Ministerul Educaţiei, Tineretului şi Sportului al Republicii Moldova

UNIVERSITATEA TEHNICA a MOLDOVEI

*Facultatea Calculatoare Informatică și Microelectronică*

**LUCRARE DE CURS**

Disciplina **“Structuri de date si algoritmi”**

**Tema: Bază de date de tip fișier**

**A elaborat:** Stud.gr.TI-125

**Sava Cristian**

**A verificat:  Luchianov Ludmila**

**Chișinău 2013**

**Cuprins**

Introducere ……………………………………………………………………….4

1. Consideratii Teoretice ………………………………………………………..5
   1. Limbajul de programare *C* ……………………………………………………5
      1. Avantajele limbajului C………..…………………………………………6
      2. Tipuri de date……………………………………………………………7
   2. Fisiere…………………………………………………..……………………8
   3. Cautarea binara………………………………………………………………10
   4. Sortarea prin interclasare (merge sort) …….……………………………………13
2. Consideratii de implementare. Module…………………………………17
   1. Stocarea si accesarea informatiei………………….……………………………17
   2. Structuri de date……………………………………………...………………21
   3. Autentificarea………………………………………………..………………22
   4. Tipuri de utilizatori……………………………………...……………………24
3. Rulare Program. Instructiuni pentru utilizator…………………………26
   1. Rularea programului ca Administrator……………….…………………………26
   2. Rularea programului ca Profesor……………………………….………………32
   3. Rularea programului ca Student……………………..…………………………32

Concluzie…………………………………………………………….……………33

Anexa Nr.1: Listingul Programului……………………….………………………35

Bibliografie…………………………………………………………..……………50

**Introducere**

Lucrarea data prezinta un registru electronic care permite inscrierea, procesarea si stocarea informatiei despre studentii unei institutii de invatamant. Registrul electronic prezinta o alternativa contemporana a banalului registru prezent acum in toate instiutiile de invatamant. Avantajele unui asemenea registru sunt destule pentru a elabora un sistem de stocare, analiza si prezentare electronica a datelor despre student. Un registru in forma electronica face posibila accesarea datelor unui student de la distanta, o confidentialitate sporita si de asemenea comoditatea cu care se opereaza datele.

Programul prezentat reprezinta doar un model al unui registru electronic si poate servi drept baza de dezvoltare a unui program mult mai complex si cu stocare online a informatiei. Deoarece limbajul de programare C nu opereaza cu baze de date, stocarea informatiei are loc cu ajutorul fisierelor.

In baza acestei lucrari sunt folosite mai multe intrumente de programare, functii si algoritmi in baza limbajului de programare.

1. **Consideratii Teoretice**

La efectuarea lucrarii date am folosit mai multe instrumente, algoritmi si tehnici de programare. Programul este efectuat in limbajul de programare C folosind ca mediu de stocare a informatiei fisierele binare. Pentru efectuarea operatiilor am am folosit algoritmul de sortare prin interplasare (merge Sort) iar cautarea este efectuata folosind algoritmul binar.

**1.1 Limbajul de programare *C***

C este un [limbaj de programare](http://ro.wikipedia.org/wiki/Limbaj_de_programare) [standardizat](http://ro.wikipedia.org/wiki/Standard), compilat, de nivel mediu. Este implementat pe majoritatea platformelor de calcul existente azi, și este cel mai popular limbaj de programare pentru scrierea de [software de sistem](http://ro.wikipedia.org/w/index.php?title=Software_de_sistem&action=edit&redlink=1). Este apreciat pentru eficiența [codului obiect](http://ro.wikipedia.org/w/index.php?title=Cod_obiect&action=edit&redlink=1) pe care îl poate genera, și pentru[portabilitatea](http://ro.wikipedia.org/w/index.php?title=Portabilitate&action=edit&redlink=1) sa.

A fost dezvoltat la începutul anilor 1970 de [Ken Thompson](http://ro.wikipedia.org/wiki/Ken_Thompson) și [Dennis Ritchie](http://ro.wikipedia.org/wiki/Dennis_Ritchie), care aveau nevoie de un limbaj simplu și portabil pentru scrierea [nucleului](http://ro.wikipedia.org/w/index.php?title=Nucleu_(informatic%C4%83)&action=edit&redlink=1)sistemului de operare [UNIX](http://ro.wikipedia.org/wiki/UNIX).

Sintaxa limbajului C a stat la baza multor limbaje create ulterior și încă populare azi: [C++](http://ro.wikipedia.org/wiki/C%2B%2B), [Java](http://ro.wikipedia.org/wiki/Limbajul_Java), [JavaScript](http://ro.wikipedia.org/wiki/JavaScript), [C#](http://ro.wikipedia.org/wiki/C_Sharp).

C este un [limbaj de programare](http://ro.wikipedia.org/wiki/Limbaj_de_programare) relativ minimalist ce operează în strânsă legătură cu [hardware](http://ro.wikipedia.org/wiki/Hardware)-ul, fiind cel mai apropiat de [limbajul de asamblare](http://ro.wikipedia.org/wiki/Limbaj_de_asamblare) față de majoritatea celorlalte limbaje de programare.

C este prezentat uneori ca "[asamblor](http://ro.wikipedia.org/wiki/Limbaj_de_asamblare) portabil", făcându-se astfel diferențele principale față de limbajele de asamblare: codul unui program C poate fi compilat și rulat pe aproape orice tip de mașină ([calculator](http://ro.wikipedia.org/wiki/Calculator)), asemănător altor limbaje de programare, în timp ce limbajele de asamblare sunt specifice unui anumit model de mașină. Limbajul C aparține clasei limbajelor de nivel scăzutsau de nivel mediu, aceasta indicând strânsa legătură între interoperabilitate și echipamentul hardware.

*[Sursa: Wikipedia.org]*

**1.1.2 Avantajele limbajului C**

C a fost creat având drept scop important de a face ca programele mari să poată fi scrise mai ușor și cu mai puține erori în [paradigma programării procedurale](http://ro.wikipedia.org/wiki/Programare_procedural%C4%83), dar fără a pune un obstacole în scrierea [compilatorului](http://ro.wikipedia.org/wiki/Compilator) de C, care este încărcat de caracteristicile complexe ale limbajului. C are urmatoarele caracteristici importante:

* Este un limbaj de bază simplu, cu importante funcționalități cum ar fi funcțiile matematice sau cele de manipulare ale [fișierelor](http://ro.wikipedia.org/wiki/Fi%C8%99ier)
* Este focalizat pe paradigma programării procedurale, care facilitează programarea într-un mod structurat
* Utilizează un set simplu de tipuri de [date](http://ro.wikipedia.org/wiki/Dat%C4%83) ce împiedică multe operații neintenționate
* Folosește un limbaj preprocesor, [preprocesorul](http://ro.wikipedia.org/w/index.php?title=Preprocesor&action=edit&redlink=1) C, pentru sarcini cum ar fi definirea de [macrouri](http://ro.wikipedia.org/w/index.php?title=Macro&action=edit&redlink=1) și includerea mai multor fișiere sursă
* Permite accesarea la nivel scăzut a [memoriei calculatorului](http://ro.wikipedia.org/wiki/Arhitectura_von_Neumann) prin utilizarea [pointerilor](http://ro.wikipedia.org/w/index.php?title=Pointer&action=edit&redlink=1)
* Permite folosirea parametrilor, care sunt comunicați funcțiilor prin valoare și nu prin referință
* Pointeri la funcții, ce permit forme rudimentare de [închidere](http://ro.wikipedia.org/w/index.php?title=%C3%8Enchidere_(limbaje_formale)&action=edit&redlink=1) (engleză closure) și polimorfism
* Declararea [variabilelor](http://ro.wikipedia.org/w/index.php?title=Variabile&action=edit&redlink=1)
* [Structuri de date](http://ro.wikipedia.org/wiki/Structur%C4%83_de_date) sau [tipuri de date agregate](http://ro.wikipedia.org/w/index.php?title=Tip_de_date_agregat&action=edit&redlink=1), definite de utilizator prin (struct), ce permit ca date înrudite să fie combinate și manipulate ca un întreg

Printre caracteristicile ce lipsesc în C, dar care pot fi găsite în alte limbaje de programare se enumeră:

* Un sistem automat de colectare a reziduurilor (de memorie) (engleză [Garbage collection](http://ro.wikipedia.org/w/index.php?title=Garbage_collection&action=edit&redlink=1))
* [Clasele](http://ro.wikipedia.org/wiki/Clas%C4%83_(programare)) și [obiectele](http://ro.wikipedia.org/wiki/Obiect_(programare)) ([programarea orientată pe obiecte](http://ro.wikipedia.org/wiki/Programare_orientat%C4%83_pe_obiecte))
* Un sistem avansat de tipuri de date
* [Programarea generică](http://ro.wikipedia.org/w/index.php?title=Programare_generic%C4%83&action=edit&redlink=1)
* [Supraîncărcarea](http://ro.wikipedia.org/w/index.php?title=Supra%C3%AEnc%C4%83rcare&action=edit&redlink=1)
* [Metaprogramarea](http://ro.wikipedia.org/wiki/Metaprogramare)
* Suport nativ pentru [programarea multifir](http://ro.wikipedia.org/w/index.php?title=Programare_multifir&action=edit&redlink=1) (engleză multithreaded) și funcțiile de [rețea](http://ro.wikipedia.org/wiki/Re%C8%9Bea_de_calculatoare)
* Procesarea [listelor](http://ro.wikipedia.org/wiki/List%C4%83_(structur%C4%83_de_date))

Deși lista minusurilor limbajului C este destul de lungă, aceasta nu reprezintă un dezavantaj suficient de mare pentru inhiba utilizarea limbajului C în practică, deoarece permite ca noi compilatoare să poată fi scrise pe noi tipuri de platforme și pentru că permite programatorului să țină bine sub control programul pe care îl scrie. Acesta este unul din motivele care face ca un cod scris în C să fie mult mai eficient decât dacă ar fi scris în alte limbaje de programare. Numai un cod scris cu foarte mare grijă într-un limbaj de asamblare poate fi mai performant, deoarece are control integral asupra mașinii, dar performanța avansată a compilatoarelor, combinată cu complexitatea noilor tipuri de procesoare, a făcut ca limbajul C să fie preferat și să fie acceptat din ce în ce mai mult de programatori.

Una din consecințele acceptării și eficienței C-ului este aceea că multe compilatoare,  [biblioteci](http://ro.wikipedia.org/w/index.php?title=Bibliotec%C4%83_(software)&action=edit&redlink=1) și [interpretoare](http://ro.wikipedia.org/w/index.php?title=Interpretor_(programare)&action=edit&redlink=1) ale limbajelor de nivel înalt sunt adesea implementate în C.

*[Sursa: Wikipedia.org]*

**1.1.3 Tipuri de date**

C are un sistem de [tipuri de date](http://ro.wikipedia.org/w/index.php?title=Tip_de_date&action=edit&redlink=1) similar cu cel al descendenților [ALGOL](http://ro.wikipedia.org/wiki/ALGOL), cum ar fi [Pascal](http://ro.wikipedia.org/wiki/Pascal), dar totuși cu anumite diferențe. Cuprinde tipuri de date cum ar fi [întregi](http://ro.wikipedia.org/w/index.php?title=Num%C4%83r_%C3%AEntreg_(informatic%C4%83)&action=edit&redlink=1) de diferite dimensiuni, cu sau fără semn, [numere în virgulă mobilă](http://ro.wikipedia.org/w/index.php?title=Num%C4%83r_%C3%AEn_virgul%C4%83_mobil%C4%83&action=edit&redlink=1), [enumerări](http://ro.wikipedia.org/w/index.php?title=Enumerare_(informatic%C4%83)&action=edit&redlink=1) (enum), structuri de date (struct) și [uniuni](http://ro.wikipedia.org/w/index.php?title=Uniune_(informatic%C4%83)&action=edit&redlink=1) (union).

C utilizează foarte mult pointerii, un tip de [referință](http://ro.wikipedia.org/w/index.php?title=Referin%C8%9B%C4%83_(informatic%C4%83)&action=edit&redlink=1) foarte simplu, care păstrează [adresa](http://ro.wikipedia.org/w/index.php?title=Adres%C4%83_de_memorie&action=edit&redlink=1) locației din memorie. Adresa poate fi manipulată cu ajutorul atribuirilor și a [aritmeticii pointerilor](http://ro.wikipedia.org/w/index.php?title=Aritmetica_pointerilor&action=edit&redlink=1). În momentul rulării unui program, un pointer reprezintă o adresă de memorie. În momentul compilării, un pointer este un tip de dată complex, ce reprezintă atât adresa de memorie cât și tipul de dată. Acest lucru permite expresiilor ce utilizează pointeri să fie [evaluate](http://ro.wikipedia.org/w/index.php?title=Evaluare_(programare)&action=edit&redlink=1) după tipul de dată. Pointerii au mai multe utilizări în C. De exemplu, [șirurile de caractere](http://ro.wikipedia.org/w/index.php?title=%C8%98ir_de_caractere&action=edit&redlink=1) (engleză text string) sunt adesea reprezentate printr-un pointer la un [vector](http://ro.wikipedia.org/w/index.php?title=Vector_(structur%C4%83_de_date)&action=edit&redlink=1) de [caractere](http://ro.wikipedia.org/wiki/Caracter). [Alocarea dinamică a memoriei](http://ro.wikipedia.org/w/index.php?title=Alocare_dinamic%C4%83_a_memoriei&action=edit&redlink=1) este realizată tot cu ajutorul pointerilor.

Un pointer null are o valoare rezervată, indicând faptul că face referire la o locație nevalidă. Acest lucru este folositor în cazuri speciale cum ar fi pointerul next (următorul) în nodul final al unei [liste înlănțuite](http://ro.wikipedia.org/wiki/List%C4%83_(structur%C4%83_de_date)). [Dereferențierea](http://ro.wikipedia.org/w/index.php?title=Dereferen%C8%9Biere&action=edit&redlink=1) unui pointer null poate cauza un comportament imprevizibil al aplicației. De asemenea, există și pointeri de tip void, fapt ce indică referirea la un obiect de tip necunoscut. Acești pointeri sunt foarte folositori în [programarea generică](http://ro.wikipedia.org/w/index.php?title=Programare_generic%C4%83&action=edit&redlink=1). Deoarece dimensiunea și tipul obiectelor la care acest tip de pointeri face referire sunt necunoscute, aceștia nu pot fi dereferențiați, dar pot fi [convertiți](http://ro.wikipedia.org/wiki/Casting) la alt tip de pointeri.

În C, anterior standardului [C99](http://ro.wikipedia.org/w/index.php?title=C99&action=edit&redlink=1), tablourile (vectorii) sunt de dimensiune fixă, statică, cunoscută la momentul compilării; în practică, acest lucru nu reprezintă o piedică, având în vedere că se pot aloca blocuri de memorie în momentul rulării, tratându-le ca pe tablouri utilizând [librăria standard](http://ro.wikipedia.org/w/index.php?title=Libr%C4%83ria_standard_C&action=edit&redlink=1). Spre deosebire de multe alte limbaje de programare, C reprezintă tablourile ca și pe pointeri: o adresă și un tip de dată. Prin urmare, valorile index pot depăși dimensiunea actuală a unui tablou.

De asemenea, C oferă posibilitatea de lucru cu tablouri multidimensionale. Din punct de vedere [semantic](http://ro.wikipedia.org/w/index.php?title=Semantic%C4%83_(limbaje_formale)&action=edit&redlink=1), tablourile multidimensionale sunt tablouri de tablouri, dar, din punct de vedere fizic, acestea sunt stocate ca un singur tablou unidimensional cu un [offset](http://ro.wikipedia.org/w/index.php?title=Offset&action=edit&redlink=1) calculat.

C este adesea folosit în programarea de nivel scăzut, unde poate fi necesar ca un întreg să fie tratat ca o adresă de memorie, un număr în virgulă mobilă ca un întreg sau un tip de pointer ca un alt tip de pointer. Pentru astfel de cazuri C oferă operatorul de [casting](http://ro.wikipedia.org/wiki/Casting), care forțează explicit conversia unei valori dintr-un tip de dată în alt tip de dată.

**1.2 Fisiere**

Prin fişier se înţelege o colecţie de date, păstrată pe un dispozitiv periferic, care de regulă este un disc magnetic sau optic. Această colecţie de date poate fi interpretată ca şi o succesiune de caractere, de cuvinte, linii sau pagini ale unui document de tip text, ca şi câmpuri sau înregistrări ale unei baze de date sau de exemplu, ca şi pixelii unei imagini grafice.

Toate fişierele, indiferent de tipul de date pe care le conţin sau de metodele folosite pentru procesarea acestor date au câteva proprietăţi importante. Astfel, acestea trebuie deschise pentru citire şi trebuie închise la încheierea prelucrării lor, în aceste fişiere se pot scrie date, pot fi adăugate date sau pot fi citite date din ele. Când un fişier este deschis, datele sale pot fi accesate de la începutul sau de la sfârşitul fişierului sau dintr-un loc oarecare din fişier. Pentru prevenirea folosirii necorespunzătoare a unui fişier, sistemul de operare instalat pe calculator trebuie să fie înştiinţat despre scopul în care este deschis fişierul şi despre operaţiile care vor fi efectuate (citire, scriere, adăugare) asupra datelor. La încheierea operaţiunilor care se efectuează asupra datelor dintr-un fişier acesta trebuie închis deoarece în caz contrar, sistemul de operare nu poate actualiza datele referitoare la acel fişier (dimensiune, data ultimei accesări, etc.).

Există două tipuri de fişiere cu care se poate opera într-un program: fişiere text şi fişiere binare. Pentru programul dat am folosit fisierele binare. Fişierele binare pot să conţină orice tip de date, codate într-o formă binară, în secvenţe de byte.

Pentru aceste fişiere nu există caracter special de sfârşit de fişier şi ele pot conţine orice fel de informaţie : text formatat, imagini, sunete,

Din punctul de vedere al sistemului de operare, fişierele binare nu sunt diferite de fişierele text. Mai mult, în limbajul C un byte este echivalent cu un caracter, astfel că un fişier binar este de asemenea văzut ca un stream de caractere. Totuşi, fişierele binare diferă de fişierele text în două puncte esenţiale. Pe de o parte, admiţând că fişierele binare conţin date care nu sunt neapărat interpretabile ca şi text, fiecare octet de date este transferat pe sau de pe disc neprocesat. În al doilea rând, interpretarea fişierelor binare este lăsată pe seama programatorului, acestea putând fi citite sau scrise în orice mod dorit de către programator. Fişierele binare pot fi procesate secvenţial sau, în funcţie de necesităţile aplicaţiei, acestea pot fi procesate folosind tehnici de accesare aleatoare a înregistrărilor pe care le conţin. Această procesare aleatoare implică poziţionarea într-un anumit loc din fişier, înainte de citirea sau scrierea datelor, astfel că aceste fişiere sunt în general procesate prin operaţii de citire/scriere simultane.Astfel, se poate face poziţionarea instantanee pe orice structură din fişier şi de asemenea poate fi modificat conţinutul unei structuri oriunde în fişier, în orice moment de timp.

Motivul pentru care am ales fisierele binare in acest program sunt ca fişierele binare sunt accesate mai rapid decât fişierele text şi sunt înţelese numai de programe scrise special să lucreze cu astfel de fişere. Bazele de date, de exemplu, sunt create şi procesate ca şi fişiere binare. Spre exemplu, o valoare întreagă, să zicem 123, este stocată într-un fişier text ca un şir de trei caractere, '1', '2' şi '3'. Aceasa face ca respectiva valoare să fie citită cu uşurinţă într-un editor de texte dar, pe de altă parte, această reprezentare a ei este ineficientă deoarece sunt necesari 3 octeţi pentru stocarea ei, câte unul pentru fiecare caracter. Intr-un fişier binar, această valoare întreagă poate fi stocată pe un singur byte (12310=11110112); în consecinţă, fişierele binare oferă un mod de stocare mult mai compactă a valorilor numerice, dar au şi avantajul că nu pot fi vizualizate direct, ca şi fişierele de tip text, un lucru foarte important pentru modulul de logare prezentat ulterior. În fişierele binare, informaţia este stocată exact ca şi în memoria calculatoarelor, iar acest lucru face ca transferul conţinutului variabilelor C să se facă direct între memoria computerelor şi aceste

Un alt aspect asupra căruia trebuie să insistăm este faptul că fişierele binare conţin pur şi simplu serii de octeţi şi nu reţin în nici un fel tipul de informaţie stocată în acestea. Aşadar, programatorul trebuie să fie foarte atent la scrierea şi citirea datelor care vor fi atribuite variabilelor programului. Acest lucru este în particular important atunci când fişierele binare vor fi folosite de către alte programme sau pe maşini de calcul diferite care stochează pe un număr diferit de octeţi date cu acelaşi tip (de exemplu întregii pot fi stocaţi pe 4 (Unix) sau pe 2 octeţi (PC-uri Intel)). Operaţiile efectuate asupra fişierelor binare sunt similare cu cele efectuate asupra fişierelor text deoarece ambele tipuri sunt considerate ca stream-uri de octeţi. În limbajul C, aceleaşi tipuri de funcţii sunt folosite pentru procesarea celor două tipuri de fişiere. Totuşi, funcţiile pentru managementul fişierelor binare sunt puţin mai pretenţioase dar pe de altă parte conduc la citiri/scrieri mai rapide.

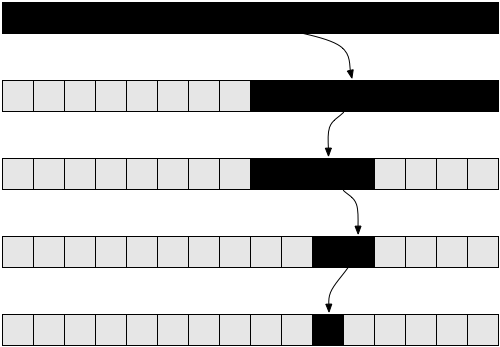
**1.3 Cautarea binara**

Algoritmul de căutare binară este un [algoritm de căutare](http://ro.wikipedia.org/w/index.php?title=Algoritm_de_c%C4%83utare&action=edit&redlink=1) folosit pentru a găsi un element într-o listă [ordonată](http://ro.wikipedia.org/wiki/Sortare_(informatic%C4%83)) (tablou unidimensional/vector). Algoritmul funcționează pe baza tehnicii [*divide et impera*](http://ro.wikipedia.org/wiki/Divide_et_impera_(informatic%C4%83)). Valoarea căutată este comparată cu cea a elementului din mijlocul listei. Dacă e egală cu cea a acelui element, algoritmul se termină. Dacă e mai mare decât acea valoare, algoritmul se reia, de la mijlocul listei până la sfârșit, iar dacă e mai mică, algoritmul se reia pentru elementele de la începutul listei până la mijloc. Întrucât la fiecare pas cardinalul mulțimii de elemente în care se efectuează căutarea se înjumătățește, algoritmul are [complexitate](http://ro.wikipedia.org/w/index.php?title=Complexitate_(informatic%C4%83)&action=edit&redlink=1) [logaritmică](http://ro.wikipedia.org/wiki/Logaritm).

Se consideră o lista simplu inaluita p  [deja sortat](http://ro.wikipedia.org/w/index.php?title=Deja_sortat&action=edit&redlink=1)a și trei variabile: ‘low’=inceput, ‘high’=sfârșit și ‘mid’=mijloc. Metoda verifică de mai multe ori dacă mijlocul vectorului/tabloului unidimensional este egal cu elementul căutat:

în cazul în care este egală, variabila mid reprezintă poziția elementului în lista; dacă nu se îndeplinește condiția de egalitate se trece la verificarea poziției elementului căutat în lsita astfel: dacă elementul căutat este mai mic decât elementul din mijlocul listei, variabila " high " ia valoarea lui "mid" iar dacă nu variabila ‘low’ ia valoarea lui ‘mid’.

Totul se repetă atât timp cât ‘low’ este mai mic decât ‘high’.



Ca în imaginea de mai sus, dorim să găsim poziția lui x în cel mai scurt mod. În loc să parcurgem pas cu pas lista de la început până la x (care poate fi chiar și ultimul nod al listei), împărțim vectorul în 2 și verificăm în care din jumătăți se află x-ul nostru. În acest exemplu, după prima înjumătățire, știm că x-ul nostru se află la dreapta față de n/2. Lăsăm ce este la stanga de n/2, și înjumătățim din nou. Ajungem la n/4, verificăm unde este x-ul și știm că x-ul este la stânga de n/4. Din nou, lăsăm ce se află la dreapta de n/4 și iar înjumătățim. Astfel ajungem la n/8 si obtinem valoarea cautata.

*struct nod \*search(struct nod \*p, char \*name[30]){*

*int low,high,mid,i;*

*element \*Pmid,\*Phead,\*ptr;*

*low=0;*

*Phead=p;*

*high=lungime(p)-1;*

*ptr=NULL;*

while(low<=high)

{

mid=(low+high)/2;

Pmid=Phead;

for(i=0;i<mid;i++)

{

Pmid=Pmid->next;

}

if(strcmp(Pmid->val.login,name)==0)

{

ptr=Pmid;

break;

}

else

{

if(strcmp(Pmid->val.login,name)>0)

{

high=mid-1;

}

else

{

if(strcmp(Pmid->val.login,name)<0)

{

low=mid+1;

}

}

}

}

return ptr;

}

**1.4 Sortarea prin interclasare (merge sort)**

Aceasta metoda de ordonare are la baza interclasarea a doi vectori: fiind dati doi vectori ordonati se obtine un al treilea vector ordonat care va contine elementele din cei doi vectori.

In cazul sortarii prin interclasare vectorii care se interclaseaza sunt doua secvente ordonate din acelasi vector

Sortarea prin interclasare utilizeaza metoda Divide et Impera:

-se imparte vectorul in secvente din ce in ce mai mici., astfel incat fiecare secventa sa fie ordonata la un moment dat si interclasata cu o alta secventa din vector corespunzatoare.

-practic interclasarea va incepe cand se ajunge la o secventa formata din doua elemente. Aceasta odata ordonata se va interclasa cu o alta corespunzatoare. Cele doua secvente vor alcatui in subsir ordonat din vector mai mare care la randul lui se va interclasa cu subsirul corespunzator s.a.m.d.

In programul prezentat am utilizat aceasta metoda pentru sortarea unei liste simplu inaluite.

Ca exemplu:

*Fie lista:*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ***8*** | ***7*** | ***9*** | ***3*** | ***6*** | ***4*** | ***17*** | ***16*** |

*Se imparte in doua secvente:*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ***8*** | ***7*** | ***9*** | ***3*** | ***6*** | ***4*** | ***17*** | ***16*** |

*In continuare pentru prima secvente se procedeaza la fel:*

|  |  |  |  |
| --- | --- | --- | --- |
| ***8*** | ***7*** | ***9*** | ***3*** |

*Dupa o noua impartire se obtine:*

|  |  |
| --- | --- |
| ***8*** | ***7*** |

*Se incepe interclasarea. Se considera ca avem doua liste de lungime 1 care se interclaseaza:*

|  |  |
| --- | --- |
| ***8*** | ***7*** |

*Rezulta:*

|  |  |
| --- | --- |
| ***7*** | ***8*** |

*La fel si pentru secventa :*

|  |  |
| --- | --- |
| ***9*** | ***3*** |

*Se considera ca avem iar doi vectori de lungime 1 care se interclaseaza:*

|  |  |
| --- | --- |
| ***9*** | ***3*** |

*Rezulta:*

|  |  |
| --- | --- |
| ***3*** | ***9*** |

*Ceea ce inseamna ca cele doua secvente determina obtinerea urmatorului subsir din lista:*

|  |  |  |  |
| --- | --- | --- | --- |
| ***7*** | ***8*** | ***3*** | ***9*** |

*Pentru care se interclaseaza cele doua zone. Rezulta:*

|  |  |  |  |
| --- | --- | --- | --- |
| ***3*** | ***7*** | ***8*** | ***9*** |

*La fel se procedeaza si cu secventa:*

|  |  |  |  |
| --- | --- | --- | --- |
| ***6*** | ***4*** | ***17*** | ***16*** |

*Se obtine:*

|  |  |
| --- | --- |
| ***6*** | ***4*** |

*Care se interclaseaza si se obtine:*

|  |  |
| --- | --- |
| ***4*** | ***6*** |

*La fel pentru:*

|  |  |
| --- | --- |
| ***17*** | ***16*** |

*Se obtine:*

|  |  |
| --- | --- |
| ***16*** | ***17*** |

*Se obtine o noua secventa:*

|  |  |  |  |
| --- | --- | --- | --- |
| ***4*** | ***6*** | ***16*** | ***17*** |

*Care prin interclasarea celor doua zone conduce la:*

|  |  |  |  |
| --- | --- | --- | --- |
| ***4*** | ***6*** | ***16*** | ***17*** |

*Cele doua secvente initiale din lista au devenit:*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ***3*** | ***7*** | ***8*** | ***9*** | ***4*** | ***6*** | ***16*** | ***17*** |

*Care se interclaseaza obtinand:*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **3** | **4** | **6** | **7** | **8** | **9** | **16** | **17** |

Codul sursa pentru sortarea prin inteclasarea folosit in program:

struct nod \*mergesort(struct nod \*head) {

struct nod \*head\_one;

struct nod \*head\_two;

if((head == NULL) || (head->next == NULL))

return head;

head\_one = head;

head\_two = head->next;

while((head\_two != NULL) && (head\_two->next != NULL)) {

head = head->next;

head\_two = head->next->next;

}

head\_two = head->next;

head->next = NULL;

return merge(mergesort(head\_one), mergesort(head\_two));

}

struct nod \*merge(struct nod \*head\_one, struct nod \*head\_two) {

struct nod \*head\_three;

if(head\_one == NULL)

return head\_two;

if(head\_two == NULL)

return head\_one;

if(strcmp(head\_one->val.sort , head\_two->val.sort)<0) {

head\_three = head\_one;

head\_three->next = merge(head\_one->next, head\_two);

} else {

head\_three = head\_two;

head\_three->next = merge(head\_one, head\_two->next);

}

return head\_three;

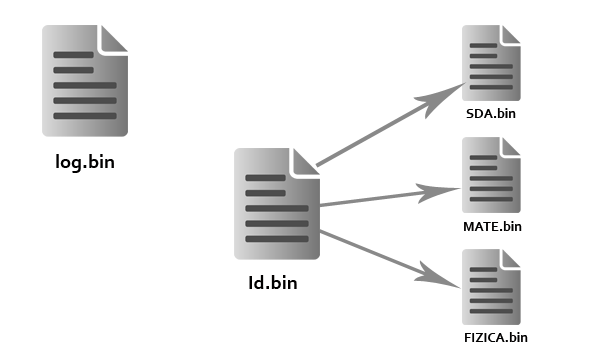
}

1. **Consideratii de implementare. Module**

**2.1 Stocarea si accesarea informatiei**

Toate datele sunt stocate in 3 tipuri de fisiere. Primul fisier – “log.bin” este un fisier independent unde sunt stocate datele de autentificare al fiecarui student. Acest fisier nu are legatura nici cu un alt fisier din program si acces asupra lui il au doar utilizatorii cu Rol:1.

Al doilea tip de fisier este *id.bin* unde se afla lista tuturor studentilor. Fiecarui student ii este atribuit un ID unic. Al treilea tip de fisiere sunt cele de disciplina. In programul dat este un singur fisier pentru o singura disciplina: *SDA.bin* si aici sunt inscrise toate notele studentilor. Fisierul SDA.bin are legatura directa cu fisierul *id.bin* si la fiecare operatie efectuata in unul din fisierul de tip disciplina, este solicitat sau ID-ul unui student dupa nume sau numele unui student dupa ID.



Functiile de apelare la fisierul *id.bin* sunt:

* **char getname(int id,char \*name)**{

int i=0;

elementID \*a,\*q,\*u;

a=readID();

q=a;

while(1){

if(id==q->valID.id){

strcpy(name,q->valID.name);

return name;

break;

} else if(id!=q->valID.id&&q->next!=NULL){

q=q->next;

} else if (q->next==NULL){

strcpy(name,"0");

return name;

break;

};

}

}

Aceasta functie returneaza numele studentului in dependenta de ID-ul trimis ca parametru. In program aceasta functie este folosita la afisarea notelor tuturor studentilor, la afisarea notelor pentru un student in particular (functia *printall()*).

* **int getid(char name[30]){**

int i=0,id;

elementID \*a,\*q,\*u;

a=readID();

q=a;

while(1){

if(strcmp(name,q->valID.name)==0){

id=q->valID.id;

return id;

break;

} else if(strcmp(name,q->valID.name)!=0&&q->next!=NULL){

q=q->next;

} else if (q->next==NULL){

return NULL;

break;

};

}

}

Functia data returneaza Id-ul studentului in dependenta de numele trimis ca paramtru. In program este folosita la adaugarea notei unui student (functia *addNot())* si la afisarea notei pentru un student in parte:

printf("Nume student:");

scanf(" %[^\n]",name);

**id=getid(name);**

printall(id, disciplina);

Informatia din fisiere este citita sub forma unei liste simplu inaluita. Fiecarui fisier ii corespunde o functie pentru citire:

* *log.bin*:

struct nod \*readUtil(){

element \*p,\*q,\*u;

FILE \*fp;

int i;

**fp=fopen("log.bin","rb");**

i=0;

while(1){

q=(struct nod\*) malloc(sizeof(struct nod));

**if(fread(&q->val, sizeof(struct nod),1,fp)!=0)**

{

if(i==0)

{

p=q;

u=p;

};

i++;

q->next=NULL;

u->next=q;

u=q;

}

else {break;}

}

**fclose(fp);**

return p;

}

* *id.bin*

struct nodID \*readID(){

int i;

elementID \*q,\*p,\*u;

FILE \*fp;

**fp=fopen("id.bin","rb");**

i=0;

while(1){

q=(struct nodID\*) malloc(sizeof(struct nodID));

**if(fread(&q->valID, sizeof(struct nodID),1,fp)!=0)**

{

if(i==0)

{

p=q;

u=p;

};

i++;

q->next=NULL;

u->next=q;

u=q;

}

else {break;}

}

return p;

fclose(fp);

}

* *SDA.bin*

elementT \*p,\*q,\*u;

FILE \*fp;

int i;

**fp=fopen("sda.bin","rb");**

i=0;

while(1){

q=(struct nodT\*) malloc(sizeof(struct nodT));

**if(fread(&q->valT, sizeof(struct nodT),1,fp)!=0)**

{

if(i==0)

{

p=q;

u=p;

};

i++;

q->next=NULL;

u->next=q;

u=q;

}

else {break;}

}

**fclose(fp);**

return p;

}

**2.2 Structuri de date**

Deoarece tipul de date inscris in fiecare fisier este diferit, este nevoie de crearea mai multor structure dinamice de date, iar pentru acest lucru am folosit mai multe structuri de date. Fiecare structura de date este folosita ca nod al listei inlantuite pentru fiecare fisier.

*Autentificare (log.bin) Lista student (id.bin) Discipline (SDA.b*

struct infoT{

int id;

int nota;

} student;

typedef struct nodT{

struct infoT valT;

struct nodT \*next;

};

typedef struct nodT elementT;

elementT \*disciplina;

struct infoID{

int id;

char name[30];

} std;

typedef struct nodID{

struct infoID valID;

struct nodID \*next;

};

typedef struct nodID elementID;

elementID \*studenti;

struct info{

char login[30];

char parola[30];

int role;

char add[30];

char sort[30];

} utilizator;

typedef struct nod{

struct info val;

struct nod \*next;

};

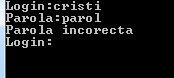
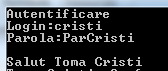
typedef struct nod element;

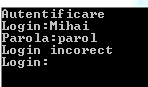
element \*utilizatori;

**2.3 Autentificarea**

Rularea programului incepe cu Autentificarea. Utilizatorului i se cere introducerea datelor de logare pentru a continua (Login/Parola). Acest bloc de program returneaza una din cele 3 posibile variante:

1. In baza de date nu a fost gasit nici un utilizator cu login-ul specificat;
2. Parola pentru utilizatorul dat nu este corecta;
3. Logare reusita. Parola si loginul corespund.





1. Login incorect; b- Parola incorecta; c- Logare reusita.

Autentificarea este realizata cu ajutorul functiei *parse log():*

struct parse log(){

element \*p,\*q,\*u;

FILE \*fp;

int i, role;

char name[30], parola[30];

p=readUtil();

p = mergesort(p);

q=p;

while(1){

printf("Login:");

scanf("%s",name);

if(strcmp(name,"0")==0){

deletelist(p);

exit(0);

}

printf("Parola:");

scanf("%s",parola);

if(strcmp(parola,"0")==0){

deletelist(p);

exit(0);

}

q=search(p,name);

if(q==NULL){

puts("Login incorect");

} else{

if(strcmp(q->val.parola,parola)!=0){

puts("Parola incorecta");

} else if(strcmp(q->val.parola,parola)==0){

strcpy(acces.login,q->val.login);

strcpy(acces.add,q->val.add);

acces.role=q->val.role;

deletelist(p);

break; } } } }

Pentru inceput datele din fisierul *log.bin* sunt citite si stocate intr-o lista dinamica simplu inlantuita: *p=readUtil().* Ulterior, informatia din lista respectiva este sortata dupa valorile din campul “Sort”: *p = mergesort(p).* Datele sunt sortare prin interclasare (Merge sort) – *vezi 3.1.* Dupa ce utilizatorul a introdus Login-ul si Parola, in lista nou creata are loc cautarea binara (*vezi 3.2*) dupa Login. In cazul in care loginul a fost gasit in aceasta lista, se verifica daca parola introdusa corespunde cu cea specificata. In dependenta de rezultatul obtinut, programul afiseaza una din cele 3 posibilitati redate mai sus. Inainte de a iesi din program sau a trece mai departe spre menu, lista create este stearsa din memoria operativa a calculatorului: deletelist(p);

Datele despre fiecare utilizator sunt stocate in fisierul de tip binar: *log.bin* sub forma de structura. Fiecare utilizator are un camp pentru | Login | Parola | Role | Add | Sort |

*struct info{*

*char login[30];*

*char parola[30];*

*int role;*

*char add[30];*

*char sort[30];*

*} utilizator;*

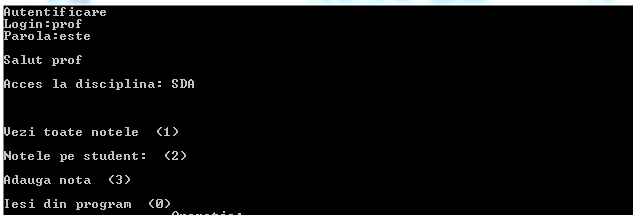
Fiecare utilizator are un rol specificat (1,2 sau 3) si fiecarui rol ii sunt atribuite diferite posibilitati in cadrul programului

**2.4 Tipuri de utilizatori**

**Rol 1:** Rolul de administrator. Poate efectua orice operatie posibila in toate fisierele programului. Poate adauga, modifica date in toate cele 3 compartimente: Utilizatori | Studenti | Disciplina

C:\Users\Krsty\Desktop\Capture.PNG

**Rol 2:** Rolul de profesor. Poate efectua operatii doar in cadrul disciplinei specificate:



Fiecarui utilizator cu rolul 2 ii este atribuit o disciplina in cadrul campului *Add*.

**Rol 3:** Student. Utilizatorul cu acest rol poate doar sa vizualizeze notele sale. In cadrul campului *Add* este specificat numele complet al studentului dupa care sunt cautate notele in fisierul disciplinei respective.

void main(){

puts("Autentificare");

log();

switch(acces.role){

case 1:

admin();

break;

case 2:

profesor();

break;

case 3:

elev();

break;

default:

break;

}

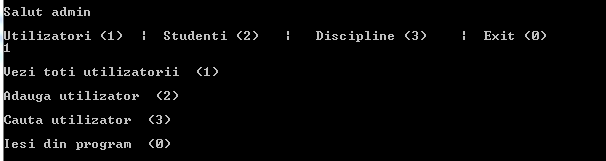
}

**3.1** **Rulare Program. Instructiuni pentru utilizator**

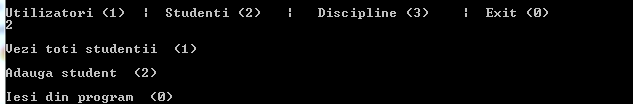
**3.1 Rularea programului ca Administrator**

In cadrul programului prezentat, administratorul are cele mai multe posibilitati si poate efectua toate operatiile posibile in toate fisierele disponibile. Oricare utilizator cu “Rol 1” este administrator. Operatiile posibile sunt clasificate in 3 categorii:

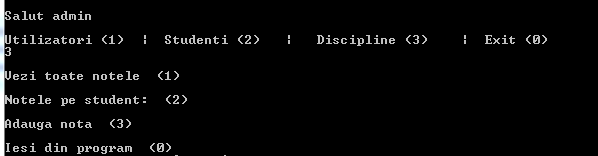
**Utilizatori :**



**Studenti:**

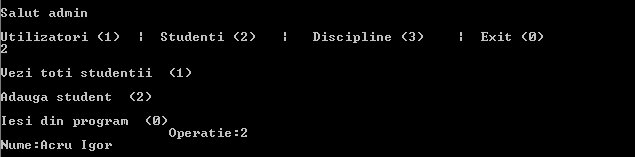
****

**Disciplina:**

****

Pentru a exemplifica operatiile posibile ca administrator, vom simula un caz de adaugarea a unui nou student si informatiile despre el.

Pentru inceput, vom adauga un nou student in rubrica “Studenti”:



Astfel, in fisierul *id.bin* a fost adaugat un nou student cu un ID unic. Id-ul pentru fiecare student introdus este selectat automat de catre program in felul urmator:

In cazul in care fisierul este gol, ID-ul este “1”, in caz contrar, cu ajutorul functiei *getmaxid(elementID \*p)* este indentificat Id-ul cu valoarea cea mai mare, pentru ca noului student sa-i fie atribuit ID-ul gasit incrementat cu o unitate.

void addStd(elementID \*p){

int i,id;

FILE \*fp;

elementID \*q;

fp=fopen("id.bin","ab+");

if(empty(fp)) {

std.id=1;

} else{

std.id=getmaxid(p)+1;

}

printf("Nume:");

scanf(" %[^\n]",std.name);

fwrite(&std,sizeof(struct nodID),1,fp);

fclose(fp);

}

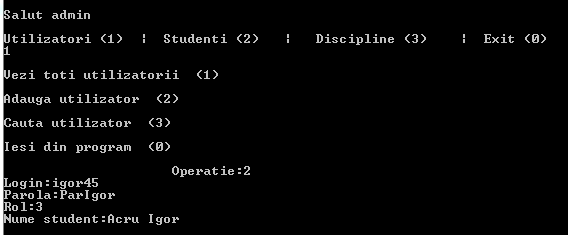
In acest mod, studentii sunt sortati automat dupa ID. In acest caz, pentru cautarea unui student dupa ID se poate folosi cautarea prin interpolare.

Vedem ca studentul “Acru Igor” a fost adaugat in lista studentilor cu ID-ul 5 si plasat la capatul listei.



Ulterior, urmeaza sa adaugam un nou utilizator pentru acest student pentru ca el sa-si poata vizualiza notele. Pentru aceasta selectam optiunea “Adauga utilizator” din rubrica “Utilizatori” si introducem informatia necesara.

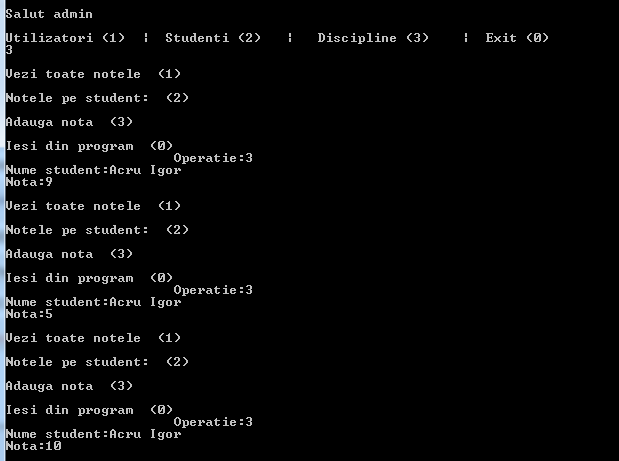
Ca login, putem folosi orice combinatie de caractere posibila. In exemplu dat am folosit o combinatie de litere si cifre. Deoarece dorim sa-i atribuim noului utilizator un rol de student, vom specifica acest lucru prin introducerea cifrei “3” in campul Rol, iar apoi ne vom asigura ca numele deplin a studentul corespunde cu cel introdus anterior in cadrul fisierului *id.bin.*



Dupa cum vedem in imaginea atasata, utilizatorul introdus a fost adaugat si afisat deja in mod sortat:



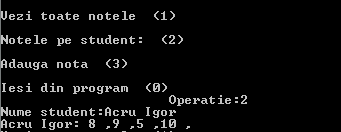
Acum putem adauga note studentului respectiv prin accesarea optiunii corespunzatoare din rubrica “Discipline”:



In acest moment, utilizatorului Acru Igor i-au fost atribuite mai multe note in fisierul SDA.bin, note pe care le putem vizualiza atat in lista tuturor studentilor:

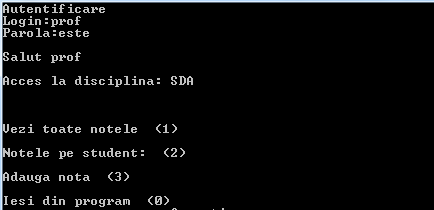
C:\Users\Krsty\Desktop\Capture.PNG

Cat si pentru studentul dat in parte:

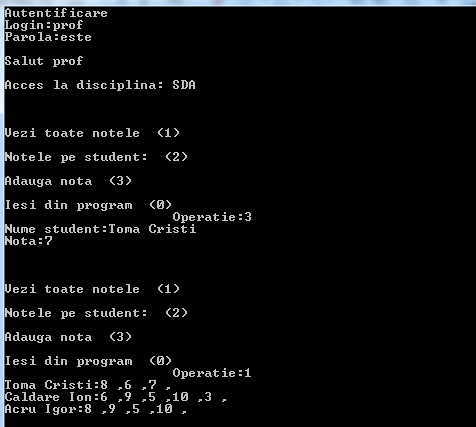


**3.2 Rularea programului ca Profesor**

Utilizatorul cu Rol: 2 poate rula programul in calitate de profesor si poate efectua modificari doar in fisierul disciplinei atribuite.

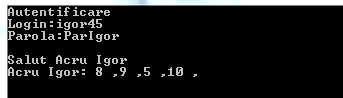


Utilizatorul dat poate doar sa adauge note studentilor in fisierul *SDA.bin* si vizualiza notele disciplinei respective.



**3.3 Rularea programului ca Student**

Utilizatorului cu Rol:3 ii sunt oferite drepturile doar de a-si vizualiza notele la disciplina respectiva:



**Concluzie**

In lucrarea de curs elaborata am modelat pe cat de posibil un proces real – procesarea datelor la o institutie de invatamant. Am elaborat un program in limbajul de programare C care face posibila inregistrarea, stocarea si procesarea datelor despre studenti/utilizatori cu ajutorul fisierelor binare. Scopul lucrarii date a fost de a insusi procedeele de programare, de a implimenta diversi algoritmi de sortare/cautare, de a lucra cu fisierele binare si efectua legatura dintre functii.

Programul are o marime de ~ 670 randuri si 17 KB. Timpul de compilare este mai mic de 3 secunde. Pe langa fisierul de executie (.exe) programul trebuie sa fie insotit de cel putin un fisier (log.bin) cu care este efectuata autentificarea.

Programul dat nu poate fi implimentat direct in practica si constituie un simplu model pentru un viitor registru electronic.

Acest program poate fi imbunatatit prin adaugarea unor noi module, functii si librarii. Lista imbunatatirilor ce pot fi aduse registrului prezentat:

1. Stocarea informatiei intr-o baza de date de tip SQL cu ajutorul modulului “*MySQL Connector/C++ driver*” sau “*SQLite*”.
2. Stocarea informatiei in mediul online cu ajutorul tehnicii “TCP Holepunch”.
3. Implimentarea unui sistem de autentificare folosind libraria Boost.Log.
4. Implimentarea unui sistem graphic pentru menu.
5. Procesarea paralela a datelor.
6. Monitorizarea activitatii fiecarui utilizator intr-un fisier de tip history.txt.

**ANEXA**

**Anexa Nr.1: Listing program**

#include <stdio.h>

#include <stdlib.h>

struct info{

char login[30];

char parola[30];

int role;

char add[30];

char sort[30];

} utilizator;

typedef struct nod{

struct info val;

struct nod \*next;

};

typedef struct nod element;

element \*utilizatori;

struct parse{

char login[30];

int role;

char add[30];

}acces;

struct infoID{

int id;

char name[30];

} std;

typedef struct nodID{

struct infoID valID;

struct nodID \*next;

};

typedef struct nodID elementID;

elementID \*studenti;

struct infoT{

int id;

int nota;

} student;

typedef struct nodT{

struct infoT valT;

struct nodT \*next;

};

typedef struct nodT elementT;

elementT \*disciplina;

int menu,id,i,n;

char name[30];

struct nod \*mergesort(struct nod \*head);

struct nod \*merge(struct nod \*head\_one, struct nod \*head\_two);

int lungime(element \*p);

struct nod \*search(struct nod \*p, char \*name[30]);

void deletelist(struct nod\* p);

struct nod \*readUtil();

struct parse log();

int empty(FILE \*fp);

int getmaxid(elementID \*p);

struct nodID \*readID();

int getid(char name[30]);

char getname(int id,char \*name);

void printall(int id, elementT \*p);

void getutilizatori(element \*p);

void addUtil();

void getstudenti(elementID \*p);

struct nodT \*readDisc();

int idexist(int i,elementT \*p);

int getmaxidDisc(elementT \*p);

void addNot();

void admin();

void profesor();

void elev();

void main(){

puts("Autentificare");

log();

switch(acces.role){

case 1:

admin();

break;

case 2:

profesor();

break;

case 3:

elev();

break;

default:

break;

}

}

// ------------------ Functii ---------------------

struct parse log(){

element \*p,\*q,\*u;

FILE \*fp;

int i, role;

char name[30], parola[30];

p=readUtil();

p = mergesort(p);

q=p;

while(1){

printf("Login:");

scanf("%s",name);

if(strcmp(name,"0")==0){

deletelist(p);

exit(0);

}

printf("Parola:");

scanf("%s",parola);

if(strcmp(parola,"0")==0){

deletelist(p);

exit(0);

}

q=search(p,name);

if(q==NULL){

puts("Login incorect");

} else{

if(strcmp(q->val.parola,parola)!=0){

puts("Parola incorecta");

} else if(strcmp(q->val.parola,parola)==0){

strcpy(acces.login,q->val.login);

strcpy(acces.add,q->val.add);

acces.role=q->val.role;

deletelist(p);

break;

}

}

}

}

struct nod \*mergesort(struct nod \*head) {

struct nod \*head\_one;

struct nod \*head\_two;

if((head == NULL) || (head->next == NULL))

return head;

head\_one = head;

head\_two = head->next;

while((head\_two != NULL) && (head\_two->next != NULL)) {

head = head->next;

head\_two = head->next->next;

}

head\_two = head->next;

head->next = NULL;

return merge(mergesort(head\_one), mergesort(head\_two));

}

struct nod \*merge(struct nod \*head\_one, struct nod \*head\_two) {

struct nod \*head\_three;

if(head\_one == NULL)

return head\_two;

if(head\_two == NULL)

return head\_one;

if(strcmp(head\_one->val.sort , head\_two->val.sort)<0) {

head\_three = head\_one;

head\_three->next = merge(head\_one->next, head\_two);

} else {

head\_three = head\_two;

head\_three->next = merge(head\_one, head\_two->next);

}

return head\_three;

}

int lungime(element \*p){

element \*q;

int k=0;

q=p;

while(q!=NULL){

q=q->next; k++;

};

return k;

};

struct nod \*search(struct nod \*p, char \*name[30]){

int low,high,mid,i;

element \*Pmid,\*Phead,\*ptr;

low=0;

Phead=p;

high=lungime(p)-1;

ptr=NULL;

while(low<=high)

{

mid=(low+high)/2;

Pmid=Phead;

for(i=0;i<mid;i++)

{

Pmid=Pmid->next;

}

if(strcmp(Pmid->val.login,name)==0)

{

ptr=Pmid;

break;

}

else

{

if(strcmp(Pmid->val.login,name)>0)

{

high=mid-1;

}

else

{

if(strcmp(Pmid->val.login,name)<0)

{

low=mid+1;

}

}

}

}

return ptr;

}

void deletelist(struct nod\* p){

struct nod \*aux;

do{

aux=p->next;

p->next=NULL;

free(p);

}while(aux!=NULL);

}

struct nod \*readUtil(){

element \*p,\*q,\*u;

FILE \*fp;

int i;

fp=fopen("log.bin","rb");

i=0;

while(1){

q=(struct nod\*) malloc(sizeof(struct nod));

if(fread(&q->val, sizeof(struct nod),1,fp)!=0)

{

if(i==0)

{

p=q;

u=p;

};

i++;

q->next=NULL;

u->next=q;

u=q;

}

else {break;}

}

fclose(fp);

return p;

}

int empty(FILE \*fp){

fseek(fp, 0, SEEK\_END);

if (ftell(fp) == 0){

return 1;

} else {

fseek(fp, 0, SEEK\_SET);

return 0;}

}

int getmaxid(elementID \*p){

elementID \*q;

int max;

q=p;

max=q->valID.id;

while(q->next!=NULL){

if(q->valID.id<q->next->valID.id){

max=q->next->valID.id;

}

q=q->next;

}

return max;

}

struct nodID \*readID(){

int i;

elementID \*q,\*p,\*u;

FILE \*fp;

fp=fopen("id.bin","rb");

i=0;

while(1){

q=(struct nodID\*) malloc(sizeof(struct nodID));

if(fread(&q->valID, sizeof(struct nodID),1,fp)!=0)

{

if(i==0)

{

p=q;

u=p;

};

i++;

q->next=NULL;

u->next=q;

u=q;

}

else {break;}

}

return p;

fclose(fp);

}

int getid(char name[30]){

int i=0,id;

elementID \*a,\*q,\*u;

a=readID();

q=a;

while(1){

if(strcmp(name,q->valID.name)==0){

id=q->valID.id;

return id;

break;

} else if(strcmp(name,q->valID.name)!=0&&q->next!=NULL){

q=q->next;

} else if (q->next==NULL){

return NULL;

break;

};

}

}

char getname(int id,char \*name){

int i=0;

elementID \*a,\*q,\*u;

a=readID();

q=a;

while(1){

if(id==q->valID.id){

strcpy(name,q->valID.name);

return name;

break;

} else if(id!=q->valID.id&&q->next!=NULL){

q=q->next;

} else if (q->next==NULL){

strcpy(name,"0");

return name;

break;

};

}

}

void printall(int id, elementT \*p) {

char nume[30];

elementT \*q;

getname(id,nume);

if(strcmp(nume,"0")==0){

puts("Nu a fost gasit student-ul");

exit(0);

}

printf("%s: ",nume);

q=p;

while(q!=NULL){

if(q->valT.id==id){

printf("%d ,",q->valT.nota);

}

q=q->next;

}

}

void getutilizatori(element \*p){

element \*q,\*u;

q=p;

while(q!=NULL){

printf("Login:%s\nParola:%s\nRol:%d\nAdd:%s\n",q->val.login,q->val.parola,

q->val.role,q->val.add);

puts("--------------------------------------------------");

q=q->next;

}

}

void addUtil(){

FILE \*fp;

fp=fopen("log.bin","ab+");

printf("Login:");

scanf("%s",utilizator.login);

strcpy(utilizator.sort, utilizator.login);

printf("Parola:");

scanf("%s",utilizator.parola);

printf("Rol:");

scanf("%d",&utilizator.role);

if(utilizator.role==2){

printf("Disciplina:");

scanf(" %[^\n]",utilizator.add);

}else if(utilizator.role==3){

printf("Nume student:");

scanf(" %[^\n]",utilizator.add);

};

fwrite(&utilizator,sizeof(struct nod),1,fp);

fclose(fp);

}

void getstudenti(elementID \*p){

elementID \*q,\*u;

q=p;

while(q!=NULL){

printf("%d: %s\n",q->valID.id,q->valID.name);

puts("");

q=q->next;

}

}

void addStd(elementID \*p){

int i,id;

FILE \*fp;

elementID \*q;

fp=fopen("id.bin","ab+");

if(empty(fp)) {

std.id=1;

} else{

std.id=getmaxid(p)+1;

}

printf("Nume:");

scanf(" %[^\n]",std.name);

fwrite(&std,sizeof(struct nodID),1,fp);

fclose(fp);

}

struct nodT \*readDisc(){

elementT \*p,\*q,\*u;

FILE \*fp;

int i;

fp=fopen("sda.bin","rb");

i=0;

while(1){

q=(struct nodT\*) malloc(sizeof(struct nodT));

if(fread(&q->valT, sizeof(struct nodT),1,fp)!=0)

{

if(i==0)

{

p=q;

u=p;

};

i++;

q->next=NULL;

u->next=q;

u=q;

}

else {break;}

}

fclose(fp);

return p;

}

int idexist(int i,elementT \*p){

elementT \*q;

int k=0;

q=p;

while(q!=NULL){

if(q->valT.id==i){

k=1;

break;

}

q=q->next;

}

if(k==1){

return 1;

} else {return 0;};

}

int getmaxidDisc(elementT \*p){

elementT \*q;

int max;

q=p;

max=q->valT.id;

while(q->next!=NULL){

if(q->valT.id<q->next->valT.id){

max=q->next->valT.id;

}

q=q->next;

}

return max;

}

void addNot(){

FILE \*fp;

char name[30];

fp=fopen("sda.bin","ab+");

printf("Nume student:");

scanf(" %[^\n]",name);

student.id=getid(name);

printf("Nota:");

scanf("%d",&student.nota);

fwrite(&student,sizeof(struct nodT),1,fp);

fclose(fp);

}

void admin(){

element \*q;

elementT \*qt,\*qi,\*qid;

printf("\nSalut %s\n",acces.login);

printf("\nUtilizatori (1) | Studenti (2) | Discipline (3) | Exit (0)\n");

scanf("%d",&menu);

switch(menu){

case 1:

while(1){

utilizatori=readUtil();

printf("\nVezi toti utilizatorii (1)\n");

printf("\nAdauga utilizator (2)\n");

printf("\nCauta utilizator (3)\n");

printf("\nIesi din program (0)\n");

printf("\n Operatie:");

scanf("%d",&menu);

switch(menu){

case 1:

utilizatori=mergesort(utilizatori);

getutilizatori(utilizatori);

break;

case 2:

addUtil();

break;

case 3:

printf("Login:");

scanf("%s",name);

q=search(utilizatori,name);

if(q==NULL){

printf("Nu a fost gasit nici un utilizator cu login-ul: %s",name);

} else{

printf("\nParola:%s\nRol:%d\nAdd:%s\n",q->val.parola,

q->val.role,q->val.add);

}

break;

case 0:

exit(0);

default:

break;

}

}

case 2:

while(1){

studenti=readID();

printf("\nVezi toti studentii (1)\n");

printf("\nAdauga student (2)\n");

printf("\nIesi din program (0)\n");

printf(" Operatie:");

scanf("%d",&menu);

switch(menu){

case 1:

getstudenti(studenti);

break;

case 2:

addStd(studenti);

break;

case 0:

exit(0);

default:

break;

}

}

case 3:

while(1){

disciplina=readDisc();

printf("\nVezi toate notele (1)\n");

printf("\nNotele pe student: (2)\n");

printf("\nAdauga nota (3)\n");

printf("\nIesi din program (0)\n");

printf(" Operatie:");

scanf("%d",&menu);

switch(menu){

case 1:

qi=qid=disciplina;

n=getmaxidDisc(disciplina);

for(i=1;i<=n;i++){

if(idexist(i,disciplina)){

getname(i,name);

printf("%s:",name);

qid=disciplina;

while(qid!=NULL){

if(qid->valT.id==i){

printf("%d ,",qid->valT.nota);

}

qid=qid->next;

}

puts("");

}

}

break;

case 2:

printf("Nume student:");

scanf(" %[^\n]",name);

id=getid(name);

printall(id, disciplina);

break;

case 3:

addNot();

break;

case 0:

exit(0);

break;

}

}

}

}

void profesor(){

element \*q;

elementT \*qt,\*qi,\*qid;

printf("\nSalut %s\n",acces.login);

if(acces.add==NULL){

printf("Nu ai acces la nici o discplina");

exit(0);

}

printf("\nAcces la disciplina: %s\n",acces.add);

while(1){

disciplina=readDisc();

printf("\n\n\nVezi toate notele (1)\n");

printf("\nNotele pe student: (2)\n");

printf("\nAdauga nota (3)\n");

printf("\nIesi din program (0)\n");

printf(" Operatie:");

scanf("%d",&menu);

switch(menu){

case 1:

qi=qid=disciplina;

n=getmaxidDisc(disciplina);

for(i=1;i<=n;i++){

if(idexist(i,disciplina)){

getname(i,name);

printf("%s:",name);

qid=disciplina;

while(qid!=NULL){

if(qid->valT.id==i){

printf("%d ,",qid->valT.nota);

}

qid=qid->next;

}

puts("");

}

}

break;

case 2:

printf("Nume student:");

scanf(" %[^\n]",name);

id=getid(name);

printall(id, disciplina);

break;

case 3:

addNot();

break;

case 0:

exit(0);

break;

}

}

}

void elev(){

disciplina=readDisc();

printf("\nSalut %s\n",acces.add);

id=getid(acces.add);

printall(id, disciplina);

getch();

}

**Bibliografie**

1. **ro.wikipedia.org** - *Enciclopedie online*
2. [**www.tutorialeprogramare.ro**](http://www.tutorialeprogramare.ro) - *Website cu tutoriale de programare*
3. **Totul despre C si C++. Manualul fundamental de programare in C si C++**  | *de* [*Dr. Kris Jamsa Lars Klander*](http://www.librarie.net/autor/1393/dr-kris-jamsa-lars-klander)
4. **Jamsa's C/C++ Programmer's Bible**
5. [**www.istrati.com/programming**](http://www.istrati.com/programming)
6. [**www.cplusplus.com**](http://www.cplusplus.com/)
7. [**www.cppreference.com**](http://www.cppreference.com)