suponer que k sera el numero de iteraciones que el predicado booleano es verdadero por lo tanto el tiempo de k quedaría de esta manera:

$$t(k) = 1 + \sum_{l=1}^{k} 5 + \sum_{l=k+1}^{n^2-3n/2} 3 \Longrightarrow t(n) = \frac{1}{\frac{n^2-3n}{2}+1} \times \left(1 + \sum_{l=1}^{k} 5 + \sum_{l=k+1}^{n^2-3n/2} 3\right)$$

$$t(n) = \frac{1}{\frac{n^2-3n}{2}+1} \times \left(1 + 5k + 3\left(\frac{n^2-3n}{2} - k\right)\right) \Longrightarrow \frac{1}{\frac{n^2-3n}{2}+1} \times \left(1 + 5k + \frac{3}{2}(n^2-3n) - 3k\right)$$

$$t(n) = \frac{1}{\frac{n^2-3n}{2}+1} \times \left(1 + 2k + \frac{3}{2}(n^2-3n)\right) \Longrightarrow \frac{1}{\frac{n^2-3n}{2}+1} \times \left(1 + 2k + \frac{3}{2}n^2 - 9n\right)$$

3. Segunda Practica

3.1. Presentación del problema:

En este caso tenemos que implementar un algoritmo de selección, osea, que tendremos que encontrar el valor de un elemento dentro de un vector desordenado dada una posición y que dicho valor de la posición sea el mismo valor que cuando el vector este ordenado.

3.2. Metodología seguida para la solución:

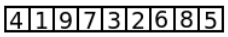
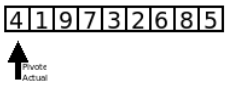
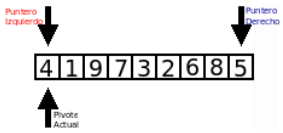
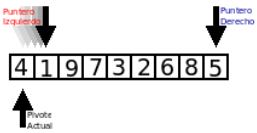
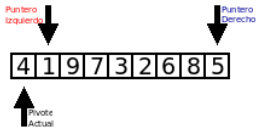
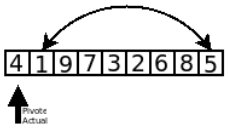
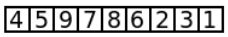
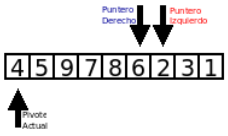
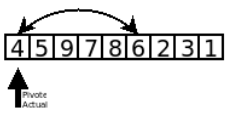
Para realizar esta practica de implementación de algoritmos primero hemos buscado cual es el lenguaje mas cómodo para la implementación del mismo, con esto me refiero al lenguaje que menos librerías tenemos que utilizar y que menos tiempo he tenido que emplear para resolver la practica, ya que en cualquier lenguaje se podría implementar la solución.

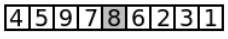
Una vez escogido, he hecho una primera aproximación utilizando el pseudocódigo que viene en las transparencias de clase, para implementar el algoritmo de QuickSort(), en este punto me he dado cuenta de que ha estas lineas le faltaban cosas y le sobraban otras pero después de muchas pruebas y búsquedas, he conseguido que el algoritmo funcionase.

Después de entender la función Pivote() que se usa en el algoritmo de QuickSort(), he comprendido que realmente dicho algoritmo debería llamarse Pivote(), ya que todo el trabajo de ordenación realmente lo realiza esta función, de hecho QuickSort() no es mas que una función wrapper por si antes de intentar ordenar quisiéramos pre procesar el vector.

Después he utilizado la función Pivote() del algoritmo QuickSort() y la he modificado para que pudiera recibir un vector y un enteros que seria la posición del valor que queremos obtener, por supuesto, aunque los ejemplos lo he hecho con números enteros es totalmente factible elaborar un template el cual después de recibir la posición nos devolviera el objeto o el valor de la posición que le hemos pedido sin tener que ordenar el vector de datos, cabe destacar que para que el algoritmo de QuickSort() funcione los objetos o datos que se encuentren en el vector deben de ser escalares, osea que, tiene que poder usarse los operadores, mayor que $>$, menor que $<$ e igual $=$, quizás este demás decirlo, puesto que si queremos ordenar algo esta observación va implícita en la propia ordenación del vector, pero como a mi me costo entender esta idea la recalco para futuras lecturas.

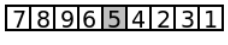
Para terminar de explicar como funciona este algoritmo nos apoyaremos en las siguientes explicaciones gráficas:

Gráfica	Descripción
	Este es nuestro vector el cual queremos encontrar el valor que se encuentra en la mediana del mismo, hay que tener en cuenta que aunque para la explicación hemos usado un vector de enteros este podría contener cualquier tipo de dato.
	En las linea 50 del código se hace la llamada a selección() nada mas llamar a la función esta utiliza tres enteros para posicionar el pivote y los dos punteros (izquierdo y derecho) que va a utilizar para llamar a la función pivote() lo primero que hace es usar el primer dato del vector como si fuera el primer pivote del algoritmo. Esto se hace en las linea de código 35.
	Después de haber posicionado el pivote se hace lo mismo con los dos punteros el izquierdo y el derecho, la primera vez que se usan estos punteros apuntan al primer dato del vector (izquierdo) y al ultimo dato del vector (derecho). Esto se hace en las lineas de código 36 y 37.
	Una vez que los punteros están en sus lugares correspondientes comienza el funcionamiento del algoritmo haciendo que el puntero de la izquierda recorra todo el vector de izquierda a derecha siempre y cuando no se pase del tamaño del vector, este parara cuando el valor que esta en la posición del puntero que estamos recorriendo sea mas grande o igual que el valor que se encuentra en la posición que se encuentra el pivote, esto es lo que hace la linea de código 39
	El puntero derecho hace lo mismo que el izquierdo pero en la otra dirección, osea que recorrerá todo el vector sin pasar desde el principio del mismo y se detendrá cuando en la posición en la que se encuentre haya un valor mayor o igual que el que se encuentra en pivote, esto es lo que hace la linea de código 40. Claro esta que en el caso que nos ocupa este puntero no se moverá del sitio por que el valor que hay en dicho puntero ya cumple con la condición esperada.
	Para el vector que nos ocupa el siguiente paso será el que se refleja en el gráfico y se intercambiaran los valores de los punteros, que como se muestra en la linea 78 si el puntero de la izquierda es menor que el puntero de la derecha se llama a la función intercambia() pasandole como parámetros ambos punteros
	Una vez realizado el intercambio de valores el vector se seguiría recorriendo el vector de tal manera que se harían los mismos pasos hasta que los punteros se cruzaran, despues de un par de iteraciones llegaríamos a una situación similar a la que se muestra en la figura.
	En el siguiente paso podemos comprobar como los dos punteros se han cruzado por lo tanto se va a proceder a intercambiar los valores pero en esta ocasión no serán entre los dos punteros.
	En esta ocasión lo que se va a cambiar es el limite inferior del vector con el puntero derecho, de esta manera, conseguimos que todos los valores que son mas grandes que el valor del dato que estamos buscando están a la izquierda y todos los valores mas chicos están a la derecha de dicho dato.



En el gráfico se puede apreciar como el valor que ha devuelto la función pivote() es 5 esta es la posición actual que tiene el pivote cuyo valor es 4, como la posición que buscamos es menor que la posición devuelta por dicha función lo que se hace a continuación es partir el vector en 2 de tal forma que la posición devuelta menos una unidad será el limite superior quedando el limite en la posición 4 que tiene el valor 8, por lo tanto esta será el nuevo vector que se le pasara a la función pivote().

El proceso que hemos explicado se repite hasta conseguir que el valor que queda en la mediana del vector corresponde al valor que quedaría en esa posición si el vector estuviese ordenado. Como referencia el siguiente gráfico muestra como sería la solución final del vector.



3.3. Traza de una ejecución completa del programa:

```
 Lista Desordenada
 4,1,9,7,3,2,6,8,5

_____________________________________________________
Posición del puntero de la Izquierda
Valor del pivote: 4, Posición del pivote: 0
4,1,9,7,3,2,6,8,5
En este momento el valor del puntero izquierdo es mas pequeño o igual que el valor del pivote
_____________________________________________________
Posición del puntero de la Derecha
Valor del pivote: 4, Posición del pivote: 0
4,1,9,7,3,2,6,8,5
En este momento el valor del puntero derecho es mas grande o igual que el valor del pivote
_____________________________________________________
Posicionados los dos punteros si el izquierdo es mas pequeño
que el derecho se intercambian los valores y se sigue recorriendo
el vector hasta que se crucen los punteros
4,1,9,7,3,2,6,8,5 Antés
4,5,9,7,3,2,6,8,1 Despúes
_____________________________________________________
Seguimos recorriendo el vector, posición del puntero de la Izquierda
Valor del pivote: 4, Posición del pivote: 0
4,5,9,7,3,2,6,8,1
En este momento el valor del puntero izquierdo es mas pequeño o igual que el valor del pivote
_____________________________________________________
Seguimos recorriendo el vector, posición del puntero de la Derecha
Valor del pivote: 4, Posición del pivote: 0
4,5,9,7,3,2,6,8,1
En este momento el valor del puntero derecho es mas grande o igual que el valor del pivote
_____________________________________________________
Posicionados los dos punteros si el izquierdo es mas pequeño
que el derecho se intercambian los valores y se sigue recorriendo
el vector hasta que se crucen los punteros
4,5,9,7,3,2,6,8,1 Antés
4,5,9,7,8,2,6,3,1 Despúes
_____________________________________________________
Seguimos recorriendo el vector, posición del puntero de la Izquierda
Valor del pivote: 4, Posición del pivote: 0
4,5,9,7,8,2,6,3,1
En este momento el valor del puntero izquierdo es mas pequeño o igual que el valor del pivote
_____________________________________________________
Seguimos recorriendo el vector, posición del puntero de la Derecha
Valor del pivote: 4, Posición del pivote: 0
4,5,9,7,8,2,6,3,1
En este momento el valor del puntero derecho es mas grande o igual que el valor del pivote
_____________________________________________________
Posicionados los dos punteros si el izquierdo es mas pequeño
que el derecho se intercambian los valores y se sigue recorriendo
el vector hasta que se crucen los punteros
4,5,9,7,8,2,6,3,1 Antés
4,5,9,7,8,6,2,3,1 Despúes
_____________________________________________________
Seguimos recorriendo el vector, posición del puntero de la Izquierda
Valor del pivote: 4, Posición del pivote: 0
4,5,9,7,8,6,2,3,1
En este momento el valor del puntero izquierdo es mas pequeño o igual que el valor del pivote
_____________________________________________________
Seguimos recorriendo el vector, posición del puntero de la Derecha
Valor del pivote: 4, Posición del pivote: 0
4,5,9,7,8,6,2,3,1
En este momento el valor del puntero derecho es mas grande o igual que el valor del pivote
_____________________________________________________
Como los punteros se han cruzado lo que se hace
es intercambiar la posición del pivote actual
por la posición del puntero de la derecha
Valor del pivote: 4, Posición del pivote: 0
4,5,9,7,8,6,2,3,1 Antés
6,5,9,7,8,4,2,3,1 Despúes
_____________________________________________________
El valor que ha devuelto la funcion pivote() es: 5
esta es la posición actual que tiene el pivote cuyo valor es: 4
```

```
Como la posición que buscamos es menor que la posición devuelta
lo que se hace a continuación es partir el vector en 2
de tal forma que la posición devuelta menos una unidad
será el limite superior quedando el limite en: 4
que tiene el valor: 8
6,5,9,7,8,4,2,3,1
-----------------------------------------------------
Posición del puntero de la Izquierda
Valor del pivote: 6, Posición del pivote: 0
6,5,9,7,8,4,2,3,1
En este momento el valor del puntero izquierdo es mas pequeño o igual que el valor del pivote
-----------------------------------------------------
Posición del puntero de la Derecha
Valor del pivote: 6, Posición del pivote: 0
6,5,9,7,8,4,2,3,1
En este momento el valor del puntero derecho es mas grande o igual que el valor del pivote
-----------------------------------------------------
Posicionados los dos punteros si el izquierdo es mas pequeño
que el derecho se intercambian los valores y se sigue recorriendo
el vector hasta que se crucen los punteros
6,5,9,7,8,4,2,3,1 Antés
6,8,9,7,5,4,2,3,1 Despúes
-----------------------------------------------------
Seguimos recorriendo el vector, posición del puntero de la Izquierda
Valor del pivote: 6, Posición del pivote: 0
6,8,9,7,5,4,2,3,1
En este momento el valor del puntero izquierdo es mas pequeño o igual que el valor del pivote
-----------------------------------------------------
Seguimos recorriendo el vector, posición del puntero de la Derecha
Valor del pivote: 6, Posición del pivote: 0
6,8,9,7,5,4,2,3,1
En este momento el valor del puntero derecho es mas grande o igual que el valor del pivote
-----------------------------------------------------
Como los punteros se han cruzado lo que se hace
es intercambiar la posición del pivote actual
por la posición del puntero de la derecha
Valor del pivote: 6, Posición del pivote: 0
6,8,9,7,5,4,2,3,1 Antés
7,8,9,6,5,4,2,3,1 Despúes
-----------------------------------------------------
El valor que ha devuelto la funcion pivote() es: 3
esta es la posición actual que tiene el pivote cuyo valor es: 6
Como la posición que buscamos es mayor que la posición devuelta
lo que se hace a continuación es partir el vector en 2
de tal forma que el la posición devuelta mas una unidad
será el limite inferior quedando el limite en: 4
que tiene el valor: 5
7,8,9,6,5,4,2,3,1
-----------------------------------------------------
Posición del puntero de la Izquierda
Valor del pivote: 5, Posición del pivote: 4
7,8,9,6,5,4,2,3,1
En este momento el valor del puntero izquierdo es mas pequeño o igual que el valor del pivote
-----------------------------------------------------
Posición del puntero de la Derecha
Valor del pivote: 5, Posición del pivote: 4
7,8,9,6,5,4,2,3,1
En este momento el valor del puntero derecho es mas grande o igual que el valor del pivote
-----------------------------------------------------
Como los punteros se han cruzado lo que se hace
es intercambiar la posición del pivote actual
por la posición del puntero de la derecha
Valor del pivote: 5, Posición del pivote: 4
7,8,9,6,5,4,2,3,1 Antés
7,8,9,6,5,4,2,3,1 Despúes
-----------------------------------------------------
 Lista Modificada
 7,8,9,6,5,4,2,3,1

Valor del elemento seleccionado: 5
Tiempo empleado en la ordenación: 0.000465214
```

3.4. Mediciones del estudio empírico:

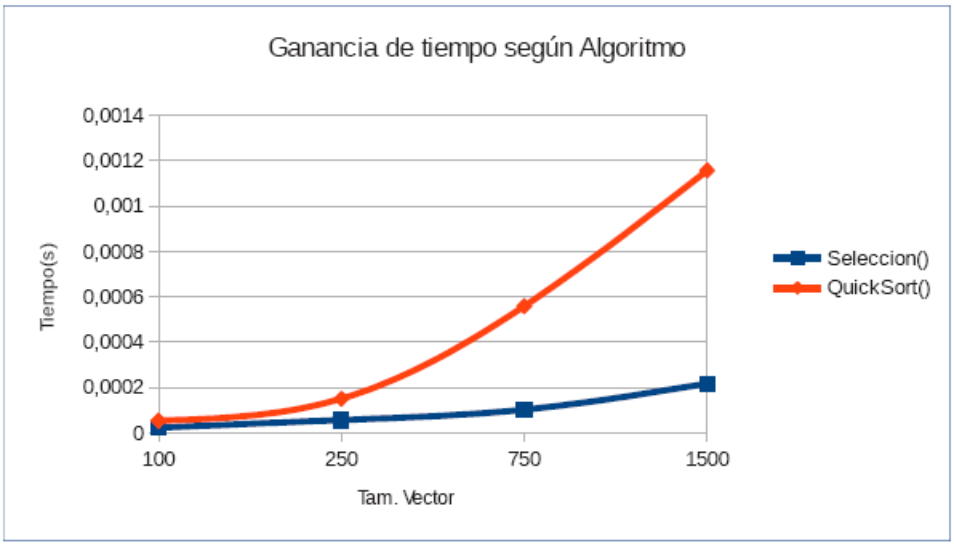
En este apartado lo que vamos ha realizar serán las mediciones necesarias para ver a partir de cuando llega a ser rentable o muy rentable utilizar este método para encontrar el valor de un dato en un vector desordenado con respecto al algoritmo QuickSort(), el cual primero tendría que ordenar el vector y luego extraer el dato de la posición que le hayamos indicado.

Bien aunque las mediciones se van a realizar en el mismo ordenador y en igualdad de condiciones, osea, con el ordenador recién arrancado se indica cual es el modelo de procesador y la cantidad de memoria que posee para posteriores comparaciones con otro tipo de maquina.

Ordenador: **Intel(R) Core(TM)2 Quad CPU Q6600 2.40GHz**

Memoria: **4 GB de DDR2 1333MHz en 4 bancos de memoria con FSB**

Tamaño del Vector	Seleccion()	QuickSort()
100	0,000022061	0,000053141
250	0,000055968	0,00015003
750	0,00010194	0,000557476
1500	0,000216664	0,001155791



3.5. Código fuente:

Aunque no es necesario incluir el código fuente para la evaluación de la practica yo lo incluyo para tener el código integrado en la lectura de la memoria.

```
/*
 * Algoritmo Selección
 *        Autor: Carlos de la Torre
 *        Fecha: 07/05/14
 */
#include <vector>
#include <iostream>
#include <iomanip>
#include <string>
#include <cstdlib>
#include <ctime>
#include "../inc/concolor.h"

using namespace std;

#define TAM_MIN_VEC 9
// con esto definimos que el tamaño mínimo
// del vector para que tenga buenos resultados
// la selección de la mediana
#define DEBUGMODE 0
// con esta definición nos aseguramos que solo
// salgan las cifras de tiempo en cada ejecución
// así de esa manera es mas fácil realizar el
// estudio empírico del programa

void intercambia (vector<int> &lista, int posi, int posd){
     int temporal;
    temporal= lista.at(posi);
    lista.at(posi)=lista.at(posd);
    lista.at(posd)=temporal;
}

int Pivote(vector<int> &lista,int limite_izq,int limite_der){
    int tmp, respuesta, pivote;

    pivote = lista.at(limite_izq);
    tmp = limite_izq;
    respuesta = limite_der;

    for(;lista.at(tmp)>=pivote and tmp<limite_der;tmp++);
    if (DEBUGMODE){
     cout <<redb<< "-------------------------------------------------------" << endl;
     cout << "Posición del puntero de la Izquierda" <<whiteb<< endl;
```

```
44      cout <<green<< "Valor del pivote: " << pivote << ", Posición del pivote: " << limite_izq <<
        whiteb<< endl;
45      for (unsigned int i=0; i<lista.size(); i++) {
46              if (i==tmp)
47                      cout <<redb<< lista.at(i) <<whiteb;
48              else
49                      cout << lista.at(i);
50
51              if(i<lista.size()-1)
52                      cout << ",";
53              else
54                      cout << endl;
55          }
56          cout << "En este momento el valor del puntero izquierdo es mas pequeño o igual que el
        valor del pivote" << endl;
57      cout <<redb<< "-------------------------------------------------------" <<whiteb<< endl;
58      }
59      for(;lista.at(respuesta)<=pivote and respuesta>limite_izq;respuesta--);
60      if (DEBUGMODE){
61      cout <<blueb<< "-------------------------------------------------------" << endl;
62      cout << "Posición del puntero de la Derecha" <<whiteb<< endl;
63      cout <<green<< "Valor del pivote: " << pivote << ", Posición del pivote: " << limite_izq <<
        whiteb<< endl;
64      for (unsigned int i=0; i<lista.size(); i++) {
65              if (i==respuesta)
66                      cout <<blueb<< lista.at(i) <<whiteb;
67              else
68                      cout << lista.at(i);
69
70              if(i<lista.size()-1)
71                      cout << ",";
72              else
73                      cout << endl;
74          }
75          cout << "En este momento el valor del puntero derecho es mas grande o igual que el valor
        del pivote" << endl;
76      cout <<blueb<< "-------------------------------------------------------" <<whiteb<< endl;
77      }
78      while (tmp<respuesta){
79      if (DEBUGMODE){
80              cout <<yellowb<< "-------------------------------------------------------" << endl;
81              cout << "Posicionados los dos punteros si el izquierdo es mas pequeño " << endl;
82              cout << "que el derecho se intercambian los valores y se sigue recorriendo" << endl;
83              cout << "el vector hasta que se crucen los punteros" <<whiteb<< endl;
84              for (unsigned int i=0; i<lista.size(); i++) {
85                      if (i==tmp)
86                              cout <<redb<< lista.at(i) <<whiteb;
87                      else if (i==respuesta)
88                              cout <<blueb<< lista.at(i) <<whiteb;
89                      else
90                              cout << lista.at(i);
91
92                      if(i<lista.size()-1)
93                              cout << ",";
94                      else
95                              cout <<green<< " Antés" <<whiteb<< endl;
96              }
97      }
98      intercambia(lista,tmp,respuesta);
99      if (DEBUGMODE){
100             for (unsigned int i=0; i<lista.size(); i++) {
101                     if (i==tmp)
102                             cout <<yellowb<< lista.at(i) <<whiteb;
103                     else if (i==respuesta)
104                             cout <<yellowb<< lista.at(i) <<whiteb;
105                     else
106                             cout << lista.at(i);
107
108                     if(i<lista.size()-1)
109                             cout << ",";
110                     else
111                             cout <<green<< " Despúes" << endl;
112             }
113             cout <<yellowb<< "-------------------------------------------------------" <<whiteb<<
        endl;
114         }
115     for(;lista.at(tmp)>=pivote;tmp++);
116     if (DEBUGMODE){
```

```
117                 cout <<redb<< "------------------------------------------------------------" << endl;
118                 cout << "Seguimos recorriendo el vector, posición del puntero de la Izquierda" <<
    whiteb<< endl;
119                 cout <<green<< " Valor del pivote: " << pivote << ", Posición del pivote: " <<
    limite_izq <<whiteb<< endl;
120                 for (unsigned int i=0; i<lista.size(); i++) {
121                       if (i==tmp)
122                             cout <<redb<< lista.at(i) <<whiteb;
123                       else
124                             cout << lista.at(i);
125
126                       if(i<lista.size()-1)
127                             cout << ",";
128                       else
129                             cout << endl;
130                 }
131                 cout << "En este momento el valor del puntero izquierdo es mas pequeño o igual que el
    valor del pivote" << endl;
132                 cout <<redb<< "------------------------------------------------------------" <<whiteb<< endl;
133           }
134       for(;lista.at(respuesta)<=pivote;respuesta--);
135           if (DEBUGMODE){
136                 cout <<blueb<< "------------------------------------------------------------" << endl;
137                 cout << "Seguimos recorriendo el vector, posición del puntero de la Derecha" <<whiteb<<
    endl;
138                 cout <<green<< " Valor del pivote: " << pivote << ", Posición del pivote: " << limite_izq
    <<whiteb<< endl;
139                 for (unsigned int i=0; i<lista.size(); i++) {
140                       if (i==respuesta)
141                             cout <<blueb<< lista.at(i) <<whiteb;
142                       else
143                             cout << lista.at(i);
144
145                       if(i<lista.size()-1)
146                             cout << ",";
147                       else
148                             cout << endl;
149                 }
150                 cout << "En este momento el valor del puntero derecho es mas grande o igual que el valor
    del pivote" << endl;
151                 cout <<blueb<< "------------------------------------------------------------" <<whiteb<< endl;
152           }
153     }
154     if (DEBUGMODE){
155           cout <<yellowb<< "------------------------------------------------------------" << endl;
156           cout << "Como los punteros se han cruzado lo que se hace" << endl;
157           cout << "es intercambiar la posición del pivote actual" << endl;
158           cout << "por la posición del puntero de la derecha" <<whiteb<< endl;
159           cout <<green<< "Valor del pivote: " << pivote << ", Posición del pivote: " << limite_izq
    <<whiteb<< endl;
160           for (unsigned int i=0; i<lista.size(); i++) {
161                 if (i==tmp)
162                       cout <<redb<< lista.at(i) <<whiteb;
163                 else if (i==respuesta)
164                       cout <<blueb<< lista.at(i) <<whiteb;
165                 else
166                       cout << lista.at(i);
167
168                 if(i<lista.size()-1)
169                       cout << ",";
170                 else
171                       cout <<green<< " Antés" <<whiteb<< endl;
172           }
173      }
174     intercambia(lista,limite_izq,respuesta);
175     if (DEBUGMODE){
176           for (unsigned int i=0; i<lista.size(); i++) {
177                 if (i==limite_izq)
178                       cout <<yellowb<< lista.at(i) <<whiteb;
179                 else if (i==respuesta)
180                       cout <<yellowb<< lista.at(i) <<whiteb;
181                 else
182                       cout << lista.at(i);
183
184                 if(i<lista.size()-1)
185                       cout << ",";
186                 else
187                       cout <<green<< " Despúes" <<whiteb<< endl;
```

```
188             }
189             cout <<yellowb<< "------------------------------------------------------------" <<whiteb<< endl;
190         }
191         return respuesta;
192 }
193
194 int seleccion(vector<int> &lista, int pos){
195         int pivote = 0;
196         int puntero_izq = 0;
197         int puntero_der = lista.size()-1;
198         while (pivote!=pos){
199             pivote = Pivote(lista,puntero_izq,puntero_der);
200             if (pos < pivote){
201                 if (DEBUGMODE){
202                     cout <<magentab<< "------------------------------------------------------------" <<whiteb<<
    endl;
203                     cout << "El valor que ha devuelto la funcion pivote() es: " << pivote << endl;
204                     cout << "esta es la posición actual que tiene el pivote cuyo valor es: " <<
    lista.at(pivote) << endl;
205                 }
206                 puntero_der = pivote-1;
207                 if (DEBUGMODE){
208                     cout << "Como la posición que buscamos es menor que la posición devuelta" <<
    endl;
209                     cout << "lo que se hace a continuación es partir el vector en 2" << endl;
210                     cout << "de tal forma que la posición devuelta menos una unidad" << endl;
211                     cout << "será el limite superior quedando el limite en: " << puntero_der << endl
    ;
212                     cout << "que tiene el valor: "<< lista.at(puntero_der) << endl;
213                     for (unsigned int i=0; i<lista.size(); i++) {
214                         if (i==puntero_der)
215                             cout <<greenb<< lista.at(i) <<whiteb;
216                         else
217                             cout << lista.at(i);
218
219                         if(i<lista.size()-1)
220                             cout << ",";
221                         else
222                             cout << endl;
223                     }
224                     cout <<magentab<< "------------------------------------------------------------" <<
    whiteb<< endl;
225                 }
226             }else if (pos > pivote){
227                 if (DEBUGMODE){
228                     cout <<magentab<< "------------------------------------------------------------" <<
    whiteb<< endl;
229                     cout << "El valor que ha devuelto la funcion pivote() es: " << pivote << endl;
230                     cout << "esta es la posición actual que tiene el pivote cuyo valor es: " <<
    lista.at(pivote) << endl;
231                 }
232                 puntero_izq = pivote+1;
233                 if (DEBUGMODE){
234                     cout << "Como la posición que buscamos es mayor que la posición devuelta" <<
    endl;
235                     cout << "lo que se hace a continuación es partir el vector en 2" << endl;
236                     cout << "de tal forma que el la posición devuelta mas una unidad" << endl;
237                     cout << "será el limite inferior quedando el limite en: " << puntero_izq << endl
    ;
238                     cout << "que tiene el valor: "<< lista.at(puntero_izq) << endl;
239                     for (unsigned int i=0; i<lista.size(); i++) {
240                         if (i==puntero_izq)
241                             cout <<greenb<< lista.at(i) <<whiteb;
242                         else
243                             cout << lista.at(i);
244
245                         if(i<lista.size()-1)
246                             cout << ",";
247                         else
248                             cout << endl;
249                     }
250                     cout <<magentab<< "------------------------------------------------------------" <<
    whiteb<< endl;
251                 }
252             }
253         }
254         return lista.at(pivote);
255 }
```

```
256
257 int main(int argc, const char * argv[]){
258      struct timespec t1, t2;
259      double tresultado;
260     int num=0;
261     if (argv[1]==NULL){
262      cout << "␣Cuantos␣numeros␣quiere␣generar:␣" << endl;
263      cin >> num;
264     }else
265      num=atoi(argv[1]);
266
267     // creamos el vector según el tamaño introducido
268     vector<int> lista(num);
269
270     // plantamos la semilla para los aleatorios
271     srand(num);
272
273     // inicializamos el vector con los datos
274 //     for (int i=0; i<num; i++){
275 //         lista.at(i)=rand()%100;
276 //     }
277     lista.at(0)=4;
278     lista.at(1)=1;
279     lista.at(2)=9;
280     lista.at(3)=7;
281     lista.at(4)=3;
282     lista.at(5)=2;
283     lista.at(6)=6;
284     lista.at(7)=8;
285     lista.at(8)=5;
286     // imprimimos la lista sin ordenar
287     if (DEBUGMODE){
288           cout <<whiteb<< "␣Lista␣Desordenada␣" << endl << "␣";
289           for (unsigned int i=0; i<lista.size(); i++) {
290                cout << lista.at(i);
291                if(i<lista.size()-1)
292                     cout << ",";
293                else
294                     cout << endl;
295           }
296           cout <<white<< endl;
297     }
298     // Medición del tiempo de ejecución del algorítmo quickSort();
299     clock_gettime(CLOCK_REALTIME, &t1);
300     int ordenado = seleccion(lista,lista.size()/2);
301     clock_gettime(CLOCK_REALTIME, &t2);
302     tresultado = (double) (t2.tv_sec - t1.tv_sec) + (double) ((t2.tv_nsec - t1.tv_nsec) / (1.e+9));
303
304     // si hémos ordenado la lista la mostramos
305     if (DEBUGMODE){
306           cout <<whiteb<< "␣Lista␣Modificada␣" << endl << "␣";
307           for (unsigned int i=0; i<lista.size(); i++) {
308                cout << lista.at(i);
309                if(i<lista.size()-1)
310                     cout << ",";
311                else
312                     cout << endl;
313           }
314           cout <<whiteb<< endl;
315           cout << "Valor␣del␣elemento␣seleccionado:␣" << ordenado << endl;
316           cout << "Tiempo␣empleado␣en␣la␣ordenación:␣" << setiosflags(ios::fixed) << setprecision(9)
      << tresultado <<normal<< endl;
317     }else{
318      cout <<whiteb<< "Valor␣del␣elemento␣seleccionado:␣" << ordenado << endl;
319      cout << setiosflags(ios::fixed) << setprecision(9) << tresultado <<normal<< endl;
320     }
321     return 0;
322 }
```