Πίνακας περιεχομένων

1 Εισαγωγή

Για την υλοποίηση της εργασίας χρησιμοποιήθηκαν το πρόγραμμα Notepad++ και το περιβάλλον GNU Prolog. Το πρόγραμμα Notepad++ χρησιμοποιήθηκε για τη σύνταξη του κώδικα, ενώ το περιβάλλον GNU Prolog χρησιμοποιήθηκε ως compiler για το τρέξιμο και τον έλεγχο του προγράμματος, καθώς και για την παραγωγή των ενδεικτικών εκτυπώσεων από τη χρήση του. Έβαλα σε ξεχωριστά αρχεία τα γεγονότα (projectFacts.pl) και τους κανόνες (projectRules.pl) για καλύτερη διαχείρισή τους.

Το συγκεκριμένο σύστημα μπορεί να δεχθεί ερωτήματα πολύ συγκεκριμένου σκοπού όπως για παράδειγμα αν μία τοποθεσία είναι γενικά προσβάσιμη ή μπορεί να εκτυπώσει μόνο τα αξιοθέατα ενός προορισμού και τίποτε άλλο. Επίσης μπορεί να επιστρέψει πολλών ειδών πληροφορίες με μόνο ένα ερώτημα, για παράδειγμα όλα τα δρομολόγια αναλυτικά που ικανοποιούν τη πρόσβαση σε έναν προορισμό ή όλες τις πληροφορίες (αξιοθέατα, καταλύματα κλπ) για έναν προορισμό. Όσο αφορά στα δρομολόγια, σαν αφετηρία θεωρεί πάντα την Αθήνα.

Συνημμένα έχω βάλει και τον φάκελο Screenshots που περιέχει ενδεικτικές εκτυπώσεις από την εκτέλεση ερωτημάτων. Οι εκτυπώσεις αυτές βρίσκονται και στην παράγραφο 5 του Εγχειριδίου Χρήστη.

2 Γεγονότα (Facts)

Για τη δημιουργία των facts της γνωσιακής βάσης, τα στοιχεία των οποίων συμπεριλαμβάνονται και στο συνημμένο αρχείο excel με όνομα «ProjectData.xlsx», χρησιμοποιήθηκαν δύο βασικά κατηγορήματα, το destinationData/5 και το route/6.

Κατηγόρημα destinationData/5:

Το κατηγόρημα αυτό περιέχει όλες τις πληροφορίες που αφορούν σε έναν προορισμό. Αν destinationData(k1, k2, k3, k4, k5) τότε τα ορίσματα φαίνονται παρακάτω:

k1: Σταθερά που δηλώνει το όνομα του προορισμού. Παράδειγμα: chios

k2: Λίστα με μία ή περισσότερες σταθερές που δηλώνει το/τα είδος/είδη του προορισμού. Παράδειγμα: [mountain, island]

k3: Κατηγόρημα sights/1 με όρισμα μία λίστα μέσα στην οποία υπάρχουν όλες οι πληροφορίες για τα αξιοθέατα του προορισμού. Η αρχική λίστα περιέχει υπολίστες καθεμιά από τις οποίες έχει ως head το είδος αξιοθέατων και ως πρώτο όρισμα μετά το head, περιέχει εντός μιας τρίτου επιπέδου λίστας ονομαστικά κάθε αξιοθέατο που αντιστοιχεί στον συγκεκριμένο προορισμό και στο συγκεκριμένο είδος αξιοθέατου.
Παράδειγμα: sights([[museums,[agios_loukas]], [archeology,[loukas_cave]]])

k4: Κατηγόρημα accommodations/1 με όρισμα μία λίστα μέσα στην οποία υπάρχουν όλες οι πληροφορίες για τα καταλύματα ενός προορισμού. Η αρχική λίστα περιέχει υπολίστες καθεμιά από τις οποίες έχει ως head το είδος καταλύματος και ως πρώτο όρισμα μετά το head, περιέχει μια τρίτου επιπέδου λίστα. Εντός της τελευταίας λίστας περιέχονται άλλες λίστες κάθε μια από τις οποίες περιέχει δύο ορίσματα που δείχνουν το είδος δωματίου και το κόστος του δωματίου. Παράδειγμα: accomodations([

```
        [airbnb, [
                [studio, 45], [apartment, 60]
        ]],
        [hostel, [
                [twoBed, 35], [fourBed, 30], [sixBed, 25]
        ]]
])
```

k5: Κατηγόρημα publicUtilityServices/1 με όρισμα μία λίστα μέσα στην οποία υπάρχουν σταθερές που δηλώνουν τα είδη υπηρεσιών κοινής ωφελείας που υπάρχουν στον προορισμό. Παράδειγμα: publicUtilityServices([pharmacy, port_authority])

Ο λόγος που χρησιμοποίησα λίστες εδώ είναι διότι ένας τυχαίος προορισμός δεν μπορώ να ξέρω από πριν αν έχει τον ίδιο αριθμό για παράδειγμα ειδών αξιοθέατων ή ειδών καταλυμάτων κλπ. Σε όποιο σημείο το πλήθος των στοιχείων που καταχωρούσα παρουσίαζε διαφοροποιήσεις από προορισμό σε προορισμό, τότε έβαζα αυτά τα στοιχεία σε λίστα για να μπορώ να έχω πρόσβαση σε αυτά με χρήση του κατηγορήματος member/2.

Επίσης, χωρίς τη χρήση λιστών το πλήθων των γεγονότων θα μεγάλωνε πολύ, διότι για παράδειγμα για έναν συγκεκριμένο προορισμό αν είχε τέσσερα είδη καταλυμάτων και για καθένα από αυτά τέσσερα είδη δωματίων, τότε θα έπρεπε να φτιάξω 4x4=16 γεγονότα μόνο για τα καταλύματα ενός μόνο προορισμού. Το πρόγραμμα όταν θα ξεκινήσει να ασχολείται με έναν προορισμό θα έκανε πολύ περισσότερες συγκρίσεις/προσπελάσεις μέχρι να βρει όλα τα καταλύματα-δωμάτια ενός προορισμού, ενώ με τις λίστες είναι όλα μαζεμένα στο κατηγόρημα που αφορά στον προορισμό αυτό.

Κατηγόρημα route/6:

Το κατηγόρημα αυτό περιέχει όλες τις πληροφορίες που αφορούν σε μία διαδρομή. Αν route(k1, k2, k3, k4, k5, k6) τότε τα ορίσματα φαίνονται παρακάτω:

k1: Σταθερά που δηλώνει το είδος του μεταφορικού μέσου. Παράδειγμα: airplane

k2: Σταθερά που δηλώνει την αφετηρία της διαδρομής. Παράδειγμα: athens

k3: Σταθερά που δηλώνει τον προορισμό της διαδρομής. Παράδειγμα: thessaloniki

k4: Σταθερά που δηλώνει το κόστος του ταξιδιού σε ευρώ. Παράδειγμα: 15

k5: Σταθερά που δηλώνει την ώρα του ταξιδιού σε ώρες. Παράδειγμα: 3.2

k6: Σταθερά που δηλώνει την απόσταση της διαδρομής σε χιλιόμετρα. Παράδειγμα: 153

3 Κανόνες (Rules) - Κεφαλές

α. directlyConnected(Start, Finish, TransportType)

Ικανοποιείται όταν υπάρχει απευθείας σύνδεση από μια τοποθεσία σε μια άλλη με τη χρήση συγκεκριμένου μεταφορικού μέσου.

β. connected(Start, Finish, TransportList)

Ικανοποιείται όταν υπάρχει σύνδεση από μια τοποθεσία σε μια άλλη άμεσα ή μέσω ενδιάμεσων σταθμών, με τη χρήση οποιουδήποτε μεταφορικού μέσω δοθεί στη λίστα.

γ. connectedAndMatchChoices(Start, Finish, CurrentCost, CurrentTime, CurrentDistance,
CurrentTransportType, TransportChoiceList,
RouteCostChoice, RouteTimeChoice,
RouteDistanceChoice)

Ικανοποιείται αν υπάρχει πρόσβαση σε μια τοποθεσία άμεσα ή έμμεσα μέσω ενδιάμεσων σταθμών και ταυτόχρονα τα συνολικά κόστη, χρόνος, απόσταση της διαδρομής βρίσκονται εντός συγκεκριμένων ορίων. Επίσης, εκτυπώνει πληροφορίες για τα δρομολόγια (συνολικά και ενδιάμεσα κόστη, χρόνους, αποστάσεις) που ικανοποιούν την πρόσβαση στην τοποθεσία.

δ. matchAllChoices(LocationChoice, SightChoice, AccommodationChoice, RoomChoice,
RoomCostChoice, TransportChoiceList, RouteCostChoice,
RouteTimeChoice, RouteDistanceChoice)

Ικανοποιείται αν υπάρχει/ουν τοποθεσία/ες που διαθέτει όλους τους περιορισμούς που προσθέτει ο χρήστης (είδος τοποθεσίας, είδος/είδη μεταφορικών μέσων κλπ), και εκτυπώνει όλες τις πληροφορίες που αφορούν σε αυτές τις τοποθεσίες.

ε. matchLocation(LocationChoice, DestinationName)

Ικανοποιείται αν μια τοποθεσία είναι ενός συγκεκριμένου είδους (πχ mountain).

στ. matchSight(SightChoice, DestinationName)

Ικανοποιείται αν μια τοποθεσία διαθέτει ένα είδος αξιοθέατου.

ζ. matchAccommodation(AccommodationChoice, RoomChoice, CostChoice,
DestinationName)

Ικανοποιείται αν μια τοποθεσία διαθέτει ένα είδος καταλύματος, δωματίου βάσει περιορισμών σε τιμή

η. matchRoute(TransportChoiceList, RouteCostChoice, RouteTimeChoice, RouteDistanceChoice, DestinationName)

Ικανοποιείται αν υπάρχει πρόσβαση σε μια τοποθεσία άμεσα ή έμμεσα μέσω ενδιάμεσων σταθμών και ταυτόχρονα τα συνολικά κόστη, χρόνος, απόσταση της διαδρομής βρίσκονται εντός συγκεκριμένων ορίων. Επίσης, εκτυπώνει πληροφορίες για τα δρομολόγια (συνολικά και ενδιάμεσα κόστη, χρόνους, αποστάσεις) που ικανοποιούν την πρόσβαση στην τοποθεσία. Στην ουσία καλεί την connectedAndMatchChoices(), χρησιμοποιώντας συγκεκριμένες τιμές σε κάποια ορίσματα ώστε η connectedAndMatchChoices() να λειτουργήσει με τον τρόπο που πρέπει.

θ. showFullInfo(DestinationName)

Ικανοποιείται αν υπάρχει μια συγκεκριμένη τοποθεσία και εκτυπώνει όλες της πληροφορίες που αφορούν σε αυτήν.

ι. showSightInfo(DestinationName)

Ικανοποιείται αν υπάρχει μια συγκεκριμένη τοποθεσία και εκτυπώνει όλα τα αξιοθέατα που αφορούν σε αυτήν.

ια. showAccommodationInfo(DestinationName)

Ικανοποιείται αν υπάρχει μια συγκεκριμένη τοποθεσία και εκτυπώνει όλες τις πληροφορίες για τα καταλύματα που αφορούν σε αυτήν.

ιβ. showPublicUtilityServices(DestinationName)

Ικανοποιείται αν υπάρχει μια συγκεκριμένη τοποθεσία και εκτυπώνει όλα τα αξιοθέτα που αφορούν σε αυτήν.

ιγ. showSinglePathInfo(Start, Finish, TransportType)

Ικανοποιείται αν υπάρχει απευθείας σύνδεση από μια τοποθεσία σε μια άλλη με χρήση ενός μεταφορικού μέσου, και εκτυπώνει πληροφορίες (κόστος, χρόνος, απόσταση) του ταξιδιού.

4 **Κανόνες (Rules) - Λειτουργία**

α. **connected**

```
directlyConnected(Start, Finish, TransportType):-
    route(TransportType, Start, Finish, _, _, _).

connected(Start, Finish, TransportList):-
    directlyConnected(Start, Finish, TransportType),
    member(TransportType, TransportList).

connected(Start, Finish, TransportList):-
    directlyConnected(Start, Middle, TransportType),
    member(TransportType, TransportList),
    connected(Middle, Finish, TransportList).
```

/*Τα παραπάνω λειτουργούν ως σύνολο για να μπορέσουμε μέσω αναδρομής να βρούμε αν όντως δύο τοποθεσίες είναι συνδεμένες μέσω ενός ή παραπάνω διαδρομών. Η directlyConnected είναι η βάση και η connected είναι αυτή που δημιουργεί την αναδρομή. Η connected χρειάζεται για να λειτουργήσει σωστά, ως είσοδο στο τρίτο όρισμά της, μια λίστα από ένα ή περισσότερα μεταφορικά μέσα, και το κατηγόρημα member/2 παίρνει θέση μέσα σε κάθε connected και μετά από την κλήση των directlyConnected για να ταυτοποιεί αν το TransportType που διάλεξε η directlyConnected βρίσκεται στη λίστα περιορισμών που δόθηκε στην connected.*/

β. **connectedAndMatchChoices**

```
directlyConnected(Start, Finish, TransportType):-
        route(TransportType, Start, Finish, _, _, _).
```

/*Η directlyConnected χρησιμοποιείται και εδώ για την επίτευξη της αναδρομικότητας*/

```
connectedAndMatchChoices(Start, Finish, _, _, _, _, TransportChoiceList, _, _, _):-
    Start \== athens,
    directlyConnected(Start, Finish, TransportType),
    member(TransportType, TransportChoiceList).
```

/*Η συγκεκριμένη connectedAndMatchAllChoices θα εκτελεστεί μόνο αν δεν βρεθεί απευθείας σύνδεση προς ένα προορισμό από την Αθήνα. Χρησιμοποιείται από την αναδρομική connectedAndMatchAllChoices που θα δούμε παρακάτω*/

```
connectedAndMatchChoices(Start, Finish, _, _, _, _, TransportChoiceList, RouteCostChoice,
                                RouteTimeChoice, RouteDistanceChoice):-
    Start = athens,
    directlyConnected(Start, Finish, TransportType),
    member(TransportType, TransportChoiceList),
    route(TransportType, athens, Finish, Cost, Time, Distance),
```

```
    Cost =< RouteCostChoice,
    Time =< RouteTimeChoice,
    Distance =< RouteDistanceChoice,
    showSinglePathInfo(athens, Finish, TransportType).

showSinglePathInfo(Start, Finish, TransportType):-
    route(TransportType, Start, Finish, RouteCost, RouteTime, RouteDistance),
    format('From ~w to ~w using ~w:', [Start, Finish, TransportType]),
    nl,
    nl,
    format('cost is ~w euro, time is ~w hours, distance is ~w km', [RouteCost, RouteTime,
            RouteDistance]),
    nl,
    nl.

/*Η συγκεκριμένη connectedAndMatchAllChoices χρησιμοποιείται ως «φρένο», ώστε αν όντως
υπάρχει απευθείας σύνδεση προς μια τοποθεσία από Αθήνα, τότε να μην μπούμε στην αναδρομική
connectedAndMatchAllChoices, και να γίνουν όλοι οι έλεγχοι επί τόπου. Επίσης, εφόσον
ικανοποιηθούν τα προαναφερθέντα, εκτυπώνει πληροφορίες του δρομολογίου με την κλήση της
showSinglePathInfo */

connectedAndMatchChoices(Start, Finish, CurrentCost, CurrentTime, CurrentDistance,
                         CurrentTransportType,
                         TransportChoiceList, RouteCostChoice, RouteTimeChoice,
                         RouteDistanceChoice):-
    directlyConnected(Start, Middle, CurrentTransportType),
    route(CurrentTransportType, Start, Middle, RouteCost, RouteTime, RouteDistance),
    member(CurrentTransportType, TransportChoiceList),
    NewCost is CurrentCost + RouteCost,
    NewTime is CurrentTime + RouteTime,
    NewDistance is CurrentDistance + RouteDistance,
    connectedAndMatchChoices(Middle, Finish, NewCost, NewTime, NewDistance,
                             NewTransportType, TransportChoiceList, RouteCostChoice,
                             RouteTimeChoice, RouteDistanceChoice),
    route(NewTransportType, Middle, Finish, FinalRouteCost, FinalRouteTime,
          FinalRouteDistance),
    member(NewTransportType, TransportChoiceList),
    TotalCost is NewCost + FinalRouteCost,
    TotalTime is NewTime + FinalRouteTime,
    TotalDistance is NewDistance + FinalRouteDistance,
    TotalCost =< RouteCostChoice,
    TotalTime =< RouteTimeChoice,
    TotalDistance =< RouteDistanceChoice,
    format('From ~w to ~w using ~w:', [Start, Middle, CurrentTransportType]),
    nl,
    nl,
```

```
    format('cost is ~w euros, time is ~w, hours, distance is ~w km', [NewCost, NewTime,
NewDistance]),
    nl,
    nl,
    format('From ~w to ~w using ~w:', [Middle, Finish, NewTransportType]),
    nl,
    nl,
    format('cost is ~w euro, time is ~w hours, distance is ~w km', [FinalRouteCost,
FinalRouteTime, FinalRouteDistance]),
    nl,
    nl,
    format('The total cost, travel time and distance to travel to ~w are:', [Finish]),
    nl,
    nl,
    format('~w euro, ~w hours and ~w km',[TotalCost, TotalTime, TotalDistance]),
    nl,
    nl.
```

/*Η παραπάνω connectedAndMatchChoices είναι και αυτή που επιτυγχάνει την αναδρομικότητα για την πρόσβαση σε τοποθεσία με χρήση ενδιάμεσου σταθμού (εφόσον δεν υπάρχει απευθείας σύνδεση και δεν σταματήσει ο έλεγχος στο «φρένο» που είπαμε πιο πάνω). Δέχεται ως είσοδο τα εξής:

Start-αφετηρία,
Finish-προορισμός,
CurrentCost - τρέχων κόστος,
CurrentTime - τρέχων χρόνος,
CurrentDistance - τρέχων απόσταση,
CurrentTransportType - τρέχων μεταφορικό μέσο,
TransportChoiceList – λίστα με επιλεγμένα μεταφορικά μέσα,
RouteCostChoice – επιλεγμένο όριο κόστους συνολικής διαδρομής,
RouteTimeChoice – επιλεγμένο όριο χρόνου συνολικής διαδρομής,
RouteDistanceChoice – επιλεγμένο όριο απόστασης συνολικής διαδρομής

Βήματα που ακολουθεί:

1. Κάνει αρχικά κλήση της directlyConnected για να βρει μια τοποθεσία που είναι προσβάσιμη απευθείας από την Start και αποθηκεύσει την νέα τοποθεσία αυτή στην Middle και το μεταφορικό μέσο που επιλέχθηκε στην CurrentTransportType. Με αυτόν τον τρόπο έχει ταυτοποιηθεί μια διαδρομή από μια τοποθεσία σε μια άλλη με τη χρήση συγκεκριμένου μεταφορικού μέσου.

2. Τρέχει την route για να αντιστοιχήσει τα RouteCost, RouteTime, RouteDistance στις τιμές της παραπάνω διαδρομής.

3. Τρέχει την member(CurrentTransportType, TransportChoiceList) για να βεβαιωθεί ότι το μεταφορικό μέσο που επιλέχθηκε από το βήμα ένα εμπίπτει στους περιορισμούς που έχουν δοθεί από το χρήστη.

4. Στη συνέχεια δίνει στα NewCost, NewTime, NewDistance τις τιμές των αποτελεσμάτων της πρόσθεσης των CurrentCost+RouteCost κλπ. Αυτό αποσκοπεί στο ότι θέλουμε τα New να μπουν στην αναδρομή για να ξαναεμφανιστούν ως Current και κάθε φορά που βρίσκουμε ενδιάμεσο σταθμό να προστίθενται τα RouteCost, RouteTime, RouteDistance της ενδιάμεσης διαδρομής.

5. Τρέχει η connectedAndMatchChoices παίρνοντας ως είσοδο τις νέες τιμές για τα κόστη, χρόνους κλπ, ενώ ταυτόχρονα παίρνει και την Middle και προσπαθεί αυτή τη φορά να βρει απευθείας ή έμμεση διαδρομή από την Middle στην Finish.

6. Όταν ικανοποιηθεί και η αναδρομική κλήση (δηλαδή βρεθεί ενδιάμεσος σταθμός που για να φτάσουμε στον προορισμό), τότε τρέχει η route για να αντιστοιχήσει τα FinalRouteCost, FinalRouteTime, FinalRouteDistance στις τιμές που αντιστοιχούν στην τελευταία διαδρομή που βρέθηκε.

7. Προσθέτει τα NewCost/Time/Distance με τα FinalRouteCost/Time/Distance και δίνει τις τιμές αυτές στα TotalCost/Time/Distance ώστε να πάρουν τα συνολικά κόστη κλπ ολόκληρου του δρομολογίου που ακολουθήθηκε.

8. Συγκρίνει τα συνολικά κόστη κλπ με τους περιορισμούς που είχε βάλει ο χρήστης. Αν ικανοποιούνται οι περιορισμοί προχωράμε παρακάτω για να εκτυπώσουμε τις πληροφορίες που αφορούν στη διαδρομή μέχρι τον ενδιάμεσο προορισμό, στη διαδρομή από τον ενδιάμεσο μέχρι τον τελικό και στο συνολικό δρομολόγιο.*/

```
matchRoute(TransportChoiceList, RouteCostChoice, RouteTimeChoice, RouteDistanceChoice,
                DestinationName):-
    connectedAndMatchChoices(athens, DestinationName, 0, 0, 0, _, TransportChoiceList,
                                    RouteCostChoice, RouteTimeChoice, RouteDistanceChoice).
```

/*Η matchRoute υπάρχει για να βεβαιωθούμε ότι η connectedAndMatchChoices θα εκτελεστεί με τις αρχικές τιμές που πρέπει στα ορίσματα. Η αφετηρία πρέπει να είναι η athens, το τρέχων κόστος/χρόνος/απόσταση πρέπει να είναι 0 αφού κατά την αρχική κλήση είναι και τα αρχικά. Έτσι ο χρήστης που θα θέλει να ελέγξει την δυνατότητα πρόσβασης σε τοποθεσία βάση περιορισμών θα εκτελέσει ερώτημα χρησιμοποιώντας την matchRoute και δε θα χρειάζεται να ανησυχεί για τις σωστές αρχικές τιμές της κλήσης της connectedAndMatchChoices*/

γ. **matchAllChoices**

```
matchAllChoices(LocationChoice, SightChoice, AccommodationChoice, RoomChoice,
                    RoomCostChoice, TransportChoiceList, RouteCostChoice, RouteTimeChoice,
                    RouteDistanceChoice):-
    matchLocation(LocationChoice, DestinationName),
```

```
    matchSight(SightChoice, DestinationName),
    matchAccomodation(AccommodationChoice, RoomChoice, RoomCostChoice,
                      DestinationName),
    matchRoute(TransportChoiceList, RouteCostChoice, RouteTimeChoice,
               RouteDistanceChoice, DestinationName).

/*Το matchAllChoices όλους τους περιορισμούς που μπορεί να βάλει ένας χρήστης, βρίσκει έναν προορισμό που ικανοποιεί όλους αυτούς τους περιορισμούς και εκτυπώνει όλες τις εναλλακτικές διαδρομές προς τον προορισμό με αναλυτικές πληροφορίες για το ταξίδι.*/

matchLocation(LocationChoice, DestinationName):-
    destinationData(DestinationName, LocationType, _, _, _),
    member(LocationChoice, LocationType).

/*Το matchLocation παίρνει την τιμή του LocationChoice από την κλήση του matchAllChoices. Στη συνέχεια αντιστοιχεί το δεύτερο όρισμα-λίστα κάθε γεγονότος destinationData με το LocationType. Μετά ελέγχει αν ο περιορισμός που έβαλε ο χρήστης, δηλαδή το LocationChoice, είναι member του LocationType. Αν ναι τότε ικανοποιείται και επιστρέφει με την τιμή του προορισμού στο DestinationName*/

matchSight(SightChoice, DestinationName):-
    destinationData(DestinationName, _, sights(List), _, _),
    member([SightChoice, _], List).

/*Το matchSight παίρνει σαν ορίσματα το SightChoice που έβαλε ο χρήστης στην κλήση του matchAllChoices και το DestinationName που ικανοποίησε το matchLocation. Στη συνέχεια με τη χρήση του destinationData βρίσκει το κατηγόρημα sights/1 που αντιστοιχεί στο συγκεκριμένο προορισμό και αντιστοιχεί τη λίστα-όρισμά του στο List. Μετά, με τη χρήση του member/2 ελέγχει αν η List περιέχει μέσα της μια άλλη λίστα με πρώτο όρισμα την τιμή του SightChoice, για να δει εν τέλη αν ο συγκεκριμένος προορισμός έχει το συγκεκριμένο είδος αξιοθέατων.*/

matchAccommodation(AccommodationChoice, RoomChoice, CostChoice, DestinationName):-
    destinationData(DestinationName, _, _, accommodations(List), _),
    member([AccommodationChoice, List1], List),
    member([RoomChoice, Cost],List1),
    Cost=< CostChoice.

/*Το matchAccommodation παίρνει σαν ορίσματα τα AccommodationChoice, RoomChoice, CostChoice που έβαλε ο χρήστης στην κλήση του matchAllChoices και το DestinationName που ικανοποίησε τα matchLocation και matchSight. Στη συνέχεια με τη χρήση του destinationData βρίσκει το κατηγόρημα accomodations/1 που αντιστοιχεί στο συγκεκριμένο προορισμό και αντιστοιχεί τη λίστα-όρισμά του στο List. Μετά, με τη χρήση του member/2 αποδομεί τη List και ελέγχει αν ο προορισμός διαθέτει συνδυασμό είδους καταλύματος, είδους δωματίου και κόστος δωματίου που να ικανοποιεί τους περιορισμούς του χρήστη.*/
```

Είναι δυνατό τα matchLocation, matchSight και matchAccommodation να χρησιμοποιηθούν ως μεμονωμένα ερωτήματα.

Η λειτουργία του matchRoute έχει αναλυθεί σε προηγούμενη παράγραφο.

δ. **showFullInfo**

```
showFullInfo(DestinationName):-
    showSightInfo(DestinationName);
    showAccommodationInfo(DestinationName);
    showPublicUtilityServices(DestinationName).

/*Το showFullInfo δέχεται σαν όρισμα το όνομα ενός προορισμού και μετά παίρνει με τη σειρά τα άλλα τρία «show» που φαίνονται και τα εκτελεί ένα ένα, εκτυπώνοντας όλες τις πληροφορίες που αφορούν στον προορισμό.*/

showSightInfo(DestinationName):-
    format('In ~w you can find the following interesing tourist sights (categorized by the type):',
            [DestinationName]),
    nl,
    nl,
    destinationData(DestinationName, _, sights(SightList), _, _),
    member([SightType|Sights], SightList),
    format('~w:', [SightType]),
    nl,
    member(SpecificSight, Sights),
    format('~w', [SpecificSight]),
    nl,
    fail.

/*Το showSightInfo παίρνει σαν όρισμα ένα προορισμό και εκτυπώνει όλες τις πληροφορίες περί αξιοθέατων που αφορούν στον προορισμό αυτό. Για να το πετύχει αυτό, μέσω της χρήσης του κατηγορήματος destinationData περνάει τη λίστα-όρισμα του sights/1, για τον προορισμό που μας ενδιαφέρει, στο SightList. Μετά παίρνει το SightList και με τη χρήση του member/2 το αποδομεί και εκτυπώνει όλες τις πληροφορίες που θέλουμε.*/

showAccommodationInfo(DestinationName):-
    format('The room choices for ~w (presented in room type-cost pairs) are:',
[DestinationName]),
    nl,
    nl,
    destinationData(DestinationName, _, _, accommodations(AccommodationList), _),
    member([AccommodationType|Rooms], AccommodationList),
    format('~w:', [AccommodationType]),
    nl,
    member(Rooms_and_Cost, Rooms),
```

```
    format('~w', [Rooms_and_Cost]),
    nl,
    fail.

/*Το showAccommodationInfo παίρνει σαν όρισμα ένα προορισμό και εκτυπώνει όλες τις πληροφορίες περί καταλυμάτων που αφορούν στον προορισμό αυτό. Για να το πετύχει αυτό, μέσω της χρήσης του κατηγορήματος destinationData περνάει τη λίστα-όρισμα του accommodatios/1, για τον προορισμό που μας ενδιαφέρει, στο AccommodationList. Μετά παίρνει το AccommodationList και με τη χρήση του member/2 το αποδομεί και εκτυπώνει όλες τις πληροφορίες που θέλουμε.*/

showPublicUtilityServices(DestinationName):-
    format('The public utility services that can be found in ~w are the following:',
            [DestinationName]),
    nl,
    nl,
    destinationData(DestinationName, _, _, _, publicUtilityServices(ServicesList)),
    member(Service, ServicesList),
    format('~w', [Service]),
    nl,
    fail.

/*Το showPublicUtilityServices παίρνει σαν όρισμα ένα προορισμό και εκτυπώνει όλες τις πληροφορίες περί υπηρεσιών κοινής ωφελείας που αφορούν στον προορισμό αυτό. Για να το πετύχει αυτό, μέσω της χρήσης του κατηγορήματος destinationData περνάει τη λίστα-όρισμα του publicUtilityServices/1, για τον προορισμό που μας ενδιαφέρει, στο ServicesList. Μετά παίρνει το ServicesList και με τη χρήση του member/2 το αποδομεί και εκτυπώνει όλες τις πληροφορίες που θέλουμε.*/
```

ε. **initialization**

Το «:-initialization(['projectFacts.pl']).» χρησιμοποιείται έτσι ώστε όταν υποβληθεί αίτηση για να γίνει compile το projectRules.pl, τότε να φορτωθεί πρώτα το projectFacts.pl, και μετά να γίνουν compile και τα δύο.