

**INDIVIDUAL ASSIGNMENT**

**TECHNOLOGY PARK MALAYSIA**

**INTRODUCTION TO C PROGRAMMING**

**ABIYYU TAJ MAHASIN BAGINDO**

**TP058652**

**APU1F2002CS(IS)**

**HAND OUT DATE: 05 OCTOBER 2020**

**HAND IN DATE: 02 NOVEMBER 2020**

**WEIGHTAGE: 50%**

**INSTRUCTIONS TO CANDIDATES:**

1. Submit your assignment online in Moodle unless advised otherwise
2. Late submission will be awarded zero(0) unless Extenuating Circumstances (EC) are upheld
3. Cases of plagiarism will be penalized
4. You must obtain at least 50% in each component to pass this module

Table of Contents

Introduction and Assumptions ……………………………………………………………………….. 3

Design of the Program ….……………………………………………………………………………...... 3

C Programming Concepts Application ................................................................. 42

Additional Features .............................................................................................. 48

Test Specification Table ....................................................................................... 49

Sample Output .................................................................................................... 56

Conclusion ........................................................................................................... 61

**Introduction and Assumptions**

This program is a management system developed for the ZeeMediLife insurance company to help them register new subscribers, look up subscribers’ data, get statistics, and help subscribers claim insurance. The system is written in C the programming language and utilizes three of C’s built in library. The program is menu-driven, and the code is separated into different modules in the form of functions.

Dividing the program into multiple functions was done because I assumed functions are great tools to build different modules in a program. This is because functions on their own are snippets of code that can be called multiple times throughout the program. Therefore, functions can serve as modules and work well.

**Design of the Program**

The program consists of more than ten functions where each function may call other functions to continue the flow of the program. For example, the main function, which is the function executed on startup, calls the menu function. Which calls other function depending on user input and calls itself when the user enters a wrong input. Below is the pseudocode for the main function after it is the pseudocode for the menu function.



Figure Pseudocode of the main function which calls the menu function.

FUNCTION menu()

INT option

PRINT "ZeeMediLife Health Insurance Management System"

PRINT "------------------------------"

PRINT "1. Insurance Plan Subscription"

PRINT "2. Insurance Claim"

PRINT "3. Accounts Information"

PRINT "4. Search"

PRINT "5. Exit"

PRINT "------------------------------"

PRINT "Choose option: "

READ option

IF option == 1

PRINT "------------------------------"

subReg()

ELSE IF option == 2

PRINT "------------------------------"

insClaim()

ELSE IF option == 3

PRINT "------------------------------"

accInfo()

ELSE IF option == 4

PRINT "------------------------------"

searchFunc()

ELSE IF option == 5

PRINT "------------------------------"

exitMenu()

ELSE

PRINT "------------------------------"

PRINT "Wrong input, please try again"

menu()

ENDIF

ENDFUNCTION

Figure Pseudocode of the function menu

As shown on the pseudocode above, the program is modular as in the program consists of multiple functions where each functions call other functions. Each of these functions have their own purposes and serve as modules to the program. Attached below is the algorithm of the menu function in the form of a flowchart where the function menu itself can be seen clearly calling five other functions depending on user input.

Diagram

Description automatically generated

Figure Flowchart of the menu function.

We will now proceed to the subReg function. A function of the program that handles the registration process of new subscribers. The simplified version of how the subReg function works is it prompts the user for data regarding themselves, such as name and age. It will then display the available plans to the user depending on the user’s age. For example, if the user is 26 years old, they can only subscribe to either Plan150 or Plan200 since Plan120 is only available for those below the age of 21. The function will then gather all the data received by the user and then writes the data down to text files for storage. The function will then call another function called returnToMenu to prompt the user whether they want to go back to the main menu and continue using the program or not.

**subReg() Function Pseudocode**

FUNCTION subReg()

INT claim = 0

INT age, plan, id, nextid, type, claimable

CHAR name[32]

OPENFILE "nextid.txt" AS idgen FOR READING

READ idgen TO id

CLOSEFILE idgen

nextid = id + 1

PRINT "ZeeMediLife Health Insurance Plan"

PRINT "-----------------------------------------------------------------------------------------"

PRINT " | Plan120(RM) | Plan150(RM) | Plan200(RM) "

PRINT "-----------------------------------------------------------------------------------------"

PRINT "Monthly Premium | 120 | 150 | 200 "

PRINT "Annual Claim Limit | 120,000 | 150,000 | 200,000 "

PRINT "Lifetime Claim Limit | 600,000 | 750,000 | 1,000,000 "

PRINT "-----------------------------------------------------------------------------------------"

PRINT "Enter first name: "

READ name

PRINT "Enter age: "

READ age

IF age < 0 OR age > 54

PRINT "We're sorry but we don't have any plans for the entered age, returning to main menu.."

menu()

ENDIF

PRINT "Plan Types:"

PRINT "1. Annual Claim"

PRINT "2. Lifetime Claim"

PRINT "Choose plan type: "

READ type

IF type != 1

IF type != 2

PRINT "Wrong input, going back to main menu.."

menu()

ENDIF

ENDIF

IF 0 < age AND age < 21

IF type == 1

PRINT "Available Plans"

PRINT "-----------------------------------------------------------------------------------------"

PRINT " | Plan120(RM) | Plan150(RM) | Plan200(RM) "

PRINT "-----------------------------------------------------------------------------------------"

PRINT "Monthly Premium | 120 | 150 | 200 "

PRINT "Annual Claim Limit | 120,000 | 150,000 | 200,000 "

PRINT "-----------------------------------------------------------------------------------------"

PRINT "1. RM 120 Plan"

PRINT "2. RM 150 Plan"

PRINT "3. RM 200 Plan"

PRINT "Choose plan: "

READ plan

IF plan < 0 OR 3 < plan

PRINT "Wrong input, returning to main menu.."

menu()

ENDIF

ELSE IF type == 2

PRINT "Available Plans"

PRINT "-----------------------------------------------------------------------------------------"

PRINT " | Plan120(RM) | Plan150(RM) | Plan200(RM) "

PRINT "-----------------------------------------------------------------------------------------"

PRINT "Monthly Premium | 120 | 150 | 200 "

PRINT "Lifetime Claim Limit | 600,000 | 750,000 | 1,000,000 "

PRINT "-----------------------------------------------------------------------------------------"

PRINT "1. RM 120 Plan"

PRINT "2. RM 150 Plan"

PRINT "3. RM 200 Plan"

PRINT "Choose plan: "

READ plan

IF plan < 0 OR 3 < plan

PRINT "Wrong input, returning to main menu.."

menu()

ENDIF

ELSE

PRINT "Wrong input, going back to main menu.."

menu()

ENDIF

ELSE IF 20 < age AND age < 41

IF type == 1

PRINT "Available Plans"

PRINT "-------------------------------------------------------------------"

PRINT " | Plan150(RM) | Plan200(RM) "

PRINT "-------------------------------------------------------------------"

PRINT "Monthly Premium | 150 | 200 "

PRINT "Annual Claim Limit | 150,000 | 200,000 "

PRINT "-------------------------------------------------------------------"

PRINT "1. RM 150 Plan"

PRINT "2. RM 200 Plan"

PRINT "Choose plan: "

READ plan

IF plan == 1

plan = 2

ELSE IF plan == 2

plan = 3

ELSE

PRINT "Wrong input, returning to main menu.."

menu()

ENDIF

ELSE IF type == 2

PRINT "Available Plans"

PRINT "-------------------------------------------------------------------"

PRINT " | Plan150(RM) | Plan200(RM) "

PRINT "-------------------------------------------------------------------"

PRINT "Monthly Premium | 150 | 200 "

PRINT "Annual Claim Limit | 750,000 | 1,000,000 "

PRINT "-------------------------------------------------------------------"

PRINT "1. RM 150 Plan"

PRINT "2. RM 200 Plan"

PRINT "Choose plan: "

READ plan

IF plan == 1

plan = 2

ELSE IF plan == 2

plan = 3

ELSE

PRINT "Wrong input, returning to main menu.."

menu()

ENDIF

ENDIF

ELSE IF 40 < age && age < 55

IF type == 1

PRINT "Available Plans"

PRINT "---------------------------------------------"

PRINT " | Plan200(RM) "

PRINT "---------------------------------------------"

PRINT "Monthly Premium | 200 "

PRINT "Annual Claim Limit | 200,000 "

PRINT "---------------------------------------------"

plan = 3;

ELSE IF type == 2

PRINT "Available Plans"

PRINT "---------------------------------------------"

PRINT " | Plan200(RM) "

PRINT "---------------------------------------------"

PRINT "Monthly Premium | 200 "

PRINT "Annual Claim Limit | 1,000,000 "

PRINT "---------------------------------------------"

plan = 3;

ELSE

PRINT "Wrong input, going back to main menu.."

menu()

ENDIF

ENDIF

IF type == 1

IF plan == 1

claimable = 120000

ELSE IF plan == 2

claimable = 150000

ELSE IF plan == 3

claimable = 200000

ELSE

PRINT "Wrong input, returning to main menu.."

menu()

ENDIF

ELSE IF type == 2

IF plan == 1

claimable = 600000

ELSE IF plan == 2

claimable = 750000

ELSE IF plan == 3

claimable = 1000000

ELSE

PRINT "Wrong input, returning to main menu.."

menu()

ENDIF

ENDIF

OPENFILE age.txt AS fage FOR APPEND

OPENFILE claim.txt AS fclaim FOR APPEND

OPENFILE claimable.txt AS fclaimable FOR APPEND

OPENFILE id.txt AS fid FOR APPEND

OPENFILE name.txt AS fname FOR APPEND

OPENFILE plan.txt AS fplan FOR APPEND

OPENFILE type.txt AS ftype FOR APPEND

OPENFILE overlimit.txt AS foverlimit FOR APPEND

OPENFILE year.txt AS fyear FOR APPEND

PRINT age TO fage

PRINT claim TO fclaim

PRINT claimable TO fclaimable

PRINT nextid TO fid

PRINT name TO fname

PRINT plan TO fplan

PRINT type TO ftype

PRINT "0" TO foverlimit

PRINT "0" TO fyear

CLOSEFILE fage

CLOSEFILE fclaim

CLOSEFILE fclaimable

CLOSEFILE fid

CLOSEFILE fname

CLOSEFILE fplan

CLOSEFILE ftype

CLOSEFILE foverlimit

CLOSEFILE fyear

OPENFILE "nextid.txt" AS idwrite FOR WRITING

PRINT nextid TO idwrite

CLOSEFILE idwrite

returnToMenu()

ENDFUNCTION

**subReg() Function Flowchart**

**Diagram, schematic

Description automatically generated**

Figure Flowchart of the function subReg.

Up next is the function insClaim. The function insClaim is used by the program to handle insurance claims. When a registered subscriber wants to claim insurance, they can simply run the program, type the number 2, and press enter. The program will then run the function insClaim to process the subscriber’s insurance claim. How the function works is the function will first prompt the user to enter their ID, which will then be used as reference when reading text files. Multiple text files that contain the subscriber’s data will then be opened and read. The line which is related to the ID entered earlier will then be assigned to variables since those lines contain the subscriber’s data. The function will then prompt the user to enter their charges such as room charges, surgical charges, etc. The program will then calculate how much insurance the subscriber will be able to claim based on their insurance plan, and then their balance will be deducted and the data will be written back to the original text files for storage. The function will then end the same way as the subReg function that is the function will call the function returnToMenu to prompt the user whether they want to go back to the main menu or not. Below are the pseudocode and flowchart for the function insClaim.

**insClaim() Function Pseudocode**

FUNCTION insClaim()

INT x, id, idlimit, limit, count

INT lineNumber

CHAR name[32]

INT claim, claimable, type, plan, year, overlimit

PRINT "Enter ID"

READ x

lineNumber = x + 1

OPENFILE "nextid" AS fidlimit FOR READING

READ fidlimit TO idlimit

limit = idlimit

IF x < 0 OR x > idlimit

PRINT "ID not found, returning to main menu..

menu()

ENDIF

CHAR cLine[32]

int iLine

count = 0

OPENFILE "name.txt" AS fname FOR READING

WHILE count < limit

READ fname TO cLine

IF count == lineNumber

name = cLine

ENDIF

count++

ENDWHILE

CLOSEFILE fname

PRINT "Welcome, " + name

count = 0

INT claimHolder[limit]

OPENFILE "claim.txt" AS fclaim FOR READING

WHILE count < limit

READ fclaim TO iLine

IF count == lineNumber

claim = iLine

ENDIF

claimHolder[count] = iLine

count++

ENDWHILE

CLOSEFILE fclaim

count = 0

INT claimableHolder[limit]

OPENFILE "claimable.txt" AS fclaimable FOR READING

WHILE count < limit

READ fclaimable TO iLine

IF count == lineNumber

claimable = iLine

ENDIF

claimableHolder[count] = iLine

count++

ENDWHILE

CLOSEFILE fclaimable

count = 0

INT overlimitHolder[limit]

OPENFILE "overlimit.txt" AS foverlimit FOR READING

WHILE count < limit

READ foverlimit TO iLine

IF count == lineNumber

overlimit = iLine

ENDIF

overlimitHolder[count] = iLine

count++

ENDWHILE

CLOSEFILE foverlimit

count = 0

OPENFILE "type.txt" AS ftype FOR READING

WHILE count < limit

READ ftype TO iLine

IF count == lineNumber

type = iLine

ENDIF

count++

ENDWHILE

CLOSEFILE ftype

count = 0

OPENFILE "plan.txt" AS fplan FOR READING

WHILE count < limit

READ fplan TO iLine

IF count == lineNumber

plan = iLine

ENDIF

count++

ENDWHILE

CLOSEFILE fplan

count = 0

INT yearHolder[limit]

OPENFILE "year.txt" AS fyear FOR READING

WHILE count < limit

READ fyear TO iLine

IF count == lineNumber

year = iLine

ENDIF

yearHolder[count] = iLine

count++

ENDWHILE

CLOSEFILE fyear

INT roomDays, icuDays, roomCharge, icuCharge

PRINT "Enter how many days spent in hospital room: "

READ roomDays

PRINT "Enter how many days spent in ICU: "

READ icuDays

IF roomDays < 0 OR icuDays < 0

PRINT "Days spent cannot be less than 0, return to main menu.."

menu()

ENDIF

IF plan == 1

roomCharge = roomDays \* 120

icuCharge = icuDays \* 250

ELSE IF plan == 2

roomCharge = roomDays \* 150

icuCharge = icuDays \* 400

ELSE IF plan == 3

roomCharge = roomDays \* 200

icuCharge = icuDays \* 700

ENDIF

INT hSupply, hSurgical, hOther

PRINT "Enter amount of hospital supplies and service charges: "

READ hSupply

PRINT "Enter amount of surgical charges: "

READ hSurgical

PRINT "Enter amount of other charges: "

READ hOther

INT sum = 0, outstand, printOutstand

sum = roomCharge + icuCharge + hSupply + hSurgical + hOther

outstand = claimable - sum

printOutstand = 0 - outstand

IF outstand < 0

PRINT "You have surpassed your claim limit and have RM " + printOutstand + " outstanding fees"

overlimit = 1

ENDIF

PRINT "---------------------"

PRINT "Subscriber Details"

PRINT "ID: " + x

PRINT "Name: " + name

PRINT "Year Claimed: 1"

PRINT "Insurance Claimed " + sum

PRINT "Balance Left: " + outstand

claimableHolder[lineNumber] = outstand

OPENFILE "claimable.txt" AS claimableWrite FOR WRITING

PRINT claimableHolder[0] TO claimableWrite

CLOSEFILE claimableWrite

OPENFILE "claimable.txt" AS claimableAppend FOR APPEND

count = 1

WHILE count < limit

PRINT claimableHolder[count] TO claimableAppend

count++

ENDWHILE

CLOSEFILE claimableAppend

claimHolder[lineNumber] = sum

OPENFILE "claim.txt" AS claimWrite FOR WRITING

PRINT claimHolder[0] TO claimWrite

CLOSEFILE claimWrite

OPENFILE "claim.txt" AS claimAppend FOR APPEND

count = 1

WHILE count < limit

PRINT claimHolder[count] TO claimAppend

count++

ENDWHILE

CLOSEFILE claimAppend

yearHolder[lineNumber] = 1

OPENFILE "year.txt" AS yearWrite FOR WRITING

PRINT yearHolder[0] TO yearWrite

CLOSEFILE yearWrite

OPENFILE "year.txt" AS yearAppend FOR APPEND

count = 1

WHILE count < limit

PRINT yearHolder[count] TO yearAppend

count++

ENDWHILE

CLOSEFILE yearAppend

IF type == 1 AND overlimit == 1

overlimitHolder[lineNumber] = 1

OPENFILE "overlimit.txt" AS overlimitWrite FOR WRITING

PRINT overlimitHolder[0] TO yearWrite

CLOSEFILE overlimitWrite

OPENFILE "overlimit.txt" AS overlimitAppend FOR APPEND

count = 1

WHILE count < limit

PRINT overlimitHolder[count] TO overlimitAppend

count++

ENDWHILE

CLOSEFILE overlimitAppend

ENDIF

INT nextclaim

IF type == 2

OPENFILE "lifeclaim.txt" AS lifeclaimAppend FOR APPEND

PRINT sum TO lifeclaimAppend

CLOSEFILE lifeclaimAppend

OPENFILE "nextclaim.txt" AS fnextclaim FOR READING

READ fnextclaim TO nextclaim

nextclaim = nextclaim + 1

CLOSEFILE fnextclaim

OPENFILE "nextclaim.txt" AS fnextclaimwrite FOR WRITING

PRINT nextclaim TO fnextclaimwrite

CLOSEFILE fnextclaimwrite

ENDIF

returnToMenu()

ENDFUNCTION

**insClaim() Function Flowchart**

**Diagram, schematic

Description automatically generated**

Figure Flowchart of the function insClaim.

The next function is not as complex as the previous one. The next function is called the accInfo function, which is separated into two different parts: accInfoA and accInfoB. The function accInfo does not do much since all it does is just call the function accInfoA, accInfoB, then returnToMEnu just like the other functions. This means the data processing takes place within the two subfunctions rather than in the function accInfo itself.

The accInfoA function’s purpose is to gather the total amount of insurance claimed by Lifetime Claim Limit subscribers in Ringgit Malaysia (MYR). How the function works is it will read the text file nextclaim.txt which contains the total amount of claims that have been made by Lifetime Claim Limit subscribers. This value will then be kept inside the variable limit. The variable limit is used as a limiter in a while loop that reads the text file lifeclaim.txt line by line which contains all the claims that have been made by Lifetime Claim Limit subscribers. All these lines are added to the variable called sum. The value of the variable sum will then be concatenated into a string and printed to the user.

The function accInfoB is also a statistical function as in the function does not do a lot of processing like the previous functions but it is simply used to read and print out data. In this case, it is the amount of Annual Claim Limit subscribers who have exhausted their balance. How it works is it will first read the text file nextid.txt which contains the total amount of subscribers, this value will then be used as a limiter for a loop just like in the function accInfoA. The loop reads the text file overlimit.txt which contains the data that indicates whether an Annual Claim Limit subscriber has exhausted their balance or not. The loop will sum everything up and then concatenate the sum into a string and print it out for the user to read. Below are the pseudocode and flowcharts for the three functions.

**accInfo() Function Pseudocode**

FUNCTION accInfo

accInfoA()

accInfoB()

returnToMenu()

ENDFUNCTION

**accInfo() Function Flowchart**

**A picture containing table, clock

Description automatically generated**

Figure Flowchart of the function accInfo.

**accInfoA() Function Pseudocode**

FUNCTION accInfoA()

INT count, sum = 0, line, limit

OPENFILE "nextclaim.txt" AS nextlimit FOR READING

READ nextlimit TO limit

CLOSEFILE nextlimit

OPENFILE "lifeclaim.txt" AS flifeclaim FOR READING

count = 0

WHILE count < limit

READ flifeclaim TO line

sum = sum + line

count++

ENDWHILE

CLOSEFILE flifeclaim

PRINT "Total amount claimed by Lifetime Claim Limit Subscribers: RM " + sum

ENDFUNCTION

**accInfoA() Function Flowchart**

**Diagram

Description automatically generated**

Figure Flowchart of the function accInfoA.

**accInfoB() Function Pseudocode**

FUNCTION accInfoB

INT count, sum = 0, line, limit

OPENFILE "nextid.txt" AS nextlimit FOR READING

READ nextlimit TO limit

CLOSEFILE nextlimit

OPENFILE "overlimit.txt" AS foverlimit FOR READING

count = 0

WHILE count < limit

READ foverlimit TO line

sum = sum + line

count++

ENDWHILE

CLOSEFILE foverlimit

PRINT "Total number of Annual Claim Limit subscribers who have exhausted all their eligible amount: " + sum

ENDFUNCTION

**accInfoB() Function Flowchart**

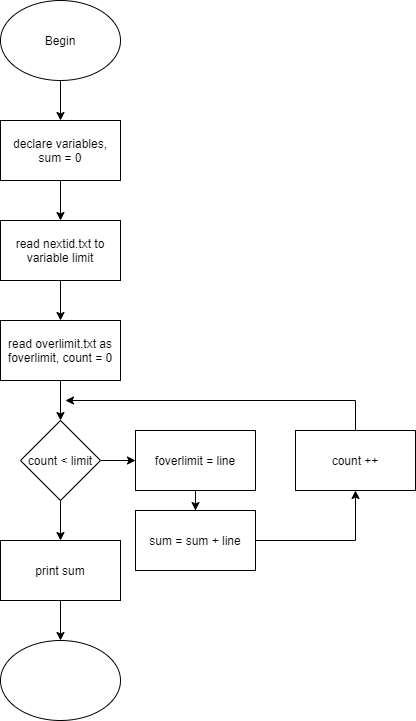


Figure Flowchart of the function accInfoB. (For some reason, the end lable is missing)

Moving on and we have the last feature of the program which is the search feature. The search feature is supported by the function searchFunc. And just like the previous function accInfo, this function also has subfunctions. This function however has five subfunctions. The searchFunc function allows the user to choose one out of five available search criteria to search for one or more patients depending on the criteria chosen. The available criteria are ID, name, claim limit type, plan, and age. The function will prompt the user to choose one of the five and call one of the five subfunctions depending on user input. If the user gives an invalid input, the function will know and return the user to the main menu. After running the one of the subfunctions, the function will call the returnToMenu function as per usual to prompt the user whether they want to go back to the main menu or not.

How it works is the function will prompt the user to choose one from the five available options, and then call a function depending on the user input. The searchFunc function itself has nothing special in it since just like the accInfo function, the process lies within the subfunctions called by the searchFunc function, and the actualSearch function which is called by searchFunc’s subfunctions.

**searchFunc() Function Pseudocode**

FUNCTION searchFunc

INT searchCriteria

PRINT "Available Search Criteria"

PRINT "1. Search by ID"

PRINT "2. Search by name"

PRINT "3. Search by plan"

PRINT "4. Search by claim limit type"

PRINT "5. Search by age"

PRINT "Choose option: "

READ searchCriteria

IF searchCriteria == 1

PRINT "------------------------------"

searchID()

returnToMenu()

ELSE IF searchCriteria == 2

PRINT "------------------------------"

searchName()

returnToMenu()

ELSE IF searchCriteria == 3

PRINT "------------------------------"

searchPlan()

returnToMenu()

ELSE IF searchCriteria == 4

PRINT "------------------------------"

searchType()

returnToMenu()

ELSE IF searchCriteria == 5

PRINT "------------------------------"

searchAge()

returnToMenu()

ELSE

PRINT "Wrong input, returning to menu.."

menu()

ENDIF

ENDFUNCTION

The subfunctions share the same algorithm with each other with the only difference being the text file they use to look for the data and searchID being the simplest among the five subfunctions. In this example, we will use the function searchAge which is used to search for subscribers of a specific age.

How it works is, the function will first declare variables and initialize the variable idIndex as 0. The program will then prompt the user to enter an age between 0 and 55, this input will be kept inside the variable age. Then, just like some of the previous functions, read the nextid.txt file to get the amount of subscribers registered to be used as a loop limiter. The variable used to store this data is called limit. The function will then initialize count as 0 and start looping until count is no longer lesser than limit with each loop incrementing the value of the variable count by 0. During each loop, the loop will read the file age.txt line by line and assign the value of those lines to iLine. The value of iLine will then be compared to the variable age which contains the age the user inputted earlier. If it matches, The program will assign the value of count + 1 to a variable called idCatch, and then assign the value of the variable idCatch to the a specific index of the array idHolder where this index is the value of variable idIndex. The variable idIndex will then be incremented and so does count and the loop starts again. When the loop closes, the text file is closed, and another process will start.

This time, the program will initialize the variable idCount as 0. A loop will then start until idCount is no longer lesser than idIndex. Within that loop, the function will call another function called actualSearch with the index idCount of the array idHolder being sent as arguments. After that the variable idCount will be incremented and the loop repeats. When the loop ends, the function will call the returnToMenu function to continue the flow of the program. There are other functions similar to this and those functions are searchID, searchName, searchType, and searchPlan. Those functions are very similar to this function on a technical standpoint.

**searchAge() Function Pseudocode**

FUNCTION searchAge()

INT count, idCount, limit, idCatch, idIndex = 0, age, iLine

INT idHolder[32]

PRINT "Enter age: "

READ age

IF age < 0 OR age > 55

PRINT "Wrong input, returning to main menu.."

menu()

ENDIF

OPENFILE "nextid.txt" AS fidlimit FOR READING

READ fidlimit TO limit

CLOSEFILE fidlimit

count = 0

OPENFILE "age.txt" AS fage FOR READING

WHILE count < limit

READ fage TO iLine

IF iLine == age

idCatch = count + 1

idHolder[idIndex] = idCatch

idIndex++

ENDIF

count++

ENDWHILE

CLOSEFILE fage

idCount = 0

WHILE idCount < idIndex

actualSearch(idHolder[idCount])

idCount++

ENDWHILE

returnToMenu()

ENDFUNCTION

**searchAge() Function Flowchart**

Diagram

Description automatically generated

Figure Flowchart of the function searchAge.

Up next is the function actualSearch, which is the function that handles the actual searching processes. This function is called by the five subfunctions of the searchFunc function such as the searchAge function from the previous paragraph. The actualSearch function is the only function in the whole program that takes in an argument. This argument is the line numbers of the matching data in the five subfunctions.

The function uses this argument as reference so the function knows which line they should get when reading a text file. This process is repeated for all the data needed. The function will then print out all the data for the user to see, and then the function ends.

**actualSearch(int x) Function Pseudocode**

FUNCTION actualSearch(INT x)

INT count, limit, line, lineNumber

CHAR name[32]

lineNumber = x - 1

OPENFILE "nextid.txt" AS fidlimit FOR READING

READ fidlimit TO limit

CLOSEFILE fidlimit

CHAR cLine[32]

INT iLine

INT year, plan, type, claimable, age

count = 0

OPENFILE "name.txt" AS fname FOR READING

WHILE count < limit

READ fname TO cLine

IF count == lineNumber

name = cLine

ENDIF

count++

ENDWHILE

CLOSEFILE fname

count = 0

OPENFILE "claimable.txt" AS fclaimable FOR READING

WHILE count < limit

READ fclaimable TO iLine

IF count == lineNumber

claimable = iLine

ENDIF

count++

ENDWHILE

CLOSEFILE fclaimable

count = 0

OPENFILE "type.txt" AS ftype FOR READING

WHILE count < limit

READ ftype TO iLine

IF count == lineNumber

type = iLine

ENDIF

count++

ENDWHILE

CLOSEFILE ftype

count = 0

OPENFILE "plan.txt" AS fplan FOR READING

WHILE count < limit

READ fplan TO iLine

IF count == lineNumber

plan = iLine

ENDIF

count++

ENDWHILE

CLOSEFILE fplan

count = 0

OPENFILE "year.txt" AS fyear FOR READING

WHILE count < limit

READ fyear TO iLine

IF count == lineNumber

year = iLine

ENDIF

count++

ENDWHILE

CLOSEFILE fyear

count = 0

OPENFILE "age.txt" AS fage FOR READING

WHILE count < limit

READ fage TO iLine

IF count == lineNumber

age = iLine

ENDIF

count++

ENDWHILE

CLOSEFILE fage

PRINT "------------------------------"

PRINT "Subscriber Record"

PRINT "ID: " + x

PRINT "Name: " + name

PRINT "Age: " + age

IF type == 1

PRINT "Claim Limit Type: Annual"

ELSE IF type == 2

PRINT "Claim Limit Type: Lifetime"

ENDIF

IF plan == 1

PRINT "Plan: Plan120"

ELSE IF plan == 2

PRINT "Plan: Plan150"

ELSE IF plan == 3

PRINT "Plan: Plan200"

ENDIF

PRINT "Year(s) Claimed: " + year

PRINT "Balance left: " + claimable

PRINT "------------------------------"

ENDFUNCTION

**actualSearch(int x) Function Flowchart**

**Diagram

Description automatically generated**

Figure Flowchart of the function actualSearch.

The last functions are not process functions, but they are still important to the flow of the program. These two functions are the returnToMenu function which has seen multiple use throughout the program and the function exitMenu. The function returnToMenu prompts the user whether they want to go back to the main menu or not and calls a function depending on user input. If the user inputs a 1, the menu function will be called. If the user inputs a 2, the exitMenu function will be called. If any other input is entered the function will print an error message and calls itself. The exitMenu function has nothing inside it. It is simply an integer function that returns 0 to the system when called, stopping the program.

**returnToMenu() Function Pseudocode**

FUNCTION returnToMenu()

int navHolder

PRINT "------------------------------"

PRINT "Return to menu?"

PRINT "1. Yes"

PRINT "2. No"

PRINT "Choose option: "

READ navHolder

IF navHolder == 1

menu()

ELSE IF navHolder == 2

exitMenu()

ELSE

PRINT "Wrong input, please try again"

returnToMenu

ENDIF

ENDFUNCTION

**returnToMenu() Function Flowchart**

**Diagram

Description automatically generated**

Figure Flowchart of the function returnToMenu.

**C Programming Concepts Applications**

The programming language of C has multiple programming concepts. These concepts can range from things as simple as variables used to store data, or functions used for the same algorithm to be used repeatedly. These concepts are of course used and applied in this program. In this chapter, we will explain what these programming concepts are, and how they are applied in this program.

**Variables**

Variables are a vital part of a program. Variables are a main way to store data temporarily in a program. Without variables, a program can not do processes and calculations as easily as they can now. Below is one line from the program’s code that shows how a variable can be used in calculations and store data.



Figure The variable id is added by one and the sum is stored in another variable called nextid.

**Reading and Writing**

Reading and writing is a fundamental part of a program. In fact, most programmers learn programming by writing the famous phrase “Hello World” typically using the programming language’s print function or its iteration of it. In C, reading and writing can be done using the functions printf and scanf, respectively. Below is a code snippet from the program’s source code that shows the program writing a line and reading input from the user.



Figure The program printing "Choose option: " and storing the user input in a variable called option.

**Assignment of Values**

After declared, a variable can be *initialized* assigning it a value. This is because, before initialized, a variable will have the value of NULL. Which is technically 0 as in there is nothing there, not the integer 0. A variable does not have to be initialized before it can be used, however.



Figure The variable overlimit being assigned with the value 1.

**Comments**

Comments do not affect the program in anyway. In fact, comments are used to prevent specific lines of code from being compiled. This is because comments are not meant for the compiler, but for humans. Comments are essentially sticky notes left by a programmer throughout the code. They are labels that can be used to explain a section of a program to someone who is not familiar with the program at all. In C, they are written by adding two forward slashes (//) to the start of a line.



Figure Two lines of comments used to explain a part of the program's code

**Control Structures**

* **If Else Statements**

An if else statement is a very basic form of a control structure. They control the flow of a program through checking for conditions. For example, “***If*** *price of vegetable is acceptable* ***then*** *buy vegetable.* ***Else*** *haggle for a lower price.”* This is a very simple and valid example of an if else statement since if else statements work the exact same way in programming as well.



Figure An if else statement used in the program's source code.

* **While Loops**

A while loop is a type of loops that will continue looping until the condition assigned is no longer met. For example, “***While*** *there is homework,* ***Then,*** *do homework.”* This is a real-life example of a while loop. This can be applied to programming however something that modifies the amount of homework has to be added unless an infinite loop is desired.



Figure A while loop used in the program's source code.

**Arrays**

If variables can be visualized as a drawer that contains values, then an array can be visualized as a collection of drawers that can store more values. An array can be used the same way as a variable however a variable cannot be used the same way as an array. This is because array has what is known as an *index*. For example, the variable x may contain the value 7 in the statement “x = 7”. The array y[3] however can contain the value 8,9, and 10 in the statement “y[3] = {8,9,10}”. The number 3 represents how many indexes does the array y have and this number can be swapped when using the array to use a specific value that belongs to the array y[3]. For example, when y[0] is printed, the console will display the number 8. This is because array indexes start from 0. So according to this, y[0] will have a value of 8, y[1] will have a value of 9, and y[2] will have a value of 10. Arrays are used in the program to store similar subscribers’ data. For example, the array nameHandler contains the names of subscribers.



Figure The array idHolder is having its specific index assigned with a value.

**Library Functions**

Functions are a vital part of a lot of programs. Functions make writing code easier since practically they are variables on a greater scale. If a variable allows you reuse the same value multiple times, a function allows you to use any length of code multiple times. A simple and basic form of a function is the print function which takes in a string and displays it on the CLI. This function belongs to the stdio.h header file, which belongs to the *C Standard Library*. This library contains predefined functions that can be called by simply including the header file that contains it. For example, this program uses the strcmp function and the strcpy function. Both functions belong to the string.h header file.



Figure Header files being included at the start of the program.



Figure The function strcmp which belongs to the string.h header file being used.

**Programmer Defined Functions**

Programmer defined functions are fundamentally the same as built-in standard library functions. They are both functions that can be called multiple times throughout the program. The only difference is programmer defined functions are custom functions that a programmer defined to suit their needs. If a built-in function only needs its corresponding header file to be included to be used, a programmer defined function must be defined by the programmer manually. This program contains more than ten programmer defined functions that are used throughout the program. One of the simples one is the function returnToMenu. Which is basically a function that prompts the user whether they want to go back to the main menu by calling the menu function, or quit the program by calling the function exitMenu which gives a return of value 0 to the program, stopping the program.



Figure Function prototypes of the multiple functions of the program which basically serves as a declaration for functions.



Figure returnToMenu function definition.



Figure The function returnToMenu called in an if statement

**File**

Text files are the most basic form of permanent data storage. When a data is stored in a variable, that data is temporary. Once the program stops running, that data that was stored within the variable will be lost. This is where text files come in. Data stored in text files are kept permanently unless the text file itself is edited or the physical storage itself is tampered with. Data can be read from text files and stored in a variable for processing and that edited data can then be written back to the same or different text file to keep the updated data permanently. Text file storage is one of the fundamentals of this program. Since all subscribers’ data are stored in text files, the program reads and writes data from and to the multiple text files used to store subscriber data.



Figure The contents of the text file "nextid.txt" is read and assigned as a value to the variable limit.

**Additional Features**

The most common type of additional features added to the program is input checks. For example, when the program prompts the user to enter their age during registration, if the user inputs a number lower than 0 or higher than 54, the program will print out a warning that says there are no plans for people with that age and returns the user to the main menu. This is only one example of the many input checks added to the program. Below are some examples of the input checks available in the program.



Figure An if else statement in the menu function will send a user back to the main menu if they enter anything that is not an integer in the range of 1 - 5.



Figure An if statement that checks for the user age that was mentioned in the previous paragraph checking the user's age.



Figure An if statement checking whether the input the user entered is valid or not.

**Test Specification Table**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| No. | Function | Input | Expectation | Result | Action | Final Result |
| 1 | menu() | 1 | Function subReg called | Function subReg called | Not required | Pass |
| 2 | menu() | 2 | Function insClaim called | Function insClaim called | Not required | Pass |
| 3 | menu() | 3 | Function accInfo called | Function accInfo called | Not required | Pass |
| 4 | menu() | 4 | Function searchFunc called | Function searchFunc called | Not required | Pass |
| 5 | menu() | 5 | Function exitMenu called | Function exitMenu called | Not required | Pass |
| 6 | menu() | 6 | Function menu called | Function menu called | Not required | Pass |
| 7 | subReg() | Pog | Function reads “Pog” as a string an assign it to the variable name | Function reads “Pog” as a string an assign it to the variable name | Not required | Pass |
| 8 | insClaim() | 20 | Function finds out the input is greater than the number of subscribers therefore that ID does not exist and returns user to the main menu | Function finds out the input is greater than the number of subscribers therefore that ID does not exist and returns user to the main menu | Not required | Pass |
| 9 | accInfo() | None | The function takes in no input and calls the functions accInfoA and accInfoB | The function takes in no input and calls the functions accInfoA and accInfoB | Not required | Pass |
| 10 | searchFunc() | 1 | The function calls the function searchID and returnToMenu | The function calls the function searchID and returnToMenu | Not required | Pass |
| 11 | searchFunc() | 2 | The function calls the function searchName and returnToMenu | The function calls the function searchName and returnToMenu | Not required | Pass |
| 12 | searchFunc() | 3 | The function calls the function searchPlan and returnToMenu | The function calls the function searchPlan and returnToMenu | Not required | Pass |
| 13 | searchFunc() | 4 | The function calls the function searchType and returnToMenu | The function calls the function searchType and returnToMenu | Not required | Pass |
| 14 | searchFunc() | 5 | The function calls the function searchAge and returnToMenu | The function calls the function searchAge and returnToMenu | Not required | Pass |
| 15 | searchFunc() | 6 | The function calls the function menu | The function calls the function menu | Not required | Pass |
| 16 | accInfoA() | None | The function gathers the total amount of insurance claimed by all Lifetime Claim Limit subscribers and displays the sum | The function gathers the total amount of insurance claimed by all Lifetime Claim Limit subscribers and displays the sum | Not required | Pass |
| 17 | accInfoB() | None | The function gathers the total count of Annual Claim Limit subscribers and who have exhausted their balance and displays the sum | The function gathers the total count of Annual Claim Limit subscribers and who have exhausted their balance and displays the sum | Not required | Pass |
| 18 | actualSearch(int x) | None | The function receives an argument x and read and display data that corresponds to the argument received | The function receives an argument x and read and display data that corresponds to the argument received | Not required | Pass |
| 19 | searchID() | 18 | The function reads the input and passes it to the function actualSearch as an argument. The records belonging to subscriber of ID 18 is then printed out | The function reads the input and passes it to the function actualSearch as an argument. The records belonging to subscriber of ID 18 is then printed out | Not required | Pass |
|  | searchID() | 20 | The function reads the input and passes it to the function actualSearch as an argument. However, nothing is printed out since there is no subscriber of ID 20. | The function reads the input and passes it to the function actualSearch as an argument. However, nothing is printed out since there is no subscriber of ID 20. | Not required | Pass |
| 20 | searchName() | Coco | The function reads the input and passes it to the function actualSearch as an argument. The records belonging to subscriber of name Coco is then printed out | The function reads the input and passes it to the function actualSearch as an argument. The records belonging to subscriber of name Coco is then printed out | Not required | Pass |
| 21 | searchPlan() | 1 | The function reads the input and passes it to the function actualSearch as an argument. The records belonging to subscribers of Plan120 is then printed out | The function reads the input and passes it to the function actualSearch as an argument. The records belonging to subscribers of Plan120 is then printed out | Not required | Pass |
| 22 | searchType() | 2 | The function reads the input and passes it to the function actualSearch as an argument. The records belonging to subscribers of the Lifetime Claim Limit is then printed out | The function reads the input and passes it to the function actualSearch as an argument. The records belonging to subscribers of the Lifetime Claim Limit is then printed out | Not required | Pass |
| 23 | searchAge() | 14 | The function reads the input and passes it to the function actualSearch as an argument. The records belonging to subscribers of age 14 is then printed out | The function reads the input and passes it to the function actualSearch as an argument. The records belonging to subscribers of age 14 is then printed out | Not required | Pass |
| 24 | returnToMenu() | 1 | The function calls the function menu | The function calls the function menu | Not required | Pass |
| 25 | returnToMenu() | 2 | The function calls the function exitMenu | The function calls the function exitMenu | Not required | Pass |
| 26 | returnToMenu() | 3 | The function calls itself | The function calls itself | Not required | Pass |
| 27 | exitMenu() | None | The function returns a value of 0 and stops the program | The function returns a value of 0 and stops the program | Not required | Pass |

**Sample Output**

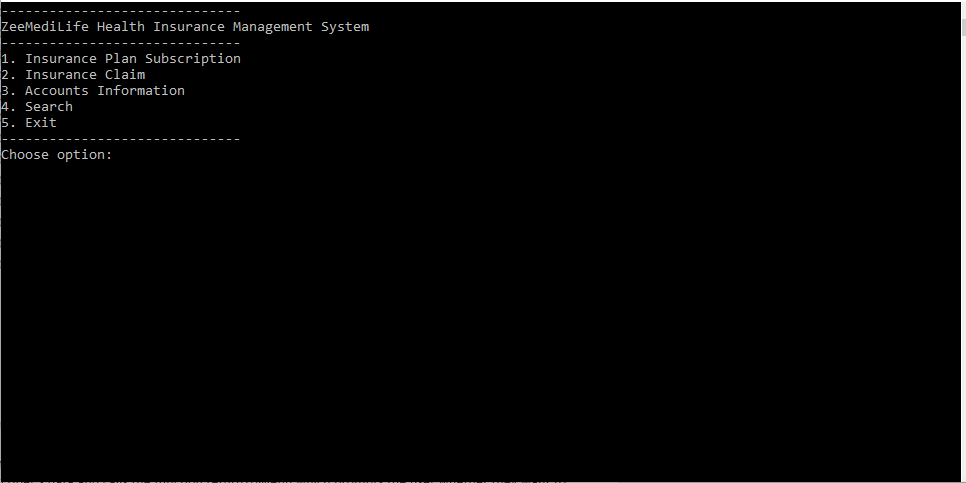


Figure The program's main menu.

The program starts by executing the main function, which calls the menu function which displays the main menu to the user. The user can then choose which one of the five options they would like to pick.

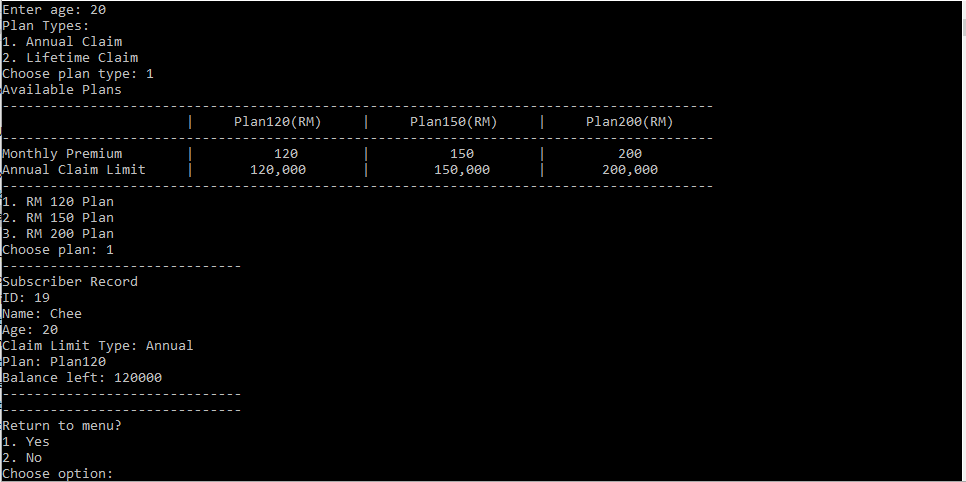


Figure A new subscriber subscribing to Plan120.

As shown in the screenshot above, the program function starts with prompting the user to enter their data. The program then prompts the user to enter what type of claim limit the user would like to be subscribed to, the program then checks available plans for the user’s age, and in this case the user’s age is 20 therefore all plans are printed on the CLI. The program then prompts the user to choose one from the three plans. Once the user enters all the relevant data, the program then displays the subscriber’s data and calls the function returnToMenu which prompts the user whether they want to continue or not.

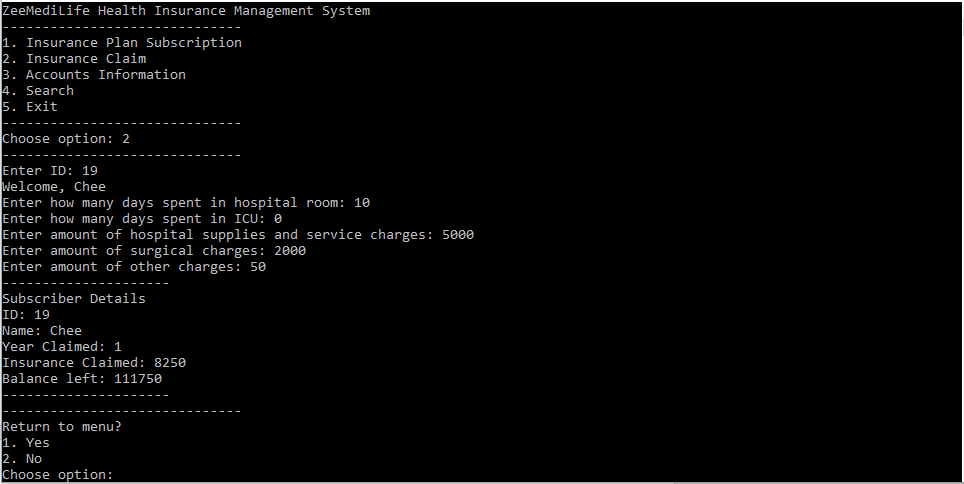


Figure Subscriber Chee claiming insurance.

In the screenshot above, the previous newly registered subscriber by the name of Chee is claiming for insurance. The program prompts the user to enter the charges they had, calculated everything, and deducted it from their balance. Once all that is done, the program displays the subscriber’s data and calls the function returnToMenu.

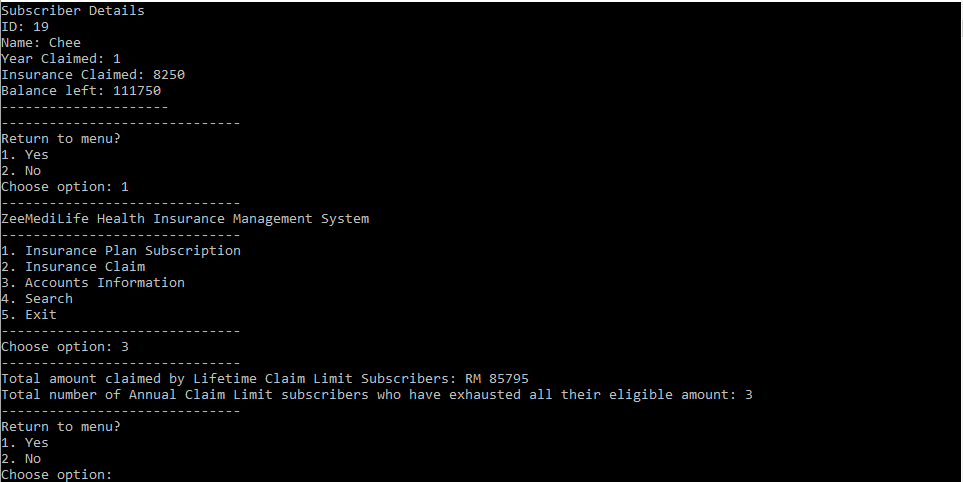


Figure Accounts Information.

In this screenshot, the option 3 is chosen from the main menu, which calls the accInfo function. This function calls two other functions: accInfoA and accInfoB. Each of these functions print each of the line printed in the Accounts Information section of the program.

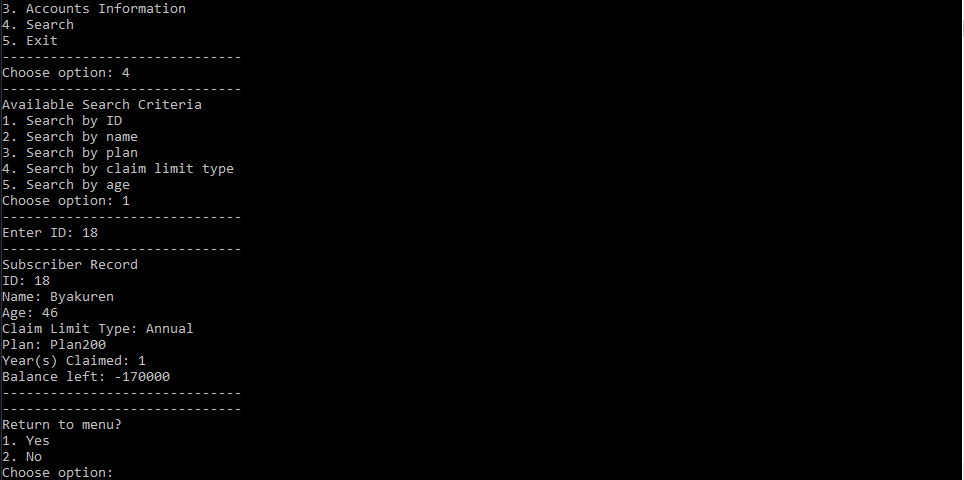


Figure A subscriber was searched by ID.

In this screenshot, the search by ID option was chosen and the number 18 was inputted. The records of the subscriber with the ID 18 is then displayed to the user.

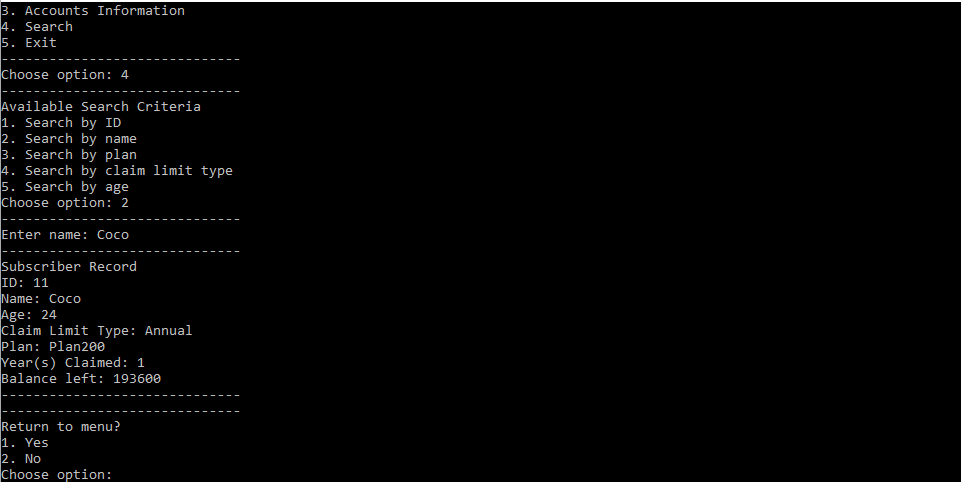


Figure A subscriber was searched by name.

In this screenshot, the search by name option was chosen and the word Coco was inputted. The records of the subscriber with the name Coco is then displayed to the user.



Figure Subscribers was searched by plan.

In this screenshot, the search by plan option was chosen and the number 2 was inputted. The records of all subscribers subscribed to Plan150 is then displayed to the user.

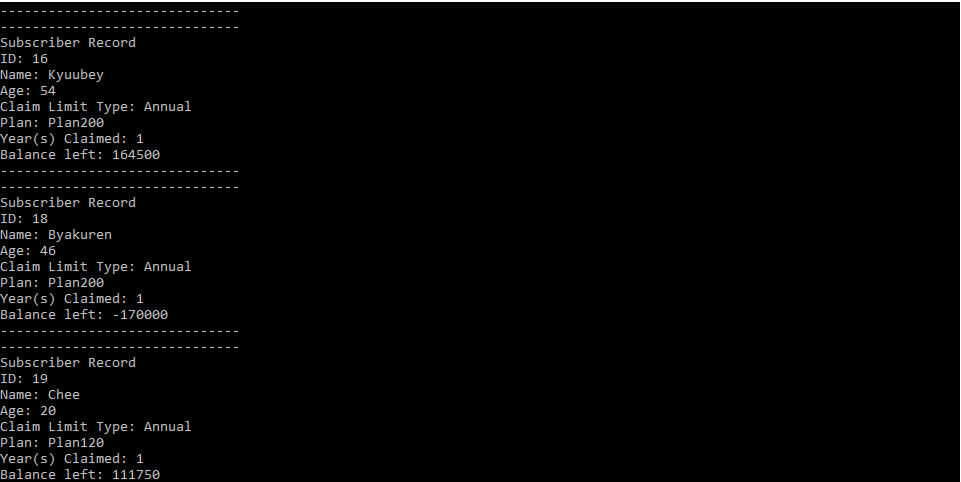


Figure Subscribers was searched by claim limit type.

In this screenshot, the search by type option was chosen and the number 1 was inputted. The records of all subscribers subscribed to annual claim limit plans is then displayed to the user.

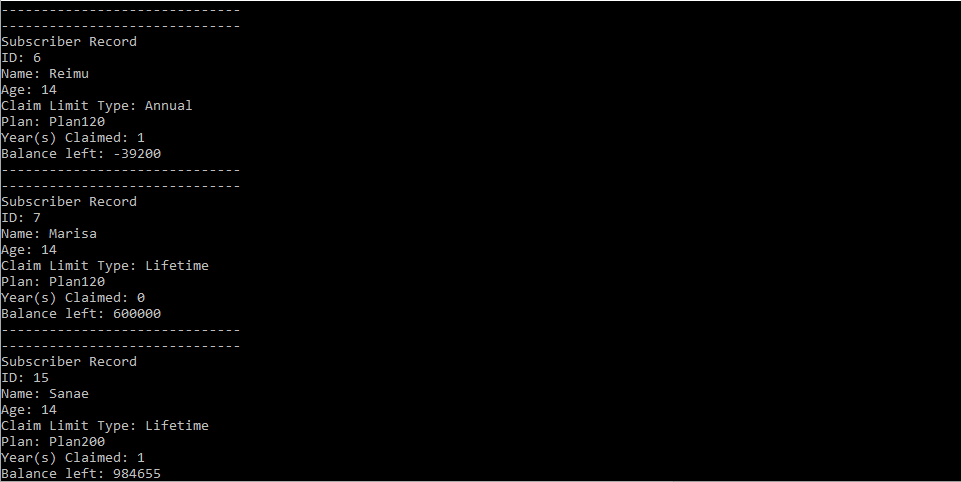


Figure Subscribers was searched by age.

In this screenshot, the search by age option was chosen and the number 14 was inputted. The records of all subscribers of age 14 is then displayed to the user.

**Conclusion**

In conclusion this program runs well and shows applications of a variety of C programming concepts that includes and is not limited to variables, control structures, and functions. The program is fundamentally a read and write program that reads data from text files, displays and/or modify the data, and write them back to the text files where they are kept and stored.

The program uses both built-in functions and programmer defined functions where each work together in harmony and help each other. These functions serve their own purposes as modules to the program as a whole and therefore the program is modular. For example, the insClaim function only handles insurance claims and the subReg function only handles subscribers’ registration.