ALGORITMI LINIARI (continuare)

R1 (1*).

Enunţul problemei: Să se descrie un algoritm pentru determinarea ariei unui pătrat, cunoscând lungimea laturii sale.

Metoda de rezolvare: Algoritmul constă în citirea de la tastatură a lungimii laturii pătratului şi apoi determinarea şi afişarea ariei acestuia (aria = latura^2). De exemplu, pentru un pătrat cu latura de 4 (unităţi de măsură), aria este 16, iar pentru un pătrat cu latura de 1.2 (unităţi de măsură), aria este 1.44.

Descrierea algoritmului în pseudocod:

```
citeşte latura
aria ← latura^2
scrie aria
```

Descrierea algoritmului în C++(CodeBlocks):

```
#include<iostream>
#include<iomanip> //pentru setprecision
using namespace std;
int main(){
    float latura, aria;
    cout<<"Dati lungimea laturii: "; cin>>latura;
    aria = latura*latura;
    cout<<"Aria patratului este: "<<setprecision(2)<<aria<<endl;
    return 0;
}
```

sau neutilizând variabila aria

```
#include<iostream>
#include<iomanip>
using namespace std;
int main(){
    float latura;
    cout<<"Dati lungimea laturii: "; cin>>latura;
    cout<<"Aria patratului este: "<<setprecision(2)
        <<latura*latura <<endl;
    return 0;
}
```

Rulare:

```
Dati lungimea laturii: 1.2 <Enter>
Aria patratului este: 1.44
```

Explicaţii cod:

Linia

```
float latura;
```

semnifică declararea unei variabile de tip "`float`" (unul din tipurile reale din C/C++, cu valori în intervalul $[-10^{308}, -10^{-308}] \cup [10^{-308}, 10^{308}]$).

Linia cout<<"Dati lungimea laturii: "; afişează pe ecran textul dintre ghilimele.

Linia cin>>latura; conţine o funcţie declarată în fişierul header "`iostream`" (similară funcţiei `cout` doar că este folosită pentru intrări de la dispozitivul standard de intrare - tastatura). Aşadar, funcţia aceasta presupune interpretarea caracterelor date de la tastatură ca un întreg zecimal şi memorarea acestora în variabila `latura`.

Linia cout<<"Aria patratului este: "<<latura*latura; afişează textul dintre ghilimele, apoi afişează valoarea expresiei latura*latura.

R2 (1*).

Enunţul problemei: Scrieţi în pseudocod şi în C++ algoritmul pentru calculul ariei unui triunghi, când se cunoaşte o bază a triunghiului şi înălţimea pe această bază (aria = baza*inaltimea/2). De exemplu, pentru baza=8, inaltimea=5 se obţine valoarea ariei: aria = 8*5 / 2 = 20, iar pentru baza=5, inaltimea=3 se obţine valoarea ariei: aria = 5*3/2 = 7,5.

Metoda de rezolvare:

Algoritmul constă în citirea valorilor variabilelor ce reprezintă baza şi înălţimea pe aceasta, apoi calculul ariei şi în final afişarea valorii variabilei calculate.

Descrierea algoritmului în pseudocod:

```
citeste b,h            *de exemplu: 5, respectiv 3
A ← b*h/2              *calculul ariei
scrie A                *data de ieşire – doar aria
```

Descrierea algoritmului în C++ (folosind CodeBlocks):

```
#include<iostream>
using namespace std;
int main()
{
 int b,h;
 float A;

 cout<<"Dati lungimea bazei: "; cin>>b;
 cout<<"Dati lungimea inaltimii: "; cin>>h;

 A = b*h / 2.0;   //b si h fiind de tip int
 //am convertit numitorul pentru a fi impartire reala

 cout<<"Aria este: "<< A << endl;
 return 0;
}
```

sau

```
#include<iostream>
using namespace std;
int main()
{
 float b,h,A;
 cout<<"Dati lungimea bazei: "; cin>>b;
 cout<<"Dati lungimea inaltimii: "; cin>>h;

 A = b*h / 2;

 cout<<"Aria este: " << A << endl;
}
```

Rulare:

```
Dati lungimea bazei: 5 <Enter>
Dati lungimea inaltimii: 3 <Enter>
Aria este: 7.5
```

R3 (1*).

Enunţul problemei: Să se descrie un algoritm care să convertească o temperatură din grade Celsius în echivalentul ei în grade Farenheit. De exemplu, 0°C = 32°F, 23°C = 73,4°F, 29°C = 84,2°F, 25°C = 77°F.

Metoda de rezolvare: Formula de echivalenţă între grade Celsius şi grade Farenheit este

$$F = 32 + \frac{9}{5}C.$$

Astfel, algoritmul se reduce la citirea valorii temperaturii în grade Celsius şi apoi afişarea expresiei din dreapta semnului egal de mai sus. De observat că valorile gradelor Celsius sunt întregi, iar gradele Farenheit sunt reale!!!

Descrierea algoritmului în pseudocod:

```
citeşte gr_C
gr_F ← 32+9/5gr_C
scrie gr_F
```

Descrierea algoritmului în C++(CodeBlocks):

```
#include<iostream>
using namespace std;
int main() {
    int gr_C;
    float gr_F;
    cout<<"Dati numarul de grade Celsius: "; cin>>gr_C;
    gr_F = 32 + 9.0*gr_C/5;
    cout<<gr_C<<" grade Celsius este echivalent cu "<<gr_F<<"
grade Farenheit"<<endl;
    return 0;
}
```

Rulare:

```
Dati numarul de grade Celsius: 7
Numarul de grade Farenheit echivalente: 44.6
```

R4 (2*- suplimentar).

Enunţul problemei: Să se descrie un algoritm pentru determinarea valorii unui depozit bancar (cu termen la 1 an şi cu capitalizarea dobânzii) după 2 ani, ştiind valoarea iniţială şi dobânda anuală.

Metoda de rezolvare:

De exemplu, pentru o valoare iniţială de 500 lei şi dobândă anuală de 10%, după primul an depozitul va avea valoarea 500+10%*500 = 550 lei, iar după al doilea an depozitul va avea suma de 550+10%*550 = 605 lei. Putem nota cu v0 valoarea iniţială a depozitului şi cu dob valoarea dobânzii (în exemplul nostru 10 (ca sa nu introducem 0.1), subînţelegându-se 10%).

Pentru o valoare iniţială de 500 lei şi dobândă anuală de 1.7%, după al doilea an depozitul va avea suma de 517.14 lei.

Descrierea algoritmului în pseudocod:

```
citeste v0,dob        *de exemplu, 500, respectiv 10
v1 ← v0 + v0*dob/100  *valoare depozit dupa un an
v2 ← v1 + v1*dob/100  *valoare depozit dupa al doilea an
scrie v2              *data de ieşire – doar v2
```

Dacă se dorea se putea afişa şi valoarea depozitului după primul an, căci valoarea sa era reţinută în variabila v1.

Se poate optimiza acest algoritm, pentru a nu folosi încă variabile diferite pentru memorarea valorilor depozitului după primul an, respectiv al doilea an.

```
citeste v0,dob          *de exemplu, 500, respectiv 10
v ← v0 + v0*dob/100     *valoare depozit dupa un an
v ← v + v*dob/100       *valoare depozit dupa al doilea an
scrie v                 *valoare depozit dupa al doilea an
```

A treia instrucţiune evaluează membrul drept (vechea valoare a variabilei *v*, adică cea de după primul an, plus aceasta înmulţită cu dob/100), iar noua valoare a variabilei *v* va fi exact rezultatul acestui membru drept.

Sau putem folosi doar o singură variabilă care reprezintă pe rând valoarea depozitului la momentul inițial, apoi după primul an, respectiv după doi ani.

```
citeste v,dob           *de exemplu, 500, respectiv 10
v ← v + v*dob/100       *valoare depozit dupa un an
v ← v + v*dob/100       *valoare depozit dupa al doilea an
scrie v                 *valoare depozit dupa al doilea an
```

Descrierea algoritmului în C++(CodeBlocks):

```
#include <iostream> //pt.cin,cout
#include <iomanip> //pt.setprecision
using namespace std;
int main()
{
 float v,dob;
 cout<<"Dati valorea initiala a depozitului: "; cin>>v;
 cout<<"Dati dobanda (de exemplu 10): "; cin>>dob;
 v = v + v*dob/100; //valoarea depozit dupa un an
 v2 = v + v*dob/100; //valoare depozit dupa al doilea an
 cout<<"Valoarea depozitului dupa 2 ani este: "<<fixed<<
setprecision(2)<< v << endl;  //fix cu 2 zecimale
 return 0;
}
```

R5 (2*).

Enunţul problemei: Să se descrie un algoritm pentru determinarea mediei aritmetice, mediei geometrice şi a mediei armonice a două valori reale strict pozitive date.

Metoda de rezolvare:

Dacă notăm cele două valori strict pozitive cu *a*, respectiv *b*, atunci:

$$m_a = \frac{a+b}{2},\ m_g = \sqrt{ab} \text{ şi } m_{arm} = \frac{2}{\frac{1}{a}+\frac{1}{b}}.$$

Descrierea algoritmului în pseudocod:

```
citeste a,b             *presupunem că ambele sunt strict pozitive
m_a ← (a+b)/2           *media aritmetică
m_g ← √(ab)             *media geometrică
m_arm ← 2/(1/a+1/b)     *media armonică
scrie m_a, m_g, m_arm   *datele de ieşire
```

Descrierea algoritmului în C++(CodeBlocks):

```
#include <iostream>
#include <iomanip>
#include <math.h> //pt.sqrt = radical de ordinul 2
using namespace std;
```

```
int main()
{
 float a,b,ma,mg,marm;

 cout<<"a= "; cin>>a;
 cout<<"b= "; cin>>b; //presupunem a>0 si b>0
 //presupunem ca s-au introdus doua valori strict pozitive

 ma = (a+b)/2;
 mg = sqrt(a*b);
 marm = 2 / (1/a + 1/b);

 cout<<"Media aritmetica = "<<fixed<<setprecision(2)<<ma<<endl;
 cout<<"Media geometrica = "<<mg<<endl; //tot fix cu 2 zecimale
 cout<<"Media armonica = "<<marm<<endl;
 return 0;
}
```

Rulare:

```
a = 2 <Enter>
b = 8 <Enter>
Media aritmetica = 5.00
Media geometrica = 4.00
Media armonica = 3.20
```

sau

```
a = 3 <Enter>
b = 8 <Enter>
Media aritmetica = 5.50
Media geometrica = 4.90
Media armonica = 4.36
```

R6 (1*-suplimentar).

Enunţul problemei: Să se determine algoritmul pentru însumarea a două fracţii (fără simplificări ulterioare).

Metoda de rezolvare: De exemplu, $\overset{8/}{\frac{1}{2}}+\overset{2/}{\frac{3}{8}}=\frac{14}{16}$.

Să notăm numărătorul, respectiv numitorul primei fracţii cu *a*, respectiv *b*, apoi numărătorul, respectiv numitorul celei de-a doua fracţii cu *c*, respectiv *d* şi în cele din urmă, numărătorul, respectiv numitorul fracţiei finale cu *e*, respectiv *f*. Atunci

$$\overset{d/}{\frac{a}{b}}+\overset{b/}{\frac{c}{d}}=\frac{e}{f},$$

unde $e = ad + bc$, iar $f = bd$.

Descrierea algoritmului în pseudocod:

```
citeste a,b,c,d        *datele de intrare
e ← ad + bc            *numărătorul fracţiei finale
f ← bd                 *numitorul fracţiei finale
scrie e,f              *datele de ieşire (fracţia finală)
```

Descrierea algoritmului în C++(CodeBlocks):

```
#include<iostream>
using namespace std;
int main()
{   int a,b,c,d,e,f; //declararea tuturor variabilelor
    cout<<"Dati numaratorul primei fractii: "; cin>>a;
         cout<<"Dati numitorul primei fractii: "; cin>>b;
```

```
    cout<<"Dati numaratorul celei de-a doua fractii: "; cin>>c;
    cout<<"Dati numitorul celei de-a doua fractii: "; cin>>d;
    e = a*d + b*c;
    f = b*d;
    cout<<"Fractia finala: "<<e<<" / "<<f<<" = "<<float(e)/f<<endl;
    return 0;
}
```

Rulare:

```
Dati numaratorul primei fractii: 1 <Enter>
Dati numitorul primei fractii: 2 <Enter>
Dati numaratorul celei de-a doua fractii: 3 <Enter>
Dati numitorul celei de-a doua fractii: 8 <Enter>
Fractia finala: 14 / 16
```

R7 (2*).

Enunţul problemei: Să se calculeze aria unui triunghi cunoscând lungimile laturilor sale *a*, *b* şi *c*.

Metoda de rezolvare: Se foloseşte formula lui Heron: *aria* = $\sqrt{p(p-a)(p-b)(p-c)}$, unde *p* este semiperimetrul triunghiului, adică $p = \frac{a+b+c}{2}$.

Descrierea algoritmului în pseudocod:

```
citeşte a,b,c                    *se citesc datele de intrare
p ← (a+b+c)/2                    *intai se calculeaza semiperimetrul
aria ← sqrt(p(p-a)(p-b)(p-c))    *se calculeaza aria triunghiului
scrie aria                       *se afiseaza valoarea variabilei aria
```

Descrierea algoritmului în C++:

```
#include <iostream>
#include <math.h> //pt sqrt
using namespace std;
int main() {
 float a,b,c,p,aria; //declararea tuturor variabilelor
 cout<<"a = "; cin>>a;
 cout<<"b = "; cin>>b;
 cout<<"c = "; cin>>c;
 //presupunem ca ele pot forma laturile unui triunghi

 p = (a+b+c)/2; //semiperimetrul
 aria = sqrt(p*(p-a)*(p-b)*(p-c));

 cout<<"Aria triunghiului este: "<<aria<<endl;
 return 0;                        //cu 4 zecimale (implicit)
}
```

Rulare:

```
a = 3 <Enter>
b = 4 <Enter>
c = 6 <Enter>
Aria triunghiului este: 6
sau
a = 2 <Enter>
b = 2 <Enter>
c = 2 <Enter>
Aria triunghiului este: 1.73205
```

R8 (1*-suplimentar).

Enunţul problemei: Să se scrie un algoritm pentru determinarea ariei unei elipsei, cunoscând lungimile semiaxelor sale.

Metoda de rezolvare: Aria unei elipse este $A = \pi ab$, unde *a* şi *b* sunt lungimile semiaxelor elipsei. Putem evita folosirea variabilei `aria` prin afişarea directă a expresiei ariei elipsei.

Descrierea algoritmului în pseudocod:

```
citeşte a,b
aria ← πab
scrie aria
```

Descrierea algoritmului în C++(CodeBlocks):

```
#include <iostream>
#include <math.h> //pentru M_PI
using namespace std;
int main(){
    float a,b,aria;
    cout<<"Dati lungimea primei semiaxe a="; cin>>a;
    cout<<"Dati lungimea celei de-a doua semiaxe b=";
    cin>>b;
    aria = M_PI*a*b;
    cout<<"Aria elipsei este: "<<aria<<endl;
    return 0;
}
```

Rulare:

Dati lungimea primei semiaxe a = 1
Dati lungimea celei de-a doua semiaxe b = 2
Aria elipsei este: 6.28319

R9 (2-3* suplimentar).

Enunţul problemei: Să se descrie un algoritm pentru determinarea ariei unui triunghi, cunosc lungimile a două laturi şi unghiul dintre acestea.

Metoda de rezolvare: Reamintim că aria unui triunghi în funcţie de două laturi şi unghiul dintre acestea este $A = \frac{ac\sin B}{2}$.

Descrierea algoritmului în pseudocod:

```
citeşte a,c,B
aria ← ac sin B / 2
scrie aria
```

Descrierea algoritmului în C++(CodeBlocks):

```
#include <iostream>
#include <math.h> //pentru functia sinus
using namespace std;
int main()
{
 float a,c,B;
 cout<<"Dati valoarea unei laturi a triunghiului: ";
 cin>>a;
 cout<<"Dati valoarea altei laturi a triunghiului: "; cin>>c;
 cout<<"Dati valoarea unghiului dintre laturi (in grade):";
 cin>>B;
 cout<<"Aria triunghiului este: "<<a*c*sin(B*M_PI/180)/2<<endl;
 //argumentul functiei sin apeleaza unghiul convertit
 //din grade in radiani
 return 0; }
```

Rulare:

```
Dati valoarea unei laturi a triunghiului: 1
Dati valoarea altei laturi a triunghiului: 2
Dati valoarea unghiului dintre laturi (in grade): 30
Aria triunghiului este: 0.5
```

R10 (2-3* suplimentar).

Enunţul problemei: Să se determine un algoritm pentru însumarea a două numere de forma $\overline{ab}+\overline{c}$ (adică un număr de două cifre cu unul de o cifră). De exemplu: 42+3=45, 49+7=56, 94+8=102.

Metoda de rezolvare:

Numărul final poate fi de maxim 3 cifre (de exemplu: 42+3=45, 49+7=56, 94+8=102); să notăm numărul final $\overline{efg}$. Se începe însumarea de la cifra unităţilor: adunăm cifra *c* cu cifra *b* şi ce trece peste 10 va fi transport la cifra zecilor, *g* fiind restul împărţirii sumei lor la 10 (lucrăm cu un sistem zecimal). Apoi însumăm *a* cu eventualul transport de la unităţi şi ce depăşeşte 10 (câtul împărţirii acestei sume la 10) trece ca cifra sutelor, adică e, iar restul rămâne ca cifra zecilor (adică *f*).

Descrierea algoritmului în pseudocod:

```
citeste a,b,c    *a,b,c cifre
g ← rest împărţire (b+c) la 10   *cifra unităţilor
f ← rest împărţire (a+cât împărţire (b+c) la 10) la 10
e ← cât împărţire (a+cât împărţire (b+c) la 10) la 10
scrie e, f, g    *sutele, zecile, unităţile
```

Descrierea algoritmului în C++:

```
#include<iostream>
#include<math.h>
using namespace std;
int main() {
 int a,b,c; //cifrele celor doua numere: ab, c
 int e,f,g; //numarul final poate fi de 3 cifre
 cout<<"Dati cifra zecilor a primului numar: ";
 cin>>a;
 cout<<"Dati cifra unitatilor a primului numar: ";
 cin>>b;
 cout<<"Dati cifra celui de-al doilea numar: ";
 cin>>c;

 g = (b+c)%10; // % = restul impartirii
 f = (a+(b+c)/10)%10;
 e = (a+(b+c)/10) / 10; //  / = cat

 cout<<"Numarul rezultat este: "<<e<<f<<g<<endl;
 return 0;
}
```

Rulare:

```
Dati cifra zecilor a primului numar: 4
Dati cifra unitatilor a primului numar: 9
Dati cifra celui de-al doilea numar: 7
Numarul rezultat este: 056
```

R11 (2-3* suplimentar).

Enunţul problemei: Să se determine permutările circulare ale unui număr de 3 cifre. De exemplu pentru *n* = 123 se se va afişa 231 şi 312.

Metoda de rezolvare:
Pornind întâi de la *n* =123, 231 = 23*10 +1 = ultimele 2 cifre ale lui *n* * 10 + prima cifra = = (rest împărţire *n* la 100) * 10 + (cât împărţire *n* la 100). Procedăm similar cu numărul obţinut, care se poate reţine tot în variabila *n*.

Descrierea algoritmului în pseudocod:

```
citeste n  *presupunem că se dă n cu 3 cifre
n ← (rest împărţire n la 100) * 10 + (cât împărţire n la 100)
scrie n    *prima permutare, de ex. 231
n ← (rest împărţire n la 100) * 10 + (cât împărţire n la 100)
scrie n    *a doua permutare
```

Descrierea algoritmului în C++:

```
#include <iostream>
using namespace std;
int n;
int main(){
  cout<<"Dati numarul n (de 3 cifre): ";
  cin >> n;
  n = n%100 * 10 + n/100;
  cout << "Prima permutare: " << n;
  n = n%100 * 10 + n/100;
  cout << endl << "A doua permutare: " << n<<endl;
  return 0;
}
```

Rulare:

```
Dati numarul n (de 3 cifre): 123 <Enter>
Prima permutare: 231
A doua permutare: 312
```

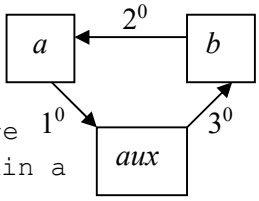
Explicaţii cod C++:

Variabila *n* în care se reţin pe rând numărul iniţial, apoi cele două permutări este dat iniţial cu 3 cifre; el este memorat într-un variabilă de tip int (reamintesc că domeniul de valori a tipului de date int este -32768..32767, deci ne încadrăm cu orice număr de 3 cifre).
La prima instructiune de atribiure, valoarea lui *n* se modifică în funcţie de vechea valoare a sa. De observat că la: n%100*10 se execută întâi % apoi *, pentru că %, * şi / au aceeaşi prioritate şi se execută de la stânga la dreapta. Imediat cum am calculat prima permutare (în variabila *n*) o şi afişăm pentru că vom modifica din nou valoarea lui *n*, când se determină tot în variabila *n* cea de-a doua permutare.

R12 (1*).

Enunţul problemei: Să se descrie algoritmul pentru interschimbarea (inversarea) conţinutul a două variabile date.

Metode de rezolvare:



O primă variantă foloseşte o a treia variabilă, aux.

Descrierea algoritmului în pseudocod:

```
citeşte a,b *se citesc valorile datelor de intrare
aux ← a     *in aux se retine temporar valoarea din a
a ← b       *in a punem acum valoarea din b
b ← aux     *in b punem ce depozitasem in aux, adica val din a
scrie a,b   *valorile interschimbate (inversate)
```

Descrierea algoritmului în C++:

```
#include <iostream>
using namespace std;
int main(){
    int a,b,aux;
    //aux este de acelasi tip de date cu a si b
    cout<<"a = "; cin>>a;
    cout<<"b = "; cin>>b;
    aux = a; a=b; b=aux;
    cout<<"a="<<a<<endl;
    cout<<"b="<<b<<endl;
    return 0;
}
```

Rulare:

```
a = 2 <Enter>
b = 5 <Enter>
a=5
b=2
```

O a doua variantă, nu foloseşte nicio variabilă suplimentară.

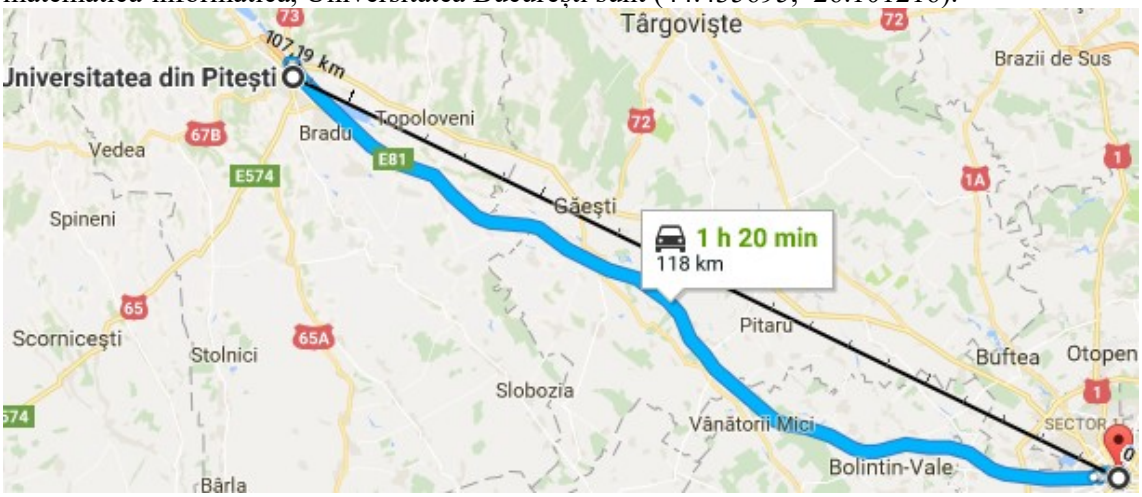
Descrierea algoritmului în pseudocod:

```
citeşte a,b
a ← a-b
b ← a+b   *b va memora valoarea initiala a var. a
a ← b-a
scrie a,b
```

De exemplu, dacă se citesc valorile a = 4 şi b = 9 se obţine

```
a ← a-b   =>   a ← -5          => a = -5 şi b = 9
b ← a+b   =>   b ← -5+9        => a = -5 şi b = 4
a ← b-a   =>   a ←  4-(-5)     => a = 9 şi b = 4
```

Pentru avansați: Să se descrie un algoritm pentru a determinara distanța dintre două puncte de pe Pământ, date prin coordonatele geografice exprimate în grade zecimale (latitudine $\in [-90^0, 90^0]$ – la 0^0 este Ecuatorul și longitudine $\in [-180^0, 180^0]$). De exemplu, coordonatele Universitatii din Pitești din Targu din Vale sunt (44.854735, 24.881994), iar a Facultății de matematică-informatică, Universitatea București sunt (44.435695, 26.101216).



Se va folosi algoritmul lui Haversine:

Date de intrare: P1(lat1, long1) și P2(lat2, long2)
R ← 6371 //raza Pamantului (in km)
Δlat ← (lat1 - lat2) * π / 180 //dif.latit. convertită în radiani
Δlong ← (long1 - long2) * π / 180
A ← $\sin^2$ (Δlat/2) + cos(lat1) * cos(long1) * $\sin^2$ (Δlong/2)
C ← 2*atan2($\sqrt{A}$, $\sqrt{1-A}$) //atan2(x,y)=tan(x/y)
D ← R*C
Dată de ieșire: D.