1.1/main.c

```c
#include <stdio.h>
#include "ordenacao.h"
int main(){

    int n;
    printf("Quantidade de elementos a serem ordenados: ");
    scanf("%d", &n);


    int vetor[n];
    printf("Digite os %d inteiros a serem ordenados:\n", n);
    for (int i = 0; i < n; i++) {
        scanf("%d", &vetor[i]);
    }

    int bubble[n], selection[n], insertion[n];
    for (int i = 0; i < n; i++) {
        bubble[i] = vetor[i];
        selection[i] = vetor[i];
        insertion[i] = vetor[i];
    }

    BubbleSort(bubble, n);
    SelectionSort(selection, n);
    InsertionSort(insertion, n);

    printf("\nValores ordenados pelo Bubble Sort: ");
    for (int i = 0; i < n; i++) {
        printf("%d ", bubble[i]);
    }

    printf("\nValores ordenados pelo Selection Sort: ");
    for (int i = 0; i < n; i++) {
        printf("%d ", selection[i]);
    }

    printf("\nValores ordenados pelo Insertion Sort: ");
    for (int i = 0; i < n; i++) {
        printf("%d ", insertion[i]);
    }
    printf("\n");


    return 0;
}

```

1.1/ordenacao.c

```c
#include <stdio.h>
#include"ordenacao.h"
int comp = 0;
int mov = 0;




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


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];
           mov++;
           comp++;
       }
       v[j] = atual;
   }
```

```
58 }
59
```

1.1/ordenacao.h

```h
#ifndef ORDENACAO_H
#define ORDENACAO_H


void BubbleSort(int *v, int n);
void SelectionSort(int *v, int n);
void InsertionSort(int *v, int n);
void troca(int *a, int *b);

#endif

```

```
[lucascosta@fedora 1.1]$ ./main
Quantidade de elementos a serem ordenados: 6
Digite os 6 inteiros a serem ordenados:
7
5
8
4
2
1

Valores ordenados pelo Bubble Sort: 1 2 4 5 7 8
Valores ordenados pelo Selection Sort: 1 2 4 5 7 8
Valores ordenados pelo Insertion Sort: 1 2 4 5 7 8
[lucascosta@fedora 1.1]$
```

1.2/main.c

```c
#include <stdio.h>
#include "decrescente.h"

int main(){
   int n;
   printf("Quantidade de elementos a serem ordenados: ");
   scanf("%d", &n);


   int vetor[n];
   printf("Digite os %d inteiros a serem ordenados:\n", n);
   for (int i = 0; i < n; i++) {
       scanf("%d", &vetor[i]);
   }


   int bubble[n], selection[n], insertion[n];
   for (int i = 0; i < n; i++) {
       bubble[i] = vetor[i];
       selection[i] = vetor[i];
       insertion[i] = vetor[i];
   }


   BubbleSort(bubble, n);
   SelectionSort(selection, n);
   InsertionSort(insertion, n);



   printf("\nValores ordenados pelo Bubble Sort: ");
   for (int i = 0; i < n; i++) {
       printf("%d ", bubble[i]);
   }
   printf("\nValores ordenados pelo Selection Sort: ");
   for (int i = 0; i < n; i++) {
       printf("%d ", selection[i]);
   }
   printf("\nValores ordenados pelo Insertion Sort: ");
   for (int i = 0; i < n; i++) {
       printf("%d ", insertion[i]);
   }
   printf("\n");

   return 0;
}

```

1.2/decrescente.h

```h
#ifndef DECRESCENTE_H
#define DECRESCENTE_H


void BubbleSort(int *v, int n);
void SelectionSort(int *v, int n);
void InsertionSort(int *v, int n);


#endif
```

1.2/decrescente.c

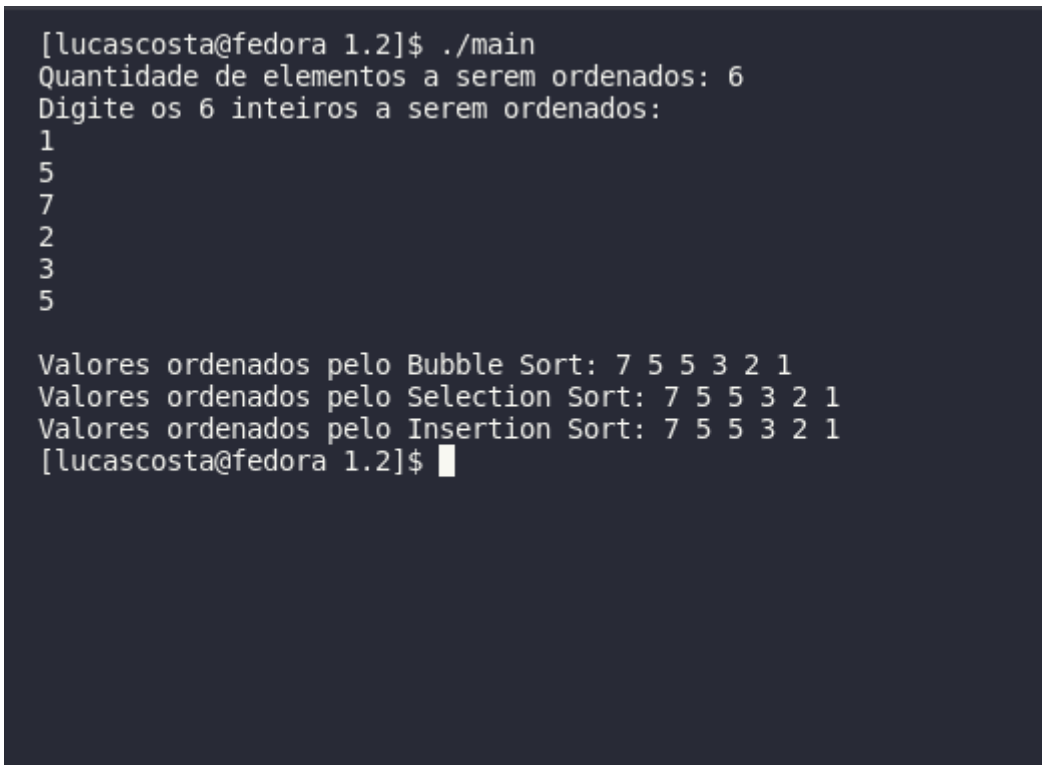
```c
#include <stdio.h>
#include "ordenacao.h"
int comp = 0;
int mov = 0;


void troca(int *a, int *b) {
   int temp = *a;
   *a = *b;
   *b = temp;
}
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+1], &v[j]);
               mov++;
           }
       }
   }
}


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++;
       }
   }
}


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];
           mov++;
           comp++;
       }
       v[j] = atual;
   }
}

```


[lucascosta@fedora 1.2]$ ./main
Quantidade de elementos a serem ordenados: 6
Digite os 6 inteiros a serem ordenados:
1
5
7
2
3
5
Valores ordenados pelo Bubble Sort: 7 5 5 3 2 1
Valores ordenados pelo Selection Sort: 7 5 5 3 2 1
Valores ordenados pelo Insertion Sort: 7 5 5 3 2 1
[lucascosta@fedora 1.2]$

1.3/main.c

```c

#include <stdio.h>
#include <stdlib.h>
#include "ordena.h"

int main(int argc, char *argv[]) {
    if (argc != 2) {
        printf("Uso: %s <nome_do_arquivo>\n", argv[0]);
        return 1;
    }

    FILE *arquivo = fopen(argv[1], "r");
    if (arquivo == NULL) {
        printf("Erro ao abrir o arquivo.\n");
        return 1;
    }

    int tamanho;
    fscanf(arquivo, "%d", &tamanho);

    int *valores = (int *)malloc(tamanho * sizeof(int));

    for (int i = 0; i < tamanho; i++) {
        fscanf(arquivo, "%d", &valores[i]);
    }

    fclose(arquivo);

    medirDesempenho(valores, tamanho, "Selecao");
    medirDesempenho(valores, tamanho, "Insercao");
    medirDesempenho(valores, tamanho, "Bubble");

    free(valores);

    return 0;
}
```

1.3/ordena.h

```c
#ifndef ORDENA_H
#define ORDENA_H

#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <string.h>

void selectionSort(int arr[], int n, long *comparacoes, long *movimentacoes) {
    int i, j, minIndex, temp;

    for (i = 0; i < n - 1; i++) {
        minIndex = i;
        for (j = i + 1; j < n; j++) {
            if (arr[j] < arr[minIndex]) {
                (*comparacoes)++;
                minIndex = j;
            }
        }
        if (minIndex != i) {
            (*comparacoes)++;
            temp = arr[i];
            arr[i] = arr[minIndex];
            arr[minIndex] = temp;
            (*movimentacoes) += 3;
        }
    }
}

void insertionSort(int arr[], int n, long *comparacoes, long *movimentacoes) {
    int i, chave, j;

    for (i = 1; i < n; i++) {
        chave = arr[i];
        j = i - 1;

        while (j >= 0 && arr[j] > chave) {
            (*comparacoes)++;
            arr[j + 1] = arr[j];
            (*movimentacoes)++;
            j = j - 1;
        }
        arr[j + 1] = chave;
        (*movimentacoes)++;
    }
}

void BubbleSort(int arr[], int n, long *comparacoes, long *movimentacoes) {
    int i, j;
    for (i = 0; i < n - 1; i++) {
        for (j = 0; j < n - i - 1; j++) {
            (*comparacoes)++;
            if (arr[j] > arr[j + 1]) {
                (*comparacoes)++;
                int temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
```

```
                (*movimentacoes) += 3;
            }
        }
    }
}


void medirDesempenho(int *valores, int tamanho, char *tipoOrdenacao) {
    int *copiaValores = (int *)malloc(tamanho * sizeof(int));
    for (int i = 0; i < tamanho; ++i) {
        copiaValores[i] = valores[i];
    }

    clock_t inicio, fim;
    double tempoExecucao;
    long comparacoes = 0, movimentacoes = 0;

    inicio = clock();
    if (strcmp(tipoOrdenacao, "Selecao") == 0) {
        selectionSort(copiaValores, tamanho, &comparacoes, &movimentacoes);
    } else if (strcmp(tipoOrdenacao, "Insercao") == 0) {
        insertionSort(copiaValores, tamanho, &comparacoes, &movimentacoes);
    } else if (strcmp(tipoOrdenacao, "Bubble") == 0) {
        BubbleSort(copiaValores, tamanho, &comparacoes, &movimentacoes);
    } else {
        printf("Algoritmo de ordenação não está disponível\n");
        free(copiaValores);
        return;
    }
    fim = clock();

    tempoExecucao = ((double)(fim - inicio)) / CLOCKS_PER_SEC;

    printf("Tipo de ordenacao: %s\n", tipoOrdenacao);
    printf("Tempo de execucao: %f segundos\n", tempoExecucao);
    printf("Numero de comparações: %ld\n", comparacoes);
    printf("Numero de movimentações: %ld\n\n", movimentacoes);

    free(copiaValores);
}

#endif

```

```
[lucascosta@fedora 1.3]$ ./main 100-contrario.txt
Tipo de ordenacao: Selecao
Tempo de execucao: 0.000056 segundos
Numero de comparações: 2550
Numero de movimentações: 150

Tipo de ordenacao: Insercao
Tempo de execucao: 0.000060 segundos
Numero de comparações: 4950
Numero de movimentações: 5049

Tipo de ordenacao: Bubble
Tempo de execucao: 0.000098 segundos
Numero de comparações: 9900
Numero de movimentações: 14850

[lucascosta@fedora 1.3]$ ./main 1000-contrario.txt
Tipo de ordenacao: Selecao
Tempo de execucao: 0.003631 segundos
Numero de comparações: 250500
Numero de movimentações: 1500

Tipo de ordenacao: Insercao
Tempo de execucao: 0.001627 segundos
Numero de comparações: 499500
Numero de movimentações: 500499

Tipo de ordenacao: Bubble
Tempo de execucao: 0.002844 segundos
Numero de comparações: 999000
Numero de movimentações: 1498500

[lucascosta@fedora 1.3]$ ./main 10000-contrario.txt
Tipo de ordenacao: Selecao
Tempo de execucao: 0.120296 segundos
Numero de comparações: 25005000
Numero de movimentações: 15000

Tipo de ordenacao: Insercao
Tempo de execucao: 0.138432 segundos
Numero de comparações: 49995000
Numero de movimentações: 50004999

^[[A^[[ATipo de ordenacao: Bubble
Tempo de execucao: 0.236691 segundos
Numero de comparações: 99990000
Numero de movimentações: 149985000

[lucascosta@fedora 1.3]$ ./main 100000-contrario.txt
Tipo de ordenacao: Selecao
Tempo de execucao: 9.800965 segundos
Numero de comparações: 2500050000
Numero de movimentações: 150000
```

.

```
Tipo de ordenacao: Insercao
Tempo de execucao: 11.815961 segundos
Numero de comparações: 4999950000
Numero de movimentações: 5000049999

Tipo de ordenacao: Bubble
Tempo de execucao: 21.415068 segundos
Numero de comparações: 9999900000
Numero de movimentações: 14999850000
```

```
[lucascosta@fedora 1.3]$ ./main 100-ordenado.txt
Tipo de ordenacao: Selecao
Tempo de execucao: 0.000019 segundos
Numero de comparações: 0
Numero de movimentações: 0

Tipo de ordenacao: Insercao
Tempo de execucao: 0.000001 segundos
Numero de comparações: 0
Numero de movimentações: 99

Tipo de ordenacao: Bubble
Tempo de execucao: 0.000017 segundos
Numero de comparações: 4950
Numero de movimentações: 0

[lucascosta@fedora 1.3]$ ./main 1000-ordenado.txt
Tipo de ordenacao: Selecao
Tempo de execucao: 0.003169 segundos
Numero de comparações: 0
Numero de movimentações: 0

Tipo de ordenacao: Insercao
Tempo de execucao: 0.000004 segundos
Numero de comparações: 0
Numero de movimentações: 999

Tipo de ordenacao: Bubble
Tempo de execucao: 0.001533 segundos
Numero de comparações: 499500
Numero de movimentações: 0

[lucascosta@fedora 1.3]$ ./main 10000-ordenado.txt
Tipo de ordenacao: Selecao
Tempo de execucao: 0.108863 segundos
Numero de comparações: 0
Numero de movimentações: 0

Tipo de ordenacao: Insercao
Tempo de execucao: 0.000035 segundos
Numero de comparações: 0
Numero de movimentações: 9999

Tipo de ordenacao: Bubble
Tempo de execucao: 0.114274 segundos
Numero de comparações: 49995000
Numero de movimentações: 0

[lucascosta@fedora 1.3]$ ./main 100000-ordenado.txt
Tipo de ordenacao: Selecao
Tempo de execucao: 9.322151 segundos
Numero de comparações: 0
Numero de movimentações: 0
```

```
Tipo de ordenacao: Insercao
Tempo de execucao: 0.000294 segundos
Numero de comparações: 0
Numero de movimentações: 99999

Tipo de ordenacao: Bubble
Tempo de execucao: 9.573903 segundos
Numero de comparações: 4999950000
Numero de movimentações: 0
```

```
[lucascosta@fedora 1.3]$ gcc main.c -o main
[lucascosta@fedora 1.3]$ ./main 100-misturado.txt
Tipo de ordenacao: Selecao
Tempo de execucao: 0.000065 segundos
Numero de comparações: 417
Numero de movimentações: 291

Tipo de ordenacao: Insercao
Tempo de execucao: 0.000038 segundos
Numero de comparações: 2426
Numero de movimentações: 2525

Tipo de ordenacao: Bubble
Tempo de execucao: 0.000118 segundos
Numero de comparações: 7376
Numero de movimentações: 7278

[lucascosta@fedora 1.3]$ ./main 1000-misturado.txt
Tipo de ordenacao: Selecao
Tempo de execucao: 0.001912 segundos
Numero de comparações: 6294
Numero de movimentações: 2982

Tipo de ordenacao: Insercao
Tempo de execucao: 0.000871 segundos
Numero de comparações: 232674
Numero de movimentações: 233673

Tipo de ordenacao: Bubble
Tempo de execucao: 0.002567 segundos
Numero de comparações: 732174
Numero de movimentações: 698022

[lucascosta@fedora 1.3]$ ./main 10000-misturado.txt
Tipo de ordenacao: Selecao
Tempo de execucao: 0.138264 segundos
Numero de comparações: 86777
Numero de movimentações: 29976

Tipo de ordenacao: Insercao
Tempo de execucao: 0.068136 segundos
Numero de comparações: 24779170
Numero de movimentações: 24789169

^[[ATipo de ordenacao: Bubble
Tempo de execucao: 0.253850 segundos
Numero de comparações: 74774170
Numero de movimentações: 74337510

[lucascosta@fedora 1.3]$ ./main 100000-misturado.txt
Tipo de ordenacao: Selecao
Tempo de execucao: 9.317687 segundos
Numero de comparações: 1044622
Numero de movimentações: 299964
```

```
Tipo de ordenacao: Insercao
Tempo de execucao: 5.810295 segundos
Numero de comparações: 2494693999
Numero de movimentações: 2494793998

Tipo de ordenacao: Bubble
Tempo de execucao: 25.389793 segundos
Numero de comparações: 7494643999
Numero de movimentações: 7484081997
```

```
[lucascosta@fedora 1.3]$ ./main 100-quaseordenado.txt
Tipo de ordenacao: Selecao
Tempo de execucao: 0.000058 segundos
Numero de comparações: 15
Numero de movimentações: 15

Tipo de ordenacao: Insercao
Tempo de execucao: 0.000010 segundos
Numero de comparações: 361
Numero de movimentações: 460

Tipo de ordenacao: Bubble
Tempo de execucao: 0.000073 segundos
Numero de comparações: 5311
Numero de movimentações: 1083

[lucascosta@fedora 1.3]$ ./main 1000-quaseordenado.txt
Tipo de ordenacao: Selecao
Tempo de execucao: 0.004436 segundos
Numero de comparações: 21
Numero de movimentações: 21

Tipo de ordenacao: Insercao
Tempo de execucao: 0.000107 segundos
Numero de comparações: 6059
Numero de movimentações: 7058

Tipo de ordenacao: Bubble
Tempo de execucao: 0.005203 segundos
Numero de comparações: 505559
Numero de movimentações: 18177

[lucascosta@fedora 1.3]$ ./main 10000-quaseordenado.txt
Tipo de ordenacao: Selecao
Tempo de execucao: 0.114086 segundos
Numero de comparações: 30
Numero de movimentações: 30

Tipo de ordenacao: Insercao
Tempo de execucao: 0.000249 segundos
Numero de comparações: 76406
Numero de movimentações: 86405

Tipo de ordenacao: Bubble
Tempo de execucao: 0.114589 segundos
Numero de comparações: 50071406
Numero de movimentações: 229218
```

```
[lucascosta@fedora 1.3]$ ./main 100000-quaseordenado.txt
Tipo de ordenacao: Selecao
Tempo de execucao: 9.509157 segundos
Numero de comparações: 36
Numero de movimentações: 36

Tipo de ordenacao: Insercao
Tempo de execucao: 0.001061 segundos
Numero de comparações: 316426
Numero de movimentações: 416425

Tipo de ordenacao: Bubble
Tempo de execucao: 9.680119 segundos
Numero de comparações: 5000266426
Numero de movimentações: 949278
```

1.4/main.c

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

int main() {
    int N;

    printf("Digite o valor de N: ");
    scanf("%d", &N);

    Pessoa vetor[N];

    printf("Digite os %d nomes e idades:\n", N);
    for (int i = 0; i < N; i++) {
        printf("Nome: ");
        scanf("%s", vetor[i].nome);
        printf("Idade: ");
        scanf("%d", &vetor[i].idade);
    }

    selectionSortCrescente(vetor, N);
    printf("Selection Sort (crescente):\n");
    imprimirVetorPessoa(vetor, N);

    selectionSortDecrescente(vetor, N);
    printf("Selection Sort (decrescente):\n");
    imprimirVetorPessoa(vetor, N);

    insertionSortCrescente(vetor, N);
    printf("Insertion Sort (crescente):\n");
    imprimirVetorPessoa(vetor, N);

    insertionSortDecrescente(vetor, N);
    printf("Insertion Sort (decrescente):\n");
    imprimirVetorPessoa(vetor, N);


    return 0;
}

```

1.4/ordena.h

```h
#ifndef ORDENA_H
#define ORDENA_H

#include <stdio.h>
#include <stdlib.h>
#include <string.h>


typedef struct {
    char nome[50];
    int idade;
} Pessoa;

void trocarPessoa(Pessoa *a, Pessoa *b) {
    Pessoa temp = *a;
    *a = *b;
    *b = temp;
}

int compararPessoa(const void *a, const void *b) {

    int comparacaoNome = strcmp(((Pessoa*)a)->nome, ((Pessoa*)b)->nome);

    if (comparacaoNome != 0) {
        return comparacaoNome;
    } else {

        return ((Pessoa*)a)->idade - ((Pessoa*)b)->idade;
    }
}

void selectionSortCrescente(Pessoa vetor[], int n) {
    int i, j, min_index;
    for (i = 0; i < n-1; i++) {
        min_index = i;
        for (j = i+1; j < n; j++)
            if (compararPessoa(&vetor[j], &vetor[min_index]) < 0)
                min_index = j;
        trocarPessoa(&vetor[min_index], &vetor[i]);
    }
}

void selectionSortDecrescente(Pessoa vetor[], int n) {
    int i, j, max_index;
    for (i = 0; i < n-1; i++) {
        max_index = i;
        for (j = i+1; j < n; j++)
            if (compararPessoa(&vetor[j], &vetor[max_index]) > 0)
                max_index = j;
        trocarPessoa(&vetor[max_index], &vetor[i]);
    }
}

void insertionSortCrescente(Pessoa vetor[], int n) {
    int i, j;
    Pessoa chave;
    for (i = 1; i < n; i++) {
```

```
58          chave = vetor[i];
59          j = i - 1;
60          while (j >= 0 && compararPessoa(&vetor[j], &chave) > 0) {
61              vetor[j + 1] = vetor[j];
62              j = j - 1;
63          }
64          vetor[j + 1] = chave;
65      }
66  }
67
68  void insertionSortDecrescente(Pessoa vetor[], int n) {
69      int i, j;
70      Pessoa chave;
71      for (i = 1; i < n; i++) {
72          chave = vetor[i];
73          j = i - 1;
74          while (j >= 0 && compararPessoa(&vetor[j], &chave) < 0) {
75              vetor[j + 1] = vetor[j];
76              j = j - 1;
77          }
78          vetor[j + 1] = chave;
79      }
80  }
81
82  void imprimirVetorPessoa(Pessoa vetor[], int n) {
83      for (int i = 0; i < n; i++) {
84          printf("Nome: %s, Idade: %d\n", vetor[i].nome, vetor[i].idade);
85      }
86      printf("\n");
87  }
88
89  #endif
90
```

```
[lucascosta@fedora 1.4]$ gcc main.c -o main
[lucascosta@fedora 1.4]$ ./main
Digite o valor de N: 5
Digite os 5 nomes e idades:
Nome: Lucas
Idade: 85
Nome: Joao
Idade: 12
Nome: Edu
Idade: 16
Nome: Lari
Idade: 45
Nome: Neusvani
Idade: 42
Selection Sort (crescente):
Nome: Edu, Idade: 16
Nome: Joao, Idade: 12
Nome: Lari, Idade: 45
Nome: Lucas, Idade: 85
Nome: Neusvani, Idade: 42

Selection Sort (decrescente):
Nome: Neusvani, Idade: 42
Nome: Lucas, Idade: 85
Nome: Lari, Idade: 45
Nome: Joao, Idade: 12
Nome: Edu, Idade: 16

Insertion Sort (crescente):
Nome: Edu, Idade: 16
Nome: Joao, Idade: 12
Nome: Lari, Idade: 45
Nome: Lucas, Idade: 85
Nome: Neusvani, Idade: 42

Insertion Sort (decrescente):
Nome: Neusvani, Idade: 42
Nome: Lucas, Idade: 85
Nome: Lari, Idade: 45
Nome: Joao, Idade: 12
Nome: Edu, Idade: 16
```