ФГБОУ ВО «Чувашский государственный университет имени И.Н. Ульянова»

Кафедра вычислительной техники

Лабораторная работа №2.
ДЕРЕВЬЯ. БИНАРНЫЕ ДЕРЕВЬЯ

Вариант 7

Выполнил: студент группы ИВТ-41-20

Галкин Дмитрий Сергеевич

Проверил:

доцент

Павлов Леонид Александрович

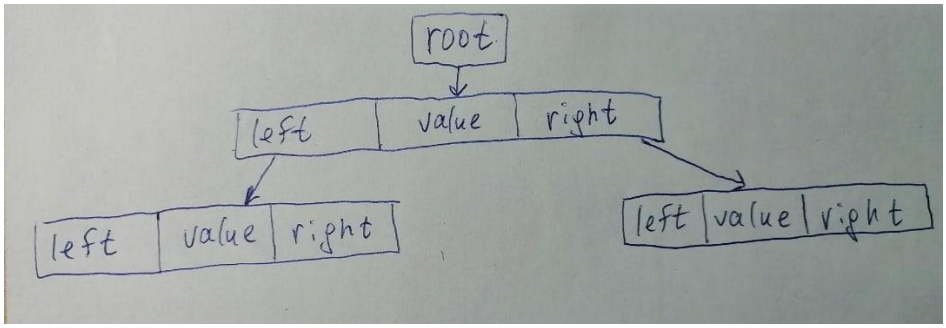
Чебоксары

2021

Цель работы: Ознакомление со способами представления деревьев и методами их прохождения, получение практических навыков программирования задач с использованием деревьев.

Подготовка к работе

1. Реализовать узловое представление бинарного дерева.



2. Разработать алгоритмы и программы, реализующие прямое, обратное, симметричное и горизонтальное прохождения бинарного дерева. Для первых трех прохождений использовать как рекурсивные, так и нерекурсивные варианты. Для реализации горизонтального прохождения требуется очередь.

   - Прямое прохождение

     Нерекурсивное:

```
// Стек S пустой
Push(S, v)
p:=root
while(p != v) {
  visit(p)
  if(p.right != v) then { Push(S, p.right) }
  if(p.left != v) then { p:=p.left }
  else { p:=Pop(S) }
}
```

Рекурсивное:

```
procedure PREORDER(p)
if(p != v) then {
  visit(p)
  PREORDER(p.left)
  PREORDER(p.right)
}
```

- Обратное прохождение

Нерекурсивное:

```
// Стек S пустой
Push(S, v)
p:=root
while(p != v) {
   if(p.right != v) then { Push(S, p.right) }
   if(p.left != v) then { p:=p.left }
   else { p:=Pop(S) }
   visit(p)
}
```

Рекурсивное:

```
procedure PREORDER(p)
if(p != v) then {
   PREORDER(p.left)
   PREORDER(p.right)
   visit(p)
}

return
```

- Симметричное

Рекурсивное:

```
procedure INORDER(p)
if(p != v) then {
   INORDER(p.left)
   visit(p)
   INORDER(p.right)
}
```

Нерекурсивное:

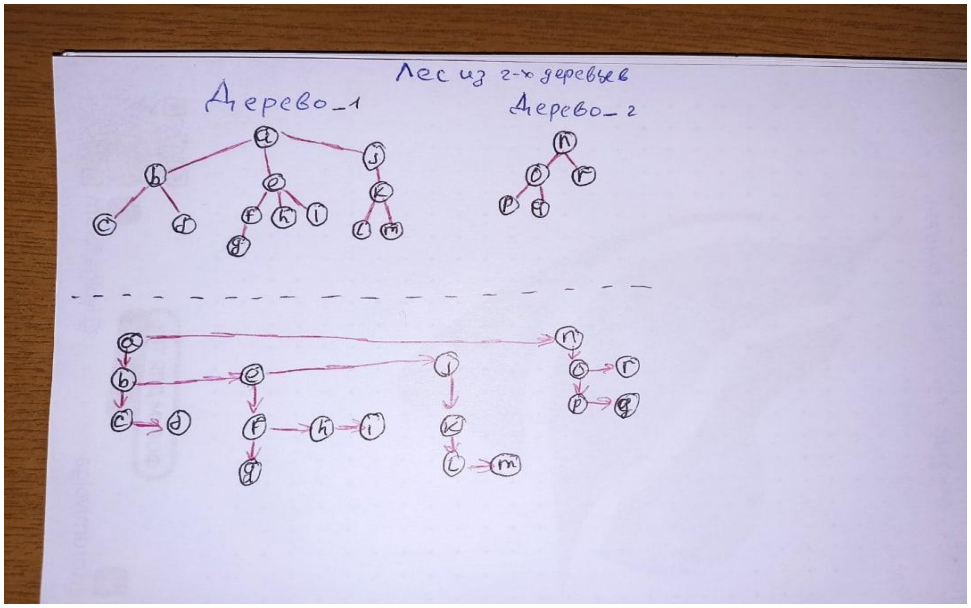
```
// Стек S пустой
p:=root
while(p != v) {
   while(p.left != v) {
      Push(S, p)
      p:=p.left
   }
   while(p.right = v and S != null) {
      p:=Pop(S)
      visit(p)
   }
   p:=p.right
}
```

- Горизонтальное:

```
// Очередь Q пустая
INSERT(Q, root)
while(Q, root) {
   p:=DEL_First(Q)
   visit(p)
   if(p.left != v) then { INSERT(Q, p.left) }
   if(p.right != v) then { INSERT(Q, p.right) }
}
```

3. Разработать алгоритм и программу для прохождения в горизонтальном порядке леса, представленного с помощью естественного соответствия бинарным деревом.

Лес, состоящий из двух деревьев:



Алгоритм прохождения:

```
// Очередь Q пустая
INSERT(Q, root)
while(Q != null) {
   p:=DEL_First(Q)
   while(p != v) {
      visit(p)
      if(p.left != v) then { INSERT(Q, p.left) }
      p:=p.right
   }
}
```

4. Разработать алгоритм и программу ввода и формирования узлового представления бинарного дерева.

```
// Стек S пустой
root:=new Pick
p:=root
root.left:=v
root.right:=v
root.value:=a() // a() - запрашивает значение у пользователя
PUSH(S, root)
while(S != null) {
  if(STACKTOP(S).left = v) then { "Crating first a son" }
  a()
  case {
    // Создание левого сына
    p:=STACKTOP(S)
    if(p.right != v) then { POP(S) }
    p.left:=new Pick
    p:=p.left
    p.left:=v
    p.right=v
    p.value:=a()
    PUSH(S, p)

    // Создание правого сына
    p:=STACKTOP(S)
    if(p.left != v) then { POP(S) }
    p.right:=new Pick
    p:=p.right
    p.left:=v
    p.right:=v
    p.value:=a()
    PUSH(S, p)

    // Не надо создавать сыновей
    POP(S)
  }
}
```

5. Разработать алгоритм и программу вывода узлового представления бинарного дерева с целью визуального контроля структуры дерева.

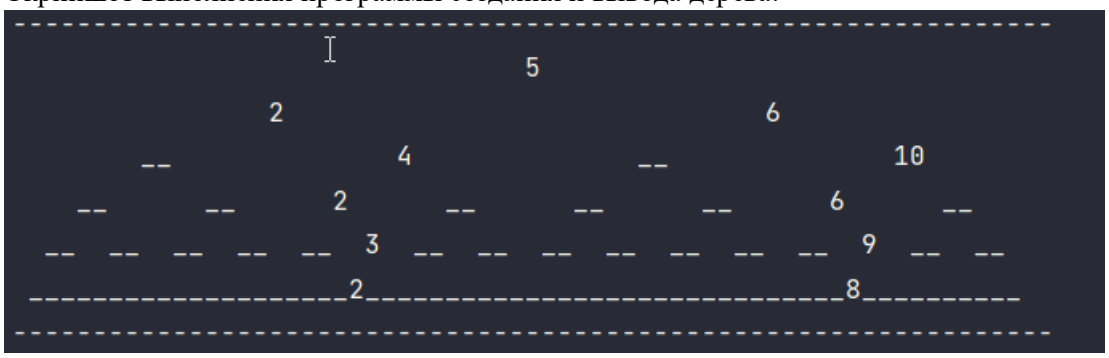
```
// Дерево непустое
// Стек S пустой
p:=root
if(p.left = v) then { print(O) }
else {
   print(p.left.value)
   PUSH(S, p.left)
}
print(p.value)
if(p.right = v) then { print(O) }
else {
   print(p.right.value)
   PUSH(S, p.right)
}
while(S != null) {
   p:=STACKTOP(S)
   if(p.left != v) then {
      POP(S)
      if(p.left = v) then { print(O) }
      else {
         print(p.left.value)
         PUSH(S, p.left)
      }
      print(p.value)
      if(p.right = v) then { print(O) }
      else {
         print(p.right.value)
         PUSH(S, p.right)
      }
   }
   p:=STACKTOP(S)
   if(p.right != v) then {
      POP(S
      if(p.left = v) then { print(O) }
      else {
         print(p.left.value)
         PUSH(S, p.left)
      }
      print(p.value)
      if(p.right = v) then { print(O) }
      else {
         print(p.right.value)
         PUSH(S, p.right)
      }
   }

   if(STACKTOP(S).left = v and STACKTOP(S).right = v) then {
      if(p.left = v) then { print(O) }
```

```
            else { print(p.left.value) }
            print(p.value)
            if(p.right = v) then { print(O) }
            else { print(p.right.value) }
            POP(S)
        }

    }
```

Скриншот выполнения программы создания и вывода дерева:



6. Разработать алгоритмы и программы решения задач в соответствии с заданными вариантами.

Задание № 7: Определить число вхождений заданного элемента в дерево

Метод добавления элементов в дерево:

```
//Стек S пустой
procedure ADD(value)

if(root = null) {
   root:=new Node(value)  // Создаем первый элемент дерева
   return
}

newNode:=new Node(value)
temp:=root
parentTemp:=null

while(true) {
   parentTemp:=temp
   if(value < temp.getValue()) {
      temp:=temp.getLeft()
      if(temp = null) then {
         parentTemp.setLeft(newNode)
         return
      }
   } else {
      temp:=temp.getRight()
      if(temp = null) then {
         parentTemp.setRight(newNode)
```

```
        return
      }
    }
}
```

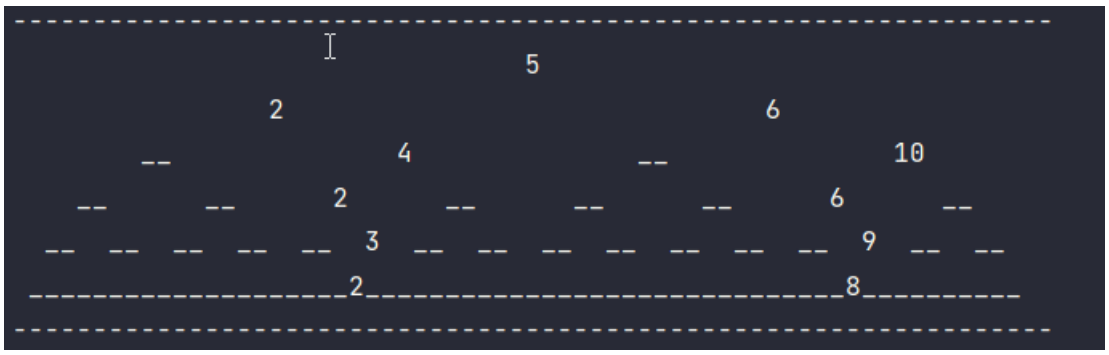
Поиск элемента в дереве

```
procedure FIND(value)

// Стек S не пустой
temp:=root
k:=0
while(temp != null) {
   if(temp.getValue() = value) then { k += 1 }
   if(value < temp.getValue()) then { temp:=temp.getLeft() }
   else { temp:=temp.getRight() }
}

return
```

Результат программы:



Вывод: в ходе данной лабораторной работы я ознакомился со способами представления деревьев и методами их прохождения, а также получил практические навыки программирования задач с использованием деревьев.