Escuela de Ingeniería y Arquitectura - Depto. de Informática e Ingeniería de Sistemas
Examen práctico de Programación 2 - 11 de septiembre de 2015 - Tiempo: hasta las 14:00

Este examen práctico individual en laboratorio forma parte de la evaluación de la asignatura. Su calificación tiene un peso del 30 % en la calificación final de la asignatura en la convocatoria de septiembre.

El trabajo a desarrollar

Se está desarrollando un programa en `C++`. En él se han definido la constante `N` y el tipo de dato `Vector`.

Se pide diseñar, sin bucles, las dos funciones **mezclarV01** `(v1, v2, v)` y **mezclarV02** `(v1, v2, v)` que se especifican en la parte posterior de esta hoja. Ambas funciones, que han presentan un comportamiento idéntico, van a diferir en la forma que han de ser diseñadas.

- La función **mezclarV01** `(v1, v2, v)` ha de diseñarse sin bucles, tanto ella como sus funciones auxiliares, y éstas han de diseñarse aplicando la técnica de inmersión mediante refuerzo de la precondición de la primera función. [5.0 puntos]
- La función **mezclarV02** `(v1, v2, v)` ha de diseñarse sin bucles, tanto ella como sus funciones auxiliares, y éstas han de diseñarse aplicando la técnica de inmersión mediante debilitamiento de la postcondición de la primera. [5.0 puntos]

En ambos diseños se exige que el comportamiento de cada función corresponda al especificado y que todas las funciones que intervengan en el diseño estén adecuadamente especificadas.

Presentación del trabajo

Como resultado del trabajo se entregará un único fichero con el código `C++` desarrollado cuyas primeras líneas sean un comentario con el nombre del alumno y, a continuación, almacene el código de las funciones **mezclarV01** `(v1, v2, v)` y **mezclarV02** `(v1, v2, v)` y de sus funciones auxiliares, dispuestas todas ellas en el orden que corresponda dentro del programa. La entrega se hará a través de la plataforma ***Moodle2*** (moodle2.unizar.es) antes de las 14:00. Se recomienda no hacerlo hasta que haya sido verificado de forma exhaustiva que el comportamiento de todo el código diseñado es correcto.

```
const int N = 200;   // Número máx. de elementos almacenados en un dato de tipo Vector

struct Vector {
    int nD;   // número de datos almacenados con nD>=0 y nD<=N
    int D[N]; // D [0],..., D[nD-1] son los datos del vector
};

/*
 * Predicados que se definen para facilitar la escritura de especificaciones :
 *
 * ORDENADOS(v,n) = (PT alfa EN [0,n-2].v.D[alfa]<=v.D[alfa+1])
 *
 * MEZCLADOS(v1,n1,v2,n2,v) =
 *                 (v.nD = n1 + n2) AND
 *                 (PT alfa EN [0, n1+n2-1].
 *                         (NUM beta IN [0, n1+n2-1].v.D[beta]=v.D[alfa])
 *                         =
 *                         (NUM beta IN [0, n1-1].v1.D[beta]=v.D[alfa])
 *                         +
 *                         (NUM beta IN [0, n2-1].v2.D[beta]=v.D[alfa]))
 */

/*
 * Pre:  ORDENADOS(v1,v1.nD) AND ORDENADOS(v2,v2.nD) AND v1.nD + v2.nD <= N
 * Post: MEZCLADOS(v1,v1.nD,v2,v2.nD,v) AND ORDENADOS(v,v.nD)
 */
void mezclarV01(const Vector& v1, const Vector& v2, Vector& v);

/*
 * Pre:  ORDENADOS(v1,v1.nD) AND ORDENADOS(v2,v2.nD) AND v1.nD + v2.nD <= N
 * Post: MEZCLADOS(v1,v1.nD,v2,v2.nD,v) AND ORDENADOS(v,v.nD)
 */
void mezclarV02(const Vector& v1, const Vector& v2, Vector& v);
```

Una solución de los problemas propuestos

```
const int N = 200;   // Número máx. de elementos almacenados en un dato de tipo Vector

struct Vector {
    int nD;     // número de datos almacenados con nD>=0 y nD<=N
    int D[N];   // D[0],..., D[nD-1] son los datos del vector
};

/*
 * Predicados que se definen para facilitar la escritura de especificaciones :
 *
 * ORDENADOS(v,n) = (PT alfa EN [0,n-2].v.D[alfa]<=v.D[alfa+1])
 * MEZCLADOS(v1,n1,v2,n2,v) =
 *                 (v.nD = n1 + n2) AND
 *                 (PT alfa EN [0, n1+n2-1].
 *                         (NUM beta IN [0, n1+n2-1].v.D[beta]=v.D[alfa])
 *                         =
 *                         (NUM beta IN [0, n1-1].v1.D[beta]=v.D[alfa])
 *                         +
 *                         (NUM beta IN [0, n2-1].v2.D[beta]=v.D[alfa]))
 */

/*
 * Pre:  ORDENADOS(v1,v1.nD) AND ORDENADOS(v2,v2.nD) AND v1.nD + v2.nD <= N AND
 *       MEZCLADOS(v1,n1,v2,n2,v) AND ORDENADOS(v,v.nD) AND
 *       n1 >= 0 AND n1 <= v1.nD AND n2 >= 0 AND n2 <= v2.nD
 * Post: MEZCLADOS(v1,v1.nD,v2,v2.nD,v) AND ORDENADOS(v,v.nD)
 */
void mezclarI01(const Vector& v1, const Vector& v2, Vector& v, int n1, int n2) {
    if ((n1<v1.nD) || (n2<v2.nD)){ // quedan datos por mezclar
        if (n1==v1.nD) { // todos los datos de v1 han sido mezclados
            v.D[v.nD] = v2.D[n2];  n2 = n2 + 1;
        }
        else if (n2==v2.nD){ // todos los datos de v2 han sido mezclados
            v.D[v.nD] = v1.D[n1];  n1 = n1 + 1;
        }
        else if (v1.D[n1]<v2.D[n2]) { // quedan por mezclar datos de v1 y de v2
            v.D[v.nD] = v1.D[n1];  n1 = n1 + 1;
        }
        else{ //v1.D[i1]>=v2.D[i2]
            v.D[v.nD] = v2.D[n2];  n2 = n2 + 1;
        }
        v.nD = v.nD + 1;            // ya que se ha incorporado un nuevo dato a v
        mezclarI01(v1, v2, v, n1, n2);
    }
}

/*
 * Pre:  ORDENADOS(v1,v1.nD) AND ORDENADOS(v2,v2.nD) AND v1.nD + v2.nD <= N
 * Post: MEZCLADOS(v1,v1.nD,v2,v2.nD,v) AND ORDENADOS(v,v.nD)
 */
void mezclarV01(const Vector& v1, const Vector& v2, Vector& v) {
    v.nD = 0;
    mezclarI01(v1, v2, v, 0, 0);
}
```

```
/*
 * Pre: ORDENADOS(v1,v1.nD) AND ORDENADOS(v2,v2.nD) AND v1.nD + v2.nD <= N AND
 *      n1 >= 0 AND n1 <= v1.nD AND n2 >= 0 AND n2 <= v2.nD
 * Post: MEZCLADOS(v1,n1,v2,n2,v) AND ORDENADOS(v,v.nD)
 */
void mezclarI02(const Vector& v1, const Vector& v2, Vector& v, int n1, int n2) {
    if ((n1>0) || (n2>0)) { // quedan datos por mezclar
        if (n1==0){ // no quedan datos por mezclar de v1
            mezclarI02(v1,v2,v,n1,n2-1);
            v.D[v.nD] = v2.D[n2-1];
        }
        else if (n2==0) {// no quedan datos por mezclar de v2
            mezclarI02(v1,v2,v,n1-1,n2);
            v.D[v.nD] = v1.D[n1-1];
        }
        else if (v1.D[n1-1]>=v2.D[n2-1]){
            // quedan datos por mezclar de v1 y de v2
            mezclarI02(v1,v2,v,n1-1,n2);
            v.D[v.nD] = v1.D[n1-1];
        }
        else{ // v1.D[n1-1]<v2.D[n2-1]
            // quedan datos por mezclar de v1 y de v2
            mezclarI02(v1,v2,v,n1,n2-1);
            v.D[v.nD] = v2.D[n2-1];
        }
        v.nD = v.nD + 1;      // ya que se ha incorporado un nuevo dato a v
    }
}

/*
 * Pre:  ORDENADOS(v1,v1.nD) AND ORDENADOS(v2,v2.nD) AND v1.nD + v2.nD <= N
 * Post: MEZCLADOS(v1,v1.nD,v2,v2.nD,v) AND ORDENADOS(v,v.nD)
 */
void mezclarV02(const Vector& v1, const Vector& v2, Vector& v){
    v.nD = 0;
    mezclarI02(v1, v2, v, v1.nD, v2.nD);
}
```