Universidade Federal de São João del Rei
Departamento de Ciência da Computação
Curso de Ciência da Computação

Roteiro 10

Adélson de Oliveira Carmo Júnior
212050019

1 Algoritmos de Ordenação

1.1 Reimplentação

Código

```c
#ifndef ORDENACAO_PT1
#define ORDENACAO_PT1

int getComp();
int getMov();
void setComp(int);
void setMov(int);
int* copiaVetor(int*, int);
void imprimeVetor(int*, int);
void preencheAleatorio(int*, int, int,
    int);
void troca(int*, int *);
void BubbleSort(int *, int);
void BubbleSort2(int *, int);
void InsertionSort(int *, int);
void SelectionSort(int *, int);

#endif
```

codigos/questao11/questao11.h

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include "questao11.h"

//Medidas de Complexidade
int comp; //Num. de comparacoes
int mov; //Num. de movimentacoes


int getComp(){
    return comp;
}

int getMov(){
    return mov;
}

void setComp(int valor){
    comp = valor;
}

void setMov(int valor){
    mov = valor;
}

int* copiaVetor(int* v, int n){
    int i;
    int *v2;
    v2 = (int*) malloc (n*sizeof(int));
    for(i=0; i<n; i++) v2[i] = v[i];
    return v2;
}

void imprimeVetor(int* v, int n){
    int i, prim = 1;
    printf("[");
    for(i=0; i<n; i++)
        if(prim){ printf("%d", v[i]);
            prim = 0; }
        else printf(", %d", v[i]);
    printf("]\n");
}

```

```c
void preencheAleatorio(int* v, int n,
    int ini, int fim){
    int i;
    for(i=0; i<n; i++)
        v[i] = ini + rand() % (fim-ini
            + 1);
}

void troca(int* a, int *b){
  int aux = *a;
  *a = *b;
  *b = aux;
}

void BubbleSort(int *v, int n){
    int i, j;
    for(i=0;i<n-1;i++)
        for(j=0;j<n-i-1;j++){
       comp++;
             if (v[j]>v[j+1]) {
                 troca(&v[j], &v[j+1]);
         mov++;
             }
        }
}

void BubbleSort2(int *v, int n){
    int j, continua, fim = n;
    do{
        continua = 0;
        for(j=0; j < fim-1; j++){
       comp++;
             if (v[j]>v[j+1]) {
                 troca(&v[j], &v[j+1]);
         mov++;
                 continua = j;
             }
        fim--;
    }while(continua != 0);
}

void InsertionSort(int *v, int n){
  int i, j, atual;
  for(i=1; i < n; i++){
    atual = v[i];
    comp++;
    for(j=i; (j>0) && (atual < v[j-1]);
        j--){
       v[j] = v[j-1];
       comp++;
       mov++;
    }
    v[j] = atual;
  }
}

void SelectionSort(int *v, int n){
  int i, j, menor;
  for(i=0; i < n-1; i++){
    menor = i;
    for(j=i+1; j < n; j++){
       comp++;
       if (v[j] < v[menor])
         menor = j;
    }
    if(i != menor){
      troca(&v[i], &v[menor]);
      mov++;
    }
  }
}
```

codigos/questao11/questao11.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include "questao11.h"

int main(){

    //Atribuicoes iniciais
    srand(time(NULL));
    setComp(0);
    setMov(0);
    clock_t t;

    int *v1, *v2, *v3;
    int n;

    /* Recebe valores e preenche */
    printf("Digite o tamanho do
        vetor:\n");
    scanf("%d", &n);
    v1 = (int*) malloc (n*sizeof(int));

    preencheAleatorio(v1, n, 1, 100);
    imprimeVetor(v1, n);

    v2 = copiaVetor(v1, n);
    v3 = copiaVetor(v1, n);

    /* BubbleSort */
    t = clock();
    BubbleSort(v1, n);
    t = clock() - t;
    printf("\nInformacoes da Ordenacao
        por BubbleSort:\n");
    printf("Tempo Execucao:  %f
        seconds.\n",
```

```
        ((float)t)/CLOCKS_PER_SEC);
34      printf("Comparacoes: %d\n",
            getComp());
35      printf("Movimentacoes: %d\n",
            getMov());
36      printf("Memoria (bytes): %ld\n",
            n*sizeof(int));
37
38      imprimeVetor(v1, n);
39
40      /* InsertionSort */
41      setComp(0);
42      setMov(0);
43      t = clock();
44      InsertionSort(v2, n);
45      t = clock() - t;
46      printf("\nInformacoes da Ordenacao
            por InsertionSort:\n");
47      printf("Tempo Execucao:  %f
            seconds.\n",
            ((float)t)/CLOCKS_PER_SEC);
48      printf("Comparacoes: %d\n",
            getComp());
49      printf("Movimentacoes: %d\n",
            getMov());
50      printf("Memoria (bytes): %ld\n",
            n*sizeof(int));
51
52      imprimeVetor(v2, n);
53
54      /* SelectionSort */
55      setComp(0);
56      setMov(0);
57      t = clock();
58      SelectionSort(v3, n);
59      t = clock() - t;
60      printf("\nInformacoes da Ordenacao
            por SelectionSort:\n");
61      printf("Tempo Execucao:  %f
            seconds.\n",
            ((float)t)/CLOCKS_PER_SEC);
62      printf("Comparacoes: %d\n",
            getComp());
63      printf("Movimentacoes: %d\n",
            getMov());
64      printf("Memoria (bytes): %ld\n",
            n*sizeof(int));
65
66      imprimeVetor(v3, n);
67
68      free(v1);
69      free(v2);
70      free(v3);
71
72      return 0;
73  }
```

codigos/questao11/main.c

```
1  all: questao11.o
2    gcc questao11.o main.c -o main
3
4  questao11.o: questao11.h questao11.c
5    gcc -c questao11.c
6
7  clean:
8    rm -f questao11.o main
```

codigos/questao11/Makefile

Saída

```
Digite o tamanho do vetor:
10
[11, 83, 88, 8, 95, 37, 71, 95, 97, 36]

Informacoes da Ordenacao por BubbleSort:
Tempo Execucao:  0.000015 seconds.
Comparacoes: 45
Movimentacoes: 16
Memoria (bytes): 40
[8, 11, 36, 37, 71, 83, 88, 95, 95, 97]
```

Figura 1: Questão 1.1 - Saída 1

```
Informacoes da Ordenacao por InsertionSort:
Tempo Execucao:  0.000012 seconds.
Comparacoes: 25
Movimentacoes: 16
Memoria (bytes): 40
[8, 11, 36, 37, 71, 83, 88, 95, 95, 97]

Informacoes da Ordenacao por SelectionSort:
Tempo Execucao:  0.000013 seconds.
Comparacoes: 45
Movimentacoes: 7
Memoria (bytes): 40
[8, 11, 36, 37, 71, 83, 88, 95, 95, 97]
```

Figura 2: Questão 1.1 - Saída 2

1.2 Aplicando modificações para decrescer

Código

```c
#ifndef ORDENACAO_PT1
#define ORDENACAO_PT1

int getComp();
int getMov();
void setComp(int);
void setMov(int);
int* copiaVetor(int*, int);
void imprimeVetor(int*, int);
void preencheAleatorio(int*, int, int,
    int);
void troca(int*, int *);
void BubbleSort(int *, int);
void BubbleSort2(int *, int);
void InsertionSort(int *, int);
void SelectionSort(int *, int);

#endif
```

codigos/questao12/questao12.h

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include "questao12.h"

//Medidas de Complexidade
int comp; //Num. de comparacoes
int mov; //Num. de movimentacoes


int getComp(){
    return comp;
}

int getMov(){
    return mov;
}

void setComp(int valor){
    comp = valor;
}

void setMov(int valor){
    mov = valor;
}

int* copiaVetor(int* v, int n){
    int i;
    int *v2;
    v2 = (int*) malloc (n*sizeof(int));
    for(i=0; i<n; i++) v2[i] = v[i];
    return v2;
}

void imprimeVetor(int* v, int n){
    int i, prim = 1;
    printf("[");
    for(i=0; i<n; i++)
        if(prim){ printf("%d", v[i]);
            prim = 0; }
        else printf(", %d", v[i]);
    printf("]\n");
}

void preencheAleatorio(int* v, int n,
    int ini, int fim){
    int i;
    for(i=0; i<n; i++)
        v[i] = ini + rand() % (fim-ini
            + 1);
}

void troca(int* a, int *b){
  int aux = *a;
  *a = *b;
  *b = aux;
}

void BubbleSort(int *v, int n) {
    int i, j;
    for (i = 0; i < n - 1; i++) {
        for (j = 0; j < n - i - 1; j++)
            {
            comp++;
            if (v[j] < v[j + 1]) {
                troca(&v[j], &v[j + 1]);
                mov++;
            }
        }
    }
}

// Bubble Sort 2
void BubbleSort2(int *v, int n) {
    int j, continua, fim = n;
    do {
        continua = 0;
        for (j = 0; j < fim - 1; j++) {
            comp++;
            if (v[j] < v[j + 1]) {
                troca(&v[j], &v[j + 1]);
                mov++;
```

```
                continua = j;
            }
        }
        fim--;
    } while (continua != 0);
}

// Insertion Sort
void InsertionSort(int *v, int n) {
    int i, j, atual;
    for (i = 1; i < n; i++) {
        atual = v[i];
        comp++;
        for (j = i; (j > 0) && (atual < v[j - 1]); j--) {
            v[j] = v[j - 1];
            comp++;
            mov++;
        }
        v[j] = atual;
    }
}

// Selection Sort
void SelectionSort(int *v, int n) {
    int i, j, maior;
    for (i = 0; i < n - 1; i++) {
        maior = i;
        for (j = i + 1; j < n; j++) {
            comp++;
            if (v[j] > v[maior])
                maior = j;
        }
        if (i != maior) {
            troca(&v[i], &v[maior]);
            mov++;
        }
    }
}
```

codigos/questao12/questao12.c

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include "questao12.h"

int main(){

    //Atribuicoes iniciais
    srand(time(NULL));
    setComp(0);
    setMov(0);
    clock_t t;

    int *v1, *v2, *v3;
    int n;

    /* Recebe valores e preenche */
    printf("Digite o tamanho do vetor:\n");
    scanf("%d", &n);
    v1 = (int*) malloc (n*sizeof(int));

    preencheAleatorio(v1, n, 1, 100);
    imprimeVetor(v1, n);

    v2 = copiaVetor(v1, n);
    v3 = copiaVetor(v1, n);

    /* BubbleSort */
    t = clock();
    BubbleSort(v1, n);
    t = clock() - t;
    printf("\nInformacoes da Ordenacao por BubbleSort:\n");
    printf("Tempo Execucao:  %f seconds.\n", ((float)t)/CLOCKS_PER_SEC);
    printf("Comparacoes: %d\n", getComp());
    printf("Movimentacoes: %d\n", getMov());
    printf("Memoria (bytes): %ld\n", n*sizeof(int));

    imprimeVetor(v1, n);

    /* InsertionSort */
    setComp(0);
    setMov(0);
    t = clock();
    InsertionSort(v2, n);
    t = clock() - t;
    printf("\nInformacoes da Ordenacao por InsertionSort:\n");
    printf("Tempo Execucao:  %f seconds.\n", ((float)t)/CLOCKS_PER_SEC);
    printf("Comparacoes: %d\n", getComp());
    printf("Movimentacoes: %d\n", getMov());
    printf("Memoria (bytes): %ld\n", n*sizeof(int));

    imprimeVetor(v2, n);

    /* SelectionSort */
    setComp(0);
    setMov(0);
    t = clock();
```

```
58      SelectionSort(v3, n);
59      t = clock() - t;
60      printf("\nInformacoes da Ordenacao
            por SelectionSort:\n");
61      printf("Tempo Execucao:  %f
            seconds.\n",
            ((float)t)/CLOCKS_PER_SEC);
62      printf("Comparacoes: %d\n",
            getComp());
63      printf("Movimentacoes: %d\n",
            getMov());
64      printf("Memoria (bytes): %ld\n",
            n*sizeof(int));
65
66      imprimeVetor(v3, n);
67
68      free(v1);
69      free(v2);
70      free(v3);
71
72      return 0;
73  }
```

codigos/questao12/main.c

```
1  all: questao12.o
2    gcc questao12.o main.c -o main
3
4  questao12.o: questao12.h questao12.c
5    gcc -c questao12.c
6
7  clean:
8    rm -f questao12.o main
```

codigos/questao12/Makefile

Saída

```
Digite o tamanho do vetor:
10
[46, 46, 33, 49, 13, 62, 42, 20, 13, 37]

Informacoes da Ordenacao por BubbleSort:
Tempo Execucao:  0.000018 seconds.
Comparacoes: 45
Movimentacoes: 15
Memoria (bytes): 40
[62, 49, 46, 46, 42, 37, 33, 20, 13, 13]
```

Figura 3: Questão 1.1 - Saída 1

```
Informacoes da Ordenacao por InsertionSort:
Tempo Execucao:  0.000013 seconds.
Comparacoes: 24
Movimentacoes: 15
Memoria (bytes): 40
[62, 49, 46, 46, 42, 37, 33, 20, 13, 13]

Informacoes da Ordenacao por SelectionSort:
Tempo Execucao:  0.000017 seconds.
Comparacoes: 45
Movimentacoes: 7
Memoria (bytes): 40
[62, 49, 46, 46, 42, 37, 33, 20, 13, 13]
```

Figura 4: Questão 1.1 - Saída 2

1.3 Utilizando entradas grandes

Código

```
1  #ifndef ORDENACAO_PT1
2  #define ORDENACAO_PT1
3
4  int getComp();
5  int getMov();
6  void setComp(int);
7  void setMov(int);
8  int* copiaVetor(int*, int);
9  void imprimeVetor(int*, int);
10 void preencheAleatorio(int*, int, int,
       int);
11 void troca(int*, int *);
12 void BubbleSort(int *, int);
13 void BubbleSort2(int *, int);
14 void InsertionSort(int *, int);
15 void SelectionSort(int *, int);
16
17 #endif
```

codigos/questao13/questao13.h

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include "questao13.h"

//Medidas de Complexidade
int comp; //Num. de comparacoes
int mov; //Num. de movimentacoes


int getComp(){
    return comp;
}

int getMov(){
    return mov;
}

void setComp(int valor){
    comp = valor;
}

void setMov(int valor){
    mov = valor;
}

int* copiaVetor(int* v, int n){
    int i;
    int *v2;
    v2 = (int*) malloc (n*sizeof(int));
    for(i=0; i<n; i++) v2[i] = v[i];
    return v2;
}

void imprimeVetor(int* v, int n){
    int i, prim = 1;
    printf("[");
    for(i=0; i<n; i++)
        if(prim){ printf("%d", v[i]);
            prim = 0; }
        else printf(", %d", v[i]);
    printf("]\n");
}

void preencheAleatorio(int* v, int n,
    int ini, int fim){
    int i;
    for(i=0; i<n; i++)
        v[i] = ini + rand() % (fim-ini
            + 1);
}

void troca(int* a, int *b){
  int aux = *a;
  *a = *b;
  *b = aux;
}

void BubbleSort(int *v, int n){
    int i, j;
    for(i=0;i<n-1;i++)
        for(j=0;j<n-i-1;j++){
       comp++;
            if (v[j]>v[j+1]) {
                troca(&v[j], &v[j+1]);
         mov++;
            }
        }
}

void BubbleSort2(int *v, int n){
    int j, continua, fim = n;
    do{
        continua = 0;
        for(j=0; j < fim-1; j++){
       comp++;
            if (v[j]>v[j+1]) {
                troca(&v[j], &v[j+1]);
         mov++;
                continua = j;
            }
        }
        fim--;
    }while(continua != 0);
}

void InsertionSort(int *v, int n){
  int i, j, atual;
  for(i=1; i < n; i++){
    atual = v[i];
    comp++;
    for(j=i; (j>0) && (atual < v[j-1]);
        j--){
       v[j] = v[j-1];
       comp++;
       mov++;
    }
    v[j] = atual;
  }
}

void SelectionSort(int *v, int n){
  int i, j, menor;
  for(i=0; i < n-1; i++){
    menor = i;
    for(j=i+1; j < n; j++){
       comp++;
       if (v[j] < v[menor])
         menor = j;
    }
    if(i != menor){
      troca(&v[i], &v[menor]);
      mov++;
    }
  }
}
```

codigos/questao13/questao13.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include "questao13.h"

int main(int argc, char *argv[]){

    //Atribuicoes iniciais
    setComp(0);
    setMov(0);
    clock_t t;

    int n;

    /* Leitura de arquivo e verificacoes */
    if (argc != 2) {
        fprintf(stderr, "Uso: %s <nome_do_arquivo>\n", argv[0]);
        return 1;
    }

    FILE *arquivo = fopen(argv[1], "r");

    if (arquivo == NULL) {
        fprintf(stderr, "Erro ao abrir o arquivo %s.\n", argv[1]);
        return 1;
    }

    /* Pega o tamanho do arquivo e cria um vetor com esse tamanho */
    fscanf(arquivo, "%d", &n);

    int *v1 = (int*)malloc(n * sizeof(int));
    int *v2 = (int*)malloc(n * sizeof(int));
    int *v3 = (int*)malloc(n * sizeof(int));

    /* Confere  se o vetor nao esta vazio */
    if (v1 == NULL) {
        fprintf(stderr, "Erro ao alocar memoria para o vetor.\n");
        fclose(arquivo);
        return 1;
    }

    /* Copia os dados do arquivo para esse vetor */
    for (int i = 0; i < n; i++) {
        fscanf(arquivo, "%d", &v1[i]);
    }

    v2 = copiaVetor(v1, n);
    v3 = copiaVetor(v1, n);

    /* BubbleSort */
    t = clock();
    BubbleSort(v1, n);
    t = clock() - t;
    printf("\nInformacoes da Ordenacao por BubbleSort:\n");
    printf("Tempo Execucao:  %f seconds.\n", ((float)t)/CLOCKS_PER_SEC);
    printf("Comparacoes: %d\n", getComp());
    printf("Movimentacoes: %d\n", getMov());
    printf("Memoria (bytes): %ld\n", n*sizeof(int));

    /* InsertionSort */
    setComp(0);
    setMov(0);
    t = clock();
    InsertionSort(v2, n);
    t = clock() - t;
    printf("\nInformacoes da Ordenacao por InsertionSort:\n");
    printf("Tempo Execucao:  %f seconds.\n", ((float)t)/CLOCKS_PER_SEC);
    printf("Comparacoes: %d\n", getComp());
    printf("Movimentacoes: %d\n", getMov());
    printf("Memoria (bytes): %ld\n", n*sizeof(int));

    /* SelectionSort */
    setComp(0);
    setMov(0);
    t = clock();
    SelectionSort(v3, n);
    t = clock() - t;
    printf("\nInformacoes da Ordenacao por SelectionSort:\n");
    printf("Tempo Execucao:  %f seconds.\n", ((float)t)/CLOCKS_PER_SEC);
    printf("Comparacoes: %d\n", getComp());
    printf("Movimentacoes: %d\n", getMov());
    printf("Memoria (bytes): %ld\n", n*sizeof(int));

    free(v1);
    free(v2);
    free(v3);

    fclose(arquivo);

```

```
90      return 0;
91  }
```

codigos/questao13/main.c

```
1  all: questao13.o
2    gcc questao13.o main.c -o main
3
4  questao13.o: questao13.h questao13.c
5    gcc -c questao13.c
6
7  clean:
8    rm -f questao13.o main
```

codigos/questao13/Makefile

Saída:

Devido à expressiva quantidade de arquivos de entrada e considerando o tempo significativo que seria demandado para processar um milhão de elementos, optou-se por utilizar exclusivamente os conjuntos que continham o segundo maior número de elementos, totalizando cem mil números.

```
Informacoes da Ordenacao por BubbleSort:
Tempo Execucao:  47.829521 seconds.
Comparacoes: 704982704
Movimentacoes: 704982704
Memoria (bytes): 400000

Informacoes da Ordenacao por InsertionSort:
Tempo Execucao:  23.967834 seconds.
Comparacoes: 705082703
Movimentacoes: 704982704
Memoria (bytes): 400000

Informacoes da Ordenacao por SelectionSort:
Tempo Execucao:  18.944656 seconds.
Comparacoes: 704982704
Movimentacoes: 50000
Memoria (bytes): 400000
```

Figura 5: Questão 2.1 - Ordem decrescente

```
Informacoes da Ordenacao por BubbleSort:
Tempo Execucao:  58.377834 seconds.
Comparacoes: 704982704
Movimentacoes: -1800273297
Memoria (bytes): 400000

Informacoes da Ordenacao por InsertionSort:
Tempo Execucao:  11.892709 seconds.
Comparacoes: -1800173298
Movimentacoes: -1800273297
Memoria (bytes): 400000

Informacoes da Ordenacao por SelectionSort:
Tempo Execucao:  18.458241 seconds.
Comparacoes: 704982704
Movimentacoes: 99988
Memoria (bytes): 400000
```

Figura 6: Questão 2.1 - Ordem misturada

```
Informacoes da Ordenacao por BubbleSort:
Tempo Execucao:  20.242315 seconds.
Comparacoes: 704982704
Movimentacoes: 0
Memoria (bytes): 400000

Informacoes da Ordenacao por InsertionSort:
Tempo Execucao:  0.000584 seconds.
Comparacoes: 99999
Movimentacoes: 0
Memoria (bytes): 400000

Informacoes da Ordenacao por SelectionSort:
Tempo Execucao:  18.408474 seconds.
Comparacoes: 704982704
Movimentacoes: 0
Memoria (bytes): 400000
```

Figura 7: Questão 2.1 - Ordem crescente

```
Informacoes da Ordenacao por BubbleSort:
Tempo Execucao:  21.178679 seconds.
Comparacoes: 704982704
Movimentacoes: 316426
Memoria (bytes): 400000

Informacoes da Ordenacao por InsertionSort:
Tempo Execucao:  0.002068 seconds.
Comparacoes: 416425
Movimentacoes: 316426
Memoria (bytes): 400000

Informacoes da Ordenacao por SelectionSort:
Tempo Execucao:  18.890879 seconds.
Comparacoes: 704982704
Movimentacoes: 12
Memoria (bytes): 400000
```

Figura 8: Questão 2.1 - Quase em ordem crescente

1.4 Reimplentação usando TAD Pessoa

Código

```c
#ifndef ORDENACAO_PT1
#define ORDENACAO_PT1

typedef struct pessoa{
    int idade;
    char* nome;
}Pessoa;


int getComp();
int getMov();
void setComp(int);
void setMov(int);
Pessoa* criaPessoa();
void liberaPessoa(Pessoa*);
Pessoa* copiaVetor(Pessoa*, int);
void imprimeVetor(Pessoa*, int);
void preencheAleatorio(int*, int, int,
    int);
void troca(Pessoa*, Pessoa*);
void BubbleSort(Pessoa*, int);
void InsertionSort(Pessoa*, int);
void SelectionSort(Pessoa*, int);
void InsertionDecres(Pessoa*, int);
void SelectionDecres(Pessoa*, int);


#endif
```

codigos/questao14/questao14.h

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <string.h>
#include "questao14.h"

//Medidas de Complexidade
int comp; //Num. de comparacoes
int mov; //Num. de movimentacoes


int getComp(){
    return comp;
}

int getMov(){
    return mov;
}

void setComp(int valor){
    comp = valor;
}

void setMov(int valor){
    mov = valor;
}

Pessoa* criaPessoa(){
    Pessoa *pessoa = (Pessoa*) malloc
        (sizeof(Pessoa));
    pessoa->idade = 0;
    pessoa->nome = "";
    return pessoa;
}


Pessoa* copiaVetor(Pessoa* v, int n){
    int i;
    Pessoa* v2 = (Pessoa*)malloc(n *
        sizeof(Pessoa));

    if (v2 == NULL){
        fprintf(stderr, "Erro ao alocar
            memoria para o vetor de
            copia.\n");
        exit(1);
    }

    for (i = 0; i < n; i++){
        v2[i].idade = v[i].idade;
        v2[i].nome = strdup(v[i].nome);
        if (v2[i].nome == NULL){
            fprintf(stderr, "Erro ao
                alocar memoria para a
                copia do nome.\n");
            exit(1);
        }
    }
    return v2;
}

void liberaPessoa(Pessoa* pessoa){
    if (pessoa != NULL){
        free(pessoa->nome);
        free(pessoa);
    }
}

void imprimeVetor(Pessoa* v, int n){
    int i, prim = 1;
    printf("[");
    for (i = 0; i < n; i++){
```

```
        if (prim){
            printf("{idade: %d, nome: %s}", v[i].idade, v[i].nome);
            prim = 0;
        } else {
            printf(", {idade: %d, nome: %s}", v[i].idade, v[i].nome);
        }
    }
    printf("]\n");
}

void preencheAleatorio(int* v, int n, int ini, int fim){
    int i;
    for(i=0; i<n; i++)
        v[i] = ini + rand() % (fim-ini + 1);
}

void troca(Pessoa* a, Pessoa* b){
    Pessoa aux = *a;
    *a = *b;
    *b = aux;
    mov++;
}

void BubbleSort(Pessoa* v, int n){
    int i, j;
    for(i = 0; i < n - 1; i++){
        for(j = 0; j < n - i - 1; j++){
            comp++;
            // Compara primeiro os nomes, em caso de empate compara as idades
            if (strcmp(v[j].nome, v[j + 1].nome) > 0 || (strcmp(v[j].nome, v[j + 1].nome) == 0 && v[j].idade > v[j + 1].idade)){
                troca(&v[j], &v[j + 1]);
            }
        }
    }
}

void InsertionSort(Pessoa* v, int n){
    int i, j;
    Pessoa atual;
    for(i = 1; i < n; i++){
        atual = v[i];
        comp++;
        for(j = i; (j > 0) && (strcmp(atual.nome, v[j - 1].nome) < 0 || (strcmp(atual.nome, v[j - 1].nome) == 0 && atual.idade < v[j - 1].idade)); j--){
            v[j] = v[j - 1];
            comp++;
            mov++;
        }
        v[j] = atual;
    }
}

void SelectionSort(Pessoa* v, int n){
    int i, j, menor;
    for(i = 0; i < n - 1; i++){
        menor = i;
        for(j = i + 1; j < n; j++){
            comp++;
            // Compara primeiro os nomes, em caso de empate compara as idades
            if (strcmp(v[j].nome, v[menor].nome) < 0 || (strcmp(v[j].nome, v[menor].nome) == 0 && v[j].idade < v[menor].idade)){
                menor = j;
            }
        }
        if(i != menor){
            troca(&v[i], &v[menor]);
        }
    }
}

void InsertionDecres(Pessoa* v, int n){
    int i, j;
    Pessoa atual;
    for(i = 1; i < n; i++){
        atual = v[i];
        comp++;
        for(j = i; (j > 0) && (strcmp(atual.nome, v[j - 1].nome) > 0 || (strcmp(atual.nome, v[j - 1].nome) == 0 && atual.idade > v[j - 1].idade)); j--){
            v[j] = v[j - 1];
            comp++;
            mov++;
        }
        v[j] = atual;
    }
}

void SelectionDecres(Pessoa* v, int n){
    int i, j, menor;
    for(i = 0; i < n - 1; i++){
        menor = i;
        for(j = i + 1; j < n; j++){
```

```
155             comp++;
156             // Compara primeiro os
                    nomes, em caso de empate
                    compara as idades
157             if (strcmp(v[j].nome,
                    v[menor].nome) > 0 ||
                    (strcmp(v[j].nome,
                    v[menor].nome) == 0 &&
                    v[j].idade >
                    v[menor].idade)){
158                 menor = j;
159             }
160         }
161         if(i != menor){
162             troca(&v[i], &v[menor]);
163         }
164     }
165 }
```

codigos/questao14/questao14.c

```
 1 #include <stdio.h>
 2 #include <stdlib.h>
 3 #include <time.h>
 4 #include <string.h>
 5 #include "questao14.h"
 6
 7 int main(){
 8     srand(time(NULL));
 9     setComp(0);
10     setMov(0);
11     clock_t t;
12
13     Pessoa *v1, *v2, *v3, *v4;
14     Pessoa *p1, *p2, *p3;
15     int n = 3;
16
17     v1 = (Pessoa*) malloc(n *
           sizeof(Pessoa));
18
19     p1 = criaPessoa();
20     p1->idade = 20;
21     p1->nome = strdup("Fulano");
22     v1[0] = *p1;
23
24     p2 = criaPessoa();
25     p2->idade = 50;
26     p2->nome = strdup("Fulano");
27     v1[1] = *p2;
28
29     p3 = criaPessoa();
30     p3->idade = 30;
31     p3->nome = strdup("Beltrano");
32     v1[2] = *p3;
33
34     imprimeVetor(v1, n);
35
36     v2 = copiaVetor(v1, n);
37     v3 = copiaVetor(v1, n);
38     v4 = copiaVetor(v1, n);
39
40     /* InsertionSort */
41     setComp(0);
42     setMov(0);
43     t = clock();
44     InsertionSort(v1, n);
45     t = clock() - t;
46     printf("\nInformacoes da ordenacao
           por InsertionSort crescente:\n");
47     printf("Tempo Execucao:  %f
           seconds.\n", ((float)t) /
           CLOCKS_PER_SEC);
48     printf("Comparacoes: %d\n",
           getComp());
49     printf("Movimentacoes: %d\n",
           getMov());
50     printf("Memoria (bytes): %ld\n", n
           * sizeof(Pessoa));
51
52     imprimeVetor(v1, n);
53
54     /* SelectionSort */
55     setComp(0);
56     setMov(0);
57     t = clock();
58     SelectionSort(v2, n);
59     t = clock() - t;
60     printf("\nInformacoes da Ordenacao
           por SelectionSort crescente:\n");
61     printf("Tempo Execucao:  %f
           seconds.\n", ((float)t) /
           CLOCKS_PER_SEC);
62     printf("Comparacoes: %d\n",
           getComp());
63     printf("Movimentacoes: %d\n",
           getMov());
64     printf("Memoria (bytes): %ld\n", n
           * sizeof(Pessoa));
65
66     imprimeVetor(v2, n);
67
68     /* InsertionSort Descrescente*/
69     setComp(0);
70     setMov(0);
71     t = clock();
72     InsertionDecres(v1, n);
73     t = clock() - t;
74     printf("\nInformacoes da ordenacao
           por InsertionSort
           decrescente:\n");
75     printf("Tempo Execucao:  %f
```

```c
        seconds.\n", ((float)t) /
        CLOCKS_PER_SEC);
    printf("Comparacoes: %d\n",
        getComp());
    printf("Movimentacoes: %d\n",
        getMov());
    printf("Memoria (bytes): %ld\n", n
        * sizeof(Pessoa));

    imprimeVetor(v1, n);

    /* SelectionSort Descrescente*/
    setComp(0);
    setMov(0);
    t = clock();
    SelectionDecres(v2, n);
    t = clock() - t;
    printf("\nInformacoes da Ordenacao
        por SelectionSort
        decrescente:\n");
    printf("Tempo Execucao:  %f
        seconds.\n", ((float)t) /
        CLOCKS_PER_SEC);
    printf("Comparacoes: %d\n",
        getComp());
    printf("Movimentacoes: %d\n",
        getMov());
    printf("Memoria (bytes): %ld\n", n
        * sizeof(Pessoa));

    imprimeVetor(v2, n);

    free(v1);
    free(v2);
    free(v3);
    free(v4);

    // Certifique-se de liberar a
        memoria alocada para cada Pessoa
    liberaPessoa(p1);
    liberaPessoa(p2);
    liberaPessoa(p3);

    return 0;
}
```

codigos/questao14/main.c

```
all: questao14.o
  gcc questao14.o main.c -o main

questao14.o: questao14.h questao14.c
  gcc -c questao14.c

clean:
  rm -f questao14.o main
```

codigos/questao14/Makefile

Saída

```
[{idade: 20, nome: Fulano}, {idade: 50, nome: Fulano}, {idade: 30, nome: Beltrano}]

Informacoes da ordenacao por InsertionSort crescente:
Tempo Execucao:  0.000004 seconds.
Comparacoes: 4
Movimentacoes: 2
Memoria (bytes): 48
[{idade: 30, nome: Beltrano}, {idade: 20, nome: Fulano}, {idade: 50, nome: Fulano}]

Informacoes da Ordenacao por SelectionSort crescente:
Tempo Execucao:  0.000005 seconds.
Comparacoes: 3
Movimentacoes: 2
Memoria (bytes): 48
[{idade: 30, nome: Beltrano}, {idade: 20, nome: Fulano}, {idade: 50, nome: Fulano}]
```

Figura 9: Questão 1.1 - Saída 1

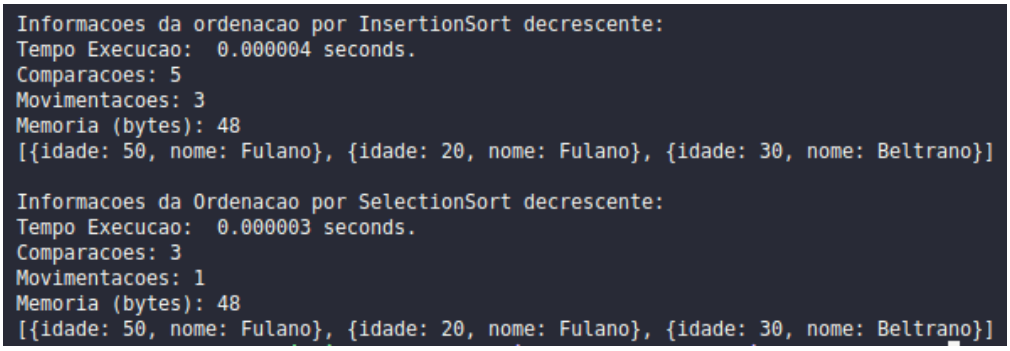

Figura 10: Questão 1.1 - Saída 2