华南理工大学
South China University of Technology

C++程序设计

课程设计报告书

题目： 背单词小程序

学　院	数学学院
专　业	信息与计算科学
学生姓名	杨程宇
学生学号	201730092420
指导教师	陆子强
课程编号	140021
课程学分	2
起始日期	2019.7.10

教师评语	教师签名： 日期：
成绩评定	
备注	

背单词小程序 1.0

一、功能需求分析

随着经济全球化的趋势不断发展，英语作为一门国际化的语言正起着越来越重要的作用。在中国，不管是工作，学习还是生活当中，人们都离不开英语，而人们学习英语需求和热情也在不断增大。而单词是英语的基础，要想学好英语，丰富的词汇量必不可少。

与此同时，计算机和智能手机的普及极大地方便了人们的各种需求的实现，但市场上的单词软件良莠不齐。这次设计尝试去实现一个简单的背单词程序，主要功能包括词库的管理，词条的查找及删改，单词测试以及逐条背单词等，目的是帮助人们通过智能设备，更加方便地进行英语地学习。

二、总体设计

本设计采用 C++语言编写，编译器为 Dev-C++。

设计理念是为用户打造一款个性化的，方便操作的，功能全面且界面美观的背单词程序。因此，我主要采用了“用户登录——主功能菜单——子功能菜单——功能实现”这样的操作模式。它的特点在于能够为每一位用户提供个性化的操纵。例如，用户可以导入自己的任意词库，除了 CET-4 的单词，TOFEL，GRE 的单词都可以导入词库进行学习。同时，不同的用户会拥有各自的动态生词库，用来存储在背单词过程中遇到的不会的生词。

本设计为每个单词设计了一个词条，词条信息包括 4 项：单词的拼写，词性，中文意思，以及笔记。用户可以自行添加笔记，例如单词的用法等等来帮助记忆。

下图所示为本设计的功能流程图：前两项为词库的管理，用户可以自行导入自己的单词库。单词搜索功能有四个子功能，分别为按英文搜索，按中文搜索，按英文精确搜索（即输入英文和词性锁定单词）以及快速搜索。快速搜索使用的是排序后的二分查找法，使得搜索的时间复杂度从 O（N）降低到 O（logN），从而提高了搜索效率。接下来四项为对词条的管理，其中修改单词信息会让用户先选择所要修改的内容，避免重复操作，是提升交互体验的一种做法。

单词测试分为中译英测试和英译中测试。中译英采用填空形式，而英译中由于很难完全准确的写出中文意思，所以以选择题的形式呈现。

最后的背单词有三个子功能，分别是复习生词，顺序记忆和乱序记忆。用户会有各自的一个动态的生词库，用来保存背单词过程中遇到的生词，在每次背单词时可选择复习生词，当掌握后生词自动从生词库删除。同时，用户也可以根据自身的喜好选择顺序记忆和乱序记忆。

经多次测试，系统的安全性和数据的完整性均符合预期，同时运行环境及性能良好。

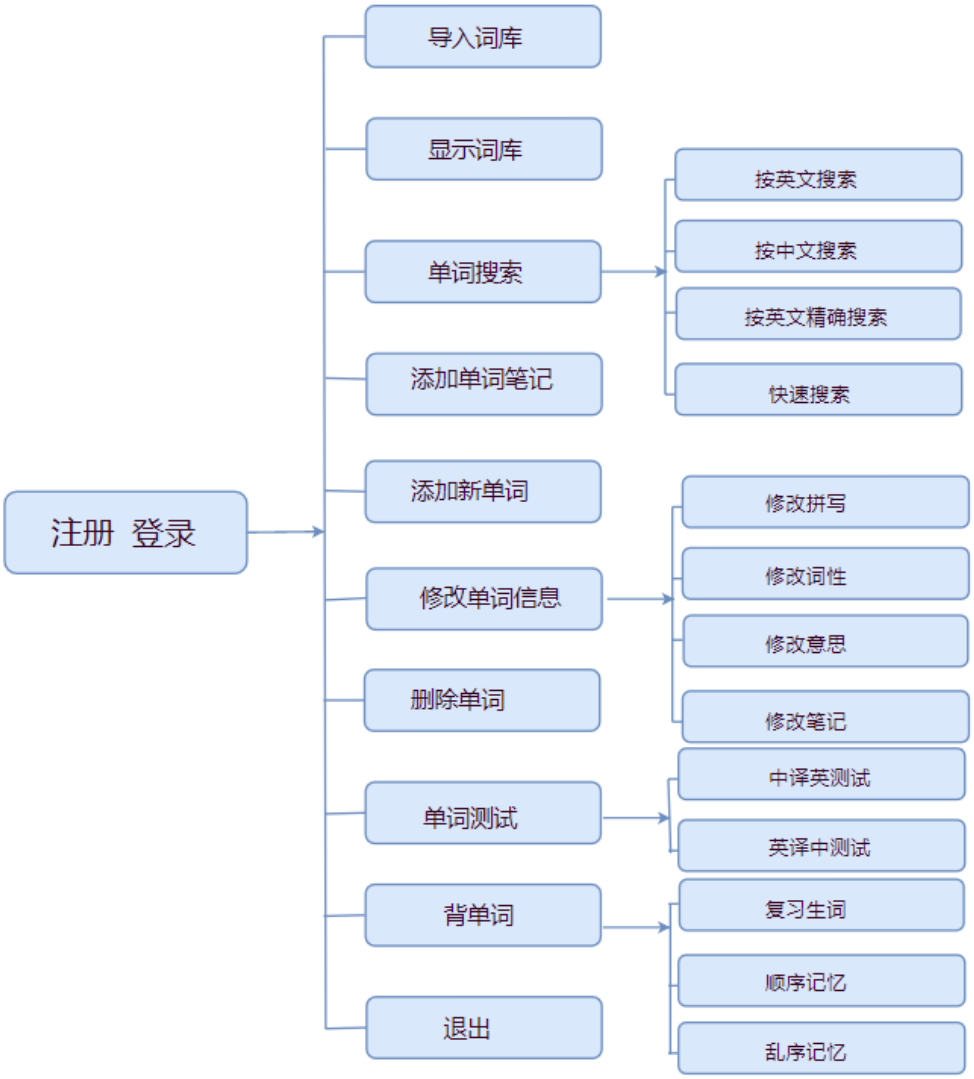
注册 登录
导入词库
显示词库
单词搜索
添加单词笔记
添加新单词
修改单词信息
删除单词
单词测试
背单词
退出
按英文搜索
按中文搜索
按英文精确搜索
快速搜索
修改拼写
修改词性
修改意思
修改笔记
中译英测试
英译中测试
复习生词
顺序记忆
乱序记忆

图 2.1　功能流程图

三、详细设计与程序实现

（一）.注册和登录

Ⅰ.结构体设计

用户信息结构体：每个结构变量包含用户名和密码。

```
struct User
{
   char username[20];
   char password[20];
};
```

Ⅱ.函数设计

1.void login();

设计思路：

用来显示登录界面，提示用户进行注册或登录。

函数代码：

```
void  login()
{
   int selection=0;
   int flag=0;//记录用户输入的用户名和密码是否匹配
   while(1)
   {  int m=0;  //用来记录内存中的用户数量
      struct User user_info[100];
      loadfile_enroll(user_info,m);
      system("cls");
      cout<<"---------------------"<<endl;
      cout<<"      背单词1.0      "<<endl;
      cout<<"---------------------"<<endl;
      cout<<"      1.登录         "<<endl;
      cout<<"      2.注册         "<<endl;
      cout<<"---------------------"<<endl;
      cout<<"请选择（1-2）：";
      cin>>selection;
      switch(selection)
      {
         case 1:flag=check_in(user_info,m);
                break;
         case 2:enrollment(user_info,m);
                break;
      }
      if(flag==1)
      {
         break;
      }
      else
```

```
        {
            continue;
        }
    }
    return;
}
```

2.int loadfile_enroll(struct User user_info[],int &m);

设计思路：

从硬盘中把用户信息导入到内存中，如果没有用户信息文本文件则新建一个。

函数参数为用户信息的结构体数组名，实际上是地址，因而数组里的内容可以实际被改变。整型变量 m 用来记录导入的用户信息数。

函数代码：

```
User newuser;
ifstream infile;
infile.open("d:\\用户信息.txt",ios::in);
if(!infile)
{
    ofstream outfile("d:\\用户信息.txt");
    outfile.close();
    m=0;
    return 0;
}
while(!infile.eof())
{
    infile>>newuser.username>>newuser.password;
    if(infile)
    {
        user_info[m]=newuser;
        m++;
    }
    else
    {
        break;
    }
}
infile.close();
return m;
}
```

3.int check_in(struct User user_info[],int m);

设计思路：

该函数用来为用户提供登录操作。但用户的输入可能有三种不同的情况，即登录成功，用户名不存在，或用户名和密码不匹配的问题。对这三种不同的情况，函数给出不同的操作。登录成功则进入主菜单，用户名不存在则返回进行注册操作，若密码错误则重新输入。

函数参数为用户信息结构体数组以及用户个数，函数内部将会进行搜索，检查是否匹配等工作。若登录成功则返回 1；若不能登录成功则返回 0。

在注册的过程中，用户需要输入密码。处于数据安全性的考虑，采用了 getch()函数进行密码的输入，这样用户的输入将不会出现在终端上，而以“*”取而代之。

函数代码:

```
int check_in(struct User user_info[],int m)
{
   while(1)
   {
      system("cls");
      char temp_username[20]={0};
      char temp_password[20]={0};
      int index=-1;
      cout<<"\n 请输入您的用户名：";
      cin>>temp_username;
      cout<<"\n 请输入您的密码：";
      int i=0;
      while(1)
      {
         char c=getch();//getch()不会向终端回显字符
         if(c=='\r')
         {
            break;
         }
         printf("*");
         temp_password[i]=c;
         i++;
```

```
    }
    cout<<endl;
    for(int i=0;i<m;i++)
    {
        if(strcmp(user_info[i].username,temp_username)==0)
        {
            index=i;
            break;
        }
    }
    if(index==-1)
    {
        cout<<"\n 用户名未注册，请先注册！\n";
        Sleep(3000);
        break;
    }
    else
    {
        if(strcmp(user_info[index].password,temp_password)!=0)
        {
            cout<<"\n 用户名和密码不匹配，请重新输入！\n";
            Sleep(3000);
            continue;
        }
        else
        {
            strcpy(userinfo_filename,temp_username);
            cout<<"\n 登录成功！\n";
            Sleep(3000);
            return 1;
        }
    }
}
return 0;
```

```
}
```

4.int savefile_enroll(struct User user_info[],int m);

设计思路：

与 loadfile_enroll 类似，在用户注册了一个新账户，即内存中的用户信息发生变动时，用来向文本文件中输出信息。

函数参数为用户信息结构体数组，用户数量。若成功存储用户信息，则返回 1，反之，返回 0。

函数代码：

```
int savefile_enroll(struct User user_info[],int m)
{
   ofstream outfile;
   outfile.open("d:\\用户信息.txt",ios::out);
   if(!outfile)
   {
      cout<<"\n 无法将数据保存到文件 d:\\用户信息.txt !";
      Sleep(3000);
      return 0;
   }
   for(int i=0;i<m;i++)
   {
      outfile<<user_info[i].username<<"
"<<user_info[i].password<<endl;
   }
   outfile.close();
   return 1;
}
```

5. void enrollment(struct User user_info[],int &m)；

设计思路：

该函数为用户提供注册操作。新注册的账号将会被保存进用户信息数组，离开函数时将调用 save_file 函数将新的信息写入硬盘中的文本文件。

函数参数为用户信息数组，以及 m 的引用，这是因为没注册成功一个账户，m 会发生变化。

函数代码；

```
void enrollment(struct User user_info[],int &m)
{
   while(1)
   {
      system("cls");
      int flag=0;
      char temp_username[20];
      char temp_password[20]={0};
      char temp_repass[20]={0};
      cout<<"\n 请输入您要注册的用户名(仅支持字母和数字)：";
      cin>>temp_username;
      cout<<"\n 请输入密码(仅支持字母和数字)：";
      int i=0;
      while(1)
      {
         char c=getch();//getch()不会向终端回显字符
         if(c=='\r')
         {
            break;
         }
         printf("*");
         temp_password[i]=c;
         i++;
      }
      cout<<"\n\n 请确认密码(仅支持字母和数字)：";
      i=0;
      while(1)
```

```
        {
            char c=getch();//getch()不会向终端回显字符
            if(c=='\r')
            {
                break;
            }
            printf("*");
            temp_repass[i]=c;
            i++;
        }
        cout<<endl;
        for(int i=0;i<m;i++)
        {
            if(strcmp(temp_username,user_info[i].username)==0)
            {
                flag=1;
                cout<<"\n 您已注册，请直接登录！";
                Sleep(3000);
                goto out;
            }
        }

        if(flag==0)
        {
            if(strcmp(temp_password,temp_repass)!=0)
            {
                cout<<"\n 两次密码不一致，请重新输入！";
                Sleep(3000);
                continue;
            }
            else
            {
//              struct User newuser;
//              strcpy(newuser.username,temp_username);
```

```
//              strcpy(newuser.username,temp_password);
//这么写是错的？user_info[m]=newuser;
                strcpy(user_info[m].username,temp_username);
                strcpy(user_info[m].password,temp_password);
                m++;
                cout<<"\n 注册成功，请登录！";
                Sleep(3000);
                break;
            }
        }
    }
    out:
    savefile_enroll(user_info,m);
    return ;
}
```

（二）.主函数

Ⅰ.结构体设计：

词条结构体：每个词条包含四个数据成员，分别是单词的拼写，单词的词性，单词的中文意思，以及单词的笔记。用户可自行添加笔记例如单词的用法，来帮助记忆。同时也可以对单词信息进行修改。

```
struct Word
{
   char word[50];
   char word_class[10];
   char word_meaning[100];
   char word_note[50];
};
```

II.全局变量

```
clock_t start,end;
```

定义了两个用来计时的全局变量。

III.函数设计

1. int main(int argc, char** argv)；

设计思路：

用来显示主菜单，同时根据用户的输入，采用 switch 语句来调用不同的函数以实现不同的功能。

函数代码：

```
int main(int argc, char** argv)
{
   login();
   static struct Word wordlist[10000];
   int n=0;//用来记录内存中的单词数
   int selection=0;
   while(1)
   {
      system("cls");
      cout<<"--------------------------------"<<endl;
      cout<<"         背单词小程序           "<<endl;
      cout<<"--------------------------------"<<endl;
      cout<<"         1.导入词库             "<<endl;
      cout<<"         2.显示词库             "<<endl;
      cout<<"         3.单词搜索             "<<endl;
      cout<<"         4.添加单词笔记         "<<endl;
      cout<<"         5.添加新单词           "<<endl;
      cout<<"         6.修改单词信息         "<<endl;
      cout<<"         7.删除单词             "<<endl;
      cout<<"         8.单词测试             "<<endl;
      cout<<"         9.背单词               "<<endl;
      cout<<"         0.退出                 "<<endl;
      cout<<"--------------------------------"<<endl;
      cout<<"选择（0-9）:";
      cin>>selection;
      getchar();
      switch(selection)
      {
```

```
        case 1:    load_file(wordlist,n);
                   break;
        case 2: print(wordlist,n);
                   break;
        case 3: Search(wordlist,n);
                   break;
        case 4: addnote(wordlist,n);
                   break;
         case 5: addword(wordlist,n);
                   break;
         case 6: changeword(wordlist,n);
                   break;
         case 7: deleteword(wordlist,n);
                   break;
         case 8: Test(wordlist,n);
                   break;
        case 9: rememberword(wordlist,n);
                   break;
        case 0: if(n==0)
                   {
                      break;
                   }
                   else
                   {
                      save_file(wordlist,n);
                      break;
                   }
    }
    if(selection==0)
    {
        break;
    }
}
return 0;
```

```
}
```

2. int load_file(struct Word wordlist[],int &n);

设计思路：

该函数用来实现对应的第一个功能。为了提供更加个性化的服务，用户可选择导入个性化的词库，因而需要用户输入词库文本文件的存储位置。

函数参数为词表数组和记录单词个数的变量 n。若成功导入则返回 1，导入失败则返回 0。

同时，函数会显示导入数据所耗费的时间。

函数代码：

```
int load_file(struct Word wordlist[],int &n)
{
   system("cls");
   Word newword;
   ifstream infile;
   cout<<"请输入单词文件的位置：";
   cin>>wordlist_filename;
   getchar();
   infile.open(wordlist_filename,ios::in);
   if(!infile)
   {
      cout<<endl<<"无数据文件 "<<wordlist_filename<<" !";
      Sleep(3000);
      n=0;
      return 0;
   }

   start=clock();
   while(!infile.eof())
   {

infile>>newword.word>>newword.word_class>>newword.word_meaning;
      infile.getline(newword.word_note,50);
      cout.setf(ios::left);
      if(infile)
      {
```

```
    //
    cout<<setw(20)<<newword.word<<newword.word_class<<newword.word_me
aning<<newword.word_note<<endl;
                wordlist[n]=newword;
                n++;
            }
            else
            {
                break;
            }
        }
        end=clock();

        infile.close();
        cout<<endl<<"成功导入"<<n<<"个单词，用时";
        cout<<double(end-start)/CLOCKS_PER_SEC;
        cout<<"秒。\n\n 按任意键回到主菜单......\n";
        getchar();
        return n;
    }
```

3. int save_file(struct Word wordlist[],int n);

设计思路：

类似于 load_file；当用户结束使用时，将内存中的词表存入文本文件中。

若函数存储成功，则返回 1，不成功则返回 0。函数参数为词表数组以及单词数量。

函数代码：

```
int save_file(struct Word wordlist[],int n)
{
   ofstream outfile;
   outfile.open(wordlist_filename,ios::out);
   if(!outfile)
   {
      cout<<"无法将数据保存到文件 "<<wordlist_filename<<" !";
      Sleep(3000);
      return 0;
   }
   for(int i=0;i<n;i++)
   {
      outfile<<wordlist[i].word<<"   "<<wordlist[i].word_class<<"
"<<wordlist[i].word_meaning<<wordlist[i].word_note<<endl;  //getling
读入时 note 前有一个空格了
   }
   outfile.close();
   return 1;
}
```

4. void print(struct Word wordlist[],int n)；

设计思路：

该函数主要用来实现第二个主功能，向屏幕上打印词库中的单词。同时显示词库中单词数量以及打印所用时间。函数参数同上不细表。

函数代码：

```
void print(struct Word wordlist[],int n)
{
   system("cls");
   if(n==0)
   {
      cout<<endl<<"词库中无单词，请导入词库！\n";
   }
   else
   {
      cout.setf(ios::left);
      cout<<endl<<endl;
      cout<<setw(20)<<"单词"<<setw(10)<<"词性"<<setw(31)<<"中文意思"<<"笔记"<<endl;

   cout<<"------------------------------------------------------------------------------------------"<<endl;

      start=clock();
      for(int i=0;i<n;i++)
      {

   cout<<setw(20)<<wordlist[i].word<<setw(10)<<wordlist[i].word_class<<setw(30)<<wordlist[i].word_meaning<<wordlist[i].word_note<<endl;
      }
      end=clock();

      cout<<endl<<"单词信息已显示完毕！\n";
      cout<<endl<<"一共显示"<<n<<"个单词，用时" ;
      cout<<double(end-start)/CLOCKS_PER_SEC<<"秒。\n";
```

```
    }
    cout<<endl<<"按任意键返回主菜单.....\n";
    getchar();
    return;
}
```

5. void Search(struct Word wordlist[],int n)；

设计思路：

该函数主要用来显示搜索菜单，英文搜索，中文搜索，精确搜索和快速搜索。并根据用户的选择借助 switch 函数调用相应的搜索函数。

函数参数同上不细表。

函数代码：

```
void Search(struct Word wordlist[],int n)
{
   while(1)
   {
      system("cls");
      int sel=0;
      cout<<endl;
      cout<<"----------------------"<<endl;
      cout<<"          搜索        "<<endl;
      cout<<"----------------------"<<endl;
      cout<<"    1.输入英文搜索    "<<endl;
      cout<<"    2.输入中文搜索    "<<endl;
      cout<<"    3.英文精确搜索    "<<endl;
      cout<<"    4.快速搜索        "<<endl;
      cout<<"    5.退出搜索        "<<endl;
      cout<<"----------------------"<<endl;
      cout<<"请输入你的选择（1-5）：";
      cin>>sel;
      getchar();
      switch(sel)
      {
         case 1: search_EtoC(wordlist,n);
                 break;
         case 2: search_CtoE(wordlist,n);
                 break;
         case 3: search_EtoC_spe(wordlist,n);
                 break;
         case 4: search_fast(wordlist,n);
```

```
                    break;
            case 5: break;
        }
        if(sel==5)
        {
            break;
        }
    }
    return ;
}
```

5.1. void search_EtoC(struct Word wordlist[],int n);

设计思路：

这是搜索功能的第一个分支，由英文搜索出其他信息。用户输入需要查找的单词，若查找到则输出该单词的其他信息，并输出查找的时间。

函数代码：

```
void search_EtoC(struct Word wordlist[],int n)
{
   while(1)
   {
      system("cls");
      int cnt=0;//记录结果个数
      char object[20];
      cout<<endl<<"请输入你想要查询的单词（区分大小写并按回车结束输入）：";
      cin>>object;

      start=clock();
      for(int i=0;i<n;i++)//查找并输出
      {
         if(strcmp(object,wordlist[i].word)==0)
         {
            if(cnt==0)
            {
               cout<<endl<<setw(10)<<"词性"<<setw(31)<<"中文意思"<<"笔记"<<endl;

   cout<<"-------------------------------------------------"<<endl;

   cout<<setw(10)<<wordlist[i].word_class<<setw(30)<<wordlist[i].word_meaning<<wordlist[i].word_note<<endl;
               cnt++;
            }
            else
```

```
                {

   cout<<setw(10)<<wordlist[i].word_class<<setw(30)<<wordlist[i].word_meaning<<wordlist[i].word_note<<endl;
                    cnt++;
                }
            }
        }
        end=clock();

        if(cnt==0)
        {
            cout<<endl<<"很抱歉没有找到您要查询的单词。\n";
        }
        else
        {
            cout<<endl<<"以上共"<<cnt<<"条搜索结果，用时";
            cout<<double(end-start)/CLOCKS_PER_SEC<<"秒。\n" ;
        }

        int _selection=0;
        cout<<endl<<"输入 0 返回搜索菜单，输入 1 继续查找（请按回车键结束输入）：";
        cin>>_selection;
        if(_selection==0)
        {
            break;
        }
        else if(_selection==1)
        {
            continue;
        }
    }
    getchar();
```

```
    return;
}
```

5.2. void search_CtoE(struct Word wordlist[],int n)；

设计思路：

该函数实现的功能与上一个函数相反，即由中文搜索出其他信息。

函数代码：

```
void search_CtoE(struct Word wordlist[],int n)
{
   while(1)
   {
      system("cls");
      int cnt=0;//记录结果个数
      char object[20];
      cout<<endl<<"请输入你想要查询的单词的中文意思：";
      cin>>object;

      start=clock();
      for(int i=0;i<n;i++)//查找并输出
      {
         if(strcmp(object,wordlist[i].word_meaning)==0)
         {
            if(cnt==0)
            {
               cout<<endl<<setw(10)<<"词性"<<setw(21)<<"英文单
词"<<"笔记"<<endl;

   cout<<"-----------------------------------------"<<endl;

   cout<<setw(10)<<wordlist[i].word_class<<setw(20)<<wordlist[i].wor
d<<wordlist[i].word_note<<endl;
               cnt++;
            }
            else
            {

   cout<<setw(10)<<wordlist[i].word_class<<setw(20)<<wordlist[i].wor
```

```
d<<wordlist[i].word_note<<endl;
                    cnt++;
                }
            }
        }
        end=clock();

        if(cnt==0)
        {
            cout<<endl<<"很抱歉没有找到您要查询的单词。\n";
        }
        else
        {
            cout<<endl<<"以上共"<<cnt<<"条搜索结果，用时";
            cout<<double(end-start)/CLOCKS_PER_SEC<<"秒。\n" ;
        }

        int _selection=0;
        cout<<endl<<"输入 0 返回搜索菜单，输入 1 继续查找（请按回车键
结束输入）：";
        cin>>_selection;
        if(_selection==0)
        {
            break;
        }
        else
        {
            continue;
        }
    }
    getchar();
    return;
}
```

5.3. void search_EtoC_spe(struct Word wordlist[],int n);

设计思路：

该函数功能是 5.1 函数的拓展。因为有些单词，比如 abstract,有两种词性，分别对应不同的意思。因此如果用户希望精确地找出某个词性地意思，则可以使用该函数。

函数代码:

```
void search_EtoC_spe(struct Word wordlist[],int n)
{
   while(1)
   {
      system("cls");
      int cnt=0;//记录结果个数
      char object_word[20];
      char object_word_class[10];
      cout<<endl<<"请输入你想要查询的单词（区分大小写并按回车结束输入）：";
      cin>>object_word;
      cout<<"请输入该单词的词性（小写并按回车键结束输入）：";
      cin>>object_word_class;

      start=clock();
      for(int i=0;i<n;i++)
      {

   if(strcmp(object_word,wordlist[i].word)==0&&strcmp(object_word_class,wordlist[i].word_class)==0)
            {
               cout<<endl<<setw(31)<<"中文意思"<<"笔记";

   cout<<endl<<"--------------------------------------";

   cout<<endl<<setw(30)<<wordlist[i].word_meaning<<wordlist[i].word_note;
               cout<<endl;
```

```
                cnt++;
            }
        }
        end=clock();

        if(cnt==1)
        {
            cout<<endl<<"    本    次    搜    索    用    时
"<<double(end-start)/CLOCKS_PER_SEC<<"秒。\n";
        }
        else
        {
            cout<<endl<<"很抱歉没有找到您要查询的单词。\n";
        }

        int _selection=0;
        cout<<endl<<"输入 0 返回搜索菜单，输入 1 继续查找（请按回车键
结束输入）：";
        cin>>_selection;
        if(_selection==0)
        {
            break;
        }
        else
        {
            continue;
        }
    }
    getchar();
    return;
}
```

5.4. void search_fast(struct Word wordlist[],int n);

设计思路：

这个函数是对英文搜索的改善，将顺序查找的方法改为二分查找，从而将时间复杂度从O（n）降为O（logn）。但是采用二分查找带来的一个问题是，由于一个单词的两个词性分属两个不同的词条，但最终只能根据英文单词查到一个词条。

函数代码：

```
void search_fast(struct Word wordlist[],int n)
{
   while(1)
   {
      system("cls");
      int cnt=0;//记录结果个数
      char object[20];
      cout<<endl<<"请输入你想要查询的单词（区分大小写并按回车结束输入）：";
      cin>>object;

      int left=0;
      int right=n-1;
      int center=(left+right)/2;
      start=clock();
      while(left<=right)
      {
         if(strcmp(wordlist[center].word,object)==0)
         {
            break;
         }
         else if(strcmp(wordlist[center].word,object)<0)
         {
            left=center+1;
            center=(left+right)/2;
         }
         else
```

```
            {
                right=center-1;
                center=(left+right)/2;
            }
        }
        end=clock();

        if(left>right)
        {
            cnt=0;
        }
        else
        {
            cout<<endl<<setw(10)<<"词性"<<setw(31)<<"中文意思"<<"笔记"<<endl;

    cout<<"-------------------------------------------------"<<endl;

    cout<<setw(10)<<wordlist[center].word_class<<setw(30)<<wordlist[center].word_meaning<<wordlist[center].word_note<<endl;
            cnt++;
        }

        if(cnt==0)
        {
            cout<<endl<<"很抱歉没有找到您要查询的单词。\n";
        }
        else
        {
            cout<<endl<<"以上共"<<cnt<<"条搜索结果，用时";
            cout<<double(end-start)/CLOCKS_PER_SEC<<"秒。\n" ;
        }
```

```
            int _selection=0;
            cout<<endl<<"输入 0 返回搜索菜单，输入 1 继续查找（请按回车键结束输入）：";
            cin>>_selection;
            if(_selection==0)
            {
                break;
            }
            else if(_selection==1)
            {
                continue;
            }
        }
        getchar();
        return;
    }
```

6. void addnote(struct Word wordlist[],int n)；

设计思路：

因为给每个词条设计了笔记这个数据成员，而原始词库中并没有笔记，因此用户可以自行为每个单词添加笔记，比如单词的用法，发音等等。这些改变会被保存进词库。

函数代码：

```
void addnote(struct Word wordlist[],int n)
{
   while(1)
   {
      system("cls");
      int cnt=0;//用来标记有没有找到
      int temp=0;//用来记录找到的单词的下标
      char input[50];
      char object_word[20];
      char object_word_class[10];
      cout<<endl<<"请输入你想要添加笔记的单词（区分大小写并按回车结束输入）：";
      cin>>object_word;
      cout<<endl<<"请输入该单词的词性（小写并按回车键结束输入）：";
      cin>>object_word_class;
      getchar();

      start=clock();
      for(int i=0;i<n;i++)
      {

   if(strcmp(object_word,wordlist[i].word)==0&&strcmp(object_word_class,wordlist[i].word_class)==0)
            {
               cnt++;
               temp=i;
            }
```

```
            }
            end=clock();

            if(cnt==1)
            {
                strcpy(wordlist[temp].word_note," ");//原来是" 暂无"
                cout<<endl<<"已找到该单词。请输入您要添加的笔记：";
                cin.getline(input,50);
                cout<<endl<<"添加成功！"<<endl;
                strcat(wordlist[temp].word_note,input);
            }
            else
            {
                cout<<endl<<"很抱歉没有找到您输入的单词。"<<endl;
            }

            cout<<endl<<"输入 0 返回主菜单，输入 1 继续添加（请按回车键结束输入）：";
            int _selection=0;
            cin>>_selection;
            if(_selection==0)
            {
                break;
            }
            else if(_selection==1)
            {
                continue;
            }
        }
        return ;
    }
```

7. void addword(struct Word wordlist[],int &n);

设计思路：

向词库中添加新的单词。

函数代码：

```
void addword(struct Word wordlist[],int &n)
{
   while(1)
   {
      system("cls");
      char input_word[50];
      char input_word_class[10];
      char input_word_meaning[100];
      char input_word_note[50];
      int cnt=0;
      cout<<endl<<"请输入您要添加的单词（按回车键结束输入）：";
      cin>>input_word;
      cout<<endl<<"请输入该单词的词性（按回车键结束输入）：";
      cin>>input_word_class;

      for(int i=0;i<n;i++)
      {

   if(strcmp(wordlist[i].word,input_word)==0&&strcmp(wordlist[i].wor
d_class,input_word_class)==0)
            {
               cnt++;
            }
      }

      if(cnt==1)
      {
         cout<<endl<<"词库里已经有您要添加的单词了哦！\n";
      }
      else
```

```
            {
                cout<<endl<<"请输入该单词的中文意思（按回车键结束输入）：
";
                cin>>input_word_meaning;
                getchar();
                cout<<endl<<"请输入该单词的笔记（按回车键结束输入）：";
                cin.getline(input_word_note,50);
                strcpy(wordlist[n].word,input_word);
                strcpy(wordlist[n].word_class,input_word_class);
                strcpy(wordlist[n].word_meaning,input_word_meaning);
                strcpy(wordlist[n].word_note," ");
                strcat(wordlist[n].word_note,input_word_note);
                n++;
                cout<<endl<<"添加成功！\n";
            }

            cout<<endl<<"输入 0 返回主菜单，输入 1 继续添加（请按回车键结
束输入）：";
            int _selection=0;
            cin>>_selection;
            if(_selection==0)
            {
                break;
            }
            else if(_selection==1)
            {
                continue;
            }
        }
        return ;
    }
```

8. void changeword(struct Word wordlist[],int n);

设计思路：

该函数的主要功能是修改已有的单词信息。为了避免每次修改的时候将所有的单词信息重新写一遍，这个函数提供了每个单词修改的选项，用户可以选择修改词条的某一个数据成员。

函数代码：

```
void changeword(struct Word wordlist[],int n)
{
   while(1)
   {
      system("cls");
      char input_word[50];
      char input_word_class[10];
      char input_word_meaning[100];
      char input_word_note[50];
      int cnt=0;
      int temp=0;
      cout<<endl<<"请输入您要修改的单词（按回车键结束输入）：";
      cin>>input_word;
      cout<<endl<<"请输入该单词的词性（按回车键结束输入）：";
      cin>>input_word_class;

      for(int i=0;i<n;i++)
      {

   if(strcmp(wordlist[i].word,input_word)==0&&strcmp(wordlist[i].word_class,input_word_class)==0)
            {
               cnt++;
               temp=i;
            }
      }

      if(cnt==0)
```

```
            {
                cout<<endl<<"很抱歉没有找到您输入的单词。\n";
            }
            else
            {
                while(1)
                {
                    system("cls");
                    int sel=0;
                    cout<<endl<<"您要修改的单词：\n";
                    cout<<endl;
                    cout<<setw(20)<<"  单  词  "<<setw(10)<<"  词  性
"<<setw(31)<<"中文意思"<<"笔记"<<endl;

    cout<<"-------------------------------------------------------------
----------------"<<endl;

    cout<<setw(20)<<wordlist[temp].word<<setw(10)<<wordlist[temp].wor
d_class<<setw(30)<<wordlist[temp].word_meaning<<wordlist[temp].word_n
ote<<endl;
                    cout<<endl;
                    cout<<endl;
                    cout<<"------------\n";
                    cout<<"    修改    \n";
                    cout<<"------------\n";
                    cout<<" 1.单词拼写 \n";
                    cout<<" 2.单词词性 \n";
                    cout<<" 3.单词意思 \n";
                    cout<<" 4.单词笔记 \n";
                    cout<<" 5.修改完毕 \n";
                    cout<<"------------\n";
                    cout<<"请输入要修改的选项（1-5）：" ;
                    cin>>sel;
```

```
                switch(sel)
                {
                    case 1: cout<<endl<<"请输入修改后的单词拼写：";
                            cin>>input_word;
                            strcpy(wordlist[temp].word,input_word);
                            cout<<"拼写修改成功！\n";
                            Sleep(3000);
                            break;
                    case 2: cout<<endl<<"请输入修改后的词性：";
                           cin>>input_word_class;

strcpy(wordlist[temp].word_class,input_word_class);
                           cout<<"词性修改成功！\n";
                           Sleep(3000);
                           break;
                    case 3: cout<<endl<<"请输入修改后的中文意思：";
                            cin>>input_word_meaning;

strcpy(wordlist[temp].word_meaning,input_word_meaning);
                           cout<<"中文意思修改成功！\n";
                           Sleep(3000);
                           break;
                    case 4: getchar();
                           cout<<endl<<"请输入修改后的笔记：";
                            cin.getline(input_word_note,50);
                           strcpy(wordlist[temp].word_note," ");

strcat(wordlist[temp].word_note,input_word_note);
                           cout<<"笔记修改成功！\n";
                           Sleep(3000);
                           break;
                    case 5: break;
                }
```

```
                if(sel==5)
                {
                    break;
                }
            }
            cout<<endl<<"该单词信息修改成功！\n";
        }

        cout<<endl<<"输入 0 返回主菜单，输入 1 继续修改其它单词（请按回车键结束输入）：";
        int _selection;
        cin>>_selection;
        if(_selection==0)
        {
            break;
        }
        else if(_selection==1)
        {
            continue;
        }
    }
}
```

9. void deleteword(struct Word wordlist[],int &n)；

设计思路：

该函数的功能为删除词库中的某一词条。因为词条存放于数组中，所以需要将后面所有的单词往前挪。

函数代码：

```
void deleteword(struct Word wordlist[],int &n)
{

   while(1)
   {
      system("cls");
      char input_word[50];
      char input_word_class[10];
      int cnt=0;
      int temp=0;
      cout<<endl<<"请输入您要删除的单词（按回车键结束输入）：";
      cin>>input_word;
      cout<<endl<<"请输入该单词的词性（按回车键结束输入）：";
      cin>>input_word_class;

      for(int i=0;i<n;i++)
      {

   if(strcmp(wordlist[i].word,input_word)==0&&strcmp(wordlist[i].word_class,input_word_class)==0)
            {
               cnt++;
               temp=i;
            }
      }

      if(cnt==0)
      {
         cout<<endl<<"很抱歉没有找到您输入的单词。\n";
```

```
            }
            else
            {
                for(int j=temp;j<=n-2;j++)
                {
                    wordlist[j]=wordlist[j+1];
                }
                n--;
                cout<<endl<<"已找到您输入的单词，删除成功！\n";
                cout<<endl<<"词库中还有"<<n<<"个单词。\n";
            }

            cout<<endl<<"输入 0 返回主菜单，输入 1 继续删除（请按回车键结束输入）：";
            int _selection;
            cin>>_selection;
            if(_selection==0)
            {
                break;
            }
            else if(_selection==1)
            {
                continue;
            }
        }
    }
```

10. void Test(struct Word wordlist[],int n)；

设计思路：

因为这款程序为用户提供了两种测试模式，因此这个函数的主要功能是为用户显示测试菜单，根据用户的选择借助 switch 语句调用不同的测试函数。

函数代码:

```
void Test(struct Word wordlist[],int n)
{
   while(1)
   {
      system("cls");
      int sel=0;
      cout<<endl;
      cout<<"----------------------"<<endl;
      cout<<"          测试        "<<endl;
      cout<<"----------------------"<<endl;
      cout<<"    1.中译英测试      "<<endl;
      cout<<"    2.英译中测试      "<<endl;
      cout<<"    3.退出测试        "<<endl;
      cout<<"----------------------"<<endl;
      cout<<"请输入你的选择（1-3)：";
      cin>>sel;
      getchar();
      switch(sel)
      {
         case 1: test_CtoE(wordlist,n);
                 break;
         case 2: test_EtoC(wordlist,n);
                 break;
         case 3: break;
       }
       if(sel==3)
       {
         break;
       }
```

```
    }
    return ;
}
```

10.1.void test_CtoE(struct Word wordlist[],int n)

设计思路：

这个函数提供的测试模式是借助 rand()函数随机给出中文意思而让用户默写出英文单词。每次测试十道题，最后会给出正确率和答题时间，并用 swtich 语句给出相应的反馈，比如超过 90%的正确率则提示“你好厉害!”。

函数代码：

```
void test_CtoE(struct Word wordlist[],int n)
{
   while(1)
   {
      system("cls");
      int cnt=0;
      int wrongans[10]={-1,-1,-1,-1,-1,-1,-1,-1,-1,-1};
      char input[50];
      cout<<endl<<"请根据给出的中文意思及词性写出相应的英文单词（共十题,每题十分）：\n";

      start=clock();
      srand(time(NULL));
      for(int i=0;i<10;i++)
      {
         int rad=rand()%n;//产生 0-n-1 的随机数
         cout<<endl<<"    第    "<<i+1<<"    题    :          "<<wordlist[rad].word_meaning<<" "<<wordlist[rad].word_class<<endl;
         cout<<"请输入你的答案：";
         cin>>input;
         if(strcmp(wordlist[rad].word,input)==0)
         {
            cnt++;
            cout<<"恭喜你，答对了！\n";
         }
         else
         {
            wrongans[i]=rad;
```

```
            cout<<"很遗憾，答错了！\n";
        }
    }
    end=clock();

    cout<<endl;
    switch(cnt)
    {
        case 10:
        case  9: cout<<"太厉害了，";
                 break;
        case  8:
        case  7: cout<<"还不错哟，";
                 break;
        case  6:
        case  5:
        case  4:
        case  3:
        case  2:
        case  1:
        case  0: cout<<"要加油啦，";
    }
    cout<<"  你 本 次 测 试 用 时 "<<double(end-start)/CLOCKS_PER_SEC<<"秒， 正确率 " <<cnt*10<<"% 。"<<endl;
    if(cnt!=10)
    {
        cout<<"\n以下为正确答案："<<endl;
        for(int i=0;i<10;i++)
        {
            if(wrongans[i]!=-1)
            {
                cout<<endl<<" 第 "<<i+1<<" 题 的 正 确 答 案 为 ："<<wordlist[wrongans[i]].word<<endl;
```

```
                }
            }
        }

        int _selection=0;
        cout<<endl<<"输入 0 返回测试菜单，输入 1 继续测试（请按回车键结束输入）：";
        cin>>_selection;
        if(_selection==0)
        {
            break;
        }
        else if(_selection==1)
        {
            continue;
        }
    }
    return ;
}
```

10.2.void test_EtoC(struct Word wordlist[],int n)

设计思路：

这个函数提供的测试模式是给出英文让用户选择正确的英文。考虑到用户很难一字不差地默写出中文意思，因此题目按照选择题的模式给出。同样，最后会给出正确率，答题时间以及反馈。

函数代码：

```
while(1)
{
    system("cls");
    srand(time(0));
    int cnt=0;
    int wrongans[10]={-1,-1,-1,-1,-1,-1,-1,-1,-1,-1};
    cout<<endl<<"请根据给出的英文单词及词性选择相应的中文意思（共十题,每题十分）：\n";

    start=clock();
    for(int k=0;k<10;k++)
    {
        char content_word[4][50];
        char content_class[4][10];
        char content_meaning[4][50];
        int rad[4];
        for(int i=0;i<4;i++)//随机产生四个单词的下标,不重复
        {
            rad[i]=rand()%n;
            int flag=1;
            while(1)
            {
                for(int j=0;j<i;j++)
                {
                    if(rad[j]==rad[i])
                    {
                        flag=0;
```

```
                    break;
                }
            }
            if(flag==1)
            {
                break;
            }
            else
            {
                continue;
            }
        }
    }

    for(int i=0;i<4;i++)
    {
        strcpy(content_word[i],wordlist[rad[i]].word);

    strcpy(content_class[i],wordlist[rad[i]].word_class);

    strcpy(content_meaning[i],wordlist[rad[i]].word_meaning);
    }

    int right_ans=rand()%4;//随机产生正确答案;
    char ans;
    cout<<endl<<"       第       "<<k+1<<"       题       :
"<<content_word[right_ans]<<"  "<<content_class[right_ans]<<endl;

    cout<<"A."<<content_meaning[0]<<"\nB."<<content_meaning[1]<<"\nC.
"<<content_meaning[2]<<"\nD."<<content_meaning[3]<<endl;
    cout<<"请输入你的选择：";
    cin>>ans;
    if(ans-'a'==right_ans||ans-'A'==right_ans)
    {
```

```
            cout<<"恭喜你，答对啦！\n";
            cnt++;
        }
        else
        {
            cout<<"很遗憾，答错了。\n";
            wrongans[k]=right_ans;
        }

    }
    end=clock();

    cout<<endl;
    switch(cnt)
    {
        case 10:
        case  9: cout<<"太厉害了，";
                 break;
        case  8:
        case  7: cout<<"还不错哟，";
                 break;
        case  6:
        case  5:
        case  4:
        case  3:
        case  2:
        case  1:
        case  0: cout<<"要加油啦，";
    }
    cout<<"    你    本    次    测    试    用    时
"<<double(end-start)/CLOCKS_PER_SEC<<"秒，正确率" <<cnt*10<<"%。
"<<endl;
    if(cnt!=10)
    {
```

```
                cout<<"\n 以下为正确答案："<<endl;
                for(int i=0;i<10;i++)
                {
                    if(wrongans[i]!=-1)
                    {
                        cout<<endl<<" 第 "<<i+1<<" 题 的 正 确 答 案 为 ："<<(char)(wrongans[i]+'A')<<endl;
                    }
                }
            }
            int _selection=0;
            cout<<endl<<"输入 0 返回测试菜单，输入 1 继续测试（请按回车键结束输入）：";
            cin>>_selection;
            if(_selection==0)
            {
                break;
            }
            else if(_selection==1)
            {
                continue;
            }
        }
        return ;
    }
```

11. void sort(struct Word wordlist[],int n)；

设计思路：

这是一个十分重要的函数，它的作用是将内存中的词条按照字典顺序进行排序。在显示词库，快速搜索等功能之前会调用这个函数。

函数代码：

```
void sort(struct Word wordlist[],int n)
{
   struct Word temp;
   //冒泡排序算法；进行 n-1 次比较
   for(int i=0;i<n-1;i++)
   {
      for(int j=0;j<n-1-i;j++)
      {
         if(strcmp(wordlist[j].word,wordlist[j+1].word)>0)
         {
            temp=wordlist[j];
            wordlist[j]=wordlist[j+1];
            wordlist[j+1]=temp;
         }
      }
   }
   return ;
}
```

（三）.背单词

考虑到为背单词设计的功能比较复杂，这里单独进行阐释。

首先，有几个较为简单的函数，其实现方法与之前类似，这里仅简要介绍其功能。

int loadfile_bdc(struct Word unknown[],int &cnt)；

设计思路：

从用户的生词库中导入生词到内存当中，如果当前用户第一次使用，则创建当前用户的生词库。

int savefile_bdc(struct Word unknown[],int cnt)；

设计思路：

将内存中的生词导入生词库。

void deleteword_bdc(struct Word unknown[],int &cnt,int i)

//i 为要删除的单词的下标

设计思路：

给出下标，删除某一单词。

void addword_bdc(struct Word unknown[],int &cnt,struct Word toadd)；

设计思路：

将用户背诵过程中遇到的不会的单词添加到生词库中。

int find_word(struct Word wordlist[],int n,char _word[],char _class[])；

设计思路：

查找单词，若找到则返回下标，没找到返回-1。

1.void rememberword(struct Word wordlist[],int n)；

设计思路：

因为给用户提供了三种背单词的模式，所以这个函数的主要功能是显示菜单，根据用户的选择调用相应的函数。

函数代码：

```
void rememberword(struct Word wordlist[],int n)
{
   while(1)
   {
      int cnt=0;//用来记录内存中的生词数
      struct Word unknown[1000];
      loadfile_bdc(unknown,cnt);
      int selection=0;

      system("cls");
      cout<<"--------------------"<<endl;
      cout<<"    背单词模式     "<<endl;
      cout<<"--------------------"<<endl;
      cout<<"    1.复习生词     "<<endl;
      cout<<"    2.顺序记忆     "<<endl;
      cout<<"    3.乱序记忆     "<<endl;
      cout<<"    4.退出         "<<endl;
      cout<<"--------------------"<<endl;
      cout<<"选择（0-4）:";
      cin>>selection;
      getchar();
      switch(selection)
      {
         case 1: revise_mode(unknown,cnt);
                  break;
         case 2: inorder_mode(wordlist,n,unknown,cnt);
                 break;
         case 3: outoforder_mode(wordlist,n,unknown,cnt);
                 break;
```

```
            case 4: break;
        }
        if(selection==4)
        {
            break;
        }
        else
        {
            continue;
        }
    }
    return ;
}
```

2.void revise_mode(struct Word unknown[],int &cnt)；

设计思路：

这个函数实现的是复习模式的背词方式。首先从生词库导入用户的生词，逐一呈现给用户，用户可选择已掌握和留在生词库来决定是否将该生词从生词库中删除。

函数代码：

```
void revise_mode(struct Word unknown[],int &cnt)
{
   system("cls");
   int d[1000]={0};
   if(cnt==0)
   {
      cout<<endl<<"您没有可以复习的单词哦！"<<endl;
   }
   else
   {
      cout<<"\n------------------------------  复  习  模  式  ----------------------------------"<<endl;
      cout<<endl<<endl;

      start=clock();
      for(int i=0;i<cnt;i++)
      {
         cout<<setw(20)<<"单词"<<setw(10)<<"词性"<<setw(31)<<"中文意思"<<"笔记"<<endl;

   cout<<"------------------------------------------------------------------------------------"<<endl;

   cout<<setw(20)<<unknown[i].word<<setw(10)<<unknown[i].word_class<<setw(30)<<unknown[i].word_meaning<<unknown[i].word_note<<endl;

   cout<<"------------------------------------------------------------------------------------"<<endl;
```

```
            cout<<endl<<"A.已掌握      B.保留在生词库      C.退出复习模式"<<endl;
            cout<<"\n 请输入你的选择："; 
            char sel=0;
            cin>>sel;
            switch(sel)
            {
                case 'a':
                case 'A': cout<<"\n 你真聪明！\n\n\n";
                          d[i]=1;
                          break;
                case 'b':
                case 'B': cout<<"\n 继续加油！\n\n\n";
                          break;
                case 'c':
                case 'C': goto out;
            }
            cout<<endl;
        }
        end=clock();
        getchar();
        cout<<" 恭 喜 您 已 复 习 完 所 有 的 生 词 , 用 时 "<<double(end-start)/CLOCKS_PER_SEC<<"秒。";
        goto out;
    }
    out:
    int cnt_0=cnt;
    for(int i=0;i<cnt_0;i++)
    {
        if(d[i]==1)
        {
            deleteword_bdc(unknown,cnt,i);
        }
    }//删除已经记住的单词
```

```
    savefile_bdc(unknown,cnt);
    cout<<"\n\n 按任意键返回背单词模式菜单...";
    getchar();
    return;
}
```

3.void inorder_mode(struct Word wordlist[],int n,struct Word unknown[],int &cnt);

设计思路：

这是顺序记忆的方式。首先用户输入从哪个单词开始，接着会按照字典顺序为用户逐一呈现其后面的单词。同时用户可选择已掌握或者添加入生词库来接着背下一个单词。

函数代码：

```
void inorder_mode(struct Word wordlist[],int n,struct Word unknown[],int &cnt)
{
   while(1)
   {
      system("cls");
      cout<<"\n------------------------------ 顺 序 记 忆 模 式 ------------------------------"<<endl;
      cout<<endl;
      char _word[20];
      char _class[10];
      cout<<"请输入你想要开始的单词（按回车结束输入）：";
      cin>>_word;
      cout<<"请输入该单词的词性（按回车结束输入）：" ;
      cin>>_class;
      int dofind=find_word(wordlist,n,_word,_class);
      if(dofind==-1)
      {
         cout<<endl<<"抱歉没有找到该单词,请按任意键重新输入。\n";
         getchar();
         getchar();
         continue;
      }
      else
      {
         cout<<endl<<endl;
         start=clock();
```

```
            for(int i=dofind;i<n;i++)
            {
                cout<<setw(20)<<"  单  词  "<<setw(10)<<"  词  性
"<<setw(31)<<"中文意思"<<"笔记"<<endl;

    cout<<"-----------------------------------------------------------
----------------"<<endl;

    cout<<setw(20)<<wordlist[i].word<<setw(10)<<wordlist[i].word_clas
s<<setw(30)<<wordlist[i].word_meaning<<wordlist[i].word_note<<endl;

    cout<<"-----------------------------------------------------------
----------------"<<endl;
                cout<<endl<<"A.已掌握      B.添加入生词库      C.退
出顺序记忆模式"<<endl;
                cout<<"\n 请输入你的选择："；
                char sel=0;
                cin>>sel;
                switch(sel)
                {
                    case 'a':
                    case 'A': cout<<"\n 你真聪明！\n\n\n";
                              break;
                    case 'b':
                    case 'B': cout<<"\n 继续加油！\n\n\n";
                              addword_bdc(unknown,cnt,wordlist[i]);
                              break;
                    case 'c':
                    case 'C': goto out;
                }
                cout<<endl;
            }
            end=clock();
            getchar();
```

```
                cout<<" 恭 喜 您 已 背 穿 词 库 ， 用 时
"<<double(end-start)/CLOCKS_PER_SEC<<"秒。";
                goto out;
            }
        }
        out:
        savefile_bdc(unknown,cnt);
        cout<<"\n\n 按任意键返回背单词模式菜单...";
        getchar();
        return;
    }
```

4.void outoforder_mode(struct Word wordlist[],int n,struct Word unknown[],int &cnt)；

设计思路：

此函数不同于上一个函数的地方在于，这里每次给出的单词是随机的，不用的用户可以根据自身的需求选择更适合的模式。

函数代码：

```
void outoforder_mode(struct Word wordlist[],int n,struct Word unknown[],int &cnt)
{
   system("cls");
   srand(time(0));
   cout<<"\n------------------------------ 乱 序 记 忆 模 式 ---------------------------------"<<endl;
   cout<<endl<<endl;
   while(1)
   {
      int rad=rand()%n;
      cout<<setw(20)<<"单词"<<setw(10)<<"词性"<<setw(31)<<"中文意思"<<"笔记"<<endl;

   cout<<"-------------------------------------------------------------------------------"<<endl;

   cout<<setw(20)<<wordlist[rad].word<<setw(10)<<wordlist[rad].word_class<<setw(30)<<wordlist[rad].word_meaning<<wordlist[rad].word_note<<endl;

   cout<<"-------------------------------------------------------------------------------"<<endl;
      cout<<endl<<"A.已掌握      B.添加入生词库      C.退出顺序记忆模式"<<endl;
      cout<<"\n 请输入你的选择：";
      char sel=0;
      cin>>sel;
```

```
        getchar();
        switch(sel)
        {
            case 'a':
            case 'A': cout<<"\n 你真聪明! \n\n\n";
                      break;
            case 'b':
            case 'B': cout<<"\n 继续加油! \n\n\n";
                      addword_bdc(unknown,cnt,wordlist[rad]);
                      break;
            case 'c':
            case 'C': goto out;
        }
        cout<<endl;
    }
    out:
    savefile_bdc(unknown,cnt);
    cout<<"\n\n 按任意键返回背单词模式菜单...";
    getchar();
    return;
}
```

四、结果分析

（一）.注册

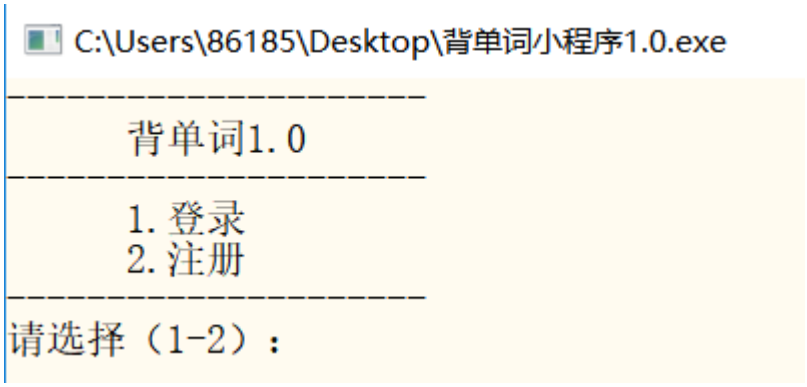

图 4.1.1　登录注册界面

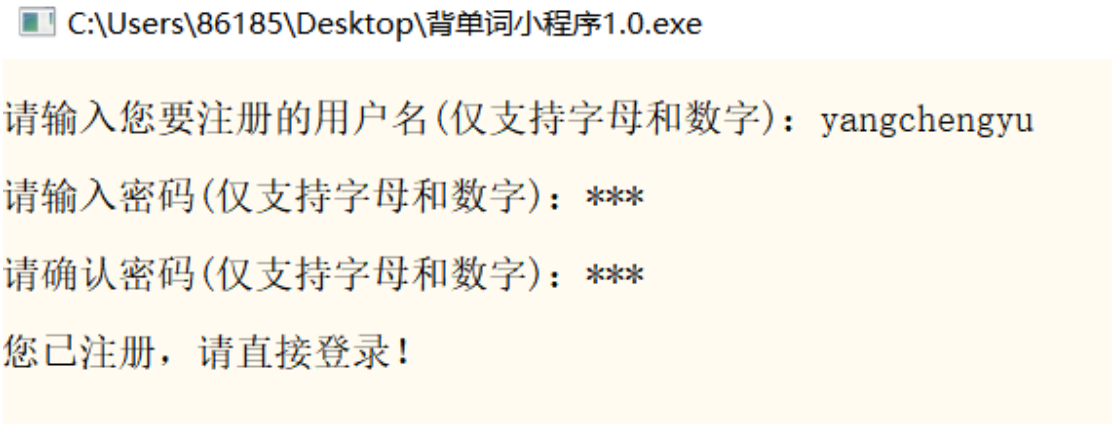

图 4.1.2 已注册

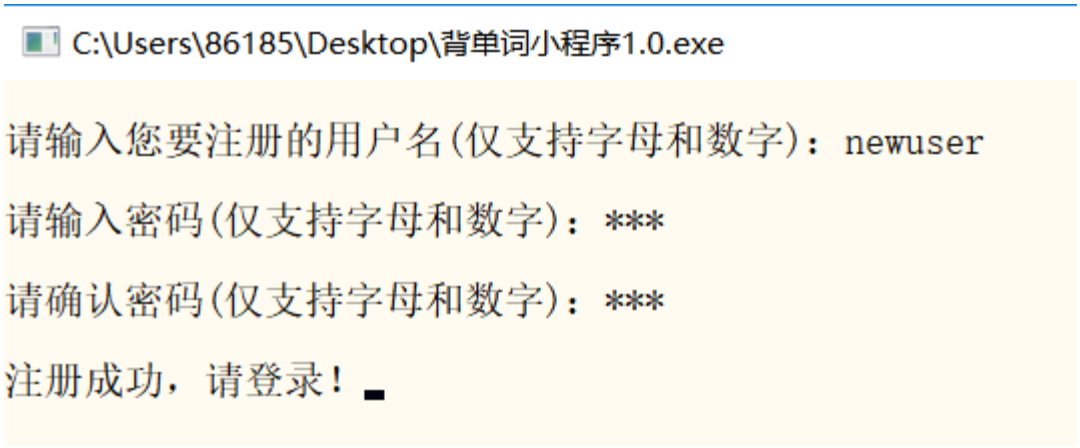

图 4.1.3 注册成功

（二）.登录

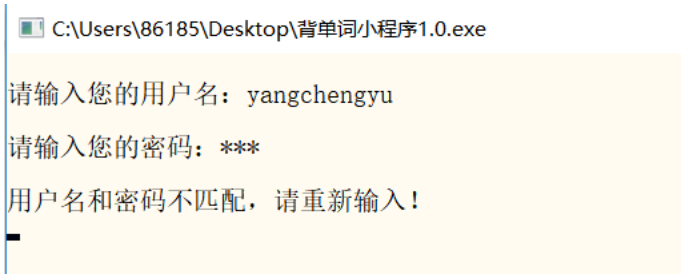

图 4.2.1 密码错误

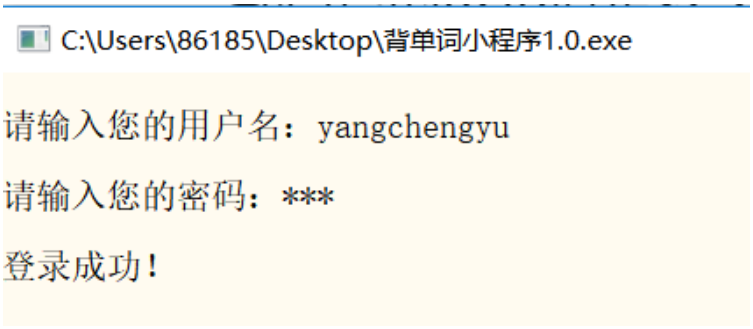

图 4.2.2 登录成功

（三）.主菜单

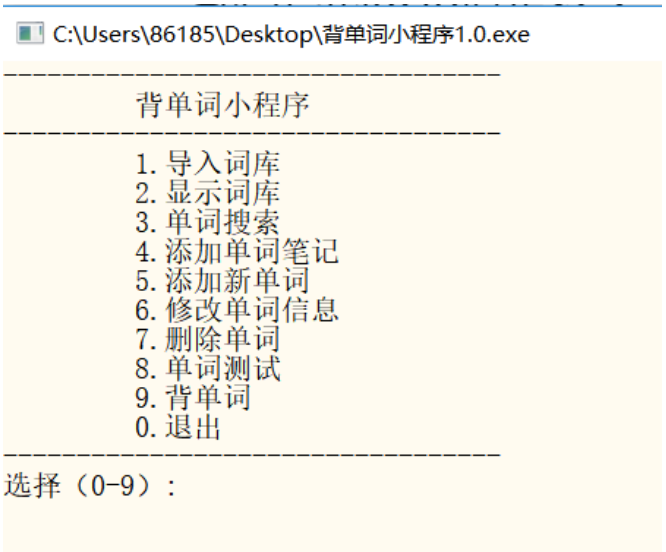

图 4.3 主菜单

（四）.导入词库

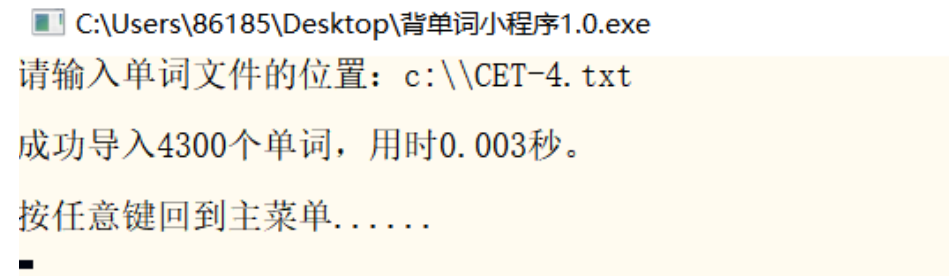

图 4.4.1　导入成功

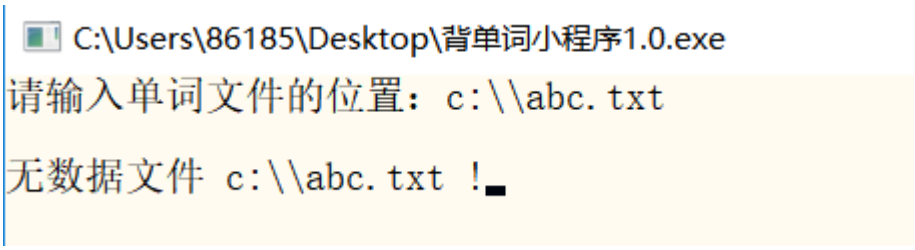

图 4.4.2　导入失败

（五）.显示词库

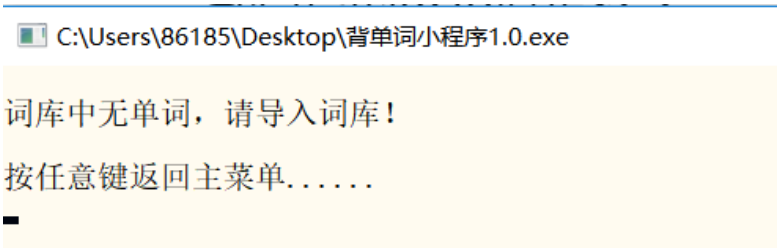

图 4.5.1　没有词可显示

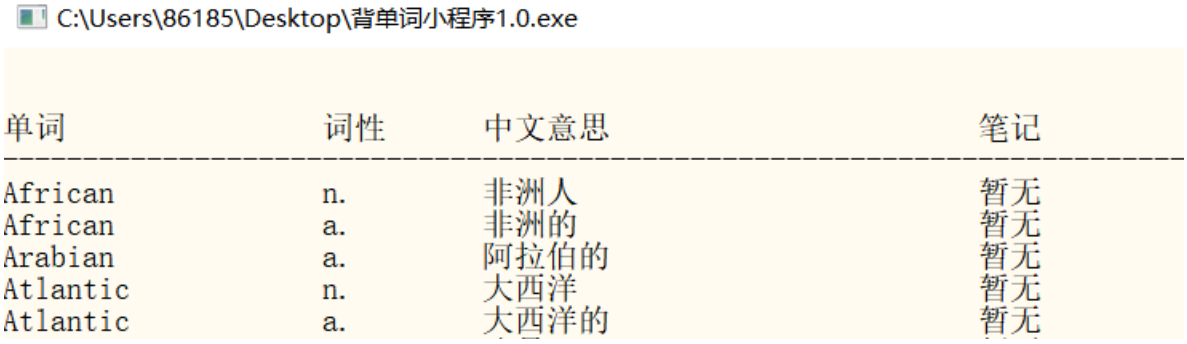

图 4.5.2　词库的开头

```
youthful        a.      年轻的，青年的                  暂无
zeal            n.      热心，热情，热忱                暂无
zealous         a.      热心的，热情的                  暂无
zebra           n.      斑马                            暂无
zero            n.      零；零点，零度                  暂无
zone            n.      地区，区域                      暂无

单词信息已显示完毕！

一共显示4300个单词，用时17.357秒。

按任意键返回主菜单.....
```

图 4.5.3　词库的结尾

（六）.搜索

```
C:\Users\86185\Desktop\背单词小程序1.0.exe

----------------------
        搜索
----------------------
    1.输入英文搜索
    2.输入中文搜索
    3.英文精确搜索
    4.快速搜索
    5.退出搜索
----------------------
请输入你的选择（1-5）：
```

图 4.6.1　搜索菜单

```
C:\Users\86185\Desktop\背单词小程序1.0.exe

请输入你想要查询的单词（区分大小写并按回车结束输入）：abstract
词性        中文意思                                笔记
--------------------------------------------------------
n.          摘要                                    暂无
a.          抽象的                                  暂无

以上共2条搜索结果，用时0.011秒。

输入0返回搜索菜单，输入1继续查找（请按回车键结束输入）：
```

图 4.6.2　输入英文搜索

```
C:\Users\86185\Desktop\背单词小程序1.0.exe

请输入你想要查询的单词（区分大小写并按回车结束输入）：dhfelhg

很抱歉没有找到您要查询的单词。

输入0返回搜索菜单，输入1继续查找（请按回车键结束输入）：
```

图 4.6.3　没有搜索到

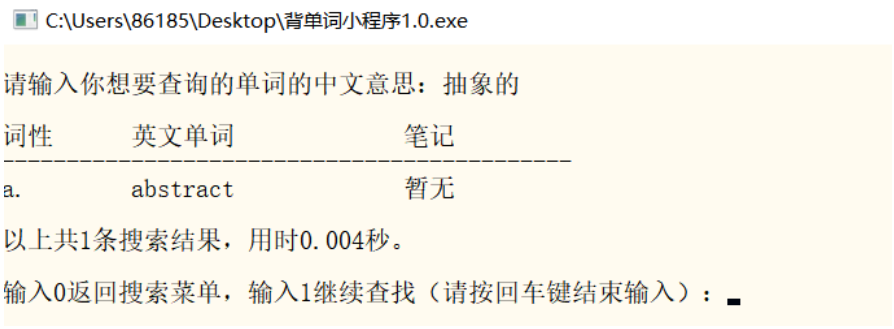

图 4.6.4　输入中文搜索

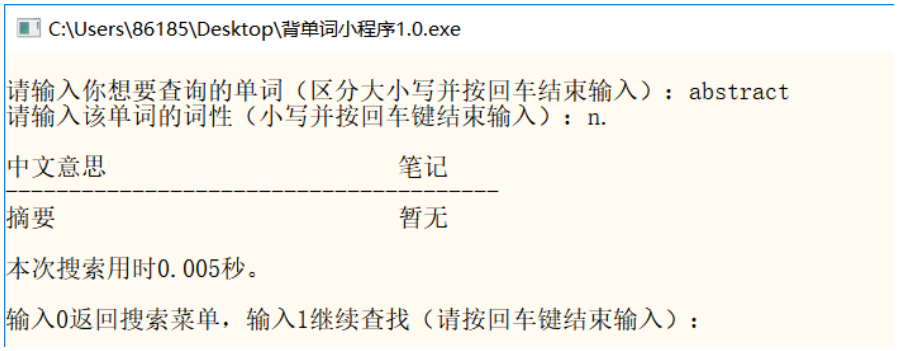

图 4.6.5 输入英文精确搜索

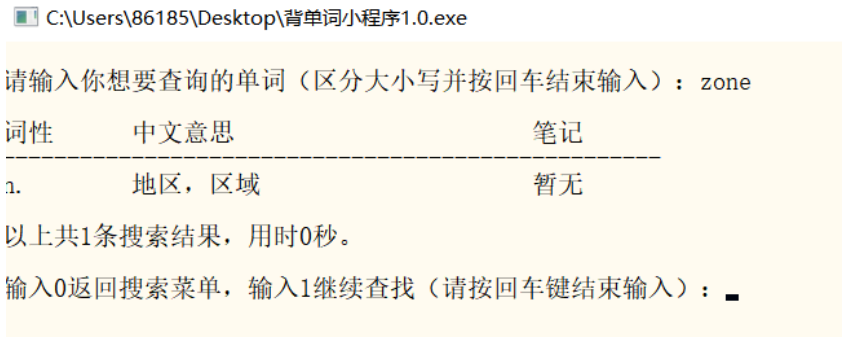

图 4.6.6　快速搜索

（七）.添加笔记

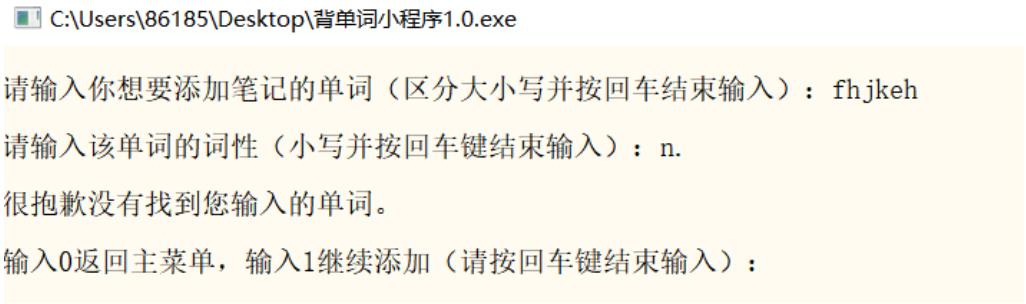

图 4.7.1　添加失败

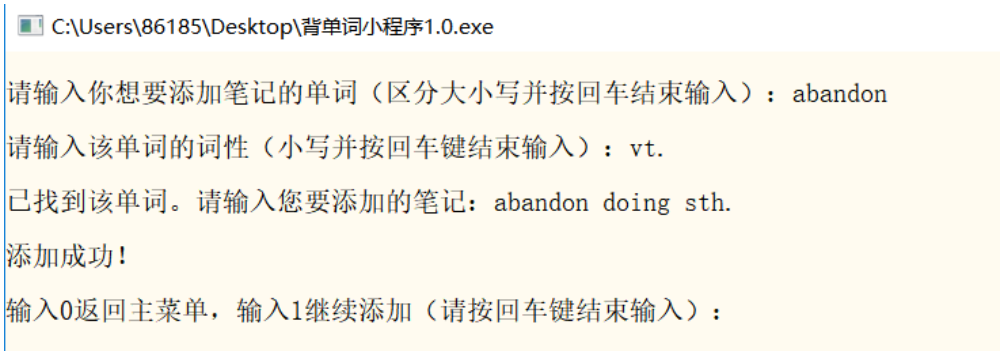

图 4.7.2　添加成功

（八）.添加新单词

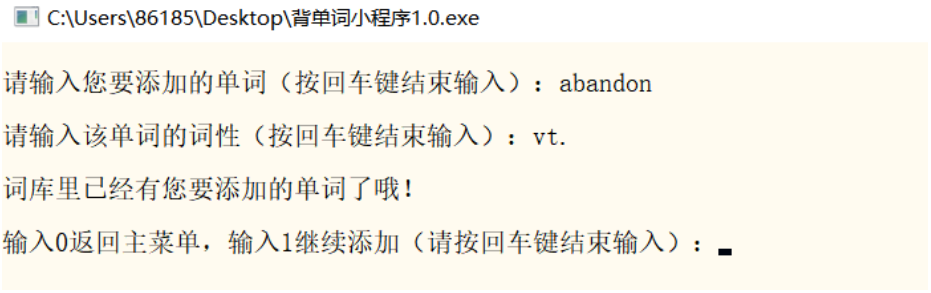

图 4.8.1　重复添加

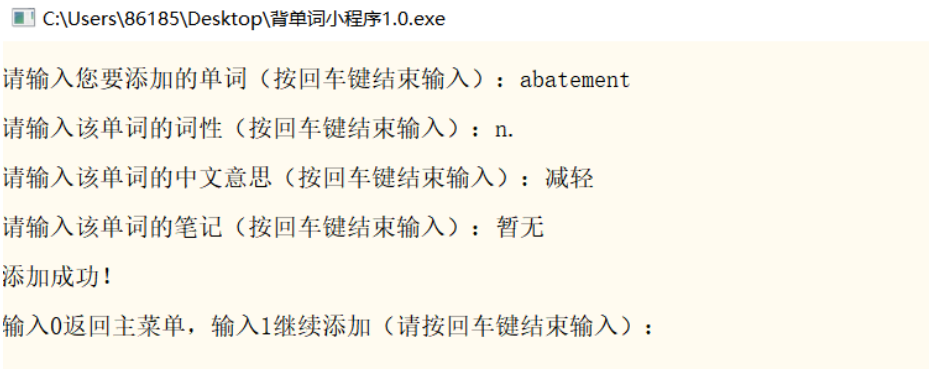

图 4.8.2　添加成功

（九）.修改单词信息

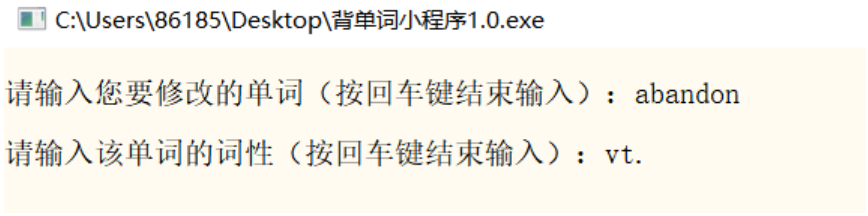

图 4.9.1　输入要修改的单词

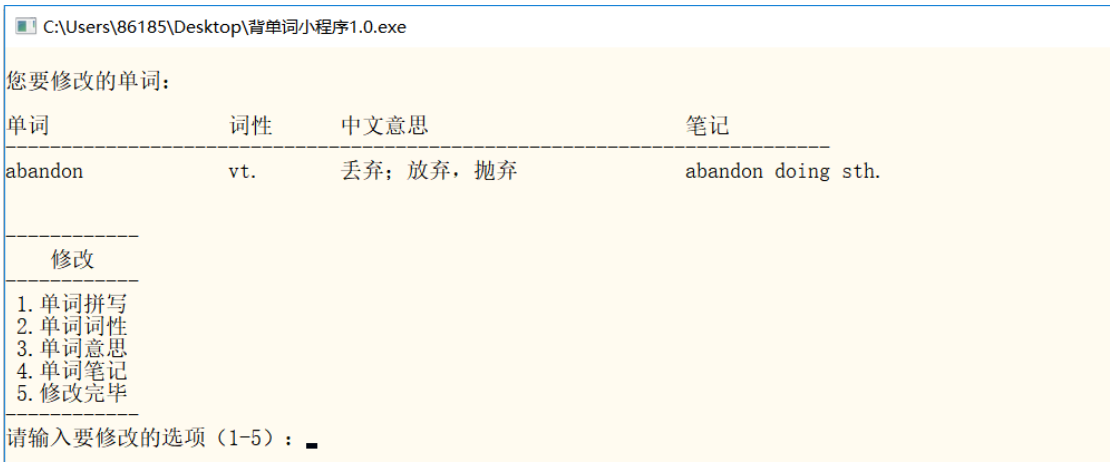

图 4.9.2　修改菜单

（十）.删除词条

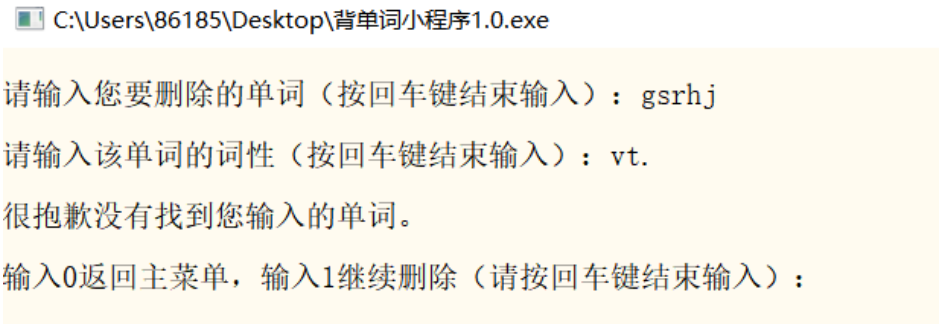

图 4.10.1　删除失败

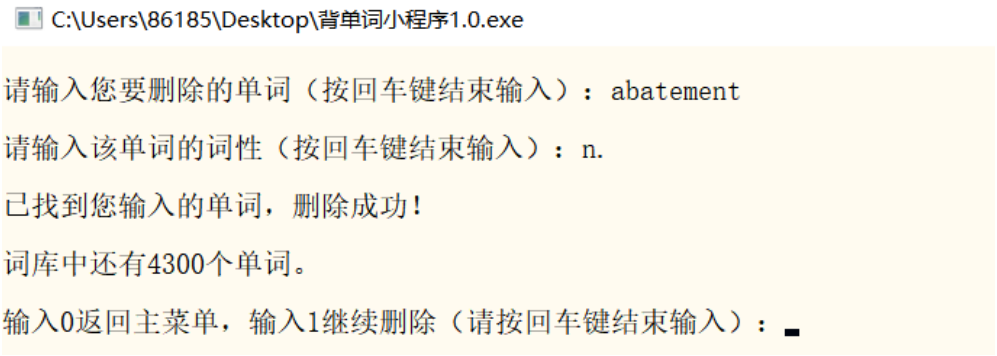

图 4.10.2　删除成功

（十一）.测试

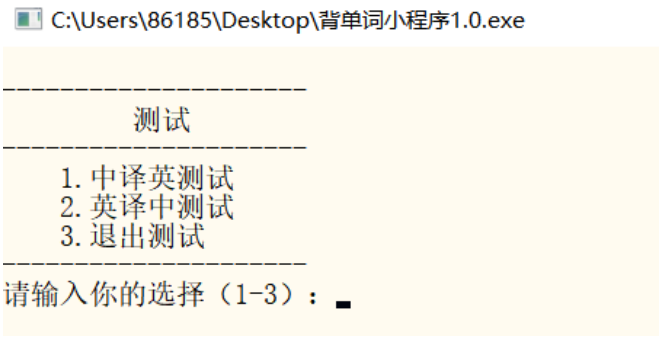

图 4.11.1 测试菜单

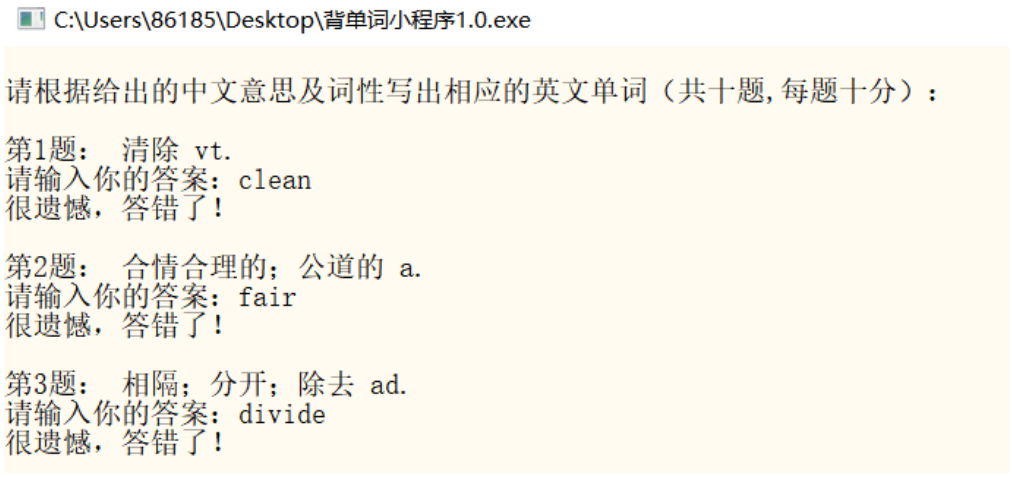

图 4.11.2 中译英测试

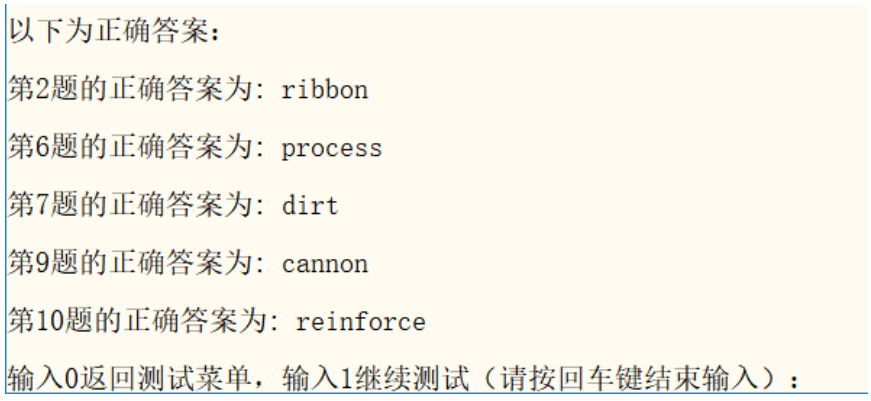

图 4.11.3　中译英测试反馈

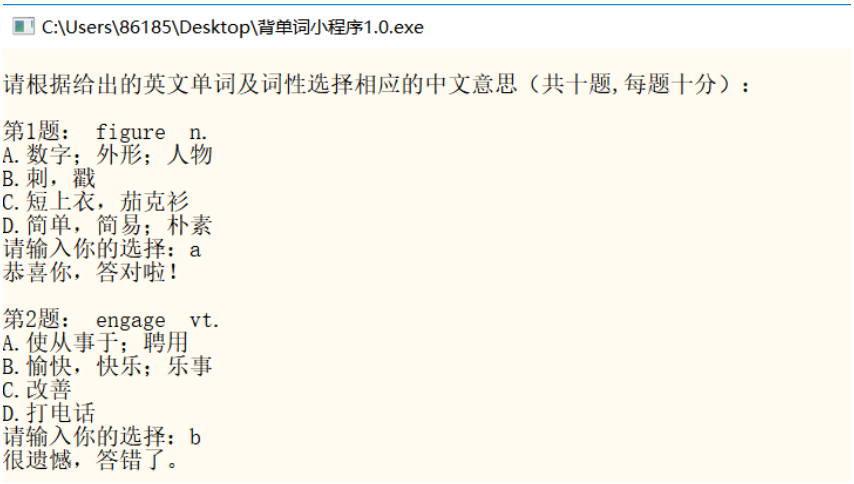

图 4.11.3　英译中测试

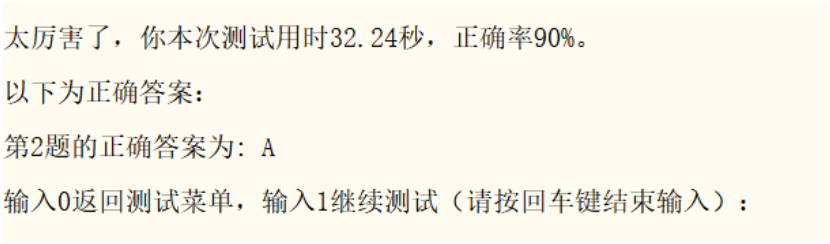

图 4.11.4　英译中测试反馈

（十二）.背单词

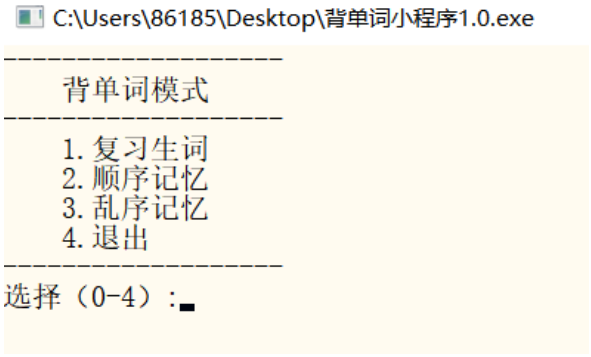

图 4.12.1　背单词模式菜单

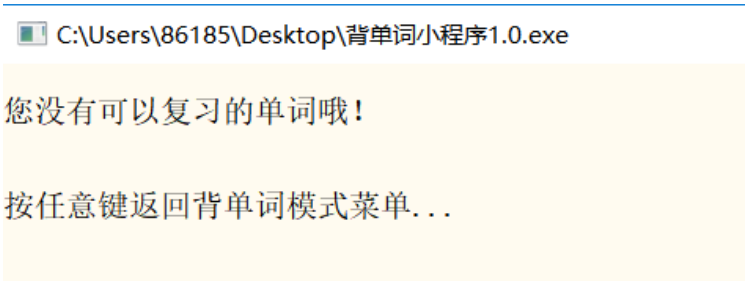

图 4.12.2　生词库中没有单词

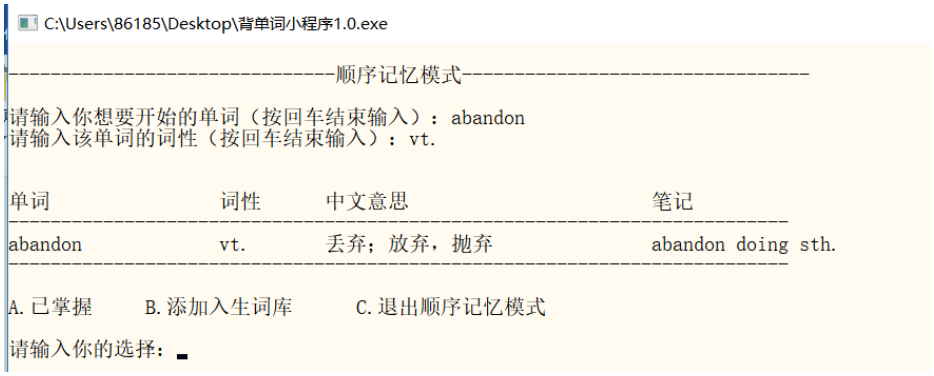

图 4.12.3　顺序记忆模式

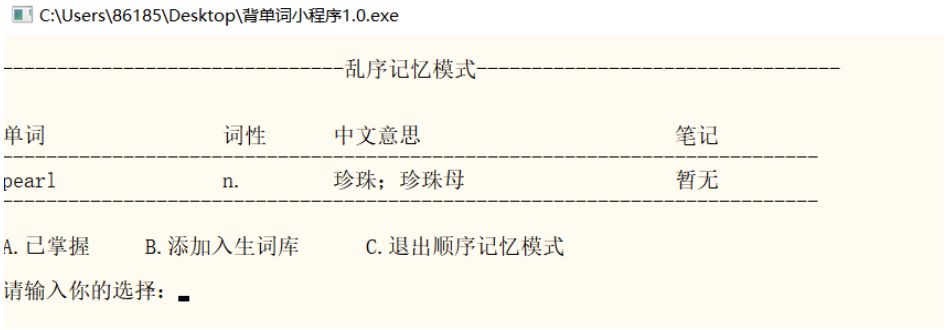

图 4.12.4　乱序记忆模式

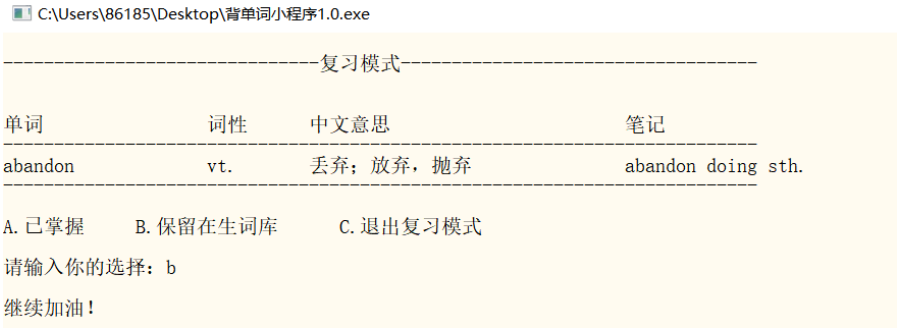

图 4.12.5　复习模式

五、课程设计总结

这一次课程设计让我更加深入地了解了 C++这门语言，也学到了很多知识，同时也锻炼了自己独立思考，独立解决问题的能力。

同时，也遇到了许多的困难。首先，如何从文本文件读入单词信息就是一个很大的难题。由于原始的词库里有不少的错误，格式繁多不方便读取，我花了几个小时的时间逐个去把单词的格式规范化，方便最终文件的读取。

此外，我用的编译器是 Dev-C++，一开始调试的时候，编译没问题，但怎么也运行不起来。后来把代码复制到 VS 里，它给出了这样的错误信息：

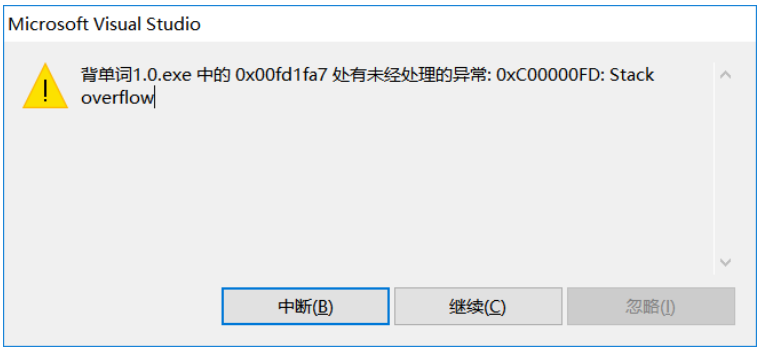

图 5.1　VS 给出的错误信息

后来，我通过查阅资料，得知是数组开的太大导致栈溢出，于是把局部变量的数组定义为静态局部变量，问题就迎刃而解了。

还有一次遇到的困难是，虽然用了 ios::in,但是当文件不存在时确不能新建该文件，于是在后面用了一个条件判断语句，当文件打不开时，新建一个同名的文件夹即可。

参考文献

1. ⊙7-302-08599-4 C++程序设计，清华大学出版社，2004.06
2. 完美 C++（第 5 版）[美] Walter Savitch，[美] Kenrick Mock 著，薛正华 译