Instrucciones

Se le contrata para realizar un IDE el cual es específico para las tareas de la empresa de desarrollo de software el cual servirá para crear aplicaciones específicas para algunos trabajos de análisis de datos. Este proyecto será creado a través de varias fases de las cuales usted y su equipo de desarrollo se encargaran. La primera fase tendrá que ser realizada por usted y entregada en la fecha que se indica en este documento.

Primera fase:

La primera fase se trata de crear la interfaz gráfica la cual estará encargada de:

- Manejo del código fuente.
- Manejo de archivos.

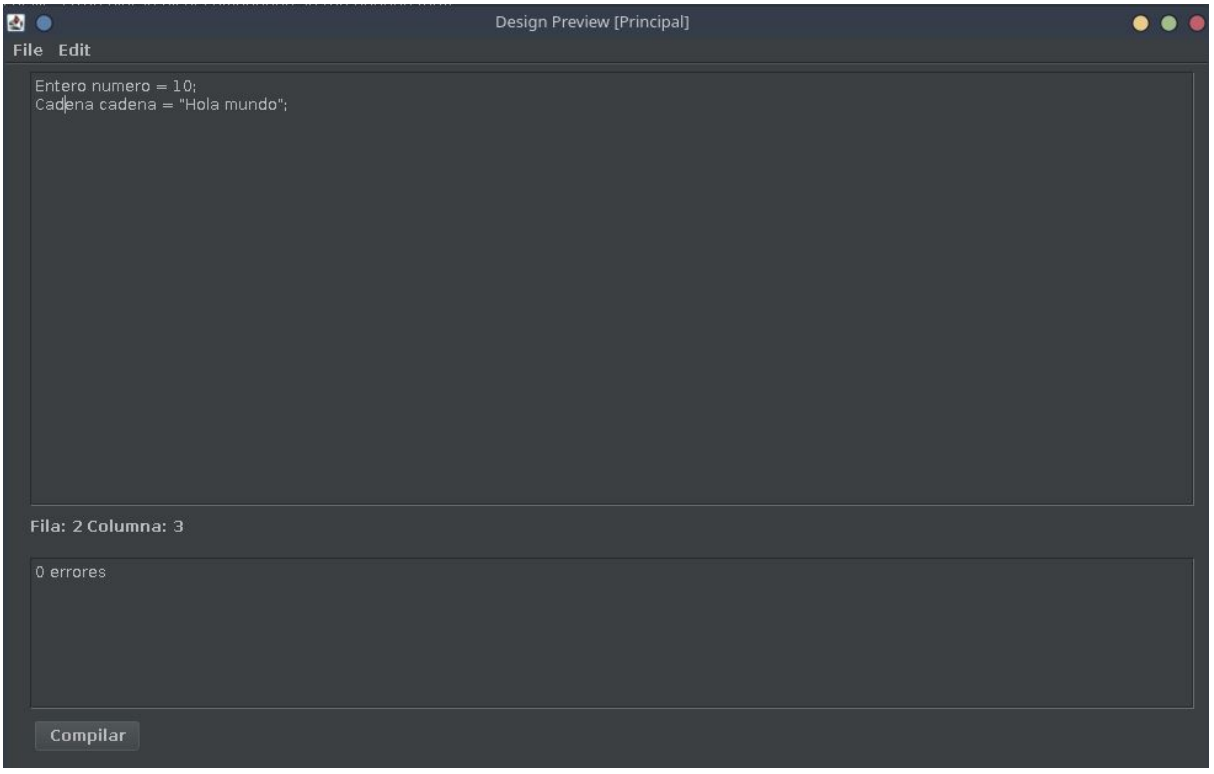
Manejo de archivos

Para el manejo de archivos se solicita que todos los archivos que puedan ser creados, editados o eliminados por este IDE tendrán que ser de la extensión .gt debido a los requerimientos de la empresa. Para poder manejar los archivos de programación las acciones que debe realizar el IDE son:
Crear, abrir, eliminar, cerrar proyecto
Crear, abrir, eliminar, editar, cerrar archivos de código fuente.

El IDE debe contar con ciertas partes esenciales la cuales usted podrá ubicar en donde crea que pueda ser lo más manejable e intuitivo,además podrá agregar otras opciones que crea conveniente, las partes esenciales son las siguientes:

- Un área para el manejo de archivos donde se podrán crear proyectos o abrir los proyectos que ya fueron creados con anterioridad.
- Un área de texto donde los desarrolladores podrán escribir su código de manera ordenada para las aplicaciones que estos tengan que realizar.
- Una etiqueta donde muestre la fila y la columna donde se encuentre el cursor en cualquier momento y para los errores.

- Un área **log** donde principalmente mostrará los errores léxicos que puedan haber dentro del área de código. (Este log podrá ser exportado a un archivo con extensión .gtE)
- Botón de compilación (opcional)



(imagen demostrativa, cada uno lo puede desarrollar de la manera que lo considere conveniente)

Todas estas partes deben estar, aparte de las que usted puede agregar, sin embargo el posicionamiento de las diferentes partes no necesariamente tiene que ser como en la imagen, queda a su creatividad pero buscando la facilidad del uso.

Estilo

Para la tarea de dar estilo usted deberá identificar el tipo de cadena ingresada y pintarla según su tipo.

Tipos de dato primitivos

Tipo	Palabra reservada	Color	Ejemplo
Entero	entero	Morado	15 -90 23
Decimal	decimal	Celeste	-25.2 12.588 5.0
Cadena	cadena	Gris	“es una cadena”
Boolean	booleano	Naranja	verdadero falso
Chart	carácter	Cafe	a A B i

Operadores aritméticos

Operador	Color	Ejemplos
+	Azul oscuro	5 + 4, -25.25 + 13, 12 + 4.55, “texto” + 12, unId + 13.25
-	Azul oscuro	5 - 4, -25.25 - 13, 12 - 4.55, unId - 13.25
*	Azul oscuro	5 * 4, -25.25 * 13, 12 * 4.55, unId * 13.25
/	Azul oscuro	5 / 4, -25.25 / 13, 12 / 4.55, unId / 13.25
++	Azul oscuro	5++, 2.5++, unId++
--	Azul oscuro	5--, 2.5--, unId--

Operadores relacionales

Operador	Color	Ejemplos
>	Azul oscuro	5 > 4, -25.25 > 13, 12 > 4.55, unId > 13.25
<	Azul oscuro	5 < 4, -25.25 < 13, 12 < 4.55, unId < 13.25
>=	Azul oscuro	5 >= 4, -25.25 >= 13, 12 >= 4.55, unId >= 13.25
<=	Azul oscuro	5 <= 4, -25.25 <= 13, 12 <= 4.55, unId <= 13.25
==	Azul oscuro	5 == 4, -25.25 == 13, 12 == 4.55, unId == 13.25, “texto == “otro”
!=	Azul oscuro	5 != 4, -25.25 != 13, 12 != 4.55, unId != 13.25, “texto !=

		“otro”

Operadores lógicos

Operador	**Color**	**Ejemplos**
||	Azul oscuro	**5 > 4, -25.25 > 13, 12 > 4.55, unId > 13.25**
&&	Azul oscuro	**5 < 4, -25.25 < 13, 12 < 4.55, unId < 13.25**
!	Azul oscuro	**5 >= 4, -25.25 >= 13, 12 >= 4.55, unId >= 13.25**

Signos de agrupación

Operadores	**Color**	**Ejemplos**
( )	Azul oscuro	**3 - (5 / (25 + 4))** **!!(5 > 25)** **(!!(a > b) && verdadero || ((c == 5) && !!falso) || d <= 4) && 5 >= g**

Asignación y fin de sentencia

Símbolo	**Color**	**Ejemplos**
=	Rosado	**unId = 25 + 25;**
;	Rosado	**entero mild;**

Palabras reservadas

Palabras	**Color**

SI, SINO, SINO_SI, MIENTRAS,HACER,DESDE,HASTA, INCREMENTO, entero, decimal, principal...	**Verde**

Sintáctico → léxico

Ademas debera trabajar con los comentarios de los siguientes tipos

Ejemplo	**Color**
//Esto es un comentario	Rojo
/* Esto también 5 es un comentario */	Rojo

Segunda Fase:

Para la segunda fase se considerarán otros puntos importantes para el desarrollo en este idea, entre los cuales destaca el análisis sintáctico de lo que el usuario escriba. todos los nuevos requerimientos se describen a continuación.

Variables locales

Para el caso de las variables funcionaran como un espacio en memoria donde se podrán almacenar datos de tipo entero, decimal, cadena, booleano y char. algunos ejemplos de la declaración de un variable son los siguientes:

```
entero _id;
entero _id1, _id2, _id3;
cadena _hola = “hola”;
```

Tome en cuenta que para que no sea considerado un error sintáctico primero tenemos que tener el tipo de la variable y luego la variable. también observe que se pueden crear varias variables del mismo tipo como se ve en el segundo ejemplo, pero cada identificador debe ser separado por medio de comas. Nótese que para ser tomado como identificador debe llevar un guión bajo antes del este.

Imprimir y leer

Imprimir: Esta función imprimirá el valor que se le pase por parámetro dentro de los paréntesis la impresión será dentro del área de texto donde se muestran los errores (sólo se imprimirá si no hay errores dentro del código).
Leer: Esta función podrá leer algún valor desde la interfaz y tendrá la capacidad de asignar dicho valor a alguna variable. Ejemplos:

```
imprimir(_id2);
imprimir("texto" + 2);
leer(_id3);
```

Estructura del código

Para que nuestro código sea válido tendrá que ir dentro de nuestro método principal como se muestra a continuación, vea que para que todo nuestro código sea válido el método principal debe ir seguido de los paréntesis, luego de la llaves, y dentro de estas llaves podremos ingresar el código necesario.

```
principal () {

        escribir( _algo + _algo1 + . . .  );
        leer( _variable);

        entero _a = 0;
        MIENTRAS _a < 5 {
                escribir ( _a + "resultado \n");
                _a++;
        }

}
```

Estructura SI, SINO_SI, SINO

Se sigue la misma función de las instrucciones if, else if, else en C#, la instrucción SINO es opcional y puede formar parte de la estructura como máximo una vez La instrucción SINO_SI es opcional y puede formar parte de la estructura n veces.

Ejemplos:

```
SI (_a > _b) {
_T = 500;
}
```

```
SI (_a > _b){
_a = 770;
}
SINO {
_b = 50;
}
```

```
SI (!!falso){
_a = 770;
}
SINO_SI (_c > _b && verdadero){
_b = 50;
}
SINO_SI (verdadero){
//sentencias
}
SINO {
//sentencias
}
```

MIENTRAS, HACER

Siguen la misma funcionalidad que las instrucciones while y do while de C#.
Ejemplos:

```
MIENTRAS(id>10){
_c = 5;
_a = 159.01;
imprimir(“datos”);
entero _vari = 15;
}
```

```
HACER(id>10) {
_c= 5;
_a = 159.01;
imprimir(“datos”) ;
enteros _var = 15;
}
MIENTRAS (_datos == 2)
```

DESDE, HASTA, INCREMENTÓ

Siguen la misma funcionalidad que la instrucción for (i = 0…..
Ejemplos:

```
DESDE _i = 0 HASTA i < 20 INCREMENTO 1{
_c= 5;
_a = 159.01;
imprimir(“datos”) ;
enteros _var = 15;
}
```

Sugerencias del IDE

Otro punto importante a considerar es que el proyecto deberá tener la funcionalidad para poder sugerir palabras mientras el desarrollador vaya escribiendo por ejemplo si dentro del área de texto el desarrollador escribe la palabra “S” el ide tendrá que sugerir las palabras reservadas “SI” , “SINO” y “SINO_SI”

Agregados proyecto final

Deberá tomar en cuenta que a su proyecto tendrá que agregarle la opción en la que se pueda graficar el árbol sintáctico de la entrada analizada esta gráfica podrá ser visualizada en su ide como también podrá ser exportada como imagen hacia alguna ruta, tome en cuenta que para esto tendrá que utilizar su autómata de pila.

Sugerencia: puede apoyarse de la herramienta llamada Graphviz

Consideraciones

Para el proyecto las reglas son las siguientes:

- Las copias del código tendrán un punteo de cero.
- El proyecto debe ser realizado en C#.
- Debe utilizar un control de versiones para su código, y demostrar al menos 10 commits, estos no pueden estar hechos el mismo día.
- Fecha de entrega: 13 de octubre del 2020.
- Solo debe de subir el enlace del git.
- A entregar
    - Código fuente
    - Documentación:
        - Definición del lenguaje
        - Definición de los autómatas y el método que utilizó para generarlos.
        - Diagrama de clases
        - Manual Tecnico
        - Manual de Usuario

console.log('doing the things right or doing the right things);