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

14 noiembrie 2020

Problema 1

Enunț

Să se realizeze câte o funcție recursivă cu un singur parametru (numărul *n*) pentru:
a) determinarea cifrei minime a lui *n*, se va returna cifra minimă;
b) determinarea cifrei pare maxime a lui n, se va returna cifra pară maximă sau -1, dacă *n* nu are cifre pare;

Analiză

Vom scrie formulele recursive pentru cele două cerințe.

a) $cifraMinima(n) = \begin{cases} n, dacă\ n\ este\ format\ dintr-o\ singură\ cifră \\ \min\{n\ mod\ 10,\ cifraMinima\left(\left[\frac{n}{10}\right]\right), \quad altfel \end{cases}$

O altă variantă recursivă pentru determinarea cifrei minime este:

$$cifraMinima(\overline{a_1a_2\dots a_{n-1}a_n}) = \begin{cases} n, dacă\ n\ este\ format\ dintr-o\ singură\ cifră \\ cifraMinima(\overline{a_1a_2\dots \min\{a_{n-1}, a_n\}}), \quad altfel \end{cases}$$

În această variantă apelul recursiv se face pentru numărul obținut prin înlocuirea ultimelor două cifre ale sale cu minimul dintre acestea.

b)

$$cifraParaMaxima(n) = \begin{cases} n, dacă\ e\ este\ format\ dintr-o\ singură\ cifră\ pară \\ -1, dacă\ n\ este\ format\ dintr-o\ singură\ cifră\ impară \\ cifraParaMaxima\left(\left[\frac{n}{10}\right]\right), \quad daca\ n\ mod\ 10\ este\ impara \\ \max\{n\ mod\ 10, cifraParaMaxima\left(\left[\frac{n}{10}\right]\right\}, daca\ n\ mod\ 10\ este\ para \end{cases}$$

Specificarea funcțiilor

Funcția **cifraMinima(n)**:

Descriere: Returneaza cifra minima a unui numar dat.

Date: n – numar natural.

Rezultate: cifra minima a numarului dat.

Funcția **cifraParaMaxima(n)**:

Descriere: Returneaza cifra para maxima a unui numar dat.

Date: n – numar natural.

Rezultate: cifra para maxima a numarului dat sau -1 daca numarul are doar cifre impare.

Implementare

Varianta C++

```
#include <iostream>

using namespace std;

/*
Descriere:  Returneaza cifra minima a unui numar dat.
Date: n – numar natural.
Rezultate: cifra minima a numarului dat.
*/

int cifraMinima_V1(int n)
{
	if (n <= 9)   // cazul in care numarul este format dintr-o singura cifra
		return n;
	int cifraMinima_restul_numarului = cifraMinima_V1(n / 10);
	return (n % 10 < cifraMinima_restul_numarului ? n % 10 :
cifraMinima_restul_numarului);
}

/*
Descriere:  Returneaza cifra minima a unui numar dat.
Date: n – numar natural.
Rezultate: cifra minima a numarului dat.
*/

int cifraMinima_V2(int n)
{
	if (n <= 9)   // cazul in care numarul este format dintr-o singura cifra
		return n;
	int minim_ultimele_doua_cifre = n % 10;
	int penultima_cifra = (n / 10) % 10;
	if (penultima_cifra < minim_ultimele_doua_cifre)
		minim_ultimele_doua_cifre = penultima_cifra;
	return cifraMinima_V2((n/100) * 10 + minim_ultimele_doua_cifre);
}
```

```
/*
Descriere:  Returneaza cifra para maxima a unui numar dat.
Date : n – numar natural.
Rezultate : cifra para maxima a numarului dat.
*/

int cifraParaMaxima(int n)
{
	if (n <= 9)		// daca n este format dintr-o singura cifra
	{
		if (n % 2 == 0)			// daca este par
			return n;
		else
			return -1;
	}

	int ultima_cifra = n % 10;
	if (ultima_cifra % 2 != 0)
		return cifraParaMaxima(n / 10);
	int cifraParaMaxima_restul_numarului = cifraParaMaxima(n / 10);
	return (ultima_cifra > cifraParaMaxima_restul_numarului ? ultima_cifra :
cifraParaMaxima_restul_numarului);
}



int main()
{
	int n = 0;
	cout << "Introduceti numarul: ";
	cin >> n;

	cout << "Cifra minima a numarului " << n << " este: " << cifraMinima_V1(n) <<
endl;
	cout << "Cifra minima a numarului " << n << " este: " << cifraMinima_V2(n) <<
endl;
	cout << "Cifra para maxima a numarului " << n << " este: " << cifraParaMaxima(n)
<< endl;

	return 0;
}
```

Varianta Pascal

```
{
Descriere:  Returneaza cifra minima a unui numar dat.
Date: n - numar natural.
Rezultate: cifra minima a numarului dat.
}
function cifraMinima_V1(n: longint): integer;
var cifraMinima_restul_numarului: integer;
begin
      if (n <= 9) then // cazul in care numarul este format dintr-o singura
cifra
            cifraMinima_V1 := n
    else
    begin
        cifraMinima_restul_numarului := cifraMinima_V1(n div 10);
        if ((n mod 10) < cifraMinima_restul_numarului) then
            cifraMinima_V1 := n mod 10
        else
            cifraMinima_V1 := cifraMinima_restul_numarului;
    end;
end;

function cifraMinima_V2(n: longint): integer;
var ultima_cifra, penultima_cifra: integer;
begin
    if(n > 9) then
    begin
        ultima_cifra := n mod 10;
        penultima_cifra := (n div 10) mod 10;
        if (penultima_cifra < ultima_cifra)
            then cifraMinima_V2 := cifraMinima_V2(n div 10)
            else cifraMinima_V2 := cifraMinima_V2(((n div 100) * 10) +
ultima_cifra);
    end
    else cifraMinima_V2 := n;
end;



{
Descriere:  Returneaza cifra para maxima a unui numar dat.
Date : n - numar natural.
Rezultate : cifra para maxima a numarului dat.
}
function cifraParaMaxima(n: longint): integer;
var ultima_cifra, cifraParaMaxima_restul_numarului: integer;
begin
    if (n <= 9)    then  {daca n este format dintr-o singura cifra}
      begin
    if (n mod 2 = 0) then           {daca este par}
                cifraParaMaxima := n
          else
                cifraParaMaxima := -1;
      end
    else
    begin
```

```
        ultima_cifra := n mod 10;
        if (ultima_cifra mod 2 <> 0) then
            cifraParaMaxima := cifraParaMaxima(n div 10)
        else
        begin
            cifraParaMaxima_restul_numarului := cifraParaMaxima(n div 10);
            if (ultima_cifra > cifraParaMaxima_restul_numarului) then
                cifraParaMaxima := ultima_cifra
            else
                 cifraParaMaxima := cifraParaMaxima_restul_numarului;
        end;
    end;
end;

var n: longint;
begin

    write('Introduceti numarul: ');
    readln(n);
    writeln('Cifra minima este: ', cifraMinima_V1(n));
    writeln('Cifra minima este: ', cifraMinima_V2(n));
    writeln('Cifra para maxima este: ', cifraParaMaxima(n));

end.
```

Problema 2

Enunț

Scrieti un subprogram recursiv care determina numarul de aparitii ale unei cifre in reprezentarea zecimala a numarului natural n.

Analiză

Formula recursiva pentru aceasta problema este:

$$aparitii(n,c) = \begin{cases} 1, daca\ n < 10\ si\ n = c \\ 0, daca\ n < 10\ si\ n \neq c \\ 1 + aparitii\left(\frac{n}{10}, c\right), daca\ n > c\ si\ n\ mod\ 10 = c \\ aparitii\left(\frac{n}{10}, c\right), daca\ n > c\ si\ n\ mod\ 10 \neq c \end{cases}$$

Specificarea subalgoritmului

Funcția **aparitii(n,c)**:

Descriere: Returneaza numarul aparitiilor cifrei c in numarul n

Date: n – numar natural, 0<=c<=9, c- numar natural.

Rezultate: numarul de aparitii ale cifrei c in numarul n.

Implementare

Varianta C++

```cpp
#include <iostream>
using namespace std;
/*
Descriere:  Returneaza numarul de aparitii ale cifrei c in numarul n.
Date : n – numar natural, c umar natural 0<=c<=9.
Rezultate : numarul de aparitii ale cifrei c in numarul n.
*/


int aparitii(int n, int c)
{
	// daca n are o singura cifra returnez 1 daca cifra = c, 0 altfel
	if (n < 10) return n == c;
	//daca ultima cifra e c o adauga la numaratoare
	else if (n % 10 == c) return 1 + aparitii(n / 10, c);
	else return aparitii(n / 10, c);
}


int main() {
	int n, c;
	cout << "Introduceti numarul: ";
	cin >> n;
	cout << "Introduceti cifra: ";
	cin >> c;
	cout << aparitii(n, c);
	return 0;
}
```

Varianta Pascal

```pascal
program aparitii;
{Descriere:
 functia determina numarul de aparitii ale cifrei c in numarul n
}
function aparitii(n:integer; c:integer):integer;
begin
if (n<10) then                {daca numarul are o singura cifra}
   if n=c then aparitii:=1    {si cifra e egala cu c, marcam o aparitie}
     else aparitii:=0         {altfel, incepem numararea aparitiilor de la 0}
else
  {daca ultima cifra a numarului e chiar c, incrementam numarul aparitiilor}
  if n mod 10=c then aparitii:=1+aparitii(n div 10, c)
  {daca ultima cifra a lui n e diferita de c, continui procesul recursiv}
    else aparitii:=aparitii(n div 10,c);
end;
```

```
var n,c:integer;
{program principal}
begin
write('Introduceti numarul n:');
 readln(n);
 write('Introduceti cifra c:');
 readln(c);
 writeln(aparitii(n,c));
end.
```

Problema 3

Enunț

În numerologie***, numărul destinului*** e folosit pentru caracterizarea unei persoane, a calităților și defectelor sale. Numărul destinului se calculează pornind de la data nașterii unei persoane, mai exact este ***suma redusă*** a tuturor cifrelor din data nașterii. Suma redusă se calculează astfel: se scrie data nașterii sub formă numerică (inclusiv luna, care va fi un număr cuprins între 1 și 12), apoi se adună toate cifrele. Se adună, apoi, toate cifrele din care este compusă această sumă. Dacă noua sumă este un număr compus din două cifre, se repetă operația, până când se obține un număr compus dintr-o singură cifră sau unul dintre numerele 11 și 22 (numere superioare).

Scrieți un subalgoritm care calculează numărul destinului pornind de la o dată calendaristică.

Exemplul 1: Data nașterii 28 noiembrie 1998 se exprimă numeric astfel: 28.11.1998.
Se adună toate cifrele: (2+8)+(1+1)+(1+9+9+8) = 10+2+27 = 39. Suma obținută este compusă din două cifre (3 și 9) și nu este un număr superior (11 sau 22). Se adună cifrele care compun suma obținută anterior: 3+9 = 12. Noua sumă este compusă tot din două cifre (1 și 2) și, din nou, nu este un număr superior (11 sau 22), așa că operațiunea de adunare trebuie continuată. Se adună cifrele care compun suma obținută anterior: 1+2 = 3. Aceasta este, în sfârșit, suma redusă a datei de 28 noiembrie 1998 și reprezintă numărul destinului pentru această dată de naștere.

Exemplul 2: Data nașterii 29 iulie 1982 (numeric 29.07.1982) are cifra destinului 11.
(2+9)+(0+7)+(1+9+8+2)=38. (3+8)=11 (număr superior)

Specificarea subalgoritmilor

Funcția **sumaCifre (x)**:

Descriere: Calculează suma cifrelor numarului natural x

Date: x – numar natural

Rezultate: suma cifrelor lui x.

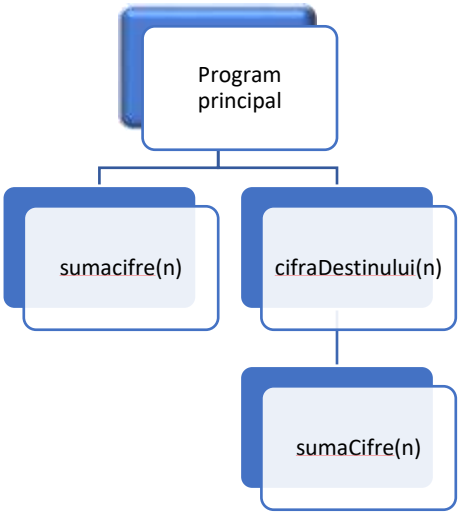
Funcția **numarulDestinului (n)**:

Descriere: Calculează numarul destinului pentru numarul n

Date: a – numar natural

Rezultate: numarul destinului asociat numarului n

Proiectare



Implementare

Varianta C++

```
/*
Descriere: Calculează suma cifrelor numarului natural x
Date: x – numar natural
Rezultate: suma cifrelor lui x.
*/
int sumaCifre(int nr){
   int s=0;
   while (nr>0){
       s+=nr % 10;
       nr/=10;
   };
   return s;
}
```

Varianta iterativă

```
/*
Descriere: Calculează numarul destinului pentru numarul nr
Date: nr – numar natural
Rezultate: numarul destinului asociat numarului nr
*/
int numarulDestinuluiIterativ(int nr) {
	while (nr >= 10 && nr != 11 && nr != 22) {
		nr = sumaCifre(nr);
	}
	return nr;
}
```

Varianta recursivă

```
/*
Descriere: Calculează numarul destinului pentru numarul nr
Date: nr – numar natural
Rezultate: numarul destinului asociat numarului nr
*/
int numarulDestinuluiRecursiv(int nr) {
	if (nr < 10 || nr == 11 || nr == 22) {
		return nr;
	}
	return numarulDestinuluiRecursiv(sumaCifre(nr));
}

int main(){
   int zi, luna, an;
   cout<<"Dati data in format zi luna an ";
   cin>>zi>>luna>>an;
   int nr = sumaCifre(zi) + sumaCifre(luna) + sumaCifre(an);
   cout<<“Numarul destinului este “ << numarulDestinuluiIterativ (nr)<<endl;
   cout<<“Numarul destinului este “ << numarulDestinuluiRecursiv (nr)<<endl;
   return 0;
}
```

Varianta Pascal

```
/*
Descriere: Calculează suma cifrelor numarului natural x
Date: x – numar natural
Rezultate: suma cifrelor lui x.
*/
function sumaCifre(nr: integer): integer;
var s:integer;
begin
   s := 0;
   while (nr>0) do
   begin
       s := s + nr mod 10;
       nr := nr div 10;
   end;
   sumaCifre:=s;
end;
```

Varianta iterativă

```
/*
Descriere: Calculează numarul destinului pentru numarul nr
Date: nr - numar natural
Rezultate: numarul destinului asociat numarului nr
*/
function numarulDestinuluiIterativ (nr: integer): integer;
begin
   While ( (nr >= 10) and (nr<>11) and (nr<>22)) do
   begin
         nr := sumaCifre(nr);
   end;
   numarulDestinuluiIterativ := nr;
end;
```

Varianta recursivă

```
/*
Descriere: Calculează numarul destinului pentru numarul nr
Date: nr - numar natural
Rezultate: numarul destinului asociat numarului nr
*/
function numarulDestinuluiRecursiv (nr: integer): integer;
begin
   if ( (nr < 10) or (nr=11) or (nr=22)) then
      numarulDestinuluiRecursiv := nr
   else
      numarulDestinuluiRecursiv := numarulDestinuluiRecursiv(sumaCifre(nr));

end;


var zi, luna, an, nr : integer;
begin
    write('Dati data in format zi luna an ');
    readln(zi);
    readln(luna);
    readln(an);
    nr := sumaCifre(zi) + sumaCifre(luna) + sumaCifre(an);
    writeln('Numarul destinului recursiv este ',
numarulDestinuluiRecursiv(nr));
    writeln('Numarul destinului iterativ este ',
numarulDestinuluiIterativ(nr));

end.
```

Probleme tip grilă

1. Ce face secvenţa de instrucţiuni pentru n=5 și şirul de numere: 100, 213, 3, 112, 210.

citeşte n {n natural} **k←0** **┌ pentru i ← 1;n execută** **│ citeşte x** **│ s←0** **│ ┌cât timp x≠0 executã** **│ │ s←s+x%10** **│ │ x←x/10** **│ └■** **│ ┌dacă s=i atunci** **│ │ k←k+1** **│ └■** **scrie k**	a) Afişează 1. b) Afişează 2. c) Afişează 3. d) Afişează 4. e) Afişează câte numere sunt egale cu numărul numărul lor de ordine de la citire. f) Afişează câte numere au suma cifrelor egală cu numărul lor de ordine de la citire.

2. Se considera subalgoritmul transformare(n), unde n este un număr natural.

Subalgoritmul transformare(n): **x ← 0;** **cattimp (n > 0) executa** **x ← x * 100 + n % 100;** **n ← n/100** **sfCattimp** **cattimp (x > 0) executa** **n ← n * 10 + x % 10;** **x ← x/10** **sfCattimp** **returneaza n;** **SfSubalgoritm**	Care din următoarele propoziţii sunt adevărate? a) transformare(123456) va returna valoarea 214365. b) transformare(2244) va returna valoarea 2244. c) Pentru un ***n*** cu număr par de cifre se vor inversa primele două cifre din ***n***, apoi următoarele două cifre din ***n***, etc. d) Se va returna oglinditul numărului.

3. Care din următoarele operații au ca rezultat valoarea True știind că variabilele întregi a și b au valorile a = 23 și b = 50
   a) (a != b) and (a > b)
   b) ((a + 10) < b) or false
   c) True and (a != b)
   d) b mod 10 > a div 7
   e) not false and (a div 10 < b)
   f) not (true or (a + b < 10)

4. Se considerî subalgoritmul calcul(n), unde n este un număr natural oarecare, iar m este un număr natural între 2 și 9.

Subalgoritmul calcul (n, m):
  x ← 0, y ← 1;
  cattimp (n!=0) executa
  x ← x + n % m * y;
  y ← y * 10;
  n ← n / m;
  sfCattimp
  returneaza x
SfSubalgoritm
	Să se determine rezultatul returnat de executarea subalgorimtului:
	a) Numărul de cifre al numărului n care sunt mai mici decât m
	b) Suma cifrelor lui n cu proprietatea că sunt mai mici decât m
	c) Reprezentarea numărului m în baza n
	d) Reprezentarea numărului n în baza m

5. Care din operațiile următoare atribuite variabilei întregi x una din cifrele sale, știind că x > 10000.
   a) x ← x mod 100
   b) x ← x mod 10
   c) x ← x div 10 mod 10
   d) x ← x div 100 mod 10
   e) x ← x mod 10 div 1
   f) x ← x mod 50