**알고리즘YD 20212127 송하성**

**11주차 과제**

**[예제 1] 이진 검색트리 구축 : 학생 ID와 SCORE를 입력받고 검색하기**

|  |
| --- |
| C |
| #define \_CRT\_SECURE\_NO\_WARNINGS  #include <stdio.h>  #include <stdlib.h>  //p. 261,260코드 복붙(key값 = 학생id)기준  struct tree {  int key;  float score;  struct key\* left;  struct key\* right;  };  int Insert\_node(struct tree\* root, int key, float score);  struct tree\* search(struct tree\* root, int key);  void inorder(struct tree\* root); // 중위  int main() {  int key; float score;  struct tree\* root = NULL, \* find;  printf("학생 ID와 Score를 입력하시오 : ");  scanf("%d %f", &key, &score);  while (key) {  if (root) {  Insert\_node(root, key, score);  }  else {  root = malloc(sizeof(struct tree));  root->left = root->right = NULL;  root->key = key;  root->score = score;  }  printf("학생 ID와 Score를 입력하시오 : ");  scanf("%d %f", &key, &score);  }  printf("\n검색할 ID를 입력하시오 : \n");  scanf("%d", &key);  find = search(root, key);  if (find) {  printf("%d %f\n", find->key, find->score);  }  else {  printf("404 not found\n");  }  printf("\n----< 성적 나열 >----\n");  inorder(root);  }  int Insert\_node(struct tree\* root, int key, float score) {  // root 값은 이미 설정 되었음  struct tree\* tptr = root, \* before = NULL, \* node;  while (tptr) {  before = tptr;  if (key < tptr->key)  tptr = tptr->left;  else if (key > tptr->key)  tptr = tptr->right;  else  return 0;  }  tptr = malloc(sizeof(struct tree));  tptr->left = tptr->right = NULL;  tptr->key = key; tptr->score = score;  if (key < before->key)  before->left = tptr;  else  before->right = tptr;  return 1;  }  struct tree\* search(struct tree\* root, int key) {  while (root) {  if (root->key == key)  return root;  else if (root->key > key)  root = root->left;  else  root = root->right;  }  return NULL;  }  void inorder(struct tree\* root) {  if (root) {  inorder(root->left);  printf("%d %f\n", root->key, root->score);  inorder(root->right);  }  } |

텍스트, 스크린샷, 블랙이(가) 표시된 사진

자동 생성된 설명

**[예제 2]**

|  |
| --- |
| C |
| #define \_CRT\_SECURE\_NO\_WARNINGS  #include <stdio.h>  #include <stdlib.h>  //p.260, 261변형 => 문자열 비교  struct tree {  char\* word;  struct key\* left;  struct key\* right;  };  int Insert\_node(struct tree\* root, char \* word);  struct tree\* search(struct tree\* root, char\*word);  void inorder(struct tree\* root); // 중위  int main() {  int word[15];  struct tree\* root = NULL, \* find;  printf("검색 트리에 저장할 단어를 입력하세요.\n 입력의 끝에는 quit를 입력하세요.\n");  printf("단어 입력 : ");  scanf("%s", word);  while (strcmp(word, "quit")) {  if (root) {  Insert\_node(root, word);  }  else {  root = malloc(sizeof(struct tree));  root->left = root->right = NULL;  root->word = malloc( (strlen(word)+1)\*sizeof(char) );  strcpy(root->word, word);  }  printf("단어 입력 : ");  scanf("%s", word);  }  printf("\n---트리 안의 단어들(사전식 순서)---\n");  inorder(root);  printf("\n검색할 단어를 입력 하시오 : ");  scanf("%s", word);  find = search(root, word);  if (find) {  printf("%s\n", find->word);  }  else {  printf("%s is not exist. \n", root->word);  }  }  int Insert\_node(struct tree\* root, char\* word) {  // root 값은 이미 설정 되었음  struct tree\* tptr = root, \* before = NULL, \* node;  int cmp;  while (tptr) {  before = tptr;  cmp = strcmp(tptr->word, word);  if (cmp == 1)  tptr = tptr->left;  else if (cmp == -1)  tptr = tptr->right;  else  return 0;  }  tptr = malloc(sizeof(struct tree));  tptr->left = tptr->right = NULL;  tptr->word = malloc((strlen(word) + 1) \* sizeof(char));  strcpy(tptr->word, word);  if (cmp == 1)  before->left = tptr;  else  before->right = tptr;  return 1;  }  struct tree\* search(struct tree\* root, char\* word) {  int cmp;  while (root) {  cmp = strcmp(root->word, word);  if (!cmp)  return root;  else if (cmp == 1)  root = root->left;  else  root = root->right;  }  return NULL;  }  void inorder(struct tree\* root) {  if (root) {  inorder(root->left);  printf("%s\n", root->word);  inorder(root->right);  }  } |

텍스트, 스크린샷, 블랙, 디자인이(가) 표시된 사진

자동 생성된 설명

**[예제3]** 위의 프로그램을 node 구조에 count를 첨가시켜 단어를 계속 읽으면서 단어의 출현빈도수를 구하기 위한 프로그램으로 수정하시오. 중위 운행한 결과 알파벳 순서로 단어와 그의 빈도수를 출력하도록 한다. 프로그램을 완성하고 실행하면서 준 입력에 대하여 프로그램 내부에서 만들어지는 이진검색트리의 모양을 그리고 실행 결과를 적으면서 프로그램을 이해하시오.

|  |
| --- |
| C |
| #define \_CRT\_SECURE\_NO\_WARNINGS  #include <stdio.h>  #include <stdlib.h>  struct tree {  char\* word;  int count;  struct key\* left;  struct key\* right;  };  int Insert\_node(struct tree\* root, char\* word);  struct tree\* search(struct tree\* root, char\* word);  void inorder(struct tree\* root);  int main() {  int word[15];  struct tree\* root = NULL, \* find;  printf("단어를 입력받고 단어의 출현빈도수를 구하기 위한 프로그램");  printf("\n(종료시 quit를 입력하시오)\n");  printf("\n단어를 입력하시오 : ");  scanf("%s", word);  while (strcmp(word, "quit")) {  if (root) {  Insert\_node(root, word);  }  else {  root = malloc(sizeof(struct tree));  root->left = root->right = NULL; root->count = 1;  root->word = malloc((strlen(word) + 1) \* sizeof(char));  strcpy(root->word, word);  }  printf("단어를 입력하시오 : ");  scanf("%s", word);  }  printf("\n----<단어 / 출현빈도수>----\n");  inorder(root);  printf("\n검색할 단어를 입력하시오 : ");  scanf("%s", word);  find = search(root, word);  if (find) {  printf("%s %d\n", find->word, find->count);  }  else {  printf("404 not found\n");  }  }  int Insert\_node(struct tree\* root, char\* word) {  // root 값은 이미 설정 되었음  struct tree\* tptr = root, \* before = NULL, \* node;  int cmp;  while (tptr) {  before = tptr;  cmp = strcmp(tptr->word, word);  if (cmp == 1)  tptr = tptr->left;  else if (cmp == -1)  tptr = tptr->right;  else {  tptr->count++;  return 0;  }  }  tptr = malloc(sizeof(struct tree));  tptr->left = tptr->right = NULL;  tptr->count = 1;  tptr->word = malloc((strlen(word) + 1) \* sizeof(char));  strcpy(tptr->word, word);  if (cmp == 1)  before->left = tptr;  else  before->right = tptr;  return 1;  }  struct tree\* search(struct tree\* root, char\* word) {  int cmp;  while (root) {  cmp = strcmp(root->word, word);  if (!cmp)  return root;  else if (cmp == 1)  root = root->left;  else  root = root->right;  }  return NULL;  }  void inorder(struct tree\* root) {  if (root) {  inorder(root->left);  printf("%s %d\n", root->word, root->count);  inorder(root->right);  }  } |

텍스트, 스크린샷, 블랙, 흑백이(가) 표시된 사진

자동 생성된 설명

**[예제4]** 학생번호(정수형)와 평균평점(실수형)을 자료로 가지는 이진검색트리의 노드 구조(SNODETYPE)를 정의하고 준비한 학생정보를 입력받아 학생번호를 기준으로 이진검색트리를 구성한다. 이렇게 구축된 데이터를 중위운행에 의하여 보여주고 특정학생의 번호를 입력하면 그 학생의 평균평점을 보여 주도록 한다. 평균평점이 4.0 이상인 학생을 보여 주도록 한다.

|  |
| --- |
| C |
| #define \_CRT\_SECURE\_NO\_WARNINGS  #include <stdio.h>  #include <stdlib.h>  // 학생 학번 점수 입력 + 4.0이상 학생 출력 함수 구현 + file 처리  struct tree {  int key;  float score;  struct key\* left;  struct key\* right;  };  int Insert\_node(struct tree\* root, int key, float score);  struct tree\* search(struct tree\* root, int key);  void inorder(struct tree\* root); // 중위  int main(int arg, char\* args[]) {  FILE\* file = fopen(args[1], "r");  int key; float score;  struct tree\* root = NULL, \* find;  fscanf(file, "%d %f", &key, &score);  while (key) {  if (root) {  Insert\_node(root, key, score);  }  else {  root = malloc(sizeof(struct tree));  root->left = root->right = NULL;  root->key = key;  root->score = score;  }  fscanf(file, "%d %f", &key, &score);  }  printf("---------< 성적 조회 >---------\n");  inorder(root);  printf("\n성적을 조회할 id를 입력하시오 : ");  scanf("%d", &key);  find = search(root, key);  if (find) {  printf("%d %f\n", find->key, find->score);  }  else {  printf("404 not found\n");  }  }  int Insert\_node(struct tree\* root, int key, float score) {  // root 값은 이미 설정 되었음  struct tree\* tptr = root, \* before = NULL, \* node;  while (tptr) {  before = tptr;  if (key < tptr->key)  tptr = tptr->left;  else if (key > tptr->key)  tptr = tptr->right;  else  return 0;  }  tptr = malloc(sizeof(struct tree));  tptr->left = tptr->right = NULL;  tptr->key = key; tptr->score = score;  if (key < before->key)  before->left = tptr;  else  before->right = tptr;  return 1;  }  struct tree\* search(struct tree\* root, int key) {  while (root) {  if (root->key == key)  return root;  else if (root->key > key)  root = root->left;  else  root = root->right;  }  return NULL;  }  void inorder(struct tree\* root) {  if (root) {  inorder(root->left);  printf("%d %f\n", root->key, root->score);  inorder(root->right);  }  } |

텍스트, 스크린샷, 블랙이(가) 표시된 사진

자동 생성된 설명