Algoritmi care lucreaza pe numere (fara tablouri sau alte elemente structurate)
Partea I
13.02.2021

1. Alin are în față ***M*** mingi de tenis roșii și ***N*** mingi de tenis galbene. El dorește să formeze mai multe grupuri, astfel încât toate grupurile să conțină același număr de mingi, toate mingile dintr-un grup să aibă aceeași culoare și numărul de mingi dintr-un grup să fie cât mai mare posibil. După ce a reușit să facă împărțeala, Alin a primit alt set de mingi roșii și galbene. Imediat a început să le împartă în grămezi de M mingi roșii și N mingi galbene. Fiecare grămadă conține mingi de aceeași culoare. El dorește să aleagă un număr de X grămezi cu mingi roșii și Y grămezi cu mingi galbene astfel încât diferența dintre numărul total de mingi roșii și numărul total de mingi galbene să fie egală cu numărul de mingi dintr-unul din grupurile construite la început.
Să se citească de la tastatură numerele M și N. Se vor afișa pe ecran numărul de mingi dintr-o grupă construită, numărul de grămezi de mingi roșii și galbene.

Exemple

Date de intrare	Rezultat
4 6	2 1 1
10 5	5 0 1
24 63	3 8 3

Analiză

- trebuie determinat cel mai mare divizor comun a două numere M și N
- trebuie să se determine două numere X și Y astfel încât |MX – NY| = cmmdc(M,N)
- algoritmul extins al lui Euclid

Rezolvare C++

```
/*
Problema a fost compilata cu Microsoft Visual Studio varianta Community 2017 si Apple
clang 12.0.0
*/
#include <iostream>

/*
Descriere:    algoritmul lui Euclid in varianta extinsa
Date:         a, b - numere naturale, x, y
Rezultat:     cmmdc pentru cele doua numere naturale
*/
int cmmdc_extins(int a, int b, int& x, int& y)
{
	x = 0;
	y = 1;
	int u = 1;
	int v = 0;
	int q, r, m, n;
	while (a != 0)
	{
		q = b / a;
		r = b % a;
		m = x + u * q;
		n = y + v * q;
		b = a;



		a = r;
		x = u;
		y = v;
		u = m;
		v = n;
	}
	return b;
}

int main()
{
	int x = 0, y = 0, cmmdc = 0;
	cmmdc = cmmdc_extins(4, 6, x, y);
	std::cout << "cmmdc extins: " << cmmdc << ", x = " << x << ", y = " << y <<
std::endl;
	cmmdc = cmmdc_extins(10, 5, x, y);
	std::cout << "cmmdc : " << cmmdc << ", x = " << x << ", y = " << y << std::endl;
	cmmdc = cmmdc_extins(24, 63, x, y);
	std::cout << "cmmdc : " << cmmdc << ", x = " << x << ", y = " << y << std::endl;
	return 0;
}
```

Rezolvare Pascal

```
{
Descriere:      algoritmul lui Euclid in varianta extinsa
Date:           a, b - numere naturale
Rezultat:       cmmdc pentru cele doua numere naturale
}
program cmmdc;
function cmmdc_extins(a:integer; b:integer):integer;
var x,y,u,v,q,r,m,n:integer;
begin
        x := 0;
        y := 1;
        u := 1;
        v := 0;
        while (a <> 0) do
        begin
                q := b div a;
                r := b mod a;
                m := x + u * q;
                n := y + v * q;
                b := a;
                a := r;
                x := u;
                y := v;
                u := m;









        v := n;
        end;
        writeln('x = ',x,', y = ', y);
        cmmdc_extins:=b;
end;
begin

   writeln( 'cmmdc extins: ',cmmdc_extins(4, 6));
        writeln( 'cmmdc extins: ',cmmdc_extins(10, 5));
        writeln( 'cmmdc extins: ',cmmdc_extins(24, 63));
end.
```

2. O companie nou înființată de taxi are următorul model de taxare pentru șoferii săi: un șofer începe săptămâna cu 0 unități monetare, în timpul săptămânii fiecare șofer trebuie să predea companiei tot câștigul realizat în momentul în care are asupra sa N unități monetare, orice câștig suplimentar rămâne în posesia șoferului. Compania are angajați K șoferi, pentru fiecare se cunoaște numărul curselor efectuate și câștigul obținut. În nici un moment un șofer nu poate avea asupra sa mai mult de N-1 unități monetare (decât imediat

după o cursă) deoarece imediat ce are mai mult de N unități monetare trebuie să le predea companiei. Mai mult, dacă după ce predă N unități monetare, numărul său de unități monetare este din nou mai mare decât N, el va preda din nou N unități monetare până când va avea un câștig mai mic decât N. Determinați câștigul de la sfârșitul săptămânii pentru un șofer.

Să se citească de la tastatură valoarea N, numărul curselor pentru un șofer și câștigul pentru fiecare cursă. Programul va afișa câștigul obținut de șofer la sfârșitul săptămânii.

Exemple

Date de intrare	Rezultat
50 4 1 2 3 4	10
50 5 10 20 30 40 50	0
50 3 874 9735 835	44

Analiză

- aritmetică modulara (însumări modulo)
- pe baza modului de efectuare a operațiilor în aritmetica modular, următoarea relație se respectă:

$$(a + b) \bmod n = ((a \bmod n) + (b \bmod n)) \bmod n$$

Rezolvare C++

```
/*
Problema a fost compilata cu Microsoft Visual Studio varianta Community 2017 si Apple
clang 12.0.0
*/
#include <iostream>
```

```
/*
Descriere:    calculeaza suma modulo n pentru doua numere naturale
Date:         a, b, n - numere naturale,
Rezultat:     (a + b) modulo n
*/
int adunare_modulo(int a, int b, int n)
{
	return (a + b) % n;
}

/*
Descriere:    detrmina prin sume modulo castigul unui sofer
Date:         n - numar natural
Rezultat:     castigul unui sofer la sfarsitul saptamanii
*/
int determinaCastigSofer(int n)
{
	int curse;
	int castig = 0;
	int castig_cursa;
	std::cin >> curse;
	while (curse)
	{
		std::cin >> castig_cursa;
		castig = adunare_modulo(castig, castig_cursa, n);
		curse--;
	}
	return castig;
}

int main()
{
	int n = 1;

	while (n)
	{
		std::cin >> n;
		std::cout << "castigul soferului este: " << determinaCastigSofer(n) <<
std::endl;
	}

	return 0;
}
```

Rezolvare Pascal

```
program castig;
var n:integer;
{
Descriere:    calculeaza suma modulo n pentru doua numere naturale
Date:         a, b, n - numere naturale,
Rezultat:     (a + b) modulo n
}
```

```
function adunare_modulo(a:integer; b:integer; n:integer):integer;
begin
        adunare_modulo:= (a + b) mod n;
end;

{
Descriere:      detrmina prin sume modulo castigul unui sofer
Date:           n - numar natural
Rezultat:       castigul unui sofer la sfarsitul saptamanii
}
function determinaCastigSofer(n:integer):integer;
var curse, castig, castig_cursa:integer;
begin
        castig := 0;
        readln(curse);
        while (curse<>0) do
        begin
                readln(castig_cursa);
                castig := adunare_modulo(castig, castig_cursa, n);
                dec(curse);
        end;
        determinaCastigSofer:=castig;
end;
begin
  n := 1;

        while (n<>0) do
        begin
                readln(n);
                if (n<>0) then writeln('castigul soferului este: ', determinaCastigSofer(n));
        end;
end.
```

3. Fie un număr n, se definește gradul numărului n ca fiind numărul de secvențe "101" din reprezentarea binară a acestuia. De exemplu, numărul 21 are gradul 2 (reprezentarea binară a lui 21 este 10101). Să se citească de la tastatură gradul k și mai multe numere (citirea se termină cu numărul 0), să se afișeze dacă un număr are gradul k.

Exemple

Date de intrare	Rezultat
1 7 5 13 21 29 25 0	numere de grad cel putin 1: 5 13 21 29
2 13 5 21 56 45 360 37 0	numere de grad cel putin 2: 21 45 360

Analiză

- operații la nivel de bit, izolarea celor mai puțin semnificativ trei biți din reprezentarea binară și verificarea dacă sunt o secvență 101, după care se deplasează la stânga cu una sau două poziții (în funcție de rezultatul comparației cu $(101)_2 = (5)_{10}$).

Rezolvare C++

```
/*
Problema a fost compilata cu Microsoft Visual Studio varianta Community 2017 si Apple
clang 12.0.0
*/
#include <iostream>
using namespace std;

const int NRBITI = 32;
int BINAR[NRBITI] = { 0 };

/*
Descriere:    toate elementele vectorului BINAR = 0
 */
void resetBinar()
{
	for (int i = 0; i < NRBITI; ++i)
	{
		BINAR[i] = 0;
	}
}

/*
 * Descriere:   reprezint un numar in baza doi
 * Date:        n - numarul natural de reprezentat in baza doi
 * Rezultat:    numarul de cifre din reprezentarea binara
 */
int calculReprezBinar(int n)
{
	resetBinar();
	int i;
	for (i = 0; n > 0; i++)
	{
		BINAR[i] = n % 2;
		n = n / 2;
	}
	return i + 1;
}

/*
Descriere:    reprezint un numar in baza doi, versiunea II folosind
              operatii pe biti
Date:         n - numarul natural de reprezentat in baza doi
Rezultat:     numarul de cifre din reprezentarea binara
 */
int calculReprezBinar2(int n)
{
	resetBinar();
	int i = 0;
	/*
	se izoleaza fiecare cifra din reprezentarea binara a numarului
	si se stocheaza in vectorul BINAR pe pozitia ei
	*/
	while (n)
	{
		BINAR[i] = n & 1;
		n >>= 1;
		i++;
	}
	return i;
}
```

```
/*
Descriere:    calcul determina gradul unui numar (numarul de secvente 101
              din reprezentarea binara)
Date:         n - numarul natural pentru care trebuie determinat gradul
Rezultat:     gradul numarului
*/
int determinaGrad(int n)
{
	int grad = 0;
	int nrCifre = 0;
	nrCifre = calculReprezBinar2(n);
	for (int i = nrCifre; i >= 2; i--)
	{
		if (BINAR[i] == 1 && BINAR[i - 1] == 0 && BINAR[i - 2] == 1)
			grad++;
	}

	return grad;
}

int main()
{
	int k, numar = 1;
	// se citeste gradul cautat
	cin >> k;
	/* sirul s va stoca numerele de grad cel putin k */
	int s[100] = { 0 };
	int i = 0;
	/* citirea numerelor se termina cu citirea lui 0 */
	while (numar)
	{
		cin >> numar;
		if (k <= determinaGrad(numar))
		{
			s[i] = numar;
			i++;
		}
	}
	cout << "numere de grad cel putin " << k << " :" << endl;
	for (int y = 0; y < i; ++y)
	{
		cout << s[y] << " ";
	}
	cout << endl;
	return 0;
}
```

O abordare mai eficientă poate fi imaginată prin izolarea celor mai puțin semnificativ trei biți din reprezentarea binară și verificarea dacă sunt o secvență 101, după care se deplasează la stânga cu una sau două poziții (în funcție de rezultatul comparației cu $(101)_2 = (5)_{10}$).

```
/*
Problema a fost compilata cu Microsoft Visual Studio varianta Community 2017 si Apple
clang 12.0.0
*/
#include <iostream>
using namespace std;
```

```
/*
Descriere:    calcul determina gradul unui numar (numarul de secvente 101
              din reprezentarea binara)
Date:         n - numarul natural pentru care trebuie determinat gradul
Rezultat:     gradul numarului
*/
int determinaGrad2(int n)
{
	int grad = 0;
	while (n)
	{
		/*
		se izoleaza cei mai putin semnificativ 3 biti si se verifica
		daca sunt o secventa 101
		*/
		if ((n & 7) == 5)
		{
			grad++;
			n >>= 2;
		}
		else
		{
			n >>= 1;
		}
	}
	return grad;
}

int main()
{
	int k, numar = 1;
	// se citeste gradul cautat
	cin >> k;
	/* citirea numerelor se termina cu citirea lui 0 */
	while (numar)
	{
		cin >> numar;
		if (k == determinaGrad2(numar))
		{
			cout << "numarul " << numar << " are gradul " << k << endl;
		}
	}
	return 0;
}
```

Exemple:

Date de intrare si rezultat
2 21 numarul 21 are gradul 2 7 5 29 45 numarul 45 are gradul 2 360 numarul 360 are gradul 2 0
3

```
362
numarul 362 are gradul 3
360
85
numarul 85 are gradul 3
0
```

Rezolvare Pascal

```
program HelloWorld;

const NRBITI = 32;
var  BINAR:array [1..32] of integer;
var k, numar,i,y:integer;
var s:array [1..100] of integer;

procedure resetBinar();
var i:integer;
begin
for i:=1 to  NRBITI do
    begin
        BINAR[i] := 0;
    end;
end;

function calculReprezBinar(n: integer ):integer;
var i:integer;
begin
    resetBinar();
    i := 1;
    while (n>0) do
    begin
    BINAR[i] := n mod 2;
        n := n div 2;
    end;
    calculReprezBinar:=i+1;
end;

function calculReprezBinar2(n: integer):integer;
var i:integer;
begin
    resetBinar();
    i:= 1;
    while (n>0) do
    begin
        BINAR[i]:=n and 1;
        n:=n>>1;
        inc(i);
    end;
    calculReprezBinar2:=i;
end;

function determinaGrad2(n: integer ):integer;
var grad:integer;
begin
    grad:= 0;
    while (n>0) do
    begin
```

```
     if ((n and 7)=5) then
        begin
            inc(grad);
            n:= n>>2;
        end
        else
            n:= n>>1;
    end;
    determinaGrad2:=grad;
end;

function determinaGrad(n: integer ):integer;
var  grad, nrCifre,i:integer;
begin
    grad:= 0;
    nrCifre:= 0;
    nrCifre := calculReprezBinar2(n);
    for  i := nrCifre downto 3 do
        if ((BINAR[i]=1)  and (BINAR[i-1]=0)  and (BINAR[i-2]=1)) then
            inc(grad);

    determinaGrad:= grad;
end;

begin
    numar := 1;
    // se citeste gradul cautat
    readln(k);
    // varainta 1 fara siruri
    // se citesc elementele sirului, citirea se termina la introducerea
numarului 0
    while(numar>0)do
    begin
        readln(numar);
        if (k = determinaGrad(numar)) then
        begin
            writeln('numarul ',numar,' are gradul= ',k);
        end;
    end;
    // varianta 2 cu siruri
    i := 1;
    numar:=1;
    while (numar>0) do
    begin
        readln(numar);
        if (k <= determinaGrad2(numar)) then
        begin
            s[i] := numar;
            inc(i);
        end;
    end;
    writeln( 'numere de grad ', k,' :');
    for y := 1 to i-1 do
       write(s[y],' ');

end.
```

4. Un număr n este "zâmbăreț" dacă duce la 1 după o secvență de pași unde în fiecare pas numărul este înlocuit cu suma pătratelor cifrelor ce formează numărul. Sa se scrie un program care citeste mai multe numere pana la citirea numarului 0 si determină câte numere "zâmbărețe" s-au citit.

Exemple

numarul 19 este "zâmbăreț"
pas_1: 1+9^2=82
pas_2: 64+4=68
pas_3: 36+64=100
pas_4: 1+0+0=1

Rezolvare C++

```
/*
Problema a fost compilata cu Microsoft Visual Studio varianta Community 2017 si Apple
clang 12.0.0
*/
#include <iostream>
using namespace std;

/*
Descriere:    calcul suma patrate cifre numar
Date:         n - numar natural
Rezultat:     suma patrate numere
 */
int sumPatrate(int n)
{
	int suma = 0;
	while (n)
	{
		suma += (n % 10) * (n % 10);
		n /= 10;
	}
	return suma;
}

/*
Descriere:    determin daca un numar este zambaret
Date:         n - numar natural
Rezultat:     True - numarul e zambaret
              False - numarul nu este zambaret
*/
bool eNumarZambaret(int n)
{
	int nr1, nr2;
	// initializare numere cu n
	nr1 = nr2 = n;
	/*
	un numar nu este zambaret daca pe parcursul iteratiilor
	atinge aceeasi valoare succesiv
	 */
	do
	{
		nr1 = sumPatrate(nr1);
		nr2 = sumPatrate(sumPatrate(nr2));
	} while (nr1 != nr2);

	// daca ambele numere sunt 1, return true
	return (nr1 == 1);
}
```

```
int main()
{
	int contor = 0;
	int numar = 1;
	while (numar)
	{
		cin >> numar;
		if (eNumarZambaret(numar))
		{
			cout << "numarul " << numar << " este zambaret" << endl;
			contor++;
		}
	}
	cout << "Am citit " << contor << " numere zambarete" << endl;
	return 0;
}
```

Rezolvare Pascal

```
function sumPatrate(n:integer):integer;
var suma:integer;
begin
    suma := 0;
    while (n>0) do
    begin
        suma := suma+ (n mod 10) * (n mod 10);
        n :=n div 10;
    end;
    sumPatrate :=suma;
end;
function eNumarZambaret(n:integer):boolean;
 var   nr1, nr2:integer;
 begin
    // initializare numere cu n
    nr1 := n;
    nr2 := n;

    // un numar nu este zambaret daca pe parcursul iteratiilor
    // atinge aceeasi valoare succesiv

    repeat
        nr1 := sumPatrate(nr1);
        nr2 := sumPatrate(sumPatrate(nr2));
    until (nr1 = nr2);

    // daca ambele numere sunt 1, return true
    eNumarZambaret:= (nr1 = 1);
end;
var contor, numar:integer;
begin
    contor := 0;
    numar := 1;
    while (numar>0) do
    begin
        readln(numar);
        if (eNumarZambaret(numar)) then
        begin
            writeln(numar, ' este zambaret');
            inc(contor);
        end;
```

```
    end;
    writeln( 'Am citit ',contor,' numere zambarete');
end.
```

Exemple:

Date de intrare	Rezultat
49 19 100 11 0	23 este zambaret 49 este zambaret 19 este zambaret 100 este zambaret Am citit 4 numere zambarete