Ministerul Educaţiei și Cercetării al Republicii Moldova

Universitatea Tehnică a Moldovei

Facultatea Calculatoare, Informatică şi Microelectronică

**RAPORT**

Lucrare de laborator Nr.6

*la Structuri de Date și Algoritmi*

Tema: Algoritmi de prelucrare a tipului abstract de date

„Arbori binari”

A efectuat: st. gr. SI-212 Șeremet Alexandru

A verificat: lect. asist. Mititelu Vitalii

Chişinău 2022

**Scopul:**

Obținerea deprinderilor practice de implementare și de utilizare

a tipului abstract de date (TAD) „Arbore binar” cu asigurarea operațiilor de

prelucrare de bază ale arborelui binar oarecare prin parcurgerea recursivă

a nodurilor arborelui folosind algoritmi recursivi sau structurile respective

de date „coadă” și „stivă”.

**Sarcina:**

Să se scrie 3 fișiere-text în limbajul C pentru implementarea și utilizarea

TAD „Arbore binar” cu asigurarea operațiilor de prelucrare de bază ale

arborilor binari oarecareprin parcurgerea nodurilor arborelui cu ajutorul

algoritmilor recursivi sau iterativi:

1. Fișier antet cu extensia .h, care conține specificarea structurii de date a

nodului arborelui binar (conform variantelor) și prototipurile funcțiilor de

prelucrare de bază ale arborilor binari.

2. Fișier cu extensia .c sau .cpp, care conține implementările (codurile)

funcțiilor declarate în fișierul antet.

3. Fișier al utilizatorului, funcția mаin() pentru prelucrarea arborelui binar

oarecare cu afișarea la ecran a următorului meniu de opțiuni de bază:

1. Crearea nodurilor arborelui binar oarecare în memoria dinamică și

introducerea informației despre nodurile arborelui de la tastatură în mod interactiv.

2. Afișarea informației despre nodurile arborelui la ecran.

3. Căutarea nodului în arbore.

4. Modificarea informației unui nod din arbore.

5. Determinarea numărului de noduri.

6. Determinarea înălțimii arborelui.

7. Eliberarea memoriei alocate pentru listă.

0. Ieșire din program.

**Varianta 15:**

Structura Computer cu câmpurile: modelul, procesorul, memoria,

viteza, prețul.

**Rezumat:**

Definiție și proprietăți

Definiție. Se numește arbore binar un arbore cu rădăcină în care fiecare nod are cel mult doi descendenți direcți: descendentul stâng și descendentul drept.

Exemplu.



În arborele de mai sus:

nodul 1 este rădăcina;

rădăcina are doi descendenți direcți (fii): 2 este descendent stâng a lui 1, iar 3 este descendent drept;

nodul 2 are un singur descendent, pe 4. Acest este descendent stâng al lui 2;

similar, 7 este descendent drept al lui 4;

nodurile 5, 6 și 7 sunt noduri terminale (frunze).

Observație. Dacă un nod are un singur descendent acesta va fi descendent stâng sau drept – acest lucru trebuie să fie precizat. Cei doi arbori de mai jos sunt considerați distincți.



O altă definiție, recursivă, a arborelui binar este următoarea: Un arbore binar este o mulțime finită de noduri, astfel:

există un nod special, numit rădăcina arborelui;

celelalte noduri sunt grupate în două submulțimi disjuncte A1 și A2, fiecare dintre ele fiind la rândul lor arbori binari. Arborii A1 și A2 reprezintă subarborele stâng, respectiv subarborele drept, al rădăcinii.



Această definiție recursivă este utilă deoarece foarte multe probleme cu arbori binari constau în prelucrarea nodurilor din arbore: a rădăcinii și a celor doi subarbori, în mod recursiv, prin metoda Divide et Impera.

Proprietății ale arborilor binari

Fiecare nivel i=0,1,2,... dintr-un arbore binar conține cel mult 2i noduri.

Un arbore binar de înălțime h conține cel mult 2h+1-1 noduri. Demonstrația se bazează pe afirmația anterioară.

Într-un arbore cu n noduri și înâlțime h avem relația h≥log2(n+1)−1.

Dacă într-un arbore binar numărul nodurilor terminale este a, iar c este numărul nodurilor care au exact 2 fii, atunci a = c + 1.

Tipuri speciale de arbori binari

Arbore binar strict

Definiție: Se numește arbore binar strict un arbore binar în care fiecare nod, cu excepția celor terminale, are exact doi descendenți.

Exemplu:



Proprietăți:

Un arbore binar strict cu k noduri terminale are n=2⋅k–1 noduri.

Un arbore binar strict are număr impar de noduri.

Arbore binar plin

Definiție: Se numește arbore binar plin un arbore binar în care fiecare nivel k∈{0,1,2,⋯,h}, unde h este înălțimea arborelui, conține 2k noduri.

Exemplu:



Proprietăți:

Arborele binar plin este un caz particular de arbore binar strict, în care toate nodurile terminale sunt situate pe același nivel.

Un arbore binar plin cu k noduri terminale are n=2⋅k–1 noduri.

Un arbore binar plin cu înălțimea h are 2h−1 noduri.

Arbore binar complet

Definiție: Se numește arbore binar complet un arbore binar în care fiecare nivel k∈{0,1,2,⋯,h–1}, unde h este înălțimea arborelui, conține 2k noduri, iar nivelul k conține eventual mai puțin de 2h noduri, acestea fiin grupate de regulă în partea stângă.

Exemplu:



Observație: Arborele binar plin este și arbore binar complet.

**Codul deplin al programului:**

**Fișierul code.c:**

#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

#include <string.h>

#include "includes.h"

Node \*root = NULL;

Computer introduce\_struct()

{

Computer in;

printf("modelul: ");

scanf("%s", in.modelul);

printf("procesorul: ");

scanf("%s", in.procesorul);

printf("memoria: ");

scanf("%f", &in.memoria);

printf("viteza: ");

scanf("%f", &in.viteza);

printf("pretul: ");

scanf("%f", &in.pretul);

printf("\n");

return in;

}

Node \*createNode(Computer value)

{

Node \*result = malloc(sizeof(Node));

if (result != NULL)

{

result->left = NULL;

result->right = NULL;

result->value = value;

}

return result;

}

bool insertStruct(Node \*\*rootptr, Computer value)

{

Node \*root = \*rootptr;

if (root == NULL)

{

(\*rootptr) = createNode(value);

return true;

}

if (strcmp(value.modelul, root->value.modelul) == 0)

return false;

if (strcmp(value.modelul, root->value.modelul) < 0)

return insertStruct(&(root->left), value);

else

return insertStruct(&(root->right), value);

}

int treeNodeNum = 0;

void createTree()

{

while (true)

{

Computer in;

printf("valoarea nodului %d:\n", treeNodeNum);

in = introduce\_struct();

if (!insertStruct(&root, in))

printf("valoarea data deja exista.\n");

treeNodeNum++;

int choice;

printf("doriti sa mai introduceti o structura? (1/0) ");

if (scanf("%d", &choice) && choice == 0)

return;

else if (choice != 1)

break;

}

}

void printTabs(int len)

{

for (int i = 0; i < len; i++)

{

printf("\t");

}

}

void show\_struct(Computer in, int level)

{

printTabs(level);

printf("modelul: ");

printf("%s\n", in.modelul);

printTabs(level);

printf("procesorul: ");

printf("%s\n", in.procesorul);

printTabs(level);

printf("memoria: ");

printf("%f\n", in.memoria);

printTabs(level);

printf("viteza: ");

printf("%f\n", in.viteza);

printTabs(level);

printf("pretul: ");

printf("%f\n", in.pretul);

}

void printTreeRec(Node \*root, int level)

{

if (root == NULL)

{

printTabs(level);

printf("<empty>\n");

return;

}

show\_struct(root->value, level);

printTabs(level);

printf("left\n");

printTreeRec(root->left, level + 1);

printTabs(level);

printf("right\n");

printTreeRec(root->right, level + 1);

printTabs(level);

printf("done\n");

}

void printTree()

{

printTreeRec(root, 0);

}

Node \*findStructRec(Node \*root, char \*value)

{

if (root == NULL)

return NULL;

if (strcmp(root->value.modelul, value) == 0)

return root;

if (strcmp(value, root->value.modelul) < 0)

return findStructRec(root->left, value);

else

return findStructRec(root->right, value);

}

Node \*findStruct(char \*value)

{

return findStructRec(root, value);

}

void modify(char \*value)

{

Node \*change;

change = findStruct(value);

change->value = introduce\_struct();

}

int countNodesRec(Node \*root)

{

int res = 0;

if (root->right != NULL)

res += countNodesRec(root->right) + 1;

if (root->left != NULL)

res += countNodesRec(root->left) + 1;

return res;

}

int countNodes()

{

if (root != NULL)

return countNodesRec(root) + 1;

else

return 0;

}

int heightRec(Node \*root)

{

int right = 0, left = 0;

if (root->right != NULL)

right += heightRec(root->right) + 1;

if (root->left != NULL)

left += heightRec(root->left) + 1;

if (right > left)

return right;

else

return left;

}

int height()

{

if (root != NULL)

return heightRec(root) + 1;

else

return 0;

}

void freeTreeRec(Node \*root)

{

if (!root)

return;

freeTreeRec(root->right);

freeTreeRec(root->left);

free(root);

}

void freeTree()

{

freeTreeRec(root);

}

**Fișierul includes.h:**

#pragma once

#define MAX\_STRLEN 20

typedef struct

{

char modelul[MAX\_STRLEN];

char procesorul[MAX\_STRLEN];

float memoria;

float viteza;

float pretul;

} Computer;

typedef struct Node

{

Computer value;

struct Node \*left;

struct Node \*right;

} Node;

void createTree();

void printTree();

Node \*findStruct(char \*value);

void modify(char \*value);

int countNodes();

int height();

void freeTree();

**Fișierul user.c:**

#include <stdio.h>

#include <stdbool.h>

#include "includes.h"

void show\_options()

{

printf("\nMeniu\n");

printf("1. Crearea nodurilor arborelui binar oarecare\n in memoria dinamica si introducerea informatiei\n despre nodurile arborelui de la tastatura in mod interactiv.\n");

printf("2. Afisarea informatiei despre nodurile arborelui la ecran\n");

printf("3. Cautarea nodului in arbore.\n");

printf("4. Modificarea informatiei unui nod din arbore.\n");

printf("5. Determinarea numarului de noduri.\n");

printf("6. Determinarea inaltimii arborelui.\n");

printf("7. Eliberarea memoriei alocate pentru lista.\n");

printf("0. Iesire din program.\n");

}

int handle\_options(int in)

{

switch (in)

{

case 1:

createTree();

break;

case 2:

printTree();

break;

case 3:

char find[MAX\_STRLEN];

printf("modelul elementului cautat: ");

if (scanf("%s", find))

{

Node \*res = findStruct(find);

if (res == NULL)

printf("elementul cautat nu se afla in arbore.\n");

else

printf("elementul este prezent la adresa %p.\n", res);

}

else

printf("valoare invalida.");

break;

case 4:

char choice[MAX\_STRLEN];

printf("modelul elementului schimbat: ");

if (scanf("%s", choice))

modify(choice);

break;

case 5:

printf("arborele dat are %d noduri.\n", countNodes());

break;

case 6:

printf("arborele dat are intaltimea de %d.\n", height());

break;

case 7:

freeTree();

break;

case 0:

return 1;

default:

printf("Valoarea %d nu exista in meniul de optiuni, introduceti o alta valoare.\n", in);

}

return (1);

}

int handle\_input()

{

int in;

do

{

show\_options();

printf("\nCe operatiune doriti sa efectuati?\n");

if (!scanf("%d", &in))

{

printf("\nvaloare invalida.\n");

return (0);

}

else if (!handle\_options(in))

{

return (0);

}

} while (in != 0);

return (1);

}

int main()

{

if (!handle\_input())

{

return (1);

}

return (0);

}

**Execuția programului:**













Nota: modificarea unui element din arbore nu presupune reordonarea lui pentru a pastra proprietatile arborelui binar.











**Concluzii:**

* În informatică, un arbore binar este un arbore în care fiecare nod are cel mult doi succesori (fii). De obicei, succesorii se numesc „nodul stânga” și „nodul dreapta”. Arborii binari sunt folosiți mai ales drept arbori binari de căutare sau și la structurile de date de tip heap.
* Un arbore binar este o mulțime de noduri care îndeplinesc următoarele condiții:

-fiecare nod are 0, 1 sau 2 succesori;

-fiecare nod are un singur predecesor, cu excepția rădăcinii care nu are niciunul;

-succesorii fiecărui nod sunt ordonați (fiul stâng, fiul drept; dacă este unul singur trebuie menționat care).

* Definiția recursivă:

-(Baza:) Arborele fără niciun nod este un arbore binar.

-(Pasul recursiv:) Fie a și b doi arbori binari, iar n un nod. Atunci arborele care îl are pe n ca rădăcină, pe a ca subarbore stâng și pe b ca subarbore drept este un arbore binar.

**Surse:**

* pbinfo.ro, Cursul de programare in C, Tema: Arbori binari: <https://www.pbinfo.ro/articole/25641/arbori-binari>
* wikipedia, "Arbore Binar":

<https://ro.wikipedia.org/wiki/Arbore_binar>