전자공학프로그래밍 과제 2 보고서

> 전자공학과
> 201820908 오병준
> 개발 환경 : gcc 6.3.0

1. 개발 요구 사항

메모 프로그램 개발하기

1. 사양

- 최대 100개의 메모 저장 기능
  - 각 메모 당 하나의 문자열과 0 ~ 100개의 해시태그 속성
  - 메모 문자열 입력은 최대 200바이트, 해시태그 입력은 최대 30바이트
- 저장한 메모 출력 기능
- 메모 해시태그 검색 기능
- 메모 삭제 기능

2. 프로그램 구조

1. 매크로 변수

1. `MAX_COUNT_MEMO`, `MAX_COUNT_HASHTAG` : 저장 가능한 메모와 해시태그의 최대 개수 지정
2. `MAX_LENGTH_TEXT`, `MAX_LENGTH_HASHTAG` : 저장 가능한 메모와 해시태그의 최대 길이 지정

2. 구조체 MEMO

1. `text` 메모 문자열 변수
   - 자료형 : 문자 배열
   - 배열 크기 : `[MAX_LENGTH_TEXT + 1]`

2. `hashtag` 해시태그 문자열 변수
   - 자료형 : 2차원 문자 배열
   - 배열 크기 : `[MAX_COUNT_HASHTAG][MAX_LENGTH_HASHTAG + 1]`

3. `nHashtag` 해시태그 개수 변수
   - 자료형 : 정수

3. 함수

0. 함수 파라미터

   - `*memo` : 메모를 저장하는 구조체 배열의 시작 주소
   - `*nMemo` : 메모의 개수를 저장하는 변수의 주소
   - `nMemo` : 메모의 개수

1. `printMenu()`
   메뉴 리스트를 화면에 출력한다.

   - 반환형 : **void**

2. `insertMemo(MEMO *memo, int *nMemo)`
   메모를 입력받아 저장한다.
   - 반환형 : **void**

3. `printAllMemo(MEMO *memo, int nMemo)`
   저장된 모든 메모를 출력한다.
   - 반환형 : **void**

4. `searchByHashtag(MEMO *memo, int nMemo)`
   저장된 메모에 대해 해시태그 검색을 수행해 일치하는 메모를 출력한다.
   - 반환형 : **void**

5. `deleteMemo(MEMO *memo, int *nMemo)`
   저장된 메모를 인덱스 번호로 찾아 삭제한다.
   - 반환형 : **void**

2. 코드 분석

1. `main()`

```
MEMO memo[MAX_COUNT_MEMO];
int nMemo = 0;
char temp[100];
int menu;
```

메모 구조체 배열, 메모 개수, 입력 변수 등을 선언한다.

```
do {
  printMenu();
  gets(temp);
  menu = atoi(temp);
  printf("\n");
  switch (menu) {
  case 1:
    insertMemo(memo, &nMemo);
    break;
  case 2:
    printAllMemo(memo, nMemo);
    break;
  case 3:
    searchByHashtag(memo, nMemo);
    break;
  case 4:
    deleteMemo(memo, &nMemo);
    break;
  case 0:
    printf("Program Ended\n\n");
    break;
  default:
    printf("Incorrect Input\n\n");
    break;
  }
} while (menu != 0);
```

`printMenu` 함수를 호출해 메뉴 리스트를 출력한다.
메뉴 번호를 입력받아 그에 해당하는 함수를 호출한다.
0을 입력받으면 프로그램을 종료한다.

---

2. `insertMemo()`

```
char hashInput[MAX_LENGTH_HASHTAG + 1], *token;
memo[*nMemo].nHashtag = 0;
```

해시태그 분리에 사용되는 문자 배열과 토큰 포인터를 선언한다. 해시태그 개수를 0으로 초기화한다.

```
printf("Write the Memo : ");
gets(memo[*nMemo].text);
printf("Write the Hashtag : ");
gets(hashInput);
```

메모를 입력받아 `memo` 구조체 배열에서 마지막으로 입력받은 요소의 다음 요소의 `text` 멤버에 저장한다.
띄어쓰기 입력 지원을 위해 `printf`가 아닌 `gets` 함수를 통해 한 줄 입력을 받는다.

```
if(strlen(hashInput)) {
  token = strtok(hashInput, "#");
  while(token) {
    strcpy(memo[*nMemo].hashtag[memo[*nMemo].nHashtag++], token);
    token = strtok(NULL, "#");
  }
}
```

해시태그 입력이 존재할 때만, 문자열을 # 단위로 분리한다.
분리한 문자열을 `memo`의 `hashtag` 멤버 배열에 차례로 저장하며, 해시태그 개수를 저장한다.
더 이상 분리가 불가능할 때까지 진행한다.

```
printf("Num Memo / Hashtag\n");
printf("\t%d %s\n\t", *nMemo + 1, memo[*nMemo].text);
for(int i = 0; i < memo[*nMemo].nHashtag; i++)
  printf("#%s ", memo[*nMemo].hashtag[i]);
printf("\n\n");
```

입력받은 메모의 인덱스와 메모 내용, 모든 해시태그를 출력한다.

```
(*nMemo)++;
```

저장중인 메모의 개수를 1 증가시킨다.

```
rewind(stdin);
```

입력 버퍼를 비워 다음 입력을 받을 수 있도록 한다.

---

3. `printAllMemo()`

```
printf("Num Memo / Hashtag\n");
for(int i = 0; i < nMemo; i++) {
  printf("\t%d %s\n\t", i + 1, memo[i].text);
  for(int j = 0; j < memo[i].nHashtag; j++)
    printf("#%s ", memo[i].hashtag[j]);
  printf("\n\n");
}
```

메모 구조체 배열의 각 요소를 순환하며 현재 저장중인 모든 메모를 양식에 맞추어 출력한다.

---

4. `searchByHashtag()`

```
char schQuery[MAX_LENGTH_HASHTAG + 1];
printf("Input Hashtag : ");
scanf("%s", schQuery);
```

검색할 해시태그 문자열을 입력받아 저장한다.

```
for(int i = 0; i < nMemo; i++) {
  for(int j = 0; j < memo[i].nHashtag; j++) {
    if(!strcmp(schQuery, memo[i].hashtag[j])) {
      printf("\t%d %s\n\t", i + 1, memo[i].text);
      for(int k = 0; k < memo[i].nHashtag; k++)
        printf("#%s ", memo[i].hashtag[k]);
      printf("\n\n");
    }
  }
}
```

구조체 배열의 각 요소와 그 해시태그를 순환한다.

```
if(!strcmp(schQuery, memo[i].hashtag[j]))
```

위 조건식에 의해 검색어와 일치하는 해시태그를 가진 메모와 해시태그만 모두 출력한다.

```
rewind(stdin);
```

입력 버퍼를 비워 다음 입력을 받을 수 있도록 한다.

---

5. `deleteMemo()`

```
int delQuery;
printf("Input Num : ");
scanf("%d", &delQuery);
```

삭제할 메모의 표시 인덱스를 입력받아 저장한다.

```
for(int i = delQuery; i < *nMemo; i++)
  memcpy(&memo[i - 1], &memo[i], sizeof(MEMO));
```

입력받은 인덱스 다음 요소부터 앞으로 한 개씩 복사해 삭제할 요소를 덮어씌운다.
표시 인덱스는 실제 요소 인덱스보다 1 크므로, 복사는 표시 인덱스부터 시작한다.

```
(*nMemo)--;
```

저장중인 메모의 개수를 1 감소시킨다.

```
rewind(stdin);
```

입력 버퍼를 비워 다음 입력을 받을 수 있도록 한다.

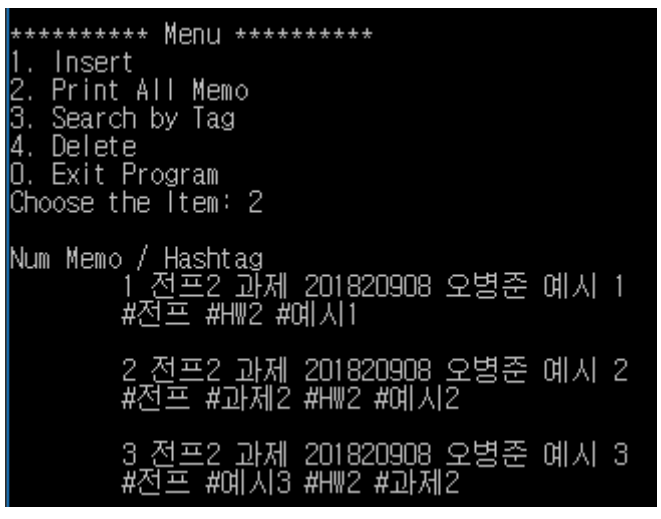
---

3. 실행 결과 분석

I. 메모 삽입

메모와 해시태그 입력을 받고 입력받은 내용과 인덱스를 출력한다.

II. 메모 출력



입력받은 모든 메모와 인덱스를 출력한다.

III. 메모 검색

```
********** Menu **********
1. Insert
2. Print All Memo
3. Search by Tag
4. Delete
0. Exit Program
Choose the Item: 3

Input Hashtag : 과제2

Num Memo / Hashtag
       2 전프2 과제 201820908 오병준 예시 2
       #전프 #과제2 #HW2 #예시2

       3 전프2 과제 201820908 오병준 예시 3
       #전프 #예시3 #HW2 #과제2
```

검색 문자열을 입력받고, 입력과 일치하는 해시태그가 존재하는 메모만 출력한다.
`strcmp` 함수는 문자열이 일치하는 경우만 `0`을 반환하므로,

```
if(!strcmp(schQuery, memo[i].hashtag[j]))
```

일치하는 문자열에 대해서만 위 조건식이 참이 되어 해당하는 메모를 출력한다.

IV. 메모 삭제

```
********** Menu **********
1. Insert
2. Print All Memo
3. Search by Tag
4. Delete
0. Exit Program
Choose the Item: 4

Input Num : 2
********** Menu **********
1. Insert
2. Print All Memo
3. Search by Tag
4. Delete
0. Exit Program
Choose the Item: 2

Num Memo / Hashtag
       1 전프2 과제 201820908 오병준 예시 1
       #전프 #HW2 #예시1

       2 전프2 과제 201820908 오병준 예시 3
       #전프 #예시3 #HW2 #과제2
```

입력받은 표시 인덱스에 해당하는 메모를 삭제한다.

V. 해시태그 미입력

```
********** Menu **********
1. Insert
2. Print All Memo
3. Search by Tag
4. Delete
0. Exit Program
Choose the Item: 1

Write the Memo : 테스트 입력
Write the Hashtag :
Num Memo / Hashtag
       3 테스트 입력


********** Menu **********
1. Insert
2. Print All Memo
3. Search by Tag
4. Delete
0. Exit Program
Choose the Item: 2

Num Memo / Hashtag
       1 전프2 과제 201820908 오병준 예시 1
       #전프 #HW2 #예시1

       2 전프2 과제 201820908 오병준 예시 3
       #전프 #예시3 #HW2 #과제2

       3 테스트 입력
```

사양서의 해시태그의 최소 입력 개수가 0개이므로, 해시태그를 입력하지 않으면

```
if(strlen(hashInput))
```

을 통과하지 못하여 `nHashtag` 멤버가 초기화 값인 0으로 유지되고, 출력에서도 나타나지 않는다.

4. 전체 코드

HW2_prob.c — C:₩Users₩LUFT-AQUILA₩Dropbox₩Atom₩Ajou_Univ₩Electronics_Pragramming₩과제2 — Atom

File Edit View Selection Find Packages Help

HW2_prob.c README.md

```c
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

#define MAX_COUNT_MEMO 100
#define MAX_LENGTH_TEXT 200
#define MAX_COUNT_HASHTAG 100
#define MAX_LENGTH_HASHTAG 30

typedef struct {
  char text[MAX_LENGTH_TEXT + 1];
  char hashtag[MAX_COUNT_HASHTAG][MAX_LENGTH_HASHTAG + 1];
  int nHashtag;
}MEMO;

void printMenu();
void insertMemo(MEMO *memo, int *nMemo);
void printAllMemo(MEMO *memo, int nMemo);
void searchByHashtag(MEMO *memo, int nMemo);
void deleteMemo(MEMO *memo, int *nMemo);

int main(void) {
  MEMO memo[MAX_COUNT_MEMO];
  int nMemo = 0;
  char temp[100];
  int menu;

  do {
    printMenu();
    gets(temp);
    menu = atoi(temp);
    printf("\n");
    switch (menu) {
    case 1:
      insertMemo(memo, &nMemo);
      break;
    case 2:
      printAllMemo(memo, nMemo);
      break;
    case 3:
      searchByHashtag(memo, nMemo);
      break;
    case 4:
      deleteMemo(memo, &nMemo);
      break;
    case 0:
      printf("Program Ended\n\n");
      break;
    default:
      printf("Incorrect Input\n\n");
      break;
    }
  } while (menu != 0);
  return 0;
}

void printMenu() {
  printf("********** Menu **********\n");
  printf("1. Insert\n");
  printf("2. Print All Memo\n");
  printf("3. Search by Tag\n");
  printf("4. Delete\n");
  printf("0. Exit Program\n");
  printf("Choose the Item: ");
  return;
}

void insertMemo(MEMO *memo, int *nMemo) {
  char hashInput[MAX_LENGTH_HASHTAG + 1], *token;
  memo[*nMemo].nHashtag = 0;
  printf("Write the Memo : ");
  gets(memo[*nMemo].text);
  printf("Write the Hashtag : ");
  gets(hashInput);
  if(strlen(hashInput)) {
    token = strtok(hashInput, "#");
    while(token) {
      strcpy(memo[*nMemo].hashtag[memo[*nMemo].nHashtag++], token);
      token = strtok(NULL, "#");
    }
  }
  printf("Num Memo / Hashtag\n");
  printf("\t%d %s\n\t", *nMemo + 1, memo[*nMemo].text);
  for(int i = 0; i < memo[*nMemo].nHashtag; i++)
    printf("#%s ", memo[*nMemo].hashtag[i]);
  printf("\n\n");
```

```
    (*nMemo)++;
    rewind(stdin);
    return;
  }

  void printAllMemo(MEMO *memo, int nMemo) {
    printf("Num Memo / Hashtag\n");
    for(int i = 0; i < nMemo; i++) {
      printf("\t%d %s\n\t", i + 1, memo[i].text);
      for(int j = 0; j < memo[i].nHashtag; j++)
        printf("#%s ", memo[i].hashtag[j]);
      printf("\n\n");
    }
    return;
  }

  void searchByHashtag(MEMO *memo, int nMemo) {
    char schQuery[MAX_LENGTH_HASHTAG + 1];
    printf("Input Hashtag : ");
    scanf("%s", schQuery);
    printf("\nNum Memo / Hashtag\n");
    for(int i = 0; i < nMemo; i++) {
      for(int j = 0; j < memo[i].nHashtag; j++) {
        if(!strcmp(schQuery, memo[i].hashtag[j])) {
          printf("\t%d %s\n\t", i + 1, memo[i].text);
          for(int k = 0; k < memo[i].nHashtag; k++)
            printf("#%s ", memo[i].hashtag[k]);
          printf("\n\n");
        }
      }
    }
    rewind(stdin);
    return;
  }

  void deleteMemo(MEMO *memo, int *nMemo) {
    int delQuery;
    printf("Input Num : ");
    scanf("%d", &delQuery);
    for(int i = delQuery; i < *nMemo; i++)
      memcpy(&memo[i - 1], &memo[i], sizeof(MEMO));
    (*nMemo)--;
    rewind(stdin);
    return;
  }

HW2_prob.c  0 0 0  133:1  CRLF  UTF-8  C  master  Fetch  GitHub  Git (0)
100%  오후 11:02
```