|  |  |  |  |
| --- | --- | --- | --- |
| 2022/2 『자료구조』실습 보고서 | | | |
| 제목 | 8장 실습( O ) 과제( ) | 제출일자 | 2022.  11 .    16 . |
| 학번 | 201911608 | 이름 | 김 지환 |

|  |
| --- |
| 1. 프로그램 8.4 - 반복적인 중위 순회  #include <stdio.h>  #include <stdlib.h>  #include <memory.h>  typedef struct TreeNode {  int data;  struct TreeNode\* left, \* right;  } TreeNode;  #define SIZE 100  int top = -1;  TreeNode\* stack[SIZE];  void push(TreeNode\* p) {  if (top < SIZE - 1) stack[++top] = p;  }  TreeNode\* pop() {  TreeNode\* p = NULL;  if (top >= 0) p = stack[top--];  return p;  }  void inorder\_iter(TreeNode\* root) {  while (1) {  for (; root; root = root->left) push(root);  root = pop();  if (!root) break;  printf("[%d] ", root->data);  root = root->right;  }  }  TreeNode n1 = { 1, NULL, NULL };  TreeNode n2 = { 4, &n1, NULL };  TreeNode n3 = { 16, NULL, NULL };  TreeNode n4 = { 25, NULL, NULL };  TreeNode n5 = { 20, &n3, &n4 };  TreeNode n6 = { 15, &n2, &n5 };  TreeNode\* root = &n6;  int main(void) {  printf("중위 순회=");  inorder\_iter(root);  printf("\n");  return 0;  }  실행결과    -> root = 15    중위 순회 - 왼쪽 노드에 있는 값들부터 출력 |
| 2. 프로그램 8.5 - 레벨 순회  #include <stdio.h>  #include <stdlib.h>  #include <memory.h>  typedef struct TreeNode {  int data;  struct TreeNode\* left, \* right;  } TreeNode;  #define MAX\_QUEUE\_SIZE 100  typedef TreeNode\* element;  typedef struct {  element data[MAX\_QUEUE\_SIZE];  int front, rear;  } QueueType;  void error(char\* message) {  fprintf(stderr, "%s\n", message);  exit(1);  }  void init\_queue(QueueType\* q) {  q->front = q->rear = 0;  }  int is\_empty(QueueType\* q) {  return (q->front == q->rear);  }  int is\_full(QueueType\* q) {  return ((q->rear + 1) % MAX\_QUEUE\_SIZE == q->front);  }  void enqueue(QueueType\* q, element item) {  if (is\_full(q)) error("큐가 포화상태입니다");  q->rear = (q->rear + 1) % MAX\_QUEUE\_SIZE;  q->data[q->rear] = item;  }  element dequeue(QueueType\* q) {  if (is\_empty(q)) error("큐가 공백상태입니다");  q->front = (q->front + 1) % MAX\_QUEUE\_SIZE;  return q->data[q->front];  }  void level\_order(TreeNode\* ptr) {  QueueType q;  init\_queue(&q);  if (ptr == NULL) return;  enqueue(&q, ptr);  while (!is\_empty(&q)) {  ptr = dequeue(&q);  printf(" [%d] ", ptr->data);  if (ptr->left) enqueue(&q, ptr->left);  if (ptr->right) enqueue(&q, ptr->right);  }  }  TreeNode n1 = { 1, NULL, NULL };  TreeNode n2 = { 4, &n1, NULL };  TreeNode n3 = { 16, NULL, NULL };  TreeNode n4 = { 25, NULL, NULL };  TreeNode n5 = { 20, &n3, &n4 };  TreeNode n6 = { 15, &n2, &n5 };  TreeNode\* root = &n6;  int main(void)  {  printf("레벨 순회=");  level\_order(root);  printf("\n");  return 0;  }  실행결과    -> root = 15    -> bfs 알고리즘을 사용  -> queue = 15 -> 15의 left랑 right가 존재할 시 left, right 순서로 queue에 삽입을 반복  -> 15 -> 4 (15.l) -> 20 (15.r) -> 1 -> (4.l) -> 16 (20.l) -> 25(20.r)  -> 4.r과 1.lr, 1.r, 16.l, 16.r, 25.l, 25.r 은 null이므로 무시 |
| 프로그램 8.6 - 수식 트리 프로그램  #include <stdio.h>  #include <stdlib.h>  typedef struct TreeNode {  int data;  struct TreeNode\* left, \* right;  } TreeNode;  TreeNode n1 = { 1, NULL, NULL };  TreeNode n2 = { 4, NULL, NULL };  TreeNode n3 = { '\*', &n1, &n2 };  TreeNode n4 = { 16, NULL, NULL };  TreeNode n5 = { 25, NULL, NULL };  TreeNode n6 = { '+', &n4, &n5 };  TreeNode n7 = { '+', &n3, &n6 };  TreeNode\* exp = &n7;  int evaluate(TreeNode\* root) {  if (root == NULL) return 0;  if (root->left == NULL && root->right == NULL) return root->data;  else {  int op1 = evaluate(root->left);  int op2 = evaluate(root->right);  printf("%d %c %d을 계산합니다.\n", op1, root->data, op2);  switch (root->data) {  case '+':  return op1 + op2;  case '-':  return op1 - op2;  case '\*':  return op1 \* op2;  case '/':  return op1 / op2;  }  }  return 0;  }  int main(void) {  printf("수식의 값은 %d입니다. \n", evaluate(exp));  return 0;  }  실행결과    ->    위와 같은 트리에서 연산자 끼리의 연산이다.  즉 h = 4 일 때 모든 노드가 null 이므로 0을 반환하게 된다.  h = 3 일 때 모든 노드가 상수이므로 상수 그대로 반환 해준다.  h = 2 일 때, 연산자를 가진 노드들은 반환 된 op1과 op2 값을 해당 연산자에 맞게 연산한 값 반환  h = 1 일 때, h = 2 일 때, 반환 된 값을 연산 후 반환한다. |
| 4. 프로그램 8.7 - 디렉토리 용량 계산 프로그램  #include <stdio.h>  #include <stdlib.h>  typedef struct TreeNode {  int data;  struct TreeNode\* left, \* right;  } TreeNode;  int calc\_dir\_size(TreeNode\* root) {  int left\_size, right\_size;  if (root == NULL) return 0;  left\_size = calc\_dir\_size(root->left);  right\_size = calc\_dir\_size(root->right);  return (root->data + left\_size + right\_size);  }  int main(void) {  TreeNode n4 = { 500, NULL, NULL };  TreeNode n5 = { 200, NULL, NULL };  TreeNode n3 = { 100, &n4, &n5 };  TreeNode n2 = { 50, NULL, NULL };  TreeNode n1 = { 0, &n2, &n3 };  printf("디렉토리의 크기=%d\n", calc\_dir\_size(&n1));  }  실행결과    -> 모든 노드를 탐색 후 더한다. |
| 5. Quiz(290쪽) 01 문제 프로그램  01. get\_node\_count(), get\_height(), get\_leaf\_count() 함수들을 테스트하는 전체 프로그램을 작성해보자.  #include <stdio.h>  #include <stdlib.h>  #include <memory.h>  #define max(a,b) (((a) > (b)) ? (a) : (b))  typedef struct TreeNode {  int data;  struct TreeNode\* left, \* right;  } TreeNode;  TreeNode n1 = { 1, NULL, NULL };  TreeNode n2 = { 4, &n1, NULL };  TreeNode n3 = { 16, NULL, NULL };  TreeNode n4 = { 25, NULL, NULL };  TreeNode n5 = { 20, &n3, &n4 };  TreeNode n6 = { 15, &n2, &n5 };  TreeNode\* root = &n6;  int get\_node\_count(TreeNode\* node) {  int count = 0;  if (node != NULL) {  int left = get\_node\_count(node->left);  int right = get\_node\_count(node->right);  count = 1 + left + right;  }  return count;  }  int get\_leaf\_count(TreeNode\* node) {  int count = 0;  if (node != NULL) {  if (node->left == NULL && node->right == NULL) return 1;  else {  int left = get\_leaf\_count(node->left);  int right = get\_leaf\_count(node->right);  count = left + right;  }  }  return count;  }  int get\_height(TreeNode\* node) {  int height = 0;  if (node != NULL) {  int left = get\_height(node->left);  int right = get\_height(node->right);  height = 1 + max(left, right);  }  return height;  }  int main(void) {  printf("노드 갯수 : %d\n", get\_node\_count(root));  printf("단말 노드 갯수 : %d\n", get\_leaf\_count(root));  printf("노드 높이 : %d\n", get\_height(root));  return 0;  } |