WUOLAH


Indalecia
www.wuolah.com/student/Indalecia

56805

# EsquemaSolucion2018Enero.pdf

*ENERO 2018 (ORDINARIA) + SOLUCION*

**1° Fundamentos de Programación**

**Grado en Ingeniería Informática**

**Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación**
**Universidad de Granada**

Pregunta 1. Asignaci□n de t□cnicos a pedidos.

Esquema del algoritmo:
   t -> t□cnico
   Usamos un vector de bool ya_asignados

   const int MAX_COSTE = 100
   Inicializar X a cero
   Inicializar ya_asignados a false

   for (t
      pos_min = Posici□n del m□nimo de la fila t (inicializar min a MAX_COSTE +
1)
         Saltar valor si ya_asignado[p]
      ya_asignado[pos_min] = true
      X[t][pos_min] = 1
      coste = coste + B[t][pos_min]

Fallos "imperdonables":
- Limitarse a calcular el m□nimo de cada fila, sin tener en cuenta si el pedido
   ya hab□a sido asignado previamente.
- Modificar la matriz de costes B.
   Por el hecho de calcular unas asignaciones y un costo total, no podemos
   modificar la matriz de costes que contiene unos datos vitales de nuestro
programa.

Fallos importantes:
- Inicializar min al primer elemento de cada fila.
   Hay que inicializarlo o bien al primer elemento no asignado previamente de
cada fila,
   o mucho m□s f□cil, inicializarlo a MAX_COSTE + 1

Fallos menores:
- Dimensionar un vector con una variable (este fallo lo detecta el compilador)

==========
====
Pregunta 2. Subcadena contenida d□bilmente en otra.

Cabecera:
   bool Contiene(SecuenciaCaracteres a_buscar)

Usemos las siguientes abreviaciones:

util_a_buscar = a_buscar.TotalUtilizados()
util: es el dato miembro total_utilizados
vector_privado -> v_p


a) Recorriendo primero los caracteres de la secuencia principal (el vector
privado v_p)

   Esquema del Algoritmo:

   for (... i < util && !encontrada
      if (v_p[i] == a_buscar.Elemento(j))
         j++
      encontrada = j == util_a_buscar

b) Recorriendo primero los caracteres de a_buscar

   Esquema del Algoritmo:

   for (... i < util_a_buscar && !caracter_enc

```
      ultima_pos_enc = PrimeraOcurrenciaEntre(ultima_pos_enc + 1,
                                              util - 1,
                                              a_buscar.Elemento(i))
      caracter_enc = ultima_pos_enc == -1;


Fallos "imperdonables":
- Pasar dos par@metros al m@todo:
   bool Contiene(SecuenciaCaracteres ppal, SecuenciaCaractere a_buscar)

Fallos importantes:
- Usar un vector o secuencia auxiliar, local al m@todo.
   Duplica memoria y es totalmente innecesario
- No salir del bucle cuando un car@cter de a_buscar no se encuentra en v_p
- No salir del bucle cuando se ha encontrado la secuencia completa a_buscar
- En el caso b), reiniciar la b@squeda dentro de a_buscar desde j=0
- Usar return dentro de los bucles -> hay que usar un bool

Fallos menores:
- Acceder a a_buscar[i]. Debe ser: a_buscar.Elemento(i)


================================================================================
====
Pregunta 3. ToASCII

Cabecera:
   TablaRectangularCaracteres ToASCII()

Esquema del Algoritmo:
   matriz_privada -> m_p

   TablaRectangularCaracteres ToASCII(){
      TablaRectangularCaracteres tab;
      SecuenciaCaracteres sec;

      for (i
         for (j
            dato = m_p[i][j]

            if (dato < 70)
               car = '@'
            else if (dato < 130)
               car = '&'
            ...
            sec.Aniade(car)

         tab.Aniade(sec)
         sec.EliminaTodos()

      return tab
   }

Fallos "imperdonables":
- Pasar una tabla de enteros como par@metro al m@todo:
   TablaRectangularCaracteres ToASCII(TablaRectangularEnteros tabla) :-(
   La llamada al m@todo debe ser as@:
      convertida = tabla.ToASCII();  :-)
   Si se hace mal, quedar@a as@:
      convertida = esta_tabla_no_sirve_de_nada.TOASCII(tabla);  :-(
- Realizar asignaciones del tipo
   tabla.Elemento(i,j) = valor
   El m@todo Elemento devuelve un valor. No es un lvalue
- Acceder a los datos miembro privados de la tabla de caracteres
```

```
   El m@todo pertenece a la clase TablaRectangularEnteros y el objeto local que
el m@todo
   devuelve es de la clase TablaRectangularCaracteres, por lo que no son del
mismo tipo
   y no se puede acceder a sus datos miembro privados.

Fallos importantes:
- Repetir c@digo:
   if (valor >= 70 && valor < 130)
      ...
   else if (valor >= 130 && valor < 170)
      ...
   Se repiten los literales, 70, 130, 170, etc
   Las expresiones (valor < 130) y (valor >= 130) son mutuamente excluyentes
- No incluir else en cada if

Fallos menores:
- No ejecutar sec.EliminaTodos()
- Declarar el objeto sec dentro del bucle.
   Conlleva una recarga innecesaria de crear y destruir el
   objeto cada vez que se realiza una iteraci@n.
   Eso s@, ya no ser@a necesario ejecutar sec.EliminaTodos()


================================================================================
====
Pregunta 4. Intervalo

Dise@o de la clase:

"Conjunto vac@o" es un estado correcto de un Intervalo.
Es como una cantidad de dinero a cero.
Para identificarlo, tenemos varias alternativas:

- Usar un bool es_vacio como dato miembro
   Es correcto, pero para que no haya "contradicciones" habr@a que
   tener el cuidado de asignarle true siempre que los extremos se
   asignen a valores imposibles como por ejemplo (3,3) [3,1] etc
- Si usamos el tipo double para los extremos, le podr@amos asignar NAN
   para representar el intervalo vac@o
   El m@todo EsVacio() comprobar@a si los extremos son NAN (con isnan)
- Lo m@s f@cil ser@a que el m@todo EsVacio() compruebe que los valores
   de los extremos sean correctos, en cuyo caso no es vac@o. False en otro caso.
   En el constructor por defecto (sin par@metros) establecer@amos los
   extremos a unos valores concretos, como por ejemplo izda = 1, dcha = 0

Para identificar el tipo de los extremos (abierto/cerrado), podemos
analizar c@mo quedar@a el programa principal.
Ser@a muy c@modo y legible tener el c@digo siguiente:

int main(){
   Intervalo ejemplo('[',0,1,']');        // [0,1]
   Intervalo otro_ejemplo('(',0,1,']');   // (0,1]

El hecho de que en el main pasemos un char no significa que usemos
un char como dato miembro, ya que luego, las comparaciones con estos valores
ser@an propensas a errores. Como datos miembros, usaremos dos bool
del tipo abierto_izda, abierto_dcha.

Incluso, mucho mejor ser@a lo siguiente:

int main(){
   Intervalo ejemplo("[0,1]");   // [0,1]
```

aunque esto conllevar�a procesar el string dentro del constructor
y se sale de la dificultad pedida en este ejercicio.

Tambi�n podr�amos haber usado directamente dos bool en el constructor:

```
int main(){
   Intervalo ejemplo(false,0,1,false);   // [0,1]
```

Esta aproximaci�n tiene el inconveniente de que hay que mirar la
cabecera del constructor para saber si false corresponde a
abierto o a cerrado.

Si se ha optado por esta soluci�n (con dos bool en el constructor),
no se penaliza ning�n punto.

En lo que sigue, se usa == para denotar igualdad.
Si los tipos de dato son double, hay que usar una funci�n global
   bool EsIgual(double uno, double otro)

```
class Intervalo{
private:
   bool abierto_izda, abierto_dcha
   double izda, dcha
public:
   Intervalo(){   // Intervalo vac�o
      izda = 1
      dcha = 0
   }
   Intervalo(char tipo_izda, double cota_izda, double cota_dcha, char tipo_dcha)
{
      Asignar datos miembros a par�metros:
      abierto_izda = tipo_izda == '('
            // NUNCA m�s se vuelve a usar en la clase Intervalo el literal '('
            // Si necesitamos saber si est� abierto llamaremos a AbiertoIzda()

      izda = cota_izda
      ...
   }
   bool EsVacio(){
      return izda > dcha   //  4,3 es vac�o ya sea abierto o cerrado en
cualquier extremo
            ||
            (izda==dcha && (abierto_izda || abierto_dcha))  // (3,3] [3,3)
(3,3) son vac�os
   }

   // M�todos "Get";
   bool AbiertoIzda() { return abierto_izda; }
   bool AbiertoDcha() ...
   double CotaIzda() ...
   double CotaDcha() ...

   // Contiene un punto
   bool Contiene(double valor){
      return (izda < valor && valor < dcha)
            ||
            (izda == valor && !abierto_izda)
            ||
            (dcha == valor && !abierto_dcha)
   }

   // Contiene otro intervalo
   bool Contiene(Intervalo otro){
      ...
```

```
   }
};
```

Fallos "imperdonables":

- No tener en cuenta si es abierto o cerrado por los dos extremos
- Usar un vector como dato miembro para almacenar expl�citamente una serie de valores
- Devolver un char 's' / 'n' en vez de un bool en el m�todo Pertenece
- Devolver un char '(' / '[' en vez de un bool en el m�todo AbiertoIzda
- Pasar dos par�metros al m�todo que comprueba si un intervalo est� dentro de otro:
   bool Contiene(Intervalo uno, Intervalo otro)   :-((
- Usar el tipo char para los datos miembro abierto_izda, abierto_dcha

Fallos importantes:
- No definir el m�todo EsVacio()
- No definir los constructores
- Dentro del m�todo Contiene, repetir expresiones del tipo
   izda < valor

Fallos menores:
- Considerar que [a,a] es vac�o
- Usar como nombre "bool estado_izda" en vez de "abierto_izda" o "cerrado_izda"
- Repetir el c�digo de la expresi�n de terminaci�n del bucle en el main.