COME0331-005 자료구조
COMP0216-004 자료구조응용
HW #2

2023. 3. 29.

1. input.txt 파일로부터 20 byte가 넘지 않는 문자열 3개를 입력받아 이차원 배열 arr에 저장한 후, 배열 arr을 가리키는 더블 포인터 변수 ptr을 이용하여 세 문자열을 output.txt 파일로 출력하는 프로그램을 작성하시오.

* 더블 포인터 변수의 선언 및 초기화

이차원 배열 int arr[2][3]은 각 원소가 int [3]이고 크기가 2인 일차원 배열과 같다. 따라서 포인터 변수를 이용하여 arr을 가리킬 때 다음과 같은 방법들을 생각해 볼 수 있다.

```
1) int *ptr = arr
2) int (*ptr)[3] = arr
3) int **ptr = arr
```

컴파일러로부터 warning 및 error가 발생하지 않게 하려면 2번과 같은 방법을 써야한다.

배열명 arr은 arr[0]의 주소이다. 그리고 arr[0]에는 int [3]이라는 요소가 들어있으므로 arr[0]은 그 속 int [3]의 배열 이름이 된다. 따라서 &arr[0]은 int [3] 배열의 주소이므로 타입은 int (*)[3]과 같다. 그런데 arr = &arr[0]이므로 결국 arr의 타입은 int (*)[3]과 같다. 따라서 2번과 같은 방법은 int b; int a = b;와 같은 방식이라고 볼 수 있다.

* 입출력 양식 예

```
%vi input.txt <CR>
Apple !
Ba na na
Cho colate _

%vi output.txt <CR>
Apple !
Ba na na
Cho colate _
```

2. 10개의 난수를 생성한 후, 5로 나눴을 때 몫이 같은 난수들의 합을 각각 출력하는 프로그램을 작성하시오.

1) 크기가 10인 1차원 배열에 0-24 범위의 난수 10개를 생성하여 저장한다.
(단, time 함수를 기반으로 시드를 적용하여, 프로그램 실행 시 매번 다른 난수가 발생하도록 해야 한다.)

2) 생성된 난수들을 5로 나눠 몫 값을 조사하여, 같은 몫을 가진 난수가 몇 개씩 있는지 파악한다.
이 때 각 몫 값은 앞으로 생성될 2차원 배열의 행의 index 값이 된다.
(e.g. 난수가 12, 5, 7, 3, 8, 1, 24, 10, 17, 21인 경우
몫 0인 수 : 2개
몫 1인 수 : 3개
몫 2인 수 : 2개
몫 3인 수 : 1개
몫 4인 수 : 2개)

3) 더블 포인터와 동적 메모리 할당을 이용하여, 행의 크기는 5, 열의 크기는 각 개수에 맞도록 2차원 배열의 메모리를 할당받은 후, 각 원소를 차례로 저장한다.
단, 해당 자릿수의 난수가 없는 경우 해당 index의 첫 원소에 -1을 저장한다.
(e.g.

arr[0]	3	1	
arr[1]	5	7	8
arr[2]	12	10	
arr[3]	17		
arr[4]	24	21	

)

4) 다음과 같은 양식으로 각 행의 원소들의 합을 구하고 출력한다.
(e.g. arr[0] = 3 + 1 = 4
arr[1] = 5 + 7 + 8 = 20
arr[2] = 12 + 10 = 22
arr[3] = 17 = 17
arr[4] = 24 + 21 = 49)

3. 스택을 이용하여 만든 주머니에 구슬을 넣는 프로그램을 만드시오.

- 구슬은 검은색과 하얀색 2가지이며, 구슬의 총 개수는 사용자의 입력으로 결정한다. 단, 사용자로부터 입력받는 구슬의 총 개수는 20개 이하로 한다.
- 구슬의 색깔은 난수를 이용하여 결정하며 난수를 생성하기 위해서 아래의 함수를 사용한다.
  - int rand(void); // 0 ~ RAND_MAX 사이의 정수 난수를 발생시킨다.
- 프로그램을 실행할 때마다 다른 결과를 얻기 위해 난수발생기를 초기화시켜야 하므로 아래의 함수를 사용한다.
  - void srand(unsigned seed); // seed는 난수발생의 초기점
- 스택 주머니에 구슬을 넣을 때 아래의 구슬 색깔에 따른 규칙을 따른다.
  - 검은색 구슬인 경우, 검은색 구슬 2개를 넣는다.
  - 하얀색 구슬인 경우, 스택 주머니에서 2개의 구슬을 꺼내고 하얀색 구슬을 1개 넣는다.
    (단, 스택 주머니에서 더 이상 꺼낼 것이 없다면 하얀색 구슬을 바로 넣는다.)
- 구슬을 넣는 작업이 다 끝났거나 스택 주머니가 가득 차서 더 이상 넣을 수 없는 경우, 스택 주머니에서 구슬을 차례대로 꺼내어 하얀색 구슬과 검은색 구슬이 각각 몇 개인지 헤아려 출력한다.
- 각 항목 선택별 입출력 양식은 다음 아래 예시를 따르며, 사용자가 종료를 선택할 때까지 반복한다.

```
****************************************
구슬 개수를 입력하세요 : 3
생성된 구슬 : 검 검 흰
주머니에 구슬을 넣었습니다.

결과
하얀색 구슬: 1개
검은색 구슬: 2개
다시 하겠습니까? [y/n] : y

****************************************
구슬 개수를 입력하세요 : 3
생성된 구슬 : 흰 검 검
주머니에 구슬을 넣었습니다.

결과
하얀색 구슬: 1개
검은색 구슬: 4개
다시 하겠습니까? [y/n] : n

프로그램을 종료합니다.
```

4. 정수를 저장하는 Queue 관리 프로그램을 작성하시오.

- 다음과 같이 4가지 항목으로 구성하고, 사용자가 종료를 선택할 때까지 반복한다. 메뉴를 선택하여 원하는 수행을 하기위해서 반복문과 switch-case문을 사용한다.
- 각 메뉴별 동작은 다음과 같다.
  - AddQ : 표준입력으로부터 정수를 입력받아 Queue에 삽입
  - DeleteQ : Queue에 존재하는 가장 오래된 정수를 모니터에 출력한 후 삭제
  - List : Queue에 존재하는 정수들을 Queue에 입력된 순서대로 모니터에 출력
  - Exit : 프로그램 종료
- Queue는 크기가 5인 배열을 이용하여 구현하며, Queue가 꽉 찼을 때 AddQ를 하거나 비었을 때 DeleteQ를 하려는 경우 에러 메시지를 출력해야한다.
- 각 항목 선택별 입출력 양식은 다음을 따른다.

```
****************************************
* 1. AddQ                              *
* 2. DeleteQ                           *
* 3. List                              *
* 4. Exit                              *
****************************************
Menu : 2
Error!

// 이하 메뉴항목 표기 생략... (프로그램에서는 정상적으로 표기되어야함)

Menu : 3
// 현재 Queue가 비어있는 상태

Menu : 1
AddQ Data : 35

Menu : 1
AddQ Data : -29

Menu : 3
List : 35 -29

Menu : 2
DeleteQ Data : 35

Menu : 1
AddQ Data : 100

Menu : 3
List : -29 100

Menu : 4
Exit
```