Ministerul Educaţiei al Republicii Moldova

Universitatea Tehnică a Moldovei

Facultatea: calculatoare, informatică şi microelectronică

**RAPORT**

Lucrare de laborator Nr.1

*la Programarea Calculatoarelor*

Теmа: Utilizarea instrucțiunilor liniare și condiționale în limbajul C

A efectuat: st. gr. SI-212 Șeremet Alexandru

A verificat: lect. asist. V. Mititelu

Chişinău 2021

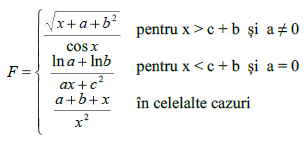
**Lucrare de laborator Nr.1**

**Tema:** Utilizarea instrucțiunilor liniare și condiționale în limbajul C

**Scopul lucrării:** Studierea tehnicilor și metodelor de utilizare a instrucțiunilor condiționale, de citire a informației de la tastatură și de afișare a informației pe ecran pentru calculul diverselor funcții matematice în limbajul C.

**Sarcina:** Să se calculeze și să se afișeze la ecran valorile argumentului x și valorile funcției F, definită prin 3 expresii date. Variabilele reale a, b, c și x se citesc de la tastatură.

**Varianta 15:**



**Rezumat:**

**Noţiuni principale din teorie şi metode folosite:**

Un algoritm este compus dintr-o mulțime finită de pași, fiecare necesitînd una sau mai multe operații. Formele convenționale cele mai folosite în reprezentarea algoritmilor sunt:

* Schemele logice sau organigramele,;
* Limbajele pseudocod;

Principala calitate a acestora este posibilitatea de a evidenția cu claritate fluxul controlului algoritmilor, adică succesiunile posibile ale acțiunilor. Astfel, **schemele logice** utilizează pentru aceasta săgeți de legătură între diversele forme geometrice care simbolizează diverse tipuri de acțiuni, în timp ce **limbajele pseudocod** folosesc cuvinte-cheie, adică niște cuvinte cu înțeles prestabilit ce identifică operația care se execută, și cîteva reguli simple de aliniere a textului scris.

Structura unui program simplu in limbajul C este urmatoarea:

* directive preprocesare;
* declarații globale;
* antetul functiei main;
* corpul functiei main.

**Instrucțiuni condiționale**:

**Instrucțiunile condiționale** determină programele să testeze diferite condiții și, în functie de acestea, să decidă execuția anumitor comenzi.  
Aceste **instrucțiuni condiționale** sunt:

* **if()** - execută comenzile dorite când o condiție scrisă în paranteze este adevarată.
* **if() ... else** - execută anumite comenzi când o condiție este adevarată și alte comenzi când aceasta este falsă.
* **switch** - selecteaza care comandă va fi executată.

**Citirea și afișarea datelor:**

Pentru citire se folosește : "scanf" sau "cin". Pentru afișare se folosește : "printf" sau "cout".

- scanf sau cin, citește valoarea introdusă de la tastatură.

- printf sau cout, afișează un mesaj.

**Descrierea variabilelor:**

a) date de intrare:

x,a,b,c - variabile simple de tip float, introduse de la tastatură;

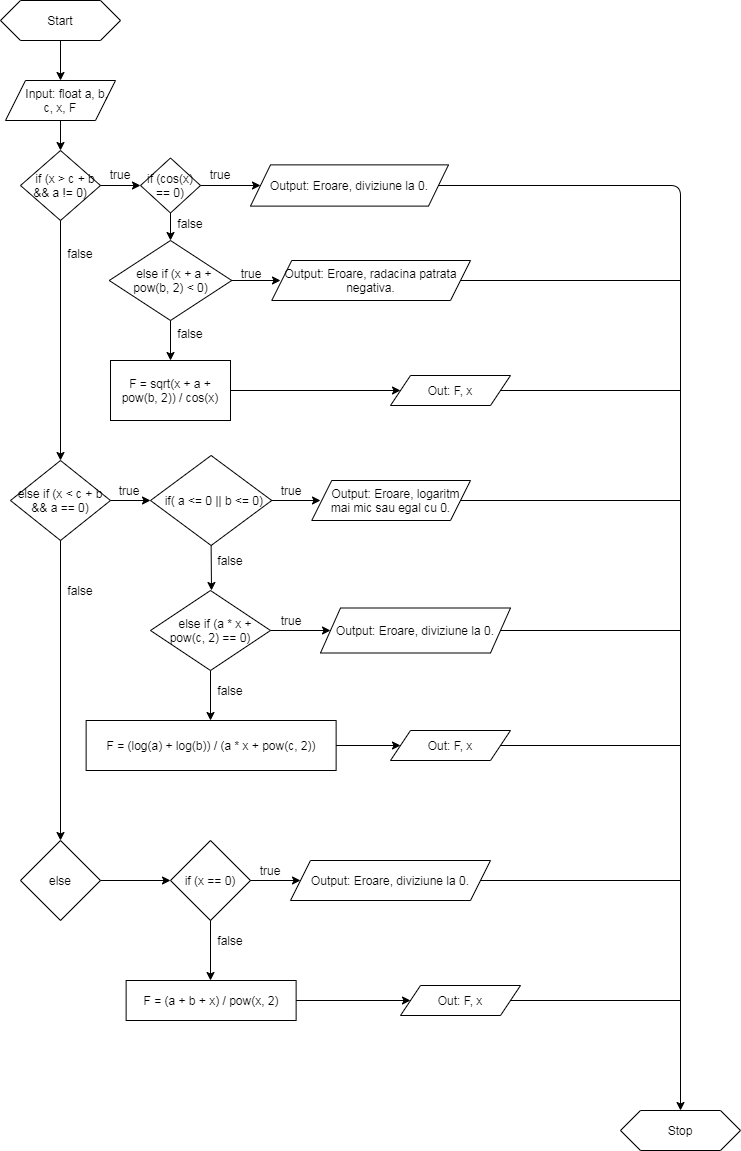
b) date de ieşire:

x,F - variabile simple de tip float, rezultatele de calcul;

c) date de lucru:

x,F - variabile simple de tip float, rezultatele de calcul.

**Schema bloc a algoritmului de soluționare a problemei:**



**Codul deplin al programului:**

#include <stdio.h>

#include <math.h>

int main ()

{

    float a, b, c, x, F;

    printf ("\nIntroduceti a, b, c si x: ");

    scanf("%f %f %f %f", &a, &b, &c, &x);

    if (x > c + b && a != 0) {

        printf("\nCazul 1: ");

        if (cos(x) == 0) {

            printf("\nEroare, diviziune la 0.\n");

            return (1);

        }

        else if (x + a + pow(b, 2) < 0) {

            printf("\nEroare, radacina patrata negativa.\n");

            return (1);

        } //exceptii^^

        F = sqrt(x + a + pow(b, 2)) / cos(x);

    }

    else if (x < c + b && a == 0) {

        printf("\nCazul 2: ");

        if (a <= 0 || b <= 0) {

            printf("\nEroare, logaritm mai mic sau egal cu 0.\n");

            return (1);

        }

        else if (a \* x + pow(c, 2) == 0) {

            printf("\nEroare, diviziune la 0.\n");

            return (1);

        } //exceptii^^

        F = (log(a) + log(b)) / (a \* x + pow(c, 2));

    }

    else {

        printf("\nCazul 3: ");

        if (x == 0) {

            printf("\nEroare, diviziune la 0.\n");

            return (1);

        } //exceptii^^

        F = (a + b + x) / pow(x, 2);

    }

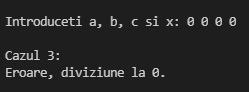
    printf ("\nF = %f \nx = %f\n", F, x);

    return (0);

}

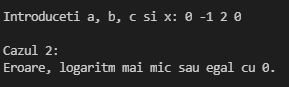
**Execuția programului:**

* Set de date nr. 1: 0 0 0 0



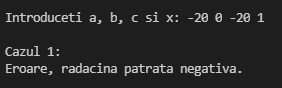
Întrucât valoarile nu corespund primelor două condiții if, x, c și b fiind egale între ele, se trece la blocul de cod desemnat de condiția else, iar, deoarece x este egal cu 0, și el e singurul numitor din fracția din formula F = (a + b + x) / pow(x, 2), codul nu poate fi executat, printând o linie de text care anunță o eroare și returnând cifra 1.

* Set de date nr. 2: 0 -1 2 0



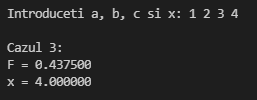
Întrucât valorile corespund condițiilor lui else if dupa primul if, x fiind mai mic decât suma lui c și b, iar a fiind egal cu 0, se trece la blocul de cod respectiv, iar, deoarece a și b sunt mai mici sau egali cu 0, ln(a) și ln(b) din formula F = (log(a) + log(b)) / (a \* x + pow(c, 2)) nu pot fi calculați, și, astfel, programul printează o linie de text care anunță o eroare și returneaza cifra 1.

* Set de date nr. 3: -20 0 -20 1



Întrucât valorile corespund condițiilor primului if, x fiind mai mare decât suma lui c și b, iar a fiind diferit de 0, se trece la blocul de cod respectiv, iar, deoarece suma lui x cu a și b^2 este negativă, radacina ei pătrată din formula F = sqrt(x + a + pow(b, 2)) / cos(x) nu poate fi calculată, și, astfel, programul printează o linie de text care anunță o eroare și returnează cifra 1.

* Set de date nr. 4: 1 2 3 4



Întrucât valoarile nu corespund primelor două condiții if, x, c și b fiind egale între ele, se trece la blocul de cod desemnat de condiția else, care calculează valoarea lui F după formula

F = (a + b + x) / pow(x, 2), după aia printându-se rezultatul împreună cu valoarea lui x și returnându-se cifra 0, care anunță executarea codului fără probleme.

**Concluzii:**

1. Pentru a scrie un program eficient, fiecare detaliu trebuie luat în considerare și implementat adecvat, pseudocodul și schema bloc ale viitorului cod fiind inalienabile în planificarea lui, ele oferind o viziune clară asupra scopurilor și modalităților de a le atinge.
2. Instrucțiunile condiționale sunt primordiale în anticiparea și implementarea valorilor divergente ale variabilelor, ele oferind un mod flexibil de a optimiza codul.
3. Citirea și printarea variabilelor stă la baza introducerii unui element interactiv în program, rezultatul variind astfel în dependență de ceea ce introduce utilizatorul, codul devenint mult mai flexibil și adaptabil drept consecință.

**Surse:**

https://marplo.net/flash/instructiuni-conditionale-if-else-switch\_as

<https://world-it.ro/software/limbaje-de-programare/c/lectia-7-c-citire-si-afisare>