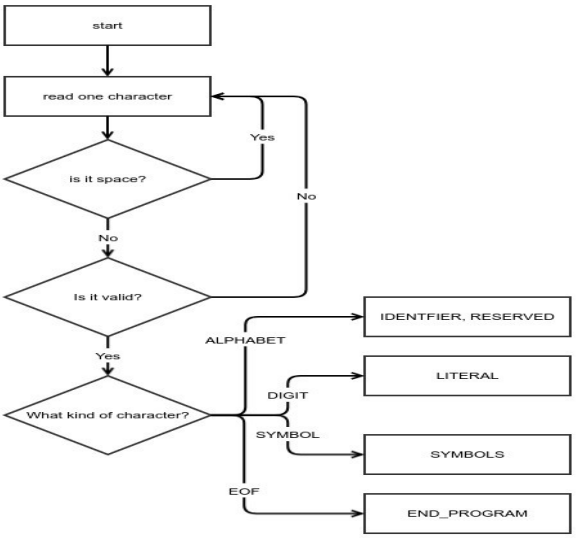
3年15組９２番　全　海成(JEON HAESUNG)
コンピュータサイエンス実習A　ソフトウェア　一回目　レポート

実習の目的
ソフトウェアコース全体の目的はC0と言うC言語を簡単なバージョンのコンパイラーを作ることです。一回目の実習の目的ははそのコンパイラーの基礎となる字句解析を行うプログラムをJAVAで作成する事です。プログラムの入力はC0で書いてあるソースコードでコードをテキストとして読み込み字句解析を行いコードの一番左上からどういう字句があるかを順番に出力します。一番の目的である字句解析をする関数はget_tokenです。

考え方の説明
C0の言語の字句には四種類があって識別子、予約語、数、記号の四つです。コードの最初から一文字づつ読む方式を基本とし度の種類の字句なのかを判断します。
まず一文字づつ読みながら空白のところを無視します。空白じゃない文字にあったらそれがC0で許容する有効か判断します。有効ではないと次の文字に進みます。有効だった場合どんな字句なのかを判断します。数の判断基準は一文字目が数だと数です。記号は‘/’以外は一文字目で記号だと判断できます。識別子と予約語は一文字目がアルファベットの場合です。この流れを図で表せたのが下の図です。




```
enum token {
    END_PROGRAM,
    IDENTIFIER, LITERAL,
    ELSE, IF, WHILE,
    COMMA, SEMICOLON,
    LEFT_BRACE, RIGHT_BRACE, LEFT_PAREN, RIGHT_PAREN,
    EQUAL, OROR, ANDAND, OR, AND,
    EQEQ, NOTEQ, LE, LT, GE, GT,
    PLUS, MINUS, STAR, SLASH, PERCENT,
    ERROR
}
```

実装の話
コードの話をすると字句はenum token型で扱います。上の図はget_token()の流れで一文字づつ読むのはnext_ch()関数で行われます。全体プログラムの流れはメインでファイルを開きファイルが終わるまでget_token()を呼びます。そして字句の種類と内容を字句ごとに出力します。
識別子と予約語を認識するアルゴリズムです。
1. 一文字目がアルファベットか‘_’である場合識別子や予約語が始まる文字だと判断してバッハー(id_string)に保存します。
2. 有効文字以外が出るまでバッハーに保存します。
3. 有効文字以外を検出したらそこで一つの字句が終わったと判断し次の文字を読むことを止めます。

4. バッハーの内容が予約語なのか確認したあと予約語だった場合予約語だと告知し終了、違った場合は識別子だと告知し内容も告知して終了します。
下は実際のコードです。

```
// identify IDENTIFIER
else if(Character.isLetter(ch) || ch == '_') {
    id_string = "";
    while(Character.isLetter(ch) || Character.isDigit(ch) || ch == '_') {
        id_string += ch;
        next_ch();
    }
    if(id_string.equals("else")) {
        sy = token.ELSE;
        return ;
    }
    else if(id_string.equals("if")) {
        sy = token.IF;
        return ;
    }
    else if(id_string.equals("while")) {
        sy = token.WHILE;
        return ;
    }
    sy = token.IDENTIFIER;
    return ;
}
```

Week1.java:211~232

次は数の認識するアルゴリズムです。ここで重要なことは C0 の数は int しかありません。そのためオーバーフローが起こる可能性があります。

1. 一文字目が数だと数だと判断して足し合わせる long 型変数(v)を 0 に初期化する。
2. 一文字づつ読みながら数字以外の文字が出るまで v = v * 10 + (読んだ数）をします。この時足し合わせた値が int の範囲外になったらオーバーフローが起きたのですぐ止めます。
3. オーバーフローじゃないなら字句が数だと告知し値も告知します。オーバーフローだったらエラーが起きたと告知して数字以外の文字が出るまで文字を読み込み一つの字句を終えて関数を終了します。

以下は実際のコードです。

```
// identify LITERAL, if there's character which is not digit, need to display error message and terminates
else if(Character.isDigit(ch)) {
    long v = 0;
    boolean flag = false;
    while(Character.isDigit(ch)) {
        v = v * 10 + (long)(ch - '0');
        if(v > Integer.MAX_VALUE) {
            flag = true;
            break;
        }
        next_ch();
    }
    if(flag) {
        // TODO : Overflow occured, need to handle. read all following digits
        error("too large integer literal");
        while(Character.isDigit(ch)) {
            next_ch();
        }
        sy = token.LITERAL;
        literal_value = 0;
        return ;
    }
    literal_value = (int) v;
    sy = token.LITERAL;
    return ;
}
```

Week1.java:185~210

次は記号を認識するアルゴリズムですが記号は一文字や二文字のものが多い上に一文字読めば何が何なのか分かるようになっていてほぼ if-else の繰り返しです。なので少し難しい二つの記号(!=,/)の認識だけ説明します。
まず!=です。一文字目が!なのにすぐ後ろに=がないと C0 では有効な文字ではないためエラーを出します。少し特殊なエラーであるためこのエラーは!の後ろに=がないと知らせます。下は実際のコードです。

```
else if(ch == '!') {
    next_ch();
    if(ch == '=') {
        next_ch();
        sy = token.NOTEQ;
        return ;
    }
    // TODO : Need handle error(exclamation mark not followed by equal
    error("exclamation mark not followed by equal");
    sy = token.ERROR;
    return ;
}
```

Week1.java:95~106

/何ですがこの記号を認識するに当たって難しいところはコメントがこの記号で始まるってことです。幸いにも C0 では一行コメントはないので/のすぐ後ろに*がついているのかだけ見ればこれがコメントの始まりなのかそれともただの記号なのかを知ることができます。だからすぐ後ろに*がなければ記号だと判断して終わります。

コメントだった場合の処理です。コメントだと判断するには/*の二文字がすでに読まれたということなので*と/が順番で連続して出るかを確認するだけです。だから*と/が連続で出るまで一文字づつ読みます。出てない場合はファイルの最後まで読むので EOF にあったらコメントが終わってないというエラーを出してプログラムを終了します。ちゃんとコメントの終わりを見つけたらコメントが終わったところから字句を探すように get_token を呼んでちゃんと字句を探すようにします。下は実際のコードです。

```
else if(ch == '/') {
    next_ch();
    if(ch != '*') {
        sy = token.SLASH;
        return ;
    }
    // TODO : need to skip comments
    // if comment does not terminate, display error message and terminate program.
    // NOTICE: There's no one line comment
    next_ch();
    while(ch != 65535) {
        if(ch == '*') {
            next_ch();
            if (ch == '/') {        // comment terminated
                next_ch();
                get_token();
                return ;
            }
        }
        else {
            next_ch();
        }
    }
    line_number--;
    error("comment not terminated");
    sy = token.END_PROGRAM;
    return ;
}
```

Week1.java:157~184

考察
今回実装した内容を書くとき注意すべきだったところはコメントを吹き飛ばした後ちゃんと次の字句を探すようにしておくのとオーバーフローだと判断した数の字句を最後まで呼んで一つの字句としての処理を終えておくことです。get_token 関数はメインで見たら一回の呼び出しで一つの字句を返すことを保証してくれる必要があるからです。このことが保証されてないと使う方法が簡単じゃなくなるからです。