**REPORT**

**[실습 과제 2주차]**



**과 목 : 심화프로그래밍02**

**담당교수 : 윤성림 교수님**

**학 과 : 컴퓨터공학과**

**이 름 : 이재혁**

**제 출 일 : 2024.3.16**

텍스트, 클립아트이(가) 표시된 사진

자동 생성된 설명

**실습문제 1**

**텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명**

**해결전략**

함수 내부에서 최댓값이 바뀔 때마다, 그 위치인 max\_index값을 따로 저장해 두어, 함수 반환 시 max\_index위치의 주소 값을 반환해, 최댓값의 주소를 출력합니다.

**소스코드**

#include <stdio.h>

double\* get\_max(double\*, int);

int main() {

double arr[10] = { 1.2, 2.3, 3.4, 4.1, 2.4, 8.2, 3.593, 0.123, 5.412, 7.26 };

printf("최댓값은%.3f이고, 해당요소의 주소는%p입니다.\n", \*(get\_max(arr, 10)), get\_max(arr, 10));

// 함수가 return 하는 값을 그대로 출력

return 0;

}

double\* get\_max(double\* A, int size) {

int max = A[0];

int max\_index = 0; // 매번 바뀌는 최댓값의 위치가 저장될 변수

for (int i = 1; i < size; i++) {

if (A[i] > max) {

max = A[i];

max\_index = i; // 최댓값이 갱신되면 index정보를 저장

}

}

return &A[max\_index]; // 배열에서 최댓값이 저장된 주소를 반환

}

**결과**



가장 큰 값과, 그 값의 주소 값이 출력 됩니다.

**실습문제 2**

텍스트, 폰트, 스크린샷, 번호이(가) 표시된 사진

자동 생성된 설명

**해결전략**

scores 배열과 grades 배열의 주소를 넘겨받아,

scores 배열에 grades 배열의 원소에 100점 만점으로 전환하여 값을 입력합니다.

**소스코드**

#include <stdio.h>

void compute(double\*, double\*);

int main() {

double grades[10] = { 0.0, 0.5, 1.0, 1.5, 2.0, 2.8, 3.0, 3.7, 4.2, 4.3 };

// 학생들의 학점을 종류별로 입력

double scores[10];

// 학점을 100점 만점으로 전환하여 입력 할 배열

compute(grades, scores);

for (int i = 0; i < 10; i++) {

printf("학생 %d: %.2f\n", i + 1, scores[i]);

}

return 0;

}

void compute(double\* arr1, double\* arr2) {

for (int i = 0; i < 10; i++) {

if (arr1[i] == 4.3) {

arr2[i] = 100.0;

// 점수 변환 비율이 소수 2자리까지만 입력했을 때 0.3의 오차 발생

// 4.3의 학점은 100점 만점이기 때문에 100점으로 무조건 전환

}

else {

arr2[i] = arr1[i] \* 23.25581;

}

}

}

// 소감

// 컴퓨터의 계산은 오차가 발생할 수 있다는 사실을 다시 확인했습니다.

// 처음 점수 전환비율을 소수 2자리까지만으로 계산을 했더니 4.3전환 기준으로 0.03의 오차가 발생했습니다.

// 소수 5번째 자리 까지는 사용해야 보다 정확한 비율로 계산이 되었습니다.

//

// 학점 계산 같은 경우는 적은 범위의 수여서 계산하는 소수 자리 수를 늘려서 해결했지만

// 더 큰 범위의 계산을 할 때 만약 메모리가 숫자의 크기를 전부 담지 못해서 계산 오차가 날 수 있다는 생각이 들었습니다.

// 따라서 선언하는 자료형의 메모리크기를 이해하고 알고 있는 것이 정교한 계산을 위해 필수적이라고 느꼈습니다.

**결과**

텍스트, 폰트, 스크린샷, 블랙이(가) 표시된 사진

자동 생성된 설명

최대한 근사 시켜 학점에 23.25581의 값을 곱한 값이 출력됩니다.

**실습문제 3**

텍스트, 스크린샷, 폰트, 디자인이(가) 표시된 사진

자동 생성된 설명

**해결전략**

구조체에는 실수, 허수의 값을 저장할 int 자료를 관련한 연산을 시행합니다.

**소스코드**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <math.h> // 제곱근 계산을 사용하기 위해 선언

typedef struct {

int real;

int imag;

}Complex; // 실수부 허수부의 자료를 가지는 Complex 구조체 선언

void Sum(Complex, Complex);

void Dis(Complex, Complex);

void Mul(Complex, Complex);

void Abs(Complex\* , Complex\* );

// 각각의 계산을 해 출력해주는 함수 선언

int main() {

Complex c1, c2;

printf("c1, 정수부 허수부 입력: ");

scanf("%d %d", &c1.real, &c1.imag);

printf("c2, 정수부 허수부 입력: ");

scanf("%d %d", &c2.real, &c2.imag);

// c1, c2가 생성될 때 실수부와 허수부의 정수 자료가 저장될 메모리가 생성, 그 주소에 값을 입력 받아 대입

Sum(c1, c2);

Dis(c1, c2);

Mul(c1, c2);

Abs(&c1, &c2);

}

void Sum(Complex c1, Complex c2) {

Complex s;

s.real = c1.real + c2.real;

s.imag = c1.imag + c2.imag;

printf("덧셈: %d + %di\n", s.real, s.imag);

}

void Dis(Complex c1, Complex c2) {

Complex d;

d.real = c1.real - c2.real;

d.imag = c1.imag - c2.imag;

printf("뺄셈: %d + %di\n", d.real, d.imag);

}

void Mul(Complex c1, Complex c2) {

Complex m;

m.real = c1.real \* c2.real - c1.imag \* c2.imag;

m.imag = c1.imag \* c2.real + c2.imag \* c1.real;

printf("곱셈: %d + %di\n", m.real, m.imag);

// 실수부는 실수부의 곱과 허수부의 곱의 차

// 허수부는 실수부와 허수부의 곱 끼리의 합

}

// 덧셈 뺄셈 곱셈함수는 구조체들의 정보를 바탕으로 새로운 객체를 복사하여 생성 후 그 값을 사용

void Abs(Complex\* c1, Complex\* c2) {

double a1, a2;

a1 = sqrt(c1->real \* c1->real + c1->imag \* c1->imag);

a2 = sqrt(c2->real \* c2->real + c2->imag \* c2->imag);

printf("절대값: %.2f\n %.2f\n", a1, a2);

}

// 절댓값 함수는 구조체의 주소를 받아와 직접 값을 참조하여 절댓값을 계산

// 소감

// 함수를 작성하면서 구조체를 call by value로 자료를 받을 경우 구조체 내부에 여러 자료형이 있어,

// 일반 자료형을 받는 함수보다 더 큰 메모리를 사용하여 복사본을 생성한다고 생각했습니다.

// 따라서 절댓값함수는 call by reference로 구조체의 주소 값을 넘겨 받아 직접 값을 참조했습니다.

// 다만 원본 구조체의 값이 변경될 수도 있기 때문에, call by value방법도 필요하다고 생각했습니다.

**결과**

텍스트, 폰트, 스크린샷, 블랙이(가) 표시된 사진

자동 생성된 설명

절댓값 : 341/2 ≈ 5.83095

81/2 ≈ 2.828427

올바르게 계산되어 출력됩니다.

**실습문제 4**

**텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명**

**해결전략**

구조체 생성 후 멤버에 값을 입력해 출력합니다.

**소스코드**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <string.h> // 문자열 복사를 위한 strcpy함수를 사용하기 위해 사용

typedef struct

{

char name[20];

char pNumber[11]; // 전화번호 -> 010\*\*\*\*\*\*\*\* 0으로 시작하기 때문에 컴퓨터가 8진수로 인식 따라서 문자열로 저장

char address[20];

} Person; // 이름, 전화번호, 주소의 값을 자료로 가지는 Person 선언

int main() {

Person p1, p2;

strcpy(p1.name, "Kim Dongguk");

strcpy(p1.pNumber, "01011111111");

strcpy(p1.address, "Dongguk University");

strcpy(p2.name, "Lee Younghee");

strcpy(p2.pNumber, "01022222222");

strcpy(p2.address, "Dongguk University");

// 문자열 자료들을 값으로 복사

printf("첫번째 사람 정보 :\n");

printf("이름 : %s\n", p1.name);

printf("전화번호 : %s\n", p1.pNumber);

printf("주소 : %s\n", p1.address);

printf("\n");

printf("두번째 사람 정보 :\n");

printf("이름 : %s\n", p2.name);

printf("전화번호 : %s\n", p2.pNumber);

printf("주소 : %s\n", p2.address);

return 0;

}

// 소감

// 숫자가 0 으로 시작할 경우 컴퓨터가 8진수로 인식하는 문제가 있었습니다.

// 문자열로 정보를 저장하는 방법을 선택했고

// 8진수로 입력을 받고 출력할 때 '0'을 출력 후 8진수로 출력하는 것이 메모리는 더 효율적으로 사용한다고 생각합니다.

// 다만 출력문을 단번에 이해하기 어렵다고 생각해서 문자열로 선언하는 방식을 선택했습니다.

**결과**

**텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명**

**실습문제 5**

**텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명**

**해결전략**

구조체 정보를 자료형으로 갖는 배열을 10칸을 생성에 각각의 칸에 학생들의 정보를 입력합니다.

**소스코드**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <string.h> // 문자열 복사를 위한 strcpy함수를 사용하기 위해 사용

typedef struct

{

char name[20];

char number[11];

float average;

char major[20];

char career[20];

} student;// 이름, 학번, 평균평점, 학과, 진로의 자료를 갖는 student 선언

int main()

{

student arr[10]; // student 자료 10개를 저장할 배열 생성

strcpy(arr[0].name, "A");

strcpy(arr[0].number, "2021000001");

arr[0].average = 4.3;

strcpy(arr[0].major, "computer science");

strcpy(arr[0].career, "AI");

strcpy(arr[1].name, "B");

strcpy(arr[1].number, "2021000002");

arr[1].average = 4.0;

strcpy(arr[1].major, "computer science");

strcpy(arr[1].career, "engineer");

strcpy(arr[2].name, "C");

strcpy(arr[2].number, "2021000003");

arr[2].average = 3.7;

strcpy(arr[2].major, "multimedia");

strcpy(arr[2].career, "engineer");

strcpy(arr[3].name, "D");

strcpy(arr[3].number, "2021000004");

arr[3].average = 3.3;

strcpy(arr[3].major, "computer science");

strcpy(arr[3].career, "AI");

strcpy(arr[4].name, "E");

strcpy(arr[4].number, "2021000005");

arr[4].average = 3.0;

strcpy(arr[4].major, "computer science");

strcpy(arr[4].career, "engineer");

strcpy(arr[5].name, "F");

strcpy(arr[5].number, "2021000006");

arr[5].average = 2.7;

strcpy(arr[5].major, "computer science");

strcpy(arr[5].career, "backend");

strcpy(arr[6].name, "G");

strcpy(arr[6].number, "2021000007");

arr[6].average = 2.3;

strcpy(arr[6].major, "computer science");

strcpy(arr[6].career, "AI");

strcpy(arr[7].name, "H");

strcpy(arr[7].number, "2021000008");

arr[7].average = 2.0;

strcpy(arr[7].major, "computer science");

strcpy(arr[7].career, "engineer");

strcpy(arr[8].name, "I");

strcpy(arr[8].number, "2021000009");

arr[8].average = 1.7;

strcpy(arr[8].major, "computer science");

strcpy(arr[8].career, "backend");

strcpy(arr[9].name, "J");

strcpy(arr[9].number, "2021000010");

arr[9].average = 1.3;

strcpy(arr[9].major, "computer science");

strcpy(arr[9].career, "AI");

for (int i = 0; i < 10; i++) {

printf("학생 %d 정보 : \n", i + 1);

printf("이름 : %s\n", arr[i].name);

printf("학번 : %s\n", arr[i].number);

printf("평균 평점 : %.1f\n", arr[i].average);

printf("학과 : %s\n", arr[i].major);

printf("진로 : %s\n", arr[i].career);

printf("\n");

}

// 배열 내의 원소들이 모두 같은 자료형을 가지고 있으므로 반복문의 i를 사용해 반복 출력

return 0;

}

**결과**

**텍스트, 스크린샷, 메뉴, 흑백이(가) 표시된 사진

자동 생성된 설명 텍스트, 스크린샷, 메뉴, 흑백이(가) 표시된 사진

자동 생성된 설명**

**실습문제 6**

**텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명**

**해결전략**

구조체 정보를 자료형으로 갖는 배열을 5칸을 생성에 각각의 칸에 교수들의 정보를 입력합니다.

**소스코드**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <string.h>

typedef struct

{

char name[20];

char pNumber[12];

char address[20];

char subject[30];

char major[25];

} professor; // 개인정보, 담당과목(여러 개), 학과 자료를 포함하는 구조체 professor 선언

int main()

{

professor p[5]; // professor의 자료를 5개 가지고 있는 배열생성 (professor 5명 생성)

strcpy(p[0].name, "교수님 1");

strcpy(p[0].pNumber, "11111111");

strcpy(p[0].address, "동국대학교");

strcpy(p[0].subject, "컴퓨터구성");

strcpy(p[0].major, "컴퓨터공학");

strcpy(p[1].name, "교수님 2");

strcpy(p[1].pNumber, "22222222");

strcpy(p[1].address, "동국대학교");

strcpy(p[1].subject, "자료구조");

strcpy(p[1].major, "컴퓨터공학");

strcpy(p[2].name, "교수님 3");

strcpy(p[2].pNumber, "33333333");

strcpy(p[2].address, "동국대학교");

strcpy(p[2].subject, "객체지향프로그래밍");

strcpy(p[2].major, "컴퓨터공학");

strcpy(p[3].name, "교수님 4");

strcpy(p[3].pNumber, "44444444");

strcpy(p[3].address, "동국대학교");

strcpy(p[3].subject, "기초프로그래밍");

strcpy(p[3].major, "컴퓨터공학");

strcpy(p[4].name, "교수님 5");

strcpy(p[4].pNumber, "55555555");

strcpy(p[4].address, "동국대학교");

strcpy(p[4].subject, "공학경제");

strcpy(p[4].major, "산업시스템공학");

// 기본정보 입력

printf("교수정보출력 : \n\n");

for (int i = 0; i < 5; i++) {

printf("교수님 %d 정보: \n", i + 1);

printf("이름: %s\n", p[i].name);

printf("전화번호: %s\n", p[i].pNumber);

printf("주소: %s\n", p[i].address);

printf("담당과목: %s\n", p[i].subject);

printf("전공: %s\n\n", p[i].major);

}

// 반복문을 통해 배열의 원소에 접근하여 출력

return 0;

}

**결과**

**텍스트, 스크린샷, 폰트, 흑백이(가) 표시된 사진

자동 생성된 설명 텍스트, 스크린샷, 폰트, 흑백이(가) 표시된 사진

자동 생성된 설명**

**실습문제 7**

**텍스트, 폰트, 스크린샷, 라인이(가) 표시된 사진

자동 생성된 설명**

**해결전략**

담당교수의 정보를 포인터로 입력 받아서 새로운 자료를 생성하지 않고 원래 있던 자료를 가지고 담당교수를 지정합니다.

**소스코드**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <string.h>

typedef struct

{

char name[20];

char pNumber[12];

char address[20];

} professor; // 교수 구조체 선언

typedef struct

{

char name[20];

char number[11];

float average;

char major[20];

char career[20];

professor\* p; // 담당교수의 자료가 저장된 주소 값을 포함해 담당교수 정보에 접근가능

} person; // 사람(학생)의 구조체 선언

int main() {

professor p[5]; //교수 정보 5개를 저장할 배열 선언

strcpy(p[0].name, "교수님 1");

strcpy(p[0].pNumber, "11111111");

strcpy(p[0].address, "동국대학교");

strcpy(p[1].name, "교수님 2");

strcpy(p[1].pNumber, "22222222");

strcpy(p[1].address, "동국대학교");

strcpy(p[2].name, "교수님 3");

strcpy(p[2].pNumber, "33333333");

strcpy(p[2].address, "동국대학교");

strcpy(p[3].name, "교수님 4");

strcpy(p[3].pNumber, "44444444");

strcpy(p[3].address, "동국대학교");

strcpy(p[4].name, "교수님 5");

strcpy(p[4].pNumber, "55555555");

strcpy(p[4].address, "동국대학교");

// 교수님의 정보 입력

person student[10]; // 학생 정보 10개 저장할 배열 선언, 교수님 1~5가 2명씩 학생을 담당한다.

strcpy(student[0].name, "stu. A");

strcpy(student[0].number, "2021000001");

student[0].average = 4.3;

strcpy(student[0].major, "computer science");

strcpy(student[0].career, "AI");

student[0].p = &p[0]; // 담당교수의 정보가 저장된 주소를 입력

strcpy(student[1].name, "stu. B");

strcpy(student[1].number, "2021000002");

student[1].average = 4.0;

strcpy(student[1].major, "computer science");

strcpy(student[1].career, "engineer");

student[1].p = &p[0];

strcpy(student[2].name, "stu. C");

strcpy(student[2].number, "2021000003");

student[2].average = 3.7;

strcpy(student[2].major, "multimedia");

strcpy(student[2].career, "engineer");

student[2].p = &p[1];

strcpy(student[3].name, "stu. D");

strcpy(student[3].number, "2021000004");

student[3].average = 3.3;

strcpy(student[3].major, "computer science");

strcpy(student[3].career, "AI");

student[3].p = &p[1];

strcpy(student[4].name, "stu. E");

strcpy(student[4].number, "2021000005");

student[4].average = 3.0;

strcpy(student[4].major, "computer science");

strcpy(student[4].career, "engineer");

student[4].p = &p[2];

strcpy(student[5].name, "stu. F");

strcpy(student[5].number, "2021000006");

student[5].average = 2.7;

strcpy(student[5].major, "computer science");

strcpy(student[5].career, "studentkend");

student[5].p = &p[2];

strcpy(student[6].name, "stu. G");

strcpy(student[6].number, "2021000007");

student[6].average = 2.3;

strcpy(student[6].major, "computer science");

strcpy(student[6].career, "AI");

student[6].p = &p[3];

strcpy(student[7].name, "stu. H");

strcpy(student[7].number, "2021000008");

student[7].average = 2.0;

strcpy(student[7].major, "computer science");

strcpy(student[7].career, "engineer");

student[7].p = &p[3];

strcpy(student[8].name, "stu. I");

strcpy(student[8].number, "2021000009");

student[8].average = 1.7;

strcpy(student[8].major, "computer science");

strcpy(student[8].career, "backend");

student[8].p = &p[4];

strcpy(student[9].name, "stu. J");

strcpy(student[9].number, "2021000010");

student[9].average = 1.3;

strcpy(student[9].major, "computer science");

strcpy(student[9].career, "AI");

student[9].p = &p[4];

printf("학생정보출력:\n");

for (int i = 0; i < 10; i++) {

printf("학생 %d 정보: \n", i + 1);

printf("이름: %s\n", student[i].name);

printf("학번: %s\n", student[i].number);

printf("평균 평점: %.1f\n", student[i].average);

printf("학과: %s\n", student[i].major);

printf("진로: %s\n", student[i].career);

printf("지도 교수 이름: %s\n", student[i].p->name);

printf("지도 교수 전화번호: %s\n", student[i].p->pNumber);

printf("지도 교수 주소: %s\n\n", student[i].p->address);

} // 학생정보를 출력, 학생정보 출력 시 담당교수의 정보는 포인터 이므로 -> 참조 연산자를 사용해 자료에 접근한다

return 0;

}

// 소감

// 구조체 내부에 다른 구조체의 관한 정보가 있을 시 새로운 멤버 변수를 만드는 것 보다

// 포인터를 통해 다른 구조체 객체의 주소를 받아 자료에 접근하는 게 메모리 사용에 효율적이라는 생각이 들었습니다.

**결과**

**텍스트, 스크린샷, 메뉴, 폰트이(가) 표시된 사진

자동 생성된 설명 텍스트, 스크린샷, 메뉴, 폰트이(가) 표시된 사진

자동 생성된 설명**

포인터로 넘겨받은 자료도 -> 참조연산자로 출력이 되는 모습입니다.