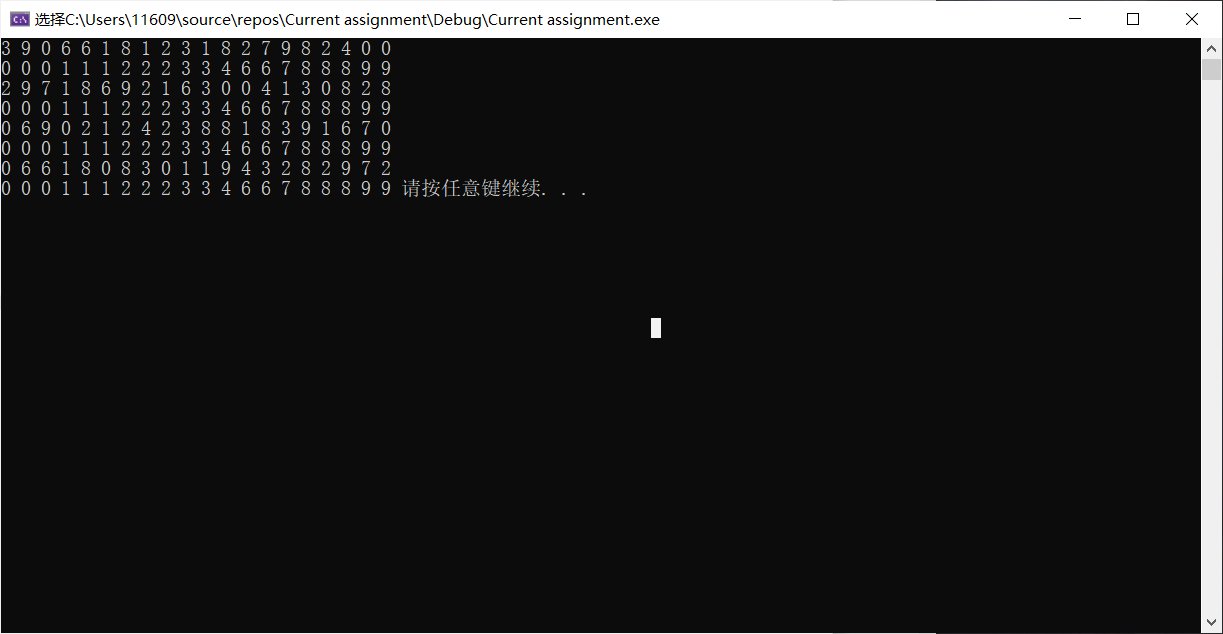
**A.**



//奇数行为排序前，偶数行为排序后，排序前数据一样，顺序不同

#include<iostream>

#include<cstdio>

#include <cstdlib>

#include<algorithm>

#include<ctime>

using namespace std;

void print(int a[], int size)

{

for (int i = 0; i < size; i++)

cout << a[i] << " ";

}

void order1(int a[], int size)

{

int min = 0;

for (int i = 0; i < size; i++)

{

min = i;

for (int j = i + 1; j < size; j++)

if (a[j] < a[min]) min = j;

swap(a[i], a[min]);

}

}

void order1plus(int a[], int size)

{

int max = (size - 1);

int t = max;

for (int i = (size - 1); i >= 0; i--)

{

if (a[max] < a[i]) max = i;

}

swap(a[max], a[t]);

size--;

if (size != 1)

order1plus(a, size);

}

void order2(int b[], int size)

{

for (int i = 1; i < size; i++)

{

for (int j = 0; j < i; j++)

{

if (b[i] < b[j]) swap(b[i], b[j]);

}

}

}

void order2plus(int b[], int size)

{

const static int t = size - 1; //保持t的值不会因为递归而变，而size可变。

for (int i = t; i > (size - 2); i--)

{

if (b[(size - 2)] > b[i]) swap(b[i], b[(size - 2)]);

}

size--;

if(size != 1)

order2plus(b, size);

}

int main()

{

srand(time(0));

int a[20];

for(int i = 0;i < 20;i++)

{

a[i] = rand() % 10;

cout << a[i] << " ";

}

order2plus(a, 20); //以下均为测试代码，先打乱顺序，后排序（第一次数据上方获取）

cout << endl;

print(a, 20);

random\_shuffle(a, a + 20);

cout << endl;

print(a, 20);

order1(a, 20);

cout << endl;

print(a, 20);

random\_shuffle(a, a + 20);

cout << endl;

print(a, 20);

order1plus(a, 20);

cout << endl;

print(a, 20);

random\_shuffle(a, a + 20);

cout << endl;

print(a, 20);

order2(a, 20);

cout << endl;

print(a, 20);

system("pause");

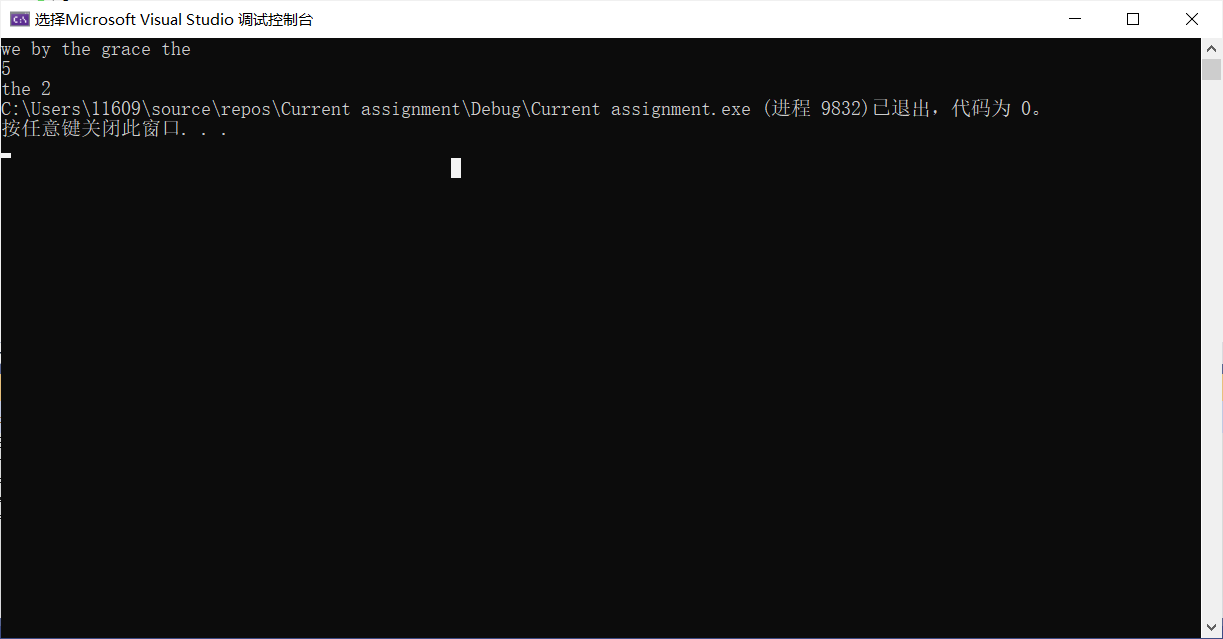
return 0;

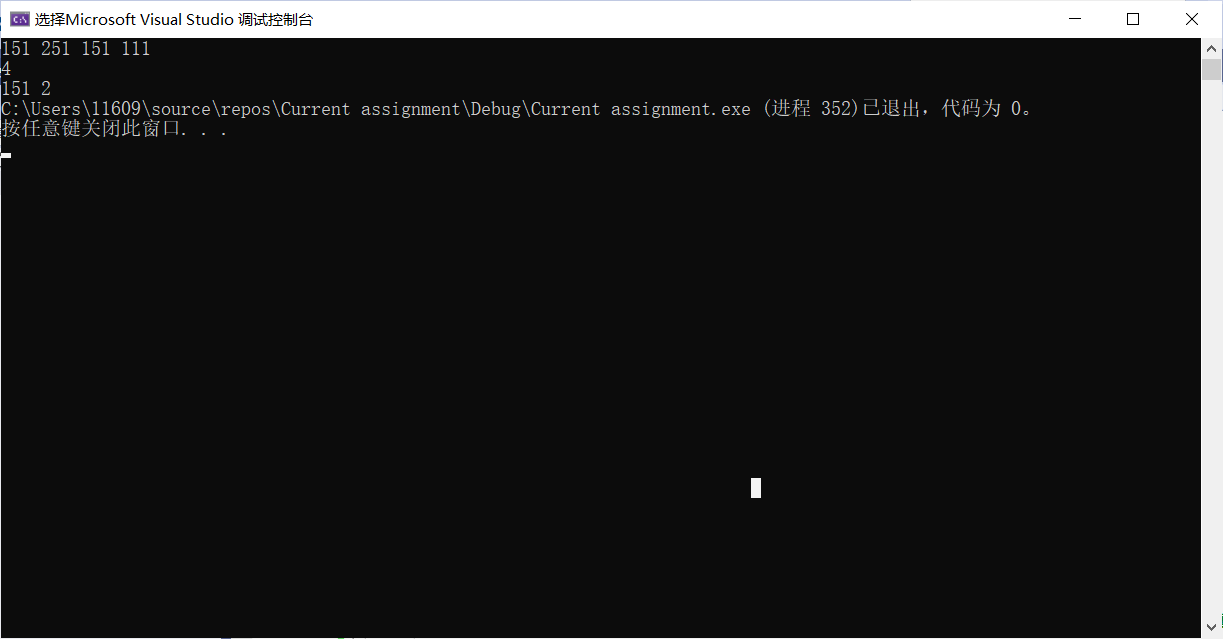
}

问题与思考：如果数据比较大，用clock函数可以明显看出printf与cout速度上的差距。这里用了algorithm中的random\_shuffle函数打乱顺序，可以便于看到同一组数据四种方法排序是否有出现错误的；rand（）函数vs（2010 && 2019）中没有声明#include <cstdlib>也正常使用，检查前还没注意到，问题有点大；algorithm库也有现成的sort函数用于排序，效率高，用less<数据类型>()

greater<数据类型>()确定升降序即可。

**B.**





#include<iostream>

using namespace std;

char b[20];

int d[20];

class Word

{

public: //下方需要访问c[20]

char c[20]; //单个单词长度上限20

Word() { for (int i = 0; i < 20; i++) c[i] = 0; }

Word(char b[])

{

for (int i = 0; i < 20; i++)

c[i] = 0;

for (int i = 0; i < 20; i++)

c[i] = b[i];

}

};

int length(char a[],int size)

{ //判断的是除开\0的有效长度，空格也算

int counter2 = 0;

for (int i = 0; i <= size - 1; i++)

{

if (a[i] != 0) counter2++;

}

return counter2;

}

int length2(char a[], int size)

{

int counter4 = 0;

for (int i = 0; i <= size - 1; i++) //原本对char数组理解错误误以为char[20]的19索引会自动填\0,只能输入19个元素， 实际上若输入了索引19，\0应该在20

{

if (a[i] != 0 && a[i] !=' ') counter4++;

}

return counter4;

}

int total(char a[], int size)

{

int counter = 0;

int set = length(a, size);

for (int i = 0; i < set; i++)

{

if (a[i] != ' ' && a[i + 1] == ' ')

counter++;

}

if (a[set - 1] != ' ') counter++;

return counter;

}

void mostfrequent(char a[], int size)

{

int k = 0,k2 = 0;

Word record[20];// 最多记录20个单词

for (int i = 0; i < size; i++)

{

if (a[i] != ' ')

{

b[k] = a[i];

k++;

}

if (a[i + 1] == ' ' || a[i + 1] == 0)

{

record[k2] = Word(b);

k2++; //k2 = 0故最后一次k2++ 后为总单词数目

for (int j = 0; j < 20; j++)

b[j] = 0,k = 0;

}

} //通过临时数组b转运单个单词至Word数组中每个record，分离得到所有单词

for (int i = 0; i < k2; i++)

{

int r1 = 0,r2 = 0,set2 = 0,counter3 = 0;

for (int j = i + 1; j < k2; j++)

{

counter3 = 0, r1 = i, r2 = j, set2 = -1;//习惯将变量都变为0但实际上set2设为0容易出问题

if (length2(record[i].c, 20) == length2(record[j].c, 20))

set2 = length2(record[i].c, 20);

/\*cout << set2 << " ";\*/ //检验点：通过是否显示长度（>0）判断是否正常判断长度相同,单词是否正常“封装”比较

for (int l = 0; l < set2; l++)

{

if (record[r1].c[l] == record[r2].c[l]) //r1,r2对应第几个单词，l限制在单词长度内

counter3++;

/\*cout << counter3 ;\*/ //与上方对照检验点

}

if (counter3 == set2) //(\*)

d[i] += 1; //与某单词相同次数

} //无相同属于非合法输入

/\*cout << i;\*/

}

int max = 0;

for (int i = 0; i < k2; i++)

{

if (d[i] > d[max]) //在这里手误加过一个分号，上方根据d[i]的错误找出：i在正确值之后变成了19

max = i; //这里实际上就是 输出的 第一个 为最大出现次数 的单词的 次数

}

/\*cout << max;\*/

for (int i = 0; i < length2(record[max].c, 20); i++)

cout << record[max].c[i];

cout << " " << d[max] + 1;

}

int main()

{

char a[20]; //上面设的都是20这里不能动

gets\_s(a);

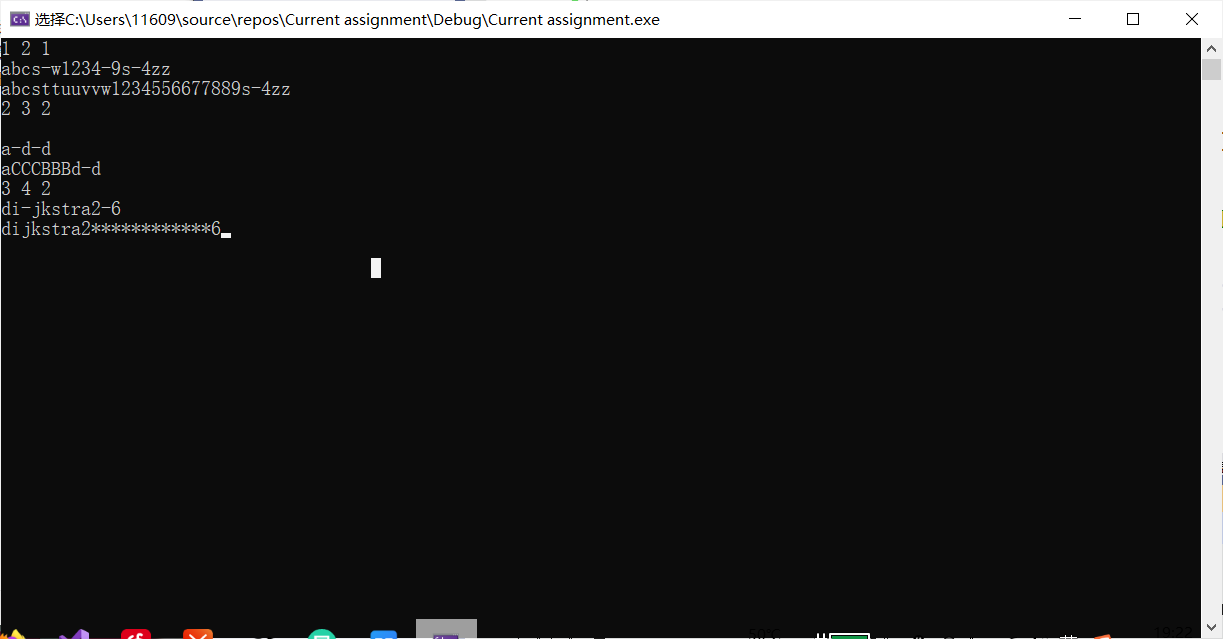
cout << total(a, 20)<<endl;

mostfrequent(a,20);

}

问题与思考：数组以0为开始挺容易出现各种问题的,和其他代码部分可能产生冲突不匹配，可以考虑从1开始方便一些；char数组补\0的特性确实使得数组的实际输入长度（不为\0部分）成了必求的一部分；变量的初始有时设为0并不是最佳选择，上方的count3（相同长度单词按顺序相同元素个数）在初始时为0，set2（单词若长度相同，即为其长度）初始若也为0，下方判断就会出错；初次用类去构造数组使用，才确切感觉到补一个空白构造的意义，空白构造可以用于构造类数组，其中的元素靠构造函数得到数据。

**C.**



#include<iostream>

using namespace std;

char str[100];

char add[100];

char pattern;

bool isnumber(char x)

{

return (x >= '0' && x <= '9');

}

bool alphabet(char x)

{

return(x >= 'a' && x <= 'z');

}

int length(char a[])

{

int c = 0;

for (int i = 0; i < 100; i++)

if (a[i] != 0) c++;

return c;

}

int main()

{

int p1, p2, p3,count,k = 0;

cin >> p1 >> p2 >> p3 >> str;

for (int i = 0; i < length(str); i++)

{

k = 0;

count = 0;//展开条件

if (i != 0 && i != 99 && str[i] == '-' && str[i + 1] > str[i - 1] && ((alphabet(str[i + 1]) == 1 && alphabet(str[i - 1]) == 1) || (isnumber(str[i + 1]) == 1 && isnumber(str[i - 1]) == 1)))

{

if ((alphabet(str[i + 1]) == 1 && alphabet(str[i - 1]) == 1))

{

if (p1 == 1)

{

count = str[i + 1] - str[i - 1];

for (int m = 1; m < count; m++)

{

pattern = str[i - 1] + m;

for (int j = 0; j < p2; j++)

{

add[k] = pattern;

k++;

}

}

}

else if (p1 == 2)

{

count = str[i + 1] - str[i - 1];

for (int m = 1; m < count; m++)

{

pattern = str[i - 1] + m - 32;

for (int j = 0; j < p2; j++)

{

add[k] = pattern;

k++;

}

}

}

}

else if ((isnumber(str[i + 1]) == 1 && isnumber(str[i - 1]) == 1))

{

if (p1 == 1 || p1 == 2)

{

count = str[i + 1] - str[i - 1];

for (int m = 1; m < count; m++)

{

pattern = str[i - 1] + m;

for (int j = 0; j < p2; j++)

{

add[k] = pattern;

k++;

}

}

}

}

if (p1 == 3)

{

count = str[i + 1] - str[i - 1];

for (int j = 0; j < (count-1) \* p2; j++)

{

add[j] = '\*';

}

}

}

if (str[i] == '-' && (str[i + 1] != str[i - 1] + 1))

{

if (p3 == 1)

{

for (int m = 0; m < (count - 1) \* p2; m++)

{

cout << add[m];

}

}

else if (p3 == 2)

{

for (int m = ((count - 1) \* p2 - 1); m >= 0; m--)

{

cout << add[m];

}

}

}

if (str[i] != '-')

cout << str[i];

if (str[i] == '-' && str[i + 1] <= str[i - 1])

cout << str[i];

else if (str[i] == '-' && ((alphabet(str[i + 1]) != alphabet(str[i - 1]))|| (isnumber(str[i + 1]) != isnumber(str[i - 1]))))

cout << str[i]; //条件判断拆开写了，上面不可else if不然会重复输出

}

system("pause");

return 0;

}

问题与思考：如果不用cin >> str的方式来输入char数组，而是按照for循环来输入，而又不输入空格，实际上会自动填到下一个元素中去，如果数组不够大，是会溢出的；if太长拆开写（多条件）需要考虑else if和if的使用不同，上方最后两个条件就不能两个if 或else if；类似的对数组处理问题通过数个辅助数组应该都能较好解决。