UTN - FRBA
Sintaxis y Semántica de los Lenguajes
Trabajo Práctico n° 1 - Autómata Finito Determinístico para Constantes Enteras de C

Enunciado

Implementar en lenguaje C un programa que **reconozca las constantes enteras de ANSI C (*C89/C90*) y las clasifique por su tipo** (**Decimal**, **Octal**, o **Hexadecimal).**

El mismo deberá leer las cadenas separadas por **coma** (**,**) **(caracter de corte o centinela)** desde un **archivo de texto de entrada** (ej. *entrada.txt*) y generar una salida (por defecto por pantalla, salida estándar/*stdout*) correspondiente donde por cada cadena leída, escriba en una nueva línea de texto dicha cadena y seguidamente:

- Si la cadena fue reconocida, qué tipo de constante entera es: **DECIMAL**, **OCTAL**, o **HEXADECIMAL**
- Si la cadena no fue reconocida, simplemente: **NO RECONOCIDA**

El programa **debe implementar un autómata finito determinístico (AFD)**, leyendo del archivo de entrada y procesando **caracter a caracter** para el reconocimiento y clasificación de cadenas según estado final.

Uso

`./bin/tp1.exe <ruta archivo entrada> [parámetros adicionales opcionales]`

La ruta al archivo de texto de entrada es el único parámetro **obligatorio**.

Adicionalmente, pueden agregar parámetros **opcionales** que consideren útiles o convenientes (ej. activar logs verbosos para facilitar el debug, etc.).

Ejemplo de aplicación

Archivo de entrada (entrada.txt) de prueba (secuencia de caracteres separadas por coma):

```
0xFF,127,0159,0xaBb1,0Xx,0,010,09,127A,120
```

Ejecución:

`./bin/tp1.exe entrada.txt > salida.txt` [1]

Archivo de salida (salida.txt) esperado:

```
0xFF HEXADECIMAL
127 DECIMAL
0159 NO RECONOCIDA
0xaBb1 HEXADECIMAL
0Xx NO RECONOCIDA
0 OCTAL
010 OCTAL
09 NO RECONOCIDA
127A NO RECONOCIDA
120 DECIMAL
```

[1] La extensión *.exe* sólo aplica para Windows. Éste comando redirige la salida por pantalla (stdout) del programa a un archivo (salida.txt). Si el archivo ya existe, lo sobreescribe; si no, lo crea.

Consideraciones

La entrega de este trabajo práctico **grupal** es **obligatoria**, su **fecha límite para consulta, entrega y revisión se encuentra indicada en el cronograma**. Luego de esa fecha, **no** se aceptarán más trabajos, y toda consulta referida quedará para la defensa final que al cierre de la cursada.

Se deberá **implementar la tabla de transiciones (TT) como una matriz** para obtener el estado correspondiente al que transicionar según el estado actual y el caracter de entrada.

No usar "números mágicos" para los nombres de los estados; utilizar enumeraciones (*enum*) o en su defecto directivas *#define*. Aspectos como la declaratividad, nombres de identificadores, abstracción y reutilización de código son tenidos en cuenta en las correcciones como buenas prácticas de programación. Así como eficiencia de uso de CPU y memoria.

Queda a criterio de cada grupo si contemplar casos borde como un archivo de entrada vacío, una entrada **,,** , o una cadena de entrada que abarca varias líneas, y cómo interpretarlos.

Para escribir el código fuente del programa pueden utilizar cualquier estándar de C, como: ANSI C (*C89/C90)*, *C99*, *C11*, *C17/C18*, etc.[2]

Como compilador de C se utilizará *gcc* (***G**NU **C** **C**ompiler*). Sin embargo, se recomienda **no utilizar** extensiones del lenguaje C específicas del compilador (*GNU C Extensions*) que no formen parte del estándar de C utilizado.[3]

El programa debe poder compilarse independientemente del sistema operativo, utilizando únicamente bibliotecas estándar de C según necesidad: *stdio.h*, *stdlib.h*, *ctype.h*, *string.h*, *math.h*, *errno.h*, etc.
En otras palabras, **el código C escrito debe ser portable**; para ello, no debe incluir bibliotecas específicas de ningún sistema (ej. *Windows*, *Linux*, *Mac OS*, etc.). Por ejemplo, bibliotecas definidas en los estándares *POSIX* (***P**ortable **O**perating **S**ystem **I**nterface/Unix*), ya que no son compatibles nativamente con Windows, entre otros sistemas.

Tener en consideración las funciones:

- De *stdio.h*: getchar, getc, fgetc, ungetc, putchar, putc, fputc, feof (así como fopen y fclose).
- De *ctype.h*: isalnum, isalpha, isblank, iscntrl, isdigit, islower, isprint, ispunct, isspace, isupper, isxdigit, tolower, toupper

Tener en cuenta los tipos de datos declarados en las variables así como los en las funciones utilizadas los tipos de datos de los parámetros y de retorno

---

[2] En *gcc* el estándar de C utilizado se especifica agregando una opción *-std* al compilar, entre las que están: *-std=c89*, *-std=c99*, *-std=c11*, *-std=c17*, etc. Para conocer todas las opciones *-std* (ergo todos los estándares) que soporta una versión de *gcc*, pueden consultar la documentación de GCC que corresponda a dicha versión. Con *gcc --version* pueden consultar la versión instalada.
Si no se indica una opción *-std* al compilar, *gcc* por defecto utiliza el último estándar de C que soporte pero aceptando las extensiones del lenguaje propias del compilador.

[3] *gcc* puede alertar del uso de extensiones de C como warnings si se le agrega la opción *-Wpedantic* y/o como errores si se le agregan las opciones *-pedantic -pedantic-errors* al compilar.

El **entorno de desarrollo a utilizar queda a elección de cada grupo** (*Eclipse*, *CodeBlocks*, CLion, ***Visual Studio Code***, etc.). Se recomienda un entorno que aplique resaltado de sintaxis para todos los archivos de especificación, que facilite la depuración de código C, así como que se integre con la consola de *Git* para poder realizar el trabajo de una forma más práctica.

Para compilar el programa, pueden utilizar herramientas como ***GNU Make*** con archivos de especificación correspondientes como ***makefiles***, ya sean de su propia autoría o los que les hayan sido provistos (en el *aula virtual* o en las plantillas de TPs en los repositorios grupales designados). En dicho caso, **esos archivos deberán formar parte de la entrega**.

Para el programa, sólo formarán parte de la entrega los **archivos fuente** de C (*.c*) **y/o archivos de cabecera** (*header files*) (*.h*) escritos por el equipo. No se considerarán archivos ejecutables (ej. *.exe*) ni archivos intermedios de salida del compilador, como los archivos objeto (ej. *.o*), ensamblador (ej. *.s*) o preprocesados (ej. *.i*).

Completar el archivo *README.md* dentro del directorio del TP, el cual actúa como carátula del mismo, con los datos que allí consignan.

La **entrega será a través del repositorio** de GitHub designado para el equipo en la carpeta correspondiente al TP **mediante Pull Request (PR)** desde la rama creada para el TP hacia la rama principal (main), como se indica en el instructivo para entrega de trabajos prácticos.

Pueden **agregar también las aclaraciones que consideren necesarias mencionar** en relación con el trabajo práctico realizado **en dicho Pull Request (PR)**.

Las **consultas** podrán ser realizadas a través de Discord en el canal correspondiente a cada grupo, o bien, si es una consulta más general del trabajo referida al enunciado por el canal de consultas denominado TP-**#**, siendo **#** el número de TP.