Lab08

Mark Ribeiro – 1612043

Jonny Russo – 1610608

Não foram utilizados os algoritmos elaborados no GPLF pois nenhum membro da dupla lembrou muito bem o seu código. Então preferimos começar um código novo juntos.

Código 1.1)

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <time.h>

#define SIZE 35

char vetor[40][SIZE];

int coli;

int hash(char\* palavra);

int main (void)

{

char \*palavra;

FILE \*f = fopen("nomes.txt","r");

int i, posi;

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

{

palavra = (char\*) malloc(40\*sizeof(char));

fread(palavra,40,1,f);

posi = hash(palavra);

}

printf("Colisoes:%d\n",coli);

fclose(f);

return 0;

}

int hash(char\* palavra)

{

int posi,i;

char aux = 0;

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

{

aux = aux + palavra[i]\*23;

}

posi = (int)aux % SIZE;

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

{

if(vetor[40][posi] == NULL)

{

strcpy(&vetor[40][posi],palavra);

return posi;

}

coli++;

if (strcmp(&vetor[40][posi],palavra)==0)

return posi;

posi++;

if (posi>SIZE)

posi = posi - SIZE;

}

return -1;

}

Código 1.2)

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <math.h>

typedef struct tree

{

int key;

struct tree \*l;

struct tree \*r;

}Tree;

int AVL(Tree\* Arv);

int verifyAVL(Tree\* Arv);

Tree\* createNode(int key);

Tree\* insert(int key, Tree\* arv);

int main(void)

{

Tree\* arv1 = NULL, \*arv2 = NULL;

int i = 0;

int vetor1[15] = {8,4,12,2,6,10,14,1,3,5,7,9,11,13,15};

int vetor2[15] = {8,4,12,2,10,14,1,3,5,6,7,9,11,13,15};

for(i = 0; i < 15; i++)

arv1 = insert(vetor1[i], arv1);

for(i = 0; i < 15; i++)

arv2 = insert(vetor2[i], arv2);

if(verifyAVL(arv1) == 1)

printf("Arv1 eh AVL\n");

else

printf("Arv1 nao e AVL\n");

if(verifyAVL(arv2) == 1)

printf("Arv2 eh AVL\n");

else

printf("Arv2 nao eh AVL\n");

return 0;

}

int AVL(Tree\* Arv)

{

int l, r;

if(Arv == NULL)

return -1;

l = AVL(Arv->l);

r = AVL(Arv->r);

return abs(l-r);;

}

int verifyAVL(Tree\* Arv)

{

return AVL(Arv) < 2;

}

Tree\* createNode(int key)

{

Tree\* new = (Tree\*) malloc(sizeof(Tree));

new->key = key;

new->r = NULL;

new->l = NULL;

return new;

}

Tree\* insert(int key, Tree\* arv)

{

if(arv == NULL)

return createNode(key);

else if(key < arv->key)

arv->l = insert(key,arv->l);

else if (key > arv->key)

arv->r = insert(key,arv->r);

return arv;

}