

天津理工大学 实验报告

学院名称：计算机科学与工程学院

姓名	王帆	学号	20152180	专业	计算机科学与技术
班级	2015 级 1 班	实验项目	实验一：词法分析		
课程名称	编译原理		课程代码	0668056	
实验时间	2018 年 5 月 23 日 第 1、2 节 2018 年 5 月 28 日 第 5、6 节		实验地点	软件实验室 7-219 软件实验室 7-212	
实验成绩考核评定分析					
实验过程 综合评价 30 分	实验目标 结果评价 20 分	程序设计 规范性评价 20 分	实验报告 完整性评价 30 分	实验报告 雷同分析 分类标注	实验 成绩
■实验过程认真专注，能独立完成设计与调试任务 30 分 ■实验过程认真，能较好完成设计与编成调试任务 25 分 ■实验过程较认真，能完成设计与编成调试任务 20 分 ■实验过程态度较好，基本完成设计与编成调试任务 15 分 ■实验过程态度欠端正，未完成设计与编成调试任务 10 分	■功能完善，且人机交互界面友好 20 分 ■满足功能要求，但人机交互界面一般 15 分 ■基本满足功能需求，人机交互界面欠缺 10 分 ■功能缺失 5 分	■程序易读性好 20 分 ■程序易读性较好 15 分 ■程序易读性欠缺 10 分 ■程序易读性较差 5 分 **注：易读性要求标识符命名见名知意，程序编制采用嵌套方式，层次结构清晰可读，关键部分具有简明注释。	■报告完整 30 分 ■报告较完整 25 分 ■报告内容一般 20 分 ■报告内容极少 10 分	凡雷同报告将不再重复评价前四项考核内容，实验成绩将按低学号雷同学生成绩除雷同人数计算而定。 标记为： S 组号-人数(组分)	前四项评价分数之总和 (**雷同报告按第五项标准核算**)
<p>实验内容： 实现标准 C 语言词法分析器</p> <p>实验目的：</p> <ol style="list-style-type: none"> 1. 掌握程序设计语言词法分析的设计方法； 2. 掌握 DFA 的设计与使用方法； 3. 掌握正规式到有限自动机的构造方法； <p>实验要求：</p> <ol style="list-style-type: none"> 1. 单词种别编码要求 基本字、运算符、界符：一符一种；标识符：统一为一种；常量：按类型编码； 2. 词法分析工作过程中建立符号表、常量表，并以文本文件形式输出； 3. 词法分析的最后结果以文本文件形式输出； 4. 完成对所设计词法分析器的功能测试，并给出测试数据和实验结果； 5. 为增加程序可读性，请在程序中进行适当注释说明； 6. 整理上机步骤，总结经验和体会； 7. 认真完成并按时提交实验报告。 					

1.单词种别编码

大类	类别	单词符号	种别编码	助记符	内码值
基本字	类型说明保留字	int	1	int	—
		long	2	long	—
		short	3	short	—
		float	4	float	—
		double	5	double	—
		char	6	char	—
		unsigned	7	unsigned	—
		signed	8	signed	—
		const	9	const	—
		void	10	void	—
		volatile	11	volatile	—
		enum	12	enum	—
		struct	13	struct	—
		union	14	union	—
	语句定义保留字	if	15	if	—
		else	16	else	—
		goto	17	goto	—
		switch	18	switch	—
		case	19	case	—
		do	20	do	—
		while	21	while	—
		for	22	for	—
		continue	23	continue	—
		break	24	break	—
		return	25	return	—
		default	26	default	—
		typedef	27	typedef	—
	存储类说明保留字	auto	28	auto	—
		register	29	register	—
		extern	30	extern	—
		static	31	static	—
	长度运算符保留字	sizeof	32	sizeof	—
运算符	前缀（后缀）运算符	++	33	++	—
		--	34	--	—
	函数调用运算符	(35	bracket	LSB
)	36	bracket	RSB
	下标运算符	[37	bracket	LMB
]	38	bracket	RMB
	复合字面量	{	39	bracket	LLB

大类	类别	单词符号	种别编码	助记符	内码值
运算符	复合字面量	}	40	bracket	RLB
	成员访问运算符	->	41	mao	TP
		.	42	mao	TO
	逻辑运算符	!	43	inverse	—
		~	44	bti	—
		_Alignof	45	_Alignof	—
	四则运算符	+	46	+	—
		-	47	-	—
		*	48	*	—
		/	49	/	—
	模运算符	%	50	mod	MOD
	移位运算符	<<	51	so	LL
		>>	52	so	RR
	关系运算符： 大（等）于、小（等）于	<=	53	relop	LE
		<	54	relop	L
		>=	55	relop	RE
		>	56	relop	R
	关系运算符： 等于、不等于	==	57	relop	EQ
		!=	58	relop	NE
	按位异或运算符	^	59	xor	—
	关系运算符:与	&	60	relop	AND
	关系运算符:或		61	relop	OR
	关系运算符:短路与	&&	62	relop	SA
	关系运算符:短路或		63	relop	SO
	三目运算符	?	64	?	—
		:	65	:	—
	双目运算符	*=	66	binop	ME
		/=	67	binop	DE
		+=	68	binop	PE
		-=	69	binop	FE
		<<=	70	binop	LDE
		>>=	71	binop	RDE
		&=	72	binop	AE
		=	73	binop	OE
		^=	74	binop	BE
	逗号运算符	,	75	,	—
界符	界符	/*	76	/*	—
		*/	77	*/	—
		"	78	"	—
		'	79	'	—
		;	80	;	—

大类	类别	单词符号	种别编码	助记符	内码值
标识符	标识符	标识符	81	id	id 在符号表中的位置
常量	常量	常量	82	num	num 在常量表中的位置

表 1 C 语言单词符号表

对于一个程序语言来说，基本字、运算符、界符的数目是确定的，通常一个单词可以定义一个类别码，单词与它的类别码为一一对应的关系。此时不需要第二元素作为识别码。

另一类是将运算符等统一归类一种，并用第二元素作为识别码进行区别。

在本实验中，我选用 C#进行界面设计，并将这些结果在图形化界面中展示，从而便于导出。

2.符号表、常量表建立与词法分析文本输出

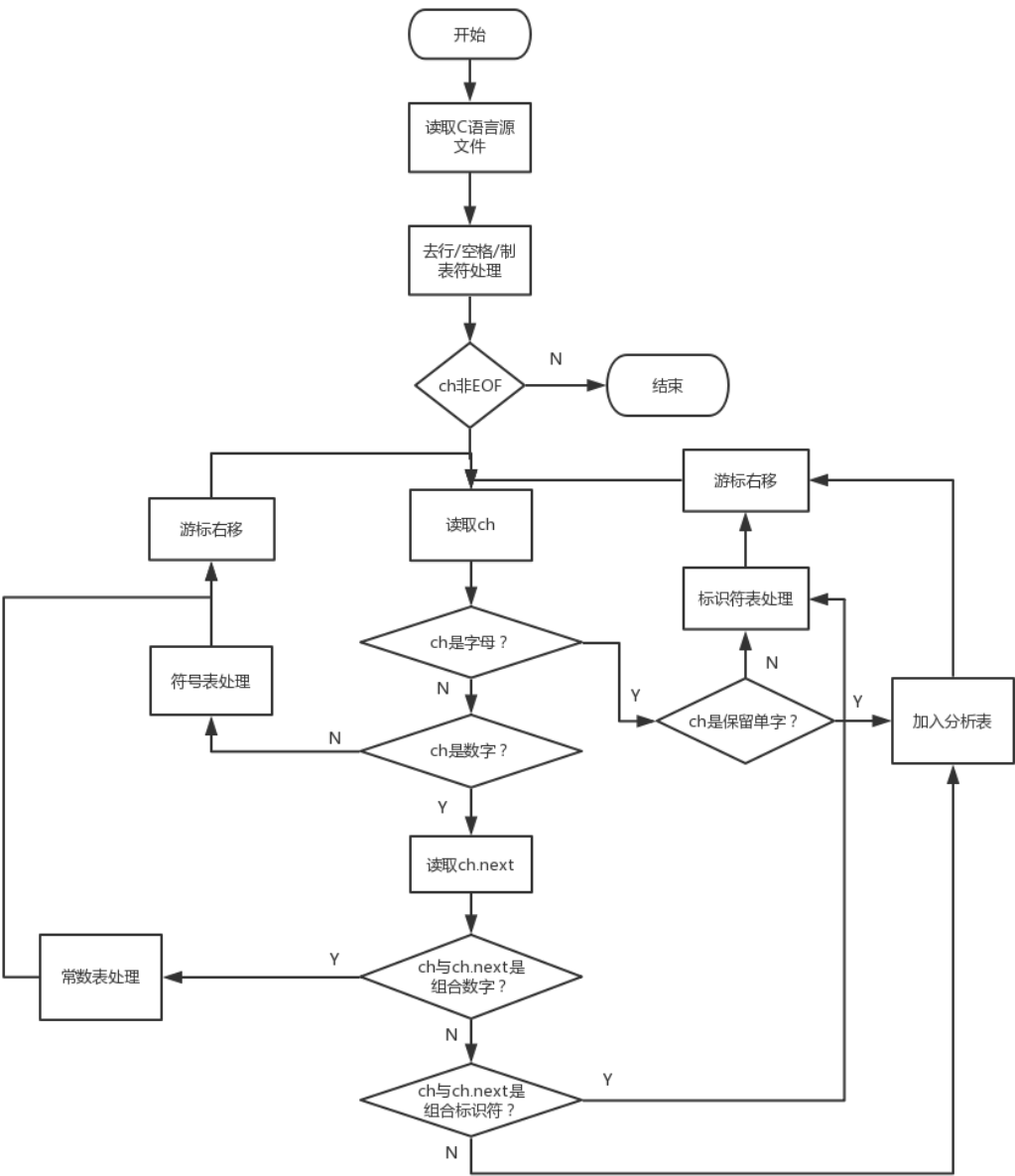


图 1 词法分析整体流程

```
//界面逻辑
using CompilerLab.ChildForm.WordAnalysis.Class;
```

第4页 共19页

```

using System;
using System.Collections;
using System.Windows.Forms;

namespace WordAnaly.ChildForm.WordAnalyse
{
    public partial class WordAnalyse : Form
    {
        CodeFormatterFactory codeFormatterFactory = new CodeFormatterFactory();
        /*C 语言所有关键字，共 32 个*/
        ArrayList KeyWordList = new ArrayList();
        /*运算、限界符*/
        ArrayList LimitList = new ArrayList();
        /*常量表*/
        ArrayList ConstList = new ArrayList();
        /*标识符*/
        ArrayList IdentifierList = new ArrayList();
        /*输出*/
        ArrayList OutputList = new ArrayList();
        public string sourceText = null;
        public WordAnalyse()
        {
            InitializeComponent();
        }
        private void WordAnalyse_Load(object sender, EventArgs e)
        {
            codeFormatterFactory.SourceCode = sourceText;
            codeFormatterFactory.init();
            codeFormatterFactory.Parse();
        }
        private void 符号表 ToolStripMenuItem_Click(object sender, EventArgs e)
        {
            this.IdentifierList = new
ArrayList(codeFormatterFactory.GetLimitList ());
            string str = "";
            foreach (object o in LimitList)
            {
                str += o.ToString()+ "\n";
            }
            this.richTextBox.Text = str;
        }
        private void 常量表 ToolStripMenuItem_Click(object sender, EventArgs e)
        {
            this.OutputList=new ArrayList(codeFormatterFactory.GetConstList ());
            string str = "";

```

```

        foreach(object o in ConstList)
        {
            str += o.ToString()+"\n";
        }
        this.richTextBox.Text = str;
    }
    private void 整体分析 AToolStripMenuItem_Click(object sender, EventArgs e)
    {
        this.OutputList=new
ArrayList(codeFormatterFactory.GetOutputList());
        string str = "";
        foreach(object o in OutputList)
        {
            str += o.ToString()+"\n";
        }
        this.richTextBox.Text = str;
    }
}
//
//词法分析逻辑
using System;
using System.IO;
using System.Text;
using System.Windows.Forms;
using System.Collections;

namespace CompilerLab.ChildForm.WordAnalysis.Class
{
    /// <summary>
    /// CodeFormatterFactory 的摘要说明。
    /// c 代码解析,不支持中文
    /// </summary>
    public class CodeFormatterFactory
    {

        /*C 语言所有关键字，共 32 个*/
        ArrayList KeyWordList = new ArrayList();
        /*运算、限界符*/
        ArrayList LimitList = new ArrayList();
        /*常量表*/
        ArrayList ConstList = new ArrayList();
        /*标识符*/
        ArrayList IdentifierList = new ArrayList();
        /*输出*/

```

```

ArrayList OutputList = new ArrayList();
public ArrayList GetOutputList()
{
    return OutputList;
}
public ArrayList GetIdentifierList()
{
    return IdentifierList;
}
public ArrayList GetConstList()
{
    return ConstList;
}
public ArrayList GetLimitList()
{
    return LimitList;
}

public CodeFormatterFactory()
{
    init();
}

public CodeFormatterFactory(string s)
{
    this.SourceCode = s;
    init();
}

public string SourceCode { get; set; } = "";

public string ParseMessages
{
    get
    {
        string pm = "";

        IEnumerator ie = this.OutputList.GetEnumerator();
        while (ie.MoveNext())
            pm += ie.Current.ToString() + "/r\n";
        return pm;
    }
}

public void init()

```

```

{
    /*C 语言所有关键字，共 32 个*/
    string[] key = new string[]{"
", "auto", "break", "case", "char", "const", "continue", "default", "do", "double",
    "else", "enum", "extern", "float", "for", "goto", "if", "int", "long", "register",
    "return", "short", "signed", "sizeof", "static", "struct", "switch", "typedef",
    "union", "unsigned", "void", "volatile", "while"};

    /*运算、限界符*/
    string[] limit = new string[]{"
", "(", ")", "[", "]", "-", ">", ".", "!", "++", "--", "&", "~",
    "*", "/", "%", "+", "-", "<<", ">>", "<", "<=", ">", ">=", "==", "!=", "&&", "||",
    "=", "+=", "-=", "*=", "/=", ",", ";", "{", "}", "#", "_", "'"};

    this.KeyWordList.Clear();
    this.KeyWordList.TrimToSize();
    for (int i = 1; i < key.Length; i++)
        this.KeyWordList.Add(key[i]);

    this.LimitList.Clear();
    this.LimitList.TrimToSize();
    for (int i = 1; i < limit.Length; i++)
        this.LimitList.Add(limit[i]);

    this.ConstList.Clear();
    this.ConstList.TrimToSize();

    this.IdentifierList.Clear();
    this.IdentifierList.TrimToSize();

    this.OutputList.Clear();
    this.OutputList.TrimToSize();
}

/*****
* 十进制转二进制函数
*****/
public string dtb(string buf)
{
    int[] temp = new int[20];
    string binary = "";
    int val = 0, i = 0;

```



```

/*先将字符转化为十进制数*/
try
{
    val = Convert.ToInt32(buf);
}
catch
{
    val = 0;
}

if (val == 0)
{
    return (val.ToString());
}

i = 0;
while (val != 0)
{
    temp[i++] = val % 2;
    val /= 2;
}

binary = "";
for (int j = 0; j <= i - 1; j++)
    binary += (char)(temp[i - j - 1] + 48);

return (binary);
}

/*****
* 根据不同命令查表或造表函数
*****/
public int find(string buf, int type, int command)
{
    int number = 0;
    string temp;

    IEnumerator ie = null;
    ArrayList al = null;
    switch (type)
    {
        case 1://关键字表
            ie = this.KeyWordList.GetEnumerator();
            break;

```

```

        case 2://标识符表
            ie = this.IdentifierList.GetEnumerator();
            break;
        case 3://常数表
            ie = this.ConstList.GetEnumerator();
            break;
        case 4://运算、限界符表
            ie = this.LimitList.GetEnumerator();
            break;
    }

    if (ie != null)
        while (ie.MoveNext())
        {
            temp = ie.Current.ToString();
            if (temp.Trim().ToLower() == buf.Trim().ToLower())
            {
                return number;
            }
            number++;
        }

    if (command == 1)
    {
        /*找不到，当只需查表，返回 0，否则还需造表*/
        return 0;
    }

    switch (type)
    {
        case 1:
            al = this.KeyWordList;
            break;
        case 2:
            al = this.IdentifierList;
            break;
        case 3:
            al = this.ConstList;
            break;
        case 4:
            al = this.LimitList;
            break;
    }
    if (al != null)
        al.Add(buf);

```

```

        return number + 1;
    }
    /*****
    * 数字串处理函数
    *****/
    public void cs_manage(string buffer)
    {
        string binary = dtb(buffer);
        int result = find(binary, 3, 2);
        this.OutputList.Add(String.Format("{0} 3 {1}", buffer, result));
    }

    /*****
    * 字符串处理函数
    *****/
    public void ch_manage(string buffer)
    {
        int result = find(buffer, 1, 1);
        if (result != 0)
        {
            this.OutputList.Add(String.Format("{0} 1 {1}", buffer,
result));
        }
        else
        {
            result = find(buffer, 2, 2);
            this.OutputList.Add(String.Format("{0} 2 {1}", buffer,
result));
        }
    }

    /*****
    * 出错处理函数
    *****/
    public void er_manage(char error, int lineno)
    {
        this.OutputList.Add(String.Format("错误关键字: {0} , 所在行: {1}",
error, lineno));
    }

    /*****
    * 转换 Char 数组为 string
    *****/
    public string joinString(char[] array, int Length)

```

```

{
    string s = "";
    if (array.Length > 0)
        for (int i = 0; i < Length; i++)
        {
            if (array[i] != '\0')
            {
                s += array[i];
            }
            else
            {
                break;
            }
        }
    return s;
}

public char getchc(ref int n)
{
    char[] c = SourceCode.ToCharArray();
    if (n < c.Length)
    {
        char r = c[n];
        n++;
        return r;
    }
    return SourceCode[SourceCode.Length - 1];
}
/*****
* 扫描程序
*****/
public void Parse()
{
    try
    {

        //StreamWriter fpout = null;
        char ch;
        int i = 0, line = 1;
        int count, result, errorno = 0;
        char[] array = new char[30];
        string word = "";

        /*按字符依次扫描源程序，直至结束*/
        int n = 0;

```

```

while (n < SourceCode.Length - 1)
{
    i = 0;
    ch = getchc(ref n);
    /*以字母开头*/
    if (((ch >= 'A') && (ch <= 'Z')) || ((ch >= 'a') && (ch <= 'z'))
|| (ch == '_'))
    {
        while (((ch >= 'A') && (ch <= 'Z')) || ((ch >= 'a') && (ch <=
'z')) || (ch == '_') || ((ch >= '0') && (ch <= '9'))))
        {
            array[i++] = ch;
            ch = getchc(ref n);
        }
        array[i++] = '\0';
        word = joinString(array, array.Length);
        ch_manage(word);
        if (n < SourceCode.Length)
            n--;
    }
    else if (ch >= '0' && ch <= '9')
    {
        /*以数字开头*/
        while (ch >= '0' && ch <= '9')
        {
            array[i++] = ch;
            ch = getchc(ref n);
        }
        array[i++] = '\0';
        word = joinString(array, array.Length);
        cs_manage(word);
        if (n < SourceCode.Length)
            n--;
    }
    else if ((ch == ' ') || (ch == '\t'))
        /*消除空格符和水平制表符*/
        ;
    else if (ch == '\n')
        /*消除回车并记录行数*/
        line++;
    else if (ch == '/')
    {
        /*消除注释*/
        ch = getchc(ref n);
    }
}

```

```

if (ch == '=')
{
    /*判断是否为‘/='符号*/
    this.OutputList.Add(String.Format("/= 4 32"));
}
else if (ch != '*')
{
    /*若为除号，写入输出*/
    this.OutputList.Add(String.Format("/ 4 13"));
    n--;
}
else if (ch == '*')
{
    /*若为注释的开始，消除包含在里面的所有字符*/
    count = 0;
    ch = getchc(ref n);
    while (count != 2)
    {
        /*当扫描到‘*’且紧接着下一个字符为‘/’才是注释的结束*/
        count = 0;
        while (ch != '*')
            ch = getchc(ref n);
        count++;
        ch = getchc(ref n);
        if (ch == '/')
            count++;
        else
            ch = getchc(ref n);
    }
}
else if (ch == '"')
{
    /*消除包含在双引号中的字符串常量*/
    this.OutputList.Add(String.Format("{0} 4 37", ch));
    while (ch != '"')
        ch = getchc(ref n);
    this.OutputList.Add(String.Format("{0} 4 37", ch));
}
else
{
    /*首字符为其它字符,即运算限界符或非法字符*/
    array[0] = ch;
    /*再读入下一个字符，判断是否为双字符运算、限界符*/
    ch = getchc(ref n);
}

```

```

/*若该字符非结束符*/
if (n < SourceCode.Length)
{
    array[1] = ch;
    array[2] = '\0';
    word = joinString(array, 2);
    result = find(word, 4, 1); /*先检索是否为双字符运算、限界符
*/

    if (result == 0)
    {
        /*若不是*/
        array[2] = '\0';
        word = joinString(array, 1);
        result = find(word, 4, 1);
        /*检索是否为单字符运算、限界符*/
        if (result == 0)
        {
            /*若还不是，则为非法字符*/
            er_manage(array[0], line);
            errorno++;
            n--;
        }
        else
        {
            /*若为单字符运算、限界符，写入输出并将扫描指针回退一个
字符*/
            this.OutputList.Add(String.Format("{0}    4    {1} ",
word, result));

            n--;
        }
    }
    else
    {
        /*若为双字符运算、限界符，写输出*/
        this.OutputList.Add(String.Format("{0}    4    {1}",
word, result));
    }
}
else
{
    /*若读入的下一个字符为结束符*/
    array[2] = '\0';
    word = joinString(array, 1);
    /*只考虑是否为单字符运算、限界符*/
    result = find(word, 4, 1);

```

```

        /*若不是，转出错处理*/
        if (result == 0)
            er_manage(array[0], line);
        else
        {
            /*若是，写输出*/
            this.OutputList.Add(String.Format("{0} 4 {1}",
word, result));
        }
    }
    }
    ch = getchc(ref n);
}
/*报告错误字符个数*/
this.OutputList.Add(String.Format("\n 共有 {0} 个错误.\n", errorno));
}
catch (Exception ex)
{
    //错误提示
    MessageBox.Show(ex.Message, "错误提示", MessageBoxButtons.OK,
MessageBoxIcon.Stop);
}
}
}
}
}

```

3.功能测试

测试用例：

//Test.c

```
#include <stdio.h>
```

```

int main(){
    int a;
    a=10;
    printf("Hello!%d",&a);
    return 0;
}

```

测试过程：

1.打开实验软件，选择文件，打开待测试 c 语言源文件，可以看到其显示在主窗体文本框内；

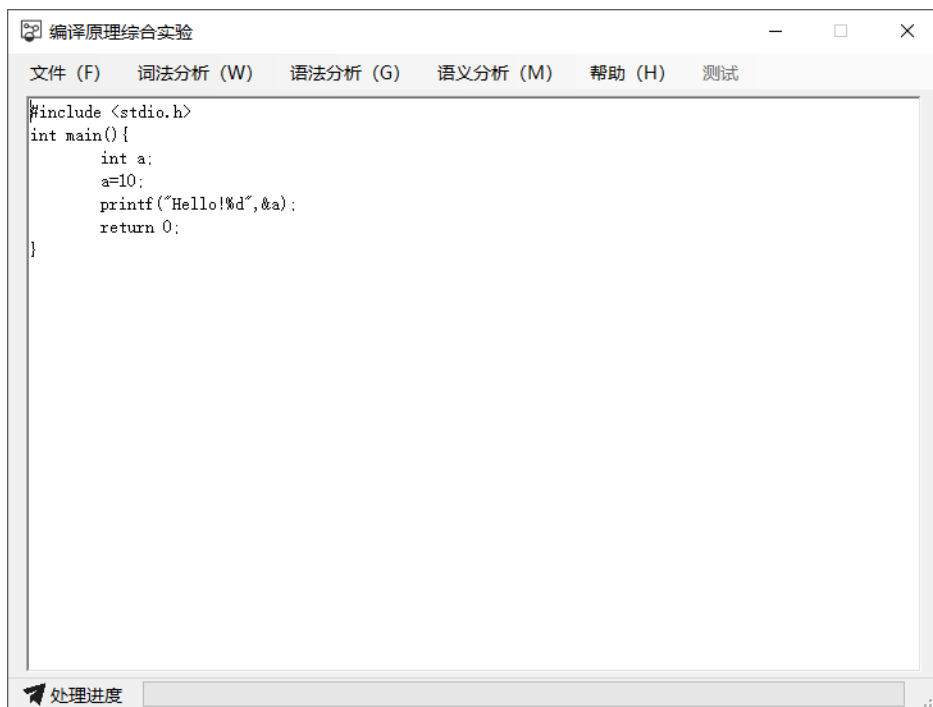


图 1 打开 c 语言源码程序

2.选择“词法分析”，进入词法分析子程序；

3.分别选择“符号表”、“常量表”、“整体分析”选项，可在下方文本框中得到所需结果；



图 2 符号表

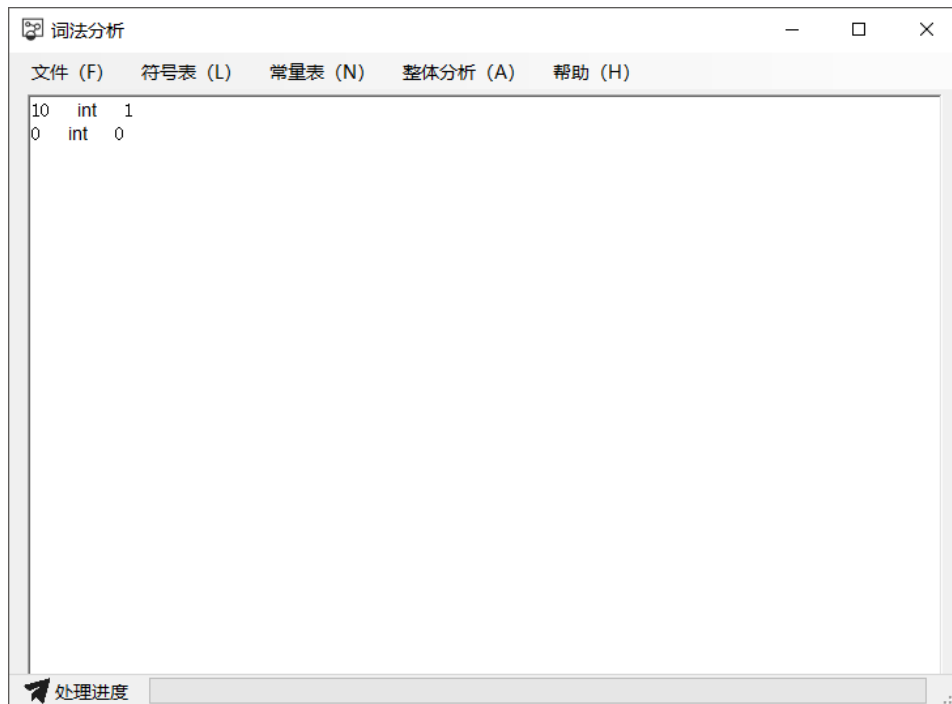


图 3 常量表

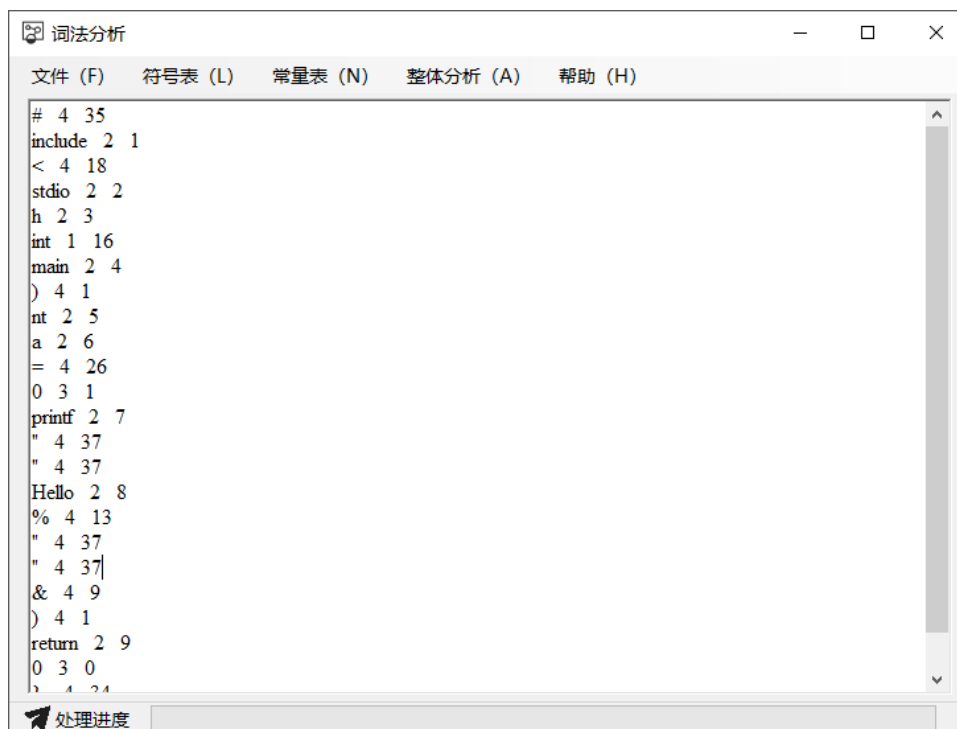



图 4 整体分析结果

4.选择“文件”-“导出”，将当前文本框内分析结果导出为txt文件。



```
符号表.txt - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
# 界符 35
< 界符 18
> 界符 19
( 界符 0
) 界符 1
{ 界符 33
= 运算符 26
( 界符 0
" 界符 37
" 界符 37
% 运算符 13
" 界符 37
" 界符 37
& 运算符 9
) 界符 1
} 界符 34
```

图 5 分析结果导出效果图

4.心得体会

通过本次实验，我加深了对词法分析器的理解，并通过对词法分析器进行实现，加深了对词法分析流程的理解。词法分析计算机科学中将字符序列转换为单词（Token）序列的过程。进行词法分析的程序或者函数叫作词法分析器，通常基于有限状态自动机实现。在实验过程中，我发现单个字符的元素是比较容易判断的，然而，对于关键字，标识符，数字这种就有点麻烦，包含多个字符的。我的思路是，因为多个字符的三种情况的一个元素里面是不可能含有分界符运算符的，所以我把分界符运算符和'#'统称为分隔符。我把遇到分隔符前的字符都收集起来组成字符串，再对其判断是否为数字（首字符为数字），再判断是否关键字，最后判断标识符。