

## 과제보고서

2019147578 이상현

Getcommandtail을 통해 프로그램의 커맨드라인을 메모리에 저장한다. 이를 필요한 인수별로 나눠서 저장한다(소스파일, 결과파일, 위치, 색). 이후 인수들 중 source에 해당하는 img이 존재할 경우 이를 읽는 용도로 불러와 그 내용을 읽는다. 수정을 하지 않는 첫번째 줄과 그 밑의 줄로 나눠서 메모리에 저장한다 이후 filehandle을 close한다.

계산의 편의성을 위해서 위치에 해당하는 x와 y를 문자형태에서 숫자로 변형해 저장한다

이 때, x와 y가 1자리수인 경우와 2자리수인 경우를 모두 상정하여 진행한다

이렇게 숫자로 변형한 뒤에 이를 통해 source에 해당하는 img의 수정을 가할 부분을 array형식으로 저장한 mapsbuffer에서의 위치를 나타내는 xyloc를 만들어 접근하기 쉽도록 한다. Mapsbuffer로 들어온 elements들은 한 row가 52개의 elements이며 50개의row가 있다. 따라서 xyloc는 숫자로 변형한 y에 52를 곱하고 숫자로 변형한 x를 더한 값이다. 이를 통해 mapsbuffer의 xyloc위치에 있는, 커맨드라인을 통해 얻은 위치인 (x,y)자리에 있던 색의 값을 xyoric에 저장한다.

커맨드라인을 통해 지정된 위치의 사방이 지정된 위치와 같은 영역에 속하는지를 recursion방식이 들어간 depth first search방식으로 판별한다.

`mov ebx, offset mapsbuffer`을 통해 dfs subroutine에서 mapsbuffer의 내용을 조작할 수 있게 한다.

mapsbuffer는 커맨드라인을 통해 얻은 source에 해당하는 img의 내용을 array형식으로 담은 것이다. 이 때, source에 해당하는 img의 내용 중 첫번째 줄에 해당하는 img의 크기에 대한 내용은 firstrow라는 변수를 통해 따로 저장하고 이 이하의 내용만 mapsbuffer에 저장한다.

`mov edx, offset visitedmap`을 통해 dfs에서 visitedmap의 내용을 조작할 수 있게 한다.

visitedmap은 dfs에서 mapsbuffer의 내용인 특정위치를 방문할 때 방문여부를 확인하는데 사용된다.

`invoke dfs, xyloc, xyoric, color`으로 subroutine인 dfs를 call한다.

xyloc은 커맨드라인을 통해 얻은 x,y값을 source를 통해 얻은 content의 array에 대응되는 위치를 지정하도록 변형한 값이다. 2차원 array에서 한 개의 row는 52개의 원소로 구성되고 50개의 row가 있으므로  $xyloc = x + (52 * y)$ 이다. Xyoric는 이 (x,y)지점에서의 원래의 색의 값이다. Color는 커맨드라인을 통해 얻은 색채우기를 할 색이다.

`dfs proc loc:word, oric:byte, dfscolor:byte`에서는 위의 invoke를 통해 얻은 parameter를 사용하고 ebx와 edx를 통해 mapsbuffer와 visitedmap을 조작한다. dfs에서 loc는 현재 위치를 나타낸다.

`local wayr:word, wayl:word, wayu:word, wayd:word`을 통해 loc인 현재 위치에서 각각 우, 좌, 상, 하의 한칸 이동한 위치를 나타내기 위해 로컬변수를 생성한다.

```

mov ax, loc
mov wayr, ax
inc wayr
mov wayl, ax
dec wayl
mov wayu, ax
sub wayu, 52
mov wayd, ax
add wayd, 52

```

을 통해 위에 적힌대로 움직인 위치를 각각의 변수에 저장한다.(source img를 통해 얻은 내용인 mapsbuffer는 한 row가 52개의 elements로 이루어져 있기 때문에 위와 같은 코드로 현재위치에서의 사방을 표현한 것이다.)

invoke paint, loc, dfscolor을 통해 ebx를 통해 subroutine인 paint를 call하고, 불러온 mapsbuffer에서 현재위치인 loc에 dfscolor의 색을 칠하고 edx를 통해 불러온 visitedmap에서 현재위치인 loc는 방문했음을 표시한다.

paint가 call 되면,

paint proc locat:word, dfscolor:byte에서 위에 제시된 것들을 parameters로 받는다

```

movzx eax, locat;
add eax, edx
mov cl, 1
mov [eax], cl

```

를 통해 visitedmap에 해당 위치인 locat는 방문하였다는 표시를 1로 남긴다(방문하지 않았던 위치라면 0이다)

```

movzx eax, locat
add eax, ebx
mov cl, dfscolor
mov [eax], cl

```

를 통해 mapsbuffer에 해당위치인 locat에 색칠할 색인 dfscolor로 색의 값을 바꾼다(색칠한다)

```

ret
paint endp

```

이후 다시 dfs subroutine으로 돌아간다

이후 현재위치인 loc에서 오른쪽위치인 wayr부터 시계반대방향으로 wayu, wayl, wayd 순서대로 방문하여 색칠할 위치를 탐색한다

이 때, mapsbuffer에서 각 row에서 색에 해당하는 유의미한 정보를 가진 contents는 row의 첫번째 element부터 50번째 element까지이고, 이 뒤의 2개의 elements는 crlf에 해당하는 것으로 다음row로 커서를 위치시키는 것이다.

probr: 이 라벨은 코드 실행동작 구분이 쉽도록 붙여서, 코드작동에 역할은 없다

```

mov ax, 0
cmp wayr, ax
jl probu

```

ax(0)를 통해 wayr이 첫번째 row보다 위에 있는지를 판단한다

0보다 wayr의 위치가 작다면 이는 첫번째row보다 위이니 다음방향을 탐색한다

즉, 이를 통과한다면 wayr위치의 row는 첫번째row이거나 이보다 아래인 것이다.(valid함)

```

mov ax, 2600
cmp wayr, ax
jg probu

```

ax(2600=52\*50)을 통해 wayr이 50번째 row보다 아래에 있는지를 판단한다

2600보다 wayr의 위치가 크다면 이는 50번째row보다 아래이니 다음방향 탐색

즉, 이를 통과한다면 wayr위치의 row는 50번째row이거나 이보다 위인 것이다(valid함)

```

movzx eax, wayr
mov cl, oric
add eax, ebx
cmp [eax], cl
jne probu

```

이 code를 통해 mapsbuffer의 wayr위치를 탐색한다.

mapsbuffer에서 wayr위치의 색이 oric와 같은지 비교한다

oric와 색이 다르다면 다른 영역이거나, crlf에 해당하는 element인 것이다

위의 2개 과정을 통과했다면 wayr은 mapsbuffer의 유의미한 elements중

하나라는 의미이고,

한 row의 맨 오른쪽 2 elements는 crlf에 해당하는 0dh,0ah이기 때문이다.  
즉, 이를 통과한다면 wayr에 위치한 색은 loc위치의 oric와 같은 영역의 색이다.

```
movzx eax,wayr
add eax,edx
mov cl,0
cmp [eax],cl
jne probu
```

이 code를 통해 visitedmap의 wayr위치를 탐색한다

visitedmap에서 wayr위치가 이전에 방문했던 위치인지 확인한다  
visitedmap에서 wayr위치의 값이 0이면 이전에 방문하지 않았다는 것이고  
1이면 이전에 방문했던 곳이다. visitedmap을 통해 방문여부를 확인하여  
무의미한 탐색을 방지하고, 커맨드라인을 통해 얻은 위치(x,y)에서의  
색(oric)과 커맨드라인을 통해 색칠할 색(color)이 같은 경우에도 무한히  
색칠하지 않고 valid한 결과를 낼 수 있도록 한다. 또한, depth first search를  
만족하는 subroutine을 만든다

즉, 이를 통과한다면 wayr은 첫 방문하는 위치이다.

위의 과정들을 모두 통과한다면 색칠할 valid한 위치라는 것이므로,

```
invoke dfs, wayr, oric, dfscolor
```

이 위치로 이동하여(현재위치를 wayr로 바꿔) dfs subroutine을 call한다

**probu:** 위의probr이하의 코드들에서 하나의 과정이라도 통과하지 못하면 다음 방향탐색인 probu로 이  
동해 온다

```
mov ax,0
cmp wayu,
jl probl
mov ax,2600
cmp wayu,ax
jg probl
movzx eax,wayu
mov cl, oric
add eax,ebx
cmp [eax],cl
jne probl
movzx eax,wayu
add eax,edx
mov cl,0
cmp [eax],cl
jne probl
invoke dfs, wayu, oric, dfscolor
```

위 코드는 과정통과 못하면 probl로 이동한다는 것과 wayu에 대한 처리인 것 외에는 probr에서  
의 과정과 같다

```

probl:
    mov ax,0
    cmp wayl, ax
    jl probd
    mov ax,2600
    cmp wayl,ax
    jg probd
    movzx eax,wayl
    mov cl, oric
    add eax,ebx
    cmp [eax],cl
    jne probd
    movzx eax,wayl
    add eax,edx
    mov cl,0
    cmp [eax],cl
    jne probd
    invoke dfs, wayl, oric, dfscolor

probd:
    mov ax,0
    cmp wayd, ax
    jl dfsret
    mov ax,2600
    cmp wayd,ax
    jg dfsret
    movzx eax,wayd
    mov cl, oric
    add eax,ebx
    cmp [eax],cl
    jne dfsret
    movzx eax,wayd
    add eax,edx
    mov cl,0
    cmp [eax],cl
    jne dfsret
    invoke dfs, wayd, oric, dfscolor

```

위 probl과 probd의 코드는 앞서 본 probr,probu와 같은 역할을 수행한다. 단, probd 코드에서 각 과정을 통과하지 못하여 jump한 곳인 dfsret는 아래와 같다

```

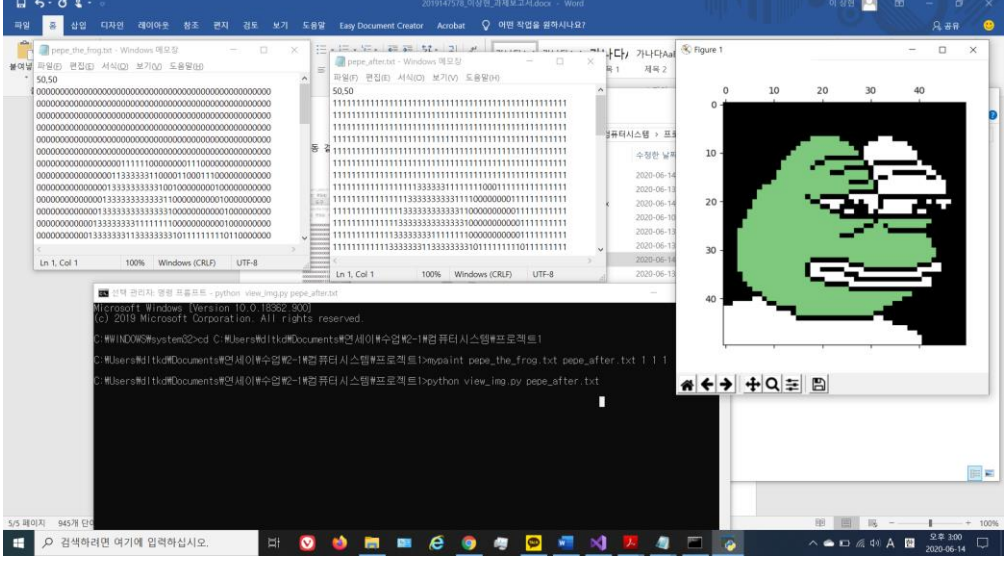
dfsret:
    ret    즉, 위의 4방향의 탐색 후 현재위치loc에 대한 dfs는 종료되고 이 dfs subroutine을
           call한 곳으로 돌아간다(즉, dfs나 main으로 돌아간다)
dfs endp

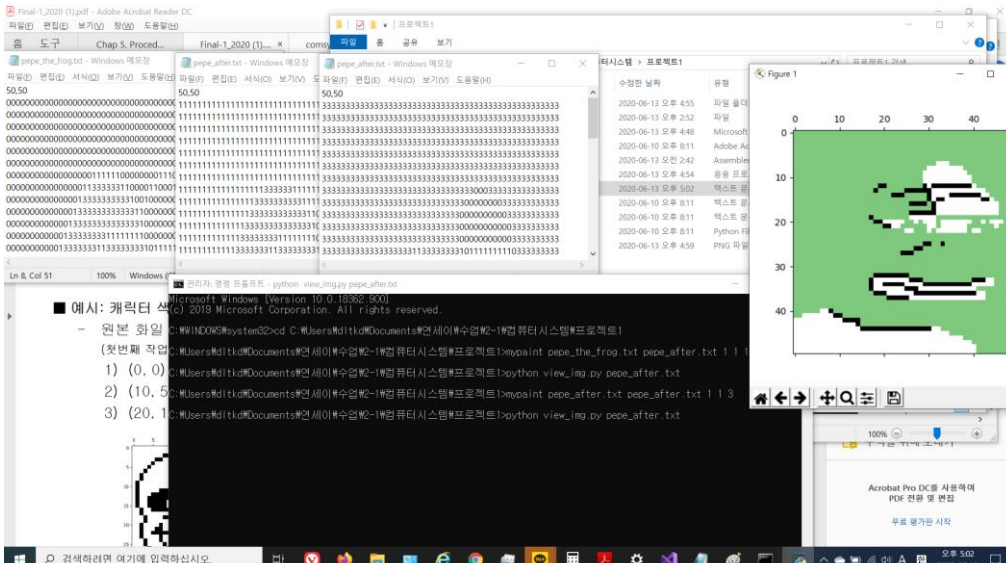
```

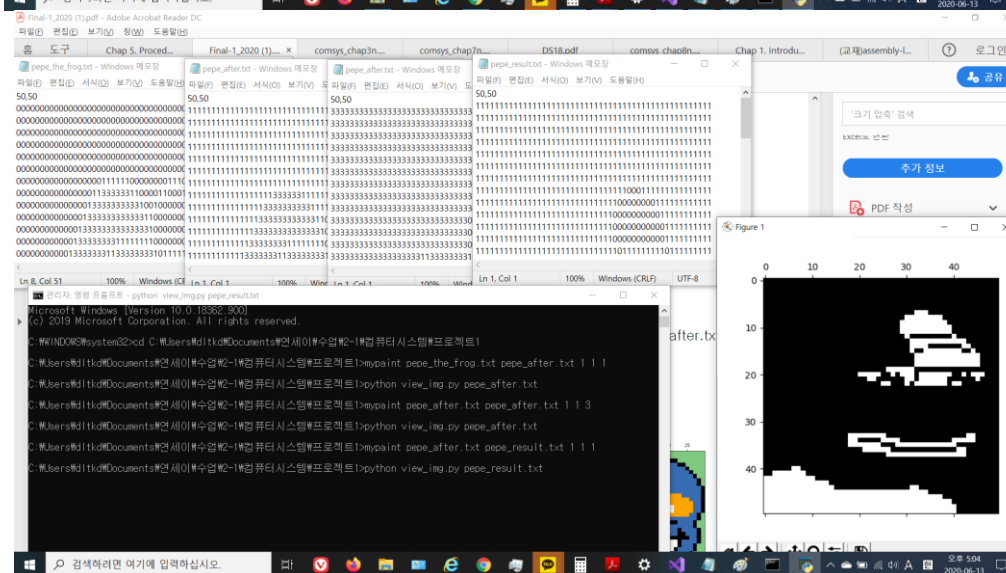
위의 recursive하게 같은 영역을 색칠한 후, 쓰는 용도로 output에 해당하는 img파일을 항상 생성한다 그리고 수정을 하지 않은 첫번째 줄(50,50)과 그 밑의 줄들(mapbuffer의 내용)을 output에 해당하는 파일에 쓴다.

Filehandle를 close하여 작업을 완료한다

## 구동 결과 화면 캡처

- 

1. Initial state of the program. The main window displays a large grid of zeros. A terminal window at the bottom shows the command `python view_img.py pepe_after.txt` and its output, which includes the file path and the command `python view_img.py pepe_after.txt`. A small window titled "Figure 1" shows a green pixelated image of a frog.
- 

2. The program is running. The main window shows a grid of zeros. A terminal window at the bottom displays the command `python view_img.py pepe_after.txt` and its output, which includes the file path and the command `python view_img.py pepe_after.txt`. A small window titled "Figure 1" shows a green pixelated image of a frog.
- 

3. The program is running. The main window shows a grid of zeros. A terminal window at the bottom displays the command `python view_img.py pepe_result.txt` and its output, which includes the file path and the command `python view_img.py pepe_result.txt`. A small window titled "Figure 1" shows a green pixelated image of a frog.