



## INFORME DEL PROYECTO:

---

### PROGRAMA PARA OPTIMIZAR EL TRÁFICO MEDIANTE RUTAS ALTERNAS Y ATAJOS EN LA CIUDAD DEL CUSCO

**Integrantes:**

Paccohuanca Alvarez Luis Miguel

Paucar Cazas Gary Elliot

Achahuanco Bayona David

Mattaque Huillcas Juan Abelardo

**Docente:**

QUISPE CCAPACCA, EDGAR

**Curso:**

Estructura de Datos

Cusco - Perú  
2025

# Informe del Proyecto: Simulador de Tráfico con Rutas Alternas en la Ciudad del Cusco

## Análisis del Problema

### 1. Descripción del Problema

En la ciudad del Cusco, se presentan altos niveles de congestión vehicular, sobre todo en avenidas muy transitadas como El Sol, La Cultura y las calles del Centro Histórico. Estas zonas concentran gran parte del flujo vehicular, tanto de transporte público como de vehículos particulares, especialmente en las horas punta. Esta situación genera demoras en los desplazamientos, aumenta la contaminación ambiental y afecta la calidad de vida de los ciudadanos. La saturación de las vías principales se debe, en gran parte, a la falta de planificación de rutas alternativas y al incremento constante de vehículos particulares y turísticos. Ante esta situación, se hace necesario buscar soluciones tecnológicas que contribuyan a mejorar la eficiencia del tránsito en la ciudad.

El problema principal radica en la falta de un sistema de gestión del tráfico que oriente a los conductores hacia rutas alternas o vías secundarias que ayuden a descongestionar las avenidas principales. Frente a ello, se propone desarrollar un programa en C++ que permita simular un sistema de control inteligente del tráfico, ofreciendo rutas alternativas que optimicen el desplazamiento urbano. Este tipo de solución tecnológica permite visualizar cómo la informática puede aplicarse a problemas reales.

### 2. Requerimientos del Sistema

Requerimientos Funcionales:

- Permitir ingresar la cantidad de intersecciones o puntos del mapa.
- Ingresar las distancias entre los puntos principales.
- Solicitar un punto de partida y calcular la mejor ruta hacia un destino.
- Mostrar la ruta más corta y las rutas alternas disponibles.
- Simular los tiempos de viaje según las rutas seleccionadas.

Requerimientos No Funcionales:

- El sistema debe ejecutarse correctamente en Dev-C++.
- Debe ser fácil de entender para los estudiantes.
- Los cálculos deben realizarse de forma rápida y precisa.
- El código debe estar bien estructurado y comentado.
- Debe permitir la simulación de tráfico urbano en zonas con alta densidad vehicular.

### 3. Estructuras de Datos Propuestas

Para el desarrollo del simulador se propone el uso de una matriz de adyacencia, donde cada posición representa la distancia entre dos puntos o intersecciones. Además, se emplearán arreglos auxiliares para registrar las distancias mínimas y los nodos visitados durante la ejecución del algoritmo de búsqueda.

### 4. Justificación de la Elección

Las matrices son estructuras simples de implementar y comprender, ideales para proyectos de nivel intermedio. Permiten acceder fácilmente a los datos y facilitan la aplicación de algoritmos como Dijkstra, que resulta muy útil para calcular rutas más cortas entre varios puntos. Por ello, esta elección es adecuada para los objetivos del proyecto.

## Diseño de la Solución

### 1. Descripción de Estructuras de Datos y Operaciones

El programa usará una matriz de tamaño  $n \times n$ , donde  $n$  es el número de intersecciones ingresadas por el usuario. Las operaciones principales serán: ingresar los valores de la matriz, procesar la información mediante el algoritmo de Dijkstra y mostrar el camino más corto junto con posibles rutas alternas.

### 2. Algoritmos Principales

El algoritmo principal será el de Dijkstra, que permite encontrar la ruta más corta desde un punto inicial hacia los demás puntos del mapa. Su aplicación es ideal para este tipo de simulaciones, ya que muestra de manera clara cómo se puede optimizar el tránsito al evitar las vías más congestionadas.

### 3. Diagrama de Flujo

1. Inicio.
2. Ingresar cantidad de puntos.
3. Ingresar matriz de distancias.
4. Seleccionar punto de inicio.
5. Ejecutar algoritmo de Dijkstra.
6. Mostrar la ruta más corta.
7. Mostrar rutas alternas.
8. Fin.

### 4. Justificación del Diseño

El diseño se centra en la claridad y simplicidad. Se busca que los estudiantes comprendan cómo los algoritmos de búsqueda pueden aplicarse a problemas reales como el tráfico urbano. Además, al realizar el proyecto, se puede avanzar de forma organizada: en primera para el análisis y diseño, la segunda para codificación, y la tercera para pruebas y presentación.

## Conclusión

El desarrollo del programa en Dev-C++ demuestra cómo los algoritmos y las estructuras de datos pueden aplicarse para resolver problemas reales. La simulación de un sistema de tráfico inteligente en la ciudad del Cusco muestra el potencial de la informática para mejorar la movilidad urbana, reducir la congestión y contribuir al bienestar de la población.

## Primer diseño:

```
=====
      SIMULADOR DE TRAFICO - CUSCO (Preliminar)
=====
1. Iniciar simulaci3n
2. Mostrar rutas actuales
3. Salir
Seleccione una opci3n: 1

Ingrese la cantidad de ciclos de simulaci3n: 2

===== CICLO DE SIMULACI3N 1 =====
Analizando niveles de trbfico...

-----
Ruta principal: Av. El Sol
Distancia: 3.5 km
Nivel de trbfico: BAJO
Tiempo estimado: 4.2 minutos
? Trbfico fluido. Puede continuar por Av. El Sol.

-----
Ruta principal: Av. La Cultura (UNSAAC)
Distancia: 4.2 km
Nivel de trbfico: ALTO
Tiempo estimado: 14.7 minutos
?? Trbfico alto detectado. Se recomienda tomar:
  - Alterna 1: Circunvalacion
  - Alterna 2: Av. Los Incas
```

Nivel de tráfico: BAJO  
Tiempo estimado: 3.0 minutos  
? Tráfico fluido. Puede continuar por Wanchaq.

Fin del ciclo 2. Actualizando datos...

=====  
SIMULADOR DE TRAFICO - CUSCO (Preliminar)  
=====

1. Iniciar simulación  
2. Mostrar rutas actuales  
3. Salir  
Seleccione una opción: 2

===== LISTADO DE RUTAS =====

Ruta Principal	Distancia(km)	Trafico	Tiempo(min)
Av. El Sol	3.5	Medio	7.0
Av. La Cultura (UNSAAC)	4.2	Alto	14.7
Av. De La Paz	2.8	Alto	9.8
Av. Tullumayo	3.9	Medio	7.8
Wanchaq	2.5	Bajo	3.0

=====  
SIMULADOR DE TRAFICO - CUSCO (Preliminar)  
=====

1. Iniciar simulación  
2. Mostrar rutas actuales  
3. Salir  
Seleccione una opción: |

```

1 // =====
2 // PROYECTO: Simulador de Tráfico - Cusco
3 // Descripción: Este programa simula el tráfico en
4 // diferentes rutas del Cusco y recomienda rutas alternas
5 // en base a niveles aleatorios de congestión.
6 // =====
7
8 #include <iostream>
9 #include <string>
10 #include <windows.h> // Para Sleep()
11 #include <cstdlib> // Para rand(), srand()
12 #include <ctime> // Para time()
13 #include <iomanip> // Para formato de salida
14
15 using namespace std;
16
17 // =====
18 // ESTRUCTURAS DE DATOS
19 // =====
20 struct Ruta {
21     string principal;
22     string alterna1;
23     string alterna2;
24     float distancia; // en kilómetros
25     int trafico; // 0 = bajo, 1 = medio, 2 = alto
26     float tiempo; // tiempo estimado en minutos
27 };
28
29 // =====
30 // FUNCIONES
31 // =====
32
33 // Genera un nivel de tráfico aleatorio y actualiza el tiempo estimado
34 void actualizarTrafico(Ruta &r) {
35     r.trafico = rand() % 3; // 0, 1 o 2
36     switch (r.trafico) {
37         case 0: r.tiempo = r.distancia * 1.2; break; // tráfico fluido
38         case 1: r.tiempo = r.distancia * 2.0; break; // tráfico medio
39         case 2: r.tiempo = r.distancia * 3.5; break; // tráfico alto
40     }
41 }

```

```

42
43 // Muestra una recomendación según el nivel de tráfico
44 void mostrarRecomendacion(const Ruta &r) {
45     cout << "\n-----\n";
46     cout << "Ruta principal: " << r.principal << endl;
47     cout << "Distancia: " << fixed << setprecision(1) << r.distancia << " km" << endl;
48     cout << "Nivel de tráfico: ";
49
50     if (r.trafico == 0) cout << "BAJO\n";
51     else if (r.trafico == 1) cout << "MEDIO\n";
52     else cout << "ALTO\n";
53
54     cout << "Tiempo estimado: " << fixed << setprecision(1) << r.tiempo << " minutos\n";
55
56     // Recomendaciones según tráfico
57     if (r.trafico == 2) {
58         cout << "?? Tráfico alto detectado. Se recomienda tomar:\n";
59         cout << " - Alternativa 1: " << r.alterna1 << "\n";
60         cout << " - Alternativa 2: " << r.alterna2 << "\n";
61     } else if (r.trafico == 1) {
62         cout << "?? Tráfico medio. Puede considerar usar " << r.alterna1 << " para reducir tiempo.\n";
63     } else {
64         cout << "?? Tráfico fluido. Puede continuar por " << r.principal << ".\n";
65     }
66 }
67
68 // Muestra todas las rutas con sus datos actuales
69 void mostrarTodasLasRutas(Ruta rutas[], int n) {
70     cout << "\n===== LISTADO DE RUTAS =====\n";
71     cout << left << setw(25) << "Ruta Principal"
72         << setw(15) << "Distancia(km)"
73         << setw(12) << "Tráfico"
74         << setw(15) << "Tiempo(min)\n";
75     cout << "\n";
76     for (int i = 0; i < n; i++) {
77         string nivel;
78         if (rutas[i].trafico == 0) nivel = "Bajo";
79         else if (rutas[i].trafico == 1) nivel = "Medio";
80         else nivel = "Alto";
81         cout << left << setw(25) << rutas[i].principal
82             << setw(15) << rutas[i].distancia
83             << setw(12) << nivel
84             << setw(15) << fixed << setprecision(1) << rutas[i].tiempo << endl;
85     }
86 }
87

```

```

87
88 // Simula varios ciclos de análisis de tráfico
89 void ejecutarSimulacion(Ruta rutas[], int n, int ciclos) {
90     for (int i = 1; i <= ciclos; i++) {
91         cout << "\n===== CICLO DE SIMULACIÓN " << i << " =====\n";
92         cout << "Analizando niveles de tráfico...\n";
93         Sleep(1000);
94
95         for (int j = 0; j < n; j++) {
96             actualizarTrafico(rutas[j]);
97             mostrarRecomendacion(rutas[j]);
98             Sleep(700);
99         }
100
101         cout << "\nFin del ciclo " << i << ". Actualizando datos...\n";
102         Sleep(1500);
103     }
104 }
105
106 // =====
107 // FUNCIÓN PRINCIPAL
108 // =====
109 int main() {
110     srand((unsigned)time(0));
111
112     const int NUM_RUTAS = 5;
113     Ruta rutas[NUM_RUTAS] = {
114         {"Av. El Sol", "Av. La Cultura", "Av. Tullumayo", 3.5, 0, 0},
115         {"Av. La Cultura (UNSAAC)", "Circunvalacion", "Av. Los Incas", 4.2, 0, 0},
116         {"Av. De La Paz", "Av. Los Angeles", "Av. 28 de Julio", 2.8, 0, 0},
117         {"Av. Tullumayo", "Av. Collasuyo", "Av. Pachacutec", 3.9, 0, 0},
118         {"Wanchaq", "Av. Garcilaso", "Av. Velasco Astete", 2.5, 0, 0}
119     };
120
121     int opcion, ciclos;
122
123     do {
124         cout << "\n===== \n";
125         cout << "    SIMULADOR DE TRÁFICO - CUSCO (Preliminar)\n";
126         cout << "===== \n";
127         cout << "1. Iniciar simulación\n";
128         cout << "2. Mostrar rutas actuales\n";
129         cout << "3. Salir\n";
130         cout << "Seleccione una opción: ";
131         cin >> opcion;

```

```

120
121 int opcion, ciclos;
122
123 do {
124     cout << "\n===== \n";
125     cout << "    SIMULADOR DE TRÁFICO - CUSCO (Preliminar)\n";
126     cout << "===== \n";
127     cout << "1. Iniciar simulación\n";
128     cout << "2. Mostrar rutas actuales\n";
129     cout << "3. Salir\n";
130     cout << "Seleccione una opción: ";
131     cin >> opcion;
132
133     switch (opcion) {
134         case 1:
135             cout << "\nIngrese la cantidad de ciclos de simulación: ";
136             cin >> ciclos;
137             ejecutarSimulacion(rutas, NUM_RUTAS, ciclos);
138             break;
139         case 2:
140             mostrarTodasLasRutas(rutas, NUM_RUTAS);
141             break;
142         case 3:
143             cout << "\nSaliendo del simulador...\n";
144             break;
145         default:
146             cout << "\nOpción inválida. Intente nuevamente.\n";
147     }
148
149     } while (opcion != 3);
150
151     cout << "\n? Simulación finalizada.\n";
152     return 0;
153 }
154

```

