În primul rând programul citește datele din fișier. Astfel, ele sunt salvate în diferite entități, cum ar fi structura de insule, matricea de adiacență a grafului, vectori, etc.

Dupa ce au fost stocate toate informațiile necesare rezolvării cerințelor se apeleaza funcția "cerințe" ce are drept scop găsirea de soluții viabile și afișarea lor in fișierul de output.

În funcția "cerințe" citim numele cerinței ce se dorește a fi îndeplinită de către program. Apoi, pentru a reuși să stochez în mod corect datele corespunzătoare etapei de gestiune, am creat o condiție prin care să se citească valori atât timp cât ele se află într-un interval fix. Acest interval este aflat cu ajutorul funcției "oprire". În cadrul ei am determinat valoarea intreagă minimă a vectorului ce conține numele cerințelor, dar și pe cea maximă. Astfel, revenind la funcția "cerințe", voi calcula de fiecare dată când citesc câte o valoare dacă se află in intervalul de referință. Dacă respectă această condiție va apela funțiile aferente fiecărei cerințe, altfel se va întrerupe procesul citirii și va converti ultimul element citit într-un întreg, acesta reprezentând numărul de insule pentru care se va realiza etapa de gestiune.

Dacă valoarea citită este identică cu cea a cerinței va îndeplini acea functie.

Funcția "conexiune" parcurge matricea grafului g și verifică daca valoarea dată de linia și coloana parcursă e diferită de 0 și afișează un mesaj corespunzător (dacă e 0 atunci nu exită conexiune între insulele citite).

Funcția legatura utilizează același principiu ca și funcția precedentă, cu excepția faptului ca în fișier chiar atunci când este îndeplinită condiția.

Funcțiile adauga\_zbor si anulare\_zbor parcurg matricea grafului, iar în momentul în care indicii care parcurg matricea ajung să fie egali cu pozițiile insulelor se va adăuga durata zborului, respectiv se va modifica costul in 0. Căutarea se face după linii.

Funcția "max\_resurse" adaugă toate resursele diferite într-un arbore în așa fel încât dacă îl parcurgem în inordine se vor afița valorile în ordine lexicografică. În cadrul parcurgerii în inordine mă folosesc de două stive: una cu ajutorul careia reusesc sa scot elementele in ordine lexicografica si alta in care salvez aceste elemente, intrucat se doreste mai intai afisarea numarului de resurse diferite. Astfel, de fiecare data cand scot cate un element din stiva cu valori numarul de resurse va creste. Apoi, afisez acest numar urmat de numele resurselor in ordine lexicografica.

Funcția "max\_cantitate" citește mai întâi resursa, iar apoi determină care este cantitatea maximă de resursă, parcurgând vectorul de insule. După realizarea acestui pas, caută, din nou, in același vector, insula care are resursa cu acea cantitate maxima gasită.

Funcția "drum\_zbor" se foloseste de algoritmul lui Dijkstra pentru a afla care este distanta minima si creeaza vectorul de predecesori cu ajutorul caruia determinam care este drumul minim dintre cele doua insule citite. Functia care determina aceasta cale este recursiva, reusind sa afiseze valorile incepand cu prima insula intermediara pana la insula destinatie.

Functiile "min\_zbor" si "timp\_zbor" se folosesc de algoritmul lui Dijkstra, cu mica diferenta ca pentru a putea determina zborul minim dupa-amiaza a trebuit sa modific algoritmul pentru a adauga si durata stationarii. Ele returneaza durata minima in conditiile date de cerinte. Pentru acest lucru am creat o functie care returneaza "costul" minim calculat pana la varful destinatie.

Din moment ce inca nu s-a ajuns la finalul fisierului, dar etapele de proiectare si analiza s-au sfarsit, inseamna ca se poate desfasura urmatoarea etapa, si anume etapa de gestiune.

Astfel, acum se continua procesul citirii. Fiecare informatie citita o introducem intr-o structura de date bine definita. In primul rand, verificam daca avem avioane de distribuit prin intermediul unei functii care returneaza o anumita valoare in cazul in care nu se pot distribui avioane, din cauza faptului nicio insula nu are locuri disponibile, desi toleranta maxima este depasita. Daca toleranta maxima nu este depasita de nicio insula atunci nu trebuie sa distribuim niciun avion, deci vom afisa input-ul. De asemenea, eliberam memoria ocupata de celalalt graf, intrucat nu mai avem nevoie de el.

In cazul in care identificam insule care depasesc toleranta maxima va trebui sa le distribuim avioanele cu id-ul cel mai mare. Astfel, ne vom folosi de functia "distribuire\_avioane" pentru a efectua aceasta operatiune.

În primul rând, cream un vector (vect\_insule) in care stocăm valorile insulelor ce trebuiesc distribuite. Apoi, ii cream o copie pe care o ordanăm crescător. Ea ne ajuta in procesul distribuirii pentru a putea afisa informatiile corect. Apoi, parcurgem noul vector de insule si verificam diferite cazuri pentru a putea realiza corect distribuirea si afisarea valorilor.

Verificam pentru fiecare insula parcursa daca se afla in legatura cu insula care depaseste toleranta. Astfel, in cazul in care se intampla acest lucru, vom proceda astfel: mai intai verificam daca insula respectiva poate sa accepte avioane, in cazul contrar ea ramanand neschimbata. Daca, totusi, poate sa accepte, vom crea o copie numarului de avioane distribuite. Ea ne ajuta sa reusim sa identificam toate valorile si cazurile posibile atunci cand parcurgem matricea viitorului graf.

Daca se distribuie doar doua avioane, iar insulele primitoare pot sa accepte aceste avioane atunci vom apela functia "distribuire\_2\_avioane". Ea afiseaza valorile, incepand cu pozitia transmisa prin parametru. Acesta este o functie recursiva. In momentul in care valoarea transmisa prin parametru ajunge sa fie egala cu numarul de avioane distribuite vom afisa prima valoare a vectorului de insule.

In cazul in care avem de distribuit mai mult de 2 avioane si toleranta insulelor cu care se afla in legatura directa nu o depaseste pe cea maxima, inseamna ca le putem distribui. La inceputul functiei "distribuire\_avioane" mi-am creat un vector in care am stocat cate avioane poate sa primeasca fiecare insula.

Astfel, cream un graf reprezentat prin matrice de adiacenta si parcurg doar matricea superior triunghiulara dupa linii. In cadrul acestei parcurgeri afisez pentru fiecare linie noua avioanele deja existente in cadrul insulei care primeste noi id-uri. Afisez valorile astfel: pentru fiecare linie noua "voi distribui" avioanele aflate in vectorul vect\_insule (se afla in ordinea primirii lor, adica descrescatoare). Apoi parcurg coloanele, verificand daca exista "legatura" in matricea de adiacenta a grafului format,folosindu-ma de copia ordonata crescator a vectorului de insule. Dupa ce parcurg coloanele, dimensiunea matricii va scadea cu o unitate si se repeta procedeul pana cand distribuim toate avioanele. In momentul in care dimensiunea scade cu o unitate, voi determina avioanele care se afla pe "ultima pozitie" ("un fel de <<frunze>>"). Astfel, voi salva avioanele intr-un vector pe care il voi transmite unei functii care se ocupa de identificarea valorilor de pe "ultimul nivel".

Functia "numar" (mentionata mai sus) returneaza valoarea "frunzei" si o salveaza intr-un alt vector de "frunze". Ea cauta avionul care nu se gaseste pe linia curenta (se poate forma o alta posibilitate de distribuire cu el) si il returneaza. Daca numarul de avioane distribuite a ajuns sa fie 0 atunci afisam valorile vectorului "de frunze". In cazul in care toleranta avionului investigat este mai mica decat numarul de avioane distribuite si este, totodata, diferita de 0 inseamna ca mai exista o posibilitate de distribuire, aceea in care afisam doar spatiu liber, intrucat inca mai poate primi avioane.

Daca, parcurgand matricea de adiacenta a grafului dat ca parametru in functia "distribuire\_avioane", nu identificam o legatura directa cu insula care nu respecta toleranta maxima atunci îi vom afișa valorile, nemodificând-o.

Pe parcursul programului, am eliberat memoria ocupata de variabile (ex. grafurile,vectorii alocati dinamic).