**Частное учреждение образования**

**«Гродненский колледж бизнеса и права»**

**Лабораторная работа № 6**

**по дисциплине**

**«Структуры и алгоритмы обработки данных»**

**Тема:** Прошитыебинарные деревья: представление, обход, включение

для учащихся 2 курса специальности

2-40 01 01 «Программное обеспечение информационных технологий»

**ЛАБОРАТОРНАЯ РАБОТА № 6**

Тема: Прошитыебинарные деревья: представление, обход, включение.

Цель: Сформировать навыки построения прошитого бинарного дерева. Научиться делать обход и включение новых элементов.

Задачи: Освоение теоретического материала и выполнение индивидуального задания.

**ЗАДАЧИ**

Условие:

Алгоритм: Предоставлен преподавателю в письменном виде.

Решение:

**program** est6;

**uses** crt;

**type**

ptr = ^elem;

elem = **record**

inf: integer;

left: ptr;

right: ptr;

ltag, rtag: boolean;

**end**;

**var**

pold: ptr;

p, g: integer;

**function** find(root: ptr; key: integer; **var** p, parent: ptr): Boolean;

**begin**

p := root;

**while** p <> nil **do**

**begin**

**if** key = p^.inf **then**

**begin**

find := true;

**exit**;

**end**;

parent := p;

**if** key < p^.inf **then**

p := p ^.left

**else** p := p ^.right;

**end**;

find := false;

**end**;

**function** spusk(p: ptr): ptr;

**var**

y: ptr;

pred: ptr;

**begin**

y := p^.right;

**if** y^.left = nil **then** y^.left := p^.left {1}

**else** {2}

**begin**

**repeat**

pred := y;y := y^.left;

**until** y^.left = nil;

y^.left := p^.left; {3}

pred^.left := y^.right; {4}

y^.right := p^.right; {5}

**end**;

spusk := y;

**end**;

**procedure** del(**var** root: ptr; key: integer );

**var**

p: ptr;

parent: ptr;

y: ptr ;

**begin**

**if not** find(root, key, p, parent) **then** {6}

**begin**

writeln(' oaeiai yeaiaioa iao ');

**exit**;

**end**;

**if** p^.left = nil **then** y := p^.right {7}

**else**

**if** p^.right = nil **then** y := p^.left {8}

**else** y := spusk(p); {9}

**if** p = root **then** root := y {10}

**else** {11}

**if** key < parent^.inf **then**

parent^.left := y

**else** parent^.right := y;

dispose(p); {12}

**end**;

**procedure** add(**var** root: ptr; data: integer);

**begin**

**if** root = nil **then begin**

**new**(root);

root^.inf := data;

root^.left := nil;

root^.right := nil;

root^.ltag := false;

root^.rtag := false;

**end**

**else**

**if** data < root^.inf **then** add(root^.left, data)

**else** add(root^.right, data);

**end**;

**procedure** HeadNode(**var** thead: ptr; p: ptr);

**begin**

**new**(thead);

thead^.left := p;

thead^.right := thead;

**end**;

**procedure** LeftSew(p: ptr);

**begin**

**if** p <> nil **then begin**

LeftSew(p^.left);

**if** p^.left <> nil **then** p^.ltag := true

**else begin**

p^.ltag := false;

p^.left := pold;

**end**;

pold := p;

**if** p^.rtag **then**

LeftSew(p^.right);

**end**;

**end**;

**procedure** RightSew(p: ptr);

**begin**

**if** p <> nil **then**

**begin**

RightSew(p^.right);

**if** p^.right <> nil **then** p^.rtag := true

**else begin**

p^.rtag := false;

p^.right := pold;

**end**;

pold := p;

**if** p^.ltag **then** RightSew(p^.left);

**end**;

**end**;

**procedure** ReadTree(root: ptr);

**begin**

**if** root <> nil **then**

**begin**

write(root^.inf, ' ');

ReadTree(root^.left);

ReadTree(root^.right);

**end**;

**end**;

**var**

data, n, str: integer;

root, head: ptr;

**begin**

root := nil;

head := nil;

pold := nil;

writeln(' 1 - Введите элементы');;

writeln(' 2 - Прошивка дерева');

writeln(' 3 - Вывод дерева');

writeln(' 4 - Удаление элемента');

writeln(' 5 - Вставка элемента');

writeln(' 6 - Выход из прораммы');

//

**repeat**

write('Выберете действие:');readln(str);

**case** str **of**

1:

**begin**

write('Введите кол-во элементов: ');

readln(n);

**repeat**

write('Добавьте элементы: ');

readln(data);

add(root, data);

n := n - 1;

**until** n = 0;

HeadNode(head, root);

**end**;

2:

**begin**

pold := head;

LeftSew(head^.left);

pold := head;

RightSew(head^.left);

writeln('Дерево прошито')

**end**;

4:

**begin**

writeln('Введите элемент, требующий удаления: ');

readln(p);

del(root, p);

**end**;

5:

**begin**

writeln('Вставьте элемент: ');

readln(g);

Add(root, g);

**end**;

3:

**begin**

ReadTree(root);

writeln;

**end**;

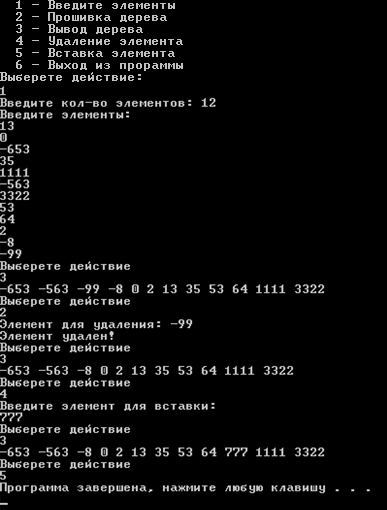
6:

**exit**;

**end**;

**until** str = 6;

**end**.



**ОТВЕТЫ НА КОНТРОЛЬНЫЕ ВОПРОСЫ**

1. Для чего выполняется прошивка бинарных деревьев?  
   Увеличивает эффективность прохождения дерева рекурсивными и нерекурсивными алгоритмами.
2. Какой указатель в дереве называется нитью?  
   Пустой указатель, использованный на отсутствующие поддеревья для хранения в них адресов узлов преемников, которые надо посетить при заданном порядке прохождения бинарного дерева.
3. Какая операция над элементами дерева называется прошивкой?  
   Операция, заменяющая пустые указатели на нити.
4. Дайте определение бинарному дереву, симметрично прошитому справа.  
   Если нити заменяют пустые указатели в узлах с пустыми правыми поддеревьями, при просмотре в симметричном порядке, то бинарное дерево называется симметрично прошитым справа.
5. Почему неэффективно использовать левостороннюю прошивку бинарных деревьев?

Правосторонняя прошивка использует только пустые указатели пустых правых поддеревьев, в то время как в левой каждый пустой указатель содержит нить. Можно сделать вывод, что при левой прошивке снижается производительность.