Universidad Andrés Bello
Facultad de Ingeniería
Ingeniería en Computación e Informática

Estructura de Datos
Laboratorio II

Observaciones:

- Todos los programas deben ser desarrollados mediante el uso de funciones que serán llamadas en la función principal.
- Cada ejercicio será desarrollado en el lenguaje C y debe generar librerías según lo visto en clases.
- La realización del taller es en grupos de máximo 3 personas.
- Al inicio del código debe ir en un comentario los nombres de los integrantes del grupo y el nombre del profesor de laboratorio.
- La copia se evaluará con la nota mínima para todas las partes involucradas.
- La solución del ejercicio 3 deberá enviarla a su profesor de laboratorio correspondiente. El correo contendrá como asunto lo siguiente: **[UNAB ED-LAB] Laboratorio II**

Los punteros

Un puntero es una variable que contiene la dirección de otra variable. Por ejemplo, un puntero a entero es una variable que se utiliza para guardar la dirección de una variable tipo int. Declaración:

```
int *punt; // declaración de un puntero a entero llamado \punt"
char *punt1; // declaración de un puntero a char llamado \punt1"
float *punt2; // declaración de un puntero a float llamado \punt2"
```

Estas declaraciones reservan espacio para un puntero, pero no inicializan el puntero, es decir, están apuntando a cualquier parte. En caso de querer obtener la dirección de memoria de una variable que no ha sido declarada como puntero se utilizará el operador `&`.

Existe una estrecha relación entre un arreglo y los punteros. El nombre del arreglo es la dirección del primer elemento del arreglo. El acceso a los distintos elementos lo podemos expresar como sigue

$$X[i] = *(X + i)$$

Estructuras

En C es posible definir tipos de datos que contengan varios datos, esto se hace mediante las estructuras. La forma de definirlas es utilizando la palabra reservada **struct**. Una forma de uso sería como se muestra a continuación:

```
struct mystruct {
    int int_member;
    double double_member;
    char string_member[25];
} variable;
```

variable es una instancia de *mystruct* y podría no escribirse. Si se omité la declaracion de *mystruct* y se puede declarar posteriormente de la siguiente manera:

```
struct mystruct variable;
```

Una práctica común es asignarle un alias al nombre de la estructura, para evitar el tener que poner *struct mystruct* cada vez que se declare una variable de ese tipo. C permite usar la palabra clave *typedef*, el que crea un alias a un tipo:

```
typedef struct {
    ...
} Mystruct;
```

La estructura misma no tiene nombre (por la ausencia de nombre en la primera linea), pero tiene de alias *Mystruct*. Entonces se puede usar así:

```
Mystruct variable;
```

Ejemplo

A continuación se define la estructura Complex que representa a los números complejos, además, se definen las operaciones *suma*, *multiplicacion* y la *comparación*

```
#include <stdio.h>

typedef struct {
    int r;
    int i;
} Complex;

Complex suma(Complex z, Complex w);
Complex multiplicacion(Complex z, Complex w);
int iguales(Complex z, Complex w);

int main() {
    Complex x, y;

    x.r = 10;
    x.i = 5;

    y.r = 7;
    y.i = 8;

    printf("(%i, %i)\n", x.r, x.i);
    printf("(%i, %i)\n", y.r, y.i);

    Complex aux;
    aux = suma(x, y);
    printf("(%i, %i) + (%i, %i) = (%i, %i)\n", x.r, x.i, y.r, y.i, aux.r, aux.i);
    aux = multiplicacion(x, y);
    printf("(%i, %i) * (%i, %i) = (%i, %i)\n", x.r, x.i, y.r, y.i, aux.r, aux.i);
    printf("(%i, %i) == (%i, %i)? %i\n", x.r, x.i, y.r, y.i, iguales(x, y));

    return 0;
}

Complex suma(Complex z, Complex w) {
    Complex aux;
    aux.r = z.r + w.r;
    aux.i = z.i + w.i;

    return aux;
}

Complex multiplicacion(Complex z, Complex w) {
    Complex aux;
    aux.r = z.r*w.r - z.i*w.i;
    aux.i = z.r*w.i + z.i*w.r;

    return aux;
}

int iguales(Complex z, Complex w) {
    if(z.r == w.r && z.i == w.i) return 1;
    return 0;
}
```

Pregunta 1

Basado en el ejemplo de los números complejos, programe las siguientes operaciones:

Sea $z = a - bi$ un número complejo, donde a es la parte real y b es la parte imaginaria, se definen las siguientes operaciones:

Operación	Representación	Fórmula
Parte Real	$Re(z)$	a
Parte Real	$Im(z)$	b
Valor absoluto	$\|z\|$	$\sqrt{Re^2(z) + Im^2(z)}$
Conjugado	$\overline{z}$	$Im(z) + Re(z)i$

Pregunta 2

Cree una estructura llamada Matrix, que contenga tres atributos:

- Un doble puntero de tipo flotante para crear una matriz dinámica.
- Una variable que indique la cantidad de filas de la matriz.
- Una variable que indique la cantidad de columnas de la matriz.

Luego cree las siguientes funciones:

1. `iniciarMatriz` : recibe por parámetro una variable de tipo `Matrix` y retorna una variable de tipo `Matrix` con una matriz inicializada.
2. `sumarMatriz` : recibe por parámetro dos variables de tipo `Matrix` y retorna una variable de tipo `Matrix`. Debe validar que la suma se pueda realizar.

Pregunta 3

Cree una estructura de nombre `Array` que contenga:

1. Los datos de tipo entero.
2. El largo del arreglo.

Luego, cree una función de nombre `iniciarArray` que reciba solo una variable de tipo entero, y que retorne una variable de tipo `Array` que contenga un arreglo de largo n con valores aleatorios.

Diseñe una función de nombre `mostrarArray` que reciba solo una variable de tipo `Array` y que muestre el contenido del arreglo.

OPCIONAL Para la función *mostrarArray* utilice **listas de argumentos de longitud variable**, de tal manera que pueda recibir una o mas variables de tipo `Array` y todas puedan ser visualizadas. Para mas información puede consultar el *Capítulo 7. Entradas y salida* del libro El Lenguaje de Programación C.