**자료구조 실습 보고서**

**실습 10. 이진탐색트리**

**2016년 5월 27일**

**학번: 201404051**

**이름: 정 용 석**

**1. 실습 문제 소개**

이진 탐색 트리를 구현하고 테스트 하는 실습으로, 사용자가 입력한 데이터의 개수만큼 정수 형 값을 랜덤으로 생성하여 출력한다. 기존에 구현했던 이진 트리를 바탕으로 형식에 맞게 노드를 생성하고 자료를 추가할 수 있는 이진 탐색 트리 삽입 함수를 구현하여, 랜덤 자료들을 차례대로 삽입한다. 이가 완료되면 좌에서 우로 퍼져나가는 트리 형식의 정렬된 자료를 출력하고, 중위 순회 방식으로 또한 자료를 출력해야 한다.

**2. 소스 코드**

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#define MAX\_SIZE 100

//struct node

typedef struct \_node {

int key;

struct \_node \*right;

struct \_node \*left;

}node;

//create a node

node \*makeBT(node \*left, int key, node \*right) {

node \*t;

//Make a new node T

t = (node\*)malloc(sizeof(node));

//Connect left, right as child nodes for T

t->left = left;

t->right = right;

//Item into T's data field

t->key = key;

return t;

}

//insert x into a binary seartch Tree

node\* insertBST(node \*bsT, int x)

{

//search insertion position

node \*p = bsT;

node \*r, \*q;

while (p != NULL) {

if (x == p->key)

return bsT; //same key exists

q = p;

if (x < p->key)

p = p->left;

else

p = p->right;

}

//create new node

r = makeBT(NULL, x, NULL);

//insert node

if (bsT == NULL)

bsT = r;

else if (x < q->key)

q->left = r;

else

q->right = r;

return bsT;

}

//print the tree inorder

void inorder(node \*t){

if (t == NULL)

return;

//order: left->current->right

inorder(t->left);

printf("%d ", t->key);

inorder(t->right);

}

//print the tree from

void printTree(node \*t, int \*sortedSet, int size) {

node \*p = t;

int i;

//search for the sortedSet data in the tree

for (i = 0; i < size; i++) {

while (p != NULL) {

//create spaces depending on the depth

printf(" ");

if (sortedSet[i] == p->key) {

printf("%d\n", p->key);

break;

}

else if (sortedSet[i] < p->key)

p = p->left;

else

p = p->right;

}

p = t;

}

}

//shell sort

int\* shellSort(int \*a, int n) {

int interval;

int i, pos, val;

for (interval = n / 2; interval > 0; interval = interval / 2)

{

for (i = interval; i < n; i++)

{

val = a[i];

for (pos = i; pos >= interval; pos -= interval)

{

if (val < a[pos - interval]) {

a[pos] = a[pos - interval];

}

else

break;

}

a[pos] = val;

}

}

return a;

}

void generateRandomList(int \*set, int size)

{

int i;

srand(time(NULL));

for (i = 0; i < size; i++)

set[i] = rand() % 1000;

}

void printList(int \*set, int size)

{

int i;

for (i = 0; i < size; i++)

printf("%d ", set[i]);

}

int main()

{

int size;

int i;

int \*randSet;

int \*sortedSet;

node \*bst;

//Get an input size from the user

printf("Enter Size = ");

scanf("%d", &size);

//Allocate space for the randSet

randSet = (int\*)malloc(sizeof(int)\*size);

sortedSet = (int\*)malloc(sizeof(int)\*size);

generateRandomList(randSet, size);

printf("Unsorted Data: ");

printList(randSet, size);

//insert randSet data into the Tree

bst = NULL;

for (i = 0; i < size; i++)

bst = insertBST(bst, randSet[i]);

//sorted set

sortedSet = shellSort(randSet, size);

printf("\n\n");

//print the Tree

printTree(bst, sortedSet, size);

//print sorted data

printf("\n\nSorted data: \n\n");

inorder(bst);

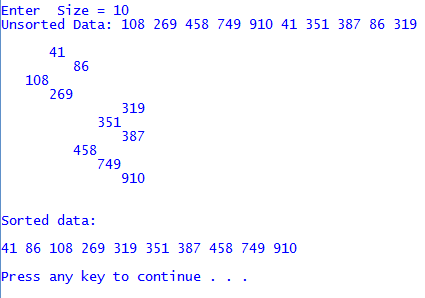
printf("\n\n");

return 0;

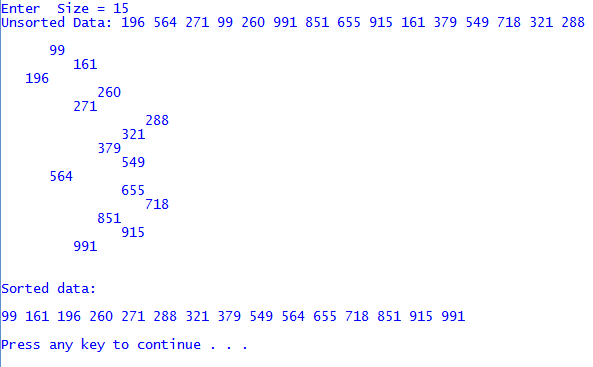
}

**3. 테스트 결과**

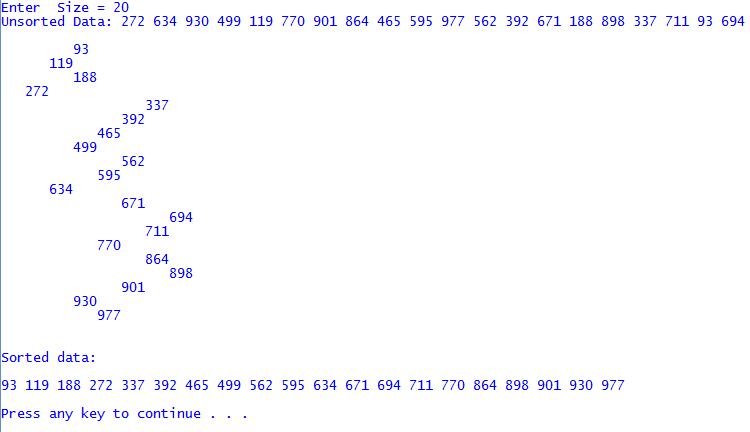
<size: 10>



<size: 15>



<size: 20>



**4. 작성자 코멘트**

난수를 생성하고 출력하는 과정이나 이진 탐색 트리 함수의 구현 부분은 저번 실습과 주어진 알고리즘을 사용하여 쉽게 구현할 수 있었다. 막상 시간이 걸린 부분이 트리 모양으로 자료를 출력하는 문제 였는데, 이는 아마도 다양한 방법이 있었지만 가장 효율적이고 간단한 방법을 찾고자 하는 마음에 더욱이 오래 걸렸던 것 같다.

저번 실습에서 사용한 중위 순회 방식으로 정렬된 자료를 출력하는 것 까지는 문제가 없었다. 하지만 이를 트리 형태로 출력하기 위해서는 각 노드의 depth를 구하고 이에 알맞게 출력 형태를 만들어야 하는데, 일단 저번 실습 간 구현했던 중위 순회 함수를 사용할 경우에는 재귀 함수 형식이기에 이를 구현하기 어렵다는 판단을 했다. 실제로 재귀 함수 안에서 어떻게든 출력을 해보려고 노력했지만 여러 번 다른 문제를 야기하여 실패했다. 일단 기본적인 문제는 depth를 구하기 위한 변수를 설정할 수가 없었다. 전역 변수로 한다고 해도 순회 방식 상 각 노드의 depth를 구하고 저장하기 위해서는 node 자체에 변수를 넣어놓고, 노드를 추가할 때마다 각각의 depth를 구하여 동시에 저장하는 방식으로 하여야 하는데, 이는 모든 함수를 조금씩 바꾸고 안에 내용 또한 조금씩 수정해야 되는 상황이 일어났다. 실제로 가능한 방식이었지만, 너무 복잡하고 난잡하다고 판단되어 이 또한 포기했다.

가장 간단하면서도 보기 편한 방법은 중위 순회 순서에 있었다. 중위 순회 순서 자체가 정렬되지 않은 자료를 트리에 삽입하면서 정렬하는 구도 이기에 트리를 출력하는 순서 또한 정렬된 자료의 순서와 동일하다. 따라서 정렬된 자료의 순서대로 트리 내에서 노드를 탐색하고, 탐색하는 과정에서 알맞은 띄어쓰기 단위를 가지게 하여 출력한다면 원하는 결과를 볼 수 있다. 여기서 나는 쉘 정렬을 이용하여 정렬되지 않는 난수의 집합을 정렬하여 새로 저장하였다. 그리고 이 저장된 배열을 이용하여 위에서 말한 방식으로 트리를 출력하였고, 성공적이었다.

생각보다 트리 형태를 출력하는 방법을 생각해 내는 것이 쉽지 않았다. 사실 상 지금 구현 한 방식도 딱히 마음에 들지 않지만 뭔가 도무지 기발하고 쉬운 방법이 생각나지가 않았다. 하지만 현재 구현 된 방식 또한 마음에 들기에 그렇게 신경 쓰진 않지만 다른 학생들의 방식 또한 볼 수 있었으면 좋겠다.