

# UFSJ - Ciências da Computação

## Laboratório de Programação 2

### Roteiro 10

Nome: Geraldo Arthur Detomi

1.1) Faça um programa para ler um valor N e em seguida N inteiros, armazenando esses inteiros em um vetor. Em seguida, ordenar esses valores utilizando os 3 metodos de ordenacao vistos: SelectionSort, InsertionSort e BubbleSort.

roteiro\_10/1-1.c

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  #define QTD_ALGORITMOS_TESTE 3
5
6  void imprimeVetor(int *v, int n) {
7      int i, prim = 1;
8      printf("[");
9      for (i = 0; i < n; i++)
10         if (prim) {
11             printf("%d", v[i]);
12             prim = 0;
13         } else
14             printf(", %d", v[i]);
15     printf("]\n");
16 }
17
18 void troca(int *a, int *b) {
19     int aux = *a;
20     *a = *b;
21     *b = aux;
22 }
23
24 void selectionSort(int *v, int n, int *comp, int *mov) {
25     int i, j, menor;
26     for (i = 0; i < n - 1; i++) {
27         menor = i;
28         for (j = i + 1; j < n; j++) {
29             (*comp)++;
30             if (v[j] < v[menor])
31                 menor = j;
32         }
33         if (i != menor) {
34             troca(&v[i], &v[menor]);
35             (*mov)++;
36         }
37     }
38 }
39
40 void insertionSort(int *v, int n, int *comp, int *mov) {
41     int i, j, atual;
42     for (i = 1; i < n; i++) {
43         atual = v[i];
44         (*comp)++;
45         for (j = i; (j > 0) && (atual < v[j - 1]); j--) {
46             v[j] = v[j - 1];
47             (*comp)++;
48             (*mov)++;
49         }
50         v[j] = atual;
51     }
52 }
53
54 void BubbleSort(int *v, int n, int *comp, int *mov) {
55     int i, j;
56     for (i = 0; i < n - 1; i++)
57         for (j = 0; j < n - i - 1; j++) {
58             (*comp)++;
59             if (v[j] > v[j + 1]) {
60                 troca(&v[j], &v[j + 1]);
61                 (*mov)++;
62             }
63         }
64 }
```

```

65
66 int *copiaVetor(int *v, int n) {
67     int i;
68     int *v2;
69     v2 = (int *)malloc(n * sizeof(int));
70     for (i = 0; i < n; i++)
71         v2[i] = v[i];
72     return v2;
73 }
74
75 void liberar_vetor(int **v) {
76     free(*v);
77     *v = NULL;
78 }
79
80 enum algoritmos_para_teste { SELECON_SORT = 0, INSERTION_SORT, BUBLE_SORT };
81
82 int main() {
83     int n;
84
85     printf("Digite o tamanho do vetor:");
86     scanf("%d", &n);
87
88     int *array_original = (int *)malloc(n * sizeof(int));
89
90     for (int i = 0; i < n; i++) {
91         scanf("%d", &array_original[i]);
92     }
93
94     for (int i = 0; i < QTD_ALGORITMOS_TESTE; i++) {
95         int *array_copia = copiaVetor(array_original, n);
96
97         int comp, mov;
98
99         switch (i) {
100             case SELECON_SORT:
101                 printf("Ordenando com Seleccion Sort:\n");
102                 selectionSort(array_copia, n, &comp, &mov);
103                 imprimeVetor(array_copia, n);
104                 break;
105             case INSERTION_SORT:
106                 printf("Ordenando com Insertion Sort:\n");
107                 insertionSort(array_copia, n, &comp, &mov);
108                 imprimeVetor(array_copia, n);
109                 break;
110             case BUBLE_SORT:
111                 printf("Ordenando com Buble Sort:\n");
112                 BubbleSort(array_copia, n, &comp, &mov);
113                 imprimeVetor(array_copia, n);
114                 break;
115             }
116
117         liberar_vetor(&array_copia);
118     }
119
120     liberar_vetor(&array_original);
121     return 0;
122 }

```

Saída do terminal:

```

└─ arthurdetomi at arthurdetomi-System-Product-Name
└─ ./1-1.out
Digite o tamanho do vetor:4
7
8
2
5
Ordenando com Seleccion Sort:
[2, 5, 7, 8]
Ordenando com Insertion Sort:
[2, 5, 7, 8]
Ordenando com Buble Sort:
[2, 5, 7, 8]

```

**1-2) Modifique os algoritmos de ordenacao anteriores, para que ordenem de forma decrescente os numeros (do maior para o menor) e teste os novos metodos.**

roteiro\_10/1-2.c

```

1 #include <stdio.h>
2 #include <stdlib.h>
3

```

```
4 #define QTD_ALGORITMOS_TESTE 3
5
6 void imprimeVetor(int *v, int n) {
7     int i, prim = 1;
8     printf("[");
9     for (i = 0; i < n; i++)
10         if (prim) {
11             printf("%d", v[i]);
12             prim = 0;
13         } else
14             printf(", %d", v[i]);
15     printf("]\n");
16 }
17
18 void troca(int *a, int *b) {
19     int aux = *a;
20     *a = *b;
21     *b = aux;
22 }
23
24 void selectionSort(int *v, int n, int *comp, int *mov) {
25     int i, j, maior;
26     for (i = 0; i < n - 1; i++) {
27         maior = i;
28         for (j = i + 1; j < n; j++) {
29             (*comp)++;
30             if (v[j] > v[maior])
31                 maior = j;
32         }
33         if (i != maior) {
34             troca(&v[i], &v[maior]);
35             (*mov)++;
36         }
37     }
38 }
39
40 void insertionSort(int *v, int n, int *comp, int *mov) {
41     int i, j, atual;
42     for (i = 1; i < n; i++) {
43         atual = v[i];
44         (*comp)++;
45         for (j = i; (j > 0) && (atual > v[j - 1]); j--) {
46             v[j] = v[j - 1];
47             (*comp)++;
48             (*mov)++;
49         }
50         v[j] = atual;
51     }
52 }
53
54 void BubbleSort(int *v, int n, int *comp, int *mov) {
55     int i, j;
56     for (i = 0; i < n - 1; i++)
57         for (j = 0; j < n - i - 1; j++) {
58             (*comp)++;
59             if (v[j] < v[j + 1]) {
60                 troca(&v[j], &v[j + 1]);
61                 (*mov)++;
62             }
63         }
64 }
65
66 int *copiaVetor(int *v, int n) {
67     int i;
68     int *v2;
69     v2 = (int *)malloc(n * sizeof(int));
70     for (i = 0; i < n; i++)
71         v2[i] = v[i];
72     return v2;
73 }
74
75 void liberar_vetor(int **v) {
76     free(*v);
77     *v = NULL;
78 }
79
80 enum algoritmos_para_teste { SELECON_SORT = 0, INSERTION_SORT, BUBLE_SORT };
81
82 int main() {
83     int n;
84 }
```

```

85     printf("Digite o tamanho do vetor:");
86     scanf("%d", &n);
87
88     int *array_original = (int *)malloc(n * sizeof(int));
89
90     for (int i = 0; i < n; i++) {
91         scanf("%d", &array_original[i]);
92     }
93
94     printf("Agora a ordenação será decrescente:\n");
95     for (int i = 0; i < QTD_ALGORITMOS_TESTE; i++) {
96         int *array_copia = copiaVetor(array_original, n);
97
98         int comp, mov;
99
100        switch (i) {
101            case SELECON_SORT:
102                printf("Ordenando com Seleccion Sort:\n");
103                selectionSort(array_copia, n, &comp, &mov);
104                imprimeVetor(array_copia, n);
105                break;
106            case INSERTION_SORT:
107                printf("Ordenando com Insertion Sort:\n");
108                insertionSort(array_copia, n, &comp, &mov);
109                imprimeVetor(array_copia, n);
110                break;
111            case BUBBLE_SORT:
112                printf("Ordenando com Buble Sort:\n");
113                BubbleSort(array_copia, n, &comp, &mov);
114                imprimeVetor(array_copia, n);
115                break;
116        }
117
118        liberar_vetor(&array_copia);
119    }
120
121    liberar_vetor(&array_original);
122    return 0;
123 }

```

Saída do terminal:

```

[arthurdetomi at arthurdetomi-System-Product-Name
:: ./1-2.out
Digite o tamanho do vetor:5
4
10
3
6
2
Agora a ordenação será decrescente:
Ordenando com Seleccion Sort:
[10, 6, 4, 3, 2]
Ordenando com Insertion Sort:
[10, 6, 4, 3, 2]
Ordenando com Buble Sort:
[10, 6, 4, 3, 2]

```

1-3) Considerando os 3 metodos vistos, utilize um programa para ordenar grande quantidade de valores e verifique o aumento do numero de comparacoes, movimentacoes e tempo de execucao conforme se aumenta o volume de dados processados

roteiro\_10/tempo.h

```

1  #ifndef TEMPO_H
2  #define TEMPO_H
3
4  #include <stdio.h>
5  #include <sys/resource.h>
6  #include <sys/time.h>
7
8  // Estrutura para armazenar os tempos de execução
9  typedef struct Temporizador {
10     struct timeval start_tv, end_tv;
11     struct rusage start_usage, end_usage;
12 } Temporizador;
13
14 // Inicia a contagem do temporizador
15 void iniciarTemporizador(Temporizador *t);
16
17 // Finaliza a contagem do temporizador
18 void finalizarTemporizador(Temporizador *t);
19
20 // Calcula o tempo real decorrido em segundos

```

```

21 double calcularTempoReal(Temporizador *t);
22
23 // Calcula o tempo de CPU em modo usuário em segundos
24 double calcularTempoUsuario(Temporizador *t);
25
26 // Calcula o tempo de CPU em modo sistema/kernel em segundos
27 double calcularTempoSistema(Temporizador *t);
28
29 // Imprime todos os tempos medidos de forma formatada
30 void imprimirTempos(Temporizador *t);
31
32 #endif

```

## roteiro\_10/tempo.c

```

1  #include "tempo.h"
2
3  // Inicia a medição do tempo, armazenando os valores atuais
4  void iniciarTemporizador(Temporizador *t) {
5      gettimeofday(&t->start_tv, NULL);
6      getrusage(RUSAGE_SELF, &t->start_usage);
7  }
8
9  // Finaliza a medição do tempo, armazenando os valores finais
10 void finalizarTemporizador(Temporizador *t) {
11     gettimeofday(&t->end_tv, NULL);
12     getrusage(RUSAGE_SELF, &t->end_usage);
13 }
14
15 double calcularTempo(struct timeval inicio, struct timeval fim) {
16     time_t seg = fim.tv_sec - inicio.tv_sec;
17     suseconds_t microseg = fim.tv_usec - inicio.tv_usec;
18
19     // Ajusta caso microsegundos do fim sejam menores que os do início
20     if (microseg < 0) {
21         seg -= 1;
22         microseg += 1000000;
23     }
24
25     return (double)seg + (double)microseg / 1e6;
26 }
27
28 // Calcula diferença entre tempos reais (start e end) em segundos
29 double calcularTempoReal(Temporizador *t) {
30     return calcularTempo(t->start_tv, t->end_tv);
31 }
32
33 // Calcula diferença entre tempos de sistema (start e end) em segundos
34 double calcularTempoSistema(Temporizador *t) {
35     return calcularTempo(t->start_usage.ru_stime, t->end_usage.ru_stime);
36 }
37
38 // Imprime todos os tempos de execução de forma legível
39 void imprimirTempos(Temporizador *t) {
40     printf("Tempo de execução:\n");
41     printf("Tempo Real: %.8f segundos\n", calcularTempoReal(t));
42     printf("Tempo Sistema: %.8f segundos\n", calcularTempoSistema(t));
43 }

```

## roteiro\_10/1-3.c

```

1  #include "tempo.h"
2  #include <stdio.h>
3  #include <stdlib.h>
4
5  #define QTD_ALGORITMOS_TESTE 3
6
7  void imprimeVetor(int *v, int n) {
8      int i, prim = 1;
9      printf("[");
10     for (i = 0; i < n; i++)
11         if (prim) {
12             printf("%d", v[i]);
13             prim = 0;
14         } else
15             printf(", %d", v[i]);
16     printf("]\n");
17 }
18
19 void troca(int *a, int *b) {
20     int aux = *a;
21     *a = *b;

```

```
22     *b = aux;
23 }
24
25 void selectionSort(int *v, int n, int *comp, int *mov) {
26     int i, j, menor;
27     for (i = 0; i < n - 1; i++) {
28         menor = i;
29         for (j = i + 1; j < n; j++) {
30             (*comp)++;
31             if (v[j] < v[menor])
32                 menor = j;
33         }
34         if (i != menor) {
35             troca(&v[i], &v[menor]);
36             (*mov)++;
37         }
38     }
39 }
40
41 void insertionSort(int *v, int n, int *comp, int *mov) {
42     int i, j, atual;
43     for (i = 1; i < n; i++) {
44         atual = v[i];
45         (*comp)++;
46         for (j = i; (j > 0) && (atual < v[j - 1]); j--) {
47             v[j] = v[j - 1];
48             (*comp)++;
49             (*mov)++;
50         }
51         v[j] = atual;
52     }
53 }
54
55 void BubbleSort(int *v, int n, int *comp, int *mov) {
56     int i, j;
57     for (i = 0; i < n - 1; i++)
58         for (j = 0; j < n - i - 1; j++) {
59             (*comp)++;
60             if (v[j] > v[j + 1]) {
61                 troca(&v[j], &v[j + 1]);
62                 (*mov)++;
63             }
64         }
65 }
66
67 int *copiaVetor(int *v, int n) {
68     int i;
69     int *v2;
70     v2 = (int *)malloc(n * sizeof(int));
71     for (i = 0; i < n; i++)
72         v2[i] = v[i];
73     return v2;
74 }
75
76 void liberar_vetor(int **v) {
77     free(*v);
78     *v = NULL;
79 }
80
81 enum algoritmos_para_teste { SELECON_SORT = 0, INSERTION_SORT, BUBLE_SORT };
82
83 int main() {
84     int n;
85
86     printf("Digite o tamanho do vetor:");
87     scanf("%d", &n);
88
89     int *array_original = (int *)malloc(n * sizeof(int));
90
91     for (int i = 0; i < n; i++) {
92         scanf("%d", &array_original[i]);
93     }
94
95     for (int i = 0; i < QTD_ALGORITMOS_TESTE; i++) {
96         int *array_copia = copiaVetor(array_original, n);
97
98         int comp = 0, mov = 0;
99
100         Temporizador tempo_teste;
101         iniciarTemporizador(&tempo_teste);
102     }
```

```

103     switch (i) {
104     case SELECON_SORT:
105         printf("Ordenando com Seleccion Sort:\n");
106         selectionSort(array_copia, n, &comp, &mov);
107         imprimeVetor(array_copia, n);
108         break;
109     case INSERTION_SORT:
110         printf("Ordenando com Insertion Sort:\n");
111         insertionSort(array_copia, n, &comp, &mov);
112         imprimeVetor(array_copia, n);
113         break;
114     case BUBBLE_SORT:
115         printf("Ordenando com Buble Sort:\n");
116         BubbleSort(array_copia, n, &comp, &mov);
117         imprimeVetor(array_copia, n);
118         break;
119     }
120
121     printf("Quantidade de movimentações realizadas: %d\n", mov);
122     printf("Quantidade de comparações realizadas: %d\n", comp);
123
124     finalizarTemporizador(&tempo_teste);
125
126     imprimirTempos(&tempo_teste);
127
128     liberar_vetor(&array_copia);
129
130     printf("\n\n");
131 }
132
133 liberar_vetor(&array_original);
134 return 0;
135 }

```

Saída do terminal:

```

[arthurdetomi at arthurdetomi-System-Product-Name in ~/Documents/UFSJ-Graduacao/UFSJ-2025_1/Lab_Prog_2/roteiro_10 on mainxxx 25-06-15]
$ ./1-3.out < input/100-misturado.txt
Digite o tamanho do vetor: Ordenando com Seleccion Sort:
[2, 3, 4, 4, 9, 10, 12, 12, 14, 17, 18, 21, 25, 25, 27, 28, 29, 29, 33, 34, 38, 38, 39, 40, 42, 43, 43, 51, 52, 58, 62, 66, 67, 67, 68, 88, 89, 94, 95, 96, 97, 100, 105, 106, 113, 113, 115, 115, 116, 117, 121, 123, 124, 127, 129, 129, 129, 134, 135, 139, 142, 142, 144, 163, 164, 165, 167, 167, 167, 172, 173, 175, 178, 178, 180, 184, 185, 189, 192, 193, 193, 195, 197, 200, 200]
Quantidade de movimentações realizadas: 97
Quantidade de comparações realizadas: 4950
Tempo de execução:
Tempo Real: 0.00004000 segundos
Tempo Sistema: 0.00004000 segundos

Ordenando com Insertion Sort:
[2, 3, 4, 4, 9, 10, 12, 12, 14, 17, 18, 21, 25, 25, 27, 28, 29, 29, 33, 34, 38, 38, 39, 40, 42, 43, 43, 51, 52, 58, 62, 66, 67, 67, 68, 88, 89, 94, 95, 96, 97, 100, 105, 106, 113, 113, 115, 115, 116, 117, 121, 123, 124, 127, 129, 129, 129, 134, 135, 139, 142, 142, 144, 163, 164, 165, 167, 167, 167, 172, 173, 175, 178, 178, 180, 184, 185, 189, 192, 193, 193, 195, 197, 200, 200]
Quantidade de movimentações realizadas: 2426
Quantidade de comparações realizadas: 2525
Tempo de execução:
Tempo Real: 0.00002900 segundos
Tempo Sistema: 0.00002900 segundos

Ordenando com Buble Sort:
[2, 3, 4, 4, 9, 10, 12, 12, 14, 17, 18, 21, 25, 25, 27, 28, 29, 29, 33, 34, 38, 38, 39, 40, 42, 43, 43, 51, 52, 58, 62, 66, 67, 67, 68, 88, 89, 94, 95, 96, 97, 100, 105, 106, 113, 113, 115, 115, 116, 117, 121, 123, 124, 127, 129, 129, 129, 134, 135, 139, 142, 142, 144, 163, 164, 165, 167, 167, 167, 172, 173, 175, 178, 178, 180, 184, 185, 189, 192, 193, 193, 195, 197, 200, 200]
Quantidade de movimentações realizadas: 2426
Quantidade de comparações realizadas: 4950
Tempo de execução:
Tempo Real: 0.00004400 segundos
Tempo Sistema: 0.00004400 segundos

```

**1-4) Modifique as funcoes que implementam o metodo Seleccion e Insercao para fazer a ordenacao de um vetor de struct do tipo Pessoa,**

roteiro\_10/1-4.c

```

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4
5 #define QTD_ALGORITMOS_TESTE 2
6
7 enum algoritmos_para_teste { SELECON_SORT = 0, INSERTION_SORT, BUBBLE_SORT };
8
9 enum ordens_ordenacao { CRESCENTE = 'c', DECRESCENTE = 'd' };

```

```
10
11 typedef struct Pessoa {
12     char nome[50];
13     int idade;
14 } Pessoa;
15
16 void imprimeVetor(Pessoa *v, int n) {
17     for (int i = 0; i < n; i++) {
18         printf("%s (%d)\n", v[i].nome, v[i].idade);
19     }
20 }
21
22 void troca(Pessoa *a, Pessoa *b) {
23     Pessoa aux = *a;
24     *a = *b;
25     *b = aux;
26 }
27
28 int comparaCrescente(Pessoa p1, Pessoa p2) {
29     int comparacao = strcmp(p1.nome, p2.nome);
30     if (comparacao == 0) {
31         return p1.idade - p2.idade;
32     }
33     return comparacao;
34 }
35
36 int comparaDecrescente(Pessoa p1, Pessoa p2) {
37     int comparacao = strcmp(p2.nome, p1.nome);
38     if (comparacao == 0) {
39         return p2.idade - p1.idade;
40     }
41     return comparacao;
42 }
43
44 int comparaPorOrdem(Pessoa p1, Pessoa p2, char ordem) {
45     if (ordem == CRESCENTE) {
46         return comparaCrescente(p1, p2);
47     }
48     return comparaDecrescente(p1, p2);
49 }
50
51 void selectionSort(Pessoa *v, int n, int *comp, int *mov, char ordem) {
52     int i, j, menor;
53     for (i = 0; i < n - 1; i++) {
54         menor = i;
55         for (j = i + 1; j < n; j++) {
56             (*comp)++;
57             if (comparaPorOrdem(v[j], v[menor], ordem) < 0) {
58                 menor = j;
59             }
60         }
61         if (i != menor) {
62             troca(&v[i], &v[menor]);
63             (*mov)++;
64         }
65     }
66 }
67
68 void insertionSort(Pessoa *v, int n, int *comp, int *mov, char ordem) {
69     int i, j;
70     for (i = 1; i < n; i++) {
71         Pessoa atual = v[i];
72         (*comp)++;
73         for (j = i; (j > 0) && comparaPorOrdem(atual, v[j - 1], ordem) < 0; j--) {
74             v[j] = v[j - 1];
75             (*comp)++;
76             (*mov)++;
77         }
78         v[j] = atual;
79     }
80 }
81
82 Pessoa *copiaVetor(Pessoa *v, int n) {
83     int i;
84     Pessoa *v2;
85     v2 = (Pessoa *)malloc(n * sizeof(Pessoa));
86     for (i = 0; i < n; i++) {
87         strcpy(v2[i].nome, v[i].nome);
88         v2[i].idade = v[i].idade;
89     }
90     return v2;
91 }
```



```
91 | }
92 |
93 | void liberar_vetor(Pessoa **v) {
94 |     free(*v);
95 |     *v = NULL;
96 | }
97 |
98 | int main() {
99 |     int n;
100 |
101 |     printf("Digite o tamanho do vetor:");
102 |     scanf("%d", &n);
103 |
104 |     char ordem;
105 |     printf("Digite a ordem de ordenação (c) crescente (d) decrescente:");
106 |     scanf(" %c", &ordem);
107 |
108 |     if (ordem != CRESCENTE && ordem != DECRESCENTE) {
109 |         printf("Ordem inserida invalida\n");
110 |         exit(1);
111 |     }
112 |
113 |     Pessoa *array_original = (Pessoa *)malloc(n * sizeof(Pessoa));
114 |
115 |     for (int i = 0; i < n; i++) {
116 |         printf("Cadastro pessoa %d:\n", i + 1);
117 |         printf("Nome: ");
118 |         scanf("%s", array_original[i].nome);
119 |         printf("Idade: ");
120 |         scanf("%d", &array_original[i].idade);
121 |     }
122 |
123 |     for (int i = 0; i < QTD_ALGORITMOS_TESTE; i++) {
124 |         Pessoa *array_copia = copiaVetor(array_original, n);
125 |
126 |         int comp = 0, mov = 0;
127 |
128 |         switch (i) {
129 |             case SELECIION_SORT:
130 |                 printf("Ordenando com Seleccion Sort:\n");
131 |                 selectionSort(array_copia, n, &comp, &mov, ordem);
132 |                 imprimeVetor(array_copia, n);
133 |                 break;
134 |             case INSERTION_SORT:
135 |                 printf("Ordenando com Insertion Sort:\n");
136 |                 insertionSort(array_copia, n, &comp, &mov, ordem);
137 |                 imprimeVetor(array_copia, n);
138 |                 break;
139 |         }
140 |
141 |         liberar_vetor(&array_copia);
142 |     }
143 |
144 |     liberar_vetor(&array_original);
145 |     return 0;
146 | }
```

Saída do terminal:

```
└─ arthurdetomi at arthurdetomi-System-Product-Name in ~/Documents/UFSJ-Graduacao/UFSJ-2025_1/Lab_Prog_2
└─ ./1-4.out < in
Digite o tamanho do vetor:Digite a ordem de ordenação (c) crescente (d) decrescente:Cadastro pessoa 1:
Nome: Idade: Cadastro pessoa 2:
Nome: Idade: Cadastro pessoa 3:
Nome: Idade: Ordenando com Selecion Sort:
Bob (25)
Alice (30)
Alice (25)
Ordenando com Insertion Sort:
Bob (25)
Alice (30)
Alice (25)
└─ arthurdetomi at arthurdetomi-System-Product-Name in ~/Documents/UFSJ-Graduacao/UFSJ-2025_1/Lab_Prog_2
└─ ./ echo 'Teste anterior foi decrescente'
Teste anterior foi decrescente
└─ arthurdetomi at arthurdetomi-System-Product-Name in ~/Documents/UFSJ-Graduacao/UFSJ-2025_1/Lab_Prog_2
└─ ./ echo 'Agora será crescente'
Agora será crescente
└─ arthurdetomi at arthurdetomi-System-Product-Name in ~/Documents/UFSJ-Graduacao/UFSJ-2025_1/Lab_Prog_2
└─ ./ ./1-4.out < in
Digite o tamanho do vetor:Digite a ordem de ordenação (c) crescente (d) decrescente:Cadastro pessoa 1:
Nome: Idade: Cadastro pessoa 2:
Nome: Idade: Cadastro pessoa 3:
Nome: Idade: Ordenando com Selecion Sort:
Alice (25)
Alice (30)
Bob (25)
Ordenando com Insertion Sort:
Alice (25)
Alice (30)
Bob (25)
└─ arthurdetomi at arthurdetomi-System-Product-Name in ~/Documents/UFSJ-Graduacao/UFSJ-2025_1/Lab_Prog_2
```