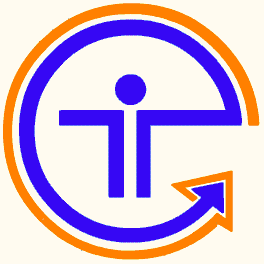
CENTRO DE ENSEÑANZA TECNICA INDUSTRIAL



By:

Daniel Arturo Fernández Raygoza #13310091

Edgar Jesús Robles Najar #12110275

Date:

13 of June of 2016

Final Project:

“Expert system for personal computer hardware fail diagnostic”

**Expert system for personal computer hardware fail diagnostic**

**Justification:**

In this practice we are going to make an expert system to filter data using multiple parameters like the age, socioeconomic position and salary. The system must storage all the data in a file so is accessible anytime. Also the users can add new persons to the database and delete the whole database allowing to create a new database when needed. Also it’s possible to seek for users by it’s name or economic class, sort the database by economic positions and if you enter a user with a no matching salary with socioeconomic class the program reassigns it to the right socioeconomic class.

We are going to use CLIPS to create the expert system algorithm.

**Code:**

;;============================================================

;;Por: Daniel Arturo Fernandez Raygoza #13310091

;;============================================================

(deffunction ask-number ()

(bind ?menu (read))

(assert (menu ?menu))

)

(deftemplate tempBusca

(slot busca

(type STRING)

(default "Not\_assigned")

)

)

(deffunction ask-name ()

(printout t "Name to seek: ")

(bind ?seekN(readline))

(assert

(tempBusca

(busca ?seekN)

)

)

)

(deffunction ask-position ()

(printout t "ecPosition to seek: ")

(bind ?ecPosition(readline))

(while (and(neq ?ecPosition "poor-class") (neq ?ecPosition "medium-class") (neq ?ecPosition "upper-class"))

(printout t "ecPosition to seek: ")

(bind ?ecPosition(readline))

)

(assert

(tempBusca

(busca ?ecPosition)

)

)

)

(deftemplate person

(slot name

(type STRING)

(default "Not\_assigned")

)

(slot coupleName

(type STRING)

(default "Not\_assigned")

)

(slot ecPosition

(type STRING)

(default "Not\_assigned")

)

(slot salary

(type INTEGER)

(default 0)

)

(slot age

(type INTEGER)

(default 40)

)

)

(deftemplate auxPerson

(slot name

(type STRING)

(default "Not\_assigned")

)

(slot coupleName

(type STRING)

(default "Not\_assigned")

)

(slot ecPosition

(type STRING)

(default "Not\_assigned")

)

(slot salary

(type INTEGER)

(default 0)

)

(slot age

(type INTEGER)

(default 40)

)

)

(deffunction fillPerson()

(printout t "Name: ")

(bind ?name(readline))

(printout t "coupleName: ")

(bind ?coupleName(readline))

(printout t "ecPosition: ")

(bind ?ecPosition(readline))

(while (and(neq ?ecPosition "poor-class") (neq ?ecPosition "medium-class") (neq ?ecPosition "upper-class"))

(printout t "ecPosition: ")

(bind ?ecPosition(readline))

)

(printout t "Salary: ")

(bind ?salary(read))

(while (not(numberp ?salary))

(printout t "Salary: ")

(bind ?salary(read))

)

(if (<= ?salary 5000)

then

(printout t "User moved to poor-class due to low salary" crlf)

(bind ?ecPosition "poor-class")

)

(if (and(> ?salary 5000)(< ?salary 10000))

then

(printout t "User moved to medium-class due to intermediate salary" crlf)

(bind ?ecPosition "medium-class")

)

(if (>= ?salary 10000)

then

(printout t "User moved to high-class due to high salary" crlf)

(bind ?ecPosition "upper-class")

)

(printout t "Age: ")

(bind ?age(read))

(while (not(numberp ?age))

(printout t "Age: ")

(bind ?age(read))

)

(assert

(person

(name ?name)

(coupleName ?coupleName)

(ecPosition ?ecPosition)

(salary ?salary)

(age ?age)

)

)

)

(defrule ask-menu

(declare (salience 12))

(not(inicial-fact ?))

=>

(load-facts memoria)

(printout t crlf crlf)

(printout t "-----Expert systems practice-----" crlf)

(printout t crlf crlf)

(printout t "1- Show screen with all people of 60 years." crlf)

(printout t "2- Show screen salary and name of all people of 40 years." crlf)

(printout t "3- Show all database." crlf)

(printout t "4- Show the name of the people with a poor economic position." crlf)

(printout t "5- From poor class couple implements one vector to work memory <f\_data <couple> Poor-position>" crlf)

(printout t "6- If the people sold the account erase the name from the work memory" crlf)

(printout t "7- Add new element to the database." crlf)

(printout t "8- Delete entire database." crlf)

(printout t "9- Sort database from upper-class to poor-class." crlf)

(printout t "10- Sort database from poor-class to upper-class." crlf)

(printout t "11- Sort database by ascent salary." crlf)

(printout t "12- Sort database by descent salary." crlf)

(printout t "13- Seek user by name." crlf)

(printout t "14- Seek user by ecPositiion." crlf)

(printout t "15- Exit program." crlf)

(printout t crlf "What do you want to do?" crlf)

(ask-number)

)

(defrule option1 "all60"

(declare (salience 10))

(menu 1)

(person

(name ?name)

(coupleName ?)

(ecPosition ?)

(salary ?)

(age 60)

)

=>

(printout t "Name: "?name crlf)

)

(defrule option2 "salary-name-40"

(declare (salience 10))

(menu 2)

(person

(name ?name)

(coupleName ?)

(ecPosition ?)

(salary ?salary)

(age 40)

)

=>

(printout t crlf "Name: "?name)

(printout t " Salary: "?salary crlf)

)

(defrule option3 "all-data"

(declare (salience 10))

(menu 3)

(person

(name ?name)

(coupleName ?coupleName)

(ecPosition ?ecPosition)

(salary ?salary)

(age ?age)

)

=>

(printout t crlf "Name: "?name)

(printout t crlf "coupleName: "?coupleName)

(printout t crlf "ecPosition: "?ecPosition)

(printout t crlf "Salary: "?salary)

(printout t crlf "Age: "?age crlf)

)

(defrule option4 "poor-class"

(declare (salience 10))

(menu 4)

(person

(name ?name)

(coupleName ?)

(ecPosition "poor-class")

(salary ?)

(age ?)

)

=>

(printout t "Name: "?name crlf)

)

(defrule option5 "vector-memory"

(declare (salience 10))

(menu 5)

=>

(printout t "This function is not implemented because the lack of data in class. Please explain how to use CLIPS." crlf)

)

(defrule option6 "sold-account"

(declare (salience 100))

(menu 6)

=>

(printout t "This function is not implemented because the lack of data in class. Please explain how to use CLIPS." crlf)

)

(defrule option7 "create-database"

(declare (salience 10))

(menu 7)

?object<-(initial-fact)

=>

(fillPerson)

(retract ?object)

(save-facts Memoria visible person)

)

(defrule option8 "deleteData"

(declare (salience 10))

(menu 8)

?object<-(person)

=>

(printout t "Deleting database..." crlf crlf)

(retract ?object)

(save-facts Memoria visible person)

)

(defrule option9\_1 "sort Upper to Poor act1"

(declare (salience 18))

(menu 9)

(person

(name ?name1)

(coupleName ?coupleName1)

(ecPosition ?ecPosition1)

(salary ?salary1)

(age ?age1)

)

=>

(assert

(auxPerson

(name ?name1)

(coupleName ?coupleName1)

(ecPosition ?ecPosition1)

(salary ?salary1)

(age ?age1)

)

)

(save-facts temp visible auxPerson)

)

(defrule option9\_2 "sort Upper to Poor act2"

(declare (salience 17))

(menu 9)

(not( nodel ?))

?object1<-(person)

=>

(retract ?object1)

(save-facts Memoria visible person)

)

(defrule option9\_3 "sort Upper to Poor act3"

(declare (salience 16))

(menu 9)

(auxPerson

(name ?name)

(coupleName ?coupleName)

(ecPosition ?ecPosition)

(salary ?salary)

(age ?age)

)

=>

(assert (nodel "check"))

(if(eq ?ecPosition "poor-class")

then

(assert

(person

(name ?name)

(coupleName ?coupleName)

(ecPosition ?ecPosition)

(salary ?salary)

(age ?age)

)

)

(save-facts Memoria visible person)

)

)

(defrule option9\_4 "sort Upper to Poor act4"

(declare (salience 15))

(menu 9)

(auxPerson

(name ?name)

(coupleName ?coupleName)

(ecPosition ?ecPosition)

(salary ?salary)

(age ?age)

)

=>

(assert (nodel "check"))

(if(eq ?ecPosition "medium-class")

then

(assert

(person

(name ?name)

(coupleName ?coupleName)

(ecPosition ?ecPosition)

(salary ?salary)

(age ?age)

)

)

(save-facts Memoria visible person)

)

)

(defrule option9\_5 "sort Upper to Poor act5"

(declare (salience 14))

(menu 9)

(auxPerson

(name ?name)

(coupleName ?coupleName)

(ecPosition ?ecPosition)

(salary ?salary)

(age ?age)

)

=>

(assert (nodel "check"))

(if(eq ?ecPosition "upper-class")

then

(assert

(person

(name ?name)

(coupleName ?coupleName)

(ecPosition ?ecPosition)

(salary ?salary)

(age ?age)

)

)

(save-facts Memoria visible person)

)

)

(defrule option9\_6 "sort Upper to Poor act6"

(declare (salience 13))

(menu 9)

?object2<-(auxPerson)

=>

(retract ?object2)

(save-facts temp visible auxPerson)

(printout t "Sorting database from upper-class to poor-class..." crlf)

)

(defrule option10\_1 "sort Poor to Upper act1"

(declare (salience 18))

(menu 10)

(person

(name ?name1)

(coupleName ?coupleName1)

(ecPosition ?ecPosition1)

(salary ?salary1)

(age ?age1)

)

=>

(assert

(auxPerson

(name ?name1)

(coupleName ?coupleName1)

(ecPosition ?ecPosition1)

(salary ?salary1)

(age ?age1)

)

)

(save-facts temp visible auxPerson)

)

(defrule option10\_2 "sort Poor to Upper act2"

(declare (salience 17))

(menu 10)

(not( nodel2 ?))

?object1<-(person)

=>

(retract ?object1)

(save-facts Memoria visible person)

)

(defrule option10\_3 "sort Poor to Upper act3"

(declare (salience 16))

(menu 10)

(auxPerson

(name ?name)

(coupleName ?coupleName)

(ecPosition ?ecPosition)

(salary ?salary)

(age ?age)

)

=>

(assert (nodel2 "check"))

(if(eq ?ecPosition "upper-class")

then

(assert

(person

(name ?name)

(coupleName ?coupleName)

(ecPosition ?ecPosition)

(salary ?salary)

(age ?age)

)

)

(save-facts Memoria visible person)

)

)

(defrule option10\_4 "sort Poor to Upper act4"

(declare (salience 15))

(menu 10)

(auxPerson

(name ?name)

(coupleName ?coupleName)

(ecPosition ?ecPosition)

(salary ?salary)

(age ?age)

)

=>

(assert (nodel2 "check"))

(if(eq ?ecPosition "medium-class")

then

(assert

(person

(name ?name)

(coupleName ?coupleName)

(ecPosition ?ecPosition)

(salary ?salary)

(age ?age)

)

)

(save-facts Memoria visible person)

)

)

(defrule option10\_5 "sort Poor to Upper act5"

(declare (salience 14))

(menu 10)

(auxPerson

(name ?name)

(coupleName ?coupleName)

(ecPosition ?ecPosition)

(salary ?salary)

(age ?age)

)

=>

(assert (nodel2 "check"))

(if(eq ?ecPosition "poor-class")

then

(assert

(person

(name ?name)

(coupleName ?coupleName)

(ecPosition ?ecPosition)

(salary ?salary)

(age ?age)

)

)

(save-facts Memoria visible person)

)

)

(defrule option10\_6 "sort Poor to Upper act6"

(declare (salience 13))

(menu 10)

?object2<-(auxPerson)

=>

(retract ?object2)

(save-facts temp visible auxPerson)

(printout t "Sorting database from poor-class to upper-class..." crlf)

)

(defrule option13 "seekName call"

(declare (salience 14))

(menu 13)

=>

(ask-name)

)

(defrule Seek\_names "seekName print"

(declare (salience 13))

(menu 13)

(person

(name ?name)

(coupleName ?coupleName)

(ecPosition ?ecPosition)

(salary ?salary)

(age ?age)

)

(tempBusca

(busca ?busca)

)

=>

;(printout t ?busca " - " ?name crlf)

(if(eq ?name ?busca)

then

(printout t crlf "Name: "?name)

(printout t crlf "coupleName: "?coupleName)

(printout t crlf "ecPosition: "?ecPosition)

(printout t crlf "Salary: "?salary)

(printout t crlf "Age: "?age crlf)

(assert (notfound 1))

)

)

(defrule Seek\_names\_notfound "seekName notfound"

(declare (salience 12))

(menu 13)

(not(notfouund 1))

(not (notfound ?))

=>

(printout t crlf "There's no user with this name in database. Do you want to add it[y/n]?." crlf)

(bind ?obt(readline))

(while (and(neq ?obt "y")(neq ?obt "n"))

(printout t crlf "There's no user with this name in database. Do you want to add it[y/n]?." crlf)

(bind ?obt(readline))

)

(if (eq ?obt "y")

then

(assert (menu 7))

)

)

(defrule option14 "seekClass call"

(declare (salience 14))

(menu 14)

=>

(ask-position)

)

(defrule Seek\_class "seekClass print"

(declare (salience 13))

(menu 14)

(person

(name ?name)

(coupleName ?coupleName)

(ecPosition ?ecPosition)

(salary ?salary)

(age ?age)

)

(tempBusca

(busca ?busca)

)

=>

;(printout t ?busca " - " ?ecPosition crlf)

(if(eq ?ecPosition ?busca)

then

(printout t crlf "Name: "?name)

(printout t crlf "coupleName: "?coupleName)

(printout t crlf "ecPosition: "?ecPosition)

(printout t crlf "Salary: "?salary)

(printout t crlf "Age: "?age crlf)

)

)

(defrule option15 "exit"

(declare (salience 10))

(menu 15)

=>

(printout t "Goodbye..." crlf crlf)

)

(defrule reboot "restart-app"

(declare (salience -1000))

(not (menu 15))

(menu ?)

=>

(reset)

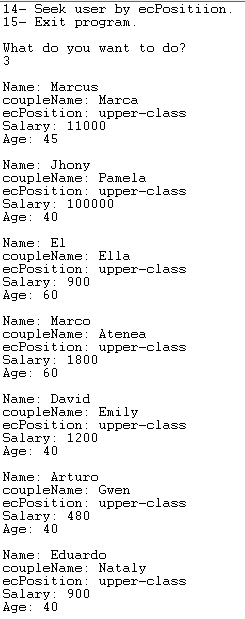
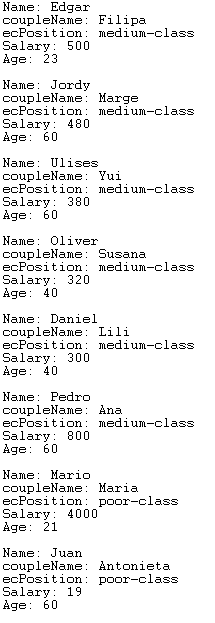
(run)

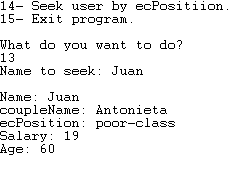
;(system "cls") ;esta funcion no funciona bajo windows. Abre una ventana CMD, la borra y la cierra.

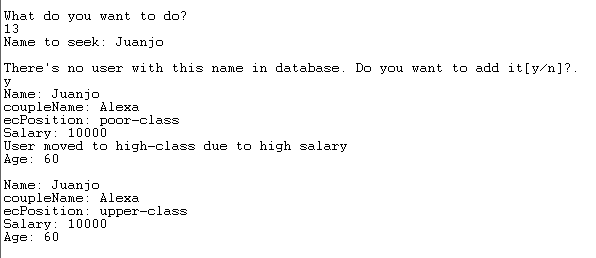
)

**Screen shots:**

****

****

****

****

**Conclusion:**

The expert system is capable of manipulate a database and use it to execute different actions with them, like filtering, searching, sorting and other functions.

With this program I have learned more about how to cycle rules and execute the desired actions in the desired order. But still being difficult to understand how CLIPS runs and how to implement certain commands that are available in other platforms.