Ministerul Educaţiei și Cercetării al Republicii Moldova

Universitatea Tehnică a Moldovei

Facultatea Calculatoare, Informatică şi Microelectronică

**RAPORT**

Lucrare de laborator Nr.2

*la Programarea Calculatoarelor*

Tema: Instrucțiuni repetitive (cicluri)

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

A verificat: lect. asist. Mititelu Vitalii

Chişinău 2021

**Lucrare de laborator Nr.2**

**Tema:** Instrucțiuni repetitive (cicluri)

**Scopul:** Utilizarea instrucțiunilor repetitive (for, while și do-while) pentru scrierea programelor în limbajul de programare C.

**Sarcina:** Scrieți un program care citește de la tastatură valorile variabilelor a, b, c, n, i, j, k (în dependență de variantă), efectuează ceea ce este indicat în variantă, apoi afișează pe ecran rezultatul:

**Varianta 15:**

Determină numerele impare succesive a căror sumă este egală cu n^3, pentru n= 1,...,20 (ca exemplu 1^3=1; 2^3=3+5; 3^3=7+9+11 etc).

**Rezumat:**

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

**Instrucţiunile repetitive** asigură repetarea unei instrucţiuni (simple sau compuse) de un anumit număr de ori sau cât timp este îndeplinită o condiţie. În C sunt în număr de trei: while, do-while, for.

* Instrucţiunea **while** are sintaxa generală:

while(<expresie\_condiţională>) <instrucţiune>;

Intrucţiunea determină repetarea instrucţiunii cât timp evaluarea expresiei se face cu valoarea true. Dacă, la un moment dat în execuţie, evaluarea se face cu false, instrucţiunea este închisă şi se trece la linia următoare din program.

- Instrucţiunea **do-while** are sintaxa generală:

do {

<instrucţiune;>

} while (<expresie\_condiţională>);

Instrucţiunea do-while are un comportament asemănător instrucţiunii while, dar se deosebeşte prin aceea că asigură execuţia instrucţiunii cel puţin o dată, indiferent de îndeplinirea condiţiei.

- Instrucţiunea **for** este numită şi instrucţiune repetitivă cu număr cunoscut de paşi. Ea este concepută pentru a se executa de un număr determinat de ori.

Sintaxa sa generală este:

for(<instrucţiune\_iniţializare>; <expresie\_condiţională>; <expresie\_creştere) {

<lista\_instrucţiuni>;

}

Precum în cazul while şi instrucţiunea din interiorul for se execută cât timp expresia condiţională este evaluată cu true.

Procesul de execuţie al instrucţiunii for poate fi rezumat astfel:

pas 1: se execută instrucţiunea de iniţializare. De obicei, în acest moment se declară o variabilă contor şi este iniţializată cu o valoare. Această instrucţiune este executată o singură dată, la început.

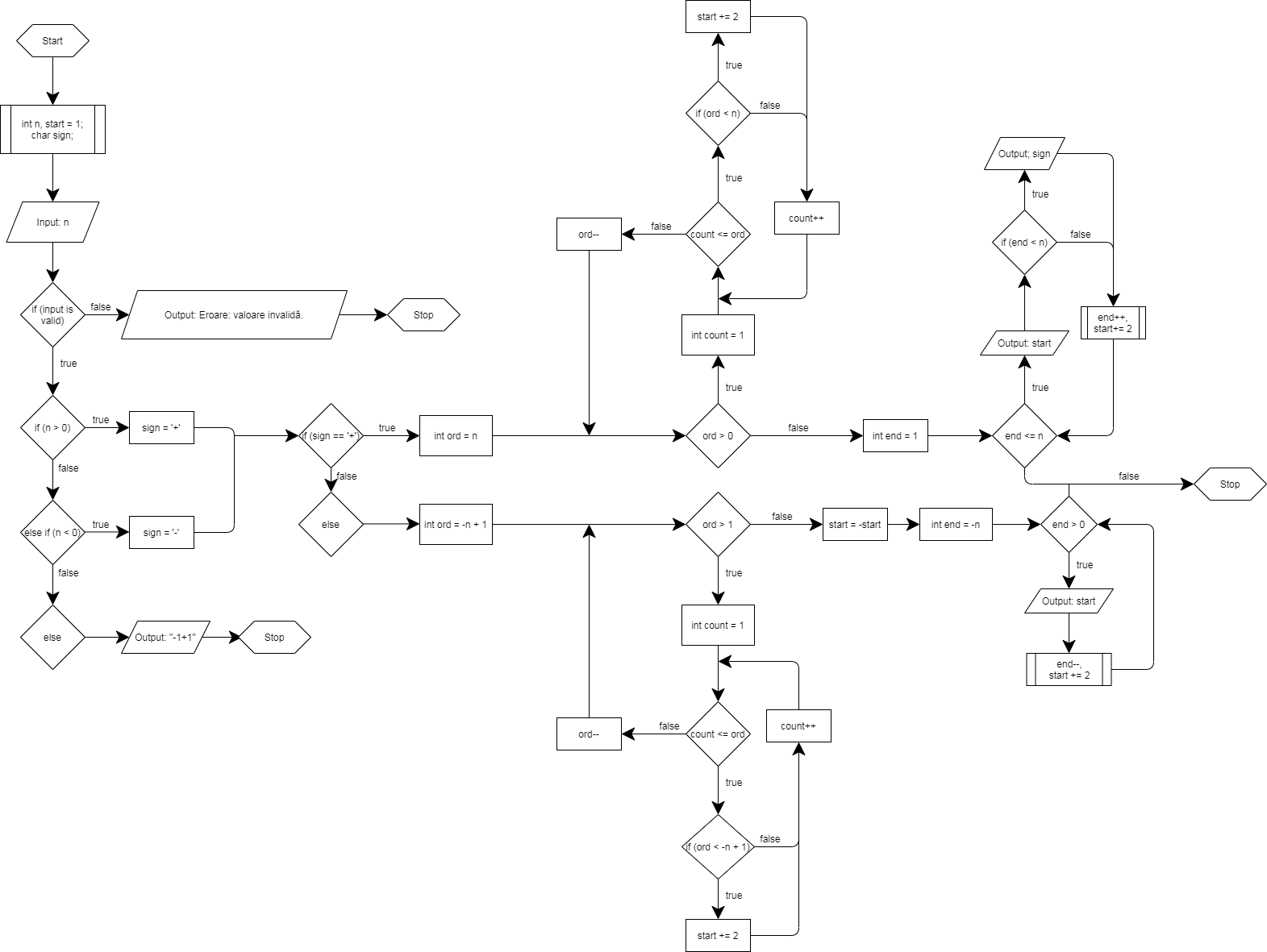
pas 2: este evaluată expresia. Dacă rezultatul evaluării este true, instrucţiunea continuă. Altfel este terminată şi programul continuă cu pasul 5.

pas 3: este executată instrucţiunea. Aceasta poate fi o instrucţiune simplă sau compusă.

pas 4: se execută expresia de creştere. Se revine la pas 2

pas 5: Instrucţiunea for este terminată şi se trece la isntrucţiunea următoare.

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



**Codul deplin al programului:**

#include <stdio.h>

#include <math.h>

int main ()

{

    int n, start = 1;

    char sign;

    printf("\nIntroduceti: ");

    if (!scanf("%d", &n)) {

        printf("\nEroare: valoare invalida.\n");

        return (1);

    }

    if(n > 0)

        sign = '+';

    else if (n < 0)

        sign = '-';

    else {

        printf("\nRezultat: -1+1\n");

        return (0);

    }

    printf("\nRezultat: ");

    if (sign == '+') {

        for (int ord = n; ord > 0; ord--){

            for(int count = 1; count <= ord; count++){

                if(ord < n)

                    start += 2;

            }

        } //afla unde se incepe

        for (int end = 1; end <= n; end++,

                                    start += 2){

            printf("%d", start);

            if (end < n)

                printf("%c", sign);

        } //printeaza n termeni impari care compun numarul n^3

    }

    else {

        for (int ord = -n + 1; ord > 1; ord--){

            for(int count = 1; count <= ord; count++){

                if(ord < -n + 1)

                    start += 2;

            }

        }

        start = -start;

        for (int end = -n; end > 0; end--,

                                    start += 2){

            printf("%d", start);

        }

} //la else actioneaza acelasi principiu, doar ca putin schimbat pentru ca

numerele sa ramana succesive

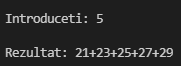
    printf("\n");

    return (0);

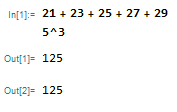
}

**Execuția programului:**

* Set de date nr. 1: 5



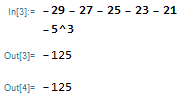
Calcule confirmative în Wolfram:



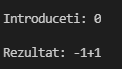
* Set de date nr. 2: -5

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

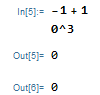
Calcule confirmative în Wolfram:



* Set de date nr. 3: 0



Calcule confirmative în Wolfram:



* Set de date nr. 4: „Nu”

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

„Nu” este de tip char \*, ceea ce nu este int și dă eroare ca rezultat.

Principiul de bază aici e că fiecare număr n^3 e compus din n termeni impari consecutivi celor din care e compus predecesorul lui n. Adică, daca 3 e compus din 7+9+11, atunci 4: 13+15+17+19, 5: 21+23+25+27+29 ș.a.m.d. Singura excepție regulei date este 0^3, care este egal cu -1+1.

**Concluzii:**

1. Instrucțiunile ciclice expandează posibilitățile limbajului C, ele, esențial, fiind o serie de if-uri omogene, care permit repetiția aceleiași instrucțiuni, sau aceluiași bloc de cod de un numar de ori definit de utilizator.
2. Instrucțiunile cilcice pot fi combinate cu cele condiționale pentru a forma o divergență în modul de executare a codului. În cazul meu, în dependență de valoarea lui n, fie n > 0, n < 0 sau n == 0, se execută regiuni diferite ale programului meu, adaptate condițiilor susmenționate.
3. Pentru a rezolva problema desemnată mie, era nevoie de a o adapta către punctele forte ale calculatorului. Așadar, varianta optimă a fost sa identific un șablon după care toate numerele se calculează, ceea ce și am facut, explicând-ul mai sus.

**Surse:**

* **Manual de programare in limbajul C++.** Capitolul: Instrucţiuni repetitive <https://cplusplusifier.fandom.com/wiki/Instruc%C5%A3iuni_repetitive>