IMPORTANTE: Debe adjuntar este documento en Schoology dentro de Examen 2, incluyendo su nombre y los códigos en C como respuestas a las preguntas. Note que el examen es individual y que debe apegarse a los códigos de Ética e Integridad Académica del Tecnológico de Monterrey, por lo que exámenes similares serán anulados.

EJERCICIO:

Su empresa debe desarrollar un Sistema Automatizado de Identificación de Huellas Digitales (AFIS) para instituciones federales como INE, SAT e INM. Los AFIS utilizan representaciones de huellas digitales basadas en minucias (puntos en las crestas donde se rompe la continuidad).

Estos puntos se pueden representar con cuatro atributos: 1) coordenada horizontal de la imagen (entero en el intervalo [0, 1023]); 2) coordenada vertical de la imagen (entero en el intervalo [0, 1023]); 3) dirección de las crestas (entero en el intervalo [0, 359]); 4) tipo de minucia (valor entero sin signo que solo toma en cuenta los dos bits más significativos).

Debe implementar lo siguiente para modelar minucias y realizar operaciones con ellas:

**1 (15 puntos).** La estructura con alias Minucia tiene cuatro campos. Los campos x e y son enteros sin signo de 16 bits. Un campo ángulo que es un número de coma flotante de 32 bits. Un campo tipo que es un entero sin signo de 8 bits donde solo importan los dos bits más significativos para representar uno de los cuatro valores posibles: Terminación (00), División (01), Punto (10) y Desconocido (11).

typedef struct

{

    uint16\_t x;

    uint16\_t y;

    double angulo;

    uint8\_t tipo;

} Minucia;

**2 (15 puntos).** La estructura con alias ArregloMinucias tiene dos miembros. Un arreglo de minucias que se representan con un puntero a Minucia. Y la longitud del arreglo (entero sin signo de 16 bits).

typedef struct

{

    Minucia\* arreglo;

    uint16\_t longitud;

} ArregloMinucias;

**3 (15 puntos).** La función crearMinucia crea dinámicamente una minucia a partir de la información pasada como parámetros (posiciones x e y, ángulo y tipo) y devuelve un puntero a la minucia creada.

Minucia\* crearMinucia(uint16\_t x, uint16\_t y, double angulo, uint8\_t tipo)

{

    Minucia \*mi = (Minucia\*)malloc(sizeof(Minucia));

    mi->x=x;

    mi->y=y;

    mi->angulo=angulo;

    mi->tipo=tipo;

    return mi;

}

**4 (15 puntos).** La función crearArregloMinucias crea dinámicamente (asignando memoria) un arreglo con el número de minucias especificadas por parámetro. Inicializa cada minucia con los campos en valor cero. Y devuelve un puntero al arreglo creado (ArregloMinucias).

ArregloMinucias\* crearArregloMinucias(int sizeArreglo)

{

    ArregloMinucias \*amPtr = (ArregloMinucias\*)malloc(sizeArreglo\*sizeof(ArregloMinucias));

    amPtr->arreglo = (Minucia\*)calloc(100, sizeof(Minucia));

    amPtr->longitud=sizeArreglo;

    return amPtr;

}

**5 (15 puntos).** La función liberarArregloMinucias libera toda la memoria ocupada por el arreglo de minucias que se pasa como un parámetro (un puntero a ArregloMinucias) de la función.

void freeQueue(ArregloMinucias\* arregloMin)

{

    free(arregloMin->arreglo);

    free(arregloMin);

}

**6 (15 puntos).** La función encontrarCentroide recibe un puntero a un ArregloMinucias y llama a una función calcularDistancia; devuelve un puntero a Minucia. La función calcularDistancia recibe dos punteros a Minucia devuelve un número de coma flotante de 64 bits. encontrarCentroide itera sobre las minucias almacenadas en ArregloMinucias; y devuelve un puntero a la minucia cuya distancia acumulada a los demás es mínima.

double calcularDistancia(Minucia \*p1, Minucia \*p2)

{

    double dif1 = pow(p1->x - p2->x, 2);

    double dif2 = pow(p1->y - p2->y, 2);

    double dist = sqrt(dif1+dif2);

    return dist;

}

Minucia\* encontrarCentroide(ArregloMinucias \*aPtr)

{

    //Minuncia minima a regresar se asume que es la primera la de menor distancia con todas

    Minucia \*minuciaMinima;

    minuciaMinima=&aPtr->arreglo[0];

    //distancia a comparar

    //minimo que se comparara con distancia

    //empieza en infinito para que la primera vez calculada lo ingrese en minimo

    double distancia=0;

    double minimo=INFINITY;

    //Para comparar todas las minucias con hasta consigo misma sumando sus distancias

    //No se valida que no se sume consigo misma ya que la suma de eso es 0

    int i,j;

    for(i=0;i<aPtr->longitud;i++)

    {

        for(j=0;j<aPtr->longitud;j++)

        {

            //Acumular la distancia para que sea distancia con todos

            distancia = distancia + calcularDistancia(&aPtr->arreglo[i],&aPtr->arreglo[j]);

        }

if(distancia<minimo)

        {

            //Cambia el apuntador de la minuciaMinima si tiene menor distancia acumulada que el anterior

            minuciaMinima=&aPtr->arreglo[i];

            minimo = distancia;

        }

        //Se reinicia el contador para seguir con otra minucia

        distancia = 0;

    }

    return minuciaMinima;

}

**7 (10** puntos). La función devolverTipoMinucia recibe un puntero a Minucia como parámetro y devuelve el tipo de la minucia. Note que el tipo está almacenado en la estructura con dos bits y debe devolver el nombre del tipo.

char \* devolverTipoMinucia(Minucia \*mi)

{

    //mask 00000011

    //&

    //ya que nos importa saber solo sus ultimos dos bits

    uint8\_t mask=0x03;

    uint8\_t tipo=mi->tipo;

    char tipoStr[]="";

    if((tipo & mask)==0){

        return "Valor Tipo Minuncia: TERMINACION\n";

    }

    if((tipo & mask)==1){

        return "Valor Tipo Minuncia: DIVISION\n";

    }

    if((tipo & mask)==2){

        return "Valor Tipo Minuncia: PUNTO\n";

    }

    if((tipo & mask)==3){

        return "Valor Tipo Minuncia: DESCONOCIDO\n";

    }

}

**8 (5 puntos extras).** Incluya la función main() con el código necesario para probar sus funciones.

¡Buena Suerte!

int main()

{

    //Declarar variables para escanear

    uint16\_t sizeArr=0,auxSize;

    uint16\_t x, cordX;

    uint16\_t y, cordY;

    double angulo;

    uint8\_t tipo;

    //pedir dimension arreglo

    printf("Dimension de Arreglo de Minucias: ");

    scanf("%d",&sizeArr);

    //Se perdia el valor por eso se genera ese auxiliar del mismo tipo de variable

    auxSize = sizeArr;

    //Crear el arreglo de Minucias con la dimension pedida

    ArregloMinucias \*arrPtr = crearArregloMinucias(auxSize);

    printf("\n");

    printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

    printf("\*\*\*    IMPRIMIENDO DATOS MINUCIAS INICIALIZADO EN 0    \*\*\*\n");

    printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

    int i;

    for(i=0;i<auxSize;i++)

    {

        printf("\n--- MINUNCIA %d ---\n",i);

        printf("Coordenada X       : %d\n", arrPtr->arreglo[i].x);

        printf("Coordenada Y       : %d\n", arrPtr->arreglo[i].y);

        printf("Angulo             : %lf\n",arrPtr->arreglo[i].angulo);

        printf("Valor Tipo Minuncia: %d\n", arrPtr->arreglo[i].tipo);

    }

    printf("\n");

    printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

    printf("\*\*\*                  ESCANEO DE DATOS                  \*\*\*\n");

    printf("\*\*\* Tipo Segun 2 Ultimos Bits                          \*\*\*\n");

    printf("\*\*\* 00-Terminacion 01-Division 10-Punto 11-Desconocido \*\*\*\n");

    printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

    for(i=0;i<auxSize;i++)

    {

        //1. Pedir cada valor para cada campo de Minuncia

        printf("\n--- MINUNCIA %d ---\n",i);

        printf("Coordenada X       : ");

        scanf("%d",&x);

        //Se perdia el valor por eso se genera ese auxiliar del mismo tipo de variable

        cordX=x;

        printf("Coordenada Y       : ");

        scanf("%d",&y);

        //Se perdia el valor por eso se genera ese auxiliar del mismo tipo de variable

        cordY=y;

        printf("Angulo             : ");

        scanf("%lf",&angulo);

        printf("Valor Tipo Minuncia: ");

        scanf("%d",&tipo);

        //2. Crear Minuncia con su apuntador

        arrPtr->arreglo[i].x=crearMinucia(cordX,cordY,angulo,tipo)->x;

        arrPtr->arreglo[i].y=crearMinucia(cordX,cordY,angulo,tipo)->y;

        arrPtr->arreglo[i].angulo=crearMinucia(cordX,cordY,angulo,tipo)->angulo;

        arrPtr->arreglo[i].tipo=crearMinucia(cordX,cordY,angulo,tipo)->tipo;

    }

    printf("\n");

    printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

    printf("\*\*\*       IMPRIMIENDO DATOS MINUCIAS Y SUS TIPOS       \*\*\*\n");

    printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

    for(i=0;i<auxSize;i++)

    {

        printf("\n--- MINUNCIA %d ---\n",i);

        printf("Coordenada X       : %d\n", arrPtr->arreglo[i].x);

        printf("Coordenada Y       : %d\n", arrPtr->arreglo[i].y);

        printf("Angulo             : %lf\n",arrPtr->arreglo[i].angulo);

        printf("Valor Tipo Minuncia: %d\n", arrPtr->arreglo[i].tipo);

        printf(devolverTipoMinucia(&arrPtr->arreglo[i]));

    }

    printf("\n");

    printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

    printf("\*\*\*               ENCONTRANDO CENTROIDE                \*\*\*\n");

    printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

    Minucia\* minCentroide = encontrarCentroide(arrPtr);

    printf("\n---  CENTROIDE  ---\n",i);

    printf("Coordenada X       : %d\n", minCentroide->x);

    printf("Coordenada Y       : %d\n", minCentroide->y);

    printf("Angulo             : %lf\n",minCentroide->angulo);

    printf("Valor Tipo Minuncia: %d\n", minCentroide->tipo);

    printf(devolverTipoMinucia(minCentroide));

    free(arrPtr);

    //Use esta parte de codigo para ver si seguia estando apartada la memoria

    //Funciona como esperado: Libera el arreglo de Minuncias

    //Aparece datos raros, pero lo bloqueo para evitar que funcione erroneamente con el resto del codigo

    /\*printf("\n");

    printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

    printf("\*\*\*   IMPRIMIENDO DATOS MINUCIAS LIBERANDO EL ESPACIO  \*\*\*\n");

    printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

    for(i=0;i<auxSize;i++)

    {

        printf("\n--- MINUNCIA %d ---\n",i);

        printf("Coordenada X       : %d\n", arrPtr->arreglo[i].x);

        printf("Coordenada Y       : %d\n", arrPtr->arreglo[i].y);

        printf("Angulo             : %lf\n",arrPtr->arreglo[i].angulo);

        printf("Valor Tipo Minuncia: %d\n", arrPtr->arreglo[i].tipo);

    }\*/

    return 0;

}