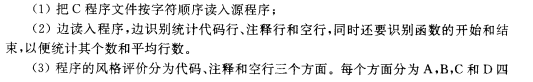
**实习报告：3.5题 程序分析**

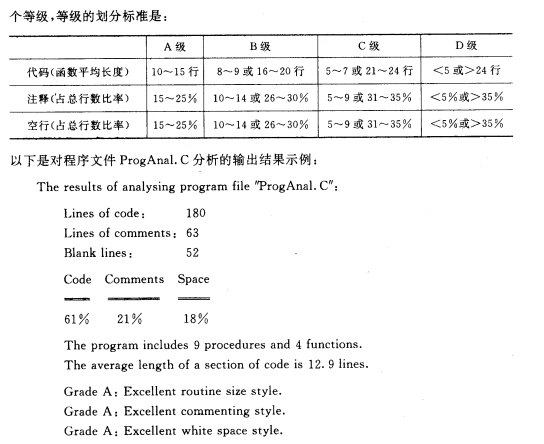
题目：读入一个C程序，统计程序中代码、注释和空行的行数以及函数个数与平均函数，根据统计数据评价代码风格

班级：1613 姓名：张丽玮，资威（孙小丁）

学号：2016K8009929029, 完成日期：2018.7.10

1. **需求分析**





1. **概要设计**

采用顺序表对读入的程序进行存储，先要进行文件的读入和转化。

1. **定义存储结果的结构体**

struct count

{

int total\_line;//总行数

int comments\_line;//注释行数

int blank\_line;//空行数

int code\_line;//代码行数

int function;//一个程序的总函数数

int maxlen;//选做的最大函数深度

int position;//选做的最大函数位置

};

//定义一个结构体

则一个文件使用一个结构体，而如果批量读入多个文件可以以数组方式批量输出，或者读入一个输出一个。

1. **输入接口**
2. **本程序主要包含5个模块（函数）**
3. 主程序模块：其中主函数为

int main()

{

读入文件

将C程序转化为顺序表

对顺序表进行分析

将分析结果与标准核对

输出结果

}

1. 读入转化模块——将C程序转化为顺序表
2. 程序分析模块——统计空行、注释和代码行数，并且统计函数个数和平均行数，找出最长函数的行数与所在位置
3. 标准判断模块——根据程序分析的统计信息进行代码风格判断
4. 输出模块——按照一定格式将结果输出
5. **详细设计**
6. **主程序中需要的全程量**

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#define MAX\_LENGTH 1000000

struct count

{

int total\_line;//总行数

int comments\_line;//注释行数

int blank\_line;//空行数

int code\_line;//代码行数

int function;//一个程序的总函数数

int maxlen;//选做的最大函数深度

int position;//选做的最大函数位置

};

//定义一个结构体

1. **主程序**

int main()

{

char file[MAX\_LENGTH], name[50];

struct count p, result;

printf("输入文件名：");

scanf("%s", name);

printf("\n");

getfile(file);//读入文件，转化为顺序表

p = analyse(file);//将转化后的顺序表进行分析

result = judge(p);//将分析结果进行代码风格判定

print(p ,result, name);//输出结果

return 0;

}

1. **getfile函数——读入文件，转化为顺序表**
2. **analyse函数——将转化后的顺序表进行分析**

struct count analyse(char s[])

{

int i=0;

int blank=1;//用于判断是否是空行

struct count p;

int brace=0,brace2=0,sign=0,sign2=0,function\_line=0,position=0;

p.total\_line=0;

p.code\_line=0;

p.blank\_line=0;

p.comments\_line=0;

p.function=0;

p.maxlen=0;

p.position=0;

//各个变量的初始化

while (s[i]!=EOF)//如果顺序表未结束

{

if (blank==1)//即为一行的开头还未读到字母字符,如果读到则置0

{

while (s[i]==' ')

i++;//排除所有空格的干扰

if (s[i]=='\n')//判断是否换行，如果除了空格就是换行则视为空行

{

p.total\_line++;//总行数

p.blank\_line++;//空格行

i++;//读入下一个字符

blank=1;//换行，仍在一行开头

}else

if (s[i]=='/'&&s[i+1]=='/')//判断是否是注释行

{

p.comments\_line++;

while (s[i]!='\n'&&s[i]!=EOF)

i++;//如果是则忽略注释内容直接读到换行符

p.total\_line++;

if (s[i]!=EOF)

i++;

if (s[i]=='\n')

blank=1;

}

else

blank=0;//如果不是注释行和空行，则进行函数判定。

}else

{

if (s[i]=='{')

{

brace++;

if (sign2==1&&brace2==0)//如果之前已经有一对（）则判定这个{}为函数

{

sign=1;

position=p.total\_line;

}

i++;

}

else if (s[i]=='}')

{

brace--;

if (brace==0&&sign==1)//一个函数体结束，根据{}的配对判定

{

if (p.maxlen<function\_line)

{

p.maxlen=function\_line+1;

p.position=position;

}

function\_line=0;

position=0;

p.function++;

sign=0;

sign2=0;

}

i++;

}

else if (s[i]=='(')

{

brace2++;

sign2=0;

i++;

}

else if (s[i]==')')

{

brace2--;

if (brace2==0&&brace==0&&sign==0&&s[i+1]!=';')

sign2=1;

i++;

}

else if (s[i]=='\n')

{

if (sign==1&&brace!=0)

function\_line++;

p.total\_line++;

blank=1;

i++;

}

else

i++;

}

}

p.code\_line=p.total\_line-p.blank\_line-p.comments\_line;

return p;

}

算法说明——

p——最终的return结构体

blank——用于判断是否处于一行开头，从判断空行与注释行

brace——用于计算{}的配对

brace2——用于计算（）的配对

sign,sign2——用于判断是否是函数

function\_line,position——临时存储当前函数的行数与位置

1、函数体特征为（）中包含调用变量，之后{}中函数体主体

2、读到）且brace2==0的时候，判断是否brace==0，如果是，则（）在结构体和函数体之外，补充判断）后的;，排除函数定义的特殊情况，则sign2=1

3、读到{且sign2==1的时候，表明之前有一个函数体外的（）配对，则接下来{}中的内容是函数体内容，令sign=1

4、用brace增减配对{}，brace==0的时候配对结束，说明结构体结束，计算行数，并且初始化sign=0,sign2=0

1. **judge函数——判断程序的风格**

struct count judge(struct count p)

{

struct count result;

float comments\_rate,blank\_rate,average\_line;

average\_line=(float)p.code\_line/p.function;

comments\_rate=(float)p.comments\_line/p.total\_line;

blank\_rate=(float)p.blank\_line/p.total\_line;

if (average\_line>=10&&average\_line<=15)

result.code\_line=1;

else if ((average\_line>=8&&average\_line<=9)||(average\_line>=16&&average\_line<=20))

result.code\_line=2;

else if ((average\_line>=5&&average\_line<=7)||(average\_line>=21&&average\_line<=24))

result.code\_line=3;

else result.code\_line=4;

if (comments\_rate>=0.15&&comments\_rate<=0.25)

result.comments\_line=1;

else if ((comments\_rate>=0.10&&comments\_rate<=0.24)||(comments\_rate>=0.26&&comments\_rate<=0.30))

result.comments\_line=2;

else if ((comments\_rate>=0.05&&comments\_rate<=0.09)||(comments\_rate>=0.31&&comments\_rate<=0.35))

result.comments\_line=3;

else result.comments\_line=4;

if (blank\_rate>=0.15&&blank\_rate<=0.25)

result.blank\_line=1;

else if ((blank\_rate>=0.10&&blank\_rate<=0.24)||(blank\_rate>=0.26&&blank\_rate<=0.30))

result.blank\_line=2;

else if ((blank\_rate>=0.05&&blank\_rate<=0.09)||(blank\_rate>=0.31&&blank\_rate<=0.35))

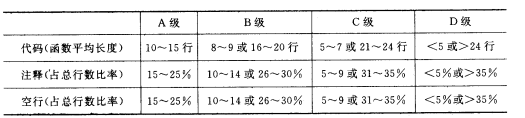
result.blank\_line=3;

else result.blank\_line=4;

return result;

}

根据表格



用if-else写的判定标准

1. **print函数——按照格式输出**

void print (struct count p,struct count result,char name[])

{

float comments\_rate,blank\_rate,average\_line,code\_rate;

average\_line=(float)p.code\_line/p.function;

code\_rate=(float)p.code\_line/p.total\_line;

comments\_rate=(float)p.comments\_line/p.total\_line;

blank\_rate=(float)p.blank\_line/p.total\_line;

printf("The results of analysing program file \"%s\"\n",name);

printf("\n");

printf(" Lines of code:%d\t\n",p.code\_line);

printf(" Lines of comments:%d\t\n",p.comments\_line);

printf(" Blank lines:%d\t\n",p.blank\_line);

printf("\n");

printf(" Code Comments Space\t\n");

printf(" ==== ======== =====\t\n");

printf(" %f%% %f%% %f%%\t\n",code\_rate\*100,comments\_rate\*100,blank\_rate\*100);

printf("\n");

printf(" The program includes %d functions\n",p.function);

printf(" The average length of a section of code is %.1f lines\n",average\_line);

printf(" The length of the deepest function is %d lines\n",p.maxlen);

printf(" The position is %d lines\n",p.position);

printf("\n");

if (result.code\_line==1)//根据judge中result的1,2,3,4来判断ABCD风格

printf(" Grade A:Excellent routine size style.\n");

else if (result.code\_line==2)

printf(" Grade B:Good routine size style.\n");

else if (result.code\_line==3)

printf(" Grade C:Normal routine size style.\n");

else if (result.code\_line==4)

printf(" Grade D:Bad routine size style.\n");

if (result.comments\_line==1)

printf(" Grade A:Excellent commenting style.\n");

else if (result.comments\_line==2)

printf(" Grade B:Good commenting style.\n");

else if (result.comments\_line==3)

printf(" Grade C:Normal commenting style.n");

else if (result.comments\_line==4)

printf(" Grade D:Bad commenting style.\n");

if (result.blank\_line==1)

printf(" Grade A:Excellent white space style.\n");

else if (result.blank\_line==2)

printf(" Grade B:Good white space style.\n");

else if (result.blank\_line==3)

printf(" Grade C:Normal white space style.n");

else if (result.blank\_line==4)

printf(" Grade D:Bad white space style.\n");

}

1. **调试分析**
2. 在编程过程中，对设计有过如下修改
3. analyse中的i++，起初是在while的末尾，这样无论进行什么判断最后都会+1进入下一轮循环。但是这样效率不高，因此改为将i++加入每个if判断中。这样可以批量跳过’ ‘和//后的注释内容，保证在读到换行符后i+1，blank=1
4. 对于函数体的统计，最初的代码设计是这样的

if (s[i]=='{')

{

brace++;

sign=1;

}

else if (s[i]=='}')

{

brace--;

if (brace==0&&sign==1)//一个函数体结束，根据{}的配对判定，sign表示已经有过{}的配对

{

if (p.maxlen<function\_line)

{

p.maxlen=function\_line;

p.position=p.total\_line-function\_line-1;

}

function\_line=0;

p.function++;

sign=0;

}

}

if (sign==1&&brace!=0)

{

if (s[i]=='\n')

{

function\_line++;

p.total\_line++;

blank=1;

}

}

if (s[i]=='s'&&s[i+1]=='t'&&s[i+2]=='r'&&s[i+3]=='u'&&s[i+4]=='c'&&s[i+5]=='t'&&brace==0&&sign==0)

{

p.function--;

}

i++;

}

}

p.code\_line=p.total\_line-p.blank\_line-p.comments\_line;

return p;

}

算法思想为计算所有的{}配对，然后排除如struct结构体的特殊情况进行function--。

最后引起的问题是在调用结构体的时候同样会出现struct和{},或者结构体作为函数输出时会造成干扰，这种算法反而更为复杂。组员在调试的时候测试这个程序本身就因为struct的问题而导致函数数量错误。

修改之后，不再单独考虑struct的情况，而是考虑函数体的特性，需要有一对（）表示调用的变量，从而增加一对brace2和sign2来判断{}前是否有这样的一对（）

1. 时间问题并没有能实现嵌套函数的统计，但是算法已经基本成型

算法思想——

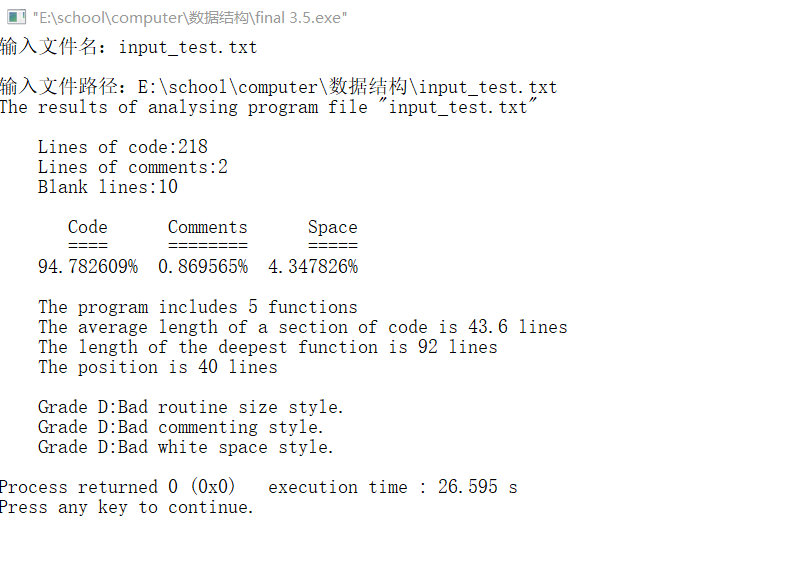
将函数统计部分单独作为一个函数，写成递归模式，在函数中判断函数再判断函数，从而实现嵌套定义函数的统计，得到最大的函数深度。

**五、用户手册**

1、本程序在主流操作系统上都可以运行，执行文件为：final 3.5.exe

2、进入程序后即显示提示信息：输入文件名，以等待用户输入需要进行程序分析的文件（以一个回车为结束标志的字符串）。之后会有另一条提示信息：输入文件路径，以等待用户输入需要进行程序分析的文件的所在路径（以一个回车为结束标志的字符串）。输入完成后即可等待程序运行。如果输入有误，会报出：错误！如果看到此提示信息需要重新执行输入。

3、输入结束后程序啧进行分析，输入输出的样例如下



第一行表示输入的程序名，即进行分析的程序。

后面三行分别表示代码行数、注释行数和空行数。

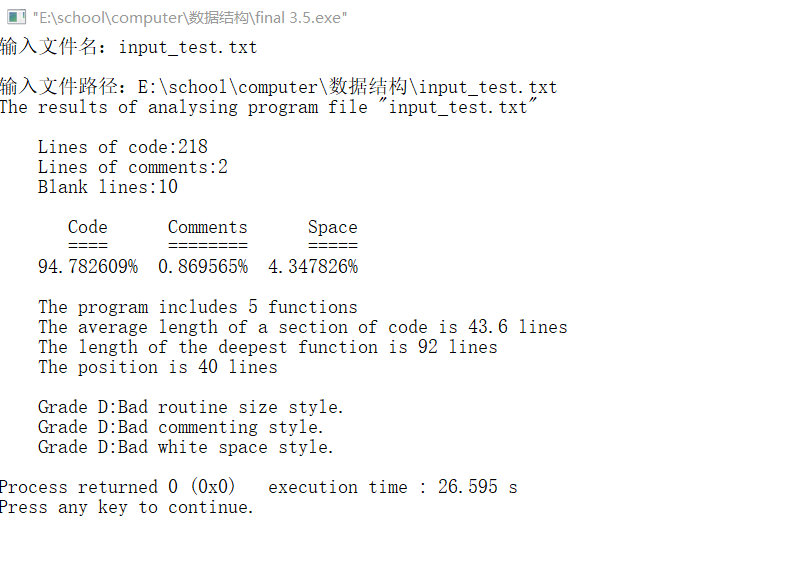
再下面三行则是代码、注释、空行占整个程序的比例。

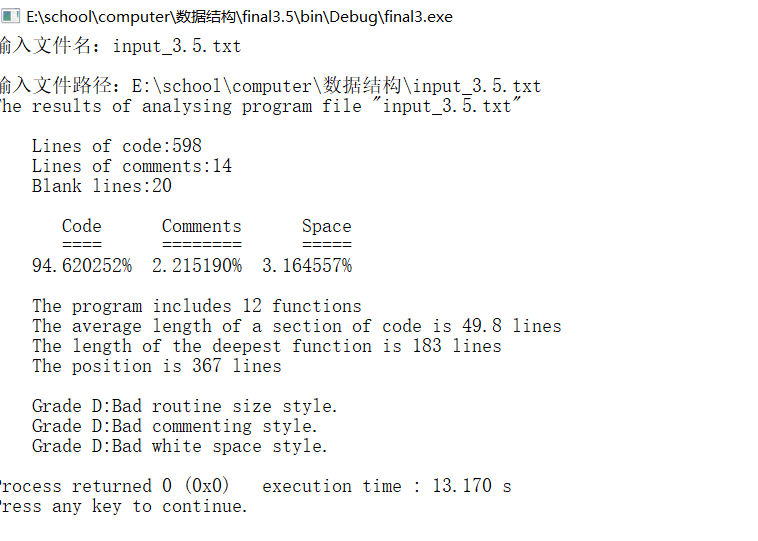
之后包括这个程序包含的函数个数，平均的函数长度，最长的函数行数，以及最长函数的所在位置。

三个Grade表示依照评价标准的代码风格评定结果。

1. 本程序只统计无语法错误的C语言代码程序。

**六、测试结果**





1. **小组成员贡献**

初期讨论——孙小丁、资威、张丽玮

读入文件与转化接口（getfile函数）——资威

程序的其他主体部分及后期调试——张丽玮