|  |
| --- |
| 2020년 1학기 |
| 프로그래밍 언어 레포트 |
| 10주차 학습활동 |

|  |
| --- |
| 송 지민  사회학과 201721899 |

***[과제#1] 실습#2, 실습#3 실습수행 및 실행화면 캡처***

1. 실습#2 코드 및 실행화면

**#include <stdio.h>**

**#define SIZE 10**

**int main(void)**

**{**

**char reserve = 0;**

**int i;**

**int num=0;**

**int seats[SIZE] = {0};**

**while(1){**

**printf("좌석을 예약하시겠습니까?(y 또는 n)");**

**scanf(" %c", &reserve);**

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

**if(reserve == 'y')**

**{**

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

**printf("1 2 3 4 5 6 7 8 9 10\n");**

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

**for(i = 0;i < SIZE;i++)**

**{**

**printf("%d ", seats[i]);**

**}**

**printf("\n\n몇번째 좌석을 예약하시겠습니까?");**

**scanf("%d", &num);**

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

**if(num <= 0 || num > SIZE)**

**{**

**printf("1부터 10사이의 숫자를 입력하세요\n\n");**

**continue;**

**}**

**if(seats[num-1] == 1)**

**{**

**printf("이미 예약된 좌석입니다.\n");**

**printf("다른 좌석을 선택하세요.\n\n");**

**continue;**

**}**

**else**

**{**

**seats[num-1] = 1;**

**printf("예약되었습니다.\n\n");**

**}**

**}**

**else if(reserve == 'n')**

**break;**

**else if(reserve != 'n' && reserve != 'y')**

**{**

**printf("y 또는 n을 입력해주세요.\n\n");**

**continue;**

**}**

**}**

**printf("예약을 종료합니다.");**

**return 0;**

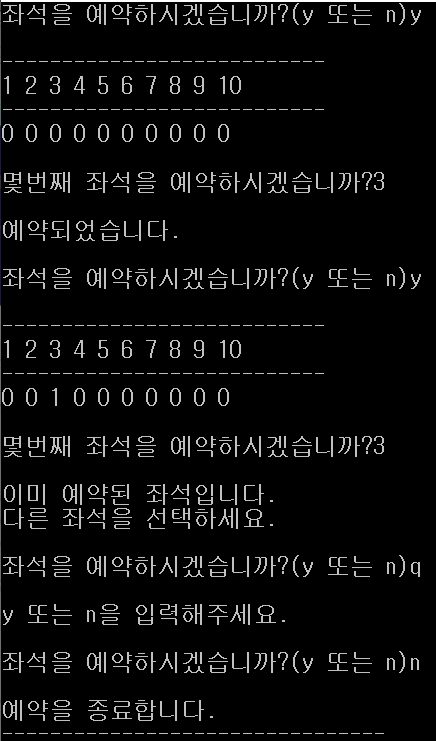
**}**

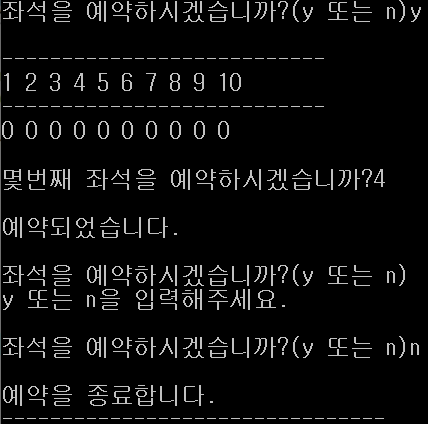
왼쪽은 정상 실행화면으로, 코드에서 형광펜으로 표시한 부분, 즉 scanf(“ %c”, &reserve);

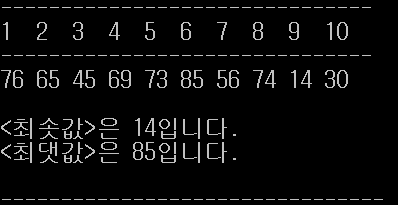
코드를 작성했을 때의 화면이다.

아래 실행화면은 코드에서 형광펜 표시 부분을 scanf(“%c”, &reserve); 으로 변경했을 때의 오류 실행화면이다.

이것으로 미루어보아, 문자형 변수 자료 입력 시에 공백문자가 입력에 영향을 주는 것을 알 수 있다.

****

****

1. 실습#3 코드 및 실행화면

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#define SIZE 10

int main(void)

{

int prices[SIZE] = {0};

int i, j;

int min = 0;

int max = 0;

printf("-------------------------------\n");

printf("1 2 3 4 5 6 7 8 9 10\n");

printf("-------------------------------\n");

srand(time(NULL));

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

{

prices[i] = rand()%100 + 1;

printf("%-3d", prices[i]);

}

min = prices[0];

for(i = 1; i < SIZE; i++)

{

if(prices[i] < min)

{

min = prices[i];

}

}

max = prices[9];

for(i = 0; i < SIZE-1; i++)

{

if(prices[i] > max)

{

max = prices[i];

}

}

printf("\n\n");

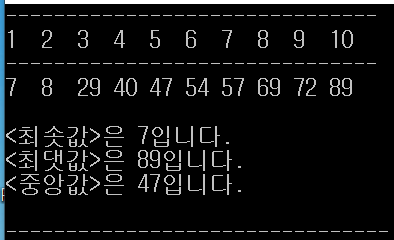
printf("<최솟값>은 %d입니다.\n", min);

printf("<최댓값>은 %d입니다.\n", max);

return 0;

}

* 다른 방식으로 최솟값, 최댓값 찾고 중앙값 출력 추가한 코드

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#define SIZE 10

int main(void)

{

int prices[SIZE] = {0};

int i, j;

int max = 0;

printf("-------------------------------\n");

printf("1 2 3 4 5 6 7 8 9 10\n");

printf("-------------------------------\n");

srand(time(NULL));

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

{

prices[i] = rand()%100 + 1;

}

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

{

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

if(prices[i] > prices[j])

{

max = prices[i];

prices[i] = prices[j];

prices[j] = max;

}

printf("%-3d", prices[i]);

}

printf("\n\n");

printf("<최솟값>은 %d입니다.\n", prices[0]);

printf("<최댓값>은 %d입니다.\n", prices[9]);

printf("<중앙값>은 %d입니다.\n", prices[4]);

return 0;

}

***[과제#2] 배열과 난수를 이용한 프로그램 작성***

1. 첫번째 프로그램 코드 및 실행화면

// 최솟값, 최댓값, 중앙값, 평균값 구하는 첫번째 프로그램

#include <stdio.h> // 표준입출력 헤더파일 선언

#include <stdlib.h> // 표준라이브러리 선언

#include <time.h> // 날짜 및 시간을 조작하는 함수가 정의된 헤더 파일

#define SIZE 10 // 기호상수 선언

/\* 배열의 크기를 기호상수로 선언하면 훗날 코드를 수정, 보완할 때 상수값을 바꿔주기만 하면 되므로 더욱 용이하다.\*/

int main(void) // 반환형이 정수이고 매개변수가 없는 메인 함수의 선언부

{

// 변수선언

int rand\_num[SIZE] = {0}; // 배열의 크기가 10인 정수형 배열 선언 및 초기화

/\* 배열은 자료형 배열이름[배열요소의 개수]형태로 선언하며, 배열을 정수형으로 선언했기 때문에 배열에 들어가는 값 또한 정수형이어야 한다.

배열을 초기화할 때는 모든 값을 0으로 입력하지 않아도 별도로 초기값을 입력하지 않으면 값이 0으로 초기화된다.

이는 초기값을 0으로 주고 싶을 때는 간편하게 쓸 수 있지만, 그렇지 않을 때는 선언과 동시에 배열의 값을 초기화하는 것이 에러 방지에 도움이 될 것으로 보인다.\*/

int i, j; // 반복 제어 변수 선언

int max = 0; // 난수를 순서대로 나열할 때 큰 수를 저장할 변수

int sum = 0; // 난수의 합을 저장할 변수

int avg = 0; // 난수의 평균을 저장할 변수

/\* 변수 선언 시에는 쓰레기값의 저장을 막기 위해 초기화하는 습관을 들이는 것이 좋다.\*/

srand(time(NULL)); // rand()함수의 시드값을 랜덤하게 주기 위한 함수

/\* 시드값을 따로 지정하지 않으면 rand()함수는 기본값 0을 시드값으로 지정하고, 이 때 rand()함수 값은 모두 같아진다.

srand()함수는 rand()함수의 시드값을 지정해주는 함수이며, time(NULL)을 시드값으로 줌으로써 비교적 완벽한 난수를 받아오게 된다.

srand(time(NULL))에서 NULL은 메모리의 어떤 유효한 위치의 개체도 가리키지 않는 포인터 값이다.

주의할 점은 time(NULL)은 시간 변화를 초 단위로 반영하기 때문에 1초가 지나기 전에 계속해서 호출하면 동일한 값이 나온다는 것이다.\*/

// 랜덤값 주기

for(i = 0; i < SIZE; i++) // 0부터 9까지 10번 반복됨

{

rand\_num[i] = rand(); // rand()함수를 통해 난수값을 받아 배열에 넣음

/\* 배열 인덱스 i는 반복문에 따라 0부터 9까지 증가하며 차례대로 난수를 받게 된다.

이 때 인덱스는 0부터 시작함을 주의해야 하며 따라서 크기가 10인 배열의 인덱스 i는 1이 아닌 0부터 시작하고 10이 아닌 9가 마지막 인덱스가 된다.

또한 배열은 주로 반복문과 함께 사용되며, 일반적으로 반복문이 실행되는 회수는 배열의 크기와 일치한다.\*/

}

printf("생성된 난수 : "); // printf()함수 이용하여 출력하기

// 배열을 작은 수부터 정렬하여 출력하기

for(i = 0; i < SIZE; i++) // 0부터 9까지 10번 반복됨

{

for(j = i + 1; j < SIZE; j++) // i+1부터 9까지 반복됨

if(rand\_num[i] > rand\_num[j]) // 만약 rand\_num[i]가 rand\_num[j]보다 크면

{

max = rand\_num[i]; // 큰 값을 max에 저장

rand\_num[i] = rand\_num[j]; // 작은 값을 rand\_num[i]에 저장

rand\_num[j] = max; // rand\_num[j]에 큰 값을 저장

}

printf("%d ", rand\_num[i]); // 배열의 인덱스 0부터 9까지 난수 출력

}

/\* 반복문을 이용해 작은 수부터 큰 수까지 차례대로 나열하는 코드를 작성하였다.

예를 들어 i가 0일 때는 내부 반복문에서 rand\_num[0]와 rand\_num[1]과 비교하여 rand\_num[0]값이 더 크면 배열의 순서를 바꾸는데 이것을 인덱스가 i보다 1큰 수부터 9까지 반복하여 전부 비교한다.

그러한 과정을 거치면, 인덱스가 0인 배열에 최솟값이 저장되므로 이를 출력하여 순서대로 보이게 한다.\*/

// 난수의 합 구하기

for(i = 0; i < SIZE; i++) // 0부터 9까지 10번 반복

{

sum += rand\_num[i]; // sum = sum+rand\_num[i]와 동일, 배열의 총합

}

avg = sum/10; // 배열의 총합을 배열요소의 개수로 나눠 평균을 구함

printf("\n\n"); // 줄바꿈

printf("<최솟값>은 %6d입니다.\n", rand\_num[0]); // 최솟값은 가장 첫번째 배열값인 rand\_num[0]이므로 이를 출력

printf("<최댓값>은 %6d입니다.\n", rand\_num[9]); // 최댓값은 가장 마지막 배열값인 rand\_num[9]이므로 이를 출력

printf("<중앙값>은 %6d입니다.\n", rand\_num[4]); // 짝수일 때 중앙값은 n/2 번째의 값이므로 5번째 값인 rand\_num[4]가 중앙값

printf("<평균값>은 %6d입니다.\n", avg); // 난수의 평균을 출력

return 0; // 0을 반환하여 메인 함수 끝내기

}



1. 두번째 프로그램 코드 및 실행화면

// 최솟값과 최댓값을 구하는 두번째 방식

#include <stdio.h> // 표준입출력 헤더파일 선언

#include <stdlib.h>

#include <time.h>

#define SIZE 10

int main(void)

{

int rand\_num[SIZE] = {0};

int i;

int min = 0; // 최솟값을 구하기 위한 변수

int max = 0; // 최댓값을 구하기 위한 변수

int sum = 0;

int avg = 0;

srand(time(NULL));

printf("생성된 난수 : "); // 첫번째 코드와 달리 생성된 난수를 난수를 받기 전에 출력

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

{

rand\_num[i] = rand();

printf("%d ", rand\_num[i]);

}

min = rand\_num[0]; // 첫번째 배열값이 최소라고 가정

// 최솟값을 구하기 위한 반복문

for(i = 1; i < SIZE; i++) // i가 1일 때부터 9일 때까지 9번 반복

if(min > rand\_num[i]) // 최솟값으로 가정한 값이 다른 배열값보다 더 클 때

min = rand\_num[i]; // 최솟값에 작은 값을 저장

max = rand\_num[9]; // 10번째 배열값이 최대라고 가정

// 최댓값을 구하기 위한 반복문

for(i = 0; i < SIZE-1; i++) // i가 0일 때부터 8이 때까지 9번 반복

if(max < rand\_num[i]) // 최댓값으로 가정한 값이 다른 배열값보다 더 작을 때

max = rand\_num[i]; // 최댓값에 큰 값을 저장

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

{

sum += rand\_num[i];

}

avg = sum/10;

printf("\n\n");

printf("<최솟값>은 %6d입니다.\n", min); // 최솟값 출력

printf("<최댓값>은 %6d입니다.\n", max); // 최댓값 출력

printf("<평균값>은 %6d입니다.\n", avg);

return 0;

}



***[과제#3] 2차원 배열을 이용한 합계 프로그램***

1. 기능 추가 프로그램 코드 및 실행화면 : 배열 값을 난수로 줌, 각 행과 열의 합계 및 평균 계산 기능 추가

// 2차원 배열을 이용한 합계 프로그램

#include <stdio.h> // 표준입출력 헤더파일 선언

#include <stdlib.h> // 표준 라이브러리 선언

#include <time.h> // 시간 및 날짜 함수 헤더파일 선언

#define ROWS 3 // 2차 배열의 행 크기 기호상수로 선언

#define COLS 5 // 2차 배열의 열 크기 기호상수로 선언

/\* 기호상수로 배열의 크기를 선언하면 유지, 보수에 용이하며 알아보기 쉬우며 #define ROWS = 3과 같이 대입연산자를 쓰지 않도록 주의해야 한다.\*/

int main(void) // 반환형이 정수형이고 매개변수가 없는 메인 함수 선언부

{

int i, j; // 반복제어변수 선언

int sum = 0; // 난수 합 저장할 변수 선언 및 초기화

int sumsum[5] = {0}; // 열의 합 저장할 배열 선언 및 초기화

int avg = 0; // 난수의 평균 저장할 변수 선언 및 초기화

int data[ROWS][COLS] = {0}; // 행이 3, 열이 5인 2차원 배열 선언 및 초기화

srand(time(NULL)); // rand()함수 시드값 주기

/\* rand()함수는 시드값을 기준으로 하여 난수를 생성하기 때문에 시드값이 중요한데, 여기서 time(NULL)은 rand()함수가 더욱 완벽한 난수를 생성할 수 있게 한다.\*/

// 난수를 2차원 배열에 저장하기 위한 반복문

for(i = 0; i < ROWS; i++) // 0에서 2까지 3번 반복

{

for(j = 0; j < COLS; j++) // 0에서 4까지 5번 반복

{

data[i][j] = rand()%100+1; // 1에서 100까지의 값 중 랜덤한 값을 저장

}

}

/\* 선 형태의 1차원 배열과 달리 2차원 배열은 가로와 세로가 존재하는 평면 형태를 지니며, 선언 시에 행과 열의 크기를 선언해야 한다.

여기서는 3행 5열의 2차원 배열을 선언하였는데, 이것을 각 층에 3가구가 거주하는 5층 아파트로 생각해볼 수 있다.

또한 선언과 동시에 초기화하는 경우, 행의 개수는 지정하지 않을 수 있으며 이 경우 컴파일러가 자동으로 행의 개수를 결정한다.

반면 열의 개수는 반드시 지정해야 함을 주의한다.\*/

printf(" 1 2 3 4 5 합 평균\n"); // 열의 번호, 행의 합, 행의 평균 알림

printf("-----------------------------------\n");

// 전체 행열을 출력하고 행의 합, 평균 계산 및 출력

for(i = 0; i < ROWS; i++) // 0부터 2까지 3번 반복

{

printf(" %d|", i+1); // 행의 번호를 출력, 인덱스가 0부터이므로 +1해줌

for(j = 0; j < COLS; j++) // 0부터 4까지 5번 반복

{

sum += data[i][j]; // 행의 총합을 계산

printf(" %3d", data[i][j]); // 행과열을 고려하여 배열값 출력

}

printf(" %4d", sum); // 행의 총합을 출력

avg = sum/5; // 행의 총합을 열의 개수로 나눠 평균 구하기

printf(" %3d\n", avg); // 행의 총합 출력하기

sum = 0; // 총합을 0으로 만들어주기

}

/\* 각 행마다 한 번만 출력하는 것은 내부 반복문의 바깥에 쓰고 행의 열마다 출력하는 내용은 내부 반복문에 작성한다.

따라서 각 행의 번호, 각 행의 총합, 각 행의 평균은 내부 반복문의 바깥에 쓰고 행의 열 값을 다 더해야 하는 총합 계산, 전체 출력은 내부 반복문에 작성한다.\*/

printf(" 합|"); // 열의 합임을 알림

// 각 열의 합 계산및 출력하는 반복문

for(i = 0; i < COLS; i++) // 0부터 4까지, 5번 반복

{

for(j = 0; j < ROWS; j++) // 0부터 2까지, 3번 반복

{

sumsum[i] += data[j][i]; // 열의 행을 누적하여 더해서 배열에 저장함

}

printf(" %3d", sumsum[i]); // 합계 출력하기

}

/\* 출력 결과가 선형적으로 나타나는 것이 아니기 때문에 열의 합과 평균은 같은 반복문 내에서 계산 및 출력이 불가하다.

따라서 따로 계산, 출력하기 위해 각 열의 행의 총합을 배열에 저장해서 평균을 계산할 때 그 값을 5로 나누기 위해 배열을 선언하였다.

여기서 열의 행의 총합은 각 열마다 있으므로 모두 5개의 값이 나온다. 따라서 인덱스는 i로 주었다.

또한 열의 행을 더하기 위해 data[j][i]와 같은 형태로 작성하였다.\*/

printf("\n"); // 줄바꿈

printf("평균|"); // 열의 평균임을 알림

// 각 열의 평균 계산 및 출력하는 반복문

for(i = 0; i < COLS; i++) // 0부터 4까지 총 5번 반복

{

avg = sumsum[i]/3; // 열의 행의 총합을 행의 수로 나누어 평균 구하기

printf(" %3d", avg); // 평균 출력하기

}

/\* 앞에서 저장했던 열의 행 총합을 반복문을 통해 행의 수로 나누어 평균을 구하고 이를 출력한다.

여기서는 열마다 한번씩 출력하면 되기 때문에 굳이 중첩 반복문을 쓸 필요는 없다.\*/

return 0; // 0을 반환하여 끝내기

}



1. 과제#3 기본 프로그램 코드 및 실행화면

// 2차원 배열 이용한 합계 프로그램(기본)

#include <stdio.h>

#define ROWS 3

#define COLS 5

int main(void)

{

int i, j;

int sum = 0;

int data[ROWS][COLS] = {{12,56,32,16,98},{99,56,34,41,3},{65,3,87,78,21}};

// 배열 선언과 동시에 초기화

// 행의 합계 계산 및 출력

for(i = 0; i < ROWS; i++) // 0부터 2까지 3번 반복

{

printf("%d행의 합계 : ", i+1); // 행의 번호를 출력, 인덱스가 0부터이므로 +1해줌

for(j = 0; j < COLS; j++) // 0부터 4까지 5번 반복

{

sum += data[i][j]; // 행의 총합을 계산

// 예를 들어, 행이 0일 때 0열부터 4열까지 누적하여 합함

}

printf("%3d\n", sum); // 행의 총합을 출력

sum = 0; // 총합을 0으로 만들어주기

// 0으로 만들어주지 않으면 다음 반복에서 누적되어 있던 합계에 또 누적하여 더함

}

// 각 열의 합 계산및 출력하는 반복문

for(i = 0; i < COLS; i++) // 0부터 4까지, 5번 반복

{

printf("%d열의 합계 : ", i+1); // 열의 번호 출력, 인덱스 0부터이므로 +1해주기

for(j = 0; j < ROWS; j++) // 0부터 2까지, 3번 반복

{

sum += data[j][i]; // 열의 행을 누적하여 더해서 배열에 저장함

// 예를 들어 열이 0일 때 0,1,2행의 값을 누적하여 합함

}

printf("%3d\n", sum); // 합계 출력하기

sum = 0; // 총합 0으로 초기화하기

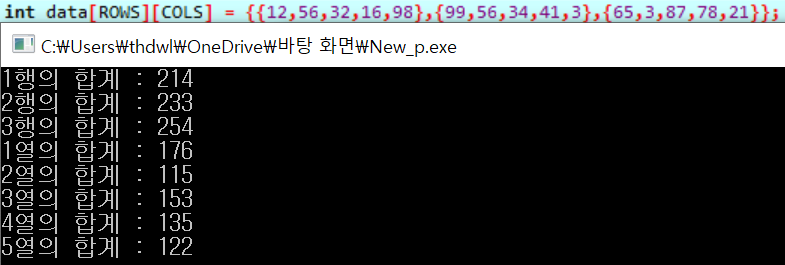
// 마찬가지로, 0으로 초기화하지 않으면 다음 반복에서 누적 합계에 또 다시 누적하여 더함

}

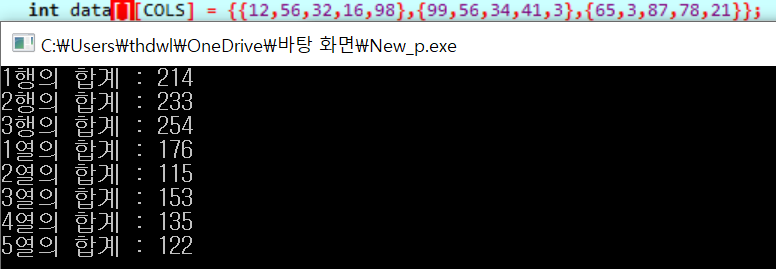
return 0; // 0을 반환하여 끝내기

}

초기화할 때 행의 크기를 쓴 경우의 실행화면



초기화할 때 행의 크기를 쓰지 않은 경우의 실행화면



초기화할 때 열의 크기를 쓰지 않자, 컴파일 에러 발생함을 확인



