REPORT

[실습 과제 1주차]



과 목 : 심화프로그래밍02

담당교수 : 윤성림 교수님

학 과 : 컴퓨터공학과

이 름 : 이재혁

제 출 일 : 2024.3.08

텍스트, 클립아트이(가) 표시된 사진

자동 생성된 설명

**실습문제 1**



**문제해결전략**

컴퓨터의 임의의 값과 입력을 조건으로 나누어 비교

**소스코드**

#include <stdio.h>

#include <stdlib.h> //random함수를 사용하기 위해서 포함시켜 줍니다.

#include <time.h> //매번 다른 값을 생성시키기 위해서 time을 통해 매번 다른 시드 값을 전달 해 줍니다.

int main()

{

srand(time(NULL));

int com, user; //컴퓨터와 사용자가 가질 값을 저장할 변수를 선언합니다.

com = rand() % 3; //컴퓨터가 무작위로 0, 1, 2 중 하나의 값을 가지게 합니다.

printf("가위 = 0, 바위 = 1, 보 = 2\n입력: ");

scanf\_s("%d", &user); //사용자의 입력을 받습니다.

if (com == 0) { //사용자의 입력을 받았을 때 컴퓨터가 가진 경우에 따라 결과를 출력합니다.

printf("com: 가위\n");

if (user == 0) printf("비겼습니다\n");

else if (user == 1) printf("이겼습니다\n");

else if (user == 2) printf("졌습니다\n");

else printf("잘못 된 입력입니다."); //범위 밖의 숫자를 입력했을 시 결과를 알 수 없기 때문에 따로 분류 합니다.

}

else if (com == 1) {

printf("com: 바위\n");

if (user == 0) printf("졌습니다\n");

else if (user == 1) printf("비겼습니다\n");

else if (user == 2) printf("이겼습니다\n");

else printf("잘못 된 입력입니다.");

}

else {

printf("com: 보\n");

if (user == 0) printf("이겼습니다\n");

else if (user == 1) printf("졌습니다\n");

else if (user == 2) printf("비겼습니다\n");

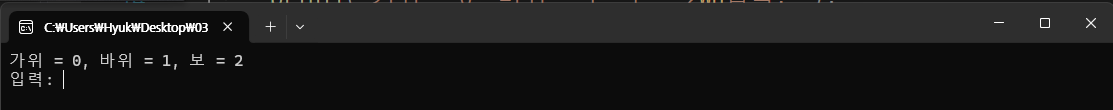
else printf("잘못된 입력입니다.");

}

return 0; // main함수의 반환형이 int이므로 종료될 때 0 을 반환합니다.

}

결과 및 결과 분석

 텍스트, 스크린샷, 블랙이(가) 표시된 사진

자동 생성된 설명 텍스트, 스크린샷, 디스플레이, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

사용자가 입력을 했을 때 컴퓨터가 임의로 가지고 있던 값과 비교해 결과를 출력합니다.

주어진 범위 이외의 값이 입력될 경우 프로그램을 수행하지 않습니다.

**실습문제 2**



**문제해결전략**

정적변수를 함수내부에 선언해 함수가 호출될 때만 접근할 수 있게 해 사용

소스코드

#include <stdio.h>

int function(); // 자신이 호출 된 횟수를 반환하는 함수입니다.

int main() {

int call; // 호출 된 횟수가 저장 될 변수입니다.

for (int i = 0; i < 3; i++) { // 처음 3번 호출했습니다.

call = function();

}

printf("현재까지 %d번 호출 됐습니다\n", call); // 호출된 횟수를 출력합니다.

for (int i = 0; i < 5; i++) { // 5번 추가 호출합니다.

call = function();

}

printf("현재까지 %d번 호출 됐습니다\n", call); // 3번 이후 5번 더 출력 됐기 때문에 8번이 출력되는지 확인합니다.

return 0;

}

int function() {

static int repeat = 0; // 이 함수 내부에서만 변화시킬 수 있고, 프로그램이 종료 될 때 사라지는 정적변수 생성합니다.

repeat++;

return repeat; //함수가 반복한 횟수를 반환합니다.

}

// 전역 변수로 함수가 반복 될 때 마다 변수를 하나씩 증가시키는 방법을 사용할 수 있었습니다.

// 하지만 그 방법은 main함수에서도 반복횟수를 수정할 수 있고, 함수가 반환하는 방법이 아니기 때문에

// 프로그램이 종료되는 시점에 사라지는 정적변수를 함수 안에 배치해서 보다 정확하게, 호출될 때만 변수가 증가하게 작성했습니다.

결과 및 결과 분석

텍스트, 스크린샷, 폰트, 블랙이(가) 표시된 사진

자동 생성된 설명

처음 3번 호출 후 호출된 횟수를 반환했습니다.

다시 5번을 추가로 호출하고 호출된 횟수 총 8번을 반환했습니다.

**실습문제 3**

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

**문제해결전략**

입력을 while반복문으로 계속 받고, printAnswer()함수가 실행될 때 종료되도록 설계

**소스코드**

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

int number;

static int min = 1;

static int max = 100;

int trycount = 0; // 모든 함수에서 시도횟수를 추가하거나, 값을 참조해야 하기 때문에 전역변수로 지정했습니다.

int is\_end = 0; // 프로그램을 종료할지 결정할 변수 입니다.

void setNumber(); // 함수들은 최대, 최소 출력이나, 변수 설정 등의 역할이므로 반환 값이 없기 때문에 void형으로 지정했습니다.

void printHead();

void printHigher(int);

void printLower(int);

void printAnswer();

int main() {

int input; // 사용자가 입력 할 변수입니다

setNumber();

while (is\_end == 0) { // 전역에 선언된 is\_end가 초기값이 0일때는 계속 반복합니다.

printHead();

scanf\_s("%d", &input);

if (input > max || input < min) {

printf("범위 밖의 수 입니다.\n"); // 입력 받은 수가 최대, 최소값의 범위 밖에 있을 경우 아래를 확인할 필요가 없습니다.

trycount++;

continue;

}

trycount++; // 입력마다 횟수를 추가합니다

if (number > input) {

printLower(input);

}

else if (number < input) {

printHigher(input);

}

else printAnswer();

}

return 0;

}

void setNumber() {

srand(time(NULL));

number = rand() % 100 + 1; // 1 ~ 100 범위의 정수를 생성합니다.

printf("난수설정완료\n");

}

void printHead() { // 수시로 변화하는 숫자의 범위를 반복의 시작 부분마다 출력해줍니다.

printf("숫자의 범위는 %d~%d 입니다.\n", min, max);

printf("숫자 입력 : ");

}

void printLower(int m) {

printf("입력받은 숫자가 작습니다.\n");

min = m + 1; // m은 이미 아닌 것을 알고 있으므로 그 다음 값을 최소 범위로 잡아줍니다.

}

void printHigher(int M) {

printf("입력받은 숫자가 큽니다.\n");

max = M - 1; // M은 이미 아닌 것을 알고 있으므로 그 이전 값을 최대 범위로 잡아줍니다.

}

void printAnswer() { // 정답을 맞췄기 때문에 시도 횟수를 출력하고 is\_end의 값을 바꿔줍니다.

printf("정답입니다. 시도횟수 : %d\n", trycount);

is\_end = 1;

}

결과 및 결과 분석

텍스트, 스크린샷, 디스플레이, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

사용자가 입력을 난수와 비교해주고 최대, 최소의 범위를 변경합니다.

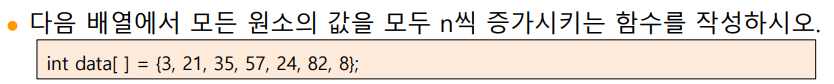
사용자의 입력이 정답이 아닐 때

입력 받은 숫자가 난수보다 작다면 최소값을 (입력은 정답이 아니므로) 입력+1

난수보다 크다면 입력-1의 값으로 조정해줍니다.

범위 밖의 숫자를 입력 받았다면 난수와 비교과정을 거치지 않습니다.

**실습문제 4**



**문제해결전략**

배열의 주소를 넘겨, 추가적인 메모리를 사용하지 않고, 값을 증가

**소스코드**

#include <stdio.h>

void increase(int\*, int, int);

// 주어진 배열에서 크기를 키워주면 되기 때문에, void에 주소 값을 넘겨 받는 함수로 작성합니다.

// 포인터로 넘겨 받기 때문에 배열의 크기를 알 수 없기 때문에, 배열의 크기도 인자로 넘겨줍니다.

int main() {

int data[] = { 3, 21, 35, 57, 24, 82, 8 };

int len = sizeof(data) / 4; //int 배열은 4바이트를 원소로 갖기 때문에 원소의 크기로 나눠서 구했습니다.

int n; // 증가시킬 값을 저장 시킬 변수

printf("원래 배열 : ");

for (int i = 0; i < sizeof(data) / 4; i++) {

printf("%d ", data[i]);

}

printf("\n");

printf("증가시킬 값 : ");

scanf\_s("%d", &n);

increase(data, len, n);

printf("증가된 배열 : ");

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

printf("%d ", data[i]);

}

printf("\n");

return 0;

}

void increase(int\* arr, int len, int n) {

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

arr[i] += n;

}

}

// call by value로 주어진 배열의 값 들을 받아온 뒤 n씩 더해 새로운 배열을 만드는 방법도 있습니다.

// 그 경우에는 배열의 크기만큼 새로운 메모리를 차지하기 때문에 비효율적이라고 생각했습니다.

// 따라서 번거로워도 주소와 배열의 크기를 함께 넘겨줘 주어진 배열을 바꾸도록 작성했습니다.

결과 및 결과 분석

텍스트, 스크린샷, 블랙, 폰트이(가) 표시된 사진

자동 생성된 설명

원래 배열을 출력하고 증가시킬 값을 입력 받은 뒤, 증가시킨 원소들을 출력합니다.

**실습문제 5**

텍스트, 폰트, 스크린샷, 라인이(가) 표시된 사진

자동 생성된 설명

**문제해결전략**

주소 값을 인자로 넘겨 주소에 해당하는 값을 수정

**소스코드**

#include <stdio.h>

void swap(int\*, int\*);

int main() {

int num1, num2;

printf("num1 : ");

scanf\_s("%d", &num1);

printf("num2 : ");

scanf\_s("%d", &num2);

printf("바뀌기 전 num1 : %d, num2 : %d\n", num1, num2);

swap(&num1, &num2);

printf("바뀐 후 num1 : %d, num2 : %d\n", num1, num2);

return 0;

}

void swap(int\* a, int\* b) {

int tmp;

tmp = \*a;

\*a = \*b;

\*b = tmp;

}

// 만약 swap함수에 두 숫자를 int로 받았다면, main함수에 있는 변수들과 완전히 다른 새로운 변수입니다.

// 또한 어차피 swap함수가 끝날 때 없어지는 변수이기 때문에, return값이 없는 swap함수에서는 의미가 없는 변수입니다.

// 따라서 int \* 즉 주소 값을 인자로 넘겨받아 그곳을 가리키는 포인터들을 만들고 가리키는 것을 바꿔 줬습니다.

결과 및 결과 분석

텍스트, 스크린샷, 블랙, 폰트이(가) 표시된 사진

자동 생성된 설명

두 수를 따로 입력 받고, 함수 실행 이전 이후에 값을 출력해 변화를 확인합니다.

**실습문제 6**

텍스트, 스크린샷, 폰트, 라인이(가) 표시된 사진

자동 생성된 설명

**문제해결전략**

새로운 배열에 저장하고 출력하기 보다는 함수 자체에서 배열을 생성하고 함수가 끝날 때 배열을 해제해 메모리 사용을 최소화

**소스코드**

#include <stdio.h>

void Sum(float[][3], float[][3]); // 2차원 배열 -> 길이가 3인 배열을 몇개 갖고 있는가 입니다.

void Distinct(float[][3], float[][3]); // 계산 수행 후 출력 까지만 기능이기 때문에 return이 없어도 된다고 생각했습니다.

int main() {

float arr1[2][3] = { {4.2, 4.3, 3.8}, {3.7, 1.5, 0.7} }; // 실수 형을 저장하는 배열 이므로 float으로 선언했습니다.

float arr2[2][3] = { {5.2, 2.1, 1.8}, {3.2, 1.4, 2.9} }; // 길이가 3인 배열을 2개 갖고 있는 형태입니다

float\* sum[3];

float\* distinct[3];

Sum(arr1, arr2);

Distinct(arr1, arr2);

return 0;

}

void Sum(float a1[][3], float a2[][3]) { // a1, a2라는 2차원 배열이 다시 선언되는 과정이기 때문에, 형태를 지정해줍니다.

float sum[2][3]; // 계산된 값을 저장할 배열입니다.

for (int i = 0; i < 2; i++) {

for (int j = 0; j < 3; j++) {

sum[i][j] = a1[i][j] + a2[i][j]; // 같은 위치에 있는 값 끼리 더한 후 저장합니다.

}

}

printf("두 배열의 합\n");

for (int i = 0; i < 2; i++) { // 배열의 내용을 출력해줍니다.

for (int j = 0; j < 3; j++) {

printf("%.1f ", sum[i][j]);

}

printf("\n");

}

}

void Distinct(float a1[][3], float a2[][3]) {

float dis[2][3]; // 계산된 값을 저장할 배열입니다.

for (int i = 0; i < 2; i++) {

for (int j = 0; j < 3; j++) {

dis[i][j] = a1[i][j] - a2[i][j]; // 같은 위치에 있는 값 끼리 뺀 후 저장합니다.

}

}

printf("두 배열의 차\n");

for (int i = 0; i < 2; i++) { // 배열의 내용을 출력해줍니다.

for (int j = 0; j < 3; j++) {

printf("%.1f ", dis[i][j]);

}

printf("\n");

}

}

결과 및 결과 분석

텍스트, 블랙, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

Sum함수는 두 배열을 합한 배열을 출력합니다.

Distinct함수는 두 배열을 뺀 배열을 출력합니다.

**실습문제 7**

텍스트, 스크린샷, 폰트, 라인이(가) 표시된 사진

자동 생성된 설명

**문제해결전략**

포인터가 가리키는 주소의 값을 증가

**소스코드**

#include <stdio.h>

int main()

{

int ary[][5] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 }; // 총 10개의 원소가 5개씩 묶여 있는 2차원 배열입니다.

int(\*ptr)[5];

int i, j;

ptr = ary; // 원소가 5개인 배열을 가리키는 포인터를 생성해 2차원 배열을 가리키게 합니다.

for (i = 0; i < 2; i++)

{ // 배열 2칸에 원소가 5개인 배열이 들어가 있는 구조입니다.

for (j = 0; j < 5; j++)

{

ptr[i][j] += 5; // ptr이 가리키는 배열의 원소에 5씩 더합니다.

}

}

for (i = 0; i < 2; i++)

{

for (j = 0; j < 5; j++)

{

printf("%d ", ary[i][j]); // 잘 더해졌는지 확인합니다.

}

}

return 0;

}

결과 및 결과 분석

텍스트, 스크린샷, 폰트, 블랙이(가) 표시된 사진

자동 생성된 설명

주어진 배열에 5씩 더해져서 출력 된 형태입니다.

**실습문제 8**

텍스트, 폰트, 스크린샷, 라인이(가) 표시된 사진

자동 생성된 설명

**문제해결전략**

새로운 공간을 만들고 그곳에 원소를 복사

**소스코드**

#include <stdio.h>

#include <stdlib.h> // 동적할당을 위해서 사용할 malloc함수를 위해 포함시킵니다.

void copyarray(double\*, double\*, int);

int main() {

double ary[5] = { 3.12, 5.14, 7.25, 7.48, 5.91 };

double\* ary\_copied;

ary\_copied = (double\*)malloc(sizeof(double) \* 5); // 복사한 원소를 저장할 메모리를 새로 할당합니다.

printf("주어진 배열 : ");

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

printf("%.2f ", ary[i]);

}

printf("\n");

copyarray(ary, ary\_copied, 5);

free(ary\_copied); // 동적할당한 메모리를 해제해줍니다.

return 0;

}

void copyarray(double\* source, double\* target, int size) {

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

target[i] = source[i]; // 새로 할당한 주소에 원래 함수의 원소를 대입합니다.

}

printf("복사 한 배열 : ");

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

printf("%.2f ", target[i]);

}

printf("\n");

}

// 처음에는 copyarray()가 동적할당 한 주소를 return해서, main에서 복사한 내용을 출력하려 했습니다.

// 하지만 그러면 굳이 target 배열을 인자로 받을 이유가 없다고 생각해서,

// 함수 내에서 원소를 복사하고 출력하고 동적할당을 해제하는 void함수로 작성했습니다.

결과 및 결과 분석



새로 할당한 배열에 같은 원소들이 복사된 것을 확인할 수 있습니다.

**실습문제 9**

텍스트, 스크린샷, 폰트, 라인이(가) 표시된 사진

자동 생성된 설명

**문제해결전략**

정수형 변수를 문자열로 변환

**소스코드**

#define \_CRT\_SECURE\_NO\_WARNINGS // sprintf 보안경고로 인한 컴파일 에러를 없애기 위해 전처리 했습니다.

#include <stdio.h>

#include <stdlib.h> // malloc 함수 사용을 위해 포함합니다.

int main() {

int n = 0x324F3A24; // 0x -> 16진수를 위한 표현입니다 따라서 324F3A24만 문자열로 저장하면 됩니다

char\* pc;

pc = (char\*)malloc(sizeof(char) \* 9); // pc는 324F3A24 총 8개와 문자열이 끝날 때 \0을 필요로 하므로 9칸의 char공간을 필요로 합니다.

sprintf(pc, "%x", n);

printf("%s\n", pc); // 문자열 pc를 출력합니다.

free(pc); // 동적할당으로 생성한 공간을 해제해줍니다.

return 0;

}

결과 및 결과 분석



16진수가 문자열인 %s로 출력된 것을 확인할 수 있습니다.