PROGRAMACIÓN 2. Curso 2017-18. Grupo de tarde
1ª prueba voluntaria de evaluación

Esta es la primera prueba voluntaria que se plantea en la asignatura ***Programación 2*** para la evaluación de los alumnos matriculados en el grupo de tarde. Tiene un valor de **10 puntos**. Debe ser resuelta individualmente y deberá ser presentada **antes de las 24 horas del 8 de abril de 2018** a través de la plataforma **Moodle2**.

```
/*
 * Pre:  n >= 0  AND n < #v   AND (PT alfa  EN [0,n-1]. v[ alfa ]  = Vo[ alfa ])    AND
 *       (PT alfa  EN [0,n-2]. v[ alfa ]  <= v[alfa+1])
 * Post:  (PT alfa  EN [0,n-1]. Vo[ alfa ]  < nuevo -> Vo[alfa] = v[ alfa ]   AND
 *                              Vo[ alfa ]  >= nuevo -> Vo[alfa] = v[alfa+1])   AND
 *        v[INS] = nuevo   AND INS >= 0  AND INS <= n AND
 *        (PT alfa  EN [0,INS-1]. v[ alfa ]  < nuevo)
 */
template <typename Dato>
void  insertar  (Dato v [],  const  int  n,  const  Dato nuevo);

/*
 * Pre:  n = K  AND n >= 0  AND n <= #v  AND (PT alfa  EN [0,n-1]. v[ alfa ]  = Vo[ alfa ])
 * Post:  (PT alfa  EN [0,K-1]. (EX beta  EN [0,n-1]. Vo[ alfa ]  = v[beta ])  )   AND
 *        (PT alfa  EN [0,n-1]. (EX beta  EN [0,K-1]. v[ alfa ]  = Vo[beta ])  )   AND
 *        (PT alfa  EN [0,n-1]. (NUM beta EN [0,n-1]. v[beta] = v[ alfa ])  = 1)
 */
template <typename Dato>
void  purgar  (Dato v [],  int& n);
```

En esta prueba se pide diseñar, sin programar ningún bucle, las funciones **insertar**`(v,n,nuevo)` y **purgar**`(v,n)` que acaban de ser especificadas, teniendo en cuenta lo siguiente:

- El diseño sin bucles de cada una de las funciones anteriores se abordará mediante inmersión, únicamente en el caso de que realizar dicha inmersión sea imprescindible.
- En el caso de que sea necesario proceder a una inmersión para diseñar sin bucles una o las dos funciones anteriores, entonces al calificar sus diseños se tendrá fundamentalmente en cuenta la adecuada especificación y el correcto diseño de cada una de las funciones auxiliares en las que se apoye el código de las funciones pedidas.
- El diseño sin bucles de la función **insertar**`(v,n,nuevo)` se calificará sobre 3 puntos y el diseño sin bucles de la función **purgar**`(v,n)` se calificará sobre 7 puntos. Para obtener alguno de los puntos asignados a cada uno de los dos diseños es condición necesaria que la función pedida se comporte al ser ejecutada de acuerdo con su especificación.

Como resultado de esta prueba se presentará a través de la plataforma **Moodle2** un único fichero que contenga exclusivamente el código **C++** de las dos funciones pedidas junto, en su caso, con el código de las funciones auxiliares en las que se apoya el diseño de cada una de ellas. En el fichero no deben incluirse otros elementos (cláusulas `#include`, función **main**`()` de un posible programa de prueba u otras funciones auxiliares utilizadas para realizar pruebas pero que no constituyan una parte del diseño de las dos funciones pedidas).

No se admitirán trabajos que no se presenten a través de la plataforma **Moodle2**, ni trabajos presentados fuera de plazo, ni trabajos plagiados total o parcialmente.

Una solución de los problemas de diseño planteados en la 1ª prueba voluntaria.

Es posible plantear un diseño sin bucles de la función **`insertar`**`(v,n,nuevo)` sin necesidad de realizar una inmersión.

```
/*
 * Pre: n >= 0 AND n < #v AND (PT alfa EN [0,n-1]. v[alfa] = Vo[alfa]) AND
 *      (PT alfa EN [0,n-2]. v[alfa] <= v[alfa+1])
 * Post: (PT alfa EN [0,n-1]. Vo[alfa] < nuevo -> Vo[alfa] = v[alfa] AND
 *                            Vo[alfa] >= nuevo -> Vo[alfa] = v[alfa+1]) AND
 *       v[INS] = nuevo AND INS >= 0 AND INS <= n AND
 *       (PT alfa EN [0,INS-1]. v[alfa] < nuevo)
 */
template <typename Dato>
void insertar (Dato v[], const int n, const Dato nuevo) {
    if (n > 0) {
        // n > 0
        if (v[n-1] <= nuevo) {
            v[n] = nuevo;
        }
        else {
            v[n] = v[n-1];
            insertar (v, n-1, nuevo);
        }
    }
    else {
        // n = 0
        v[0] = nuevo;
    }
}
```

Se ha realizado un diseño sin bucles de la función **`purgar`**`(v,n)`. La función **`esta`**`(v,d,hasta)` facilita el diseño del código de la función anterior, discriminando qué elementos se deben ***purgar*** del vector **`v`**.

```
/*
 * Pre: hasta < #v
 * Post: esta(v, d, hasta) = (EX alfa EN [0,hasta]. v[alfa] = d)
 */
template <typename Dato>
bool esta (const Dato v[], const Dato d, const int hasta) {
    if (hasta >= 0) {
        if (d == v[hasta]) {
            return true;
        }
        else {
            return esta(v, d, hasta - 1);
        }
    }
    else {
        return false;
    }
}

/*
 * Pre: n = K AND n >= 0 AND n <= #v AND (PT alfa EN [0,n-1]. v[alfa] = Vo[alfa])
 * Post: (PT alfa EN [0,K-1]. (EX beta EN [0,n-1]. Vo[alfa] = v[beta]) ) AND
 *       (PT alfa EN [0,n-1]. (EX beta EN [0,K-1]. v[alfa] = Vo[beta]) ) AND
 *       (PT alfa EN [0,n-1]. (NUM beta EN [0,n-1]. v[beta] = v[alfa]) = 1)
 */
template <typename Dato>
void purgar (Dato v[], int& n) {
    if (n > 1) {
        if (esta(v, v[n-1], n - 2)) {
            // v[n-1] hay que 'purgarlo' ya que está repetido en v[0..n-2]
            n = n - 1;
            purgar(v, n);
        }
        else {
            // v[n-1] hay que 'memorizarlo' ya que es único en v[0..n-1]
            Dato ultimo = v[n-1];
            // Se va a purgar el resto del vector, es decir, v[0..n-2]
            n = n - 1;
            purgar(v, n);
            // Se repone el dato memorizado previamente
            v[n] = ultimo;
            n = n + 1;
        }
    }
}
```