**SD – Seminar 7 – ARBORI**

**17.11.2020**

Exemplu: arbore binar ce memoreaza in noduri caractere (litere mari) – Elt = char.

struct LLin {

Elt tab[0..Max-1]

int ultim

}

***I Reprezentare cu structuri inlantuite***

struct nod {

Elt inf

nod \* stg

nod \* drp

}

ArbBin alias pentru nod \* (adresa nodului radacina sau NULL daca arborele este vid).

***II Reprezentare cu tablouri***

* ArbBin – alias pentru un tablou cu tipul de baza Elt si dimensiune nMax (=?).
* Radacina se memoreaza la pozitia 0.
* Un nod memorat la pozitia i are fiul stang / drept memorat la pozitia 2i+1 / 2i+2. Adresa parintelui unui nod memorat la pozitia i, in afara de radacina este data de expresia (i-1)/2.
* ! Necesitatea unei valori de marcaj pentru fiii lipsa.
* !Conditia de nod frunza.

Exemplu:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | **13** | **14** |
| **A** | **B** | **C** | **$** | **$** | **D** | **$** | **$** | **$** | **$** | **$** | **E** | **F** | **$** | **$** |

nMax = 2^{hMax + 1} -1

**Pr. 1**

Crearea arborelui binar asociat unei secvente de caractere. Secventa este constituita din **litere mari** si caracterele **( , ) $**. Semnificatia acestora este urmatoarea:

* literele reprezinta informatia dintr-un nod;
* **(** defineste un nivel inferior;
* **,** separa subarborele stang de cel drept;
* **)** marcheaza sfarsitul nivelului inferior;
* **$** marcheaza lipsa subarborelui respectiv.

Exemplu:

Expresia asociata: “A(B,C(D(E,F),$))”

***I Implementarea cu structuri inlantuite***

**function** build\_tree ( int n, char sir[0..n-1] )

**begin**

\\ returnam adresa nodului radacina a arborelui creat

**end**

***II Implementarea cu tablouri***

**function** build\_tree ( int n, char sir[0..n-1] )

**begin**

\\ returnam t[0..nMax-1] reprezentarea arborelui asociat expresiei sir[]

**end**

A(B,C(D(E,F),$))

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | **13** | **14** |
| **A** | **B** | **C** | **$** | **$** | **D** | **$** | **$** | **$** | **$** | **$** | **E** | **F** | **$** | **$** |

**Pr. 2**

Scrieti un subprogram care obtine reprezentarea imaginii *in oglinda* a unui arbore binar (pentru cele doua implementari).

|  |  |
| --- | --- |
|  |  |

***I Implementarea cu structuri inlantuite***

**procedure** oglina\_rec ( nod \* t )

**begin**

\\ returnam adresa nodului radacina a arborelui transformat

if t == NULL return

else {

swap(t->stg, t->drp)

oglinda\_rec(t->stg)

oglinda\_rec(t->drp)

}

**end**

**procedure** oglina\_nerec ( nod \* t )

**begin**

\\ returnam adresa nodului radacina a arborelui transformat

\\ TEMA

**end**

***II Implementarea cu tablouri***

**procedure** oglinda ( char t[0..nMax-1] )

**begin**

\\

**end**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | **13** | **14** |
| **C** | **A** | **E** | **D** | **B** | **U** | **$** | **$** | **P** | **X** | **Y** | **$** | **$** | **$** | **$** |
| **C** | **E** | **A** | **$** | **U** | **B** | **D** | **$** | **$** | **$** | **$** | **Y** | **X** | **P** | **$** |

nivel = 0, 1, 2, 3 = k

nr de noduri pe nivel : 1, 2, 4, 8

indicii capetelor nivelului k: stanga = 2^k – 1 , dreapta = 2^{k+1}-2 = 2(2^k-1) = 2 \* stanga

doilak <- 1 ;

while (stanga < nMax) do{

p <- doilak - 1; q <- 2\*p

while (p<q) do {swap(t[p], t[q]); p <- p+1; q <- q-1}

doilak <- 2 \* doilak

}

\\ TEMA

**Pr 3 \\ TEMA**

Afisarea celui de-al k-lea element din traversarea inordine a unui arbore binar.

Exemple:

Parcurgerea inordine:

pt k = 1 🡪

pt k = 4 🡪