编译原理实验报告

实验环境

类目	详情
操作系统	macOS Big Sur 11.2.3
CPU	Intel Core i5-7260U@2.3Ghz x2
IDE	CLion 2020.3.3 Build #CL-203.7717.62
Compiler	Apple clang version 11.0.0 (clang-1100.0.33.8)

实验要求

实验项目

以下为正则文法所描述的C语言子集单词符号的示例，请补充单词符号： ++，--， >>，<<，+= ，-= ，=，/= ， &&（逻辑与）， ||（逻辑或）， ！（逻辑非）等等，给出补充后描述C语言子集单词符号的正则文法，设计并实现其词法分析程序。

<*标识符*>→*字母* | <*标识符*>*字母* | <*标识符*>*数字*

<*无符号整数*>→*数字* | <*无符号整数*>*数字*

<*单字符分界符*> →+ | － | ； | , | (|) | { | }

<双字符分界符>→<大于>= | <小于>= | <小于>> | <感叹号>= | <等于>= | <斜竖>* <小于>→< <等于>→= <大于>→>

<斜竖> →/

<感叹号>→!

该语言的保留字： void、int、float、double、if、else、for、do、while 等等（也可补充）。

设计说明

- 可将该语言设计成大小写不敏感，也可设计成大小写敏感，用户定义的标识符最长不超过32个字符；
- 字母为a-z A-Z，数字为0-9；
- 可以对上述文法进行扩充和改造； (4) “/……/”和“//”(一行内)为程序的注释部分。

设计要求

- 给出各单词符号的类别编码；
- 词法分析程序应能发现输入串中的错误；
- 词法分析作为单独一遍编写，词法分析结果为二元式序列组成的中间文件； (4) 设计两个测试用例（尽可能完备），并给出测试结果。

实验内容

文件列表

文件	说明
main.cpp	程序入口
Classification.h / Classification.cpp	判断字符是否为字母、数字、空格（回车等）或分隔符的函数
Handler.h / Handler.cpp	错误处理或结尾处理
Output.h / Output.cpp	文件输出

文件功能介绍

Classification.cpp

Classification.cpp共包含了四个字符类型判断相关的函数，返回值均为bool型：

isAplhabet

函数isAlphabet通过输入字符与字母边界的ASCII码比较判断，其具体实现如下：

```
bool isAlphabet(char ch) {
    return ((ch >= 'a' && ch <= 'z') || (ch >= 'A' && ch <= 'Z'));
}
```

isNumber

函数isNumber通过输入字符与数字边界的ASCII码比较判断，其具体实现如下：

```
bool isNumber(char ch) {
    return (ch >= '0' && ch <= '9');
}
```

isWhiteSpace

函数isWhiteSpace通过直接匹配进行判断，其具体实现如下：

```
bool isWhiteSpace(char ch) {
    return (ch == ' ' || ch == '\n' || ch == '\t' || ch == '\r');
}
```

isSeparator

函数isSeparator通过直接匹配来判断分隔符，其具体实现如下：

```
bool isSeparator(char ch) {
    return (ch == ';' || ch == ',' || ch == '(' || ch == ')' || ch == '{' || ch == '}');
}
```

Handler.cpp

Handler.cpp包含了进行打印或错误处理的相关函数：

handleError

函数handleError是若程序判断到输入错误后，程序直接读取到下一分隔符并打印，其具体实现如下：

```
void handleError() {
    char ch;
    do {
        ifile >> ch;
    } while (!isWhiteSpace(ch));
    output("error");
}
```

handleEndOfWord

函数handleEndOfWord是若程序判断输入正确，读取到下一分隔符后打印，其具体实现如下：

```
void handleEndOfWord(string str) {
    char ch;
    ifile >> ch;
    // 正确结束
    if (isWhiteSpace(ch) || isSeparator(ch)){
        output(str);
    }
        // 错误结束
    else {
        handleError();
    };
    ifile.seekg(-1, ios::cur);  // 文件指针回退
}
```

Output.cpp

Output.cpp包含了一个STL中map的数据结构以及一个文件输出相关函数:

map

此map具体实现如下：

```
std::map<std::string, std::string> IDofWords;
```

map的key值为标识符，value对应为其值。

output

函数output包含两个参数

参数	说明
type	标识符类型
item	标识符的值

其具体实现如下：

```
void output(string type, string item = "") {
    if (type == "error") {
        ofile << "error" << endl;
        cout << "error" << endl;
    }
```

```
    else if (type == "integer") {
        ofile << "[" + IDofWords[type] + ", " + item + "]" << endl;
        cout << "[" + IDofWords[type] + ", " + item + "]" << endl;
    }
    else if (type == "identifier") {
        // 判断是否为保留字
        if (IDofWords.count(item) == 1) {
            ofile << "[" + IDofWords[item] + ", " + item + "]" << endl;
            cout << "[" + IDofWords[item] + ", " + item + "]" << endl;
        }
        else {
            ofile << "[" + IDofWords[type] + ", " + item + "]" << endl;
            cout << "[" + IDofWords[type] + ", " + item + "]" << endl;
        }
    }
    else {
        ofile << "[" + IDofWords[type] + ", " + type + "]" << endl;
        cout << "[" + IDofWords[type] + ", " + type + "]" << endl;
    }
}
```

main.cpp

在**main.cpp**中，首先声明了两个全局变量，分别为std::ifstream类型的ifile，用作文件读入以及std::ofstream类型的ofile用于输出文件写入。

随后进入主函数，在主函数中首先声明一个字符（character）型变量ch用于存放当前读入字符，同时将ifile通过 `ifile >> noskipws;` 设置为允许读空格。随后若文件不能打开，则直接退出程序：

```
if (!ifile.is_open()) {
    cout << "Failed to open file." << endl;
    return 0;
}
```

若文件正常打开，则一直读取知道文件结尾，即 `!ifile.eof()` 。随后进行分支判断：

```
while (!ifile.eof()) {
    ifile >> ch;
    string token(1, ch);  // 将当前字符装入字符串
    switch (ch) {
        // 纯单字符分界符
    case '{': case '}': case '(': case ')': case ',': case ';':
        output(token);
        break;
        // 单、双字符分解符 + - < >
    case '+': case '-': case '<': case '>':
        ifile >> ch;
        if (isWhiteSpace(ch)) {
            output(token);
        }
        else if (ch == '=' || ch == token[0] || (token[0] == '<' && ch == '>')) {
            token = token.append(1, ch);
            handleEndOfWord(token);
        }
        else {
            handleError();
        }
        break;
        // 单、双字符分解符 * / ! =
    case '*': case '/': case '!': case '=':
```

```
            ifile >> ch;
            if (isWhiteSpace(ch)) {
                output(token);
            }
            else if (token[0] == '/' && ch == '/') {
                char temp[255];
                ifile.getline(temp, 255);
            }
            else if (token[0] == '/' && ch == '*') {
                bool isWellEnded = false;  // 注释是否正确结束
                while (!ifile.eof()) {
                    ifile >> ch;
                    if (ch == '*') {
                        ifile >> ch;
                        if (ch == '/') {
                            isWellEnded = true;
                            break;
                        }
                        else {
                            ifile.seekg(-1, ios::cur);
                        }
                    }
                }
                if (!isWellEnded) output("error");
            }
            else if (ch == '=') {
                token = token.append(1, ch);
                handleEndOfWord(token);
            }
            else {
                handleError();
            }
            break;
            // 单、双字符分解符 & |
        case '&': case '|':
            ifile >> ch;
            if (isWhiteSpace(ch)) {
                output(token);
            }
            else if (ch == token[0]) {
                token = token.append(1, ch);
                handleEndOfWord(token);
            }
            else {
                handleError();
            }
            break;
            // 空
            case ' ': case '\n': case '\t': case '\r':
            break;
        default:
            // 整数
            if (isNumber(ch)) {
                token = "";
                do {
                    token += ch;
                    ifile >> ch;
                } while (isNumber(ch));
                // 正确结束
                if (isWhiteSpace(ch) || isSeparator(ch)) {
                    output("integer", token);
                }
```

```
                // 错误结束
            else {
                do {
                    ifile >> ch;
                } while (!isWhiteSpace(ch));
                output("error");
            };
            ifile.seekg(-1, ios::cur);
        }
            // 标识符
        else if (isAlphabet(ch)) {
            token = "";
            int length = 0;
            do {
                token += ch;
                length++;
                ifile >> ch;
            } while (isAlphabet(ch) || isNumber(ch));
            // 正确结束
            if ((isWhiteSpace(ch) || isSeparator(ch)) && length <= 32) {
                output("identifier", token);
            }
                // 错误结束
            else {
                do {
                    ifile >> ch;
                } while (!isWhiteSpace(ch));
                output("error");
            };
            ifile.seekg(-1, ios::cur);
        }
            // 啥都不是
        else {
            output("error");
        }
    }
}
```

最后关闭文件，退出程序：

```
ifile.close();
ofile.close();
```