一、组员分工

小组序号 M

组长　章门　55180220
组员　李墨馨 55180105
**　　王泽辰 55180212**
**　　魏来　55180223**

章门(28%)
　基本框架：从文件中读取数据，建立基本数据库（配件、供货商等）
　　数据打印函数，错误判断与标记函数
　记录的【修改】【删除】功能（将错误记录予以退账）
　【单价与赠品动态变动】功能
　【存储多个历史记录】功能

李墨馨(24%)
　记录的【查询】功能（按时间、配件、供货商等进行查询）
　【模糊查询】功能

王泽辰(24%)
　记录的【增加】功能（从文件中增加、从程序中增加）
　【价格预测】功能

魏来(24%)
　记录的【统计】功能（生成库存报表、统计赠品）
　【进货建议】功能

其余零碎的小功能（如按照时间排序、计算平均价格等）在制作过程中由四个人合作随时补充。

二、程序结构

1.模块划分

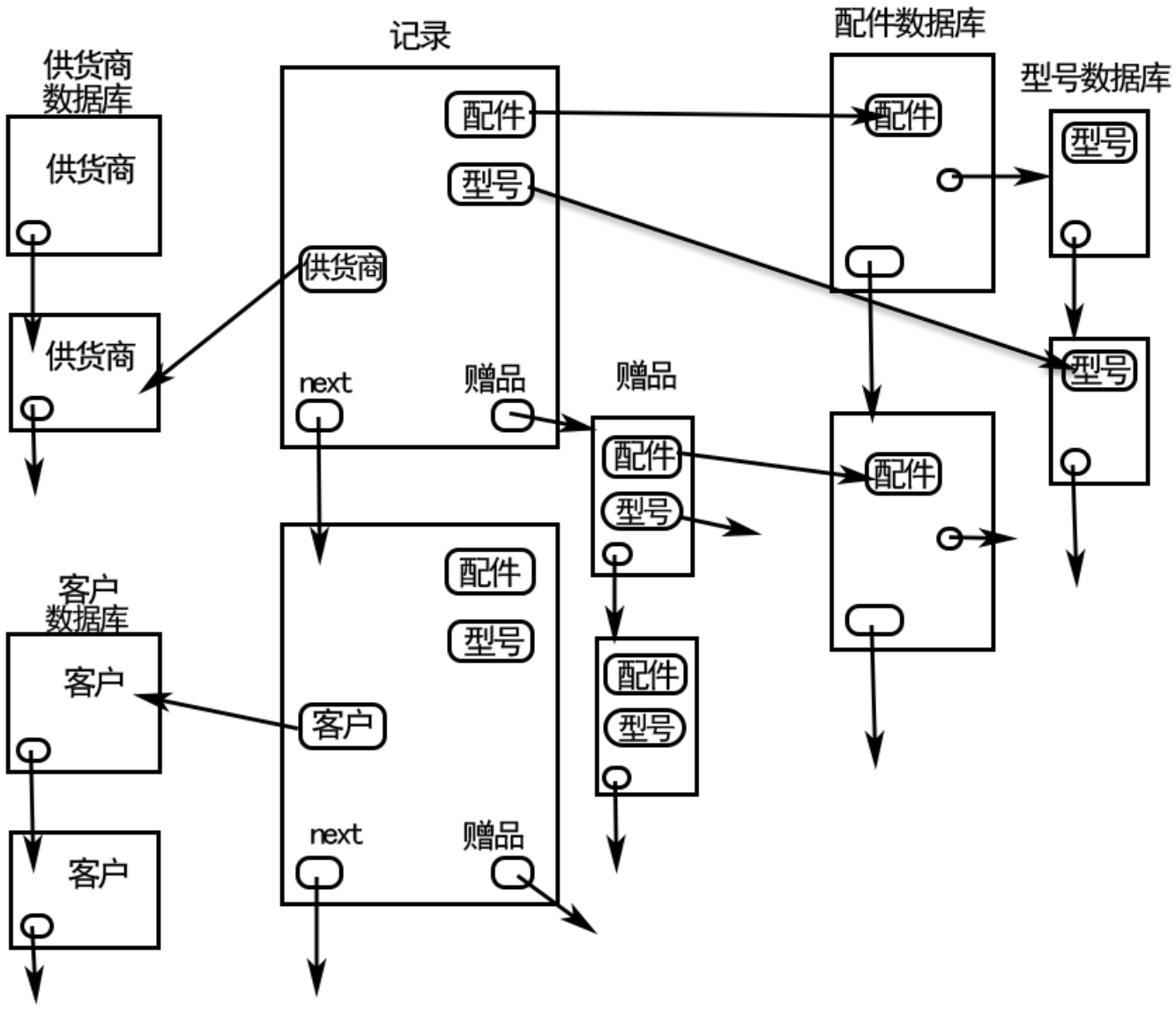
头文件一共有 4 个，分别是：
　structtype.h（结构体和全局变量的定义）
　smallfunction.h（零碎的小函数的声明，如输出函数、错误判断函数等）
　mainfunction.h（实现主要功能的函数声明）
　interface.h（实现界面的函数声明）
　目录下有同名的.c 源文件，是对应头文件的实现。
　main 函数在 main.c 源文件中。

主体的记录结构体 Record 的主要成员如下(部分成员未列出)：

```
Recordtype recordtype;      //枚举类型:空,进货,批发,零售
Date b_date;                //日期
```

```
Part *b_part;                //配件指针
Parttype *b_parttype;        //配件型号指针
Supply *b_supply;            //供货商指针
Client *b_client;            //客户指针
int amount;                  //数量
double unit_price;           //单价
double total_price;          //总价，由数量*单价即时计算得出
Present *b_present;          //指向赠品链表头的指针
int color;                   //颜色，默认为0，标记该条记录是否出错
Record *b_recordnext;        //指向下一个结点的指针
```

能够反映其主要结构的图表如下：



链表大致分为两大类，一类是【可变记录】，一类是【基础数据】。

如上所示，Record 和 Present 就属于【可变记录】，除了单价数量等，其余的信息，例如配件、配件型号、供货商等，在 Record 中都是存储相应【基础数据】中信息的【地址】。因而，程序的模块划分大致就是针对这两大类的数据进行区别操作。

(1) 文件读写

【读写可变记录】

【读写基础数据】包含读写【配件】【供货商】【客户】等

【存储历史记录】

(2) 记录的查询

【普通查询】包括按照【配件】、【供货商】、【时间范围】等标准进行查询。

【模糊查询】

(3) 记录的更改

【增加】包含【从文件中读取完整信息】和【从控制台中进行实时贸易】

【修改】【删除】【结算】

【错误检测】

【赠品动态变化】

(4) 记录的统计与测算

【统计营业额与盈利额】【统计赠品】

【生成库存报表】【价格预测】【进货建议】

(5) 基础数据的管理

【增加】【删除】【价格动态变化】

(6) 用户可视化操作（界面）

【数据打印】包括【记录】【配件】【供货商】等数据的打印

【数据选择】对打印出的数据进行选择

2.主函数的主要功能

用于索引的函数（存于 interface.c）：

```
void main_title();                  //主菜单
void searchfunc();                  //1.查询记录
void change_record();               //2.更改记录
void add_record();                  //2.1.增加记录
void trans_record();                //2.2.修改记录
void trans_one_record(Record *p);   //2.2 修改记录项目
void del_record();                  //2.3 删除记录
void count_data();                  //3.统计数据
void smart_analyse();               //4.智能分析
void price_predict_choose();        //4.1.价格预测选择
void high_change();                 //5.高级设置
void high_change_supply();          //5.1.供货商管理菜单
void high_change_client();          //5.2.客户管理菜单
void high_change_del();             //5.3.清空记录
void high_change_replace();         //5.4.替换记录
```

部分主要功能函数（存于 mainfunction.c）

```
load 系列函数                                  //从文件中读取配件客户记录等
```

```
save 系列函数                                    //将配件客户记录等存入文件中
add 系列函数                                     //从文件/控制台增加记录
change 系列函数                                  //修改记录的各种属性
search 系列函数                                  //以时间等各种标准查询记录
void history_file();                             //保存历史记录
void del_one_record(Record *key);                //删除一条记录
void store_form(Part_list* p);                   //生成库存报表
void present_form(Record *p);                    //统计所有赠品情况
void fuzzysearch();                              //模糊查询
void unit_price_change();                        //单价的动态变化
void suggest_in();                               //进货建议
void price_predict(Part* key_part,Parttype* key_parttype);
                                                 //价格预测
void del_all();                                  //删除所有记录
```

3.各子函数主要功能（存于 smallfunction.c）

```
output 系列函数                                  //打印函数
key 系列函数                                     //根据名称返回对应结点的地址
void sorttime(Record *p);                        //按照时间顺序插入指定结点
void dropoff(Record *p);                         //将某一节点摘下而不删除
void solve_record(Record *key_record,int mode);
                                                 //结算记录
int judge_choice(int maxid);                     //对输入的编号进行判断
int error_judge();                               //输入错误判断
void wait_enter();                               //操作成功，等待
void strtrans(Record *p,char *recstr);//记录转为字符串（用于模糊查询）
int KmpSearch(char* s, char* p);                 //KMP 算法（用于模糊查询）
double all_sell_amount(Part *keypart);//计算不同型号产品的总销售数量
double total_sales(Record *p);                   //从记录中求销售总额
double total_profit(Record* p);                  //从记录中求总盈利
void linear_regression(double x[], double y[], double *b, double *a);
                                                 //线性回归方程（用于价格预测）
int judgedate(Date date);                        //判断给定日期是否合法
int timecmp(Date t1,Date t2);                    //比较两个日期的先后
void timestr(char *savestr,Date date,int mode);
                                                 //将日期转换为字符串
```

4.函数之间的调用关系

大致调用关系为：打印->选择->处理->检测。

以执行一次【修改】功能，修改【时间】与【单价】为例，

(1)通过【索引函数】，选择修改功能。

(2)执行【打印函数】，打印已有的记录。

(3)用户输入数字编号，通过【选择函数】定位到相应记录的指针。

(4)传入相应记录的指针，执行【修改函数的索引函数】，选择修改【时间】项目。

(5)执行【摘下函数】，将该结点从链表上摘下，再执行【按时间插入】函数，

从而确保链表按时间有序。

(6)修改成功，返回上一步对项目的索引。不需要执行【结算函数】，因为时间的改变不影响该条记录对金钱和库存的影响。

(7)继续执行【修改函数的索引函数】，选择修改【单价】项目。

(8)修改前，执行【结算函数】，将该条记录对金钱和库存带来的影响抹除。

(9)修改成功，执行【结算函数】，增加修改后的记录对金钱和库存带来的影响。

(10)下一次执行打印函数时，执行【错误检测】函数，标记错误的记录。

三、设计思想

1.总体思路

程序的整体思路为：

【读取文件】->【数据处理】->【写入文件】

【数据处理】可以细化为：

【用户操作】<-->【数据组织】

关于【数据组织】，在这个程序中，我们将其划分为两大类：

【可变记录】与【基础数据】

可变记录，在这个程序中就是 Record 结构体包含的内容。可以看出，这个结构体大部分成员都是指针。

基础数据则是指针所指向的内容，存储着基础的供货商、客户等信息。

2.各模块总体思想

(1)文件读写

文件读写包括对可变记录的读写和对基础数据的读写。无论是可变数据还是基础数据，都需要在程序启动的时候执行读取函数，读取一组【初始数据】，建立链表。

对基础数据的改写需要慎重。举例而言，若基础数据中存在供货商 A，而某条记录中的供货商指针也指向了 A，那么程序会在客户删除供货商 A 时会提示无法删除，以此防止错误。

【存储历史记录】功能，会在用户每次进行存储的时候，额外备份一份 historyX.txt 的文档存于 save 文件夹。每存储一次就会进行一次备份，方便用户对数据进行恢复。

(2)记录的查询

查询功能在写的时候结合打印函数，编程的效率较高。

关于【按时间段查询】这一功能，在前后设计时进行了改进。当用户输入了相反的时间，那么系统不会判其错误输入，而是按照反序的时间进行查询，使得程序更加人性化。

(3)记录的更改

增加删除修改这三种操作都会对当前的资金和对应配件的库存造成影响，需要一个统一的函数对删除或修改的原数据进行退还，对新增数据进行增加。我们程序中将这个操作命名为【结算】。

其中，记录的更改可能会导致错误，比如资金不够，库存不够等。程序仅仅对其进行【错误检测】的操作，并不禁止这一行为，因为用户可能并不按照时间顺序录入记录。

同时，错误检测应该考虑到“当前时间点”下的记录是否合法，例如，在6月1日进货100件A，此时库存100。然后，用户录入在5月1日卖出100件A，此时A的库存为0，但实际上并不合法。因为5月1日卖出的A不可能来自6月1日的进货，因此5月1日的卖出属于不合法的记录。程序对于这种不合法的记录也可以检测并标记。

标记错误的记录将不会用作预测、进货建议等功能的计算数据。

【赠品动态变化】功能，会基于批发商品的量是否超过某个确定值（这个值keynum记录在配件型号的结构体中）。如果超过，会赠送赠品。倘若该件商品的库存较多，说明这件商品供过于求，商家会通过赠送更多的赠品来吸引客户批发。在这种情况下，赠品数量会增加。

(4)记录的统计与测算

【输出库存报表】可以直接模仿【打印】函数快速写出。

【预测】功能采取了简单的一元线性回归方程，但实施过程中需要考虑到数据为0个或者为1个的情况。

【进货建议】功能从两个角度给出了进货建议：获利最高的商品建议进货，价格降至较低的商品建议囤货。

(5)基础数据的管理

如上所述，基础数据会在程序运行的一开始读取一组初始数据建立初始链表。程序中有几个全局变量专门存放这些基础数据的头指针，方便随时调用。

【价格动态变化】放在基础数据这一栏。关于价格，我们的程序是通过这样的公式计算出最终价格的：

最终价格 = 基础价格 * 价格动态变化参数 * 客户/供货商参数

采取这样的形式，可以避免价格变化得离谱的现象（限制了参数最低为0.75，最高为1.5），始终围绕着不变的基础价格在一定范围内波动。

客户/供货商参数由每个客户或供货商决定，而价格动态变化参数则是根据以下标准：

若该种配件的某种型号的出售量高于该种配件所有型号的平均售出量，则说明这种型号的商品卖得好，供不应求，价格上涨。

基于以上标准，价格变动参数的计算公式为：

价格变动参数 = 该种配件的售出量 / 该种配件的平均售出量

参数不低于0.75，不高于1.5，由此便可以模拟出价格变动的情况。

(6)用户的可视化操作

【数据打印】和【数据选择】这两种功能结合得比较紧密。大体思路是将链表存入一个临时的指针数组，从而可以让用户输入编号进行随机访问。这一操作在本程序中使用十分频繁，但仅需要两三行代码，这将在后文放出。

3.界面设计

(1)采取层进式的界面，从而使用户可以快速索引到自己想要的功能。

(2)每到下一层功能都用一个新的函数，避免多层循环嵌套。

(3)选择恰当时机使用清屏函数，尽可能让每个功能在执行时都能保持控制台的简洁。

(4)与界面相关的函数全部放在interface.c中，便于修改。

4.特色设计

(1)错误数据使用绿色标记。前文已提到记录可能会互相矛盾而导致出错。那么在打印函数中会执行【错误判断】函数，对不合法的数据进行【标记】。而

标记后的函数会在打印时变成绿色，这样能醒目地提醒客户去修改与完善。

(2)模糊查询功能。这一功能使用 KMP 算法，将一条记录转为字符串然后进行匹配。若部分匹配成功则输出该条记录。

(3)高级设置功能，如【替换记录】等，方便开发者进行调试。

四、关键代码

1.数据打印与数据选择（例：选择记录进行修改）

```
maxrecord=output_record(choose_record);
printf("请选择你要修改的记录编号，输入%d 返回上一步\n",\
      maxrecord+1);
choose_id=judge_choice(maxrecord+1);
```

choose_record 是一个指针数组，作为 output_record 函数的参数。

output 函数会在打印记录的同时，把记录的地址存入传入的数组中去。

output 函数的返回值是记录链表结点的数量。

judge_choice(X)函数是让用户输入 1 到 X 范围内的整数，返回值为用户输入的数,该函数结合了错误输入判断。当检测到用户输入 maxrecord+1 时 return，返回上一层的菜单。

2.层进式菜单的设计（例：更改记录的菜单）

```
void change_record()
{
    while(1)
    {
        system("cls");//清屏
        output_money();
        printf("1.增加记录\n2.删除记录\n3.修改记录\n4.上一步\n");
        int option_choice=judge_choice(4);
        switch(option_choice)
        {
        case 1:
            add_record();//增加记录
            break;
        case 2:
            if(recordall->b_recordnext==NULL)
            {
                printf("当前记录为空，无法删除！");
                wait_enter(0);//等待按下回车后继续
                break;
            }
            del_record();//删除记录
            break;
        case 3:
            if(recordall->b_recordnext==NULL)
            {
```

```
                printf("当前记录为空，无法修改！");
                wait_enter(0);
                break;
            }
            trans_record();//修改记录
            break;
        case 4:
            return;//返回上一步
        }
    }
}
```

这段代码在外层用 while(1)进行循环，在内侧用 break 代表重新进行本次操作，用 return 则代表返回上一层操作。

同时，对记录的判空放在外层，而不放在功能函数内，减轻了功能函数的负担。

3.错误标记打印为绿色

打印函数中，有一个【打印一条记录】的函数，对本函数进行简单修改就可以完成这样的效果。

```
if(head->color==1)
{
    color_change(10);//如果记录不合法，则输出绿色
}
……//这些是函数的主要内容
color_change(7);//把颜色改回来
```

而错误判断与标记的过程，是在每次【打印记录】前进行。这样不管是在查询记录还是修改，都可以看到绿色。同时，在主菜单也会进行错误判断。倘若当前的记录在主菜单有错误，会提示用户当前的记录有误。

五、程序测试

1.测试计划

(1)测试程序的健壮性

对于每一次让用户输入编号选择的情况，尝试输入以下类型的数据：

范围小于 1 或者大于编号最大值的数字

杂乱无章的字符

对于要求输入字符串的情况，尝试输入以下类型数据：

超长的字符串

(2)测试文件读写的准确性

对于读写文件的情况，尝试在文件中录入以下类型数据：

含有不合法的日期

数据库中不存在的供货商/客户/配件

(3)测试基本功能的有效性

对于增加功能，让进货超过库存

对于删除功能，把记录全部删除

对于修改功能，修改所有的项目

对于查询功能，用所有的标准进行查询，检测其准确性

对于统计功能，用所有的标准进行统计，检测其准确性

(4)测试界面的人性化程度

尝试通过主菜单索引所有的功能

尝试在所有功能执行结束后返回主菜单

2.测试结果

(1)输入超长字符串时，输出表格格式错乱。

(2)文件读写时，不合法的日期会被录入程序中。

(3)当出现不存在的供货商/客户/配件时，程序会崩溃。

(4)删除赠品后，输出记录会导致程序崩溃。

(5)把批发修改为进货时，赠品保留。

(6)多次重复查询时，会输出重复的记录。

(7)生成库存报表时，每种配件只会显示一种。

(8)部分功能在执行完成后，提示文本显示一瞬间就消失了。

(9)操作失败后，仍显示“操作成功”的字样

(10)用空格分隔多个编号时，按下一次回车会进行多个选择步骤。

3.测试结果分析

(1)输入超长字符串时，输出表格格式错乱。

原因分析：录入的字符串太长，导致表格格式错乱。

解决方案：录入字符串时，对字符串的长度作出限制，超出限制的字符串将提示让用户重新录入。

(2)文件读写时，不合法的日期会被录入程序中。

原因分析：不合法的日期被直接读入，没有进行判断。

解决方案：读入日期的 4 个属性（月日时分）后对日期合法性进行判断，若不合法则停止录入。

(3)当出现不存在的供货商/客户/配件时，程序会崩溃。

原因分析：录入不存在的供货商/客户时，无法从数据库中检索到，记录中的指针是无效指针。

解决方案：在通过字符串访问基本数据的函数中，倘若无法访问到相应的基本数据，则返回值为 NULL。在录入函数中，判断若访问函数返回的指针为 NULL，则停止录入，并予以错误提示。

(4)删除赠品后，输出记录会导致程序崩溃。

原因分析：赠品删除时，未初始化头结点。

解决方案：初始化头结点。

(5)把批发修改为进货时，赠品保留。

原因分析：没有考虑到要求中的赠品只有在“批发”的情况下才可能出现。

解决方案：当记录由批发改为其他类型时，删除所有的赠品。

(6)多次重复查询时，会输出重复的记录。

原因分析：用于计数的变量在每次查询时累积增加。

解决方案：在每次查询时对计数变量进行初始化。

(7)生成库存报表时，每种配件只会显示一种。

原因分析：只遍历了配件，没有对每种配件的配件型号进行遍历。

解决方案：新设置一个配件型号类型的临时变量，对配件型号进行遍历。

(8)部分功能在执行完成后，提示文本显示一瞬间就消失了

原因分析：显示文本后直接 return 到上一层。

解决方案：在显示文本后增加 wait_enter 暂停函数。

(9)操作失败后，仍显示“操作成功”的字样

原因分析：在操作后直接打印了操作成功，没有考虑到操作失败的文本。

解决方案：将“操作成功”改为“操作结束”。

(10)用空格分隔多个编号时，按下一次回车会进行多个选择步骤。

原因分析：多余的数据存入缓冲区。

解决方案：用 fflush(stdin)清空缓冲区。

4.测试结论

对该程序中一切功能的测试都要从可变记录与基本数据两方面进行考虑。比如，增加记录时，要考虑增加的数据在基础数据中是否存在。同样，删除基础数据时，要考虑可变记录中是否有指向该条基础数据的指针。

提高函数的复用度，一旦某个功能出现 BUG，只需要修改一个函数就够了，否则需要在整个项目中寻找使用过这一功能的模块，降低效率。例如增加删除修改函数对库存和金钱的影响，不要对每个具体的增加删除修改功能进行处理，而是采用一个统一的【结算】函数。一旦发现计算出错，只需要对结算函数本身进行调试即可。