Manual del Usuario: Microcompilador Web

Introducción

Bienvenido al Microcompilador Web, una herramienta educativa diseñada para aprender los fundamentos de programación y compilación de código. Este manual te guiará a través de todas las funcionalidades disponibles y te enseñará cómo utilizarlas de manera efectiva.

Interfaz de Usuario

La interfaz del Microcompilador Web está diseñada para ser intuitiva y fácil de usar:

![Interfaz del Microcompilador Web]

La interfaz principal consta de tres secciones principales:

1. **Barra de botones**: Proporciona acceso a todas las funciones del microcompilador
2. **Editor de código**: Donde puedes escribir y editar tu código
3. **Panel de salida**: Muestra los resultados de la compilación y ejecución

Funciones Principales

Gestión de Archivos

- **Nuevo Archivo**: Crea un nuevo archivo en blanco. Si ya tienes contenido en el editor, este será eliminado.
- **Abrir Archivo**: Permite abrir un archivo de texto existente en tu ordenador.
- **Eliminar Archivo**: Elimina todo el contenido actual del editor.
- **Descargar**: Guarda el contenido actual del editor como un archivo de texto en tu ordenador.

Compilación y Ejecución

- **Ver Tokens**: Muestra un análisis de los tokens (palabras, símbolos, etc.) presentes en tu código.
- **Compilar**: Verifica la sintaxis de tu código y muestra errores si los hay.
- **Ejecutar**: Compila tu código y, si no hay errores, lo ejecuta paso a paso.

Documentación

- **Ver/Descargar Documentación**: Permite acceder a este manual de usuario o al manual del programador para información técnica más detallada.

Sintaxis del Lenguaje

El microcompilador utiliza un lenguaje sencillo con las siguientes características:

1. Instrucciones Básicas

Cada instrucción debe terminar con `::` para indicar su finalización, excepto las estructuras de control.

Salida de datos

```
write(expresión) ::
```

Ejemplo:

```
write("Hola mundo") ::
write(2 + 2) ::
write(nombre) ::
```

Entrada de datos

```
capture(variable) ::
```

Ejemplo:

```
capture(edad) ::
```

Asignación de variables

```
variable = expresión ::
```

Ejemplo:

```
nombre = "Juan" ::
edad = 25 ::
suma = a + b ::
```

2. Estructuras de Control

Condicional IF

```
if (condición) then
    instrucciones
end-if
```

Ejemplo:

```
if (edad >= 18) then
    write("Eres mayor de edad") ::
end-if
```

Bucle WHILE

```
while (condición)
    instrucciones
end-while
```

Ejemplo:

```
contador = 1 ::
while (contador <= 5)
    write(contador) ::
    contador = contador + 1 ::
end-while
```

3. Operadores

Operadores Relacionales

- `<`: Menor que
- `>`: Mayor que
- `<=`: Menor o igual que
- `>=`: Mayor o igual que
- `=`: Igual a
- `<>`: Diferente de

Operadores Lógicos

- `and`: Y lógico

- `or`: O lógico
- `not`: Negación

Operadores Aritméticos

- `+`: Suma
- `-`: Resta
- `*`: Multiplicación
- `/`: División

Ejemplos de Programas

Ejemplo 1: Calculadora Simple

```
write("CALCULADORA SIMPLE") ::
write("Ingrese el primer número:") ::
capture(num1) ::
write("Ingrese el segundo número:") ::
capture(num2) ::
write("Suma: " + (num1 + num2)) ::
write("Resta: " + (num1 - num2)) ::
write("Multiplicación: " + (num1 * num2)) ::
write("División: " + (num1 / num2)) ::
```

Ejemplo 2: Verificar Número Par o Impar

```
write("Verificar si un número es par o impar") ::
write("Ingrese un número:") ::
capture(numero) ::
resto = numero % 2 ::
if (resto = 0) then
    write("El número es par") ::
end-if
if (resto <> 0) then
    write("El número es impar") ::
end-if
```

Ejemplo 3: Contador con While

```
write("Contador del 1 al 10") ::
contador = 1 ::
while (contador <= 10)
    write(contador) ::
    contador = contador + 1 ::
end-while
write("¡Contador finalizado!") ::
```

Ejemplo 4: Cálculo de Factorial

```
write("Cálculo de factorial") ::
write("Ingrese un número:") ::
capture(numero) ::
factorial = 1 ::
contador = 1 ::
while (contador <= numero)
    factorial = factorial * contador ::
    contador = contador + 1 ::
end-while
write("El factorial de " + numero + " es: " + factorial) ::
```

Ejecución Paso a Paso

La función "Ejecutar" te permite ver cómo se procesa tu código instrucción por instrucción:

1. Haz clic en el botón "Ejecutar"
2. El sistema compila tu código para verificar que no haya errores
3. Si no hay errores, comienza la ejecución paso a paso
4. En cada paso, verás:
   - La línea que se está ejecutando
   - El resultado de la ejecución
   - El estado actual de las variables
5. La ejecución avanza automáticamente cada segundo
6. Cuando finaliza, se muestra el mensaje "Ejecución finalizada"

Esta función es especialmente útil para entender cómo se ejecuta el código y para identificar posibles problemas lógicos en tus programas.

Solución de Problemas Comunes

Error: "Palabra reservada mal escrita"

Verifica que estés utilizando correctamente las palabras reservadas como `write`, `capture`, `if`, `then`, `end-if`, `while`, `end-while`. Respeta mayúsculas y minúsculas.

Error: "Falta un operador relacional válido en la condición"

En las condiciones de `if` y `while` debe haber al menos un operador relacional (`<`, `>`, `<=`, `>=`, `=`, `<>`).

Error: "La instrucción debe finalizar con '::'"

Todas las instrucciones excepto las estructuras de control deben terminar con `::`.

Error: "Condicional if sin cierre adecuado con end-if"

Asegúrate de cerrar cada `if` con su correspondiente `end-if`.

Error: "Bucle while sin cierre adecuado con end-while"

Asegúrate de cerrar cada `while` con su correspondiente `end-while`.

Error: "Expresión inválida"

Verifica que las expresiones matemáticas y de texto estén correctamente formadas.

Recomendaciones y Buenas Prácticas

1. **Indentación**: Aunque el compilador no lo requiere, indenta tu código para hacerlo más legible.
2. **Nombres descriptivos**: Usa nombres de variables que describan su propósito.
3. **Pruebas incrementales**: Escribe y prueba pequeñas partes de código a la vez.
4. **Guarda tu trabajo**: Usa la función "Descargar" para guardar tu código frecuentemente.
5. **Consulta los ejemplos**: Si tienes dudas sobre la sintaxis, revisa los ejemplos proporcionados.

Conclusión

El Microcompilador Web es una herramienta educativa excelente para aprender los fundamentos de la programación y la compilación. Con práctica, podrás crear programas cada vez más complejos y entender mejor cómo funcionan los lenguajes de programación por dentro.

¡Feliz programación!