上海大学计算机工程与科学学院

《编译原理》实验报告

实验一　识别标识符

姓名：林艺珺

学号：18120189

日期：2021 年 04 月 08 日

1 实验目的

- 根据 PL/0 语言的文法规范，编写 PL/0 语言的标识符识别程序。
- 通过设计调试标识符识别程序，实现从源程序中分出各个标识符的方法；加深对课堂教学的理解；为后序词法分析程序的实现打下基础。
- 掌握从源程序文件中读取有效字符的方法和产生源程序的内部表示文件的方法。
- 掌握识别标识符的实现方法。
- 上机调试完成的识别标识符程序的实现。

2 实验内容

输入 PL/0 语言源程序，输出源程序中所有标识符的出现次数。

3 实验要求

- 识别程序读入 PL/0 语言源程序 (文本文件)，识别结果也以文本文件保存。
- 按标识符出现的顺序输出结果，每个标识符一行，采用二元式序列，即：（标识符值，标识符出现次数）。
- 源程序中字符不区分大小写，即："a1"和"A1"是同一个标识符。
- 准备至少 5 组测试用例，每组测试用例包括：输入源程序文件和输出结果。
- 测试用例应该考虑各种合法标识符的组合情况。

4 设计思路

根据实验要求，源程序中字符不区分大小写，因此先对文件进行大写转换，即全部转换为大写。为方便实验二的内容，即区分每个单词的类别，因此将 PL/0 语言的词汇表全部转换成为编码形式。由于 C++ 的 Stringstream 的特性可直接根据空格分隔字符串流中的字符串，因此在替换的过程中，在编码前后各加一空格，方便处理。将 Stringstream 类型数据逐个读入一字符串数组，至此完成文本文件中源程序的分词。

分词完成后，根据词汇表，判断单词类型，将标识符类型的单词保存至一个动态数组中。由于要输出不同标识符出现的次数，因此，将数组进行排序（此处数据量少，直接使用冒泡）。排序结果的判断最后一个与前一个字符串相同的后一个数组下标减去上一个记录的数组下标，即是重复个数。最后根据格式要求输出。

5 代码

5.1 PL/0 语言的词汇表

根据 PL/0 的词汇规范，建立一个二维数组，将基本字、运算符及界符的单词与编码一一对应。由于测试样例中包含了 ELSE 的扩展，及自定义运算符 ®，因此将两者一并加入二维数组，分别对

应编码 ELSESYM 及 REGISTED。

在实际程序中，对于运算符来说，替换的顺序不可颠倒。包含两个字符的运算符必须先于一个字符的进行替换，否则将出现重复替换的错误。

```
string phases[31][2] = {
            {"BEGIN",       "BEGINSYM"},
            {"CALL",        "CALLSYM"},
            {"CONST",       "CONSTSYM"},
            {"DO",          "DOSYM"},
            {"END",         "ENDSYM"},
            {"IF",          "IFSYM"},
            {"ODD",         "ODDSYM"},
            {"PROCEDURE",   "PROCEDURESYM"},
            {"READ",        "READSYM"},
            {"THEN",        "THENSYM"},
            {"VAR",         "VARSYM"},
            {"WHILE",       "WHILESYM"},
            {"WRITE",       "WRITESYM"},
            {"ELSE",        "ELSESYM"},     // 增添ELSE

            {"<=",  "LEQ"},
            {">=",  "GEQ"},
            {":=",  "BECOMES"},

            {"+",   "PLUS"},
            {"-",   "MIUNS"},
            {"*",   "TIMES"},
            {"/",   "SLASH"},
            {"=",   "EQL"},
            {"#",   "NEQ"},
            {"<",   "LSS"},
            {">",   "GTR"},
            {"(",   "LPAREN"},
            {")",   "RPAREN"},
            {",",   "COMMA"},
            {";",   "SEMICOLON"},
            {".",   "PERIOD"},

            {"®",   "REGISTED"},           // 增添®
};
```

5.2 大小写统一

下列函数完成指定字符串的大写转换。

```
void toUppercase(string & srcStr) {
    for (int i = 0; i < srcStr.length(); i++)
        if (srcStr[i] >= 'a' && srcStr[i] <= 'z')
            srcStr[i] -= 32;
    return;
}
```

5.3 单词替换为编码

下列函数完成指定单词的编码转换。

```
void changePhases(string & srcStr, string phases[31][2]) {
    srcStr = srcStr.replace(srcStr.find('\r'), 1, "");
    for (int i = 0; i < 31; i++) {
        int pos = 0;
        string subStr = srcStr.substr(pos, srcStr.length());
        while (pos < srcStr.length() && pos > -1) {
            subStr = srcStr.substr(pos, srcStr.length());
            int subPos =  subStr.find(phases[i][0]);
            if (subPos < subStr.length() && subPos > -1) {
                srcStr = srcStr.replace(pos + subPos, phases[i][0].
                   length(), ' ' + phases[i][1] + ' ');
                pos += subPos + phases[i][1].length() + 2;
                continue;
            }
            break;
        }
    }
    return;
}
```

5.4 输出指定结果

根据实验一要求判断单词类型并根据格式要求输出。

```
string lab1(stringstream & changedFile, string phases[31][2]) {
    string res[1024] = {""};
    int i = 0;
    int cnt = 0;
    while (!changedFile.eof()) {
        string temp = "";
        changedFile >> temp;
        for (int k = 0; k < 31 ; k++) {
            if (temp == phases[k][1]) {
```

```
                break;
            } else if (k == 30 && temp != phases[k][0] && temp != ""
                && temp[0] != '0' && temp[0] != '1' && temp[0] != '2'
                && temp[0] != '3' && temp[0] != '4' && temp[0] != '5'
                && temp[0] != '6' && temp[0] != '7' && temp[0] != '8'
                && temp[0] != '9') {
                res[i] = temp;
                cnt++;
            }
        }
        i++;
    }
    string * resIN = new string[cnt];
    int p = 0;
    for (int j = 0; j < 1024; j++) {
        if (res[j] != "") {
            resIN[p] = res[j];
            p++;
        }
    }
    for (int bi = 0; bi < cnt; bi ++) {
        for (int bb = 0; bb < cnt - 1; bb ++) {
            string bbtemp;
            if (resIN[bb] > resIN[bb + 1]) {
                bbtemp = resIN[bb];
                resIN[bb] = resIN[bb + 1];
                resIN[bb + 1] = bbtemp;
            }
        }
    }
    stringstream fOut;
    int last = 0;
    for (int bb = 1; bb < cnt + 1; bb ++) {
        if (resIN[bb] != resIN[bb - 1]) {
            fOut << "(" << resIN[bb - 1] << ": " << bb - last << ")" <<
                endl;
            last = bb;
        }
    }
    cout << fOut.str();
    return fOut.str();
}
```

5.5 完整读写操作

从文本文件中读入，经过一系列处理后，最终将结果写入文本文件的全过程（包含了实验二的选择选项）。

```
int readCase(string path) {
    string phases[31][2] = {省略};
    ifstream caseFile;
    char data[256];
    string strFile;
    stringstream ssFile;
    caseFile.open(path);
    if (!caseFile.is_open()) {
        cout << "error" << endl;
    } else {
        while (getline(caseFile, strFile)) {
            if (strFile.empty() || strFile == "\r") continue;
            toUppercase(strFile);
            changePhases(strFile, phases);
            ssFile << strFile << endl;
        }
    }
    string input;
    cout << "请选择实验（1/2）：";
    cin >> input;
    if (input == "1") {
        string fo = lab1(ssFile, phases);
        string outpath = path.replace(path.find(".txt"), 4, "lab1out.
            txt");
        ofstream fileOut(outpath);
        fileOut << fo;
        fileOut.flush();
        fileOut.close();
    } else if (input == "2") {
        省略
    } else {
        cout << "请重新输入。" << endl;
        return 1;
    }
    caseFile.close();
    return 0;
}
```

5.6 主函数调用

```
int main() {
    string filePath;
    while (1) {
        cout << "请输入测试文件名: ";
        cin >> filePath;
        if (filePath == "exit") break;
        filePath = "../" + filePath;
        readCase(filePath);
    }
    return 0;
}
```

6 测试/运行结果

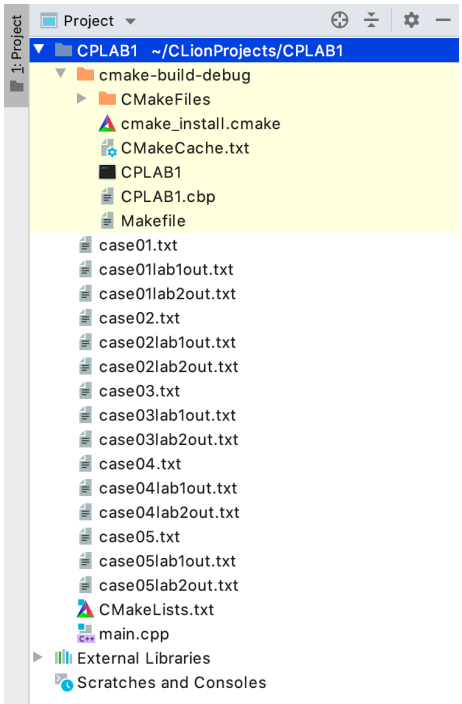

图 1: 运行结果目录截图

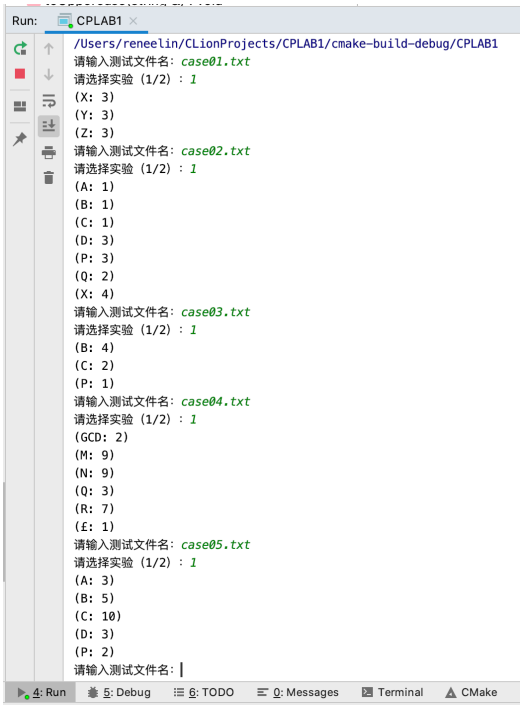

图 2: 运行结果命令行截图

7 实验体会

本次实验可以算是编译原理的初步体验，从根本层面上理解程序是如何被计算机阅读和理解的，让我们对于计算机的内部原理有了更深刻的理解，而不是简单的停留在使用者的身份上。作为计算机科学与技术专业的学生，对于计算机基本原理的掌握一定是必要和基础的，这是我们未来在实际生产应用中会优于其它专业学生理解的根本基础。很久没有这样使用 C++ 来编写程序了，不得不说遗忘得还是很厉害，要逐渐重新拾起。当然时隔这么久，也发现自己对语言的理解与之前不同了，也算是有一些进步。