|  |
| --- |
| 2020 1학기 |
| 프로그래밍 언어 레포트 |
| 7주차 학습과제 |

|  |
| --- |
| 송 지민  사회학과 201721899 |

**[과제#1] 실습#3, #4 실습수행 및 실행화면 캡처**

* 실습#3 코드 및 실행화면

#include <stdio.h>

int get\_integer(void);

void is\_prime(int n);

int main(void)

{

int n = get\_integer();

is\_prime(n);

return 0;

}

int get\_integer(void)

{

int n;

printf("정수를 입력하시오 : ");

scanf("%d", &n);

return n;

}

void is\_prime(int n)

{

int i, j, check;

for(i=2; i<=n; i++){

check = 0;

for(j=1; j<i; j++){

if(i%j == 0){

check++;

}

}

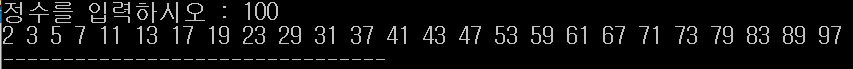
if(check != 1)

continue;

printf("%d ", i);

}

}



* 실습#4 코드 및 실행화면

#include <stdio.h>

int is\_multiple();

int main(void)

{

int n=0;

int m=0;

printf("첫번째 정수를 입력하시오: ");

scanf("%d", &n);

printf("두번째 정수를 입력하시오: ");

scanf("%d", &m);

int result = is\_multiple(n,m);

if(result==1)

printf("%d은(는) %d의 배수입니다.", n, m);

else

printf("%d은(는) %d의 배수가 아닙니다.", n, m);

return 0;

}

int is\_multiple(int n, int m)

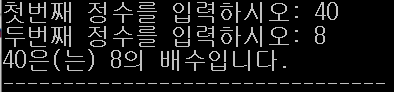
{

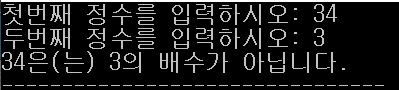
if(n%m==0)

return 1;

else

return 0;

}



**[과제#2] 함수를 이용한 사칙연산 프로그램 작성**

* 코드

// 함수를 이용한 사칙연산 프로그램

#include <stdio.h> // 표준입출력 헤더파일

#include <math.h> // 수학라이브러리 함수, pow() 함수 사용을 위해서 선언

/\* math.h 은 삼각함수, 역삼각함수, 쌍곡선함수, 지수함수, 실수와 정수의 절대값, 제곱근 등 여러 수학 함수를 포함하는 C언어의 표준 라이브러리이다.\*/

// 함수 원형 : 컴파일러에게 함수에 대하여 미리 알리는 것

double add(double x, double y); // 두 수를 덧셈연산하기 위한 함수

double sub(double x, double y); // 두 수를 뺄셈연산하기 위한 함수

double multi(double x, double y); // 두 수를 곱셈연산하기 위한 함수

double div(double x, double y); // 두 수를 나눗셈 연산하기 위한 함수

int mod(int x, int y); // 두 수를 나눈 나머지를 구하기 위한 함수

// 나머지 연산의 경우, 실수를 가지고 연산이 불가능하기 때문에 반환유형과 매개변수 모두 int로 선언

double Pow(double x, double y); // 첫번째 수를 두번째 수만큼 거듭제곱 하기 위한 함수

/\* 메인 함수를 상위에 두는 것이 일반적이며, 코드를 보기 편하기 때문에 메인 함수에서 이용되지만 메인 함수 뒤에 자리한 함수 원형을 사용해 주어야 한다.

이 때, 함수 원형을 정의하는 것이기 때문에 변수를 선언할 때처럼 반환유형과 매개변수의 자료형을 반드시 써줘야 한다.\*/

int main(void) // 메인 함수 선언부, 종료를 누르기 전까지 반복작업을 하도록 만들어야 함

{

// 변수선언 및 초기화

double a=0.0; // 첫번째로 입력받을 수, 실수로 선언하며 초기화

double b=0.0; // 두번째로 입력받을 수, 실수로 선언하며 초기화

int op=0; // 연산메뉴를 입력받을 변수, 정수형으로 선언

double answer=0.0; // 연산의 답을 저장할 변수, 실수형으로 선언

double change=0.0; // 8번의 두 수 순서바꾸기에 이용할 변수

// 프로그램명 출력하고 수 입력받기

printf("☞☞연산 프로그램☜☜\n\n"); // printf()함수 이용하여 프로그램 이름 출력

printf("두 수를 입력하세요: "); // printf()함수 이용하여 두수 입력안내 출력

scanf("%lf %lf", &a, &b); // scanf()함수 이용하여 두 수 입력받기

// 종료 메뉴를 입력하기 전까지 반복하기

while(op!=9){ //사용자로부터 받은 op값이 9일 때 반복이 멈춤

printf("\n\*\*\*\*\*\*\* 연산메뉴 \*\*\*\*\*\*\*\n"); // printf()함수 이용하여 연산메뉴바 출력

printf("1. 덧셈\n"); // printf()함수 이용하여 덧셈메뉴 출력

printf("2. 뺄셈\n"); // printf()함수 이용하여 뺄셈메뉴 출력

printf("3. 곱셈\n"); // printf()함수 이용하여 곱셈메뉴 출력

printf("4. 나눗셈\n"); // printf()함수 이용하여 나눗셈메뉴 출력

printf("5. 나머지 연산\n"); // printf()함수 이용하여 나머지연산메뉴 출력

printf("6. 첫번째 수를 두번째 수만큼 거듭제곱\n"); // printf()함수 이용하여 거듭제곱연산메뉴 출력

printf("7. 숫자 재입력\n"); // printf()함수 이용하여 숫자재입력메뉴 출력

printf("8. 두 수의 순서 바꾸기\n"); // printf()함수 이용하여 두수 순서바꾸기메뉴 출력

printf("9. 종료\n"); // printf()함수 이용하여 종료메뉴 출력

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n"); // printf()함수 이용하여 연산메뉴 닫아주기

printf("수행할 연산을 선택하세요: "); // printf()함수 이용하여 연산 선택 안내하기

scanf("%d", &op); // scanf()함수 이용하여 연산메뉴 입력받기

// 입력받은 op는 이어질 제어문에서 경우에 따라 연산을 수행할 때 사용됨

// 입력받은 op에 따라 다르게 처리하기

if(op<1||op>9) // 사용자가 메뉴의 숫자를 벗어난 수를 입력했을 때

{

printf("\n메뉴를 잘못 입력하셨습니다.\n"); // printf()함수 이용하여 오류 안내

printf("메뉴를 다시 입력해주세요.\n"); // printf()함수 이용하여 메뉴 재입력 안내

continue; // 메뉴를 재입력 받으러 반복문의 시작으로 올라감

}

if(op==4||op==5)

if(b==0) // 사용자가 메뉴 중 4, 5를 선택했는데 나누는 수에 0이 올 때

{

printf("\n나눗셈, 나머지 연산에서 나누는 수는 0이 될 수 없습니다.\n");

printf("연산을 다시 선택해주세요.\n"); // 오류메시지 출력

continue; // 연산 재선택받기

}

if(op==7) // 사용자가 메뉴 중 7을 선택했을 때(숫자 재입력)

{

printf("\n두 수를 다시 입력하세요: "); // printf()함수 이용하여 안내

scanf("%lf %lf", &a, &b); // scanf()함수 이용하여 재입력받기

}

if(op==8) // 사용자가 메뉴 중 8을 선택했을 때(두 수 순서 바꾸기)

{

change=a; // change에 a 저장하기

a=b; // a에 b값 저장하기

b=change; // change에 넣어둔 원래 a값을 b에 저장하기

printf("\n순서가 %.3lf %.3lf와 같이 바뀌었습니다.\n", a, b); // 바뀐 형태 안내하기

continue; // 연산메뉴 출력 및 입력으로 돌아가기

}

if(op>=1)

if(op<=6) // 사용자가 덧셈부터, 거듭제곱 메뉴 중 하나를 선택했을 때

{

switch(op){ //switch문을 이용하여 제어, break;를 통해 빠져나올 수 있음

case 1: // 덧셈을 선택한 경우

answer = add(a,b); // 덧셈 함수 호출

break;

case 2: // 뺄셈을 선택한 경우

answer = sub(a,b); // 뺄셈 함수 호출

break;

case 3: // 곱셈을 선택한 경우

answer = multi(a,b); // 곱셈 함수 호출

break;

case 4: // 나눗셈을 선택한 경우

answer = div(a,b); // 나눗셈 함수 호출

break;

case 5: // 나머지 연산을 선택한 경우

answer = mod((int)a,(int)b); // 나눗셈 연산 함수 호출

// 나머지 연산은 정수를 통해서만 가능하므로 반드시 강제 형변환을 해줘야 함

// a, b는 앞서 실수로 선언했으므로 정수로 형변환을 해줌

printf("\n나머지 연산에서 두 수는 정수로 변환되어 계산됩니다.\n");

// 나머지 연산에서 발생하는 강제 형변환을 안내

break;

case 6: // 거듭제곱 연산을 선택한 경우

answer = Pow(a,b); // 거듭제곱 함수 호출

break;

/\* 함수를 정의할 때, 매개변수를 두개로 했으므로 호출할 때 인수를 두개 주어야 하며 정수형으로 정의했으면 정수로, 실수로 정의했으면 실수로 주어야 한다. 호출할 때는 반환유형을 빼고 이름(인수)(매개변수를 정의했다면) 형태로 쓴다.\*/

}

printf("\n★답은 %.3lf입니다.★\n", answer);

/\* 연산메뉴 1부터 6까지는 연산을 했을 때 답이 출력되어야 하므로

switch문으로 묶어주어 switch문을 빠져나왔을 때 답이 출력되게 한다.\*/

}

}

printf("\n연산이 종료되었습니다.");

// 사용자로부터 op를 9로 입력받았을 때 연산을 종료되고 이를 안내하므로 종료 안내는 반복문 바깥에 써야 함

return 0; // 0을 반환하여 끝내기

}

// 연산에 사용되는 함수

/\* 함수는 특정 작업을 수행하는 명령어들의 모음에 이름을 붙인 것이다.

함수는 반환유형에 따라 반환값이 없을 수도 있고 있을 수도 있으며 입력을 받을 수도 있다.

라이브러리 함수는 컴파일러에서 지원되는 함수들이며 사용자가 직접 함수를 만들 수도 있다(사용자 정의 함수).

반환형 함수이름(매개변수)의 형태로 정의되며 매개변수가 없을 수도 있다.

매개변수는 호출 프로그램에 의하여 함수에 실제로 전달되는 인수를 전달받는 변수이다.

매개변수가 없으면 (void) 혹은 ()처럼 쓸 수 있고 반환값이 없을 때는 void라고 반환유형을 표시한다.\*/

double add(double x, double y) // 반환값 및 매개변수를 실수로 갖는 덧셈 함수 정의

{

return x+y; // 덧셈수식을 반환

}

double sub(double x, double y) // 반환값 및 매개변수를 실수형으로 갖는 뺄셈 함수 정의

{

return x-y; // 뺄셈수식을 반환

}

double multi(double x, double y) // 반환값 및 매개변수를 실수형으로 갖는 곱셈 함수 정의

{

return x\*y; // 곱셈수식을 반환

}

double div(double x, double y) // 반환값 및 매개변수를 실수형으로 갖는 나눗셈 함수 정의

{

return x/y; // 나눗셈수식을 반환

}

int mod(int x, int y) // 반환값 및 매개변수를 정수형으로 갖는 나머지 연산 함수 정의

{

return x%y; // 나머지연산수식을 반환

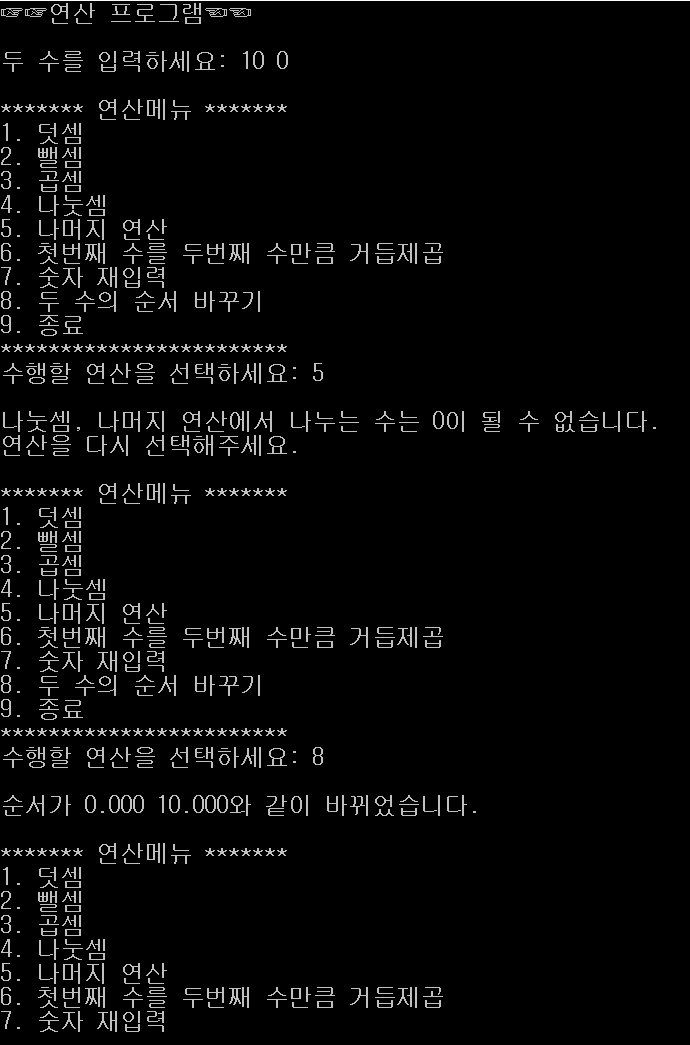
}

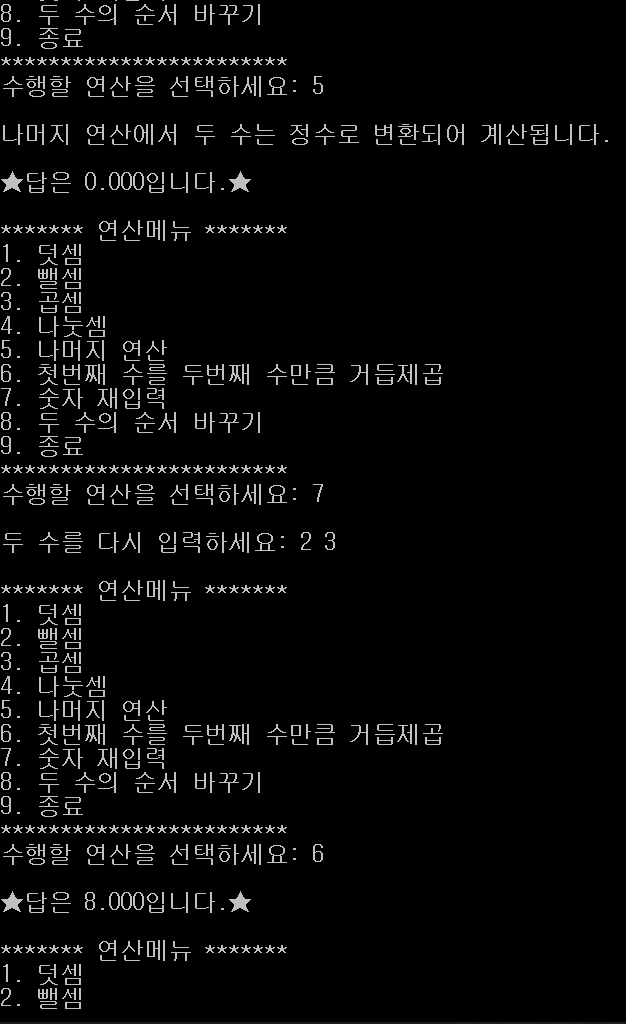
double Pow(double x, double y) // 반환값 및 매개변수를 실수형으로 갖는 거듭제곱 함수 정의

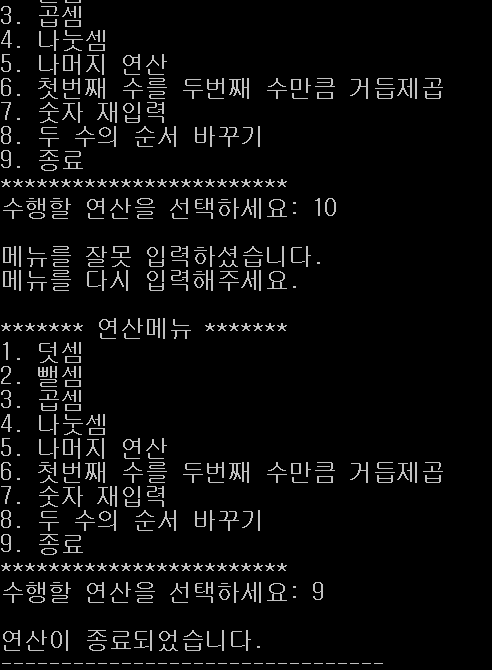
{

return pow(x,y); // x를 y번 거듭제곱하는 함수를 반환

}

* 실행화면





**[과제#3] 로또번호 생성기 프로그램 작성**

* 코드

// 로또번호 생성기 프로그램

// 전처리문

#include <stdio.h> // 표준입출력라이브러리 포함

#include <stdlib.h> // 난수를 발생시키기 위한 표준라이브러리 포함

#include <time.h> // C언어의 표준 라이브러리 중 하나, 시간과 날짜 얻거나 조작하는 함수 포함

/\* 이 프로그램에서 time.h는 srand()함수와 관련하여 시드값을 다르게 줘서 보다 더 완전한 난수를 생성하기 위해 사용된다.

이 밖에도 시간 조작을 하는 함수 clock\_t clock(void); time\_t time(time\_t\*timer); 등이 포함되며 clock\_t는 틱 수를 저장하는 형식, time\_t는 UTC 1970년 1월 1일 0시 0분 0초를 기준으로 하는 초 단위의 시간을 저장하는 형식이다.\*/

// 상수 만들기

#define MAX 45 // 로또 번호는 1부터 45까지이므로 이에 관한 상수를 만듦

#define NUM 6 // 로또 번호는 45개 중 6개를 고를 수 있으므로 이에 관한 상수를 만듦

/\* 기호상수는 가독성을 높여주며 값을 쉽게 변경할 수 있다는 장점이 있다.

기호상수를 만드는 방법에는 두 가지가 있으며 위에서 사용한 것과 같이 전처리문처럼 메인함수 밖에 기호의 값을 정의해주는 방식과 메인함수 안에 const NUM = 6처럼 선언해 주는 방식이 있다.\*/

int get\_quantity(void); // 함수 원형 : 컴파일러에게 함수에 대하여 미리 알리는 것

/\* 함수 정의가 함수 호출보다 먼저 오지 않으므로 반드시 함수 원형을 정의해 주어야 한다.

또한 일반적으로 메인함수가 코드 작성 순서상 앞서기 때문에 대부분의 경우에서 함수 함수 정의가 함수 호출보다 뒤에 위치하므로 함수 원형을 정의하는 습관을 기르는 것이 좋다.\*/

int main(void) //메인 함수 선언부, 반환유형이 정수형이며 매개변수는 없음

{

// 변수 선언

int i; // 반복제어변수

int j; // 반복제어변수

int k; // 반복제어변수

int s; // 반복제어변수

int tmp; // 번호 순서를 정렬할 때 필요한 변수

int n = get\_quantity(); // 사용자로부터 입력받은 구매 로또 개수(함수 호출)를 저장할 변수

int number[NUM]; // 배열 선언, 배열을 사용하여 생성된 임의의 난수를 저장할 변수

/\* get\_quantity();는 앞서 원형을 정의한 함수를 호출한 것이다. 함수를 호출할 때는 반환유형을 빼고 호출한다.

보통의 경우 변수는 오직 하나의 값만을 저장할 수 있으나 배열을 이용하면 많은 값을 한 번에 저장할 수 있다. 자료형 배열이름[배열의 크기];의 형태로 사용되며 앞서 NUM을 6으로 정의했기 때문에 number에는 6개의 변수를 저장할 수 있다.

배열을 선언할 때 주의할 점은 배열 크기를 반드시 지정해야 하며, 이때 배열의 크기는 음수나 실수가 될 수 없다. 또한 변수도 배열의 크기로 사용할 수 없지만 여기서 #define을 사용하여 NUM을 기호 상수로 정의했기 때문에 int number[NUM];의 형태가 가능하다.\*/

srand(time(NULL)); // 난수를 생성할 때 시드값을 불특정하게 주어 난수의 정확도를 높이기 위함

// time 함수는 계속해서 값이 바뀌기 때문에 위의 방식으로 같은 난수가 생성되는 것을 막을 수 있음

// 같은 로또 번호 생성을 막는 반복문

for(j = 1;j <= n;j++){ // 사용자가 구매할 로또의 개수만큼 실행됨

printf("\n%d 번 로또 : ", j); // 사용자가 구매할 로또의 개수만큼 출력됨

for(i = 0;i < NUM;i++){ // NUM은 6이므로 0부터 5까지 총 6번 반복됨

number[i] = 1 + rand() % MAX;// 1부터 45까지의 난수 중 하나를 저장

/\* 반복문을 사용했으므로 i가 0일 때부터 5일 때까지 실행되어 number[i]에 0부터 5까지의 배열요소가 채워지고 총 6개의 난수를 받게 된다.\*/

for(k = 0; k < i; k++){ // 중복 번호를 피하기 위한 반복문, 0부터 i보다 작을 때까지 실행

if(number[i] == number[k]){

i--;

/\* 만약 number[i]가 그전에 받은 값인(혹은 그 값들 중 하나인) number[k]와 같다면 중복된 번호를 피하기 위해 i를 1감소시켜 번호를 다시 받는다. 값이 모두 중복 없이 다르다면 i는 감소되지 않으므로 다음 배열에 저장될 난수를 받는다.\*/

break; // for문을 탈출하여 난수를 다시 받음

}

}

}

// 작은 수부터 정렬하기 위한 반복문

for(i = 0; i < NUM; i++){ // 6번 실행됨

// number[i]가 그 뒤에 받은 번호들보다 클 때 순서를 바꿔 줄 반복문

for(s = i + 1; s < NUM; s++){ // 5번 실행됨

if(number[i] > number[s]){ // 만약 number[i]가 그 다음에 받은 번호인 number[s]보다 크다면

tmp = number[i]; // tmp에 큰 수를 저장

number[i] = number[s]; // number[i]에 작은 수를 저장

number[s] = tmp; // 배열요소가 더 큰 번호에 큰 수를 저장

}

}

printf("%d ", number[i]); // i가 0부터 5까지 number[i]를 출력

/\* 앞서 배열요소가 작을 때 가장 작은 수가 저장되어 있도록 처리를 했기 때문에 number[0]<number[1]<number[2]<number[3]<number[4]<number[5]이므로 i가 0일 때부터 5일 때까지 순서대로 반복출력하면 결과적으로 작은 수부터 출력이 이루어진다.\*/

}

}

return 0; // 메인 함수에 0을 반환하여 끝내기

}

// 로또 구매 개수 입력받는 함수

int get\_quantity(void) // 함수 정의, 반환유형은 정수이며 매개변수는 없음

{

int n; // 사용자로부터 입력받을 수를 저장할 변수 선언

printf("구매할 로또의 개수를 입력하세요 : "); // 구매 개수 입력 안내하기

scanf("%d", &n); // 구매 개수 입력 받기

return n; // 입력받은 정수를 반환하기

// 반환값이 n이라는 것은 함수가 호출한 곳으로 반환하는 작업의 결과값이 사용자에게 입력받은 n값이라는 것이다.

}

* 실행화면

