Casas y Pueblos S71653_es

Tenemos un conjunto de casas situadas a lo largo de una carretera, en posiciones enteras (que son como la distancia desde el inicio de la carretera hasta la casa). Un **pueblo** es una subsecuencia máxima de 2 casas o más, consecutivas en la carretera, que están a menos de 3 unidades de distancia entre sí. Por esta definición, los pueblos no se pueden solapar entre sí. (A veces también hay casas solas, porque están a una distancia de 3 o más de sus casas vecinas).

Por ejemplo, consideremos las siguientes casas en la carretera:

Nombre	`Can_Sola`	`Can_Verdet`	`Cal_Quer`	`Ca_n_Aurelia`	`Can_Roca`
Posición	0	1	4	8	10

En este caso, hay 2 pueblos: `Can_Sola` y `Can_Verdet` por un lado; y `Ca\_n\_Aurelia` y `Can_Roca` por el otro. `Cal_Quer` no forma parte de ningún pueblo.

Debes hacer una **función** `detecta_pobles` con la siguiente declaración:

```
struct Casa {
    string nom;
    int pos;
};

struct Poble {
    int inici, fi;
    int num_cases;
};

/**
 * @brief Detecta los pueblos de la carretera.
 *
 * @pre   El vector `cases` está ordenado por nombre.
 * @post  El resultado está ordenado por posición.
 */
vector<Poble> detecta_pobles(vector<Casa>& cases);
```

El vector `cases` contiene las casas, pero éstas vienen ordenadas por *nombre*. El resultado debe ser un vector de pueblos (`vector<Poble>`), con todos los pueblos (usando la definición) en el orden en que se encuentran en la carretera (es decir, ordenados por posición).

Observación

El icono de nombre '.CPP' contiene el programa principal para hacer las pruebas.

Sólo debes enviar un fichero que contenga la función requerida, con los `include` necesarios y las funciones auxiliares que hayas declarado (si hay), y **nada más**.

No se puede usar la función `sort` de `C++`. Si quieres ordenar un vector, debes implementarlo la ordenación tú (puedes usar cualquier algoritmo).

Para poder poner las estructuras en la solución, hace falta copiar y pegar el siguiente código (que evita que salga un error que dice que la declaración de `Casa` y `Poble` está duplicada):

```
#ifndef TIPUS
#define TIPUS

struct Casa {
    string nom;
    int pos;
};

struct Poble {
    int inici, fi;
    int num_cases;
};

#endif
```

Este código también se encuentra en el fichero `main.cc` que te damos.

Entrada

La entrada ya la procesa el programa principal que se da (icono .CPP). La entrada consiste en varias secuencias de casas, en donde cada una empieza con un entero que indica cuántas casas hay en la carretera, y después siguen las casas ordenadas por nombre.

Salida

La salida ya la produce el programa principal que se da (icono .CPP). Para cada secuencia de casas, se muestran los pueblos detectados, cada uno con la posición inicial, la final, y el número de casas.

Ejemplo de entrada 1

```
10
Ca_n_Aurelia 8
Cal_Mestre 14
Cal_Quer 7
Can_Pla 13
Can_Roca 20
Can_Sola 0
Can_Torrent 3
Can_Verdet 2
Can_Vila 25
Can_Xic 30
```

Ejemplo de salida 1

```
___
0 3 3
7 8 2
13 14 2
```

Información del problema

Autoría: Pau Fernández

Generación: 2026-01-25T12:47:50.384Z