Estruturas de Dados II: ed202_ex01

1 – Alterar o algoritmo de *insert_sort* para, ao final, mostrar o número de trocas realizadas.

2 – Executar o algoritmo com o vetor usado na aula: v = {1, 9, 8, 5, 3, 7, 4}

3 – Executar o algoritmo com o seguinte vetor: v = {40, 37, 95, 42, 39, 51, 60}

4 – Escreva um algoritmo que verifique se um vetor v = [1, ..., n] está em ordem crescente.

5 – Escreva um algoritmo que verifique se um vetor v = [1, ..., n] está em ordem decrescente.

6 – Escreva uma versão recursiva do algoritmo de ordenação por inserção.

7 – No algoritmo de *insert_sort*, troque a comparação da variável auxiliar "v[i] > x" por "v[i] >= x". A nova função continua produzindo uma ordenação crescente de v = [1, ..., n] ?

8 – Critique a seguinte implementação do algoritmo de ordenação por inserção:

```
void ins( int n, int v[]) {
  int i, j, x;
  for (j = 1; j < n; ++j) {
    x = v[j];
    for (i = j-1; i >= 0 && v[i] > x; --i) {
      v[i+1] = v[i];
      v[i] = x;
    }
  }
}
```

9 – Critique a seguinte implementação do algoritmo de ordenação por inserção:

```
void ins( int n, int v[]) {
  int i, j, x;
  for (j = 1; j < n; ++j) {
    for (i = j-1; i >= 0 && v[i] > v[i+1]; --i) {
      x = v[i]; v[i] = v[i+1]; v[i+1] = x;
    }
  }
}
```

10 – Critique a seguinte implementação do algoritmo de ordenação por inserção:

```
void ins( int n, int v[]) {
  int h, i, j, x;
  for (j = 1; j < n; ++j) {
```

```
      x = v[j];
      for (h = 0; h < j && v[h] <= x; ++h) ;
      for (i = j-1; i >= h; --i)
         v[i+1] = v[i];
      v[h] = x;
   }
}
```

11 – Escreva um algoritmo que permute os elementos de um vetor inteiro v = [1, ..., n] de modo que eles fiquem em ordem decrescente.

12 – Escreva um algoritmo que coloque em ordem lexicográfica um vetor de strings. Use o algoritmo de inserção.