Univesitatea „Ștefan cel Mare” Suceava
Facultatea de Inginerie Electrică și Știința Calculatoarelor

Proiect Programarea Orientată pe Obiect

Gestiunea unei Biblioteci

Student: Chiperi Alin-Ioan

Grupa:3122A

Profesor Coordonator:Ștefănescu Narcisa

Cuprins

I. Introducere

I.1 Descrierea proiectului

Tema proeictului are la bază proiectarea unei aplicații ca realizează gestiunea unei bibilioteci. Aplicația a fost gândită astfel încât sa fie utilă atât studenților dar și administratorilor și biblotecarilor.

Din persepectiva utilizatorului de tip „Student”, acesta are mai multe optiuni printre care: vizualizarea cartilor din biblioteca, a căuta o anumită carte, sau a împrumuta.

Din perspectiva administratorului, acesta își poate gestiona mult mai ușor biblioteca folosind opțiunele din meniul lui, printre care amintim: adăugare de carte, vizualizare studenti, vizualizare carți împrumutate și data acestora.

II.2 Noțiuni teoretice si tehnologii folosite

Aplicația „Gestiunea unei Bibloteci ” a fost dezvloltată cu ajutorul limbajului de programare C++, utilizând conceptele Programării Orientate pe Obiect cum ar fi încapsularea și polimorfismul. Cu ajutorul claselor implementate, a proprietăților și metodelor acestora, dar și a lucrurlui cu fișiere, proiectul a prins contur, iar utilizarea aplicației este una facilă la îndemana utilizatorului.

Proiectul a fost dezvoltat utilizând mediul **IDE Code::Blocks 20.03,** compilatorul GCC, iar sistemul de operare Windows 10.

II Schița proiectului

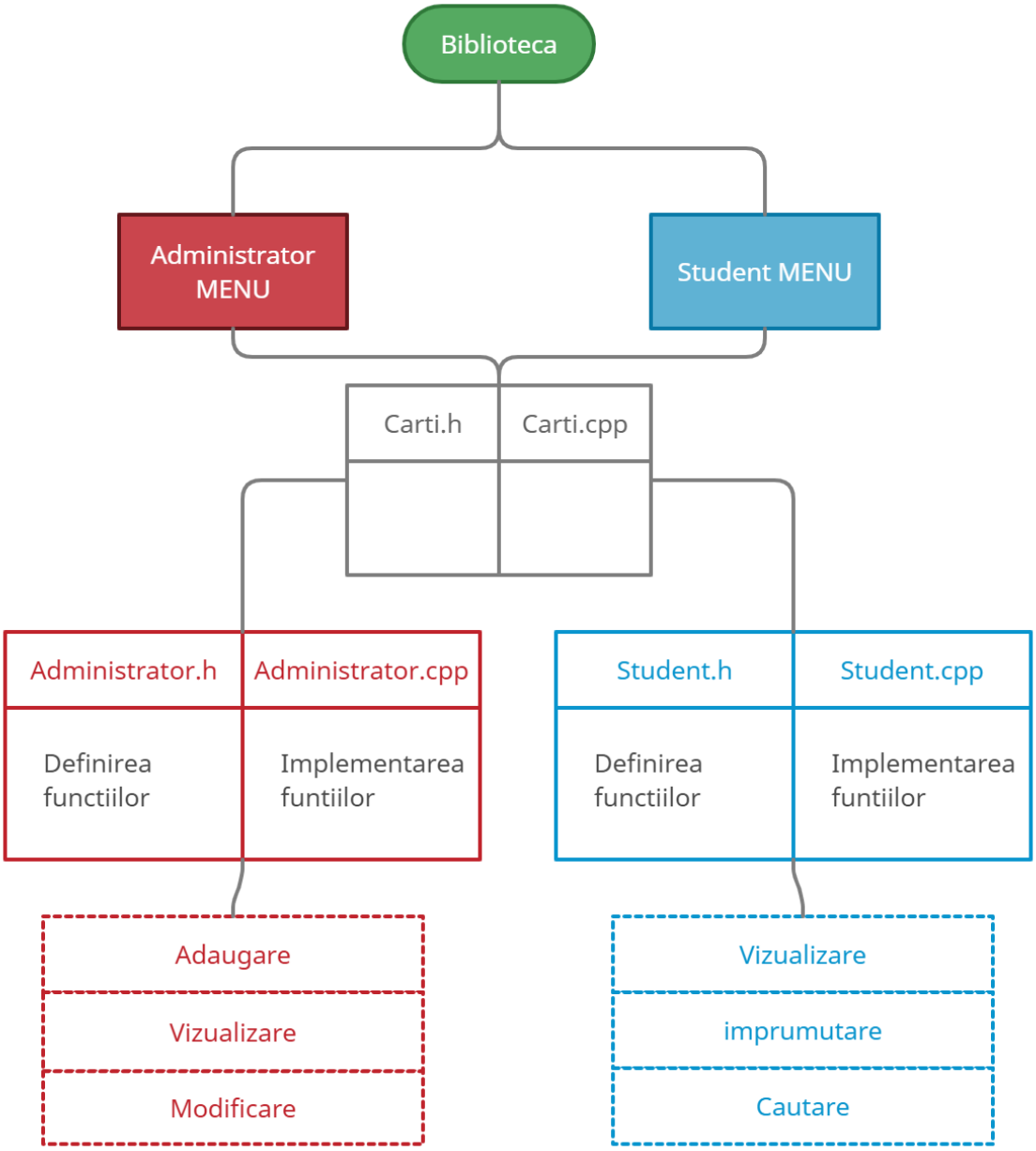

Fig.1- *Schema proiectului*

III Implementare

Pentru realizarea aplicației am implementat trei clase:

- Clasa Carte
- Clasa Administrator
- Clasa Student

Deși clasele sunt independente din punct de vedere a implementării întrucât nu exista o relație de moștenire, ele lucrează corelat, iar legăturile dintre ele ajută la definirea produsului final.

III.1 Clasa Carte

Clasa carte cupride atributele (*isbn, denumire, autor, etc.*) și metodele specifice (*constructori, getere și settere ,functii de citire și afișare*)

Aceasta facilitează lucrul cu un obiect de tip Carte, îngloband atributele principale într-o singură entitate .

```
class Carte
{
    char isbn[30], denumire[30], autor[30], editura[30];
    float pret;
    char buff[100];
    public:
        Carte();
        Carte(char _isbn[30], char _denumire[30], char _autor[30], char _editura[30], float _pret);
        inline char* getIsbn() {return isbn;}
        inline char* getDenumire() {return denumire;}
        inline char* getAutor() {return autor;}
        inline char* getEditura() {return editura;}
        inline float getPret(){return pret;}
        char* getDenumireComplet();
        char* ConversieFisier(); /// functie pentru crearea intr-un format specific scrierii in fisier
        /// functii de citire si afisare
        friend istream& operator >> (istream& c, Carte x);
        friend ostream& operator << (ostream& c, Carte x);
```

Fig.2-*Clasa Carte*

III.2 Clasa Administrator

Aceasta clasă este utilizată în principal pentru gestionarea bibliotecii. Cu ajotorul clasei Administrator se realizează funcțiile de adăugare sau vizualizare cărți și studenți . Aceata folosește în mare parte lucrul cu fișiere, astfel informațiile stocate rămân după închiderea utilizării programului, fiind ușor de vizualizat.

Totodată, gestiunea se poate realiza doar de către persoanele autorizate care au user-ul și parola.

```
#define User "admin"
#define Password "admin"
class Administrator
{
    int id;
    public:
        Administrator();
        void AdaugaCarte(Carte c1);
        void StergeCarte();
        void AfisareStudenti();
        void AfisareCarti();
        void CautaStudentId(char ID[50]);
        void ModificaCarte(Carte c1, Carte c2);
        void AfisareCartiImprumutate();
        virtual ~Administrator();
    protected:
```

Fig.3-*Clasa Administrator*

III.3 Clasa Student

Clasa Student reușește să pună în relație obiectele de tip carte cu utilizatorul student. Fiecare student se loghează cu numele, prenumele și emailul în aplicație, iar programul îi va genera automat un cod unic utilizând un membru static ID.

La împrumutarea unei cărți, datele studentului și a cărții se vor scrie într-un fisier separt disponibil pentru vizualizare doar administratorului; totodată, atașat se va insera și data la care acesta o împrumută.

```
class Student
{
    static int nextId;
    char buff[100];
    char nume[30];
    char prenume[30];
    char email[30];
    int id;
    public:
        Student();
        Student( char nume [30],char prenume [30],char email[30]);
        inline char* getNume(){return nume;}
        inline char* getPrenume(){return prenume;}
        inline int getId(){return id;}
        inline char* getEmail(){return email;}
        char* getDenumireStudent();
        char*getnumeComplet();
        char* CautaCarte(char* carte);
        virtual ~Student();
```

Fig.4.-*Clasa Student*

III.4 Main

În header-ul main este implementată interfață cu utilizatorul. Aceasta este constituită din meniul principal, divizat in alte două meniu. În funcție de utilizator, acesta poate alege meniul studentului, sau meniul de administrare.

Totadă, în main.h mai sunt implementate și funcții auxiliare pentru organizarea mai ușoară a datelor și relațiilor dintre clase . Spre exemplu este realizată funcția Data(), cu ajutorul căreia putem obtine data la care un student închiriază o carte.

```
string Data()
{
    time_t tt;
    struct tm * ti;
    time (&tt);
    ti = localtime(&tt);

    return asctime(ti);
}
```

Fig.5-*Funția Data()*

IV. Manual Utilizare

Aplicația dispune de un meniu simplu divizat în funcție de tipul de utilizator

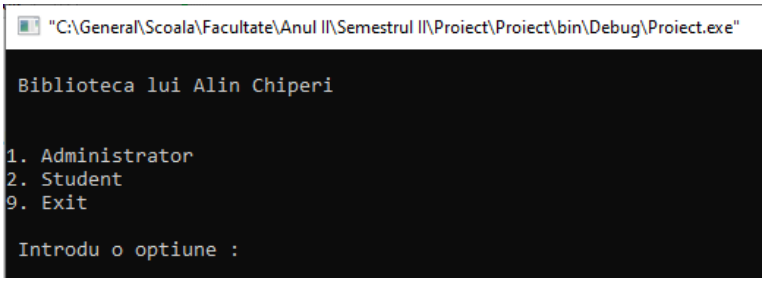

Fig 6- *Main Menu*

Prin apăsarea testei 1, se deschide meniul de logare, iar după verificarea datelor de administrator (User: admin, Password: admin) se deschide meniul ADMINISTRSTOR. Prin apasărea testei 2, se deschide meniul de logare a unui nou student, iar după autentificare, se afișeaza meniul STUDENT. Fiecare dintre dintre cele două meniuri având opțiunele aferente.

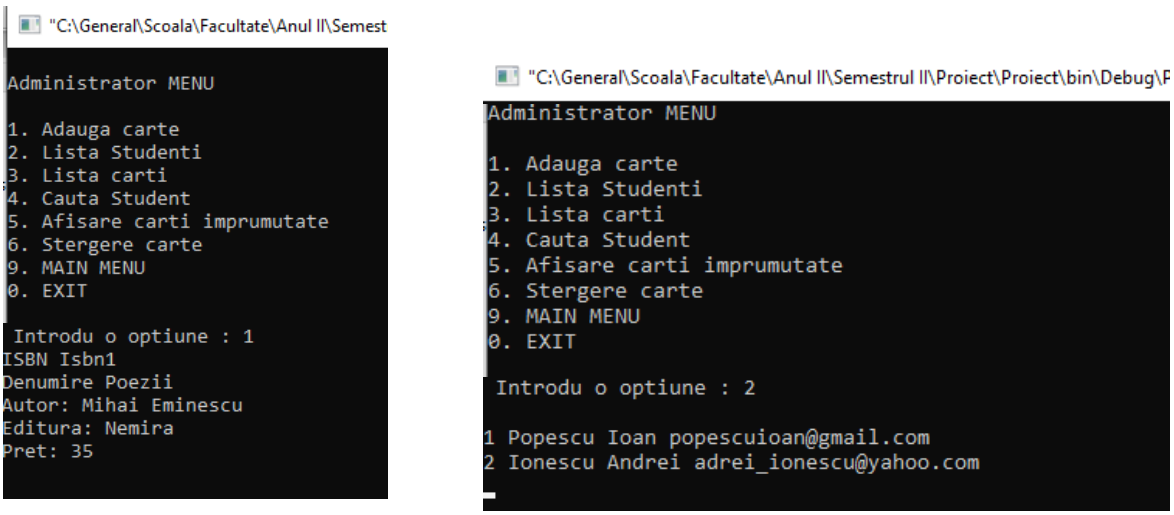

Fig.7a-*Administrator MENU*

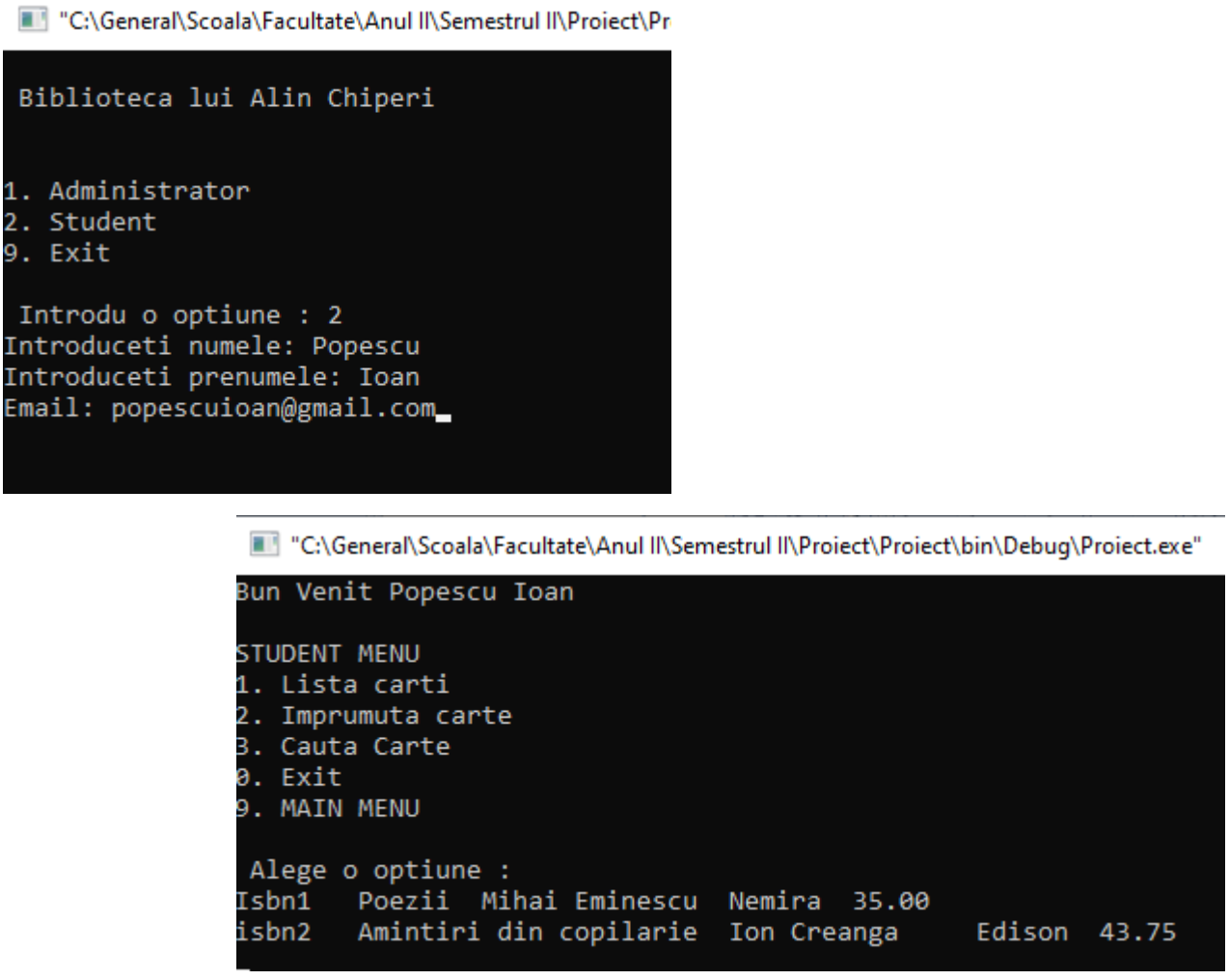

Fig.7b-*Student Menu*

V. Concluzii și direcții de dezvoltare

În cadrul acestui proeict am avut ocazia de a exersa cunoștiințele dobândite în ceea ce privește Programarea Orientată pe Obiect în limbajul c++. Din punctul meu de vedere am reușit să implementez o aplicație de gestiune a unei bibloteci, atât din perspectiva administratorului cât și din cea a studentului.

O dezvoltare viitoare ar fi introducerea unei baze de date întrucat este mult mai facilă în ceea ce privește organizarea datelor, spre deosebire de fișierele text, în care informația este foarte vulnerabilă. Totadă, aș dezvolta și interfața grafice în alte medii de dezvoltare precum Visual Studio care permite realizarea unei interfețe mult mai prietenoase.

VI. Biblliografie

- http://apollo.eed.usv.ro/~remus/
- *Informatică* Sorin Tudor, București, Editura L&S Soft 2008
- https://stackoverflow.com/
- https://www.codeblocks.org/