윈도우 프로그래밍 워밍업 2

2024년도 1학기

3. [2차원 배열 사용하기] 4x5 행렬 만들고 행렬의 연산 구현하기

- 4행 5열의 행렬을 만들고 숫자(중복 불가)를 행렬에 저장한다.
  - 1 ~ 50 사이의 랜덤한 숫자를 행렬에 자동 입력
  - 열의 위치를 맞춰서 출력한다.

- 다음의 명령어를 수행한 후 결과를 출력한다. 모든 명령어는 다시 누르면 실행이 취소되고 원래의 값으로 변경된다.
  - a: 1행1열을 시작으로 **행을 따라 오름차순으로 정렬**
  - d: 1행5열을 시작으로 **열을 따라 내림차순으로 정렬**
  - e: 짝수만 출력하기
  - o: 홀수만 출력하기
  - m: 최대값을 찾아 모든 값에 더한 값을 출력
  - n: 최소값을 찾아 모든 값에서 뺀 값을 출력
  - p: 한 행의 값을 모두 더해 각 행의 1열에 저장
  - u: 한 열의 값을 모두 곱해 **각 열의 4행에 저장 (수정 부분) (→ 기존의방법인 5행을 새롭게 만들어 출력해도 무관함)**
  - r: 기존의 숫자를 랜덤하게 재정렬
  - s: 새로운 랜덤한 숫자를 다시 자동 입력 (리셋)
  - q: 프로그램 종료

3. [2차원 배열 사용하기] 4x5 행렬 만들고 행렬의 연산 구현하기

결과 예)

행렬값:

3	10	24	11	18
39	41	5	17	22
1	29	28	13	20
25	14	33	43	50

명령어 입력 a:

1	3	5	10	11
13	14	17	18	20
22	24	25	28	29
33	39	41	43	50

명령어 입력 d:

10	17	24	33	50
5	14	22	29	43
3	13	20	28	41
1	11	18	25	39

명령어 입력 e:

0	10	24	0	18
0	0	0	0	22
0	0	28	0	20
0	14	0	0	50

명령어 입력 m:

53	60	74	61	68
89	91	55	67	72
61	79	78	63	70
75	64	83	93	100

명령어 입력 r:

10	41	18	3	29
5	28	43	25	14
33	17	24	11	50
39	22	13	20	1

명령어 입력 p:

101	41	18	3	29
152	28	43	25	14
135	17	24	11	50
95	22	13	20	1

4. [구조체 사용하기] 텍스트를 사용하여 2인 돌 이동하기

- 화면에 10x10 바둑판을 그린다. 바둑판의 임의의 위치에 2개의 다른 돌을 놓고 사용자의 명령어(wasd 와 ijkl)에 따라 돌을 좌/우/상/하로 번갈아 입력 받아 움직인다.
  - 두 종류의 다른 돌은 번갈아 가며 움직일 수 있다.
  - 두 사용자의 이동 명령어는 두 종류의 다른 조합을 사용한다. (wasd와 ijkl)
- 가장자리에 돌이 닿으면 반대편 가장자리에 나타난다. 반대편에 나타날 때 돌의 모양이 바뀐다.
  - 돌의 모양은 5개 중 한 개로 랜덤하게 선택한다: 5개의 돌 모양 예) o, x, #, @, *
- 움직이는 돌이 다른 돌과 부딪치면
  - 움직인 돌 (위의 돌)의 모양만 나타난다. 이때 비프 소리를 출력한다.
  - 다음 순서인 아래의 돌을 움직이게 되면 바뀐 모양으로 나타난다.

- 돌의 위치와 상태를 가지고 있는 구조체를 사용한다.
- 키보드 명령으로 이동 키보드 외에 프로그램 리셋/종료 명령을 추가한다.
  - s: 프로그램 리셋 (시작위치에서 새로운 돌 모양으로 시작)
  - q: 프로그램 종료

** 비프음 소리내기: 함수 Beep 사용

```
#include <windows.h>
Beep (음의 높이, 음의 지속시간);
```

4. [구조체 사용하기] 텍스트를 사용하여 2인 돌 이동하기

결과 예)

```
--------------------------------
|  |  |x|  |  |  |  |  |  |  |
--------------------------------
|  |  |  |  |O|  |  |  |  |  |
--------------------------------
...
```

Input command: w (원이 위로 이동)

```
-----------------------  ---------
|  |  |x|  |O|  |  |  |  |  |
-----------------------  ---------
|  |  |  |  |  |  |  |  |  |  |
--------------------------------
...
```

Input command: j (x가 좌측 이동)

```
 -----------------------------
|  |x|  |  |O |  |  |  |  |  |
--------------------------------
|  |  |  |  |  |  |  |  |  |  |
--------------------------------
...
```

Input command: d (원이 우측으로 이동)

```
-----------------------  ---------
|  |x|  |  |  |O|  |  |  |  |
-----------------------  ---------                    ...
|  |  |  |  |  |  |  |  |  |  |
--------------------------------
...
```

Input command: a (O가 왼쪽으로 가니 x와 부딪치게 됨. O이 x위에 올려져서 O만 출력된다.)

```
 -----------------------------
|  |  |  |x |O |  |  |  |  |  |
--------------------------------
|  |  |  |  |  |  |  |  |  |  |
--------------------------------
```

```
-----------------------  ---------
|  |  |  |O|  |  |  |  |  |  |
-----------------------  ---------
|  |  |  |  |  |  |  |  |  |  |
--------------------------------
```

Input command: k (다음 순서로 x를 아래로 움직이면, x의 모양이 다른 모양으로 바뀌어 나타난다)

```
 -----------------------------
|  |  |  |O |  |  |  |  |  |  |
--------------------------------
|  |  |  |@|  |  |  |  |  |  |
--------------------------------
```

.....

5. [조건문 사용하기] 109명의 대의원 선출하기

- **109명의 대의원 선출하기**
  - 13개의 반이 있고 각 반을 대표하는 대의원을 선출한다.
    - 13개의 반의 대의원 수는 6~13 사이이고 13개 반의 대의원의 총 합은 109이다.
    - 13개의 대의원의 수는 각각 9/12/9/13/10/6/10/6/9/5/6/7/7으로 앞의 13개의 숫자를 13개의 반에 랜덤하게 배치한다.
  - 사용자는 대의원 후보자 숫자 (N)를 입력한다.
    - 대의원 후보자 수는 150 ~ 250명 이다.
    - 각 대의원 후보는 13개의 반 중 한 개의 반으로 정한다. 랜덤하게 설정한다.
  - 이제 1000명의 인원이 대의원 후보 중에서 각자 109명을 뽑는다.
    - 2가지 방식으로 대의원을 뽑는다.
      1) 각 반별로 설정된 대의원 수만큼 뽑는다. 만약 어떤 반의 대의원 후보가 대의원 수보다 작으면, 남은 수의 대의원은 탈락된 대의원에서 다득표 순으로 뽑는다. 즉, 1반의 대의원 수가 9명인데 후보가 7명만 있다면, 나머지 2명은 탈락된 대의원에서 다득표 순으로 2명을 뽑는다.
      2) 전체 다득표 순으로 대의원 수만큼 뽑는다.
  - 명령어를 실행한다.
    - p: 대의원 수를 각 반에 배정하고 배정된 숫자를 화면에 출력하고, 1000명의 인원이 각각 109명의 대의원을 뽑는다.
    - **1~1000: 입력한 번호의 인원이 뽑은 대의원 번호와 그 대의원의 반 번호를 출력한다. (명령어 추가)**
    - a: 반 별로 투표를 받은 대의원의 후보의 번호를 오름차순으로 정렬하여 출력한다.
    - d: 반 별로 투표를 받은 대의원의 후보의 번호를 내림차순으로 정렬하여 출력한다.
    - m: 반 별로 당선된 대의원의 번호를 출력한다. **(1번 방식으로 선출된 대의원 출력)**
    - n: 전체 다득표 순으로 109명의 대의원의 번호를 출력한다. **(2번 방식으로 선출된 대의원 출력)**
    - r: 위의 과정을 다시 시작한다.
    - q: 프로그램 종료

5. [조건문 사용하기] 109명의 대의원 선출하기

- **예)**
  - 대의원 후보자 숫자 입력: 200
    - 출력: 1번 – 1반, 2번 – 3반, 3번 – 10반, 4번 - 7반, 5번 – 9반, 6번 – 1반, …, 10번 – 6반

      …

      191번 – 2반, 192번 – 2반, 193번 – 7반, ….. 200번 – 12반
  - 명령어: p (*대의원 수를 출력*)
    - 1반 – 9명, 2반 – 12명, 3반 – 9명, 4반 – 13명, … 12반 – 7명, 13반 – 7명
  - 명령어: d (*각 반별 후보자들을 모두 출력*)
    - 1반: 15번 – 50표, 27번 – 42표, 93번 – 41표,… (1반의 후보자들의 득표수에 따라 내림차순으로 출력)
    - 2반: 83번 – 83표, 1번 – 65표, … (2반의 후보자들의 득표수에 따라 내림차순으로 출력)
    - …
    - 13반: 72번 – 101표, 16표 – 95표, … (13반의 후보자들의 득표수에 따라 내림차순으로 출력)
  - 명령어: m (*각 반별 후보자들 중 당선자만 출력*)
    - 1반: 15번 – 50표, 27번 – 42표, 93번 – 41표,… (1반의 후보자들의 득표수에 따라 9명을 내림차순으로 출력)
    - 2반: 83번 – 83표, 1번 – 65표, … (2반의 후보자들의 득표수에 따라 12명을 내림차순으로 출력)
    - …
    - 13반: 72번 – 101표, 16표 – 95표, … (13반의 후보자들의 득표수에 따라 7명을내림차순으로 출력)
  - 명령어: n (*전체 다득표순으로 당선자 출력, 당선자 번호와 득표수, 반 이름을 같이 출력*)
    - 72번 – 101표 13반, 16번 – 95표 13반, 83번 - 83표 2반… (이런 식으로 109명을 출력한다.

6. [함수 사용하기] 계산 순위 정하여 정해진 순위 따라 연산하기

- 사용자가 입력하는 숫자 5개와 연산자 4개를 랜덤하게 설정한다.
  - 숫자의 개수는 더 많게는 변경 가능, 숫자가 변경되면 연산자수도 같이 변경된다.
- 숫자는 한자리 또는 두 자리의 정수다.
- 사용자의 명령어를 입력 받아 우선순위대로 또는 순서대로 또는 역 순서로 계산하여 출력한다.
  - 명령어 1: 연산자 우선순위로
  - 명령어 2: 순서대로
  - 명령어 3: 역 순서대로
  - 명령어 4: 새로 숫자와 연산자 설정
  - 명령어 q: 프로그램 종료

결과 예)

```
Numbers to calculate:          3+4*5*2-1
Input the calculating order: 1
Result: 3 + ((4*5)*2) - 1 = 42              //--- 연산자의 우선순위대로 계산한다.
Input the calculating order: 2
Result: (((3+4)*5)*2)-1 = 69                //--- 앞에서부터 순서대로 계산한다.
Input the calculating order: 3
Result: 3+(4*(5*(2-1))) = 23                //--- 뒤에서부터 계산한다.
Input the calculating order: 4
Numbers to calculate:          3+6/4*2+5
Input the calculating order: 1
Result: 3 + ( (6/4) *2) + 5 = 11
...
```

워밍업 프로그램은

- **문제 당 채점 기준**
  - 1문제당 3점으로 계산하여 적용됨
  - 각 문제당 100% 이상 구현했을 때: 3점
  - 각 문제당 50% 이상 구현했을 때: 2점
  - 각 문제당 30% 이상 구현했을 때: 1점

- **한 학기를 위한 워밍업이니 최대한 스스로의 힘으로 구현하세요!**