**알고리즘YD 20212127 송하성**

**12주차 과제**

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

|  |
| --- |
| C |
| #define \_CRT\_SECURE\_NO\_WARNINGS  #include <stdio.h>  #include <stdlib.h>  #include<string.h>  typedef struct node {  struct node\* left;  int eid;  int syear;  char grade;  struct node\* right;  } ENODE;  int insert\_node(ENODE\* root, int num, int year, char score);  ENODE\* search(ENODE\* root, int key);  void inorder(ENODE\* root);  void year\_search(ENODE\* root, int year);  void print\_below\_A(ENODE\* root);  void main(int argc, char\* argv[]) {  ENODE\* tree = NULL, \* ptr;  int id, year;  char score;  FILE\* input;  if ((input = fopen(argv[1], "r")) == NULL) {  printf("Invalid Open input file...\n");  exit(1);  }  while (fscanf(input, "%d %d %c", &id, &year, &score) != EOF) {  if (!tree) {  tree = (ENODE\*)malloc(sizeof(ENODE));  tree->eid = id;  tree->syear = year;  tree->grade = score;  tree->left = tree->right = NULL;  }  else  insert\_node(tree, id, year, score);  }  fclose(input);  printf("\n트리에 구축된 사원 정보 : \n");  inorder(tree);  printf("\n 입사년도 : ");  scanf("%d", &year);  year\_search(tree, year);  printf("\n 사원 번호 : ");  scanf("%d", &id);  ptr = search(tree, id);  if (ptr)  printf("%d번 사원의 정보 : %d\t%c \n", ptr->eid, ptr->syear, ptr->grade);  else  printf("%d번 사원에 대한 정보를 찾을 수 없습니다.\n", id);  printf("\n인사 등급이 A가 아닌 사원 번호 : \n");  print\_below\_A(tree);  }  ENODE\* search(ENODE\* root, int key) {  ENODE\* tptr = root;  while (tptr) {  if (key < tptr->eid)  tptr = tptr->left;  else if (key > tptr->eid)  tptr = tptr->right;  else  return tptr;  }  return NULL;  }  int insert\_node(ENODE\* root, int id, int year, char score) {  ENODE\* tptr = root, \* before = NULL;  while (tptr) {  if (id < tptr->eid) {  before = tptr;  tptr = tptr->left;  }  else if (id > tptr->eid) {  before = tptr;  tptr = tptr->right;  }  else  return 0;  }  tptr = (ENODE\*)malloc(sizeof(ENODE));  tptr->eid = id;  tptr->syear = year;  tptr->grade = score;  tptr->left = tptr->right = NULL;  if (id < before->eid)  before->left = tptr;  else  before->right = tptr;  return 1;  }  void inorder(ENODE\* ptr) {  if (ptr) {  inorder(ptr->left);  printf("%d\t%d\t%c\n", ptr->eid, ptr->syear, ptr->grade);  inorder(ptr->right);  }  }  void year\_search(ENODE\* ptr, int year) {  if (ptr) {  year\_search(ptr->left, year);  if ((ptr->syear) == year)  printf("%d\t%c\n", ptr->eid, ptr->grade);  year\_search(ptr->right, year);  }  }  void print\_below\_A(ENODE\* ptr) {  if (ptr) {  print\_below\_A(ptr->left);  if (ptr->grade != 'A')  printf("%d\n", ptr->eid);  print\_below\_A(ptr->right);  }  } |

**텍스트, 스크린샷, 블랙이(가) 표시된 사진

자동 생성된 설명**

**[예제 2] car id와 grade를 받아, grade가 ‘A’ 가 아닌 car의 id 검색하기**

|  |
| --- |
| C |
| #define \_CRT\_SECURE\_NO\_WARNINGS  #include <stdio.h>  #include <stdlib.h>  typedef struct node {  struct node\* left;  unsigned carid;  char cgrade;  struct node\* right;  } CARTYPE;  int make\_bst(CARTYPE\*\* troot, unsigned id, char grade);  void nonpasscar(CARTYPE\* troot);  int make\_bst(CARTYPE\*\* troot, unsigned id, char grade) {  CARTYPE\* tptr = \*troot;  CARTYPE\* before = NULL;  // Traversing the tree to find the correct position for the new node  while (tptr) {  before = tptr;  if (id < tptr->carid) {  tptr = tptr->left;  }  else if (id > tptr->carid) {  tptr = tptr->right;  }  else {  return 0; // Duplicate id, do not insert  }  }  // Creating the new node  CARTYPE\* newNode = (CARTYPE\*)malloc(sizeof(CARTYPE));  if (!newNode) {  printf("Memory allocation failed\n");  exit(1);  }  newNode->carid = id;  newNode->cgrade = grade;  newNode->left = newNode->right = NULL;  // Inserting the new node in the correct position  if (before == NULL) {  \*troot = newNode; // Tree was empty  }  else if (id < before->carid) {  before->left = newNode;  }  else {  before->right = newNode;  }  return 1;  }  void nonpasscar(CARTYPE\* root) {  if (root) {  nonpasscar(root->left);  if (root->cgrade != 'A') {  printf("%u\t", root->carid);  }  nonpasscar(root->right);  }  }  void inorder(CARTYPE\* root) {  if (root) {  inorder(root->left);  printf("%d\t%c\n", root->carid, root->cgrade);  inorder(root->right);  }  }  void main(int argc, char\* argv[]) {  CARTYPE\* tree = NULL;  unsigned id;  char grade;  FILE\* input;  input = fopen(argv[1], "r");  if (input == NULL) {  printf("Cannot open input file...\n");  return 1;  }  while (fscanf(input, "%u %c", &id, &grade) != EOF) {  make\_bst(&tree, id, grade);  }    fclose(input);  printf("차량 정보 트리 구축 완료:\n");  inorder(tree);  printf("\n등급이 A가 아닌 차량의 ID:\n");  nonpasscar(tree);  printf("\n");  return 0;  } |

**텍스트, 스크린샷, 블랙이(가) 표시된 사진

자동 생성된 설명**

**[예제 3] 버블정렬**

|  |
| --- |
| C |
| //p.398 버블정렬 405, 406  #define \_CRT\_SECURE\_NO\_WARNINGS  #include <stdio.h>  #include <stdlib.h>  #include<string.h>  #define INUM 100  int bsearch\_stock(unsigned a[][2], int n, unsigned key);  void print\_item(char\* heading, unsigned a[][2], int n);  void bubble(unsigned a[][2], int n);  void main(int argc, char\* argv[]) {  FILE\* stockdb;  unsigned stock[INUM][2], item\_code;  int k = 0, s;  if ((stockdb = fopen(argv[1], "r")) == NULL) {  printf("Cannot open read file ....\n");  exit(1);  }  while ((fscanf(stockdb, "%u %u", &stock[k][0], &stock[k][1])) != EOF) k++;  print\_item("임의로 저장된 물품코드 : \n", stock, k);  //물품코드의 내림차순 정렬  bubble(stock, k);  print\_item("\n내림차순으로 정렬된 물품 코드 : \n", stock, k);  //검색  printf("\n검색할 item Number 입력 : ");  scanf("%u", &item\_code);  s = bsearch\_stock(stock, k, item\_code);  if (s == -1) printf("제고 물품이 없습니다.\n");  else printf("\n%u의 재고개수 = %u\n", stock[s][0], stock[s][1]);  }  void print\_item(char\* heading, unsigned a[][2], int n) {  int i;  printf(heading);  for (i = 0; i < n; i++) printf("%u\n", a[i][0]);  }  int bsearch\_stock(unsigned a[][2], int n, unsigned key) {  int left = 0, right = n - 1, mid;  while (left <= right) {  mid = (left + right) / 2;  if (key < a[mid][0]) left = mid + 1;  else if (key > a[mid][0]) right = mid - 1;  else return mid;  }  return -1;  }  void bubble(unsigned a[][2], int n) {  int i = n - 1, j, flag = 1;  unsigned tmp0, tmp1;  while (flag && i != 0) {  flag = 0;  for (j = 0; j <= i - 1; j++) {  if (a[j][0] < a[j + 1][0]) {  flag = 1;  tmp0 = a[j][0]; a[j][0] = a[j + 1][0]; a[j + 1][0] = tmp0;  tmp1 = a[j][1]; a[j][1] = a[j + 1][1]; a[j + 1][1] = tmp1;  }  }  i--;  }  } |

****

**[예제 4]**

|  |
| --- |
| C |
| //삽입정렬 p412  #define \_CRT\_SECURE\_NO\_WARNINGS  #include <stdio.h>  typedef struct student {  long id;  char name[20];  double score;  } sdatatype;  void insert\_sort1(sdatatype a[], int n);  void insert\_sort2(sdatatype a[], int n);  void main(int argc, char\* argv[])  {  sdatatype s[50];  FILE\* infile, \* out;  int i = 0, j;  char choice;  if (argc != 3) {  printf("실행 인수를 잘못 주었습니다. \n");  exit(1);  }  if ((infile = fopen(argv[1], "r")) == NULL) {  printf("입력 파일을 열 수 없습니다.\n");  exit(1);  }  if ((out = fopen(argv[2], "w")) == NULL) {  printf("결과 파일을 열 수 없습니다. \n");  exit(1);  }  while (fscanf(infile, "%ld %s %lf", &(s[i].id), s[i].name, &(s[i].score)) != EOF)  i++;  printf("학번순으로 저장하려면 1을 \n");  printf("성적순으로 저장하려면 2를 입력하시오 : \n");  scanf("%c", &choice);  if (choice == '1')insert\_sort1(s, i);  else if (choice == '2') insert\_sort2(s, i);  else {  printf("선택번호를 잘못 입력했습니다.");  exit(1);  }  for (j = 0; j < i; j++) {  fprintf(out, "%ld\t%s\t%.2f\n", (s[j].id), s[j].name, (s[j].score));  printf("%ld\t%s \t %.2f\n", (s[j].id), s[j].name, (s[j].score));  }  printf("Program terminated ... \n");  }  void insert\_sort1(sdatatype a[], int n) {  int i, j;  sdatatype temp;  for (i = 1; i < n - 1; i++) {  temp = a[i];  j = i - 1;  while (a[j].id > temp.id && j >= 0) {  a[j + 1] = a[j];  j--;  }  a[j + 1] = temp;  }  }  void insert\_sort2(sdatatype a[], int n) {  int i, j;  sdatatype temp;  for (i = 1; i < n - 1; i++) {  temp = a[i];  j = i - 1;  while (a[j].id < temp.id && j >= 0) {  a[j + 1] = a[j];  j--;  }  a[j + 1] = temp;  }  } |

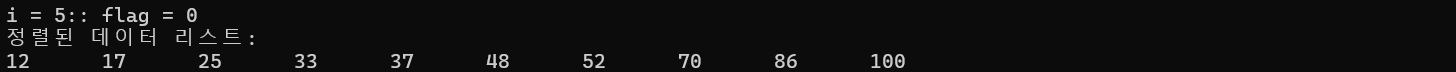
**텍스트, 스크린샷, 블랙이(가) 표시된 사진

자동 생성된 설명텍스트, 스크린샷, 블랙이(가) 표시된 사진

자동 생성된 설명**

**[예제 5]**

|  |
| --- |
| C |
| #define \_CRT\_SECURE\_NO\_WARNINGS  #include<stdio.h>  #include<stdlib.h>  void bubble(int a[], int n);  main() {  int a[] = { 12, 17, 25, 33, 48, 52, 86, 37, 100, 70 };  int i, n;  n = sizeof(a) / sizeof(int);  bubble(a, n);  printf("정렬된 데이터 리스트: \n");  for (i = 0; i < n; i++)  printf("%d\t", a[i]);  }  void bubble(int a[], int n)  {  int i = n - 1, j, tmp, flag = 1;  while (flag && i != 0) {  flag = 0;  for (j = 0; j <= i - 1; j++) {  if (a[j] > a[j + 1]) {  flag = 1;  tmp = a[j]; a[j] = a[j + 1]; a[j + 1] = tmp;  }  }  i--;  }  printf("i = %d:: flag = %d\n", i, flag);  } |

****