설계과제 소스파일

```
/**********************************
        프로젝트 과제#1
        2009112441 김영빈
        항공기 예약관리
        Reserve_Plane.cpp 메인함수 포함 파일
**********************************/

#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <string.h>
#include "Reserve_Plane.h"

void main(void)
{
        FILE * fp=NULL, *fp2= NULL;
        Plane * _newP = NULL, *P = NULL;
        Reserv* _newR = NULL;
        char Command[3], flight_Number[5], name[21];
        int error_check_cnt=0;
        int cnt= 0;
        C_list = (Customer *)malloc(sizeof(Customer));
        C_list->next = NULL;
        C_list->prev = NULL;


        // 파일 오픈
        while((fp = fopen("SAMPLEIN.P1", "r"))==NULL){
                printf("Fail to open file.₩n");
                error_check_cnt++;
                if(error_check_cnt==5) exit(1);
        }

        while(fscanf(fp, "%s", Command)!=EOF){ // 몸체 부분

                 // 명령 받아오기
                Command[0]      =       toupper(Command[0]);      Command[1]      =
toupper(Command[1]);
                printf("[%d]Command : %s₩n",cnt++, Command);
```

```
		//getch();
		if(!strcmp(Command, "IF")){
			_newP = (Plane *)malloc(sizeof(Plane));

			if(_newP == NULL){ // 메모리 할당 오류
				printf("! Fail to allocate memory.₩n");
				continue;
			}
			if(fscanf(fp, "%s %s %s %s", _newP->flight_Number,
_newP->from_city,_newP->to_city,_newP->capacity) != 4)
			{ // 입력 오류
				printf("! 비어있는 필드가 있습니다.₩n");
				free(_newP); // 할당 필드 반환
				continue;
			}

			insert_Plane(&P, _newP);
			_newP = NULL; // 초기화
		}
		else if(!strcmp(Command, "DF")){
			if(fscanf(fp, "%s", flight_Number) != 1)
			{ // 입력 오류
				printf("! 비어있는 필드가 있습니다.₩n");
				continue;
			}
			delete_Plane(&P,flight_Number);
		}
		else if(!strcmp(Command, "RP")){

			if(fscanf(fp, "%s %s", name, flight_Number) != 2)
			{ // 입력 오류
				printf("! 비어있는 필드가 있습니다₩n");
				continue;
			}

			reserv_Plane(P, name, flight_Number);

		}
		else if(!strcmp(Command, "PA")){
			print_All(P);
		}
```

```
		else if(!strcmp(Command, "CP")){
			if(fscanf(fp, "%s %s", name, flight_Number) != 2)
			{ // 입력 오류
				printf("! 비어있는 필드가 있습니다₩n");
				continue;
			}
			Cancel_res(P, name, flight_Number);
		}
		else if(!strcmp(Command, "PF")){
			print_Flight(P);
		}
		else if(!strcmp(Command, "PR")){
			if(fscanf(fp, "%s", flight_Number) != 1)
			{ // 입력 오류
				printf("! 비어있는 필드가 있습니다₩n");
				continue;
			}
			print_FlightC(P, flight_Number);
		}
		else if(!strcmp(Command, "PP")){
			if(fscanf(fp, "%s", name) != 1)
			{ // 입력 오류
				printf("! 비어있는 필드가 있습니다₩n");
				continue;
			}
			print_Customer(name);
		}
		else {
			printf("! 명령어가 잘못 되었습니다.₩n");
			continue;
		}

	}
	fclose(fp);
	while(P){ delete_Plane(&P, P->flight_Number); } // 힙 free
}

void insert_Plane(Plane ** P, Plane * _new)
{
	Plane * loop = *P;
```

```
	// 노드 초기화
	_new->res_List= _new->front = _new->rear = NULL;
	_new->next = _new->prev=NULL;
	_new->isFull = _new->res_Num = 0;

	if(!loop){
		*P = _new;
	} else {
		while(loop) // 저 -> 고 배열
		{
			if(!strcmp(loop->flight_Number, _new->flight_Number)) {
	printf("이미 [%s] 에 대한 정보가 존재합니다.₩n"); return ; }
			if(strcmp(loop->flight_Number, _new->flight_Number)<0)
break;
			loop = loop->next;
		}

		_new->isFull = FALSE;
		_new->next = loop->next;
		if(_new->next) _new->next->prev = _new;
		loop->next = _new;
		_new->prev = loop;
	}
	printf("# [%s] 항공편이 등록되었습니다.(탑승 가능 인원: %s)₩n",
_new->flight_Number, _new->capacity);
	printf(" %s ▷▶▷ %s₩n", _new->from_city, _new->to_city);
}

void delete_Plane(Plane ** P, char * flight_Number)
{
	Plane * temp = *P;
	//	Customer * ctemp;
	Reserv * rtemp, * delete_node;

	if(!temp){ getch(); printf("등록된 비행편이 없습니다.₩n"); return ; }
	while(strcmp(flight_Number, temp->flight_Number)){temp = temp->next;
if(!temp) break;}// 요청 번호 검색

	if(!temp){ // 존재하지 않을때
		printf("요청하신 [%s] 비행편이 존재 하지 않습니다.₩n",
flight_Number);
```

```
		return ;
	}
	rtemp = temp->front;

	while(rtemp){
		delete_node = rtemp;
		if(rtemp->thisC_prev)	rtemp->thisC_prev->thisC_next	=
rtemp->thisC_next;
		if(rtemp->thisC_next)	rtemp->thisC_next->thisC_prev	=
rtemp->thisC_prev;
		if(rtemp->next_res) rtemp->next_res->prev_res = rtemp->prev_res;
		if(rtemp->prev_res) rtemp->prev_res->next_res = rtemp->next_res;
		if(rtemp->ticket->res_List->prev_res	==
rtemp->ticket->res_List->next_res){	rtemp->ticket->res_List = NULL;	}
		if(rtemp == rtemp->ticket->res_List){	rtemp->ticket->res_List	=
rtemp->next_res;	}
		if(rtemp	==	rtemp->ticket->front){	rtemp->ticket->front	=
rtemp->ticket->front->next_res; }
		rtemp = rtemp->thisC_next;
		free(delete_node);
	}

	rtemp = temp->res_List;

	while(rtemp){
		delete_node = rtemp;
		if(rtemp->thisC_prev)	rtemp->thisC_prev->thisC_next	=
rtemp->thisC_next;
		if(rtemp->thisC_next)	rtemp->thisC_next->thisC_prev	=
rtemp->thisC_prev;
		if(rtemp->next_res) rtemp->next_res->prev_res = rtemp->prev_res;
		if(rtemp->prev_res) rtemp->prev_res->next_res = rtemp->next_res;
		if(rtemp->ticket->res_List->prev_res	==
rtemp->ticket->res_List->next_res){	rtemp->ticket->res_List = NULL;	}
		if(rtemp == rtemp->ticket->res_List){	rtemp->ticket->res_List	=
rtemp->next_res;	}
		if(rtemp	==	rtemp->ticket->front){	rtemp->ticket->front	=
rtemp->ticket->front->next_res; }
		rtemp = rtemp->thisC_next;
		free(delete_node);
	}
```

```
        // 삭제
        if(temp->prev){ temp->prev->next = temp->next; }
        if(temp->next){ temp->next->prev = temp->prev; }
        if(*P == temp){ *P = temp->next; temp->prev = NULL;}
        if(!(*P)->next)){ *P=NULL; }

        free(temp); // 메모리 반환


}

void  reserv_Plane(Plane * P, char * name, char * flight_Number)
{
        Plane * ploop = P;
        Reserv * rloop, *_newR;
        Customer * cloop = C_list, *_newC;

        /*
        while(ploop){ printf("->%s %s₩n", ploop->flight_Number, ploop->capacity);
ploop= ploop->next;}
        ploop = P;
        */

        while(cloop && strcmp(cloop->name, name)) cloop = cloop->next;

        if(!cloop){
                _newC = add_Customer(name); // 총 고객 리스트에 저장
                if(!_newC) return ;
        } else _newC = cloop;

        while(ploop && strcmp(ploop->flight_Number, flight_Number)) ploop =
ploop->next;

        if(!ploop){ printf("[%s] 항공편이 준비되어 있지 않습니
다.₩n",flight_Number); return ;}

        // 예약자 일치여부 찾기
        rloop = ploop->res_List;

        if(rloop){
```

```
                if(!strcmp(rloop->host->name, name)){ printf("%s 고객님은 이미 [%s] 항공편에 예약되어 있습니다.₩n", name, flight_Number); return ; }
                else rloop = rloop->next_res;

                while(strcmp(rloop->host->name,                    name)){ if(rloop==ploop->res_List) break; rloop= rloop->next_res;   }
                if(rloop != ploop->res_List){    printf("%s 고객님은 이미 [%s] 항공편에 예약되어 있습니다.₩n", name, flight_Number); return ;}
        }

        // 대기자 일치여부 찾기
        rloop = ploop->front;
        if(rloop){
                while(rloop  &&  strcmp(rloop->host->name,  name)){  rloop= rloop->next_res; }
                if(rloop){        printf("%s 고객님은 이미 [%s] 항공편에 대기중에 있습니다.₩n", name, flight_Number); return ;}
        }


        if(!(_newR  =  (Reserv  *)malloc(sizeof(Reserv)))){printf("Fail  to  allocate memory.₩n"); return ;}

        // 예약하기
        if(!ploop->isFull){

                // 포인터 정리
                _newR->host = _newC;
                _newR->ticket = ploop;
                add_Reserv(&(ploop->res_List), _newR, name);
                conf_this(_newC, _newR);

                // 자료 정리
                _newR->isRes = TRUE;

                (ploop->res_Num)++;
                if(compare_C(ploop->capacity, ploop->res_Num)==2) ploop->isFull = TRUE;


        } else {
```

```
                // 대기
                _newR->host =_newC;
                _newR->ticket = ploop;
                add_Wait(ploop, _newR);
                conf_this(_newC, _newR);


                // 자료 정리
                strcpy(_newR->host->name, name);
                _newR->isRes = FALSE;
        }

}

void conf_this(Customer * _newC, Reserv * _newR)
{
        Reserv * rloop = _newC->res;
        _newR->thisC_next = _newR->thisC_prev = NULL;

        if(!rloop){
                _newC->res = _newR;
                return ;
        }

        if(strcmp(rloop->ticket->flight_Number, _newR->ticket->flight_Number)>0){
                _newR->thisC_next = _newC->res;
                _newC->res = _newR;
                _newR->thisC_next->thisC_prev = _newR;
                return ;
        }

        while(rloop)
        {
                if(strcmp(rloop->ticket->flight_Number,
_newR->ticket->flight_Number)<0) break;
                rloop = rloop->thisC_next;
        }
        _newR->thisC_next = rloop->thisC_next;
        if(_newR->thisC_next) _newR->thisC_next->thisC_prev = _newR;
        rloop->thisC_next = _newR;
```

```
        _newR->thisC_prev = rloop;
}


void add_Reserv(Reserv ** R_head, Reserv * _newR ,char * name)
{
        Reserv * rloop = (*R_head), *before=NULL;
        strcpy(_newR->host->name, name);

        if(!rloop){
                *R_head = _newR;
                _newR->next_res = _newR->prev_res = _newR;
                return ;
        }

        rloop = rloop->next_res;

        while(strcmp(rloop->host->name, name)>0) {
                if(rloop->next_res == *R_head){rloop = rloop->next_res; break;}
                rloop = rloop->next_res; }

        _newR->prev_res = rloop->prev_res;
        _newR->next_res = rloop;
        rloop->prev_res->next_res =_newR;
        rloop->prev_res = _newR;
}

void add_Wait(Plane * P, Reserv * _newR)
{
        if(!(P->front)){

                P->front = _newR;
                P->rear = _newR;
                _newR->prev_res = _newR->next_res = NULL;
        } else {

                _newR->prev_res = P->rear;
                P->rear->next_res = _newR;
                P->rear = _newR;
                _newR->next_res = NULL;
        }
```

```
}

Customer * add_Customer(char * name)
{
        Customer * _newC= (Customer *)malloc(sizeof(Customer));
        if(!_newC) { printf("Fail to allocate memory.₩n"); return _newC; }
        strcpy(_newC->name, name);

        _newC->res = NULL;

        _newC->next = C_list;
        C_list = _newC;
        return _newC;
}

int compare_C(char * pre, int obj)
{
        int i, _pre=0;

        for(i=0; pre[i]; i++) _pre = _pre*10 + (pre[i]-'0');

        if( _pre < obj ) return 0;
        else if(_pre > obj) return 1;
        else return 2; // 같을때
}

void print_Flight(Plane * P)
{
        Plane * ploop = P;
        FILE * fp=NULL;

        if(!(fp=fopen("OUTFILE.P1", "a+"))){ printf("! Fail to open file.₩n"); }

        if(!ploop){ printf("! 등록된 항공편이 없습니다.₩n"); fprintf(fp,"! 등록된 항공편이
없습니다.₩n"); return ; }
        printf("□■□■□■□■□■□§ 비행편 리스트 §■□■□■□■□■□■₩n₩n");
        fprintf(fp,"□■□■□■□■□■□§ 비행편 리스트 §■□■□■□■□■□■₩n₩n");
        while(ploop){
                printf(" ☞   항공편: [ %s ] ", ploop->flight_Number);
                fprintf(fp," ☞   항공편: [ %s ] ", ploop->flight_Number);
                printf("        From: %s To: %s₩n", ploop->from_city, ploop->to_city
```

```
);
                fprintf(fp,"              From:  %s  To:  %s₩n", ploop->from_city,
ploop->to_city );
                printf("          예약인원 / 탑승인원 : %03d / %s", ploop->res_Num,
ploop->capacity);
                fprintf(fp,"          예약인원 / 탑승인원 : %03d / %s", ploop->res_Num,
ploop->capacity);
                if(ploop->isFull){ printf("  √예약가능₩n"); fprintf(fp,"  √예약가능₩n");
} else { printf("  ×예약불가₩n"); fprintf(fp,"  ×예약불가₩n"); }
                ploop= ploop->next;
        }
        if(!ploop){ printf("! 등록된 항공편이 없습니다.₩n"); fprintf(fp,"등록된 항공편이
없습니다.₩n"); return ; }
}

void print_All(Plane * P)
{
        Plane * ploop = P;
        Reserv * rloop = P->res_List;
        FILE * fp=NULL;

        if(!(fp=fopen("OUTFILE.P1", "a+ "))){ printf("! Fail to open file.₩n"); }

        if(!ploop){ printf("! 등록된 항공편이 없습니다.₩n"); fprintf(fp,"등록된 항공편이
없습니다.₩n"); return ; }



        printf("□■□■□■□■□■□§ 비행편 리스트 §■□■□■□■□■□■₩n₩n");
        fprintf(fp,"□■□■□■□■□■□§ 비행편 리스트 §■□■□■□■□■□■₩n₩n");
        while(ploop){
                printf(" ☞   항공편: [ %s ] ", ploop->flight_Number);
                fprintf(fp," ☞   항공편: [ %s ] ", ploop->flight_Number);
                printf("        From: %s To: %s₩n", ploop->from_city, ploop->to_city
);
                fprintf(fp,"              From:  %s  To:  %s₩n", ploop->from_city,
ploop->to_city );
                printf("          예약인원 / 탑승인원 : %03d / %s", ploop->res_Num,
ploop->capacity);
                fprintf(fp,"          예약인원 / 탑승인원 : %03d / %s", ploop->res_Num,
ploop->capacity);
```

```
                if(!ploop->isFull){  printf("    √예약가능₩n");  fprintf(fp,"    √예약가능
₩n"); } else { printf("  ×예약불가₩n"); fprintf(fp,"  ×예약불가₩n"); }
                ploop= ploop->next;
        }
        ploop = P;
        if(!ploop){ printf("! 등록된 항공편이 없습니다.₩n"); fprintf(fp,"등록된 항공편이
없습니다.₩n"); return ; }

        printf("₩n");
        fprintf(fp,"₩n");
        ploop = P;
        printf("■□■□■□■□§비행편 별 예약, 대기 리스트§■□■□■□■□₩n");
        fprintf(fp,"■□■□■□■□§비행편 별 예약, 대기 리스트§■□■□■□■□₩n");
        while(ploop){
                rloop = ploop->res_List;
                printf("₩n");
                fprintf(fp,"₩n");
                printf("₩n[%s]  비행편  예약정보(%03d  /  %s)  :  ₩n",
ploop->flight_Number, ploop->res_Num, ploop->capacity);
                fprintf(fp,"₩n[%s]  비행편  예약정보(%03d  /  %s)  :  ₩n",
ploop->flight_Number, ploop->res_Num, ploop->capacity);
                printf("From:  %s  ▶▷▶  To  :  %s₩n",  ploop->from_city,
ploop->to_city);
                fprintf(fp,"From:  %s  ▶▷▶  To  :  %s₩n",  ploop->from_city,
ploop->to_city);
                printf("%-20s %-9s₩n", "이름", "비행기편");
                fprintf(fp,"%-20s %-9s₩n", "이름", "비행기편");

                if(rloop){

                        do{
                                rloop = rloop->prev_res;
                                printf("%-20s  %-9s  ₩n",  rloop->host->name,
ploop->flight_Number);
                                fprintf(fp,"%-20s  %-9s  ₩n",  rloop->host->name,
ploop->flight_Number);
                                //getch();
                        }while(rloop!=ploop->res_List);
                }
                rloop = ploop->front;
                printf("₩n[%s] 비행편 대기 리스트 : ₩n", ploop->flight_Number);
```

```
                fprintf(fp,"₩n[%s] 비행편 대기 리스트 : ₩n", ploop->flight_Number);
                printf("From:  %s  ▶▷▶  To  :  %s₩n",  ploop->from_city,
ploop->to_city);
                fprintf(fp,"From:  %s  ▶▷▶  To  :  %s₩n",  ploop->from_city,
ploop->to_city);
                printf("%-20s %-9s₩n", "이름", "비행기편");
                fprintf(fp,"%-20s %-9s₩n", "이름", "비행기편");

                while(rloop){
                        printf("%-20s                %-9s                ₩n",
rloop->host->name,ploop->flight_Number);
                        fprintf(fp,"%-20s                %-9s                ₩n",
rloop->host->name,ploop->flight_Number);
                        rloop = rloop->next_res;
                }

                ploop = ploop->next;
                printf("₩n▒▒▒▒▒▒▒▒₩n₩n");
                fprintf(fp,"₩n▒▒▒▒▒▒▒▒₩n₩n");
        }
        fclose(fp);
}

void print_FlightC(Plane * P, char * flight_Number)
{
        Plane * ploop = P;
        Reserv * rloop;
        FILE * fp=NULL;

        if(!(fp=fopen("OUTFILE.P1", "a+ "))){ printf("! Fail to open file.₩n"); }

        if(!ploop){ printf("! 등록된 항공편이 없습니다.₩n");fprintf(fp,"등록된 항공편이 없
습니다.₩n"); return ; }

        while(strcmp(ploop->flight_Number, flight_Number)){ ploop = ploop->next; }
        if(!ploop){ printf("! 항공편이 존재하지 않습니다.₩n"); fprintf(fp,"항공편이 존재하
지 않습니다.₩n"); return ; }
        ploop = P;
        printf("■□■□■□■□§비행편 별 예약, 대기 리스트§■□■□■□■□₩n");
```

```
	fprintf(fp,"■□■□■□■□§비행편 별 예약, 대기 리스트§■□■□■□■□₩n");
	rloop = ploop->res_List;
	printf("₩n");
	fprintf(fp,"₩n");
	printf("₩n[%s] 비행편 예약정보(%03d / %s) : ₩n", ploop->flight_Number,
ploop->res_Num, ploop->capacity);
	fprintf(fp,"₩n[%s] 비행편 예약정보(%03d / %s) : ₩n", ploop->flight_Number,
ploop->res_Num, ploop->capacity);
	printf("From: %s ▶▷▶ To : %s₩n", ploop->from_city, ploop->to_city);
	fprintf(fp,"From: %s ▶▷▶ To : %s₩n", ploop->from_city, ploop->to_city);
	printf("%-20s %-9s₩n", "이름", "비행기편");
	fprintf(fp,"%-20s %-9s₩n", "이름", "비행기편");

	if(rloop){
		do{
			rloop = rloop->prev_res;
			printf("%-20s	%-9s	₩n",	rloop->host->name,
ploop->flight_Number);
			fprintf(fp,"%-20s	%-9s	₩n",	rloop->host->name,
ploop->flight_Number);
			//getch();
		}while(rloop!=ploop->res_List);
	}
	rloop = ploop->front;
	printf("₩n[%s] 비행편 대기 리스트 : ₩n", ploop->flight_Number);
	fprintf(fp,"₩n[%s] 비행편 대기 리스트 : ₩n", ploop->flight_Number);
	printf("From: %s ▶▷▶ To : %s₩n", ploop->from_city, ploop->to_city);
	fprintf(fp,"From: %s ▶▷▶ To : %s₩n", ploop->from_city, ploop->to_city);
	printf("%-20s %-9s₩n", "이름", "비행기편");
	fprintf(fp,"%-20s %-9s₩n", "이름", "비행기편");

	while(rloop){
		printf("%-20s %-9s ₩n", rloop->host->name,ploop->flight_Number);
		fprintf(fp,"%-20s	%-9s	₩n",
rloop->host->name,ploop->flight_Number);
		rloop = rloop->next_res;
		//getch();
	}
	printf("₩n▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒
₩n₩n");
	fprintf(fp,"₩n▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒
```

```
▒₩n₩n");
        fclose(fp);
}

void Cancel_res(Plane * P, char * name, char * flight_Number)
{
        Plane * ploop = P;
        Customer * cloop;
        Reserv * rloop, *delete_node, * Q;

        if(!ploop){ printf("! 등록된 항공편이 없습니다.₩n"); return ;}

        while(strcmp(ploop->flight_Number, flight_Number)){ ploop = ploop->next;
if(!ploop) break; }
        if(!ploop && strcmp(flight_Number, "0")){ printf("! [%s] 항공편이 존재하지 않
습니다.₩n", flight_Number); return ;}

        cloop = C_list;

        while(strcmp(cloop->name, name)){ cloop = cloop->next; if(!cloop) break; }
        if(!cloop){ printf("! 등록된 고객이 아닙니다.₩n"); return ;}

        if(!cloop->res){ printf("! 고객님의 예약정보가 없습니다.₩n"); return ; }

        if(!strcmp(flight_Number, "0") || !strcmp(flight_Number, "0000")){
                rloop = cloop->res;

                while(rloop){
                        printf("₩n*%p**/", rloop);
                        delete_node = rloop;

                        if(!(rloop->isRes) && !(rloop->ticket->front->next_res) )
{rloop->ticket->front = NULL;}
                        if(rloop->thisC_next) { rloop->thisC_next->thisC_prev =
rloop->thisC_prev; }
                        if(rloop->thisC_prev) { rloop->thisC_prev->thisC_next =
rloop->thisC_next; }
                        if(!rloop->thisC_next && !rloop->thisC_prev) cloop->res =
rloop->thisC_next = rloop->thisC_prev = NULL;

                        if(rloop->ticket->res_List->prev_res                    ==
```

```
rloop->ticket->res_List->next_res){     rloop->ticket->res_List = NULL;        }
                        rloop->next_res->prev_res = rloop->prev_res;
                        rloop->prev_res->next_res = rloop->next_res;
                        // 처음꺼 걸렸을때 처리
                        if(rloop == rloop->ticket->res_List){
rloop->ticket->res_List = rloop->next_res;      }
                        if(rloop  ==  rloop->ticket->front){  rloop->ticket->front  =
rloop->ticket->front->next_res; }

                        rloop = rloop->thisC_next;

                        printf("1%p//₩n",delete_node->thisC_prev,
delete_node->thisC_next);
                        if(delete_node->isRes){
                                if(delete_node->ticket->front){
                                        Q = delete_node->ticket->front;
                                        delete_node->ticket->front                    =
delete_node->ticket->front->next_res;

add_Reserv(&(delete_node->ticket->res_List), Q, Q->host->name);
                                }else (delete_node->ticket->res_Num)--;
                        }

                        printf("      ※ 다음의 예약(대기)정보가 취소되었습니다 : [%s]
%s₩n",delete_node->ticket->flight_Number,delete_node->host->name );
                        free(delete_node);
                }

                return ;
        }

        rloop = cloop->res;
        while(strcmp(rloop->ticket->flight_Number,     flight_Number)){     rloop     =
rloop->thisC_next; if(!rloop) break; }
        if(!rloop){  printf("!  %s  고객님의  %s  항공편에  대한  예약(대기)정보가  없습니
다.₩n", name, flight_Number); return ;}

        delete_node = rloop;

        if(!(rloop->isRes) && !(ploop->front->next_res) ) {ploop->front = NULL; }
        if((rloop->isRes)   &&   (ploop->res_List   ==   ploop->res_List->next_res))
```

```
{ploop->res_List = NULL;}
        if(rloop->thisC_next) { rloop->thisC_next->thisC_prev = rloop->thisC_prev;
}
        if(rloop->thisC_prev) { rloop->thisC_prev->thisC_next = rloop->thisC_next;
}
        if(!rloop->thisC_next && !rloop->thisC_prev) cloop->res =
rloop->thisC_next = rloop->thisC_prev = NULL;

        rloop->next_res->prev_res = rloop->prev_res;
        rloop->prev_res->next_res = rloop->next_res;
        if(rloop == rloop->ticket->res_List){ rloop->ticket->res_List =
rloop->next_res; }
        if(rloop == rloop->ticket->front){ rloop->ticket->front =
rloop->ticket->front->next_res; }

        if(rloop->isRes){
                if(ploop->front){
                        Q = ploop->front;
                        ploop->front = ploop->front->next_res;
                        add_Reserv(&(ploop->res_List), Q, Q->host->name);
                }else (ploop->res_Num)--;
        }

        printf("          ※ 다음의 예약(대기)정보가 취소되었습니다 : [%s]
%s₩n",delete_node->ticket->flight_Number,delete_node->host->name );
        free(delete_node);
/*
        rloop = cloop->res;printf("%p %p₩n", rloop, rloop->ticket);
        while(rloop){printf("%p %p₩n", rloop->thisC_prev, rloop->thisC_next); rloop
= rloop->thisC_next;}
        */
}


void print_Customer(char * name)
{
        Customer * cloop = C_list;
        Reserv * rloop;
        FILE * fp=NULL;

        if(!(fp=fopen("OUTFILE.P1", "a+ "))){ printf("! Fail to open file.₩n"); }
```

```
	if(!cloop){
		printf("! 등록된 고객이 없습니다.₩n");
		fprintf(fp,"! 등록된 고객이 없습니다.₩n");
		return ;
	}

	while(cloop){
		if(!strcmp(cloop->name, name)) break;
		cloop = cloop->next;
	}
	if(!cloop){printf("! 등록되지 않았습니다.₩n"); fprintf(fp,"! 등록되지 않았습니다.₩n"); return ;}

	rloop = cloop->res;
	printf("■ %s 고객님의 예약 정보 : ₩n", cloop->name);
	fprintf(fp,"■ %s 고객님의 예약 정보 : ₩n", cloop->name);
	printf("	예약(대기)된 비행 정보₩n");
	fprintf(fp,"	예약(대기)된 비행 정보₩n");
	while(rloop){
		printf("%s",rloop->ticket->flight_Number);
		fprintf(fp,"%s",rloop->ticket->flight_Number);
		if(rloop->isRes) {printf("	● 예약₩n"); fprintf(fp,"	● 예약₩n");}
else {printf("	○ 대기₩n");fprintf(fp,"	○ 대기₩n");}

		printf("From:	%s	▷▶▷	To:	%s₩n₩n",rloop->ticket->from_city, rloop->ticket->to_city);
		fprintf(fp,"From:	%s	▷▶▷	To:	%s₩n₩n",rloop->ticket->from_city, rloop->ticket->to_city);
		rloop = rloop->thisC_next;
	}
	fclose(fp);
}

//헤더파일

/*********************************
	프로젝트 과제#1
	2009112441 김영빈
	항공기 예약관리
	Reserve_Plane.h 헤더파일
```

```
**********************************/

// 논리형 정의
#define Bool unsigned int
#define TRUE  1
#define FALSE 0

typedef struct Plane * pPlane;          // 비행기편 포인터
typedef struct Reserv * pReserv;                // 예약 정보 노드
typedef struct Customer * pCustomer;  // 고객 정보 노드


// 기본 노드 설정
typedef struct Plane {

        // 비행기편 정보
        char flight_Number[5];  // 비행기 번호
        char from_city[31];             // 출발지
        char to_city[31];               // 도착지
        char capacity[4];               // 최대 탑승객 수

        pPlane prev, next;              //

        Bool isFull;                    // 예약이 끝났는지 표시

        pReserv res_List;               // 예약 리스트 (링크드)
        pReserv front, rear;     // 대기 리스트 (큐)

        int res_Num;                    // 승인된 예약자 수
}Plane;

typedef struct Reserv {
        // 예약 정보
        pCustomer host;                                         // 예약자
        pPlane ticket;                                      // 비행기편
        pReserv thisC_prev, thisC_next;// 이 고객의 이전, 다음 노드
        pReserv prev_res, next_res;             // 이 예약의 이전, 다음 노드
        Bool isRes;                                             // 예약인지, 대기인지
여부
} Reserv;
```

```
typedef struct Customer {
        // 고객 정보
        char name[21];                  // 이름
        pCustomer prev, next;  // 고객 link 포인터
        pReserv res;                    // 예약 정보
} Customer;

Customer * C_list; //  총 고객 리스트

//////////////// 기본 오퍼레이션
void insert_Plane(Plane ** P, Plane * _new);              // 비행편 입력
void delete_Plane(Plane ** P, char * flight_Number); // 요청된 비행편 삭제
int compare_C(char * pre, int obj);                                        // 한쪽이 int
형인 비교
void reserv_Plane(Plane * P, char * name, char * flight_Number);
void print_All(Plane * P);
void add_Reserv(Reserv ** R_head, Reserv * _newR, char * name);
void add_Wait(Plane * P, Reserv * _newR);
Customer * add_Customer(char * name);
void Cancel_res(Plane * P, char * name, char * flight_Number);
void print_Flight(Plane * P);
void print_FlightC(Plane * P, char * flight_Number);
void conf_this(_newC, _newR);
void print_Customer(char * name);
void save_all(Plane * P, FILE * fp);
```

P1 파일
sample
IF 0056 Lafayette Detroit 004
IF 1234 Chicago Baltimore 006
IF 0196 Dayton Indianapolis 003
PA
RP Smith 0056
RP Hubbard 0196
RP Jones 0056
RP Churchill 0056
RP Sauvage 0056
RP Smith 0196
RP Smith 0056
RP Smith 1234
RP Jones 0196
RP Hubbard 0056
PF
PR 0056
PP Smith
CP Churchill 0056
RP Churchill 0196
RP Wiley 0196
PA
CP Smith 0
PR 0196
DF 0196
PA

outfile
□■□■□■□■□■□§ 비행편 리스트 §■□■□■□■□■□■

☞ 항공편: [0056] From: Lafayette To: Detroit
예약인원 / 탑승인원 : 000 / 004 √예약가능
☞ 항공편: [0196] From: Dayton To: Indianapolis
예약인원 / 탑승인원 : 000 / 003 √예약가능
☞ 항공편: [1234] From: Chicago To: Baltimore
예약인원 / 탑승인원 : 000 / 006 √예약가능

■□■□■□■□§비행편 별 예약, 대기 리스트§■□■□■□■□

[0056] 비행편 예약정보(000 / 004) :
From: Lafayette ▶▷▶ To : Detroit
이름 비행기편

[0056] 비행편 대기 리스트 :
From: Lafayette ▶▷▶ To : Detroit
이름 비행기편

[0196] 비행편 예약정보(000 / 003) :
From: Dayton ▶▷▶ To : Indianapolis
이름 비행기편

[0196] 비행편 대기 리스트 :
From: Dayton ▶▷▶ To : Indianapolis
이름 비행기편

[1234] 비행편 예약정보(000 / 006) :
From: Chicago ▶▷▶ To : Baltimore
이름 비행기편

[1234] 비행편 대기 리스트 :
From: Chicago ▶▷▶ To : Baltimore
이름 비행기편

■□■□■□■□§비행편 별 예약, 대기 리스트§■□■□■□■□

[0056] 비행편 예약정보(004 / 004) :
From: Lafayette ▶▷▶ To : Detroit
이름 비행기편
Churchill 0056

Jones 0056
Sauvage 0056
Smith 0056

[0056] 비행편 대기 리스트 :
From: Lafayette ▶▷▶ To : Detroit
이름 비행기편
Hubbard 0056

■ Smith 고객님의 예약 정보 :
예약(대기)된 비행 정보
0056 ● 예약
From: Lafayette ▷▶▷ To: Detroit

1234 ● 예약
From: Chicago ▷▶▷ To: Baltimore

0196 ● 예약
From: Dayton ▷▶▷ To: Indianapolis

□■□■□■□■□■□§ 비행편 리스트 §■□■□■□■□■□■

☞ 항공편: [0056] From: Lafayette To: Detroit
예약인원 / 탑승인원 : 004 / 004 ×예약불가
☞ 항공편: [0196] From: Dayton To: Indianapolis
예약인원 / 탑승인원 : 003 / 003 ×예약불가
☞ 항공편: [1234] From: Chicago To: Baltimore
예약인원 / 탑승인원 : 001 / 006 √예약가능

■□■□■□■□§비행편 별 예약, 대기 리스트§■□■□■□■□

[0056] 비행편 예약정보(004 / 004) :
From: Lafayette ▶▷▶ To : Detroit
이름 비행기편
Hubbard 0056
Jones 0056
Sauvage 0056
Smith 0056

[0056] 비행편 대기 리스트 :
From: Lafayette ▶▷▶ To : Detroit
이름 비행기편

[0196] 비행편 예약정보(003 / 003) :
From: Dayton ▶▷▶ To : Indianapolis
이름 비행기편
Jones 0196
Smith 0196
Hubbard 0196

[0196] 비행편 대기 리스트 :
From: Dayton ▶▷▶ To : Indianapolis
이름 비행기편
Churchill 0196
Wiley 0196

[1234] 비행편 예약정보(001 / 006) :
From: Chicago ▶▷▶ To : Baltimore
이름 비행기편
Smith 1234

[1234] 비행편 대기 리스트 :
From: Chicago ▶▷▶ To : Baltimore
이름 비행기편

■□■□■□■□§비행편 별 예약, 대기 리스트§■□■□■□■□

[0056] 비행편 예약정보(003 / 004) :

From: Lafayette ▶▷▶ To : Detroit
이름 비행기편
Hubbard 0056
Jones 0056
Sauvage 0056

[0056] 비행편 대기 리스트 :
From: Lafayette ▶▷▶ To : Detroit
이름 비행기편

□■□■□■□■□■□§ 비행편 리스트 §■□■□■□■□■□■

☞ 항공편: [0056] From: Lafayette To: Detroit
예약인원 / 탑승인원 : 003 / 004 ×예약불가
☞ 항공편: [1234] From: Chicago To: Baltimore
예약인원 / 탑승인원 : 000 / 006 √예약가능

■□■□■□■□§비행편 별 예약, 대기 리스트§■□■□■□■□

[0056] 비행편 예약정보(003 / 004) :
From: Lafayette ▶▷▶ To : Detroit
이름 비행기편
Hubbard 0056
Jones 0056
Sauvage 0056

[0056] 비행편 대기 리스트 :
From: Lafayette ▶▷▶ To : Detroit
이름 비행기편

[1234] 비행편 예약정보(000 / 006) :
From: Chicago ▶▷▶ To : Baltimore
이름 비행기편

[1234] 비행편 대기 리스트 :
From: Chicago ▶▷▶ To : Baltimore
이름　　　　　　　　　　비행기편

□■□■□■□■□■□§ 비행편 리스트 §■□■□■□■□■□■

☞　항공편: [0056]　　　From: Lafayette To: Detroit
　　예약인원 / 탑승인원 : 004 / 004 √예약가능
☞　항공편: [0196]　　　From: Dayton To: Indianapolis
　　예약인원 / 탑승인원 : 003 / 003 √예약가능
☞　항공편: [1234]　　　From: Chicago To: Baltimore
　　예약인원 / 탑승인원 : 001 / 006 ×예약불가
등록된 항공편이 없습니다.