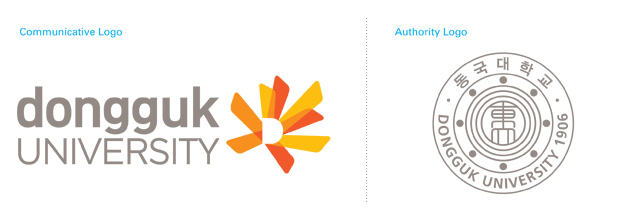
**Report**

**자료구조**



|  |
| --- |
| **Report번호 :** **O번** |
| **강좌 번호 :**   **SCS???** |
| **담 당 : 박 성 철 교수님** |
| **학 과 : 경제학과** |
| **학 번 : 2012112850** |
| **성 명 : 김 치 현** |
| **제 출 일 : 2017년 월 일** |



**1. 실습 과제**

|  |
| --- |
| /\*  작성자: 김치현  내  용: binary tree  파  일: binary\_tree.cpp\*/    #include <stdio.h>  #include <conio.h>  #include <ctype.h>  #include <stdlib.h>  #include <string.h>  #include "binary\_tree.h"  int main ()  {  char c;  int n;  tree\_pointer t;  t = build\_simple\_tree(); //create tree  printf("\*\*\*\*\*\*\*\*\*\*\*\*\* Command \*\*\*\*\*\*\*\*\*\*\*\*\n");  printf("C: Count tree, A: Add tree data \n");  printf("H: Height of tree, S: Show preorder \n");  printf("F: Free tree, Q: Quit \n");  printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");    while (1) {  printf("\nCommand> ");  c = getch();  putch(c);  c = toupper(c);  switch (c) {  case 'C':  n = bt\_count(t);  printf("\n Total number of node = %d \n", n);  break;    case 'A':  n = bt\_add(t);  printf("\n Sum of tree data = %d \n", n);  break;    case 'H':  n = bt\_height(t);  printf("\n Height of tree = %d \n", n);  break;    case 'S':  printf("\n");  bt\_show\_preorder(t);  printf("\n");  break;    case 'F':  printf("\n");  bt\_free(t);  printf("\n");  break;    case 'Q':  printf("\n");  exit(1);  default: break;  } //switch  } //while  } //main  tree\_pointer build\_simple\_tree() // create binary tree having nine node  {  tree\_pointer n1, n2, n3, n4, n5, n6, n7, n8, n9;    n1= (tree\_pointer)malloc(sizeof(tree\_pointer)); n2= (tree\_pointer)malloc(sizeof(tree\_pointer)); n3= (tree\_pointer)malloc(sizeof(tree\_pointer));  n4= (tree\_pointer)malloc(sizeof(tree\_pointer)); n5= (tree\_pointer)malloc(sizeof(tree\_pointer)); n6= (tree\_pointer)malloc(sizeof(tree\_pointer));  n7= (tree\_pointer)malloc(sizeof(tree\_pointer)); n8= (tree\_pointer)malloc(sizeof(tree\_pointer)); n9= (tree\_pointer)malloc(sizeof(tree\_pointer));    n1->data = 10; n1->left = n2; n1->right = n3;  n2->data = 20; n2->left = n4; n2->right = n5;  n3->data = 30; n3->left = n6; n3->right = n7;  n4->data = 40; n4->left = n8; n4->right = n9;  n5->data = 50; n5->left = NULL; n5->right = NULL;  n6->data = 60; n6->left = NULL; n6->right = NULL;  n7->data = 70; n7->left = NULL; n7->right = NULL;  n8->data = 80; n8->left = NULL; n8->right = NULL;  n9->data = 90; n9->left = NULL; n9->right = NULL;    return n1;  }  int bt\_count(tree\_pointer ptr)// calculate sum tree  {  int count =0;    if(ptr != NULL)  count = 1 + bt\_count(ptr->left) + bt\_count(ptr->right);    return count;    }  int bt\_add(tree\_pointer ptr) // calculate sum node  {  int sum = 0;    if(ptr){  sum += ptr->data + bt\_add(ptr->left) + bt\_add(ptr->right);  }    return sum;  }  int bt\_height(tree\_pointer ptr) // calculate height tree  {  int height = 0;    if (ptr != NULL)  height = 1 + max( bt\_height(ptr->left), bt\_height(ptr->right) );    return height;  }  void bt\_show\_preorder(tree\_pointer ptr) // print tree to preorder  {  if(ptr){  printf("%d ", ptr->data);  bt\_show\_preorder(ptr->left);  bt\_show\_preorder(ptr->right);  }    }  void bt\_free(tree\_pointer ptr) // memory free all tree  {  if( ptr!= NULL){  printf("free node with item %d ... \n", ptr->data);  bt\_free(ptr->left);  bt\_free(ptr->right);  free(ptr);  }  }  int max(int i, int j) //for height tree calculate  {  } |

**2. 캡쳐 화면**

|  |
| --- |
|  |

**3. 결과 부연 설명**

|  |
| --- |
| 이진 트리를 구현했습니다. |

**4. 소감**

|  |
| --- |
| 연결리스트, 큐와 함께 이진 트리 역시 어려운 놈입니다. 이진트리의 노드 계산, 합 계산 등을 혼자서 구현하라고 하면 절대로 못할 것 같습니다. 주어져 있는 함수를 가져다 쓰는 건 쉽지만, 그 함수가 왜 이렇게 만들어졌는지 이해하는 과정이 더 중요할 것 같습니다.  이진트리의 삭제, 삽입 구문도 빨리 연습하여 확실하게 이해할 수 있도록 하겠습니다. |