```c
/*******************************************************************************
    Clientes:
        Universidade Estadual do Maranhão
        Centro de Ciências Tecnológicas
        Departamento de Engenharia da Computação
        Curso de Engenharia da Computação
        Disciplina: Estrutura de Dados Básica (ASL092N321)
        Semestre: 2022.2         Turma: 01
        Professores:
            Lúis Carlos Fonseca
            Pedro Brandão Neto

    Autores:
        Alunos:
            Alexsandro Lucena Mota   Código: 20210024710

    Título: Ordenação - Desafio 1

    Propósito do Programa:

        Crie uma programa para preencher arbitrariamente 3 vetores
        com no mínimo 150000 números inteiros e então ordenar cada
        um deles usando os métodos bolha, seleção e inserção. Crie um
        menu de texto com as seguintes opções:
             1) Preencher aleatoriamente o primeiro vetor
             2) Preencher aleatoriamente o segundo vetor
             3) Preencher aleatoriamente o terceiro vetor
             4) ImprimirVetor 1
             5) ImprimirVetor 2
             6) ImprimirVetor 3
             7) Busca Sequencial (vetor1)
             8) Ordenação por trocas - bubbleSort - (vetor1)
             9) Ordenação por seleção (vetor2)
            10) Ordenação por inserção (vetor3)
            11) Busca Binária (vetor 3)
            12) Sair

    Dados de Manutenção do programa

      Data            Programador                 Descrição da Mudança
    ==========    ====================      ================================
    2022/13/10    Alexsandro Lucena Mota      - Código original (versão 0.1).
*******************************************************************************/

#include <stdio.h>  /*> Entrada e Saída de Dados                              */
#include <stdlib.h> /*> Para o uso da função rand e srand.                    */
#include <time.h>   /*> Para mudar o ponto de partida da função srand()      */
#define tam 150000      /*> Tamanho dos vetores                                 */

/*******************************************************************************/
void preencher_vetor(int *vetor,int dim){ // Preenche um vetor
    srand(time(NULL)); // Muda o ponto de partida de rand a cada nova execução.
    for(int i = 0; i < dim; i++){ // Varre cada posição do vetor    */
        *(vetor + i) = rand()%100; // Atribui valor aleatório de 0 até 99.
    }
}
/*-----------------------------------------------------------------------------*/
void imprimir_vetor(int *vetor,int dim){
    for(int i = 0; i < dim; i++){
        printf("%d ",*(vetor + i));
    }
};
/*-----------------------------------------------------------------------------*/
int busca_sequencial(int valor, int *vetor, int dim){
    printf("\n");
    for(int i = 0; i < dim; i++){
        if(valor == *(vetor + i)){
            return i;
        }
    }
    return -1;
};
```

```c
/*-------------------------------------------------------------------------------*/
void bublesoft(int *vetor, int dim){// Ordenação por troca
    for(int i = 0; i < dim; i++){
        for(int j = 0; j < dim - i; j++){
            if( *(vetor + j) > *(vetor + j + 1) ){
                int x = *(vetor + j);
                *(vetor + j ) = *(vetor + j + 1);
                *(vetor + j + 1) = x;
            }
        }
    }
}
/*-------------------------------------------------------------------------------*/
int menor_item(int *vetor, int dim, int enesimo_item){
    int minitem = enesimo_item; // item mínimo recebe o item inicial (valor)
    for(int j = enesimo_item + 1; j < dim; j++){
        if( *(vetor + minitem) > *(vetor + j) ){
            minitem = j; // item mínimo recebe o indice do menor valor da lista
        }
    }
    return minitem;
}
/*.....................................................................*/
void selectionsort(int *vetor, int dim){
    int x = 0, minitem = 0;
    for(int i = 0; i < dim - 1; i++){
        minitem = menor_item(vetor, dim, i);
        x = *(vetor + i);
        *(vetor + i) = *(vetor + minitem);
        *(vetor + minitem) = x;
    }
};
/*-------------------------------------------------------------------------------*/
void inserctionsort(int *vetor, int dim){
    int copia, indice;
    for(int i = 1; i < dim; i++){
        copia = *(vetor + i);
        indice = i - 1;
        do{
            *(vetor + indice + 1) = *(vetor + indice);
            indice--;
        }while( (indice >= 0) && ( copia < *(vetor + indice) ) );
        *(vetor + indice + 1) = copia;
    }
}
/*-------------------------------------------------------------------------------*/
int busca_binaria(int valor,int *vetor, int dim){
    printf("\n");
    int inicio = 0, fim = dim - 1, pmedia = 0;
    do{
        pmedia = (fim + inicio)/2;
        if(valor == *(vetor + pmedia)){
            return pmedia;
        }else if(valor < *(vetor + pmedia)){
            fim = pmedia - 1;
        }else{
            inicio = pmedia + 1;
        }
    }while(inicio <= fim);
    return -1;
};
/*-------------------------------------------------------------------------------*/
int menu_select(){
    int ch = 0;
    do{
        printf("Menu de Opcoes\n");
        printf("\t 1 - Preencher aleatoriamente o primeiro vetor\n");
        printf("\t 2 - Preencher aleatoriamente o segundo vetor\n");
        printf("\t 3 - Preencher aleatoriamente o terceiro vetor\n");
        printf("\t 4 - Imprimir vetor 1\n");
        printf("\t 5 - Imprimir vetor 2\n");
        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 - Burble, 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 preechido 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                 bublesoft(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 /*****************************************************************************/
279
```