БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Кафедра программного обеспечения информационных технологий

Факультет ФНиДО

Специальность ПОИТ

Индивидуальная практическая работа № 2.2

«Бинарные деревья»

по дисциплине «Структуры и алгоритмы обработки данных»

Выполнила: Карпеко Н. Г.

Договор № 941 от 20.02.2019 г.

Минск 2020

**Задание:** Ввести 10-15 целых чисел и построить из них бинарное дерево поиска. Выполнить симметричную прошивку бинарного дерева поиска. Обойти его согласно симметричному порядку следования элементов. Реализовать поиск и вставку элементов симметрично прошитого бинарного дерева.

**Листинг программы:**

**program** IPR2\_2;

**uses** crt;

**type**

PNode = ^TNode;

TNode = **Record**

Key: Integer;

LTag, RTag: Boolean;

Left, Right: PNode;

**End**;

//----------- Функция поиска элемента -----------------------

**function** Find(**const** Key: Integer; Node: PNode; **var** Rez: PNode): Boolean;

**begin**

Result := False;

Rez := nil;

**while** (**not** Result) **and** (Node <> nil) **do**

**begin**

Rez := Node;

**if** Node^.Key = Key **then**

Result := True

**else**

**if** Key < Node^.Key **then**

Node := Node^.Left

**else**

Node := Node^.Right;

**end**;

**end**;

//------------- добавление элемента -----------------------

**procedure** Add(**const** Key: Integer; **var** Node: PNode);

**var**

NodeTO, NewNode: PNode;

**begin**

**if not** Find(Key, Node, NodeTO) **then**

**begin**

**New**(NewNode);

NewNode^.Key := Key;

NewNode^.LTag := True;

NewNode^.RTag := True;

NewNode^.Left := nil;

NewNode^.Right := nil;

**if** Node = nil **then**

Node := NewNode

**else**

**if** Key < NodeTO^.Key **then**

NodeTO^.Left := NewNode

**else**

NodeTO^.Right := NewNode

**end**;

**end**;

//----- Процедура вывода дерева на экран -----------

**procedure** Print\_Tree(**var** Node: PNode; level: integer);

**var**

i: integer;

**begin**

**if** Node <> nil **then**

**begin**

Print\_Tree(Node^.Right, level + 1);

**for** i := 0 **to** level **do**

write(' ');

writeln(Node^.Key);

Print\_Tree(Node^.Left, level + 1);

**end**;

**end**;

//--- симметричный обход бинарного дерева поиска ----

**procedure** sim\_print(**var** Node: PNode);

**begin**

**if** Node <> nil **then**

**begin**

write(Node^.Key, ' ');

sim\_print(Node^.Left);

write('(', Node^.Key, ') ');

sim\_print(Node^.Right);

write(Node^.Key, ' ');

**end**

**else**

write('0 ');

**end**;

**var**

Prev, Head: PNode;

**procedure** sim\_print\_rightsew(**const** Node: PNode);

**procedure** rightsew(**const** Curr: PNode);

**begin**

**if** Prev <> nil **then**

**begin**

**if** Prev^.right = nil **then**

**begin**

Prev^.rtag := false;

Prev^.right := Curr;

**if** Prev^.Right <> Head **then**

Write(Prev^.Key, '-', Prev^.Right^.Key, ' ')

**else**

Write(Prev^.Key, '-Head ')

**end**

**else**

Prev^.rtag := true;

**end**;

Prev := Curr;

**end**; //rightsew()------

**begin** //sim\_print\_rightsew()------

**if** Node <> nil **then**

**begin**

sim\_print\_rightsew(Node^.left);

rightsew(Node);

**if** node <> head **then**

sim\_print\_rightsew(Node^.right);

**end**;

**end**;

//--- Процедура обхода после прошитого дерева -------------

**procedure** obhod\_proshiv(Node: PNode);

**begin**

**while** (Node <> Head) **and** (Node <> nil) **do**

**begin**

**while** Node^.Left <> nil **do**

**begin**

Write(Node^.Key, ' ');

Node := Node^.Left;

**end**;

Write(Node^.Key, ' ');

Write('0 ');

Write('(', Node^.Key, ') ');

**while** Node^.RTag = false **do**

**begin**

Node := Node^.Right;

**if** Node = Head **then**

**begin**

write('(Head) ');

**Exit**;

**end**;

write('(', Node^.Key, ') ');

**end**;

Node := Node^.Right;

**end**;

**end**;

**VAR**

Node: PNode;

s: string;

delnode, addnode: integer;

//i, a: integer;

**BEGIN**

Node := nil;

{for i:=1 to 10 do

begin

write('Введите ', i,'-й', ' элемент дерева: '); readln(a); Add(a, Node);

end;}

Add(5, Node);

Add(7, Node);

Add(12, Node);

Add(9, Node);

Add(3, Node);

Add(1, Node);

Add(2, Node);

Add(8, Node);

Add(17, Node);

Add(4, Node);

Print\_Tree(Node, 0); writeln;

**New**(Head);

Head^.Left := Node;

Head^.Right := Head;

Prev := Head;

writeln(' После прошивки бинарного дерева поиска:');

write(' '); sim\_print\_rightsew(Head); writeln; writeln;

Head^.Left := Node;

Head^.Right := Head;

Prev := Head;

writeln(' Симметричный обход прошитого бинарного дерева поиска:');

write(' '); obhod\_proshiv(Node); writeln; writeln;

readln;

**END**.

**Результат работы программы:**

