Trabalho Prático 2

Estudante: Tarcísio de Oliveira Rocha Nº de matricula: 2010016950

1. **Introdução:**

O trabalho consiste em criar uma classificação para o Brasileirão, campeonato brasileiro de futebol. Ele deve classificar os times por numero de pontos obtidos (times utilizando o algoritmo shellsort ou quicksort que serão indicados na entrada), seguindo o proposto no enuncia, que cada vitoria dá ao time 3 pontos e um empate 1 ponto. Em caso de empate haverão outros critérios para desempate da classificação, serão avaliados vitórias, saldo de gols, gols marcados, nesta ordem. Um arquivo de entrada nos fornecerá o tamanho requerido para uma tabela hash, a quantidade de times, o numero de rodadas do campeonato e numero de jogos por rodada. Em cada rodada aparecerão os times do confronto, o placar e a data deste. A data de cada jogo deverá ser utilizada para acesso de uma tabela hash, e quando houver mais de 1 jogo no mesmo dia, o programa deve utilizar de uma arvore SBB para resolver a colisão.

1. **Implementação:**

**Organização do Código, Decisões de Implementação e Detalhes Técnicos**

O código foi separado em 10 arquivos, 2 arquivos do formato .h e 8 .c.

Foi definido valores específicos para algumas constantes:

Em futebol.h :

#define ZERO 0

#define SHELLSORT 1 //legibilidade do código

#define QUICKSORT 2 //legibilidade do código

#define NTIME 30 // Este é o valor Maximo para o nome do time.

#define VITORIA 3

#define EMPATE 1

Em hash\_SBB.h:

#define TRUE 1

#define FALSE 0

#define SBB\_VERTICAL 0

#define SBB\_HORIZONTAL 1

#define hash(v, M) (v % M) // Transforma uma chave v em um índice no intervalo 0..M-1.

**Estrutura de Dados**

Existem duas estruturas de dados no programa, separadas em dois arquivos.

Em futebol.h **(**

**typedef struct** Time{

**char** nome\_do\_time[NTIME];

**int** pontos;

**int** vitorias;

**int** saldo\_de\_gols;

**int** gols\_marcados;

}Time;

**)**

A estrutura Time contém o nome do time, pontos obtidos, vitórias conquistadas, o saldo de gols e gols marcados pela equipe.

Em hash\_SBB.h:**(**

**typedef struct** Celula{

**char** time1[NTIME];

**char** time2[NTIME];

**char** data[NTIME];

**int** placar1;

**int** placar2;

}Celula;

**struct** sbb{

Celula reg;

**struct** sbb \*esq;

**struct** sbb \*dir;

**int** esqtipo;

**int** dirtipo;

};

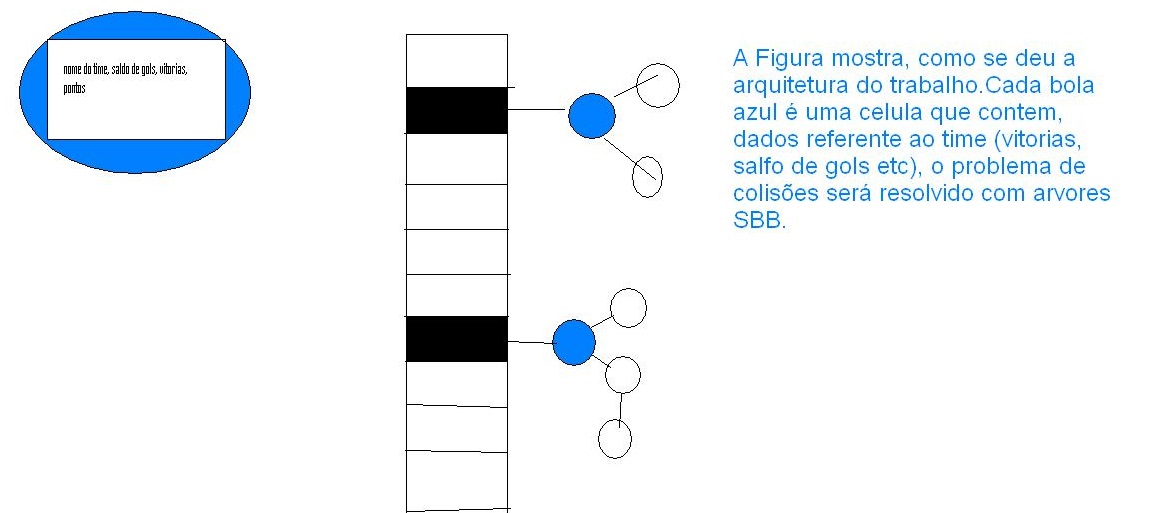
**typedef struct** Tabela\_hash{

**struct** sbb \*A;

}Tabela\_hash;

**)**

A estrutura Tabela\_hash tem um ponteiro para uma arvore SBB. Uma variável deste tipo deve criar um vetor com o tamanho da tabela que será fornecido pelo arquivo de entrada. Um exemplo da estrutura é apresentado abaixo.



**Funções e Procedimentos**

**Programa Principal**

O programa principal cria duas variável do tipo File uma para a entrada e outra para saída, abre os 2 arquivos, entrada.txt e saida.txt, da linha 14 a 41 são declaradas variáveis para os tipos apresentados, inicializados os valores para vitorias, saldo de gols, etc, que pertencem ao time, e declaradas as variáveis para a leitura dos dados.

1. **Análise de Complexidade**
2. **Programa Principal**

**Funções e Procedimentos**

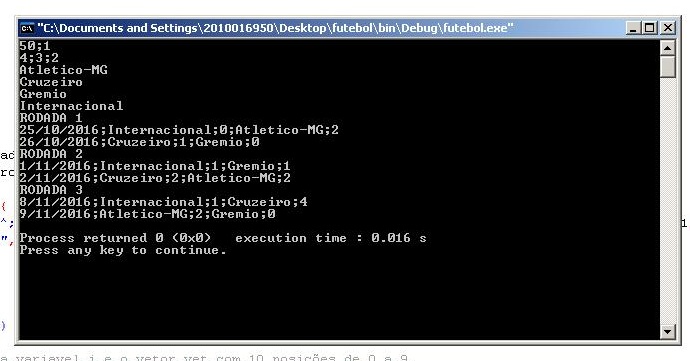
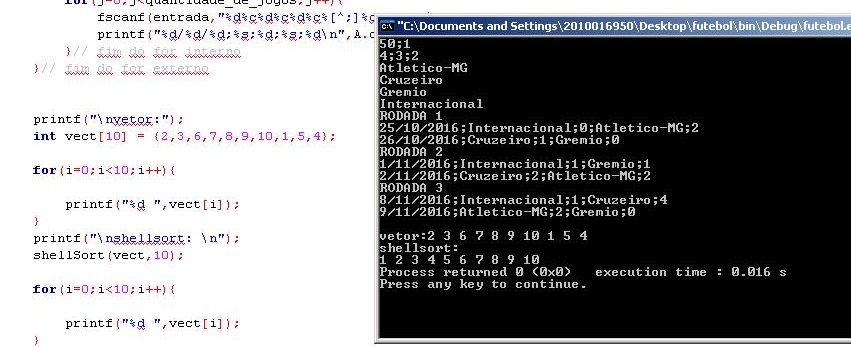
Testes:

FIGURA 1: Coleta de dados. O teste apresenta os dados coletas no arquivo de entrada do enunciado.



FIRURA 2: Neste teste executo o algoritmo de ordenação shellsort para um vetor de 10 posições(ao lado o código referente a impressão). Ocorreu tudo como esperado.

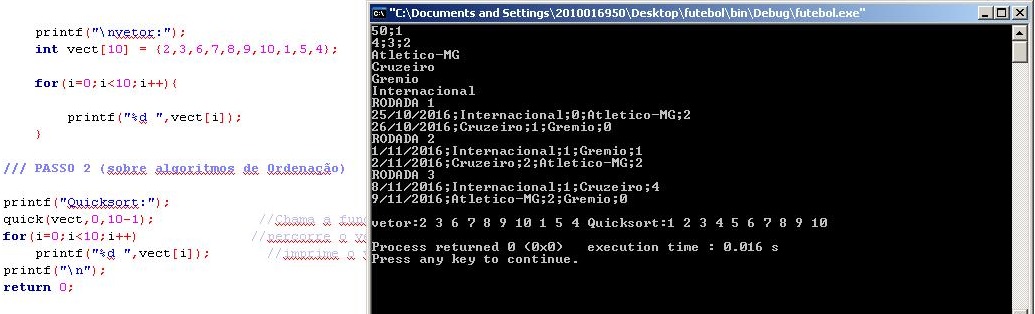


FIGURA3: Esta figura representa a execução do algoritmo de ordenação quicksort para um vetor de 10 posições, como podem ver ocorreu tudo dentor do esperado.

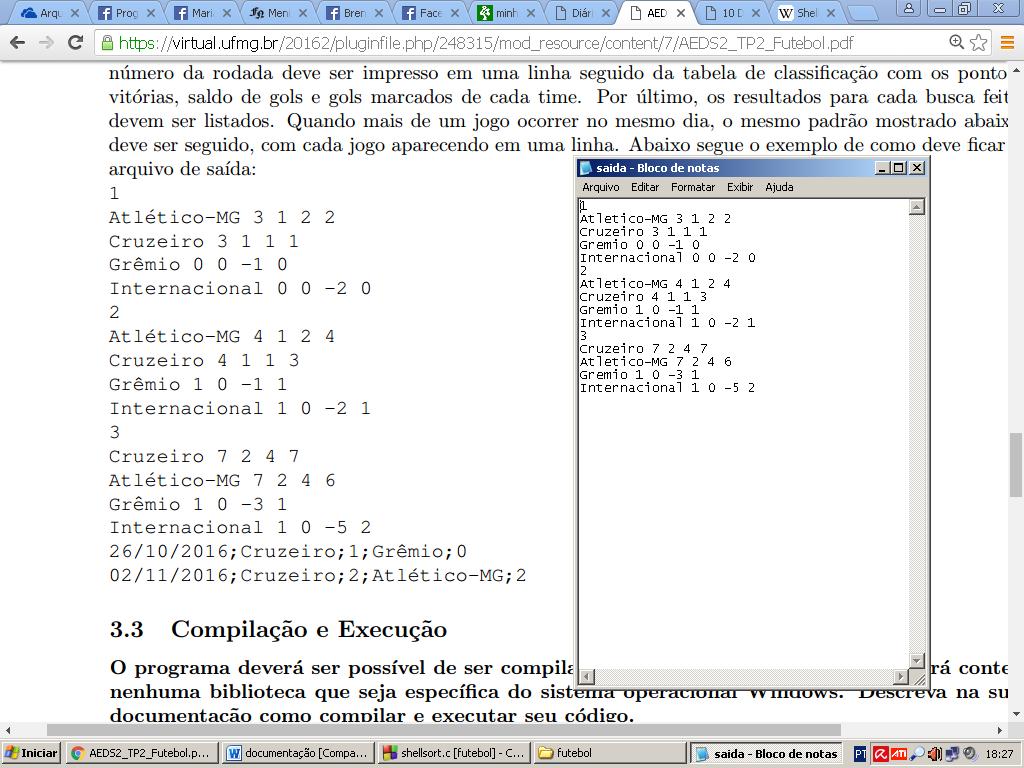


Figura 4. Neste passo o programa já está quase no seu fim, como podem o meu algoritmo atendeu a todos os criterios de ordenação.

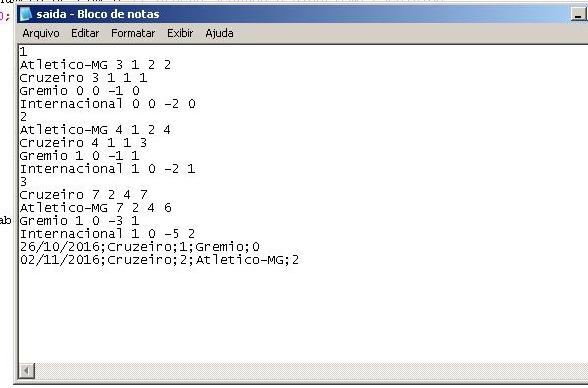


Figura 5: A figura mostra como ficou o trabalho para o caso apresentado no enunciado.

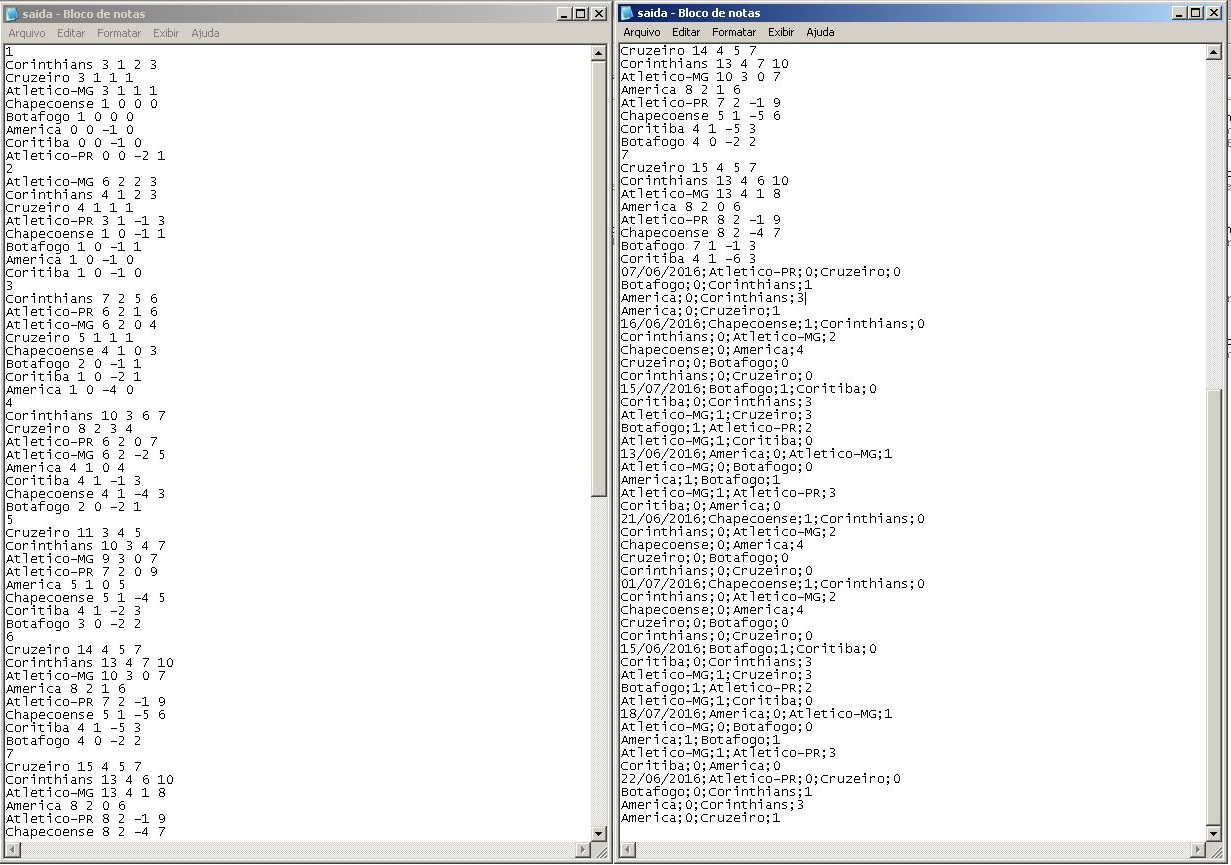


Figura 6: teste para o arquivo entrada1.txt fornecido pelos monitores. A imagem mostra o arquivo saída1.txt .

Tempo gasto para o trabalho prático 1:

Inicio: segunda-feira, 21 de novembro de 2016, 16:48

Termino:segunda-feira,5 de dezembro de 2016: 20:47

Conclusão:

No inicio do programa as dificuldades foram, perceber que o %s (mascara para string) . Tudo estava sendo lido até mesmo o ponto e vírgula, ocasionando erro (40 minutos, tempo perdido com o erro) que infelizmente não é apurado pelo compilador. No site homepages.dcc.ufmg.br, os arquivos que contem o funcionamento das arvores SBB estão errados, diferenciando-se até dos arquivos que estão no moodle da disciplina (que também não compilam), pude notar um erro na função iinsere(), quando ocorria a chamada desta função o programador errou ao passar por referência. O correto seria iinsere(reg,&(\*ptr)->esq,&(\*ptr)->esqtipo, fim); e não iinsere(reg,&(\*ptr->esq),&(\*ptr->esqtipo), fim); como está no arquivo. No entanto o programa saio dentro do esperado.

Referências:

<https://pt.wikipedia.org/wiki/Shell_sort#C.C3.B3digo_em_C> acesso: segunda-feira , 21 de novembro de 2016 as 18:26.

<https://pt.wikipedia.org/wiki/Quicksort#C> acesso: segunda-feira, 21 de novembro de 2016 as 18:28.

<http://www.cprogressivo.net/2012/11/A-funcao-printf-Caracteres-Especais.html> acesso: terça-feira, 22 de novembro de 2016 as 20:33.

<http://homepages.dcc.ufmg.br/~cunha/teaching/20121/aeds2/sbbs.pdf>acesso: terça-feira, 22 de novembro de 2016 as 20:48.

<https://virtual.ufmg.br/20162/pluginfile.php/250420/mod_resource/content/1/aeds2_aula_020_hashing-fab.pdf> acesso: terça-feira, 22 de novembro de 2016 as 21:29.

<https://www.ime.usp.br/~pf/algoritmos/aulas/stru.html> acesso : quarta-feira, 23 de novembro de 2016 as 17:33.

<https://www.ime.usp.br/~pf/mac0122-2002/aulas/hashing.html> acesso : quarta-feira, 23 de novembro de 2016 as 18:44.

<https://tentandoblogar.wordpress.com/2009/03/15/comparando-palavras-em-c-a-funcao-strcmp/> acesso: quarta-feira, 23 de novembro de 2016 as 21:23.

<http://www.cprogressivo.net/2013/09/Header-cabecalho-o-que-sao-para-que-servem-como-criar-e-usar-seus-arquivos-.h.html> acesso: segunda-feira, 23 de novembro de 2016 as 20:17.

<http://www.guj.com.br/t/resolvido-como-comparar-strings-em-c/283211> acesso: segunda-feira, 23 de novembro de 2016 as 21:37.

<http://www.hardware.com.br/comunidade/v-t/1226612/> acesso: quarta-feira, 30 de novembro de 2016 as 14:15.

**Anexos**

Listagem dos programas:

futebol.h

hash\_SBB.h

hash.c

SBB.c

futebol.c

quisort.c

shellsort

saida.c

Leia-me.c

main.c