Algoritmos & Estructuras de Datos

Guía Práctica

Universidad de San Andrés

Magalí Marijuán I. Javier Mermet Matía Sandacz

2 de diciembre de 2022

Índice general

I Introducción **2**

1 Lenguaje C **3**
- 1.1 Introductorios . . . 3
- 1.2 Punteros y Arreglos . . . 3
- 1.3 Más arreglos! . . . 4
- 1.4 Búsqueda y Ordenamiento . . . 5

2 Recursividad **6**

3 Lectura de Archivos **7**

4 Magia con Punteros **8**

II Estructuras de Datos **9**

5 Listas **10**

6 Pilas **11**

7 Colas **12**

8 Heaps **13**

9 Árboles **14**

10 Grafos **15**

11 Hashing **16**

Parte I

Introducción

Capítulo 1

Lenguaje C

1.1 Introductorios

1. Escribir la función que dado n ∈ N devuelve si es primo. Recuerden que un número es primo si los únicos divisores que tiene son 1 y el mismo.

2. Escribir la función que dado n ∈ N devuelve la suma de todos los números impares menores que n.

1.2 Punteros y Arreglos

3. ¿Cuál es el valor de a y b luego de ejecutar el programa?

```
void myFunc(int* a, int b)
{
(*a)++;
b++;
}

void main()
{
    int a = 10;
    int b = 10;
    myFunc(&a, b);
}
```

4. ¿Que valor se imprime por consola luego de cada llamado a *printf*?

```
void main()
{
   int x = 10;
   int* px = &x;

   printf("%d \n", px);
   printf("%d \n", (*px));
```

```
    (*px)++;

    printf("%d \n", px);
    printf("%d \n", (*px));
}
```

5. Programar las siguientes funciones en C:

   (a) `void crearArreglo(int v)`

   Crea un arreglo estático de enteros de tamaño 8, inicializando todos sus elementos con v, y lo imprime en pantalla.

   (b) `int* crearArregloDinamico(int n)`

   Dado un natural n, crea un arreglo dinámico de enteros de ese tamaño, lo inicializa con ceros, y devuelve un puntero al mismo.

   (c) Invocar la siguiente función con cualquiera de los arreglos inicializados anteriormente y convencerse de que sus elementos están ubicados de manera **contigua** en memoria. *Recordar que cada elemento de tipo int ocupa 4 bytes*

```
void mostrarMemoria(int* arr, int size)
{
    for(int i=0; i<size; i++)
    {
        printf("Elemento: %d, Direccion: %d\n", i, &arr[i]);
    }
}
```

1.3 Más arreglos!

6. Programar las siguientes funciones en C:

   (a) `int maximo(int* arr, int size)`

   Dado un arreglo de enteros *arr* de tamaño *size*, devuelve el máximo elemento.

   (b) `void sumador(int* arr, int c, int size)`

   Suma c a todos los elementos de *arr*.

   (c) `char* copiar(char* arr)`

   Crea una copia de *arr*.

   (d) `int* reverso(int* arr, int size)`

   Dado un arreglo de enteros *arr* de tamaño *size*, devuelve su reverso.
   Ejemplo: Dado [1, -2, 85, 65] se debe devolver [65, 85, -2, 1].

   i. Se puede modificar *arr*.

   ii. Sin modificar *arr*.

   (e) `bool estaOrdenado(int* arr, int size)`

   Dado un arreglo *arr* de enteros de tamaño *size*, retorna true si es monótonamente creciente o monótonamente decreciente.

(f) `bool esPalindromo(char* s)`

Dado un String s, retorna true si es un palíndromo. *Recuerden que un palíndromo es una palabra que se lee igual en un sentido que en otro (por ejemplo; Ana, Anna, Otto).*

1.4 Búsqueda y Ordenamiento

7. Dado un arreglo de enteros arr, escribir una función que lo ordene de menor a mayor y dar su complejidad temporal.
   (a) Utilizando ***selection sort***.
   (b) Utilizando ***insertion sort***.
8. Dado un arreglo de enteros arr y un entero $target$, escribir una función que busque al $target$ en arr. Si existe, entonces devolver su índice. En caso contrario, devolver -1.
9. Repetir el ejercicio anterior, pero ahora suponiendo que arr está ordenado. El algoritmo debe tener complejidad $O(logn)$.

Capítulo 2

Recursividad

Capítulo 3

Lectura de Archivos

Capítulo 4

Magia con Punteros

Parte II

Estructuras de Datos

Capítulo 5

Listas

Capítulo 6

Pilas

Capítulo 7

Colas

Capítulo 8

Heaps

Capítulo 9

Árboles

Capítulo 10

Grafos

Capítulo 11

Hashing