ALGORITHM

Pseudocode

// This program manages clients information for a garbage disposal company such as their information, jobs that are done and payments made.

DECLARATIONS

Constants

// Maximum record of clients that can be stored

Maxrecord = 100

// Maximum services that can be stored

Maxservice = 10

Records

// Stores data for one client

Clientrecord

Compname, contperson, address1, address2, phonenum – data type string

Endrecord

// Stores the services offered

Servicerecord

Service – data type string

Cost – data type real

Endrecord

// Stores the Job Entries

Jobrecord

Comp – data type integer

service – data type string

Freq – data type real

Date – data type string

Total – data type real

Endrecord

Paymentrecord

Compname , date – data type string

Amtpaid – data type real

Endrecord

Invoicerecord

Information[15] – data type jobrecord

Ttotal – data type real

Endrecord

Clientsservicerecord

id – data type integer

Amountofservices – data type integer

Clientservice [ maxservice] – data type string

Endrecord

Variables

Option, count, scount ,cscount,ncount,balonly– data type integer

Clientdata[maxrecord] – data typeclientrecord

Servicedata [maxrecord] – data type servicerecord

// This file contains information stored on the client such as the name , contact person, //address and phone number

Clientsservicerec – data type Clientsservicerecord

Clientservicedata [maxrecord] – data type Clientsservicerecord

Clientfile– data type text file

// This file contains the services that the company offers with the price

servicefile– data type text file

// This file stores all records of jobs done for a client

jobfile– data type text file

// This file stores the payment done by clients

Paymentfile – data type text file

//This file will send the jobs done and total cost of job to a document to create and print an //invoice

Invoicefile – data type textfile

Clientservicefile – data type text file

// Main Program

PROCESS

Call popservicedata( servicedata[], scount)

Repeat

Count 🡨 0

Balonly 🡨 0

Call popclientsdata(clientdata[], count)

Call popclientsservicedata( clientservicedata[],cscount)

Call menuoption(option)

Case option of

1 - call updateclientsacc( clientdata[], servicedata[], count, scount, clientservicedata[],cscount)

2 - display “View Clients Balance”

Call choosecompany( clientdata[ ], ncount, count)

call calclientsbal( clientdata[], count,clientservicedata,balonly,ncount )

3 - call viewclientsinfo( clientdata[], count,clientservicedata)

4 - call addclient(clientservicedata[], servicedata[],

cscount, scount)

5 - display”Program ending”

Otherwise - display “Invalid response”

Endcase

Until (option = 5)

STOP

//00000000000000000000000000000000000000000000000000000000000000000000000000000000000000

FUNCTION popservicedata()

// This function puts the information stored in the service file( name of service and cost) in an array so that the //user can choose the option when required

Parameters

Servicedata[ ] - data type servicerecord // passed by reference

scount – data type integer // passed by reference

Variables

Servicerec – data type service record

PROCESS

scount 🡨 0

Open servicefile as an input file

Read servicefile servicerec

While not eof (servicefile) do

Servicedata[scount] 🡨 servicerec

Read servicefile servicerec

Scount 🡨 scount +1

Endwhile

Close servicefile

STOP

//00000000000000000000000000000000000000000000000000000000000000000000000000000000000

FUNCTION popclientdata()

// This function puts all the clients information stored in the clients file in an array of records

Parameters

Clientdata[ ] – data type clientrecord //passed by reference

Count – data type integer //passed by reference

Variables

Clientrec – data type clientrecord

PROCESS

Count 🡨0

Open clientfile as an input file

Read clientfile, clientrec

While not eof (clientfile) do

Clientdata[count] 🡨 clientrec

Read clientfile clientrec

Count 🡨 count + 1

Endwhile

Close clientfile

STOP

//00000000000000000000000000000000000000000000000000000000000000000000000000000000000000

FUNCTION popclientservicedata()

// This function puts all the clients information stored in the clients service file in an array of records

Parameters

Clientservicedata[ ] – data type clientservicerecord //passed by reference

cscount – data type integer //passed by reference

Variables

Clientservicerec – data type clientservicerecord

i – data type integer

PROCESS

cscount 🡨0

Open clientservicefile as an input file

Read clientservicefile, clientservicerec

While not eof (clientservicefile) do

Clientservicedata[csount].id 🡨 clientservicerec.id

Clientservicedata[csount].amountofservice 🡨 clientservicerec.amountofservice

For i 🡨 0 to clientservicerec.amountofservice step 1 do

Clientservicedata[cscount].clientservice[i] 🡨 clientservicerec.clientservice[i]

endfor

Read clientfile clientrec

cscount 🡨 cscount + 1

Endwhile

Close clientfile

STOP

FUNCTION menuoption ()

// This function will display the menu options and accepts the choice of the user

Declarations

Parameters

Option - data type integer // variable parameter

PROCESS

Display ”00000000000000000000000000000000000000000000000000”

Display “0 Hans Services System 0”

Display “0 Welcome!! 0”

Display “0 0”

Display “0 1 . Update Clients account 0”

Display “0 2.View balance due by Clients 0”

Display “0 3. View Clients Information 0”

Display “0 4. Set up New Clients 0”

Display “0 5. Exit 0”

Display “00000000000000000000000000000000000000000000000000”

Display “Type response here :”

Read option

STOP

//00000000000000000000000000000000000000000000000000000000000000000000000000000000000000

FUNCTION updateclientsacc()

// This function gives the user an option to add a payment to a clients account or add a new job that was done .

Parameters

clientdata[]– data type clientrecord //pass by reference

servicedata[] – data type servicerecord //pass by reference

clientservicedata[] – data type clientservicerecord // pass by reference

count, scount, cscount – data type integer // pass by copy

Variables

Response – data type character

Ncount, endcount , endcount2, response2, balonly- data type integer

Balance- data type real

PROCESS

Balonly 🡨 1

Endcount 🡨 0

Display “ Update Clients Accont”

Repeat

Display “ Hans Services Update Client Menu”

Display ” Enter A – to add payment to client account”

Display ” B – to add new job entry “

Display ” C – to exit“

Read response

If ( response = ‘A’) or ( response = ‘a’) then

Display “ Adding Job ”

Call choosecompany( clientdata[ ], ncount, count)

Repeat

Display “ Enter 1 for regular services or 2 for special services done”

Read response2

If (response2 = 1)

Call jobentry1 (servicedata[], clientservicedata[ ], ncount ,cscount,scount)

Else

If(response2 = 2)

Call jobentry2 ( servicedata[ ], clientservicedata[],

clientdata[ ], ncount, scount)

Else

Display “ Invalid Response”

Endif

endif

Endcount2 🡨 Encount2 + 1

Until ( response = 1) or ( response = 2) or (Encount2 = 3)

Else

If ( response = ‘B’) or ( response = ‘b’) then

Display ”Adding payment”

Call choosecompany( clientdata[ ], ncount, count)

Call clientpayment( clientdata[ ], ncount)

Else

If ( response = ‘C’) or ( response = ‘c’) then

// program goes to main menu

else

Display “ Invaild response”

Endcount 🡨 Encount + 1

Endif

Endif

Endif

Until ( response = ‘C’) or ( response = ‘c’) or (Encount = 3)

STOP

//00000000000000000000000000000000000000000000000000000000000000000000000000000000000000

Function jobentry1()

Parameters

clientservicedata[ ] – data type clientservicerecord // pass by reference

clientdata[] – data type clientrecord // pass by reference

servicedata[] – data type servicerecord // pass by reference

ncount , cscount, scount– data type integer // passed by copy

Variable

Invoicerec – data type invoicerecord

Jobrec – data type jobrecord

Count1, count2, endcount, found, option, endcount– data type integer

Ttotal, frequency– data type real

Process

Count2 🡨 0

For count1 🡨 1 to clientservicedata[ncount].amountofservice step 1 do

Jobrec.comp 🡨 clientservicedata[ncount].id

Invoicerec.information[count].compname 🡨 clientservicedata[ncount].id

Found 🡨 0

While( found <> 1)

If ( clientservicedata[ncount].clientservice[count1]= servicedata[count2].service) then

repeat

Display “ How much “, servicedata[count2].service,”was collected”

Read frequency

Endcount 🡨 endount + 1

Until (frequency >= 0) or (endcount = 3)

If (encount = 3) then

// program ends and goes to main menu

endif

Jobrec.freq 🡨 frequency

Invoicerec.information[count1].freq 🡨jobrec.freq

Display”What is the date when it was performed”

Read jobrec.date

Invoive.information[count1].date 🡨 jobrec.date

Jobrec.total🡨 jobrec.freq \* servicedata[count2].cost

Invoicerec.information[count1].total 🡨 Jobrec.total

Ttotal 🡨ttotal + jobrec.total

Open jobfile for output

Output jobfile, jobrec

Close jobfile

Display “Job successfully added to file”

Endif

Count2 🡨 count2 + 1

Endfor

Endfor

Display “Total cost of job is $ “,ttotal

Invoicerec.ttotal 🡨 ttotal

Display “ Do you want an invoice printed 1 for yes and 2 for no”

Read option

If (option = 1) then

Display” Hans Services “

display”Company Name : ", clientdata[ncount].compname

display”Date Quantity Jobs Done Cost"

for j 🡨 0 to jobsdone step 1 do

display invoicerec.information[j].date

display" ", invoicerec.information[j].freq

display” ",invoicerec.information[j].service

display” $", invoicerec.information[j].total

endfor

display ”TOTAL AMOUNT DUE: $", ttotal

display “ HAVE A GOOD DAY "

endif

STOP

//0000000000000000000000000000000000000000000000000000000000000000000000000000000

FUNCTION jobentry2()

// This function adds jobs done to the jobfile

Parameters

Servicedata[ ] – data type servicerecord // passed by reference

Clientdata[ ] – data type clientrecord // passed by reference

clientservicedata[ ] – data type clientservicerecord // pass by reference

Ncount , scount– data type integer // passed by copy

Variables

Nservice, endcount, jobsdone, j, x – data type integer

Ttotal, cost, jobfreq – data type real

Jobrec – data type jobrecord

Invoicerec – data type invoicerecord

Option – data type integer

PROCESS

Endcount 🡨 0

Repeat

Display “ How many Jobs were done for “,clientdata[ncount].compname

Read Jobsdone

If (jobsdone >= 1) then

Endcount 🡨 3

else

Display “ Invalid response”

Endcount 🡨 Endcount + 1

Endif

If ( endcount = 3)

// the program goes back to menu

endif

Until (endcount = 3)

Endcount 🡨 0

Ttotal 🡨0

For j 🡨 1 to jobsdone step 1 do

Jobrec.comp 🡨 clientservicedata[ncount].id

Invoicerec.information[j].compname 🡨 clientservicedata[ncount].id

// Displaying the name of the service to choose from

Endcount 🡨 0

repeat

For x 🡨 1 to scount step 1 do

display x “---“ servicedata[x].service

endfor

display “ choose the number for the service performed”

read nservice

Endcount 🡨 endount + 1

until( nservice < scount) and (nservice>= 0) or (endcount = 3)

If (encount = 3) then

// program ends and goes to main menu

endif

display “Adding “servicedata[nservice].service

jobrec.service 🡨 servicedata[nservice].service

invoicerec.information[j].service 🡨 servicedata[nservice-1].service

cost 🡨 servicedata[nservice-1].cost

endcount 🡨 0

repeat

Display” What is the frequency”

Read jobfreq

If (jobfreq <= 0)

Endcount 🡨 endount + 1

If (encount = 3) then

// program ends and goes to main menu

Else

Endcount 🡨3

Endif

Endif

Until (jobfreq >= 0) or (endcount = 3)

Jobrec.freq 🡨 jobfreq

Invoicerec.information[j].freq 🡨jobrec.freq

Display”What is the date when it was performed”

Read jobrec.date

Invoive.information[j].date 🡨 jobrec.date

Jobrec.total🡨 jobrec.freq \* cost

Invoicerec.information[j].total 🡨 Jobrec.total

Ttotal 🡨ttotal + jobrec.total

Open jobfile for output

Output jobfile, jobrec

Close jobfile

Display “Job successfully added to file”

Endfor

Display “Total cost of job is $ “,ttotal

Invoicerec.ttotal 🡨 ttotal

Display “ Do you want an invoice printed 1 for yes and 2 for no”

Read option

If (option = 1) then

display” Hans Services “

display”Company Name : ", clientdata[ncount].compname

display”Date Quantity Jobs Done Cost"

for j 🡨 1 to jobsdone step 1 do

display invoicerec.information[j].date

display" ", invoicerec.information[j].freq

display” ",invoicerec.information[j].service

display” $", invoicerec.information[j].total

endfor

display ”TOTAL AMOUNT DUE: $", ttotal

display “ HAVE A GOOD DAY "

endif

STOP

//00000000000000000000000000000000000000000000000000000000000000000000000000000000000000

FUNCTION clientpayment()

// This function gets the payment a client is paying and adds it to the payment file

Parameters

Clientdata[ ] – data type clientrecord // passed by reference

Clientservicedata [] – data type clientservicerecord // passed by reference

Ncount, count– data type integer // passed by copy

Balonly – data type integer // passed by reference

Variables

Paymentrec – data type paymentrecord

Option – data type integer

Amountpaid, change,companybal,ramptpaid – data type real

PROCESS

Balonly = 1

Paymentrec.compname 🡨 Clientdata[ncount].compname

Call calclientbal ( clientdata, clientservicedata, count, balonly,ncount, companybal)

Display” When was this paid“

Read Paymentrec.date

Display” How much is being paid“

Read amountpaid

If ( amountpaid >= 1) then

Ramptpaid 🡨 amountpaid

Change 🡨 companybal – amountpaid

If( change < 0)

Change = change /-1

Display “ Your change is”, change

Amountpaaid = companybal

Else

Display “ You owe”, change

Change 🡨 0

Endif

Paymentrec.amtpaid 🡨 amountpaid

Open paymentfile for output

Output paymentfile, paymentrec

Close paymentfile

Display “Payment successfully added to file”

Else

Display ” Invalid response”

Display ” How much is being paid“

Read amountpaid

If ( amountpaid >= 1) then

Ramptpaid 🡨 amountpaid

Change 🡨 companybal – amountpaid

If( change < 0)

Change = change /-1

Display “ Your change is”, change

Amountpaaid = companybal

Else

Display “ You owe”, change

Change 🡨 0

Endif

Paymentrec.amtpaid 🡨 amountpaid

Open paymentfile for output

Output paymentfile, paymentrec

Close paymentfile

Display “Payment successfully added to file”

Else

// Program ends

Endif

endif

Display “ Do you want a receipt 1 for yes and 2 for no”

Read option

If (option = 1) then

display" Hans Services "

display"Date of Payment : ", paymentrec.date

display"Company : ", paymentrec.compname

display"TOTAL AMOUNT PAID $ : ", ramtpaid

display “ CHANGE : ”, change

display" Have a good day "

endif

STOP

//00000000000000000000000000000000000000000000000000000000000000000000000000000000000000

FUNCTION calclientsbal()

//This function displays the balance on a clients account by finding the difference between all the payments //made and the total cost of the jobs

Parameters

clientdata[] – data type clientrecord // pass by reference

clientservicedata[] – data type clientservicerecord // pass by reference

companybal – data type integer // pass by reference

count, balonly, ncount– data type integer // pass by copy

variables

Jobrec – data type jobrecord

Paymentrec – data type paymentrecord

totalcostjob, totalpaymentjob, compowe – data type real

PROCESS

Totalcostjob 🡨 0

Open jobfile as an input file

Read jobfile, jobrec

While not eof (jobfile) do

If (jobrec.comp = clientservicedata[ncount].id) then

Totalcostjob 🡨 totalcostjob + jobrec.total

endif

Read jobfile, jobrec

Endwhile

Close jobfile

Totalpaymentjob 🡨 0

Open paymentfileas an input file

Read paymentfile, paymentrec

While not eof (paymentfile) do

If (paymentfile.compname = clientdata[ncount].compname) then

Totalpaymentjob🡨 totalpaymentjob + paymentfile.amtpaid

endif

Read paymentfile paymentrec

endwhile

close paymentfile

compowe🡨 Totalcostjob - Totalpaymentjob

if (balonly = 0)

If (compowe < 0) then

Compowe 🡸 compowe / -1

Endif

Display “ Hans Services System”

Display “ Clients Balance Information Report”

Display “ Name of Client”,clientdata[ncount].compname

Display “ Amount Owed $”,compowe

Endif

Companybal 🡨 compowe

STOP

//00000000000000000000000000000000000000000000000000000000000000000000000000000000000000

FUNCTION viewclientsinfo

//This function allows the user to view the information stored on a client such as the address, contact person //and phone number

Parameters

clientdata[] – data type clientrecord // pass by reference

clientservicedata[] – data type clientservicerecord // pass by reference

count – data type integer // pass by copy

Variables

Ncount, x– data type integer

PROCESS

Call choosecompany( clientdata[ ], ncount, count)

Display “Company Name- “,clientdata[ncount].compname

Display “ Company ID number “, clientservicedata[ncount].id

Display” Contact person - ” , clientdata[ncount].contperson

Display” Address – “,clientdata[ncount].address1

Display” “ , clientdata[ncount].address2

Display” Phone number - ” , clientdata[ncount].phonenum

Display” amount of contracted services “, clientservicedata[ncount].amountofservice

Display “ The services are : “

For x 🡨 1 to clientservicedata[ncount].amountofservice step 1 do

Display clientservicedata[ncount].clientservice[x]

endfor

STOP

//00000000000000000000000000000000000000000000000000000000000000000000000000000000000000

FUNCTION choosecompany()

//This function displays the list of the company names and puts the clients information in an array so that when //the user chooses the company from the list the information can be retrieved from the array

Parameters

Clientdata[ ] – data type clientrecord // passed by reference

Ncount data type integer // passed by reference

count– data type integer // passed by copy

Variables

comresponse, j – data type integer

PROCESS

Repeat

// Displays the company names to choose from

For j 🡨 1 to count step 1 do

Display j+1 “ ---“ clientdata[ j ].compname

endfor

Display” Which of these companies do you want to process“

Display” Choose the number“

Read comresponse

If (comresponse <= count) and ( comresponse>= 1) then

Ncount 🡨comresponse

Else

Displayed “Invalid response choose between 1 and “,count +1

endif

Until (comresponse <= count) and ( comresponse > 0)

Display “Processing ”,clientdata[ncount].compname

STOP

//00000000000000000000000000000000000000000000000000000000000000000000000000000000000000

FUNCTION call addclient()

// This function allows the user to enter a new client and save this information in the client file

Parameters

Clientservicedata – data type clientservicerecord // pass by reference

Servicedata – data type servicerecord // pass by reference

scount, cscount – data type integer // pass by copy

variables

Clientrec – data type clientrecord

Working – data type integer

PROCESS

Display ”What is the name of the new company”

read clientrec.compname

Display ”What is the name of the contact person for new company”

read clientrec.contperson

Display ”What is the 1st line of address for the new company”

Read clientrec.address1

Display ” What is the 2nd line of address for the new company”

read clientrec.address2

Display ” What is the phonenumber for the new company”

read clientrec.phonenum

Call chooseservices ( clientservicedata, servicedata, cscount, scount,working)

If (working = 0)

Open clientfile for output

Output clientfile, clientrec

Close clientfile

Display “Client successfully added”

endif

STOP

FUNCTION call chooseservices

Parameter

Servicedata[ ] – data type servicerecord // passed by reference

clientservicedata[ ] – data type clientservicerecord // pass by reference

cscount , scount– data type integer // passed by copy

working – data type integer // passed by reference

Variables

Nservice, x, k,endcount , amount– data type integer

Clientservicerec – data type clientservicerecord

Process

Working 🡨 0

Open clientservicefile for output

Clientservicesrec.id🡨 (clientservicesdata[cscount].id + 1)

Output clientservicefile Clientservicesrec.id

encount 🡨 0

repeat

Display “ Amount of services :”

Read amount

Endcount 🡨 endount + 1

Until (amount >= 1) or (endcount = 3)

If (encount = 3) then

Working 🡨 working + 1

// Program returns to Addclient function

endif

clientservicerec.amountofservice 🡨 amount

Output clientservicefile clientservicerec.amountofservice

encount 🡨 0

For k 🡨 1 to amount step 1 do

repeat

For x 🡨 1 to scount step 1 do

display x “---“ servicedata[x].service

endfor

display “ choose the number for the service performed”

read nservice

Endcount 🡨 endount + 1

until( nservice < scount) and (nservice>= 0) or (endcount = 3)

If (encount = 3) then

Working 🡨 working + 1

// function ends and goes to Add Client function

Else

Display “ Adding”, Servicedata[nservice].service, “ to file”

clientservicerec.clientservice[k] 🡨 Servicedata[nservice].service

output clientservicefile clientservicerec.clientservice[k]

endif

endfor

Close clientservicefile

STOP