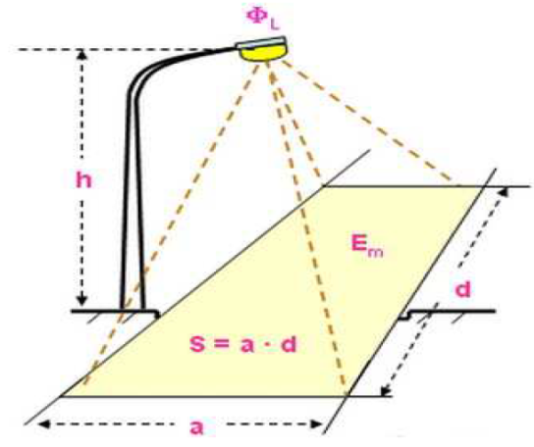
**Fundamentos de Programación (8 de junio de 2020)**

**(Grados en Ingeniería Mecánica, Eléctrica, Electrónica Industrial y Química Industrial)**

**Ejercicio 1 – Programación Estructurada (2 puntos):** construir un programa en ***C*** para calcular la distancia **d** de separación adecuada entre las luminarias de un alumbrado público de una vía urbana de manera que se garantice un nivel de iluminancia medio determinado (**Em**).



El programa solicitará inicialmente por teclado la iluminancia media sobre la calzada (entre 10 y 20 lux) que queremos conseguir, la anchura de la calzada (entre 3 y 15 m) y la altura de montaje prevista (entre 3 y 15 m). Seguidamente solicitará por teclado el flujo luminoso de la lámpara en función de la altura prevista (ver tabla de la derecha).

A continuación, se calculará el factor de utilización η a partir de la gráfica y la distancia de separación entre luminarias. Se ha de comprobar que el resultado obtenido está dentro de los límites indicados en la siguiente tabla:

|  |  |
| --- | --- |
| **Iluminancia media (lux)** | **Separación/altura** |
| 2≤ Em<7 | 4≤d/h<5 |
| 7≤ Em <15 | 3.5≤d/h<4 |
| 15≤ Em <30 | 2≤d/h<3.5 |

En caso afirmativo habremos acabado y se presentarán en pantalla los resultados. De lo contrario, el programa presentará un mensaje en pantalla y preguntará al usuario si quiere probar con otro valor del flujo de la lámpara, repitiéndose de nuevo el proceso de cálculo si la respuesta es afirmativa o finalizándose la ejecución si es negativa.

Em: iluminancia media sobre la calzada (lux) 1lux=1lm/m2

η: factor de utilización de la instalación

fm: factor de mantenimiento (considerar un valor de 0.70 fijo)

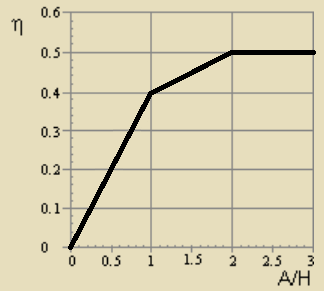
ΦL: flujo luminoso de la lámpara (lm)

a: anchura a iluminar de la calzada (m)

d: separación entre luminarias (m)

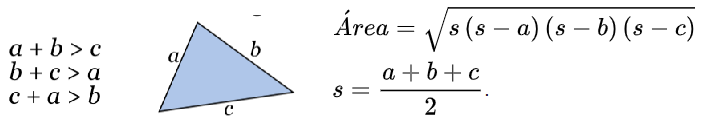
h: altura de montaje (m)

|  |  |
| --- | --- |
| **Flujo de la lámpara (lm)** | **Altura (m)** |
| ΦL<3000 | h<6 |
| 3000≤ ΦL<10000 | 6≤h<8 |
| 10000≤ ΦL<20000 | 8≤h<10 |
| 20000≤ ΦL<40000 | 10≤h<12 |
| 40000≤ ΦL<100000 | ≥12 |



**Ejercicio 2 – Diseño Modular (2 puntos): 2.1)** construir una función en ***C*** para determinar la pieza plana triangular con mayor relación área/perímetro, conocidas las longitudes de dos de sus lados en cm. La función devolverá la longitud del tercer lado de la pieza que maximiza dicha relación como parámetro de salida, y también devolverá el valor de dicha relación a través del identificador de la función. La longitud del tercer lado se determinará con una precisión de 1mm. Considerar el siguiente prototipo:

double area\_per(double a, double b, double \*c);



**2.2)** Construir un programa en ***C*** que presente en pantalla una tabla con las longitudes del tercer lado que maximizan la relación área/perímetro, para longitudes de los otros lados comprendidas entre 5 cm y 50 cm, equi-espaciadas a intervalos de 5 cm.

**Ejercicio 3 – Estructuras de Datos (2 puntos):** Considerar las siguientes estructuras de datos para representar una lista de circuitos lógicos combinatorios. Cada circuito está formado por un máximo de 8 entradas lógicas (cada una admite una señal digital binaria: 0/1), un conjunto de puertas lógicas elementales y una salida lógica.

#define MAX\_C 10 // Nº máximo de circuitos combinatorios

#define MAX\_E 8 // Nº máximo de entradas lógicas

#define MAX 256 // Nº máximo de filas tabla de verdad (2^n)

typedef char tipo\_mapa[MAX/2][MAX/2+MAX%2];

typedef char tipo\_tabla[MAX];

typedef struct{

int ne; // Nº de entradas lógicas

tipo\_tabla tv; // Tabla de verdad

}tipo\_cc;

typedef tipo\_cc tipo\_vector[MAX\_C];

typedef struct{

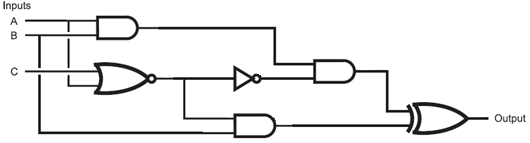
int n; // Nº de circuitos combinatorios

tipo\_vector v; // Vector de circuitos combinatorios

}tipo\_lista;

|  |  |
| --- | --- |
| **I** | **O** |
| 0 | '0' |
| 1 | '0' |
| 2 | '1' |
| 3 | '0' |
| 4 | '0' |
| 5 | '0' |
| 6 | '1' |
| 7 | '1' |

|  |  |  |  |
| --- | --- | --- | --- |
| **Inputs** | | | **Output** |
| **A** | **B** | **C** |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |



Construir las dos funciones siguientes en C con el prototipo especificado:

|  |  |
| --- | --- |
| void eliminar(tipo\_lista \*l); | Eliminar de la lista todos los circuitos combinatorios en los que todas sus salidas sean iguales (bien 0 o bien 1). |
| void mapa\_karnaugh(tipo\_cc c, tipo\_mapa m); | Devolver el mapa de Karnaugh asociado a un circuito lógico combinatorio que se le pasa como argumento. |

Nota: El mapa de Karnaugh consiste en una representación bidimensional de la tabla de verdad, donde los valores de las entradas lógicas se agrupan por filas y columnas, y cada celda de la matriz se corresponde con la salida lógica resultante. Los agrupamientos de las variables no están escritos de forma ordenada 00, 01, 10, 11 en binario: cada agrupamiento difiere del anterior en un solo bit (código Grey). Ejemplo con 5 entradas lógicas:

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | | | **ABC** | | | | | | | |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|  | 000 | 001 | 011 | 010 | 110 | 111 | 101 | 100 |
| **DE** | | 0 | 00 | '1' | '0' | '0' | '1' | '0' | '0' | '1' | '1' |
| 1 | 01 | '1' | '1' | '1' | '1' | '0' | '1' | '0' | '1' |
| 2 | 11 | '1' | '0' | '1' | '1' | '1' | '0' | '1' | '0' |
| 3 | 10 | '0' | '0' | '1' | '0' | '1' | '1' | '1' | '1' |

int grey(int n){

/\* Devuelve el código Grey (entero) de un

entero no negativo \*/

return(n^(n/2));

}

**Ejercicio 4 – Archivos (2 puntos):** Un nuevo aparato de diagnóstico es capaz de generar un archivo de texto con la secuencia genética del ARN de un virus extraída a partir de una muestra de un paciente. La máquina puede trabajar a un ritmo de 1000 tests por ciclo de trabajo, y los nombres de los archivos generados van desde "**test000**" hasta "**test999**". La secuencia genética consiste en un mensaje escrito con combinaciones de cuatro letras (**A**,**G**,**C**,**U**). Cada una de ellas es la inicial de un compuesto químico con diferentes cantidades de carbono, hidrógeno, nitrógeno y oxígeno (**A**: adenina , **G**: guanina, **C**: citosina, **U**: uracilo). La gran novedad del SARS-CoV-2 respecto a otros coronavirus es la aparición de 12 letras extra insertadas en su genoma. Los expertos señalan a esta brevísima secuencia como principal culpable de su contagiosidad y virulencia: **ccucggcgggca**

Construir un programa lo más modular posible, atendiendo a criterios de modularidad, que analice los 1000 archivos de texto generados en un ciclo de trabajo de la máquina en busca de una ocurrencia de la secuencia de esas doce letras, y que grabe en un archivo de texto de nombre "ResultadosTest.txt" los números de los tests que han dado resultado positivo y negativo. Ej:

"Test000" **cugcccucggcgggcaccgu...** Positivo

"Test015" **cugcccuggcggcaccgu......** Negativo

"Test999" **cugcccugcggcaccgu........** Negativo