**MINISTERUL EDUCAȚIEI ȘI CERCETĂRII AL REPUBLICII MOLDOVA**

**Universitatea Tehnică a Moldovei**

**Facultatea Calculatoare, Informatică şi Microelectronică**

**Departamentul Informatică şi Ingineria Sistemelor**

**RAPORT**

Lucrare de laborator nr. 3

la cursul ***„Structuri de date şi algoritmi”***

**Tema: Recursia**

**A efectuat :**   **St. gr. CR-221FR Serba Cristina**

**A verificat: Asis.univ. Toma Olga**

**Chișinău 2023**

**Cuprins**

[INTRODUCERE 2](#_Toc136204232)

[REALIZAREA PRACTICĂ A SARCINII DE LUCRU 3](#_Toc136204233)

[CONCLUZII 5](#_Toc136204234)

[BIBLIOGRAFIE 6](#_Toc136204235)

# 

# INTRODUCERE

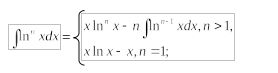
Recursivitatea reprezintă un concept esențial în programarea și este utilizată pentru a rezolva probleme complexe prin împărțirea acestora în subprobleme mai mici și mai simple. Atunci când o funcție este apelată în interiorul ei însăși, aceasta creează un lanț de apeluri recursive, unde fiecare apel se încheie numai atunci când condiția de oprire este îndeplinită. Recursivitatea poate fi o tehnică puternică și versatilă în programare, dar trebuie utilizată cu atenție. Este important să avem în vedere eficiența și gestionarea stivei de apeluri recursive, deoarece funcțiile recursive pot consuma multă memorie și pot duce la depășirea stivei (stack overflow) dacă numărul de apeluri recursive este prea mare.

În cazul dat recursia este folosită pentru calcularea integralei, folosind formula Newton-Leibniz.

# REALIZAREA PRACTICĂ A SARCINII DE LUCRU

**Sarcina lucrării de laborator (varianta 13):**

Să se calculeze pentru limitele indicate de integrare [a,b]:

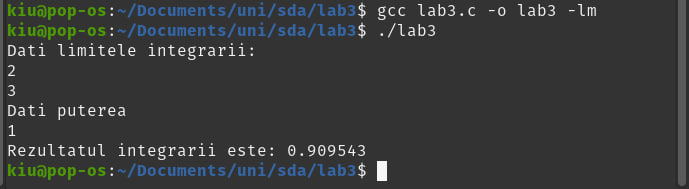


|  |
| --- |
| #include <stdio.h>  #include <math.h>  /\* Să se calculeze pentru limitele indicate de integrare [a,b]: \*/    float integral(float A, float B, int n)  {  if(n > 1) return ((B\*pow(log(B), n) - n\*integral(A, B, n-1)) - (A\*pow(log(A), n) - n\*integral(A, B, n-1)));  else if(n = 1)  return ((B\*log(B) - B) - (A\*log(A) - A));  }  int main()  {  float A, B; //limitele  int n; //puterea  printf("Dati limitele integrarii:\n");  scanf("%f%f", &A, &B);  printf("Dati puterea\n");  scanf("%d", &n);  printf("Rezultatul integrarii este: %f\n", integral(A, B, n));  return 0;  } |

Acest program folosește recursivitatea pentru a calcula rezultatul integralei definite. În interiorul funcției integral, sunt tratate două cazuri folosind instrucțiunea if-else.

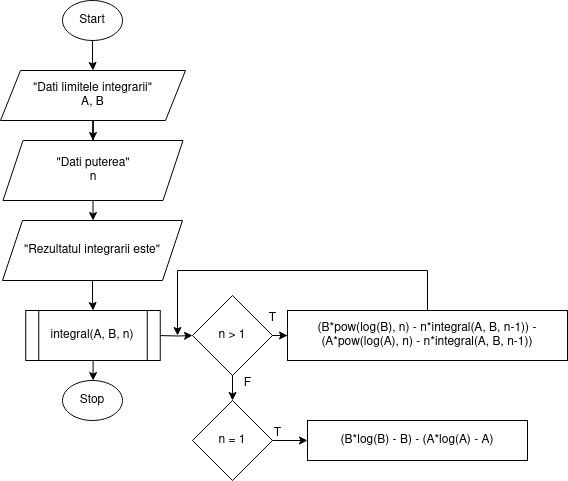
a. Dacă n este mai mare decât 1, se calculează valoarea integrală folosind formula regulei trapezoidale recursive: (B \* pow(log(B), n) - n \* integral(A, B, n-1)) - (A \* pow(log(A), n) - n \* integral(A, B, n-1)).

b. Dacă n este 1, se calculează valoarea integrală folosind formula regulei trapezoidale pentru puterea 1: (B \* log(B) - B) - (A \* log(A) - A). Prin apelarea recursivă a funcției integral, se realizează împărțirea problemei în subprobleme mai mici și se obține rezultatul final.



**Fig.1.1 *Rezultatul execuției***

Figura 1.1 prezintă rezultatul executiei programului



**Fig.1.2 *Schema bloc a programului***

Figura 1.2 prezintă schema bloc a programului

# CONCLUZII

În concluzie, programul furnizat utilizează recursivitatea pentru a calcula rezultatul unei integrale definite folosind regula trapezoidală. Aceasta demonstrează capacitatea de a rezolva probleme matematice complexe prin împărțirea lor în subprobleme mai simple și repetând procesul de calcul până la atingerea unei condiții de oprire. Utilizarea recursivității permite evitarea nevoii de a implementa o buclă iterativă complexă, și în schimb, subproblemele sunt rezolvate prin apeluri recursive ale aceleiași funcții.

Deși recursivitatea poate oferi soluții elegante și concize, este important să acordăm atenție eficienței și gestionării stivei de apeluri recursive. Un număr mare de apeluri recursive poate duce la depășirea stivei, ceea ce poate avea un impact negativ asupra performanței programului și poate duce la erori.

# BIBLIOGRAFIE

[***https://app.diagrams.net/***](https://app.diagrams.net/)