Avaliação #2

Ordenação Externa

Matheus Henrique
Vinicius Gasparini
Professor Douglas Dutra
13/09/2018

UDESC
UNIVERSIDADE
DO ESTADO DE
SANTA CATARINA

Linguagem & Atributos

- **C++**
  - Praticidade
  - Desempenho

- Bibliotecas utilizadas:

```
#include <iostream>
#include <fstream>
#include <algorithm>
#include <vector>
#include <string>

#include <cstdio>
#include <cstdlib>
#include <ctime>
```

Estrutura e Globais

```
#define endl "\n"
#define MEDIUM_LENGTH 69

typedef struct futebol{
  /*
  This struct are used to describe a match
  */
  string teamA;
  int scoreA;
  string teamB;
  int scoreB;
  int day, month, year;
  int audience;
  string local;
}Futebol;

string original_path = "data.txt";
string sorted_path = "sorted_data.txt";
string paged_path = "paged_data.txt";
string names[4] = {"Palmeiras","Barcelona","Joinville","Liverpool"};
string locals[4] = {"Maracana","Marcilio","Esnestao","Anfieldd"};
unsigned long long id = 0;
int seletor = 0;
bool isSorted = false;
int pag = 0;

FILE *fp;
FILE *fps;
FILE *fpp;
Futebol aux;
vector<vector<string>> data;
```

UDESC

Protótipos

```
void menu();
bool sair();
void cabecalho();
void createAndStores();
void createBySize();
void createByInstance();
void fileRead();
void showData();
void sortProcess();
void sortedToFile();
void showSortedData();
void paginationToScreen();
```

UDESC

Função *main*

```c
int main(void){

    srand(time(NULL));
    menu();
    return 0;
}
```

Função *menu*

```
void menu(){
    /*
    This function is the menu os program
    */
    if (system("CLS")) system("clear");
    int op;
    cout << "Selecione modo de funcionamento" << endl << "---------" << endl << endl;
    cout << "1 - Limitar por tamanho" << endl << "2 - Limitar por partida(s)" << endl;
    cout << "3 - Mostra na tela" << endl << "4 - Ordenar e salvar no arquivo" << endl;
    cout << "5 - Mostra na tela arquivo ordenado" << endl << "6 - Paginacao na tela" << endl;
    cout << "9 - Sair" << endl << endl << "-> ";
    cin >> op;
```

UDESC

Função *menu*

```
switch (op) {
    case 1:
        fp = fopen(original_path.c_str(), "w");
        createBySize();
        fclose(fp);
        sair();
        break;
    case 2:
        fp = fopen(original_path.c_str(), "w");
        createByInstance();
        fclose(fp);
        sair();
        break;
    case 3:
        showData();
        sair();
        break;
    case 4:
        sortProcess();
        sair();
        break;
    case 5:
        showSortedData();
        sair();
        break;
    case 6:
        paginationToScreen();
        sair();
        break;
    default:
        cout << "Saindo!\n\n";
        return;
    }
}
```

UDESC

Função *sair*
Função *cabecalho*

```
bool sair(){
    /*
    This function is used to continue or not in program
    */
    int op;
    cout << endl << "Deseja sair? 0-Não | 1-Sim" << endl;
    cin >> op;
    if(op==1) return true;
    menu();
}
void cabecalho(){
    cout << "|      0-ID      | 1-Time A | 2-Placar A | 3-Time B | 4-Placar B | 5-dd/mm/aaaa | 6-Publico | 7-Local " << endl;
}
```

UDESC

Função *createAndStores*

```
void createAndStores(){
    /*
    This function simulates a random match and stores in @arq
    */
    aux.teamA = names[rand()%4];
    aux.scoreA = rand()%9;
    aux.teamB = names[rand()%4];
    aux.scoreB = rand()%9;
    aux.day = rand()%30+1;
    aux.month = rand()%12+1;
    aux.year = 2017;
    aux.audience = rand()%100000+1;
    aux.local = locals[rand()%4];
    fprintf(fp,"%'0'18ld;%s;%d;%s;%d;%'0'2d/%'0'2d/%d;%'0'6d;%s\n",id,aux.teamA.c_str()
    aux.scoreA,aux.teamB.c_str(),aux.scoreB,aux.day,aux.month,aux.year
    aux.audience,aux.local.c_str());
    id += 69;
}
```

UDESC

Função *createBySize*

```
void createBySize(){
    /*
    This function generates a file until size @limit is reached
    */
    int bytes = 0, size = 0;
    char unit;
    id = 1;
    printf("Informe o tamanho desejado ([B]ytes, [K]B, [M]B ou [G]B)\n ->");
    cin >> size >> unit;
    switch (unit){
        case 'B':
            bytes = (int)(size)/MEDIUM_LENGTH;
            break;
        case 'K':
            bytes = (int)(size * 1024)/MEDIUM_LENGTH;
            break;
        case 'M':
            bytes = (int)(size * 1048576)/MEDIUM_LENGTH;
            break;
        case 'G':
            bytes = (int)(size * 1073741824)/MEDIUM_LENGTH;
            break;
        default:
            printf("Opcao invalida... Saindo\n");
            return;
    }
    clock_t begin = clock();
    for(int i = 0; i < bytes; i++) createAndStores();
    printf("Tempo de execucao: %.4lf\n", (double)(clock() - begin) / CLOCKS_PER_SEC);
```

Função *createByInstance*

```
void createByInstance(){
    /*
    This function generates a file until instances @limit is reached
    */
    int limit;
    id = 1;
    cout << "Informe a quantidade de instancias\n ->";
    cin >> limit;
    clock_t begin = clock();
    for(int i = 0 ; i < limit; i++) createAndStores();
    printf("Tempo de execucao: %.4lf\n", (double)(clock() - begin) / CLOCKS_PER_SEC);
}
```

Função *explode* e *fileRead*

```
vector<string> explode(const string& s,const char& c){
    /*
    This function is used to separate a line each ';' in a new instance of vector
    */
    string buff{""};
    vector<string> v;

    for(auto n:s){
        if(n != c) buff+=n; else
        if(n == c && buff != "") { v.push_back(buff); buff = ""; }
    }
    if(buff != "") v.push_back(buff);
    return v;
}
void fileRead(string path){
    /*
    This function is used to read a file in @path to @data
    */
    data.clear();
    string line;
    ifstream file (path.c_str());
    if(file.is_open()){
        while(!file.eof()){
            getline(file,line);
            vector<string> v{explode(line, ';')};
            data.push_back(v);
        }
        data.pop_back();
        file.close();
    }
```

UDESC

Função *showData*

```
void showData(){
    /*
    This function show data from file to console
    */
    if(!data.size() || isSorted) fileRead(original_path);
    isSorted = false;
    cabecalho();
    for(auto n:data){
        for(auto m:n){
            cout << m << ';';
        }
        cout << endl;
    }
}
```

UDESC

Função *decide*

```cpp
bool decide(vector<string> a, vector<string> b){
    /*
    This function is a boolean type for sort algorithm
    */
    return (a[seletor] != b[seletor]) ? a[seletor] < b[seletor] : a[0] < b[0];
}
```

UDESC

Função *sortProcess*

```
void sortProcess(){
    /*
    This function is the sort process, uses a std::sort parametrized to
    rearrange from colluns. After that, rewrite into a new file usinf sortedToFile
    */
    cout << "Escolha o parametro de ordenacao\n ->";
    cabecalho();
    cin >> seletor;
    cout << "Iniciado leitura" << endl;
    clock_t begin_read = clock();
    if(!data.size()) fileRead(original_path);
    clock_t end_read = clock();
    cout << "Leitura realizada" << endl << "Ordenacao iniciada" << endl;
    clock_t begin_sort = clock();
    sort(data.begin(), data.end(), decide);
    clock_t end_sort = clock();
    clock_t begin_save = clock();
    sortedToFile();
    printf("Tempo de leitura: %.4lf\nTempo de ordenacao: %.4lf\nTempo de escrita ordenada %.4lf\n"
    (double)(end_read - begin_read) / CLOCKS_PER_SEC, (double)(end_sort - begin_sort) / CLOCKS_PER_SEC,
    (double)(clock() - begin_save) / CLOCKS_PER_SEC);
    isSorted = true;
```

UDESC

Função *sortedToFile*

```
void sortedToFile(){
    /*
    This function is used to rewrite sorted data into a new file named "data_sorted.txt"
    */
    if(data.size()){
        fps = fopen("data_sorted.txt", "w");
        string line;
        for(auto n:data){
            for(auto m:n){
                fprintf(fps, "%s",(m.append(";")).c_str());
            }fprintf(fps, "\n");
        }
        fclose(fps);
    } else {
        cout << "Memoria ainda vazia. Executando o processo de sort" << endl;
        sortProcess();
    }
}
```

UDESC

Função *showSortedData*

```
void showSortedData(){
    /*
    This function show data from file to console
    */
    if(!data.size() || !isSorted) fileRead(sorted_path);
    isSorted = true;
    cabecalho();
    for(auto n:data){
        for(auto m:n){
            cout << m << ';';
        }
        cout << endl;
    }
}
```

UDESC

Função *paginationToScreen*

```
void paginationToScreen(){
    /*
    This function shows in console paged data
    */
    if(!data.size()) fileRead(original_path);
    string line;
    int cont=0;
    cout << "Digite o tamanho da paginacao desejada" << endl << "->";
    scanf("%d",&pag);
    getchar();
    for(auto n:data){
        for(auto m:n){
            printf("%s",(m.append(";")).c_str());
        }
        printf("\n");
        cont++;
        if(cont==pag){
            getchar();
            cont=0;
            printf("\n\n");
```

UDESC

Execução

UDESC

Obrigado

matheushenriquecct@hotmail.com
viniciuszeiko@gmail.com

UDESC