PROCEDURE Reschedule;

VAR

JobDay,Today,IntialRow,IntialCol,Time,AircraftPostion,CurrentJobId:LONGINT;

Free,Present,Complete:BOOLEAN;

Finished:CHAR;

This procedure is responsible for determining whether or not a job requires rescheduling and locating the earliest possible time slot to reschedule the job.

BEGIN

RESET(JobFile); {Sets the filoe poiunter to the start of the file}

REPEAT

READ(JobFile,Job);

UNTIL Job.jobid=JetJob.JobId; {Locates job specific job within the file}

BEGIN

WITH Jet DO

BEGIN

WRITELN('This job has not yet been complete');

WRITELN; {Blank line}

WRITELN('Job title: ',Job.Jobtitle);

WRITELN('JobId: ',Job.Jobid);

WRITELN('Job description: ', job.Jobdescribtion);

WRITELN('Aircraft Id: ',AircraftId);

WRITELN('Call sign: ',CallSign);

WRITELN('Status: ',AircraftStatus);

WRITELN('Flight Hours: ',FlightHours);

WRITELN('Time since last service: ',Flight\_TimeFromservice);

IF JetJob.Complete=FALSE THEN

WRITELN('Progress: Incomplete');

WRITELN;

END;{Blank line}

REPEAT

WRITELN('This job is marked as incomplete do you wish to re-schedule or has the job been completed');

WRITELN; {Blank line}

WRITELN('Y= Mark as complete,N= Re-Schedule');

Finished:=UPCASE(READKEY);

IF (Finished <>'Y') AND (Finished <>'N') THEN

WRITELN('Invalid input please try again');

UNTIL(Finished='Y') OR (Finished='N'); {Look up check}

IF Finished='Y' THEN

BEGIN

SEEK(AircraftJobFile,AircraftPostion);

JetJob.Complete:=TRUE; {Marks the job as complete}

WRITE(AircraftJobFile,JetJob);

SEEK(AIRCRAFTJOBFILE,AIRCRAFTPOSTION);

{Sets file pointer to the start of the record}

READ(AircraftJobFile,jetjob);

END;

Initially the user will select whether or not to mark the job as complete or to reschedule. If the job requires rescheduling the procedure will locate the earliest free time slot within the users schedule and assign the job to this date.

IF Finished='N' THEN

BEGIN{Reschedules the given job}

REPEAT

This procedure unties several blank lines to improve user friendliness as the use of blank lines helps to natural divided information making the information on display easier to distinguish by the user thus making the information more readable and understandable.

Time:=1;{Time slot within schedule}

BEGIN

REPEAT

BEGIN

IF Schedule[Time,ToDay]='-1' THEN {Rouge value}

Free:=TRUE

ELSE

Time:=Time+1;

END;

UNTIL(Free=TRUE) OR (Time=8);

IF Free=TRUE THEN

BEGIN

SEEK(StaffFile,StaffRecNo);

[Sets file pointer to the start of teh record}

READ(StaffFile,STaff);

Schedule[Time,ToDay]:=INTTOSTR(CurrentJobId);

Schedule[IntialRow,JobDay]:='-1';

WRITE(StaffFile,Staff);{Writes teh job to the new date}

END;

END;

IF Free=FALSE THEN

ToDay:=ToDay+1; {Increases the day by one form the day of the job}

UNTIL Free=TRUE; {Repeats until a free slot is found within the schedule}

END;

END;

END;

PROCEDURE Delay(Year,Month,Day,Number:LONGINT){Paramters passed on by another subroutine};

VAR

Subtract,JobDay,Today,IntialRow,IntialCol,TodayYear,TodayMonth,TodayDay,TodayNo,Time,AircraftPostion,CurrentJobId,Add:LONGINT;

Free,Present,Complete:BOOLEAN;

Finished:CHAR;

BEGIN{calculates number of days from the previous Monday}

CASE Number OF {Multiple if statement tests if the input is equal to any of the a values below}

0:Subtract:=0;

1:Subtract:=0;

2:Subtract:=1;

3:Subtract:=2;

4:Subtract:=3;

5:Subtract:=4;

6:Subtract:=5;

END;

CASE Number OF {Calculate number of days from the weekend to next Monday}

0:Add:=1;

6:Add:=2;

ELSE

Add:=0;

END;

BEGIN {Data is initialised to prevent undefined values and undesired results}

Complete:=FALSE;

This procedure is used to identify any previous jobs that are yet to have been completed from the previous week to ensure that no jobs go uncompleted.

Free:=FALSE;

Found:=FALSE;

Present:=FALSE;

ToDay:=(Month-1)\*31+Day+Add;{Current date}

JobDay:=(Month-1)\*31 +Day- Subtract;{Date of previous Monday}

WITH Staff DO

BEGIN

REPEAT

ROW:=0;

BEGIN

REPEAT

Row:=Row+1;

IF (Schedule[Row,JobDay] <>'-1'){Rouge value} AND (JobDay<ToDay) THEN

{Test for job and that the job date has been passe3d}

BEGIN

Present:=TRUE;

END;

Initially this procedure calculates the date of the previous Monday, Following this each section of the users schedule is tested for a job. The date of the job is then stored and the user can reschedule the job that has been located. This is repeated up to the current date.

UNTIL (Present) OR (Row=8);

END;

IF (NOT(Present)) AND (Jobday<>Today) THEN

BEGIN

JobDay:=JobDay+1;{The date from the previous Monday is increased by d one}

END;

UNTIL (Present) OR (Jobday=Today);

IF Present=TRUE THEN

BEGIN

IntialRow:=Row; {Stores the valie of the current timeslot}

CurrentJobID:=STRTOINT(Schedule[Row,JobDay]);{Coverts the id number to r text format}

RESET(AircraftJobFile);{Sets the file pointer to the strat of the file}

REPEAT

READ(AircraftJobFile,JetJob);

IF (Jetjob.AircraftJObID=CurrentJobId) AND (Jetjob.Complete=FALSE) THEN

Found:=TRUE;

UNTIL (Found) OR (EOF(AircraftJobFile));

IF Found THEN

BEGIN

AircraftPostion:=FILEPOS(AircraftJobFile)-1;{Stores position of record}

BEGIN

RESET(AircraftFile); {Sets the file pointer to the start of the d record}

REPEAT

This procedure is used to identify any previous jobs that are yet to have been completed from the previous week to ensure that no jobs go uncompleted.

READ(AircraftFile,Jet);

UNTIL Jet.AircraftId=JetJob.AircraftId;

END;

END;

END;

END;

END;

END;

FUNCTION WeekDay: DayNameArray ;

VAR

DayName:DayNameArray;

Pos:LONGINT;

BEGIN

WeekDay[0]:='Sunday';

WeekDay[1]:='Monday';

WeekDay[2]:='Tuesday';

WeekDay[3]:='Wendsday';

WeekDay[4]:='Thursday';

WeekDay[5]:='Friday';

WeekDay[6]:='Saturday';

END;

PROCEDURE DisplayJobInfo(Times,Count,TaskID:LONGINT;Plane:STRING);{Paramters passed on by another subroutine}

VAR

MatchId:STRING[5];

This procedure displays all the information regarding a specific job within the user’s roster.

Found:BOOLEAN;

BEGIN

TaskId:=STRTOINT(Staff.Schedule[Row,Times]);

Count:=Count+1;

RESET(AircraftJobFile);

REPEAT

READ(AircraftJobFile,JetJob);

UNTIL Jetjob.AircraftJobId=TaskID;

MatchId:=JetJob.JobId;

Plane:=Jetjob.AircraftId;

RESET(JobFile);

REPEAT

READ(JobFile,Job);

UNTIL Job.jobId=MatchID;

BEGIN

WITH Job DO

BEGIN

WRITELN;

WRITELN('JobId: ',JobId);

WRITELN('Job title: ',JobTitle);

WRITELN('Job description: ',JobDescribtion);

WRITELN('How the fault was deected: ',WhenHowFound);

WRITELN;

RESET(AircraftJobFile);

Found:=FALSE;

REPEAT

READ(AircraftJobFile,JetJob);

IF (Jetjob.complete=FALSE) AND (JobId=Jetjob.jobId) THEN

Found:=TRUE;

UNTIL(Found) OR(EOF(AircraftJobFile));

RESET(AircraftFile);

REPEAT

READ(AircraftFile,Jet);

UNTIL(Jet.Aircraftid=Plane) OR (EOF(AircraftFile));

WITH Jet DO

BEGIN

WRITELN('Aircraft');

WRITELN;

WRITELN('Aircraft Id: ',Aircraftid);

WRITELN('Call sign: ',CallSign);

WRITELN('Total Flight hours: ',FlightHours);

WRITELN('Flight hours since last service: ',Flight\_timefromservice);

END;

WRITELN('Press any key to continue');

READKEY;

WRITELN;

END;

Initially the record containing the information regarding the specific task is located. From here the id of the job that is being preformed is stored and used to access the record containing the information regarding the job which is then displayed to the user. The same process is repeated for the aircraft that the job is being performed on and all information regarding the task is displayed to the user.

END;

END;

END;

PROCEDURE ShowDate;

VAR

TodayYear, TodayMonth,TodayDay,TodayNo,Times, StaffJobRecNo, Day,Add,Count,TaskID: LONGINT;

MatchId,JobMatch,Plane:STRING[5];

Found:BOOLEAN;

DayName:DayNameArray;

BEGIN

DayName:=WeekDay;

CLRSCR;{Clears the contents of the screen}

GETDATE(TodayYear,TodayMonth,TodayDay,TodayNo);{Retrives the current date}

SEEK(StaffFile,StaffRecNo);{Sets the file pointer to the start of the stored record}

READ(StaffFile,Staff);

Delay(TodayYEar,TodayMOnth,TodayDay,TodayNo);{Test for any previous jobs not yet complete passing on s the following paramters form this procedure}

BEGIN

GOTOXY(35,2);{Sets the position of the text to the coordinates on the screen}

BEGIN

CLRSCR;{Clears the contents of the screen}

GOTOXY(35,2); {Sets the position of the text to the coordinates on the screen}

WRITELN('Weekley roster');

WRITELN;{Blank line}

WRITELN('Today''Date is: ',TodayDay,'/',TodayMonth,'/',TodayYear);{Dispalys the current date in d the form DD/MM/YY}

WRITELN('Press any key to continue');

READKEY;{waits until the user enters a value to progress to the next instruction}

WRITELN;{Blank line}

CASE TodayNo OF{Calculates the number of days to add on to reach

the nearest Monday form the weekend}

0:Add:=1;

6:Add:=2;

ELSE

Add:=0;

END;

END;

Col:=(TodayMonth-1) \*31 +TodayDay+Add; {Stores the value of the day out of 365}

IF (TodayNo=0) OR (TodayNo=6) THEN {Determines if weekend or weekday}

TodayNo:=1; {Sets the day of the week to Monday}

CLRSCR;

FOR Times:=Col TO COL+(5-TodayNo) DO{Dislays the roster until Friday has been passed]

BEGIN

CASE TodayNo OF {Determines the day of the week}

1:WRITELN('Monday');

2:WRITELN('Tuesday');

3:WRITELN('Wendsday');

4:WRITELN('Thursday');

5:WRITELN('Friday');

END;

Count:=0;{Number of jobs on a given day}

TodayNo:=TodayNo+1;

{Increase the day of the week by one}

FOR Row:= 1 TO 8 DO

BEGIN

CASE Row OF {Determines the time of the job}

1:WRITELN('8:00');

2:WRITELN('10:00');

3:WRITELN('12:00');

4:WRITELN('14:00');

5:WRITELN('16:00');

6:WRITELN('18:00');

7:WRITELN('20:00');

8:WRITELN('22:00');

END;

Located:=FALSE;

IF Staff.Schedule[Row,TIMES] <>'-1' THEN

Located:=TRUE;

IF Located=TRUE THEN

DisplayJobInfo(Times,Count,TaskID,Plane);

{Displays the information regarding the job}

IF Located=FALSE THEN

BEGIN

WRITELN('You have no jobs at this time');

END;

END;

IF Count=0 THEN

WRITELN;{Blank line}

WRITELN('You have no jobs on this day');

WRITELN;{Blank line}

WRITELN('Press anykey to continue');

READKEY;

WRITELN; {Blank line}

END;

END;

END;

This procedure displays the contents of the user’s weekly roster, calculating the date and time slot that a job is taken place and displaying the relevant information to the user form the current day to the nearest Friday. This procedures makes use of serval blank line and clear screens to aid in user friendliness and present information in a clear logical manner. In addition to this the procedure also has several points where the user will be asked to enter a key to continue, this allows the user to progress at their own pase allowing them to read as much information as required before continuing and presenting more information on screen. This also helps the naturally divide and categorise the information into the different days making the information more understandable.

Initially the current date is retried from the system and the day of the year (out of 365 days) is calculated. A numerical value will also be obtained that corresponds to a given day of the week. If the current date is a weekend the system will determine the number of days to add on to the current date to reach the next Monday. Starting form here or the current date each time slot in the user’s record is tested. The time corresponding to each time slot (row within the user’s schedule) is displayed along with the day of the week. If a rogue value is detected than a message is displayed informing the user that no job has yet been assigned to this time slot. Otherwise the details of the specific job are display.

PROCEDURE DisplayRecord;

This procedure displays the personal details of the current user logged into the system. In order to do this each field within the current record inside the staff file is read and displayed.

BEGIN{Start of procedure}

WITH Staff DO {Selects type of record to be displayed}

BEGIN{Start of display}

CLRSCR;{Clears the contents of the screen.

WRITELN('Here is your personal information');

WRITELN; {Blank line}

WRITELN('ID code: ',StaffIdcode);

WRITELN('Name: ',Forename,' ', Surname);

WRITELN('Job feild: ',Jobfeild);

WRITELN('Press anykey to continue');

READKEY;

END;{End of display}

END;{End of procedure}

To aid user friendliness several blank lines have been outputted by the system to help split up information making the information easier to identifier thus making the data more understandable. In addition to this upon instating the sub routine the previous contents of the screen is cleared ensuring that only the information required by the user is being displayed. This allows for more of the screen to be utilised by the sub routine and makes the information easier to read and unserstand.

PROCEDURE DisplayStaffFile;

BEGIN

SEEK(StaffFile,StaffRecNo);{Sets the file pointer to

The start of the stored record}

READ(StaffFile,Staff);

IF Staff.Rank='Flight Sergant'THEN {tests the rank of the current user}

BEGIN {Displays each record within the file}

RESET(StaffFile);{Sets the file pointe to the start of the fiel}

WHILE NOT EOF(StaffFile) DO{repeats until the end of the file}

BEGIN

READ(StaffFile,Staff);

WITH Staff DO

BEGIN

IF StaffIDcode <>'-1' THEN {Tests for deleted record by t detecting a rouge value }

BEGIN

WRITELN; {Blank line}

WRITELN('ID code: ',StaffIdcode);

WRITELN('Name: ',Forename,' ', Surname);

WRITELN('Job feild: ',Jobfeild);

WRITELN('password: ',Password);

WRITELN('Press anykey to continue');

READKEY;

END;

This procedure displays the personal details of each member of staff whose details are currently stored by the system to an admin member of staff.

END;

END;

END;

SEEK(StaffFile,StaffRecNo);{Sets the file pointer to the

start of the stored record}

READ(StaffFile,Staff);

IF Staff.Rank <> 'Flight Sergant' THEN

BEGIN

WITH Staff DO

BEGIN

WRITELN;

WRITELN('ID code: ',StaffIdcode);

WRITELN('Name: ',Forename,' ', Surname);

WRITELN('Job feild: ',Jobfeild);

WRITELN('Press anykey to continue');

READKEY;

END;

END;

This procedure makes use of incorporating blank lines to split up the information this helps to naturally categorise the information showing which data belongs to which record. As a result the data is more meaningful, easier to read and easier to understand. There are also several points where the user will be asked to enter a value to continue this allows the user to proceed at their own pace allowing them to read as much information as required before continuing and presenting more information on screen. This helps to ensure that the user can easily distinguish between each record on display and that the data is not too concentrated on screen for the user to understand.

END;

The procedure reads each record in turn within the staff file using serial access and displays each field within the record in the order the records are read. .

This procedure displays the details regarding each aircraft currently in use by the squadron.

PROCEDURE DisplayAircraftFile;

BEGIN

RESET(AircraftFile);

WHILE NOT EOF(AircraftFile) DO

BEGIN

READ(AircraftFile,Jet);

WITH JET DO

BEGIN

IF AircraftId <> '-1' THEN

BEGIN

WRITELN;

WRITELN('Aircraft ID: ',AircraftId);

WRITELN('Call sign: ',CallSign);

WRITELN('Aircraft Status: ',AircraftStatus);

WRITELN('Flight hours since last service: d d ',Flight\_timefromservice);

WRITELN('Total flight time (Hours): ',Flighthours);

WRITELN('Press anykey to continue');

READKEY;

END;

END;

END;

END;

This procedure makes use of incorporating blank lines to split up the information this helps to naturally categorise the information showing which data belongs to which record. As a result the data is more meaningful, easier to read and easier to understand. There are also several points where the user will be asked to enter a value to continue this allows the user to proceed at their own pace allowing them to read as much information as required before continuing and presenting more information on screen. This helps to ensure that the user can easily distinguish between each record on display and that the data is not too concentrate don screen for the user to understand.

The procedure reads each record in turn within the aircraft file using serial access and displays each field within the record in the order the records are read. .

PROCEDURE ViewFile;

This procedure provided a menu to the user allowing them to select which file they want to view.

VAR

FileChoice:CHAR;

RecordNo:LONGINT;

Authentrey:STRING[7];

BEGIN

REPEAT

CLRSCR;

WRITELN('Press A to view the staff file');

WRITELN('Press B to view the aircraft file');

WRITELN('Press E to exit');

Filechoice :=UPCASE(READKEY);

CASE Filechoice OF

'A': DisplayStafffile;

'B':DisplayAircraftFile;

'E':WRITELN;

ELSE

WRITELN('Invalid input please select from the list provided');

WRITELN('Press anykey to continue');

READKEY;

END;

UNTIL Filechoice='E';

END;

This procedure uses the input for the user to compare the input t a list of available options in order to determine what file the user wishes the view. The selected procedure will be called, the user will exit the current procedure or an error message will be displayed.

This once again reinforces the menu based user interface helping users to quickly familiarise themselves with how the system operates. As menus are used throughout the system the user will be familiar with this interface aiding user friendliness. As stated in previous sections the use of a menu also helps to categorise tasks into distinctive groups making it easier for the user to identify which file they wish to view and the necessary input required by the system. By limiting the number of inputs from the user the range of error available is also significantly decreased.

PROCEDURE DisplayMemberAuths;

This procedure displays a list of authorisations possessed by a chosen member of staff.

VAR

Found:BOOLEAN;

StaffAUthRecNo:LONGINT;

AuthIdentfier:STRING[5] ;

BEGIN

WRITELN('Authirisations');

BEGIN

RESET(StaffAuthsFile);

StaffAuthRecNo:=1;{Postion of record within staff auths file}

WHILE NOT EOF(StaffAUthsFile) DO {Repeats until the end of the file}

BEGIN

REPEAT

Found:=FALSE;

SEEK(StaffAuthsFile,StaffAuthRecNo);{Sets the file pointer to the h start of the record}

READ(StaffAuthsFile,StaffAuths);

IF StaffAuths.StaffID=Staff.StaffIdCode THEN

Found:=TRUE;

StaffAuthRecNo:=StaffAuthRecNo+1; ;{The postion of the next e record is stored

IF Found=TRUE THEN

BEGIN

AuthIdentfier:=StaffAuths.AuthorisationID;

RESET(AuthFile);

WHILE NOT EOF(AuthFile) DO

BEGIN

REPEAT

READ(AuthFile,Auth);

IF Authidentfier=Auth.AuthirisationId THEN

BEGIN

WRITELN;{Blank line}

WRITELN('Authirisation: ',Auth.Description);

WRITELN('Press anykey to continue');

READKEY;

END;

UNTIL (AuthIdentfier=Auth.AuthirisationId) OR (EOF(AuthFile));

END;

END;

UNTIL (Found) OR (EOF(StaffAUthsFile));

END;

END;

END;

Blank lines have been incorporated to improve the user interface of the system by splitting up the information making the information on display easier to distinguish. This makes the information easier to read and more understandable. There are also several points where the user will be asked to enter a value to continue this allows the user to proceed at their own pace allowing them to read as much information as required before continuing and presenting more information on screen. This helps to ensure that the data regarding each authorisation is not confused by the user and the information is displayed in a clear and logical manor.

Each record within the staff authorisation record is read in turn using serial access. If a record is found containing the id of the selected staff member then the authorisation id is stored. The authorisation file is then searched in the same manner as the staff authorisations file until the record containing the stored authorisation id is found. The contents of this records is then displayed to the user . This is repeated until the end of they

PROCEDURE EnterID;

VAR

Found:BOOLEAN;

UserOption,AuthChoice:CHAR;

BEGIN

REPEAT

WRITELN('Please enter the Id number of the staff member you require');

READLN(SearchId);

Found:=FALSE;

IF SearchID =Staff.StaffIdCode THEN

Found:=TRUE;

BEGIN

IF Found=FALSE THEN

BEGIN

REPEAT

CLRSCR;

WRITELN('The code you entered was invalid do you wish to try again');

WRITELN('Press Y for yes');

WRITELN('Press N for no');

UserOption:=UPCASE(READKEY);

IF (UserOption<>'Y') AND (UserOption<>'N') THEN

BEGIN

WRITELN;

WRITELN('Invalid input please try again');

WRITELN;

WRITELN('Press any key to continue');

READKEY;

CLRSCR;

END;

This procedure enables the user to uniquely identify a member of staff by entering the id code of the desired member as displayed on screen. Once the member of staff has been identified all information regarding the member of staff is displayed.

UNTIL(UserOption='Y') OR (UserOption='N');

END;

END;

CLRSCR;

WITH Staff DO

IF UserOption<>'N' THEN

BEGIN

WRITELN;

WRITELN('ID code: ',StaffIdcode);

WRITELN('Name: ',Forename,' ', Surname);

END;

UNTIL (UserOption='N') OR (Found);

IF Found=TRUE THEN

Upon entering the id code the file pointer is set to the start of the staff file and each record is read using serial access until the given id is found. The contents of the record are then displayed to the user. If the desired record is not found an error message is displayed.

BEGIN

RESET(StaffFile);

REPEAT

READ(StaffFile,Staff);

UNTIL Staff.StaffIdCode=SearchID;

WITH Staff DO

BEGIN

WRITELN;

WRITELN('ID code: ',StaffIdcode);

WRITELN('Name: ',Forename,' ', Surname);

WRITELN('Job feild: ',Jobfeild);

WRITELN('Press anykey to continue');

READKEY;

WRITELN;

REPEAT

WRITELN('Do you wish to view the authirsations for this member of staff(Y=Yes,N=No)');

AuthChoice:=UPCASE(READKEY);

IF (AuthChoice<>'Y') AND (AuthChoice<>'N') THEN

BEGIN

WRITELN('Invalid input please try again');

WRITELN;

Several techniques have been utilised within this procedure to produce a user friendly environment such as the use of blank lines, clear screens and several points where the user can proceed at their digression. This helps to modularise the information making the information more identifiable. This also makes the data on display more clearly to the user making it more understandable.

WRITELN('Press anykey to continue');

READKEY;

END;

UNTIL (AuthChoice='Y') OR (AuthChoice='N');

IF AuthChoice='Y' THEN

DisplayMemberAuths;

END;

END;

END;

This procedure enables the user to search for information regarding a specific member of staff if required by the user.

PROCEDURE SearchStaff;

VAR

ViewList,UserOption:CHAR;

DisplayMessage:STRING[2];

BEGIN

WITH Staff DO

REPEAT

CLRSCR;

RESET(StaffFile);

WRITELN('Please enter the surname of the desired staff member');

READLN(SearchName);

Located:=FALSE;

REPEAT

READ(StaffFile,Staff);

IF UPPERCASE(SearchName) = UPPERCASE(Surname) THEN

Located:=TRUE;

UNTIL (Located) OR (EOF(StaffFile));

BEGIN

WRITELN;

WRITELN('ID code: ',StaffIdcode);

WRITELN('Name: ',Forename,' ', Surname);

END;

IF Located=FALSE THEN

BEGIN

REPEAT

CLRSCR;

WRITELN('No matches found do you wish to reenter the desired name');

WRITELN('Y: YES');

WRITELN('N: NO');

Choice:=UPCASE(READKEY);

CASE Choice OF

'Y':WRITELN;

'N':WRITELN;

ELSE

BEGIN

WRITELN;

WRITELN('Invalid input please try again');

WRITELN;

WRITELN('Press any key to continue');

READKEY;

END;

END;

UNTIL (Choice='Y') OR(Choice='N');

END;

UNTIL (Located) OR (Choice='N');

IF Choice <> 'N' THEN

EnterId;

END;

The user will enter the surname of the desired member of staff. Key information will be displayed for each member of staff matching the criteria such as . This will provide the information needed to allow the user to refine their search. Upon en

PROCEDURE SearchJet;

VAR

ViewList:CHAR;

BEGIN

REPEAT

CLRSCR;

RESET(AircraftFile);

REPEAT

WRITELN('Do you wish to view the list of current aircraft (Y=Yes,N=No)');

ViewList:=UPCASE(READKEY);

IF (ViewList<>'Y') AND (ViewList<>'N') THEN

BEGIN

WRITELN('Error invalid input please try again');

WRITELN;

WRITELN('Press any key to continue');

READKEY;

WRITELN;

END;

UNTIL (ViewList='Y') OR (ViewList='N');

IF ViewLIst='Y' THEN

DisplayAircraftFile;

WRITELN;

WRITELN('Please enter the Id number of the desired aircraft');

READLN(SearchName);

Located:=FALSE;

RESET(AircraftFile);

BEGIN

REPEAT

READ(AircraftFile,Jet);

IF UPPERCASE(Jet.AircraftId) = UPPERCASE(SearchName) THEN

Located:=TRUE;

UNTIL (Located) OR (EOF(AircraftFile));

BEGIN

IF Located=FALSE THEN

BEGIN

REPEAT

WRITELN('Invalid input do you wish to search again? (Y=Yes, N=No)');

Options:=UPCASE(READKEY);

IF Options ='Y' THEN

WRITELN;

IF (Options <>'Y') AND (Options<>'N') THEN

WRITELN('Invalid input please try again');

UNTIL (Options='Y') OR (Options='N');

END;

END;

BEGIN

IF Located=TRUE THEN

WITH Jet DO

BEGIN

WRITELN;

WRITELN('AircraftId:',AircraftId);

WRITELN('CallSign: ',Callsign);

WRITELN('Aircraft status: ',AircraftStatus);

WRITELN('Flight hours since last service: ',Flight\_timefromservice);

WRITELN('Total Flight time: ',Flighthours);

WRITELN;

WRITELN('Press anykey to continue');

READKEY;

END;

END;

END;

UNTIL (Located) OR (Options='N');

END;

PROCEDURE Search(SearchItem:CHAR);

VAR

AuthRecNo:LONGINT;

StaffAuthRecNo:LONGINT;

AuthIdentfier:STRING[5];

AnotherSearch,AuthChoice,ViewList,UserOption:CHAR;

DisplayMessage:STRING[2];

BEGIN

DisplayMessage:='-1';

REPEAT

IF (SearchItem <>'A') AND (SearchItem<>'B') THEN

BEGIN

CLRSCR;

WRITELN('Please select one of the following options');

WRITELN;

WRITELN('Press A to search for a member of staff');

WRITELN('Press B to search for a aircraft');

WRITELN('Press E to exit');

SearchItem:=UPCASE(READKEY);

END;

CASE SearchItem OF

'A': SearchStaff;

'B':SearchJet;

'E':WRITELN;

ELSE

BEGIN

WRITELN('Invalid input please try again');

WRITELN;

WRITELN('Press anykey to continue');

READKEY;

WRITELN;

END;

END;

BEGIN

REPEAT

WRITELN('Do You wish to search for another Record (Y=Yes, N=No)');

AnotherSearch:=UPCASE(READKEY);

IF (AnotherSearch <>'Y') AND (AnotherSearch <>'N') THEN

WRITELN('Invalid input please try again');

UNTIL(AnotherSearch='Y') OR (AnotherSearch='N');

END;

UNTIL (Choice='E') OR (AnotherSearch='N');;

END;

FUNCTION RoleTest (JobCode:STRING){parameter passed on by another routine}:STRING;{Data type of value returned by the function}

BEGIN {Start of function}

IF JobCode='A0001' THEN {Job id}

This function returns the value of the job field that is required by members of staff to perform a given job. This function is used to ensure that members that are selected to perform a job are qualified and have received the appropriate training.

RoleTest:='Mechanics';{Job field}

IF JobCode='A0002' THEN

RoleTest:='Mechanics';

IF JobCode='A0003' THEN

RoleTest:='Mechanics';

IF JobCode='A0004' THEN

RoleTest:='Mechanics';

IF JobCode='A0005' THEN

RoleTest:='Avionics';

IF JobCode='A0006' THEN

RoleTest:='Mechanics';

IF JobCode='A0007' THEN

RoleTest:='Mechanics';

IF JobCode='A0008' THEN

RoleTest:='WeaponsTechnican';

END;{End of function}

The job is identified by its id code which is passed on as a parameter by another subroutine . This function will then return the required job field back to this subroutine.

FUNCTION RoleCheck(StaffId,Role:STRING){Paramters passed on by another sub routine}:BOOLEAN{Data type of value returned by function};

BEGIN

RoleCheck:=FALSE;

WITH Staff DO

This function returns a specific value determining whether the selected member of staff is of the necessary job field.

BEGIN

RESET(StaffFile);

REPEAT

READ(StaffFile,Staff);

UNTIL StaffIdCode=StaffId;

IF JobFeild=Role THEN

RoleCheck:=TRUE;

END;

END;

The staff member is identified by their id code which is passed on as a parameter by another subroutine along with the required role. The function then a returns a value to this subroutine indicating if the chosen member is of this role.

FUNCTION RankTest(MemberRank:STRING){Parameter passed on by another sub routine}:LONGINT{Data type of value returned by function};

BEGIN{Start of function}

IF MemberRank='SAC' THEN {Rank of staff member}

RankTest:=1; {Numerical value of the given rank}

IF MemberRank='Corporal' THEN

RankTest:=2;

IF MemberRank='Sergant' THEN

This function returns a value assigning a numerical value corresponding to a given rank allowing for the rank of each member to be better compared against certain conditions.

Ranktest:=3;

IF memberRank='Flight Sergant' THEN

RankTest:=4;

END; {End of function}

The rank of the selected member of staff is passed on as a parameter by another subroutine. The function will then return the numerical value corresponding to this rank back to the subroutine.

FUNCTION Jobtest(Jobnumber:STRING){Paramter passed on by another subroutine}:LONGINT{Data type of value passed on by the function};

BEGIN {Start of function}

IF JobNumber='A0001' THEN {Job Id}

This function returns the value of the minimum number of staff members required to perform a given job (as provided by my client), This function compares the id of the selected job against a list of available id’s and assigns the minimum number of staff members required accordingly.

JobTest:=2;{Minimum number of members required}

IF JobNumber='A0002' THEN

JobTest:=5;

IF JobNumber='A0003' THEN

JobTest:=3;

IF JobNumber='A0004' THEN

JobTest:=2;

IF JobNumber='A0005' THEN

JobTest:=1;

IF JobNumber='A0006' THEN

JobTest:=4;

IF JobNumber='A0007' THEN

JobTest:=3;

IF JobNumber='A0008' THEN

JobTest:=4;

END; {End of function}

The id of the selected job is passed on as a parameter by another subroutine where the job is selected. The function than returns the minimum number of staff members required to this subroutine based on the id that has been passed on.

FUNCTION MemberTest(JobStaff:StaffArray;JobValue:LONGINT;JobCode:STRING){Paramters passed on by another subroutine}:BOOLEAN;{data type of value retunred by the function}

VAR

Pos,Count,TotalCount,RankValue:LONGINT;

Valid,SergantTest,StaffRequirment:BOOLEAN;

BEGIN {Start of the function}

Membertest:=FALSE;

TotalCount:=0;{Total number of memebrs added}

Count:=0;{Number of overseeing officers}

Valid:=False;

SergantTest:=TRUE;

Pos:=0; {The data is initialised to prevent any undefined values}

FOR POS:=1 TO 5 DO {Postion of subscript within array}

BEGIN{Start of test}

This function returns a specific value determine whether the combination of staff members assigned is valid by determining if there is the correct number of overseeing officers of the required rank to oversee the completion of the task. In addition to this this function also determined if the minimum requirement of staff members has been met.

IF JobStaff[Pos]<> '-1'{Rouge value} THEN

BEGIN

TotalCount:=TotalCount+1;

RESET(StaffFile);

REPEAT

READ(StaffFile,Staff);

UNTIL Staff.StaffIdCode=JobStaff[Pos];

RankValue:=RankTest(Staff.Rank);

IF RankValue>=2 THEN

Count:=Count+1;

IF JobCode='A00002' THEN

BEGIN

IF RankValue<3 THEN

SergantTest:=FALSE

ELSE

SergantTest:=TRUE;

END;

StaffRequirment:= (Count>=1) AND (TotalCount>=JobValue);

MemberTest := (StaffRequirment=TRUE) AND(SergantTest=TRUE);

END;

END;{End of test}

END;{End of the function}

The data tested within this function is passed on to this function as parameters including the list of members assigned to the job, the minimum number of members required and the id o the selected job.

This function ensures that the requirements of the selected task have been met by calculating the total members that have been assigned to the job to ensure that the minimum number of staff members required has been met. In addition to this the function calculates the number of staff members that are of the correct rank to overseeing the completion of the job and ensures that the correct number of overseeing officers have been assigned to the job.

FUNCTION AuthTest(JobCode,MemberId:STRING){Paramters passed on by another subroutine}:BOOLEAN; {Dta type of value returned by the function}

VAR

MatchCount,AuthCount:LONGINT;

BEGIN {Start of function}

AuthTest:=FALSE;

Found:=FALSE;

AuthCount:=0;{Total number of authorisations required}

matchCount:=0; ;{number of required authorisation possessed by the member of staff}

{The data is intalised to preven any undefined values}

RESET(JobFile);

REPEAT

READ(JobFile,Job);

UNTIL Job.JobId=JobCode; {The selected job is located}

BEGIN{Start of test}

REPEAT

AuthCount:=AuthCount+1;

RESET(StaffAuthsFile);

REPEAT {Locates the authorisations possessed by the chosen staff member}

READ(StaffAuthsFile,StaffAuths);

IF (StaffAuths.StaffId=memberId) AND (StaffAuths.AuthorisationID=Job.AuthId[AuthCount]) THEN

Found:=TRUE;

IF Found =TRUE THEN

MatchCount:=MatchCount+1;

UNTIL (Found) OR EOF(StaffAuthsFile);

This function returns a specific value determine whether the selected member of staff possess the necessary authorisations/qualifications to perform the selected job.

UNTIL Job.AuthID[AuthCount+1]='-1';{Rouge value}

IF MatchCount=AuthCount THEN

AuthTest:=TRUE;

END;{end of test}

END;{End of function}

This function compares a list of authorisations required for the given job against the authorisations within the staff authorisation file to see how many authorisations are possessed by the chosen member of staff. The id of the selected job and staff member are passed on as parameter’s from another subroutine after which the function will return the value to the subroutine indicating if the member of staff is qualified or not.

Initially the record relating to the selected job is located , following this the staff authorisations file is read until a record is found containing the id of the staff member and the id of the first authorisation within the authorisation array stored within the job file or until the end of the file has been reached. A count is collected of each authorisation and each authorisation possessed by the member of staff. If the counts are equal when the end of the array within the record inside the job file has been reached then the member of staff is qualified and the function returns the value true.

FUNCTION DaysOfMonth(Month:LONGINT):LONGINT;

BEGIN

IF Month=1 THEN

DaysOfMonth:=31;

IF Month=2 THEN

DaysOfMonth:=28;

IF Month=3 THEN

DaysOfMonth:=31;

IF Month=4 THEN

DaysOfMonth:=30;

IF Month=5 THEN

DaysOfMonth:=31;

IF Month=6 THEN

DaysOfMonth:=30;

IF Month=7 THEN

DaysOfMonth:=31;

IF Month=8 THEN

DaysOfMonth:=31;

IF Month=9 THEN

DaysOfMonth:=30;

IF Month=10 THEN

DaysOfMonth:=31;

IF Month=11 THEN

DaysOfMonth:=30;

IF Month=12 THEN

DaysOfMonth:=31;

END;

FUNCTION TodayNumber(JobDate:LONGINT):LONGINT;

VAR

Add,Today,Count,Differance,TodayYear,TodayMonth,TodayDay,TodayNo:LONGINT;

Valid:BOOLEAN;

BEGIN

GETDATE(TodayYear ,TodayMonth ,TodayDay,TodayNo);{Retrieves todays date from the computer that the user is using}

Today:=(TodayMonth-1) \*31 +TodayDay;{The current day out of 365 days in the year}

Differance:=JobDate-Today; {The diffranc ein days between the current date and the day of the job}

Count:=0;{A count of eah day added on to the current date to reach the the of the job}

IF Differance >0 THEN

This function returns a specific numerical value corresponding to a given day within the week. This allows the system to determine whether the given date entered by the user for a job to take place is a weekend or weekday, as per the information provided by my client jobs cannot be performed on a weekend.

BEGIN

REPEAT

TodayNo:=TodayNo+1;

IF TodayNo>6 THEN

BEGIN

TodayNo:=0;

END;

Count:=Count+1;

UNTIL (Count=Differance) OR ( Differance=0);

END;

TodayNumber:=TodayNo;

END;

The day that the job is selected to take place (out of 365 days within the year) is passed on by another subroutine as a parameter, the function then returns the numerical value corresponding to the day of the week that this date is .

This function first retrieves the current date form the computer and calculates the day within the year out of 365. The function then calculates the difference between the current date and the date of the job. The function then increase the value corresponding to the day of the week by one , resetting the value for each new week ,eventually reaching the value of the selected date for the job.

FUNCTION SelectMonth(Date:LONGINT):LONGINT;

VAR

Select:CHAR;

BEGIN

IF (Date >= 29) AND (Date<>31) THEN

BEGIN

REPEAT

CLRSCR;

WRITELN('Press A for January');

WRITELN('Press B for March');

WRITELN('Press C for April');

WRITELN('Press D for May');

WRITELN('Press E for june');

WRITELN('Press F for July');

WRITELN('Press G for August');

WRITELN('Press H For September');

WRITELN('Press I for October');

WRITELN('Press J for november');

WRITELN('Press K for December');

Select:=UPCASE(READKEY);

CASE Select OF

'A':SelectMonth:=1;

'B':SelectMonth:=3;

'C':SelectMonth:=4;

'D':SelectMonth:=5;

'E':SelectMonth:=6;

'F':SelectMonth:=7;

'G':SelectMonth:=8;

'H':SelectMonth:=9;

'I':SelectMonth:=10;

'J':SelectMonth:=11;

'K':SelectMonth:=12;

ELSE

WRITELN('Invalid input please try again');

WRITELN;

WRITELN('Press any key to continue');

READKEY;

END;

UNTIL (ORD(Select)>64) AND (ORD(Select)<75);

END;

BEGIN

IF Date= 31 THEN

BEGIN

REPEAT

CLRSCR;

WRITELN('Press A for January');

WRITELN('Press B for March');

WRITELN('Press C for May');

WRITELN('Press D for July');

WRITELN('Press E for August');

WRITELN('Press F for October');

WRITELN('Press G for December');

Select:=UPCASE(READKEY);

CASE Select OF

'A':SelectMonth:=1;

'B':SelectMonth:=3;

'C':SelectMonth:=5;

'D':SelectMonth:=7;

'E':SelectMonth:=8;

'F':SelectMonth:=10;

'G':SelectMonth:=12;

ELSE

WRITELN('Invalid input please try again');

WRITELN;

WRITELN('Press any key to continue');

READKEY;

END;

UNTIL (ORD(Select)>64) AND (ORD(Select)<70);

END;

END;

BEGIN

IF (Date >= 29) AND (Date<>31) THEN

BEGIN

REPEAT

CLRSCR;

WRITELN('Press A for January');

WRITELN('Press B for March');

WRITELN('Press C for April');

WRITELN('Press D for May');

WRITELN('Press E for june');

WRITELN('Press F for July');

WRITELN('Press G for August');

WRITELN('Press H For September');

WRITELN('Press I for October');

WRITELN('Press J for november');

WRITELN('Press K for December');

Select:=UPCASE(READKEY);

CASE Select OF

'A':SelectMonth:=1;

'B':SelectMonth:=3;

'C':SelectMonth:=4;

'D':SelectMonth:=5;

'E':SelectMonth:=6;

'F':SelectMonth:=7;

'G':SelectMonth:=8;

'H':SelectMonth:=9;

'I':SelectMonth:=10;

'J':SelectMonth:=11;

'K':SelectMonth:=12;

ELSE

WRITELN('Invalid input please try again');

WRITELN;

WRITELN('Press any key to continue');

READKEY;

END;

UNTIL (ORD(Select)>64) AND (ORD(Select)<76);

END;

END;

BEGIN

IF (Date< 31) AND (Date<>29) AND (Date<>30) THEN

BEGIN

REPEAT

CLRSCR;

WRITELN('Press A for January');

WRITELN('Press B for Febuary');

WRITELN('Press C for March');

WRITELN('Press D for April');

WRITELN('Press E for May');

WRITELN('Press F for june');

WRITELN('Press G for July');

WRITELN('Press H for August');

WRITELN('Press I For September');

WRITELN('Press J for October');

WRITELN('Press K for November');

WRITELN('Press L for December');

Select:=UPCASE(READKEY);

CASE Select OF

'A':SelectMonth:=1;

'B':SelectMonth:=2;

'C':SelectMonth:=3;

'D':SelectMonth:=4;

'E':SelectMonth:=5;

'F':SelectMonth:=6;

'G':SelectMonth:=7;

'H':SelectMonth:=8;

'I':SelectMonth:=9;

'J':SelectMonth:=10;

'K':SelectMonth:=11;

'L':SelectMonth:=12;

ELSE

WRITELN('Invalid input please try again');

WRITELN;

WRITELN('Press any key to continue');

READKEY;

END;

UNTIL (ORD(Select)>64) AND (ORD(Select)<77);

END;

END;

END;

FUNCTION SelectTime:LONGINT;

VAR

Entered:CHAR;

Correct:BOOLEAN;

BEGIN

REPEAT

Correct:=TRUE;

CLRSCR;

WRITELN('Press A for 8:00AM');

WRITELN('Press B for 10:00AM');

WRITELN('Press C for 12:00AM');

WRITELN('Press D for 2:00PM');

WRITELN('Press E for 4:00PM');

WRITELN('Press F for 6:00PM');

WRITELN('Press G for 8:00PM');

Entered:=UPCASE(READKEY);

CASE Entered OF

'A':SelectTime:=8;

'B':SelectTime:=10;

'C':SelectTime:=12;

'D':SelectTime:=14;

'E':SelectTime:=16;

'F':SelectTime:=18;

'G':SelectTime:=20;

ELSE

Correct:=FALSE;

IF correct=FALSE THEN

BEGIN

WRITELN;

WRITELN('Invalid input please select from one of the options above');

WRITELN;

WRITELN('Press any key to continue');

READKEY;

END;

END;

UNTIL Correct;

END;

FUNCTION GenerateId:LONGINT{Data type of value returned by the function};

VAR

CurrentId:LONGINT;

BEGIN

RESET(TaskIdFile);

READ(TaskIdFile,NewId);{Retrives value of current id}

CurrentId:=NewID.LAstId; {The current id is stored}

GenerateId:=NewId.LastId+1; {The current id is increased by one}

NewId.LastId:=NewId.lastID+1;{The currnet id within the task file is overwritten with the new id}

RESET(TaskIdFile);

This function returns a specific numerical value of the id of the new job that is being assigned.

WRITE(TaskIdFile,NewId);

END;

This function retrieves the value of the current id from the id file then increases the id by one, storing the new id and overwriting the current id within the file with the new id.

This procedure allows the user to assign new authorisations to members of staff as they gain an authorisation. This ensures that the information stored about each member is up to date and that members of staff can be immediately assigned to new jobs once they are qualified.

PROCEDURE AddAuths;

VAR

SerachCriteria,AuthChoice:CHAR;

AuthCode:STRING[5];

AuthIdenification:STRING[3];

BEGIN

SerachCriteria:='A';{parameter indicating which search to perform}

CLRSCR; {Clears the current contents of the screen}

WRITELN('Please select the member of staff that you wish to assign a authrisation');

SeArch(SerachCriteria); {Search routine is called with the parameter being passed to the routine}

CLRSCR;{Clears the current conets of the screen }

WRITELN('PLease slect the authrisation you wish to assign to the member of staff');

WRITELN('Press A to assing Aircraft jack');{Menu of available authorisations}

WRITELN('Press B to assing Cockpit entry ');

WRITELN('Press C to assing Hydrolic systems');

WRITELN('Press D to assing Apply external power');

WRITELN('Press E to assing Hydrolic rig');

WRITELN('Press F to assing Engine ringe');

WRITELN('Press G to assing Live weapons saftey');

WRITELN('Press X to exit');

Authchoice:=UPCASE(READKEY);

WITH Auth DO

CASE AuthChoice OF

'A':AuthIdenification:='A2J';

'B':AuthIdenification:='C2C';

'C':AuthIdenification:='H2S';

'D':AuthIdenification:='AEP';

'E':AuthIdenification:='H2R';

'F':AuthIdenification:='ENG';

'G':AuthIdenification:='LWS';

END;

WITH StaffAuths DO [Selects the type of record in use}

BEGIN

CLRSCR;

WRITELN('Please enter the given staff authrisation id');

READLN(AuthCode);

RESET(StaffAuthSFile);

REPEAT

READ(StaffauthSFile,StaffAuths);

UNTIL(StaffAuthId='-1'){Rouge value locating deleted record} OR (EOF(StaffAuthsFile));{Locates the end of the file]

StaffId:=Staff.StaffIdcode;

AuthorisationID:=AuthIdenification;

StaffAuthId:=AuthCode;

END;

WRITE(StaffAuthsFile,StaffAuths); {Writes/overwrites the information entered by the user to the file}

END;

Several clear screens have been utilised to aid user friendliness as by clearing the contents displayed on screen the new information being displayed is easier to identify and is therefore made more clear and is easier to read and understand.

First the user will select the member of staff that the authorisation is being assigned to (the search routine is called}. The user will then select the authorisation that is being assigned form a list of available authorisations, the user will then enter the id of the authorisation and the data is written to the file.

FUNCTION EnterPassword:STRING;

VAR

PassChar,EntryChar:CHAR;

Passphrase,Entry,Password1,Password2:STRING[15];

BEGIN

REPEAT

BEGIN

REPEAT

WRITELN('Please enter your password');

Passphrase:='';

REPEAT

Passchar:=READKEY;

IF ORD(Passchar) <> 13 THEN

BEGIN

Passphrase := Passphrase + passchar;

WRITE('\*');

END;

BEGIN

IF ORD(Passchar) =8 THEN

BEGIN

Passphrase := COPY(Passphrase,1,LENGTH(Passphrase)-2);

GOTOXY(WHEREX-2,WHEREY);

WRITE(' ');

GOTOXY(WHEREX-2,WHEREY);

END;

END;

UNTIL ORD(Passchar) =13;

IF (LENGTH(Passphrase)>=5) AND (LENGTH(Passphrase)<=10) THEN

Password1:=Passphrase

ELSE

BEGIN

WRITELN('A password must consist of between 5 to 10 charecters');

WRITELN;

WRITELN('Please re enter your password');

WRITELN;

WRITELN('Press any key to continue');

READKEY;

END;

UNTIL(LENGTH(Passphrase)>=5) AND (LENGTH(Passphrase)<=10);

BEGIN

REPEAT

WRITELN;

WRITELN('Please verify this password');

WRITELN;

WRITELN('Please enter your password here');

Entry:='';

REPEAT

EntryChar:=READKEY;

IF ORD(EntryChar) <> 13 THEN

BEGIN

Entry := Entry + EntryChar;

WRITE('\*');

END;

BEGIN

IF ORD(EntryChar) =8 THEN

BEGIN

Passphrase := COPY(Entry,1,LENGTH(Entry)-2);

GOTOXY(WHEREX-2,WHEREY);

WRITE(' ');

GOTOXY(WHEREX-2,WHEREY);

END;

END;

UNTIL ORD(EntryChar) =13;

IF (LENGTH(Entry)>=5) AND (LENGTH(Entry)<=10) THEN

Password2:=Entry

ELSE

BEGIN

WRITELN('A password must consist of between 5 to 10 charecters');

WRITELN;

WRITELN('Please re enter your password');

WRITELN;

WRITELN('Press any key to continue');

READKEY;

END;

UNTIL(LENGTH(Entry)>=5) AND (LENGTH(Entry)<=10);

IF Password1=Password2 THEN

EnterPassword:=Password1

ELSE

BEGIN

WRITELN;

WRITELN('Error the passwords do not match please re enter the desired password');

WRITELN;

WRITELN('Press any key to continue');

READKEY;

END;

END;

END;

UNTIL Password1=Password1;

END;

PROCEDURE CollectData;

VAR

Option:CHAR;

JobSelection:CHAR;

Entry,Passphrase:STRING[15];

LargestId,JetId:STRING[5];

Day:STRING[2];

AnotherAdd,Find,AnotherSearch,Select,RankChoice,PassChar:CHAR;

Weekday,Valid,Exit,Requirment,StaffTest,SearchAgain:BOOLEAN;

NewId,StaffPostion,NewDay,MonthChoice,TodayNo,Count,MemberCount,Time,Sub,MembersAdded,Nostr,ErrCode,SelectedDay:LONGINT;

NewtaskID:LONGINT;

JobMember:StaffArray;

BEGIN

WRITELN('Please slect the file you wish to appened');

WRITELN('Press A to appened the staff file');

WRITELN('Press B to appened the Aircraft file');

WRITELN('Press C to appened the Job file');

WRITELN('Press D to assign a new job to the calander');

WRITELN('Press E to assign new authrisations');

Choice:=UPCASE(READKEY);

CASE Choice Of

'A': BEGIN

WITH NewRecStaff DO

BEGIN

CLRSCR;

REPEAT

WRITELN('Please enter your assinged Id code');

READLN(StaffIdcode);

IF (LENGTH(StaffIdcode)<>5) THEN

BEGIN

WRITELN('Invalid entry please re enter the assinged id code');

WRITELN;

WRITELN('press anykey to continue');

READKEY;

WRITELN;

END;

UNTIL (LENGTH(StaffIdcode)=5);

REPEAT

WRITELN('Please enter your forename');

READLN(Forename);

IF (LENGTH(Forename)>15) OR (LENGTH(Forename)=0) THEN

BEGIN

WRITELN('Error there is an invalid amount of charecters within this name pleaase re enter your name or contact the system admisartor');

WRITELN;

WRITELN('Press anykey to continue');

READKEY;

WRITELN;

END;

UNTIL(LENGTH(Forename)<=15) AND (LENGTH(Forename)>0);

REPEAT

WRITELN('Please enter your surname');

READLN(Surname);

IF (LENGTH(Surname)>15) OR (LENGTH(Surname)=0) THEN

BEGIN

WRITELN('Error there is an invalid amount of charecters within this name pleaase re enter your name or contact the system admisartor');

WRITELN;

WRITELN('Press anykey to continue');

READKEY;

WRITELN;

END;

UNTIL(LENGTH(Surname)<=15) AND (LENGTH(Surname)>0);

WRITELN('Please enter your rank');

BEGIN

REPEAT

CLRSCR;

WRITELN('Please select one of the following ranks');

WRITELN('Press 1 for SAC');

WRITELN('Press 2 for Corporal');

WRITELN('Press 3 for sergant');

WRITELN('Press 4 for flight Sergant');

RankChoice:=UPCASE(READKEY);

CASE RankChoice OF

'1':Staff.Rank:='SAC';

'2':Staff.Rank:='Corporal';

'3':Staff.Rank:='Sergant';

'4':Staff.Rank:='Flight Sergant';

ELSE

WRITELN('Invalid input please try agian');

WRITELN;

WRITELN('Press anykey to continue');

READKEY;

END;

UNTIL (ORD(RankChoice) <53) AND (ORD(RankChoice) >48);

END;

REPEAT

WRITELN('Please select your job role');

WRITELN('1:Avionics');

WRITELN('2:Mechainc');

WRITELN('3 Weapons technican');

Option:=UPCASE(READKEY);

CASE Option OF

'1':JobFeild:='Avionics';

'2':JobFeild:='Mechanics';

'3':Jobfeild:='Weapons technican';

ELSE

WRITELN('Invalid input please try agian');

WRITELN;

WRITELN('Press amy key to continue');

READKEY;

END;

UNTIL (ORD(Option) > 48) AND (ORD(OPTION) <52);

REPEAT

WRITELN('Please enter Your username below');

READLN(Entry);

IF (LENGTH(Entry)=0) OR (LENGTH(Entry)>25) THEN

BEGIN

WRITELN('Error there is an ivalid amount of charecters within the user name please re enter your username');

WRITELN;

WRITELN('Press any key to continue');

READKEY;

WRITELN;

END;

UNTIL(LENGTH(Entry)>0)AND (LENGTH(Entry)<=25);

UserName:=UPPERCASE(Entry);

WRITELN;

Password:=Enterpassword;

WRITELN;

END;

END;

'B':BEGIN

WITH NewRecJet DO

BEGIN

REPEAT

WRITELN('Pleasse enter the provided aircraft Id');

READLN(AircraftId);

IF LENGTH(AircraftId)<>3 THEN

BEGIN

WRITELN('Error you have entered an invalid Id');

WRITELN;

WRITELN('Please re enter the aircaftId');

WRITELN;

WRITELN('Press any key to continue');

READKEY;

WRITELN;

END;

UNTIL LENGTH(AircraftId)=3;

REPEAT

WRITELN('Please enter the provided call sign');

READLN(Callsign);

IF (LENGTH(Callsign)=0) OR (LENGTH(Callsign)>15) THEN

BEGIN

WRITELN('Error you have entered an invalid amoubt of charecters');

WRITELN;

WRITELN('Please re enter the aircaft call-sign');

WRITELN;

WRITELN('Press any key to continue');

READKEY;

WRITELN;

END;

UNTIL (LENGTH(CallSign)>0) AND (LENGTH(CallSign)<=15);

REPEAT

CLRSCR;

WRITELN('Please assign the suitable aircraft status');

WRITELN;

WRITELN('Please select one of the following options');

WRITELN('1:Serviable');

WRITELN('2:Job in progress');

WRITELN('3:Unservicable');

Option:=UPCASE(READKEY);

CASE Option OF

'1':AircraftStatus:='=Servicable';

'2':Aircraftstatus:='Job in progress';

'3':Aircraftstatus:='Unservicable';

ELSE

BEGIN

WRITELN('Invalid input please try again');

WRITELN;

WRITELN('Press anykey to continue');

READKEY;

END;

END;

UNTIL (ORD(Option) > 48) AND (ORD(OPTION) <52);

END;

END;

'C':BEGIN

WITH NewrecJob DO

BEGIN

WRITELN('Please enter the given job Id');

READLN(JobId);

WRITELN('Please enter the Job title');

READLN(JobTitle);

WRITELN('Please enter how the fault was identfied');

READLN(WhenhowFound);

END;

END;

'D':BEGIN

WRITELN('Please select the aircraft that needs servecing');

WRITELN;

WRITELN('Press anykey to continue');

Find:='B';

Search(Find);

JetId:=Jet.AircraftId;

REPEAT

CLRSCR;

WRITELN('Please select the job you wish to assign');

WRITELN('Press 1 to select wheel change');

WRITELN('Press 2 to select engine change');

WRITELN('Press 3 to select brake pad change');

WRITELN('Press 4 to select oil filter change');

WRITELN('Press 5 to select bulb change');

WRITELN('Press 6 to select flight control surface change');

WRITELN('Press 7 to select hydrolic actulator');

WRITELN('Press 8 to select role change');

JobSelection:=UPCASE(READKEY);

WITH Job DO

CASE JobSelection OF

'1':JobId:='A0001';

'2':JobId:='A0002';

'3':JobId:='A0003';

'4':JobId:='A0004';

'5':JobId:='A0005';

'6':JobId:='A0006';

'7':JobId:='A0007';

'8':JobId:='A0008';

ELSE

BEGIN

WRITELN('Invalid input please try again');

WRITELN;

WRITELN('Press anykey to continue');

READKEY;

END;

END;

UNTIL (ORD(JobSelection)>48) AND (ORD(JobSelection)<57);

BEGIN

MemberCount:=JobTest(Job.Jobid);

CLRSCR;

WRITELN;

BEGIN

FOR Sub:=1 TO 5 DO

JobMember[Sub]:='-1';

END;

BEGIN

REPEAT

POS:=0;

Count:=0;

MembersAdded:=0;

Exit:=FALSE;

Valid:=FALSE;

SearchAgain:=FALSE;

WRITELN('Please select each member of staff you wish to assign to the job');

BEGIN

WITH Staff DO

REPEAT

BEGIN

//REPEAT

CLRSCR;

RESET(StaffFile);

WRITELN('You have added ',Count,' Out of ',MemberCount,' ',MemberCount-count,' More members required');

READKEY;

WRITELN;

WRITELN('curently assinged to the job');

BEGIN

FOR Sub:=1 TO 5 DO

IF JobMember[Sub] <>'-1' THEN

BEGIN

MembersAdded:=MembersAdded+1;

REPEAT

READ(StaffFile,Staff);

UNTIL(StaffIdcode=Jobmember[Sub]) OR(EOF(StaffFile));

IF StaffIdcode=Jobmember[Sub] THEN

WRITE('ID Code: ',StaffIdcode,' Forenmae:',Forename,' Surname: ' ,Surname);

WRITELN;

END;

END;

IF MembersAdded=0 THEN

WRITELN('No members have been added yet');

WRITELN;

Find:='A';

Search(Find);

Valid:=FALSE;

Requirment:=FALSE;

StaffPostion:=FILEPOS(StaffFile);

IF AuthTest(Job.JobId,Staff.StaffIdCode)=TRUE THEN

Valid:=TRUE;

IF Valid=FALSE THEN

BEGIN

REPEAT

WRITELN('This staff member is not qualified');

WRITELN;

WRITELN('Do you wish to serach for another member of staff (Y=Yes N=No)');

AnotherSearch:=UPCASE(READKEY);

IF (AnotherSearch<>'Y') AND (AnotherSearch<>'N') THEN

BEGIN

WRITELN;

WRITELN('Invalid input please try again');

WRITELN('Press any key to continue');

READKEY;

END;

UNTIL(Anothersearch='Y') OR (AnotherSearch='N');

IF AnotherSearch='Y' THEN

SearchAgain:=TRUE;

END;

END;

IF Valid=TRUE THEN

BEGIN

Pos:=Pos+1;

Jobmember[Pos]:=StaffIdcode;

Count:=Count+1;

END;

IF Count>=MemberCount THEN

REPEAT

WRITELN('Do you wish to add another member of staff?(Y=Yes,N=No)');

Select:=UPCASE(READKEY);

IF (Select<>'Y') AND (Select<>'N') THEN

BEGIN

WRITELN('Invalid Input please try again');

WRITELN;

WRITELN('Press any key to continue');

READKEY;

END;

UNTIL(Select='Y') OR (Select='N');

IF (Count>=MemberCount) AND (Select='N') THEN

Requirment:=TRUE;

UNTIL(SearchAgain=FALSE) AND (Requirment=TRUE);

StaffTest:=MemberTest(Jobmember,MemberCount,Job.JobId);

IF StaffTest=FALSE THEN

BEGIN

WRITELN('The combanation of staff members is invalid plaese try again');

WRITELN;

WRITELN('Press any key to continue');

READKEY;

FOR Pos:=1 TO 5 DO

JobMember[Pos]:='-1';

END;

END;

UNTIL(StaffTest=TRUE);

BEGIN

WeekDay:=TRUE;

REPEAT

CLRSCR;

WRITELN('Please select the date that the job will take place');

WRITELN;

REPEAT

BEGIN

WRITELN('Please enter the day of the month that the job will take place');

READLN(Day);

VAL(Day,NoStr,Errcode);

IF Errcode<>0 THEN

BEGIN

WRITELN('Invalid input please enter an integar');

WRITELN;

WRITELN('Press any key to continue');

READKEY;

WRITELN;

END;

IF ErrCode=0 THEN

BEGIN

SelectedDay:=STRTOINT(Day);

IF SelectedDay>31 THEN

BEGIN

WRITELN('Invalid input the date entered must be smaller than the 31');

WRITELN;

WRITELN('Press any key to continue');

READKEY;

END;

END;

END;

UNTIL (SelectedDay<=31) AND (ErrCode=0);

WRITELN;

WRITELN('Press any key to continue');

READKEY;

WRITELN;

WRITELN('Please select the month during which the job will take place');

WRITELN;

MonthChoice:=SelectMonth(SelectedDay);

Newday:=(MonthChoice-1) \*31+SelectedDay;

ToDayNo:=Todaynumber(NewDay);

IF (TodayNo=6) OR (TodayNo=0) THEN

WeekDay:=FALSE

ELSE

Weekday:=TRUE;

IF WeekDay=FALSE THEN

BEGIN

WRITELN('Invalid input a job can not be assinged on a weekend');

WRITELN;

WRITELN('Press any key to continue');

READKEY;

WRITELN('TodayNo: ',TodayNo);

END;

UNTIL WeekDay=TRUE;

BEGIN

WRITELN('Please enter The time at which the job will take place');

Time:=SelectTime;

IF Time=8 THEN

Row:=1;

IF Time=10 THEN

Row:=2;

IF Time=12 THEN

Row:=3;

IF Time=14 THEN

Row:=4;

IF Time=16 THEN

Row:=5;

IF Time=18 THEN

Row:=6;

IF Time=20 THEN

Row:=7;

IF Time=22 THEN

Row:=8;

END;

BEGIN

NewTaskID:=GenerateID ;

FOR Sub:= 1 TO 5 DO

IF JobMember[Sub]<>'-1' THEN

BEGIN

RESET(StaffFile);

REPEAT

READ(StaffFile,Staff);

UNTIL (Staff.StaffIdcode=Jobmember[Sub]);

StaffPostion:=FILEPOS(StaffFile);

SEEK(StaffFile,StaffPostion-1);

Staff.Schedule[Row,NewDay]:=INTTOSTR(NewTaskID);

WRITE(StaffFile,Staff);

END;

END;

BEGIN

RESET(AircraftJobFile);

REPEAT

READ(AircraftJobFile,JetJob);

UNTIL(JetJob.AircraftJobId=-1) OR (EOF(AircraftJobFile));

WITH JetJob DO

BEGIN

AircraftJobId:=NewTaskId;

JobId:=job.JobId;

AircraftId:=JetId;

Complete:=FALSE;

WRITE(AircraftJobFile,JetJob);

END;

END;

END;

END;

END;

END;

'E':Addauths;

END;

END;

This procedure is responsible for adding new records to the system by allowing the user to select which file they wish to append and identifying an insertion point (deleted record or the end of the file) to write the data to once the relevant data has been collected.

PROCEDURE AddRecord;

VAR

AnotherAdd:CHAR;

FileId:STRING[7];

BEGIN

REPEAT

CLRSCR;

Collectdata;

CASE Choice OF

'A':BEGIN

RESET(StaffFile);{Sets file pointer to the start of the file}

REPEAT {Searches for insertion point}

READ(StaffFile,Staff);

UNTIL (Staff.StaffIdcode ='-1') OR (EOF(StaffFile));

IF Staff.StaffIdcode ='-1' THEN {tests for rogue value}

SEEK(StaffFile,FILEPOS(StaffFile)-1); );{Sets file pointer to the start of the record}

WRITE(StaffFile,NewRecStaff);

WRITELN('Do you whis to add another record?(Y=yes N=No)');

AnotherAdd:=UPCASE(READKEY);

END;

'B':BEGIN

RESET(AircraftFile); {Sets file pointer to the start of the file}

REPEAT {Searches for insertion point}

READ(AircraftFile,Jet);

UNTIL (Jet.AIrcraftId ='-1') OR (EOF(AIrcraftFile));

IF Jet.AircraftId='-1' THEN {tests for rogue value}

SEEK(AircraftFile, FILEPOS(AircraftFile)-1); );{Sets file pointer to the start of the record}

WRITE(AircraftFile, NewRecJet);

WRITELN('Do you want to add another record?(Y/N Y=Yes N=No)');

AnotherAdd :=UPCASE(READKEY);

END;

'C':BEGIN

RESET(JobFile); {Sets file pointer to the start of the file}

REPEAT {Searches for insertion point}

READ(JobFile,Job);

UNTIL (Job.JobId ='-1') OR (EOF(JobFile));

IF Job.JobId ='-1' THEN {tests for rogue value}

SEEK(JobFile, FILEPOS(JobFile)-1);{Sets file pointer to the start of the record}

WRITE(JobFile,NewRecJob);

WRITELN('Do you want to add another record?(Y/N Y=Yes N=No)');

AnotherAdd :=UPCASE(READKEY);

END;

'D':BEGIN

REPEAT

WRITELN('Do you want to add another record?(Y/N Y=Yes N=No)');

AnotherAdd :=UPCASE(READKEY);

IF (AnotherAdd <>'Y') AND (Anotheradd<>'N') THEN {look up check}

Upon collecting the relevant data to add to the selected file (see collect data) a point of insertion is identified by testing the id of each record for a rogue value (-1) to locate a deleted record to overwrite. If no deleted record is found the data is written to the end of the file. The user is then asked if they wish to add another record. This procedure is repeated until the user selects no,

WRITELN('Invalid input please try again');

WRITELN;

WRITELN('Press anykey to continue');

READKEY;

UNTIL (Anotheradd='Y') OR (AnotherAdd='N');

END;

END;

UNTIL AnotherAdd ='N';

END;

PROCEDURE Addjob;

VAR

Complete,Qualified,Valid,Free,Trained:BOOLEAN;

Times,Count,RankRequirment,Pos,TodayYear,TodayMonth,TodayDay,TodayNo:LONGINT;

Differance,Largest,day,JobDay,Latest,StaffLocation,NewTask:LONGINT;

JobMembers:StaffArray;

Find,JobSelection:CHAR;

JetId:STRING[5];

JobRole:STRING[15];

BEGIN

WRITELN('Please select the aircraft that needs servecing');

WRITELN;

WRITELN('Press any key to continue');

Find:='B';

Search(Find);

JetId:=Jet.AircraftId;

REPEAT

CLRSCR;

WRITELN('Please select the job you wish to assign');

WRITELN('Press 1 to select wheel change');

WRITELN('Press 2 to select engine change');

WRITELN('Press 3 to select brake pad change');

WRITELN('Press 4 to select oil filter change');

WRITELN('Press 5 to select bulb change');

WRITELN('Press 6 to select flight control surface change');

WRITELN('Press 7 to select hydrolic actulator');

WRITELN('Press 8 to select role change');

JobSelection:=UPCASE(READKEY);

WITH Job DO

CASE JobSelection OF

'1':JobId:='A0001';

'2':JobId:='A0002';

'3':JobId:='A0003';

'4':JobId:='A0004';

'5':JobId:='A0005';

'6':JobId:='A0006';

'7':JobId:='A0007';

'8':JobId:='A0008';

ELSE

BEGIN

WRITELN;

WRITELN('Invalip input please enter a value displayed within the menu');

WRITELN;

WRITELN('Press any key to continue');

READKEY;

END;

END;

UNTIL (ORD(JobSelection)>48) AND (ORD(JobSelection)<57);

GETDATE(TodayYear,TodayMonth,TodayDay,TodayNo);

Day:=(TodayMonth-1)\*31+TodayDay;

Row:=1;

Count:=0;

Qualified:=FALSE;

BEGIN

FOR POS:=1 TO 5 DO

JobMembers[Pos]:='-1';

END;

Free:=FALSE;

Valid:=FALSE;

Times:=JobTest(Job.JobId);

RankRequirment:=RankTest(Staff.Rank);

JobRole:=RoleTest(Job.JobId);

WITH Staff DO

BEGIN

Pos:=0;

RESET(StaffFile);

REPEAT

READ(StaffFile,Staff);

Trained:=RoleCheck(StaffIdcode,JobRole);

Qualified:=AuthTest(Job.JobId,StaffIdcode);

IF (Qualified=TRUE) AND (Trained=TRUE) THEN

BEGIN

Pos:=Pos+1;

Jobmembers[Pos]:=StaffIdcode;

Count:=Count+1;

Qualified:=FALSE;

END;

UNTIL (Count=Times) OR (EOF(StaffFile));

IF Count=Times THEN

BEGIN

Valid:=MemberTest(JobMembers,Times,Job.JobId);

IF Valid = TRUE THEN

BEGIN

Latest:=0;

Largest:=0;

FOR Pos:=1 TO Times DO

BEGIN

RESET(StaffFile);

REPEAT

READ(StaffFile,Staff);

UNTIL(Staffidcode=JobMembers[Pos]);

BEGIN

REPEAT

IF Free=FALSE THEN

BEGIN

Day:=Day+1;

Row:=1;

END;

REPEAT

IF Schedule[Row,Day] ='-1' THEN

Free:=TRUE

ELSE Row:=Row+1;

UNTIL (Free) OR (Row>8);

UNTIL Free;

JobDay:=(TodayMonth-1)\*31 + TodayDay;

DIfferance:=Day-JobDay ;

IF Differance>Largest THEN

Largest:=Differance;

IF Row>Latest THEN

Latest:=Row;

END;

END;

END;

BEGIN

JobDay:=Day+Differance;

Newtask:=GenerateId;

BEGIN

FOR Pos:=1 TO Times DO

IF Jobmembers[Pos] <>'-1' THEN

BEGIN

RESET(StaffFile);

REPEAT

READ(StaffFile,Staff);

UNTIL Staff.StaffIdCode=JobMembers[Pos];

StaffLocation:=FILEPOS(StaffFile);

SEEK(StaffFile,StaffLocation-1);

Schedule[Row,day]:=INTTOSTR(NewTask);

WRITE(StaffFile,Staff);

END;

END;

WRITELN('Job succesfuly added press any key to continue');

READKEY;

WITH JetJob DO

BEGIN

RESET(AircraftJobFile);

REPEAT

READ(AircraftJobFile,JetJob);

UNTIL(AircraftJobId=-1) OR (EOF(AircraftJobFile));

AircraftJobId:=NewTask;

AircraftId:=JetId;

JobID:=Job.JObID;

Complete:=FALSE;

WRITE(AircraftJobFile,JetJob);

END;

END;

IF Count<> Times THEN

BEGIN

WRITELN('There are not enough valid staff members to complete this job');

WRITELN;

WRITELN('Press any key to continue');

READKEY;

END;

END;

END;

END;

PROCEDURE Delete;

VAR

DelOption,SearchChoice:CHAR;

AnotherDel:CHAR;

MemberID:STRING[5];

BEGIN

REPEAT

CLRSCR;

WRITELN('Please select one of the following options');

WRITELN;

WRITELN('Press A to search for a member of staff');

WRITELN('Press B to search for a aircraft');

Choice:=UPCASE(READKEY);

CASE Choice OF

'A': BEGIN

SearchChoice:='A';

Search(SearchChoice);

MemberId:=Staff.StaffIdCode;

REPEAT

WRITELN('Are you sure you want to delete this item(Y/N)');

DelOption:=UPCASE(READKEY);

IF (DelOption<'Y') AND (DelOption<>'N') THEN

BEGIN

WRITELN('Invalid input please try again');

WRITELN;

WRITELN('Press anykey to continue');

READKEY;

END;

UNTIL (DelOption='Y') OR (DelOption='N');

CASE DelOption OF

'Y': BEGIN

REPEAT

Staff.StaffIdcode := '-1';

SEEK(StaffFile, FILEPOS(StaffFile)-1);

WRITE(StaffFile,Staff);

RESET(StaffAuthsFile);

WHILE NOT EOF(StaffAUthsFile) DO

BEGIN

WITH StaffAuths DO

BEGIN

READ(STaffAuthsFile,StaffAuths);

IF StaffAuths.StaffID=MemberID THEN

StaffauthID:='-1';

StaffId:='-1';

WRITE(StaffAuthsFile,StaffAuths);

END;

END;

WRITELN('Your item has been deleted');

WRITELN('Do you wish to delete another record (Y/N)');

AnotherDel:=UPCASE(READKEY);

IF (AnotherDel<>'Y') AND (AnotherDel<>'N') THEN

BEGIN

WRITELN('Invalid input please try again');

WRITELN;

WRITELN('Press anykey to continue');

READKEY;

END;

UNTIL (AnotherDel='Y') OR (AnotherDel='N');

END;

'N':BEGIN

REPEAT

WRITELN;

WRITELN('Do you wish to search for another item (Y/N)');

AnotherSearch:=UPCASE(READKEY);

IF (AnotherSearch <>'Y') AND (AnotherSearch <>'N') THEN

BEGIN

WRITELN('Invalid Input please try again');

WRITELN;

WRITELN('Press anykey to continue');

READKEY;

END;

UNTIL (AnotherSearch ='Y') OR (AnotherSearch = 'N');

END;

ELSE

WRITELN('Invalid input please try again');

END;

END;

'B':BEGIN

SearchChoice:='B';

Search(SearchChoice);

REPEAT

WRITELN;

WRITELN('Are you sure you want to delete this item(Y/N)');

DelOption:=UPCASE(READKEY);

IF (DelOption<'Y') AND (DelOption<>'N') THEN

BEGIN

WRITELN('Invalid input please try again');

WRITELN;

WRITELN('Press anykey to continue');

READKEY;

END;

UNTIL (DelOption='Y') OR (DelOption='N');

CASE DelOption OF

'Y': BEGIN

REPEAT

Jet.AircraftId := '-1';

SEEK(AircraftFile, FILEPOS(AircraftFile)-1);

WRITE(AircraftFile,Jet);

WRITELN('Your item has been deleted');

WRITELN('Do you wish to delete another record (Y/N)');

AnotherDel:=UPCASE(READKEY);

IF (AnotherDel<>'Y') AND (AnotherDel<>'N') THEN

BEGIN

WRITELN('Invalid input please try again');

WRITELN;

WRITELN('Press anykey to continue');

READKEY;

END;

UNTIL (AnotherDel='Y') OR (AnotherDel='N');

END;

'N':BEGIN

REPEAT

WRITELN('Do you wish to search for another item (Y/N)');

AnotherSearch:=UPCASE(READKEY);

IF (AnotherSearch <>'Y') AND (AnotherSearch <>'N') THEN

BEGIN

WRITELN('Invalid Input please try again');

WRITELN;

WRITELN('Press anykey to continue');

READKEY;

END;

UNTIL (AnotherSearch ='Y') OR (AnotherSearch = 'N');

END;

ELSE

WRITELN('Invalid input please try again');

END;

END;

ELSE

BEGIN

WRITELN('Invalid input please try again');

WRITELN;

WRITELN('Press any key to continue');

READKEY;

END;

END;

UNTIL (AnotherDel ='N') OR (AnotherSearch='N');

END;

PROCEDURE AmendPersonalData(AdminMember:BOOLEAN);

VAR

Selection:CHAR;

EmployeePostion:LONGINT;

BEGIN

REPEAT

WITH Staff DO

BEGIN

CLRSCR;

WRITELN('ID number: ',StaffIdCode);

WRITELN('Name: ',Forename,' ',Surname);

WRITELN('Job feild: ',Jobfeild);

WRITELN('Rank: ',Rank);

WRITELN;

REPEAT

WRITELN('Please select the information you wish to amend');

WRITELN;

WRITELN('Press 1 to amend the staff members name');

WRITELN('Press 2 to amend the staff members Job role');

IF AdminMember THEN

WRITELN('Press 3 to amend the staff members rank');

WRITELN('Press 4 to amend the staff members password');

WRITELN('Press E to exit');

Options:=UPCASE(READKEY);

CASE Options OF

'1':BEGIN

WRITELN('Please enter the required forename:');

READLN(Forename);

WRITELN;

WRITELN('Please enter the required surname');

READLN(Surname);

END;

'2':BEGIN

REPEAT

CLRSCR;

WRITELN('Please select your job role');

WRITELN('4:Avionics');

WRITELN('5:Mechainc');

WRITELN('6: Weapons technican');

Selection:=UPCASE(READKEY);

CASE Selection OF

'4':JobFeild:='Avionics';

'5':JobFeild:='Mechanics';

'6':Jobfeild:='Weapons technican';

ELSE

WRITELN('Invalid input please try agian');

WRITELN;

WRITELN('Press anykey to continue');

READKEY;

END;

UNTIL (ORD(Selection) < 55) AND (ORD(Selection) >51);

END;

'3':BEGIN

REPEAT

CLRSCR;

WRITELN('Please select one of the following ranks');

WRITELN('Press 1 for SAC');

WRITELN('Press 2 for Corporal');

WRITELN('Press 3 for sergant');

WRITELN('Press 4 for flight Sergant');

Selection:=UPCASE(READKEY);

CASE Selection OF

'1':Staff.Rank:='SAC';

'2':Staff.Rank:='Corporal';

'3':Staff.Rank:='Sergant';

'4':Staff.Rank:='Flight Sergant';

ELSE

WRITELN('Invalid input please try agian');

WRITELN;

WRITELN('Press anykey to continue');

READKEY;

END;

UNTIL (ORD(Selection) <53) AND (ORD(Selection) >48);

END;

'4':Password:=Enterpassword;

'E':WRITELN;

ELSE

BEGIN

IF Options <>'E' THEN

BEGIN

WRITELN('Invalid input please try again');

WRITELN('Press anykey to continue');

READKEY;

END;

END;

END;

UNTIL (ORD(Options) >51) OR (ORD(Options) <49);

SEEK(StaffFile,FILEPOS(StaffFile)-1);

WRITE(StaffFile,Staff);

END;

UNTIL Options='E';

END;

PROCEDURE AmendRecord;

VAR

Selection,SearchOption:CHAR;

RecordNo,StrNo,ErrCode:lONGINT;

Admin:BOOLEAN;

LastService:STRING[3];

BEGIN

Admin:=FALSE;

IF Staff.Rank='Flight Sergant' THEN

Admin:=TRUE;

REPEAT

CLRSCR;

WRITELN('please select which record you wish to amend');

WRITELN('Press A to amend Personal information');

IF Staff.Rank='Flight Sergant' THEN

BEGIN

WRITELN('press B to amend staff infomration');

WRITELN('Press C to amend aircraft information');

WRITELN('Press D to amend Job information');

END;

WRITELN('Press E to exit');

Choice:=UPCASE(READKEY);

CASE choice OF

'A':AmendPersonalData(Admin);

'B':BEGIN

SearchOption:='A';

Search(SearchOption);

AmendPersonalData(Admin);

SEEK(StaffFile,StaffRecNo);

READ(StaffFile,Staff);

END;

'C':BEGIN

REPEAT

SearchOption:='B';

SEARCH(SearchOption);

BEGIN

REPEAT

WITH Jet DO

BEGIN

REPEAT

WRITELN;

WRITELN('Please select the data that you wish to amend');

WRITELN('Press 1 to amend the flight hours since last y service');

WRITELN('Press 2 to amend the status of the aircraft');

WRITELN('Press E to exit');

Options:=UPCASE(READKEY);

CASE Options OF

'1':BEGIN

REPEAT

BEGIN

REPEAT

WRITELN('Please enter the flight houes J since last service:');

READLN(LastService);

WRITELN;

VAL(LastService,StrNo,ErrCode);

IF ErrCode <> 0 THEN

BEGIN

WRITELN('Invalid input please enter J an integar');

WRITELN;

WRITELN('Press any key to G continue');

READKEY;

END;

UNTIL ErrCode=0;

U Flight\_timefromService:=STRTOINT(LastService);

IF Flight\_timefromService<0 THEN

BEGIN

WRITELN('Invalid input please re enter g the data');

WRITELN;

WRITELN('Press any key to continue');

READKEY;

END;

END;

UNTIL Flight\_timefromService>=0;

END;

'2':BEGIN

REPEAT

CLRSCR;

WRITELN('Please select the status of the aircraft');

WRITELN('4: Servicable');

WRITELN('5: Unservicable');

WRITELN('6: Job in progress');

WRITELN('7: Airborn');

Selection:=UPCASE(READKEY);

CASE Selection OF

'4':Aircraftstatus:='Servicable';

'5':Aircraftstatus:='Unservicable';

'6':BEGIN

Aircraftstatus:='Job in progress';

Flight\_timefromservice:=0;

END;

'7':Aircraftstatus:='Airborn';

ELSE

BEGIN

WRITELN;

WRITELN('Invalid input please try agian');

WRITELN;

WRITELN('Press any key to continue');

READKEY

END;

END;

UNTIL (ORD(Selection) < 56) AND (ORD(Selection) >51);

END;

'E':WRITELN;

ELSE

BEGIN

WRITELN('Invalid input please try again');

WRITELN('Press anykey to continue');

READKEY;

END;

END;

UNTIL Options='E';

END;

BEGIN

REPEAT

WRITELN;

WRITELN('Do you wish to amend another data item?(Y=yes, N=No)');

Entrey:=UPCASE(READKEY);

IF (Entrey <> 'Y') AND (Entrey <>'N') THEN

BEGIN

WRITELN('Invalid input please try again');

WRITELN;

WRITELN('Press anykey to continue');

READKEY;

END;

UNTIL (Entrey ='Y') OR (Entrey='N');

END;

UNTIL (ORD(Options) >51) AND (ORD(Options) <49) OR (Entrey='N');

END;

BEGIN

REPEAT

WRITELN;

WRITELN('Do you wish to search for another recrod to amend? (Y=Yes, N=No)');

Anothersearch:=UPCASE(READKEY);

IF (AnotherSearch <> 'N') AND (AnotherSearch <>'Y') THEN

BEGIN

WRITELN('Invalid input please try again');

WRITELN;

WRITELN('Press anykey to continue');

READKEY;

END;

UNTIL (AnotherSearch='Y') OR (AnotherSearch='N');

END;

UNTIL (Located) OR (Options='N') OR (AnotherSearch='N');

SEEK(AircraftFile,FILEPOS(AircraftFile)-1);

WRITE(AircraftFile,Jet);

END;

'E':WRITELN;

ELSE

BEGIN

WRITELN;

WRITELN('Invalid input please try again');

WRITELN;

WRITELN('Press any key to continue');

READKEY;

END;

END;

UNTIL Choice='E';

END;

PROCEDURE Backup;

VAR

StaffBackup : FILE OF StaffRec;

JobBackUp : FILE OF JobRec;

AircraftBackup : FILE OF AircraftRec;

AuthBackup : FILE OF Authorisationcodes;

StaffAuthBackup: FILE OF StaffAuthorisations;

WorkBackup : FILE OF WorkCodes;

WorkJobBackup : FILE OF WorkJobRec;

JetJobBackup : FILE OF AircraftJobRec;

TaskBackup : FILE OF TaskIdRec;

Select:CHAR;

BEGIN

REPEAT

WRITELN('Are you sure you wish to create a backup of the current data(Y=Yes,N=No)');

Select:=UPCASE(READKEY);

IF (Select<>'Y') AND (Select<>'N') THEN

BEGIN

WRITELN('Error invalis input please try again');

WRITELN;

WRITELN('Press anykey to continue');

READKEY;

END;

UNTIL (Select='Y') OR (Select='N');

IF Select='Y' THEN

BEGIN

WRITELN('Please select the drive that you wish to use');

Drive:=UPCASE(READKEY);

ASSIGN(StaffBackup, Drive+':\Adam sixth form\Cg4 Backup files\StaffBackup.dta');

ASSIGN(JobBackup, Drive+':\Adam sixth form\Cg4 Backup files\JobBackUp.dta');

ASSIGN(AircraftBackup, Drive+':\Adam sixth form\Cg4 Backup files\AircraftBackup.dta'); ASSIGN(AuthBackup, Drive+':\Adam sixth form\Cg4 Backupfiles\AuthBackup.dta');

ASSIGN(StaffAuthBackup, Drive+':\Adam sixth form\Cg4 Backup files\StaffAuthBackup.dta');

ASSIGN(WorkBackup, Drive+':\Adam sixth form\Cg4 Backup files\WorkBackup.dta');

ASSIGN(WorkJobBackup, Drive+':\Adam sixth form\Cg4 Backup files\WorkJobBackup.dta');

ASSIGN(JetJobBackup, Drive+':\Adam sixth form\Cg4 Backup files\JetJobBackup.dta');

ASSIGN(TaskBackup, Drive+':\Adam sixth form\Cg4 Backup files\TaskIdBackup.dta');

REWRITE(StaffBackup);

RESET(StaffFile);

REPEAT

READ(StaffFile,Staff);

WRITE(StaffBackup,Staff);

UNTIL EOF(StaffFile);

REWRITE(JobBackup);

RESET(JobFile);

REPEAT

READ(JobFile,Job);

WRITE(JobBackup,Job);

UNTIL EOF(JobFile);

REWRITE(AircraftBackup);

RESET(AircraftFile);

REPEAT

READ(AircraftFile,Jet);

WRITE(AircraftBackup,Jet);

UNTIL EOF(AircraftFile);

REWRITE(AuthBackup);

RESET(AuthFile);

REPEAT

READ(AuthFile,Auth);

WRITE(AuthBackup,Auth);

UNTIL EOF(AuthFile);

REWRITE(StaffAuthBackup);

RESET(StaffAuthsFile);

REPEAT

READ(StaffAuthsFile,StaffAuths);

WRITE(StaffAuthBackup,StaffAuths);

UNTIL EOF(StaffAuthsFile);

REWRITE(WorkBackup);

RESET(WorkFile);

REPEAT

READ(WorkFile,Work);

WRITE(WorkBackup,Work);

UNTIL EOF(WorkFile);

REWRITE(WorkJobBackup);

RESET(WorkJobFile);

REPEAT

READ(WorkJobFile,WorkJob);

WRITE(WorkJobBackup,WorkJob);

UNTIL EOF(WorkJobFile);

REWRITE(JetJobBackup);

RESET(AircraftJobFile);

REPEAT

READ(AircraftJobFile,JetJob);

WRITE(JetJobBackup,JetJob);

UNTIL EOF(AircraftJobFile);

REWRITE(TaskBackup);

RESET(TaskIdFile);

REPEAT

READ(TaskIdFile,NewId);

WRITE(TaskBackup,NewId);

UNTIL EOF(TaskIDFile);

WRITELN('Backup complete press any key to continue');

READKEY;

END;

END;

PROCEDURE AdminMenu;

BEGIN

REPEAT

CLRSCR; {Clears the contents of the screen}

SEEK(StaffFile,StaffRecNo);{Sets the file pointer to the users record}

READ(StaffFile,Staff);{Reas in the users information}

WRITELN(' Welcome to RAf losimouth servecing schedule');

WRITELN; {Blank line}

WRITELN('Press A to view the data within the file');

WRITELN('Press B to search the system');

WRITELN('Press C to add a new Record to a file');

WRITELN('Press D to delete a record');

This procedure provides the first menu to admin users allowing them to navigate the system and access each task. This procedure also establishes the menu based user interface for the user aiding user friendliness as it helps to quickly familiarise the user with how the system operates. This also helps the user to quickly identify the desired task.

WRITELN('Press E to amend a record');

WRITELN('Press F to view your weekly roster');

WRITELN('Press G to auto assign a job');

WRITELN('Press H to Back up the system');

WRITELN('Press X to exit');

Choice:=UPCASE(READKEY);

CASE Choice OF

'A':ViewFile;

'B':Search('0');

'C':AddRecord;

'D':Delete;

'E':AmendRecord;

'F':ShowDate;

'G':AddJob;

'H':Backup;

'X':WRITELN;

ELSE {Look up check}

BEGIN

WRITELN;

WRITELN('Invalid input please selectedone of the options within the menu');

WRITELN;

WRITELN('Press anykey to continue');

READKEY;

END;

END;

UNTIL Choice ='X';

END;

This procedure utilises both blank lines and the clear screen function to aid user friendliness as by clearing the screen the information is easier to identify and less concentrated less screen space is required. As a result the information is more clear and understandable. The use of blank lines also helps to split up information helping to present information in a more logical manner making the information easier to distinguish.

This procedure allows the user to select the given task they wish to perform by providing a menu of all available tasks allowing the user to enter a character selecting the task they wish to undergo form the menu. A selection test is undergone comparing the character entered by the user against each value in the menu to identify the desired task. The file pointer is set to the users record each time the menu is accessed to ensure that the user is not confused with a record they previously accessed and that they can still access each of the desired tasks. This also helps to prevent non admin members accessing the menu.

This procedure provides the first menu to non admin users allowing them to navigate the system and access each task .Much like the admin menu this procedure also establishes the menu based user interface for the user aiding user friendliness as it helps to quickly familiarise the user with how the system operates. This also helps the user to quickly identify the desired task.

PROCEDURE MainMenu;

BEGIN

REPEAT

CLRSCR;{Clears the content of the screen}

WRITELN('Press A to view your personal data');

WRITELN('Press B to amend your personal data');

WRITELN('Press C to view your weekly roster');

WRITELN('Press X to exit');

Choice:=UPCASE(READKEY);

CASE Choice OF

'A':DisplayRecord;

'B':AmendRecord;

'C':ShowDate;

'X':WRITELN;

ELSE{Look up check}

BEGIN

WRITELN;

WRITELN('Invalid input please selected one of the options within the menu');

WRITELN;

WRITELN('Press any key to continue');

READKEY;

END;

END;

UNTIL Choice='X';

END;

A clear screen is unlisted to clear the previous contents of the screen to ensure that the information is clear and easy to identify by the user aiding user friendliness.

This procedure allows the user to select the given task they wish to perform by providing a menu of all available tasks allowing the user to enter a character selecting the task they wish to undergo form the menu.

This procedure allows the system to identify the current user trying to access the system in order to prevent unauthorised access and to assign the appropriate level of access to the user. The current user is identified based on the user name entered , following this confirm the users identity.

PROCEDURE LOGIN;

VAR

Search:BOOLEAN;

Found:BOOLEAN;

PassChar:CHAR;

Passphrase:STRING[10];

BEGIN

WITH Staff DO

REPEAT

CLRSCR; {Clears contents of screen}

RESET(StaffFile); {Sets file pointer to the start of the file}

WRITELN(' RAF Lossiemouth');

WRITELN; {Blank line}

WRITELN('Welcome to RAF lossiemouths servecing database');

RESET(StaffFile); {Sets file pointer to the start of the file}

WRITELN('Please enter your username');

READLN(Entrey);

WRITELN;

Search:=FALSE;

BEGIN

REPEAT

READ(StaffFile,Staff);

IF UPPERCASE(Username) = UPPERCASE(ENTREY) THEN

{Tests input against the user name within the current record}

BEGIN

Search:=TRUE;

BEGIN

WRITELN('Thankyou press anykey to continue');

READKEY;{allows the user to select when to proceed}

END;

END;

UNTIL (Search) OR (EOF(StaffFile));

Passphrase:='';{Sets the password to blank}

WRITELN('Please enter your password');

REPEAT

Passchar:=READKEY;

IF ORD(Passchar) <> 13 THEN {Test if enter has been pressed}

BEGIN

Passphrase := Passphrase + passchar;

{Adds the character entered to the password}

WRITE('\*'); {Dispalys teh character as an \*}

END;

BEGIN

IF ORD(Passchar) =8 THEN {Tests for backspace}

BEGIN

Passphrase := COPY(Passphrase,1,LENGTH(Passphrase)-2);

{Copies the current password }

GOTOXY(WHEREX-2,WHEREY);{Moves the curser two spaces to the lest}

WRITE(' ');{Inserts a blank character in place of current character}

GOTOXY(WHEREX-2,WHEREY);

END;

END;

UNTIL ORD(Passchar) =13; {Look up check}

BEGIN

Found:=FALSE;

IF UPPERCASE(Passphrase) =UPPERCASE(Password) THEN

{Compares password entered against the password stores within the current record}

Found:=TRUE;

StaffRecNo:=FILEPOS(StaffFile)-1; {Stores position of current record}

BEGIN

IF (Found=TRUE) AND (Search=TRUE) THEN

{Tests if both the username and password match}

BEGIN

WRITELN('Welcome!');

SEEK(StaffFile,StaffRecNo);

READ(StaffFile,Staff);

IF Staff.Rank ='Flight Sergant' THEN

{assigns level of access}

AdminMenu

ELSE

Mainmenu;

END;

IF (Found=FALSE) OR (Search=FALSE) THEN

BEGIN

WRITELN;

WRITELN('Incorrect password or username please try again');

WRITELN('Press anykey to continue');

READKEY;

CLRSCR;

END;

END;

END;

END;

UNTIL (FOUND=TRUE) AND(Search=TRUE); {look up check}

END;

Initially the user will be required to enter their user name at which point the staff file is searched serially until the user name entered by the user has been located. The user will then be required to enter their password which will be compared against the password store within the current record relating to the username entered. If either the user name or password are not found an error message is displayed. The rank of the user is then tested to determine the level of access to provide the user to directing them to either the admin menu or the main menu.

Several clear screens and blank lines have been utilised in order to provide a more user friendly environment. By providing clear screens more of the screen is made available to display the data as a result the info4rmation is less concentrated and is easier to identify by the user. The use of blank lines helps to naturally split up information making it easier for the user to distinguish between different bits of information.