Administrator manual

[Basic Information 3](#_Toc9043)

[Guide to the files in the main folder 3](#_Toc10414)

[Documents 3](#_Toc22615)

[Appointments 3](#_Toc1822)

[Doctors 3](#_Toc2743)

[Laz and module test files 3](#_Toc13456)

[Lib and backup 4](#_Toc11205)

[Patients 4](#_Toc28627)

[FileFile.txt 4](#_Toc20182)

[IDFile.txt 4](#_Toc30650)

[Guide to all variables 5](#_Toc5361)

[Integers: 5](#_Toc30864)

[Texts: 5](#_Toc21271)

[Strings: 5](#_Toc19911)

[Arrays: 5](#_Toc12188)

[T String Lists: 6](#_Toc613)

[Constants 6](#_Toc5987)

[Guide to the Procedures 7](#_Toc3535)

[Patient Module 7](#_Toc12807)

[AddPatient 7](#_Toc30678)

[ListPatients 9](#_Toc29747)

[EditPatient 10](#_Toc9776)

[RemovePatient 12](#_Toc10669)

[ViewPatient 13](#_Toc26313)

[Patient Screen 15](#_Toc23789)

[Doctor Module 16](#_Toc9614)

[AddDoctor 16](#_Toc28721)

[EditDoctor 18](#_Toc29090)

[GPScreen 20](#_Toc7280)

[Appointment Module 21](#_Toc21784)

[AddAppointment 21](#_Toc27798)

[ViewAppointment 24](#_Toc23468)

[ScheduleMenu 25](#_Toc16365)

[Print Daily List Module 26](#_Toc18445)

[PrintDailyList 26](#_Toc29870)

[Admin Module 27](#_Toc13499)

[AdminMenu 27](#_Toc9210)

[MainMenu Procedure 31](#_Toc1151)

[Troubleshooting 32](#_Toc25301)

[Procedure wont compile 32](#_Toc26639)

[My files aren't been created 32](#_Toc22497)

[The program keeps crashing for the users 32](#_Toc10584)

[The program freezes 32](#_Toc13279)

# Basic Information

The default username and password for the admin login system is:

* Username: Admin
* Password: Password

This can be changed easily, refer to the Changing admin log in in the Admin Module section of the manual.

# Guide to the files in the main folder

## Documents

This folder contains four word documents. The first is the Administrator manual which you are reading right now, it contains all the information neccesary for the administrator of the system, the second is the program information document which contains supplementary information about the system. The third is the progress report which contains information about the development of the system. The fourth is the user manual which contains information for users of the system.

## Appointments

The appointments folder contains three types of files. The first is the times and looks like 0600pm.txt for example. These files store all the information for a booked appointment such as the patient details and doctor details.

The second file is the AvApps.txt file which stores the list of times that have not been booked out and are shown to the user when they go to book a new appointment.

The third file is the BookedApps.txt file and stores all the times of the currently booked appointments and is used to generate a printout of the list of daily appointments.

## Doctors

This folder contains all the txt documents for the doctors that contains all the information such as filename, ID and name. The files typically follow a naming format of FName\_LName-ID.

## Laz and module test files

Laz and module test files are folders that contains a program I was using to test the patient module before adding it to the main program.

## Lib and backup

These folders simply contain files needed for lazarus the compiler used to create the program.

## Patients

Patients is a folder that contains all the patient file that contain details like filename, ID, names, addresses etc. They typically follow a naming format like FName\_LName-ID.

## FileFile.txt

The FileFile is a file that stores all the filenames of the patients registered with the system and is used to list all patients to the user.

## IDFile.txt

The ID File is used to store the last used ID so that a new one is assigned each time when creating a new patient or doctor, this prevents duplicated from occurring.

# Guide to all variables

### Integers:

* Choice: This variable is used in various menus to see what the user wants to do
* ID: This variable is used to assign and track what Ids have been used by which patients
* Counter and Counter2: These variables are used as counters in various parts of the program in order to keep track of what the program has already done.
* Index: This is used with a T String List to allow deletion of specific terms from a file
* KeyInteger: This variable stores what key a user has pressed as an integer for use in a case

### Texts:

* PatientFile: This file is used to store patient details
* IDFile: This file stores what ID was last used
* FilenameFile: This file stores the filenames of the patients registered in the system
* DoctorFile: This file stores the details of the doctors in the system
* AvApps: This file stores what appointment times are still available
* Appointment: This file stores the details of an appointment
* BookedApps: This file stores what times have been booked out for appointments

### Strings:

* Filename: This variable stores the filename of the various files used in the program
* Patient strings: This collection of strings store the various details of a patient
* Doctor strings: This collection of strings store the various details of a doctor
* Log in details: This collection of strings are used in the log in system to verify that the user is allowed into the admin menu
* Time: This variable is used to see what time the user wants to book an appointment for

### Arrays:

* Filearray: This array is used with the filename file to keep track of all the filenames and write them to the console and then write them back to the file.

### T String Lists:

* Filelist: This list is used to modify the filenamefile by removing a specific filename when removing a patient before writing the modified list back to the file
* Applist: This list reads all the appointments from the specified file and writes them to the terminal for viewing by the user.

### Constants

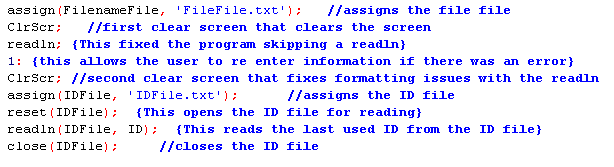
* AUsername: The username for the log in system
* APassword: The password for the log in system

# Guide to the Procedures

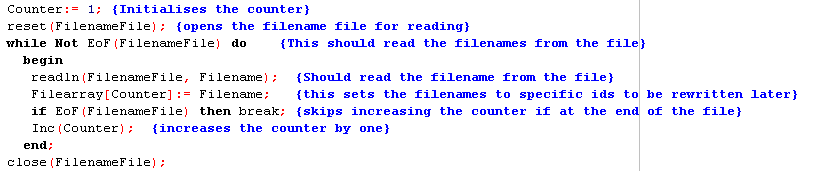
## Patient Module

### AddPatient

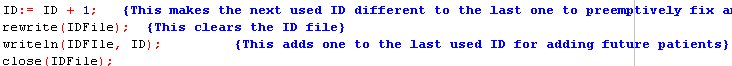
This procedure creates a new patient in the system. The way it works is it first assigns the fil file to the name FilenameFile, assigns the ID file to the name IDFile and reads the last used ID from the ID file.



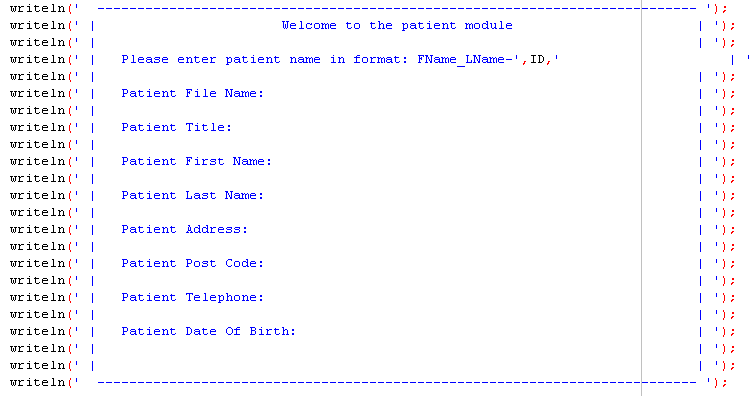
Next it opens the filename file and reads into an array the list of filenames already registered while also incrementing a counter to keep track of how many names were in the file, this doesn’t do anything at the moment but is needed later on in the program.



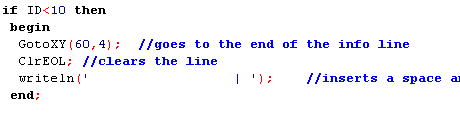
Next the program takes the ID it read at the beginning of the and increments it by one in order to get a fresh unique ID so as to prevent any duplicates from been made, it then writes this ID back to the ID File so it can be accessed next time a doctor or patient is made.



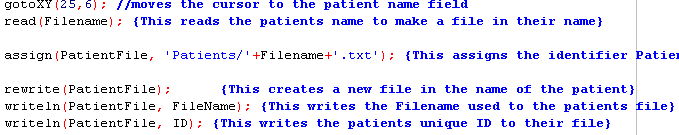
Next it loads up a menu with fields for each of the patients details as well as displaying what format the user needs to enter the name in with the new unique ID been shown at the end such as FName\_LName-ID



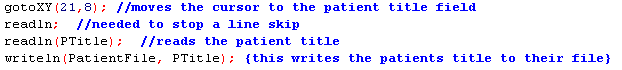
Next it checks if the ID been used is less then 10, if it is it inserts a small space in the GUI to fix a formatting issue.



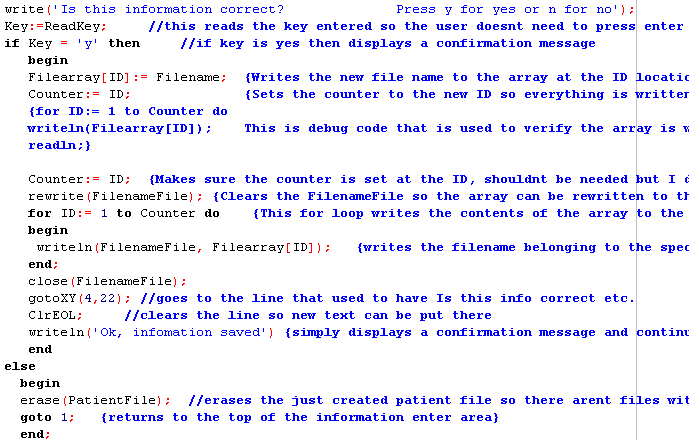
Next the program moves the cursor to the Patient File Name and reads the filename from the user, it then assigns the patient file to the filename just read and creates a file with that name as well as writing the filename and ID to that file.



The next section of the procedure just moves to the appropriate time and reads the patient detail before writing it to the file.

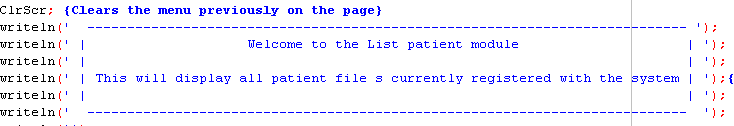


The procedure then checks with the user that the information that they entered is correct, if it isn't the procedure erases the just created file and returns to the top and starts over, if it is then it creates a new filename entry in the array with the entered filename at the ID and then writes the whole array back into the file.

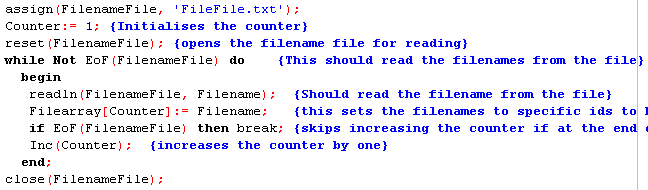


### ListPatients

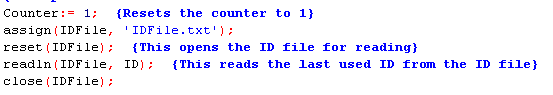
The List Patients procedure is used to list all patients registered in the system.The procedure starts by loading up a GUI that tells the user what the procedure does.



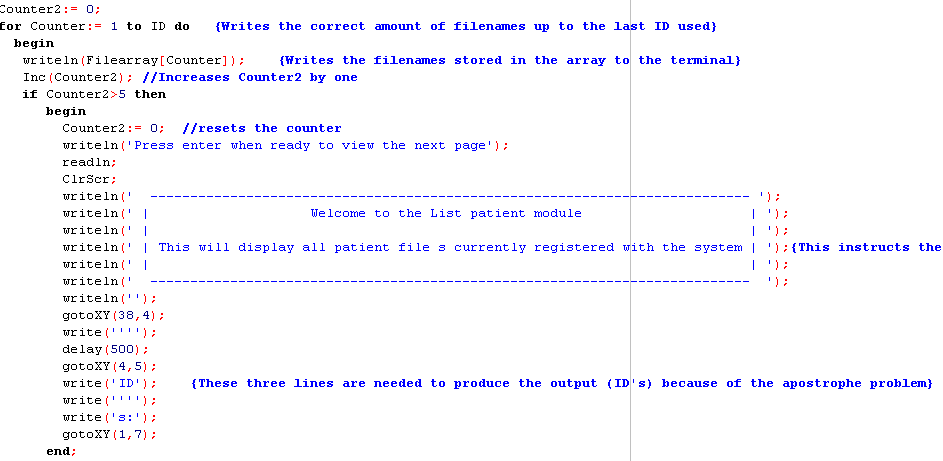
The procedure then assigns the FileFile to the name FilenameFile and reads the file names stored in the file into an array while also incrementing a counter to keep track of how many filenames there are.



The procedure then reads the last used ID from the ID file so that the correct number of names are written to the terminal.

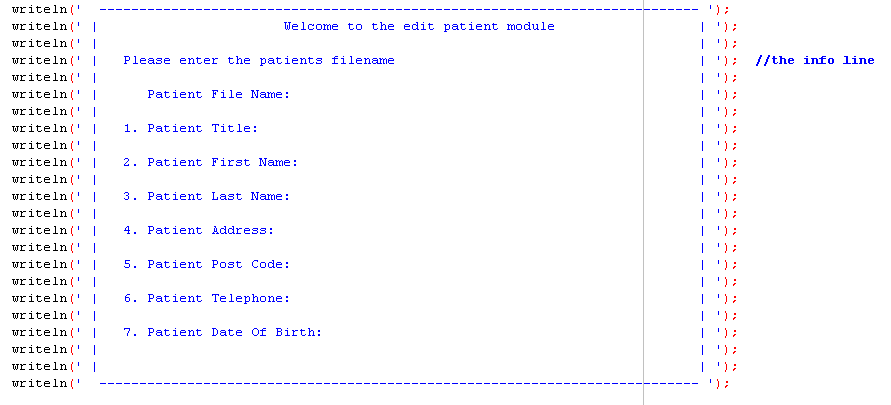


The procedure then writes the filenames to the terminal five at a time. Once it reaches five names it asks the user to press enter to view the next page, this prevents too many files showing at once as some would get pushed off the page.



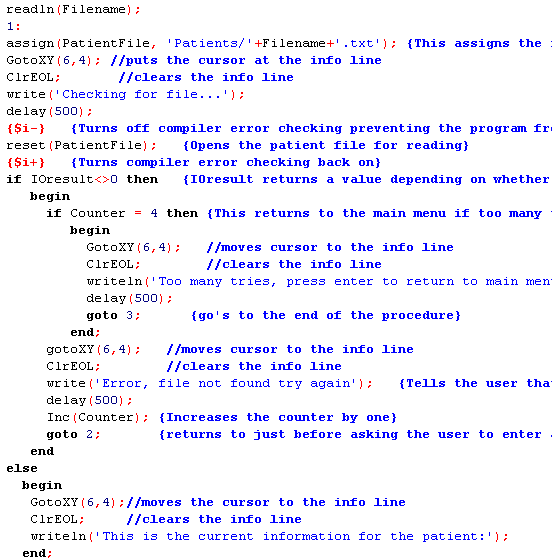
### EditPatient

The Edit Patient procedure allows a user to edit a users previously entered details.The first part of the procedure loads up a GUI to display the patient details in.

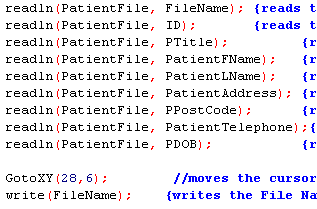


The next part asks the user to enter the filename belonging to the patient they wish to view. The procedure then checks if the file exists using the IOResult function which returns a number if an error occurs, if an error does occur it warns the user and returns to the top of the procedure, if four errors are made the procedure exits to the main menu to prevent been stuck in an infinite loop, if the file exists however the procedure simply proceeds

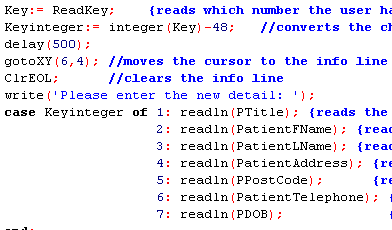
.



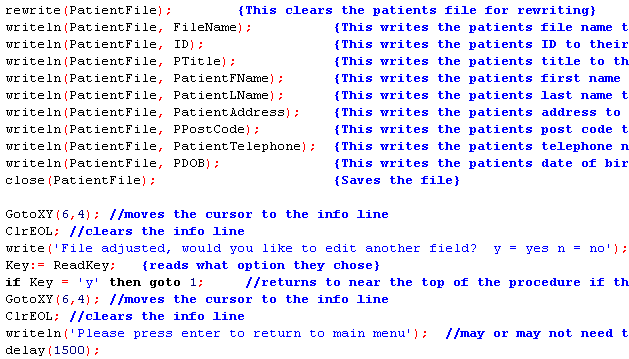
The procedure then reads the details from the patient file and writes them to the appropriate line in the menu.



The procedure then asks the viewer to select a number corresponding to which field they wish to edit. The procedure then reads a new value over the existing one for the variable they selected to edit.

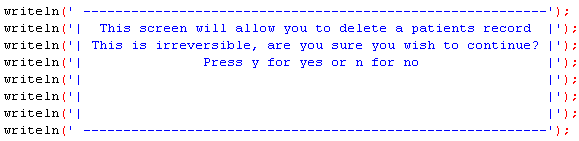


The procedure then rewrites the patients file and writes all the variables back to the file. After that it asks them if they wish to edit another field, if they select yes it goes back to the top of the procedure, otherwise if no it simply exits to the main menu.

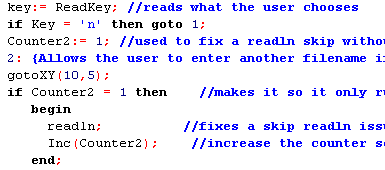


### RemovePatient

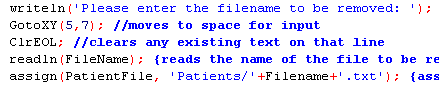
This procedure simply deletes a patients file. The procedure first loads up a simple GUI asking if the user really wants to delete the patients file.



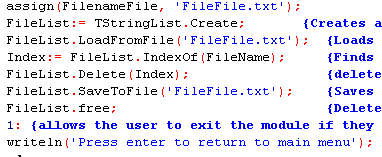
The procedure then reads what key they press, if they press n for no it sends them to the bottom of the procedure effectively exiting the procedure, if they press y for yes it continues on. This part of the procedure also features a small part that detects if this is the first time the procedure has been loaded through the counter2 variable, if it is the first time it runs a readln to fix a line skip issue, it then increments the counter so it wont run again if the user returns to the top of the procedure.



The procedure then asks the user for the filename of the patient to be deleted and then assigns the term patientfile to the filename just read from the user.



The procedure then checks that the file exists by using the IOResult function, for more information on this function read the Edit Patient procedure guide. If the file exists it is erased and then the procedure deletes its filename from the FileFile through a T String List and associated functions to prevent it showing up in the list patients procedure.

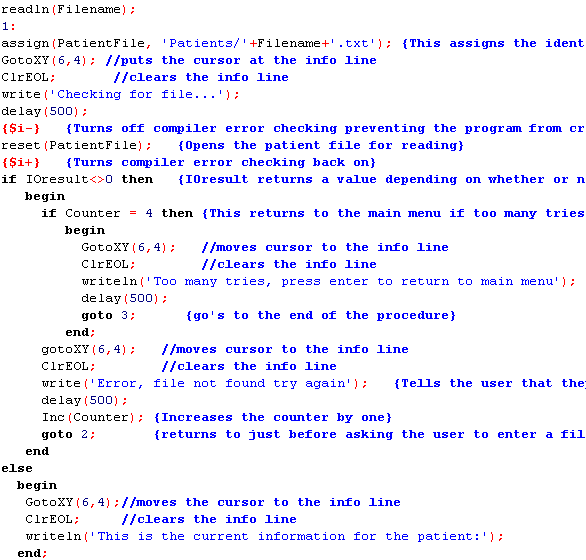


### ViewPatient

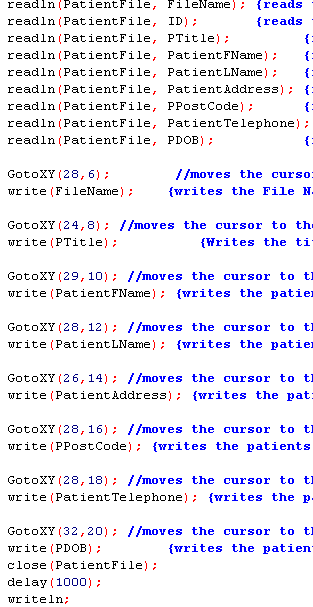
The View Patient procedure allows a user to view all the details stored on a particular patient. The procedure starts by loading up a GUI with lines for each detail.



The procedure then asks the user for the filename of the patient they wish to view. It then assigns that filename to the patientfile and checks if it exists through the IOResult function, for more information on this see the Edit Patient procedure guide, if it does it opens it for reading.

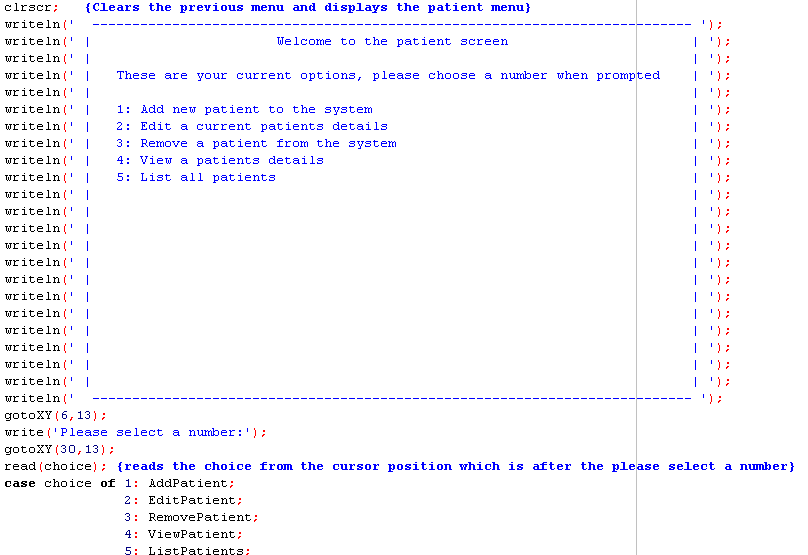


The procedure then reads the variables for the patient from the patient file and writes them to their appropriate line in the GUI.



### Patient Screen

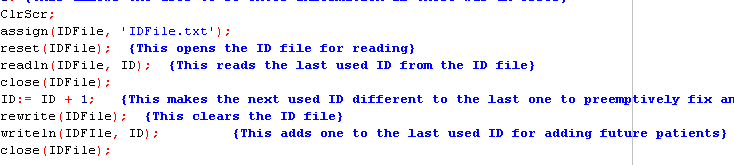
The patient screen simply serves as a menu to get to which ever procedure the user wishes to get to within the patient module. The first thing it loads up is a menu GUI. It then asks the user to select a number corresponding to which procedure they wish to go to, it then reads that number and loads the appropriate procedure.



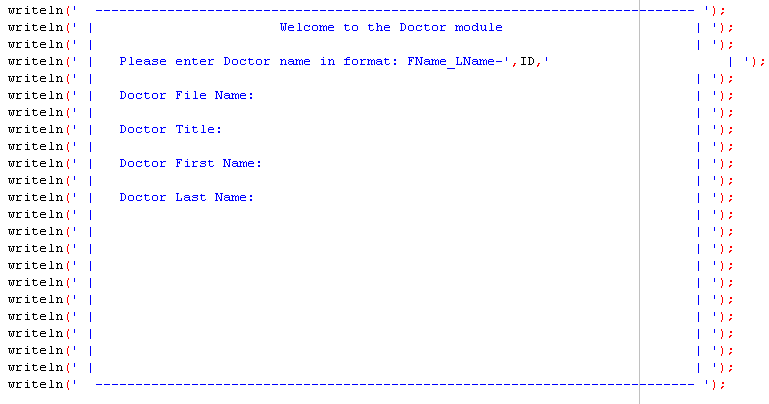
## Doctor Module

### AddDoctor

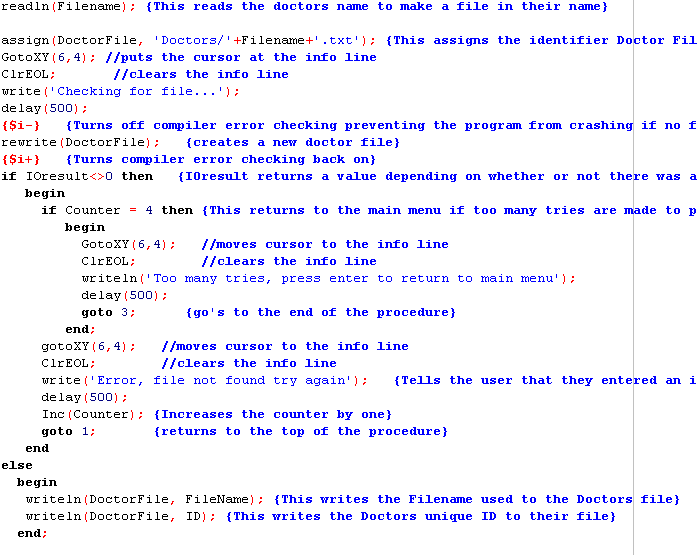
The Add Doctor procedure allows the user to add a new doctor and their details to the system. The procedure starts by reading the ID file for the last used ID. The procedure then increments the ID so that a new unique one is used. It then saves this new ID back to the ID File.



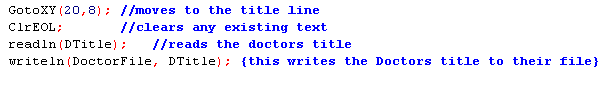
The program then loads up a GUI which has fields for all the details to be added and also tells the user what format they need to enter the name in with the unique ID at the end.



The procedure then reads the filename off the user and attempts to create a new doctor file with that name, if there is an error during creation it loops back up to the top otherwise it continues and writes the filename and ID to the new file. For more information on the error checking see the Edit Patient procedure guide.



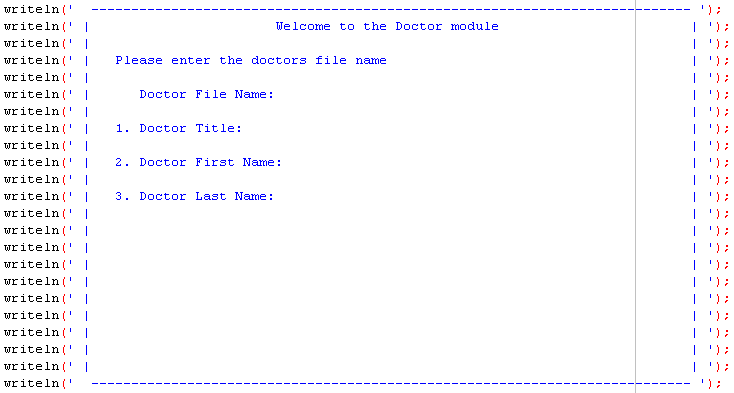
The procedure then moves the cursor to each appropriate line and asks the user to input that detail, the procedure then saves the detail to the doctor file and continues to the next detail.



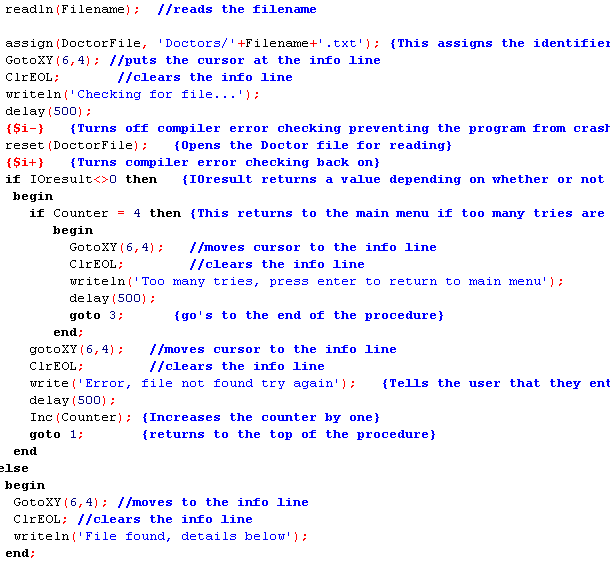
The procedure then checks with the user that the details are correct, if they are the user presses y and the program saves and continues, if the user presses n the program erases the just created doctor file and returns to the top of hte procedure.

### EditDoctor

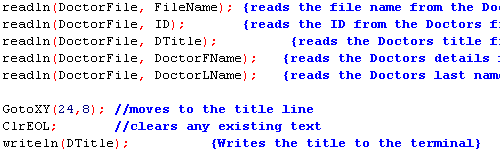
The Edit Doctor procedure is used by the user to edit the details of a created doctor. The first thing the procedure does is load up a GUI with lines for each detail of the doctor.



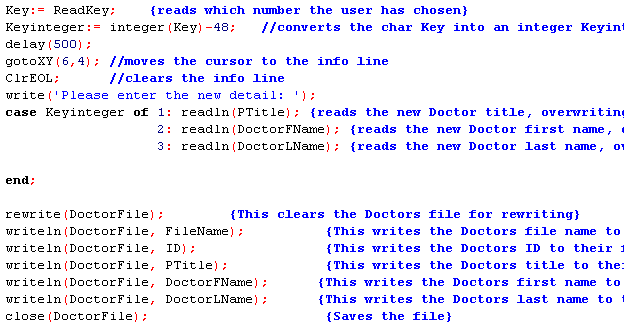
The procedure then reads the filename for the desired doctor from the user, assigns it and then verifies the file exists, if the file does exist it is then opened for reading. For more information on error checking and file verification see the Edit Patient procedure guide.



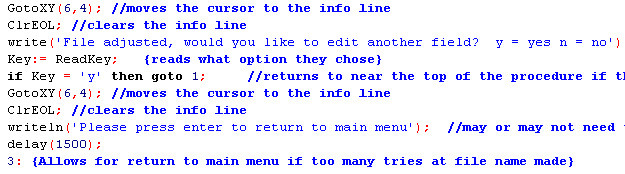
The procedure then reads the doctors details from the doctor file and writes them to the terminal at the appropriate line.



The procedure then asks the user to press a number corresponding to the field they wish to edit, it then converts the number to an integer and uses it in a case to select which field to edit, it then reads from the user a new value for the chosen variable over the existing one and then rewrites the doctor file with the variables including the new one.

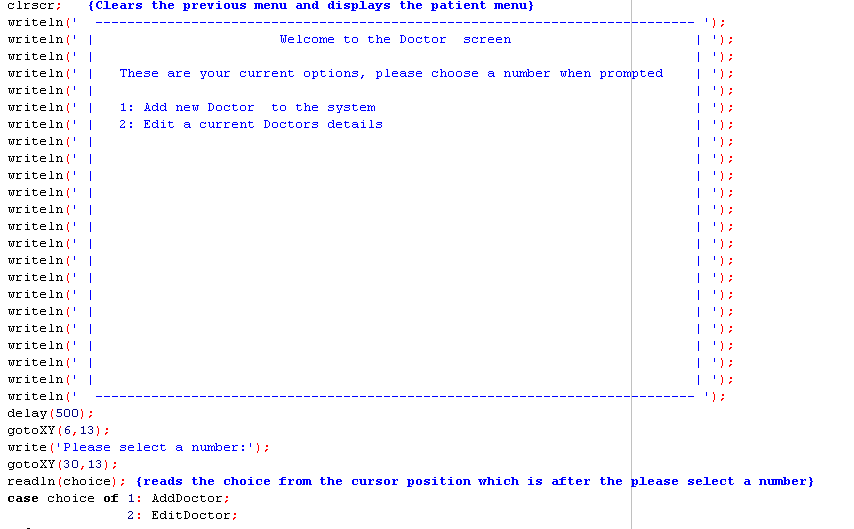


The procedure then asks the user if they would like to edit another field, if they press y it returns to the top of the procedure, if they press anything else it continues on to the main menu.



### GPScreen

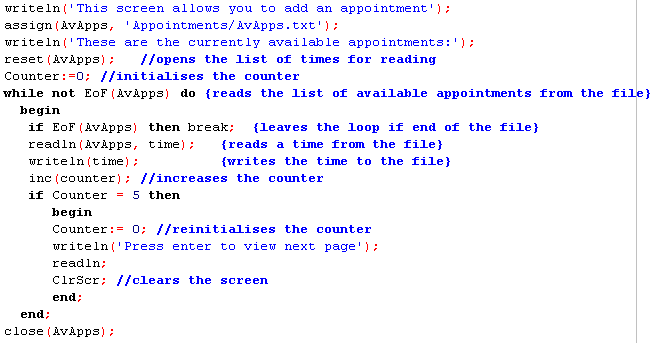
The GP Screen is the hub for the doctor module and serves as a gateway to all the procedures for the doctor module. It first loads up a GUI menu and then asks the user what number corresponding to a procedure they want to choose. It then reads that choice into a case and loads up a procedure depending on which choice they made.



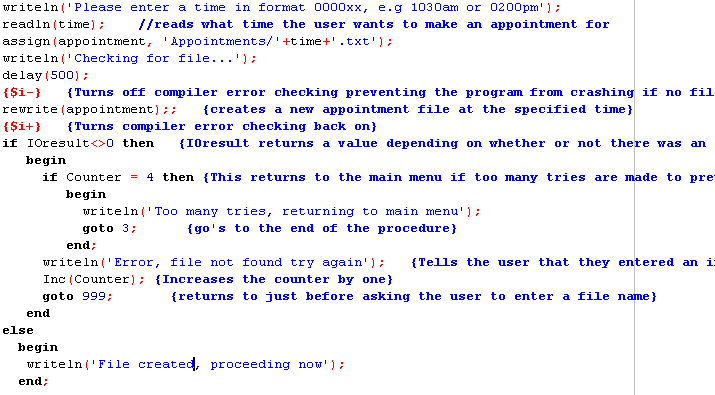
## Appointment Module

### AddAppointment

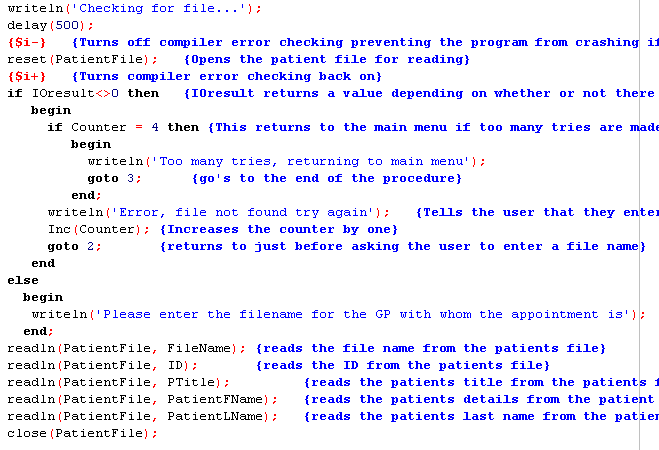
The Add Appointment procedure is used to create a new appointment in the system. The first thing the procedure does is assign the AvApps txt file to the variable AvApps. It then resets this file for reading and reads the times from the file and writes them onto the terminal, it does this for five times at a time by using the Counter variable, once five times have been written to the terminal it pauses and prompts the user to press enter to continue. It does this so as to prevent too many times from been written up which can be confusing to the user and can cause some times to get pushed off the top of the screen.



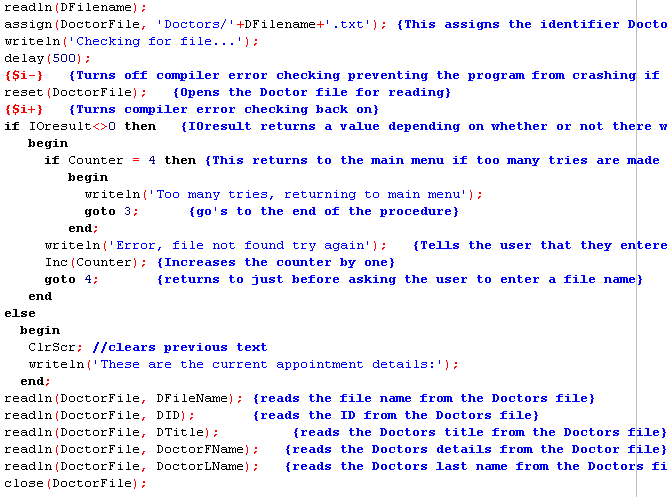
The procedure then prompts the user to enter the time they desire in the format xxxx00 (This is due to naming constraints). Once the user has entered their time it reads it into the time variable and assigns the text variable appointment to the time. It then attempts to create a new file under the desired time, if an error occurs during this process the procedure resets, otherwise it continues. For more information on error checking see the Edit Patient procedure.



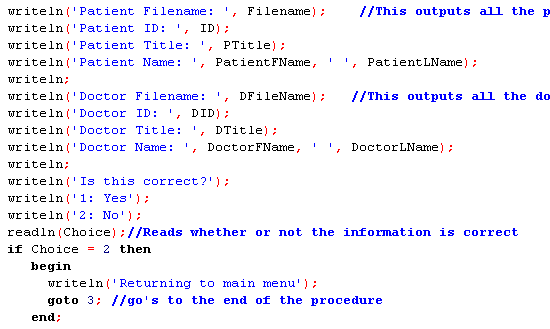
The procedure then asks the user for the filename of the patient who is booking the appointment. The procedure then reads this filename and attempts to reset the file for reading, if the reset fails due to the file not been there the procedure resets, for more information on this error detection see the Edit Patient procedure guide. Once the file has been reset the procedure reads the necessary variables out of the file.



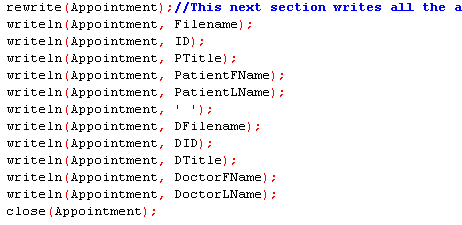
The procedure then asks for the filename of the doctor with whom the appointment is. It then attempts to reset the doctors file for reading, if the file doesn't exist the procedure loops back up to before reading the filename allowing the user to try again. For more information on this process see the Edit Patient procedure guide. If the file exists then the procedure reads the doctors details out of the file into the appropriate variables.



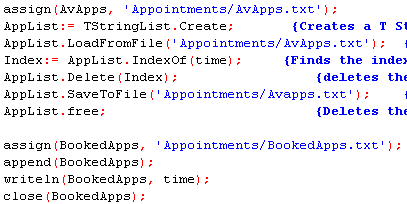
The procedure then writes out all the details of the appointment such as patient details, doctor details and time and then verifies with the user that these details are correct. If the user selects two for no then the procedure skips to the bottom allowing it to exit back to the main menu.



If the user selects option one yes however the procedure then writes all the variables containing the details into the created appointment file.

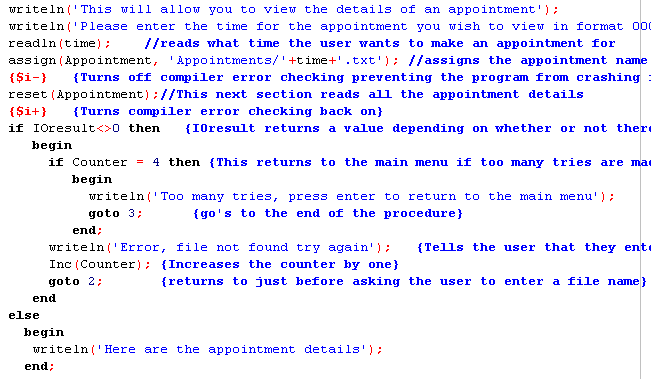


The procedure finishes off by loading the AvApps file into a T String List where it locates the string for the time just booked out and deletes if from the list before writing the list back to the file sans booked time. The procedure then appends the BookedApps file adding the just booked time in.

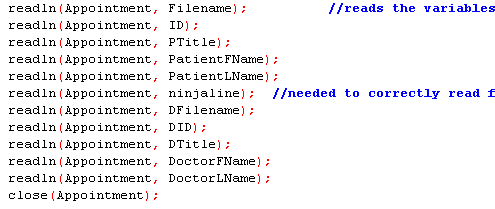


### ViewAppointment

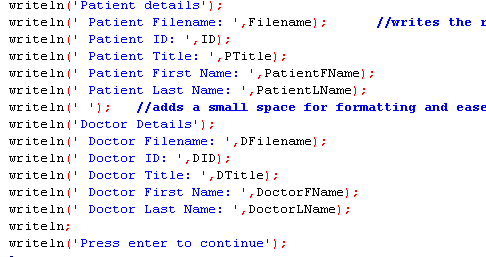
The View Appointment procedure allows a user to view the details for an appointment at a a particular time. The procedure starts by asking the user for the time of the appointment they wish to view, it then assigns the text variable Appointment to the time and attempts to reset the file for reading. If the file doesn't exist then the procedure restarts. For more information on this error catching method see the Edit Patient procedure guide. If the file is there however it is reset for reading and the procedure continues.



The procedure then reads all the appointment variable details out of the file in the order they were put in. The variable ninjaline is there to account for the space in formatting that the file was written with.

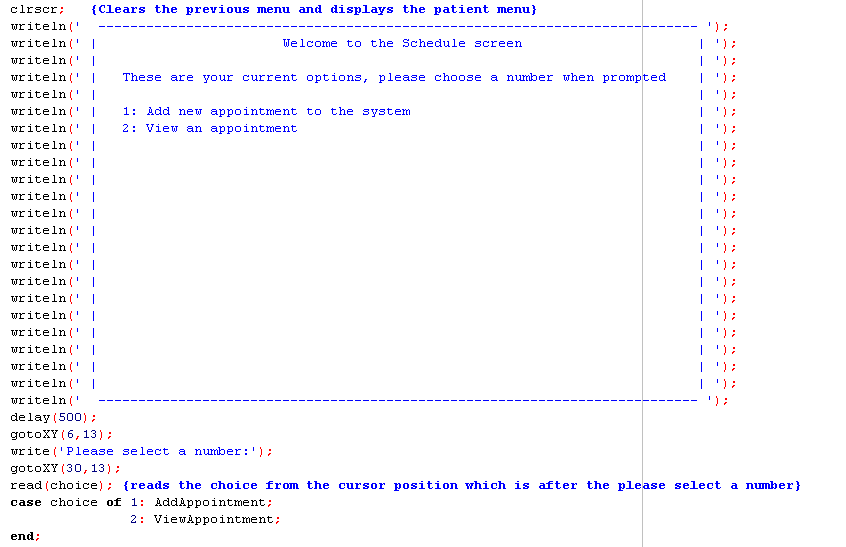


The procedure then writes the variables for the appointment to the terminal with text indicating what each variable is.



### ScheduleMenu

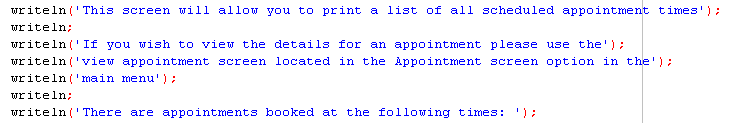
The Schedule Menu procedure simply serves as a gateway to the procedures that make up the function of the Appointment Module. It starts by loading a GUI menu and then asks the user to select a number corresponding to the procedure they wish to load. It then reads this choice into a case and loads the chosen procedure.



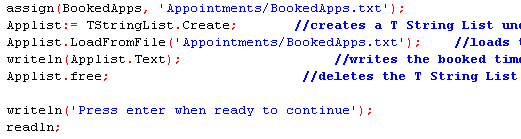
## Print Daily List Module

### PrintDailyList

The Print Daily List procedure serves to print out a list of all the currently booked times for the day. It first displays some text telling the user what the module does and how they can view the details for a particular appointment if they so desire.



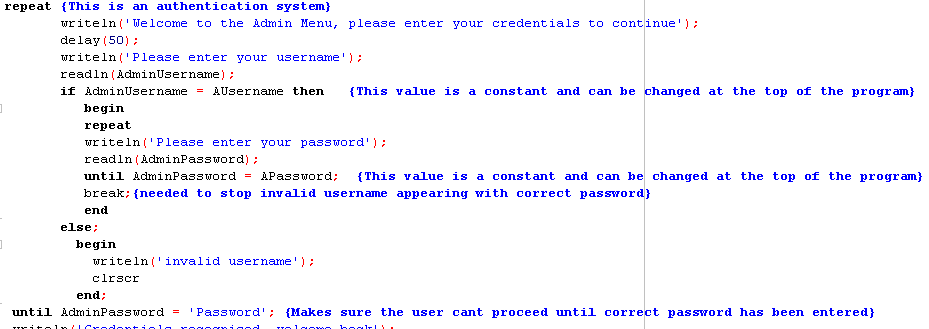
The procedure then creates a T String List and loads all the strings from the BookedApps file into the List. It then outputs all the strings in the list to the terminal before asking the user to press enter to continue.



## Admin Module

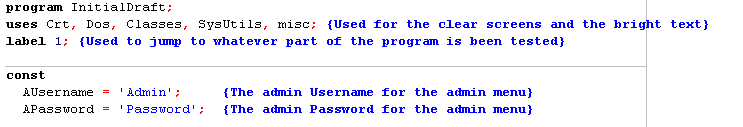
### AdminMenu

The Admin Menu serves as a collection of routines which are useful in the administration of the program by automating several processes. The Admin Menu starts by loading up a log in system where it asks you for a username, if you enter the right username it then asks you for a password, if you enter the right password it admits you to the actual menu.



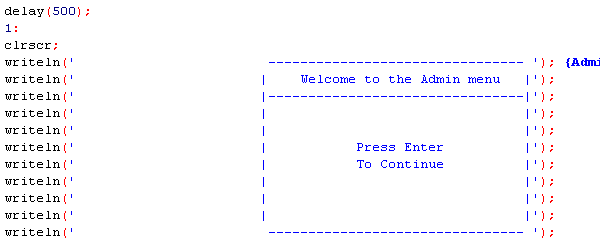
#### Changing the Admin Username and Password

By default the admin username is Admin and the password is Password. However this can be changed very easily within the compiler, simply go to this part of the program:



And change the text in the apostrophes under const next to AUsername and APassword to whatever you wish, just note that the text still has to be enclosed in apostrophes for it to compile and work properly.

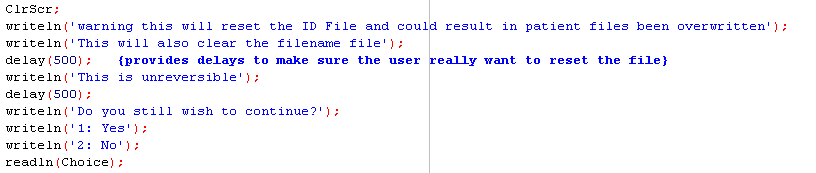
Once you have gotten past the log in system the admin menu loads up.



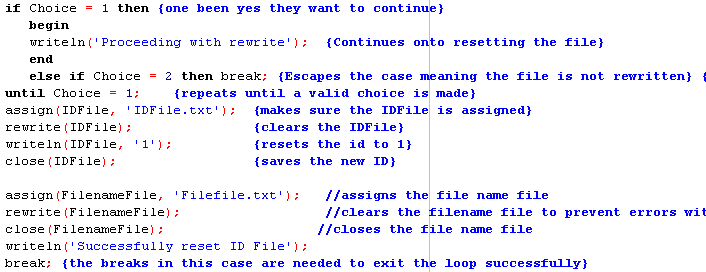
Once you press enter the actual options will load up and the user will be instructed to choose an option based on what they want to do.

#### Reset ID File

If the user chooses to reset the ID File the screen will clear and a warning message will show up telling them that this process is irreversible and could result in patient files been overwritten. It then asks them if they wish to continue.

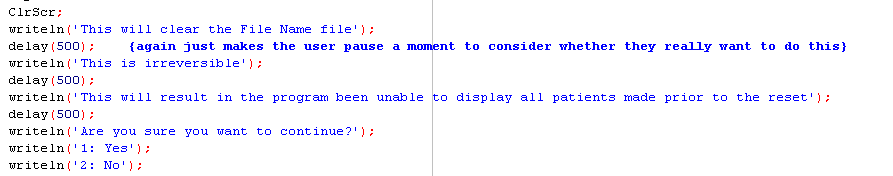


If the user selects two, no they don't want to continue the procedure breaks out of the loops returning to the menu. If they choose one that they do want to continue the procedure then rewrites the ID File and FileFile so that they are both blank and ready to be set up fresh. This procedure should only be used when the program is been completely wiped and set up fresh.

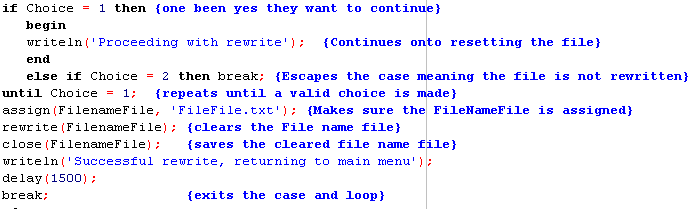


#### Reset Filename File

This procedure resets the filename file only not the ID File, it should only be used if the filename file has been corrupted and needs to be manually cleaned up. Upon loading this procedure a warning message displays warning how this is irreversible and means patient filenames will no longer be able to be listed. It then asks them if they wish to continue.

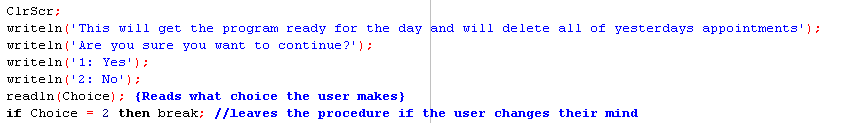


If the user selects no they do not wish to continue the procedure breaks out of the loop and returns to the menu. However if the user chooses to continue then the procedure assigns and rewrites the filename file and returns to the main menu.

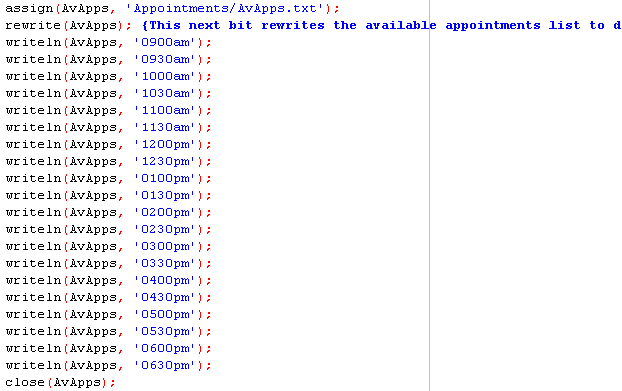


#### Initialize day

The Initialise procedure resets all the appointment related files and gets ready for a new day, it should be used every morning so that no legacy data is left over. When this procedure is loaded the first thing it does is confirm that the user really want to delete all of yesterdays appointments. If they select no it breaks out of the loop and returns to the main menu, if they select yes then it continues on.



Once the procedure has confirmed the user really want to continue with the wipe it first assigns and rewrites the AvApps file. It then writes all the appointment times back to the AvApps file so that it correctly lists the available appointments when booking a new appointment.



The procedure then goes on and assigns each file for each time to the appointment variable and rewrites it so it is blank for the coming day, it does this for all twenty appointment times.

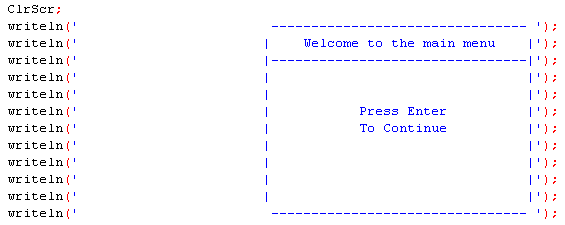


The procedure lastly assigns the BookedApps file to the text variable BookedApps and rewrites it so it is blank and correctly lists the booked appointments.

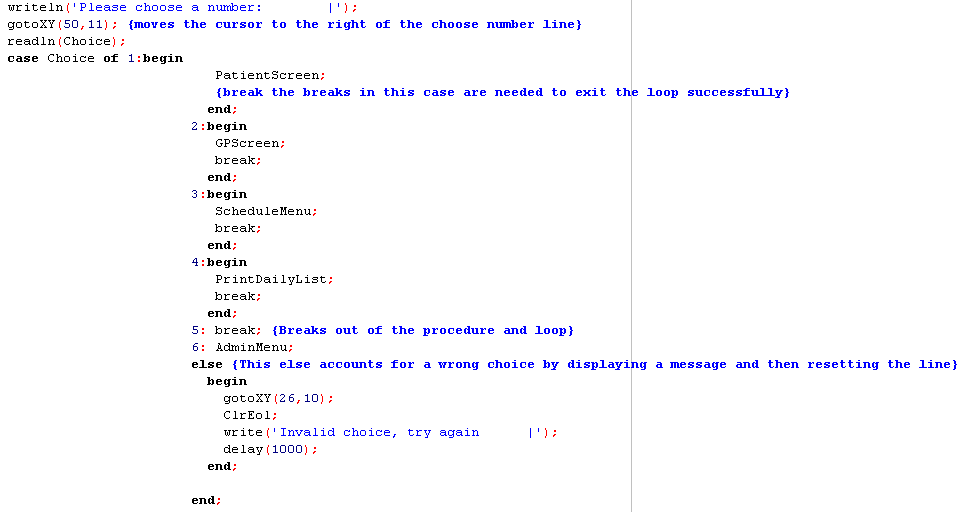


## MainMenu Procedure

The very last procedure is the Main menu procedure and it simply serves as a gateway to each of the five modules. Upon loading it up it displays a blank menu but pressing enter will load up the choices.



The procedure will then prompt the user to select a choice. It will read that choice into a case where it will then load up the chosen procedure.



# Troubleshooting

## Procedure wont compile

There could be any reason for this but the main one is to first look at the error console and see what it says. Most commonly there is probably a missing semicolon or a semicolon where there shouldn't be. The next probable reason is that someone changed the admin username and password but didn't enclose it in apostrophes, so check that.

## My files aren't been created

If the files are not been created it is likely because the name of the folder they are meant to be stored in was changed, compare the name of the folder in the program directory with the name of the folder in the assign lines, they should match up. (HINT: the assign line follows the format FOLDERNAME/FILENAME.txt. The pluses are there to join strings together so leave them there.)

## The program keeps crashing for the users

Make sure the users are correctly entering their choices, i.e if it asks for a number make sure they enter the actual number and not the word i.e 2 vs two. Also tell them to give it a second after they press the number, if the cursor is still flashing then they can press enter, if not it will have automatically read it.

## The program freezes

This is likely due to a stray readln, if the program freezes for longer then a couple of seconds merely press the enter button and it should pick back up again.