

Programming Language

HW04



과목명 : 프로그래밍언어론

담당교수 : 강승식 교수님

학과 : 컴퓨터 공학과

학년 : 3학년

학번 : 20133191

이름 : 고준규

5. Sorting program – read a data from 'input.txt'

```
package main

import (
    "fmt"
    "io"
    "os"
)

func check(e error){
    if e != nil{
        panic(e)
    }
}

func main(){
    dat, err := os.Open("input.txt")
    check(err)

    var perline int
    var num []int
    for{
        _,err := fmt.Fscanf(dat, "%d\n",&perline);

        if err != nil{
            if err == io.EOF{break}
            fmt.Println(err)
            os.Exit(1)
        }
        num = append(num, perline)
    }
    fmt.Println(num)
    for i:= 0; i< len(num); i++){
        for j:= 0; j< i; j++){
            if(num[i] < num[j]){num[i], num[j] = num[j], num[i]}
        }
    }
    fmt.Println(num)

    f, err := os.Create("output.txt")
    if err != nil{
        fmt.Println(err)
        return
    }
    for i := range num{
        fmt.Fprintf(f, "%d\n", num[i])
    }
    f.Close()
}
```

6. 트리 순회 문제

1) channel 사용 안하고 tree traverse 함수 작성

- 데이터값(value)을 left->right 순(소팅 순)으로 출력하시오.

```
package main

import(
    "fmt"
    "math/rand"
)

type Tree struct{
    Left *Tree
    Value int
    Right *Tree
}

func NewTree(k int) *Tree{
    var t *Tree
    for _, v := range rand.Perm(10){
        t = insert(t, (1+v) * k)
    }
    return t
}

func insert(t *Tree, v int) *Tree{
    if t == nil{
        return &Tree{nil, v, nil}
    }
    if v < t.Value{
        t.Left = insert(t.Left, v)
    }else{
        t.Right = insert(t.Right, v)
    }
    return t
}

func Traverse(t *Tree){
    if t == nil{
        return
    }
    Traverse(t.Left)
    a := t.Value
    fmt.Print(a, " ")
    Traverse(t.Right)
}

func main() {

    newTree:=NewTree(1)
    Traverse(newTree)
    fmt.Println()
}
```

- tree 모양을 괄호 모양으로 출력하시오.

예 1: (3 (1 (1) (2)) (8 (5) (13))) --> (value l-tree r-tree)

예 2: (8 (3 ((1 (1) (2))) (5)) (13))

```
package main

import(
    "fmt"
    "math/rand"
)

type Tree struct{
    Left *Tree
    Value int
    Right *Tree
}

func NewTree(k int) *Tree{
    var t *Tree
    for _, v := range rand.Perm(10){
        t = insert(t, (1+v) * k)
    }
    return t
}

func insert(t *Tree, v int) *Tree{
    if t == nil{
        return &Tree{nil, v, nil}
    }
    if v < t.Value{
        t.Left = insert(t.Left, v)
    }else{
        t.Right = insert(t.Right, v)
    }
    return t
}

func (t *Tree) Traverse() string{
    if t == nil{
        return "()"
    }
    s := ""
    if t.Left != nil{
        s += t.Left.Traverse() + ""
    }
    s += fmt.Sprintf("%d", t.Value)
    if t.Right != nil{
        s += "" + t.Right.Traverse()
    }
    return "(" + s + ")"
}
```

```
func main() {  
    newTree:=NewTree(1)  
    fmt.Println(newTree.Traverse())  
}
```

- tree 의 모양을 tab 문자 이용, 좌로 90 도 회전하여 출력하시오.(아래 put-tree.c 참조)

```
package main

import(
    "fmt"
    "math/rand"
)

var N_nodes int= 0

type Node struct{
    data int
    nchild int
    down *Node
    right *Node
}

func put_tree(node *Node, depth int){
    var p *Node

    if node == nil{
        return
    }
    for j:= 0; j< depth; j++){
        fmt.Print("          ")
    }
    fmt.Println(node.data, "(", node.nchild, ")")

    if node.nchild > 0{
        p = node.down
        for i:= node.nchild-1; i>=0; i--{
            put_tree(p, depth+1)
            if p == nil{return}
            p = p.right
        }
    }
}

func del_tree(node *Node){
    if node == nil{return}

    if(node.down != nil){
        del_tree(node.down)
    }
    if(node.right != nil){
        del_tree(node.right)
    }

    N_nodes = N_nodes-1
}

func get_newnode() *Node{
```

```

var node *Node = new(Node)

node.nchild = rand.Intn(4)+1
node.down = nil
node.right = nil

N_nodes = N_nodes+1
return node
}

func build_tree(n int) *Node{

var node *Node = get_newnode()
node.data = n * 1000

if n-1 > 0{
    node.down = build_tree(n-1)
    node.down.data = node.down.data + 1

    n2 := node.down
    for i:=0; i < node.nchild-1; i++){
        n2.right = build_tree(n-1)
        n2.right.data = (n2.right.data + (i+2))

        n2 = n2.right
    }
}else{
    node.nchild = 0
}
return node
}

func main(){
    depth := 4
    var root *Node

    root = build_tree(depth)
    fmt.Println("Total", N_nodes, " nodes are created!")

    put_tree(root,0)
    del_tree(root)
    if(N_nodes != 0){
        fmt.Println("Total %d nodes are remaining!\n", N_nodes)
    }
}

```

2) channel 사용하여 데이터값(value)을 left->right 순(소팅 순)으로 출력하시오.

```
package main

import(
    "fmt"
    "math/rand"
)

type Tree struct{
    Left *Tree
    Value int
    Right *Tree
}

func NewTree(i int) *Tree{
    var t *Tree
    for _,v := range rand.Perm(10){
        t = insert(t, (1+v) * i)
    }
    return t
}

func insert(t *Tree, v int) *Tree{
    if t == nil{
        return &Tree{nil, v, nil}
    }
    if v < t.Value{
        t.Left = insert(t.Left, v)
        return t
    }else{
        t.Right = insert(t.Right, v)
        return t
    }
    return t
}

func channel_tree(a int, c chan int){
    prints := a
    c <- prints
}

func Traverse(t *Tree){
    if t == nil{
        return
    }

    Traverse(t.Left)
    var a int
    c := make(chan int)
    a = t.Value
    go channel_tree(a, c)
    x := <- c
    fmt.Print(" ",x)
    Traverse(t.Right)
}
```



```
}
```

```
func main(){
```

```
    newTree := NewTree(1)
```

```
    Traverse(newTree)
```

```
    fmt.Println()
```

```
}
```

실행화면

5.

```
[1 5 10 2010 0 -1 -100 1 12345 -12345]
[-12345 -100 -1 0 1 1 5 10 2010 12345]
```

6-1-1.

```
1 2 3 4 5 6 7 8 9 10
```

6-1-2.

```
(((((1(2))3(4))5((6)7((8)9)))10)
```

6-1-3.

```
Total 33 nodes are created!
4000 ( 2 )
      3001 ( 4 )
            2001 ( 4 )
                  1001 ( 0 )
                  1002 ( 0 )
                  1003 ( 0 )
                  1004 ( 0 )
            2002 ( 1 )
            2003 ( 1 )
            2004 ( 4 )
                  1001 ( 0 )
                  1002 ( 0 )
                  1003 ( 0 )
                  1004 ( 0 )
      3002 ( 4 )
            2001 ( 2 )
            2002 ( 4 )
            2003 ( 4 )
            2004 ( 2 )
                  1001 ( 0 )
                  1002 ( 0 )
```

6-2

```
1 2 3 4 5 6 7 8 9 10
```