Ordenamiento – Algoritmos básicos

En esta oportunidad se desea elaborar un programa que nos permita comparar el desempeño de los tres algoritmos básicos de ordenamiento: **Selección**, **Inserción** y **Burbuja**, cuando ordenamos en orden descendente un vector de números enteros positivos no repetidos desordenado aleatoriamente. Aunque es conocido que el orden de complejidad de estos tres algoritmos es n^2, debido a los detalles propios de implementación es posible observar diferencias en tiempo de ejecución según el tamaño del vector que se desea ordenar.

El programa a desarrollar deberá generar un conjunto de tres vectores desordenados aleatoriamente con tamaños 10000, 100000, 1000000.

Para obtener el vector desordenado::
1. Genere un vector de enteros con valores 1, 2, 3, 4, ….N.
2. Desordene aleatoriamente el vector usando el algoritmo de **Fisher–Yates**.
(https://en.wikipedia.org/wiki/Fisher%E2%80%93Yates_shuffle)

```
-- Para barajar un arreglo de n elementos (indices 0..n-1)
para i desde 0 hasta n−2 hacer:
     j ← random entero tal que 0 ≤ j < n-i
     intercambiar a[i] y a[i+j]
```

Cada vector generado se debe copiar a un vector de trabajo que se usará como entrada para cada uno de los tres algoritmos de ordenamiento.

Para medir el tiempo que tarda cada ordenamiento se sugiere el código mostrado en el siguiente link:
http://stackoverflow.com/questions/5248915/execution-time-of-c-program

```
#include <sys/time.h>

struct timeval  tv1, tv2;
gettimeofday(&tv1, NULL);
/* llamada al algoritmo de ordenamiento */
gettimeofday(&tv2, NULL);

printf ("Total time = %f seconds\n",
         (double) (tv2.tv_usec - tv1.tv_usec) / 1000000 + (double) (tv2.tv_sec -
tv1.tv_sec));
```

Como parte del proyecto se sugiere programar las siguientes funciones y/o procedimientos:

```
void intercambiar(int *a, int *b);
int *copiar(int *v, int tam);
void llenarVectorMenoraMayor(int v[], int n);
int *genRndArray(int size);
void seleccion(int v[], int n);
void insercion(int v[], int n);
void burbuja(int v[], int n);
```

`intercambiar`: Procedimiento usado por los algoritmos de ordenamiento y `genRndArray` para

intercambiar dos elementos del vector.

`copiar`: función que retorna una copia del vector de entrada `v[]`, de forma que se puede tener:

```
int *x;
int tamano = 1000;
x = copiar(v, tamano);

int *copiar(int *v, int tam)
{
      int *w, i;

      /* Asignar espacio para un vector con tam elementos de tipo int */
      w = (int *)malloc(tam * sizeof(int));
      for(i = 0; i < tam; i++)
            w[i] = v[i]; /* copia componente del vector v en el vector w */
      return w;
}

int *copiar(int const *v, int tam)
{  /* mas eficiente */
   int *p = (int *)malloc(tam * sizeof(int));
   memcpy(p, v, tam * sizeof(int));
   return p;
}
```

Copia estática de arreglos (solo si no puede hacerlo de la forma sugerida en los dos anteriores).

```
void copiar(int v[], int w[], int tam)
{
      int *w, i;

      for(i = 0; i < tam; i++)
            w[i] = v[i]; /* copia componente del vector v en el vector w */
}
```

`llenarVectorMenoraMayor`: Procedimiento que llena el vector `v[]` con valores desde `1` hasta `n`, utilizado por `genRndArray`.

`genRndArray`: Función que genera un vector desordenado de tamaño `size`.

`insercion`, `seleccion`, `burbuja`: Procedimientos clásicos de ordenamiento que ordenan el vector `v[]` de tamaño `n` en orden descendente.

Además de medir los tiempos de ejecución se desea determinar para cada algoritmo de ordenamiento el número de `intercambios` y de `comparaciones` usando para esto compilación condicional. Observe en el código a continuación el uso de la variable `DEBUG` para la compilación condicional del procedimiento `insercion` (**atención**: este procedimiento ordena el vector `v[]` en forma ascendente).

```
void insercion(int v[], int n)
{
      int i, j, aux;
```

```
	#ifdef DEBUG
	int comparaciones = 0;
	int intercambios = 0;
	#endif
	for(i = 1; i < n; i++)
	{
		j = i;
		aux = v[j];
		#ifdef DEBUG
		comparaciones++;
		#endif
		while(j > 0 && aux < v[j-1])
		{
			v[j] = v[j-1];
			j--;
			#ifdef DEBUG
			intercambios++;
			comparaciones++;
			#endif
		}
		v[j] = aux;
	}
	#ifdef DEBUG
	printf("Insercion -- Comparaciones %d -- Intercambios %d\n", comparaciones,
intercambios);
	#endif
}
```

Para compilar el código de forma que se ejecuten las lineas en color, se sugiere usar el siguiente comando:

```
$ cc -o RUN -DDEBUG ordenamientos.c
```

La razón de usar compilación condicional en esta parte del proyecto es que el código que totaliza el número de comparaciones e intercambios no altere las mediciones del tiempo de ejecución de los ordenamientos.

Finalmente, no olvide formatear el código producido con el comando:

```
$ astyle --style=allman ordenamientos.c
```

Pruebe su programa y envíe por correo:

- Código fuente (archivo.c)
- El resultado de las corridas, sin y con la opcion DEBUG activada.
- Si realizó las pruebas de su programa en una máquina con sistema operativo Linux envíe el resultado de los comandos:

```
$ uname -a
$ cc -v
$ lshw
```

En mi caso estos son los resultados:

```
$ ./RUN
Vector de 10000. Tiempo ordenamiento por selección: 0.157173 seg
Vector de 10000. Tiempo ordenamiento por insercion: 0.070657 seg
Vector de 10000. Tiempo ordenamiento por burbuja: 0.104734 seg
Vector de 100000. Tiempo ordenamiento por selección: 10.460976 seg
Vector de 100000. Tiempo ordenamiento por insercion: 6.924901 seg
```

```
Vector de 100000. Tiempo ordenamiento por burbuja: 10.516829 seg
Vector de 1000000. Tiempo ordenamiento por selección: 1057.533866 seg
Vector de 1000000. Tiempo ordenamiento por insercion: 696.532028 seg
Vector de 1000000. Tiempo ordenamiento por burbuja: 1055.234037 seg

$./RUN (con DDEBUG)
Selection -- Comparaciones 49995000 -- Intercambios 9999
Vector de 10000. Tiempo ordenamiento por selección: 0.150917 seg
Insercion -- Comparaciones 25013450 -- Intercambios 25003451
Vector de 10000. Tiempo ordenamiento por insercion: 0.078152 seg
Selection -- Comparaciones 49995000 -- Intercambios 9999
Vector de 10000. Tiempo ordenamiento por burbuja: 0.112428 seg
Selection -- Comparaciones 4999950000 -- Intercambios 99999
Vector de 100000. Tiempo ordenamiento por selección: 11.274860 seg
Insercion -- Comparaciones 2498727280 -- Intercambios 2498627281
Vector de 100000. Tiempo ordenamiento por insercion: 7.885815 seg
Selection -- Comparaciones 4999950000 -- Intercambios 99999
Vector de 100000. Tiempo ordenamiento por burbuja: 11.163541 seg
Selection -- Comparaciones 499999500000 -- Intercambios 999999
Vector de 1000000. Tiempo ordenamiento por selección: 1118.088425 seg
Insercion -- Comparaciones 250162400694 -- Intercambios 250161400695
Vector de 1000000. Tiempo ordenamiento por insercion: 775.710766 seg
Selection -- Comparaciones 499999500000 -- Intercambios 999999
Vector de 1000000. Tiempo ordenamiento por burbuja: 1118.748659 seg

$ uname -a
Linux concordia 4.4.3-1-ARCH #1 SMP PREEMPT Fri Feb 26 15:09:29 CET 2016 x86_64
GNU/Linux

$ cc -v
Usando especificaciones internas.
COLLECT_GCC=cc
COLLECT_LTO_WRAPPER=/usr/lib/gcc/x86_64-unknown-linux-gnu/5.3.0/lto-wrapper
Objetivo: x86_64-unknown-linux-gnu
Configurado con: /build/gcc-multilib/src/gcc-5-20160209/configure --prefix=/usr
--libdir=/usr/lib --libexecdir=/usr/lib --mandir=/usr/share/man
--infodir=/usr/share/info --with-bugurl=https://bugs.archlinux.org/ --enable-
languages=c,c++,ada,fortran,go,lto,objc,obj-c++ --enable-shared --enable-
threads=posix --enable-libmpx --with-system-zlib --with-isl --enable-__cxa_atexit
--disable-libunwind-exceptions --enable-clocale=gnu --disable-libstdcxx-pch
--disable-libssp --enable-gnu-unique-object --enable-linker-build-id --enable-lto
--enable-plugin --enable-install-libiberty --with-linker-hash-style=gnu --enable-
gnu-indirect-function --enable-multilib --disable-werror --enable-checking=release
Modelo de hilos: posix
gcc versión 5.3.0 (GCC)

$ lshw
WARNING: you should run this program as super-user.
concordia
    description: Computer
    width: 64 bits
    capabilities: vsyscall32
  *-core
       description: Motherboard
       physical id: 0
     *-memory
          description: System memory
          physical id: 0
```

```
     size: 7884MiB
*-cpu
     product: Intel(R) Core(TM) i5-3450 CPU @ 3.10GHz
     vendor: Intel Corp.
     physical id: 1
     bus info: cpu@0
     size: 3328MHz
     capacity: 3500MHz
     width: 64 bits
```

Notas:

- Sus resultados pueden ser diferentes dependiendo del orden aleatorio de los vectores, del hardware del computador y del sistema operativo en que realice las pruebas.
- Si realiza sus pruebas en Wndows, indicar la versión del sistema operativo, versión del compilador y plataforma de hardware. Indique que programa usó para obtener la información.