고급C프로그래밍 최종 발표보고서

2018605059 정수연

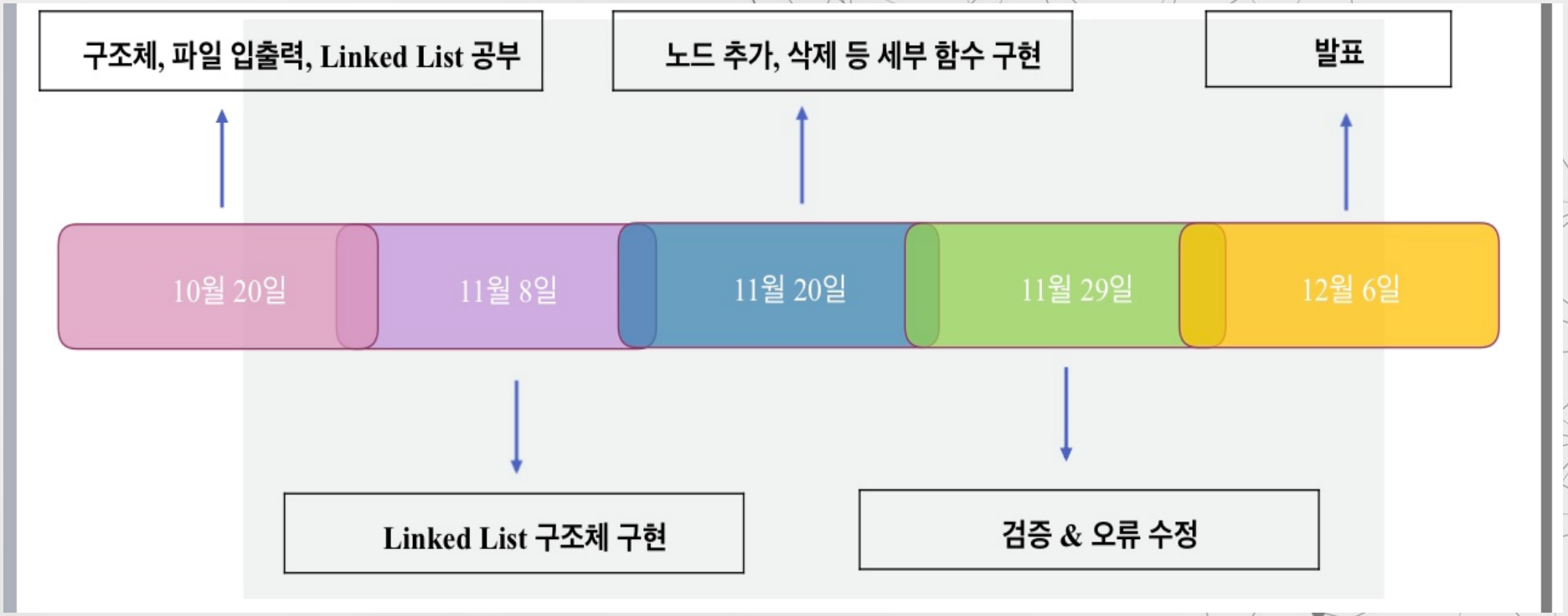
TABLE OF CONTENTS

01

Introduction

Introduction

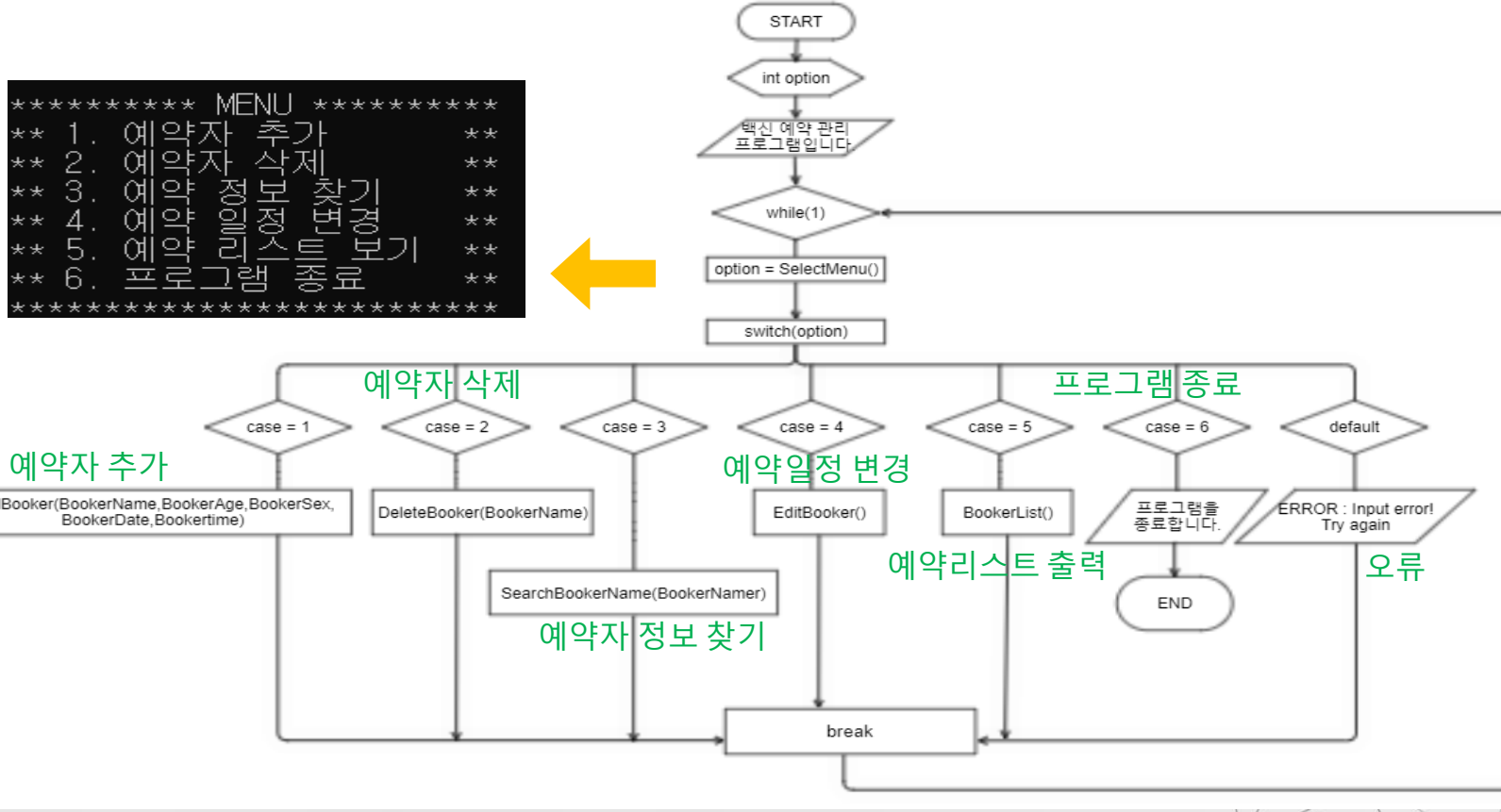
- 프로그램 설명 - 링크드리스트를 이용한 **백신 예약 프로그램**



02

Flow Chart

Flow Chart


********** MENU **********
** 1. 예약자 추가 **
** 2. 예약자 삭제 **
** 3. 예약 정보 찾기 **
** 4. 예약 일정 변경 **
** 5. 예약 리스트 보기 **
** 6. 프로그램 종료 **
****
START
int option
택신 예약 관리 프로그램입니다.
while(1)
option = SelectMenu()
switch(option)
예약자 삭제
프로그램 종료
case = 1
case = 2
case = 3
case = 4
case = 5
case = 6
default
예약자 추가
AddBooker(BookerName,BookerAge,BookerSex, BookerDate,Bookertime)
DeleteBooker(BookerName)
SearchBookerName(BookerNamer)
예약자 정보 찾기
예약일정 변경
EditBooker()
BookerList()
예약리스트 출력
프로그램을 종료합니다.
END
ERROR : Input error! Try again
오류
break

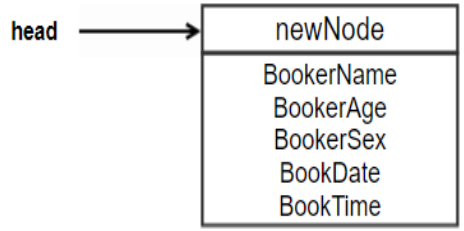
03

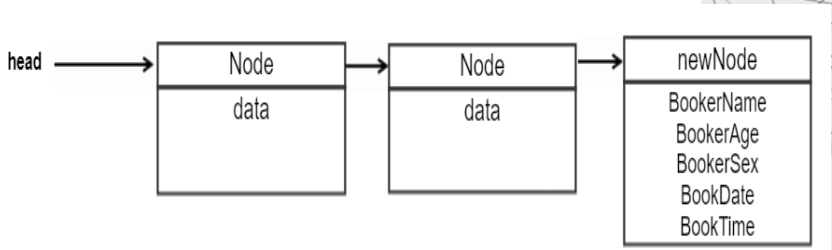
Algorithm

Algorithm

〈 예약자 추가〉

```
void AddBooker(char BookerName[], char BookerAge[], char BookerSex[], char BookerDate[], char
    BookerTime[]) {
    예약자 정보 노드 구조체에서 newNode를 동적으로 생성
    입력받은 정보들을 각각 strcpy를 통해 newNode에 저장
    newNode->next = NULL;
    if (head가 비었다면)
        head = newNode;
    else {
        while (연결리스트의 head로부터 가장 끝 노드로 이동);
        기존 리스트에 newNode 연결
    }
    printf("성공적으로 추가되었습니다.₩n₩n");
}
```

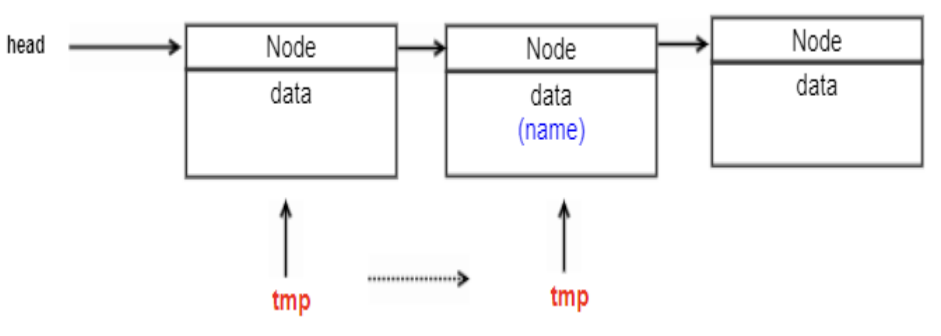




Algorithm

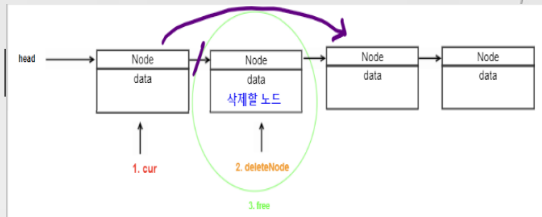
< 예약자 정보 찾기>

```
struct Booker_information* SearchBooker(char* name) {
    Booker_information* tmp = head;
    while (연결리스트의 head부터 끝 노드까지 반복) {
        if (strcmp를 통해 연결리스트에서 name을 찾으면)
                해당 노드 반환
        tmp = tmp->next;
    }
    return NULL;
}
```

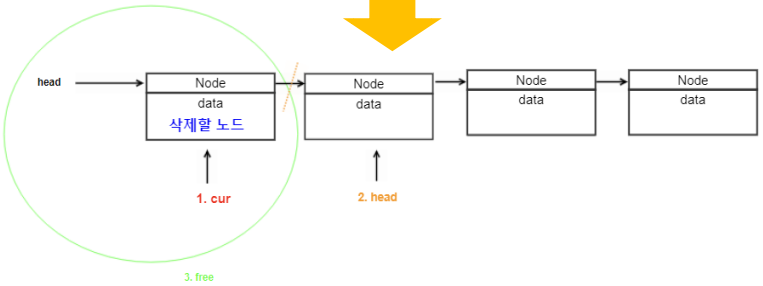


Algo

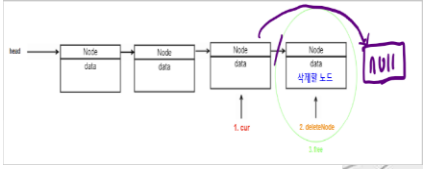
〈 예약자 삭제〉




```
void DeleteBooker(char BookerName[]) {
    if (리스트가 비었으면) {
        printf("해당 예약자는 존재하지 않습니다.₩n₩n");
        return;
    }

    if (SearchBooker함수로 삭제할 예약자가 연결리스트에
    있다면) {
        Booker_information* cur = head;
        if (strcmp를 통해 이름이 같다면) {
            head = cur->next;
            free를 통해 노드 메모리를 해제해서 예약자 정보 삭제
            printf("성공적으로 삭제되었습니다.₩n₩n");
```

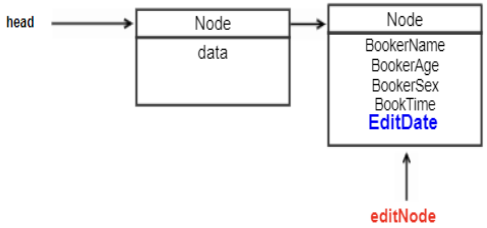



```
        else {
            while (연결리스트의 tail까지 반복) {
                if (strcmp를 통해 이름이 같다면) {
                    Booker_information* deleteNode = cur->next;
                    if (삭제할 노드가 리스트의 중간에 있으면)
                        cur->next = deleteNode->next;
                    else {
                        삭제할 노드가 리스트의 마지막에 있으므로
                        cur->next = NULL;
                    }
                    free(deleteNode);
                    printf("성공적으로 삭제되었습니다.₩n₩n");
                    return;
                }
                연결리스트의 다음 노드로 이동      }
            }
        }
        else {
            printf("해당 예약자는 존재하지 않습니다.₩n₩n");
            return;
        }
    }
```

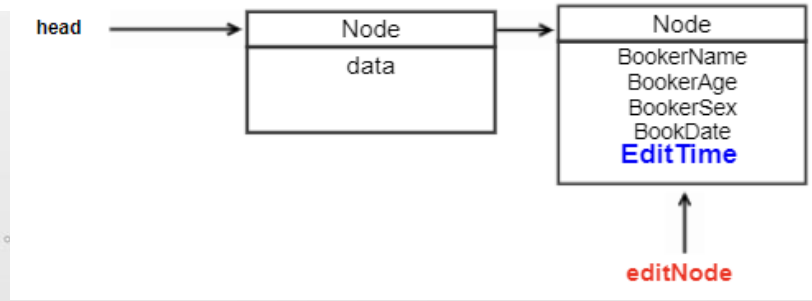


Algorithm

〈 예약 일정 변경〉




```
void EditBooker() {
    char BookerName[16];
    scanf로 수정할 예약자 이름 입력받아 BookerName에 저장

    int option = 0;

    Booker_information* editNode = SearchBooker(BookerName);
    if (editNode) {
        BookerName이 연결리스트에 존재하므로
        while (1) {
            변경메뉴 출력 후 사용자가 번호 선택
```

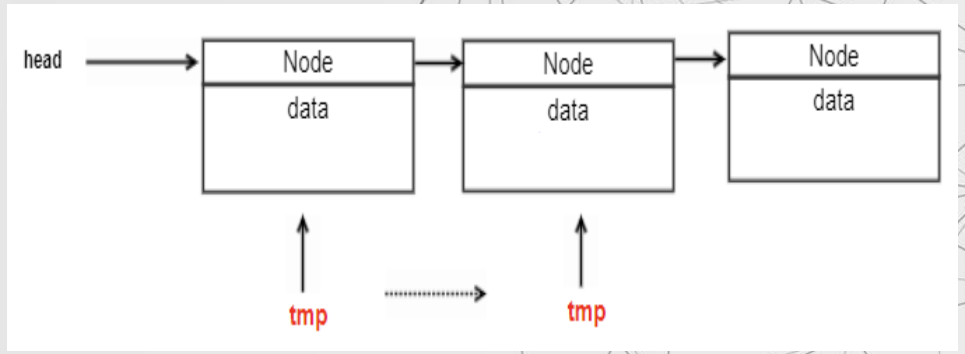



```
            switch (option) {
            case 1:// 날짜 변경
            {
                char EditDate[16];
                변경할 날짜를 사용자로부터 입력받아 EditDate에 저장
                strcpy로 기존 노드에 EditDate를 복사하여 업데이트
                break;
            }
            case 2:// 시간 변경
            {
                char EditTime[16];
                printf("변경할 시간: ");
                변경할 시간을 사용자로부터 입력받아 EditTime에 저장
                Strcpy로 기존 노드에 EditTime을 복사하여 업데이트
                break;
            }
            case 3:
                변경이 잘 되었음을 공지 후 종료
            default:
                printf("ERROR : Input error! Try again.₩n₩n");
            }
        }
    }
    else {
        printf("₩n해당 예약자는 존재하지 않습니다.₩n₩n");
        return;
```

Algorithm

〈 예약 리스트 보기〉

```
void BookerList() {
    if (리스트가 비었다면) {
        printf("예약자 리스트가 비었습니다.₩n₩n");
        return;
    }

    printf("/////// 예약리스트 ///////₩n");
    Booker_information* tmp = head;
    for (int i = 1; tmp != NULL; i++) {
        연결리스트의 head부터 tail까지 이동하며 각 노드의 정보를 출력함
        다음 노드로 이동
    }
    printf("₩n");
    return;
}
```



04

Code & Result Screen

Code & Result Screen

〈 예약자 정보 노드 구조체 〉

```
typedef struct Booker_information {
    char BookerName[16];                // 이름
    char BookerAge[16];  // 나이
    char BookerSex[16];   // 성별
    char BookerDate[16]; // 날짜
    char BookerTime[16]; // 시간
    struct Booker_information* next;   // 다음 노드를
    가리킬 구조체 포인터
}Booker_information;
```

```
//첫 노드 가리킬 head포인터 선언 후 null로 초기화
Booker_information* head = NULL;
```

〈 예약자 추가〉

```
void AddBooker(char BookerName[], char BookerAge[], char BookerSex[],
char BookerDate[], char BookerTime[]) {
    // 노드를 동적으로 생성하여 데이터를 저장한다
    Booker_information* newNode =
    (Booker_information*)malloc(sizeof(Booker_information));
    strcpy(newNode->BookerName, BookerName);
    strcpy(newNode->BookerAge, BookerAge);
    strcpy(newNode->BookerSex, BookerSex);
    strcpy(newNode->BookerDate, BookerDate);
    strcpy(newNode->BookerTime, BookerTime);
    newNode->next = NULL;

    // 연결리스트 만들기
    // 리스트가 비었다면 생성한 노드를 리스트의 head로 설정
    if (!head)
        head = newNode;
    else {
        Booker_information* tmp = head;
        // 연결리스트의 가장 끝 노드로 이동
        while (tmp->next != NULL)
            tmp = tmp->next;
        tmp->next = newNode;// 새로 생성한 노드를 연결
    }
    printf("성공적으로 추가되었습니다.₩n₩n");
}
```

Code & Result Screen

〈예약자 정보 찾기〉

```
struct Booker_information* SearchBooker(char* name) {
    Booker_information* tmp = head;
    // 연결리스트의 head부터 tail까지 반복
    while (tmp) {
            // 찾으면 해당 노드를 반환
            if (strcmp(tmp->BookerName, name) == 0)
                    return tmp;
            tmp = tmp->next;
            // 못 찾으면 연결리스트의 다음 노드로 이동
    }
    return NULL;
    // 찾는 이름이 연결리스트에 없으면 NULL 포인터 반환
}
```

Code & Result Screen

< 예약자 삭제>

```
void DeleteBooker(char BookerName[]) {
    // 연결 리스트가 비었을 경우 종료
    if (!head) {
        printf("해당 예약자는 존재하지 않습니다.₩n₩n");
        return;
    }

    // 삭제하려는 예약자가 연결리스트에 존재하고
    if (SearchBooker(BookerName)) {
        Booker_information* cur = head;
        // 삭제할 노드가 head일 때
        if (strcmp(cur->BookerName, BookerName) == 0) {
            head = cur->next;// head의 next를 head로 설정
            free(cur);// 삭제하려는 노드의 메모리 해제
            printf("성공적으로 삭제되었습니다.₩n₩n");
            return;
    }
```

```
    else {
        // 연결리스트의 tail까지 반복
        while (cur->next != NULL) {
            if (strcmp(cur->next->BookerName, BookerName) == 0) {
                Booker_information* deleteNode = cur->next;
                // 삭제할 노드가 연결리스트의 중간에 위치할 경우
                if (deleteNode->next)
                    cur->next = deleteNode->next;// 리스트 재연결
                // 삭제할 노드가 연결리스트의 마지막에 위치할 경우
                else
                    cur->next = NULL;// 현재 노드를 tail로 설정
                free(deleteNode);// 삭제하려는 노드의 메모리 해제
                printf("성공적으로 삭제되었습니다.₩n₩n");
                return;
            }
            cur = cur->next;// 연결리스트의 다음 노드로 이동
        }
    }
    // 삭제할 예약자가 연결리스트에 존재하지 않을 경우
    else {
        printf("해당 예약자는 존재하지 않습니다.₩n₩n");
        return;
    }
}
```

Code & Result Screen

〈예약 일정 변경〉

```
int EditMenu() {
    int option = 0;
    printf("₩n********** MENU **********₩n");
    printf("** 1. 날짜 변경       **₩n");
    printf("** 2. 시간 변경       **₩n");
    printf("** 3. 변경 종료       **₩n");
    printf("**************************₩n");
    printf("선택하세요: ");
    scanf("%d", &option);
    return option;
}

void EditBooker() {
    char BookerName[16];
    printf("예약자 이름: ");
    scanf("%s", BookerName);
    int option = 0;

    Booker_information* editNode = SearchBooker(BookerName);
    // 변경하려는 예약자가 연결리스트에 존재하면
    if (editNode) {
        while (1) {
            option = EditMenu();// 변경메뉴 출력 후 선택
            switch (option) {
            case 1:// 날짜 변경
            {
                char EditDate[16];
                printf("변경할 날짜: ");
                scanf("%s", EditDate);
                strcpy(editNode->BookerDate, EditDate);// 변경할 날짜를 기존 노드에
                복사해서 업데이트
                break;
            }
            case 2:// 시간 변경
            {
                char EditTime[16];
                printf("변경할 시간: ");
                scanf("%s", EditTime);
                strcpy(editNode->BookerTime, EditTime);// 변경할 시간을 기존 노드에
                복사해서 업데이트
                break;
            }
            case 3:// 변경 종료
                printf("₩n성공적으로 변경되었습니다.₩n₩n");
                return;
            default:
                printf("ERROR : Input error! Try again.₩n₩n");
            }
        }
    }
    // 변경하려는 예약자가 연결리스트에 존재하지 않으면
    else {
        printf("₩n해당 예약자는 존재하지 않습니다.₩n₩n");
        return;
    }
}
```

Code & Result Screen

〈 예약 리스트 보기〉

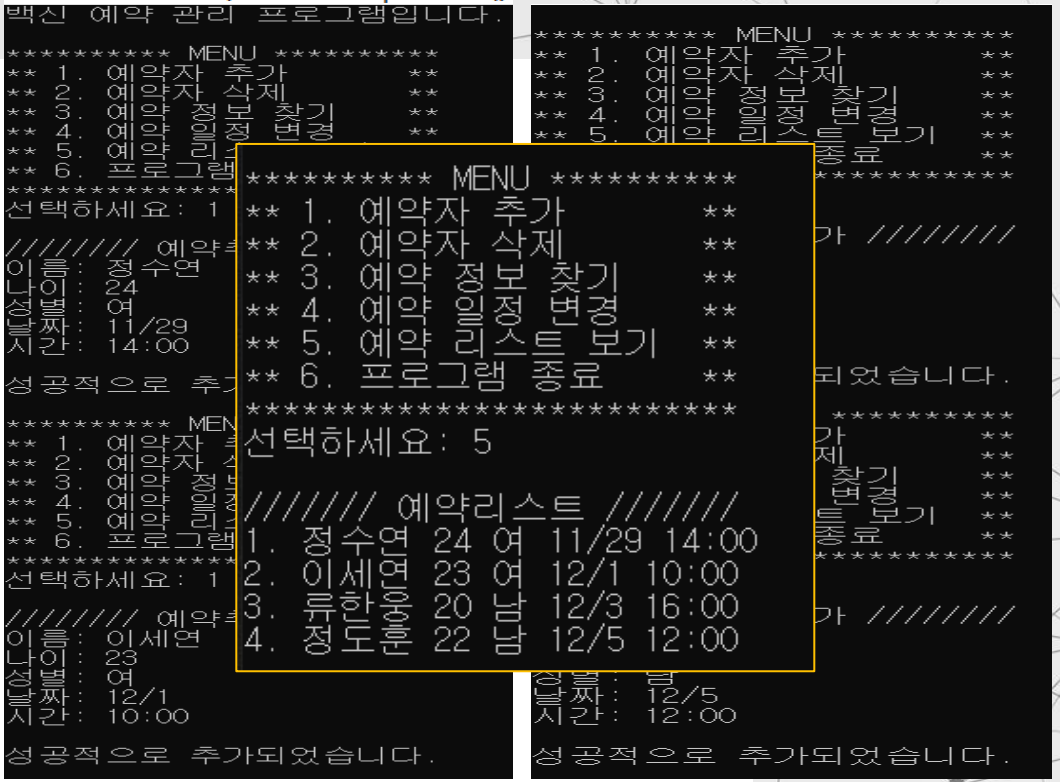
```
void BookerList() {
    // 리스트가 비었다면 종료
    if (!head) {
        printf("예약자 리스트가 비었습니다.₩n₩n");
        return;
    }

    printf("/////// 예약리스트 ///////₩n");
    Booker_information* tmp = head;
    // 연결리스트의 head부터 tail까지 반복
    for (int i = 1; tmp != NULL; i++) {
        printf("%d. %s %s %s %s %s₩n", i, tmp->BookerName,
        tmp->BookerAge, tmp->BookerSex, tmp->BookerDate, tmp-
        >BookerTime);// 예약자 정보 출력
        tmp = tmp->next;// 연결리스트의 다음 노드로 이동
    }
    printf("₩n");
    return;
}
```

〈메뉴 출력〉

```
int SelectMenu() {
    int option = 0;
    printf("********** MENU **********₩n");
    printf("** 1. 예약자 추가      **₩n");
    printf("** 2. 예약자 삭제      **₩n");
    printf("** 3. 예약 정보 찾기   **₩n");
    printf("** 4. 예약 일정 변경   **₩n");
    printf("** 5. 예약 리스트 보기 **₩n");
    printf("** 6. 프로그램 종료    **₩n");
    printf("**************************₩n");
    printf("선택하세요: ");
    scanf("%d", &option);
    printf("₩n");
    return option;
}
```

Code(main) & Result Screen

```
int main() {
    int option;
    printf("백신 예약 관리 프로그램입니다.₩n₩n");
    while (1) {
        option = SelectMenu();// 프로그램 메뉴 출력 후 선택
        switch (option) {
        case 1:// 예약자 추가
        {
            char BookerName[16];
            char BookerAge[16];
            char BookerSex[16];
            char BookerDate[16];
            char BookerTime[16];
            printf("//////// 예약추가 ////////₩n");
            printf("이름: ");
            scanf("%s", BookerName);
            printf("나이: ");
            scanf("%s", BookerAge);
            printf("성별: ");
            scanf("%s", BookerSex);
            printf("날짜: ");
            scanf("%s", BookerDate);
            printf("시간: ");
            scanf("%s", BookerTime);
            printf("₩n");
            AddBooker(BookerName, BookerAge, BookerSex, BookerDate, BookerTime);
            break;
}
```



Code(main) & Result Screen

```
case 2:// 예약자 삭제
{
   char BookerName[16];
   printf("//////// 예약삭제 ////////₩n");
   printf("삭제할 이름: ");
   scanf("%s", BookerName);
   printf("₩n");
   DeleteBooker(BookerName);
   break;
}
```

```
********** MENU **********
** 1. 예약자 추가          **
** 2. 예약자 삭제          **
** 3. 예약 정보 찾기       **
** 4. 예약 일정 변경       **
** 5. 예약 리스트 보기     **
** 6. 프로그램 종료        **
*************************
선택하세요: 2

//////// 예약삭제 ////////
삭제할 이름: 정수연

성공적으로 삭제되었습니다.

********** MENU **********
** 1. 예약자 추가          **
** 2. 예약자 삭제          **
** 3. 예약 정보 찾기       **
** 4. 예약 일정 변경       **
** 5. 예약 리스트 보기     **
** 6. 프로그램 종료        **
*************************
선택하세요: 5

/////// 예약리스트 ///////
1. 이세연 23 여 12/1 10:00
2. 류한중 20 남 12/3 16:00
3. 정도훈 22 남 12/5 12:00
```

```
********** MENU **********
** 1. 예약자 추가          **
** 2. 예약자 삭제          **
** 3. 예약 정보 찾기       **
** 4. 예약 일정 변경       **
** 5. 예약 리스트 보기     **
** 6. 프로그램 종료        **
*************************
선택하세요: 2

//////// 예약삭제 ////////
삭제할 이름: 이하이

해당 예약자는 존재하지 않습니다.
```

Code(main) & Result Screen

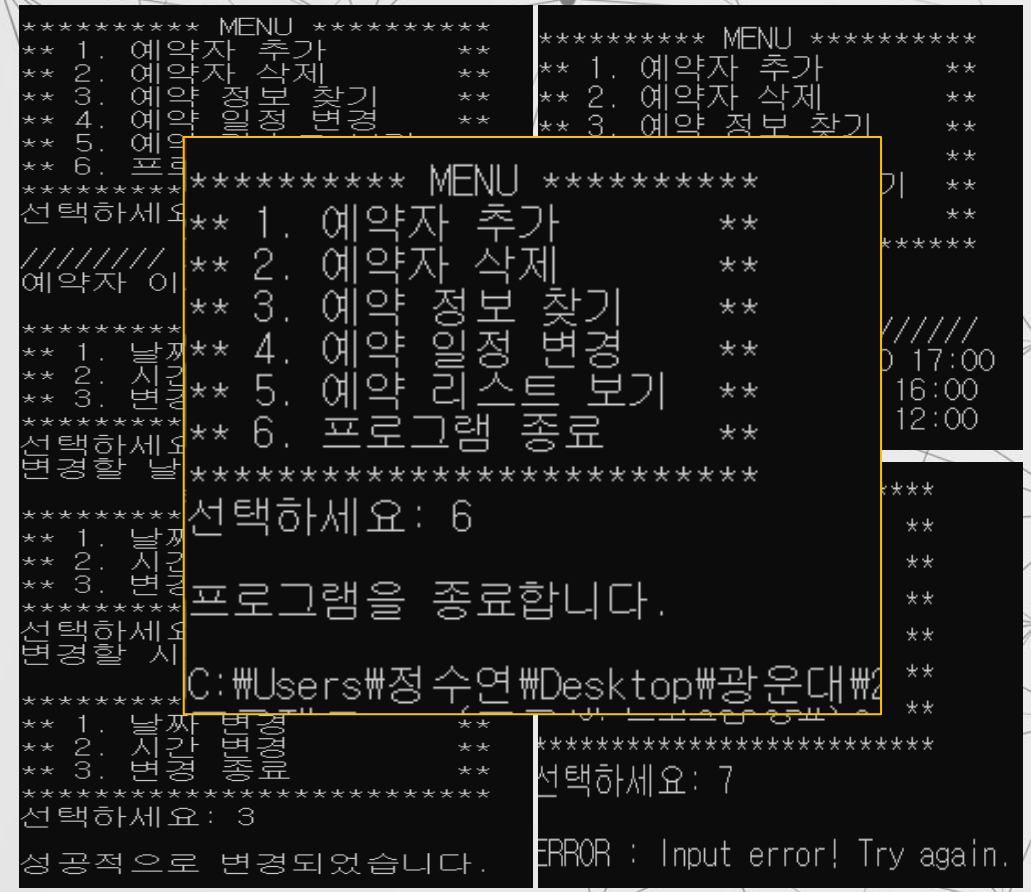
```
case 3:// 예약 정보 찾기
{
    char BookerName[16];
    printf("//////// 예약검색 ////////₩n");
    printf("예약자 이름: ");
    scanf("%s", BookerName);

    Booker_information* booker = SearchBooker(BookerName);
    // 예약자가 연결리스트에 존재하면 예약 정보 출력
    if (booker) {
        printf("₩n-------- 예약정보 --------₩n");
        printf("%s %s %s %s %s₩n₩n", booker->BookerName,
        booker->BookerAge, booker->BookerSex, booker->BookerDate,
        booker->BookerTime);
    }
    else
        printf("₩n찾는 예약자가 없습니다.₩n₩n");
    break;
}
```

```
********** MENU **********
** 1. 예약자 추가        **
** 2. 예약자 삭제        **
** 3. 예약 정보 찾기     **
** 4. 예약 일정 변경     **
** 5. 예약 리스트 보기   **
** 6. 프로그램 종료      **
*****************************
선택하세요: 3

//////// 예약검색 ////////
예약자 이름: 류한웅

-------- 예약정보 --------
류한웅 20 남 12/3 16:00
```

```
********** MENU **********
** 1. 예약자 추가        **
** 2. 예약자 삭제        **
** 3. 예약 정보 찾기     **
** 4. 예약 일정 변경     **
** 5. 예약 리스트 보기   **
** 6. 프로그램 종료      **
*****************************
선택하세요: 3

//////// 예약검색 ////////
예약자 이름: 이하이

찾는 예약자가 없습니다.
```

Code(main) & Result Screen

```
case 4:// 예약 일정 변경
        printf("//////// 예약변경 ////////₩n");
        EditBooker();
        break;
case 5:// 예약 리스트 출력
        BookerList();
        break;
case 6:// 프로그램 종료
        printf("프로그램을 종료합니다.₩n");
        return 0;
default:
        printf("ERROR : Input error! Try
        again.₩n₩n");
    }
  }
}
```



05
회의록 & 고찰

회의록

- 10월 20일 회의 - 구조체 파일 입출력 링크드리스트 등을 공부하며 시험공부를 하기로 함
- 11월 11일 회의 - 구조도를 작성하고 코드 작성을 시작함. 마감이 한 달 안 남았다는 것을 인지함
- 11월 24일 회의 - 코드 작성의 마무리 단계에 접어들어 각 기능을 테스트하고 수정함
- 11월 26일 회의 - 모든 프로젝트를 마무리하고 보고서 작성 및 ppt 담장자와 발표자를 정하는 시간을 가짐

고찰

- 팀프로젝트의 결과물이 화려해야 한다는 생각에 마음만 앞서 여러가지 경우의 수를 생각하지 않고 코딩하여 많은 오류를 야기하였다. 따라서, 다같이 프로젝트를 함에 있어서 퍼포먼스보다는 안정성을 우선으로 해야 한다는 것을 깨달았다.
- 연결 리스트를 통하여 사용자에 대한 많은 정보를 하나의 노드에 저장할 수 있었고 배열을 남발하는 것보다 훨씬 편리하고 유지보수가 쉽다는 것을 경험을 통해 알게 되었다.

06

참고문헌 및 사이트

참고문헌 및 사이트

https://blog.naver.com/ljk2231/222160409724

https://blog.naver.com/ljk2231/222175431341

https://a6ly.dev/65

https://kldp.org/node/161651

https://dojang.io/mod/page/view.php?id=645

Thank you.