Práctica 2 - Flex ++

Alumnos: Raúl Rodríguez Pérez
Santiago Muñoz Castro
Raúl Castro Moreno
Grupo: A2

1. Problema abordado

Para la realización de esta práctica nos hemos planteado el siguiente problema; la creación de un programa que obtenga toda información relevante respecto a las operaciones del código de un fichero escrito en C++. Es decir, que obtenga el número de apariciones de una gran variedad de operaciones/funciones como pueden ser: sumas, restas, multiplicaciones, llamadas a funciones, condicionales, bucles…

2. Solución del problema

Tal y como nos pide la práctica, hemos abordado la solución del problema, en primer lugar, creando los macros de nuestro problema, es decir, los oportunos emparejamientos de cadenas con expresiones regulares.

- Declaración de los macros de nuestro problema:

```
%}
/*Aqui definiremos las macros para definir todas las operaciones que vamos a contabilizar*/
SUMA    "+"|"++"|"+="
RESTA   "-"|"--"|"-="
MULTIPLICACION  "*"|"*="
DIVISION    "/"|"/="
CONDICIONAL "if"|"switch"
BUCLE   "for"|"while"
COMPARACION "<"|">"|"=="|"!="|"<="|">="

/*Tambien tenemos macros definidas para las posibles variables del codigo, los tipos para los cast y los cierres de parentesis y corchetes*/
VARIABLE   [A-Za-z0-9"_"."]
TIPO   "float"|"int"|"double"|"char"|"bool"|"string"|"long"
CIERRES     ")"|"]"

%x comment


%%
```

Tras la creación de los macros, hemos pasado a la sección de reglas en donde definimos la asociación de acciones a cada emparejamiento creado en el apartado anterior. Como podemos observar, dichas reglas hacen uso tanto de los macros que hemos creado como de los signos propios de la herramienta flex.

- La sección de reglas declarada para nuestro problema:

```
%%
    /*-Seccion de Reglas-*/

"//"(.*)                        /*---En estas 2 líneas quitamos los comentarios tanto con "//" como con "/*"---*/
"/*"    BEGIN(comment);         /*---Para quitar los comentarios que empiezan con '/*' hemos tenido que usar start conditions---*/
<comment>[^*]*
<comment>"*"+[^*/]*
<comment>"*"+"/"    BEGIN(INITIAL);

\".*\"                      /*Con esta regla hemos omitido todos las frases entre comillas del codigo, como pueden ser rutas o frases para la salida por ter

{VARIABLE}+{CIERRES}*" "*{SUMA}" "*{VARIABLE}+|{VARIABLE}+{SUMA}         {operaciones[0]++;}    /*Con esta regla omitimos todas las sumas definidas en el ma
{VARIABLE}+{CIERRES}*" "*{RESTA}" "*{VARIABLE}+|{VARIABLE}+{RESTA}[^">"]    {operaciones[1]++;} /*Con esta regla omitimos todas las restas definidas en el m
{VARIABLE}+{CIERRES}*" "*{MULTIPLICACION}" "*[-]*{VARIABLE}+     {operaciones[2]++;}     /*Con esta regla omitimos todas las multiplicacion definidas en el
{VARIABLE}+{CIERRES}*" "*{DIVISION}" "*[-]*{VARIABLE}+    {operaciones[3]++;} /*Con esta regla omitimos todas las divisiones definidas en el macro DIVISION
{VARIABLE}+{CIERRES}*" "*"%"" "*{VARIABLE}+      {operaciones[4]++;}    /*Con esta regla omitimos todos los modulos  definidos como % y sumamos en uno al co
{VARIABLE}+{CIERRES}*" "*{COMPARACION}" "*[-]*{VARIABLE}+    {operaciones[5]++;} /*Con esta regla omitimos todas las comparaciones  definidas en el macro CO
{CONDICIONAL}" "*"("   {operaciones[6]++;}  /*Con esta regla omitimos todas las condiciones definidas en el macro CONDICIONAL y sumamos en uno al contador d
{BUCLE}" "*"("    {operaciones[7]++;}   /*Con esta regla omitimos todos bucles definidos en el macro BUCLE y sumamos en uno al contador de bucles*/
"::"{VARIABLE}+"("  /*Con esta regla omitimos todas las declaraciones de metodos*/
{TIPO}" "*"("   /*Con esta regla omitimos todos los tipos de datos definidos en el macro TIPO, para evitar problemas con casteos*/
("."|"->")*{VARIABLE}+"("    {operaciones[8]++;} /*Con esta regla omitimos todas las llamadas a funciones, recononiendolas al encontrar un . o -> seguido de

[^ \n\t]   /*Con estas dos reglas omitiremos todo lo restante que haya quedado en el codigo*/
[ \n\t]


%%
```

En esta sección se puede apreciar, que la segunda regla utiliza un recurso de **flex++** llamado “Start Conditions” del cual hemos sacado la información para poder usarlo de estas 2 paginas web:

-https://stackoverflow.com/questions/25960801/f-lex-how-do-i-match-negation

-http://westes.github.io/flex/manual/Start-Conditions.html

Finalmente tenemos la sección de Procedimientos del Usuario, en donde pondremos a usar todo lo creado anteriormente sobre el flujo del fichero que abramos, el cual queremos analizar

-La sección de procedimiento de usuario declarada para nuestro problema:

```
    /*-Seccion Procedimientos del Usuario-*/

/*En el main, primero tendremos una seccion de abrir el archivo para ver si falla o no.
    Una vez que este correcto, inicializamos el vector de operaciones a 0
    Y ya despues utilizamos yyFlexLexer para el fichero, usamos en ese flujo la funcion yylex() para realizar el proceso y terminamos escribiendo los datos, pasandole el vector*/
int main(int argc, char *argv[])
{

    if (argc == 2){
        fichero.open(argv[1]);
        if (!fichero.is_open()){
            cout<<"Error de lectura"<<endl;
            exit(1);
        }
    }
    else{
        exit(1);
    }

    operaciones.resize(9);
    for(int i=0; i<operaciones.size();i++){
        operaciones[i] = 0;
    }

    yyFlexLexer flujo(&fichero, 0);
    flujo.yylex();
    escribir_datos(operaciones);
    return 0;
}

/*Funcion para mostrar los datos recogidos en el terminal */
void escribir_datos(vector<int> datos){

    cout << "Numero de sumas: " << datos[0] << endl;
    cout << "Numero de restas: " << datos[1] << endl;
    cout << "Numero de multiplicaciones: " << datos[2] << endl;
    cout << "Numero de divisiones: " << datos[3] << endl;
    cout << "Numero de modulos: " << datos[4] << endl;
    cout << "Numero de comparaciones: " << datos[5] << endl;
    cout << "Numero de condicionales: " << datos[6] << endl;
    cout << "Numero de bucles: " << datos[7] << endl;
    cout << "Numero de llamadas a funciones: " << datos[8] << endl;
}
```

A continuación mostramos un ejemplo de ejecución de nuestro código

-Muestra del funcionamiento de nuestro analizador de código de C++

```
santiago@santiago-OMEN-by-HP-Laptop-15-dc0xxx:~/Escritorio/MC$ flex++ analizador.l
santiago@santiago-OMEN-by-HP-Laptop-15-dc0xxx:~/Escritorio/MC$ g++ lex.yy.cc -o prog
santiago@santiago-OMEN-by-HP-Laptop-15-dc0xxx:~/Escritorio/MC$ ./prog escena.cc
Numero de sumas: 5
Numero de restas: 4
Numero de multiplicaciones: 7
Numero de divisiones: 10
Numero de modulos: 0
Numero de comparaciones: 17
Numero de condicionales: 50
Numero de bucles: 1
Numero de llamadas a funciones: 116
```