Warming-up 프로그램

2021-2 컴퓨터 그래픽스

1. 행렬 다루기

- 3 X 3 행렬의 덧셈, 뺄셈, 곱셈하는 프로그램을 구현한다.
- 연산을 할 2개의 행렬에는 0 또는 1의 값을 랜덤하게 설정한다.
- 다음의 명령어로 수행한다. 종료 명령어를 입력할 때까지 명령을 연속적으로 수행할 수 있도록 한다.
  - m: 행렬의 곱셈
  - a: 행렬의 덧셈
  - d: 행렬의 뺄셈
  - r: 행렬식의 값 (Determinant) ➔ 입력한 2개의 행렬의 행렬식 값을 모두 출력한다.
  - t: 전치 행렬(Transposed matrix)과 그 행렬식의 값 ➔ 입력한 2개의 행렬에 모두 적용한다.
  - h: 3X3 행렬을 4X4 행렬로 변환하고 행렬식의 값 (4행4열 행렬식 값) 출력
    (단위 행렬의 값, 즉, 4행에 0을, 4열에 0을, 4행 4열의 위치에는 1을 추가한다)
  - s: 행렬의 값을 새로 랜덤하게 설정한다.
  - q: 프로그램 종료

실행 예)

(명령어 입력) m: $\begin{vmatrix}0&0&1\\1&1&1\\0&1&0\end{vmatrix} \cdot \begin{vmatrix}1&0&1\\0&0&0\\0&1&0\end{vmatrix} = \begin{vmatrix}0&1&0\\1&1&1\\0&0&0\end{vmatrix}$

(명령어 입력) a: $\begin{vmatrix}0&0&1\\1&1&1\\0&1&0\end{vmatrix} + \begin{vmatrix}1&0&1\\0&0&0\\0&1&0\end{vmatrix} = \begin{vmatrix}1&0&2\\1&1&1\\0&2&0\end{vmatrix}$

(명령어 입력) r: $\begin{vmatrix}0&0&1\\1&1&1\\0&1&0\end{vmatrix} = 1$ $\begin{vmatrix}1&0&1\\0&0&0\\0&1&0\end{vmatrix} = 0$

(명령어 입력) t: $\begin{vmatrix}0&0&1\\1&1&1\\0&1&0\end{vmatrix} \rightarrow \begin{vmatrix}0&1&0\\0&1&1\\1&1&0\end{vmatrix} = 1$ $\begin{vmatrix}1&0&1\\0&0&0\\0&1&0\end{vmatrix} \rightarrow \begin{vmatrix}1&0&0\\0&0&1\\1&0&0\end{vmatrix} = 0$

(명령어 입력) h: $\begin{vmatrix}0&0&1\\1&1&1\\0&1&0\end{vmatrix} \rightarrow \begin{vmatrix}0&0&1&0\\1&1&1&0\\0&1&0&0\\0&0&0&1\end{vmatrix} = 1$ $\begin{vmatrix}1&0&1\\0&0&0\\0&1&0\end{vmatrix} \rightarrow \begin{vmatrix}0&0&1&0\\1&1&1&0\\0&1&0&0\\0&0&0&1\end{vmatrix} = 0$

2. 파일에서 데이터 읽기

- 문자와 숫자가 10줄 저장된 파일을 만든다.
- 데이터 파일 이름을 입력 받고, 파일에서 데이터를 읽어 문자열에 저장한다.
- 읽은 문자열을 출력하고, **공백을 기준으로** 단어의 개수를 세고, 숫자는 따로 출력하고 개수를 센다.
  - 알파벳이나 숫자 외의 특수문자들은 (예, / , - * 같은 문자)는 구분하지 않고 연결된 값으로 카운트 한다.
  - 숫자와 문자가 연결된 문자인 경우에는 숫자로 카운트 하도록 한다.

예) input data file name: ***data.txt***

This is computer graphics warming up program.
1 2 3 4 5 6 7 8 9 10
File input output sample program
3D computer graphics
2021-ab
Korea Polytechnic University
Game and multimedia engineering department
1001ada_ada1001
Monday Tuesday Wednesday Thursday Friday
abcdefghidcba

word count: 28
number count: 13 (1 2 3 4 5 6 7 8 9 10 3D 2020-ab 1001ada_ada1001)

3. 문자열 다루기 (2번 문제 확장)

- 2번 문제같이 파일에서 문자열들을 입력받는다.
- 다음의 명령어를 실행한다.
  - d: 문장 전체를 뒤집기
  - e: 동일한 간격으로 특정 문자를 일정 갯수 삽입하기 (3문자 후 2개의 * 문자 삽입. 공백도 하나의 문자로 취급하기)
  - f: 띄어쓰기 (또는 e 명령어로 삽입된 특정 문자)를 기준으로 문자 뒤집기
  - g: 문자 내부의 특정 문자를 다른 문자로 바꾸기 (바꿀 문자, 새롭게 입력할 문자 입력받음)
  - h: 앞뒤에서 읽었을 때 어디까지 같은 문자인지 출력하기
  - q: 프로그램 종료

예) input data file name: ***data.txt (여기에서는 앞의 예에서 두 문장만을 예제로 적용함. 실제로는 파일에서 읽은 모든 문장에 각각 적용)***

This is computer graphics warming up program.

command: d: 문장 전체를 뒤집기
e: 동일한 간격으로 띄어쓰기를 일정 개수 삽입하기
f: 띄어쓰기를 기준으로 문자 뒤집기
g: 문자 내부의 특정 문자열을 다른 문자열로 바꾸기
h: 앞뒤에서 읽었을 때 어디까지 같은 문자인지 출력하기
q: 종료

* d, e, f 명령어는 기능을 켰다 껐다 하는 토글키처럼 사용됨.
예) d 명령어인 경우에 문장을 뒤집기/원래대로를 번갈아 가며 적용
예) e 명령어인 경우에 일정 문자를 삽입/삭제를 번갈아 가며 적용
예) f 명령어인 경우에 각 단어를 뒤집기/원래대로를 번갈아 가며 적용

//--- 다음 페이지에 계속 소개

3. 문자열 다루기 (2번 문제 확장)

```
input the command: d
        .margorp pu gnimraw scihparg retupmoc si sihT
input the command: d          //--- 다시 원래 순서로 바뀜
        This is computer graphics warming up program.
input the command: e
        Thi**s i**s c**omp**ute**r g**rap**hic**s w**arm**ing**up **pro**gra**m.
input the command: f
        ihT**i s**c s**pmo**etu**g r**par**cih**w s**mra**gni**pu **orp**arg**.m
input the command: f          //--- 다시 원래 순서로 바뀜
        Thi**s i**s c**omp**ute**r g**rap**hic**s w**arm**ing**up **pro**gra**m.
input the command: e          //--- e 명령어를 다시 입력하면 의한 별표 삭제
        This is computer graphics warming up program.
input the command: g i K      //--- i를 K로 바꾸기 (입력 방법은 변경해도 무관)
        ThKs Ks computer graphKcs warmKng up.
input the command: h          //--- 마지막 문자열: abcdefghidcba
                              //--- 8번째 줄 문자열: 1001ada_ada1001
        This is computer graphics warming up program.: 0
        ...
        1001ada_ada1001: 1001ada
        abcdefghidcba : abcd
```

4. 사각형과 직선 간의 충돌 체크

- 비교할 사각형과 직선의 좌표값을 [800, 600] 사이의 랜덤한 값으로 정한다.
  - 각각의 도형은 두 개의 좌표값으로 표시하고, 각 좌표값은 x와 y의 두 개의 정수값으로 표시한다.
  - 즉, 사각형의 (x1, y1) (x2, y2) 값과 직선의 (x3, y3) (x4, y4) 값을 랜덤하게 설정한다.
- 키보드 명령을 사용하여 도형의 좌표값을 이동시킨 후, 두 도형의 충돌 체크 여부를 출력한다.
  - 키보드 명령어 wasd: 사각형을 위/좌/하/우 측으로 일정 길이만큼 이동
  - 키보드 명령어 ijkl: 직선을 위/좌/하/우 측으로 일정 길이만큼 이동
  - 도형의 좌표는 범위를 벗어나지 않는다. 범위를 벗어나면 에러메시지를 출력하고 이동하지 않는다.

```
예)             (이동 Command: 사각형 이동 wasd / 직선 이동 ijkl)
Input Shape Coordinates value:
                Rect: (100, 100) (300, 400)
                Line: (400, 400) (500, 500)
Command: d              //--- 사각형을 오른쪽으로 이동시킨다: x 좌표값을 50 이동
                Rect : (150, 100) (350, 400)
                Line: (400, 400) (500, 500)
Command: k              //--- 직선을 아래쪽으로 이동시킨다: y 좌표값을 -60 이동
                Rect : (150, 100) (350, 400)
                Line: (400, 340) (500, 440)
Command: d              //--- 사각형을 오른쪽으로 이동시킨다: x 좌표값을 50 이동
                Rect : (200, 100) (400, 400)
                Line: (400, 340) (500, 440)                     Rectangle & Line collide!!
```

5. 저장 리스트 만들기 (구조체 데이터 사용)

- 점 (x, y, z) 데이터 값을 저장하는 리스트를 만든다. (점 데이터는 각각 정수이고, 구조체를 사용하도록 한다.)
- 최대 10개의 점 데이터를 저장하도록 한다.
- 리스트에 데이터를 입력하거나 삭제하고 출력하는 명령어를 실행한다.
  - 각 명령어를 입력 받으면 결과 리스트를 다음페이지의 그림과 같이 항상 10개의 항목을 가진 리스트로 (인덱스와 데이터 값) 출력한다.

- 구현 함수 프로토타입과 명령어:
  - + x y z: 리스트의 맨 위에 입력 (x, y, z: 숫자)
  - -: 리스트의 맨 위에서 삭제한다.
  - e x y z: 리스트의 맨 아래에 입력 (명령어 +와 반대의 위치, 리스트에 저장된 데이터값이 위로 올라간다.)
  - d: 리스트의 맨 아래에서 삭제한다. (리스트에서 삭제된 칸이 비어있다.)
  - l: 리스트에 저장된 점의 개수를 출력한다.
  - c: 리스트를 비운다. 리스트를 비운 후 다시 입력하면 0번부터 저장된다.
  - m: 원점에서 가장 먼 거리에 있는 점의 좌표값을 출력한다.
  - n: 원점에서 가장 가까운 거리에 있는 점의 좌표값을 출력한다.
  - s: 원점과의 거리를 정렬하여 오름차순 (또는 내림차순)으로 정렬하여 출력한다. 인덱스 0번부터 빈 칸없이 저장하여 출력한다. 다시 s를 누르면 원래의 인덱스 위치에 출력한다.
  - q: 프로그램을 종료한다.

** 리스트에서 맨 위(인덱스 9번)까지 차고 아래칸(인덱스 0번)이 비어있으면 다음 데이터 입력할 때는 0번에 입력된다. 즉, 10개의 칸을 다 채울 수 있어야 함.

5. 저장 리스트 만들기 (구조체 데이터 사용)

실행 예)

+ 0 1 0	→ 리스트 1번 출력, (리스트 각 항목 옆에 length 출력)
+ 0 1 1	→ 리스트 2번 출력, (리스트 각 항목 옆에 length 출력)
-	→ 리스트 3번 출력, (리스트 각 항목 옆에 length 출력)
e 1 1 1	→ 리스트 4번 출력, (리스트 각 항목 옆에 length 출력)
e 1 0 1	→ 리스트 5번 출력, (리스트 각 항목 옆에 length 출력)
+ 1 1 0	→ 리스트 6번 출력, (리스트 각 항목 옆에 length 출력)
d	→ 리스트 7번 출력, (리스트 각 항목 옆에 length 출력)
s	→ 리스트 8번 출력, (리스트 각 항목 옆에 length 출력)
m	→ (1, 1, 1) 출력
n	→ (0, 1, 0) 출력
l	→ 리스트 길이: 3

1

9	
8	
7	
6	
5	
4	
3	
2	
1	
0	0 1 0

2

9	
8	
7	
6	
5	
4	
3	
2	
1	0 1 1
0	0 1 0

3

9	
8	
7	
6	
5	
4	
3	
2	
1	
0	0 1 0

4

9	
8	
7	
6	
5	
4	
3	
2	
1	0 1 0
0	1 1 1

5

9	
8	
7	
6	
5	
4	
3	
2	0 1 0
1	1 1 1
0	1 0 1

6

9	
8	
7	
6	
5	
4	
3	1 1 0
2	0 1 0
1	1 1 1
0	1 0 1

7

9	
8	
7	
6	
5	
4	
3	1 1 0
2	0 1 0
1	1 1 1
0	

8

9	
8	
7	
6	
5	
4	
3	
2	1 1 1
1	1 1 0
0	0 1 0

6. 경로 만들기

- 50x50 크기의 2차원 평면을 만든다.
- 시작점 (0, 0)에서 대각선에 놓여있는 끝점 (49, 49)까지 랜덤한 경로를 설정하여 출력한다.
  - 길이 아닌 곳은 0으로 표시하고, 경로는 시작점을 1로 출력하고 연결되는 경로를 1보다 큰 정수를 순서대로 출력한다.
  - 이때, 길은 4방향인 좌/우/상/하로 연결되고, 다음의 조건에 맞게 길을 만든다.
    - ✓ 조건 1) 경로는 한 방향으로 5칸 이상 계속 이동할 수 없다.
    - ✓ 조건 2) 경로는 좌우상하로 최소한 1번 이상 방향을 전환한 적이 있어야 한다.
    - ✓ 조건 3) 한번 지나갔던 길을 다시 지나갈 수 있다. 한번 이상 거친 경로는 최종적으로 지나가는 순서의 정수값 (1, 2, 3, ...)으로 출력한다.
    - ✓ 출력할 때 포맷에 맞춰서 출력한다.
  - 명령어: r – 경로를 다시 만든다. Q – 프로그램을 종료한다.

예) (샘플로 10X10으로 나타냈음: 2는 두 번 지나간 길)

1	**2**	0	0	0	0	0	0	0	0
0	**3**	**4**	**5**	0	0	0	0	0	0
0	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	0
0	**27**	0	**7**	0	0	0	0	**18**	0
0	**28**	0	**8**	**9**	**10**	0	0	**17**	0
0	**29**	0	0	0	**11**	0	0	*16*	0
0	**30**	**31**	0	0	**12**	**13**	**14**	**15**	0
0	0	**32**	0	0	0	0	0	0	0
0	0	**33**	0	0	0	0	**40**	**41**	**42**
0	0	**34**	**35**	**36**	**37**	**38**	**39**	0	**43**

- 워밍업 문제는 각 1점으로 채점되고, 최소 3개 이상 구현할 것.
- 9월 7일 (화요일) 22:00까지 이클래스에 제출
  - 늦은 제출: 9월 10일 (금요일) 22:00 까지 (약간의 패널티 있음)

- **제출물: 구현한 파일들, 리드미 파일**
  - **파일들: warming1.cpp ~ warming6.cpp로 저장**
  - **리드미 파일: 구현한 프로그램 소개**
    - **구현한 내용, 실행 키보드 명령어, 본인만의 특이 사항 또는 변경하여 구현한 내용 등이 있으면 설명**

- 다음주에는 오픈지엘 기초에 대하여 수업 진행

- 안전하고 활기찬 2021년 2학기의 첫번째 주 보내세요!