	Carátula para entrega de prácticas
Facultad de Ingeniería	Laboratorio de docencia

Laboratorios de computación salas A y B

Profesor:	M.C. JOSE MAURICIO MATAMOROS DE MARIA Y CAMPOS
Asignatura:	ESTRUCTURA DE DATOS Y ALGORITMOS I
Grupo:	12
No de Práctica(s):	02
Integrante(s):	ARELLANES CONDE ESTEBAN
No. de Equipo de cómputo empleado:	
No. de Lista o Brigada:	01
Semestre:	2022-2
Fecha de entrega:	28 feb, 23:59
Observaciones:	

CALIFICACIÓN: _____

Práctica 2.
Aplicaciones de apuntadores

Estructuras de Datos y Algoritmos I

Autor: Arellanes Conde Esteban

Índice

1. Objetivo

Utilizar apuntadores en lenguaje C para acceder a las localidades de memoria tanto de datos primitivos como de arreglos.

2. Introducción

En el lenguaje C existen tipos de variables llamadas “apuntadores”, o punteros en algunos casos, que sirven para almacenar la dirección de memoria de otra variable, esto mediante el operador unario “ * ”que retorna el valor de la variable referenciada junto con el operador “ & ” (Indirección) para acceder a la dirección de memoria de la variable, a esto se le conoce como desreferencia.

3. Actividad 1: Apuntadores como direcciones de memoria

El programa de la actividad uno ("`random.c`") proporcionado por el profesor consistía en generar un número pseudo-aleatorio con base en la semilla 69 para posteriormente imprimir este número y su dirección en memoria. Al compilar y ejecutar dicho programa tres veces los números generados y sus direcciones de memoria son:

3.1. Actividad 1: Tabla de resultados 1

	Dirección	Número
Ejecución 1	(0x7ffe7447d164)	790
Ejecución 2	(0x7ffed2415a44)	790
Ejecución 3	(0x7fff0e4951f4)	790

[2 puntos] ¿Las direcciones coinciden? Explique: No, porque varía dependiendo del tamaño de la memoria en cada procesador, sistema operativo de la computadora, entre otros. Aunque la semilla es la misma se asignan diferentes espacios de memoria para cada momento de ejecución del programa.

3.2. Actividad 1: Tabla de resultados 2

Continuando con la compilación y ejecución del programa "`randomv.c`" que consistía en generar e imprimir un arreglo de números pseudo-aleatorios y sus direcciones en memoria. Completamos la siguiente tabla:

Índice	Número	Dirección
a	(0x7ffdc7bebda0)	= { }
0	(0x7ffdc7bebda0)	790
1	(0x7ffdc7bebda4)	930
2	(0x7ffdc7bebda8)	615
3	(0x7ffdc7bebdac)	681
4	(0x7ffdc7bebdb0)	205
5	(0x7ffdc7bebdb4)	645
6	(0x7ffdc7bebdb8)	17
7	(0x7ffdc7bebdbc)	93
8	(0x7ffdc7bebdc0)	684
9	(0x7ffdc7bebdc4)	174

[2 puntos] Las direcciones en memoria, ¿son contiguas? ¿Están espaciadas a intervalos regulares?

Explique: No necesariamente porque son aleatorias y varía dependiendo el tipo de variable a la que apunte y el tamaño que ocupe en memoria, pero en este caso al ser de tipo entero ("`int`") están separadas a intervalos regulares de cuatro bytes.

4. Actividad 2: Aritmética de apuntadores

El programa "`bubble.c`" de acuerdo a lo visto en el laboratorio ordena un arreglo de valores tipo char generados aleatoriamente utilizando el algoritmo "`Bubble Sort`" implementado con aritmética de apuntadores. Modificando el programa anterior logramos que:

i) Ordenara enteros de 32 bits (`int`) en lugar de enteros de 8 bits (`char`).

ii) Utilizara arreglos en lugar de aritmética de apuntadores

4.1. Actividad 2: Resultado 1

[1 punto] ¿Qué salida produce el programa modificado?: a: 12 17 21 30 93 142 154 185 191 237

4.2. Actividad 2: Resultado 2

[3 puntos] Explique qué cambios tuvieron que hacerse a la función "`bubble_sort`" para que esta usara arreglos en lugar de aritmética de apuntadores: Reemplazar las variables tipo "`char`" por "`int`", de esa manera cumplimos el inciso uno y para el inciso dos lo que cambié en las líneas de código con arreglos en lugar de aritmética de apuntadores fue de esta manera:

$$\text{for (int j = a[0], i = a[0]; j} < (a[j+i]); ++j)\{$$
$$\text{if (\&a[j]} > \&a[j+1])swap(\&a[j], \&a[j+1]);\}$$

5. Actividad 3: Cadenas

El último programa realizado en el laboratorio tenía como propósito el de cifrar y descifrar una cadena de texto por medio del algoritmo del César, que consiste en aplicar un corrimiento a las letras del alfabeto de un número K de tamaño. Por línea de comandos el programa recibe dos parámetros: un entero y un doble apuntador a caracter. Estos pasan los argumentos (palabras) al programa ejecutable. Completando el programa, compilándolo y ejecútandolo con la línea: [`./crypto "Hola mundo!!!"]` obtenemos los siguientes resultados:

5.1. Actividad 3: Resultado 1

[1 punto] ¿Qué salida produce el programa?: La salida que produce el programa es:

```
Original scrypt: ‘‘Hola Mundo!!!"
Encrypted scrypt: ‘‘LSPE QYRHS!!!"
Decrypted scrypt: ‘‘HOLA MUNDO!!!"
```

5.2. Actividad 2: Resultado 2

[1 punto] ¿Qué salida produce el programa si se usa un corrimiento de 7 unidades?: La salida que produce el programa si ahora se usa un corrimiento de 7 unidades es:

```
Original scrypt: ‘‘Hola Mundo!!!"
Encrypted scrypt: ‘‘OVSH TBUKV!!!"
Decrypted scrypt: ‘‘HOLA MUNDO!!!"
```

6. Conclusión y Referencias

6.1. Conclusión:

El uso de apuntadores y arreglos en algunos casos es equivalente cuidando el formato de cada uno para evitar confusiones y pérdidas o problemas en el programa y más que recomendable tener dominancia en cada uno.

6.2. Referencias:

- Thomas H. Cormen. (2009). Introduction to Algorithms 3e. United States of America: Massachusets Institute of Technology.
- Stephen R. Davis. (2015). Beginning Programming with C++ For Dummies. New Jersey, United States of America: John Wiley Sons.