#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <string.h>

#include "lista.h"

#define GB 1073741824

void nome(char\* x);

int criaCodigo(void);

void criaTempo (int\* tempo);

void criaData(int\* data);

void escreveDado (FILE \*fp, Corredor maratonista);

int cria1GB (FILE\* fp);

void escreveParcionado(FILE\* fp, Lista\* l);

int leArquivo (FILE\* fr, Lista \*l, int escolha,int index);

int insereOrdenado(Lista\* lista,Corredor corredor, int metodoDeInsercao);

int ordenaData(No\* aux, No\* atual);

int ordenaCodigo(No\* aux, No\* atual);

int oredenaNome(No\* aux, No\* atual);

int oredenaTempo(No\* aux, No\* atual);

void incrementaChar(char\* var, int index);

FILE\* juntaArq(int quantidadeArq, int escolha);

Lista\* guardaArqMemoria(FILE\* arq);

int main(int argc, char \*\*argv)

{

int i = 9090;

int escolha = 0;

int index =0;

while (i!=0){

printf("Bem vindo ao programa de maratona \n");

printf("Escolha uma opcaoo: /n 1-Criar arquivo de 1GB \n 0-Para sair \n");

scanf("%i",&i);

if(i==1){

while(escolha<1 || escolha>4){

printf("Escolha como quer ordenar o arquivo \n 1-Pelo codigo \n 2-Por nome \n 3- Pelo tempo \n4-Pela data \n");

scanf("%i",&escolha);

}

int a=0,b=0; //variaveis para controlar o intervalo que serÃ¡ percorrido

srand((unsigned)time(NULL));

FILE \*fp = NULL; //ponteiro para escrever

FILE \*fr = NULL; //ponteiro para ler o arquivo

Lista \*l;

fp = fopen("registro.bin", "wb");

if(fp==NULL){

printf("Erro na abertura do arquivo\n");

system("pause");

exit(1);

}

printf("\nCriou 1g\n");

index = cria1GB(fp);

printf("\nSaiu do Cria 1GB\n");

fr = fopen("registro.bin","rb");

if(fr==NULL){

printf("Erro na abertura do arquivo\n");

system("pause");

exit(1);

}

l = cria();//cria a lista

printf("\nEntrando no ler arquivo\n");

int quantidadeArq = leArquivo(fr,l, escolha,index);

printf("\n\nQUANTIDADE DE ARQUIVOS: %d\n\n",quantidadeArq);

printf("\nSaiu do ler arquivo\n");

//aqui ira construir o arquivo de 1GB ordenado

printf("Entrando juntaArq\n\n");

FILE\* final = juntaArq(quantidadeArq,escolha);

rewind(final);

FILE\* theOriginal = fopen("ArquivoMestre.bin","rb");

printf("Saindo juntaArq\n\n");

Lista\* lista = guardaArqMemoria(theOriginal);

printf("Digite o intervalo de a ate b que serao percorrido na lista\n");

scanf("%d %d",&a,&b);

percorre(lista,a,b);

fclose(fp);

fclose(fr);

//printf("Escolha o intervalo de a ate b que serao percorrido nos registros\n");

//scanf("%d %d",&a,&b);

//percorre(l,a,b);

//limpa(l);

//printf("PERCORRENDO OS ARQUIVOS DEPOIS DA LIMPA\n");

//percorre(l);

//percorreInvertido(l);

int deuBoa = remove("registro.bin");

printf("Deu boa = %d\n",deuBoa);

}

if(i==0){

break;

}

}

return 0;

}

void nome(char\* x)

{

int y=0,letra= 0;

char name[8];

for(y=0;y<7;y++)

{

letra = 97+(rand()%26);

name[y] = letra;

}

name[7] = '\0';

strcpy(x,name);

}

int criaCodigo(void)

{

int x=0;

x = rand()%150000;

return x;

}

void criaTempo (int\* tempo){

int hora;

int min;

int mm;

int seg;

hora = rand()%59;

min = rand()%59;

seg = rand()%59;

mm = rand()%99;

tempo[0] = hora;

tempo[1] = min;

tempo[2] = seg;

tempo[3] = mm;

}

void criaData(int\* data)

{

struct tm \*local;

time\_t t;

t= time(NULL);

local=localtime(&t);

data[0] = local->tm\_mday;

data[1] = local->tm\_mon;

data[2] = local->tm\_year;

}

void escreveDado (FILE \*fp, Corredor maratonista)

{

fwrite(&maratonista,sizeof(Corredor),1,fp);

}

int cria1GB (FILE\* fp){

int i=0;

static int index = 0;

Corredor teste;

for(i=0; i< GB/(100\*sizeof(Corredor)); i++){

index++;

Corredor maratonista;

maratonista.codigo = criaCodigo();

criaData(maratonista.data);

criaTempo(maratonista.tempo);

nome(maratonista.name);

maratonista.index = index;

escreveDado (fp, maratonista);

teste = maratonista;

}

printf("Index: %d\n\n",teste.index);

return index;

}

int leArquivo (FILE\* fr, Lista \*l, int escolha,int index2){

printf("\nEntrou ler arquivo\n");

int index = 3;

int tam = 9,i=0;

FILE\* aux = fr;

printf("\nEntrando while do ler arquivo\n");

char\* nome= malloc(sizeof(char)\*tam);

for(i=0; i<tam-5; i++){

nome[i] ='0';

nome[tam-5]='.';

nome[tam-4]='b';

nome[tam-3]='i';

nome[tam-2]='n';

nome[tam-1]='\0';

}

int vezes = 0;

while(!feof(fr)){

vezes++;

printf("\nEntrou while do ler arquivo\n");

float tamanho = 131072000/(100\*sizeof(Corredor));

int anda = 0;

//int parametro = 0;

printf("\nEntrando while do tamanho ler arquivo\n");

while (tamanho--){

Corredor corredor; //vai guardar as informacoes

fread(&corredor, sizeof(Corredor),1,fr);//leu um corredor

//dantes de adicionar, comparar com o anterior da lista, e organizar conforme o pedido(nome,data,tempo,outros).

insereOrdenado(l,corredor,escolha);

//printf("\tSaiu do insere Ordenado\n\n");

}

printf("Criando um arquivo com nome diferente\n");

FILE\* arq = NULL;

incrementaChar(nome,index);

arq = fopen(("%s",nome),"wb");

escreveParcionado(arq, l);

fclose(arq);

printf("Vai limpar a lista\n");

limpa(l);

printf("Limpou a lista\n");

}

return vezes;

}

void escreveParcionado(FILE\* fp, Lista\* l)

{

No\* aux = NULL;

for(aux = l->primeiro;aux != NULL; aux = aux->proximo)

{

fwrite(&aux->c,sizeof(Corredor), 1, fp);

}

}

Lista\* guardaArqMemoria(FILE\* arq)

{

Lista\* lista = NULL;

lista = cria();

float tamanho = GB/(100\*sizeof(Corredor));

while (tamanho--){

Corredor corredor; //vai guardar as informacoes

fread(&corredor, sizeof(Corredor),1,arq);//leu um corredor

//dantes de adicionar, comparar com o anterior da lista, e organizar conforme o pedido(nome,data,tempo,outros).

adiciona(lista,corredor);

//printf("\tSaiu do insere Ordenado\n\n");

}

return lista;

}

FILE\* juntaArq(int quantidadeArq, int escolha){

printf("Entrou juntaArq\n\n");

int i =0,x=0;//contadores

int trocouMenor = 0;

int menor = -1;

char\* menorNome = "\0";

int menorTempo[4];

int menorData[3];

int indice = 0;

int condParada = 0;

int tam = 9;

int\* valores = malloc((sizeof(int))\*quantidadeArq);

Corredor\* corredor = malloc((sizeof(Corredor))\*quantidadeArq);

char\* nome= malloc(sizeof(char)\*tam);

printf("Fez os mallocs\n\n");

for(i=0; i<tam-5; i++){

nome[i] ='0';

nome[tam-5]='.';

nome[tam-4]='b';

nome[tam-3]='i';

nome[tam-2]='n';

nome[tam-1]='\0';

}

printf("Fez os for do nome\n\n");

FILE\* arq1G = fopen("ArquivoMestre.bin","wb");

printf("Criou o arquivo final de 1GB\n\n");

//criar vetores, no qual neles estarão o topo de cada arquivo

//ler o primeiro de cada arquivo e comparar qual eh o menor

//depois de descobrir qual eh o menor tem que ler o proximo indice do qual arquivo foi lido

//IDEIA: Guardar em pilha cada arquivo? Iria acabar guardando um giga na memoria

//Outra ideia, soh guardar o valor lido, e depois ver qual indice foi o vencedor, para saber qual arquivo esta o de menor valor - EU ACHO QUE EH ISSO

FILE\*\* vetor = malloc((sizeof(FILE\*))\*quantidadeArq);

printf("Criando e abrindo o vetor de arquivos\n\n");

for(i=0;i<quantidadeArq;i++)

{

incrementaChar(nome,3);

printf("Incrementou o nome\n");

vetor[i] = fopen(("%s",nome),"rb");

printf("Abriu o arquivo em uma posicao do vetor\n");

fread(&corredor[i],sizeof(Corredor),1,vetor[i]); //ta dando ruim aqui

printf("Abriu e Leu arquivo %d\n\n",(i+1));

printf("Nome do arquivo das operacoes realizadas %s\n\n",nome);

printf("Codigo do corredor: %d\nNome do corredor: %s\n\n",corredor[i].codigo,corredor[i].name);// e aqui

}

printf("Criou o vetor de arquivos\n");

//parte do codigo que ira comparar os valores de codigo

if(escolha == 1)

{

printf("Entrou no if da escolha 1(codigo)\nEntrando no While da condParada\n\nAto DE FEH\n\n");

while(condParada<quantidadeArq)

{

menor = 999999; //NUMERO MAIOR QUE O VALOR MAXIMO DE corredor.codigo

indice = -1;

//printf("Entrando for que compara os corredores\n");

for(i =1; i<quantidadeArq; i++)

{

if(corredor[i].index == -100)

{

continue;

}

if(menor > corredor[i].codigo)

{

//printf("Fazendo a substituicao no menor\n");

menor = corredor[i].codigo;

indice = i;

//trocouMenor = 1;

}

}

if(indice<0)//condicao de seguranca, significa que nao encontrou nenhum valor de corredor

{

//printf("Deu ruim no indice\n");

break;

}

fwrite(&corredor[indice],sizeof(Corredor),1,arq1G);

//printf("Dando fread no arquivo que tinha o menor dado\n");

if(!(fread(&corredor[indice],sizeof(Corredor),1,vetor[indice])))

{

//printf("Acabou o arquivo, ou nao foi possivel ler o arquivo");

corredor[indice].index = -100; //mudar o valor de index de corrdeor para ser ignorado na hora de comparar

//vetor[indice] = NULL; fica a duvida, pois eu perco o ponteiro para o arquivo, acho melhor nao ter isso

condParada++;

}

//printf("Escrevendo no arquivo final\n");

}

printf("Saiu While!!\n\n");

}

//aqui compara caso queira ordenar com base no nome

if(escolha==2)

{

printf("Entrou escolha = 2\n\n");

while(condParada<quantidadeArq)

{

//printf("Comeco While \n\n");

menorNome = "zzzzzzz\0"; //maior que valor maximo permitido

//printf("Atribuiu valor para menorNome\n\n");

indice = -1;

//printf("Entrando for que compara os corredores\n");

for(i =1; i<quantidadeArq; i++)

{

if(corredor[i].index == -100)

{

//printf("Entrou no index do corredor\n\n");

continue;

}

if(strcmp(corredor[i].name,menorNome) < 0)

{

//printf("Uma nova menor string\n\n");

menorNome = corredor[i].name;

//printf("Copiou a nova menor string para a variavel certa\n\n");

indice = i;

}

}

if(indice<0)

{

//printf("Entrou no indice<0\n\n");

break;

}

//printf("Escrevendo no arquivo o menor string\n\n");

fwrite(&corredor[indice],sizeof(Corredor),1,arq1G);

//printf("Escreveu no arquivo\n\n");

if(!(fread(&corredor[indice],sizeof(Corredor),1,vetor[indice])))

{

//printf("Acabou o arquivo, ou nao foi possivel ler o arquivo");

corredor[indice].index = -100; //mudar o valor de index de corrdeor para ser ignorado na hora de comparar

//vetor[indice] = NULL; fica a duvida, pois eu perco o ponteiro para o arquivo, acho melhor nao ter isso

condParada++;

}

//printf("Deu o fread, voltando pro começo\n\n");

}

}

//aqui ira comparar caso queira guardar os dados com base no tempo.

if(escolha == 3)

{

while(condParada<quantidadeArq)

{

for(x=0;x<4;x++){

menorTempo[x] = 9999;

}

indice = -1;

for(i =1; i<quantidadeArq; i++)

{

if(corredor[i].index == -100)

{

continue;

}

// CONDICOES PARA ACHAR O MENOR TEMPO

if(menorTempo[0] > corredor[i].tempo[0])//compara as horas

{

//printf("Fazendo a substituicao no menor\n");

for(x=0;x<4;x++)

{

menorTempo[x] = corredor[i].tempo[x];

}

indice = i;

//trocouMenor = 1;

}else{

if(menorTempo[0] == corredor[i].tempo[0])//se as horas for igual

{

if(menorTempo[1] > corredor[i].tempo[1])//compara os minutos

{

for(x=0;x<4;x++)

{

menorTempo[x] = corredor[i].tempo[x];

}

indice = i;

}else{

if(menorTempo[1] == corredor[i].tempo[1])//se os minutos for iguais

{

if(menorTempo[2] > corredor[i].tempo[2])//compara os segundos

{

for(x=0;x<4;x++)

{

menorTempo[x] = corredor[i].tempo[x];

}

indice = i;

}else{

if(menorTempo[2] == corredor[i].tempo[2])//se os segundos forem iguais

{

if(menorTempo[3] > corredor[i].tempo[3])//compara os milesimos

{

for(x=0;x<4;x++)

{

menorTempo[x] = corredor[i].tempo[x];

}

indice = i;

}//if dos milesimos

}//if dos segundos iguais

}//else

}//if dos segundos

}//else

}//if do tempo igual

}//else

}//o for das condicoes para achar o menor indice

if(indice<0)//condicao de seguranca, significa que nao encontrou nenhum valor de corredor

{

//printf("Deu ruim no indice\n");

break;

}

fwrite(&corredor[indice],sizeof(Corredor),1,arq1G);

//printf("Dando fread no arquivo que tinha o menor dado\n");

if(!(fread(&corredor[indice],sizeof(Corredor),1,vetor[indice])))

{

//printf("Acabou o arquivo, ou nao foi possivel ler o arquivo");

corredor[indice].index = -100; //mudar o valor de index de corrdeor para ser ignorado na hora de comparar

//vetor[indice] = NULL; fica a duvida, pois eu perco o ponteiro para o arquivo, acho melhor nao ter isso

condParada++;

}

}

}

if(escolha == 4)

{

//data da corrida data[0] = dia data[1] = mes data[2] = ano

while(condParada<quantidadeArq)

{

for(x=0;x<3;x++){

menorData[x] = 9999;

}

indice = -1;

for(i =1; i<quantidadeArq; i++)

{

if(corredor[i].index == -100)

{

continue;

}

if(menorData[0] > corredor[i].data[0])//compara as horas

{

//printf("Fazendo a substituicao no menor\n");

for(x=0;x<3;x++)

{

menorData[x] = corredor[i].data[x];

}

indice = i;

//trocouMenor = 1;

}else{

if(menorData[0] == corredor[i].data[0])

{

if(menorData[1] > corredor[i].data[1])

{

for(x=0;x<3;x++)

{

menorData[x] = corredor[i].data[x];

}

indice = i;

}else{

if(menorData[1] == corredor[i].data[1])

{

if(menorData[2] > corredor[i].data[2])

{

for(x=0;x<3;x++)

{

menorData[x] = corredor[i].data[x];

}

indice = i;

}//if do ano

}//if do mes igual

}//else

}//if do dia igual

}//else

}//for

if(indice<0){

break;

}

fwrite(&corredor[indice],sizeof(Corredor),1,arq1G);

//printf("Dando fread no arquivo que tinha o menor dado\n");

if(!(fread(&corredor[indice],sizeof(Corredor),1,vetor[indice])))

{

//printf("Acabou o arquivo, ou nao foi possivel ler o arquivo");

corredor[indice].index = -100; //mudar o valor de index de corrdeor para ser ignorado na hora de comparar

//vetor[indice] = NULL; fica a duvida, pois eu perco o ponteiro para o arquivo, acho melhor nao ter isso

condParada++;

}

}//while

}//if do escolha == 4

//testes aqui para o arq1G

FILE\* teste = fopen("ArquivoMestre.bin","rb");

Corredor testeC;

fread(&testeC,sizeof(Corredor),1,teste);

printf("Nome: %s\n",testeC.name);

printf("Codigo: %d\n",testeC.codigo);

printf("Data: %d %d %d\n",testeC.data[0],testeC.data[1],testeC.data[2]);

printf("Tempo: %d %d %d %d\n",testeC.tempo[0],testeC.tempo[1],testeC.tempo[2],testeC.tempo[3]);

//O menor, que foi passado por parâmetro, será o que irá no arquivo final

//O topo desse arquivo, que foi tirado a informação, será atualizado para o proximo

//Assim sucessivamente até todos os arquivos chegarem no final

//Fechar todos arquivos

//aqui ira fechar os arquivos

printf("Fechando os arquivos finais\n");

for(i = 0;i<quantidadeArq;i++)

{

fclose(vetor[i]);

}

//aqui ira excluir os arquivos

printf("Excluindo os arquivos\n");

for(i=0; i<tam-5; i++){

nome[i] ='0';

nome[tam-5]='.';

nome[tam-4]='b';

nome[tam-3]='i';

nome[tam-2]='n';

nome[tam-1]='\0';

}

printf("FEz os for dos nomes e ira excluir cada arquivo\n");

for(i=0;i<quantidadeArq;i++)

{

incrementaChar(nome,3);

remove(("%s",nome));

}//Excluir os outros arquivos

return arq1G;

}

/\*

Ler o primeiro de cada arquivo organizado, depois ve qual eh o menor e joga para o arquivo mestre,

depois voce vai no arquivo que tem o menor indice e le mais um, e vai fazendo esse processo ate ter lidos todos os arquivos,

arvore de peso, nao sei o nome dessa parada, o negocio do slide

\*/

// Vou implementar aqui a ideia de insercao ordenada

int insereOrdenado(Lista\* lista,Corredor corredor, int metodoDeInsercao)

{

//printf("Entrando insere ordenado\n");

int resultado = 0;

No\* contador=NULL;

No\* aux = malloc(sizeof(No));

aux->anterior = NULL;

aux->c = corredor;

aux->proximo = NULL;

if(lista->primeiro == NULL)

{

lista->primeiro = aux;

lista->ultimo = aux;

//printf("Imprimindo lista ordenada\n\n");

/\*for(contador=lista->primeiro;contador!=NULL;contador = contador->proximo)

{

printf("%d\n",contador->c.codigo);

}\*/

return 1;

}

No\* atual = lista->primeiro;

while(atual!=NULL)

{

int valorDeRetorno;

switch (metodoDeInsercao){

case 1:

//printf("Entrou caso 1\n\n");

resultado = ordenaCodigo(aux, atual);//retorna 1 se o codigo de aux for menor que atual

break;

case 2:

resultado = ordenaNome(aux, atual);

break;

case 3:

resultado = ordenaTempo(aux, atual);

break;

case 4:

resultado = ordenaData(aux, atual);

break;

}

if(resultado)//escrever condicional com valor de retorno da função de comparação de vocês.

{

if(atual == lista->primeiro)

{

aux->proximo = atual;

atual->anterior = aux;

lista->primeiro = aux;

/\*printf("Imprimindo lista ordenada\n\n");

for(contador=lista->primeiro;contador!=NULL;contador = contador->proximo)

{

printf("%d\n",contador->c.codigo);

}\*/

return 1;

}

//Rotina para inserir entre 2 registro (atual->anterior / atual) da lista

aux->anterior = atual->anterior;

atual->anterior->proximo = aux;

aux->proximo = atual;

atual->anterior = aux;

/\*printf("Imprimindo lista ordenada\n\n");

for(contador=lista->primeiro;contador!=NULL;contador = contador->proximo)

{

printf("%d\n",contador->c.codigo);

}\*/

return 1;

}

if(atual==lista->ultimo){

//Rotina para inserir no final da lista

aux->anterior = atual;

lista->ultimo = aux;

atual->proximo = aux;

/\*printf("Imprimindo lista ordenada\n\n");

for(contador=lista->primeiro;contador!=NULL;contador = contador->proximo)

{

printf("%d\n",contador->c.codigo);

}\*/

return 1;

}

atual=atual->proximo;

}

/\*printf("Imprimindo lista ordenada\n\n");

for(contador=lista->primeiro;contador!=NULL;contador = contador->proximo)

{

printf("%d\n",contador->c.codigo);

}\*/

return 1;

}

///////////////ARRUMAR ESSA PORRA AQUI EMBAIXO////////////////////////

//Verificar porque ordena codigo,nome e tempo esta dando ruim

int ordenaCodigo(No\* aux, No\* atual)

{

if(aux->c.codigo < atual->c.codigo)

{

//printf("Retornou 1 em ordenaCodigo\n");

//printf("Valor do codigo novo: %d\n",aux->c.codigo);

return 1;

}

//printf("Valor do codigo novo: %d\n",aux->c.codigo);

//printf("Retornou 0\n");

return 0;

}

int ordenaNome(No\* aux, No\* atual)

{

if(strcmp(aux->c.name,atual->c.name) < 0)

{

return 1;

}

return 0;

}

int ordenaTempo(No\* aux, No\* atual)

{

if(aux->c.tempo[0] < atual->c.tempo[0])

{

return 1;

}

if(aux->c.tempo[1] < atual->c.tempo[1])

{

return 1;

}

if(aux->c.tempo[2] < atual->c.tempo[2])

{

return 1;

}

if(aux->c.tempo[3] < atual->c.tempo[3])

{

return 1;

}

return 0;

}

int ordenaData(No\* aux, No\* atual)

{

if(aux->c.data[2] < atual->c.data[2])

{

return 1;

}

if(aux->c.data[1] < atual->c.data[1])

{

return 1;

}

if(aux->c.data[0] < atual->c.data[0])

{

return 1;

}

return 0;

}

void incrementaChar(char\* var, int index){

int i;

var[index]=var[index]+1;

for(i=index; i>0; i--){

if(var[i]>'9'){

var[i]='0';

var[i-1]=var[i-1]+1;

}

}

if(var[0]>'9')

var[0]='A';

}

/\*TODO

//Lembrar de fechar os arquivos

//Lembrar de excluir os arquivos ANTERIORES que ja foram usados

//arrumar os ordena

//e implementar o juntaARq

//Se der tempo: fazer busca, fazer o excluir no arquivo. Mostrar parcionado

\*/