实验一 预处理实验报告

一、 实验目的

设计一个程序，从任意字符串中剔除 C 语言形式的注释，包括：

1. 形如：//……的单行注释；
2. 形如：/*……*/的多行注释

二、 实验要求

1. 使用 C 语言或 C++语言编写
2. 输入为字符串形式的源程序，输出为处理之后的字符串形式的源程序
3. 该实验是在词法分析之前，对程序员提交的源程序进预处理，剔除注释等不必要的字符，以简化词法分析的处理

三、 实验过程

1. 状态转换图的分析

我们通过状态转换图的状态变化来确定程序的整体思路

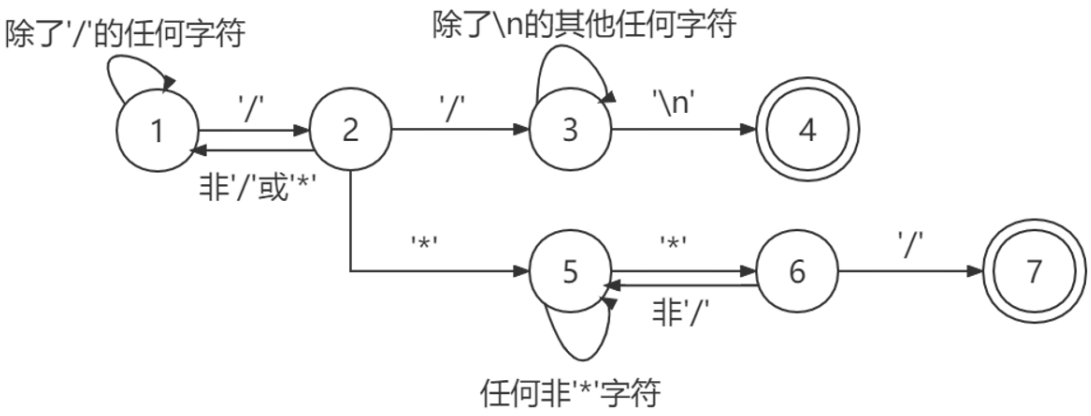

图 1 剔除注释的状态转换图

具体思路如下：

a. 当输入一个字符串后，开始读取字符

b. 当读入的字符为’/’时，进入状态 2

c. 上一个读取的字符为’/’，当前状态为 2，当前读取的字符为’/’则进入状态 3

d. 进入状态 3 则表明该行后面的都是注释内容可以删除，直到遇见换行符，进入结束状态 4

e. 上一个读取的字符为’/’，当前状态为 2，当前读取的字符不为’/’也不为’*’则返回状态 1

f. 上一个读取的字符为’/’，当前状态为 2，当前读取的字符为’*’则进入状态 5

g. 进入状态 5 后表明后续的内容可能都是注释，一直读直到遇见’*’进入状态 6

h. 进入状态 6 后，如果后一个字符为’/’说明/* */之间的内容为多行注释内容，可以删除，然后进入结束状态 7

i. 若进入状态 6 后，后一个字符不为’/’则返回状态 5

2. 输入输出的分析

这里使用了两种输入和输出方式进行测试：

a. 使用 getchar 从标准输入流中读取多行字符，将字符存入字符数组中传入 scanner 进行扫描并返回删除注释后的字符数组；

b. 使用 FILE 类型的指针读写.txt 类型的文件，将要处理的字符串存入 input.txt 中，通过指针的操作将其读入字符数组中，传入 scanner 进行扫描并返回删除注释的字符数组后，由另一个文件 result.txt 所指指针将结果写入输出文件中

3. 实验中问题的考虑

a. 当一开读入的字符就是'/'时，如何判断其是否应该保存？

b. 为了简化处理操作，这里不对多余的空格和换行符再做处理

c. 当存在多行注释时，应当先将原来的指针位置保存，等待确定了多行注释的具体位置之后，再将其位置更新

d. 需要时刻考虑遍历读入字符串时指针的溢出情况

4. 源程序的编写

a. 从标准输入流中输入字符串的输入方式的主程序编写

```
int MAX_LEN = 1000;
int main()
{
    char str[MAX_LEN];
    char ch;
    int i = 0;
    //读入多行字符串 使用CTRL+Z结束读入
    while((ch=getchar())!=EOF)
    {
        str[i++] = ch;
    }
    str[i] = '\0';
    char*p = scanner(str);
    printf("\n原输入为:\n");
    printf("%s\n", str);
    printf("\n删除注释后输出为:\n");
    printf("%s", p);
    return 0;
}
```

b. 从文件中输入字符串的主程序编写

```
int main()
{
    FILE *fp, *fp1;//文件指针定义
    //读写的错误处理
    if ((fp = fopen("input.txt", "r")) == NULL)
    {
        cout << "read error!" << endl;
        exit(1);
    }
    if ((fp1 = fopen("result.txt", "w")) == NULL)
    {
        cout << "write error!" << endl;
        exit(1);
    }
    char str[MAX_LEN];
    int i = 0;
    char ch = fgetc(fp);
    //从文件中读入字符串
    for(i=0;feof(fp)==0;i++)
    {
      str[i] = (char)ch;
      ch = fgetc(fp);
    }
    str[i] = '\0';
    printf("%s", str);
    char * p  = scanner(str);
    while( (*p != '\0') && fputc(*(p++),fp1) != EOF ) ;//将字符串写入文件
    fclose(fp);
    fclose(fp1);
    system("pause");
    return 0;
}
```

c. 删除注释的 scanner 函数的编写

函数功能：将输入的源程序字符串中的单行注释和多行注释删除

函数输入：char 类型的数组

函数输出：char 类型的删除注释后的数组

利用变量 state 进行状态的转换

```
char*scanner(char str[])
{
    int pos = 0;
    static char str_out[1000];
    int length = strlen(str);
    // 遍历字符串形式的源程序
    // 通过状态转换达到剔除目的
    int i = 0;
    int state;
    //刚开始进行特判
    if(str[0]=='/')
    {
        state = 1;
        if(i+1<length)
            i++;
        else
        {
            str_out[0] = '/';
            return str_out;
        }
    }
    else
        state = 0;
    while(i<length)
    {
        //进入状态1
        if(state == 0)
        {
            // 当前字符写入结果
            str_out[pos++] = str[i++];
            if(str[i]=='/')
            {
                state = 1;
                if(i+1<length)
```

```
                    i++;
                else
                    break;
            }
        }
        //同一行的都是注释
        if(state == 1)
        {
            if(str[i]=='/')
            {
                //同一行内后续的都是注释，不需要输出
                while(i<length&&str[i]!='\n')
                    i++;
                if(i>=length)
                    break;
                else
                    //把该行剩余的\n除去
                    i++;
                state = 0;
            }
            else if(str[i]=='*')
                //直到找到第二个*否则都先认为是注释
                state = 2;
            else
                //返回状态0
                state = 0;
        }
        if(state==2)
        {
            int p = i+1;
            //往后寻找第一个*
            while(p<length&&str[p]!='*')
                p++;
            if(str[++p]=='/')
            {
                //说明从i-p该段为注释应该删除
                state = 0;
                i = p;
                if(i+1<length)
                    i++;
                else
                    break;
            }
            //如果不是则继续往前查看
            else
                i++;
        }
    }
    return str_out;
}
```

5. 结果测试

a. 测试一：

b. 测试二：

输入：input.txt

输出：result.txt

输入输出的结果正确

四、 实验总结

1. 利用状态转换图或者有穷自动机来完成程序的算法设计可以起到事半功倍的效果
2. 学习了利用 C 语言实现文件的读写和字符串中类 C 注释的删除的程序设计
3. 算法和程序还可以继续改进，比如识别三斜杠时的处理’///’；在删除注释的同时将多余的空格和换行符一并进行删除等