```cpp
//Dericson Pablo e Gabriel Simioni

#include <iostream>
#include <algorithm>
#include <locale.h>
#include <time.h>
#define TAM 10

using namespace std;

int vetor[TAM];
float mediaCompBuscaSeq = 0, mediaCompBuscaBin = 0, mediaCompBuscaArvB = 0,
mediaCompBuscaArvAVL = 0;

struct no
{
    int dado;
    struct no *esq;
    struct no *dir;
    int bal;
};

struct no *raiz, *raiz_avl;

void OrdenaVetor()
{
    //BUBBLE SORT

    int pass = 1, aux,i;
    bool sorted = false;
    while (sorted == false && pass < TAM)
    {
        sorted = true;
        for (i=0; i<=TAM-pass-1; i++)
        {
            if (vetor[i] > vetor[i+1])
            {
                aux = vetor[i];
                vetor[i] = vetor[i+1];
                vetor[i+1] = aux;
                sorted = false;
            }
        }
        pass = pass +1;
    }

}

void ImprimeVetor()
{

    for(int i = 0; i < TAM; i++)
        cout << vetor[i] << endl;

}

void GeraVetor()
{

    for(int i = 0; i < TAM ; i++)
    {
        vetor[i] = rand() % 100;
    }

}

```

```
 66  // Rotação para a esquerda
 67  void esquerda(struct no *p)
 68  {
 69
 70      struct no *q, *hold;
 71
 72      q = p->dir;
 73      hold = q->esq;
 74      q->esq = p;
 75      p->dir = hold;
 76
 77  }
 78
 79  //Rotação para a direita
 80  void direita(struct no *p)
 81  {
 82
 83      struct no *q, *hold;
 84
 85      q = p->esq;
 86      hold = q->dir;
 87      q->dir = p;
 88      p->esq = hold;
 89
 90  }
 91
 92  // Cira um no e preenche os membros
 93  struct no *cria_no(int valor)
 94  {
 95      struct no *aux = new (struct no);
 96      aux->dado=valor;
 97      aux->dir=NULL;
 98      aux->esq=NULL;
 99      aux->bal=0;
100      return aux;
101  };
102
103  void insere_bal(int chave)
104  {
105
106      struct no *pp=NULL, *p=raiz_avl, *pajovem=NULL, *ajovem=raiz_avl, *q, *
filho;
107      int imbal;
108
109      if (p==NULL)                /* Arvore vazia */
110      {
111          raiz_avl = cria_no(chave);  /* Funcao para criacao de um novo no */
112          return;
113      }
114
115      /* Insere chave e descobre ancestral mais jovem a ser desbalanceado */
116      while (p!=NULL)
117      {
118          if (chave < p->dado)
119              q = p->esq;
120          else q = p->dir;
121          if (q!=NULL)
122              if (q->bal != 0)
123              {
124                  pajovem=p;
125                  ajovem=q;
126              }
127          pp = p;
128          p = q;
129      }
130
```

```
131       q = cria_no(chave);
132
133       if (chave<pp->dado)
134           pp->esq=q;
135       else pp->dir=q;
136
137       /* Balanceamento de todos os nós entre ajovem e q devem ser ajustados */
138       if (chave<ajovem->dado)
139           filho = ajovem->esq;
140       else filho = ajovem->dir;
141
142       p = filho;
143
144       while (p!=q)
145       {
146           if (chave < p->dado)
147           {
148               p->bal=1;
149               p=p->esq;
150           }
151           else
152           {
153               p->bal = -1;
154               p=p->dir;
155           }
156       }
157
158       if (chave<ajovem->dado)
159           imbal = 1;
160       else imbal = -1;
161
162       if (ajovem->bal==0)         /*Não houve desbalanceamento */
163       {
164           ajovem->bal=imbal;
165           return;
166       }
167
168
169       if (ajovem->bal!=imbal)      /*Não houve desbalanceamento */
170       {
171           ajovem->bal=0;
172           return;
173       }
174
175       /* Houve desbalanceamento */
176       if (filho->bal == imbal)
177       {
178           p=filho;
179           if (imbal==1)         /* Faz rotação simples */
180               direita(ajovem);
181           else esquerda(ajovem);
182           ajovem->bal=0;
183           filho->bal=0;
184       }
185       else
186       {
187           if (imbal==1)      /*Faz rotação dupla */
188           {
189               p=filho->dir;
190               esquerda(filho);
191               ajovem->esq=p;
192               direita(ajovem);
193           }
194           else
195           {
196               p=filho->esq;
```

```
197             direita(filho);
198             ajovem->dir=p;
199             esquerda(ajovem);
200         }
201         if (p->bal==0)
202         {
203             ajovem->bal=0;
204             filho->bal=0;
205         }
206         else
207         {
208             if (p->bal == imbal)
209             {
210                 ajovem->bal = - imbal;
211                 filho->bal = 0;
212             }
213             else
214             {
215                 ajovem->bal = 0;
216                 filho->bal = imbal;
217             }
218         }
219         p->bal=0;
220     }
221
222     if (pajovem == NULL)  /* Ajusta ponteiro do pai do ancestral mais jovem
*/
223         raiz_avl = p;
224     else if (ajovem==pajovem->dir)
225         pajovem->dir = p;
226     else pajovem->esq = p;
227
228     return;
229 }
230
231 //Insere valor em uma árvore binária
232 void insere(int valor)
233 {
234
235     struct no *atual, *aux;
236
237     aux = new(struct no);
238     aux->dado = valor;
239     aux->esq = NULL;
240     aux->dir = NULL;
241
242     atual=raiz;
243
244     if (atual==NULL)
245     {
246         raiz = aux;
247         return;
248     }
249
250     while (1)
251     {
252         if (valor < atual->dado)
253             if (atual->esq==NULL)
254             {
255                 atual->esq=aux;
256                 return;
257             }
258             else atual=atual->esq;
259         else if (atual->dir==NULL)
260         {
261             atual->dir=aux;
```

```
262             return;
263         }
264         else atual=atual->dir;
265     }
266 }
267
268 void buscaArvoreBinaria(int valor)
269 {
270
271     struct no *atual;
272     atual = raiz;
273     int cont = 0;
274     //cout << "EXECUTANDO BUSCA EM ÁRVORE BINÁRIA" << endl;
275
276     while(atual != NULL)
277     {
278         if(valor < atual -> dado)
279         {
280             cont++;
281             atual = atual->esq;
282         }
283         else if(valor > atual -> dado)
284         {
285             atual = atual -> dir;
286             cont += 2;
287         }
288         else
289         {
290             //cout<<"Encontrei o valor: "<< valor <<endl;
291             //cout << "numero de comparações busca em árvore binária: " <<
cont << endl;
292             cont +=2;
293             return;
294         }
295     }
296     mediaCompBuscaArvB += cont;
297     //cout << "numero de comparações busca em árvore binária: " << cont <<
endl;
298     //cout<<"Não encontrei o valor: "<< valor << endl;
299     //cout << endl;
300 }
301
302 void buscaArvoreAVL(int valor)
303 {
304
305     struct no *atual;
306     atual = raiz_avl;
307     int cont = 0;
308     //cout << "EXECUTANDO BUSCA EM ÁRVORE AVL" << endl;
309
310     while(atual != NULL)
311     {
312         if(valor < atual -> dado)
313         {
314             cont++;
315             atual = atual->esq;
316         }
317         else if(valor > atual -> dado)
318         {
319             atual = atual -> dir;
320             cont += 2;
321         }
322         else
323         {
324             //cout<<"Encontrei o valor: "<< valor <<endl;
325             //cout << "numero de comparações busca em árvore avl: " << cont
```

```
<< endl;
326             cont +=2;
327             return;
328         }
329     }
330     mediaCompBuscaArvAVL += cont;
331     //cout << "numero de comparações busca em árvore avl: " << cont << endl;
332     //cout<<"Não encontrei o valor: "<< valor << endl;
333     //cout << endl;
334 }
335
336
337
338 void em_ordem(struct no *atual)
339 {
340     if (atual!=NULL)
341     {
342         em_ordem(atual->esq);
343         cout << atual->dado << endl;
344         em_ordem(atual->dir);
345     }
346 }
347
348 void InsereVetorEmArvoreAVL()
349 {
350
351     for(int i = 0; i < TAM; i++)
352     {
353         insere_bal(vetor[i]);
354     }
355
356 }
357
358 void InsereVetorEmArvoreBinaria()
359 {
360
361     for(int i = 0; i < TAM; i++)
362     {
363         insere(vetor[i]);
364     }
365
366 }
367
368 void BuscaSequencial(int valor)
369 {
370
371     //cout << "EXECUTANDO BUSCA SEQUENCIAL" << endl;
372     int cont = 0;
373
374     for (int i = 0; i < TAM; i++)
375     {
376         if(vetor[i] == valor)
377         {
378             cont ++;
379             //cout << "Achei o valor: " << valor << endl;
380             //cout << "numero de comparações busca seq: " << cont << endl;
381             mediaCompBuscaSeq+= cont;
382             //cout << endl;
383             return;
384         }
385         cont++;
386     }
387     mediaCompBuscaSeq+= cont;
388     //cout << "numero de comparações busca seq: " << cont << endl;
389     //cout << "Nao achei o valor: " << valor << endl;
390     //cout << endl;
```

```
391  }
392
393  void BuscaAleatoria()
394  {
395
396      int valor = rand() % 100;
397      cout << "EXECUTANDO CHAVE DE BUSCA ALEATÓRIA" << endl;
398
399      for (int i = 0; i < TAM; i++)
400      {
401          if(vetor[i] == valor)
402          {
403              cout << "Achei o valor: " << valor << endl;
404              cout << endl;
405              return;
406          }
407      }
408
409  }
410
411  void BuscaBinaria(int valor)
412  {
413
414      int inicio = 0, fim = TAM-1, meio;
415      int cont = 0;
416
417      //cout << "EXECUTANDO BUSCA BINÁRIA" << endl;
418      while(inicio <= fim)
419      {
420          meio = (inicio+fim)/2;
421          if(vetor[meio] == valor)
422          {
423              //cout << "Achei o valor: " << valor << endl;
424              cont++;
425              // cout << "Numero de comparações busca binária: " << cont <<
endl;
426              mediaCompBuscaBin += cont;
427              //cout << endl;
428              return;
429          }
430          if(vetor[meio] > valor)
431          {
432              fim = meio-1;
433              cont+=2;
434          }
435          else
436          {
437              inicio = meio+1;
438              cont += 2;
439          }
440
441      }
442      //cout << "Numero de comparações busca binária: " << cont << endl;
443      mediaCompBuscaBin += cont;
444      //cout << "Nao achei o valor: " << valor << endl;
445      //cout << endl;
446  }
447
448  void calculaMediaBuscas()
449  {
450
451      mediaCompBuscaBin /= 100;
452      mediaCompBuscaArvAVL /= 100;
453      mediaCompBuscaArvB /= 100;
454      mediaCompBuscaSeq /= 100;
455      cout << "media de comparações da busca binária: " << mediaCompBuscaBin
```

```
<< endl;
456       cout << "media de comparações da busca sequencial: " <<
mediaCompBuscaSeq << endl;
457       cout << "media de comparações da busca em Árvore Binária: " <<
mediaCompBuscaArvB << endl;
458       cout << "media de comparações da busca em Árvore AVL: " <<
mediaCompBuscaArvAVL << endl;
459
460   }
461
462   int main()
463   {
464
465       setlocale(LC_ALL, "");
466       srand(time(NULL));
467       GeraVetor();
468       OrdenaVetor();
469       InsereVetorEmArvoreAVL();
470       InsereVetorEmArvoreBinaria();
471       for(int i = 0; i < 100; i++)
472       {
473           BuscaBinaria(rand() % 100);
474           BuscaSequencial(rand() % 100);
475           buscaArvoreBinaria(rand() % 100);
476           buscaArvoreAVL(rand() % 100);
477       }
478
479       calculaMediaBuscas();
480
481       return 0;
482   }
```