Analizador Léxico de C creado con Flex

Instituto Tecnológico de Costa Rica

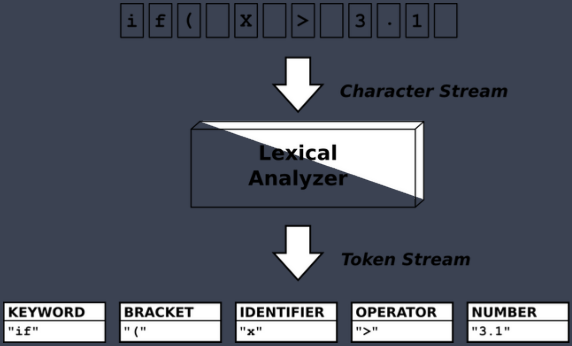
Compiladores e Intérpretes
Semestre II 2020

by Ronald Herrera Gámez

on November 25, 2020

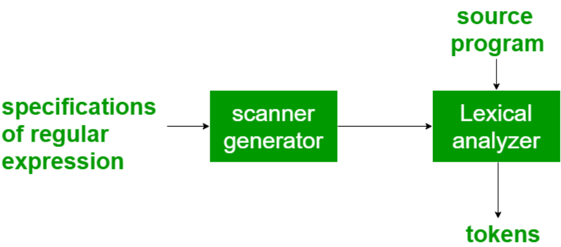
» Proceso de Scanning

Consiste en determinar las diferentes unidades elementales de un programa fuente, es decir, identifica los distintos lexemas de un lenguaje. Para este proceso, el scanner busca patrones dentro del fuente que cumplan con instrucciones, operadores, identificadores, constantes, entre otros, del lenguaje que se escanea.



» Flex (Fast Lexical Analyzer)

Flex es una herramienta para generar analizadores léxicos basados en la teoría de autómatas finitos. Para ello, se crea la descripción del escáner en forma de pares de expresiones regulares y código C, llamados reglas. Flex genera un archivo fuente en C llamado "lex.yy.c" que se puede compilar y vincular para producir un ejecutable. Este ejecutable analiza su entrada en busca de ocurrencias de texto que coinciden con las expresiones regulares para cada regla y siempre que encuentra una coincidencia, ejecuta el código C correspondiente.



» Código Después Del Preproceso

TOKENS:

KEYWORD
IDENTIFIER
LITERAL
OPERATOR
PUNTUACTOR
COMMENT
LEXICALERROR
PREPROCESSOR

```
#ifndef _STDIO_H #define _STDIO_H 1 #define __GLIBC_INTERNAL_STA
#include <bits/libc-header-start.h> __BEGIN_DECLS #define __need_size_t
#define __need_NULL #include <stddef.h> #define __need___va_list
#include <stdarg.h> #include <bits/types.h> #include <bits/types/__fpos_t.
#include <bits/types/__fpos64_t.h> #include <bits/types/__FILE.h>
#include <bits/types/FILE.h> #include <bits/types/struct_FILE.h> #ifdef
__USE_GNU # include <bits/types/cookie_io_functions_t.h> #endif
#if defined __USE_XOPEN || defined __USE_XOPEN2K8 # ifdef __GNUC__
# ifndef _VA_LIST_DEFINED typedef __gnuc_va_list va_list ; # define
_VA_LIST_DEFINED # endif # else # include <stdarg.h> # endif
#endif #if defined __USE_UNIX98 || defined __USE_XOPEN2K
# ifndef __off_t_defined # ifndef __USE_FILE_OFFSET64 typedef
__off_t off_t ; # else typedef __off64_t off_t ; # endif # define
__off_t_defined # endif # if defined __USE_LARGEFILE64 && !defined
__off64_t_defined typedef __off64_t off64_t ; # define __off64_t_define
# endif #endif #ifdef __USE_XOPEN2K8 # ifndef __ssize_t_defined
typedef __ssize_t ssize_t ; # define __ssize_t_defined # endif #endif
#ifndef __USE_FILE_OFFSET64 typedef __fpos_t fpos_t ; #else
typedef __fpos64_t fpos_t ; #endif #ifdef __USE_LARGEFILE64 type-
```

def __fpos64_t fpos64_t ; #endif #define _IOFBF 0 #define _IOLBF 1 #define _IONBF 2 #define BUFSIZ 8192 #define EOF (-1) #define SEEK_SET 0 #define SEEK_CUR 1 #define SEEK_END 2 #ifdef __USE_GNU # define SEEK_DATA 3 # define SEEK_HOLE 4 #endif #if defined __USE_MISC || defined __USE_XOPEN # define P_tmpdir "/tmp" #endif #include <bits/stdio_lim.h> extern FILE * stdin ; extern FILE * stdout ; extern FILE * stderr ; #define stdin stdin #define stdout stdout #define stderr stderr extern int remove (const char * __filename) __THROW ; extern int rename (const char * __old , const char * __new) __THROW ; #ifdef __USE_ATFILE extern int renameat (int __oldfd , const char * __old , int __newfd , const char * __new) __THROW ; #endif #ifdef __USE_GNU # define RENAME_NOREPLACE (1 « 0) # define RENAME_EXCHANGE (1 « 1) # define RENAME_WHITEOUT (1 « 2) extern int renameat2 (int __oldfd , const char * __old , int __newfd , const char * __new , unsigned int __flags) __THROW ; #endif #ifndef __USE_FILE_OFFSET64 extern FILE * tmpfile (void) __wur ; #else # ifdef __REDIRECT extern FILE * __REDIRECT (tmpfile , (void) , tmpfile64) __wur ; # else # define tmpfile tmpfile64 # endif #endif #ifdef __USE_LARGEFILE64 extern FILE * tmpfile64 (void) __wur ; #endif extern char * tmpnam (char * __s) __THROW __wur ; #ifdef __USE_MISC extern char * tmpnam_r (char * __s) __THROW __wur ; #endif #if defined __USE_MISC || defined __USE_XOPEN extern char * tempnam (const char * __dir , const char * __pfx) __THROW __attribute_malloc__ __wur ; #endif extern int fclose (FILE * __stream) ; extern int fflush (FILE * __stream) ; #ifdef __USE_MISC extern int fflush_unlocked (FILE * __stream) ; #endif #ifdef __USE_GNU extern int fcloseall (void) ; #endif #ifndef __USE_FILE_OFFSET64 extern FILE * fopen (const char * __restrict __filename , const char * __restrict __modes) __wur ; extern FILE * freopen (const char * __restrict __filename , const char * __restrict __modes , FILE * __restrict __stream) __wur ; #else # ifdef __REDIRECT extern FILE * __REDIRECT (fopen , (const char * __restrict __filename , const char * __restrict __modes) , fopen64) __wur ; extern FILE * __REDIRECT (freopen , (const char * __restrict __filename , const char * __restrict __modes , FILE * __restrict __stream) , freopen64) __wur ; # else # define fopen fopen64 # define freopen freopen64 # endif #endif #ifdef __USE_LARGEFILE64 extern FILE * fopen64 (const char * __restrict

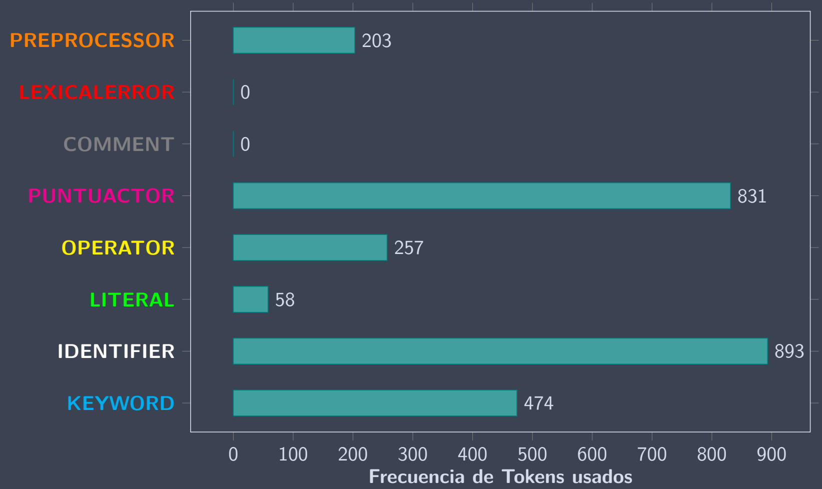
```
__filename , const char * __restrict __modes ) __wur ; extern FILE
* freopen64 ( const char * __restrict __filename , const char * __restrict
__modes , FILE * __restrict __stream ) __wur ; #endif #ifdef
__USE_POSIX extern FILE * fdopen ( int __fd , const char * __modes
) __THROW __wur ; #endif #ifdef __USE_GNU extern FILE * fopencookie
( void * __restrict __magic_cookie , const char * __restrict
__modes , cookie_io_functions_t __io_funcs ) __THROW __wur ;
#endif #if defined __USE_XOPEN2K8 || __GLIBC_USE (LIB_EXT2)
extern FILE * fmemopen ( void * __s , size_t __len , const char *
__modes ) __THROW __wur ; extern FILE * open_memstream ( char
* * __bufloc , size_t * __sizeloc ) __THROW __wur ; #endif extern
void setbuf ( FILE * __restrict __stream , char * __restrict __buf
) __THROW ; extern int setvbuf ( FILE * __restrict __stream , char
* __restrict __buf , int __modes , size_t __n ) __THROW ; #ifdef
__USE_MISC extern void setbuffer ( FILE * __restrict __stream , char
* __restrict __buf , size_t __size ) __THROW ; extern void setlinebuf (
FILE * __stream ) __THROW ; #endif extern int fprintf ( FILE * __restrict
__stream , const char * __restrict __format , ... ) ; extern int
printf ( const char * __restrict __format , ... ) ; extern int sprintf ( char *
__restrict __s , const char * __restrict __format , ... ) __THROWNL ;
extern int vfprintf ( FILE * __restrict __s , const char * __restrict __format
, __gnuc_va_list __arg ) ; extern int vprintf ( const char * __restrict
__format , __gnuc_va_list __arg ) ; extern int vsprintf ( char * __restrict
__s , const char * __restrict __format , __gnuc_va_list __arg )
__THROWNL ; #if defined __USE_ISOC99 || defined __USE_UNIX98
extern int snprintf ( char * __restrict __s , size_t __maxlen , const
char * __restrict __format , ... ) __THROWNL __attribute__ ( (
__format__ ( __printf__ , 3 , 4 ) ) ) ; extern int vsnprintf ( char *
__restrict __s , size_t __maxlen , const char * __restrict __format ,
__gnuc_va_list __arg ) __THROWNL __attribute__ ( ( __format__
( __printf__ , 3 , 0 ) ) ) ; #endif #if __GLIBC_USE (LIB_EXT2) extern
int vasprintf ( char * * __restrict __ptr , const char * __restrict
__f , __gnuc_va_list __arg ) __THROWNL __attribute__ ( ( __format__
( __printf__ , 2 , 0 ) ) ) __wur ; extern int __asprintf ( char *
* __restrict __ptr , const char * __restrict __fmt , ... ) __THROWNL
__attribute__ ( ( __format__ ( __printf__ , 2 , 3 ) ) ) __wur ; extern
int asprintf ( char * * __restrict __ptr , const char * __restrict __fmt
, ... ) __THROWNL __attribute__ ( ( __format__ ( __printf__ , 2
, 3 ) ) ) __wur ; #endif #ifdef __USE_XOPEN2K8 extern int vdprintf
( int __fd , const char * __restrict __fmt , __gnuc_va_list __arg )
__attribute__ ( ( __format__ ( __printf__ , 2 , 0 ) ) ) ; extern int
dprintf ( int __fd , const char * __restrict __fmt , ... ) __attribute__
( ( __format__ ( __printf__ , 2 , 3 ) ) ) ; #endif extern int fscanf
( FILE * __restrict __stream , const char * __restrict __format , ...
) __wur ; extern int scanf ( const char * __restrict __format , ... )
__wur ; extern int sscanf ( const char * __restrict __s , const char *
__restrict __format , ... ) __THROW ; #if !__GLIBC_USE (DEPRECATED_SCANF)
&& !defined __LDBL_COMPAT # ifdef __REDIRECT
extern int __REDIRECT ( fscanf , ( FILE * __restrict __stream
, const char * __restrict __format , ... ) , __isoc99_fscanf ) __wur ;
extern int __REDIRECT ( scanf , ( const char * __restrict __format , ...
) , __isoc99_scanf ) __wur ; extern int __REDIRECT_NTH ( sscanf ,
( const char * __restrict __s , const char * __restrict __format , ... ) ,
__isoc99_sscanf ) ; # else extern int __isoc99_fscanf ( FILE * __restrict
__stream , const char * __restrict __format , ... ) __wur ; extern int
__isoc99_scanf ( const char * __restrict __format , ... ) __wur ; extern
int __isoc99_sscanf ( const char * __restrict __s , const char * __restrict
__format , ... ) __THROW ; # define fscanf __isoc99_fscanf # define
scanf __isoc99_scanf # define sscanf __isoc99_sscanf # endif #endif
#ifdef __USE_ISOC99 extern int vfscanf ( FILE * __restrict __s , const
char * __restrict __format , __gnuc_va_list __arg ) __attribute__
( ( __format__ ( __scanf__ , 2 , 0 ) ) ) __wur ; extern int vscanf
( const char * __restrict __format , __gnuc_va_list __arg ) __attribute__
( ( __format__ ( __scanf__ , 1 , 0 ) ) ) __wur ; extern int
vsscanf ( const char * __restrict __s , const char * __restrict __format
, __gnuc_va_list __arg ) __THROW __attribute__ ( ( __format__ (
__scanf__ , 2 , 0 ) ) ) ; # if !__GLIBC_USE (DEPRECATED_SCANF)
# if defined __REDIRECT && !defined __LDBL_COMPAT extern int
__REDIRECT ( vfscanf , ( FILE * __restrict __s , const char * __restrict
__format , __gnuc_va_list __arg ) , __isoc99_vfscanf ) __attribute__
( ( __format__ ( __scanf__ , 2 , 0 ) ) ) __wur ; extern
int __REDIRECT ( vscanf , ( const char * __restrict __format ,
__gnuc_va_list __arg ) , __isoc99_vscanf ) __attribute__ ( ( __format__
( __scanf__ , 1 , 0 ) ) ) __wur ; extern int __REDIRECT_NTH (
vsscanf , ( const char * __restrict __s , const char * __restrict __format
, __gnuc_va_list __arg ) , __isoc99_vsscanf ) __attribute__ ( ( __for-
mat__ ( __scanf__ , 2 , 0 ) ) ) ; # elif !defined __REDIRECT extern
int __isoc99_vfscanf ( FILE * __restrict __s , const char * __restrict
__format , __gnuc_va_list __arg ) __wur ; extern int __isoc99_vscanf
( const char * __restrict __format , __gnuc_va_list __arg ) __wur ;
extern int __isoc99_vsscanf ( const char * __restrict __s , const char
* __restrict __format , __gnuc_va_list __arg ) __THROW ; # define
vfscanf __isoc99_vfscanf # define vscanf __isoc99_vscanf # define vsscanf
__isoc99_vsscanf # endif # endif #endif extern int fgetc ( FILE *
__stream ) ; extern int getc ( FILE * __stream ) ; extern int getchar (
void ) ; #ifdef __USE_POSIX199506 extern int getc_unlocked ( FILE
* __stream ) ; extern int getchar_unlocked ( void ) ; #endif #ifdef
__USE_MISC extern int fgetc_unlocked ( FILE * __stream ) ; #endif
extern int fputc ( int __c , FILE * __stream ) ; extern int putc (
int __c , FILE * __stream ) ; extern int putchar ( int __c ) ; #ifdef
__USE_MISC extern int fputc_unlocked ( int __c , FILE * __stream
) ; #endif #ifdef __USE_POSIX199506 extern int putc_unlocked ( int
__c , FILE * __stream ) ; extern int putchar_unlocked ( int __c ) ;
#endif #if defined __USE_MISC \ || ( defined __USE_XOPEN && !
defined __USE_XOPEN2K ) extern int getw ( FILE * __stream ) ; extern
```

```
int putw ( int __w , FILE * __stream ) ; #endif extern char * fgets (
char * __restrict __s , int __n , FILE * __restrict __stream ) __wur ;
#if __GLIBC_USE (DEPRECATED_GETS) extern char * gets ( char *
__s ) __wur __attribute_deprecated__ ; #endif #ifdef __USE_GNU
extern char * fgets_unlocked ( char * __restrict __s , int __n , FILE *
__restrict __stream ) __wur ; #endif #if defined __USE_XOPEN2K8
|| __GLIBC_USE (LIB_EXT2) extern __ssize_t __getdelim ( char * *
__restrict __lineptr , size_t * __restrict __n , int __delimiter , FILE
* __restrict __stream ) __wur ; extern __ssize_t getdelim ( char * *
__restrict __lineptr , size_t * __restrict __n , int __delimiter , FILE *
__restrict __stream ) __wur ; extern __ssize_t getline ( char * * __restrict
__lineptr , size_t * __restrict __n , FILE * __restrict __stream
) __wur ; #endif extern int fputs ( const char * __restrict __s , FILE *
__restrict __stream ) ; extern int puts ( const char * __s ) ; extern int
ungetc ( int __c , FILE * __stream ) ; extern size_t fread ( void * __restrict
__ptr , size_t __size , size_t __n , FILE * __restrict __stream
) __wur ; extern size_t fwrite ( const void * __restrict __ptr , size_t
__size , size_t __n , FILE * __restrict __s ) ; #ifdef __USE_GNU extern
int fputs_unlocked ( const char * __restrict __s , FILE * __restrict
__stream ) ; #endif #ifdef __USE_MISC extern size_t fread_unlocked
( void * __restrict __ptr , size_t __size , size_t __n , FILE * __restrict
__stream ) __wur ; extern size_t fwrite_unlocked ( const void * __restrict
__ptr , size_t __size , size_t __n , FILE * __restrict __stream
) ; #endif extern int fseek ( FILE * __stream , long int __off , int
__whence ) ; extern long int ftell ( FILE * __stream ) __wur ; extern
void rewind ( FILE * __stream ) ; #if defined __USE_LARGEFILE
|| defined __USE_XOPEN2K # ifndef __USE_FILE_OFFSET64 extern
int fseeko ( FILE * __stream , __off_t __off , int __whence ) ; extern
__off_t ftello ( FILE * __stream ) __wur ; # else # ifdef __REDIRECT
extern int __REDIRECT ( fseeko , ( FILE * __stream , __off64_t __off
, int __whence ) , fseeko64 ) ; extern __off64_t __REDIRECT ( ftello ,
( FILE * __stream ) , ftello64 ) ; # else # define fseeko fseeko64 # define
ftello ftello64 # endif # endif #endif #ifndef __USE_FILE_OFFSET64
extern int fgetpos ( FILE * __restrict __stream , fpos_t * __restrict
__pos ) ; extern int fsetpos ( FILE * __stream , const fpos_t * __pos ) ;
#else # ifdef __REDIRECT extern int __REDIRECT ( fgetpos , ( FILE *
__restrict __stream , fpos_t * __restrict __pos ) , fgetpos64 ) ; extern
int __REDIRECT ( fsetpos , ( FILE * __stream , const fpos_t * __pos
```

) , fsetpos64) ; # else # define fgetpos fgetpos64 # define fsetpos fsetpos64 # endif #endif #ifdef __USE_LARGEFILE64 extern int fseeko64 (FILE * __stream , __off64_t __off , int __whence) ; extern __off64_t ftello64 (FILE * __stream) __wur ; extern int fgetpos64 (FILE * __restrict __stream , fpos64_t * __restrict __pos) ; extern int fsetpos64 (FILE * __stream , const fpos64_t * __pos) ; #endif extern void clearerr (FILE * __stream) __THROW ; extern int feof (FILE * __stream) __THROW __wur ; extern int ferror (FILE * __stream) __THROW __wur ; #ifdef __USE_MISC extern void clearerr_unlocked (FILE * __stream) __THROW ; extern int feof_unlocked (FILE * __stream) __THROW __wur ; extern int ferror_unlocked (FILE * __stream) __THROW __wur ; #endif extern void perror (const char * __s) ; #include <bits/sys_errlist.h> #ifdef __USE_POSIX extern int fileno (FILE * __stream) __THROW __wur ; #endif #ifdef __USE_MISC extern int fileno_unlocked (FILE * __stream) __THROW __wur ; #endif #ifdef __USE_POSIX2 extern FILE * popen (const char * __command , const char * __modes) __wur ; extern int pclose (FILE * __stream) ; #endif #ifdef __USE_POSIX extern char * ctermid (char * __s) __THROW ; #endif #if (defined __USE_XOPEN && !de-

```
fined __USE_XOPEN2K) || defined __USE_GNU extern char * cuserid
( char * __s ) ; #endif #ifdef __USE_GNU struct obstack ; extern int
obstack_printf ( struct obstack * __restrict __obstack , const char *
__restrict __format , ... ) __THROWNL __attribute__ ( ( __format__
( __printf__ , 2 , 3 ) ) ) ; extern int obstack_vprintf ( struct
obstack * __restrict __obstack , const char * __restrict __format ,
__gnuc_va_list __args ) __THROWNL __attribute__ ( ( __format__
( __printf__ , 2 , 0 ) ) ) ; #endif #ifdef __USE_POSIX199506 extern
void flockfile ( FILE * __stream ) __THROW ; extern int ftrylockfile
( FILE * __stream ) __THROW __wur ; extern void funlockfile (
FILE * __stream ) __THROW ; #endif #if defined __USE_XOPEN
&& !defined __USE_XOPEN2K && !defined __USE_GNU # include
<bits/getopt_posix.h> #endif extern int __uflow ( FILE * ) ; extern
int __overflow ( FILE * , int ) ; #ifdef __USE_EXTERN_INLINES
# include <bits/stdio.h> #endif #if __USE_FORTIFY_LEVEL > 0
&& defined __fortify_function # include <bits/stdio2.h> #endif #ifdef
__LDBL_COMPAT # include <bits/stdio-ldbl.h> #endif __END_DECLS
#endif typedef unsigned char bit ; struct Nodo * primero = NULL ; struct
Persona { char * nombre ; int edad ; bit id ; } ; struct Nodo { struct
Persona persona ; struct Nodo * sig ; } ; void agregarPersona ( const char
* nombre , int edad , bit id ) { struct Nodo * nuevo = ( struct Nodo *
) malloc ( sizeof ( struct Nodo ) ) ; nuevo -> sig = NULL ; nuevo ->
persona . nombre = ( char * ) malloc ( 1 ) ; strcpy ( nuevo -> persona .
nombre , nombre ) ; nuevo -> persona . edad = edad ; nuevo -> persona
. id = id ; if ( primero == NULL ) { primero = nuevo ; } else { nuevo
-> sig = primero ; primero = nuevo ; } } unsigned char aislarBit ( int n ,
unsigned char ID ) { unsigned char bit = ID ; bit = bit « ( n -1 ) ; bit = bit
» 7 ; return bit ; } void imprimir ( ) { puts ( "\nLista de personas:\n" ) ;
struct Nodo * aux = primero ; while ( aux != NULL ) { printf ( "Nombre:
%s, Edad: %i, ID: %i, Bit2: %d\n" , aux -> persona . nombre , aux ->
persona . edad , aux -> persona . id , aislarBit ( 2 , aux -> persona . id )
) ; aux = aux -> sig ; } } int main ( ) { agregarPersona ( "Ronald" , 21 ,
12 ) ; agregarPersona ( "Rose" , 18 , 13 ) ; agregarPersona ( "Ashly" , 17
, 12 ) ; agregarPersona ( "Jack" , 14 , 18 ) ; agregarPersona ( "Chester" ,
13 , 21 ) ; agregarPersona ( "Mike" , 12 , 32 ) ; agregarPersona ( "Jen" ,
38 , 21 ) ; imprimir ( ) ; return 0 ; }
```

» Histograma Tokens Usados


PREPROCESSOR
LEXICALERROR
COMMENT
PUNTUACTOR
OPERATOR
LITERAL
IDENTIFIER
KEYWORD
Frecuencia de Tokens usados