Logotipo, nombre de la empresa

Descripción generada automáticamente

Ingeniería en sistemas de Información

Sintaxis y semántica de los lenguajes – 2023

TURNO: Noche CURSO: K2055

DOCENTE A CARGO: Ing. Roxana Leituz

AYUDANTE:

TRABAJO PRÁCTICO TEORICO N°2

“Autómatas*”*

ÁREA TEMÁTICA: Autómatas

GRUPO N° 30

Rodríguez Lucas Ariel

Golato Barcia Ivan Nahuel

Sayago Pablo

Rabahia Maron Leonel

Schinca Mauro

FECHA DE VENCIMIENTO: 1/10/2023

FECHA DE PRESENTACIÓN:1/10/2023

FECHA DE DEVOLUCIÓN: \_\_/\_\_/\_\_

CALIFICACIÓN: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ FIRMA PROFESOR: \_\_\_\_\_\_\_\_\_

índice

* Autómatas
  + Números decimales
  + Números octales
  + Números hexadecimales
* Calculadora

Autómatas:

Nosotros decidimos utilizar un solo autómata que tenga la capacidad de identificar números decimales (con signo + o -), octales y hexadecimales.

Tabla de transición:

Imagen que contiene biombo, edificio, reloj

Descripción generada automáticamente

Modo de uso:

Texto

Descripción generada automáticamente

Primero debemos compilar nuestros archivos **.c** con los archivos **.h**, para eso utilizamos el **Makefiel**, por medio de nuestra consola ingresamos la palabra **make** que realiza la compilación en orden.

Luego de tener nuestro archivo automata.exe ya creado procedemos a ejecutarlo en consola con el comando:

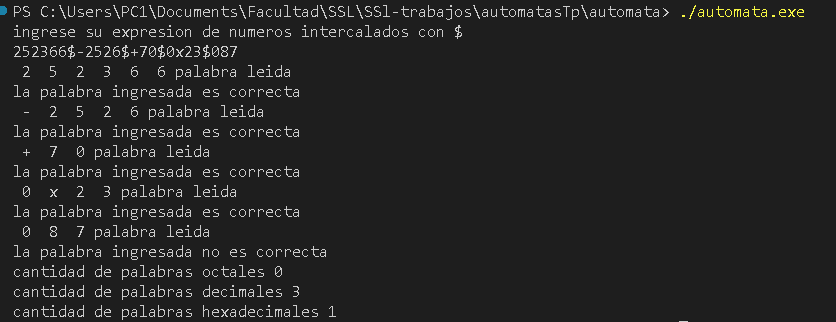
./automata.exe

Esto nos llevaría a ejecutar nuestro programa donde debemos introducir una expresión de números tanto decimales con o sin signo, como octales y hexadecimales. Todos ellos intercalados por el símbolo **“$”** que lo utilizaremos como un carácter para saber el final de un número y el comienzo de otro.

Importante la expresión debe comenzar sin el carácter **“$”**, ya que al inicio de la expresión sería el primer número en cuestión.

En caso de que todos los números ingresados sean correctos el programa nos brindara la cantidad de cada uno, como se muestra en la imagen superior.

En caso contrario mostrara el siguiente mensaje:



Como vemos además de indicarnos cuales números fueron correctos, nos muestra cual fue el caso erróneo y la cantidad de cada grupo leído hasta el momento.

Funciones y estructura del código:

Texto

Descripción generada automáticamente

columnas.h contiene la función columna() y isAToF(), que las utilizamos para determinar la columna de la tabla de transición y si un carácter pertenece al conjunto [ A-F] respectivamente.

Cuerpo principal del programa:

Texto

Descripción generada automáticamente

Consiste principalmente en ir tomando cada carácter de la entrada estándar y guardarlos en un array, que se sobrescribe constantemente por la aparición del carácter **“$”.**

Y al encontrar el carácter “/n” (salto de línea) sin haber tenido ningún error muestra la cantidad de números de cada conjunto encontrados. En caso contrario nos muestra el error pertinente.

Texto

Descripción generada automáticamente

La función palabraPertenece() recibe lo que podría ser un numero y determina si todos sus miembros pertenecen a nuestro alfabeto predefinido.

Texto

Descripción generada automáticamente

Es palabra () recibe los punteros a enteros con el fin de usarlos como contadores de los tipos de números que vamos recibiendo, además implementa la función autómata que como vemos devuelve el estado final del autómata si le pasamos por parámetros el numero a analizar.

Texto

Descripción generada automáticamente

La función autómata recibe el numero y con la tabla de transición va cambiando el estado utilizando la función columna que se aloja en **columna.h** y por último devuelve el estado final del autómata.

Texto

Descripción generada automáticamente

Por ultimo dentro de **columna.c** podemos encontrar estas dos funciones, isAToF() determina si c pertenece al conjunto [A-F] y columna nos devuelve el valor que coincide con la columna de la tabla de transición según el carácter que sea c.

Calculadora:

La calculadora consiste en dos partes fundamentalmente, la primera parte consiste en leer la expresión aritmética a realizar y analizar que esta correctamente armada, tanto la expresión en su totalidad como cada uno de sus miembros, para ello utilizamos el siguiente autómata.

Imagen en blanco y negro

Descripción generada automáticamente con confianza media Tabla

Descripción generada automáticamente

El cual lo implementamos de la siguiente forma:

Texto

Descripción generada automáticamente

Dentro del archivo automata.c pero dentro de la capeta principal que contiene el autómata anterior, el cual utilizamos para determinar si un numero es un decimal o no.Y en columna.c agregamos la función columnaAritmetica.

Imagen que contiene Texto

Descripción generada automáticamente

El nuevo autómata se utiliza en el archivo **verificacion.c**, el cual se encarga de analizar toda la expresión como así sus elementos

Texto

Descripción generada automáticamente

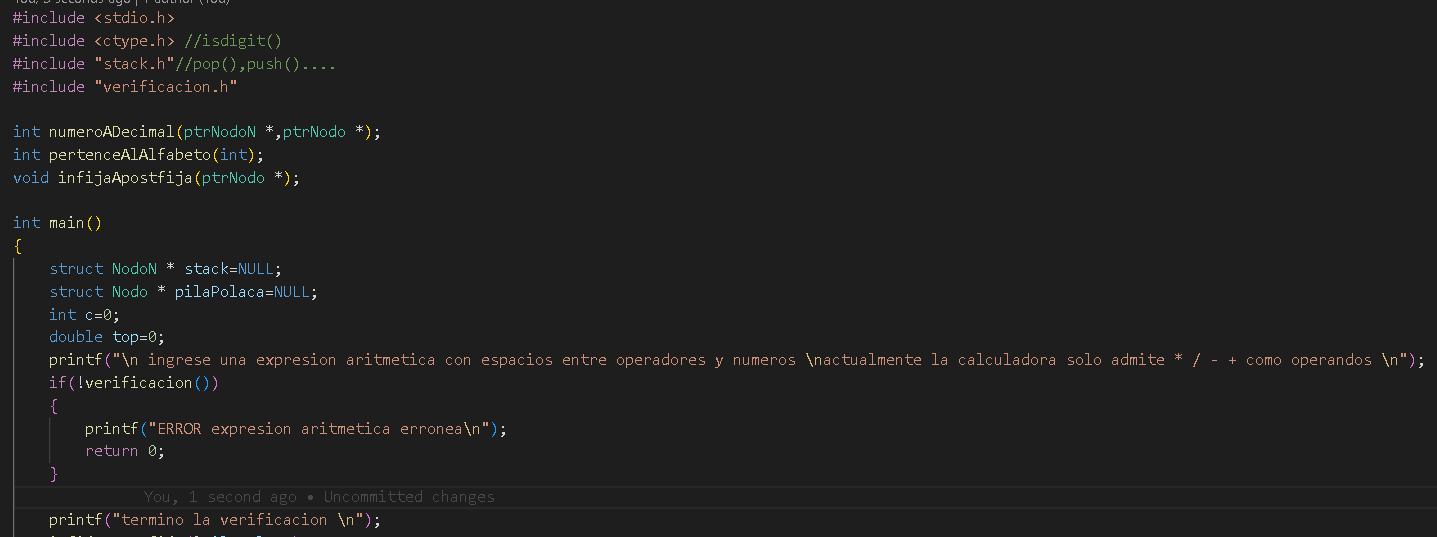
Texto

Descripción generada automáticamente

Ya teniendo la expresión verificada podemos continuar con la siguiente parte que consiste en pasar de una expresión aritmética infija a postfija, con el fin de resolver el problema de la precedencia en las operaciones. Para hacer esto seguimos el siguiente algoritmo que utiliza dos pilas, una final y otra de operadores.

Enlace:

<https://www.youtube.com/watch?v=kpLlufrQiTk&t=2s>



Dentro de la función infijaApostfija() es donde se convierte la expresión y se guarda en una pila para su uso posterior por la calculadora polaca.

La calculadora polaca funciona leyendo datos desde la pilaPolaca que obtenemos de la función infijaApostfija(), por como se transformó la expresión ahora el switch va a ir encontrando sucesivamente dos numero y un operador, lo cual va a permitirle realizar las operaciones que vaya leyendo con los números que hay guardados en la pila y finalmente guardando los resultados en la pila para seguir operando sucesivamente.

Texto

Descripción generada automáticamente

Y para obtener los números en un formato mas preciso utilizamos la función de numeroAdecimal() que consiste en ir transformando los numero de char a int y sumarlos en una sumatoria de doubles, por lo tanto el numero obtenido va a una nueva pila pero en este caso una de doubles que será nuestra pila final.

La función numeroAdecimal() realiza el push por cada numero obtenido y si lee algo que no es un numero lo devuelve para que el switch de la calculadora polaca realice la operación , lo ignore , devuelva un erro o por ultimo nos de el resultado que va estar en el tope de la pila final.

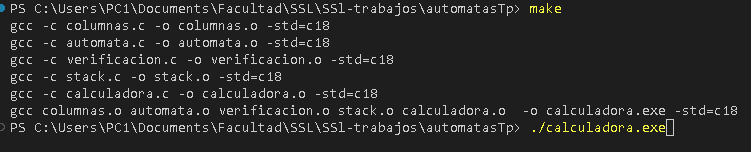
Texto

Descripción generada automáticamente

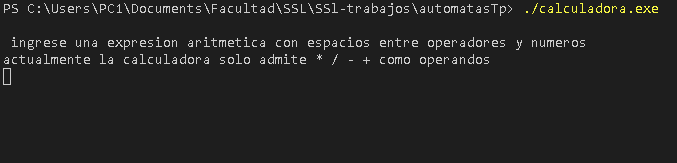
Por último, en **stack.c** especificamos las dos versiones de pop y push que utilizamos para cada uno de los tipos de pilas que utilizamos.

Modo de uso:

Primero ejecutamos el comando **make** en nuestra consola y obtendremos el **calculadora.exe**



Al ejecutarlo nos va mostrar la siguiente pantalla donde ingresamos la expresión sin espacios



Al ingresar la expresión oprimiendo el botón **enter** obtenemos el resultado.

Texto

Descripción generada automáticamente

La calculadora principalmente se inspiro en muchas idea del libro de Brian Kernighan y Dennis Ritchie, específicamente el capítulo cuatro.