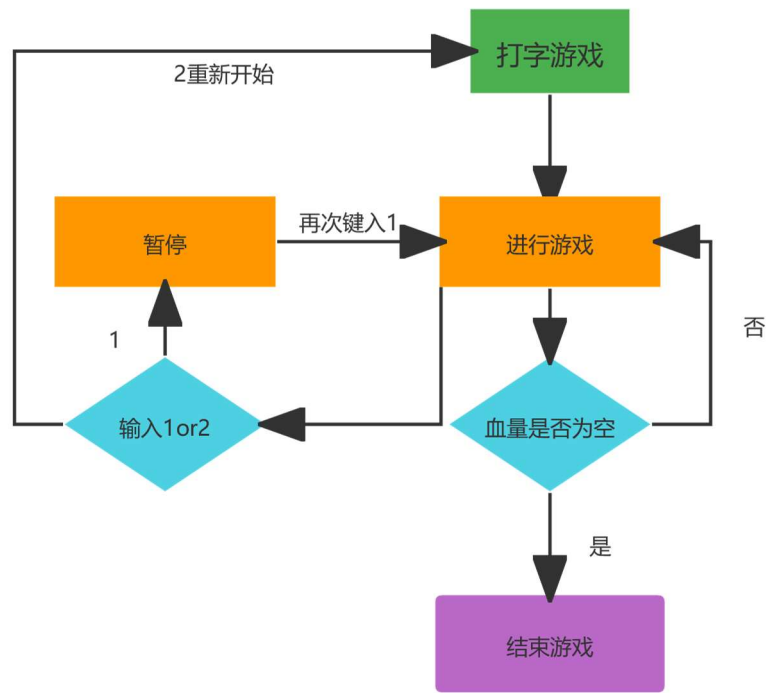
排名	用户名	完成度	文章	正确率	用时
1	Mi	98	MyFrien	100	11
2	Forever	95	Forever	94	73
3	asdf	80	Me	90	14
4	ta	92	ta	90	38
5	Talks	91	Talks	88	42
6	wzc	80	LOL	83	66
7	malo	78	Dom	80	10
8	I	58	Talks	68	23
9	Mi	95	Me	57	9
10	NEMO	90	DrugHou	24	25
11	Talks	18	Talks	5	9
12	small	99	Talks	1	4
13	small	99	Tals	1	4
14	small	99	Talks	1	4
15	sa11	99	Ilks	1	4

Navigation buttons: 上一页, 下一页, 首页, 尾页

Page info: 第1页/共2页

图表 14 排行榜界面

IV) 打字游戏



图表 15 打字游戏逻辑



图表 16 打字游戏界面

V) 单词练习



图表 17 单词练习界面

本项目对这几个功能（除退出外）分别建立了不同的 void 类型函数予以实现。

2.代码实现

具体的代码实现如下：

首先本项目通过枚举建立几个变量来分别代表这些函数：

```
enum Operator
{
    Display,//文章训练
    Add,//文章管理
    Erase,//排行榜
    Modfy,//打字游戏
    Wordpractice = 4,//单词练习
    Menu = 66 //菜单界面
};
```

然后本项目通过按钮返回上面的信息就可以在 run（）函数中来管理菜单，下面是 run（）函数中的部分代码：

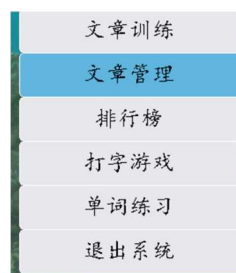

```

void Management::run()
{
    int op=66;//初始化为菜单界面的值
    switch (op)//利用 switch 分别对不同情况进行处理
    {
        case 66:
            op = mainmenu(m_msg); //只要不返回其他值，一直进行菜单页面
            break;
        case Management::Display://文章训练
            display();
            break;
        case Management::Add://文章管理
            add();
            break;
        case Management::Erase://排行榜
            erase();
            break;
        case Management::Modify://打字游戏
            modify();
            break;
        case Management::Wordpractice://单词训练
            wordpractice();
            break;
        default://如果没有点击以上按钮，则退出
            saveFile("./images/change.txt");//保存了文件，后面详解
            exit(666);
            break;
    }
}
}

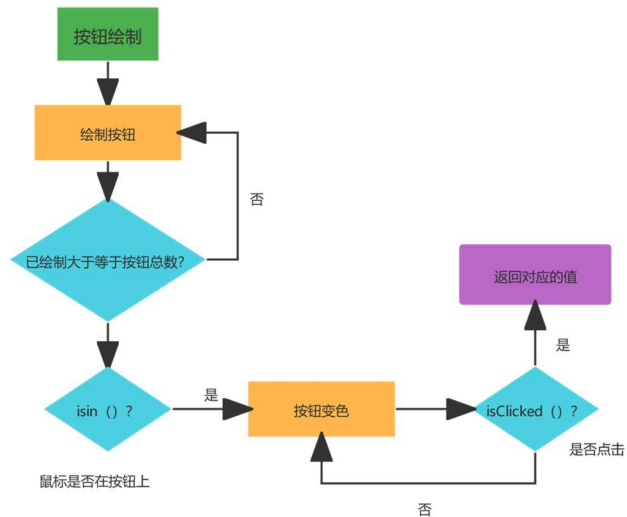
```

本项目用变量 op 来接受按钮返回的信息,后面根据 op 的不同值，利用 switch 来分别处理这些情况。当然，这段代码里含有很多函数是后续功能的函数，这里不做介绍。

1.1.2 按钮



图表 18 按钮展示



图表 19 按钮绘制逻辑

按钮的代码实现过程包括：按钮的设立、接收按钮返回的信息。

按钮的设立

在 PushButton 类中专门对按钮进行了定义。

```

PushButton(const std::string& text="Button", int x = 0, int y = 0, int
w = 100, int h = 30);
//下面的代码是 Managemeng 类中设置菜单时设置的，在这里以其中一个当作示例
menu_btns.push_back(new PushButton("文章训练"));

```

建立按钮时在这行代码中初始化了按钮的大小，以及一开始按钮上面写的文字。

```

void PushButton::setBackgroundColor(COLORREF c)
{
    normal_c = c;
}
void PushButton::setHoverColor(COLORREF c)
{
    hover_c = c;
}

```

这几行代码则是设置了按钮的颜色，以及当鼠标在按钮上时按钮的颜色，

以便于用户判断鼠标是否在按钮上。

II)接收按钮返回的信息

首先需要有一个参数来代表这个信息，本项目在 PushButton 类中用了 m_msg 来接受信息。其定义方法是 easyx 中的 ExMessage，具体如下

```

ExMessage m_msg;

```

随后专门建立 eventLoop () 函数来接受按钮信息：

```

void PushButton::eventLoop(const ExMessage& msg)
{

```

```

    m_msg = msg;
    if (!isin())//下面会对 isin () 函数进行讲解
    {
        cur_c = normal_c;//如果鼠标不在按钮上，按钮是寻常的颜色
    }
    else
    {
        cur_c = hover_c;//如果鼠标在按钮上，按钮是特殊的颜色
    }
}

```

要判断用户的鼠标是否在按钮上，这里本项目单独写了 bool 类型的 isin

() 函数来判断这一点。

```

bool PushButton::isin()
{
    if (m_msg.x >= m_x && m_msg.x < m_x + m_w && m_msg.y >= m_y &&
m_msg.y <= m_y+m_h)
    {
        return true;
    }
    return false;
}

```

下一步需要判断按钮是否被点击，同样本项目用 bool 类型的函数 isClicked () 来进行判断：

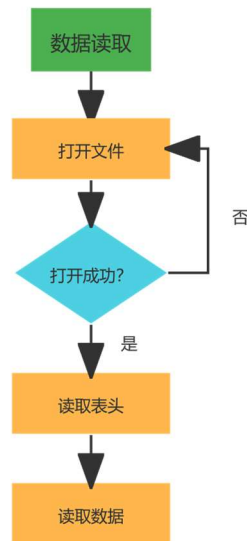
```

bool PushButton::isClicked()
{
    if (isin())
    {
        if (m_msg.message == WM_LBUTTONDOWN)
        {
            Sleep(200);
            return true;
        }
    }
    return false;
}

```

然后通过 m_msg 就可以判断按钮的信息了。

1.1.3 数据的读取



图表 20 数据读取逻辑

由于有几种文件，而文件的读取方法类似，此处以文章管理中的表格数据读取为例进行这一部分的介绍。

首先是文件中应该读取的数据类型的设置，本项目为此建立了 Student 类来设置。

```

std::string name;      //文章名称
uint32 number;        //文章字数, uint32 是 unsigned int
std::string intime;    //文章录入时间
  
```

然后在 Management 类中提前设置好读取将会用到的：

```

std::string m_header; //表头
std::vector<Student> vec_stu;
  
```

之后再 readFile () 函数中进行文件的读取：

```

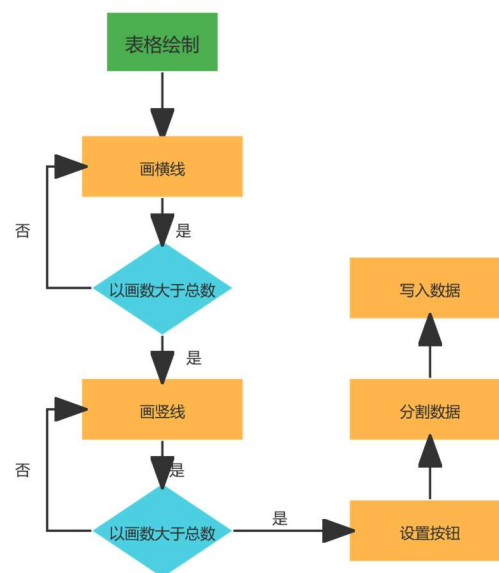
void Management::readFile(const std::string& fileName)
{
    fstream read(fileName, ios::in);
    if (!read.is_open())//判断是否成功打开文件
    {
        return;
    }
    //读取表头
    char buf[1024] = { 0 };
    read.getline(buf, 1024);
    m_header = buf;
    //开始读取数据
    while (!read.eof())
    {
        char data[1024] = { 0 };
        read.getline(data, 1024);
    }
}
  
```

```

//跳过空行
if (strlen(data) == 0)
    break;
// 格式化读取
Student stu;
stringstream ss(data);
ss >> stu.name >> stu.number >> stu.intime;
vec_stu.push_back(stu);
}
read.close();
}

```

1.1.4 表格的建立:



图表 21 表格绘制逻辑

1.表格样式:

文章名称	文章字数	录入时间
ForeverTw	283	2021年12月19日
DrugHouse	316	2021年12月19日
Talks	102	2021年12月19日
Me	41	2021年12月19日
abd	91	2021年12月19日
LOL	192	2021年12月19日

图表 22 表格样式

2.代码实现:

表格的建立包括了表格的绘制与分页处理，以及相应文件数据的导入，而本软件共绘制了两次表格，本项目以文章管理中的表格为例进行介绍。

表格的绘制与分页处理

首先得定义表格的行数与列数，表格的高度与宽度，文字的大小等基本参数，以及表头等

```
int m_rows;//行数
int m_cols;//列数

int m_gridW;//表格宽度
int m_gridH;//表格高度

int m_tw;//文字宽度
int m_th;//文字高度
```

之后根据这些基本数据，利用 easyx 图形库中的画线函数来绘制表格：

```
void Table::drawTableGrid()
{
    //画横线
    setlinecolor(BLACK);
    for (size_t i = 0; i < m_rows + 1; i++)
    {
        line(m_x, m_y+i*m_gridH, m_x+ m_cols * m_gridW, m_y+i*m_gridH);
    }
    //画竖线
    for (size_t i = 0; i < m_cols + 1; i++)
    {
        line(m_x+i*m_gridW, m_y, m_x + i * m_gridW, m_y + m_rows *
m_gridH);
    }
    drawButton();//画出按钮，作用后面详解
}
```

如此表格便绘制完毕。然而由于数据数量的问题，一页的表格肯定是无法显示全部的数据的，否则字体大小将会非常的小。因此要进行分页处理。

为了实现这个目的，首先定义：

```
int m_curPage;//当前页
int m_maxPage;//最大页
int m_extraData;//未处理的页
```

然后对数据进行分割，以便于用页数来显示：

```
std::vector<std::string> Table::split(std::string str, char separator)
{
    std::vector<std::string> res;

    for (size_t pos = 0; pos !=std::string::npos;)
    {
```

```

        pos = str.find(separator);

        res.push_back(str.substr(0,pos));

        str = std::string(str.c_str()+pos+1);
    }
    return res;
}

```

通过这些分割的数据和页数，可以一次只展现部分数据，从而达到分页处理的效果。
之后设立按钮，并将之显示出来，用于切换表格页数：

```

PushButton* m_prevBtn;
PushButton* m_nextBtn;
PushButton* m_firstBtn;
PushButton* m_LastBtn;

```

然后单独定义各个按钮被点击的情况：（此处以 m_preBtn 被点击为例）

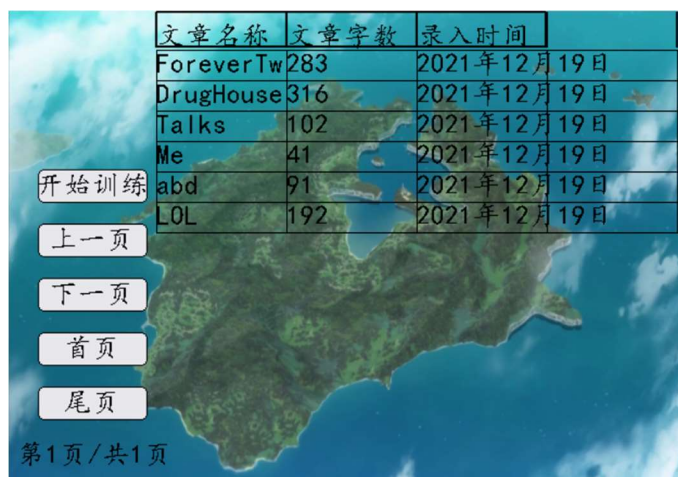
```

if (m_prevBtn->isClicked()) //如果上一页被点击
{
    if (m_curPage != 0)
    {
        m_curPage--;
    }
}

```

这样我们就可以通过 m_curPage 这个参数来代表当前页数，作为信息来显示表格数据了。这样一来即实现了表格的分页处理。

下面是成品图：



文章名称	文章字数	录入时间
ForeverTw	283	2021年12月19日
DrugHouse	316	2021年12月19日
Talks	102	2021年12月19日
Me	41	2021年12月19日
abd	91	2021年12月19日
LOL	192	2021年12月19日

开始训练

上一页

下一页

首页

尾页

第1页 / 共1页

图表 23 表格成品图

1.2 文件管理的实现

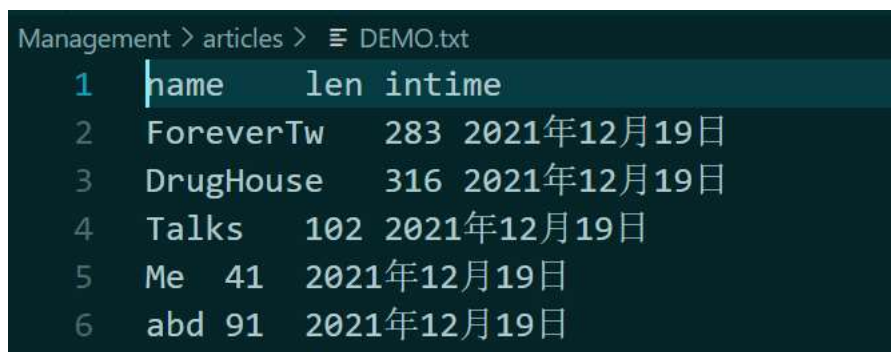
1.2.1 数据索引文件的结构设置

本软件的文件索引系统主要依托三个文本文件：

ARTICLELIST.txt, Ranking.txt, passage.txt。

1. 其中，ArticleList.txt 存储文章标题和入库时间、字数等信息；
2. 使用 Ranking.txt 存储排行榜：包括用户名和正确率等数据；
3. 而 passage.txt 存储当前需要打字的文章内容。

详细内容见下图：



1	name	len	intime
2	ForeverTw	283	2021年12月19日
3	DrugHouse	316	2021年12月19日
4	Talks	102	2021年12月19日
5	Me	41	2021年12月19日
6	abd	91	2021年12月19日

使用 ArticleList 作为文章名称文件名称的原因在于调试时使用该名称，作为调试的产物，在正式版本中就没有再修改



1	num	username	complete	passage	accuracy	time
2	1	Mi	98	MyFrien	100	11
3	2	ForeverT	95	Forever	94	73
4	3	ta	92	ta	90	38
5	4	Talks	91	Talks	88	42
6	5	T	58	Talks	68	23
7	6	Mi	95	Me	57	9
8	7	NEMO	90	DrugHou	24	25
9	8	Talks	18	Talks	5	9
10	9	small	99	Talks	1	4

在 Ranking.txt 中，文件位置位于素材库，未调整位置也是由于调试时的遗留需要。

在这两个文件当中，本项目使用了较为规范的文件数据编排方法，统一使用 Tab 键制表符来分割数据。方便上文提到的表格中数据的读取和显示。

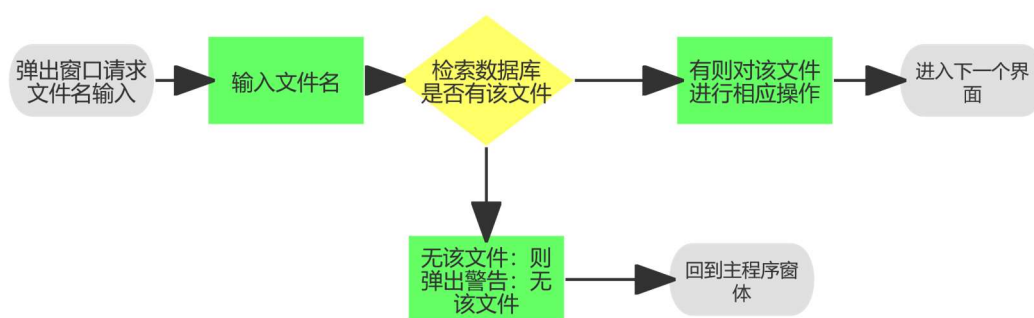
在 passage.txt 中，则存储暂时需要打字练习的文章，单独使用 passage.txt 来暂时存储当前打的文章的原因在于，打字界面读取文章时，认为 '+' 这一特殊字符为文章结尾符号。所以使用 passage 在存储文件时，单独加上 '+' 让文章读取能够更为符合打字练习软件的函数调用。


```
Management > E passage.txt
1 I don't know whether it was because work at the office slowed during February or because the football
season was over. But Valentine's Day was the time my father chose to show his love for the special
people in his life. Over the years I fondly2 thought of him as my "Valentine Man."+
```

如这篇示例文章所展示：在文章末尾使用 '+' 来作为末尾的分隔符。

1.2.2 文章索引的流程和程序实现

在对于文件数据的读取和处理以及再存储的过程中，基本思路如下：



图表 24 基本思路

在进行文件处理的过程中，基本设计如下 4 种功能，主要在处理时展现文件读写的逻辑不同，故先罗列如下，再选取其中一种进行详细的介绍：

1. 文章的增加：在检索到没有同名文章后，在 ARTICLELIST.txt 的末尾通过追加模式添加该文章的信息（包括使用 time.h 添加入库时间相关信息）；若有同名文章，则进行报错。
2. 文章的删除：在检索到有同名文件时，在 ARTICLELIST.txt 中对这条文章记录进行删除，具体做法为新建巨大的字符数组，将原文章内容除了需要删除的行数进行存入；在没有同名文件时，进行报错提醒。
3. 文章的开始训练：在检索到具有同名文件时，在文章库中找到该文章并读取文章的数据，利用大字符串数组存入相关文章信息并且追加 '+' 作为结尾符号，存入 passage.txt 中。
4. 排行榜的添加：在文章结束之后，通过变量获取训练这篇文章的：用户名、文章名、文章字数、文章完成程度、正确率这些数据。打开 Ranking.txt 后，对文件内的相关数据和排行榜中数据的异同进行比较，再在合适的位置进行数据插入处理，以上操作通过大数组完成后，重新对 Ranking.txt 进行写入。

进行如上文件操作的文件指针利用形式较为相似，但又有异同，暂且选用文章增加这一功能的代码作为范例：

```
if (add->isClicked())
{
    char name[10];
```

```

    InputBox(name, 10, "请输入想添加的文章名称(不要超过十位)");

    FILE* f1;
    f1 = fopen("./articles/ARTICLELIST.txt", "a+");
    char existnames[100];
    int flag = 1;
    bool canInput = 1;
    while (fgets(existnames, 100, f1) != NULL){
        flag = memcmp(existnames, name, strlen(name));
        if (flag == 0) {
            MessageBox(NULL, "文章名称重复, 无法录入!", "错误!", MB_OK);
            canInput = 0;
            break;
        }
    }
}

```

使用 InputBox 作为文章名称的接收入口, 通过文件指针 f1 的操作, 通过对录入数据 name 和 f1 所指向内容的比较, 判断是否有所选文章。

这样做的副好处在于: 即使输入的文章具有缺省字母的情况, 也能够该代码进行文章名称的缺省搜索。

```

    if (canInput) {
        char article[8000];
        InputBox(article, 8000, "请输入文章内容", "ArticleInput", 0,
400, 300);
        int len = strlen(article);
        char content[100];
        char tmp[20];
        time_t timep;
        struct tm* p;
        time(&timep);
        p = gmtime(&timep);
        int time[3];
        time[0] = 1 + p->tm_mon; /*获取当前月份, 范围是0-11, 所以要加1*/
        time[1] = 1900 + p->tm_year; /*获取当前年份, 从1900 开始, 所以要
加1900*/
        time[2] = p->tm_mday; //当前月份的天数
        sprintf(tmp, "%d年%d月%d日\n", time[1], time[0], time[2]);
        strcat(content, tmp);

        MessageBox(NULL, content, "输入内容", MB_OK);
        for (int kk = 0; kk < strlen(article); kk++)
            if (article[kk] == '\n' || article[kk] == '\r')
                article[kk] = ' ';
    }
}

```

```
fputs(content, f1);
fclose(f1);
```

通过打开新的 InputBox 进行文章内容的输入，并存入一个大数组当中。并且自动获取当前的时间，自动计算字数，通过 sprintf()函数写入新的字符串中，进行数据的重新录入。

```
FILE* FWRITER;
char passagearea[50] = "./articles/";
strcat(passagearea, name);
strcat(passagearea, ".txt");

FWRITER = fopen(passagearea, "w+");

fputs(article, FWRITER);
fclose(FWRITER);

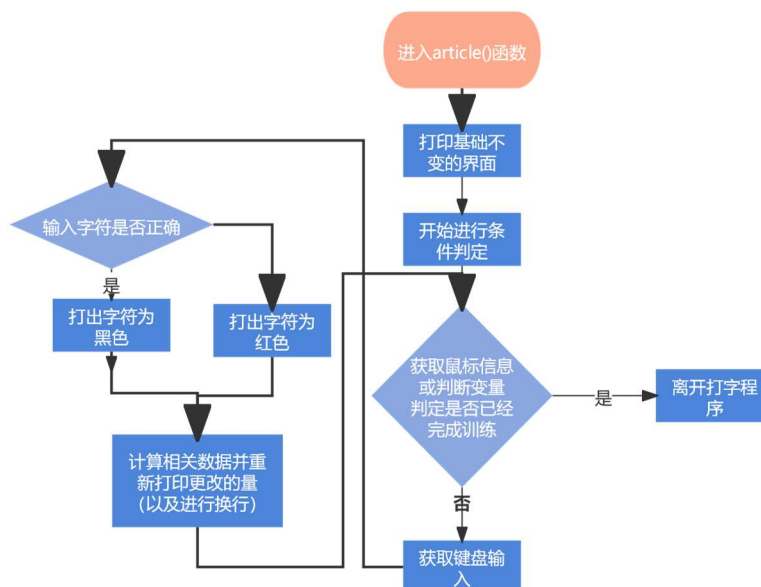
InitPage();
}
```

打开新的文件指针来对录入的文章内容进行保存。

以上操作大体为文章索引引用并添加新文件的做法，其他在文件中通过字符数组实现文件内容增删查改的方法和使用函数大致如上，仅为判定条件和存储方式和读写方式的差异，具体流程相似，不再赘述。

1.3 打字界面的实现

打字界面主要通过 article()函数进行实现，该函数的大致流程图如下：



图表 25 打字界面逻辑

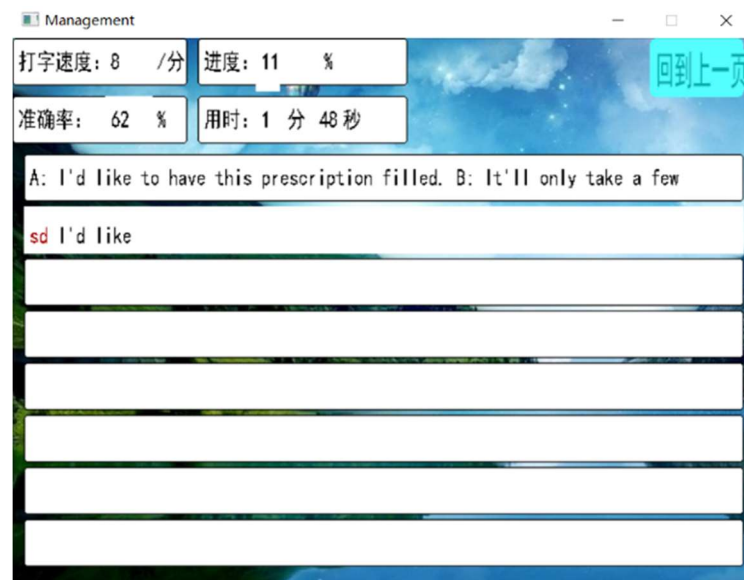
1.3.1 打字界面的绘制

```
BeginBatchDraw();
cyan(550, 0, str);
for (i1 = 0; i1 < 8; i1++) {
    others(10, 100 + i1 * 45, 630, 140 + i1 * 45, " ");
}
i1 = 0;
others(0, 0, 150, 40, "打字速度:      /分");
others(0, 50, 150, 90, "准确率:      %");
others(160, 50, 340, 90, "用时:      分      秒");
others(160, 0, 340, 40, "进度:      %      ");
EndBatchDraw();
```

打字界面主要使用 BeginBatchDraw()这一函数进行批量绘制，让绘制出的图形能够被单次刷新出，同时在图片刷新前定义好相关变量，让后续变量的调用较为快捷，避免卡顿。

其中 others()函数为特别定义的输出白色矩形的函数，同时可以在白色矩形中输入字符：后续在白色矩形中打印文章也是通过该函数进行快捷实现的。

所打印出的打字界面大致如下所示，具有清晰的文章行数分隔和显著的数据显示。



图表 26 打字界面

1.3.2 文章的录入和显示

文章的录入主要使用 filea()函数进行，该函数返回给 article () 函数一个存储文章内容，大小为 8000 的字符串数组作为文章内容存储的空间，方便后续每行文章的打印。

```
char* filea() {
    char* a;
```

```

a = (char*)malloc(80000);
int n = 0;
FILE* fp;
fp = fopen("passage.txt", "rt");
while ((a[n] = fgetc(fp)) != EOF)
{
    if (a[n] == '+')
        break;
    n++;
}
fclose(fp);
a[n] = '\0';
return a;
}

```

在获取文章内容之后，进行界面中每行内容的打印，在这里，使用 Line 字符数组来保存每行所需要容纳的字符内容，并直接使用 others 函数进行绘制：

```

for (i = 0; i < 70 && *line != '\0'; i++, line++)
    a[i] = *line;
a[70] = '\0';
if (*line == '\0')
    a[i] = '\0';
if (i1 == 4)
{
    for (i1 = 0; i1 < 8; i1++) {
        clearrectangle(10, 100 + i1 * 45, 630, 140 + i1 * 45);
        others(10, 100 + i1 * 45, 630, 140 + i1 * 45, " ");
    }
    i1 = 0;
}
others(10, 100 + 90 * i1, 630, 140 + 90 * i1, a);
clearrectangle(10, 145 + 90 * i1, 630, 185 + 90 * i1);

```

值得一提的是，在上述代码中，当打字进行到显示在界面的一半的时候，我们会重新对界面进行打印，来保证每行所打的字在软件中的显示应该是在居中或者起始位置的。

1.3.3 键盘录入的实现

```

if (kbhit()){
    s = getch();
    if (s == 8)//退格符号
    {clearrectangle(8 + 8 * i, 145 + i1 * 90, 18 + i *
8, 180 + i1 * 90);
        i = i - 2;
        q--;}else {

```

```

        q++;j++;
        if (s == a[i])
        {k++;
         letterinput(10 + 8 * i, 150 + i1 * 90,
s); }

        else
         letterinputwhite(10 + 8 * i, 150 + i1 * 90,
s); }}

```

使用 EasyX 库中提供的 kbhit () 函数判定是否有字符的键入情况：如果有，则在进行计时器操作（上述代码块省略），以及对于输入字数增加的操作后，进行输入字符的特判：如果符合该字符在文章中的位置，则进行函数：letterinput(10 + 8 * i, 150 + i1 * 90, s)，打印黑色的字母，否则则执行另外一个函数，打印错误颜色提示的字母，这里我们选用的是红色。

其中 letterinput 这个函数的写法和上文中提到的 others()矩形（带字符）输出的函数较为类似，将 letterinput 这一函数的实现逻辑附在下：

```

void letterinput(int i, int j, char le)
{
    setbkcolor(WHITE);
    setcolor(BLACK);
    LOGFONT f;
    gettextstyle(&f);
    f.LfHeight = 20;
    f.LfWidth = 8;
    _tcscpy(f.LfFaceName, _T("黑体"));
    f.LfQuality = ANTIALIASED_QUALITY;
    setttextstyle(&f);
    outtextxy(i + 5, j + 10, le);
}

```

可以看出，该函数类型主要使用的就是利用 EasyX 库中对于字体进行特定格式定义的函数先定义函数之后，对于字符进行输出：这样进行重复定义后输出虽然一定程度上会降低输出的效率、增加输出前的处理时间，但是在高速计算机的帮助下，我们可以将这一切暂时忽略。

1.3.4 打字状态的获取

1.3.4.1 时间的获取

```

timer = time(NULL);
tblock = localtime(&timer);
if ((tblock->tm_min - d1) * 60 + tblock->tm_sec - d2 >
1 && flag != 1)

```

```

        {
            clearrectangle(80, 5, 120, 35);
            sprintf(b, "%d", j * 60 / ((tbblock->tm_min - d1) *
60 + tbblock->tm_sec - d2));
            numinput(80, 0, b);
            l = l + 1;
            clearrectangle(210, 55, 230, 85);
            sprintf(b, "%d", l / 60);
            numinput(210, 50, b);
            clearrectangle(260, 55, 280, 85);
            sprintf(b, "%d", l % 60);
            numinput(260, 50, b);
        }

```

较为复杂，使用了系统提供的计时器类型：tm(在这里被重定向为 tbclock 类型)，使用 flag 来控制计时器的开和关（在上文省略打开 flag 并 sleep 让时间增加，和后文关闭 flag 防止在下一次接受前防止计时器溢出的操作），通过计时器一秒一次的计数，能够让该函数以较良好的基础格式获取秒数（利用 sprintf 等函数进行规范化输出），并且通过 numinput()这一函数进行输出，这一函数的原型和 others()长得也很像，不再赘述。

1.3.4.2 进度、正确率和状态的获取

这些量的获取较为简单，在前块代码中也有所展示：主要通过对几个变量的增加（正确字数，已经键入字数），和总字数进行比较，并进行简单处理即可输出。

其中，clearrectangle（）函数主要是绘制一个遮盖上一次所绘图形框的函数，来让画面在不进行整体刷新的情况下就可以进行新数字的绘制。

```

if (q > 0)
{
    clearrectangle(210, 5, 230, 45);
    sprintf(b, "%d", q * 100 / n);
    numinput(210, 0, b);
}

```

1.3.4.3 退出打字函数的实现

主要有 2 种退出函数的情况：

1. 完成训练正常退出
2. 点击退出按钮进行退出

这里以第一种情况的实现为例：

```

if (j == (n-1)) {
    MessageBox(NULL, "成功完成训练, 分数将计入排行榜", "成功", MB_OK);
}

```

```
accuracy = (k * 100 / j);  
finishTime = l;  
finishPercentage = (q * 100 / n);  
GetRank();  
InitPage();}
```

J 为已经键入的字数的统计变量，而 n 为总字数的统计变量（统计了'+',而不显示，所以为 n-1 即可），当这两个量满足条件时，对于准确度，完成度，结束时间进行统计，然后进行排名函数 GetRank()的调用，该函数的主要内容即为文件读写相关内容，上文有所涉及，不再赘述。

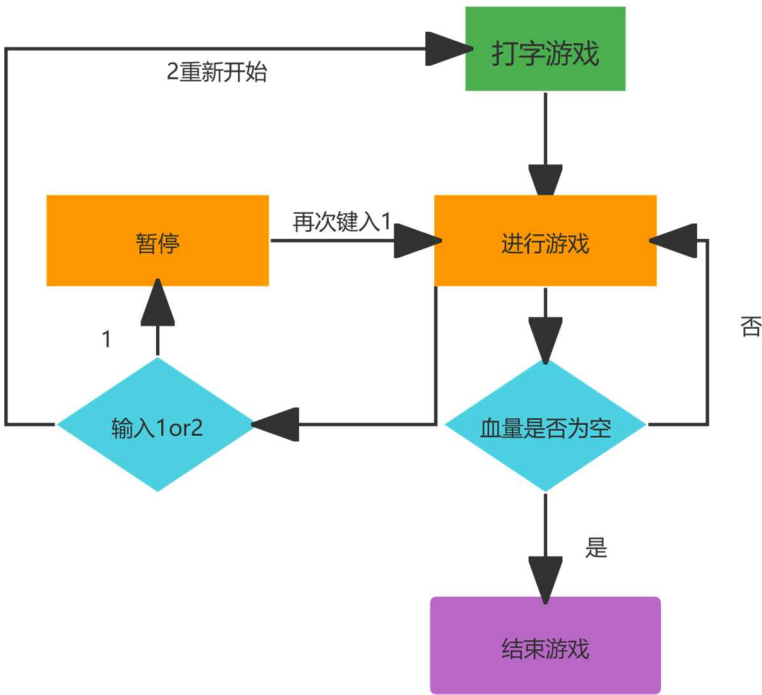
随后对于页面进行重置，使用函数 InitPage()，该函数定义如下，主要功能为新建一个主界面的对象然后重新初始化主界面：

```
void InitPage() {  
    windows w1(640, 480, EW_SHOWCONSOLE);  
    w1.setWindowTilte("ghosttyper");  
    Management m1;  
    m1.run();}
```


1.4 额外功能的实现

1.4.1 打字游戏的实现

基本介绍：



图表 27 打字游戏逻辑



图表 28 打字游戏界面

这个打字游戏类似于俄罗斯方块，会在游戏窗口上方以一定的速度落下字符，键盘上键入字符可以消除屏幕上对应的字符。如果字符落到屏幕底端而没有被消除，玩家的血量将会降低，当玩家的血量为空时，游戏结束。

当玩家正确的键入字符时，游戏会对玩家的操作进行计分，每次输入错误都会相应扣

除分数、血量，正确输入则会增加分数。

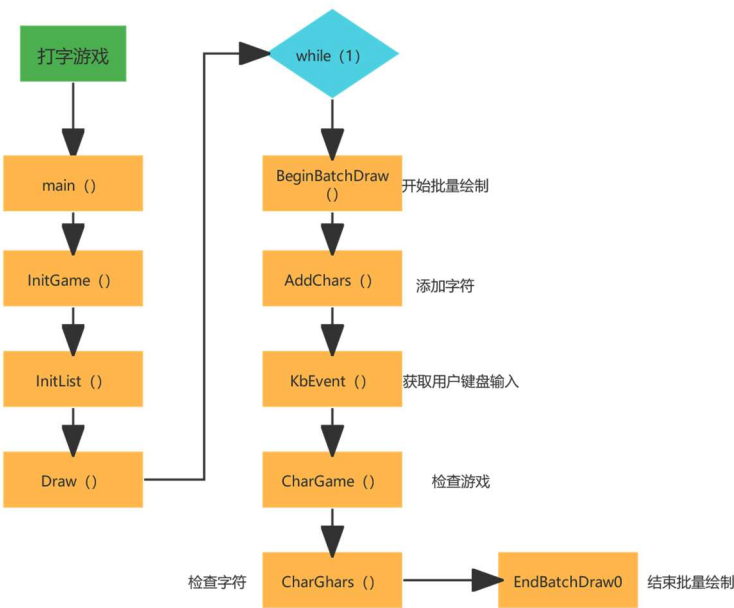
而随着游戏的进行，字符落下的速度会越来越快，对玩家的要求也就越来越高。最后当玩家结束游戏时会给出玩家本次游戏中达到的最高分，并询问玩家是否重玩，还是退出。

二、实现

以下是为了实现游戏功能而专门定义的部分函数：

```
void InitList(); //初始化顺序表
void InitGame(); //初始化游戏
void KbEvent(); //按键处理
void AddChars(); //每隔一段时间增加字符
void Draw(); //画界面
void Over(); //检查游戏是否结束
```

游戏实行内部逻辑：



图表 29 代码逻辑

首先需要字符不断地从界面顶部下降，为了实现这一点，首先要有一个字符下落的顺序表，明确接下来会下落哪个字符，因此本项目定义了 InitList () 函数。

之后要确定字符会在何时，何地下降，为此本项目定义了 AddChars () 函数。下落后对字符的顺序表进行及时的处理，以保证顺序表可以得到更新，维持游戏的持续进行，这就是 Remove ()，与 Add () 的作用。

之后我们需要收集用户的按键信息，并对按键信息做出处理，为此定义了 KbEvent()。

游戏进行过程判断完毕之后，需要判断何时结束游戏，因此建立 Over () 函数。

之后对游戏数据进行保存，为此定义 Save () 函数。

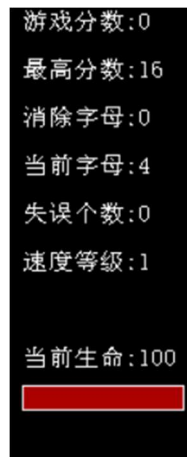
2.界面设计

为了使用户可以看到这个字符，需要绘制字符在界面上。所以本项目定义了 Draw () 函数。



图表 30 绘制的字符

通过 Draw () 用户还可以看到一些游戏的基本信息：

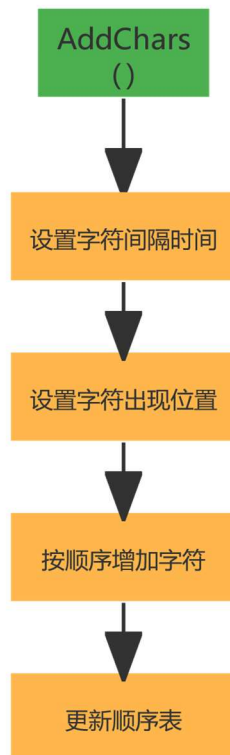


图表 31 绘制的信息框

接下来，我们具体介绍一下上面提到的主要的函数的定义。

(二) 实现函数：

1) AddChars () 函数：



图表 32 `AddChars()` 实现逻辑

该函数用来每过一段时间增加字符，也就是游戏的基本规则的实现，让字符不断地从顶部下降。

```
void AddChars()
{
    g.t2 = timeGetTime(); //当前字符下落的变化时间

    if (g.t2 - g.t1 >= ADDCHARSTIME)
    {
        g.t1 = g.t2;
        PTYPE p = (PTYPE)malloc(sizeof(TYPE));
        if (p == NULL)
        {
            MessageBox(g.hwnd, "游戏内存不足", "", MB_OK);
            exit(0);
        }

        p->c = A + rand() % 26;
        p->x = GAP + 1 + (rand() % (GAMEWIDTH - CHARR * 2 - 1)) +
(CHARR - CHARSIZE / 2);
        p->y = GAP + 1 + (CHARR - CHARSIZE / 2);
        p->color = RGB(rand() % 256, rand() % 256, rand() % 256); //颜色
    }
}
```

变化

```

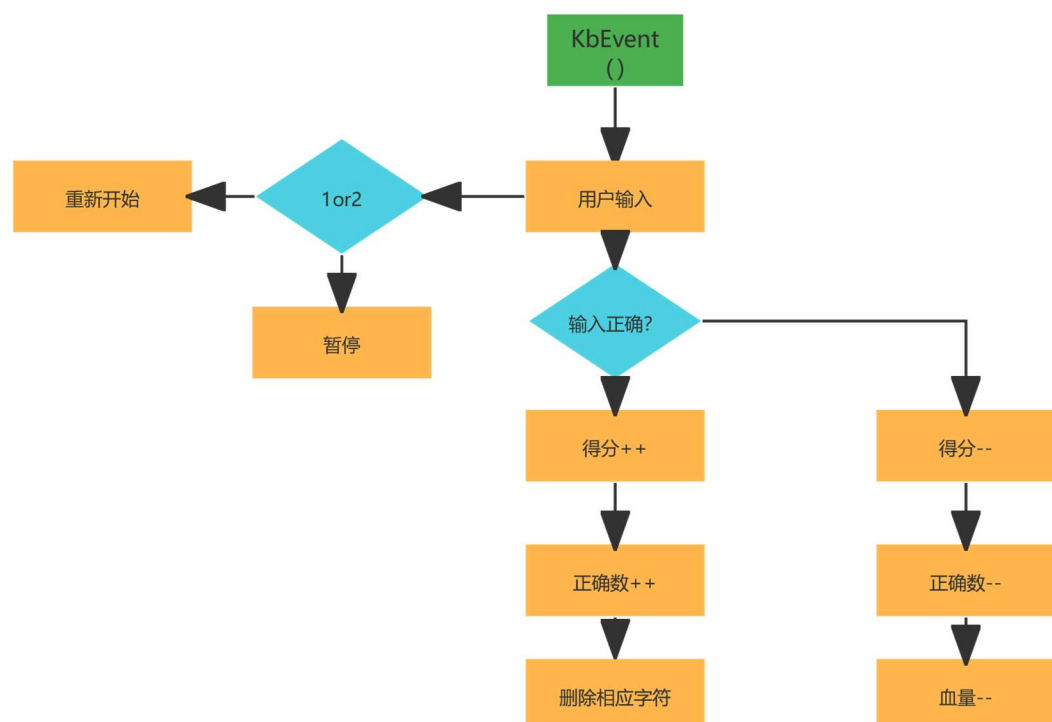
    p->t1 = timeGetTime();//间隔时间
    p->t2 = p->t1;
    p->dt = g.speed + rand() % 8;
    p->ct1 = timeGetTime();//间隔时间
    p->ct2 = p->t1;
    Add(p);//添加字符

    Draw();//绘制字符
}
}

```

首先我们获取当前字符下落变化的时间，（timeGetTime()函数是 timeapi.h 中定义的）。然后通过字符结构体指针来进行操作，最后使得字符随机在各个位置，每隔一段时间进行生成，最后通过 draw（）函数来绘制出这些字符。

II）KbEvent（）函数：



图表 33KbEvent（）逻辑

该函数用来处理用户按键的信息。

一下是部分代码

```

void KbEvent()
{
    if (kbhit())
    {
        char con = getch();//获取按键输入
        bool flag = false;    //标志此次按键是否正确
    }
}

```

```

// 暂停游戏
if (con == '1')//如果键入 1, 则会暂停游戏
{
    while (1)
    {
        ...
    }
    return;
}
// 重新开始
else if (con == '2')//如果键入 2 则会重新开始游戏
{
    ...
}

// 检查按键是否正确
for (int i = 0; i < list.size; i++)
{
    if (con == list.p[i]->c || con == list.p[i]->c + DIS)
    {
        ...
        flag = true;
        g.score++;
        g.remnum++;
        Remove(i);
        break;
    }
}
// 按键不正确的处理办法
if (!flag)
{
    ...
    g.life--;//生命减少
    g.errnum++;//错误增加
    g.score--;//分数减少
    if (g.life < 0)//生命小于 0 时, 变为 0
    {
        g.life = 0;
    }
    if (g.score < 0)//分数小于 0 时, 变为 0, 即不存在负值
    {
        g.score = 0;
    }
}

```

```

        Over();
    }
}

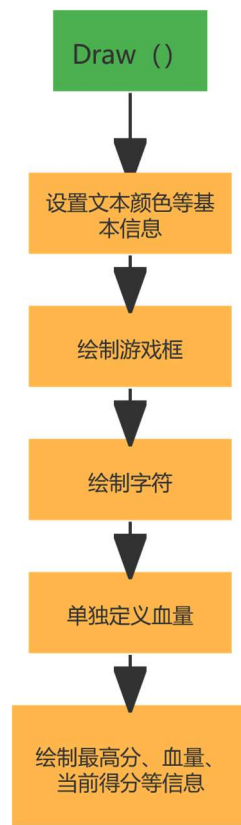
```

开始时我们通过 col 变量来接受用户的按键输入，然后用 bool 类型变量 flag 来判断按键是否正确。

然后开始暂停游戏与重新开始部分的编写，这里按 1 的话就会等待用户下一次输入，如果不是 1 就一直循环，这样就暂停了游戏。如果按下 2，则会提示用户是否重新开始，若重新开始就进行重新初始化，重新开始游戏。

接着我们检查按键是否是正确的，即是否与屏幕上已经有的字符匹配。这里通过一个循环，判断信息是否正确的时候，对分数，正确数进行了变动，随后消除该字符。

III) Draw () 函数



图表 34 Draw () 逻辑

本函数用于画出字符，游戏分数，最高分数等游戏基本内容。

```

void Draw()
{
    char str[200];    //临时数组
    cleardevice();
    setlinecolor(WHITE);
}

```

```

rectangle(GAP, GAP, GAP + GAMEWIDTH, GAP + GAMEHEIGHT); //画游戏框

//输出字母
LOGFONT f;
COLORREF c;
gettextstyle(&f);
c = gettextcolor();
for (int i = 0; i < list.size; i++)
{
    settextstyle(CHARSIZE, CHARSIZE, "宋体");
    settextcolor(list.p[i]->color);
    sprintf(str, "%c", list.p[i]->c);
    outtextxy(list.p[i]->x, list.p[i]->y, str);
    circle(list.p[i]->x + CHARSIZE / 2, list.p[i]->y + CHARSIZE /
2, CHARR);
}
//settextstyle(&f);
settextstyle(15, 8, "宋体");
settextcolor(WHITE);

rectangle(GAP * 2 + GAMEWIDTH, GAP, GAP * 2 + GAMEWIDTH +
INFORWIDTH, GAP + INFORHEIGHT);

sprintf(str, "游戏分数:%d", g.score);
outtextxy(GAP * 2 + GAMEWIDTH + 10, GAP + 10, str);

.....

//当前生命
sprintf(str, "当前生命:%d", g.Life);
outtextxy(GAP * 2 + GAMEWIDTH + 10, GAP + 220, str);
setfillcolor(RED);
solidrectangle(GAP * 2 + GAMEWIDTH + 10, GAP + 245, GAP * 2 +
GAMEWIDTH + 10 + g.Life, GAP + 260);
rectangle(GAP * 2 + GAMEWIDTH + 10, GAP + 245, GAP * 2 + GAMEWIDTH
+ 10 + 100, GAP + 260);
}

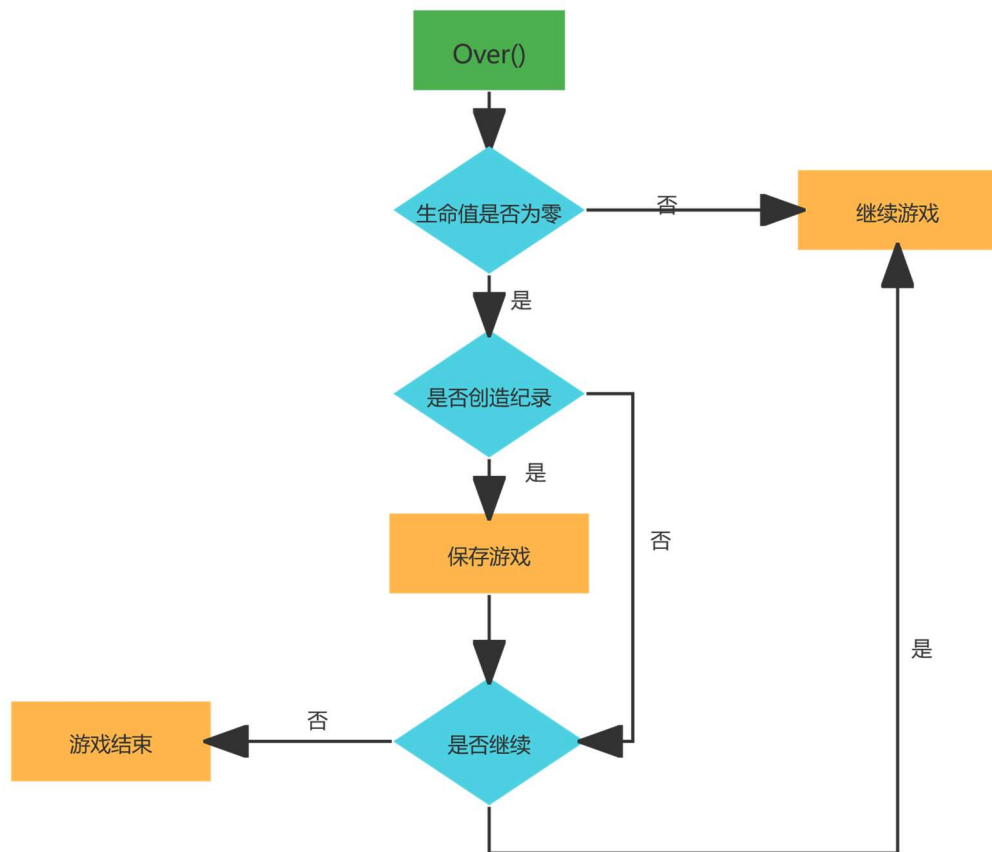
```

首先我们建立一个临时数组用于输出，之后设置划线的颜色，然后画出矩形框作为游戏的信息框。

然后我们对字母进行输出，绘制，然后再信息框内将游戏分数等基本信息进行输出。

然后绘制当前生命条，用红色的小矩形来直观地代替数字，当生命减少时，生命条的长度相应地减短。

IV) Over () 函数



图表 35Over（）逻辑

该函数用于判断游戏是否结束。

```

void Over()
{
    //如果生命为0
    if (g.life == 0)
    {
        mciSendString("close BGM", 0, 0, 0);
        mciSendString("close defeat", 0, 0, 0);
        mciSendString("open ../res/defeat.mp3 alias defeat", 0, 0, 0);
        mciSendString("play defeat", 0, 0, 0);

        //如果创造新记录
        if (g.score > g.maxscore)
        {
            g.maxscore = g.score;
            Save();
        }

        Draw();
        EndBatchDraw();
    }
}
  
```

```

    MessageBox(g.hwnd, "游戏结束!", "", MB_OK);
    int ID = MessageBox(g.hwnd, "是否重新开始? ", "", MB_YESNO);

    if (ID == IDNO)
    {
        Des();
        mciSendString("close BGM", 0, 0,
0);
        closegraph();

        windows w1(640, 480, EW_SHOWCONSOLE);
        w1.setWindowTilte("ghosttyper");
        Management m1;
        m1.run();
        m1.run();
    }
    else if (ID == IDYES)
    {
        Des();
        InitList();
        InitGame();
        Draw();
    }
}
}
}

```

判断游戏是否结束的方法是判断生命值。

如果生命值为零，则游戏结束，这种情况下如果创造了记录，则会把当前的分数值赋给最高分数，并且保存这个数据。

在结束游戏后会询问“是否重新开始”，来等待用户输入，若选择重新开始，则会将函数重新运行一遍，从而重启游戏。

在打字游戏的实现过程中，我们同时还实现了声音的播放功能，具有消除提示音和背景音乐以及错误判定的提示声音，具体原理是依靠 mciSendString()函数，比较简单，不再赘述。

1.4.2 单词训练的实现

本模块使用的功能和打字训练几乎完全一致，只不过额外打印了屏幕中的虚拟键盘，现将虚拟键盘的打印函数部分简要附在下：

```

void keyboard()
{
    num(5, 238, '`', '~');
    num(47, 238, '!', '1');
}

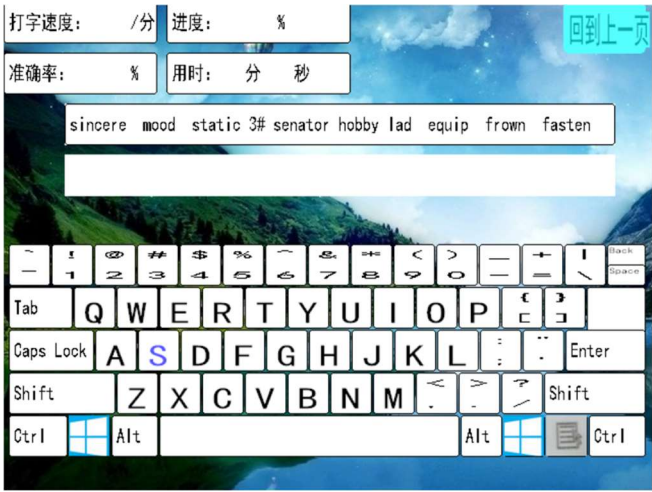
```

```
num(89, 238, '@', '2');
num(131, 238, '#', '3');
num(173, 238, '$', '4');
num(215, 238, '%', '5');
num(257, 238, '^', '6');
.....
}
```

主要功能即是在屏幕上将各个字符分别按照计算好的位置打印出来，num()函数为字符打印函数，逻辑同 others（）函数。

同时，打印单词训练的时候，追加对于字符预测的判定，让下一个需要打印的字母在键盘的显示中被打印为蓝色；打完后重新绘制为白色，代码简单，不再赘述。

最终成品见下图：



图表 36 单词训练界面

三、项目开发进度设置

表 1 项目开发进度表

时间		周数	进度内容
11.1	11.13	10-11	备战其他科目的同时自学 EasyX 及其他
11.13	11.28	11-13	分别进行 UI 界面和基础内容的编写
11.29	12.05	14	进行代码的合并和调试 编写预留接口并做其余内容前瞻
12.05	12.19	15-16	编写附加功能：如小游戏

项目按照进度开发并且在主程序完成之后撰写报告顺利。

评价、经验与总结

一、项目的最终完成情况

各部分程序完成良好，具体程序名称及代码行数*见上述完成过程。

本项目设计所需要达到的基本功能全部达成，包括：基础的文章训练功能，文章的增删查改功能，排行榜的功能。

同时达成部分进阶功能，包括：较有趣味性的打字游戏功能、具有引入性质的单词联系功能。

四、项目的优势

具有可视化界面和较低的上手难度，以及本地化的排行榜功能和文章管理功能，这是我和商业化竞争产品之间的优势。

同时基本达成商业化产品相当的界面。

三、项目的各方面评价

3.1 生产率评价

两位同学齐心协力，效率较高，能够在规定时间内完成本项目的编程任务。

3.2 技术方案评价

应用 EasyX 图形库结束进行可视化编程，同时引入 C++ 面向对象化的编程，以及较为复杂的字符串处理和判定工作标准。

在使用的技术上较为简单，但是模块之间满足高内聚、低耦合的接口标准。

在全局变量的使用上较为慎重。结合 C++ 面向对象的编程手段和 vector 容器类，能够根据文章列表信息，进行内存的动态分配。

3.3 产品质量评价

本项目采用 EasyX 进行面向对象的界面搭建，同时使用磁盘上的文件作为数据的索引文件和资源文件，在数据的安全性方面较高，不易因为断电丢失数据。同时在界面的美化方面，采用圆角矩形案件，精心在不同界面挑选不同字体，美观程度较高。和商业产品相

比，具有全本地化的优势，同时基本实现商用软件基础功能。

在经过多为同学的测试中，已排除除了设计缺陷外的其余 bug，能够满足各种规范的输入输出，甚至能够满足缺省文件名的输入和输出情况，可靠性在本项目试用成员的测试下还是较高的。

3.4 产品成品的展示

本项目已经通过 VS2022 自带的打包工具打包了一份可以直接安装的.msi 文件，可以通过安装该文件直接体验该项目。

同时，本项目组的同学，为该项目撰写了一份 README.md 文档，并且渲染了对应的.html 文件，方便后续使用者进行第一次使用和对项目的文件结构，使用方法，以及编译条件等有大体的了解。

项目的成品文件直接在：.\GhostTyper\GhostTyper.exe 就可以打开，直接放在项目目录的原因在于，需要连接到的资源文件均在该目录下，如果放在输出目录中则无法连接到资源文件，其他打开方式详见 readme.md 文档。