实验报告

题目：3.2文本格式化

班级：信卓1901班 姓名：刘可立 学号：U201913498

一、需求分析

1. 用户从键盘输入待格式化文本的文件名，处理后以文本文件形式输出。
2. 实现多余空格符的压缩，文本左对齐，以“@”符号指示换行。任何完整的字不能分占两行。
3. 若所要求的页底空行数不少于3行，则在第二行输出页码。
4. 让用户可自定义页长、页宽、左空白、头长、脚长、起始页号。对上述参数设置缺省值，分别取56，60，10，5，5，1。
5. 测试数据：

英文文本文件：hamlet.txt (181kb)

二、概要设计

为实现对文本进行格式化的功能，需对字符串进行读入，处理和输出。涉及到以下串的抽象数据类型：

1. 所使用的串的抽象数据结构定义如下：

ADT String {

数据对象：D = {ai | ai ∈ CharacterSet, i = 1, 2, ..., n, n>=0}

数据关系：R = {< ai-1 , ai > | ai-1 , ai ∈ D, i = 2, ..., n}

基本操作：

StrAssign(&T, chars)

构造一个其值等于chars的字符串

Strcpy(&L, s)

复制字符串L到s

StrCmp(L)

按字典序比较两个字符串

StrLen(s, t)

返回s的字符串⻓度

StrCat(&s1, s2)

把s2拷贝到s1的后面，接成一个⻓的字符串

DestroyString(&S)

销毁字符串S

}

1. 本程序包含两个模块：
2. 主程序模块：

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

接受用户输入：需格式化文件的位置以及控制参数;

while (未读到文件尾){

逐个读入数据；

处理数据；

输出到文件；

}

}

1. 实现串的抽象数据类型

三、详细设计

1. 主程序需要的全程量

//串的定长顺序存储表示

#define MAXSTRLEN 160

typedef unsigned char SString[MAXSTRLEN + 1];

//赋有缺省值的控制参数

int PageLen = 56, PageWed = 60, LeftMargin = 10, HeadingLen = 5, FootingLen = 5, StartPageNum = 1;

//分别进行读、写的文件指针

FILE \* fp1;

FILE \* fp2;

//记录行数，每一行字数来控制换行、翻页

int length = 0, width = 0, pagenum = StartPageNum;

1. 串抽象数据结构的基本操作伪代码如下：

int StrLength(SString s){

遍历串，直到读到串尾的'\0'，记录循环次数

}

void StrCat(SString s1, SString s2){

确保s1有足够的剩余空间

将s2的第一个字符拷贝到s1的第StrLength(s1)个字符

不断拷贝，直到s2的最后一个字符

}

char \* StrCpy(char \* dst, const char \* src){

从src的第一个字符开始，只要非0就拷贝到dst的相应位置

}

int StrCmp(const char \* s1, const char \* s2){

从第一个字符开始，逐个比较s1和s2对应位置上的字符

若相等则比较下一个

若不相等则输出两字符的差

若s1和s2完全相等，则返回0

}

1. 主函数和其他函数的伪代码算法

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

初始化

用户输入文档的文件名

调用SignValue函数让用户输入格式化的控制参数

执行Format函数对输入文档进行格式化

return 0;

}

void Format(FILE \* out, FILE \* in){

输出第一页的空行

while (文档未读完) {

fscanf(一个字);

如果字是"@"，则换行并且空8个字符

如果输出后会超过页宽限制，则调用换行函数

输出到本行，记录本行总字符数

}

}

void nextline(FILE \* out){

如果换行后需要翻页：

换行

调用翻页函数

否则：

输出一个'\n'

输出左空白

将新行的总字符记为0

}

void nextpage(FILE \* out){

如果页底空白不少于三行，则输出页码

输出若干个'\n'

将新页的总行数和新行的总字符记为0

页码+1

}

1. 函数调用关系

在主函数调用 SignValue() 函数以请求用户输入

在主函数调用 Format() 函数进行文档的格式化

Format() 函数调用了nextpage()、nextline()函数进行换行、翻页

Format() 函数调用对串操作的基本函数：Strlen()、StrCmp()、StrCat()、StrCpy()

四、调试分析

1. 对设计过程总结反思

在编程过程中，对设计进行过如下修改：

1. 输入数据时原本采用运行时在命令行中附加参数的形式来输入控制参数，考虑到参数过多，运行时太过繁琐。在用户输入参数前先询问是否采用参数的缺省值，以简化用户的输入。
2. 原先的程序只能以指定文件名进行输入，改进后可让用户能够自定义输入的文件路径，可采用相对路径或绝对路径。

在实验过程中遇到的问题：

1. 原本希望通过程序运行时重定向指定输入文档(% ./a.out < input.txt)，但是无法判断什么时候读入结束，所以改为让用户输入文件名。
2. 主要算法的时空分析
3. 空间复杂度

算法均在原地工作，使用的储存空间为常数O(1)。

1. 时间复杂度
2. StrLen()、StrCmp()、StrCat()、StrCpy()均只需要把输入的字符串遍历一遍便可得到结果，假设输入字符串长度为n（StrCmp两个输入的字符串中较短的长度为n），则时间复杂度均为O(n)。
3. 假设文档大小为n，则Format()函数的时间复杂度为O(n)。因为 Format() 将整个文档读入了一遍，相当于遍历了一次整个文档，而每一次的visit()进行对数据的处理和输出。

五、用户手册

1. 本程序的测试运行环境为MacOS，在 Terminal 运行 a.out
2. 在 Terminal 中定位到 a.out 所在的文件夹，输入 ./a.out 回车运行
3. 键入待处理文档的位置 例如：

/Volumes/Files/hamlet.txt

1. 按程序提示选择是否自定义控制参数，输入自定义的参数
2. 输出在可执行文件同目录下的output.txt文件。
3. 运行程序后的用户界面：

手机屏幕的截图

描述已自动生成

(若Terminal窗口太小可能导致提示文字显示不正确)

六、测试结果

1. 输入文件：hamlet.txt (额外在第一页加入了几个@以测试控制效果)

页长：50 页宽：60 左空白8 头长3 脚长3 起始页号1

输出文件见hamletoutput.txt

七、附录

清单：

/TextFormation.xcodeproj 项目文件

/TextFormation/main.cpp 主程序

/TextFormation/Header.h 头文件

/TextFormation/a.out 可执行文件

/TextFormation/hamlet.txt 测试输入

/TextFormation/hamletoutput.txt 测试输出