上海财经大学《程序设计基础》上机考试卷(A)

（ 2018 至 2019 学年 第 1 学期 ）

学号　　姓名　　机器号

答题及答案提交要求：

(1) 务必在 D 盘上创建一个**答题文件夹（目录）**，该文件夹的名字用你的学号+姓名命名，如你的学号为 2016000123, 姓名为“李明”，则你需要创建一个目录“2016000123 李明”。**将你做的答案保存在 D 盘你的文件夹下。**

(2) 请将你修改完成后的 **1.cpp, 2.cpp, 3.cpp, 4.cpp, 5.cpp, 6.cpp, 7.cpp** 保存在 D 盘你的文件夹下。

注意事项： **以下试题中没留出足够空行，应根据需要确定代码长度。**

1. (循环控制。 15 分）根据输入整数 N（奇数），输出 N 行菱形（上下对称）。

例如，N=7 时，输出图案如下。

```
   *
  ***
 *****
*******
 *****
  ***
   *
```

```
#include <iostream>
using namespace std;
int main()
{
    double N,x,y,z;
    cout << "Please input the number of rows(N>0): ";
    cin >> N;
    //请补充完成以下缺少的代码  15 分
    for(x=0;x<N/2;++x){                    //上三角正确 5 分
        cout<<endl;
        for(z=0;z<N/2-x-1;++z)
          cout<<" ";
        for(y=0;y<2*x+1;++y){cout<<"*";}
        }
     for(x=1;x<=N/2;x++)                   //下三角正确 5 分
        {
        cout<<endl;
        for(y=1;y<=x;y++) //
          cout<<" ";
        for(z=1;z<=N-2*x;z++)
```

```
            cout<<"*";
        }

    return 0;
}
  // 结果正确 5 分
```

2. (函数调用。 15 分）设计一个支持整型、实型和字符型数据的选择排序的函数模板。

```
#include <iostream>

using namespace std;
// 请补充函数模板代码 15 分
template<class T>        //   函数模板正确 5 分
void sort1(T data[],int x)
{int i,j,k;                          //   选择排序算法正确 5 分
 T tmp;
 for(i=0;i<x;++i){
     k=i;
     for(j=i;j<x;++j)
         if(data[j]<data[k])
         {tmp=data[j];
          data[j]=data[k];
          data[k]=tmp;
          }
     }
}
// 运行正确 5 分

int main()
{
   int i,x=10,a[ ]={1,0,8,9,6,7,2,4,3,5};
   sort1(a,x);
   for(i=0;i<10;++i){cout<<a[i] <<"    ";};
   cout<<endl;

   double b[ ]={1.2,0.8,8.1,5.7,9.2,3.8,20.2,15.6,5.0,16.9};
   sort1(b,x);
   for(i=0;i<10;++i){cout<<b[i] <<"    ";};
   cout<<endl;

   char c[ ]={'Z','s','P','3','*','q','L','O','z','$'};
   sort1(c,x);
   for(i=0;i<10;++i){cout<<c[i] <<"    ";};
```

```
    cout<<endl;

    return 0;
}
```

3. (改错题，数组与字符串，15 分)

输入一个字符串，把字符串中所有的大写字母转换为小写字母，并且输出字符串中的整数。例如输入字符串“AAAAA3456bb32“，输出字符串：“aaaaa3456bb32“和“345632“。以下是存在错误的程序，请改正程序中的逻辑错误。

```
#include <iostream>
#include <cstring>
using namespace std;
int main()
{
    int j;
    int n=0;
    char str[80];
    cin.getline(str,80);

    for(j=0;str[j]!='\0';j++)                    //修改为 str[j]!='\0' ， 5 分
    {
        if(str[j]>='A'&&str[j]<='Z')   // ||修改为&&， 3 分
            str[j]=str[j]-'A'+'a';      //==修改为=，2 分

        if(str[j]>='0'&& str[j]<='9')   //||修改为&& ，2 分
            n=n*10+str[j]-'0';          //修改为-'0'，3 分
    }
    cout<<str<<endl;
    cout<<n<<endl;

    return 0;
}
```

4. (改错题，指针与动态数组，15 分)

输入 5 个字符串（每个字符串的长度最长为 10），输出其中最短字符串的长度。例如，输入字符串：“aaaaa“，“bbbb“，“ccc“，“dd“，“e“，输出结果为 1。以下是存在错误的程序，请改正程序中的逻辑错误。

```
#include <iostream>
#include <cstring>     //添加语句（5 分）
```

```
using namespace std;
int minlen(char *str[], int n);

int main()
{
    int i;
    char *str[5]={};

    for(i=0;i<5;++i)
    {
    str[i]=new char[11];   //添加语句（5 分）
    cin>>str[i];
    }
    cout<<"这组字符串最小长度为"<<minlen(str,5)<<endl;

     for(i=0;i<5;i++)
     {
       delete [] str[i];
     }

    return 0;
}

int minlen(char *str[], int n)
{
    int i;
    unsigned int minlen;

    minlen=strlen(str[0]);   //修改语句（3 分）

    for(i=1;i<n;++i)
    {

    if(minlen>strlen(str[i]))     //修改语句（1 分）
        minlen=strlen(str[i]);   //修改语句（1 分）
    }

    return minlen;
 }
```

5.（递归。 12 分）

编写程序，以递归的形式，将一个整数逆序输出。例如，给定整数 1230，则逆序输出结果为 0321。注

意一条输出占用一行。

```
#include <iostream>
using namespace std;

void reverse_print(int n);
int main()
{
    reverse_print(12345678);
    reverse_print(12300);
    reverse_print(12300123);

    return 0;
}

void reverse_print(int n)
{
    // 请将该函数补充完整，共 12 分
    cout << n % 10;                 // 正确输出个位数值，3 分；
    if(n / 10 != 0){                // 正确写出递归条件，3 分；
        reverse_print(n / 10);      // 正确递归调用，3 分；
    }else{
        cout << endl;               // 输出换行符，3 分。
    }
}
```

6.（类。 12 分）

在数组许多应用中，经常不能事先确定数组的大小。因此，本题需要你建立一个元素为整数的动态数组类型，该类型可以根据元素个数动态地扩充数组的大小，即当元素个数超过数组现有最大允许个数时，该类型可以自动扩充数组的存储空间的大小。请你构建一个类来实现这种新的数据类型，该类提供如下功能：

1） 产生一个动态数组。即申请一块动态内存，用于存储 N 个整数，N 的默认值是 2

DynamicIntArray(int N=2);

2） 产生对某动态数组的一个拷贝

DynamicIntArray(const DynamicIntArray & x);

3）在数组中增加新的元素

void addelem(int elem); // 将 elem 添加到数组当前最后一个元素的后面

4）删除数组中的某个元素

void delelem(int idx);

// 如果存在下标为 idx 的元素，则删除该元素。

// 否则打印提示越界，并显示当前数组中的元素个数。注：下标默认从 1 开始。

5） 查找给定的元素

int findelem(int elem) const;
// 在数组中查找第一次出现 elem 的位置，如果找到,返回该位置元素的下标，
// 否则输出-1， 注：下标默认从 1 开始。

6） 设定元素的值

void setelem(int idx, int elem);
// 将下标为 idx（下标从 1 开始）的元素设置为 elem

7） 查找分配给数组的存储空间的大小（注：此函数设定为私有的）

int getTotalNum() const;
// 返回数组能够存放的整数的个数；

8）查找数组中元素的个数

int getElemNum() const; // 返回数组中元素的个数

9） 打印动态数组的信息

void display() const; // 打印数组元素个数，依次打印数组中的每个元素

10）析构函数，释放数组的存储空间

~DynamicIntArray();

该工程由 3 个文件构成：dynamicarray.h, 6.cpp, main.cpp。

请完成 6.cpp 中的

（1）拷贝构造函数 DynamicIntArray(const DynamicIntArray & x);

（2）void addelem(int elem);

（3）void delelem(int idx);

补充其中缺少的代码。

// DynamicIntArray 类的头文件 dynamicarray.h 内容如下

```
#ifndef DYNAMICARRAY_H_INCLUDED
#define DYNAMICARRAY_H_INCLUDED

class DynamicIntArray {
    private:
        int totalNum;
        int elementNum;
        int *storage;
    public:
        DynamicIntArray(int N=2);
        // 构造函数，申请一块动态内存，用于存储 N 个整数，给 totalsize, realsize 赋值
        DynamicIntArray(const DynamicIntArray & x);     //拷贝构造函数
        void addelem(int elem); // 将 elem 添加到数组的最后
        int findelem(int elem) const;
        // 在数组中查找第一次出现 elem 的位置，如果找到,返回该位置元素的下标，
        //否则输出-1.下标默认从 1 开始。
        void delelem(int idx);
        // 如果存在下标为 idx 的元素，则删除该元素，
        //否则打印提示越界，并显示当前数组中的元素个数。注：下标默认从 1 开始。
        void setelem(int idx, int elem);
```

```
        // 将下标为 idx（从 1 开始）的元素设置为 elem
        int getElemNum() const;
        // 返回数组中元素的个数；
        void display() const; // 依次打印数组中的每个元素
        ~DynamicIntArray(); //析构函数

    private:
        int getTotalNum() const;
        // 返回数组中存储空间的大小（此处是能够存放的整数的个数）；
};
#endif
```

// DynamicIntArray 类的实现在 6.cpp

```
#include <iostream>
#include "dynamicarray.h"
using namespace std;

DynamicIntArray::DynamicIntArray(int N)
{
    // 构造函数，申请一块动态内存，用于存储 N 个整数，给 totalsize, realsize 赋值。
    totalNum=N;
    elementNum=0;
    storage= new int[N];
}

DynamicIntArray::DynamicIntArray(const DynamicIntArray & arr)
{
    // 拷贝构造函数，申请一块动态内存，用于存储数组 arr 中的元素，
    // 并给 totalsize, realsize 赋值。
    // 请补充完成以下缺少的代码   4 分
    totalNum=arr.getTotalNum();              //    1 分
    elementNum=arr.getElemNum();             //    1 分
    storage= new int[totalNum];              //    1 分
    for(int i=0;i<elementNum;i++)
        storage[i]=arr.storage[i];           //    1 分
}
void DynamicIntArray::addelem(int elem){
    // 将 elem 添加到数组的最后
    if(elementNum<totalNum){
      // 如果当前存储空间还没有用完，将 elem 的值添加到数组的末尾
      // 修改 elementNum 的值
      storage[elementNum]=elem;
      elementNum++;
    }
```

```
    else {
        // 如果当前存储空间已经用完了，申请一块新空间，该空间比原空间大一倍
        // 将数组的值依次序拷贝到新申请的空间中
        // 释放掉数组原来占据的空间
        // 修改 storage 指向新的空间
        // 增加新元素  elem
        // 修改 realsize 的值，修改 totalsize 的值,
        int * intp;
        // 请补充完成以下缺少的代码   6 分
        intp=new int[2*totalNum];                //    1 分
        for(int i=0;i<elementNum;i++){
            intp[i]=storage[i];                  //     2 分
        }
        delete [] storage;                       //    1 分
        storage = intp;                          //    0.5 分
        storage[elementNum]=elem;                //    0.5 分
        elementNum++;                            //    0.5 分
        totalNum=2*totalNum;                     //    0.5 分
    }
}

int DynamicIntArray::findelem(int elem) const{
    // 在数组中查找第一次出现 elem 的位置，如果找到，返回该位置元素的下标(规定下标从 1 开始)，
    // 否则输出-1;
    int i;
    for(i=0;i<elementNum;i++){
        if(storage[i]==elem) return i+1;
    }
    return -1;
}

void DynamicIntArray::delelem(int idx){
    // 如果存在下标为 idx 的元素，则删除该元素,否则打印提示，数组越界，
    // 并显示当前的数组的元素个数
    if(idx < 1 || idx > elementNum)
    {
        cout << "index:" << idx << " out of range! failed...";
        return;
    }
    int elemdel=storage[idx-1];
    // 请补充完成以下缺少的代码   2 分
    for(int i=idx-1;i<elementNum;i++){           //    1 分
        storage[i]=storage[i+1];
    }
```

```
        elementNum--;                                   //     1 分

        cout << "the " << idx <<"-th element " << elemdel << " has been deleted.\n";
}

int DynamicIntArray::getTotalNum() const{
        return totalNum;
}

int DynamicIntArray::getElemNum() const{
        return elementNum;
}

void    DynamicIntArray::setelem(int idx, int elem){
        storage[idx-1]=elem;
}

void DynamicIntArray::display() const{
        // 依次打印数组中的每个元素
        int i;

        cout << "the allocated total memory size is :" << totalNum << endl;
        cout << "the number of elements in the array is :" << elementNum<< endl;
        if(elementNum>0)
                cout << "the elements in the array are:" << endl;
        for(i=0;i<elementNum-1; i++) {
                cout << storage[i] << ",";
        }
        if(elementNum>0) cout << storage[i]<< endl;
}

DynamicIntArray::~DynamicIntArray(){
        delete [] storage;
}
```

// 应用程序 main.cpp

```
#include <iostream>
#include <stdlib.h>
#include "dynamicarray.h"
using namespace std;

int main()
{
        DynamicIntArray arr1;
```

```
int choice;
int x, idx, i;

// 以下循环，通过人机交互过程构建数组对象 arr1
while(true){
    cout << "\nPlease make your choice:\n";
    cout << "1. Add an integer\n";
    cout << "2. Delete an element\n";
    cout << "3. Display the entire array\n";
    cout << "4. Find an element in the array\n";
    cout << "5. Quit\n";
    cin >> choice;

    switch(choice){
        case 1: cout << "please input the integer to be added: ";
                cin >> x;
                arr1.addelem(x);
                break;
        case 2: cout << "please input the index of the element to be deleted:";
                cin >> idx;
                arr1.delelem(idx);
                break;
        case 3: arr1.display();
                break;
        case 4: cout << "please input the the element to be found:";
                cin >> x;
                i=arr1.findelem(x);
                if(i==-1)
                    cout << x << " does not exist in the array!\n";
                else
                    cout << x << " is the " << i << "-th element\n";
                break;
        default: break;
    }
    if(choice==5) break;
}

// 以下代码根据 arr1, 新建数组对象 arr2, 分别显示 arr1, arr2 的内容
if(arr1.getElemNum()>0){
    DynamicIntArray arr2=arr1;
    arr2.setelem(1,-1);
    cout << "the first array is:\n";
        arr1.display();
    cout << "the second array is:\n";
```

```
            arr2.display();
    }
}
```

7. （链表。16 分）

本题是一个简单的文本文件单词分析程序，其中使用链表作为数据结构来存储单词信息。程序打开 TestData.txt 文件，从文件中逐行读取字符串，提取单词并建立单词表，并分析所有单词信息，包括单词的字符串、单词出现的次数，以及单词在文件中的位置（所在的行号和列号）等。**注意：有些单词可能会出现多次，本题中也使用链表来保存单词的多个位置信息。**

单词表、单词和位置的数据结构设计如下：

```
struct Position{       // 单词位置结构体
    int lineNo;        // 单词在文件中所在行的行号
    int columnNo;      // 单词第一个字母所在列的列号
    Position *next; // 指向下个位置结点的指针
};
struct Word{           // 单词结点的结构体
    char *str;         // 单词字符串指针
    int occurence; // 单词在文件中出现的次数
    Position *ps_head,*ps_rear; // 单词位置链表的头指针和尾指针
    Word *next;        // 指向下个单词结点的指针
};
struct WordList{       // 单词表的结构体
    int typeCounter;   // 单词种类数量
    Word *wl_head,*wl_rear; // 单词链表的头指针和尾指针
};
```

程序完成如下功能：

1） 通过 createWordList 函数建立包含所有单词的单词表：首先从文件每次读入一行字符串，然后扫描并逐个提取其中的单词，如果提取的单词不在单词表中，生成相应的单词结点，并插入到单词链表的尾部。否则，只要将提取单词的位置信息插入到表中单词的位置链表的尾部。例如，假设文本文件内容如下：

"**The** unknown as we know,
there are known knowns.
There are things we know we know.
We also know there are known unknowns.
That is to say we know there are some things we do not know.
But there are also unknown unknowns,
the ones we do not know we do not know."

我们注意到单词 the 在第 0 行和第 6 行都出现了。当程序开始运行时，单词表是空的，不包含任何单词结点。程序从第 0 行开始扫描，会首先扫描并提取“the”这个单词字符串，通过 searchWordList 函数判断出单词表中没有 the 单词结点（单词表实际上是空的），程序就会生成 the 单词结点，并插入到单词链表中，同时生成一个 the 的位置结点，包含单词位置信息，即第 0 行，第 1 列，简写为

[0,1]，插入到 the 单词结点对应的位置链表中；当程序扫描到第 6 行时，“the”再次被提取，此时通过 searchWordList 函数判断 the 单词结点已经在单词表中，只要把它再次出现的位置[6,0]插入到 the 结点对应的位置链表的尾部即可。单词表的结构信息示意图如下图 1 所示。

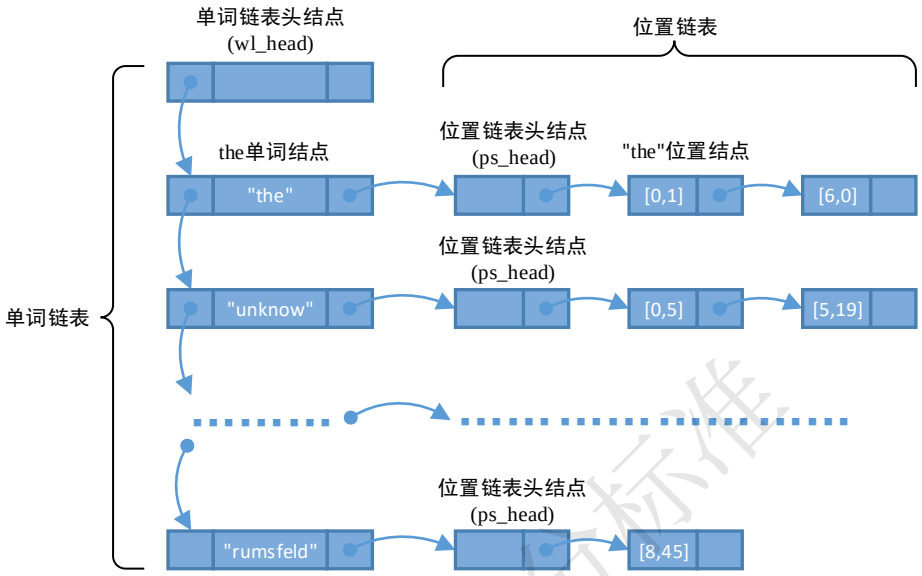

图 1 单词链表和位置链表示意图

2） 在第 1）步生成的单词表基础上，输出表中所有单词信息，包括单词种类数量，每个单词字符串和它们出现的位置信息。

3） 在第 1）步生成的单词表基础上，搜索出现最多次数的单词，并输出该单词字符串和出现的位置信息。

4） 在第 1）步生成的单词表基础上，在程序结束之前，应用循环和 delete 清理单词表，包括删除单词链表和相应的位置链表。

注意：本题目录下有 3 个文件，7.cpp, 7_demo.exe 和 TestData.txt。7.cpp 是源程序，TestData.txt 是要扫描的文本文档，务必把它和 7.cpp 放在同一目录下，而 7_demo.exe 是可以执行的演示程序，可以鼠标双击它看运行结果。

题目要求补充实现 7.cpp 的 3 个函数 insertWordPosition, searchMaxOccurenceWord 和 removeWordList 中缺少的代码。程序 7.cpp 的代码如下：

```
#include <iostream>
#include <cstring>
#include <fstream>
using namespace std;

struct Position{       // 单词位置结构体
    int lineNo;        // 单词在文件中所在行的行号
    int columnNo;      // 单词第一个字母所在列的列号
    Position *next; // 指向下个位置结点的指针
```

```
};

struct Word{            // 单词结点的结构体
    char *str;        // 单词字符串指针
    int occurence; // 单词在文件中出现的次数
    Position *ps_head,*ps_rear; // 单词位置链表的头指针和尾指针
    Word *next;       // 指向下个单词结点的指针
};

struct WordList{          // 单词表的结构体
    int typeCounter;    // 单词种类数量
    Word *wl_head,*wl_rear; // 单词链表的头指针和尾指针
};

bool createWordList(WordList &wl,const char fileName[]); // 从文件读取字符串，创建单词表
void processLineString(WordList &wl,const char lineString[]); // 提取一行字符串中的单词，插入单词表
// 处理提取的单词：提取的单词不在单词表中，生成相应的单词结点，并插入到单词
// 链表的尾部。否则，只要将提取单词的位置信息插入到表中单词的位置链表的尾部。
void processWord(WordList &wl,char *str,int lineNo,int columnNo);
void printWordList(const WordList &wl);    // 输出单词链表中每个单词的信息
void printWordInfo(const Word *wd); // 输出单词信息，包括单词字符串、在文件中出现次数和出现位置
Word *searchWordList(WordList &wl,const char str[]); // 在单词表中搜索与字符串 str 匹配的单词
void searchMaxOccurenceWord(const WordList &wl); // 搜索并输出单词表中出现次数最大的单词
void insertWordList(WordList &wl,Word *wd); // 将单词结点插入到单词链表的尾部
void insertWordPosition(Word *wd, int lineNo, int columnNo); // 将位置信息插入到单词位置链表的尾部
void removeWordList(WordList &wl);          // 释放单词表和其中的单词位置链表
bool isPunctuation(char s); // 检查字符 s 是标点符号或是分隔符吗？

int main()
{
    WordList wordlist ={0,NULL,NULL};
    wordlist.wl_head = wordlist.wl_rear = new Word;
    wordlist.wl_head->ps_head = NULL;
    wordlist.wl_head->next = NULL;

    const char *fileName = "TestData.txt";
    createWordList(wordlist,fileName);
    printWordList(wordlist);
    searchMaxOccurenceWord(wordlist);
    removeWordList(wordlist);
    getchar();
    return 0;
}
```

```
bool createWordList(WordList &wl,const char fileName[])
{    // 从文件逐行读取字符串，创建单词表
     char lineString[256];
     ifstream textFile;

     textFile.open(fileName); // 打开文件
     if (textFile.fail()){       // 判断文件打开是否失败？
          cout<<"文件打开失败！检查当前目录下是否有 TestData.txt 文件！";
          return false;
     }
     else{ // 打开文件成功，开始循环: 每次从文件中读取一行到字符串 lineString 中
          while(textFile.getline(lineString,256)) {
               cout<<lineString<<endl;
               processLineString(wl,lineString);
          }
          textFile.close();   // 处理完成后，关闭文件
          cout<<endl<<endl<<"------------文件处理完成!-------------"<<endl;
          return true;
     }
}

void processLineString(WordList &wl,const char lineString[])
{    // 提取一行字符串 lineString 中的单词，插入单词链表或位置链表中
     char str[256];     // 用来保存提取到单词的字符串信息
     static int lineCounter = 0;   // 记录当前处理的行号
     int i = 0, j = 0;

     while(lineString[i]){    // 扫描一行字符串直到'\0'为止
          while(isPunctuation(lineString[i])) i++;   // 跳过标点符号和分割符
          while(!isPunctuation(lineString[i])&&lineString[i])
               str[j++] = lineString[i++];
          if (j>0){       // 提取到一个单词，使用 str 保存它的字符串
               str[j] = '\0';
               j = 0;
               // 处理一个提取的单词
               processWord(wl,strlwr(str),lineCounter,i-strlen(str));
          }
     }
     lineCounter++;
}

// 处理提取的单词：提取的单词不在单词表中，生成相应的单词结点，并插入到单词
// 链表的尾部。否则，只要将提取单词的位置信息插入到表中单词的位置链表的尾部。
void processWord(WordList &wl,char *str,int lineNo,int columnNo)
```

```
{
    Word *wd ;

    if(wd = searchWordList(wl,str)){          // 如果单词表已经有该单词
        wd->occurence++;                      // 单词的出现次数增加 1 次
        insertWordPosition(wd,lineNo,columnNo); // 插入单词的位置结点
    }
    else{   // 提取的单词不在单词表中，生成单词结点并插入到单词链表
        wd = new Word;
        wd->str = new char[strlen(str)+1];
        strcpy(wd->str,str);
        wd->occurence = 1;
        wd->next = NULL;
        wd->ps_head = wd->ps_rear = new Position;
        insertWordPosition(wd,lineNo,columnNo);
        insertWordList(wl,wd);
    }
}

void printWordInfo(const Word *wd)
{
    if (wd){ // 输出单词信息，包括该单词字符串、在文件中出现次数和出现位置
        Position *ps = wd->ps_head->next;
        cout<<"单词  "<<wd->str<<"  出现次数："<<wd->occurence<<endl;
        cout<<"        出现位置: ";
        while(ps){
            cout<<"["<<ps->lineNo<<","<<ps->columnNo<<"]    ";
            ps = ps->next;
        }
        cout<<endl;
    }
}

void printWordList(const WordList &wl)
{
    Word *wd = wl.wl_head->next;

    cout<<endl<<"----------单词表信息----------"<<endl;
    cout<<"单词种类："<<wl.typeCounter<<endl<<endl;
    while(wd){
        printWordInfo(wd);
        wd = wd->next;
    }
}
```

```
Word *searchWordList(WordList &wl,const char str[])
{
    Word *wd = wl.wl_head->next;
    // 在单词表中搜索与字符串 str 匹配的单词
    while(wd){
        if(!strcmp(wd->str,str))
            break;
        wd = wd->next;
    }
    return wd;
}

void searchMaxOccurenceWord(const WordList &wl)
{
    Word *wd = wl.wl_head->next;
    Word **maxArray = new Word*[wl.typeCounter*sizeof(Word *)];
    int counter = 0;
    maxArray[counter] = wd;

    cout<<endl<<"--------出现次数最大的单词信息--------"<<endl;
    // 请补充完成以下缺少的代码   6 分
    while(wd){                                          //   ---- 1 分
        if (wd->occurence > maxArray[counter]->occurence)   //   ---- 2 分
            maxArray[counter = 0] = wd;
        else
            if((wd != maxArray[counter])&&(wd->occurence == maxArray[counter]->occurence)) // ----3 分
                maxArray[++counter] = wd;
        wd = wd->next;
    }
    while (maxArray[counter]&&(counter>-1))
        printWordInfo(maxArray[counter--]);
    delete maxArray;
}

bool isPunctuation(char s) //  检查字符 s 是标点符号或是分隔符吗？
{
    char punctuationSet[] = {' ','\t',';',',',',','!',':','\?','\"','\'','-','\0'};
    int i=0;

    while(punctuationSet[i])
        if (s == punctuationSet[i++])
            return true;
    return false;
```

```
}

void insertWordPosition(Word *wd, int lineNo, int columnNo)
{    // 生成位置结点，并插入到单词位置链表的尾部
     Position *ps = new Position;
     // 请补充完成以下缺少的代码   4 分
     ps->lineNo = lineNo;                // --------- 2 分
     ps->columnNo = columnNo;
     ps->next = NULL;
     wd->ps_rear->next = ps;             // --------- 2 分
     wd->ps_rear = ps;
}

void insertWordList(WordList &wl,Word *wd)
{    // 将单词结点插入到单词链表的尾部
     wl.typeCounter++;
     wl.wl_rear->next = wd;
     wl.wl_rear = wd;
}

void removeWordList(WordList &wl)
{
     Word *ptr,*wd = wl.wl_head;
     // 请补充完成以下缺少的代码   6 分
     // 删除位置链表和单词链表，注意删除的顺序
     while(wd){                                  //   ------- 2 分
          Position *temp,*ps = wd->ps_head;
          while(ps){                             //   ------- 2 分
               temp = ps;
               ps = ps->next;
               delete temp;
          }
          ptr = wd;                              //   ------- 2 分
          wd = wd->next;
          delete ptr;
     }
     cout<<endl<<"----------  单词表清理完成!----------"<<endl;
}
```