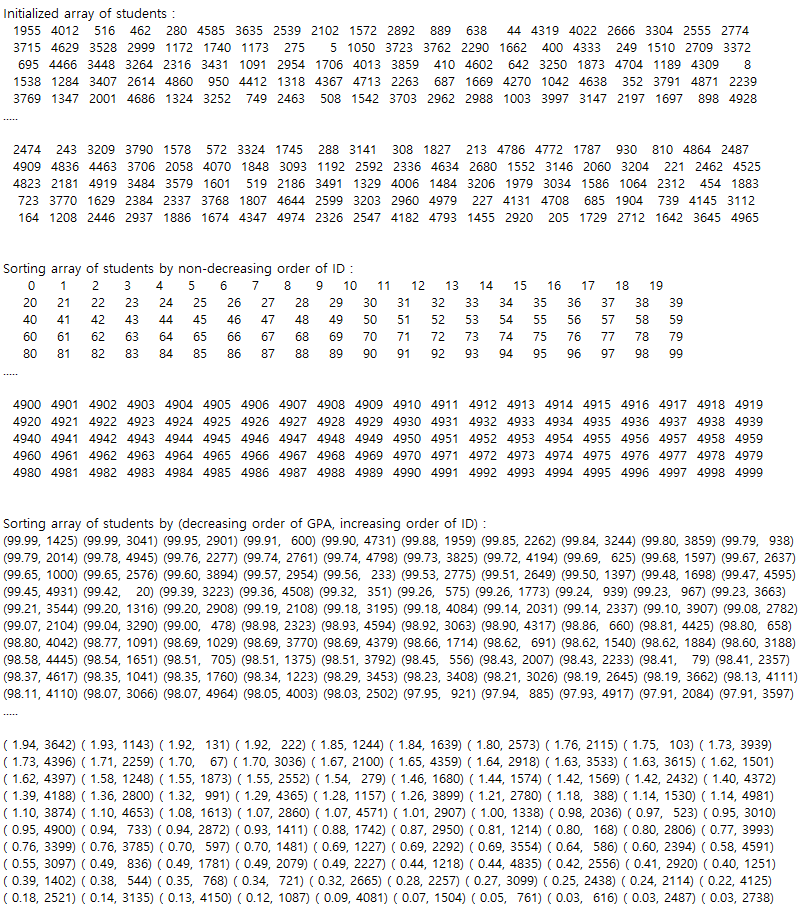
**Lab1**

|  |
| --- |
| 그림입니다. 원본 그림의 이름: YU_UI_RGB-10.png 원본 그림의 크기: 가로 2256pixel, 세로 3047pixel 프로그램 이름 : Adobe ImageReady |

|  |  |
| --- | --- |
| 과목명 | 객체지향프로그래밍과자료구조 |
| 교수님 | 김영탁 교수님 |
| 이 름 | 김주환 |
| 학 번 | 21812158 |
| 일 자 | 2021.09.03.금 |

|  |
| --- |
| /\* main.cpp \*/  /\* Description:  \* 학생 정보를 구조체로 생성  \* 학생 정보를 원하는 크기만큼 동적 할당  \* 학생 정보를 무작위로 초기화  \* ID로 오름차순 선택정렬  \* gpa로 내림차순 선택정렬, 같은 gpa에 ID로 오름차순 선택정렬  \* ID 이진 탐색  \* Programmed by J. H. Kim  \* Last updated : 2021. 09. 03 \*/  #include <iostream>  #include <iomanip>  #include <fstream>  //#include <stdlib.h>  #include "Student.h"  using namespace std;  #define NUM\_STUDNETS 5000  #define NUM\_SEARCH 5  #define NUM\_FIRST\_LAST\_BLOCK 100  void genBigRandArray(int\* randArray, int num\_rands);  void main() {  ofstream fout;  int\* student\_ids;  Student\* students, \* pSt;  int studnet\_search[NUM\_SEARCH] = { 1, 123, 999, 2500, 4999 };  fout.open("output.txt");  if (fout.fail()) {  cout << "Error in opening output.txt" << endl;  exit;  }  student\_ids = (int\*)malloc(sizeof(int) \* NUM\_STUDNETS);  students = (Student\*)malloc(sizeof(Student) \* NUM\_STUDNETS);  genBigRandArray(student\_ids, NUM\_STUDNETS);  initStudents(students, student\_ids, NUM\_STUDNETS);  // 학생 정보 생성  //printStudent\_IDs(students, NUM\_STUDENTS);  fout << endl << endl;  fout << "Initialized array of students : " << endl;  fprintBigArrayOfStudent\_IDs(fout, students, NUM\_STUDNETS, NUM\_FIRST\_LAST\_BLOCK);  // ID 오름차순 선택정렬  fout << endl << endl;  fout << "Sorting array of students by non-decreasing order of ID : " << endl;  sortStudentsByID(students, NUM\_STUDNETS);  fprintBigArrayOfStudent\_IDs(fout, students, NUM\_STUDNETS, NUM\_FIRST\_LAST\_BLOCK);    // GPA 내림차순 정렬 후 동일 ID 내림차순 정렬  fout << endl << endl;  fout << "Sorting array of students by (decreasing order of GPA, increasing order of ID) : " << endl;  sortStudentsByGPA\_ID(students, NUM\_STUDNETS);  fprintBigArrayOfStudent\_GPA\_IDs(fout, students, NUM\_STUDNETS, NUM\_FIRST\_LAST\_BLOCK);  // 탐색을 위한 재정렬  fout << endl << endl;  fout << "Sorting array of students by non-decreasing order of ID : " << endl;  sortStudentsByID(students, NUM\_STUDNETS);  fprintBigArrayOfStudent\_IDs(fout, students, NUM\_STUDNETS, NUM\_FIRST\_LAST\_BLOCK);  // 학생 ID 이진 탐색  fout << endl << endl;  fout << "Searching student with student\_ID : " << endl;  for (int i = 0; i < NUM\_SEARCH; i++) {  pSt = searchStudentByID(students, NUM\_STUDNETS, studnet\_search[i]);  fout << "Student search by ID (" << setw(4) << studnet\_search[i] << ") : ";  if (pSt != NULL)  fprintStudent(fout, pSt);  else  fout << "NULL - student was not found !!";  fout << endl;  }  fout.close();  }  void genBigRandArray(int\* randArray, int num\_rands) {  char\* flag;  int count = 0;  unsigned int u\_int32 = 0;  unsigned int bigRand;  flag = (char\*)calloc(sizeof(char), num\_rands);  while (count < num\_rands) {  u\_int32 = ((long)rand() << 15) | rand(); // 비트이동으로 bigRand 생성  bigRand = u\_int32 % num\_rands; // 원하는 크기로 스케일링  if (flag[bigRand] == 1) continue; // 중복 확인  flag[bigRand] = 1; // 중복 체크  randArray[count++] = bigRand;  }  } |
| /\* Student.h \*/  #include <iostream>  using namespace std;  typedef struct s\_Student {  int st\_id;  char name[16];  char dept[16];  int grade;  double gpa;  } Student;  void initStudents(Student students[], int st\_ids[], int num\_students);  void fprintStudent(ostream& fout, Student\* pSt);  void fprintStudent\_IDs(ostream& fout, Student students[], int num\_students);  void fprintBigArrayOfStudent\_IDs(ostream& fout, Student studnets[], int num\_students, int num\_first\_last);  void sortStudentsByID(Student students[], int num\_students);  void sortStudentsByGPA\_ID(Student students[], int num\_students);  void fprintBigArrayOfStudent\_GPA\_IDs(ostream& fout, Student students[], int num\_students, int num\_first\_last);  Student\* searchStudentByID(Student students[], int num\_studnets, int st\_ID); |
| /\* Student.cpp \*/  #include <iostream>  #include <iomanip>  using namespace std;  typedef struct s\_Student {  int st\_id;  char name[16];  char dept[16];  int grade;  double gpa;  } Student;  void initStudents(Student students[], int st\_ids[], int num\_students) {  int name\_len;  int j, grade;  double br;  srand(time(0));  for (int i = 0; i < num\_students; i++) {  students[i].st\_id = st\_ids[i];  name\_len = rand() % 26 + 'A';  students[i].name[0] = rand() % 26 + 'A';  for (j = 1; j < name\_len - 1; j++)  students[i].name[j] = rand() % 26 + 'a';  students[i].name[name\_len - 1] = '\0';  name\_len = rand() % 4 + 2;  for (j = 0; j < name\_len - 1; j++)  students[i].dept[j] = rand() % 26 + 'A';  students[i].dept[name\_len - 1] = '\0';  students[i].grade = rand() % 8 + 1;  students[i].gpa = (rand() % 10000) / 100.0;  }  }  void fprintStudent(ostream& fout, Student\* pSt) {  int count = 0;  fout.setf(ios::fixed);  fout.setf(ios::showpoint);  fout.precision(2);  fout << "Student(ID : " << setw(4) << pSt->st\_id;  fout << ", Name : " << setw(16) << pSt->name;  fout << ", Dept : " << setw(6) << pSt->dept;  fout << ", Grade : " << pSt->gpa;  }  void fprintStudent\_IDs(ostream& fout, Student students[], int num\_students) {  }  void fprintBigArrayOfStudent\_IDs(ostream& fout, Student students[], int num\_students, int num\_first\_last) {  Student\* pSt;  int count = 0;  if (num\_students <= num\_first\_last \* 2) {  for (int i = 0; i < num\_students; i++) {  pSt = &students[i];  fout << setw(6) << pSt->st\_id;  count++;  if ((count % 20) == 0) fout << endl;  }  return;  }  for (int i = 0; i < num\_first\_last; i++) {  pSt = &students[i];  fout << setw(6) << pSt->st\_id;  count++;  if ((count % 20) == 0) fout << endl;  }  fout << "....." << endl << endl;    for (int i = num\_students - num\_first\_last; i < num\_students; i++) {  pSt = &students[i];    fout << setw(6) << pSt->st\_id;  count++;  if ((count % 20) == 0) fout << endl;  }  }  // increasing order  void sortStudentsByID(Student students[], int num\_students) {  int i, j, mx;  char minName[16] = { 0 };  Student tmp;  int min\_st, min\_ID;  for (i = 0; i < num\_students - 1; i++) {  min\_st = i;  min\_ID = students[min\_st].st\_id;  for (j = i + 1; j < num\_students; j++) {  if (students[j].st\_id < min\_ID) {  min\_st = j;  min\_ID = students[j].st\_id;  }  }  if (min\_st != i) {  tmp = students[i];  students[i] = students[min\_st];  students[min\_st] = tmp;  }  }  }  // decreasing order in gpa, increasing order in ID  void sortStudentsByGPA\_ID(Student students[], int num\_students) {  Student tmp;  int max\_gpa\_index;  double max\_gpa;  for (int i = 0; i < num\_students - 1; i++) {  max\_gpa = students[i].gpa;  for (int j = i + 1; j < num\_students; j++) {  if (students[j].gpa > max\_gpa) { // gpa 내림차순 정렬  max\_gpa = students[j].gpa;  tmp = students[i];  students[i] = students[j];  students[j] = tmp;  }  else if (students[j].gpa == max\_gpa) { // gpa 같은 경우 ID 오름차순 정렬  if (students[i].st\_id > students[j].st\_id) {  tmp = students[i];  students[i] = students[j];  students[j] = tmp;  }  }  }  }  }  void fprintBigArrayOfStudent\_GPA\_IDs(ostream& fout, Student students[], int num\_students, int num\_first\_last) {  Student\* pSt;  int count = 0;  if (num\_students <= num\_first\_last \* 2) {  for (int i = 0; i < num\_students; i++) {  pSt = &students[i];  fout << "(";  fout.setf(ios::fixed);  fout.setf(ios::showpoint);  fout.precision(2);  fout << pSt->gpa << ",";  fout << setw(5) << pSt->st\_id << ") ";  count++;  if ((count % 10) == 0) fout << endl;  }  return;  }  for (int i = 0; i < num\_first\_last; i++) {  pSt = &students[i];  fout << "(";  fout.setf(ios::fixed);  fout.setf(ios::showpoint);  fout.precision(2);  fout << pSt->gpa << ",";  fout << setw(5) << pSt->st\_id << ") ";  count++;  if ((count % 10) == 0) fout << endl;  }  fout << "....." << endl << endl;  for (int i = num\_students - num\_first\_last; i < num\_students; i++) {  pSt = &students[i];  fout << "(";  if (pSt->gpa < 10) fout << " ";  fout.setf(ios::fixed);  fout.setf(ios::showpoint);  fout.precision(2);  fout << pSt->gpa << ",";  fout << setw(5) << pSt->st\_id << ") ";  count++;  if ((count % 10) == 0) fout << endl;  }  }  Student\* searchStudentByID(Student students[], int num\_studnets, int st\_ID) {  int low, high, mid;  int round = 0;  low = 0;  high = num\_studnets - 1;  while (low <= high) {  mid = (low + high) / 2;  if (st\_ID == students[mid].st\_id) return &students[mid];  else if (st\_ID < students[mid].st\_id) high = mid - 1;  else low = mid + 1;  }  return NULL;  } |



테이블이(가) 표시된 사진

자동 생성된 설명

**2. 2021-2 객체지향형 프로그래밍과 자료구조 실습 1 Oral Test**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 학번 | 21812158 | 성명 | 김주환 | 점수 |  |

|  |
| --- |
| **(1) 구조체와 배열의 차이점에 대하여 설명하라.**  구조체와 배열은 단순 작업을 위해 사용된다. 정렬 과 탐색 등 많은 작업의 기초가 된다.  구조체를 배열로 만들 수도 있고 배열을 구조체로 만들 수도 있다. 혼용도 가능하다. 그러므로 각자의 기능만 확인하겠다.   * 구조체 내부에 멤버를 가진다. 각 멤버는 자료형을 달리 할 수 있다. * 배열 내부의 모든 정보들이 같은 자료형을 가진다. |
| **(2) genBigRandArray(int \*randArray, int num\_rands) 가 0 … num\_rands – 1 범위의 중복되지 않는 난수들을 생성하고, 주어진 동적 배열 randArray[]에 저장하는 기능에 대하여 설명하라.**   * 지정된 크기의 동적 배열 randArray[] 생성 * call-by-pointer로 randArray[]를 genBigRandArray() 함수로 전달 * 중복되었는지 확인하는 flag를 randArray[]와 같은 크기로 동적 할당 flag가 0이면 신규 난수고 1이면 중복된 난수다. * 비트 이동으로 통해 u\_int32에 생성한 수를 저장 생성한 난수를 좌로 15비트 이동해서 연산자 ‘|’를 통해 총 30비트 크기의 난수를 생성한다. * RAND\_MAX는 15 비트다. 새로 생성한 난수는 RAND\_MAX(32,767) 보다 크다. %를 통해 원하는 크기로 스케일링 * 해당 수에 해당하는 flag 값이 1인지 0인지 확인하여 중복 여부를 판단한다. 중복되었다면 해당 난수를 randArray[]에 순서대로 저장하고 중복되지 않았다면 해당 flag 값을 1로 저장하고 continue한다.   이 때 randArray[]는 call-by-pointer로 전달되었기 때문에 따로 반환해주지 않아도 위에서 저장된 중복되지 않은 난수 배열이 randArray[]에 저장된다.  void genBigRandArray(int\* randArray, int num\_rands) {  char\* flag;  int count = 0;  unsigned int u\_int32 = 0;  unsigned int bigRand;  flag = (char\*)calloc(sizeof(char), num\_rands);  while (count < num\_rands) {  u\_int32 = ((long)rand() << 15) | rand();  bigRand = u\_int32 % num\_rands;  if (flag[bigRand] == 1) continue;  flag[bigRand] = 1;  randArray[count++] = bigRand;  }  } |
| **(3) 구조체 배열에 대한 선택 정렬의 내부 절차에 대하여 예를 들어 설명하라.**  선택정렬은 기준 값부터 마지막까지 이동하면서 가장 작은 값을 기준위치로 이동하는 정렬 방법이다.   * 구조체 배열 자체를 call-by-pointer로 전달 * 가장 작은 학생 인덱스와 그 학생의 ID를 받기위한 변수와 가장 작은 값을 기준 위치로 옮기기 위한 변수를 선언한다. * 오름차순 선택 정렬을 위해 이중 반복문 사용 * 외부 반복문(i)은 0 ~ (배열의 크기) – 1 만큼 반복 * 기준 인덱스(i)를 min\_st에 저장하고 해당 학생의 ID를 min\_ID에 저장한다. * 내부 반복문(j)은 1 ~ (배열의 크기) 만큼 반복 * 기준 인덱스(i) + 1 부터 마지막까지 전부 기준 값과 비교해가며 가장 작은 값을 찾는다. 이 때 가장 작은 값을 찾을 때 마다 해당 인덱스와 ID 값을 min\_st와 min\_ID에 저장한다. * 끝가지 비교한 후 기준 값보다 작은 값을 찾았을 경우에만 SWAP을 실행해준다. 이 때 SWAP은 구조체 자체를 SWAP해준다.   void sortStudentsByID(Student students[], int num\_students) {  int i, j, mx;  char minName[16] = { 0 };  Student tmp;  int min\_st, min\_ID;  for (i = 0; i < num\_students - 1; i++) {  min\_st = i;  min\_ID = students[min\_st].st\_id;  for (j = i + 1; j < num\_students; j++) {  if (students[j].st\_id < min\_ID) {  min\_st = j;  min\_ID = students[j].st\_id;  }  }  if (min\_st != i) {  tmp = students[i];  students[i] = students[min\_st];  students[min\_st] = tmp;  }  }  } |
| **(4) Windows 운영체제 환경에서 C/C++ 프로그램 모듈이 실행하는 시간을 마이크로 초 단위로 측정하는 방법에 대하여 예를 들어 설명하라.**   * 헤더파일 Windows.h, time.h * 사용 함수 QueryPerformanceFrequency() : 전달 받은 인수에 주파수를 저장한다. QueryPerformanceCounter() : 전달 받은 인수에 각 시간을 저장한다. * 절차 - 시간의 차이를 저장할 변수와 주파수를 저장할 변수 선언 이 때 변수는 LARGE\_INTEGER로 선언한다. - QueryPerformanceFrequency()를 통해 주파수 저장 - 모듈 실행 전 후를 나눠 QueryPerformanceCounter()를 실행하여 시간을 저장 - 위에서 저장한 시간의 차이를 저장한다. 이 때 차이는 LONGLONG 자료형으로 선언된 변수에 저장한다. 추가로 시간이 저장된 변수는 “시간변수.QuadPart”와 같은 형식으로 사용한다. - 두 시간의 차이를 주파수로 나누어 경과된 시간을 double 자료형으로 저장 이 때 freq도 freq.QuadPart와 같이 사용해야한다. - 결과 값에 1000을 곱해서 us로 표현 * 계산식     int\* array;  LARGE\_INTEGER freq, t1, t2;  LONGLONG t\_diff;  double elapsed\_time;  QueryPerformanceFrequency(&freq);  QueryPerformanceCounter(&t1);  hybridQuickSelectionSort(array, size);  QueryPerformanceCounter(&t2);  t\_diff = t2.QuadPart - t1.QuadPart;  elapsed\_time = (double)t\_diff / freq.QuadPart;  fprintf(fout, "hybrid\_QS\_SS sort took %11.2lf [milliseconds], ", elapsed\_time \* 1000.0); |