Universidad Simón Bolívar
Departamento de Computación
y Tecnología de la Información
Laboratorio de Computación II
Sep-Dic 2019

> Solía permitirme creer que la mente es un subproducto del cuerpo. ¿Dónde se llega primero, al puerto o a ciudad, a la ciudad o al Estado, al Estado o al mundo?
>
> Mi hermana y yo, Federico Nietzsche.

Lab. 1—Sem. 2—Búsqueda Lineal y Binaria.

En este laboratorio tenemos cuatro tareas que cumplir, a saber:

1. Escribir y probar funciones que permitan leer y escribir los elementos de un arreglo.
2. Codificar los algoritmos de búsqueda lineal acotada y no-acotada vistos en clase.
3. Usar el esquema de búsqueda lineal para hallar la posición del primer primo de un arreglo de enteros en sus n primeras casillas.
4. Codificar el algoritmo **iterativo** de búsqueda binaria que se verá en clase y hacer pruebas usando dicho algoritmo.

Tareas

1. *Recordar cómo se leen y escriben los arreglos de la entrada y salida estandard.*

   a) A continuación se le dan varias funciones que permiten leer, escribir e intercambiar dos elementos de un arreglo de enteros y un main() que permite probar dichas funciones. Su primera tarea es sólo ejecutar y entender dichas funciones. (Tiempo estimado: 15min.) Consulte lo que no entienda.

```
//Pre: n >= 0
void leeArreglo(int a[], int n){
   int i = 0;
   while (i<n) {
     printf("\na[%d] = ", i);
     scanf("%d", &a[i]);
     i++;
   }
}

//Pre: n >= 0
void escribeArreglo(int a[], int n){
   int i = 0;
   printf("\n[ ");
   while (i < n-1) printf(" %d,",a[i++]);
   if (n > 0) printf(" %d]",a[i]); else printf("]");
}

void swap(int *a, int *b){
   int t = *a; *a = *b; *b =t;
}

int main(){
   int n = 8; int a[n];
   leeArreglo(a,5);
   escribeArreglo(a,5);
   swap(&a[0],&a[4]);
   escribeArreglo(a,5);
   return 0;
}
```

   b) Escriba las funciones que se indican a continuación y use *leerArreglo* y *escribeArreglo* para probarlas.

      1) Una función que reciba tres arreglos de enteros a, b, c y sume los elementos de los arreglos a, b en el arreglo c.
      2) Una función que halle la posición de la primera ocurrencia del máximo de los elementos del arreglo de enteros a en sus primeras n casillas.

2. *Búsqueda Lineal Acotada y no Acotada.*

   a) Haga un **NUEVO** programa que permita probar los algoritmos de búsqueda lineal vistos en clase:

```
//Retorna 1 si x pertenece o 0 si no.
int pertenece(int x, int n, int a[]){
   int k = 0;
   while(k<n && a[k] != x) k++;
   return k < n;
}

//Retorna la primera ocurrencia de x, o n si no está
int primaOcur(int x, int n, int a[]){
   int k = 0;
   while(k<n && a[k] != x) k++;
   return k;
}
```

   b) Modifique los algoritmos anteriores agregando lo que se busca en a[n]—primera casilla vacía—y eliminando $k < n$ de la condición. Nota: De esta manera la búsqueda se fuerza a ser *fructuosa y no-acotada!*

   c) *Búsquedas sin Arreglos—Primalidad* Use el esquema de búsqueda del ejercicio anterior para decidir si un entero positivo mayor que 1 es primo.

   d) Use el hecho de que n divide a n para sabiendo que la búsqueda va ser fructuosa eliminar $k < n$ de la condición.

   e) Si hay tiempo halle dos mejoras más sustanciales, según lo discutido en clases.

3. *Búsqueda de la posición del primer primo de un arreglo.* Use el esquema de búsqueda lineal visto para hallar la posición del primer número primo de un arreglo de enteros a en sus primeras n casillas.

```
int ppp(int a[], int n);
```

4. *Búsqueda Binaria*

   a) Implemente el algoritmo de búsqueda binaria iterativa que se le da a continuación y use el código de ordenamiento que se da para leer y ordenar un arreglo de tamaño ocho y hacer varias búsquedas sobre él.

```
int bb(int E, int n, int a[]){
   int x = 0, y = n, m;
   while (x+1 != y){
      m = (x+y)/2;
      if (E < a[m]) y = m; else x = m;
   }
   return a[x] == E;
}

void intercambioM(int a[], int n){
   int i,j,m;
   for (i = 0; i< n-1; i++){
      m = i;
      for (j = i+1; j<n; j++) if(a[j]<a[m]) m = j;
      if (i != m) swap(&a[i], &a[m]);
   }
}
```

b) Escriba el algoritmo de búsqueda binaria en el caso que el arreglo de entrada esté ordenado **no crecientemente**.

c) Escriba un algoritmo que genere un arreglo de enteros aleatorios en el intervalo $[0, 100)$, lo ordene y luego haga varias búsquedas binarias sobre él.

d) Modifique el algoritmo de búsqueda binaria que se da para que devuelva el lugar donde lo encontró o n si no lo encontró.

Nota: Se recomienda seriamente que no se hagan *cut and paste*. **Debe copiar su código para que lo entienda mejor!!!**