**Tema si cerintele proiectului:**

Evidenta studentilor cazati intr-un camin. Exista doua structuri cu urmatoarele campuri:

1. – ID

-Nume

-Prenume →camin

-Facultate

-An

-Camera

1. – ID

-Nume

-Prenume →restantier

-Restanta

-Luna

**Continutul proiectului :**

* Fisiere text si binare ,cu operatiile aferente care se pot face asupra lor
* Liste simplu inlantuite, cu operatiile aferente care se pot face asupra lor
* Arbori binari de cautare, cu operatiile aferente care se pot face asupra lor

Schita proiectului ar putea fi urmatoarea:

Golireww

Creare

Golireww

Creare

**Meniul programului :**

La executarea programului apare un meniu , care contine taste de selectie, corespunzatoare celor de la 1 la 4. S-a incercat crearea unui meniu „user-friendly” , cu taste de navigatie intuitive. Acesta este:

Meniu principal:

1.Fisier Text

2.Liste

3.Arbori

4.Fisier Binar

0.Iesire

Apasarea oricarei taste de la 1 la 4, face aparitia unui nou submeniu, aliniat mai in stanga, pentru a putea fi urmarit mai usor de utilizator. Submeniurile si functiile lor, vor fii descrise in cele ce urmeaza:

* **Apasarea tastei 1 in meniul principal va duce la aparitia unui nou submeniu:**

MENIU FISIER TEXT

1.Creare fisier text

2.Adaugare student in fisierul text

3.Stergere student din fisierul text

4.Stergere fisier text

5.Modificare student din fisier text

6.Afisare studenti din fisierul text

* Apasara tastei 1 ,constituie crearea unui fisier text si adaugarea de inregistari atat timp cat doriti apasand tasta y/Y;
* Apasarea tastei 2 permite adaugarea unei noi inregistrari in fisierul text
* Apasarea tastei 3 duce la stergerea unui student din fisier,cautat dupa campul ” Id”;
* Apasarea tastei 4 duce la stergerea tuturor inregistrarilor din fisier;
* Apasarea tastei 5 permite modificarea datelor unui student cautat dupa campul ”Id”;
* Apasarea tastei 6 conduce la afisarea sub forma unui tabel a tuturor inregistrarilor existente in fisier la momentul acela;
  + **Apasarea tastei 2 in meniul principal va duce la aparitia unui nou submeniu:**

MENIU LISTE

0.Iesire

1.Creare fisier

Acest meniu va fi diferit in diferite stadii ale programului. In acest pas, daca nu a fost creata o lista, este disponibila decat o optiune, pentru ca este singura operatie valida pe care o puteti face la acest pas.Lista va fi creata cu inregistrarile din fisierul text.

Daca s-a facut o adaugare in lista, sau lista contine cel putin un element, atunci submeniul afisat va fi urmatorul:

0.Iesire

1.Creare lista

2.Adaugare student in lista

3.Stergere student din lista

4.Modificarea unui student din lista

5.Cautarea unui nod in lista

6.Afisare lista camin

7.Pemutari studenti an

9.Sortare dupa nume-interclasare

A.Golire lista in fisier txt

* Apasarea tastei 2 permite adaugarea unei noi inregistrari la sfarsitul listei;
* Apasarea tastei 3 permite stergerea unei inregistari din lista,cautata dupa campul „id”;
* Apasarea tastei 4 permite modificarea datelor unui student din lista.cautat dupa campul „id”;
* Apasarea tastei 5 conduce la cautarea unei inregistrari din lista dupa campul „Id”,dat de utilizator;
* Apasarea tastei 6 constituie afisarea tuturor inregistrarilor din lista sub forma unui tabel pentu a fi vizualizate mai usor;
* Apasarea tastei 7 constituie permutarea studentilor dintr-un an,citit de la tastatura;
* Apasarea tastei 8 conduce la sortarea listei dupa campul „id”,folosind metoda Quicksort(Divide et Impera);
* Apasarea tastei 9 conduce la sortarea listei in ordine alfabetica dupa campul „nume” folosind metoda de interclasare(Divide et Impera);
* Apasarea tastei a/A constituie golirea inregistrarilor din lista inapoi in fisierul text;
  + **Apasarea tastei 4 din meniul principal conduce la afisarea urmatorului submeniu:**

MENIU FISIERE BINARE

0.Iesire

1.Creare fisier binar

2.Adaugare student in fisierul binar

3.Stergere student din fisierul binar

4.Stergere fisier binar

5.Modificare student fisier binar

6.Afisare studenti din fisier binar

* Apasarea tastei 1 ,constituie crearea unui fisier binar si adaugarea de inregistrari atat timp cat doriti apasand tasta y/Y;
* Apasarea tastei 2 permite adaugarea unei noi inregistrari in fisierul binar;
* Apasarea tastei 3 duce la stergerea unui student din fisier,cautat dupa campul ” id”;
* Apasarea tastei 4 duce la stergerea tuturor inregistrarilor din fisier;
* Apasarea tastei 5 permite modificarea datelor unui student cautat dupa campul ”id”;
* Apasarea tastei 6 conduce la afisarea sub forma unui tabel a tuturor inregistrarilor existente in fisier la momentul acela;
  + **Apasarea tastei 3 in meniul principal conduce la afisarea urmatorului submeniu:**

MENIU ARBORI

0.Iesire

1.Creare arbore

Acest meniu va fi diferit in diferite stadii , ale programului. In acest pas, daca nu a fost creat un arbore, este disponibila decat o optiune, pentru ca este singura operatie valida pe care o puteti face la acest pas.Arborele va fi creat cu inregistrarile din fisierul binar.

MENIU ARBORI

0.Iesire

1.Creare arbore

2.Afisare aborescenta

3.Inserare nod

4.Stergere nod

5.Afisare Inordine

6.Afisare Preordine

7.Afisare Postordine

8.Afisarea grupurilor de studenti-restanta luna x

9.Golire arbore in fisier

A.Permutari studenti-backtracking

B.Sortare dupa suma restanta-numarare

Daca arborele a fost creat,noul submeniu va arata astfel:

* Apasarea tastei 2 conduce la afisarea arborescenta a id-urilor inregistrarilor din arbore;
* Apasarea tastei 3 permite inserarea unei noi inregistrari in arbore.Inserarea se va face dupa campul „id”;
* Apasarea tastei 4 permite stergerea unui nod din arbore;
* Apasarea tastei 5 conduce la afisarea inregistrarilor din arbore,folosind parcurgerea in Inordine;
* Apasarea tastei 6 conduce la afisarea inregistrarilor din arbore,folosind parcurgerea in Preordine;
* Apasarea tastei 7 conduce la afisarea inregistrarilor din arbore,folosind parcurgerea in Postordine;
* Apasarea tastei 8 duce la afisarea tuturor grupurilor de studenti care au restanta pe luna X,citita de la tastatura;
* Apasarea tastei 9 conduce la golirea inregistrarilor din arbore in fisierul binar.Inregistrarile in fisier se vor adauga in ordinea parcurgerii Inordine,Preordine sau Postordine a arborelui;
* Apasarea tastei a/A va permuta studentii din arbore astel incat sa nu existe 2 studenti unul langa celalalt care au restanta pe aceeasi luna;
* Apasarea tastei b/B conduce la sortarea listei crescator dupa suma restanta folosind metoda de sortare prin numarare ;

**Specificatii avansate:**

In programul prezent se foloseste pentru stocarea informatiilor, mai multe campuri.

* Tabelul ce descrie structura campului CAMIN:

|  |  |  |
| --- | --- | --- |
| **Denumire camp** | **Tip de date** | **Tip de date in C** |
| id | Intreg | int |
| nume | Sir de caractere | Char [] |
| prenume | Sir de caractere | Char [] |
| Facultate | Sir de caractere | Char [] |
| An | Intreg | int |
| Camera | Intreg | int |

In structura programului, lista studentilor este stocata sub forma de inregistrari alocate dinamic.

* Tabelul ce descrie campul RESTANTIER:

|  |  |  |
| --- | --- | --- |
| **Denumire camp** | **Tip de date** | **Tip de date in C** |
| id | Intreg | int |
| nume | Sir de caractere | char [] |
| prenume | Sir de caractere | char [] |
| restanta | real | float |
| luna | Sir de caractere | Char [] |

* Pentru crearea listei se foloseste tipul de date nod.Tabelul acestuia este:

|  |  |  |
| --- | --- | --- |
| **Denumire camp** | **Tip de date** | **Tip de date in C** |
| inf | Pointer catre inregistrarea camin |  |
| urm | Pointer catre membrul urmator |  |

* Pentru crearea arborelui se foloseste tipul de date abore.Tabelul acestuia este urmatorul:

|  |  |  |
| --- | --- | --- |
| **Denumire camp** | **Tip de date** | **Tip de date in C** |
| inf | Pointer catre inregistrearea restantier |  |
| dr | Pointer la membrul drept |  |
| st | Pointer la membrul stang |  |

Pentru crearea unui program flexibil, in meniul principal se desfasoara decat meniul principal, impreuna cu submeniurile adecvate,si se citesc toate datele , urmand ca sa fie apelate functiile care fac lucrul deservit, cu parametrii corespunzatori.

* Tabelul acestor functii este urmatorul :

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Returneaza** | **Nume** | | | **Parametri** | | | | **Descriere** | |
| ***Functii asupra fisierului text*** | | | | | | | | | |
| void | creareFisierTxt | | | fara | | | Creeaza un fisier text,adaugand inregistrari pana la apasarea tastei n/N | | |
| void | afisareFisierTxt | | | fara | | | Afiseaza inregistrarile din fisier sub foma unui tabel | | |
| void | stergereFisierTxt | | | fara | | | Sterge toate inregistrarile din fisier | | |
| Void | adaugareFisierTxt | | | fara | | | Adauga o noua inregistrare in fisier | | |
| void | stergereStudFisierTxt | | | fara | | | Sterge un student din fisier,cautat dupa id | | |
| void | modificareStudFisierTxt | | | fara | | | Modifica datele unui student din fisier | | |
| Void | FisierTxt | | | fara | | | Submeniul aferent fisierului text | | |
| ***Functii asupra listei*** | | | | | | | | | |
| void | adaugareNodLista | | | fara | | | Adauga un nou nod in lista | | |
| void | stergereNodLista | | | fara | | | Sterge un nod din lista,cautat dupa id | | |
| Void | afisareLista | | | fara | | | Afiseaza inregistrarile din lista sub forma unui tabel | | |
| void | creareLista | | | fara | | | Creaza lista simplu inlantuita | | |
| void | modificareNodLista | | | fara | | | Modifica datele unul student din lista | | |
| void | cautareNodLista | | | fara | | | Cauta un nod in lista dupa id | | |
| int | Returneaza\_nr | | | **Nume** | | **Tip** | Returneaza nr de studenti dintr-un an,citit de la tastatura | | |
| x | | int |
| void | back | | | **Nume** | | **Tip** | Permuta toti studentii dintr-un anumit an,citit de la tastatura | | |
| t | | int |
| x | | Int |
| nr\_studenti | | int |
| void | quicksort | | | fara | | | Sorteaza inregistrarile din lista dupa id,folosind metoda quicksort.Aceasta functie foloseste si functia sortare\_id care face sortarea propriu-zisa | | |
| void | interclasare | | | fara | | | Sorteaza inregistrarile din lista in ordine alfabetica dupa nume,folosind metoda de inteclasare.Aceasta functie mai foloseste si alte functii care fac sortarea propriu-zisa(sort,interc,divimp) | | |
| void | Golire\_lista | | | fara | | | Goleste inregistrarile din lista in fisierul text | | |
| void | | Liste | fara | | | | Submeniul aferent listei | |
| ***Functii asupra fisieruluir binar->aceleasi operatii ca si in cazul fisierului text,diferind doar numele functiilor si putin functiile de citire si scriere in fisier binar*** | | | | | | | | |
| ***Functii asupra arborelui*** | | | | | | | | |
| void | | Inordine1 | **Nume** | | **Tip** | | Parcurge arborele in Inordine | |
| a1 | | arbore | |
| void | | Preordine1 | Nume | | Tip | | Parcurge arborele in Preordine | |
| a1 | | arbore | |
| void | | Postordine1 | Nume | | Tip | | Parcurge arborele in Postordine | |
| a1 | | arbore | |
| void | | afisare\_arbore | Nume | | Tip | | Afiseaza arborele sub forma aborescenta.Nodul cel mai din stanga este radacina | |
| n | | int | |
| a1 | | arbore | |
| void | | AdaugareInArb | Nume | | Tip | | Adauga un nou nod in arbore | |
| arb | | arbore | |
| x | | restantier | |
| void | | creareArb | Nume | | Tip | | Creaza arborele | |
| arb | | arbore | |
| void | | stergereNodArbore | fara | | | | Sterge un nod din arbore | |
| void | | golire\_arbore | fara | | | | Goleste inregistrarile din arbore in fisierul binar ,arborele fiind parcurs in Inordine,Preordine sau Postordine | |
| void | | permutari | fara | | | | Permuta toti studentii din arbore astfel incat sa nu fie unul langa altul cu restanta pe aceasi luna | |
| void | | sortare\_numarare | fara | | | | Sorteaza inregistrarile din arbore dupa suma restanta | |
| void | | Arbori | fara | | | | Submeniul aferent arborilor | |

**Descrierea detaliata:**

Functiile care efectueaza diferitele operatii sunt urmatoarele:

* **Functii asupra fisierelor**
* **void creareFisierTxt()**

Aceasta functie deschide un fisier de tip text pentru scriere,si citeste pe rand datele unui student de la tastatura.

In primul rand se deschide fisierul „fis1.txt” pentru scriere,si se foloseste optiunea „wt” pentru al doilea parametru al functiei **fopen,**iar apoi intr-un ciclu do-while se citesc de la tastatura datele unui student pana la tastarea caracterului ‚n’ sau ‚N’.Datele se scriu in fisier pe aceeasi linie,fiecare linie reprezentand un student cu datele acestuia.

* **void afisareFisierTxt()**

Aceasta functie deschide fisierul text pentu citire,folosind optiunea “rt”.Intr-un **while** se citeste din fisier atata timp cat nu s-a ajuns la sfarsitul acestuia,iar inregistrarile se scriu intr-un tabel pentru a fi vizualizate mai usor.Daca fisierul contine mai multe inregistari pentru a putea fi urmarite mai usor dupa fiecare 15 inregistrari se reia capul de tabel.

* **void stergereFisierTxt()**

Aceasta functie ofera posibilitatea de a sterge fisierul text.Imediat apelata pe ecran apare o interogare-daca se doreste stergerea tuturor inegistrarilor din fisier.In caz afirmativ fisierul se va deschide cu optiunea “wt”,iar toate inregistrarile vor fi sterse,astfel se va anula si la apasarea oricarei taste va reveni in submeniul fisierului.

* **void adaugareFisierTxt()**

Dupa cum ii spune si numele aceasta functie are facilitatea de a adauga o noua inregistrare in fisierul text.Fisierul se deschide pentru adaugare cu optiunea “at” ,se citesc datele noului student,iar apoi se citeste in fisier pentru a verifica daca mai exista o inregistrare cu acelasi id,iar in caz contrar noua inregistrare se va adauga la sfarsitul fisierului.

* **void stergereStudFisierTxt()**

Aceasta functie sterge un student din fisier.Se citeste de la tastatura id-ul studentului care se doreste a fi sters,iar apoi se citeste din fisier atata timp cat nu s-a ajuns la sfarsitul acestuia,iar in cazul in care id-ul este diferit de cel citit informatiile se scriu intr-un al doilea fisier “fis2.txt” .Primul fisier se va sterge iar fisierul “fis2.txt” va fi redenumit cu numele primului fisier “fis1.txt”.

* **void modificareStudFisierTxt()**

Dupa cum ii spune si numele aceasta functie are rolul de a modifica datele unui student din fisier.Se citeste id-ul studentului ale carui date se doresc modificate,iar apoi se citesc noile date care se pun intr-o variabila **b** de tip **camin** .Se deschide fisierul pentru citire si se citesc inregistrarile din acesta atata timp cat nu s-a ajuns la sfarsitul acestuia,iar inregistrarile citite se vor scrie in alt fisier.In cazul in care s-a gasit studentul cu id-ul respectiv pentru a fi modificat,in noul fisier se scriu noile date ale acestuia.Primul fisier se va sterge,iar cel de-al doilea va fi redenumit cu numele primului fisier.

Pentu fisierul binar se folosesc aceleasi functii ca si in cazul celui text,diferind doar numele acestora ,tipul de deschidere si tipul de citire din fisier si scriere in fisier.Din fisierul binar datele se citesc cu functia **fread**,iar datele se scriu in fisier cu functia **fwrite** specifice acestora.

* + **Functii asupra listei**

* **void adaugareNodLista()**

Aceasta functie are rolul de a adauga o noua inregistrare la sfarsitul listei.Se citesc datele noului student de la tastatura intr-o variabia **t1** de tip nod,iar legatura urmatoare va deveni **NULL** deoarece nu mai exista niciun nod dupa aceasta.Se verifica apoi daca lista este vida,iar in caz afirmativ noul nod va deveni primul nod al listei,in caz contrar se va parcurge lista pana la ultimul nod iar urmatorul dupa ce acesta va fi noul nod.

* **void stergereNodLista()**

Apelarea acestei functii are facilitatea de a sterge un nod al listei.

Daca lista este vida se afiseaza un mesaj corespunzator pe ecran,iar in caz contrar se citeste id-ul studentului care se doreste a fi sters din lista.Intr-un **while** se parcurge lista cu 2 variabile pana la ultimul nod al listei si penultimul,atata timp cat nu s-a ajuns la sfarsitul listei si nu a fost gasit studentul cu id-ul citit de la tastura.La iesirea din while se verfica daca id-ul ultimului nod din lista difera de id-ul citit,aceasta insemnand ca studentul cu id-ul respectiv nu a fost gasit in lista.Daca lista are un singur element va fi stearsa toata lista,iar daca nodul care se doreste sters este primul nod al listei,atunci prim va deveni urmatorul,iar acesta se va sterge.In cazul in care nodul dorit sters este un nod oarecare in lista,urmatorul nod dupa nodul din stanga va deveni nodul din dreapta sa,iar acesta va fi sters.

* **void afisareLista()**

Aceasta functie afiseaza inregistrarile din lista sub forma unui tabel pentru a fi vizualizate mai usor.Intr-un **while**,atata timp cat lista nu s-a sfarsit se parcurge element cu element,iar acesta va fi afisat in tabel. Daca lista contine mai multe inregistari pentru a putea fi urmarite mai usor dupa fiecare 15 inregistrari se reia capul de tabel.

* **void creareLista()**

Aceasta functie creaza o lista simplu inlantuita..Fisierul text se deschide pentru citire,si atata timp cat nu s-a ajuns la sfarsitul acestuia se citeste cate o inregistrare si cu ajutorul functiei **void insertLista(nod \*p1)** se adauga inregistrarea in lista.Daca lista este vida inregistrarea devine prim,ia in caz contrar se parcurge lista pana la utlimul nod si inregistrarea este adaugata la sfarsitul listei.

* + - **void modificareNodLista()**

Cu aceasta functie putem modifica datele unui student din lista.Se citesc noile date intr-o variabila **p2** de tip **nod**(id-ul studentului nu se modifica),iar apoi se parcurge lista pana la gasirea studentului cu id-ul care se doreste a fi modificat,si cu functia **strcpy** noile date ale studentului vor deveni cele nou citite.

Daca s-a ajuns la sfarsitul listei si nu s-a gasit nici un student cu id-ul respectiv se afiseaza un mesaj corespunzator.

* **void cautareNodLista()**

Aceasta functie are rolul de a cauta un nod in lista.Se citeste de la tastatura id-ul studentului care se cauta,iar intr-un **while** atata timp cat lista nu s-a sfarsit si studentul nu a fost gasit se parcuge lista.Daca studentul a fost gasit se afiseaza un mesaj corespunzator,urmat de afisarea datelor acestuia,iar daca s-a parcurs lista si nodul nu a fost gasit de asemenea se afiseaza un mesaj corespunzator.

* + **Permutari**

Programul poate permuta elemente. S-a evitat folosirea tehnicii backtraking, si realizarea folosind o metoda iterativa recursiva. Functia care realizeaza aceste permutari este **back** .

Problema permutarilor ar suna cam in felul urmator:

Sa se permute studentii din acelasi an, iar la fiecare permutare realizata, sa se verifice daca exista sau nu 2 studenti asezati unul langa altul care locuiesc in aceeasi camera. Daca nu exista 2 studenti care nu indeplinesc aceasta conditie, sa se afiseze OK, in dreptul fiecarei permutari, iar in caz contrar sa se afiseze Gresit.

La apasarea tastei 7 corespunzatoare permutarilor, se va introduce un numar, care reprezinta numarul anului al carui studenti se doresc sa fie permutati. Dupa apasarea tastei ENTER, programul afiseaza un mesaj prin care utilizatorul este informat despre numarul de studenti din anul respectiv. La apasarea oricarei taste, se afiseaza permutarile,si conditia daca este valida sau nu, iar dupa fiecare solutie se asteapta apasareai unei taste, pentru a fi vizualizate mai usor solutiile.

Functiile care sunt folosite la generarea permutarilor sunt:

* **int returneaza\_nr(int x)**

Returneaza numarul de studenti apartinand anului dat ca parametru formal.

* **nod \*returneaza\_al\_k\_student\_al\_caminului(int k, int x)**

Aceasta functie returneaza datele celui de-al k-lea student din anul dat ca parametru. Numarul k trebuie sa fie mai mic decat numarul de student din an, iar studentul care se doreste sa se returneze exista in mod obligatoriu. Aceasta functie este folosita atunci cand dupa generarea solutiilor de catre functia **back**, adica t>numarul de solutii, vectorul **d** va contine pozitiile elementelor din lista, ale elementelor ale firmei care se doreste realizarea permutarilor. Obtinerea acestor elemente se realizeaza apeland functia, cu **d[i]+1**, unde **i** este un contor cuprins intre 0 si numarul de solutii care se doresc generate.

* **void back(int t,int x,int nr\_studenti)**

Modul de lucru al functiei este de ai genera un vector intre un numar  **t** si numarul maxim , in cazul nostru numarul de studenti care apartin unui bloc..

Functia  **back**  se apeleaza initial cu t=0, acest parametru semnificand intervalul in care se genereaza numele, adica 0 si numarul de studenti. Pe parcursul rularii ea se apeleaza recursiv pana cand t=numarul de studenti ai unui bloc. In acest caz inseamna ca am gasit o solutie, si se poate afisa. Numerele generate sunt intr-un vector **v**  care contin pozitii permutate ale elementelor. In continuare afisam studentii situati pe pozitia **v[i]**.

Modul de functionare al functiei este urmatorul:

Se considera 2 vectori, de o dimensiune fixa. In cazul nostru vectorii sa fie de marimea 30. Unul dintre ei este folosit pentru validarea pozitiei la care ne aflam, **v**, iar celalalt se folosete pntru retinerea valorilor generate.

La apelarea **back(0,anb,nr\_studenti)** aceasta aloca memorie pentru cei 2 vectori, alocarea de memorie facandu-se static, deoarece functia se apeleaza recursiv, dorinta fiind de a pastra valorile la o apelare ulterioara, si de a nu aloca o noua zona de memorie.

La fiecare apelare, se genereaza o solutie posibil valida, validitatea solutiei fiind data de vectorul **v**  care va retine valoarea 1 daca solutia a fost epuizata. Acest lucru se efectueaza intr-un **for** care porneste de la 0 si continua pana la numarul de solutii care se doresc generate. Dupa validarea solutiei, se scrie in vectorul **d** numarul solutie, apelandu-se recursiv functia cu t+1, adica pornind de la elementul urmator. Dupa fiecare apel terminat se invalideaza **v[i]** ca o solutie din nou valida.

Aceast lucru se repeta, pana cand **t** este egal cu numarul de solutii care doresc sa fie generate. Ajuns in acest pas inseamna ca solutia validata este buna, iar numerele din vectorul **d** reprezinta decat pozitiile elementelor care se doresc permutate. Obtinerea acestor elemente se realizeaza apeland returneaza\_al\_k\_student\_al\_caminului(a[i]).

* + **Sortari**

Programul poate sorta elementele din lista crescator dupa id,folosind metoda quicksort,si in ordine alfabetica dupa nume,folosind metoda de interclasare.

**Quicksort**

Functia care realizeaza sortarea propriu-zisa este sortare\_id,algoritm bazat pe metoda Divide et Impera.

* **void sortare\_id(int sg,int dp)**

Aceasta functie trateaza o portiune dintr-un vector, cuprinsa intre indicii sg (limita inferioara) si dp (limita superioara).

**Se alege o valoare pivot.** Se ia valoarea elementului din mijloc ca valoare pivot.

Se rearanjeaza elementele in asa fel incat, toate elementele care sunt mai mici decat pivotul merg in partea stanga a pivotului si toate elementele care sunt mai mari decat pivotul merg in partea dreapta a tabloului. **Se sorteaza amandoua partile.** Se aplica recursiv algoritmul de sortare rapida in partea stanga si in partea dreapta.

Exista 2 indici i si j, si la inceputul algoritmului de partitionare i indica primul element din tablou iar j indica ultimul element din tablou. La pasul urmator algoritmul muta i inainte, pana cand un element cu o valoare mai mare sau egala cu pivotul este gasita. Indicele j este mutat inapoi, pana cand un element cu o valoare mai mica sau egala cu pivotul este gasita. Daca i <= j atunci i merge pe pozitia (i +1 ) iar j merge pe pozitia (j - 1). Algoritmul se opreste, cand i devine mai mare decat j. Dupa partitie, toate valorile dinaintea celui de-al i-lea element sunt mai mici sau egale cu pivotul si toate valorile de dupa cel de-al j-lea element sunt mai mari sau egale cu pivotul.

* **void quicksort()**

Aceasta functie creaza cu elementele listei un vector de tip **nod**,pentru a putea fi sortat,folosind functia de mai sus,care initial este apelata de 1 si n,primul si respectiv ultimul element al vectorului.

Lista devine vida,iar dupa sortarea acesteia se ia pe rand element cu element din vectorul vect sortat anterior si se insereaza in lista,si astfel lista este sortata.

**Interclasare**

Functia care realizeaza sortarea alfabetica dupa nume a studentilor din lista este **void interclasare().**

Algoritmul de sortare prin interclasare se bazeaza pe ideea ca pentru a sorta un vector cu n elemente il impartim in doi vectori care o data sortati se interclaseaza. Deci conform strategiei DIVIDE ET IMPERA, pe care se bazeaza ai acest algoritm, problema este descompusa in alte doua subprobleme de acelasi tip si, dupa rezolvarea lor, rezultatele se combina (in cazul de fata se interclaseaza). Descompunerea unui vector in alti doi vectori care urmeaza a fi sortati are loc pana cand avem de sortat vectori de unul sau doua elemente.

Procedura **void sort(int t,int g)** sorteaza un vector de maxim 2 elemente, interclaseaza rezultatele, **void divimp(int t,int g)** implementează strategia DIVIDE ET IMPERA.

* Functia **void interclasare()** are acelasi rol ca si functia **void quicksort()**.
  + **void golire\_lista()**

Aceasta functie goleste toate inregistrarile din lista inapoi in fisierul text,folosind alt fisier,care la sfarsit va fi redenumit cu numele fisierului initial.

Se parcurge elementele listei unul cate unul si sunt scrise int-un fisier text “fis4.txt”,deschis pentru adaugare cu optiunea “at”.Dupa ce a fost parcursa toata lista,fisierul “fis1.txt” se sterge iar fisierul “fis4.txt” este redenumit cu numele fisierului anterior.

* **Functii asupra arborelui**
  + **void Inordine1(arbore \*p1)**

Apeland aceasta functie elementele arborelui sunt parcurse si afisate in Inordine(stanga,radacina,dreapta),incepand de la radacina care este pimita ca parametru,cu ajutorul recursivitatii.

* **void Preordine1(arbore \*a1)**

Apeland aceasta functie elementele arborelui sunt parcurse si afisate in Preordine(stanga,dreapta,radacina),incepand de la radacina care este primita ca parametru,cu ajutorul recursivitatii.

* **void Postordine1(arbore \*a1)**

Apeland aceasta functie elementele arborelui sunt parcurse si afisate in Postordine(dreapta,stanga,radacina),incepand de la radacina care este primita ca parametru,cu ajutorul recursivitatii.

* **void afisare\_arbore(int n,arbore \*a1)**

Functia afisare,face afisarea arborelui binar de cautare pe ecran,asa cum este el retinut in memorie.Primul paramtru al functiei este numarul de spatii care se afiseaza inainte sa se afiseze elementul.Acest numar este folosit pentru alinierea elementelor,astfel incat sa aiba aspect de arbore.

* **arbore \* creareArb(arbore \*arb)**

Aceasta functie creaza un arbore binar de cautare din fisierul text.Crearea arborelui se realizeaza pe principiul urmator:se citeste inregistrare cu inregistrare din fiser,pana la sfarsit,iar pentru fiecare element se apeleaza functia AdaugareInArb,care adauga elementul dupa campul id,astfel incat studentii cu id-urile mai mici ca radacina se afla in stanga,iar studentii cu id-urile mai mari ca radacina se afla in dreapta.

* **void inserare\_nod\_arb()**

Aceasta functie adauga un element in arbore.Pentru a functiona,arborele in care se adauga trebuie sa aiba cel putin un element.Procedura de inserare se face prin compararea campului “id” al fiecarui student,astfel: elementele care au id < elementul curent,se adauga in stanga,iar cele care au id-ul mai mare sunt adaugate in dreapta..

* **void gaseste(arbore \*a1,arbore \*a2)**

Aceasta functie recursiva cauta un element in arbore,primind ca parametri radacina arborelui si o alta variabia **a2** de tip arbore care primeste infomatiile din nodul gasit.

* **void Stergere(int nr)**

Algoritmul acestei functii este:daca nodul p ce trebuie eliminat are cel mult un fiu,se modifica campul referinta al nodului predecesor nodului p(nodul tata) astfel incat sa indice unicul fiu al nodului p,daca acesta exista sau “NULL” daca p nu are niciun fiu.Daca p are 2 fii atunci:

-se cauta nodul cu cea mai mare cheie din subarborele stang atasat lui **p**,fiind acesta **r**.Cautarea se deruleaza astfel:se construieste o secventa de noduri care incepe cu fiul stang al nodului p, dupa care se alege drept succesor al fiecarui nod, fiul sau drept. Primul nod al secventei care nu are fiu drept este nodul **r**.

-se modifica nodul **p** asigurandu‑i cheia corespunzatoare nodului **r.** In acest moment nodul **r** exista in dublu exemplar: in locul sau si in locul fostului nod **p**.

-se suprima nodul **r** initial, aplicand primul caz al algoritmului.

Aceasta functie foloseste si functia Elimina() care realizeaza cautarea nodului cu cea mai mare cheie din subarborele stang sau cautarea nodului cu cea mai mica cheie din subarborele drept si copierea sa, mai putin legaturile, in nodul ce trebuie suprimat si apoi suprimarea sa.

* Functiile Inordine2(),Preordine2() si Postordine2() formeaza vectorii corespunzatorii parcurgerii respective.
  + - **Submultimi**

Programul poate afisa subgrupuri de studenti,folosind o metoda iterativa de combinatorica,generearea submultimilor.

**Problema submultimilor ar suna cam in felul urmator:**

Sa se afiseze toate grupurile de studenti care au restanta pe o luna citita de la tastatura.

La apasarea tastei 8 corespunzatoare submultimilor, se va introduce un sir de caractere, care reprezinta luna pentru care sa se afiseze studentii cu restanta pe luna respectiva. Se va parcurge arborele in Inordine si elementele sale vor fi preluate intr-un vector **comb** de tip **restantier**,dupa care se va apela intr-un **while** atata timp cat nu se mai pot forma submultimi, functiile din algoritmul submultimilor.Dupa fiecare subgrup,apasarea taste y duce la afisarea urmatorului subgrup,iar cu apasarea tastei n nu se mai afiseaza grupurile si cu apasarea oricarei taste se va reveni in submeniu.

* **void INIT()**

Apelarea acestei functii are ca scop initializarea elementelor vectorului v si a indicatorului de generare ig la 1.

* **void GEN\_SUBM()**

Aceasta functie aisgura generarea unei noi submultimi in vectorul **v** si daca este cazul modifica valoarea indicatorului **ig**.

* **void PREL()**

Aceasta functie realizeaza operatiile corespunzatoare problemei cu elementele submultimii generate continute in vectorul v,fara a modifica elementele tabloului v.

In aceasta functie se verifica daca toti studentii submultimii generate indeplinesc conditia,ci anume cea de a avea restanta pe luna citita de la tastatura.

* + - **Permutari-elementele arborelui-tehnica backtracking**

**Enuntul problemei:**

Sa se permute toti studentii astfel incat 2 studenti vecini(unu langa altul) sa nu aiba restanta pe aceeasi luna.Pentru a genera permutarile sa se foloseasca tehnica backtracking.

**Functii cu care se realizeaza aceste permutari sunt:**

* **void init()**

Apeland aceasta functia stiva este initializata in pozitia **k** cu 0.

* **int solutie(int k)**

Aceasta functie testeaza daca s-a ajuns sau nu la solutie,adica daca **k** a ajuns la numarul de studenti existenti atunci in arbore,si in caz afirmativ returneaza **TRUE**,iar in caz negativ **FALSE**.

* **void succesor(int k)**

Aceasta functie gaseste elemntul urmator al multimii **st[k]**+1.Daca este succesor acesta este pus in stiva si **ok** primeste 1,astfel **ok** primeste 0.

* **void tipar()**

Aceasta procedura tipareste solutia.

* **void valid()**

Aceasta functie testeaza conditiile de continuare,adica daca 2 studenti unul langa altul au restanta pe aceeasi luna, si returneaza 1 in caz afirmativ,astfel returneaza 0 prin variabila **gasit**.

* **void permutari()**

Aceasta functie impreuna cu cele descrise anterior realizeaza permutarile studentilor.

In aceasta functie se parcurge arborele in Inordine,iar elementele acestuia sunt preluate intr-un vector de tip **restantier**,iar apoi se apeleaza functia **init()** de initializare,dupa care, folosind functiile de mai sus se permuta toti studentii astfel incat sa nu existe 2 studenti unul langa altul cu restanta pe aceeasi luna.

Functia consta in urmatorul algoritm:atata timp cat nu s-au generat toate posibilitatile de a permuta studentii se intra in **while**,unde se verifica daca nu s-a ajuns la numarul de studenti, iar daca nu s-a ajuns,daca exista succesor si este valid se verifica daca s-a ajuns la o solutie,iar in caz afirmativ afiseaza numele studentilor din permutarea curenta,astfel se ia urmatorul student si se initializeaza stiva.Daca s-a ajuns la ultimul student si nu mai exista posibilitati, **k** scade si se reia algoritmul.

* **Sortari**

Pentru elementele arborelui s-a folosit o sortare prin numarare a studentilor, crescator dupa suma restanta.

Functia care realizeaza sortarea prin numarare este **void sortare\_numarare().**

Mai intai in aceasta functie se apeleaza functia Inordine2() care creeaza un vector de tip **restantier** cu elementele arborelui parcurse in Inordine,iar metoda de sortare prin numarare va fi aplicata pe acest vector.

Aceasta functie consta in a numara pentru fiecare student s i cati studenti au suma restanta mai mica decat acesta. Numerele obtinute le vom memora intr-un vector **aux**. Pe baza acestuia vom rearanja elementele vectorului **comb** intr-un alt vector notat **num**. In final vectorul **num** va fi copiat in **comb**, deoarece **num** contine elementele lui **comb** ordonate crescator dupa suma restanta..

* + - **void golire\_arbore()**

Aceasta functie are facilitatea de a goli elementele din arbore intr-un fisier binar “fis4.bin”,inregistarile fiind scrise in fisier folosind parcurgerea arborelui in Inordine(la apasarea tastei ‘1’),in Preordine(la apsarea tastei ‘2’) sau in Postordine(la apasarea tastei ‘3’).

Dupa ce elementele arborelui au fost scrise in fisier,fisierul “fis2.bin” se va sterge, iar noul fisier va fi redenumit cu numele celui sters anterior.