

```

1  #include "Methods.h"
2  #include <iostream>
3
4  void Methods::copyAll(const Methods& N){
5      int i=0;
6
7      while ( i <= N.last ){
8          this->AllMethods[i] = N.AllMethods[i];
9          i++;
10     }
11     this->last = N.last;
12 }
13
14 Methods::Methods() : last(-1) {}
15
16 bool Methods::isFull() {
17     return last == -1;
18 }
19
20 void Methods::setNumber(const RandomNumbers& n) {
21     AllMethods[++last] = n;
22 }
23
24 long int Methods::getNumber(const int& pos) {
25     return AllMethods[pos].getRandom();
26 }
27
28 void Methods::swapAllMethods(RandomNumbers& a, RandomNumbers& b) {
29     RandomNumbers aux;
30     aux = a;
31     a = b;
32     b = aux;
33 }
34 int Methods::getLastPos() {
35     return last;
36 }
37 ///Metodo Burbuja
38 void Methods::bubbleSort() {
39     int i(last), j;
40     bool flag;
41
42     do {
43         flag = false;
44         j = 0;
45         while(j < i) {
46             if(AllMethods[j].getRandom() > AllMethods[j+1].getRandom()) {
47                 swapAllMethods(AllMethods[j], AllMethods[j+1]);
48                 flag = true;
49             }
50             j++;
51         }
52         i--;
53     }
54     while(flag);
55 }
56 ///Metodo Shell
57 void Methods::shellSort() {
58     float fact(3.0/4.0);
59     int dif( (last + 1) * fact), lim, i;
60
61     while(dif>0) {
62         lim = last - dif;
63
64         i=0;
65         while( i <= lim ) {
66             if( AllMethods[i].getRandom() > AllMethods[i+dif].getRandom()) {

```

```

67         swapAllMethods(AllMethods[i],AllMethods[i+dif]);
68     }
69
70     i++;
71 }
72
73     dif*=fact;
74 }
75 }
76 ///Metodo insercion
77 void Methods::insertSort() {
78     int i(1), j;
79     RandomNumbers aux;
80
81     while(i <= last) {
82         aux=AllMethods[i];
83         j=i;
84         while( j > 0 and aux.getRandom() < AllMethods[j-1].getRandom() ) {
85             AllMethods[j] = AllMethods[j-1];
86             j--;
87         }
88         if(i!=j)
89             AllMethods[j]= aux;
90         i++;
91     }
92 }
93 ///Metodo seleccion
94 void Methods::selectionSort() {
95     int i(0), j, m;
96     while( i < last ) {
97         m = i;
98         j=i+1;
99         while(j<last) {
100             if(AllMethods[j].getRandom() < AllMethods[m].getRandom() )
101                 m = j;
102             j++;
103         }
104         if(m!=i)
105             swapAllMethods(AllMethods[i],AllMethods[m]);
106         i++;
107     }
108 }
109 ///Metodo Mezcla
110 void Methods::mergeSort() {
111     mergeSort(0,getLastPos());
112 }
113
114 void Methods::mergeSort(const int& leftEdge, const int& rightEdge) {
115
116     if( leftEdge >= rightEdge ) {
117         return;
118     }
119
120     int m((leftEdge + rightEdge)/2);
121
122     mergeSort(leftEdge,m);
123     mergeSort( m + 1, rightEdge);
124
125     for( int z(leftEdge) ; z <= rightEdge ; z++ ) {
126         temp[z] = AllMethods[z];
127     }
128
129     int i(leftEdge), j( m + 1 ), x(leftEdge);
130
131     while( i <= m and j <= rightEdge ) {
132         while( i <= m and temp[i].getRandom() <= temp[j].getRandom() ) {

```

```

133         AllMethods[x++] = temp[i++];
134     }
135     if( i <= m ) {
136         while( j <= rightEdge and temp[j].getRandom() <= temp[i].getRandom() ) {
137             AllMethods[x++] = temp[j++];
138         }
139     }
140 }
141 while ( i <= m ) {
142     AllMethods[x++] = temp[i++];
143 }
144 while( j <= rightEdge ) {
145     AllMethods[x++] = temp[j++];
146 }
147 }
148 //Metodo QuickSort
149 void Methods::quickSort() {
150     quickSort(0,getLastPos());
151 }
152
153 void Methods::quickSort(const int& leftEdge, const int& rightEdge) {
154     if( leftEdge >= rightEdge ) {
155         return;
156     }
157
158     int i(leftEdge), j(rightEdge);
159
160     while( i < j ) {
161         while( i < j and AllMethods[i].getRandom() <= AllMethods[rightEdge].getRandom() ) {
162             i++;
163         }
164         while( i < j and AllMethods[j].getRandom() >= AllMethods[rightEdge].getRandom() ) {
165             j--;
166         }
167         if( i != j ) {
168             swapAllMethods(AllMethods[i],AllMethods[j]);
169         }
170     }
171     if( i != rightEdge ) {
172         swapAllMethods(AllMethods[i],AllMethods[rightEdge]);
173     }
174
175     quickSort(leftEdge, i - 1);
176     quickSort(i + 1, rightEdge);
177 }
178 Methods& Methods::operator=(const Methods& n){
179     copyAll(n);
180     return *this;
181 }

```