Desafio 2

Dados dois arquivos, A.txt e B.txt, você deverá fazer o seguinte:

1- Ordenar os elementos contidos nesses arquivos usando quicksort(valores numéricos aleatórios).

2- Realizar um merge dos arquivos

3- Gerar como saída um arquivo C.txt

Para verificar se o arquivo gerado no item 3 está consistente (contém todos os elementos do arquivo A e do arquivo B ordenados), será usado um programa, durante o processo de correção, para verificar se o seu arquivo está igual ao gabarito. Se estiver, o output será “CERTO” e, se estiver errado, o programa irá mostrar onde há inconsistência e exibir a mensagem “Tente novamente”. A correção será baseada no nível de acertividade.

A fim de auxiliar na organização, parte dos programas a serem desenvolvidos serão disponibilizados abaixo, bem como o programa de verificação. Esse programa permite verificar se o seu arquivo C resultante é igual ao arquivo CGabarito.txt, e será utilizado no processo de correção. O seu trabalho estará correto se o seu arquivo resultante passar no teste. Observe que você não tem acesso ao arquivo de gabarito, o programa está sendo disponibilizado apenas para fins de esclarecimento.

Além disso, convenciona-se que o vetor começa no índice 1.

**ORDENAR:**

#include <stdio.h>

#include <stdlib.h>

#define SIZE 1001 // vetor comeca em indice 1 e ambos tem tamanho 1000

void quicksort(int \*v, int n);

void particiona(int \*v, int l, int r);

int main()

{

FILE \*fileA = fopen("A.txt","r");

FILE \*fileB = fopen("B.txt","r");

// ler do arquivo e inserir em um vetor: arrayA, arrayB

// abriu para ler, fechar

fclose(fileA);

fclose(fileB);

// ordenar vetores

quicksort(arrayA,SIZE);

quicksort(arrayB, SIZE);

// abrir arquivo para escrever

// escrever: inserir dados ordenados

// liberar espaco dos vetores e fechar arquivos depois de fazer todos os procedimentos necessarios

free(arrayA);

free(arrayB);

fclose(A);

fclose(B);

return 0;

}

//QUICKSORT

//particao

void particiona(int \*v, int l, int r)

{

} //particionamentos feitos ate que tenha 1 elemento

void quicksort(int \*v, int n)

{

particiona(v,1,n);

}

**MERGE:**

#include <stdio.h>

void mergeFiles(FILE \*C)

{

FILE \*fA = fopen("A.txt","r");

FILE \*fB = fopen("B.txt","r");

// merge

}

fclose(fA);

fclose(fB);

}

int main()

{

FILE \*file;

file = fopen("C.txt","w+"); // file resultante

mergeFiles(file);

fclose(file);

return 0;

}

**Verificação:**

#include <stdio.h>

#include <stdlib.h>

#define SIZE 4000 // vetor comeca em indice 1

int verifica(int \*arrayC, int \*arrayCGabarito);

int main()

{

// abrir arquivo C e verificar se ele esta correto:

FILE \*fileC = fopen("C.txt","r"); // seu arquivo C

FILE \*fileCGabarito = fopen("CGabarito.txt","r");

int i;

int \*arrayC;

int \*arrayCGabarito;

arrayC = (int\*)malloc(SIZE\*sizeof(int));

arrayCGabarito = (int\*)malloc(SIZE\*sizeof(int));

// ler do arquivo e inserir no vetor

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

{

fscanf(fileC,"%d",&arrayC[i]);

fscanf(fileCGabarito,"%d",&arrayCGabarito[i]);

}

int verificacao = verifica(arrayC,arrayCGabarito);

if(verificacao == 1)

printf("CERTO! PARABENS!");

else

printf("Tente novamente.");

free(arrayC);

free(arrayCGabarito);

fclose(fileCGabarito);

fclose(fileC);

}

int verifica(int \*arrayC, int \*arrayCGabarito)

{

int i=0;

int certo = 1;

for(i=1;i<SIZE;i++)// foi convencionado que os vetores comecem no indice 1

{

if(arrayC[i] != arrayCGabarito[i])

{

printf("\nErro no indice %d, %d != %d\n",i,arrayC[i],arrayCGabarito[i]);

certo = 0;

}

}

return certo;

}