

1. Por cada camioneta: tiempo en cada etapa, ID de la camioneta, impresión matricial o 0 si no cumplió

Este bloque de código se encarga de imprimir una tabla que muestra el tiempo que cada camioneta ha tomado para completar cada etapa del rally. Si una camioneta no completó una etapa, se muestra un 0 en lugar del tiempo. La tabla tiene columnas para el ID de la camioneta y cada una de las etapas.

- Se inicia un bucle **for** que recorre todas las camionetas (i de 0 a CAM).
- Dentro del bucle, se utiliza otro bucle **for** para recorrer cada etapa (j de 0 a ETA).
- Se imprime el ID de la camioneta con formato (setw(11)) para asegurarse de que ocupe 11 caracteres en la salida.
- Luego, se imprime el tiempo tomado por esa camioneta en la etapa actual con un formato similar.
- Este proceso se repite para todas las etapas de todas las camionetas.

```
cout << endl;
cout << "ID Camioneta - Etapa 0 - Etapa 1 - Etapa 2 - Etapa 3 - Etapa 4"
<< endl;
cout << "-----" <<
endl;

for (i = 0; i < CAM; i++) {
    cout << setw(11) << idCam[i] << " - ";
    for (j = 0; j < ETA; j++) {
        cout << setw(7) << matRally[i][j] << " - ";
    }
    cout << endl;
}
```

2. Por cada camioneta, indicador que informe si tardó (-, =, +) en cada etapa

En esta sección, se muestra una tabla similar a la anterior, pero en lugar de mostrar los tiempos reales, muestra un indicador (-, =, +) que indica si la camioneta tardó menos (-), igual (=) o más (+) que el tiempo esperado en cada etapa.

- Se utiliza un bucle **for** anidado similar al anterior para recorrer todas las camionetas y etapas.
- Se calcula el indicador comparando el tiempo de la camioneta en esa etapa con el tiempo esperado (tiempoEt[j]).
- Luego, se imprime el indicador en la tabla.

```
cout << endl;
cout << "ID Camioneta - Etapa 0 - Etapa 1 - Etapa 2 - Etapa 3 - Etapa 4"
<< endl;
cout << "-----" <<
```

```
endl;

for (i = 0; i < CAM; i++) {
    cout << setw(11) << idCam[i] << " - ";
    for (j = 0; j < ETA; j++) {
        if (matRally[i][j] < tiempoEt[j]) {
            indicador = '-';
        } else if (matRally[i][j] > tiempoEt[j]) {
            indicador = '+';
        } else {
            indicador = '=';
        }
        cout << setw(7) << indicador << " - ";
    }
    cout << endl;
}
```

3. Por cada camioneta, la cantidad de etapas finalizadas

En esta parte, se calcula y muestra cuántas etapas ha finalizado cada camioneta. Si una camioneta tiene al menos un tiempo registrado en una etapa, se considera que ha finalizado esa etapa.

- Se utiliza un bucle **for** que recorre todas las camionetas (i de 0 a CAM).
- Dentro del bucle, se utiliza otro bucle **for** para recorrer todas las etapas (j de 0 a ETA).
- Si se encuentra un tiempo registrado en una etapa (`matRally[i][j] > 0`), se incrementa el contador `contEtFin[i]` para esa camioneta.
- Luego, se imprime el resultado, mostrando cuántas etapas ha finalizado cada camioneta.

```
cout << endl;
cout << "Cantidad de etapas finalizadas por camioneta: " << endl;
for (i = 0; i < CAM; i++) {
    for (j = 0; j < ETA; j++) {
        if (matRally[i][j] > 0) {
            contEtFin[i]++;
        }
    }
    cout << "ID Camioneta " << idCam[i] << " - Finalizó " << contEtFin[i]
    << " etapas" << endl;
}
```

4. Máximo y mínimo en cada etapa

En esta parte, se encuentra y muestra el tiempo máximo y mínimo registrado en cada etapa.

- Se utilizan dos bucles `for`, uno para encontrar el máximo y otro para encontrar el mínimo en cada etapa.
- En el bucle del máximo, se inicializa `maxEt` en 0 y se compara el tiempo de cada camioneta en la etapa actual con el tiempo de la camioneta que tiene el máximo tiempo (`matRally[maxEt][j]`). Si se encuentra un tiempo mayor, se actualiza `maxEt` con el índice de esa camioneta.
- Se repite un proceso similar en el bucle del mínimo para encontrar el tiempo mínimo en cada etapa.

```
// Max
for (j = 0; j < ETA; j++) {
    int maxEt = 0;
    for (i = 0; i < CAM; i++) {
        if (matRally[i][j] > matRally[maxEt][j]) {
            maxEt = i;
        }
    }
    cout << "Etapa " << j << ": Máximo tiempo - ID Camioneta " <<
    idCam[maxEt] << " - " << matRally[maxEt][j] << " minutos" << endl;
}

// Min
for (j = 0; j < ETA; j++) {
    int minEt = 0;
    for (i = 0; i < CAM; i++) {
        if (matRally[i][j] < matRally[minEt][j]) {
            minEt = i;
        }
    }
    cout << "Etapa " << j << ": Mínimo tiempo - ID Camioneta " <<
    idCam[minEt] << " - " << matRally[minEt][j] << " minutos" << endl;
}
```

5. Promedio de tiempos, solo para vehículos que finalizaron

En esta sección, se calcula y muestra el promedio de tiempos para cada etapa, pero solo para las camionetas que han finalizado cada etapa.

- Se utiliza un bucle `for` para recorrer todas las etapas (`j` de 0 a `ETA`).
- Dentro del bucle, se utilizan otros bucles para contar cuántas camionetas han finalizado esa etapa (`cont`) y sumar los tiempos de todas las camionetas que lo hicieron (`acum`).
- Luego, se calcula y muestra el promedio de tiempos para esa etapa, pero solo si al menos una camioneta la ha finalizado (para evitar divisiones por cero).
- Esas son las explicaciones detalladas de cómo funcionan esas partes específicas del código. Cada una de ellas se encarga de generar información relevante sobre el rendimiento de las camionetas en cada etapa del rally.

```
for (j = 0; j < ETA; j++) {  
    int cont = 0;  
    int acum = 0;  
    for (i = 0; i < CAM; i++) {  
        if (matRally[i][j] > 0) {  
            cont++;  
            acum += matRally[i][j];  
        }  
    }  
    if (cont > 0) {  
        cout << "Promedio de tiempos para etapa " << j << ": " << (acum /  
cont) << " minutos" << endl;  
    }  
}
```