

```

1  /*****
2  Clientes:
3      Universidade Estadual do Maranhão
4      Centro de Ciências Tecnológicas
5      Departamento de Engenharia da Computação
6      Curso de Engenharia da Computação
7      Disciplina: Estrutura de Dados Básica (ASL092N321)
8      Semestre: 2022.2          Turma: 01
9      Professores:
10         Luís Carlos Fonseca
11         Pedro Brandão Neto
12
13  Autores:
14      Alunos:
15         Alexsandro Lucena Mota    Código: 20210024710
16
17  Título: Ordenação - Desafio 1
18
19  Propósito do Programa:
20
21      Crie uma programa para preencher arbitrariamente 3 vetores
22      com no mínimo 150000 números inteiros e então ordenar cada
23      um deles usando os métodos bolha, seleção e inserção. Crie um
24      menu de texto com as seguintes opções:
25          1) Preencher aleatoriamente o primeiro vetor
26          2) Preencher aleatoriamente o segundo vetor
27          3) Preencher aleatoriamente o terceiro vetor
28          4) ImprimirVetor 1
29          5) ImprimirVetor 2
30          6) ImprimirVetor 3
31          7) Busca Sequencial (vetor1)
32          8) Ordenação por trocas - bubbleSort - (vetor1)
33          9) Ordenação por seleção (vetor2)
34          10) Ordenação por inserção (vetor3)
35          11) Busca Binária (vetor 3)
36          12) Sair
37
38  Dados de Manutenção do programa
39
40      Data          Programador          Descrição da Mudança
41      =====          =====          =====
42      2022/13/10    Alexsandro Lucena Mota    - Código original (versão 0.1).
43  *****/
44
45  #include <stdio.h>    /*> Entrada e Saída de Dados          */
46  #include <stdlib.h>   /*> Para o uso da função rand e srand.          */
47  #include <time.h>     /*> Para mudar o ponto de partida da função srand()    */
48  #define tam 150000    /*> Tamanho dos vetores          */
49
50  /*****/
51  void preencher_vetor(int *vetor, int dim){ // Preenche um vetor
52      srand(time(NULL)); // Muda o ponto de partida de rand a cada nova execução.
53      for(int i = 0; i < dim; i++){ // Varre cada posição do vetor          */
54          *(vetor + i) = rand()%100; // Atribui valor aleatório de 0 até 99.
55      }
56  }
57  /*-----*/
58  void imprimir_vetor(int *vetor, int dim){
59      for(int i = 0; i < dim; i++){
60          printf("%d ", *(vetor + i));
61      }
62  };
63  /*-----*/
64  int busca_sequencial(int valor, int *vetor, int dim){
65      printf("\n");
66      for(int i = 0; i < dim; i++){

```

```

67         if(valor == *(vetor + i)){
68             return i;
69         }
70     }
71     return -1;
72 };
73 /*-----*/
74 void bubblesort(int *vetor, int dim){// Ordenação por troca
75     for(int i = 0; i < dim; i++){
76         for(int j = 0; j < dim - i; j++){
77             if( *(vetor + j) > *(vetor + j + 1) ){
78                 int x = *(vetor + j);
79                 *(vetor + j) = *(vetor + j + 1);
80                 *(vetor + j + 1) = x;
81             }
82         }
83     }
84 }
85 /*-----*/
86 int menor_item(int *vetor, int dim, int enesimo_item){
87     int minitem = enesimo_item; // item mínimo recebe o item inicial (valor)
88     for(int j = enesimo_item + 1; j < dim; j++){
89         if( *(vetor + minitem) > *(vetor + j) ){
90             minitem = j; // item mínimo recebe o índice do menor valor da lista
91         }
92     }
93     return minitem;
94 }
95 /*-----*/
96 void selectionsort(int *vetor, int dim){
97     int x = 0, minitem = 0;
98     for(int i = 0; i < dim - 1; i++){
99         minitem = menor_item(vetor, dim, i);
100        x = *(vetor + i);
101        *(vetor + i) = *(vetor + minitem);
102        *(vetor + minitem) = x;
103    }
104 };
105 /*-----*/
106 void inserctionsort(int *vetor, int dim){
107     int copia, indice;
108     for(int i = 1; i < dim; i++){
109         copia = *(vetor + i);
110         indice = i - 1;
111         do{
112             *(vetor + indice + 1) = *(vetor + indice);
113             indice--;
114         }while( (indice >= 0) && ( copia < *(vetor + indice) ) );
115         *(vetor + indice + 1) = copia;
116     }
117 }
118 /*-----*/
119 int busca_binaria(int valor,int *vetor, int dim){
120     printf("\n");
121     int inicio = 0, fim = dim - 1, pmedia = 0;
122     do{
123         pmedia = (fim + inicio)/2;
124         if(valor == *(vetor + pmedia)){
125             return pmedia;
126         }else if(valor < *(vetor + pmedia)){
127             fim = pmedia - 1;
128         }else{
129             inicio = pmedia + 1;
130         }
131     }while(inicio <= fim);
132     return -1;

```

```

133 };
134 /*-----*/
135 int menu_select(){
136     int ch = 0;
137     do{
138         printf("Menu de Opcoes\n");
139         printf("\t 1 - Preencher aleatoriamente o primeiro vetor\n");
140         printf("\t 2 - Preencher aleatoriamente o segundo vetor\n");
141         printf("\t 3 - Preencher aleatoriamente o terceiro vetor\n");
142         printf("\t 4 - Imprimir vetor 1\n");
143         printf("\t 5 - Imprimir vetor 2\n");
144         printf("\t 6 - Imprimir vetor 3\n");
145         printf("\t 7 - Busca Sequencial (vetor 1)\n");
146         printf("\t 8 - Ordenação por trocas - bubbleSort - (vetor1)\n");
147         printf("\t 9 - Ordenação por seleção (vetor2)\n");
148         printf("\t10 - Ordenação por inserção (vetor3)\n");
149         printf("\t11 - Busca Binária (vetor 3)\n");
150         printf("\t12 - Sair.\n");
151         printf("Entre com a opcao desejada: ");
152         scanf("%d",&ch); /** Lê do teclado a seleção */
153         if((ch < 1) || (ch > 12)){
154             printf("\nOpcao Invalida! Tente novamente.\n");
155         }
156     }while( (ch < 1) || (ch > 12) );
157     return ch;
158 }
159 /*****/
160 int main(){
161     printf("PROGRAMA TRES VETORES: ORDENACAO - Burbble, Selection e Insection\n");
162     int choose = 0, *vetor1 = NULL, *vetor2 = NULL, *vetor3 = NULL;
163     int elemento_vetor = 0, retorno = 0;
164     vetor1 = (int *) malloc(tam*sizeof(int));
165     if(vetor1 == NULL){ // teste de alocação de memória
166         printf("Erro: Memória Insuficiente!\n");
167         exit(1);
168     }
169     vetor2 = (int *) malloc(tam*sizeof(int));
170     if(vetor2 == NULL){ // teste de alocação de memória
171         printf("Erro: Memória Insuficiente!\n");
172         exit(1);
173     }
174     vetor3 = (int *) malloc(tam*sizeof(int));
175     if(vetor3 == NULL){ // teste de alocação de memória
176         printf("Erro: Memória Insuficiente!\n");
177         exit(1);
178     }
179     for(int i = 0; i < tam; i++){
180         *(vetor1 + i) = NULL;
181         *(vetor2 + i) = NULL;
182         *(vetor3 + i) = NULL;
183     }
184     do{
185         printf("\n");
186         choose = menu_select();
187         switch(choose){
188             case 1:
189                 printf("\n");
190                 preencher_vetor(vetor1,tam);
191                 printf("Vetor preechido com sucesso!\n");
192                 system("pause");
193                 break;
194             case 2:
195                 printf("\n");
196                 preencher_vetor(vetor2,tam);
197                 printf("Vetor preechido com sucesso!\n");
198                 system("pause");

```

```

199         break;
200     case 3:
201         printf("\n");
202         preencher_vetor(vetor3,tam);
203         printf("Vetor preenchido com sucesso!\n");
204         system("pause");
205         break;
206     case 4:
207         printf("\n");
208         printf("vetor_1 = { ");
209         imprimir_vetor(vetor1,tam);
210         printf("}\n");
211         system("pause");
212         break;
213     case 5:
214         printf("\n");
215         printf("vetor_2 = { ");
216         imprimir_vetor(vetor2,tam);
217         printf("}\n");
218         system("pause");
219         break;
220     case 6:
221         printf("\n");
222         printf("vetor_3 = { ");
223         imprimir_vetor(vetor3,tam);
224         printf("}\n");
225         system("pause");
226         break;
227     case 7:
228         printf("Informe o valor que deseja buscar: ");
229         scanf("%d",&elemento_vetor);
230         retorno = busca_sequencial(elemento_vetor,vetor1,tam);
231         if(retorno == -1){
232             printf("Valor nao encontrado. ");
233             printf("Nao eh uma componente deste vetor.\n");
234         }else{
235             printf("valor encontrado na posicao %d do vetor: ", retorno);
236             printf("v[%d] = %d.\n", retorno, *(vetor1 + retorno));
237         }
238         system("pause");
239         break;
240     case 8:
241         bubblesort(vetor1, tam);
242         printf("Ordenação realizada como sucesso!\n");
243         system("pause");
244         break;
245     case 9:
246         selectionsort(vetor2, tam);
247         printf("Ordenação realizada como sucesso!\n");
248         system("pause");
249         break;
250     case 10:
251         inserctionsort(vetor3, tam);
252         printf("Ordenação realizada como sucesso!\n");
253         system("pause");
254         break;
255     case 11:
256         printf("Informe o valor que deseja buscar: ");
257         scanf("%d",&elemento_vetor);
258         retorno = busca_binaria(elemento_vetor,vetor3,tam);
259         if(retorno == -1){
260             printf("Valor nao encontrado. ");
261             printf("Nao eh uma componente deste vetor.\n");
262         }else{
263             printf("valor encontrado na posicao %d do vetor: ", retorno);
264             printf("vetor[%d] = %d.\n", retorno, *(vetor3 + retorno));

```

```
265         }
266         system("pause");
267         break;
268     case 12:
269         printf("\nVoce escolheu sair do programa.\n");
270         exit(0); //return 0;
271     }
272 }while(choose != 12);
273 free(vetor1);
274 free(vetor2);
275 free(vetor3);
276 return 0;
277 }
278 /*****/
```