아주대학교 프로그래밍언어

201821048 미디어학과 이서영

2019/10/28 함수, 배열 사용

# 소스코드 및 주석

/\*

[과제#5-1] 함수를 이용한 사칙연산 프로그램

문제: 지난주 과제#5로 수행했던 사칙연산 프로그램에 대해 각각 덧셈함수, 뺄셈

함수, 곱셈함수, 나눗셈 함수 등등 필요한 함수들을 가급적 많이 만들어서

다시 작성하시오.

두 수를 입력하세요:

\*\*\*\*\*\*\* 연산메뉴 \*\*\*\*\*\*\*

1. 덧셈

2. 뺄셈

3. 곱셈

4. 나눗셈

5. 종료

수행할 연산을 선택하세요:

추가로 고민할 문제

1. 오류메시지 처리 (분모가 0일 때, 메뉴 이외의 숫자 입력 시)

2. 반복문(While)을 이용해서 반복해서 연산 수행, 종료메뉴를 선택할 때까지

3. 메뉴 추가 : 두 수 재 입력 / 나머지 연산

\*/

#include <stdio.h>

#include <stdlib.h>

void add(int fir, int sec);

//덧셈 함수 있음 표시

void sub(int fir, int sec);

//뺄셈 함수 있음 표시

void mul(int fir, int sec);

//곱셈 함수 있음 표시

void divide(int fir, int sec);

//나누기 함수 있음 표시

void fir\_sec(int\* firr, int\* secr);

//다시 입력 함수 있음 표시

void remain(int fir, int sec);

//나머지 함수 있음 표시

int main() {

int fir, sec;

int\* firr, \* secr;

//다시 입력할 때 함수를 사용하기 위함

firr = &fir;

secr = &sec;

//fir과 sec에 각 주소값 연결

int choice = 0;

fir\_sec(&fir, &sec);

//첫번째로 입력받기

while (choice != 7) {

//7번을 눌러 종료할때까지 반복하기

printf("\*\*\*\*\*\*\* 연산메뉴 \*\*\*\*\*\*\*\n");

printf("1. 덧셈");

printf("2. 뺄셈");

printf("3. 곱셈");

printf("4. 나눗셈");

printf("5. 두 수 재 입력");

printf("6. 나머지 연산");

printf("7. 종료\n");

printf("수행할 연산을 선택하세요:");

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

//실행할 메뉴 보여주고 선택 값 입력받기

switch (choice)

{

case 1:

add(fir, sec);

break;

//더하기 함수 실행

case 2:

sub(fir, sec);

break;

//빼기 함수 실행

case 3:

mul(fir, sec);

break;

//곱하기 함수 실행

case 4:

divide(fir, sec);

break;

//나누기 함수 실행

case 5:

fir\_sec(firr, secr);

break;

//입력받는 함수 실행

case 6:

remain(fir, sec);

break;

//나머지 함수 실행

case 7:

break;

break;

//7이면 빠져나가기

default:

printf("잘못된 입력입니다. 다시 입력하세요");

break;

//그 외의 숫자가 들어갔을 때 잘못된 알려줌

}

}

return 0;

}

void add(int fir, int sec) {

int result = fir + sec;

printf("\n덧셈의 결과는 %d\n", result);

} //더해서 나온 값 출력

void sub(int fir, int sec) {

int result = fir - sec;

printf("\n뺄셈의 결과는 %d\n", result);

} //빼서 나온 값 출력

void mul(int fir, int sec) {

int result = fir \* sec;

printf("\n곱셈의 결과는 %d\n", result);

} //곱해서 나온 값 출력

void divide(int fir, int sec) {

double result = (double)fir / sec;

printf("\n나눗셈의 몫은 %lf\n", result);

} //나누어서 나온 값 출력

void remain(int fir, int sec) {

int result = fir % sec;

printf("\n나눗셈의 나머지는 %d\n", result);

} //나머지 연산해서 나온 값 출력

void fir\_sec(int\* firr, int\* secr) {

printf("\n두 수를 입력하세요:");

scanf\_s("%d %d", firr, secr);

}//포인터를 이용해 연결해 함수로 값을 바꿀 수 있도록 함

/\*

요구사항해결

1. switch문에서 defalut를 사용해 예외처리를 함

2. 7번이 아니면(종료조건이 아닐 시) 계속 반복하도록 구현함

3.

3-1. 두 수 재입력 : 함수에서 포인터를 이용해 값을 변경 가능하도록 함

3-2. 나눗셈 나머지 함수 : 나머지 연산자를 이용해 함수로 만들어 처리함

\*/

/\*

[과제#5-2] 로또번호 생성기 프로그램

요구사항 : [실습#1]의 로또번호 생성기를 바탕으로 1부터 45까지의 숫자 중에서

임의의 숫자 6개가 맞아야 하는 로또 번호 생성기 프로그램 작성.

사용자로부터 몇 장의 로또를 구매할 것인지를 입력 받아

6개의 임의의 숫자가 출력된 로또 1매를 사용자로부터 입력 받은

구매숫자만큼 출력.

구매할 로또의 개수를 입력하세요 : 3

1번 로또 : 13 20 34 8 42 31

2번 로또 : 41 22 15 19 33 38

3번 로또 : 7 36 24 19 20 44

고려사항 :

1) 1장의 로또에서 출력되는 임의의 숫자 6개는 중복되면 안된다.

“중복번호 피하는 알고리즘”이 핵심.

2) chapter10 장의 배열부분을 보고 배열을 사용하여 생성된 임의의

난수를 저장한다

\*/

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

int sorting(int lottery[6]) {

// 비효율적인 이중 for문으로 배열을 돌리면서 비교하는 방법을 사용함

for (int i = 0; i < 6; i++) {

//각 칸을 돈다.

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

//그 전 칸의 값들과 비교한다

if (lottery[i] < lottery[j]) {

//만약 해당 칸이 전의 칸들보다 작다면

int t = lottery[j];

lottery[j] = lottery[i];

lottery[i] = t;

//서로 위치 바꾸기

//작은 값을 앞으로, 큰 값을 뒤로 가도록 정렬함

}

}

}

}

int main() {

srand(time(NULL));

//시간에 따라 랜덤하도록 함

int number\_of\_lottery;

//로또 개수입력

int lottery[6];

//로또의 숫자는 6개로 고정됨

printf("구매할 로또의 개수를 입력하세요 :");

scanf\_s("%d", &number\_of\_lottery);

//로또 개수입력 안내 및 입력받기

for (int i = 1; i < number\_of\_lottery + 1; i++) {

printf("\n%2d번째 로또 : ", i);

//입력받은 로또의 개수 세기, 몇번쨰인지 안내 출력

for (int j = 0; j < 6; j++) {

//매 로또 칸에 랜덤하게 숫자 생성

lottery[j] = (rand() % 45) + 1;

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

//각 로또를 전 값들과 비교해 같으면 새로운 값 입력함

if (lottery[j] == lottery[k]) {

lottery[j] = (rand() % 45) + 1;

}

}//만들어진 중복없는 로또 내용 출력함

printf("%4d", lottery[j]);

}//정렬한 후의 값 안내

printf(" 정렬하기");

sorting(lottery);

//정렬한 후 보내줌

//배열은 그 자체가 포인터 역할을 할 수 있기 떄문에 상관 없음

//각 주소에 연결되어 함수로 바로 사용 가능

for (int i = 0; i < 6; i++) {

//정렬된 로또 출력

printf("%4d", lottery[i]);

}

}

return 0;

}

/\*1. 중복을 피하기 위해

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

//각 로또를 전 값들과 비교해 같으면 새로운 값 입력함

if (lottery[j] == lottery[k]) {

lottery[j] = (rand() % 45) + 1;

}

부분을 넣음

2. 랜덤한 값을 각 배열에 넣기 위해서

for (int j = 0; j < 6; j++) {

//매 로또 칸에 랜덤하게 숫자 생성

lottery[j] = (rand() % 45) + 1;

부분을 넣음

3. 보기 편하도록 정렬 함수를 만듦

for (int i = 0; i < 6; i++) {

//각 칸을 돈다.

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

//그 전 칸의 값들과 비교한다

if (lottery[i] < lottery[j]) {

//만약 해당 칸이 전의 칸들보다 작다면

int t = lottery[j];

lottery[j] = lottery[i];

lottery[i] = t;

//서로 위치 바꾸기

//작은 값을 앞으로, 큰 값을 뒤로 가도록 정렬함

}

}

}

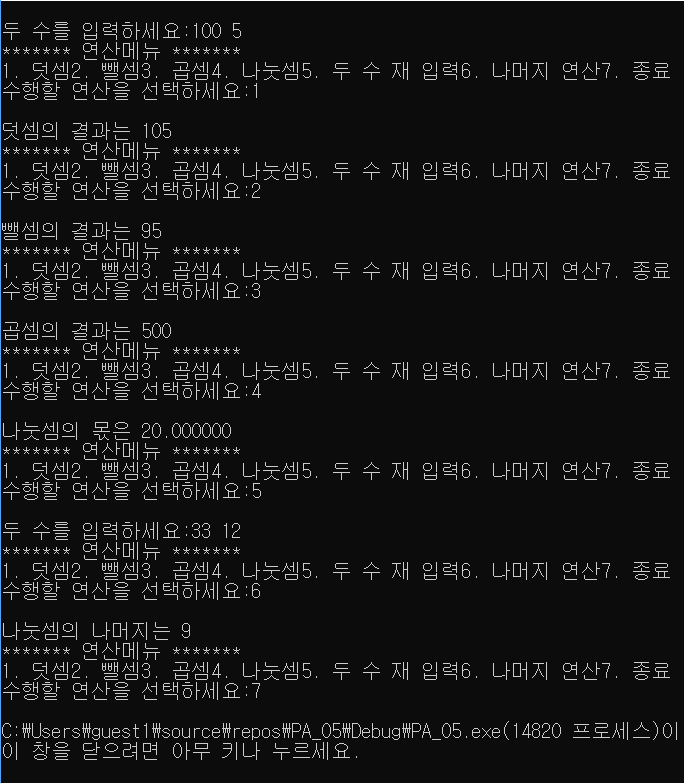
비효율적이지만 로또를 만 장 이상 사지 않는다고 가정하고

이러한 코드를 사용했다.

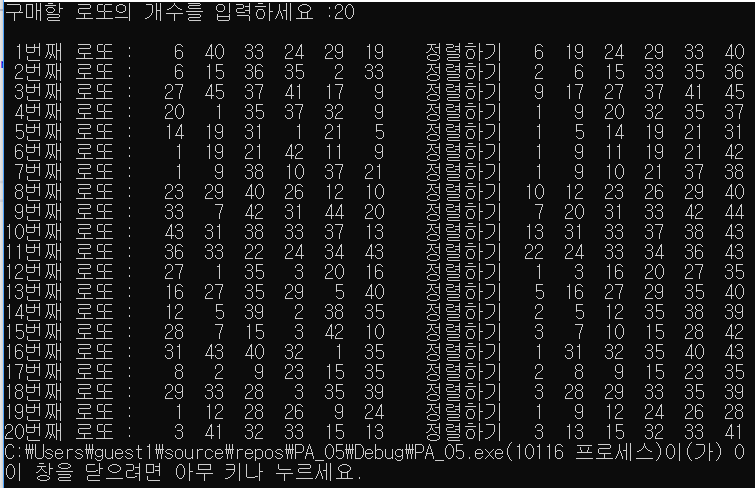
\*/

# 실행파일

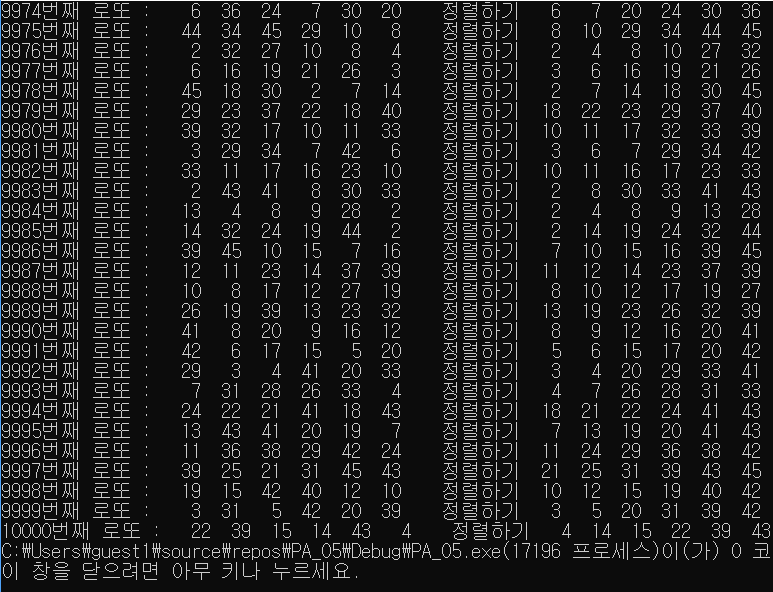
No1



No. 2



No2 에서 개선이 필요한 점



이처럼 만 장 이상을 산다면 연산 속도가 매우 느려진다 왜냐하면 o(n^3)으로 메인함수 부분에서 삼중 for문을 사용했기 떄문이다. 하지만 문제 자체가 로또를 산다고 해서 각 개인은 한 주에 몇 개 밖에 사지 못하기 떄문에 입력 개수가 많아질 때의 문제는 발생하지 않는다고 가정했다.