

Manual De Usuario SpectrumIDE



Contenido

Introducción.....	4
Partes del IDE.....	5
Menú Archivo.....	6
Menú Editar.....	7
Menú Compilar.....	8
Menú ayuda.....	8
Comience Con Spectrum.....	9
Hola mundo con Spectrum.....	9
Comentarios en Spectrum.....	9
Tipos de variables en Spectrum.....	10
Casting en Spectrum.....	10
If Else en Spectrum.....	11
Múltiples sentencias if en Spectrum.....	11
Arreglo 1D en Spectrum.....	12
Arreglo 2D en Spectrum.....	12
Entrada de datos en Spectrum.....	13
Sentencia Switch en Spectrum.....	14
Sentencia While en Spectrum.....	15
Sentencia Do While en Spectrum.....	16
Sentencia for en Spectrum.....	17
Ejemplo del uso de ciclos y condiciones en Spectrum.....	18
Ejemplo de la salida de datos en Spectrum.....	19
Funciones en Spectrum.....	19
Función en Spectrum con tipo de retorno y parámetros en su declaración.....	20

Equipo - 3

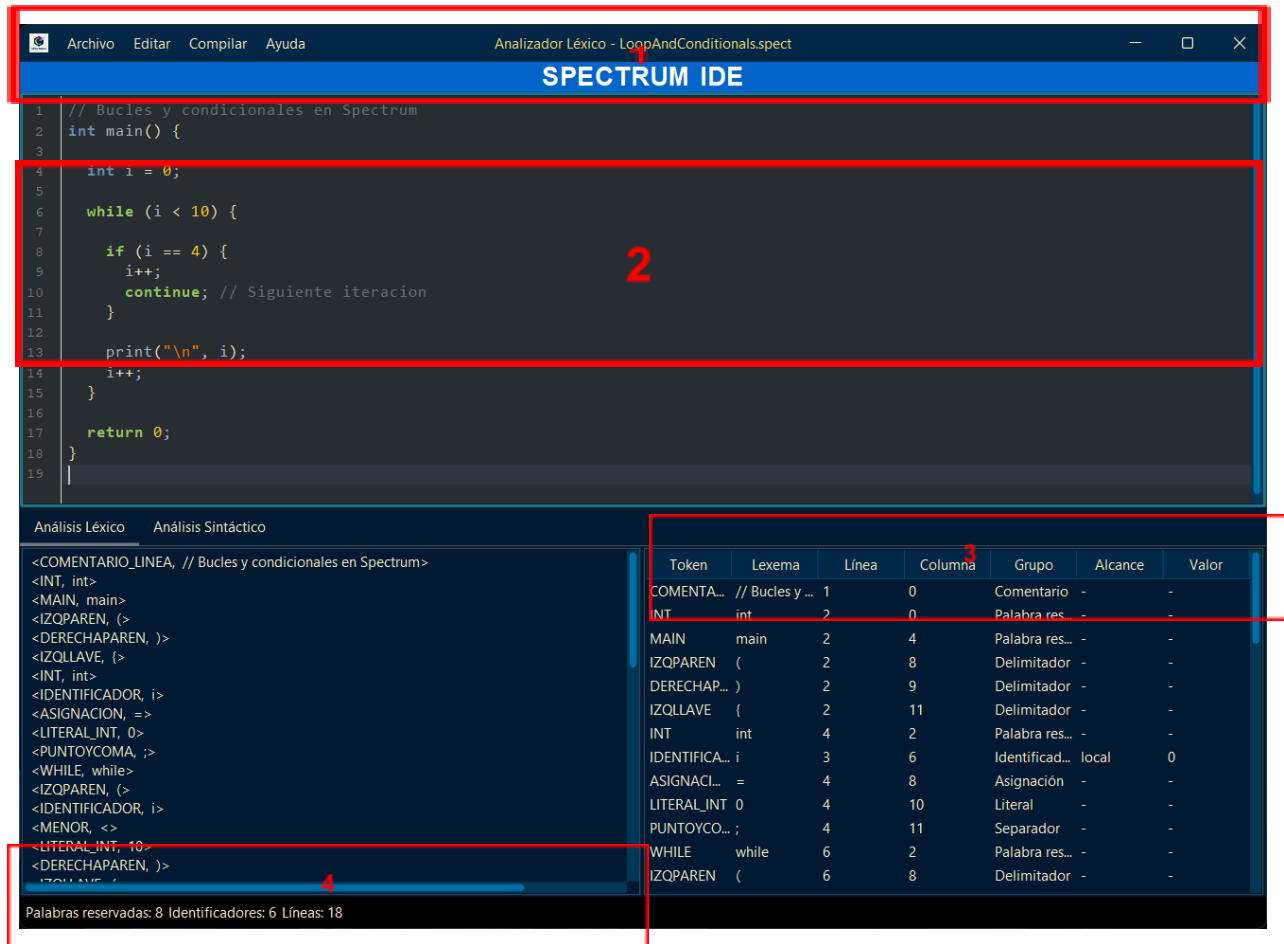
- **Gustavo Benito Reséndiz**
- **Cruz Alejandro Matías González**
- **Maximiliano Guzmán Arguelles**
- **Ángel Ricardo Juárez Del Ángel**
- **Jahaziel Emir Pérez Vicente**

Introducción

Este manual tiene como finalidad explicar el uso de la interfaz gráfica del entorno de desarrollo integrado (IDE), destacando las funciones de sus menús y submenús. Su propósito es facilitar al usuario las tareas de edición, compilación y análisis de código fuente de manera clara y organizada.

Partes del IDE

Componentes del IDE Spectrum.



Parte 1: Barra de menús (parte superior). Contiene las opciones principales del IDE. Archivo, Editar, Compilar, Ayuda.

Parte 2: Área de editor de texto. Este apartado está diseñado para poder escribir código, y muestra el número de líneas.

Parte 3: Tabla de símbolos. En esta área se encuentra el resumen del análisis léxico.

Parte 4: En esta parte se encuentra un resumen de palabras reservadas identificadores y líneas de código. Se actualiza solo en la ejecución.

Menú Archivo

El menú Archivo permite gestionar archivos de código fuente de forma eficiente.

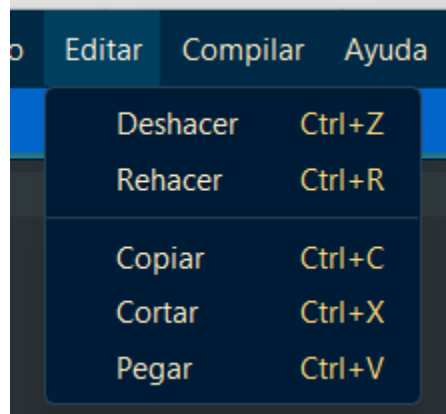


TABLA DESCRIPTIVA

Limpiar	Permite limpiar el área del editor de código.	Ctrl + n
Cargar	Nos despliega una ventana para poder buscar el archivo a utilizar.	Ctrl + o
Guardar	Nos permite guardar los cambios del archivo actual.	Ctrl + s
Guardar como	Nos despliega un menú para poder guardar el archivo en en una ruta específica y con extensión .spect	Ctrl + g
Salir	Permite salir del IDE de forma segura	Ctrl + q

Menú Editar

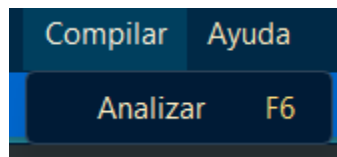
El menú Editar proporciona herramientas para modificar el texto dentro del IDE.



Deshacer	Revierte la última acción realizada.	Ctrl + z
Rehacer	Re aplica una acción que fue deshecha anteriormente.	Ctrl + r
Copiar	Copia el fragmento de código seleccionado al portapapeles.	Ctrl + c
Cortar	Elimina el fragmento de código seleccionado y lo copia al portapapeles	Ctrl + x
Pegar	Inserta el contenido del portapapeles en la posición actual del cursor.	Ctrl + v

Menú Compilar

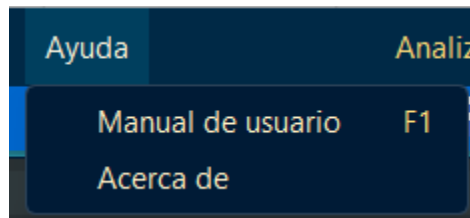
El menú Editar proporciona herramientas para modificar el texto dentro del IDE.



Analizar	Ejecuta el análisis léxico y sintáctico.	F6
-----------------	--	-----------

Menú ayuda

El menú ayuda Brinda soporte e información sobre el uso de la aplicación.



Manual de usuario	Abre el presente documento de manual F1 del usuario.	F1
Acerca de	Muestra la información general del sistema, incluyendo carrera,	NA

	materia, equipo desarrollador y versión.	
--	---	--

Comience Con Spectrum

Hola mundo con Spectrum.

```
1 // Un Hola Mundo en Spectrum.
2 int main() {
3     print("Hello World!");
4     return 0;
5 }
```

La instrucción print le ayudará a mostrar resultados por pantalla. Spectrum necesita una función de entrada a cualquier programa para su ejecución. Usted puede tener diferentes sentencias dentro de la función main.

Comentarios en Spectrum.

Spectrum permite comentarios de múltiples líneas y una sola línea. Utilice `/**/` para múltiples líneas y `//` para una sola línea.

```
1 /*
2  * Comentarios de multiples lineas en spectrum.
3  */
4 int main() {
5     print("Hello World!"); // Comentario de linea.
6 }
7
```

Tipos de variables en Spectrum.

Los tipos de variables en Spectrum son los siguientes: int, float, char, string y bool. Además de los arreglos. Para declarar una constante en Spectrum basta con anteponer la palabra reservada const antes del tipo.

```
1 // Variables en Spectrum
2 int main() {
3     int numeroInt = 15; // Entero
4     float numeroFloat = 3.1415; // Flotante
5     char caracter = '@'; // Caracter
6     string cadena = "Hola, desde Spectrum"; // Cadena
7     bool band = true; // Boolean
8     // Constantes en Spectrum
9     const float NUMERO_PI = 3.14159; // Definicion de una constante de tipo float
10    return 0;
11 }
12
```

Casting en Spectrum.

Para realizar el casting entre tipos de datos siga las siguientes nomenclaturas. int(), float(), string(), char(), bool() char solo podra ser casting de un string y bool igual.

```
1 // Casting en Spectrum
2 int main() {
3     // Por defecto input() retorna un string
4     int x = int(input()); // Casting a int
5     float y = float(x); // Casting a float
6     char c = char("@"); // Casting a char '@'
7     bool band = bool("true"); // Solo se puede el casting de string a bool.
8     return 0;
9 }
```

If Else en Spectrum.

Para utilizar la sentencia if-else en Spectrum utilice la siguiente estructura.

```
1 // Sentencia condicional if-else en Spectrum
2 int main() {
3     int time = 20;
4     if (time < 18) {
5         print("Good day.");
6     } else {
7         print("Good evening.");
8     }
9     return 0;
10 }
```

Múltiples sentencias if en Spectrum.

Usted puede añadir distintos tipos de condiciones en el lenguaje Spectrum. A continuación se muestra un ejemplo.

```
1 // Sentencia if - Múltiples condiciones
2 int main() {
3     int time = 22;
4     if (time < 10) {
5         print("Good morning.");
6     } else if (time < 20) {
7         print("Good day.");
8     } else {
9         print("Good evening.");
10    }
11    return 0;
12 }
13
```

Arreglo 1D en Spectrum.

Usted puede definir de la siguiente manera un arreglo en Spectrum. Observe la manera en que la sentencia for recorre el arreglo.

```
1 // Arreglo 1D con asignacion de datos en la definicion.
2 int main() {
3
4     int numeros[] = {25, 50, 75, 100};
5
6     for (int i = 0; i < len(numeros[]); i++) {
7         print("\n", numeros[i]);
8     }
9
10    return 0;
11 }
12
```

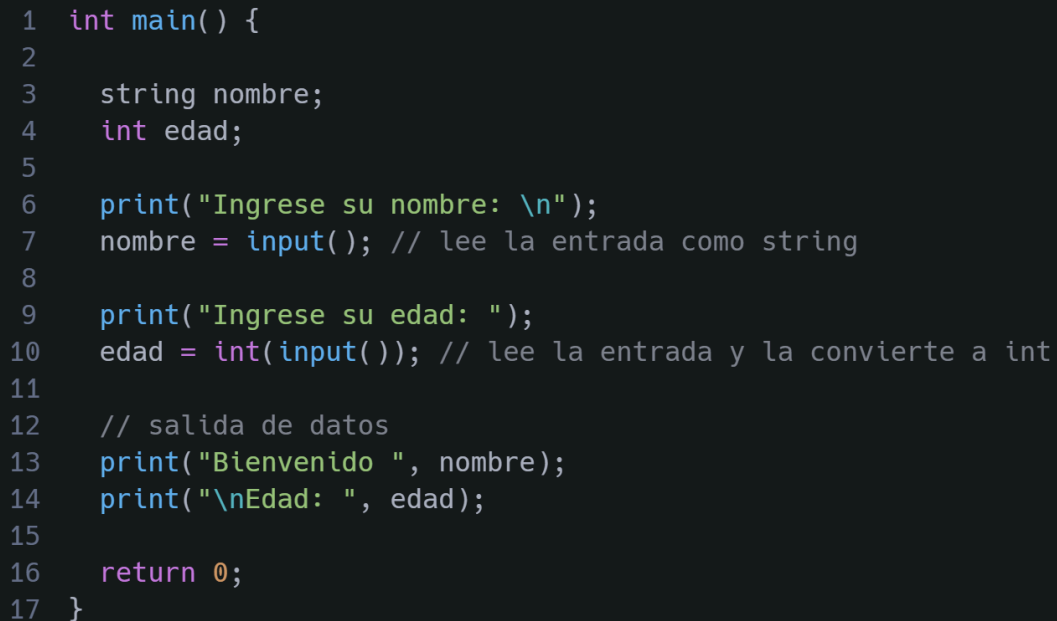
Arreglo 2D en Spectrum.

A continuación se muestra un ejemplo de como crear un arreglo de 2 dimensiones en Spectrum. Note cómo es muy similar a C en la definición.

```
1 // Arreglo 2D en Spectrum
2 int main() {
3
4     int matriz[2][3] = { { 1, 4, 2 }, { 3, 6, 8 } };
5
6     for (int i = 0; i < len(matriz); i++) {
7         for (int j = 0; j < len(matriz[i]); j++) {
8             print(matriz[i][j] + " ");
9         }
10        print("\n");
11    }
12
13    return 0;
14 }
15
```

Entrada de datos en Spectrum.

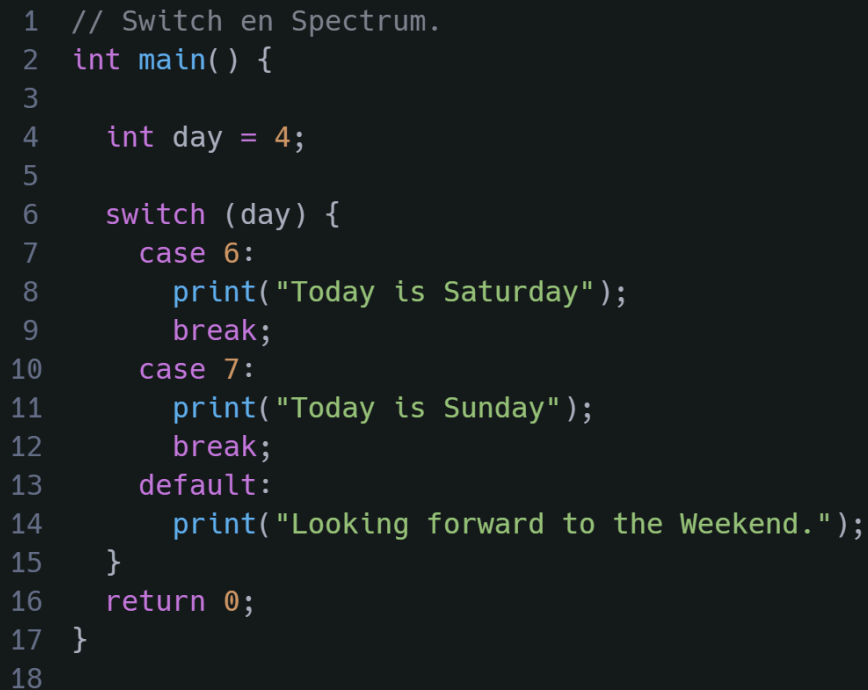
Usted puede leer entradas desde el teclado desde el teclado de la siguiente manera en Spectrum. Utilice la reservada `input()` por defecto `input` retorna un string es necesario el casting a otros tipos.



```
1  int main() {
2
3      string nombre;
4      int edad;
5
6      print("Ingrese su nombre: \n");
7      nombre = input(); // lee la entrada como string
8
9      print("Ingrese su edad: ");
10     edad = int(input()); // lee la entrada y la convierte a int
11
12     // salida de datos
13     print("Bienvenido ", nombre);
14     print("\nEdad: ", edad);
15
16     return 0;
17 }
```

Sentencia Switch en Spectrum.


Usted puede declarar la sentencia condicional múltiple de la siguiente manera. Switch solo puede evaluar literales enteros o identificadores de tipo entero.



```
1 // Switch en Spectrum.
2 int main() {
3
4     int day = 4;
5
6     switch (day) {
7         case 6:
8             print("Today is Saturday");
9             break;
10        case 7:
11            print("Today is Sunday");
12            break;
13        default:
14            print("Looking forward to the Weekend.");
15    }
16    return 0;
17 }
18
```

Sentencia While en Spectrum.

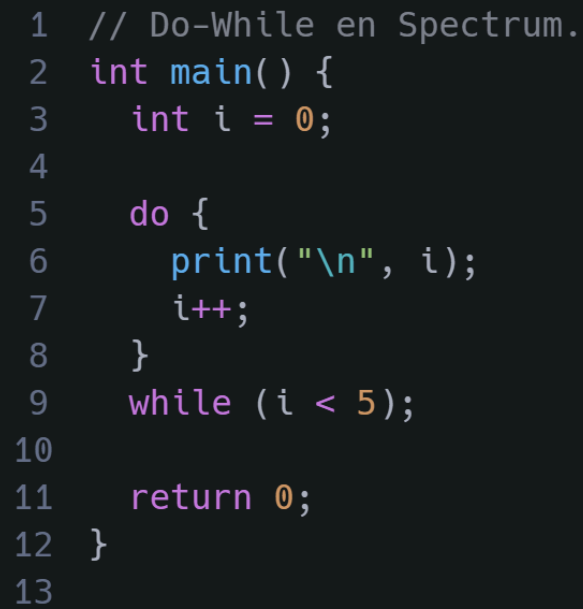
Puede hacer uso de los ciclos perfectamente en Spectrum, para el caso del ciclo mientras siga la estructura del siguiente ejemplo.



```
1 // While en Spectrum
2 int main() {
3
4     int i = 0;
5
6     while (i < 5) {
7         print("\n", i);
8         i++;
9     }
10
11     return 0;
12 }
13
```

Sentencia Do While en Spectrum.

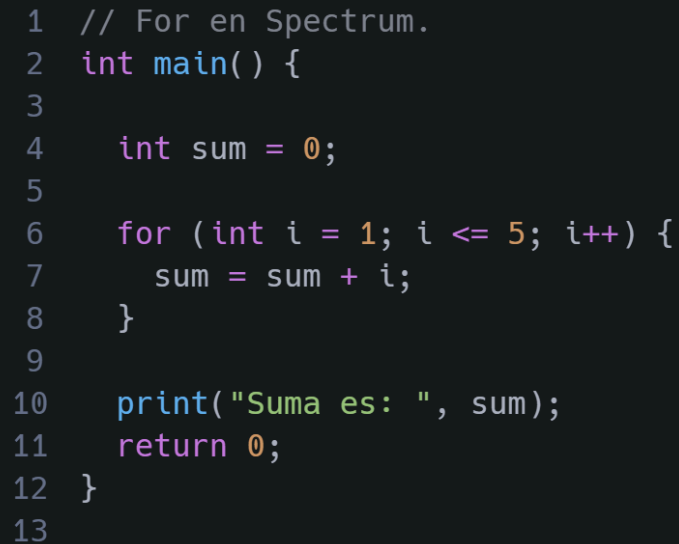
Usted puede hacer uso de la sentencia do-while en Spectrum de la siguiente manera.



```
1 // Do-While en Spectrum.
2 int main() {
3     int i = 0;
4
5     do {
6         print("\n", i);
7         i++;
8     }
9     while (i < 5);
10
11     return 0;
12 }
13
```


Sentencia for en Spectrum.


Usted puede hacer uso de la sentencia for de la siguiente manera en Spectrum. Muy útil al recorrer estructuras como los arreglos.



```
1 // For en Spectrum.
2 int main() {
3
4     int sum = 0;
5
6     for (int i = 1; i <= 5; i++) {
7         sum = sum + i;
8     }
9
10    print("Suma es: ", sum);
11    return 0;
12 }
13
```

Ejemplo del uso de ciclos y condiciones en Spectrum.

Ejemplo del uso de los ciclos y condiciones en Spectrum. Un ejemplo de lo que usted puede lograr con su imaginación. Las instrucciones break y continue funcionan perfectamente con los ciclos.



```
1 // Bucles y condicionales en Spectrum
2 int main() {
3
4     int i = 0;
5
6     while (i < 10) {
7
8         if (i == 4) {
9             i++;
10            continue; // Siguiendo iteracion
11        }
12
13        print("\n", i);
14        i++;
15    }
16
17    return 0;
18 }
19
```

Ejemplo de la salida de datos en Spectrum.

```
1 // Salida de datos en Spectrum.
2 int main() {
3     // Datos del estudiante
4     int edadEstudiante = 20;
5     float alturaEstudiante = 72.2;
6     char grupoEstudiante = 'A';
7
8     // Salida de datos en terminal
9     print("Edad del estudiante: ", edadEstudiante);
10    print("\nAltura del estudiante: ", alturaEstudiante);
11    print("\nGrupo del estudiante: ", grupoEstudiante);
12    return 0;
13 }
14
```

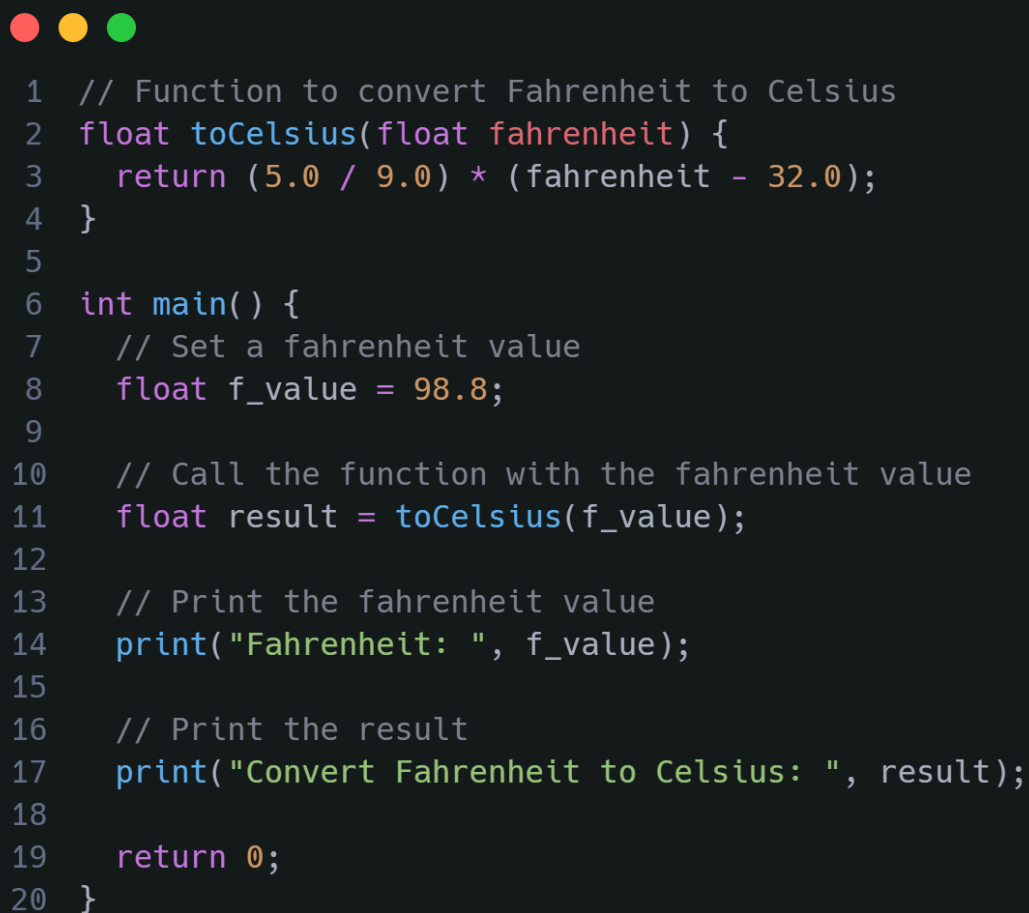
Funciones en Spectrum.

Usted puede crear sus propias funciones y resolver problemas de forma independiente y modular. Observe como la siguiente función se encarga de calcular e imprimir un resultado. Las funciones en Spectrum pueden o no retornar un valor en este caso solo muestra el resultado.

```
1 // Funcion sin retorno en Spectrum
2 void calculateSum() {
3     int x = 5;
4     int y = 10;
5     int sum = x + y;
6     print("The sum of x + y is: ", sum);
7 }
8
9 int main() {
10    calculateSum(); // call the function
11    return 0;
12 }
13
```

Función en Spectrum con tipo de retorno y parámetros en su declaración.

Observe como ahora se muestra una función con tipo de retorno float y además debe recibir un argumento del mismo tipo para poder realizar la conversión de grados celsius a fahrenheit. Solo necesita llamar a la función en el main y pasar un argumento válido.



```
1 // Function to convert Fahrenheit to Celsius
2 float toCelsius(float fahrenheit) {
3     return (5.0 / 9.0) * (fahrenheit - 32.0);
4 }
5
6 int main() {
7     // Set a fahrenheit value
8     float f_value = 98.8;
9
10    // Call the function with the fahrenheit value
11    float result = toCelsius(f_value);
12
13    // Print the fahrenheit value
14    print("Fahrenheit: ", f_value);
15
16    // Print the result
17    print("Convert Fahrenheit to Celsius: ", result);
18
19    return 0;
20 }
```