당직 서열 근무표

**설계 문서 수정 1판**

**A05**

**Workingcheck.txt 용도 변경 (6페이지)**

**리턴 처리 변경 (16페이지)**

**2차 기획 수정 추가(5페이지)**

**주말근무 설계 수정 (10~14페이지)**

**보라색 – 설계 문서 수정 내용**

201711836 송호영

201811168 김재연

201910162 문무현

202011396 황정승

202111307 서현하

플랫폼 : window 프로그래밍 언어 : C++

# 1차 기획 보강 – [명단 삭제 – 미등록 근무자 삭제 오류]

기존에 구현되어 있는 함수 중 중복되는 부분을 삭제한다.

## 기존 함수 1) Memberlist.cpp 의 Delete()

입력: 없음 출력:

1. “삭제할 근무자의 아이디를 입력하시오. 입력: “
2. 아이디가 유효하지 않은 경우 – “아이디가 유효하지 않습니다.”, (string) “-1”.
3. 아이디가 유효한 경우 – “해당 아이디 및 정보가 삭제되었습니다.”, (string) ID

기능: 사용자가 삭제하고자 하는 아이디의 파일 내 존재 유무를 확인하고, 없으면 입력이 유효하 지 않다고 알려주고, 있으면 아이디를 삭제함.

## 기존 함수 2) Excute.cpp 의 listMenu()의 switch문 중 case 51

입력: 없음.

출력: 삭제하고자 하는 인원이 근무표에 있는 경우 – “현재 근무표에 등록되어 있는 근무자입니다. 삭제하시겠습니까?”

기능: 사용자가 삭제하고자 하는 아이디가 파일 내에 존재하지 않으면 유효하지 않다고 사용자에 게 알려주고, 명단에 등록되어 있는 경우 아이디를 명단에서도 지워줌.

## 구현 방법

listMenu() 내 case 51을 통해 Delete() 진행 중 아이디가 파일에 없는 경우, index가 -1이 되어 아 이디가 없는 경우의 조건분기로 들어간 뒤 (string) -1을 리턴해준다. index가 -1이 아닌 경우 erase를 이용해 아이디를 저장하고 있는 memberList 내에서 해당 아이디를 지워준다. Delete()의

리턴된 값이 -1이면 아이디가 유효하지 않다고 출력하고, 리턴된 값이 -1이 아니면 이미 근무표 에서 삭제할 것인지 물어본다. Y의 경우, 날짜의 수 만큼 반복문을 돌며 해당 날짜에 지울 아이디 가 있으면 아이디도 지우고 날짜의 상태도 vacant로 바꿔준다. N인 경우 그냥 리턴한다.

# 1차 기획 보강 – [명단 관리 규칙 6-1-1. 작성 중인 근무표가 있는 경우 명단 등록을 진 행할 수 없습니다.]

기존에 사용하던 변수와 함수를 통해 구현한다.

## 기존 변수 int isWorking

isWorking 변수는 프로그램이 실행될 때 명단 파일의 첫 줄에서 정수 값을 읽어온다.

이 값은 작성 중인 근무표가 있다면 해당 연월을 가지며, 작성 중인 근무표가 없다면 -1을 가진 다.

## 기존 함수 void listMenu()

listMenu() 함수는 명단 관리 메뉴를 구현한다. 사용자의 선택에 따라 명단 보기, 명단 등록, 명단 삭제 기능을 수행한다.

## 구현 방법

listMenu() 함수의 명단 등록 기능 진입 이전에 조건문을 통하여 isWorking 변수의 값을 확인한다. 작성 중인 근무표가 있는 경우(-1이 아닌 경우) 해당 기능에 진입하지 못하도록 제한한다.

또한 기획서의 8. 예외 처리 규칙 – 오류에 따라 작성 시작이 불가한 경우 listMenu() 함수로 돌 아갈 수 있도록 구현한다.

# 2차 기획 수정 – [명단 관리 규칙 6-1-2. 작성 중인 근무표가 있는 경우 유효 근무자는 명단 삭제가 불가능 합니다.]

## 기존 변수

vector<pair<UserInformation, int>> validlist;

근무에 투입이 가능한 유효 근무자를 저장하는 벡터이다.

근무자의 이름과 근무 횟수가 pair로 입력된다.

## 구현 방법

삭제할 근무자의 아이디를 입력받은 후, 현재 연월의 유효 근무자를 저장하는 validlist 벡터를 먼저 확인한다. Search 함수를 통해서 벡터에 포함된 근무자라면 삭제할 수 없도록 하고, 벡터에 포함되지 않은 근무자라면 기존처럼 삭제를 진행한다.

# 1차 기획 보강 – [근무일을 다시 선택해야 하는 경우(p.21)의 경고 메시지가 제대로 출력 되지 않음]

## 기존 변수

지역 변수 Pair<User Information, int> target: 현재 사용자가 입력한 근무자이다. first에는 ID와 시 작연월, second에는 근무투입횟수가 저장된다.

## 기존 함수

Void ChoiceDay(): 근무표 작성을 시작하는 함수이다. 이 함수 내에 패스 테스트를 작성한 부분을 수정한다.

## 추가 사항

Workingcheck.txt: ~~근무투입횟수~~ 이전에 패스한 기록을 저장한다. 이는 확정되기 전까지 유동적이므로 매번 불러오고, 매번 저장해야 한다.

패스 조건을 만족했으며, 수정하는 경우 이전에 패스한 근무자들의 조건을 다시 검사하여 체크해 야 하는 부분을 수정한다.

이 때 패스 조건은 6-2-6으로, 근무표 상에 남은 근무일이 후임보다 적은 근무자는 패스를 선택 할 수 있어야 한다.

## 구현 방법

현재 남은 근무일은 int vcount, 투입될 수 있는 후임의 수는 int tcount에 저장된다. vcount는 캘린 더를 모두 돌며 비어 있는 배열의 수를 체크하면 되고, tcount는 시작달과 근무투입횟수를 target 과 비교하면 되는데, 시작달이 target보다 늦고 근무투입횟수가 target보다 작거나 같은 후임의 수 를 저장한다.

tcount<vcount인 경우 ChoiceDay()를 종료한다.

tcount>=vcount인 패스 조건을 만족하고 수정하는 경우이면, 현재 target을 캘린더에서 제외시킨

다.

그 후 패스가 완료되었다고 완료 메시지를 출력한 후, second 값을 int PASS(999)값으로 변경시킨 다.

이전에 패스한 근무자들의 조건을 다시 검사한다.

target이 아니면서 Workingcheck.txt에 저장된 ~~second 값이 PASS인~~ 근무자를 선택해 target으로 재설정한다.

바뀐 target과 vcount와 tcount를 위와 똑같이 계산하고 비교해서 패스 조건(6-2-6)을 만족하는지 체크한다.

tcount<vcount인 경우 target의 ID와 함께 이 근무자는 근무일을 다시 선택해야 한다고 다음과 같은 메시지를 출력한다. “[id] 근무자는 근무일을 다시 선택해야 합니다. - 패스 조건 불만족”

tcount>=vcount인 경우 변경할 사항은 없다. target을 캘린더에서 제외시킨다.

Target의 근무투입횟수를 1 줄인다. Workingcheck.txt를 업데이트 한다.

# 2차 기획 확장 – [파일 입출력]

## CalendarFileInput 함수

입력: (int) 불러올 달, (string\*) 팀별 근무일정, (vector<Team>) 팀 목록, (int\*) 일별 상태 출력: Boolean (정상적인 저장파일이 존재할 경우 True/ 그 외 False)

기능: 저장파일을 확인하고 존재할 경우 각 정보를 입력 받은 3개의 배열 (팀별 근무일정, 팀 목 록, 일별 상태)에 저장하고 True를 출력하며, 존재하지 않을 경우 새로운 저장파일을 생성하고 False를 출력함.

## CalendarFileOutput 함수

입력: (int) 불러올 달, (string\*) 팀별 근무일정, (vector<Team>) 팀 목록, (int\*) 일별 상태 출력: Void

기능: 저장파일을 확인하고 존재할 경우 각 정보를 입력 받은 3개의 배열 (팀별 근무일정, 팀 목 록, 일별 상태)정보를 저장파일에 문법에 맞게 저장.

**저장파일 문법** (가독성을 위해 공백은 \_ 로 표현함.)

요약: 최초 팀 정보를 저장하며 이후 구분자”-“를 사용 후 근무일 정보를 저장함.

팀 이름1\_1번 ID\_1번 시작 달 2번 ID\_2번 시작 달\_3번 ID\_3번 시작 달 팀 이름2\_1번 ID\_1번 시작 달 2번 ID\_2번 시작 달\_3번 ID\_3번 시작 달

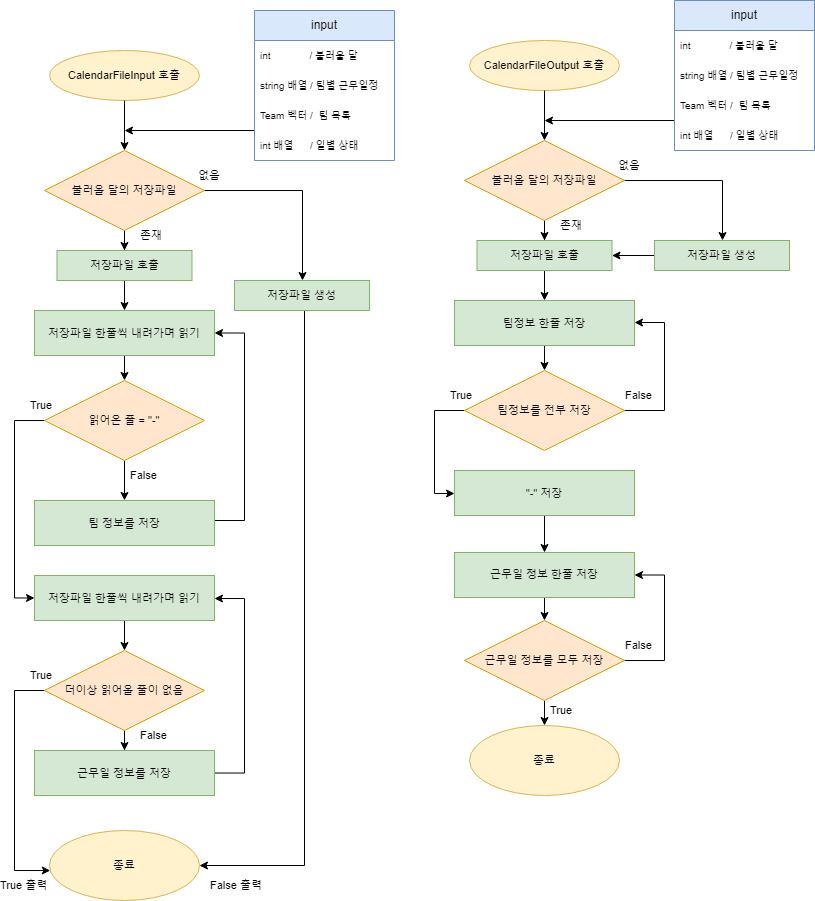
팀 이름3\_1번 ID\_1번 시작 달 2번 ID\_2번 시작 달\_3번 ID\_3번 시작 달 …

-

1\_팀 이름1\_1

2\_팀 이름2\_0 …

**함수 설계** (좌 CalendarFileInput / 우 CalendarFileOutput)



사용자의 저장파일 사용을 상정하지 않았음으로 문법이 달라 나타날 수 있는 예외처리는 시행하 지 않음.

# 2차 기획 확장 – [주말 근무를 평일 근무의 2회로 취급한다.]

근무 투입되는 인원의 수만큼 주말근무여부를 확인하는 자료구조를 만들어 주말 근무를 선택하면, 그 다음 근무 선택에서는 제외되도록 한다.

## 추가 사항

전역변수 vector<pair<string, bool>> pre\_Shift– 이전 사이클에서 주말 근무 여부를 확인해서 저장하고있는 자료구조. 최초 초기화 시 validlist의 ID를 전부 받아온 뒤, occupied된 날짜의 해당 ID의 bool값을 true로 저장한다. 사이클 종료 후 cur\_Shift값이 복사된다.

전역변수 vector<pair<string, bool>> cur\_Shift – 현재 사이클에서 주말 근무 여부를 확인해서 저장하는 자료구조. 사이클이 끝난 뒤 pre\_Shift에 복사한다.

ID(string) 주말근무여부(bool) 형식으로 저장된다.

## 구현 방법

ChoiceDay 함수에 진입하면, 동년 동월 근무표 체크 이후 주말근무여부를 초기화 한다. 주말 근무 한 인원들을 전부 체크해서 pre\_Shift에 받아온 뒤, 주말 근무 여부를 전부 true로 바꾸어준다.

근무 날짜를 선택해서 점유할 때, 이전 근무 선택에서의 주말근무여부를 체크해 진입 여부를 결정한다.

점유할 날짜의 주말 여부를 체크해서 주말이 맞으면 cur\_Shift에서 ID를 찾아 주말 근무 여부를 true로 바꾸어준다. 평일의 경우는 반대로 cur\_Shift에서 ID를 찾아 false로 바꾸어준다.

한 사이클이 끝난 뒤, cur\_Shift의 정보를 pre\_Shift에 저장하고, 해당 인원들의 근무 횟수를 +1 해준다.

# 2차 기획 확장 – [근무 인원 설정]

## 기존 변수

vector<pair<UserInformation, int>> validlist;

: 근무에 투입이 가능한 유효 근무자를 저장하는 벡터이다.

근무자의 이름과 근무 횟수가 pair로 입력된다.

struct Team

{

string TeamName = "ERROR"; // 조 이름

UserInformation userinfo[3]; // 각 조에 해당하는 근무자의 이름

};

: 조를 저장하는 구조체이다. int hopeday;

: 근무자가 근무를 희망하는 날짜로 입력받는 값이다. int STATE[DAYMAX];

: 해당 날짜의 상태를 나타낸다.

vacant : 비어있음 , occupied : 선택함 , confirmed : 확정됨

string ID[DAYMAX];

: 배열의 인덱스를 날짜로 지정하고 해당 날짜의 근무자를 나타내는 배열이다. vector<Team> teamlist;

: 각 조의 정보가 각 인덱스에 들어있는 벡터이다.

## 추가 변수

int dayworker;

: 일일 근무인원이 몇명인지 입력 받을 때, 값을 저장할 변수이다. int totalworkers;

: 총 근무 인원을 받아올 변수이다.

int teamindex;

: 조를 구분해주는 변수이다.

int remainder;

: totalworkers % dayworker 의 값을 저장해주는 변수로 조에 안들어가는 인원이 있는지 확인해준다.

int remtemp;

: remainder 변수를 잠시 저장해주는 변수로 구현에 필요해 선언하였다.

int tempidx;

: 조를 짜는 과정에서 구현에 필요한 변수로 teamList의 userinfo의 인덱스값을 저장한다.

char ans;

: 처음 입력받을 일일근무인원을 저장할 변수이다.

## 구현 방법

총 4단계로 진행된다.

1. 몇 명이 조를 맺을지 입력 받는 부분
2. 입력 받은 값으로 조를 짜는 부분
3. 조 출력하는 부분
4. 근무일을 선택했을 때, 같은 조끼리는 공유하는 부분

이 과정을 하기 전에 신청된 날짜에 해당하는 조의 최선임을 알기 위해서 team과 teamList.Teamname을 비교하여 ID값에 그 조에 해당하는 최선임을 넣어두었다.

## 몇 명이 조를 맺을지 입력 받는 부분

cout << "일일근무인원원을 입력하십시오. 입력 : " ;

\_getch()로 입력받아서 조건문을 시행시켜 1 2 3 이 아닌 값이 들어오면 에러문을 출력한다.

입력을 받을 때, 입력 받는 값은 1 ~ 3 사이의 정수 값이어야 한다. ( 조건문 ) 입력 받은 값이 문제가 없다고 판단 될 시 -> dayworker 변수에 값을 넣는다.

입력 받은 값이 문제가 있다고 판단 될 시 -> 에러문과 함께 이전 페이지로 돌아간다.

처음 코드가 실행되기 전에 teamList 의 size를 확인하여서 조가 만들어 졌는지 먼저 확인한 후에 입력 값을 받는다.

## 입력 받은 값으로 조를 짜는 부분

totalworkers= validlist.size(); 를 통해서 총 근무 인원을 받아온다. 조건문을 이용하여서 totalworkers % dayworker == 0 인 경우와 아닌 경우로 나눈다.

if(true) : 반복문을 이용해서 앞에서부터 dayworker씩 잘라서 team 구조체에 넣는다. if(false) : 똑같이 반복문을 이용하지만 for문에서 i의 시작점을 totalworkers % dayworker 로 해서 나머지만큼 배열에서 앞부분을 빼준다.

처음 코드가 실행되기 전에 teamList 의 size를 확인하여서 조가 만들어 졌는지 먼저 확인한 후에 조를 짠다.

## 조 출력하는 부분

반복문을 사용해서 Team의 userinformation을 조별로 나눠서 출력해준다. for(int i =0; i< team.size(); i++)

for(int j=0; j<dayworker; j++) cout << ID;

\*문법을 확인하지 않은 코드이다.

1. **근무일을 선택했을 때, 같은 조끼리는 공유하는 부분** 근무일을 선택, 수정 하거나 근무일을 확정 할 때, 기존에는 STATE[hopeday] = occupied;

ID[hopeday] = id;

이런식으로 근무일을 선택 한 날짜의 상태를 바꿔주고 id값을 넣어주었다.

이 방식을 조금 바꿔서

1. 아이디 입력하고 근무일 선택까지 한 후에 그 아이디에 해당하는 조의 teamlist 내에서의 인덱스 값을 찾는다.

int teamindex;

for(int i=0; i<teamlist.size(); i++)

if(id == teamlist[i].userinfo[0]) teamindex = i;

if(id == teamlist[i].userinfo[1]) teamindex = i;

if(id == teamlist[i].userinfo[2]) teamindex = i;

\*문법을 확인하지 않은 코드이다.

전부 반복문을 사용하여서 작성하였다.

1. teamlist[teamindex]에서 userinfo를 받아와서 전부 validlist의 pair에서 두 번째 값인 근무

횟수를 +1 해준다.

1. ID 배열에서 선택한 날짜를 조에서 가장 계급이 높은 근무자의 id를 넣는다.

\*문법을 확인하지 않은 코드이다.

1. STATE[hopeday] = occupied;

\*확정을 할 때에도 같은 방식을 사용한다.

# 2차 기획 확장 – 연도 제한

## 기존 변수

Bool checkdate(string str) : str이 연월 입력 의미 규칙(7-3-2)에 부합하면 true를, 부합하지 않으면 false를 리턴한다.

## 추가 변수

Int checkyear : 지역변수이며, str에서 연도를 의미하는 4자리만 저장한 값이다.

현재 연도는 상관없이 월 01~12만 체크했던 의미 규칙을 1900년 01~2099년 12월으로 제한하기 위해서 연도 제한 조건을 추가한다.

## 구현 방법

str 문자열에서 연도만 숫자로 변환한다. str에서 연도는 str[0]~str[3]의 숫자이다. 변환한 값은 지 역변수 int checkyear를 만들어서 저장한다.

checkyear > 2000이면, |2000-checkyear| 값이 1부터 99까지 중의 값인지 확인한다. 아니면 false를, 맞으면 ~~true를 리턴한다.~~ 다음 문장(월 조건 판별)을 실행한다.

checkyear <= 2000이면, |2000-checkyear| 값이 0부터 100까지 중의 값인지 확인한다. 아니면 false

를, 맞으면 ~~true를 리턴한다.~~ 다음 문장(월 조건 판별)을 실행한다.