ACH2023 - Atividades Semanais

Prof. Luciano Antonio Digiampietri

Atividade Semanal 10

1 Exercício 1 (exercício único da semana)

Nesta atividade semanal trataremos árvores binárias (árvores nas quais cada nó possui até dois filhos: um à esquerda e um à direita). De fato, este será o assunto principal do restante do semestre.

As árvores binárias vistas nesta atividade são árvores que não possuem uma ordenação em relação às suas chaves. A inserção ocorre de acordo com o pedido do usuário (o usuário informa, a cada inserção, quem será o pai do nó inserido e se o nó inserido deverá ficar à esquerda ou à direita desse nó).

Esta atividade é dividida em duas partes.

1) Escreva o que será impresso pela execução do código desta atividade (o arquivo .c deste código está disponível no site da disciplina). Tente entender cada um dos resultados produzidos por esta execução.

2) Desenhe a árvore binária resultante da execução deste código.

Para esta atividade, entregue um arquivo PDF contendo o resultado da execução do programa (o conteúdo que é impresso pela execução do código desta atividade) e também o desenho da árvore binária resultante da execução.

```
#include <stdio.h>
#include <malloc.h>
#define true 1
#define false 0
typedef enum{esquerdo,direito} LADO;
typedef int bool;
typedef int TIPOCHAVE;

typedef struct aux{
  TIPOCHAVE chave;
  struct aux *esq, *dir;
} NO, *PONT;



PONT buscarChave(TIPOCHAVE ch, PONT raiz){
```

```
  if (raiz == NULL) return NULL;
  if (raiz->chave == ch) return raiz;
  PONT aux = buscarChave(ch,raiz->esq);
  if (aux) return aux;
  return buscarChave(ch,raiz->dir);
}

void apagar(PONT raiz){
  if (!raiz) return;
  apagar(raiz->esq);
  apagar(raiz->dir);
  free(raiz);
}

PONT criarNovoNo(TIPOCHAVE ch){
  PONT novoNo = (PONT)malloc(sizeof(NO));
  novoNo->esq = NULL;
  novoNo->dir = NULL;
  novoNo->chave = ch;
  return novoNo;
}

bool inserirFilho(PONT raiz, TIPOCHAVE novaChave, TIPOCHAVE chavePai, LADO lado){
  PONT pai = buscarChave(chavePai,raiz);
  if (!pai) return false;
  PONT novo = criarNovoNo(novaChave);
  if (lado == esquerdo){
    apagar(pai->esq);
    pai->esq = novo;
  }else{
    apagar(pai->dir);
    pai->dir = novo;
  }
  return true;
}

void exibirArvoreOrdemW(PONT raiz){
  if (raiz == NULL) return;
  exibirArvoreOrdemW(raiz->esq);
  exibirArvoreOrdemW(raiz->dir);
  printf("%i ",raiz->chave);
}



int max(int a, int b){
```

```
  if (a>b) return a;
  return b;
}

int funcaoZZZ(PONT raiz){
  if (!raiz ) return -1;
  return 1 + max(funcaoZZZ(raiz->esq), funcaoZZZ(raiz->dir));
}

int funcaoX(PONT raiz){
  if (!raiz ) return 0;
  return 1 + funcaoX(raiz->esq) + funcaoX(raiz->dir);
}

void inicializar(PONT* raiz){
  *raiz = NULL;
}

void criarRaiz(PONT* raiz, TIPOCHAVE novaChave){
  *raiz = criarNovoNo(novaChave);
}

int main(){
  PONT raiz;
  inicializar(&raiz);
  criarRaiz(&raiz,1);
  inserirFilho(raiz,2,1,direito);
  inserirFilho(raiz,3,1,esquerdo);
  printf("FuncaoZZZ (1a execucao): %i\n",funcaoZZZ(raiz));
  printf("FuncaoX (1a execucao): %i\n",funcaoX(raiz));
  printf("Imprimindo (1a execucao): ");
  exibirArvoreOrdemW(raiz);
  printf("\n");

  inserirFilho(raiz,4,2,esquerdo);
  inserirFilho(raiz,5,2,direito);
  inserirFilho(raiz,6,2,esquerdo);
  inserirFilho(raiz,7,6,direito);
  printf("FuncaoZZZ (2a execucao): %i\n",funcaoZZZ(raiz));
  printf("FuncaoX (2a execucao): %i\n",funcaoX(raiz));
  printf("Imprimindo (2a execucao): ");
  exibirArvoreOrdemW(raiz);
  printf("\n");
  return 0;
}
```