Universidad Rafael Landívar
Laboratorio de Programación
Ingeniería Química
Catedrático: Ing. Edwin Chocoy

PROYECTO PRÁCTICO NO. 1
TRANSMETRO

Barrientos Mancilla, Jessica Valeria- 1097723
Ruiz Vásquez, Nadia María- 1140023

Ciudad de Guatemala, 18 de octubre de 2023

ÍNDICE

INTRODUCCIÓN

El presente trabajo es una compilación de habilidades en el ámbito de programación adquiridas a lo largo del curso, con el propósito de elaborar proyectos cuya aplicación recaiga en actividades cotidianas. Por ello, se desarrolló un programa para el sistema de Transmetro en Guatemala, donde los usuarios tengan la facilidad de comprar boletos, identificar las rutas, programar sus viajes considerando variables como el tiempo y la velocidad, entre otros factores relevantes.

De esta manera, tomando de referencia cuatro estaciones con sus códigos correspondientes, se verificará la información previa a la compra del boleto, como los códigos, las fechas, la edad y la condición de los pasajeros, las rutas y los precios del viaje.

Finalmente, se espera conseguir que los usuarios del Transmetro obtengan una herramienta digital para recopilar información sobre sus viajes, específicamente el tiempo total viajado y los gastos asociados, todo esto por medio de la simplificación con los códigos utilizados.

ANÁLISIS

ENTRADAS

Tabla No. 1: Entradas y descripciones

Entrada	Descripción
Código de entrada	El usuario debe ingresar información sobre el código de entrada de la estación a la cual abordará, el cual puede ser “51”, “61”, “71” o “82”, correspondientes a las estaciones “Javier”, “Trébol”, “Don Bosco” y “Plaza Municipal”.
Código de salida	El usuario debe ingresar información sobre el código de salida de la estación de destino, el cual puede ser “51”, “61”, “71” o “82”, correspondientes a las estaciones “Javier”, “Trébol”, “Don Bosco” y “Plaza Municipal”.
Fecha	El usuario debe ingresar la fecha en la cual abordará el Transmetro, colocando el día, el mes y por último el año.
Edad	El usuario debe ingresar su edad.
Embarazo	En caso de ser mujer, el usuario debe ingresar si está embarazada o no, por medio de las opciones “sí” o “no”.
Niños menores de 3 años	El usuario debe ingresar si le acompañará o comprará un boleto para un niño menor de 3 años, por medio de las opciones “sí” o “no”.

Fuente: Elaboración propia en base al “Proyecto Práctico No. 1”, (2023).

SALIDAS

Tabla No. 2: Salidas y descripciones

Salida	Descripción
Ruta	El sistema indicará los puntos que recorrerá la persona y los kilómetros de distancia; en caso de ingresar varias rutas, el sistema mostrará como salida la consideración o suma de todas ellas.
Tiempo	El sistema indicará el tiempo estimado de viaje, tras considerar las rutas y la velocidad. Se imprimirá en horas; en caso de ingresar varias rutas, el sistema mostrará como salida la consideración o suma del tiempo de todas ellas.
Precio	El sistema indicará el precio del boleto, tras considerar las restricciones, descuentos y demás fórmulas. Se imprimirá en

	quetzales; en caso de ingresar varias rutas, el sistema mostrará como salida la consideración o suma del total gastado en boletos.

Fuente: Elaboración propia en base al “Proyecto Práctico No. 1”, (2023).

PROCESOS

Tabla No. 3: Procesos y descripciones

Salida	**Descripción**
Ruta	El sistema debe generar la ruta que tomará en base del código de entrada con el código de salida, tomando de referencia los siguientes datos y los kilómetros a recorrer:

Estación partida	Estación destino	Distancia (km)
51	61	14
51	72	28
71	82	13
61	51	7
82	51	21

Precio	Las mujeres embarazadas o con niños menores a 3 años viajan gratis y todo ciudadano entre 15 y 25 años es considerado estudiante y tendrá un 21% de descuento; sobre el descuento, se tomará como un 79%, (100% - 21%), ya que esta es la cantidad exacta a pagar. Además, los primeros 10 km cuestan 7 centavos, cada km adicional cuesta 2 centavos, por lo que primero se estimarán los kilómetros recorridos.
Tiempo	Ya que todos los buses viajan a una velocidad constante de 40km/h, el programa realizó una conversión teniendo en cuenta la fórmula de tiempo, distancia y velocidad.

Fuente: Elaboración propia en base al “Proyecto Práctico No. 1”, (2023).

RESTRICCIONES

Tabla No. 4: Restricciones y descripciones

Restricciones	**Descripción**
Embarazo y niños menores de 3 años	Las mujeres embarazadas o con niños menores de 3 años viajan gratis, por lo que no se le aplicará todo el análisis para el precio de los boletos, únicamente las rutas y el tiempo.
Restricción	Todo ciudadano entre 15 y 25 años es considerado estudiante y

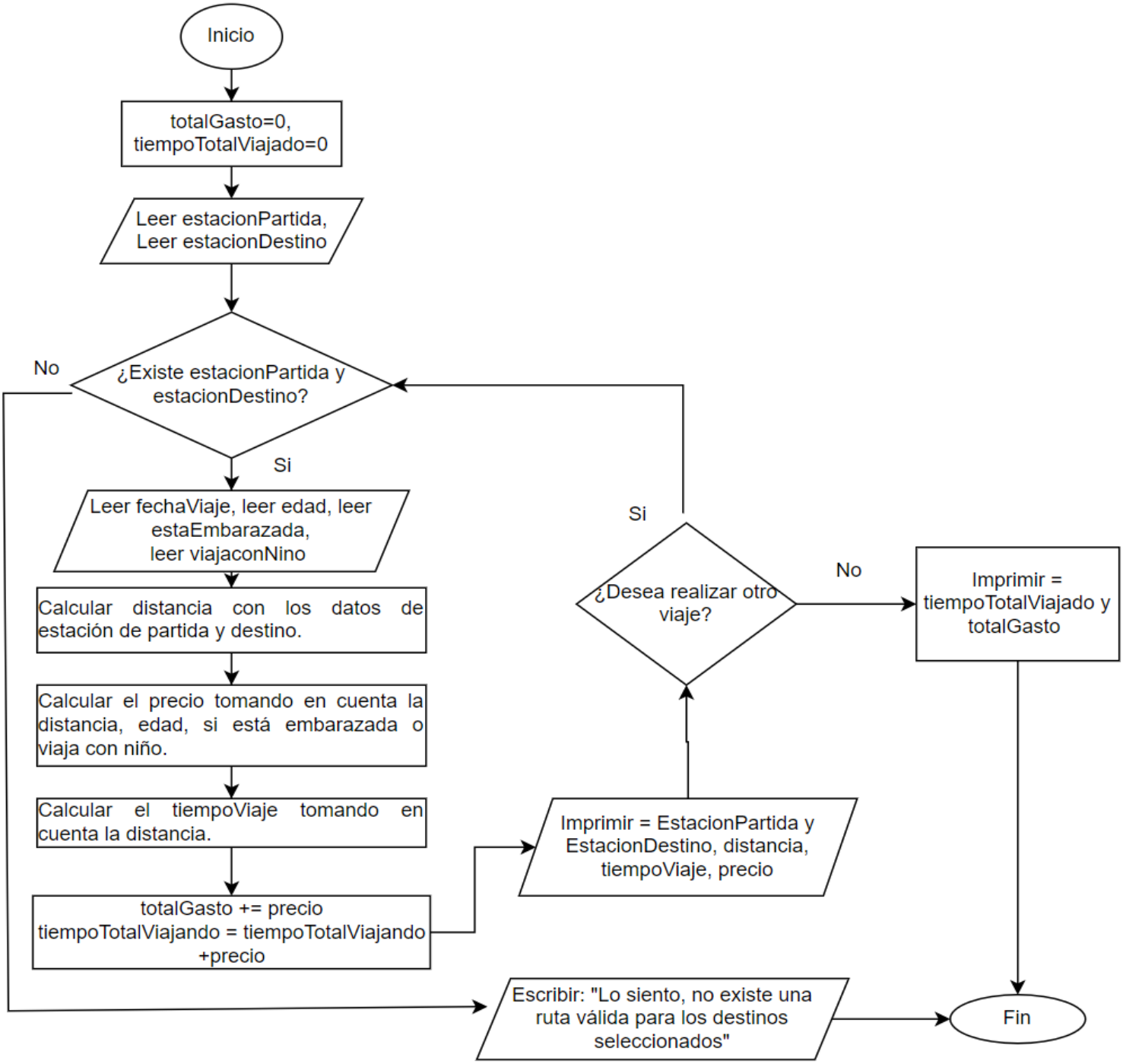
por edades de 15 a 25 años	tendrá un 21% de descuento en el precio, afectando su análisis y proceso.
Continuar travesía	Al completar el viaje el usuario puede continuar con sus rutas, por lo que el programa debe consultarlo. Si el usuario acepta, la estación de destino pasará a ser la estación de partida por lo que únicamente se solicitará la nueva estación de destino; en caso contrario, se imprimirán los datos finales de tiempo y total gastado.
Ingresar rutas erróneas	En caso de ingresar los códigos de rutas erróneas, el programa desplegará un mensaje de “Lo siento, no existe una ruta válida para los destinos ingresados.", impidiendo que el programa continúe.

Fuente: Elaboración propia en base al “Proyecto Práctico No. 1”, (2023)

DISEÑO

DIAGRAMA DE FLUJO

Diagrama No. 1: Código de transmetro



CONCLUSIONES

1. El uso de codificaciones permite que el usuario tenga información sobre cualquier ámbito de manera simplificada y sencilla, como fue el caso del presente trabajo con la información de los viajes de Transmetro, obteniendo datos como el tiempo recorrido o el total de gastos en boletos.

2. La realización de códigos en los que se utilicen ciclos y condiciones permite al programador simplificar y aprovechar el banco de memoria del programa, evitando errores por la cantidad de líneas codificadas.

3. La programación implica un papel importante en la resolución de problemas dentro de las actividades cotidianas, puesto que proporciona herramientas para automatizar tareas con eficiencia y soluciones específicas, facilitando la recopilación de información sin necesidad de la múltiple asistencia humana.

RECOMENDACIONES

1. Asegurarse que toda la información recopilada del usuario sea la necesaria para ejecutar el programa.

2. Ejecutar el programa varias veces para verificar que no exista ningún error y funcione correctamente.

3. Crear un diagrama de flujo para verificar que el programa cumpla con todos los requisitos solicitados.

REFERENCIAS

1. Microsoft (2023) *Array.GetLength(Int32) Método*
   https://learn.microsoft.com/es-es/dotnet/api/system.array.getlength?view=net-7.0
   Se utilizó para comprender el uso de GetLength y para establecerlo en la ubicación de datos en la matriz indicada.

2. Microsoft (2023) *CA1309: Utilizar StringComparison ordinal*
   https://learn.microsoft.com/es-es/dotnet/fundamentals/code-analysis/quality-rules/ca1309
   Se utilizó para utilizar un código que permitiera mejores resultados en las comparaciones para las variables del tipo bool.

3. Microsoft (2023) *Convert Class*
   https://learn.microsoft.com/en-us/dotnet/api/system.convert?view=net-7.0&redirectedfrom=MSDN
   Se utilizó para identificar los tipos de conversiones en el programa.

4. Microsoft (2023) *Crear un diagrama de flujo básico en Visio*
   https://support.microsoft.com/es-es/office/crear-un-diagrama-de-flujo-b%C3%A1sico-en-visio-e207d975-4a51-4bfa-a356-eeec314bd276
   Se utilizó para crear el diagrama de flujo del proyecto.

5. Microsoft (2023) *DateTime.Parse Method*
   https://learn.microsoft.com/en-us/dotnet/api/system.datetime.parse?view=net-7.0
   Se utilizó para identificar cómo colocar la fecha en el programa y el método para hacerlo.

6. Oregoom (2016) *Tipos de datos en C#*
   https://oregoom.com/c-sharp/tipos-de-datos/
   Se utilizó para identificar el tipo de dato en cada variable y cómo emplear “decimal” para los valores.

ANEXOS

MANUAL DEL USUARIO

Forma de uso

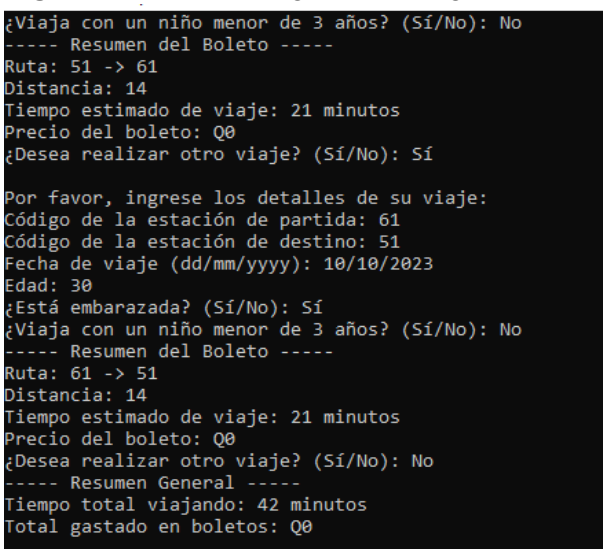
Para emplear el programa, primero se desplegarán las opciones de las rutas con sus correspondientes códigos al correrlo. Luego, pedirá al usuario colocar los detalles de su viaje:

1. Código de partida
2. Código de destino
3. La fecha del viaje, indicando primero el día, luego el mes y por último el año, separándolo con “/”.
4. La edad
5. Si la persona que realizará el viaje está embarazada, respondiendo con un “Sí” o con un “No”.
6. Si la persona que comprará el boleto viajará con un niño menor de 3 años, respondiendo con un “Sí” o con un “No”.

Tras esto, se desplegará el resumen del boleto, indicando la ruta, la distancia, el tiempo estimado de viaje y el precio del boleto. En caso de estar embarazada o viajar con un niño menor de 3 años, no se le cobrará; en caso de tener entre 15 y 25 años, se le realizará un descuento del 21%. En cuanto al tiempo, se consideró la velocidad del transmetro, 40 km/h, y los kilómetros recorridos, convirtiéndolo finalmente en minutos.

Agregado a esto, se le preguntará si desea realizar otro viaje. En caso de responder “Sí”, todo el proceso anterior se repetirá, desplegando de nuevo el resumen del boleto. Finalmente, se volverá a preguntar si desea realizar otro viaje; en caso de colocar “No”, el código acabará, desplegando el “Resumen General” con el tiempo total viajando y el total gastado en boletos, terminando así el programa.

Figura No. 2: Despliegue del código



Fuente: Elaboración propia en base a Visual Studio, (2023).

Códigos empleados

1. **Console WriteLine**
   Empleado para desplegar la información al correr el programa.

   Figura No. 3: ConsoleWriteLine

```
Console.WriteLine("Bienvenido al sistema de Trasmetro.");
Console.WriteLine("\nPrevio a comenzar, le recordamos las opciones de las rutas: ");
Console.WriteLine("Estación Javier: 51");
Console.WriteLine("Estación Trébol: 61");
Console.WriteLine("Estación Don Bosco: 71");
Console.WriteLine("Estación Plaza Municipal: 82");
```

   Fuente: Elaboración propia en base a Visual Studio, (2023).

2. **Console.ReadLine:**
   Empleado para que el usuario escriba información y se utilice en el análisis posterior dentro del código.

   Figura No. 4: ConsoleReadLine

```
Console.Write("Edad: ");
edad = int.Parse(Console.ReadLine());
```

   Fuente: Elaboración propia en base a Visual Studio, (2023).

3. **Tipos de datos:**
   - **Decimal:** se utilizó el tipo “decimal” para la variable “totalGasto” ya que el precio por 10 km era de 7 centavos (Q 0.07), por lo que se requería de un tipo de dato que permitiera el uso de decimales sin aproximaciones y con exactitud de pocas cifras después del punto. Además, al agregar la "m" al final de los valores literales, se indica que son del tipo decimal.

     Figura No. 5: Decimal

```
decimal precioBase = 0.07m;
```

     Fuente: Elaboración propia en base a Visual Studio, (2023).

   - **int:** se utilizaron los datos del tipo entero para valores exactos, tales como el número de estaciones o la distancia recorrida y la edad.

     Figura No. 6: Int

```
int estacionPartida;
int estacionDestino;
int edad;
```

     Fuente: Elaboración propia en base a Visual Studio, (2023).

- **bool:** se emplearon para las preguntas de embarazo o de la presencia de niños menores de 3 años, ya que se igualaron las respuestas de “Sí” y “No” con true y false para continuar con el programa.

Figura No. 7: Bool

```
bool estaEmbarazada;
bool viajaConNino;
```

Fuente: Elaboración propia en base a Visual Studio, (2023).

4. **Do y While:**

Parte importante del código. Se empleó “Do” para indicar que el código corriera al menos una vez y luego se repitiera mientras se cumpliera con la condición.

Luego, en la sección de “While”, indica que el bucle continuará ejecutándose mientras el texto ingresado para la pregunta de “Realizar otro viaje” sea igual a "Sí" (se especificará el uso de tal código más adelante). Al ser un texto diferente a "Sí", el bucle se detendrá.

Figura No. 8: Do

```
do
{
    int estacionPartida;
    int estacionDestino;
    int edad;
    bool estaEmbarazada;
    bool viajaConNino;

    Console.WriteLine("\nPor favor, ingrese los detalles de su viaje:");

    Console.Write("Código de la estación de partida: ");
    estacionPartida = int.Parse(Console.ReadLine());
```

Fuente: Elaboración propia en base a Visual Studio, (2023).

Figura No. 9: While

```
    Console.Write("¿Desea realizar otro viaje? (Sí/No): ");
} while (Console.ReadLine().Equals("Sí", StringComparison.OrdinalIgnoreCase));
```

Fuente: Elaboración propia en base a Visual Studio, (2023).

5. **int.Parse:** se utilizó para realizar conversiones cuando se recibía información por parte del usuario y se requería del tipo entero para su análisis dentro del programa.

   Figura No. 10: While

```
Console.Write("Código de la estación de destino: ");
estacionDestino = int.Parse(Console.ReadLine());
```

   Fuente: Elaboración propia en base a Visual Studio, (2023).

6. **if, "continue":** se empleó un if dentro del programa en caso de que el usuario ingrese una ruta inexistente. Para ello, se utilizó como condición o argumento "!ExisteRuta(estacionPartida, estacionDestino)", empleando el signo ! para indicar que si no existe tal ruta, mostrara el siguiente mensaje: "Lo siento, no existe una ruta válida para los destinos ingresados."

   Además, se utilizó el código de "continue", el cual se utiliza para omitir cualquier código que siga inmediatamente después en el bucle actual y que continúe con la siguiente parte del bucle.

   Figura No. 11: Continue

```
if (!ExisteRuta(estacionPartida, estacionDestino))
{
    Console.WriteLine("Lo siento, no existe una ruta válida para los destinos ingresados.");
    continue;
}
```

   Fuente: Elaboración propia en base a Visual Studio, (2023).

7. **DateTime.Parse:** según bibliografía consultada, el DateTime.Parse es un método que se utiliza para convertir una cadena de texto que representa una fecha.

   Figura No. 12: Tiempo

```
Console.Write("Fecha de viaje (dd/mm/yyyy): ");
DateTime fechaViaje = DateTime.Parse(Console.ReadLine());
```

   Fuente: Elaboración propia en base a Visual Studio, (2023).

8. **Equals("Sí", StringComparison.OrdinalIgnoreCase);**
   Para esta parte del código se buscó en bibliografía y con consultas a personas conocedoras de programación, ya que al ejecutar el código se encontró el posible error de que, cuando el usuario conteste a las preguntas de "Sí" y "No" escriba erróneamente cualquiera de las dos opciones. Es por ello que se recomendó el uso de StringComparison.OrdinalIgnoreCase, el cual indica que se compara el texto ingresado por el usuario con la cadena preestablecida "Sí", ignorando la diferencia entre mayúsculas, minúsculas y tildes. De esta manera, si se ingresa "Sí", "sí", "SI", "si", la comparación seguirá siendo "true".

De igual forma, el código funciona colocando únicamente **Equals("Sí")**, pero considerando como “true” únicamente si el usuario escribe el texto tal y como se indica.

Figura No. 13: StringComparison

```
Console.Write("¿Está embarazada? (Sí/No): ");
estaEmbarazada = Console.ReadLine().Equals("Sí", StringComparison.OrdinalIgnoreCase);
```

Fuente: Elaboración propia en base a Visual Studio, (2023).

9. **+=**

Se empleó para indicar una suma entre la variable de la izquierda más la variable de la derecha.

Figura No. 14: Sumas

```
totalGasto += precio;
```

Fuente: Elaboración propia en base a Visual Studio, (2023).

10. **Matrices**

Se emplearon matrices para almacenar las rutas de manera eficiente, permitiendo buscar información entre estaciones y distancias. Así, para el presente código, cada fila representa una ruta, la primera columna representa el código de la estación de partida, la segunda columna representa el código de la estación de destino y la tercera columna representa la distancia en kilómetros.

Especificando los códigos posteriores a la matriz, por ejemplo, al utilizar routes[0, 2] dará la distancia entre las estaciones de la primera ruta, ya que el 0 corresponde al primer valor de la estación de entrada.

Figura No. 15: Matrices

```
int[,] distancias = {
    { 51, 61, 14 }, { 51, 72, 28 }, { 71, 82, 13 }, { 61, 51, 7 }, { 82, 51, 21 }
};

for (int i = 0; i < distancias.GetLength(0); i++)
{
    if ((distancias[i, 0] == estacionPartida && distancias[i, 1] == estacionDestino) ||
        (distancias[i, 0] == estacionDestino && distancias[i, 1] == estacionPartida))
    {
        return distancias[i, 2];
    }
}
```

Fuente: Elaboración propia en base a Visual Studio, (2023).

11. ||

Símbolo que significa el operador “or”, indicando que la condición general se cumple si cualquiera de las dos condiciones es verdadera. Es por ello que, para la presente sección del código, se utilizó para indicar que si el código de partida y el código de salida corresponden a los establecidos en las matrices se considera la condición como verdadera, siendo válido también para el orden inverso por si el usuario aplica para la opción de “Realizar otro viaje”.

Figura No. 16: OR

```
if ((rutas[i, 0] == estacionPartida && rutas[i, 1] == estacionDestino) ||
    (rutas[i, 0] == estacionDestino && rutas[i, 1] == estacionPartida))
{
    return true;
}
```

Fuente: Elaboración propia en base a Visual Studio, (2023).

12. For y GetLength(0)

Para empezar, este método se empleó dentro del bucle for, el cual se utiliza para recorrer las filas de la matriz previamente indicada. Luego, se indicó un “int i = 0”, donde se le da el valor 0 a la variable i, utilizándose como contador en el bucle y para indicar la posición en la matriz.

Tras esto, se empleó i < rutas.GetLength(0), la cual es una condición que determina que el bucle continuará si i es menor que rutas.GetLength(0). Por último, se empleó i++, el cual aumenta el valor de i en 1 en cada repetición del bucle, continuando hasta que i sea igual al número de filas en la matriz rutas.

GetLength indica la longitud (cantidad de elementos) en la matriz, es decir, el número de filas en la matriz rutas. De esta forma, con rutas.GetLength(0) se obtiene la longitud de la primera dimensión de la matriz (el número de filas).

Figura No. 17: For

```
for (int i = 0; i < distancias.GetLength(0); i++)
```

Fuente: Elaboración propia en base a Visual Studio, (2023).