COLEGIUL NAȚIONAL "CALISTRAT HOGAȘ" TECUCI

TEMĂ ATESTAT

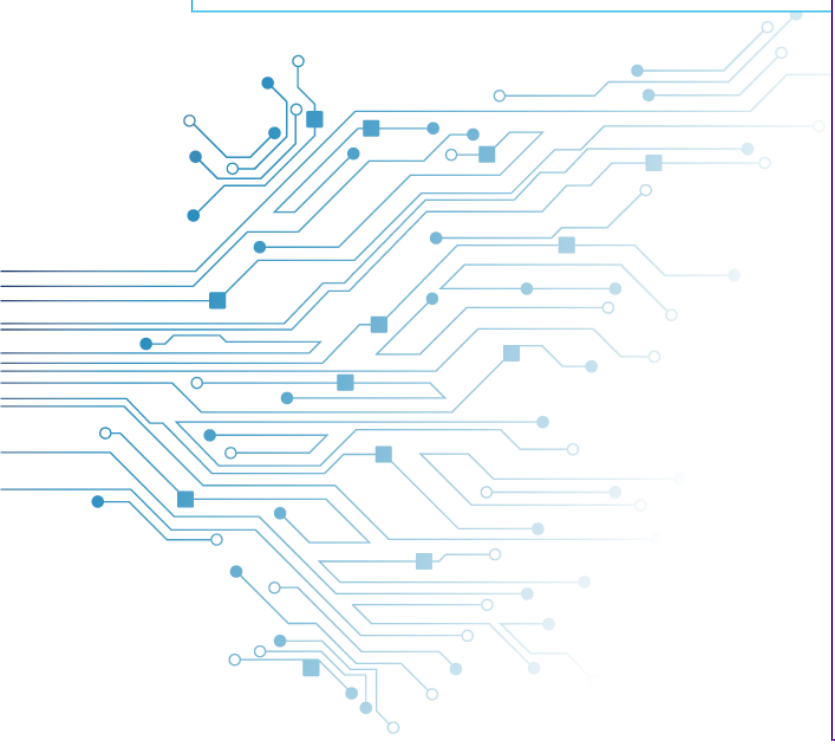
CRIPTAREA ȘI DECRIPTAREA UNUI TEXT FOLOSIND ALGORITMUL CIFRUL LUI CAESAR

Elev: Roșca Ionuț

Clasa: XII A

Profesor îndrumător:

Ciocănel Adriana



Mai 2019

Cuprins

I. Motivarea temei

Curiozitatea este unul din ingredientele cele mai inedite ale speciei umane. Insa, ce se intampla atunci cand cei curiosi au acces la ceea ce nu ar trebui sa aiba? Astfel ia nastere domeniul criptografiei, al carui principal scop este sa ascunda informatia prin utilizarea unor algoritmi matematici ce transforma informatia intr-o forma ce nu mai poate fi inteleasa.

Pasionat de domeniul securitatii cibernetice, consider ca protejarea datelor si intimitatea online sunt de o importanta deosebita, in realizarea acestor obiective sarindu-ne in ajutor criptografia. Fiecare dintre noi interactionam cu domeniul criptografic, indiferent daca suntem sau nu constienti de asta. De exemplu, lacatul de culoare verde pe care il vedem in fereastra browser-ului atunci cand navigam pe internet ne indica faptul ca traficul dintre noi si site-ul web este criptat. De ce este important ca traficul sa fie criptat? Deoarece, in caz contrar, orice persoana conectata la aceeasi retea cu noi ar putea sa vada in totalitate datele pe care noi le comunicam cu site-ul web, inclusive parole sau date bancare.

Prin intermediul acestui program doresc sa prezint si sa exemplific modul de functionare al unui cifru criptografic simplu si interesant, lucru ce ne va ajuta in intelegerea principiilor de baza ale criptografiei.

II. Prezentarea algoritmului: Cifrul lui Caesar

În criptografie, cifrul lui Caesar, numit și cifrul cu deplasare, codul lui Caesar sau deplasarea lui Caesar, este una dintre cele mai simple și mai cunoscute tehnici de criptare. Este un tip de cifru de substituție, în care fiecare literă din textul inițial este înlocuită cu o literă care se află în alfabet la o distanță fixă față de cea înlocuită.Cifrul Caesar este denumit după Iulius Caesar, care, conform Suetoniu, îl folosea cu o deplasare de 3 pentru protejarea mesajelor cu importanță militară

"Dacă avea ceva confidențial de comunicat, scria încifrat, adică schimba ordinea literelor din alfabet, astfel încât nu se putea înțelege nici un cuvânt. Dacă cineva dorește să descifreze și să înțeleagă, trebuie să înlocuiască a patra literă din alfabet, adică D, cu A, și așa mai departe pentru celelalte." — Suetoniu, Viața lui Iulius Caesar 56.

Cum functioneaza ?

Daca am presupune ca valoarea numerica a literelor ar fi:

```
A->1,
B->2,
C->3,
....,
Z->26
```

Atunci procesul de criptare poate fi descris printr-o functie matematica, numita ***Cr(x).*** Practic, literele sunt mutate cu x pozitii catre dreapta.

Criptarea unei litere x (A-Z) cu o cheie n (1-26), poate fi descrisa astfel:

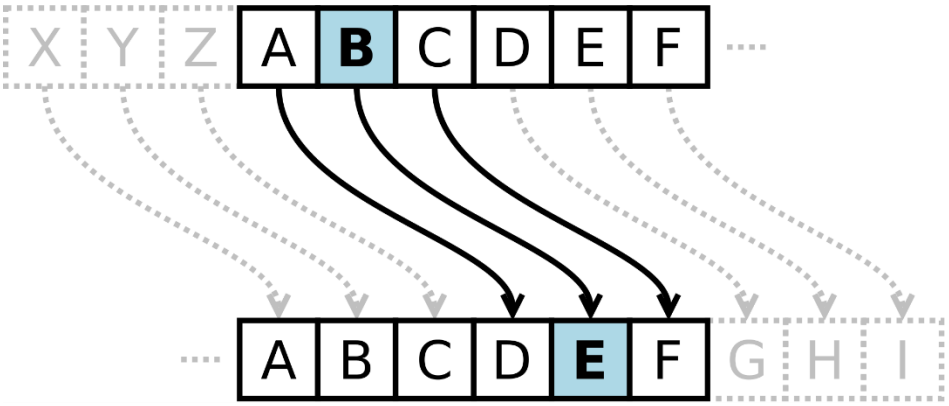
$$Cr(x) = (x + n) \% 26$$

Procesul de decriptare poate fi descris, de asemenea, printr-o functie matematica, numita ***Dr(x).*** Practic, literele sunt mutate cu x pozitii catre stanga.

Decriptarea unei litere x (A-Z) cu o cheie n (1-26), poate fi descrisa astfel:

$$Dr(x) = (x - n) \% 26$$

Exemplu



Text normal (necriptat) → VOM ATACA LA NOAPTE

Text criptat cu cheia 7 → CVT HAHJH SH UVHWAL

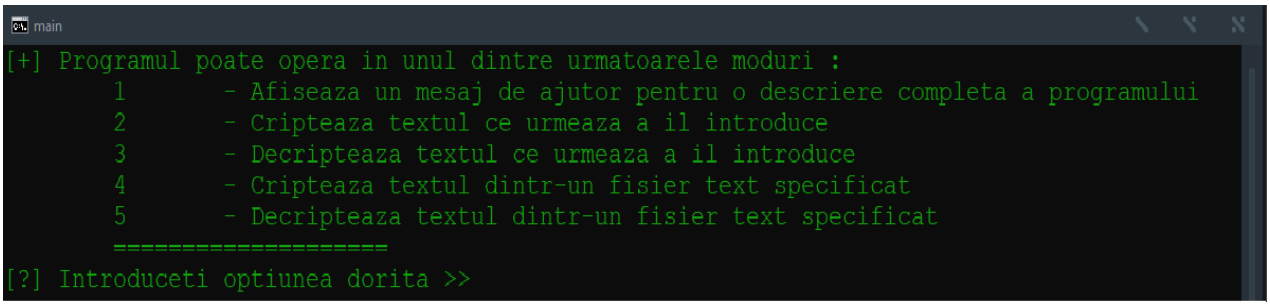
O variatie a cifrului lui Caesar este algoritmul ROT13, care, de fapt, este tot algoritmul cifrul lui Caesar, dar cu cheia fixa 13.

Text necriptat → SUPER SECRET

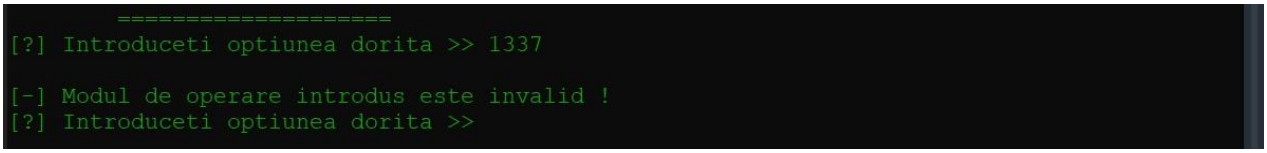
Text criptat → FHCRE FRPERG

III. Prezentarea meniului

Meniul programului este sugestiv: el prezinta utilizatorului optiunile pe care acesta le are la dispozitie. Fiecarei optiune (notata cu o cifra de la 1 la 5) ii este atasata in partea dreapta o scurta descriere.



In cazul in care utilizatorul introduce o optiune gresita, acesta va avea posibilitatea sa introduca din nou optiunea dorita, numarul de incercari fiind limitat la 3 (in afara de prima gresita).



Dupa ce utilizatorul si-a finalizat lucrul, acesta va fi intrebat daca doreste sa se intoarca inapoi la meniu sau daca doreste sa termine executia programului. Raspunsul sau trebuie sa fie litera 'Y' in caz afirmativ, respective litera 'N' in caz contrar.

La fel ca si in cazul anterior, daca utilizatorul introduce gresit optiunea, acesta va avea posibilitatea sa introduca din nou optiunea dorita, intrucat va fi intrebat din nou. Numarul de incercari este limitat la 3.

IV. Prezentarea programului

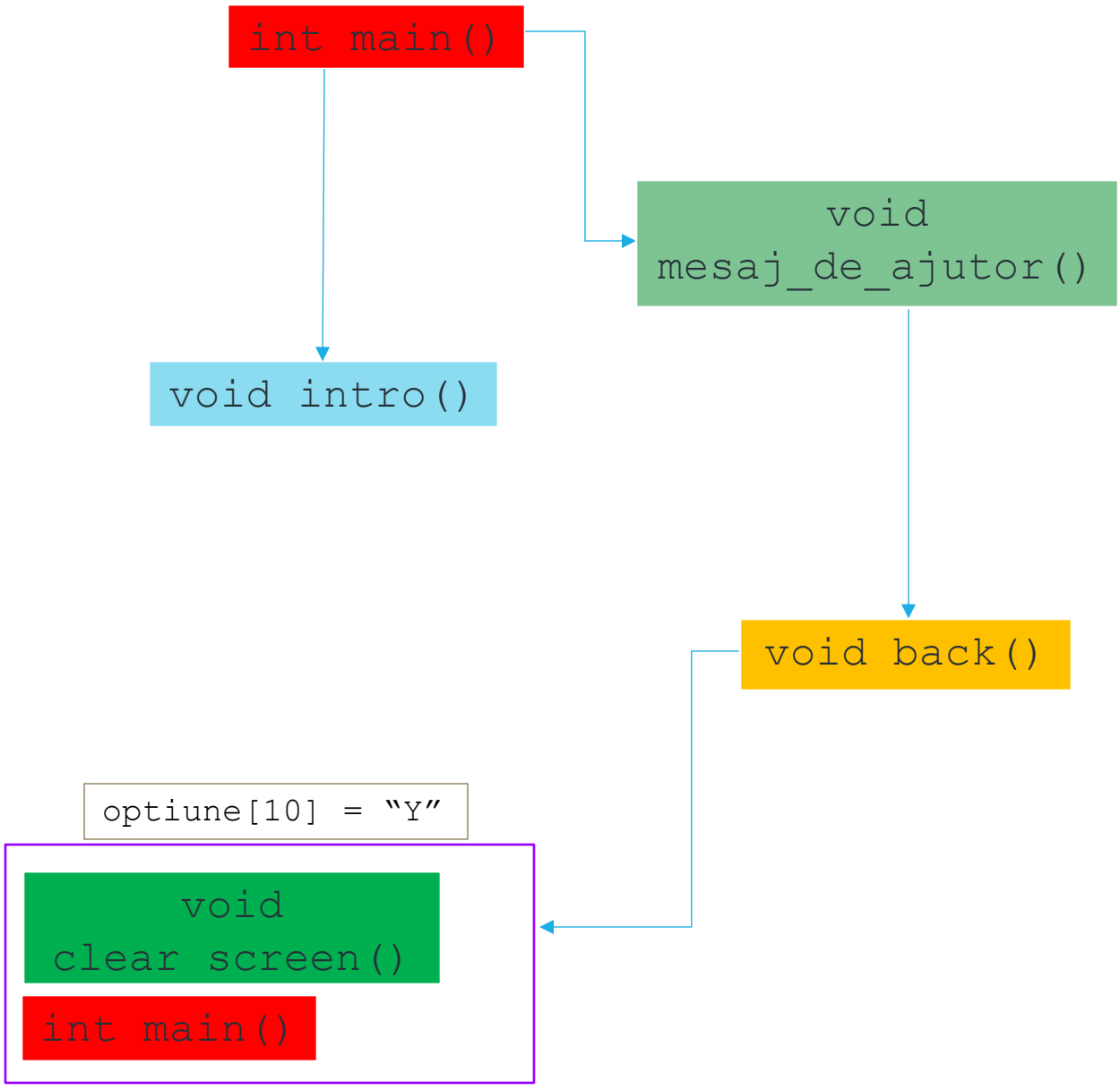
In cadrul programului, functia radacina (int main) poate fi vazuta ca un panou de control, intrucat ea prezinta meniul si directioneaza utilizatorul conform optiunii sale. Fiecarei optiuni din meniu ii corespunde o functie ce va fi apelata pentru indeplinirea sarcinii solicitate, iar unele dintre ele vor apela in continuare alte functii pentru indeplinirea sarcinilor mai mici. Pentru indeplinirea cerintei utilizatorului se vor apela astfel mai multe functii inlantuite.

In cazul in care utilizatorul doreste sa revina la meniul principal pentru efectuarea unei noi actiuni, acesta va fi solicitat sa raspunda corespunzator dorintei sale. Astfel, programul capata un flux continuu, acesta terminandu-si executia doar daca utilizatorul decide acest lucru.

Lantul de apeluri poate fi reprezentat grafic prin intermediul unei scheme logice pentru fiecare optiune in parte.

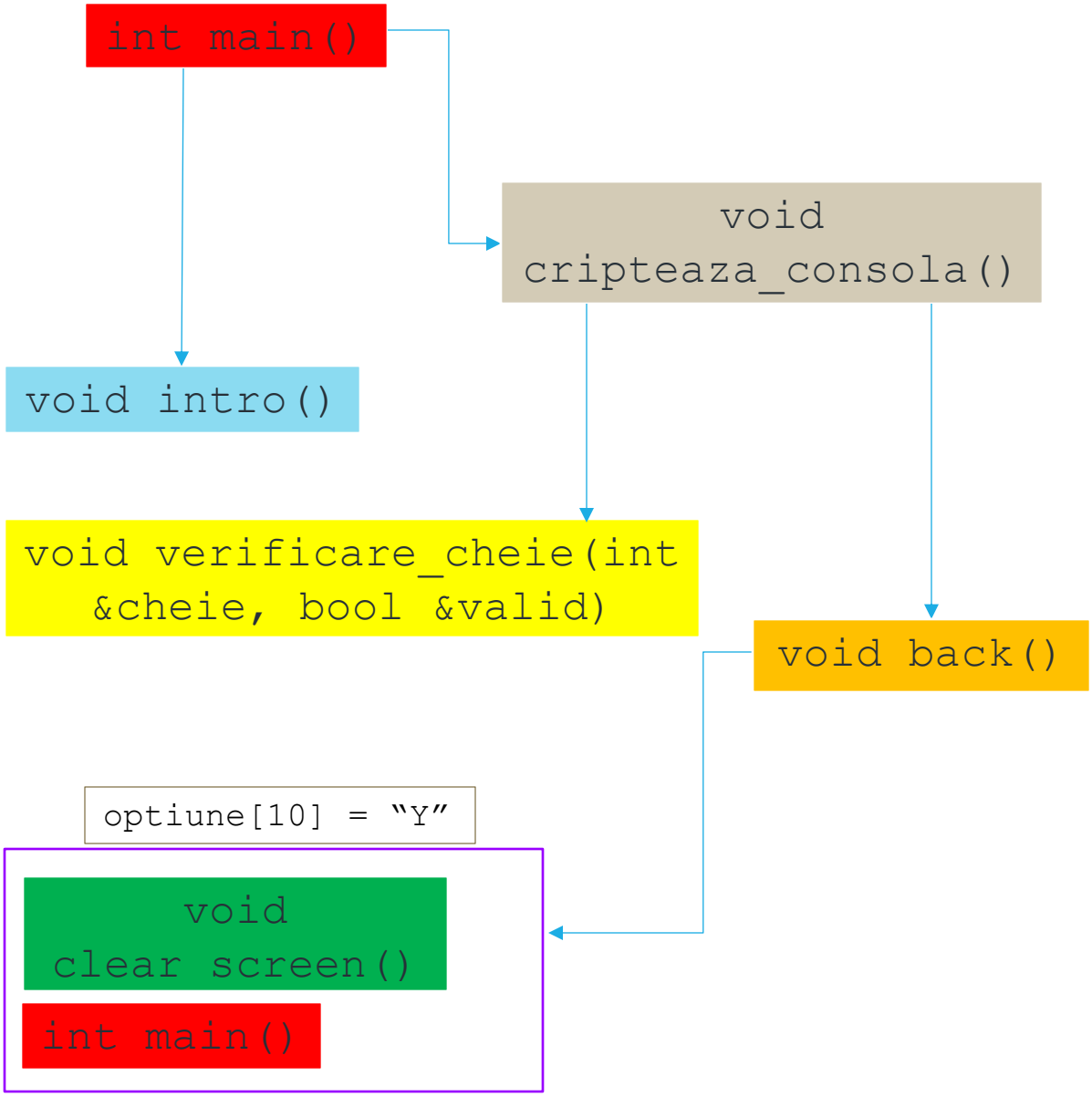
- Optiunea nr. 1

```
cin >> mod;        // 1
```



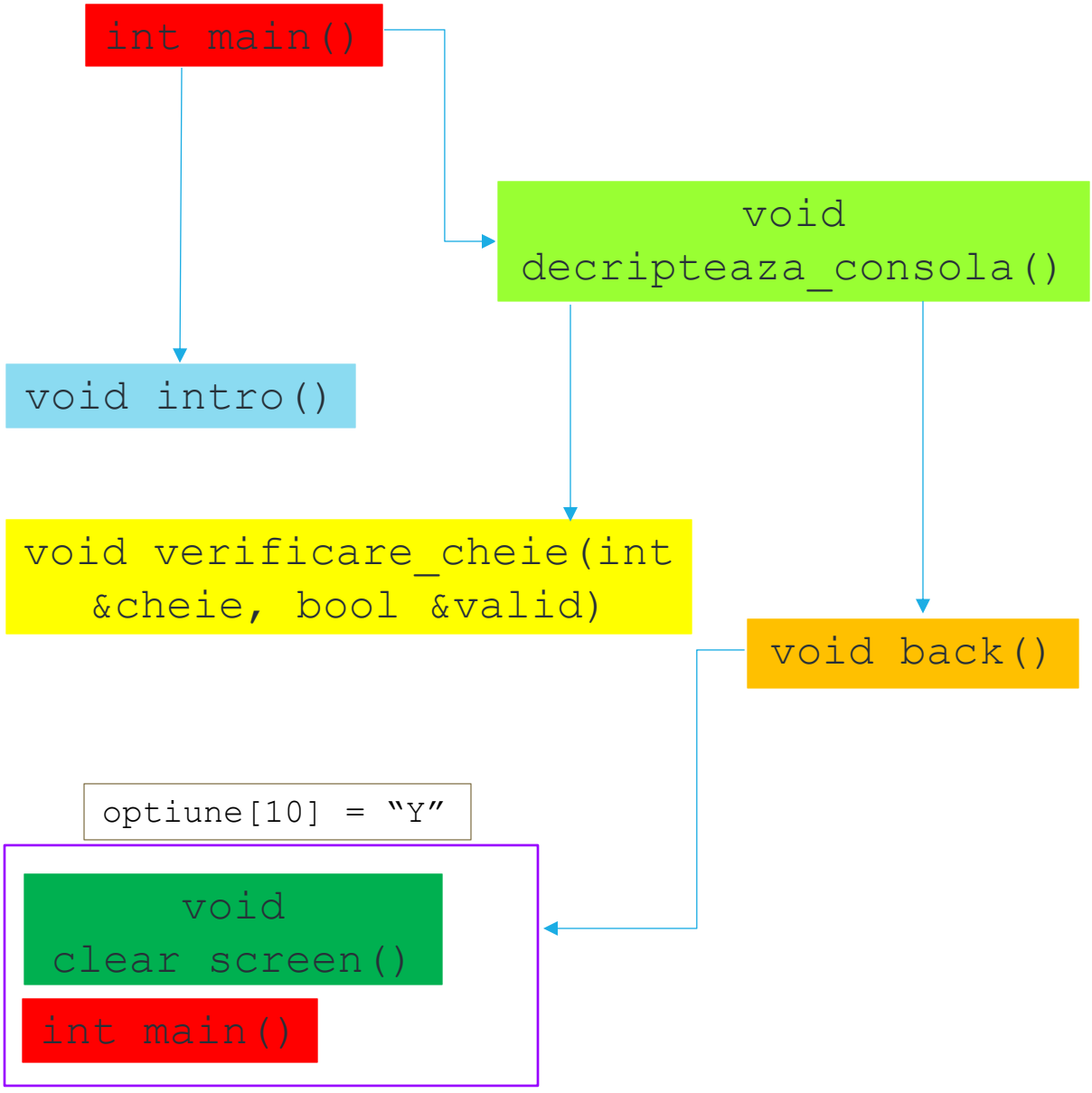
- Optiunea nr. 2

```
cin >> mod;        // 2
```



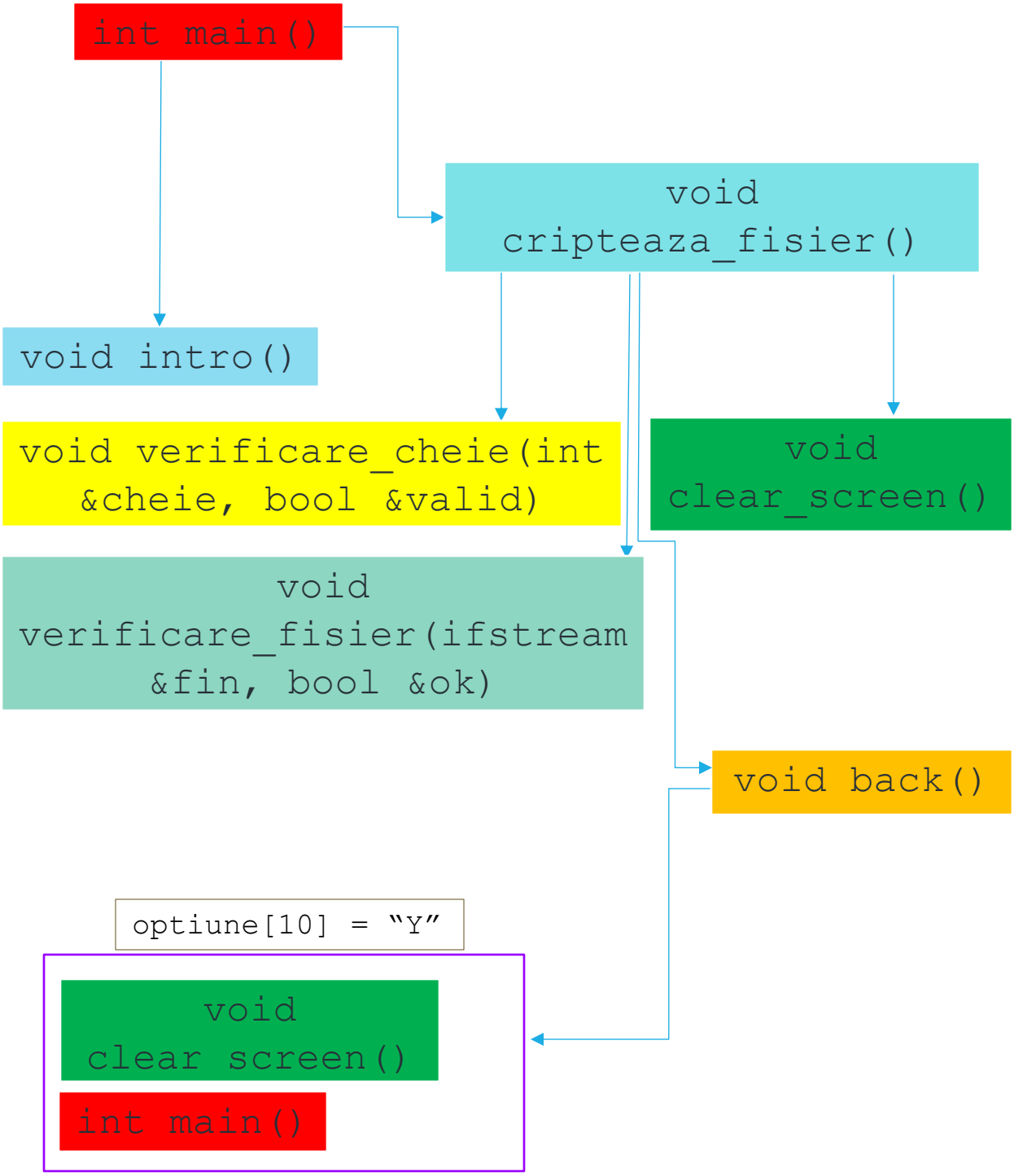
- Optiunea nr. 3

```
cin >> mod;        // 3
```



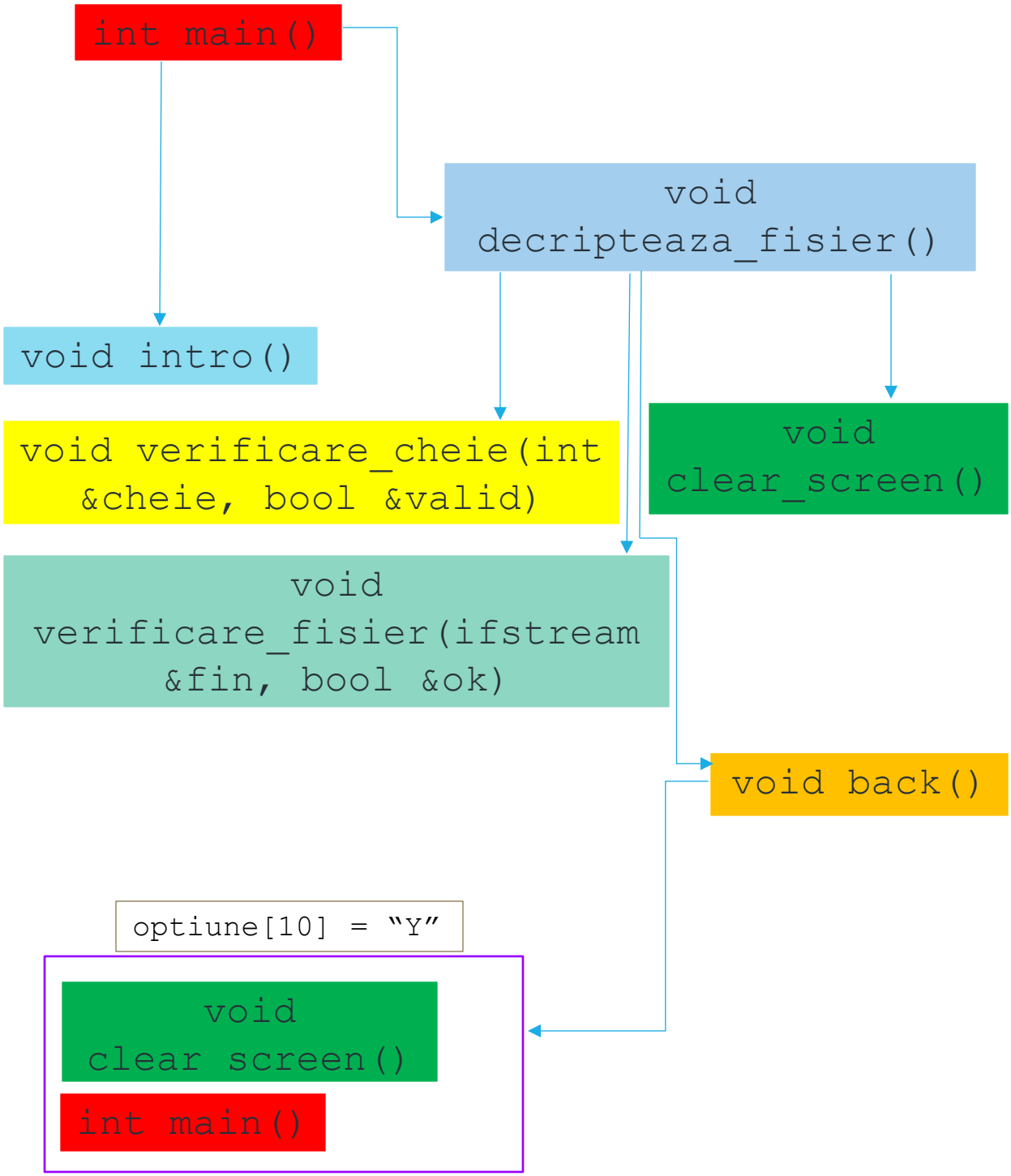
- Optiunea nr. 4

```
cin >> mod;        // 4
```



- Optiunea nr. 5

```
cin >> mod;        // 5
```

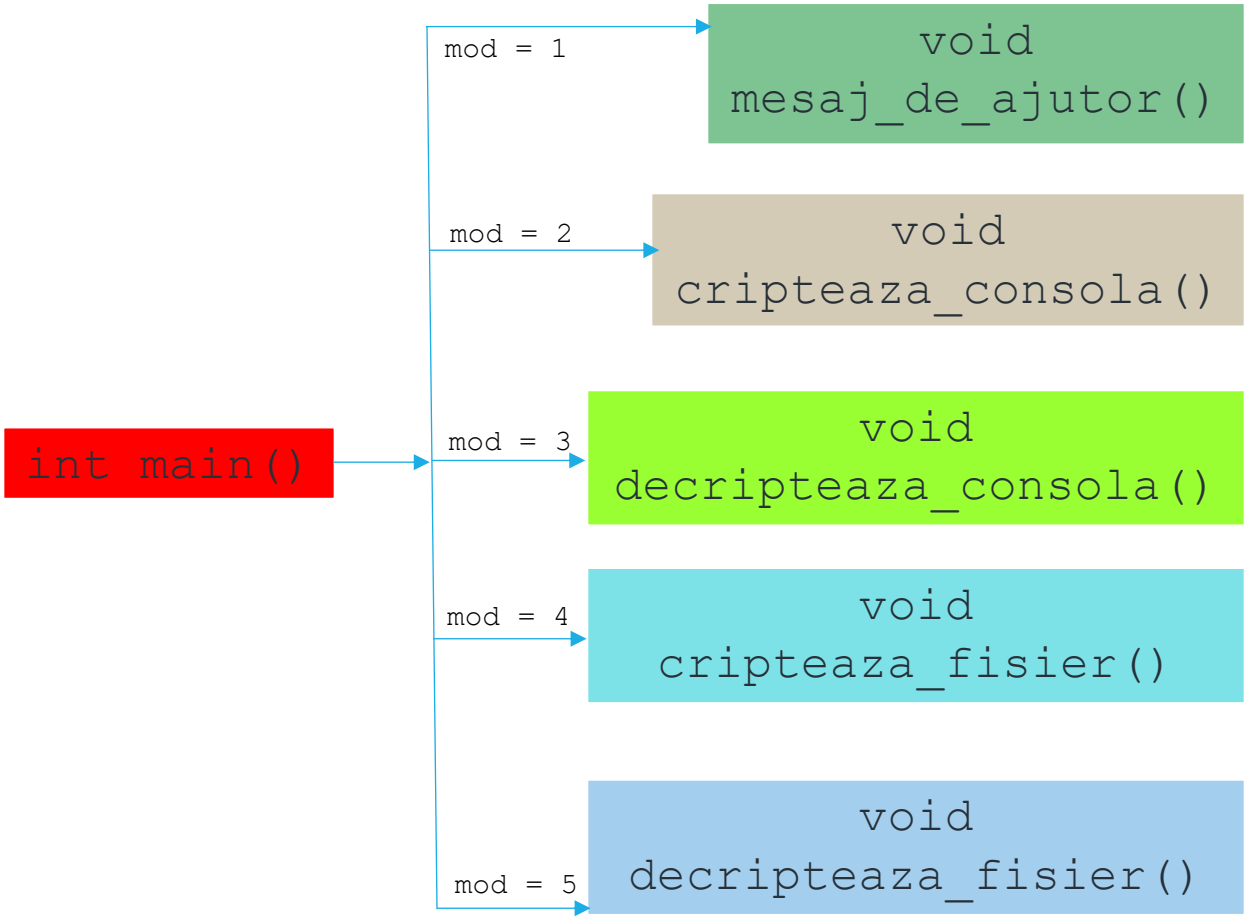


V. Structura functiei radacina

```
int main() {
    int mod, incercari = 3;  // Modul de operare al programului
                               => functia intro() contine valorile

    intro();     // contine meniul programului (o serie de afisari)

    while (incercari > 0) {
        cout << "[?] Introduceti optiunea dorita >> ";     cin >> mod;
        if (mod >= 1 && mod <= 5)   // verific daca modul de operare
                                       corespunde cerintelor
            incercari = 1;  // in caz afirmativ, numarul de incercari
                            // va deveni 1, caz in care se va apela
                            // functia corespunzatoare optiunii cerute

        switch(mod) {
            case 1:
                mesaj_de_ajutor();
                break;
            case 2:
                cripteaza_consola();
                break;
            case 3:
                decripteaza_consola();
                break;
            case 4:
                cripteaza_fisier();
                break;
            case 5:
                decripteaza_fisier();
                break;
            default:  // caz in care optiunea introdusa nu este corecta
                cout << "\n[-] Modul de operare introdus este invalid
!\n";
                break;
        }
        incercari--;  // pentru fiecare optiune introdusa gresit, nr de
    }                 // incercari se va micsora cu 1
    return 0;
}
```

Functia radacina gestioneaza celelalte functii ale programului. Utilizatorului ii este prezentat meniul prin intermediul functiei intro(), dupa care acesta este solicitat sa introduca optiunea dorita (optiunea este stocata in variabila de

tip intreg, **mod**). Daca optiunea introdusa de utilizator este in concordanta cu optiunile existente, functia **main** va apela functia corespunzatoare optiunii introduse. In cazul in care se introduce o valoare gresita, utilizatorul va avea la dispozitie inca 3 incercari pentru a introduce optiunea dorita. Acest lucru este realizat cu ajutorul unei structuri repetitive cu test initial (while) si a unei variabile care memoreaza numarul de incercari (**int incercari**).



VI. Prezentarea functiilor

1. void intro()

- Informeaza utilizatorul cu privire la optiunile disponibile, sub forma unui meniu.

```
void intro() {
```

```
    cout << "[+] Programul poate opera in unul dintre urmatoarele moduri
:" << endl;
    cout << "\t1\t- Afiseaza un mesaj de ajutor pentru o descriere
completa a programului" << endl;
    cout << "\t2\t- Cripteaza textul ce urmeaza a il introduce" << endl;
    cout << "\t3\t- Decripteaza textul ce urmeaza a il introduce" <<
endl;
    cout << "\t4\t- Cripteaza textul dintr-un fisier text specificat"
<< endl;
    cout << "\t5\t- Decripteaza textul dintr-un fisier text specificat"
<< endl;
    cout << "\t====================" << endl;
}
```

2. void back()

- Solicita utilizatorul cu privire la dorinta de a se intoarce la meniul principal. Utilizatorului ii sunt acordate 3 incercari pentru a alege optiunea corespunzatoare dorintei sale.

- Utilizatorul are la dispozitie 2 optiuni: Y (reprezentand cazul afirmativ) si N (reprezentand cazul negativ).

- Deoarece aceasta functie este utilizata de mai multe functii, iar ea trebuie sa apeleze alte 2 functii (clear_screen() si main()) apare nevoia sa ii declaram prototipul, dupa care, la sfarsitul programului, sa o definim.

- Variabila de tip intreg **incercari** retine numarul de incercari pe care utilizatorul le are la dispozitie.

- Variabila de tip sir de caractere **optiune[10]** memoreaza optiunea introdusa de utilizator. Optiunea este memorata intr-un sir de caractere pentru a preveni eventuale crash-uri ale programului in cazul in care utilizatorul introduce mai mult de un caracter.

```
void back() {
    char optiune[10];
    int incercari = 3;      // numarul de incercari

    while (incercari >= 0) {
        cout << "Doriti sa va intoarceti inapoi la meniu ? [Y/N] >>> ";
        cin >> optiune;

        if (strcmp(optiune, "Y") == 0 || strcmp(optiune, "N") == 0)
```

```
            incercari = -1;
        else
            incercari--;
    }

    if (strcmp(optiune, "Y") == 0) {
        clear_screen();     // apel catre functie
        main();       // apel catre functia radacina
    }
}
```

3. void clear_screen()

- "Curata" consola de tot textul care era inainte

- Realizeaza aceasta sarcina prin afisarea repetata a caracterului '\n' (newline).

```
void clear_screen() {
    /* "Curata" consola de tot textul care era inainte */

    for (int n = 0; n < 10; n++)
        cout << "\n\n\n\n\n\n\n\n\n\n";
}
```

4. void verificare_cheie(int & cheie, bool & valid)

- Verifica validitatea cheii introduse

- Cheia trebuie sa aiba o valoare din intervalul [1, 26]

- Valorile mai mari de 26 se repeta (de exemplu cheia 27 va avea acelasi efect precum cheia 1, intrucat 26 reprezinta sfarsitul alfabetului englez, caz in care acesta va fi parcurs din nou de la inceput; cheia 5743 este echivalenta cu cheia 23 -> 5743 mod 26 = 23).

- Daca valorarea ar fi negativa, atunci s-ar schimba sensul parcurgerii albafetului. De exemplu, criptarea se realizeaza mutand literele spre partea dreapta, iar daca valoarea ar fi negativa atunci literele ar fi mutate in partea stanga, ceea ce inseamna decriptare.

- Variabila de tip intreg **cheie** retine cheia introdusa de utilizator

- Variabila de tip logic **valid** retine valoarea *true* in cazul in care cheia este valida, si valoarea *false* in caz contrar.

- Deoarece variabilele sunt transmise prin referinta, valorile lor vor fi furnizate mai departe functiei apelante.

```
void verificare_cheie(int & cheie, bool & valid) {
    valid = false;
    for (int i = 3; !valid && i >= 0; i--) {
        cout << "[?] Introduceti cheia >> ";      cin >> cheie;
        if (cheie < 0 || cheie > 26) {     // Verificarea validitatii
cheiii
            cout << "\n[-] Cheie invalida ! Cheia poate avea valori in
intervalul [1, 26] !\n";
        }else
            valid = true;
    }
}
```

5. void verificare_fisier(ifstream &fin, bool &ok)

- Verifica daca fisierul introdus de la tastatura exista si poate fi deschis pentru citirea datelor.

- La fel ca si in cazul celorlalte functii, utilizatorul are 3 incercari la dispozitie sa introduca numele fisierului.

- Pentru deschiderea fisierului este utilizata functia open(), iar pentru verificarea existentei acestuia este utilizata functia good() -> returneaza *true* daca fisierul exista si este accesibil, *false* in caz contrar.

- Variabila **ok** de tip logic retine valoarea *true* daca numele fisierului este valid si valoarea *false* in caz contrar.

- Variabila **fin** de tip ifstream asigura comunicarea intre program si fisier.

- In cazul in care numele fisierului este valid, variabila ok va avea valoarea *true*, iar numarul de incercari va deveni -1 pentru terminarea instructiunii repetitive (while).

- Parametrii sunt transmisi prin referinta, ceea ce inseamna ca valoarea lor va fi utilizata de catre functia apelanta.

- In cazul in care fisierul nu exista, utilizatorului I se va afisa un mesaj specific, iar daca acesta mai are incercari disponibile, va fi solicitat sa introduca iar numele fisierului.

```
void verificare_fisier(ifstream &fin, bool &ok) {
    /* Verifica daca fisierul introdus de la tastatura exista si poate
fi deschis pentru citirea datelor */

    char nume_fisier[256];
    int incercari = 3;
    ok = false;

    while (!ok && incercari >= 0) {
        cin.get();
        cout << "[?] Introduceti numele fisierului >> ";
        cin.get(nume_fisier, 256);
        fin.open(nume_fisier);
        if (fin.good()) {
            incercari = -1;
            ok = true;
        }else {
            cout << "[-] Fisierul specificat nu exista !" << endl;
            incercari--;
        }
    }
}
```

6. void mesaj_de_ajutor()

- Afiseaza in consola un text informativ pentru a ajuta utilizatorul sa inteleaga cum functioneaza algoritmul *Cifrul lui Caesar*, oferindu-I si un scurt istoric.

- La sfarsit va fi apelata functia back() care il va intreba pe utilizator daca doreste sa se intoarca la meniul principal.

- Caracterul **'\n'** (newline) va afisa o linie noua plina cu spatii.

- Caracterul **'\t'** este echivalentul tastei TAB, afisand un numar prestabilit de caractere spatiu.

```
void mesaj_de_ajutor() {
    cout << "\n[+] Scurta introducere" << endl;
    cout << "\t====================" << endl;
```

```
    cout << "\n\t- Cifrul lui Caesar este numit dupa Julius Caesar, care,
conform lui Suetonius,";
    cout << "\n\tl-a folosit, mutand cu 3 litere textul pentru a proteja
mesaje cu importanta militara.";
    cout << endl;
    cout << "\n[+] Cum functioneaza?" << endl;
    cout << "\t====================" << endl;
    cout << "\n\t- Daca am presupune ca valoarea numerica a literelor
ar fi: A=1, B=2, ..., Z=26, atunci:";
    cout << "\n\t- Procesul de criptare poate fi descris printr-o functie
matematica, numita Cr(x).";
    cout << "\n\t- Criptarea unei litere x (A-Z) cu o cheie n (1-26),
poate fi descrisa astfel:";
    cout << "\n\t\t Cr(x) = (x + n) % 26";
    cout << "\n\t- Procesul de decriptare poate fi descris printr-o
functie matematica, numita Dr(x).";
    cout << "\n\t- Decriptarea unei litere x (A-Z) cu o cheie n (1-26),
poate fi descrisa astfel:";
    cout << "\n\t\t Dr(x) = (x - n) % 26" << endl;
    cout << "\n[+] Informatii suplimentare" << endl;
    cout << "\t====================" << endl;
    cout << "\n\t- In mod implicit, criptarea se realizeaza mutand
literele spre partea dreapta,";
    cout << "\n\tiar decriptarea spre partea stanga.";
    cout << "\n\t- De exemplu, criptarea cu cheia 3 spre stanga este
echivalenta cu criptarea cu";
    cout << "\n\tcheia 23 spre dreapta.";
    cout << "\n\t- Criptarea/Decriptarea din fisier este recomandata
pentru texte de dimensiuni mari.";
    cout                <<                "\n\t\t                (Sursa:
https://en.wikipedia.org/wiki/Caesar_cipher)" << endl;
    cout << "\n[+] Exemplu" << endl;
    cout << "\t====================" << endl;
    cout << "\n\t- Text normal (necriptat/plaintext) : VOM ATACA LA
NOAPTE";
    cout << "\n\t- Text criptat (ciphertext) cu cheia 7: CVT HAHJH SH
UVHWAL" << endl;

    back();     // intoarcere la meniul principal
}
```

7. void cripteaza_consola()

- Cripteaza textul introdus de la tastatura

- La inceputul executiei functiei, aceasta va apela subprogramul verificare_cheie de unde va prelua valorile pentru variabilele **cheie** si **valid**. Daca variabila **valid** are valoarea *true*, executia programului continua, altfel este terminata.

- Variabila de tip sir de caractere **text[256]** stocheaza textul introdus de utilizator de la tastatura (maxim 256 de caractere)

- La fiecare pas variabila de tip intreg **ascii** va retine codul ASCII corespunzator caracterului din text, caruia I se adauga valoarea cheii, obtinandu-se astfel mutarea caracterului in partea dreapta cu valoarea variabilei **cheie** pozitii. Se testeaza pentru caractere de tip litera mare sau mica, caz in care se aplica modificarile corespunzatoare**.**

- In cazul literelor mari se verifica daca valoarea variabilei **ascii** este mai mare de 90 (corespunzatoare literei Z), in caz afirmativ va trebui sa parcurgem alfabetul de la inceput, iar in caz contrar caracterul modficat va fi chiar valoarea variabilei **ascii**. Parcurgerea alfabetului de la inceput se realizeaza astfel: din variabila **ascii** se scade valoarea 90, iar diferenta va fi adunata cu 64 (caracterul din spatele literei A), iar aplicand conversia implicita spre tipul caracter obtinem caracterul substituit corespunzator spre partea dreapta.

- In cazul literelor mici se urmeaza acelasi procedeu, diferenta constand in valorile ce sunt adunate si scazute datorita faptului ca literele mici au codul ASCII mai mare cu 32 de unitati fata de litera mare corespunzatoare.

- La final, textul criptat este afisat pe ecran, iar functia **back()** este apelata.

```
        ====================
[?] Introduceti optiunea dorita >> 2
[?] Introduceti cheia >> 13
[?] Introduceti textul >> criptografia simetrica este rapida

        [!] Textul criptat >>> pevcgbtensvn fvzrgevpn rfgr encvqn
Doriti sa va intoarceti inapoi la meniu ? [Y/N] >>>
```

```
void cripteaza_consola() {

    bool valid;
    int cheie;
    verificare_cheie(cheie, valid);
```

```
    if (valid) {
        char text[256];
        cin.get();
        cout << "[?] Introduceti textul >> ";       cin.get(text, 255);

        for(unsigned int i=0; i < strlen(text); i++) {
            int ascii = int(text[i]) + cheie;
            if (isupper(text[i])) {         // verificare pentru litera
mare
                if (ascii > 90)
                    text[i] = (char)ascii - 90 + 64;    // 64 inseamna
inclusiv "A"
                else
                    text[i] = (char)ascii;
            }else if (islower(text[i])) {        // verificare pentru
litera mica
                if (ascii > 122)
                    text[i] = (char)ascii - 122 + 96;   // 96 inseamna
inclusiv "a"
                else
                    text[i] = (char)ascii;
            }
        }
        cout << "\n\t[!] Textul criptat >>> " << text << endl;
    }
    back();     // intoarcere la meniul principal
}
```

8. void decripteaza_consola()

- Decripteaza textul introdus de la tastatura.

- Validitatea cheii este testata la inceputul executiei functiei.

- La fiecare pas se executa aceleasi transformari ca si in cazul functiei **cripteaza_consola()**, diferenta consta in faptul ca inainte de intrare in structura repetitiva (for), variabila **cheie** va avea valoarea 26 – cheie, realizandu-se astfel substitutia literelor spre partea stanga.

- La finalul executiei functiei va fi afisat textul decriptat, iar functia **back()** va fi apelata.

- Variabilele utilizate au aceeasi semnificatie ca si in cazul functiei **cripteaza_consola()**.

```
void decripteaza_consola() {

    bool valid;
    int cheie;
    verificare_cheie(cheie, valid);

    if (valid) {
        char text[256];
        cin.get();
        cout << "[?] Introduceti textul >> ";       cin.get(text, 255);

        cheie = 26 - cheie;     // Pentru decriptare, se muta caracterele
in partea stanga
        for(unsigned int i = 0; i < strlen(text); i++) {
            int ascii = int(text[i]) + cheie;
            if (isupper(text[i])) {          // verificare pentru litera
mare
                if (ascii > 90)
                    text[i] = (char)ascii - 90 + 64;     // 64 inseamna
inclusiv "A"
                else
                    text[i] = (char)ascii;
            }else if (islower(text[i])) {        // verificare pentru
litera mica
                if (ascii > 122)
                    text[i] = (char)ascii - 122 + 96;    // 96 inseamna
inclusiv "a"
                else
                    text[i] = (char)ascii;
            }
        }
        cout << "\n\t[!] Textul decriptat >>> " << text << endl;
    }
    back();      // revenire la meniul principal
}
```

9. cripteaza_fisier()

- Cripteaza textul dintr-un fisier, al carui nume este introdus de la tastatura, la fel si cheia de criptare.

- Diferenta dintre aceasta functie si functia cripteaza_consola() consta modul din citire a datelor: din fisier, respectiv de la tastatura.

- Variabila de tip ifstream **fisier** ne permite sa comunicam cu fisierul.

- Subprogramul apeleaza functia *verifica_fisier()*, cu parametrii specifici pentru a asigura existenta si disponibilitatea acestuia.

- Textul criptat va fi stocat in fisierul **criptat.txt**, de acest fisier ocupandu-se variabila de tip ofstream **criptat**.

- Inainte de afisarea mesajului informativ e ste apelata functia **clear_screen()** pentru o estetica mai buna a consolei.


```
void cripteaza_fisier() {

    bool valid, ok;
    int cheie;
    verificare_cheie(cheie, valid);

    if (valid) {      // prima conditie
        ifstream fisier;

        // Verificarea existentei si accesibilitatii fisierului
        verificare_fisier(fisier, ok);

        if (ok) {     // a doua conditie
            ofstream criptat("criptat.txt");
            char x;
            while (fisier >> noskipws >> x) {    // Citeste din fisier
si spatiile
                int ascii = int(x) + cheie;
                if (isupper(x)) {           // verificare pentru litera
mare
                    if (ascii > 90)
                        x = (char)ascii - 90 + 64;     // 64 inseamna
inclusiv "A"
                    else
                        x = (char)ascii;
```

```
                }else if (islower(x)) {      // verificare pentru litera
mica
                    if (ascii > 122)
                        x = (char)ascii - 122 + 96;    // 96 inseamna
inclusiv "a"
                    else
                        x = (char)ascii;
                }
                criptat << x;
            }
            clear_screen();
            cout                                                        <<
"\t\t##################################################" << endl;
            cout << "\t\t#\t[!] Criptare reusita !\t\t\t #" << endl;
            cout << "\t\t#\t[!] Mesajul criptat in > [criptat.txt]\t #"
<< endl;
            cout                                                        <<
"\t\t##################################################" << endl;
            criptat.close();
        }
    }
    back();      // revenire la meniul principal
}
```

10. decripteaza_fisier()

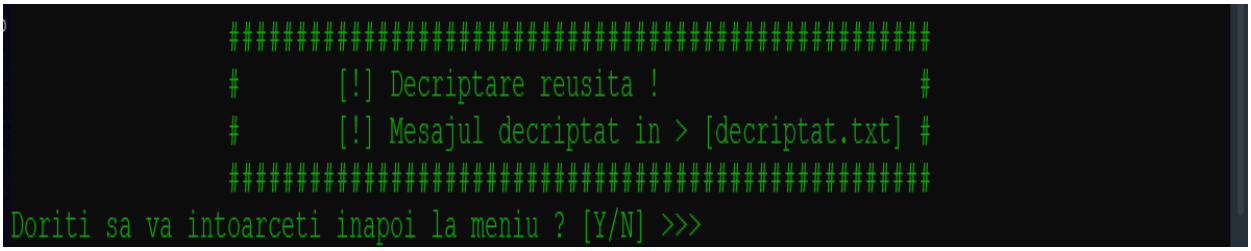
- Utilizeaza acelasi principiu precum functia decripteaza_consola(), diferenta constand in faptul ca citirea datelor se face dintr-un fisier.

- Afisarea mesajului informativ si verificarea integritatii fisierului este aceeasi ca si in cazul functiei **cripteaza_fisier()**.

- Ambele functii (**decripteaza_fisier() si cripteaza_fisier()**) isi continua executia doar daca sunt indeplinite simultan urmatoarele 2 conditii: cheia este valida (verificarea este asigurata prin intermediul variabilei **valid**) si fisierul este valid (verificarea este asigurata prin intermediul variabilei **ok**).

- Textul decriptat este stocat in fisierul **decriptat.txt**

- *Nota: Este recomandata utilizarea functiilor **cripteaza_fisier()** si **decripteaza_fisier()** (prin alegerea optiunii 4, respectiv 5 din meniu) pentru criptarea, respectiv decriptarea textelor de dimensiuni mari (ce depasesc 256 de caractere).*




```
void decripteaza_fisier() {
    bool valid, ok;
    int cheie;
    verificare_cheie(cheie, valid);

    if (valid) {      // prima conditie
        ifstream fisier;

        // Verificarea existentei si accesibilitatii fisierului
        verificare_fisier(fisier, ok);

        if (ok) {     // a doua conditie
            ofstream decriptat("decriptat.txt");
            char x;
            cheie = 26 - cheie;       // Pentru decriptare, se muta
caracterele in partea stanga
            while (fisier >> noskipws >> x) {
                int ascii = int(x) + cheie;
                if (isupper(x)) {          // verificare pentru litera
mare
                    if (ascii > 90)
                        x = (char)ascii - 90 + 64;     // 64 inseamna
inclusiv "A"
                    else
                        x = (char)ascii;
                }else if (islower(x)) {     // verificare pentru litera
mica
                    if (ascii > 122)
                        x = (char)ascii - 122 + 96;    // 96 inseamna
inclusiv "a"
                    else
                        x = (char)ascii;
                }
                decriptat << x;
            }
            clear_screen();
            cout                                                    <<
"\t\t#################################################" << endl;
```

```
            cout << "\t\t#\t[!] Decriptare reusita !\t\t   #" << endl;
            cout << "\t\t#\t[!] Mesajul decriptat in > [decriptat.txt]
#" << endl;
            cout                                                    <<
"\t\t#######################################################" << endl;
            decriptat.close();
        }
    }
    back();
}
```

VII. Prezentare headere si functiile utilizate

```
#include <fstream>
```

- open()
- close()
- good()

```
#include <cstring>
```

- cin.get()
- strcmp()
- strlen()

```
#include <cctype>
```

- isupper()
- islower()

VIII. Codul sursa

```
#include <iostream>
#include <fstream>
#include <cstring>
```

```
#include <cctype>

using namespace std;

void back();    // prototipul functiei

void intro() {
    /*  Afiseaza pe ecran o scurta descriere a modurilor de functionare
a programului:
    -> 0    - Mesaj de ajutor
    -> 1    - Cripteaza textul introdus in consola
    -> 2    - Decripteaza textul introdus in consola
    -> 3    - Cripteaza textul dintr-un fisier introdus ca argument
    -> 4    - Decripteaza textul dintr-un fisier introdus ca argument
    */

    cout << "[+] Programul poate opera in unul dintre urmatoarele moduri
:" << endl;
    cout << "\t1\t- Afiseaza un mesaj de ajutor pentru o descriere
completa a programului" << endl;
    cout << "\t2\t- Cripteaza textul ce urmeaza a il introduce" << endl;
    cout << "\t3\t- Decripteaza textul ce urmeaza a il introduce" <<
endl;
    cout << "\t4\t- Cripteaza textul dintr-un fisier text specificat"
<< endl;
    cout << "\t5\t- Decripteaza textul dintr-un fisier text specificat"
<< endl;
    cout << "\t====================" << endl;
}

void mesaj_de_ajutor() {
    cout << "\n[+] Scurta introducere" << endl;
    cout << "\t====================" << endl;
    cout << "\n\t- Cifrul lui Caesar este numit dupa Julius Caesar, care,
conform lui Suetonius,";
    cout << "\n\tl-a folosit, mutand cu 3 litere textul pentru a proteja
mesaje cu importanta militara.";
    cout << endl;
    cout << "\n[+] Cum functioneaza?" << endl;
    cout << "\t====================" << endl;
    cout << "\n\t- Daca am presupune ca valoarea numerica a literelor
ar fi: A=1, B=2, ..., Z=26, atunci:";
    cout << "\n\t- Procesul de criptare poate fi descris printr-o functie
matematica, numita Cr(x).";
    cout << "\n\t- Criptarea unei litere x (A-Z) cu o cheie n (1-26),
poate fi descrisa astfel:";
    cout << "\n\t\t Cr(x) = (x + n) % 26";
```

```
    cout << "\n\t- Procesul de decriptare poate fi descris printr-o
functie matematica, numita Dr(x).";
    cout << "\n\t- Decriptarea unei litere x (A-Z) cu o cheie n (1-26),
poate fi descrisa astfel:";
    cout << "\n\t\t Dr(x) = (x - n) % 26" << endl;
    cout << "\n[+] Informatii suplimentare" << endl;
    cout << "\t====================" << endl;
    cout << "\n\t- In mod implicit, criptarea se realizeaza mutand
literele spre partea dreapta,";
    cout << "\n\tiar decriptarea spre partea stanga.";
    cout << "\n\t- De exemplu, criptarea cu cheia 3 spre stanga este
echivalenta cu criptarea cu";
    cout << "\n\tcheia 23 spre dreapta.";
    cout << "\n\t- Criptarea/Decriptarea din fisier este recomandata
pentru texte de dimensiuni mari.";
    cout               <<               "\n\t\t               (Sursa:
https://en.wikipedia.org/wiki/Caesar_cipher)" << endl;
    cout << "\n[+] Exemplu" << endl;
    cout << "\t====================" << endl;
    cout << "\n\t- Text normal (necriptat/plaintext) : VOM ATACA LA
NOAPTE";
    cout << "\n\t- Text criptat (ciphertext) cu cheia 7: CVT HAHJH SH
UVHWAL" << endl;

    back();
}

void verificare_cheie(int & cheie, bool & valid) {
    /* Verifica daca cheia introdusa de la tastatura are valoarea
cuprinsa in intervalul [1, 26] */

    valid = false;
    for (int i = 3; !valid && i >= 0; i--) {
        cout << "[?] Introduceti cheia >> ";      cin >> cheie;
        if (cheie < 0 || cheie > 26) {     // Verificarea validitatii
cheiii
            cout << "\n[-] Cheie invalida ! Cheia poate avea valori in
intervalul [1, 26] !\n";
        }else
            valid = true;
    }
}

void verificare_fisier(ifstream &fin, bool &ok) {
    /* Verifica daca fisierul introdus de la tastatura exista si poate
fi deschis
    pentru citirea datelor */
```

```
    char nume_fisier[256];
    int incercari = 3;
    ok = false;

    while (!ok && incercari >= 0) {
        cin.get();
        cout << "[?] Introduceti numele fisierului >> ";
        cin.get(nume_fisier, 256);
        fin.open(nume_fisier);
        if (fin.good()) {
            incercari = -1;
            ok = true;
        }else {
            cout << "[-] Fisierul specificat nu exista !" << endl;
            incercari--;
        }
    }
}

void clear_screen() {
    /* "Curata" consola de tot textul care era inainte */

    for (int n = 0; n < 10; n++)
        cout << "\n\n\n\n\n\n\n\n\n\n";
}

void cripteaza_consola() {
    /* Cripteaza textul introdus de la tastatura.
    Criptarea se realizeaza prin adunarea codului ASCII al fiecarui
caracter
    introdus de la tastatura cu cheia intr-o variabila numita ascii. La
fiecare pas
    se scade 90 din acea valoare si se aduna 64 (in cazul literelor
mici). Acelasi
    procedeu este implementat si in cazul literelor mari, doar ca sunt
folosite
    valorile corespunzatoare.
    */

    bool valid;
    int cheie;
    verificare_cheie(cheie, valid);

    if (valid) {
        char text[256];
        cin.get();
        cout << "[?] Introduceti textul >> ";       cin.get(text, 255);
```

```
        for(unsigned int i = 0; i < strlen(text); i++) {
            int ascii = int(text[i]) + cheie;
            if (isupper(text[i])) {         // verificare pentru litera
mare
                if (ascii > 90)
                    text[i] = (char)ascii - 90 + 64;    // 64 inseamna
inclusiv "A"
                else
                    text[i] = (char)ascii;
            }else if (islower(text[i])) {        // verificare pentru
litera mica
                if (ascii > 122)
                    text[i] = (char)ascii - 122 + 96;   // 96 inseamna
inclusiv "a"
                else
                    text[i] = (char)ascii;
            }
        }
        cout << "\n\t[!] Textul criptat >>> " << text << endl;
    }
    back();
}

void decripteaza_consola() {
    /* Decripteaza textul introdus de la tastatura.
    Decriptarea se realizeaza adunand succesiv codul ASCII al
caracterelor textului
    cu cheia(cate un caracter la fiecare pas) in variabila ascii. Daca
caracterul
    de la acel pas este litera mare se verifica daca valoarea variabilei
ascii depaseste
    valoarea 90 (insemnand litera Z), in caz afirmativ se va scadea 90
si aduna 64 din
    variabila ascii, rezultand astfel litera decriptata conform cheii.
Acelasi
    lucru se intampla si in cazul literelor mici, caz in care se scade
122 si se aduna 96
    din variabila ascii.
    */

    bool valid;
    int cheie;
    verificare_cheie(cheie, valid);

    if (valid) {
        char text[256];
        cin.get();
        cout << "[?] Introduceti textul >> ";       cin.get(text, 255);
```

```
        cheie = 26 - cheie;    // Pentru decriptare, se muta caracterele
in partea stanga
        for(unsigned int i = 0; i < strlen(text); i++) {
            int ascii = int(text[i]) + cheie;
            if (isupper(text[i])) {         // verificare pentru litera
mare
                if (ascii > 90)
                    text[i] = (char)ascii - 90 + 64;    // 64 inseamna
inclusiv "A"
                else
                    text[i] = (char)ascii;
            }else if (islower(text[i])) {        // verificare pentru
litera mica
                if (ascii > 122)
                    text[i] = (char)ascii - 122 + 96;   // 96 inseamna
inclusiv "a"
                else
                    text[i] = (char)ascii;
            }
        }
        cout << "\n\t[!] Textul decriptat >>> " << text << endl;
    }
    back();
}

void cripteaza_fisier() {
    /* Utilizeaza acelasi principiu precum functia cripteaza_consola(),
diferenta constand in faptul
    ca citirea datelor se face dintr-un fisier */

    bool valid, ok;
    int cheie;
    verificare_cheie(cheie, valid);

    if (valid) {
        ifstream fisier;

        // Verificarea existentei si accesibilitatii fisierului
        verificare_fisier(fisier, ok);

        if (ok) {
            ofstream criptat("criptat.txt");
            char x;
            while (fisier >> noskipws >> x) {   // Citeste din fisier
si spatiile
                int ascii = int(x) + cheie;
```

```
                if (isupper(x)) {          // verificare pentru litera
mare
                    if (ascii > 90)
                        x = (char)ascii - 90 + 64;     // 64 inseamna
inclusiv "A"
                    else
                        x = (char)ascii;
                }else if (islower(x)) {     // verificare pentru litera
mica
                    if (ascii > 122)
                        x = (char)ascii - 122 + 96;    // 96 inseamna
inclusiv "a"
                    else
                        x = (char)ascii;
                }
                criptat << x;
            }
            clear_screen();
            cout                                                    <<
"\t\t#####################################################" << endl;
            cout << "\t\t#\t[!] Criptare reusita !\t\t\t #" << endl;
            cout << "\t\t#\t[!] Mesajul criptat in > [criptat.txt]\t #"
<< endl;
            cout                                                    <<
"\t\t#####################################################" << endl;
            criptat.close();
        }
    }
    back();
}

void decripteaza_fisier() {
    /*     Utilizeaza     acelasi     principiu     precum     functia
decripteaza_consola(), diferenta constand in faptul
    ca citirea datelor se face dintr-un fisier */

    bool valid, ok;
    int cheie;
    verificare_cheie(cheie, valid);

    if (valid) {
        ifstream fisier;

        // Verificarea existentei si accesibilitatii fisierului
        verificare_fisier(fisier, ok);

        if (ok) {
            ofstream decriptat("decriptat.txt");
```

```
            char x;
            cheie = 26 - cheie;       // Pentru decriptare, se muta
caracterele in partea stanga
            while (fisier >> noskipws >> x) {
                int ascii = int(x) + cheie;
                if (isupper(x)) {          // verificare pentru litera
mare
                    if (ascii > 90)
                        x = (char)ascii - 90 + 64;     // 64 inseamna
inclusiv "A"
                    else
                        x = (char)ascii;
                }else if (islower(x)) {     // verificare pentru litera
mica
                    if (ascii > 122)
                        x = (char)ascii - 122 + 96;    // 96 inseamna
inclusiv "a"
                    else
                        x = (char)ascii;
                }
                decriptat << x;
            }
            clear_screen();
            cout                                                    <<
"\t\t#######################################################" << endl;
            cout << "\t\t#\t[!] Decriptare reusita !\t\t   #" << endl;
            cout << "\t\t#\t[!] Mesajul decriptat in > [decriptat.txt]
#" << endl;
            cout                                                    <<
"\t\t#######################################################" << endl;
            decriptat.close();
        }
    }
    back();
}

int main() {
    int mod, incercari = 3;   // Modul de operare al programului =>
functia intro() contine valorile

    intro();     // contine meniul programului (o serie de afisari)

    while (incercari > 0) {
        cout << "[?] Introduceti optiunea dorita >> ";     cin >> mod;
        if (mod >= 1 && mod <= 5)   // verific daca modul de corespunde
cerintelor
            incercari = 1;  // in caz afirmativ, numarul de incercari
va deveni 1,
```

```
                            //  caz  in  care  se  va  apela  functia
corespunzatoare optiunii cerute

        switch(mod) {
            case 1:
                mesaj_de_ajutor();
                break;
            case 2:
                cripteaza_consola();
                break;
            case 3:
                decripteaza_consola();
                break;
            case 4:
                cripteaza_fisier();
                break;
            case 5:
                decripteaza_fisier();
                break;
            default:  // caz in care optiunea introdusa nu este corecta
                cout << "\n[-] Modul de operare introdus este invalid
!\n";
                break;
        }
        incercari--;  // pentru fiecare optiune introdusa gresit, nr de
    }                     // incercari se va micsora cu 1
    return 0;
}


void back() {
    /* Solicita utilizatorul cu privire la dorinta de a se intoarce la
meniul principal.
    Utilizatorului ii sunt acordate 3 incercari pentru a alege optiunea
corespunzatoare
    dorintei sale. */

    char optiune[10];
    int incercari = 3;  // numarul de incercari

    while (incercari >= 0) {
        cout << "Doriti sa va intoarceti inapoi la meniu ? [Y/N] >>> ";
        cin >> optiune;

        if (strcmp(optiune, "Y") == 0 || strcmp(optiune, "N") == 0)
            incercari = -1;
        else
            incercari--;
```

```
    }

    if (strcmp(optiune, "Y") == 0) {
        clear_screen();
        main();
    }
}
```

IX. Bibliografie

1. https://stackoverflow.com/questions/21647/reading-from-text-file-until-eof-repeats-last-line
2. https://stackoverflow.com/questions/5605125/why-is-iostreameof-inside-a-loop-condition-considered-wrong
3. https://en.wikipedia.org/wiki/Caesar_cipher
4. https://www.xarg.org/tools/caesar-cipher/ -> pentru testarea programului
5. http://www.cplusplus.com/ -> pentru documentatie asupra functiilor din headere