MANUAL DE C.JS COMPILER

Instalación de IDE

Requerimientos

- Node.js 10+

Ejecución

En el directorio del proyecto:

```
$ npm install
$ npm start
```

Corre la aplicación en modo desarrollo. Abre http://localhost:3000 para mirar en el navegador.

Elementos del lenguaje

Comentarios

Los comentarios se hacen anteponiendo el signo '#' y luego la cadena de texto.

El intérprete omite las líneas de comentarios, lo cuál es bastante útil si deseamos poner información explicativa en el código. Por ejemplo alguna explicación sobre el comportamiento o objetivo de una función.

```
#Comentario de prueba
```

Variables

Las variables se definen de forma estática, lo que significa que se tiene que especificar cuál es el tipo de la variable, y esta puede tomar distintos valores. Se usa el símbolo = para asignar valores.

```
<Tipo de dato> nombre_variable = valor;
```

Ejemplo de sintaxis:

```
#Asignando el número 5 a la variable entera num
int num = 5;
```

Arreglos

- Para declarar un *arreglo* se usan los corchetes `[]`,los elementos se separan por comas.
- Los arreglos contienen elementos del mismo tipo..
- Para acceder a los elementos de un arreglo se utiliza un índice entero (empezando por "0", no por "1"). Se pueden utilizar índices negativos para acceder elementos a partir del final.
- Los *arreglos* se caracterizan por ser mutables, es decir, se puede cambiar su contenido en tiempo de ejecución,
- Los *Arreglos* se puede inicializar al momento de declararlos

Formas de declaración:

```
<tipo_dato> nombre [longitud];
<tipo_dato> nombre = {valor1,valor2,valor3};
```

Ejemplo de sintaxis:

```
int numerosNaturales[9];
int numerosNaturales = {0,1,2,3,4,5,6,7,8,9};
```

Ejemplo de asignación:

```
numerosNaturales[0] = 312;
```

Tipos de datos

Nombre	Descripción
Int	**Entero**
String	**Cadena**
Bool	**Booleano**
Char	**Carácter**
Float	**Flotante**

Instrucciones de selección:

Nombre	Descripción
If	**Condicional if**
else	**Condicional else**
elif	**Condicional else if**

Sintaxis de if

```
if (condition)
{
   #Código de instrucción
}
```

Sintaxis de if-else

```
if (condition)
{
   #Instrucción
}
else
{
   #Instrucción
}
```

Sintaxis de elif

```
if (condition) {
     #Instrucción
} elseif (condition) {
    #Introducción
} else {
     #Instrucción
}
```

Instrucciones iterativas:

Nombre	Descripción
for	**Iteración for**
while	**Iteración while**
do while	**Iteración do while**

Sintaxis de for

```
if (inizializador;condicion;iterador) {
     #Instrucción
}
```

Sintaxis while

```
int n = 0;
while (n < 5)
{
   cjprint(n);
    n++;
}
```

Sintaxis do-while

```
int n = 0;
do
{
    cjprint(n);
    n++;
} while (n < 5);
```

Instrucción de función

```
<Tipo> nombre(args){
    retorna variable o valor;
}
```

Ejemplo de sintaxis

```
int imprimir_numero(numero){
      cjprint(numero)
}
```

Palabras reservadas

Tipo	Palabras
Encierro de programa	**BEGIN, END**
Variables	**int, float, char, string, bool**
Instrucciones de selección	**if, else,elif**
Instrucciones iterativas	**for, while, do**
Entrada y salida de datos	**cjread, cjprint**
Manejo de archivos	**cjopenfile, cjreadfile, cjclosefile**
Retorno de función	**return**

Expresiones regulares

Operaciones

- L U M: Union de 2 lenguajes L y M
- LM: Interseccion entre 2 lenguajes L y M
- L^*: cero o mas ocurrencias de un lenguaje L

Ocurrencia de símbolos

- letra = [a-z] or [A-Z]
- dígito = [0-9]
- signo = [+ | -]

Representación de tokens utilizando expresiones regulares

- identificador = (letra | _) (letra | digito | _)^*
- entero = $(signo)^{?}$ $(digito)^{+}$
- número real = $(signo)^{?}$ $(digito)^{+}$ $(.(digito)^{+})^{?}$

Operadores

TIPO	OPERADORES
Matemáticos	**+,-,*,/**
Lógicos	**&&, ||**

Caracteres especiales

TIPO	CARACTERES
Comparación	**==,<,<=,>,>=**
Asignación	**=**
Manejo de arreglos	**[]**, **‘,’**
Apertura de instrucción	**{}**
Parámetros	**()**
Manejo de cadenas	**“ “**
Comentario	**#**