华北电力大学

课程设计报告

（ 2024--2025 年度第一学期）

名　　称：编译技术课程设计

题　　目：C 语言的预处理程序

院　　系：控制与计算机工程

班　　级：计算 2201

学　　号：120221080320

学生姓名：薛春晖

指导教师：王红

设计周数：一周

成　　绩：

日期：2024 年 12 月 29 日

1 课程设计目的和要求

1.1 课程设计目的

本次设计的时间为 1 周，目的是使用高级语言实现部分算法加强对编译技术和理论的理解。设计一个 C 语言的预处理程序。将给出 C 语言中所有的宏常量进行计算，并生成另外一个文件，将宏常量展开和计算的结果全部显示出来。将定义的宏在源程序中全部进行替换。

1.2 课程设计的要求

1、需要完成算术、关系、逻辑、位、赋值、条件、逗号、下标运算符的语法分析

2、定义上述表达式的文法

3、给出语法分析过程和计算过程

4、通过界面与文件输出翻译后的文件

2 系统分析

本次实验使用 IntelliJ IDEA 社区版作为编程工具，将输入的代码 txt 文件中的每一个单词分割并进行信息的存储，然后在利用递归将处理好的表达式进行语法分析，对于语法分析中结果为可求解的宏常量进行计算，通过求解后缀表达式将宏常量展开。

C 语言预处理程序的所有中间过程以及按钮交互都通过 GUI 界面展示，最后的翻译文件保存为 answer 文件。

2.1 语法分析

首先将算术、关系、逻辑、位、赋值、条件、逗号、下标运算符分为四大类：单目前置运算符，单目后置运算符，双目运算符，三目运算符。

单目前置运算符：！和 ~；

单目后置运算符：++和--；

双目运算符：+，-，*，/，%，<，<=，==，>，>=，!=，&&，||，<<，>>，|，^，&，，；

三目运算符：？：；

将以上运算符存储到对应的哈希表中。

2.2 表达式文法

（1）S->num

（2）S->S

（3）S->(S)

（4）S->OP S

（5）S->S OP

（6）S->S OP S

（7）S->S1?S2:S3

将语句通过以上表达式来进行判断是否可规约，如果可以规约将进行下一步计算。

2.3 语法分析过程与计算

该程序主要进行的是“#define key value；”类型的计算。通过语法分析判断 value 是

否是可计算的。如果是可计算的，将其转换为后缀表达式，然后利用栈与运算符之间的优先级关系进行求解。如果不可以计算，那么直接跳过，执行下一条语句。

3 概要设计

3.1 概要设计

本功能主要分为五个部分：读入程序，词法分析，语法分析，语义分析，最后结果生成。

第一步（读入程序）：利用 Reader 类建立读管道，从被选择的文件中读入数据流，并存储到 dataCode（ArrayList 类型）中。

第二步（词法分析）：事先建立好一个符号哈希表，将所有运算符存入其中。然后利用 while 循环判断 dataCode 中读到的数据是不是分隔符，如果是，那么将分隔符前面的字符串写入 dataCode 中，如果为操作符，那么将其直接写入 dataCode 中。并将程序中出现的十六进制转换为十进制。

```
String[] iniData = {"+", "-" , "*" , "/", "%", "(", ")",
        "<", "<=", "==", ">=", ">", "&&", "||", "!",
        "<<", ">>", "~", "|", "^", "&",
        "=", "+=", "=+", "-=", "=-", "++", "--",
        "?", ":",
        ",",
        "[", "]", "[]",
        "{", "}", ";", " ", "\n", "\r", "\t", String.valueOf((char)9)};
```

图 3-1 分隔符定义

第三步（语法分析）首先将经过词法分析处理好的 dataCode 进行传入，dataObject 中存储的是变量名所对应的类型值，和把“#define key value；”中的 value 放在 object_valueStr 中当作对应的求值 String。dataDefineData.dataWrite 将（关键字，对应的 dataObject）映射存储到 dataDefineData 中。然后进行递归，从 value 第一项开始扫描，如果在 dataDefineData 中找到了对应项，那么就用他去替换。最后通过表达式文法自下而上进行判断是否可规约，将是否可规约的结果存储在 boolean 类型的 isOperate 中。

第四步（语义分析）首先知道运算符的优先级关系（[] > (! ~) > (* / %) > (+ -) > (<< >>) > (> >= < <=) > (== !=) > & > ^ > | > && > || > ?: > (= += -= /= *= %=) > ,(逗号)），然后构建 priorityOperator 运算符矩阵，接下来求解后缀表达式，最后通过 getTheEndAnswer 判断是字符串还是单个值，如果是串的话，对应前面的 dataDictionary 寻找对应的下标进行替换；如果是单个值，直接通过映射关系得到运算结果。

第五步（最后结果生成）首先将 dataCode 每一行的值清空，然后逐行扫描单词，如果发

现该变量有定义，那么用其 dataObject.operate 的值进行替换，如果没有被定义就复制下来。

3.2 系统用例图

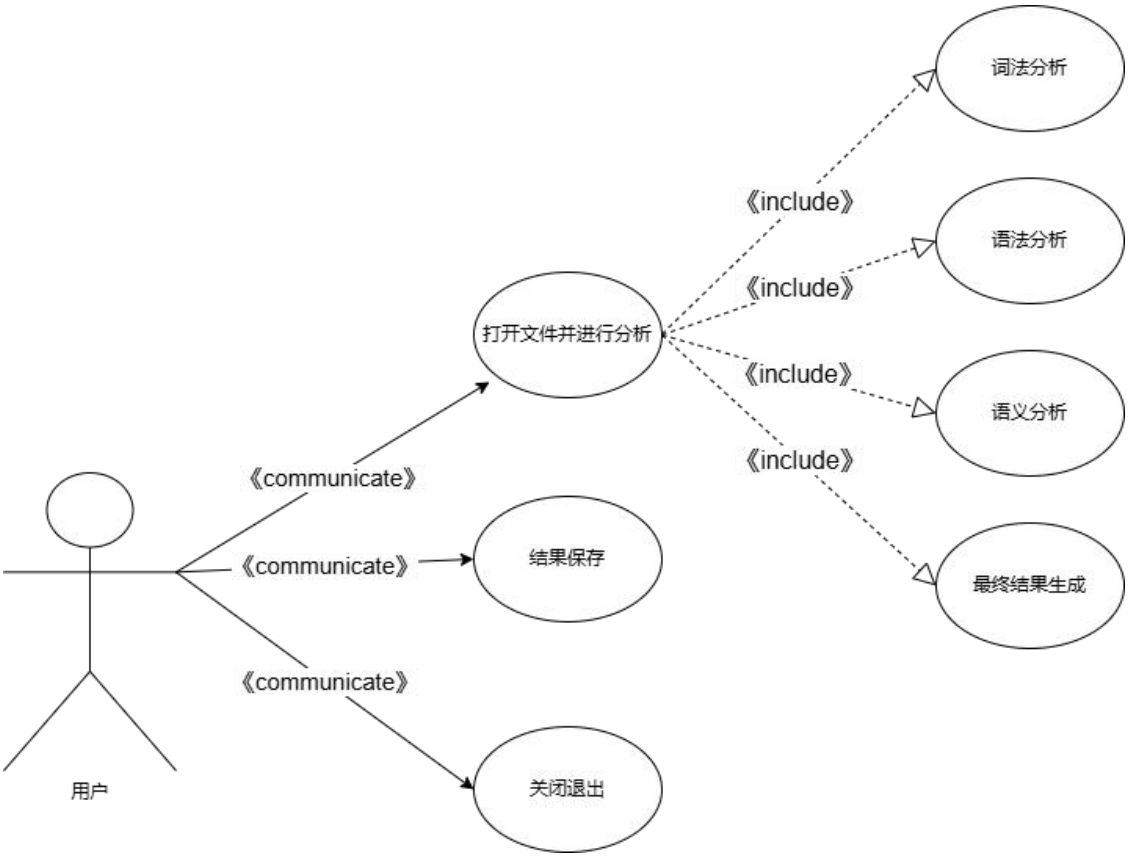

图 3-2 C 语言预处理程序的用例图

3.3 系统用例

表 3-3 C 语言预处理程序用例

用例名称	C 语言预处理程序
简述	将给出 C 语言中所有的宏常量进行计算，将宏常量展开和计算的结果全部显示出来，将定义的宏在源程序中全部进行替换，并生成另外一个文件。
角色	程序操作者
前置条件	已经进入 C 语言预处理程序的 GUI 界面
基本流	1. 程序操作者在 C 语言预处理程序的 GUI 界面中点击“打开” 2. 出现文件浏览对话框，选择需要导入的 txt 文件，点击“确定” 3. 系统输出提示信息 4. 系统输出读入的程序代码文件 5. 系统进行词法分析，并在完成时输出提示信息

	6. 系统进行语法分析，并在完成时输出提示信息 7. 系统输出关键字的名字，关键字的类型，关键字对应的求值字符串 8. 系统进行语义分析，并在完成时输出提示信息 9. 系统输出求得的后缀表达式 10. 系统输出后缀表达式求得的值 11. 系统进行最终结果输出，并在完成时输出提示信息 12. 退出并给出提示信息 【用例结束】
备选流	2.1 txt 文件不存在 a 系统提示文件读取失败 b 系统返回程序初始界面 6.1 运算符使用不规范导致判断错误 a 信息异常的记录数量+1 b 跳到【12】 【用例失败】
后置条件	导入成功，成功进行宏常量展开计算，成功将代码进行翻译
特殊需求	只能导入 txt 文件
待解决问题	变量命名不规范无法精准识别

3.4 开发环境

本次实验使用 IDEA Community Edition 2024.1.4 软件，整体采用 Java 语言。

4 详细设计

4.1 系统的类图

画出系统的所包括的所有类及类之间的关系图，如 4-1 所示。

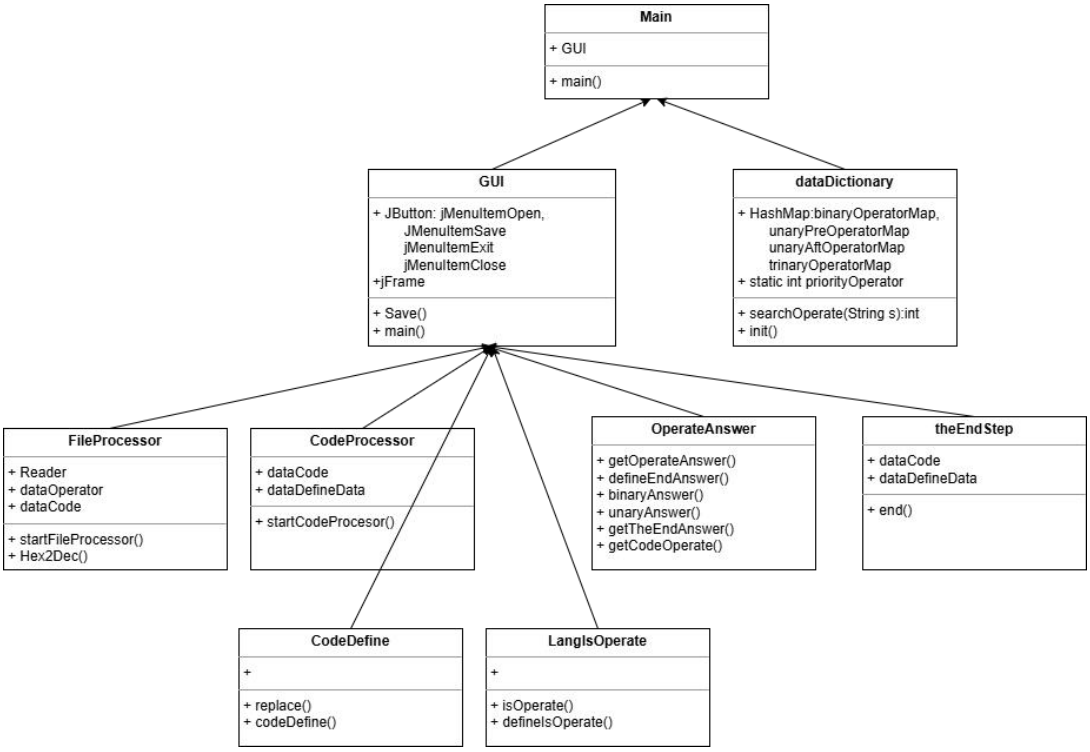

图 4-1　C 语言预处理程序类图

4.2 主要算法的流程图

算法主要流程如图 4-2 所示。

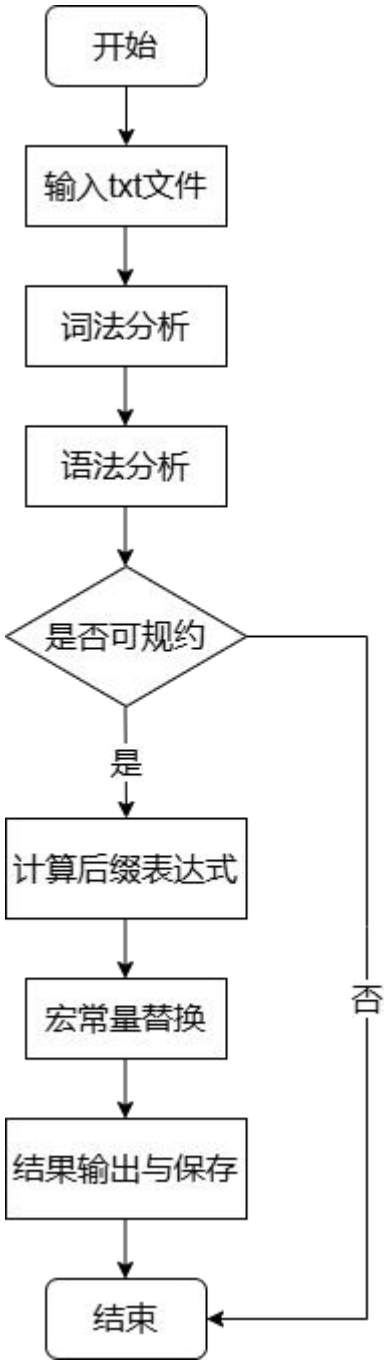

图 4-2　C 语言预处理程序流程图

4.3 数据分析与定义

表 4-3-1 C 语言预处理程序的数据类

类名	功能	说明
data	对使用到的数据格式进行封装	拥有四个子类：data_code,data_defineData,data_object,data+operator
data_code	将整段代码进行存储	负责读入整段代码，用 ArrayList<ArrayList>进行存

		储。拥有 BetterCode 进行代码优化
data_defineData	保存一个关键字的所有信息，类似于字典	
data_object	保存一个关键字的相关信息	xxvalue 表示该关键字的值，object_isOperate 进行存储判断是不是可以进行后续操作的可操作数，object_valueStr 存储该值的定义字符串，operate 存储该关键字的后缀表达式，含有 mustDo 方法，进行相对应的值计算
data_operator	存储所有的运算符,用于分割字符串	

表 4-3-2 C 语言预处理程序的过程类

类名	功能	说明
Main	进行函数的初始化和开始	
dataDictionary	存储所有的 operator 并进行分类,将其转换为 hashmap,而且存储优先关系表	binaryOperatorMap（二元） unaryPreOperatorMap（一元前） unaryAftOperatorMap（一元后） trinaryOperatorMap（三元） 运算符到相应数字的映射 priorityOperator 存储运算符优先关系
GUI	界面类	拥有四个事件处理：打开，保存，退出，关闭
judge	判断是不是十六进制，并将其类型值进行返回（十六进制，int，float，String）	
FileProcessor	将代码进行分割，分割为一个一个不含运算符的语句	进行词法分析
CodeProcessor	进行判断，并给 a 打上标签，划分属性	进行语法分析
CodeOpen	将所有#define a x 中的 x 进行替换	进行语法分析
LangIsOperate	判断每一个 x 是否是可以运算的	进行语法分析
OperateAnswer	将 x 转换为后缀表达式，并进行计算	进行语义分析
theEndStep	进行最后值的替换（翻译）	进行结果输出

4.4 系统界面设计

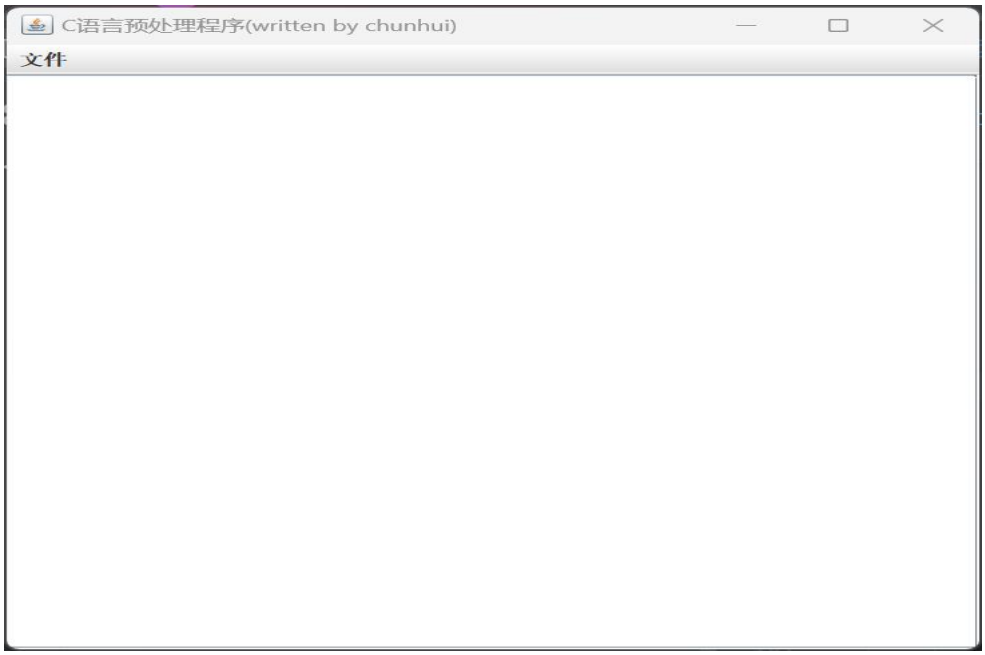

图 4-4-1 C 语言预处理程序的初始界面

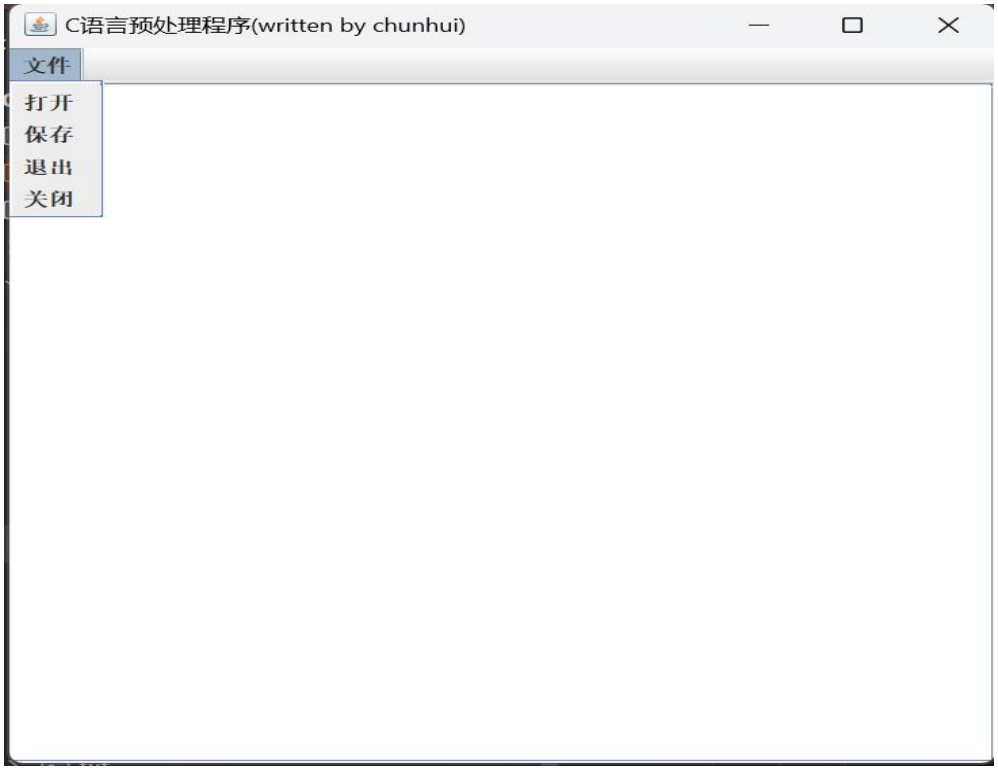

图 4-4-2 C 语言预处理程序的选择界面图

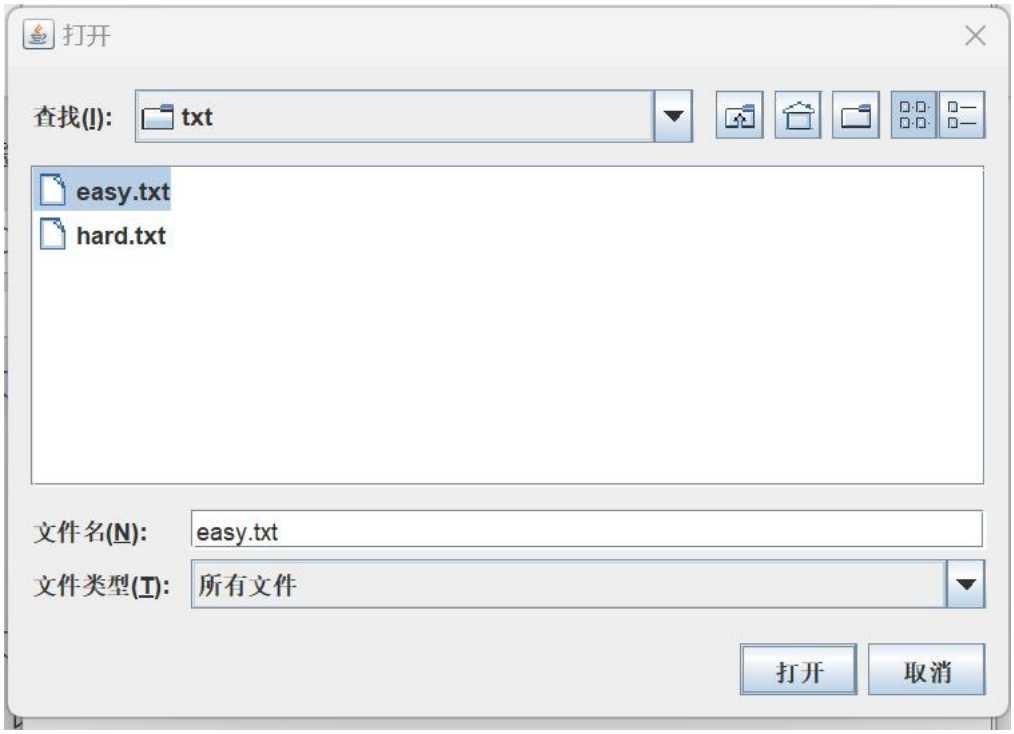

图 4-4-3 C 语言预处理程序的打开选择界面

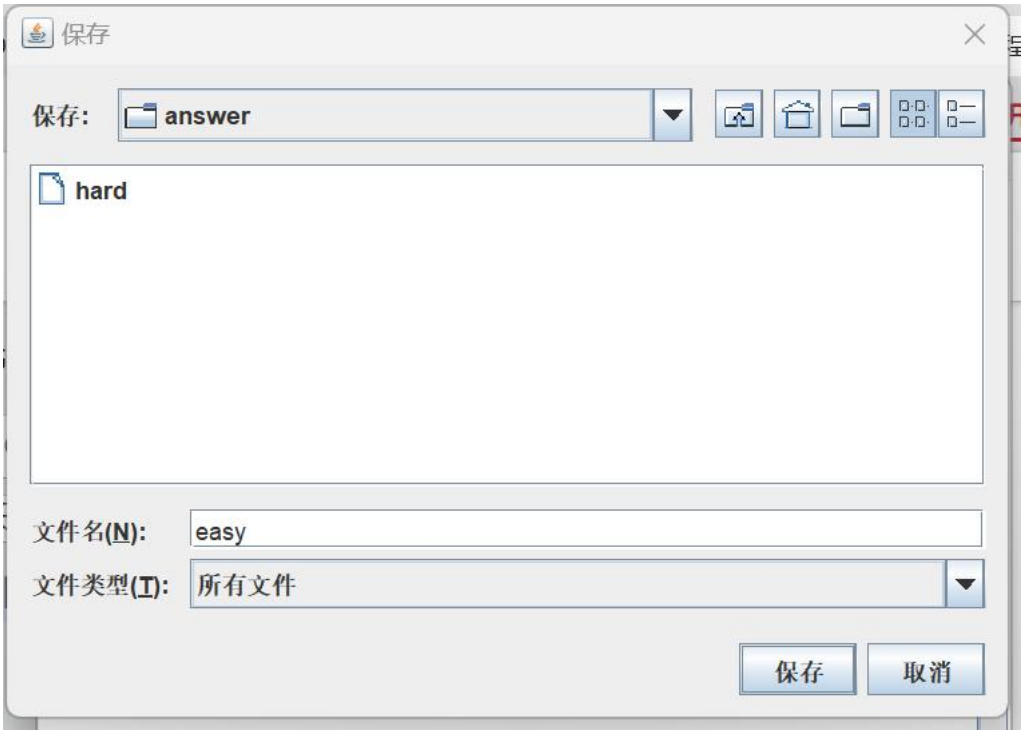

图 4-4-4 C 语言预处理程序的保存选择界面

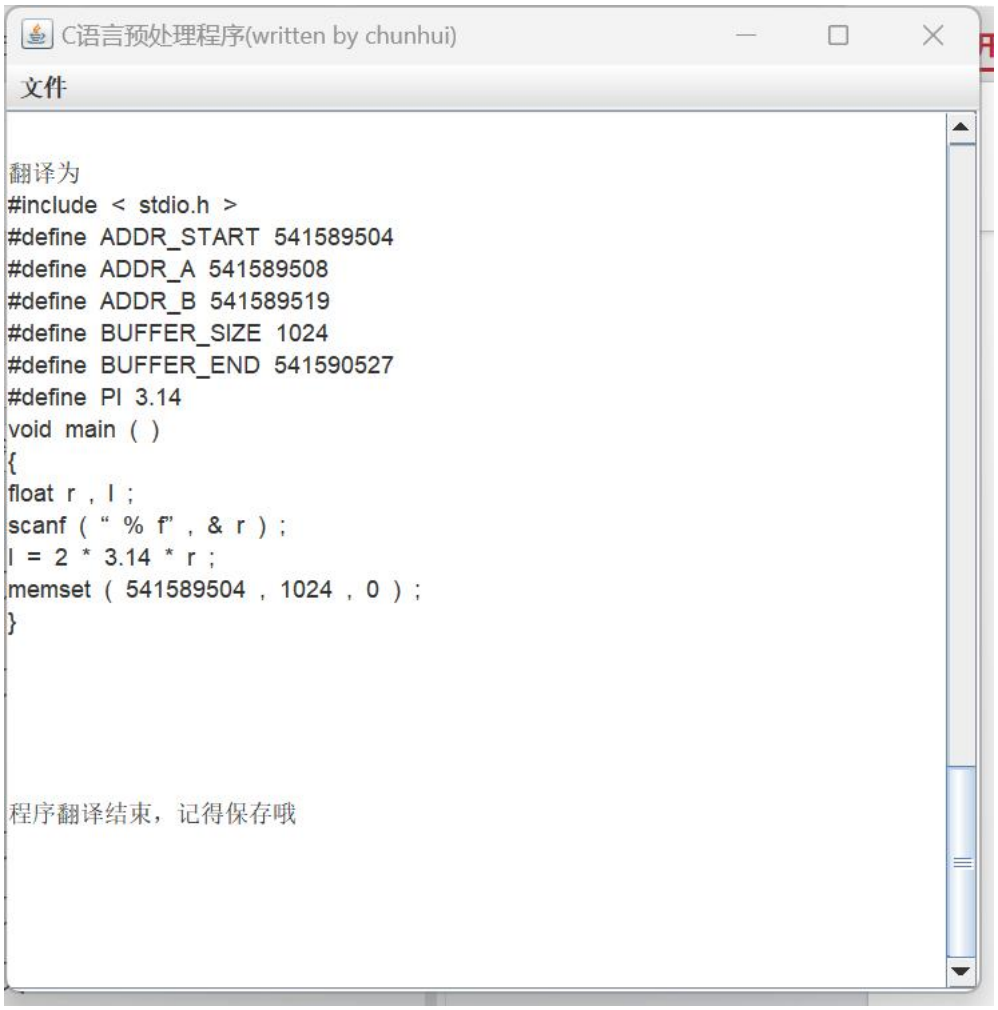

图 4-4-5 C 语言预处理程序的最终结果界面

5 测试方法和测试结果

5.1 测试用例 1

测试目的：是否能够将宏常量进行计算和替换，并将程序进行翻译。

输入如程序 5-1 所示。

```
#include <stdio.h>
#define   ADDR_START     0x20480000
#define   ADDR_A             ADDR_START + 0x04
#define   ADDR_B             ADDR_START + 15
#define   BUFFER_SIZE    1024
#define   BUFFER_END       ADDR_START + BUFFER_SIZE-1
#define   PI          3.14
void main()
{
      float r, l;
      scanf(“%f”, &r);
      l = 2 * PI *r;
      memset(ADDR_START, BUFFER_SIZE, 0x00);
}
```

程序 5-1 输入运算符较少的源程序

宏常量计算过程如图 5-1 所示。

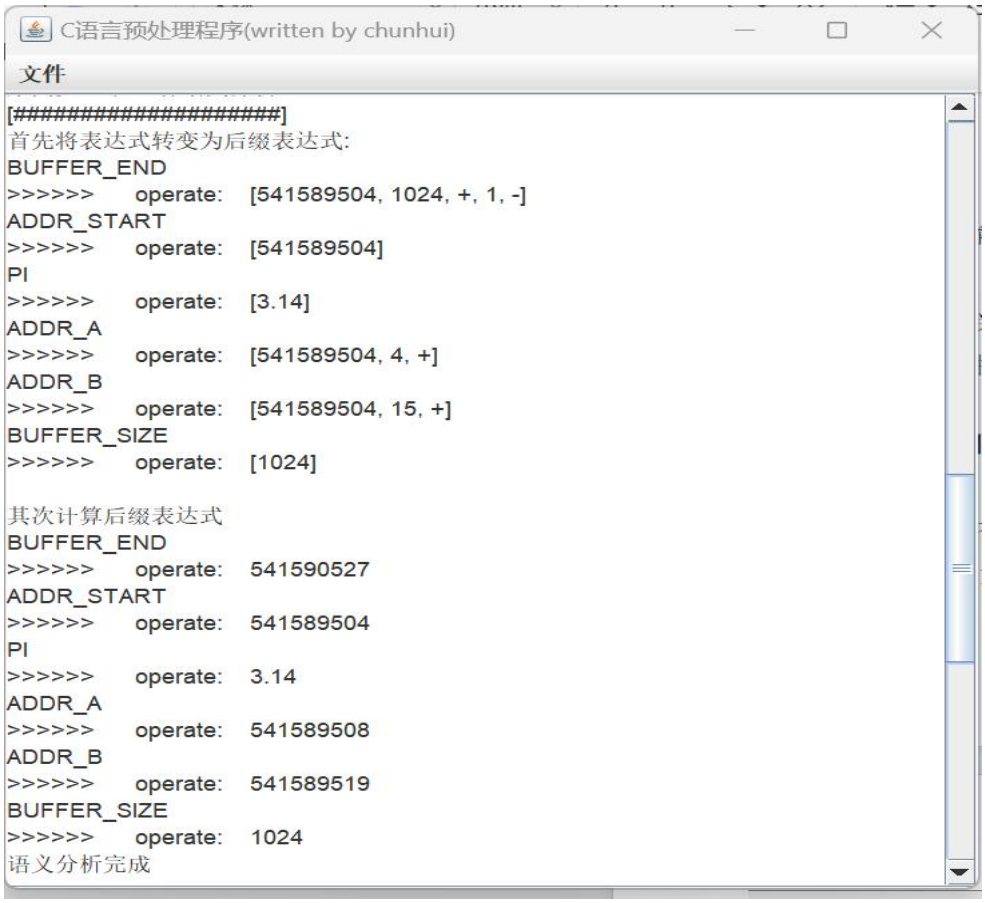

图 5- 1-1 宏常量的计算过程

最终翻译结果输出如图 5-1-2 所示

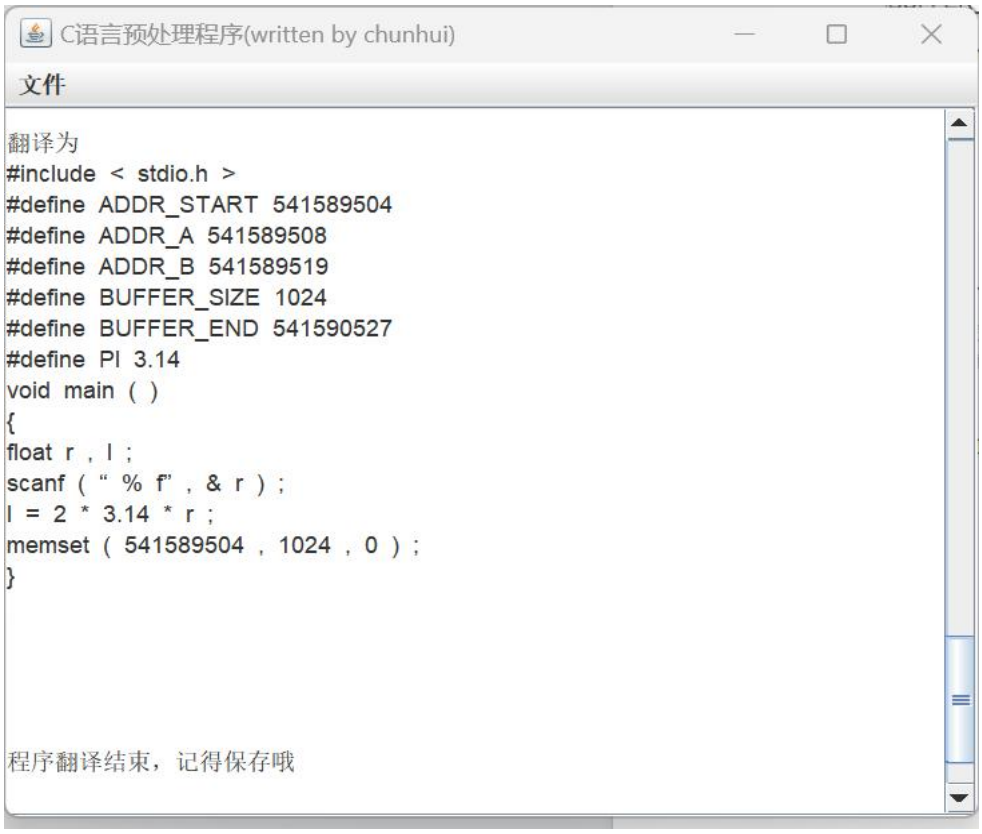

图 5- 1-2 将宏常量替换得到的结果

5.2 测试用例 2

测试目的：在运算符种类较多、运算表达式较为复杂的情况下，是否还能够将宏常量进行计算和替换，并将程序进行翻译。

输入内容如程序 5- 2 所示。

```
#include <stdio.h>
#define a 0xff
#define b 2
#define c a&&b
#define d a+b-1
#define e 3+(8-6>>1)*2+3
#define g 3.14*2
#define z printf("%d",a);
void main()
{
    float l,r;
    z
    scanf("%f", &r);
    l = 2 * g *r;
    memset(a, e, 0x00);
}
```

程序 5-2 运算符种类较多、运算表达式较为复杂的源程序

宏常量计算过程如图 5- 1 所示。

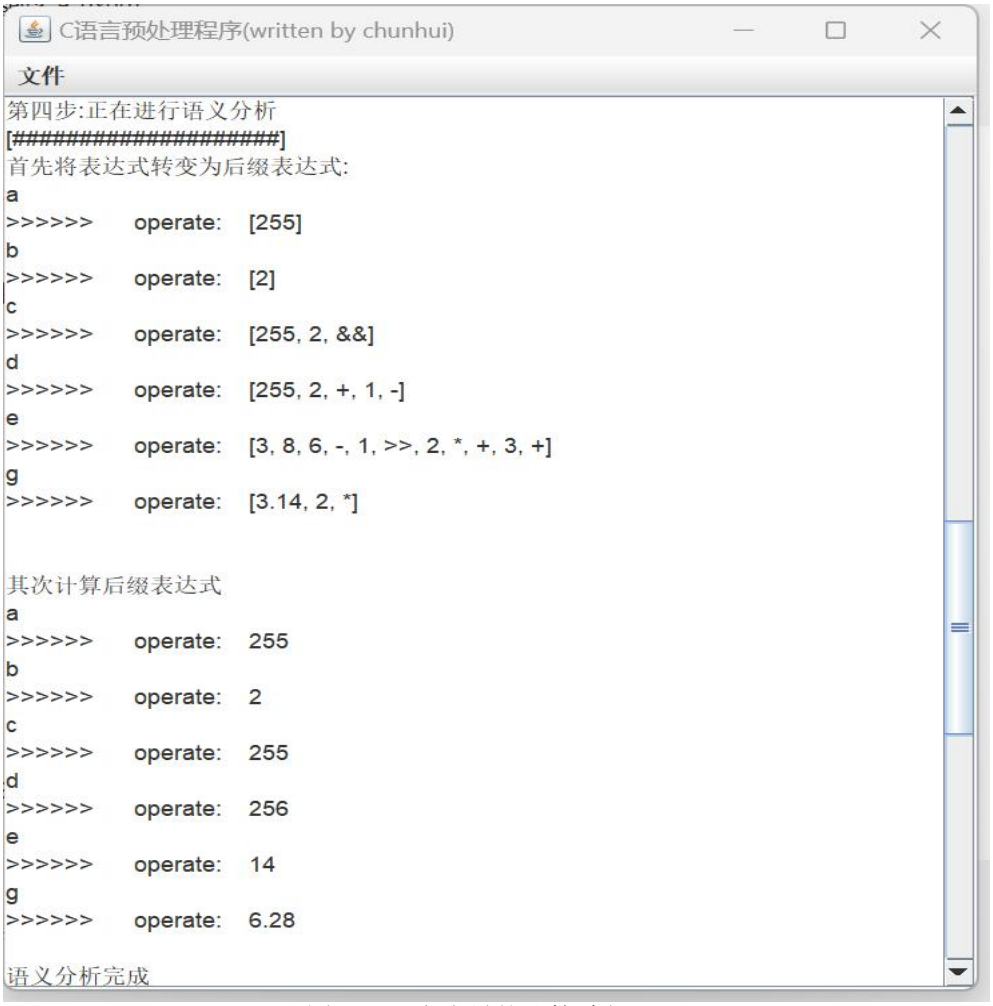

图 5-2-1 宏常量的计算过程

输出结果如图 5- 3 所示。

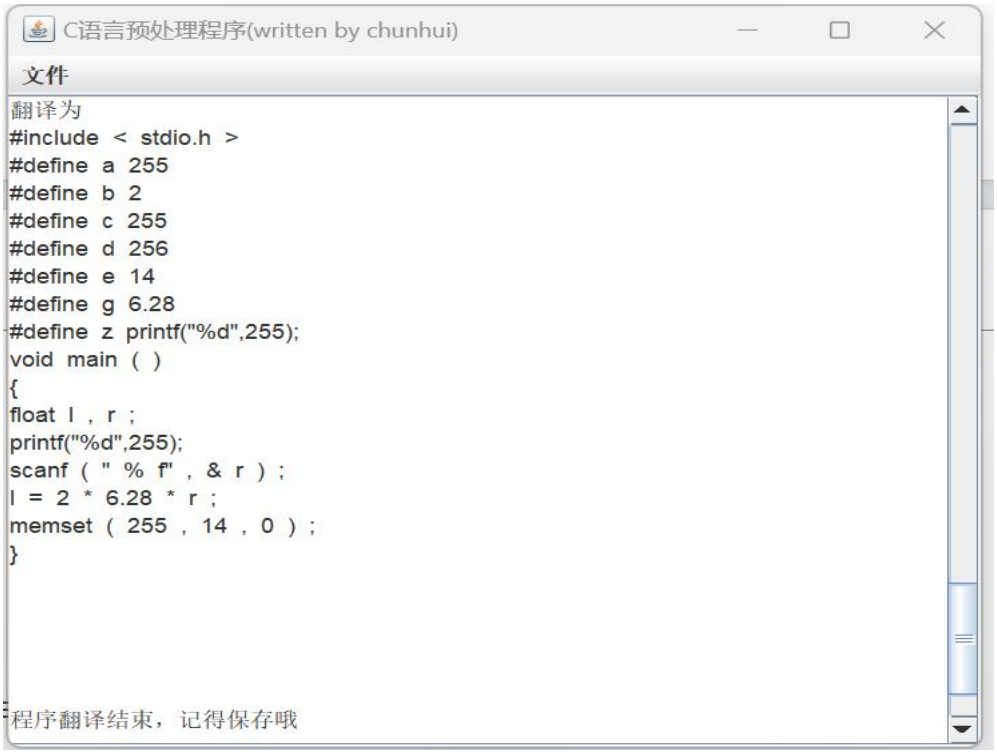

图 5-2-2 将宏常量替换得到的结果

5.3 测试用例 3

```
#include <stdio.h>
#define a 0xff
#define b 2
#define c a||b
#define d a+b-1
#define e 3+(8-6>>1)*2+3
#define g 6%5
void main()
{
    int a;
    area=g*e*e;
    float l,r;
    scanf("%f", &r);
    l = 2 * g *r;
    printf("%d",a+b-c);
}
```

程序 5-3 运算符种类较多、运算表达式较为复杂的源程序

宏常量计算过程如图 5- 3-1 所示。

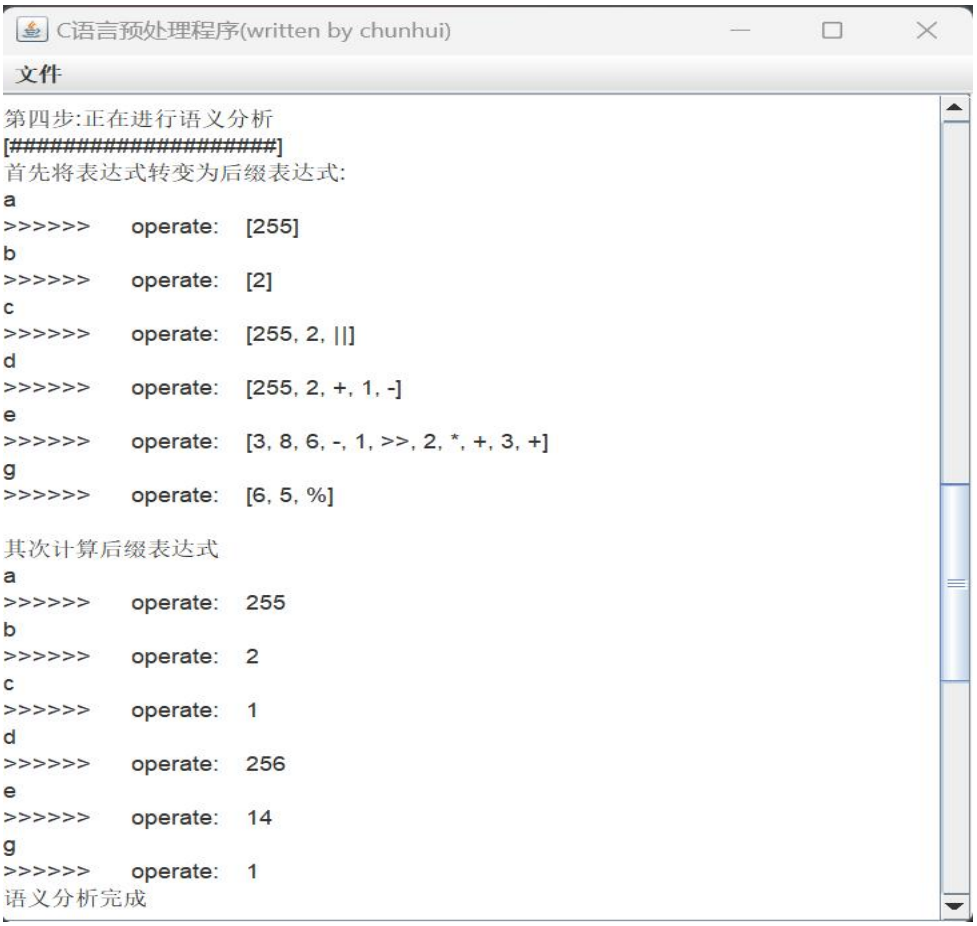

图 5-3-1 宏常量的计算过程

输出结果如 5-3-2 所示。

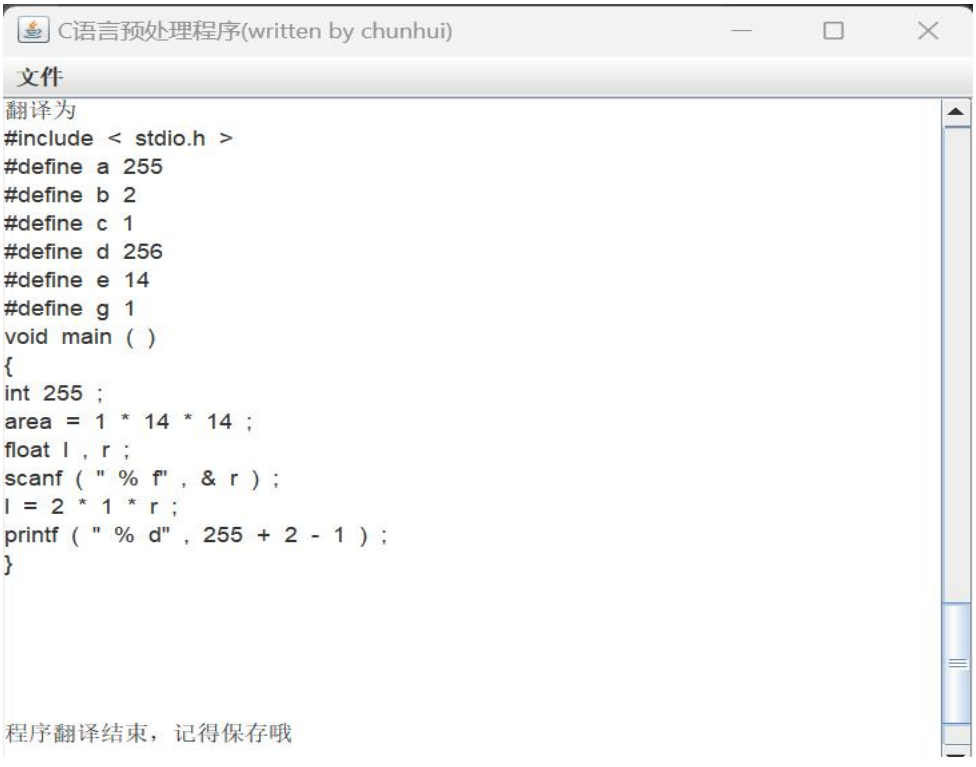

图 5-3-2 将宏常量替换得到的结果

6 结论和展望

6.1 结论

本选题目的是使用高级语言设计一个C语言的预处理程序，将给出C语言中所有的宏常量进行计算，将宏常量展开和计算的结果全部显示出来，并将定义的宏在源程序中全部进行替换。加深对编译原理的理解。通过这个项目，我深入了解了课本中形式语言理论中的关键概念，并且实践了这些概念在实际编程中的应用。

实际操作过程复合上课所讲七个步骤：词法分析，语法分析，语义分析，中间代码生成，代码优化，目标代码生成，差错处理。首先词法分析负责将预处理文件进行扫描，识别单词并进行分割。语法分析负责判断宏常量的定义是否是可计算可规约的。语义分析负责通过求解后缀表达式计算宏常量。中间代码生成负责将代码中宏常量部分进行替换。但是代码优化，差错处理这两部分由于时间不够没有进行拓展。

6.2 展望

本次实验的不足还有很多，由于时间不足，对程序一些优化还没有做好。例如，代码优化部分没有将生成的代码文件中空格和制表符进行优化输出，导致输出的代码层次结构不分明。缺少差错处理，默认输入的文件都是正确的，当遇到连续使用运算符或者变量命名不规范时代码容易出错。而且，输出界面不是很友好，在分析过程中没有达到交互式的要求。

收获：通过本次实验，我收获了很多东西。首先对编译有了进一步的深刻理解，对编译预处理的过程有了更加清晰的认识，同时对于语法分析的原理和过程都有了进一步的巩固，对于Java的编程水平也起到了锻炼的作用，巩固了平时所学的知识，真正做到了学以致用，对于各种结构体的运用也有了深刻的认识。

6.3 学习体会和教学评价

在《编译技术》这门课程的学习过程中，我深刻体会到了计算机科学理论与实践相结合的魅力。通过深入研究编译器的工作原理，我不仅理解了编程语言如何被转换成机器码，还掌握了词法分析、语法分析、语义分析以及代码生成和优化等关键步骤。这些知识让我对程序执行的本质有了更清晰的认识，并提升了我对代码效率和性能的关注。同时，C语言预处理程序的设计让我对编译过程有了更加清晰的认知。

在整个教学过程中，理论讲解与实际操作的结合使得我们能够在实践中加深对理论知识的理解。课程设计是一个非常好的学习机会。课程内容对于我来说还是有一定难度的，从基础概念逐步过渡到复杂的技术细节，这样的安排有助于我们建立坚实的知识体系。教师提供了大量的参考资料、在线教程和工具链支持，帮助我们在课外进一步扩展学习范围。然而，更多的课堂互动和实战机会将会使课程更加生动有趣，也能够让学生获得更加全面的经验。

参考文献

参考文献要至少列出 1 个，格式如下。

[1] 王生原.编译原理(第 3 版)[M].北京:清华大学出版社,1998 年

[2] 舍里(shelly,G.B)等. 李芳等译. 系统分析与设计教程[M], 北京: 机械工业出版社, 2004, 79-96.

[3] 刘新民. Java 程序设计[M]. 北京: 清华大学出版社, 2004, 61-101.