

객체지향프로그래밍

Assignment Report 2-2

2019202050 이강현

1. Stack(Last-In, First-Out)을 구현하시오. 프로그램은 두개의 명령어 push, pop을 가지고 있다. push는 1개의 수를 추가로 입력 받으며 입력 받은 정수를 저장하는 역할을 한다. pop은 추가 인수가 없으며 가장 마지막에 입력 받은 정수를 삭제하는 역할을 한다. 프로그램은 명령어를 입력 받아 동작할 때 마다 저장 되어있는 정수들을 모두 출력하며 구현할 때 필요한 함수 및 기능은 모두 직접 구현한다. (STL 및 Built-in Function 사용 불가)

필요한 개념: 동적할당, string사용법, push, pop 확인 조건문

Push혹은 pop이라는 문자열을 입력받아야 하므로 string변수를 이용하여 입력받고 조건문을 통해 push와 pop각각을 입력받았을 때 행동을 다르게 한다. int*형으로 stack이라는 이름으로 동적할당 한 후 값을 저장한다. 얼마나 값을 받을지 모르므로 stack을 값의 수만큼 재할당하도록 한다.

```
Please Enter Command(push, pop) :push 5
5
Please Enter Command(push, pop) :push 7
5
7
Please Enter Command(push, pop) :push 8
5
7
8
Please Enter Command(push, pop) :push 9
5
7
8
9
Please Enter Command(push, pop) :pop
5
7
8
Please Enter Command(push, pop) :po
5
7
8
Please Enter Command(push, pop) :pop
5
7
Please Enter Command(push, pop) :pop
5
Please Enter Command(push, pop) :pop
Please Enter Command(push, pop) :
```

<예시>

고찰: push일 때 값을 배열의 맨 마지막에 저장하고 pop일때는 가장 나중에 저장된 값을 삭제한다는 알고리즘을 이해하니 쉽게 코드구현이 가능했다. 문제에선 종료조건이 따로 명시되지 않아 아무값도 저장되지 않았을 때 pop을 입력하면 프로그램을 종료하도록 설정했다.

2. . 문자열을 입력 받아 '/'단위로 나눠 출력하는 프로그램을 작성하시오. 프로그램은 `char* my_strtok(char * string)`라는 직접 구현한 함수를 사용해 문자열을 나누어 출력하며 `my_strtok` 함수는 아래와 같이 동작한다.

Argument(string)	동작
string이 NULL이 아닌 경우	String의 첫 '/'을 NULL로 바꾼 후 처음부터 '/' 전까지(만약 '/'가 없을 경우 문자열 끝까지) 문자열을 Return
string이 NULL인 경우	이미 전에 사용한 경우 : 이전 출력 '/'이후의 다음 '/'을 NULL로 바꾼 후 이전 출력의 '/' 이후부터 NULL로 바뀐 '/' 전까지(만약 '/'가 없을 경우 문자열 끝까지) 문자열을 Return
	처음 사용하는 경우 : NULL을 Return



구현할 때 필요한 함수 및 기능은 모두 직접 구현한다. (hint : static 혹은 global 변수 사용)

필요한 개념: string의 사용법, 동적할당의 사용, 전역변수의 특성, 함수의 개념

string으로 문자열을 입력받고 그 길이를 구하고 /의 개수를 파악하여 나눠지는 문자열의 개수를 알아낸다. 그 값을 통해 동적할당의 크기를 정하고 전역변수를 이용하여 strtok함수의 인자가 null값이더라도 문자열에 접근할 수 있게 한다.

```
Microsoft Visual Studio 디버그 콘솔
Please Enter Any String :this/is/my/room/123
this
nis
my
room
123
```

<예시>

고찰: strtok에 처음에는 인자를 받지만 그 이후로는 NULL값을 넣기 때문에 문자열에 계속 접근을 할 수 있어야 했다. 따라서 arr이라는 전역변수를 이용하였고 또한 함수의 반환형식이 char*이었으므로 함수내에서 동적할당을 하여 char*형식이 변수를 또 하나 선언하였다. 하지만 반환하기전에 메모리를 해제하면 값을 반환하지 못하였고 값을 반환하면 메모리누수가 발생한다는 문제점이 있었다. 따라서 일단 값을 반환하였고 함수내에서 동적할당한 변수의 주소를 받는 main함수내의 변수를 추가적으로 메모리해제를 해주는 방식으로 문제를 해결하였다.

3. Matrix의 길이 N를 입력 받아 내부가 0부터 N²-1까지 정수로 채워져 있는 Matrix를 출력하는 프로그램을 구현하시오. 이 때 Matrix의 정수는 아래의 그림과 같이 나선형 오름차순 채워져 있으며 N의 길이는 unsigned int형의 모든 범위내에서 동작한다.

필요한 개념: matrix의 개념, unsigned int형의 범위, 나선형 오름차순이 배열에서 어떻게 표현될 수 있는지

2차원 배열을 입력받은 값을 토대로 동적할당 한 후 모두 값을 초기화해주고 배열의 열과 행에 접근하여 조건문을 통해 움직여가며 값을 넣도록 구현했다.

```
Microsoft Visual Studio 디버그 콘솔
Please enter the length of matrix : 5
0 1 2 3 4
15 16 17 18 5
14 23 24 19 6
13 22 21 20 7
12 11 10 9 8
```

<예시>

고찰: 행렬을 구현한 후 값이 없는 부분에 값을 이동하며 채워넣어줘야 하기 때문에 처음에는 0으로 초기화를 진행하였다. 그러나 행렬을 따라 한바퀴를 나선형으로 돌고 나서 [0][0]에 접근하였을 때 0을 아무값도 입력되지 않은 것으로 인식하여 문제가 발생했었다. 따라서 행렬의 크기가 얼마나 되든 1은 이동하며 마주치지 않으므로 1로 초기화하는 방법을 선택했다. 또한 unsigned int형이므로 0을 값이 음수가 될 때 탈출한다라는 조건문을 사용했을 때 음수값을 저장하지 못하므로 문제가 생겼다. 따라서 양수 범위에서 조건

문을 수정하는 것으로 해결했다.

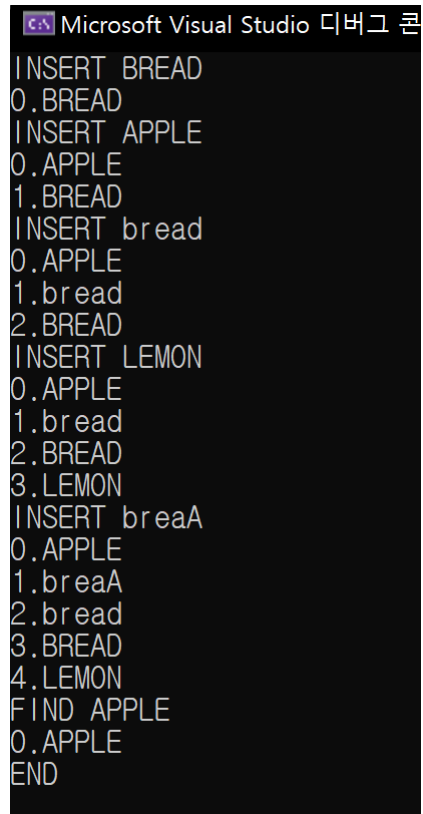
4. 단어장 프로그램을 구현하시오. 프로그램은 INSERT, FIND, END 3개의 기능을 가지며 기능은 아래의 Table과 같다.

명령어	사용법	동작
INSERT	INSERT word	최대 100 개의 Word 를 단어장에 저장한다. 만약 저장된 단어가 100 개일 경우 임의의 경고문구를 출력하고 단어는 저장하지 않는다. 단어는 아래 규칙에 따라 저장되며 매 저장마다 저장되어 있는 모든 단어를 출력한다. 저장규칙 a. 알파벳 순서(a-b-c ...) b. 알파벳이 같을 경우 길이순서 c. 알파벳 대소문자는 구별하지 않는다.
FIND	FIND word	word 가 프로그램에 저장되어 있는지 출력한다. 만약 저장되어 있다면 단어의 순서와 단어를 출력하며 저장되어 있지 않다면 "Not Found"를 출력한다.
END	END	Program 종료

위 프로그램을 구현할 때 필요한 기능은 모두 구현해 사용하며 STL, Built-in Function 등은 사용하지 않는다.

필요한 개념: 문자열 받는 법, 문자열을 저장할 공간을 설정할 동적할당, 각각 명령어를 받았을 때 다른 행동을 취할 조건문, 단어정렬을 위한 조건문, 알파벳의 비교, 단어를 찾기 위해 모든 값에 접근하는 법등

동적할당을 이용하여 100개의 단어를 저장할 수 있는 저장소를 마련하고 명령어에 따라서 INSERT를 받았을 때는 정렬방식에 맞게 저장하고 FIND를 받았을 때는 반복문을 이용하여 모든 단어에 접근하여 동일하면 출력하게끔 END를 만났을 때는 프로그램을 종료하도록 나눠 코드를 작성한다.



```

Microsoft Visual Studio 디버그 콘솔
INSERT BREAD
0.BREAD
INSERT APPLE
0.APPLE
1.BREAD
INSERT bread
0.APPLE
1.bread
2.BREAD
INSERT LEMON
0.APPLE
1.bread
2.BREAD
3.LEMON
INSERT breaA
0.APPLE
1.breaA
2.bread
3.BREAD
4.LEMON
FIND APPLE
0.APPLE
END

```

<예시>

고찰: 단어를 INSERT하는 경우에 문제점들이 많았는데 단어장에 아무것도 없을 때 단어를 비교하려고 하니 문제가 하나 발생하여 단어장의 단어 개수를 표현하는 변수를 이용해 해결하였고 알파벳이 동일할 때 길이비교로 넘어가는데 앞에서부터 차례대로 저장소 인덱스를 늘려가며 접근하다보니 조건을 잘못 설정하면 인덱스가 범위를 넘어가는 문제도 생겼다. 또한 알파벳이 동일하여 길이비교로 넘어갔는데 인덱스를 늘려가며 접근할 때 앞자리가 바뀌면 길이 비교만 되는 문제도 생겼다.(ex 새로운 단어:abcdef 1.app 2.apple 3.ball에서 비교할 때 app->apple->ball순으로 비교하므로 apple을 넘어가고 ball을 만났는데 길이가 더 길어 ball 뒤에 저장하는 경우) 조건문을 결국 수정하여 문제는 해결하였다. 조건이 많이 까다로워 함수도 적절히 잘 이용해야 했던 문제였다.