컴퓨터 프로그래밍 3

텀 프로젝트 보고서

과목명 : 컴퓨터프로그래밍3

분반 : 00반

담당교수 : 조은선 교수님

소속 학과 : 컴퓨터융합학부

학번 : 2020025146

이름 : 임우진

목 차

1. 프로그램 요구 분석

2. 프로그램 설계 방식

3. 프로그램 동작 원리

4. 느낀 점

1. 프로그램 요구 분석

텀 프로젝트 주제 : 간단한 Line Editor 만들기 프로젝트

Base :
사용 프로그래밍 언어 : C
개발 환경 : Visual Studio Code with Ubuntu

Function :

자료구조 - 이중 연결 리스트

동적 할당의 적절한 사용 - malloc

문자열의 추가(Append), 삽입(Insert), 삭제(delete)를 구현

Page 최적화 및 반복 실행 구현

Operate :

WSL환경의 cmd에서 프로그램(cnuled)이 실행되며

같은 디렉토리 내 testfile.txt에서 읽기/쓰기(r/w)를 진행

Struct Page :

구조체 Page를 구현 (Page 구조체끼리 이중 연결 리스트 형태로 연결)

- Page당 최대 10개의 문자열
- Page의 문자열 수
- 이전 Page 포인터
- 다음 Page 포인터

2. 프로그램 설계 방식

구현 사항 개요

전역 변수와 선행처리기

구조체 : Page

함수 : appendLine

insertLine

removeLine

printPages

main함수 - main함수는 동작과 관련성이 높기에 3장에서 다룹니다.

세부 구현 사항

전역 변수와 선행처리기

```
#define MAX_LINES 10 //10줄
#define MAX_CHARS 80 //최대 80자의 문자열
#define MAX_READ_LINES 5 //읽을 때 5줄

int total = 0; // 전체 존재하는 문자열 개수
```

Page 구조체와 전역 포인터 변수

```
typedef struct Page{ //Page 구조체
    char line[MAX_LINES][MAX_CHARS];
    int count;
    struct Page *nextPage;
    struct Page *prevPage;
}Page;

Page *firstPage = NULL; //첫 페이지 포인터
Page *currentPage = NULL; //현재 페이지 포인터
```

appendLine 함수 설명과 코드

가장 마지막 페이지 line[count]에 문자열 삽입하고 count 1 증가
빈 line 없을 시 Page추가하고 이전 Page와 연결하고 새로운 Page에 삽입
한 페이지당 5개까지 appen하여 페이지 최적화

```
void appendLine(char *str){
    if(currentPage == NULL){ //아무것도 없으면 새페이지 생성해서 첫페이지로 지정
        currentPage = (Page *)malloc(sizeof(Page));
        currentPage->count = 0;
        currentPage->nextPage = NULL;
        currentPage->prevPage = NULL;
        firstPage = currentPage;
    }

    strcpy(currentPage->line[currentPage->count], str); //마지막줄에 추가
    currentPage->count++; //페이지의 카운트 1 증가

    if(currentPage->count == MAX_READ_LINES){ //페이지에 문자열이 5개가 되면
        Page *newPage = (Page *)malloc(sizeof(Page)); //새페이지 만들기
        if(newPage == NULL){ //동적할당 실패 시 예외처리
            printf("Error: Failed to allocate memory for a new page.\n");
            return;
        }
        newPage->count = 0;
        newPage->nextPage = NULL;
        newPage->prevPage = currentPage;
        currentPage->nextPage = newPage;
        currentPage = newPage;
    }
    total++; //전체 문자열 개수 증가
}
```

insertLine 함수

모든 페이지에서 존재하는 요소 중 i번째 라인에 삽입하고 이 후 값들은 뒤로 한 칸 씩 이동
i case:
    I > total_line -> error occur!
    i < total_line -> 각 페이지의 page->count에 의거하여 page 넘겨가며 알맞은 위치를 탐색
탐색한 페이지에서 count case :
  count = 10 //가득 차 있음
    알맞은 위치에 삽입하고 한 칸씩 뒤로 밀고 마지막에 있던 값은 바로 뒤에 새로운 페이지
    만들어서 그 페이지의 첫번째 줄로 이동
    페이지 연결 구성 : str삽입한 페이지 - 새로운 페이지 - str삽입한 페이지의 원래 nextPage
  count != 10
    알맞은 위치에 삽입하고 한 칸 씩 뒤로 밀기
삽입 성공 시 count와 total 증가

removeLine 함수

모든 페이지에서 존재하는 요소 중 i번째 라인 삭제 후 그 페이지에서만 한 칸 씩 당겨오기
페이지의 count와 전역변수 total 감소

insertLine 함수 코드

```
void insertLine(int i, char *str) { //인자는 인덱스와 문자열
    if(i > total){ //범위를 벗어남
        printf("Error: Index out of range.\n");
        return;
    }

    Page *page = firstPage; //첫페이지부터 순회


    while(page != NULL && i >= page->count){ //인덱스 세는 반복문 : i - (그 페이지의 카운트)
        i -= page->count;

        page = page->nextPage; //페이지 넘기기
    }

    if(page == NULL){
        printf("Error: Index out of range.\n");
        return;
    }

    int lineIdx = i; //위의 반복문을 마치고 나온 결과물 줄 위치

    if(page->count == MAX_LINES){ //꽉 찼으면 새 페이지 생성하고 연결하고
        Page *newPage = (Page *)malloc(sizeof(Page));
        if(newPage == NULL){
            printf("Error: Failed to allocate memory for a new page.\n");
            return;
        }
        //연결 알고리즘 : (원)page1->page2 => page1->newpage->page2
        newPage->count = 0;
        newPage->prevPage = page;
        newPage->nextPage = page->nextPage;
        page->nextPage = newPage;

        if(newPage->nextPage != NULL){
            newPage->nextPage->prevPage = newPage;
        }

        if(lineIdx == 0){ //lineIdx가 0이면 페이지 넘기기
            page = newPage;
        }else{
            strcpy(newPage->line[newPage->count++], page->line[MAX_LINES - 1]);
            page->count--;
        }
    }

    for(int j = page->count; j > lineIdx; j--){ //한칸씩 뒤로 밀기
        strcpy(page->line[j], page->line[j - 1]);
    }

    strcpy(page->line[lineIdx], str); //밀고 남은 빈자리(삽입하고픈 위치)에 삽입
    page->count++;
    total++;
}
```

removeLine 함수 코드

```
void removeLine(int i) { //index를 받아서 삭제하는 연산
    i--;
    if(i >= total){ //범위 벗어남
        printf("Error: Index out of range.\n");
        return;
    }

    Page *page = firstPage;

    //페이지 넘기기
    while(page != NULL && i >= page->count){
        i -= page->count;

        page = page->nextPage;
    }

    if(page == NULL){
        printf("Error: Index out of range.\n");
        return;
    }

    // 페이지 내의 요소들만 이동시킵니다 - 다음 페이지의 요소는 건드리지 않고!
    for(int j = i; j < page->count - 1; j++){
        strcpy(page->line[j], page->line[j + 1]);
    }

    page->count--;
    total--;

    // 페이지가 비어있다면 메모리를 해제하고 연결을 조정합니다.
    if(page->count == 0){
        if(page->prevPage){
            page->prevPage->nextPage = page->nextPage;
        }else{
            firstPage = page->nextPage;
        }

        if(page->nextPage){
            page->nextPage->prevPage = page->prevPage;
        }

        free(page);
    }
}
```

printPages 함수 설명 및 코드

Page 순회하며 저장된 문자열들을 요구 양식대로 출력

```
void printPages(){ //페이지 출력
    Page *page = firstPage;
    int pageNum = 1;

    while(page != NULL){
        printf("<Page %d>\n", pageNum);
        for(int i = 0; i < page->count; i++){
            printf("%s\n", page->line[i]);
        }
        printf("----------------------\n");
        page = page->nextPage;
        pageNum++;
    }
}
```

3. 프로그램 동작 원리

main함수 및 구성

main함수(int argc, char* argv[])

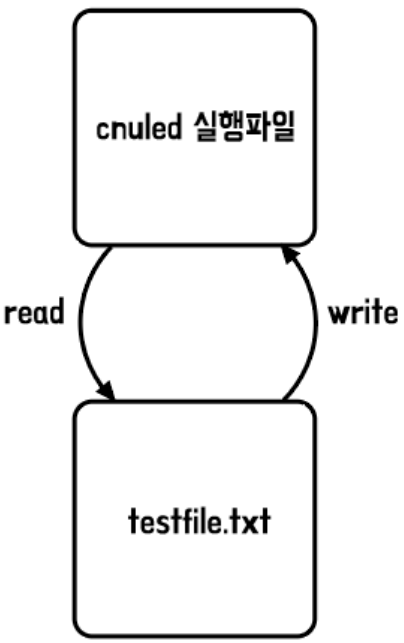
맨 처음에 argc와 argv를 받아
(argc = 3) file열기 ./cnuled -f testfile.txt

단발성 실행이 아닌, 연속적인 명령을 받기 위해
exit라는 명령어가 나오기 전까지 실행(반복 실행)

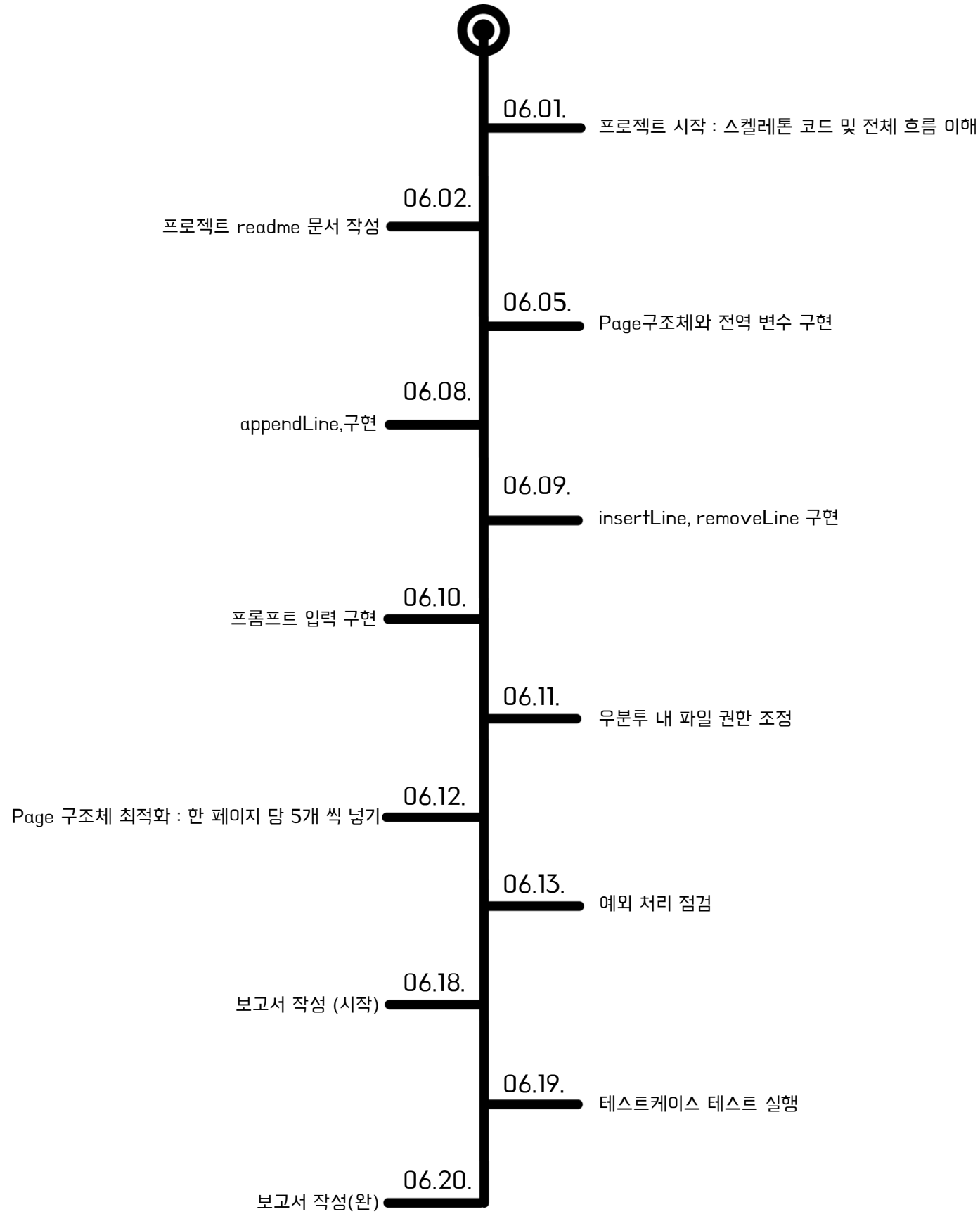
- argc = 2 or 3
- argv
  - [0] : exit or 명령어
  - [1] : 명령어 a일 경우 문자열, r일 경우 인덱스
  - [2] : 명령어 i일 경우 문자열
- 현재 상태 출력 - 페이지별로

프로그램 동작 방식

```
FILE *file = fopen(argv[2], "r"); //read
char line[MAX_CHARS];
        while(fgets(line, sizeof(line), file)){
            // 파일의 이미 존재하는 문자열을 구조체 삽입
            line[strcspn(line, "\n")] = '\0';
            appendLine(line);
        }

        fclose(file);


FILE *outputFile = fopen("testfile.txt", "w"); //write
Page *page = firstPage; //첫 페이지부터
    while(page != NULL){
        for(int i = 0; i < page->count; i++){
            fprintf(outputFile, "%s\n", page->line[i]);
        }
        page = page->nextPage;
    }

    fflush(outputFile);
    fclose(outputFile);
```



4. 프로젝트 진행 타임라인 및 느낀점



이번 텀 프로젝트를 진행하며 느낀 점

텀 프로젝트 **라인 에디터**를 진행하며
C언어의 기초 문법과 포인터, 구조체, 동적할당 등
한 학기 동안 학습하고 숙지한 것들을 총정리할 수 있었고
하나의 프로그램을 구현할 수 있는 기회가 된 것 같다.

처음에 readme 문서를 작성하고
함수, 구조체 하나하나 구현해가며 많은 것을 수정하고
예외 처리를 추가하며 프로젝트를 진행했다.
깃허브에 올라온 테스트 케이스뿐 아니라
다른 케이스들도 실행해 본 결과 자유자재로 작성/수정이 가능했다.

프로젝트를 진행하며 오류, 오작동, 구현 때문에 힘들기도 했지만
기능이 구현되는 순간마다 너무 뿌듯하고 재미있었다.

+)
원래 C++를 공부하다가 C를 공부하게 되었는데
익숙하면서도 반가운 느낌이 들었고
우분투나 WSL도 낯설었지만 한 번 설정해놓고 계속 사용하다 보니
익숙해졌다.