0.1 – Planificación de horarios (B)

Estructuras de Datos
Facultad de Informática - UCM

Supongamos que queremos planificar las actividades de un congreso. Cada actividad tiene una duración diferente. Para almacenar la duración de cada una de ellas utilizamos el siguiente tipo de datos:

```
using Duracion = int;
```

De este modo, una duración se expresa mediante un número entero que indica una cantidad de segundos determinada. Si no conoces la palabra clave `using` en C++, posiblemente sí conozcas `typedef`. La declaración anterior es equivalente a lo siguiente:

```
typedef int Duracion;
```

Por otro lado, el congreso también tiene una duración máxima, y queremos saber si hay tiempo suficiente para poder abarcar todas las actividades propuestas. Supongamos que tenemos una lista con las duraciones de cada una de estas actividades, y que también sabemos la duración del congreso.

Implementa la siguiente función:

```
bool caben_todas(
    const vector<Duracion> &duracion_actividades,
    const Duracion &duracion_congreso
    );
```

Esta función recibe el listado con la duración de las actividades, la duración máxima del congreso y debe devolver `true` si el tiempo ocupado por todas las actividades es menor o igual que la duración del congreso, o `false` en caso contrario.

Solución

```
#include <iostream>
#include <vector>

using namespace std;

using Duracion = int;


Duracion sumar_duracion(const Duracion &t1, const Duracion &t2) {
  return t1 + t2;
}

bool duracion_menor_que(const Duracion &t1, const Duracion &t2) {
  return t1 < t2;
}

Duracion duracion_cero() {
  return 0;
}

bool caben_todas(const vector<Duracion> &duracion_actividades,
                 const Duracion &duracion_congreso) {

  Duracion actual = duracion_cero();

  for (int i = 0; i < duracion_actividades.size(); i++) {
    actual = sumar_duracion(actual, duracion_actividades[i]);
  }

  return duracion_menor_que(actual, duracion_congreso);
}
```