**Table of Contents**

[**Introduction 2**](#_Toc82112682)

[**Pseudocode 3**](#_Toc82112683)

[**Flowchart 43**](#_Toc82112684)

[**Explanation of program 55**](#_Toc82112685)

[**ANSI C Programming Concept: 63**](#_Toc82112686)

[**Test statement table 64**](#_Toc82112687)

[**Output of Program 65**](#_Toc82112688)

[**Conclusion 67**](#_Toc82112689)

# **Introduction**

The C programming language is one of the most frequently used languages. Official and machine-independent programming languages ​​are also widely used in applications. The C programming language was used to create everything in the operating system, from complex programs. This program builds a console for health insurance and health insurance management systems. We offer a variety of choices depending on the age of our customers, but there are restrictions on the elderly. The most important feature of the console is that people can check in insurance, payment process account information and search function. Create this terminal using programming features such as struct pointer variables, functions, etc.

This approach saves paper by retaining information about the software on the device. On the one hand, it helps recover some important data that is missing. The quick steps in this system involve tracking subscribers and displaying the total lifetime subscriber bill and annual subscriber bill. This method helps the subscriber determine the desired quote type.

**Assumptions:**

First Reader will fill out a paper form with all the information and submit it to the counter. After that, the paper is more likely to be lost. It took a long time to complete the task. Finally, it takes time to report the supporter.

# **Pseudocode**

#include<stdio.h>

#include<string.h>

#include<stdlib.h>

#include<time.h>

void subscribe();

void OUTPUTPlan();

int planChecker(int);

char claimCheck();

void OUTPUTClaims();

int genId();

int incId();

void OUTPUTData();

int searchSub(char[]);

void claimInsur();

int calcClaim(char,int,int,int,int,int);

void WriteClaims();

void OUTPUTAccountInfo();

void main(){ //Main Function

FILE \*claim;

claim=fopen("claimData.txt","r");

if(claim==NULL){

FILE \*c;

c=fopen("claimData.txt","w");

fclose(c);

}

FILE \*count;

count=fopen("counter.txt","r");

if(count==NULL){

FILE \*newFile;

newFile=fopen("counter.txt","w");

int id=1;

fOUTPUTf(newFile,"%d",id);

fclose(newFile);

}

char b='f';

int yw;

OUTPUTf("\tWelcome to ZeeMedLife Health Insurance Company\n");

OUTPUTf("\n");

OUTPUTf("1)Make Subscription 2)Claim Insurance 3)Accounts Information 4)Search\n");

// subscribe();

do{

OUTPUTf(">>");

scanf("%d",&yw);

if(yw==1 || yw==2 || yw==3 || yw==4){

if(yw==1){

subscribe();

b='k';

}else if(yw==4){

FILE \*lamo;

lamo=fopen("data.txt","r");

if(lamo==NULL){

OUTPUTf("\n");

OUTPUTf("No Subscriber Found!\n");

exit(0);

}

char name[100];

int c=getc(stdin);

OUTPUTf("Enter subscriber nam: ");

fgets(name,100,stdin);

OUTPUTf("\n");

name[strlen(name)-1]= '\0'; //removing new line of strings

searchSub(name);

b='k';

}else if(yw==2){

FILE \*lamo;

lamo=fopen("data.txt","r");

if(lamo==NULL){

OUTPUTf("\n");

OUTPUTf("No Subscriber Found!\n");

exit(0);

}

claimInsur();

b='k';

}else if(yw==3){

OUTPUTAccountInfo();

b='k';

}

}else{

OUTPUTf("Invalid Input!\n");

}

}while(b=='f');

}

void subscribe(){

int sl=50;

int plantype;

struct Customer{

int id;

char name[sl];

char addr[sl];

int age;

long int phone;

char planType[sl];

char claimType[sl];

char healthHistory[sl];

int year;

};

int c=getc(stdin);

struct Customer c1;

SET time\_t t TO time(NULL);

SET struct tm tm TO \*localtime(&t);

c1.year=tm.tm\_year + 1900;

OUTPUTf("Enter your Name: ");

fgets(c1.name,sl,stdin);

OUTPUTf("Enter your Address: ");

fgets(c1.addr,sl,stdin);

OUTPUTf("Enter your Phone Number: ");

scanf("%li",&c1.phone);

OUTPUTf("Enter your age: ");

scanf("%d",&c1.age);

if(c1.age>54){

OUTPUTf("\n");

OUTPUTf("Insurance not available!!\n");

exit(0);

}

c1.name[strlen(c1.name)-1]= '\0'; //removing new line of strings

c1.addr[strlen(c1.addr)-1]= '\0';

c1.claimType[strlen(c1.claimType)-1]= '\0';

c1.planType[strlen(c1.planType)-1]= '\0';

OUTPUTf("\n");

OUTPUTPlan();

OUTPUTClaims();

plantype=planChecker(c1.age);

if(plantype==1){

strcpy(c1.planType,"Plan120");

}else if(plantype==2){

strcpy(c1.planType,"Plan150");

}else if(plantype==3){

strcpy(c1.planType,"Plan200");

}

OUTPUTf("\n");

SET int claimType TO claimCheck();

if(claimType==1){

strcpy(c1.claimType,"Annual Claim");

}else{

strcpy(c1.claimType,"Lifetime Claim");

}

FILE \*fp;

c1.id=genId();

incId();

fp=fopen("data.txt","a");

if(fp==NULL){

OUTPUTf("Error");

}else{

fOUTPUTf(fp,"%d-%s-%d-%li-%s-%s-%s-%d\n",c1.id,c1.name,c1.age,c1.phone,c1.addr,c1.planType,c1.claimType,c1.year);

}

fclose(fp);

}

void OUTPUTPlan(){ //this OUTPUT insurance plans

OUTPUTf("ZeeMedLife Health Insurance Plan (Choose by plan number): \n");

OUTPUTf("\n");

OUTPUTf("Types of Plans\t\tMonthly Premium\t\tAnnual Claim Limit\t\tLifetime Claim limit\n");

OUTPUTf("1)Plan120 \t\t120RM \t\t120000RM \t\t600000RM\n");

OUTPUTf("2)Plan150 \t\t150RM \t\t150000RM \t\t750000RM\n");

OUTPUTf("3)Plan200 \t\t200RM \t\t200000RM \t\t1000000RM\n");

OUTPUTf("(Note: Annual Claim is available till age of 60 and Lifetime Claim till exausted. )\n");

OUTPUTf("\n");

OUTPUTf("\n");

}

int planChecker(int age){ //this is FOR chossing plan type

int planType;

char b='f';

do{

OUTPUTf(">>");

scanf("%d",&planType);

if(planType==1 || planType ==2 ||planType ==3){

if(planType==1){

if(age>=1 && age<=20){

OUTPUTf("Plan120 Successfully chosen\n");

b='t';

}else{

OUTPUTf("plan not available!\n");

}

}else if(planType==2){

if(age>=1 && age<40){

OUTPUTf("Plan150 Successfully chosen\n");

b='t';

}else if(age>41 && age<=54){

OUTPUTf("Plan not available!\n");

}else{

OUTPUTf("Plan not available!\n");

}

}else if(planType==3){

OUTPUTf("Plan200 successfully chosen\n");

b='t';

}

}else{

OUTPUTf("Invalid INPUT!\n");

}

}while(b=='f');

RETURN planType;

}

char claimCheck(){

int cType;

OUTPUTf("Enter your Claim type:\n");

OUTPUTf("1)Annual Claim\t 2)Lifetime Claim\n");

char b='f';

do{

OUTPUTf(">>");

scanf("%d",&cType);

if(cType==1 || cType==2){

if(cType==1){

OUTPUTf("You've Chosen Annual Claim!\n");

b='q';

}else{

OUTPUTf("You've Chosen Lifetime Claim!\n");

b='q';

}

}else{

OUTPUTf("Invalid Input!\n");

}

}while(b=='f');

RETURN cType;

}

int genId(){

int id;

FILE \*fp;

fp=fopen("counter.txt","r");

rewind(fp);

fscanf(fp,"%d",&id);

RETURN id;

fclose(fp);

}

int incId(){

FILE \*fp;

int id;

fp=fopen("counter.txt","r");

fscanf(fp,"%d",&id);

fclose(fp);

FILE \*nFile;

id++;

nFile=fopen("counter.txt","w");

fOUTPUTf(nFile,"%d",id);

fclose(nFile);

// FILE \*fp;

// int id;

// fp=fopen("counter.txt","r");

// rewind(fp);

// fscanf(fp,"%d",&id);

// id++;

// fclose(fp);

// FILE \*nFile;

// nFile=fopen("counter.txt","w");

// fOUTPUTf(nFile,"%d",id);

// fclose(nFile);

}

void OUTPUTClaims(){

OUTPUTf("Hospital and Surgical Benefits\n");

OUTPUTf("\n");

OUTPUTf("Types of Claim plan120\t\tplan150\t\tplan200\n");

OUTPUTf("Room Charges 120/day\t\t150/day\t\t200/day\n");

OUTPUTf("ICU Charges 250/day\t\t400/day\t\t700/day\n");

OUTPUTf("(Note: For Hospital service and supplies, Surgical fees, other fees are as approved by ZeeMedLife)\n");

}

int searchSub(char name[]){

int check;

char\* arr[8];

FILE \*fp;

fp=fopen("data.txt","r");

char lamo[400];

while(fscanf(fp,"%[^\n] ",lamo)!=EOF){

// OUTPUTf("%s\n",lamo);

SET char\* token TO strtok(lamo, "-");

int n=0;

while(token!=NULL){

// OUTPUTf("%s\n",token);

arr[n]=token;

n++;

token=strtok(NULL, "-");

}

check=strcmp(name,arr[1]);

if(check==0){

break;

}

}

if(check==0){

OUTPUTf("Your Details:\n");

OUTPUTf("id\t\t%s\n",arr[0]);

OUTPUTf("Name\t\t%s\n",arr[1]);

OUTPUTf("Age\t\t%s\n",arr[2]);

OUTPUTf("Phone No.\t%s\n",arr[3]);

OUTPUTf("Address.\t%s\n",arr[4]);

OUTPUTf("Plan Type\t%s\n",arr[5]);

OUTPUTf("Claim Type\t%s\n",arr[6]);

OUTPUTf("Subscribed Year\t%s\n",arr[7]);

int b=atoi(arr[0]);

RETURN b;

}else{

OUTPUTf("Subscriber Doesn't exist!\n");

RETURN 0;

}

}

void claimInsur(){

OUTPUTf("\n");

char name[100];

int c=getc(stdin);

OUTPUTf("Enter Subscriber name: ");

fgets(name,100,stdin);

name[strlen(name)-1]= '\0';

OUTPUTf("\n");

SET char id TO searchSub(name);

if(id){

char type;

int hd,hSS,sF,oF;

OUTPUTf("\n");

OUTPUTf("Enter Claim details:\n");

OUTPUTf("\n");

OUTPUTf("Number of Days IN Hospital: ");

scanf("%d",&hd);

OUTPUTf("ICU(I) or Normal(N): ");

scanf(" %c",&type);

OUTPUTf("Hospital Supplies and Services: ");

scanf("%d",&hSS);

OUTPUTf("Surgical Fees: ");

scanf("%d",&sF);

OUTPUTf("Other Fees: ");

scanf("%d",&oF);

calcClaim(type, id, hd, hSS,sF, oF);

}

}

int calcClaim(char type, int id, int hD, int hSS, int sF, int oF){

char str1[400];

int RoomClaim=0;

int claimLimit=0;

char claimtype[100];

int rDate;

FILE \*fp;

char\* arr[8];

fp=fopen("data.txt","r");

while(fscanf(fp,"%[^\n] ",str1)!=EOF){

SET char\* token TO strtok(str1, "-");

int n=0;

while(token!=NULL){

// OUTPUTf("%s\n",token);

arr[n]=token;

n++;

token=strtok(NULL, "-");

}

int newId=atoi(arr[0]);

if(id==newId){

rDate=atoi(arr[7]);

strcpy(claimtype,arr[6]);

if(strcmp(arr[5],"Plan120")==0){

if(strcmp(arr[6],"Lifetime Claim")==0){

claimLimit=600000;

}else if(strcmp(arr[6],"Annual Claim")==0){

claimLimit=120000;

}

if(type=='N'){

RoomClaim=hD\*120;

}else if(type=='I'){

RoomClaim=hD\*250;

}

}else if(strcmp(arr[5],"Plan150")==0){

if(strcmp(arr[6],"Annual Claim")==0){

claimLimit=150000;

}else if(strcmp(arr[6],"Lifetime Claim")==0){

claimLimit=750000;

}

if(type=='N'){

RoomClaim=hD\*150;

}else if(type=='I'){

RoomClaim=hD\*400;

}

}else if(strcmp(arr[5],"Plan200")==0){

if(strcmp(arr[6],"Annual Claim")==0){

claimLimit=200000;

}else if(strcmp(arr[6],"Lifetime Claim")==0){

claimLimit=1000000;

}

if(type=='N'){

RoomClaim=hD\*200;

}else if(type=='I'){

RoomClaim=hD\*700;

}

}

}

}

FILE \*checkClaim;

checkClaim=fopen("claimData.txt","r");

SET time\_t t TO time(NULL);

SET struct tm tm TO \*localtime(&t);

int claimDate=tm.tm\_year + 1900;

if(checkClaim!=NULL){

fseek(checkClaim,0,SEEK\_END);

size\_t size= ftell(checkClaim);

if(0==size){

int totalClaim=RoomClaim+hSS+sF+oF;

SET int RemainingClaim TO claimLimit-totalClaim;

WriteClaims(id,totalClaim,RemainingClaim,claimtype,claimDate,rDate);

}else{

char str1[400];

FILE \*readClaim;

char\* arr[7];

readClaim=fopen("claimData.txt","r");

char checkId=6;

while(fscanf(readClaim,"%[^\n] ",str1)!=EOF){

SET char\* token TO strtok(str1, "-");

int n=0;

while(token!=NULL){

// OUTPUTf("%s\n",token);

arr[n]=token;

n++;

token=strtok(NULL, "-");

}

if(id==atoi(arr[0])){

checkId=1;

FILE \*sClaim;

char sss[500];

char\* sArr[7];

int sId;

sClaim=fopen("claimData.txt","r");

while(fscanf(sClaim,"%[^\n] ",sss)!=EOF){

SET char\* token TO strtok(sss, "-");

int p=0;

while(token!=NULL){

// OUTPUTf("%s\n",token);

sArr[p]=token;

p++;

token=strtok(NULL, "-");

}

sId=atoi(sArr[0]);

if(id==sId){

claimLimit=atoi(sArr[2]);

if(claimLimit==0){

OUTPUTf("\n");

OUTPUTf("You've exausted Your claims!\n");

exit(0);

}

int OldClaim=atoi(sArr[1]);

}

}

}

}

int totalClaim=RoomClaim+hSS+sF+oF;

int RemainingClaim;

if(totalClaim>claimLimit){

RemainingClaim=claimLimit;

}else{

RemainingClaim=claimLimit-totalClaim;

}

if(checkId!=1){

int totalClaim=RoomClaim+hSS+sF+oF;

SET int RemainingClaim TO claimLimit-totalClaim;

WriteClaims(id,totalClaim,RemainingClaim,claimtype,claimDate,rDate);

}else{

WriteClaims(id,totalClaim,RemainingClaim,claimtype,claimDate,rDate);

}

}

}

fclose(checkClaim);

}

void WriteClaims(int id, int totalClaim, int RemainingClaim, char claimType[100], int claimDate, int rDate){

if(RemainingClaim==0){

OUTPUTf("\n");

OUTPUTf("Claim Successful!\n");

OUTPUTf("Total Claim: %dRM\n",totalClaim);

OUTPUTf("Remaining Claim: %dRM\n",RemainingClaim);

FILE \*claimWrite;

claimWrite=fopen("claimData.txt","a");

fOUTPUTf(claimWrite,"%d-%d-%d-%s-%d-%d\n",id,totalClaim,RemainingClaim,claimType,claimDate,rDate);

fclose(claimWrite);

}else if(totalClaim>RemainingClaim){

OUTPUTf("\n");

OUTPUTf("Error!Your claim is greater than Available balance\n",RemainingClaim);

OUTPUTf("Your claim: %dRM\n",totalClaim);

OUTPUTf("Available Balance:%dRM\n",RemainingClaim);

}else{

OUTPUTf("\n");

OUTPUTf("Claim Successful!\n");

OUTPUTf("Total Claim: %dRM\n",totalClaim);

OUTPUTf("Remaining Claim: %dRM\n",RemainingClaim);

FILE \*claimWrite;

claimWrite=fopen("claimData.txt","a");

fOUTPUTf(claimWrite,"%d-%d-%d-%s-%d-%d\n",id,totalClaim,RemainingClaim,claimType,claimDate,rDate);

fclose(claimWrite);

}

}

void OUTPUTAccountInfo(){

FILE \*readData;

FILE \*readClaim;

readClaim=fopen("claimData.txt","r");

readData=fopen("data.txt","r");

if(readClaim!=NULL){

fseek(readClaim,0,SEEK\_END);

SET size\_t size TO ftell(readClaim);

if(size EQUALS 0 && readData ==NULL){

OUTPUTf("\n");

OUTPUTf("No Datas found!\n");

}else if(readData!=NULL){

char spp[500];

char\* arr[8];

int lCount=0,aCount=0;

while(fscanf(readData,"%[^\n] ",spp)!=EOF){

SET char\* token TO strtok(spp, "-");

int n=0;

while(token!=NULL){

arr[n]=token;

n++;

token=strtok(NULL, "-");

}

if(strcmp(arr[6],"Lifetime Claim")==0){

lCount++;

}else if(strcmp(arr[6],"Annual Claim")==0){

aCount++;

}

}

OUTPUTf("\n");

OUTPUTf("Account Details\n");

OUTPUTf("\n");

OUTPUTf("Total Lifetime Claim Subscribers: %d\n",lCount);

OUTPUTf("Total Annual Claim Subscribers: %d\n",aCount);

}

if(size!=0){

FILE \*rrClaim;

char stt[500];

char\* arr[6];

int countA=0,countL=0,eSubs=0,totalL=0,totalA=0;

rrClaim=fopen("claimData.txt","r");

while(fscanf(rrClaim,"%[^\n] ",stt)!=EOF){

SET char\* token TO strtok(stt, "-");

int n=0;

while(token!=NULL){

// OUTPUTf("%s\n",token);

arr[n]=token;

n++;

token=strtok(NULL, "-");

}

if(strcmp(arr[3],"Annual Claim")==0){

totalA+=atoi(arr[1]);

}else if(strcmp(arr[3],"Lifetime Claim")==0){

totalL+=atoi(arr[1]);

}

if(atoi(arr[2])==0){

eSubs++;

}

}

fclose(rrClaim);

// OUTPUTf("\n");

OUTPUTf("Total Exausted subscribers: %d\n",eSubs);

OUTPUTf("Total Amount Claimed by Lifetime Subscriber: %d\n",totalL);

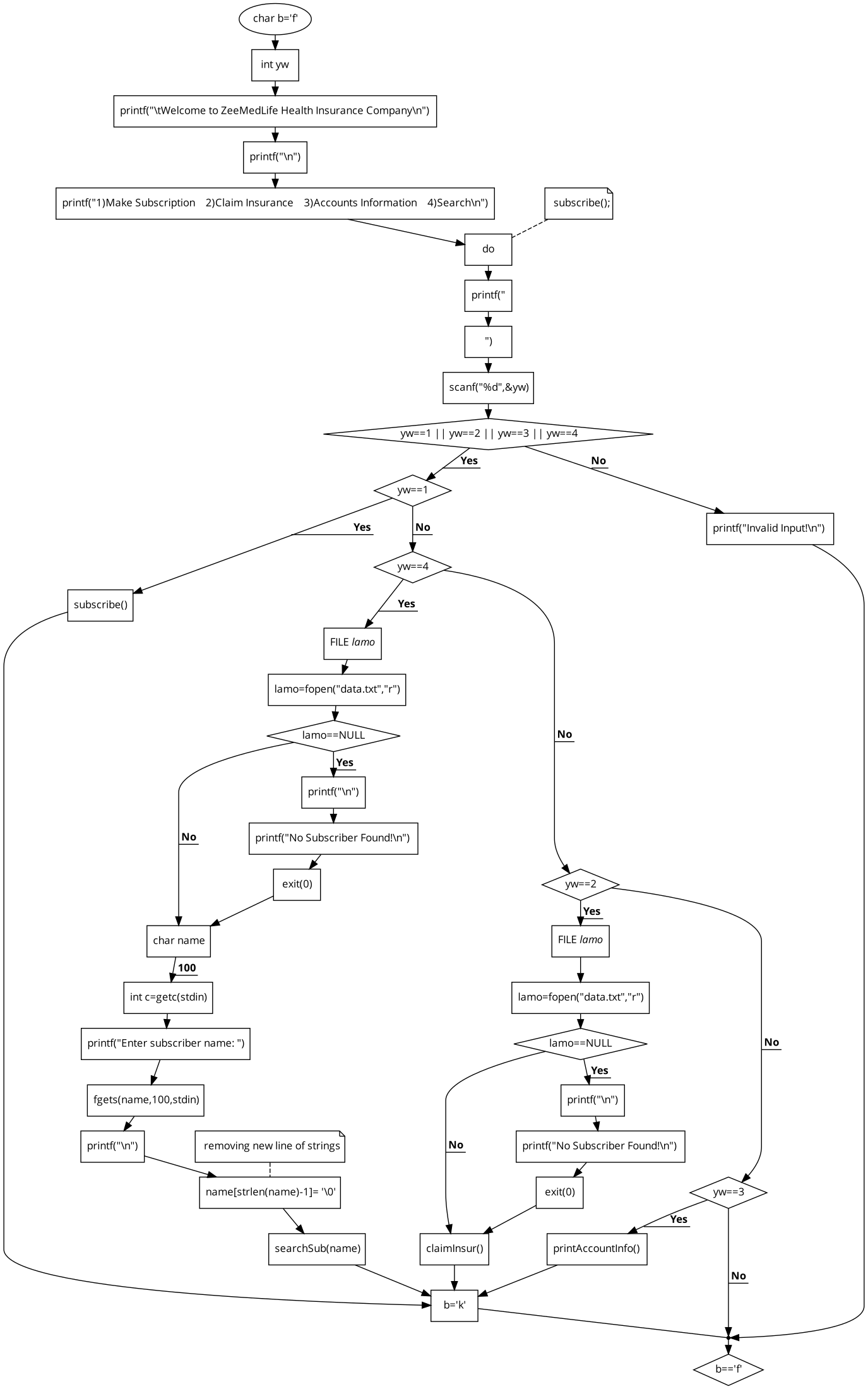
OUTPUTf("Total Amount Claimed by Annual Subscriber: %d\n",totalA);

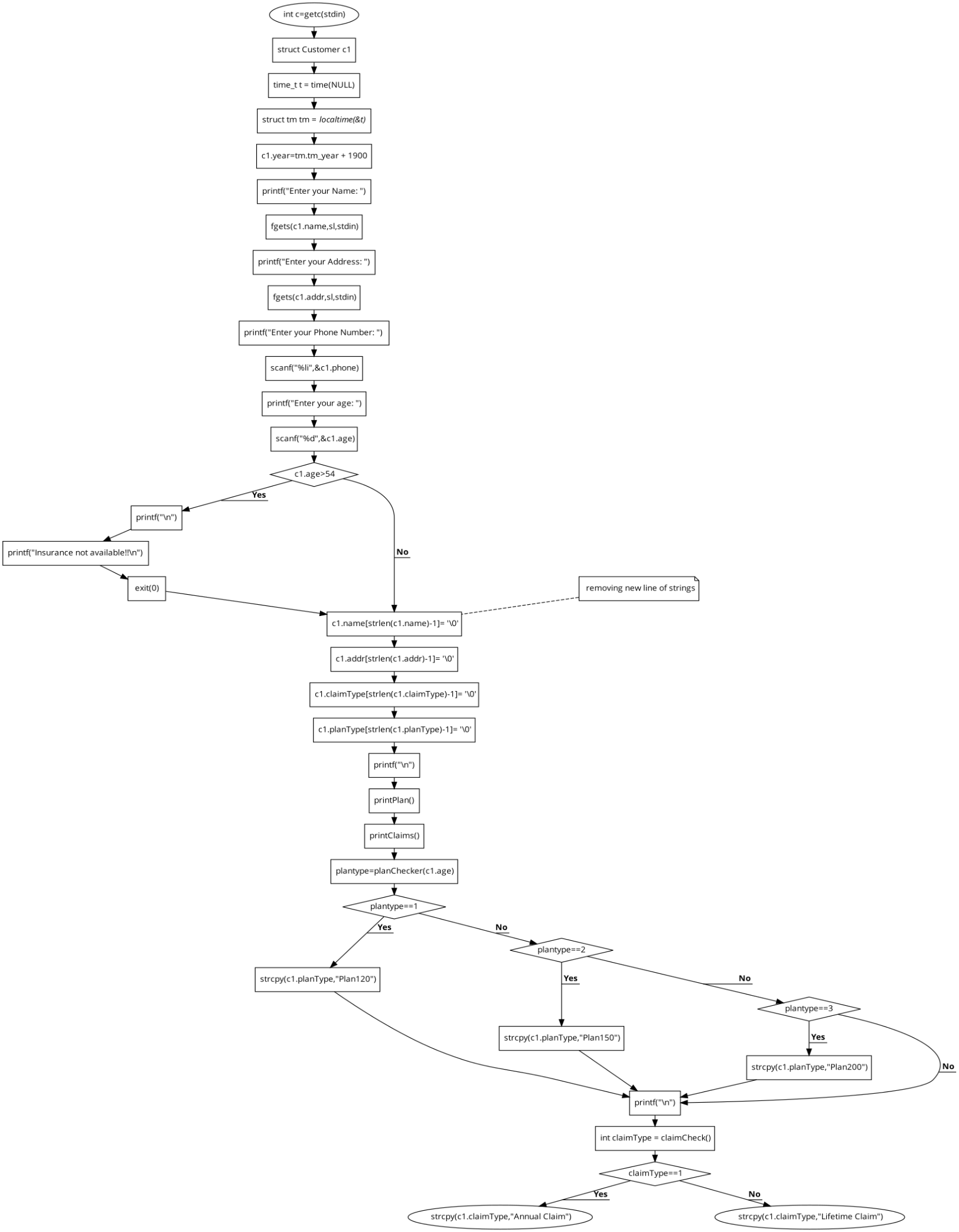
}

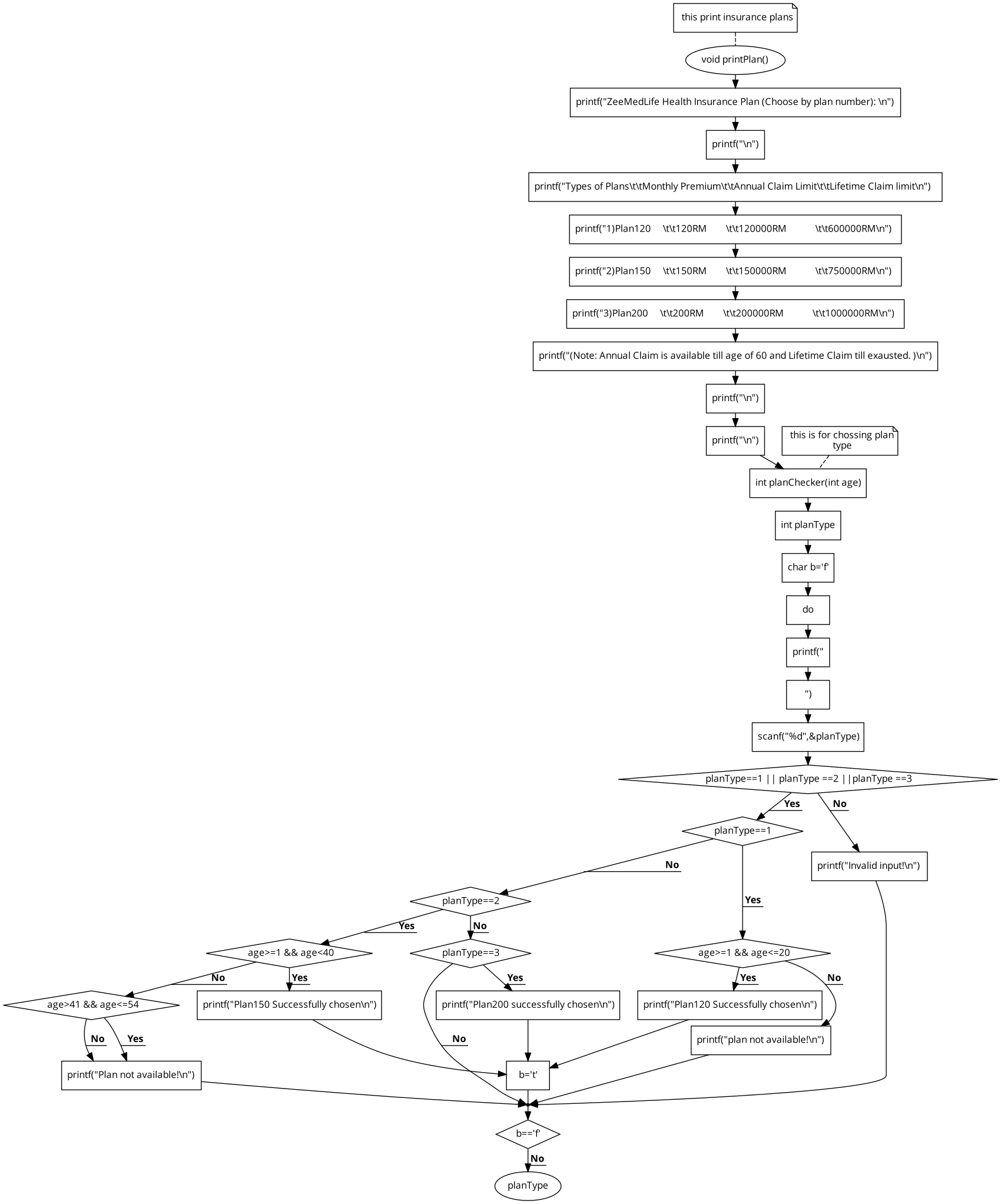
}

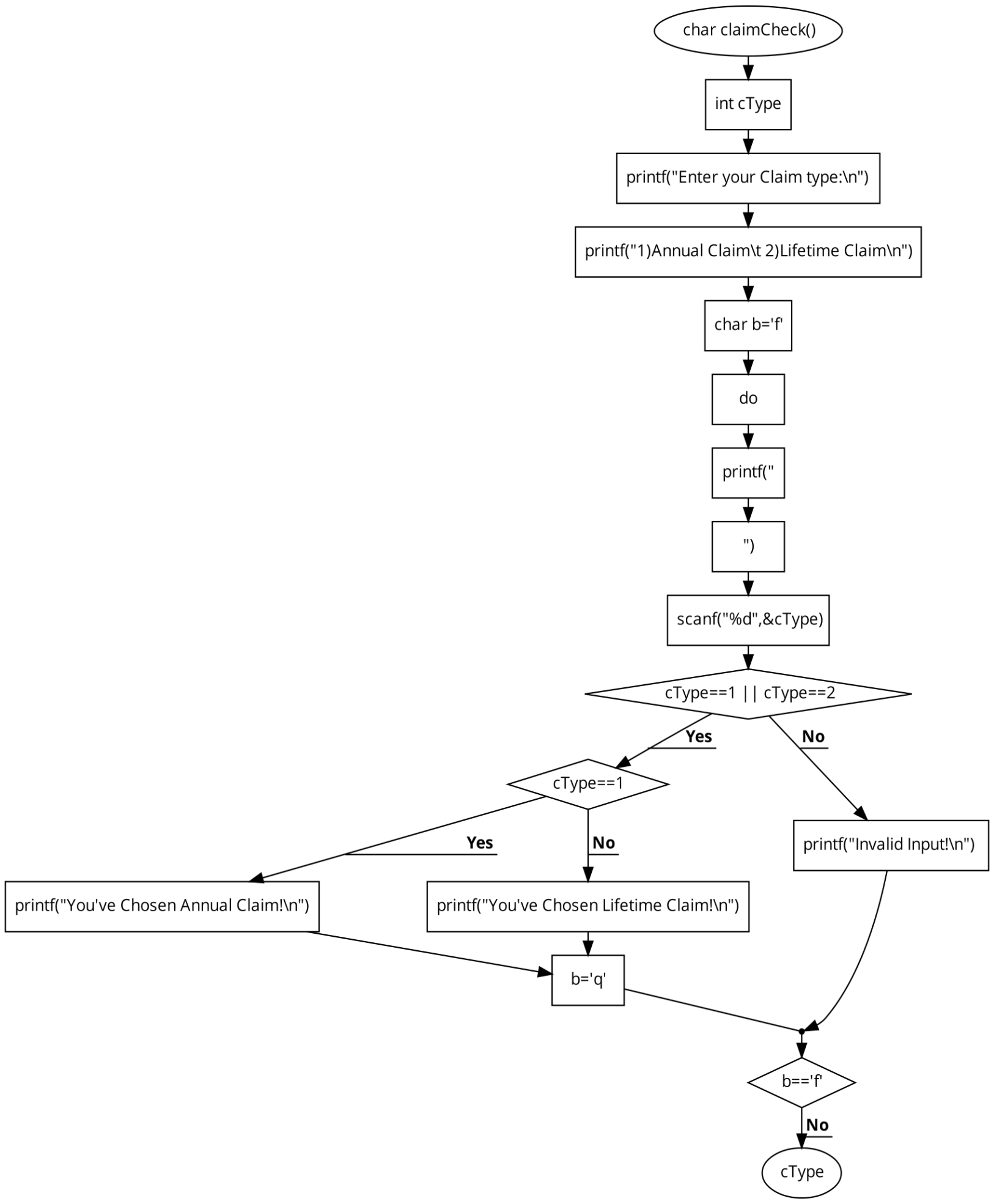
}

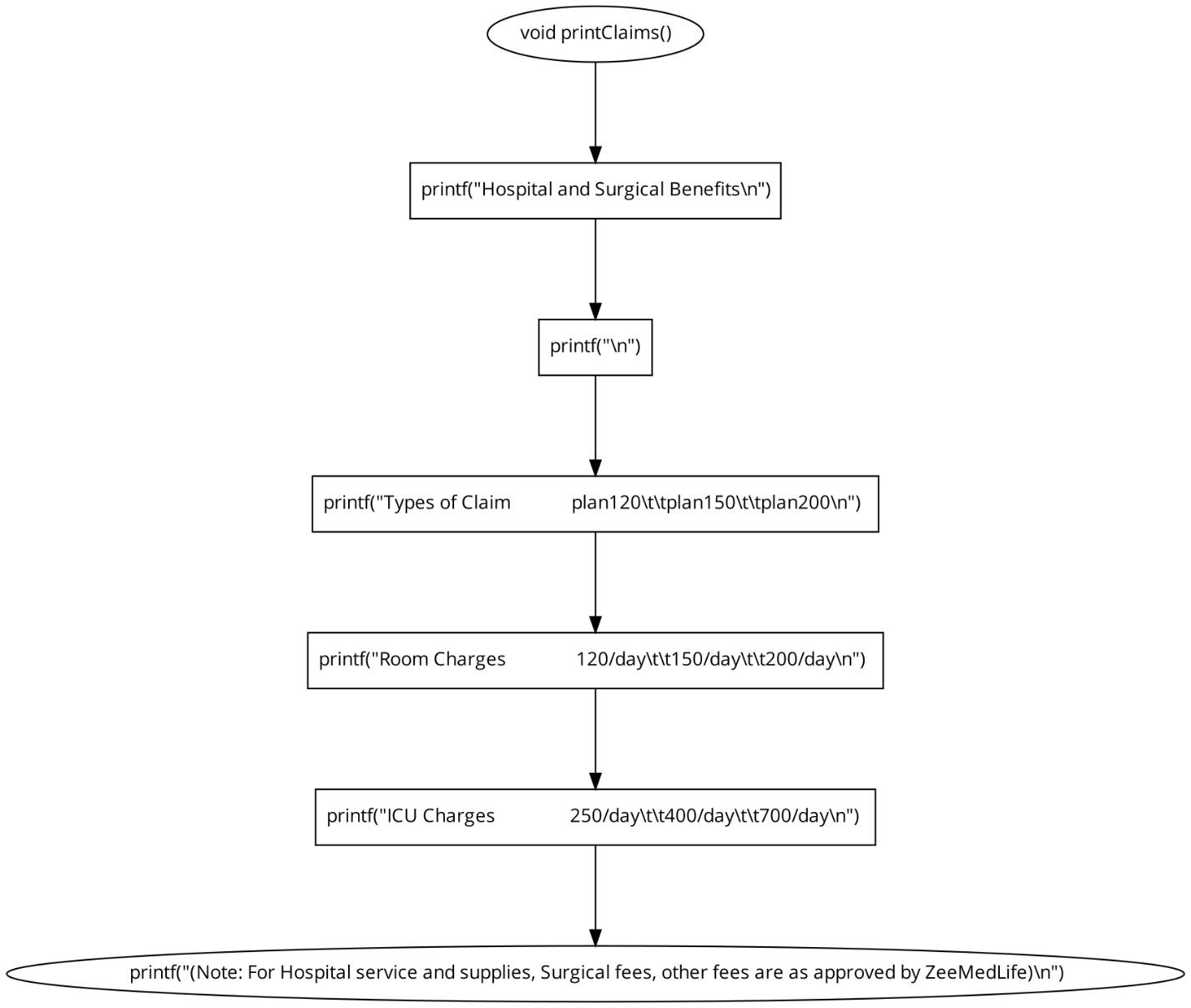
# **Flowchart**

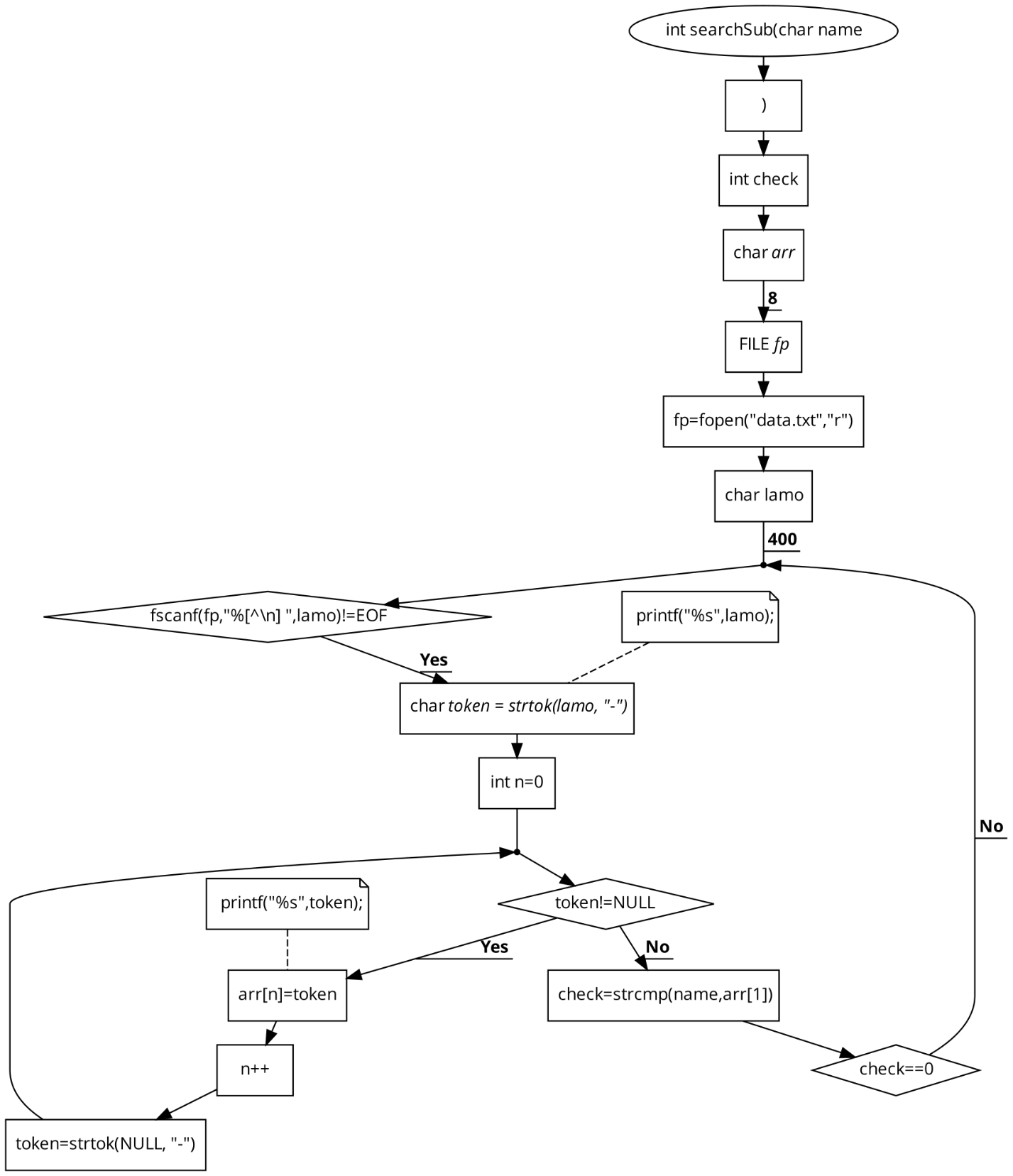


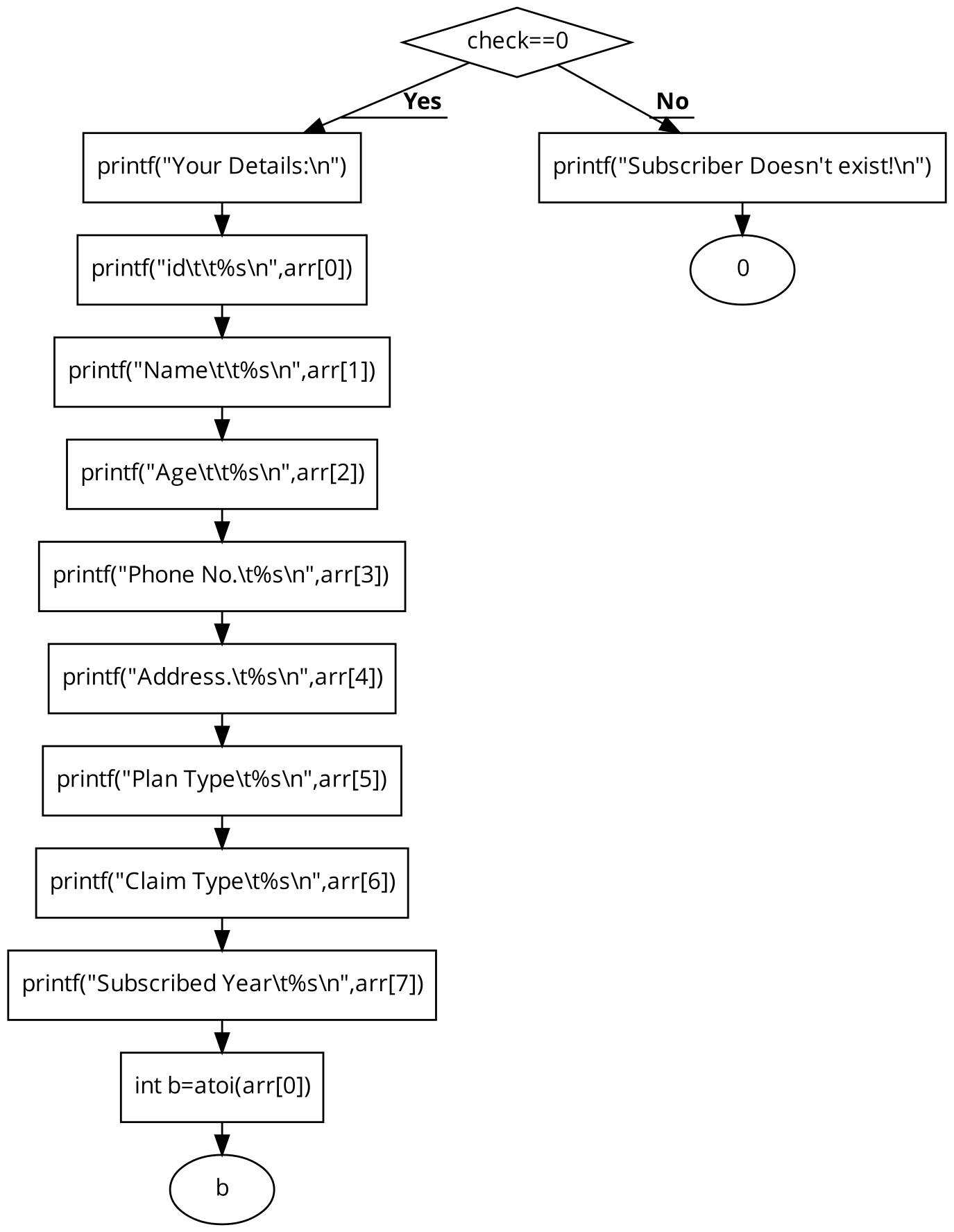


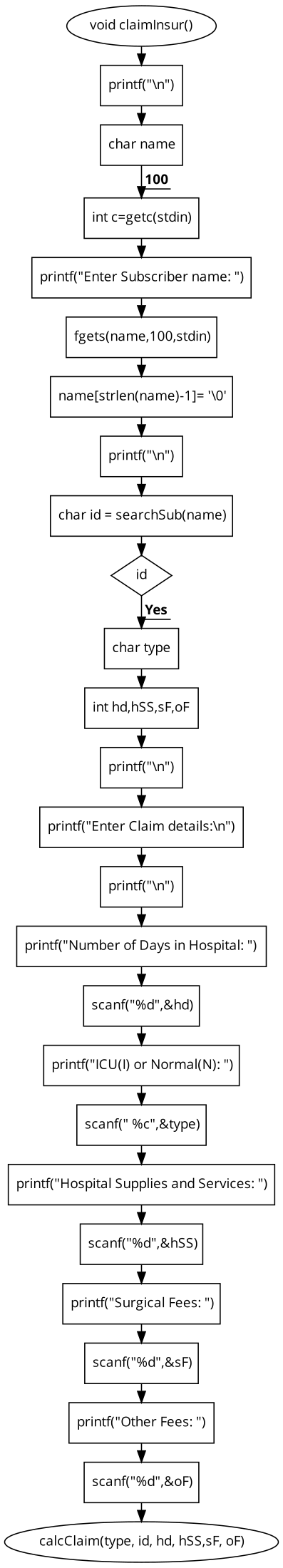


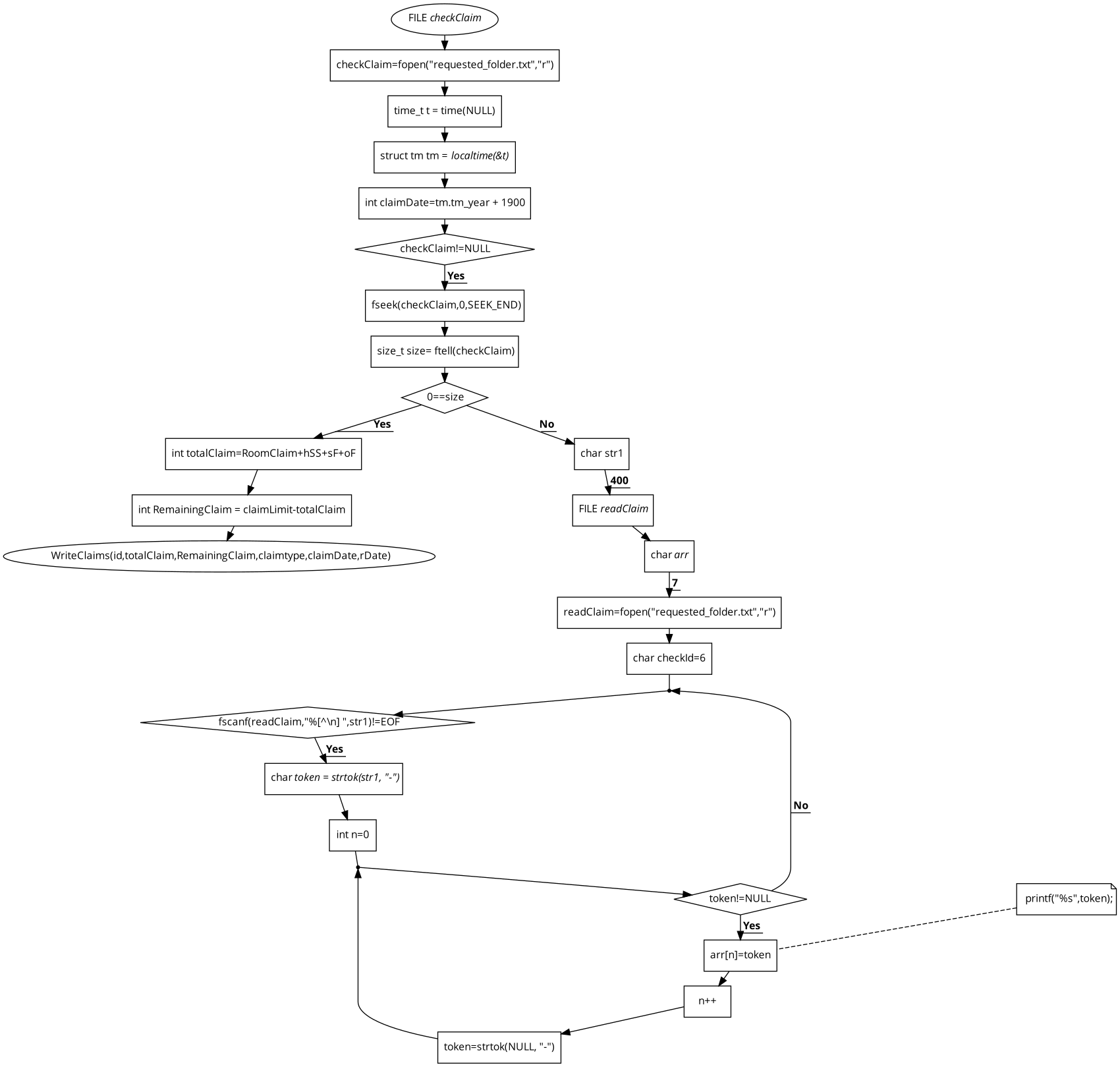


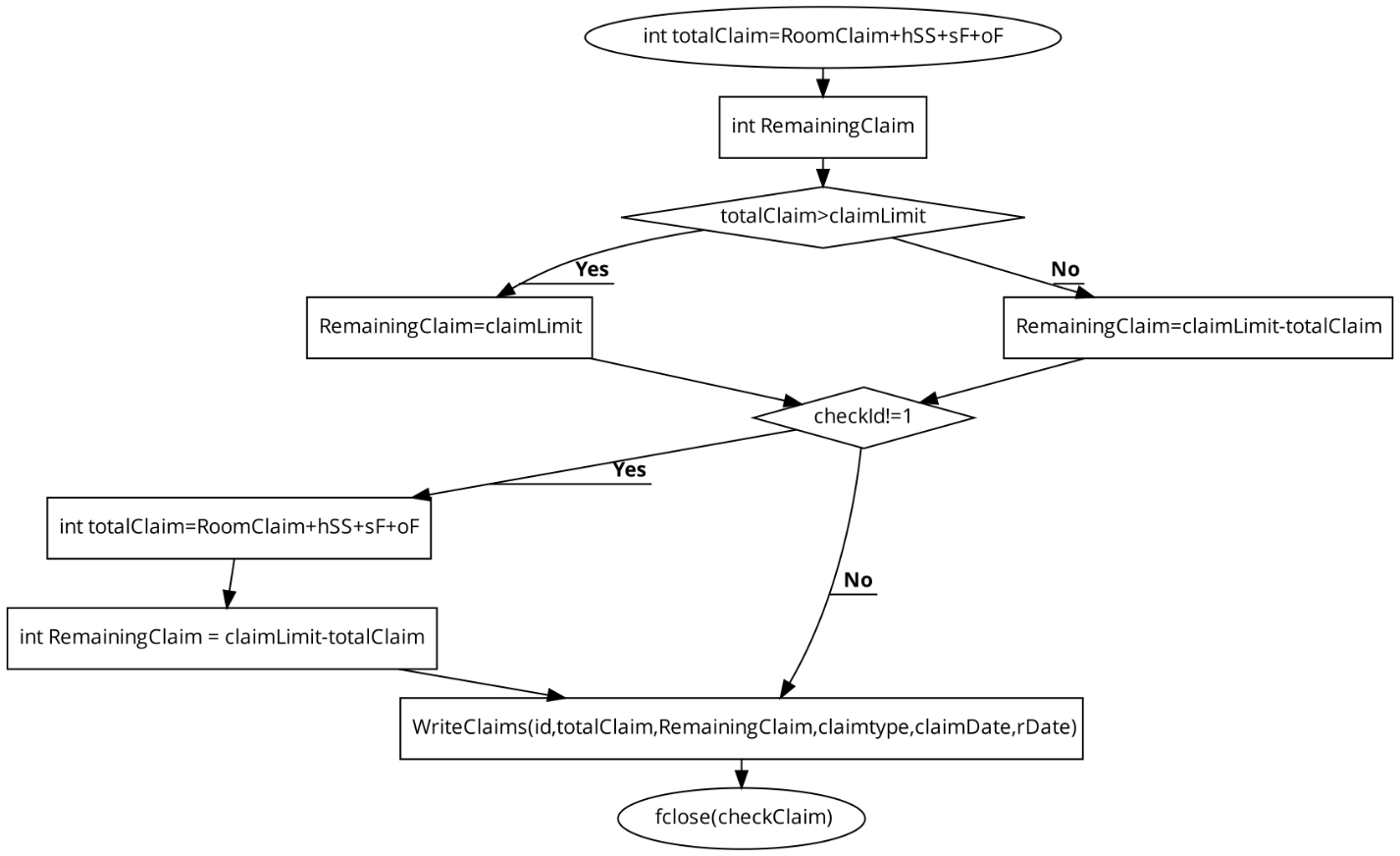


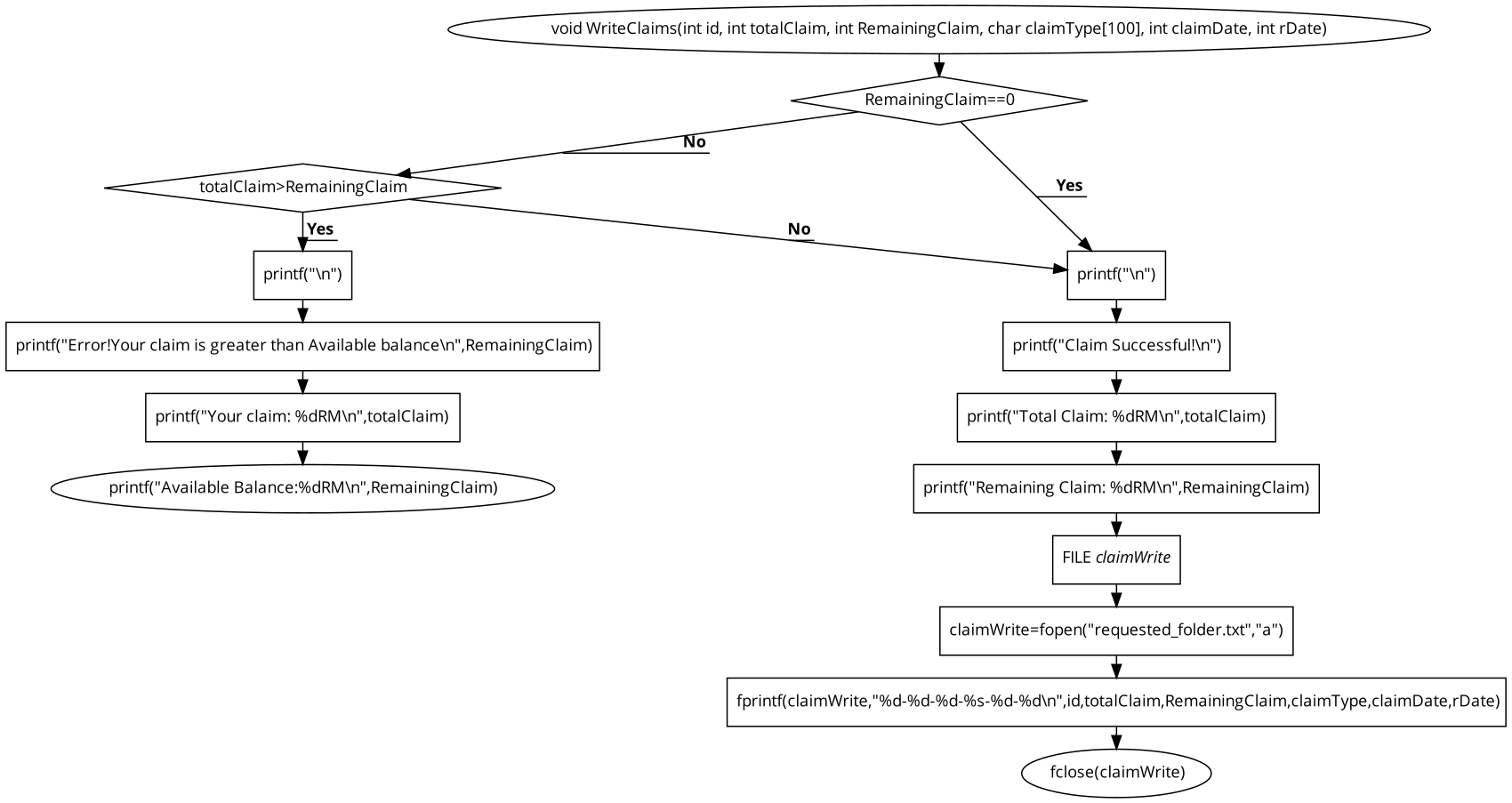


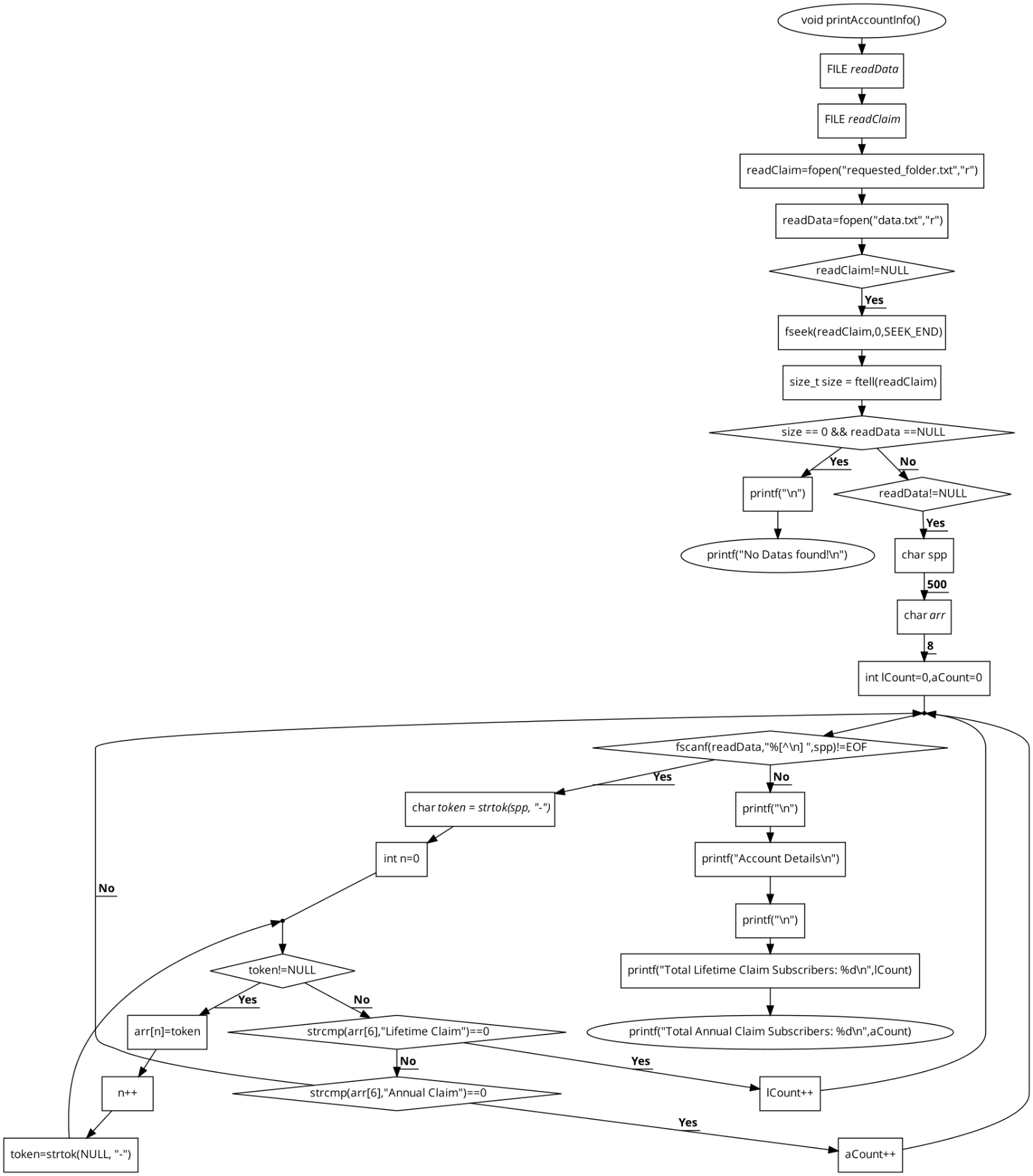


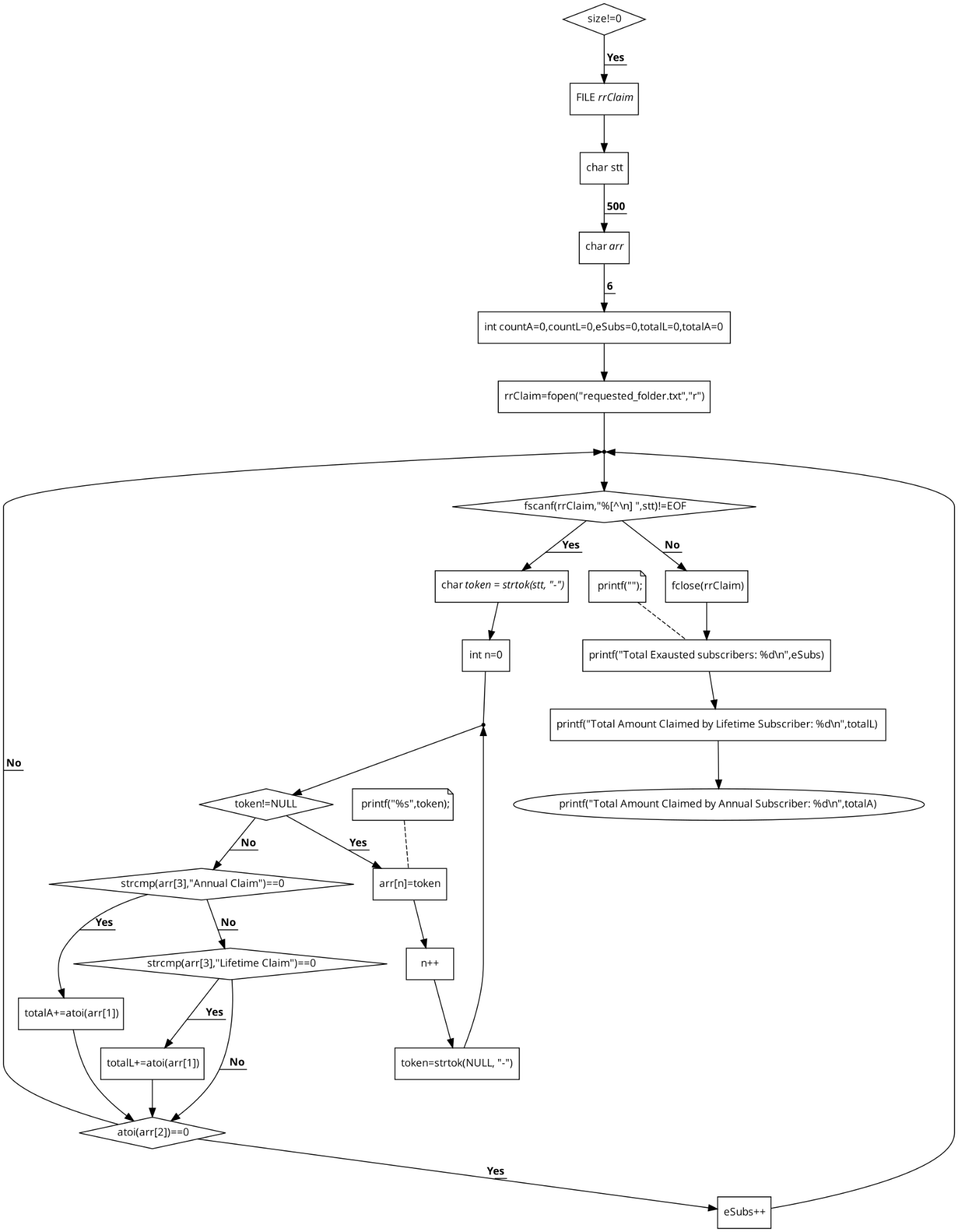




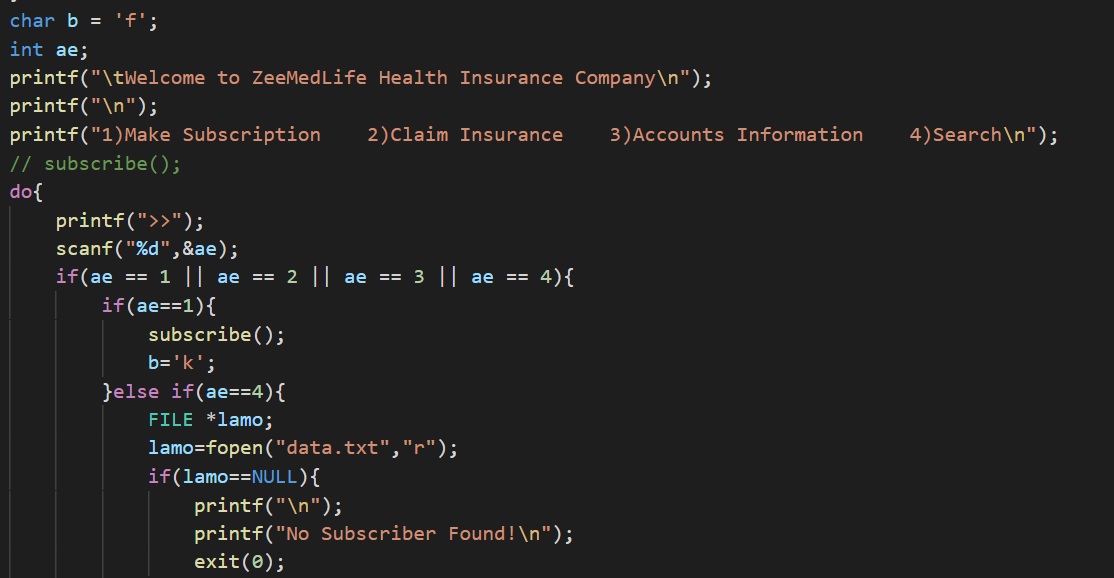








# **Explanation of program**

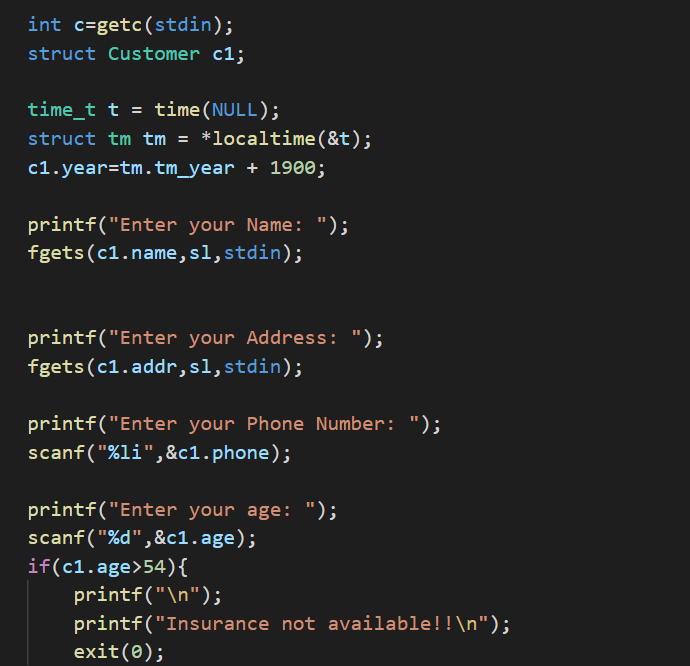


Allows us to select from 1 to 4 the items we require, such as:

1. Create a Subscription
2. File an Insurance Claim
3. Accounts Data
4. Search



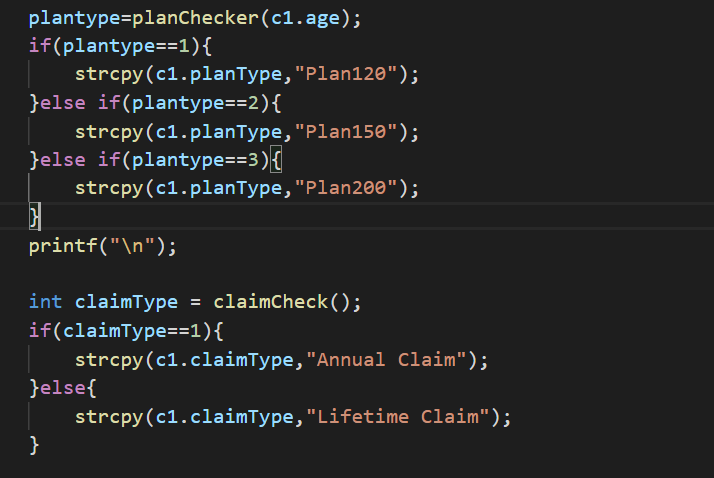
Examine the subscriber's name.



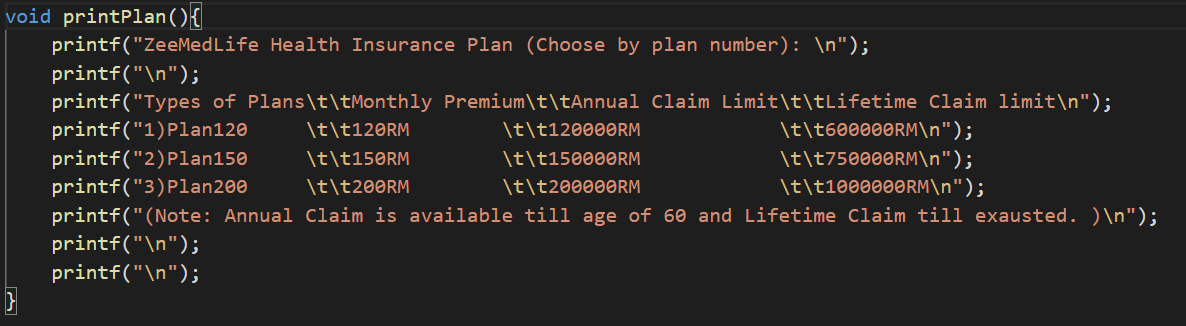
Add details about the subscriber.



Add a new string line and delete it.



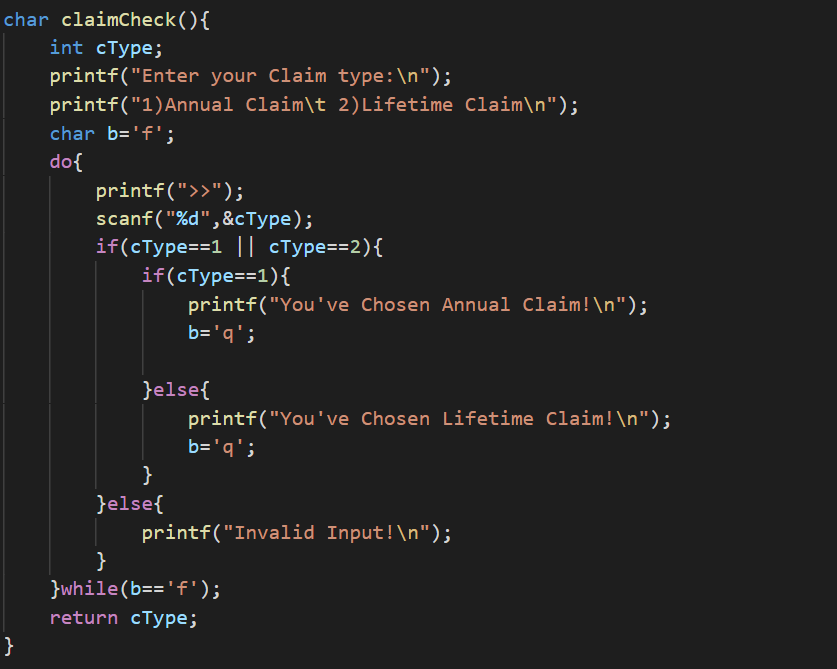
Check out the different types of plans and claims.



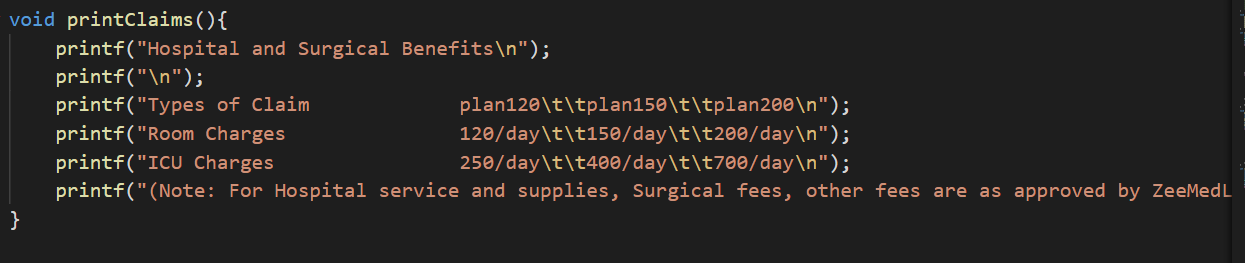
Policies should be printed out.



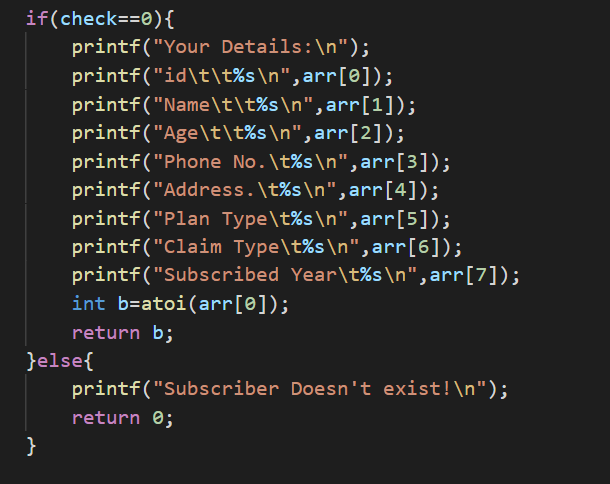
It aids in the decision-making process while choosing plan types.



It aids in the selection of the claims kinds that we require.



It aids in the selection of the claims kinds that we require. It facilitates the printing of various sorts of claims as well as medical services and fees.

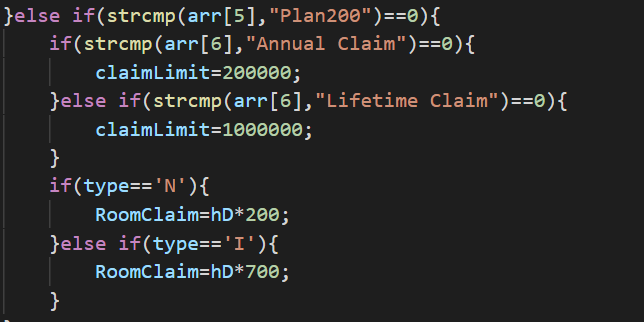


Checks our characteristics such as name, age, phone number, and many more.

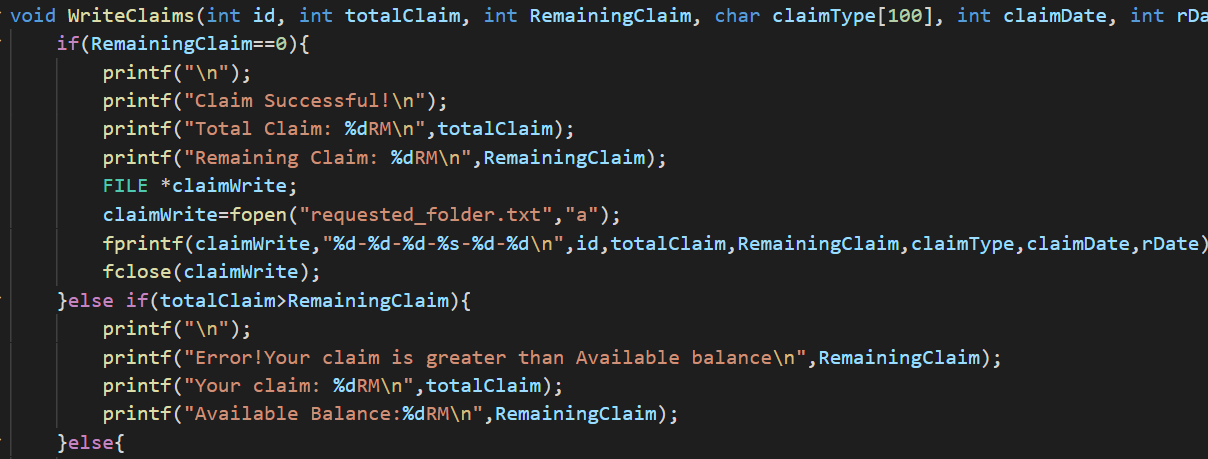


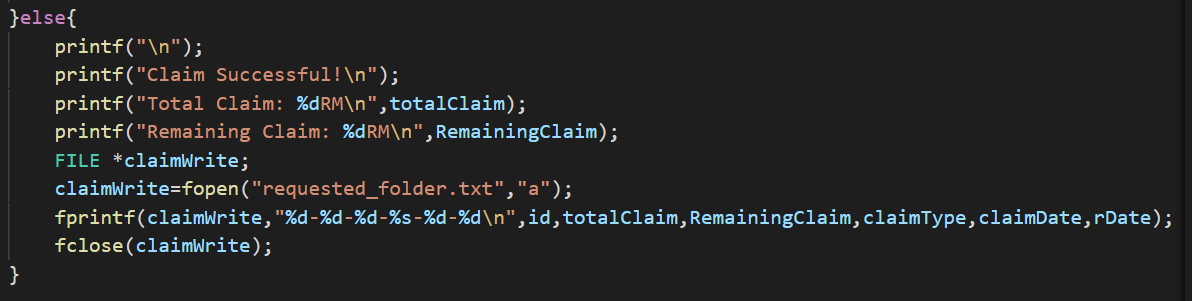
You can print the member's name and charge for what you need, such as the number of days in the hospital, the type of bed you need, and the cost.

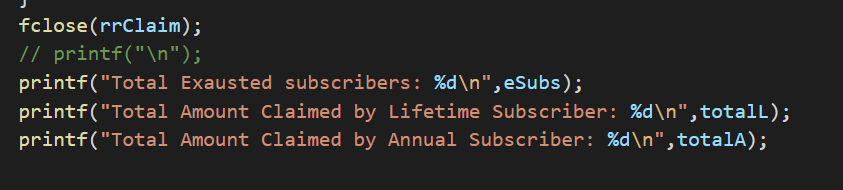




It assists in entering the most recent id and determining the types of plans and claims that a new client requires.





It helps determine if the bill was successful, otherwise you can enter the bill and check again.

It is possible to print the total number of subscribers and the total volume claimed by lifetime or annual subscribers using this tool.

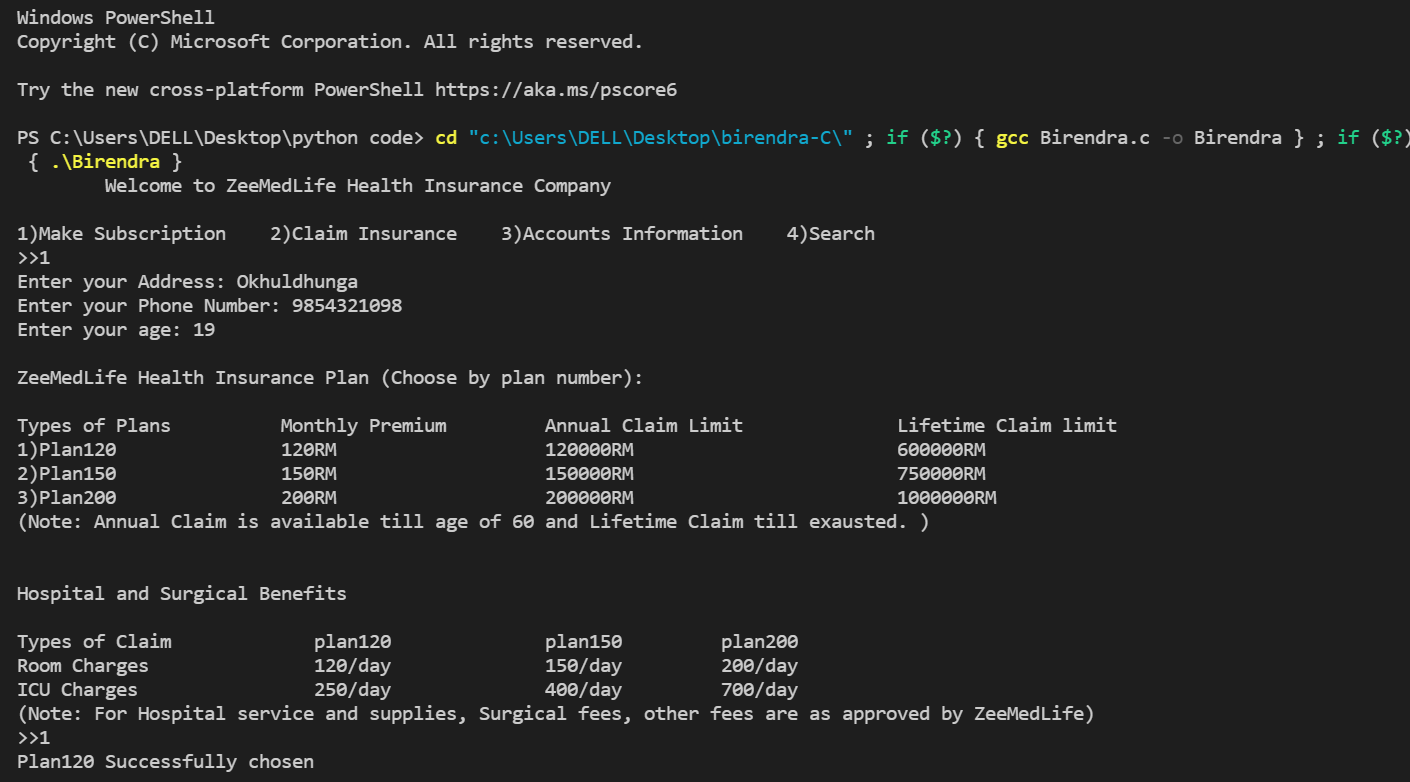
# **ANSI C Programming Concept:**

ANSI C is a commonly used term in the C programming language level, which is still consistent with the current quality, for all major changes in named files over time, The current standard is still mostly used. This is the beginning of a well-known name (National Institute of Standards). The basic standardized form of C is suggestive in "ANSI C", but can be implied by "C89" or "C90", depending on the change. C99 and the latest C11 are standard variations that accompany ANSI C. Other current versions (small C, Arduino, etc.).

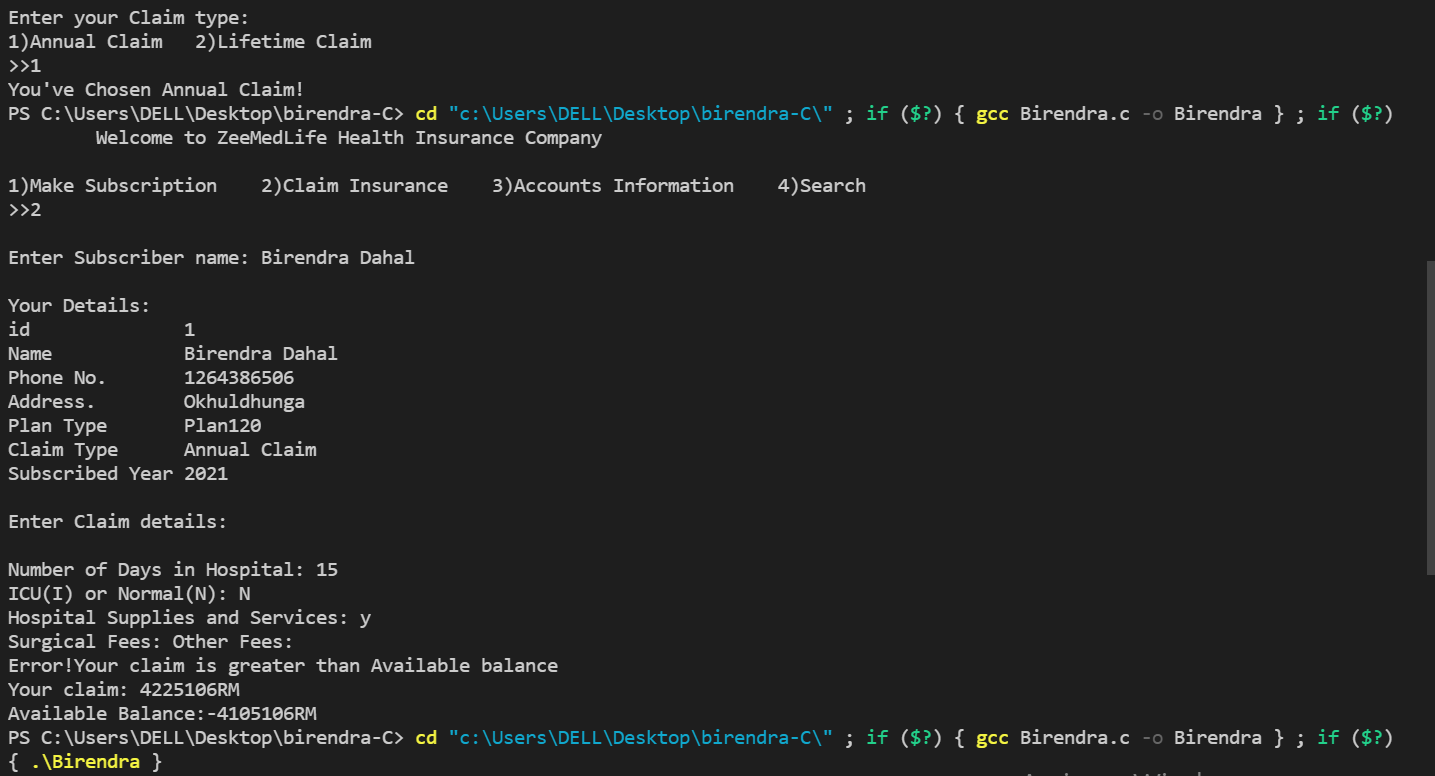
# **Test statement table**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Experiment No.** | **Illustration** | **Trial proof**  **(Insert)** | **Anticipated**  **result** | **Real**  **result** | **Extra act needed** |
| a | The main menu |  | Greetings and welcome to the EZMedilife health insurance website | Good day and welcome to the EZMedilife health insurance website | Not at all |
| b | The procedure of subscribing and claiming | Fill in the subscriber's name, hospitalization days, ICU/Normal ward, and all expenses | Insurance policy and claim table, as well as conditions to meet and a message indicating that the claim was successfully processed | Insurance policy and claim table, as well as conditions to meet and a message indicating that the claim was successful | Not at all |
| c | Account specific details | Account information is printed | Account information is presented | Account information is presented. | Not at all |
| d | In search of data | Accounts may be searched by name, id, plan, and claim type | Accountant detail with care | Accountant detail with care | Not at all |

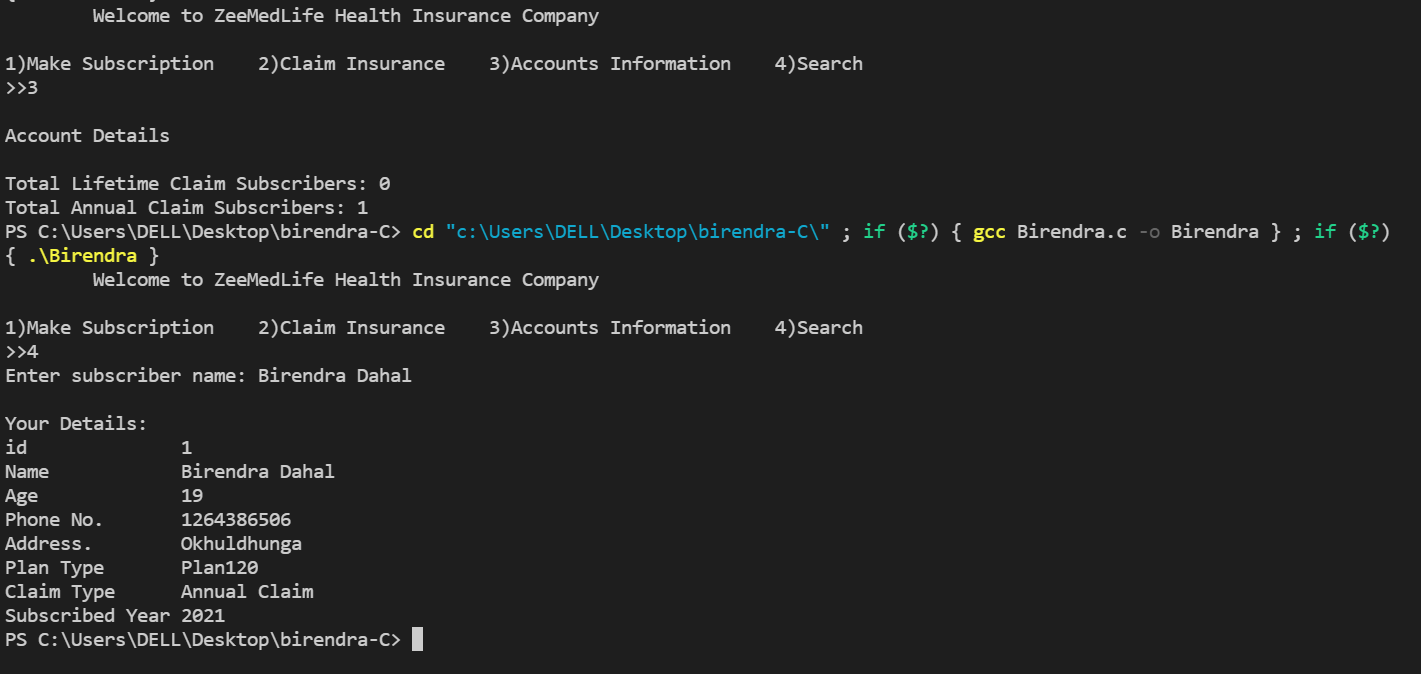
# **Output of Program**



The zeeMedLife Health Insurance plan, as well as hospital and surgical coverage, appear to be available here.



This output displays our information and allows us to submit claim information.



Our name and account information are displayed in this output.

# **Conclusion**

We may describe this program as having a structure that allows both the subscriber and the user to pick an item from a list. Subscribers may select between hospital and surgical benefits simply by providing detailed information, as well as claim types and plan options. They can also reserve specific sorts of beds and the number of days they wish to spend in the hospital. The subscriber may also verify whether the claim was successful or not, and if it wasn't, they can assist in the claim's entry and review. Finally, the total number of exhausted subscribers, as well as the Total Amount Claimed by Lifetime Subscriber and Total Amount Claimed by Annual Subscriber, will be displayed.