

算法与数据结构设计报告

（ 2021 / 2022学年 第 一 学期）

题 目： Top-N问题

代码分析器

|  |  |
| --- | --- |
| **专 业** | **计算机科学与技术** |
| **学 生 姓 名** | **茹兴奥** |
| **班 级 学 号** | **B19030824** |
| **指 导 教 师** | **柯昌博** |
| **指 导 单 位** | **计算机科学与技术系** |
| **日 期** | **2021.10.25-2021.11.05** |

|  |  |  |
| --- | --- | --- |
| **支撑指标点** | **评价准则** | **计分（每项10分）** |
| 课程目标1：文献调研与资料收集能力，问题发现、研究、分析与解决能力（20分） | 1、能够掌握算法与数据结构设计的相关基础知识，并能够针对求解的工程问题，收集资料进行合理的分析与设计 |  |
| 2、通过调研，能够选择合适的程序设计语言与编程开发平台，对求解的工程问题进行编程实现 |  |
| 课程目标2：通过课程设计，培养学生综合应用算法和数据结构等知识解决工程问题的实践能力（20分） | 3、能够给出数据结构和算法的设计描述，给出关键算法的流程图或伪代码，并给出各算法之间的结构关系描述 |  |
| 4、具备一定的人机交互设计意识，人机交互设计合理、友好，操作简便 |  |
| 课程目标3：培养解决工程问题的开发工具运用能力，能够利用程序设计软件或系统对问题求解进行模拟和实现，能够设计测试数据验证问题解决方法的正确性，并能够对问题解决方法的性能和效率进行分析（40分） | 5、具备一定的算法与数据结构设计分析能力，能够完成课题要求的各项任务和指标 |  |
| 6、能够结合计算机软硬件资源，合理选用算法、数据结构、数据存储方式等技术手段，对求解的工程问题进行有效建模和求解 |  |
| 7、具备一定自学能力与探索创新意识，能够充分利用教科书及其资源（如网络等）自学新知识与新技能 |  |
| 8、掌握调试方法与工具，对程序开发过程中出现的问题进行分析、跟踪与调试，并能够进行充分测试 |  |
| 课程目标4：选择同类课题的学生能够通过讨论和交流解决课程设计中的难题，能在实验报告中准确阐述课程设计的内容，能够清晰陈述观点和回答问题（20分） | 9、能够正确、完整地回答指导教师关于课题的问询，反映其对课题内容，以及相关的工程基础知识具有较好的理解和掌握 |  |
| 10、具备一定的语言表达能力与文字处理能力，能够结合复杂工程问题撰写报告，报告内容和实验数据详实，格式规范 |  |
| 算法与数据结构设计能力测评总分 | |  |
| **指导教师： 年 月 日** | | |
| **备注：** | | |

**Top-N问题**

**一、课题内容和要求**

假设全国有10000人参加物理竞赛，只录取成绩优异的前10名，并将他们从

高分到低分输出。而对落选的其他考生，无需排出名次，问此种情况下，试编写

尽可能快的算法解决Top-10问题？

扩展：如何在10亿数中找出前1000大的数？

**二、数据结构说明**

在该部分我使用到了冒泡排序，位置交换，和堆排序

位置交换：

void swap(int \*a,int \*b)

{

int temp;

temp=\*a;

\*a=\*b;

\*b=temp;

}

冒泡排序：

#include <stdio.h>

void sort(int \*a,int len)

{int i=0;

int j;

int t;

for(i=0;i<len-1;i++)[1]

{

for(j=0;j<len-i-1;j++)

{

if(a[j]>a[j+1])

{

t=a[j];

a[j]=a[j+1];

a[j+1]=t;

}

}

}

}

int main(int argc, char \*argv[])

{

int a[10]={

-999,2,3,77,12,88,0,-8,99,100

};

int i=0;

sort(a,10);

for(i=0;i<10;i++)

{

printf("%d ",a[i]);

}

return 0;

}

堆排序：

#**include** <stdio.h>

#**include** <stdlib.h>

**void** swap(**int**\* a, **int**\* b)

{

**int** temp = \*b;

    \*b = \*a;

    \*a = temp;

}

**void** max\_heapify(**int** arr[], **int** start, **int** end)

{

    //建立父节点指标和子节点指标

**int** dad = start;

**int** son = dad \* 2 + 1;

**while** (son <= end)  //若子节点指标在范围内才做比较

        {

**if** (son + 1 <= end && arr[son] < arr[son + 1])

            //先比较两个子节点大小，选择最大的

            son++;

**if** (arr[dad] > arr[son]) //如果父节点大於子节点代表调整完毕，直接跳出函数

**return**;

**else**  //否则交换父子内容再继续子节点和孙节点比较

        {

            swap(&arr[dad], &arr[son]);

            dad = son;

            son = dad \* 2 + 1;

        }

    }

}

**void** heap\_sort(**int** arr[], **int** len)

{

**int** i;

    //初始化，i从最後一个父节点开始调整

**for** (i = len / 2 - 1; i >= 0; i--)

        max\_heapify(arr, i, len - 1);

    //先将第一个元素和已排好元素前一位做交换，再重新调整，直到排序完毕

**for** (i = len - 1; i > 0; i--)

    {

        swap(&arr[0], &arr[i]);

        max\_heapify(arr, 0, i - 1);

    }

}

**int** main() {

**int** arr[] = { 3, 5, 3, 0, 8, 6, 1, 5, 8, 6, 2, 4, 9, 4, 7, 0, 1, 8, 9, 7, 3, 1, 2, 5, 9, 7, 4, 0, 2, 6 };

**int** len = (**int**) sizeof(arr) / sizeof(\*arr);

    heap\_sort(arr, len);

**int** i;

**for** (i = 0; i < len; i++)

        printf("%d ", arr[i]);

    printf("\n");

**return** 0;

}

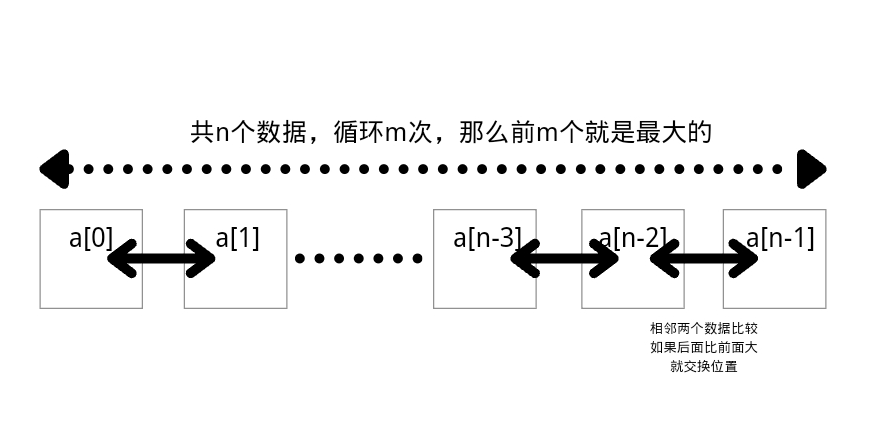
**三、算法设计**

1.对于第一小问，运用冒泡排序和堆排序都是很简单的，10000个取前十，冒泡排序就是从后往前两两比较，如果后面的比前面的大就交换位置，这样经过一轮循环之后，最前面的就是最大的，以此类推，循环十次，前十个就是最大的十个，而且这十个还有顺序：

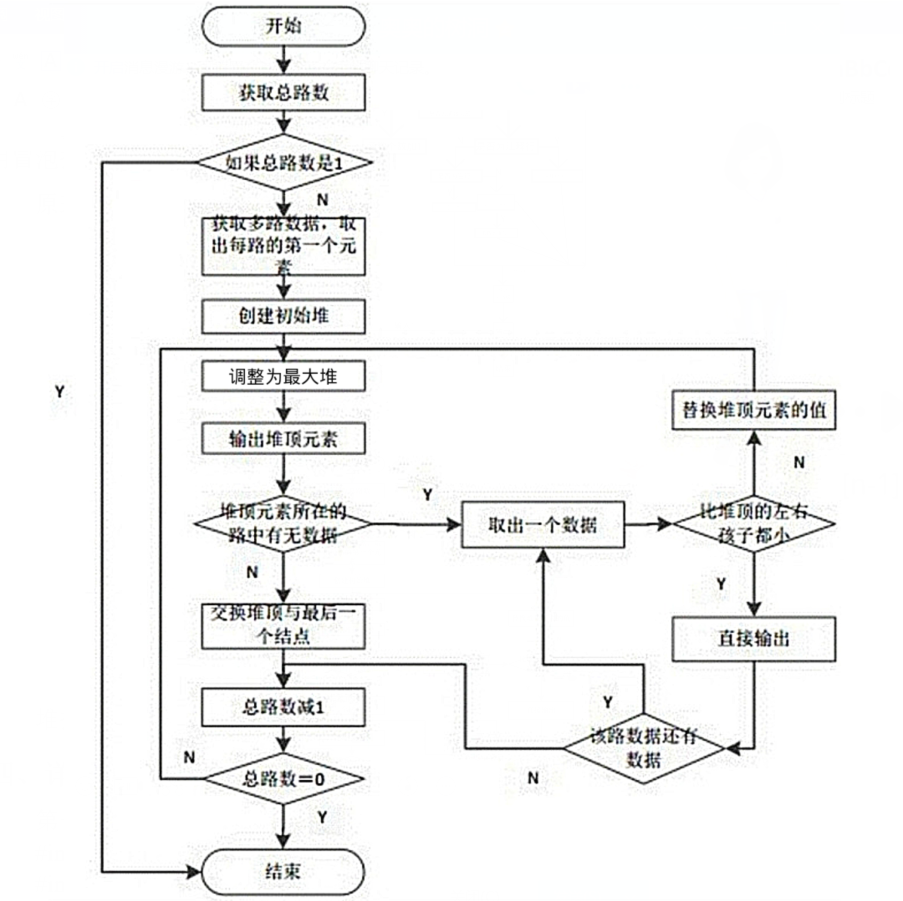
2.如果是堆排序，只要从后往前保证父母大于孩子就行，如果不大于就交换父母与孩子的位置，这样一轮循环下来，就能保证最上面的是最大的，把顶部的拿出来，再次对这个树进行维护，再次使最顶部的是最大的，这样循环十次，就能找到最大的十个了。

3.而对于拓展题，因为数据太多，有1000000000个数据，而且还是取前1000，再次使用冒泡排序就会很慢，所有还是使用堆排序简单一些。

原理图：冒泡：



堆排序：



**四、详细设计**

第一题冒泡排序源代码：

#include<iostream>

#include<string>

#include <stdlib.h>

using namespace std;

double a[10000];

int main(){

int i,j,k;

int b,c;

double d;

for(i=0;i<10000;i++){

b=rand()%101;

c=rand()%10;

a[i]=b+c\*0.1;

if(a[i]>100)

a[i]--;

}

for(j=0;j<100;j++){

for(k=9999;k>=0;k--)

{

if(a[k]<=a[k+1])

{

d=a[k];

a[k]=a[k+1];

a[k+1]=d;

}

}

}

for(j=0;j<10;j++){

cout<<a[j]<<endl;

}

return 0;

}

第一问堆排序源代码：

#include<iostream>

#include<string>

#include <stdlib.h>

using namespace std;

unsigned int length=10000;

void swap(int \*m,int \*n)

{

int temp;

temp=\*m;

\*m=\*n;

\*n=temp;}

void fix\_tree(float \*tree, unsigned int start, unsigned int end){

int dad = start;

int son = dad \* 2 + 1;

while (son <= end) {

if (son + 1 <= end && tree[son] < tree[son + 1])

son++;

if (tree[dad] > tree[son])

return;

else {

swap(tree[dad], tree[son]);

dad = son;

son = dad \* 2 + 1;

}

}

}

void build\_tree(float \*tree, unsigned int length){

for (int i = length / 2 - 1; i >= 0; i--)

fix\_tree(tree, i, length - 1);

}

int main(){

int b,c,i,d=length;

float a[length];

for(i=0;i<length;i++){

b=rand()%101;

c=rand()%10;

a[i]=b+c\*0.1;

if(a[i]>100)

a[i]--;}

build\_tree(a,length-1);

swap(a[0],a[length]);

cout<<a[length]<<endl;

for(i=0;i<9;i++){

fix\_tree(a,0,d-2);

swap(a[0],a[d-1]);

cout<<a[d-1]<<endl;

d--;

}

return 0;

}

拓展题堆排序源代码：

#include<iostream>

#include<string>

#include <stdlib.h>

using namespace std;

unsigned int length=100000;

void swap(int \*m,int \*n)

{

int temp;

temp=\*m;

\*m=\*n;

\*n=temp;}

void fix\_tree(float \*tree, unsigned int start, unsigned int end){

int dad = start;

int son = dad \* 2 + 1;

while (son <= end) {

if (son + 1 <= end && tree[son] < tree[son + 1])

son++;

if (tree[dad] > tree[son])

return;

else {

swap(tree[dad], tree[son]);

dad = son;

son = dad \* 2 + 1;

}

}

}

void build\_tree(float \*tree, unsigned int length){

for (int i = length / 2 - 1; i >= 0; i--)

fix\_tree(tree, i, length - 1);

}

int main(){

int b,c,i,d=length;

float a[length];

for(i=0;i<length;i++){

b=rand()%101;

c=rand()%10;

a[i]=b+c\*0.1;

if(a[i]>100)

a[i]--;}

build\_tree(a,length-1);

swap(a[0],a[length]);

cout<<a[length]<<endl;

for(i=0;i<999;i++){

fix\_tree(a,0,d-2);

swap(a[0],a[d-1]);

cout<<a[d-1]<<endl;

d--;

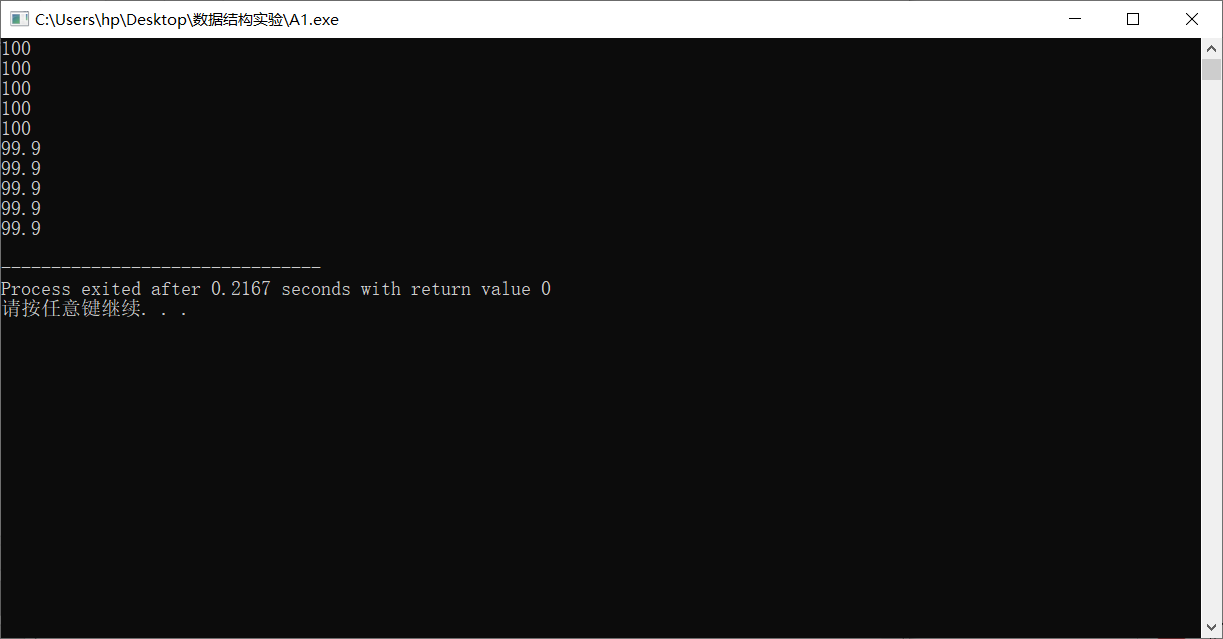
}

return 0;

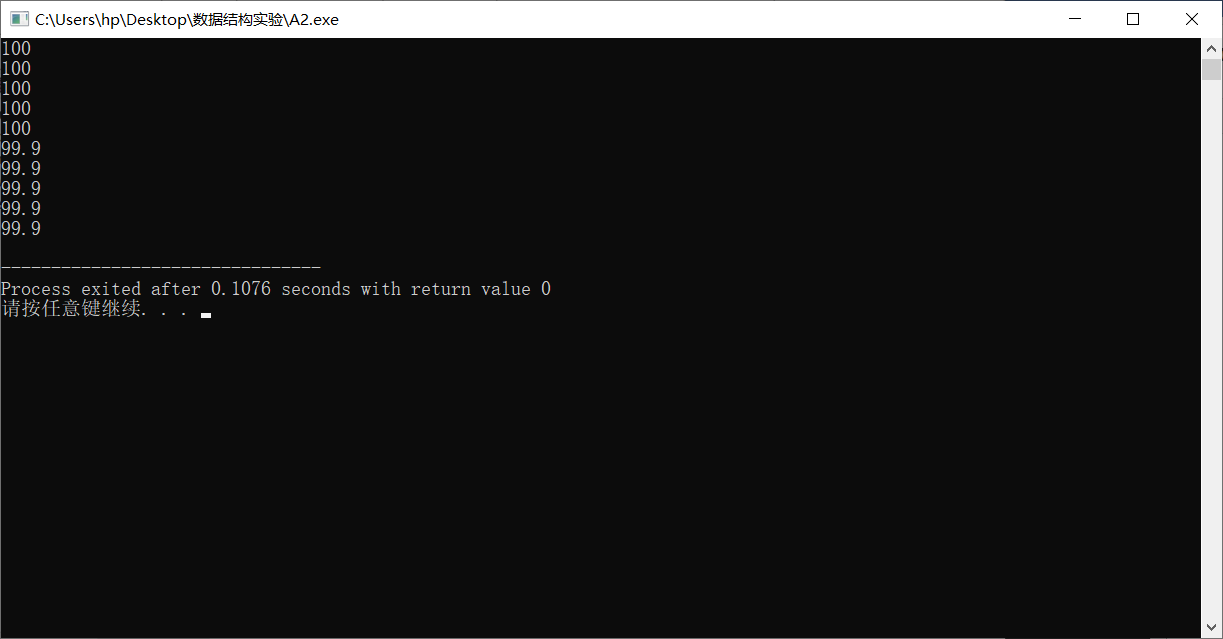
}

**五、测试数据及其结果分析**

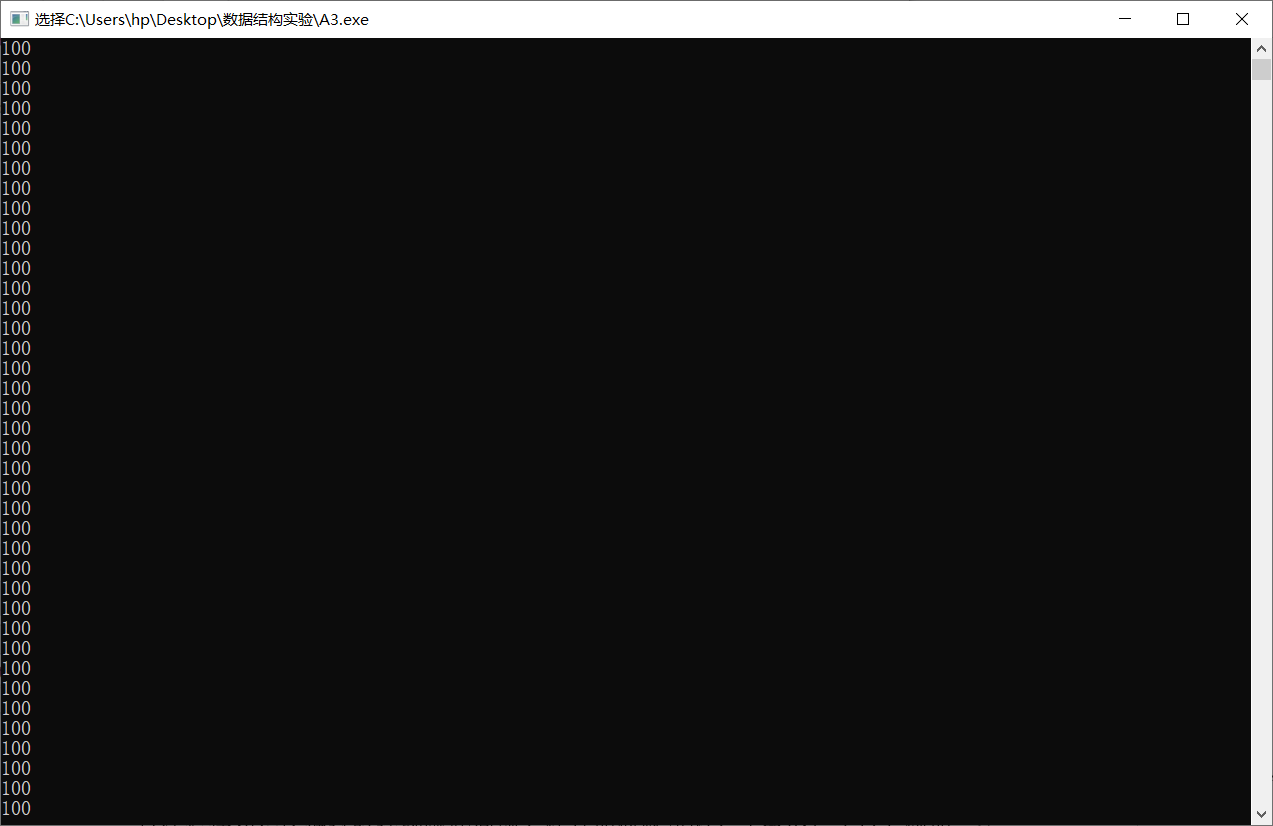
第一问冒泡排序:



第一问堆排序：



拓展题堆排序：



**六、算法设计和程序调试过程中的问题**

**问题1**：冒泡排序比较顺序和循环老是搞错

**解决方法**：打开课本，看书，找到原理把他看懂

**问题2**：堆排序我以为可以直接截取前多少层，但发现，只能保证父母比孩子大，别的不能。

**解决方法**：多看书，多了解代码原理。

**问题3**：不知道怎么找一万个数据。

**解决方法：**利用随机算法，创建一万个数据的数组。

**七、课程设计总结**

算法进一步改进的设想：应该把数据从文档中读取出来，也可以把数据存入文档，方便查看，也更符合实际。

课程设计过程的收获和感受：通过这次设计实验，让我对堆排序和冒泡排序和堆排序有了更深入的了解，也让我明白了，这两种算法的优缺点，学到了很多东西。

**代码分析器**

**一、课题内容和要求**

（一）课题内容

读入 C 源文件，统计文件中的代码、注释和空行的行数以及函数的个数和平

均行数，并利用统计信息分析评价该程序的风格。

（二）课题要求

(1) 将程序文件按字符顺序读入源程序；

(2) 边读入程序，边识别统计代码行、注释行和空行，同时还要识别函数的

开始和结束，以便统计其个数和平均行数。

(3) 程序的风格评价分为代码（函数平均长度）、注释（占总行数比例）和空

行（占总函数比例）三个方面。每个方面分为 A，B，C:和 D 四个等级。

等级的划分标准如下：



(4) 测试时先对较小程序进行分析。当你的程序能正确运行时，对你的程序

本身进行分析。

【其他要求】

(1) 变量、函数命名符合规范。

(2) 注释详细：每个变量都要求有注释说明用途；函数有注释说明功能，对

参数、返回值也要以注释的形式说明用途；关键的语句段要求有注释解

释。

(3) 程序的层次清晰，可读性强。

**二、数据结构说明**

在这一题中使用到了打开读取文件：

#define \_CRT\_SECURE\_NO\_WARNINGS // 新版本提示要用安全函数，此处添加宏\_CRT\_SECURE\_NO\_WARNINGS 屏蔽告警

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define MAX\_LINE 1024

int main()

{

char buf[MAX\_LINE]; /\*缓冲区\*/

FILE\* fp; /\*文件指针\*/

int len; /\*行字符个数\*/

if ((fp = fopen("test.txt", "r")) == NULL)

{

perror("fail to read");

exit(1);

}

while (fgets(buf, MAX\_LINE, fp) != NULL)

{

len = strlen(buf);

buf[len - 1] = '\0'; /\*去掉换行符\*/

printf("%s %d \n", buf, len - 1);

}

return 0;

}

还有正则函数的使用：

#include <stdio.h>

#include <sys/types.h>

#include <regex.h>

/\* 取子串的函数 \*/

static char\* substr(const char\*str,

unsigned start, unsigned end)

{

unsigned n = end - start;

static char stbuf[256];

strncpy(stbuf, str + start, n);

stbuf[n] = 0;

return stbuf;

}

/\* 主程序 \*/

int main(int argc, char\*\* argv)

{

char \* pattern;

int x, z, lno = 0, cflags = 0;

char ebuf[128], lbuf[256];

regex\_t reg;

regmatch\_t pm[10];

const size\_t nmatch = 10;

/\* 编译正则表达式\*/

pattern = argv[1];

z = regcomp(?, pattern, cflags);

if (z != 0){

regerror(z, ?, ebuf, sizeof(ebuf));

fprintf(stderr, "%s: pattern '%s' \n",ebuf, pattern);

return 1;

}

/\* 逐行处理输入的数据 \*/

while(fgets(lbuf, sizeof(lbuf), stdin))

{

++lno;

if ((z = strlen(lbuf)) > 0 && lbuf[z-1] == '\n')

lbuf[z - 1] = 0;

/\* 对每一行应用正则表达式进行匹配 \*/

z = regexec(?, lbuf, nmatch, pm, 0);

if (z == REG\_NOMATCH) continue;

else if (z != 0) {

regerror(z, ?, ebuf, sizeof(ebuf));

fprintf(stderr, "%s: regcom('%s')\n", ebuf, lbuf);

return 2;

}

/\* 输出处理结果 \*/

for (x = 0; x < nmatch && pm[x].rm\_so != -1; ++ x)

{

if (!x) printf("%04d: %s\n", lno, lbuf);

printf(" $%d='%s'\n", x, substr(lbuf, pm[x].rm\_so, pm[x].rm\_eo));

}

}

/\* 释放正则表达式 \*/

regfree(?);

return 0;

}

执行下面的命令可以编译并执行该程序：

# gcc regexp.c -o regexp

# ./regexp 'regex[a-z]\*' < regexp.c

0003: #include <regex.h>

$0='regex'

0027: regex\_t reg;

$0='regex'

0054: z = regexec(?, lbuf, nmatch, pm, 0);

$0='regexec'

**三、算法设计**

首先要打开读取文件，使用者输入文件地址，打开他，如果无法打开，就会循环，直到打开正确的文件；讲每一行复制到一个字符串；然后使用到了正则函数regex()，regex函数就是看是否这个字符串和我们提前设定的格式一样。比如：

注释行：如果检索到“//”或者/\*就会判断是注释行，而且对于/\*，只要遇到就开始注释加一，直到遇到\*/;

空行：如果这个字符串是空的或者全是空格，那么就是空行；

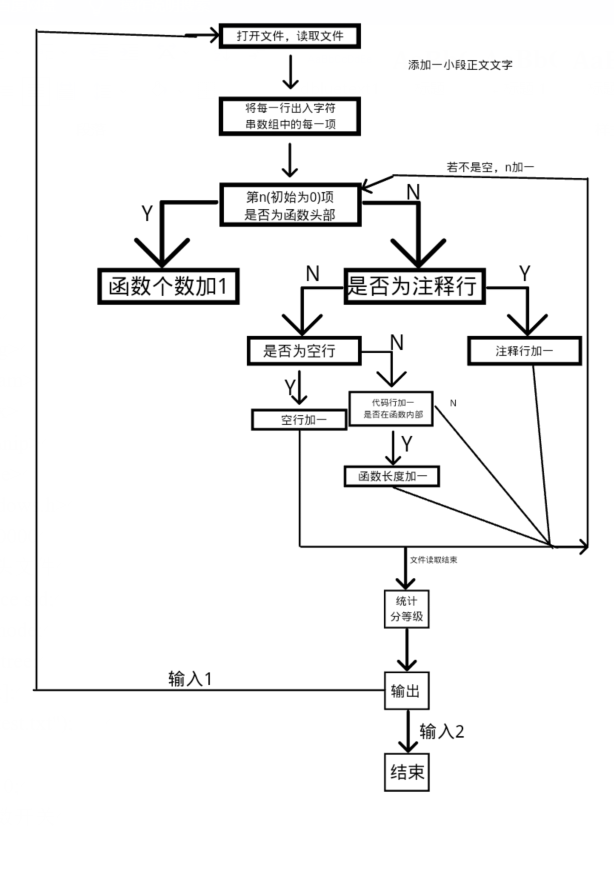
代码行：除了代码行和空行就是代码行了。

因为要统计函数长度和个数，所以对于一个函数的基本格式，应该是英文 空格 英文 左括号 有或者没有的内容 右括号 空格或者没有东西，但是一旦有了；就不行，因为这属于函数声明，不算是函数个数，如果遇到就要排除。

接下来是函数的长度，从匹配到函数开始，因为函数有一个“{”所以对于函数里的代码来说，他前面的“{”个数减去“}”个数一定是大于零的，就利用这一点判断函数的长度，什么时候个数相减等于零，就说明函数结束，就能统计出函数的长度。

最后就是输出文件地址、函数个数、平均长度、注释行行数、代码行行数，以及对应的比例还有等级的判定。

流程图：



**四、详细设计**

源代码：  
#include<iostream>

#include<string>

#include<fstream>

#include<regex>

#include<iomanip>

#include<queue>

#include<Windows.h>

#define max 10000

//正则表达式头文件

using namespace std;

typedef struct node;

typedef node \*tree;

int visited[max];

string name ("test.txt");

//括号匹配

int kuohao = 0;

//函数长度计数开关

int flag = 0;

//函数行数

int fun = 0;

//代码总行数

int line = 0;

struct result

{

int Code;//代码行

int Comments;//注释行

int Blanklines;//空行

int scode ;//代码行数

int scomment;//注释行数

int sspace;

int countfun ;//函数个数

int funlen ;//函数总长度

double avelen ;//函数平均长度

}list;

string evaluate(result l)

//根据获取到的数据，对程序进行评分

{

string value;

if (l.avelen >= 10 && l.avelen <= 15)

value += 'A';

else if ((l.avelen >= 8 && l.avelen < 10) || (l.avelen > 15 && l.avelen <= 20))

value += 'B';

else if ((l.avelen >= 5 && l.avelen < 8) || (l.avelen > 20 && l.avelen <= 24))

value += 'C';

else

value += 'D';

if (list.scomment >= 15 && list.scomment <= 25)

value += 'A';

else if ((list.scomment >= 10 && list.scomment < 15) || (list.scomment > 25 && list.scomment <= 30))

value += 'B';

else if ((list.scomment >= 5 && list.scomment < 10) || (list.scomment > 30 && list.scomment <= 35))

value += 'C';

else

value += 'D';

if (list.sspace >= 15 && list.sspace <= 25)

value += 'A';

else if ((list.sspace >= 10 && list.sspace < 15) || (list.sspace > 25 && list.sspace <= 30))

value += 'B';

else if ((list.sspace >= 5 && list.sspace < 10) || (list.sspace > 30 && list.sspace <= 35))

value += 'C';

else

value += 'D';

return value;

}

/\*根据成绩给出相印的评语\*/

string remark(char s)

{

if (s=='A')

return " ";

else if (s == 'B')

return " ";

else if (s == 'C')

return " ";

else

return " ";

}

void analyze(string s) //判断是哪一类

{

//匹配到函数定义(regex\_match测试正则表达式是否与整个目标字符串相匹配)

if (regex\_match(s, regex("(\\w{1,10}) (\\w{1,100})\\(.{0,100}\\).{0,2}")))

//函数个数+1

{

list.countfun++;

//开启计数器

flag = 1;

}

if (regex\_search(s, regex("\\{")))

{

kuohao++;

}

if (regex\_search(s, regex("\\}")))

{

if (kuohao)

kuohao--;

if(kuohao==0&&flag)

{

fun++;

//加上尾花括号

list.funlen++;

fun = 0;

flag = 0;

}

}

if (regex\_search(s, regex("//")))

//注释行搜索;

list.Comments++;

else if (s == "")

//空行搜索

list.Blanklines++;

else {

if (flag)

{

list.funlen++;

}

list.Code++;

}

if (flag) {

fun++;

}

}

/\*展示面板\*/

void show()

{

list.scode = (int)((double)list.Code / (list.Code + list.Comments + list.Blanklines)\*100+0.5);

list.scomment = (int)((double)list.Comments / (list.Code + list.Comments + list.Blanklines) \* 100 + 0.5);

list.sspace = 100 - list.scode - list.scomment;

list.avelen = (double)list.funlen / list.countfun;

string res = evaluate(list);

cout << "The result of analysing program file \"" + name + "\":" << endl;

cout << " 代码行数 : " << list.Code << endl;

cout << " 注释行数 : " << list.Comments << endl;

cout << " 空行数 : " << list.Blanklines << endl;

cout << " 代码 注释 空行" << endl;

cout << " ==== ======== =====" << endl;

cout << " " <<list.scode<< "% "<<setw(6)<<list.scomment << "% " << list.sspace<<"%"<< endl;

cout << " 包含了 " << list.countfun << " 条函数." << endl;

cout << " 平均长度 " <<list.avelen<< " 行" << endl;

cout << " 代码等级 " << res[0] <<" "<< remark(res[0]) << endl;

cout << " 注释等级 " << res[1] <<" "<< remark(res[1]) << endl;

cout << " 空行等级 " << res[2] <<" "<< remark(res[2]) << endl;

}

int main()

{

int key;

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

cout << " 请输入进行分析的程序文件路径及其后缀（可使用相对路径）" << endl;

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

cout << "分析文件路径：";

getline(cin, name);

ifstream in(name);

//读入文件

system("cls");

if (!in)

{

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

cout << " 文件打开失败 " << endl;

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

Sleep(2000);

system("cls");

main();

}

string str;

while (1)

{

line++;

if (in.eof())

//判断文件读取结束eof有东西就不会返回false

break;

getline(in, str);

analyze(str);

}

in.close();

show();

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

cout << " 1.分析其他程序 2.退出 " << endl;

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

cout << "请输入序号 ： ";

cin >> key;

if (key == 1) {

system("cls"); //清屏

getchar(); //清除缓存

main();

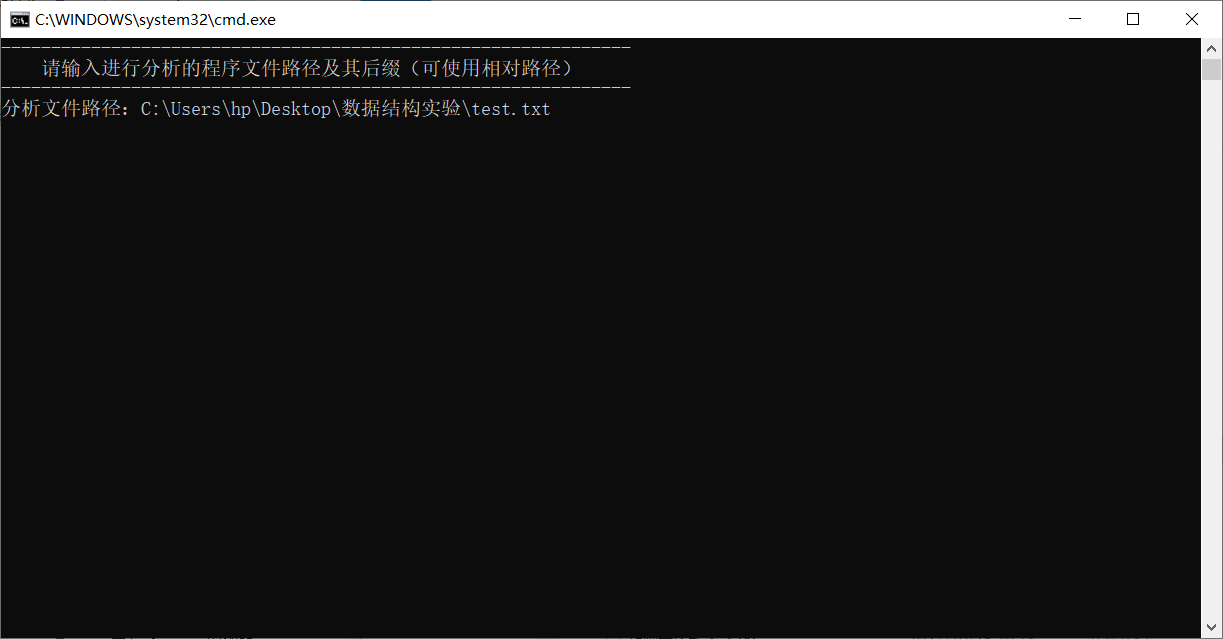
}

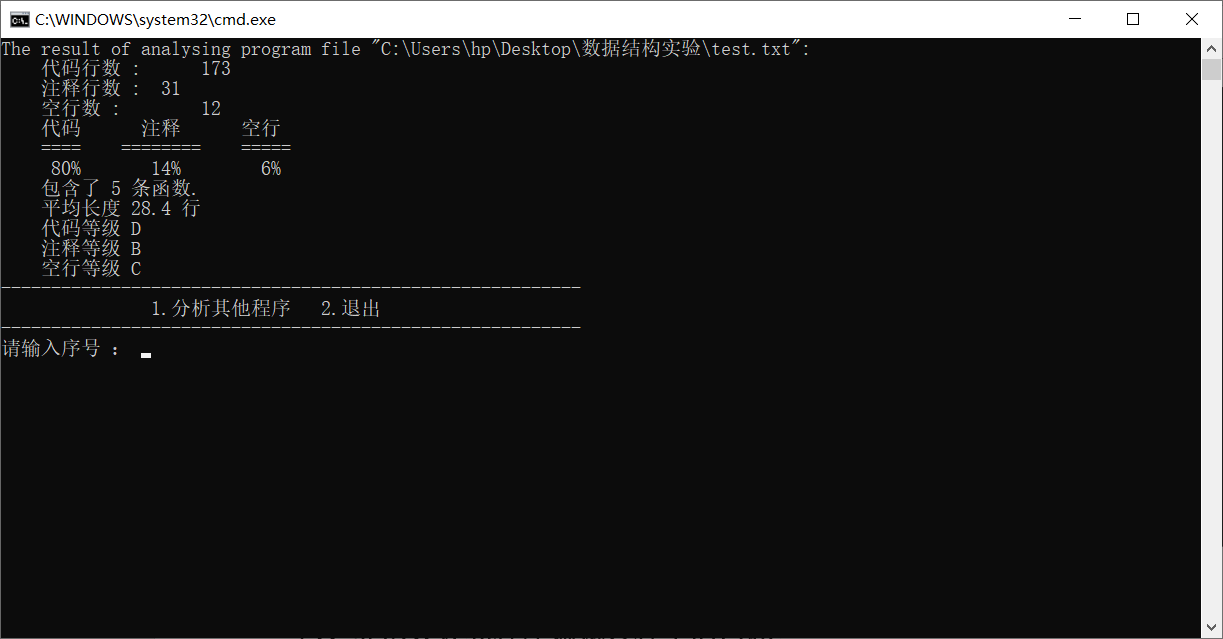
return 0;

}

**五、测试数据及其结果分析**

运行之后的结果：





**六、算法设计和程序调试过程中的问题**

算法设计和程序调试时遇到哪些问题？你是如何解决的？（至少写3点）例如：

**问题1**：不知道怎么去判断一个字符串是不是代码

**解决方法**：在网上找，找到了神奇的regex正则函数匹配，解决了最难的问题

**问题2**：不知道怎么统计函数的长度

**解决方法**：请教同学教会了我可以统计大括号的数量，判断是不是函数内部

**问题3**：用dev一直报错，但是转到MVS就不会报错

**解决方法**：dev版本太低了，无法完美运行regex代码，转到MVS运行即可

**七、课程设计总结**

算法进一步改进的设想：无法克服如果只是类似：

Cout<<”//ababababab”<<endl；

的影响，而且如果大部分代码和注释在同一行，就会判断为注释行，影响最后结果。

Regex也是有一定的局限性。

课程设计过程的收获和感受：通过这次学习，让我学会了熟练的使用regex函数，这是一个全新的函数，让我学到了很多东西，以及有了解决问题的方法，对以后的学习有很大的帮助。