Edson Yudi Toma --- 9791305

Trabalho 3 – ICC2

Pseudocódigo – Distribuição dos pontos

*Usar um laço para percorrer o vetor checando os nomes e parar no time desejado;
*Incrementar o número de jogos;
*Somar a quantidade de gols marcados;
*Somar a quantidade de gols sofridos;
*Somar a diferença de gols (saldo de gols);
*Checar se o time ganhou (marcou mais gols do que sofreu)
    *Se sim, incrementa o numero de vitorias;
    *Se sim, soma-se 3 pontos aos pontos do time;
*Se não ganhou, checar se empatou (gols marcados = gols sofridos)
    *Se sim, incrementar o número de empates;
    *Se sim, somar 1 ponto aos pontos do time;
*Se não, então o time perdeu
    *Incrementar o número de derrotas

Código – Distribuição dos pontos

```
void distributepoints (TIMES *times, char *nome, int gols_favor, int gols_contra) {
    int j = 0;
    while (strcmp(nome, times[j].nome))
        j++;
    times[j].njogos++;
    times[j].golsM += gols_favor;
    times[j].golsS += gols_contra;
    times[j].difgols += (gols_favor - gols_contra);
    if (gols_favor > gols_contra) {
        times[j].nvitorias++;
        times[j].pontos += 3;
    }
    else if (gols_favor == gols_contra) {
        times[j].nempates++;
        times[j].pontos++;
    }
    else
    times[j].nderrotas++;
}
```

Análise de Complexidade – Distribuição dos pontos

Como há um laço, no pior dos casos esse laço percorre n elementos.
Incrementar o número de jogos, gols e a diferença de gols são operações únicas!
Há uma checagem com 2 incrementos;
Se a primeira checagem não der certo há outra checagem com 2 incrementos;
Caso contrário há apenas um incremento;

No pior dos casos ocorrem N+4+3 = N+7. Portanto O(n);
    Onde N é o número de times!

Pseudocódigo – Ordenação

*Um laço que começa em 0 e vai até n (número de times)
*Um laço que começa no anterior + 1 e vai ate n (número de times)
Checar os prerrequisitos de ordenação e se baterem trocar os times da posição do 1 laço com a posição do segundo laço;

Código – Ordenação

```
void sortTournament (TIMES *times, int n) {
    int i, j;
    TIMES aux;
    for (i = 0; i < n; i++) {
        for (j = i; j < n; j++) {
            if (times[j].pontos > times[i].pontos) {
                aux = times[i];
                times[i] = times[j];
                times[j] = aux;
            }
            else if (times[j].pontos == times[i].pontos) {
                (...)*
            }
```

*Há um encadeamento de IFs e ELSEs para testar os casos de ordenação e casos eles batam ocorre a troca!

```
void sortTournament (TIMES *times, int n) {
   int i, j;
   TIMES aux;
   for (i = 0; i < n; i++) {
      for (j = i; j < n; j++) {
         if (times[j].pontos > times[i].pontos) {
            aux = times[i];
            times[i] = times[j];
            times[j] = aux;
         }
         else if (times[j].pontos == times[i].pontos) {
            if (times[j].nvitorias > times[i].nvitorias) {
               aux = times[i];
               times[i] = times[j];
               times[j] = aux;
            }
            else if (times[j].nvitorias == times[i].nvitorias) {
               if (times[j].difgols > times[i].difgols) {
                  aux = times[i];
                  times[i] = times[j];
                  times[j] = aux;
               }
               else if (times[j].difgols == times[i].difgols) {
                  if (times[j].golsM > times[i].golsM) {
                     aux = times[i];
                     times[i] = times[j];
                     times[j] = aux;
                  }
                  else if (times[j].golsM == times[i].golsM) {
                     if (times[j].njogos < times[i].njogos) {
                        aux = times[i];
                        times[i] = times[j];
                        times[j] = aux;
                     }
                     else if (times[j].njogos == times[i].njogos) {
                        if (strcmp(times[j].nome, times[i].nome) < 0) {
                           aux = times[i];
                           times[i] = times[j];
                           times[j] = aux;
                        }
```

Análise de Complexidade – Ordenação;

Como o segundo laço começa com o valor do laço anterior + 1, o número de vezes que os laços ocorrem é de Σ(N-i), com i de 1 à n-1, portanto temos a soma de P.A.: (N-1 + 1)/2 * N Obtendo $N^2/2$, como ocorrem no máximo 11 checagens e 3 trocas, há no máximo 14 operações.

Portanto $14*N^2/2$, $O(N^2)$, onde N = número de times;