Coduri. Sisteme de numeratie. Pseudocod. Algoritmi (Codes. Numeration systems. Pseudo-code. Algorithms)

Obiective:

- Câştigarea abilităţii de a face conversii între diferitele baze de numeraţie
- Scrierea de algoritmi simpli în pseudocod
- Dezvoltarea gândirii algoritmice

Objectives:

- To learn about base conversions
- To develop simple algorithms in pseudo-code.
- To develop an algorithmically thinking mode

Rezumat:

Principalele sisteme de numeraţie implicate în domeniul calculatoarelor sunt:

- *binar* cu simbolurile *0, 1;*
- *octal* cu simbolurile *0, 1, ..., 7;*
- *zecimal* cu simbolurile *0, 1, ..., 9;*
- *hexazecimal* cu simbolurile *0, 1, ..., 9, A, B, C, D, E, F.*

Într-un sistem de numeraţie în baza *b*, un număr *N* format din parte întreagă şi fracţionară se poate reprezenta într-una din formele:

- sub forma de şir:

$$a_n a_{n-1} a_{n-2} \ldots a_1 a_0, a_{-1} a_{-2} \ldots a_{-m} \quad (1)$$

- sub forma unui polinom :

$$a_n b^n + a_{n-1} b^{n-1} + \ldots + a_1 b^1 + a_0 b^0 + a_{-1} b^{-1} + a_{-2} b^{-2} + \ldots + a_{-m} b^{-m} \quad (2)$$

- sub forma unei sume:

$$\sum_{i=-m}^{n} a_i b^i \quad (3)$$

- *b* reprezintă baza de numeraţie;
- a_i cifrele;
- *n (+1)* numărul de cifre al părţii întregi;
- *m* numărul de cifre al părţii fracţionare;
- a_n cifra cea mai semnificativă;
- a_m cifra cea mai puţin semnificativă.

Exemplu: $35_{10} = 100011_2 = 43_8 = 23_{16}$

$18_{10} = 10010_2 = 22_8 = 12_{16}$

Conversia bazei

Prin conversie înţelegem transcrierea unui număr dintr-o bază în alta. Există două metode mai frecvent utilizate în procesul de conversie:

- metoda împărţirii/înmulţirii bazei;
- metoda substituţiei.

Conversia numerelor întregi dintr-o baza b în baza 10

Definim în acest caz partea întreagă prin:

$$I = a_n b^n + a_{n-1} b^{n-1} + \ldots + a_1 b^1 + a_0 b^0 \qquad (4)$$

Fie cazul în care *b=2*, adică conversia din baza 2 în baza 10 şi în acest caz aplicând regula de mai sus (substituţiei) utilizând relaţia (4) avem:

$$11011_2 = 1*2^4 + 1*2^3 + 0*2^2 + 1*2^1 + 1*2^0 = 16 + 8 + 2 + 1 = 27$$

Pentru *b=8* avem:

$$533_8 = 5*8^2 + 3*8^1 + 3*8^0 = 347$$

Pentru *b=16* avem:

$$AC3_{16} = A_{16}*16^2 + C_{16}*16^1 + 3*16^0 = 10*256 + 12*16 + 3 = 2755$$

Conversia numerelor întregi din baza 10 în altă bază b

În acest caz vom da un procedeu simplu pornind de la relaţia (4) care se bazează pe împărţirea acestei relaţii de definiţie cu baza *b*, aşa numita *regulă de împărţire a bazei*.

$$\frac{I}{b} = C_1 + \frac{R_1}{b} = (a_n b^{n-1} + a_{n-1} b^{n-2} + \ldots + a_1 b^0) + a_0/b, \qquad (5)$$

unde C_1 şi R_1 sunt câtul respectiv restul.

Termenii din paranteză fiind întregi, ne vor da valoarea C_1 şi deci :

$$R_1 = a_0. \qquad (6)$$

Continuând cu acelaşi procedeu vom avea:

$$\frac{C_1}{b} = C_2 + \frac{R_2}{b} = (a_n b^{n-2} + a_{n-1} b^{n-3} + \ldots + a_2 b^0) + a_1/b\ , \qquad (7)$$

şi deci: $R_2 = a_1$. (8)

Procesul se continuă până când câtul este nul. Împărţirile se fac în baza 10.

Exemplu:

Convertiţi numărul 27 în baza 2.

27	13	6	3	1	0
1	1	0	1	1	-
a_0	a_1	a_2	a_3	a_4	

Citirea se va face de la coadă spre început: $27 = a_4a_3a_2a_1 = 110011_2$

Conversia numerelor reale

Deci în acest caz avem reprezentarea numerelor întregi sub forma de şir, polinom sau sumă, dată de relaţiile 1, 2, 3.

a) Partea întreagă se converteşte conform regulilor specificate anterior în subcapitolul precedent.

b) Dacă considerăm partea fracţionară reprezentată sub forma polinomială:

$$F = a_{-1}b^{-1} + a_{-2}b^{-2} + \ldots + a_{-m}b^{-m} \quad (9)$$

Atunci:

- un număr fracţionar se converteşte din baza *b* în baza 10 folosind regula substituţiei pe baza acestei relaţii.

Exemple:

$0.1101_2 = 1*2^{-1} + 1*2^{-2} + 0*2^{-3} + 1*2^{-4} = ½ + ¼ + 1/16 = 0.8125$
$0.35_8 = 3*8^{-1} + 5*8^{-2} = 29/64 = 0.453125$
$0.2C_{16} = 2*16^{-1} + C_{16}*16^{-2} = 44/256 = 0.171875$

- un număr fracţionar din baza 10 se converteşte în baza *b* folosind *regula înmulţirii bazei* pe baza următorului algoritm:

$$F = a_{-1} + (a_{-2}b^{-1} + a_{-3}b^{-2} + \ldots + a_{-m}b^{-m+1}) = a_{-1} + F_1 = a_{-1,}a_{-2}\ldots a_{-m} \quad (10)$$

şi deci

$a_{-1} = \lceil F*b \rceil$, adică partea întreagă din $F*b$. (11)

Asemănător avem:

$$a_{-2} = \lceil F_1*b \rceil, \quad (12)$$

şi procesul se încheie dacă există un *i* astfel încât produsul F_i*b să fie întreg. Dacă nu există se obţine o fracţie periodică.

Exemple:

Convertiţi numărul N = 0.8125 din baza 10 în baza 2.

	0	8125	*2
a_{-1}	1	6250	
a_{-2}	1	250	
a_{-3}	0	50	
a_{-4}	1	0	

Deci rezultatul este: 0.1101_2

Convertiţi numărul N = 0.453125 din baza 10 în baza 8.

	0	453125	*8
a_{-1}	3	625	
a_{-2}	5	0	

Deci rezultatul este 0.35_8

Observatie:
Numerele reale negative se pot reprezinta în calculator în codul "Complement faţă de doi", prescurtat C2 (această reprezentare o numim "în virgulă fixă") sau "în virgulă mobilă", care imită notaţia ştiinţifică, cu exponent şi mantisă.
La reprezentarea in virgula fixa rămân valabile cele două metode de conversie prezentate mai înainte, singura diferenţă fiind că pentru conversia numerelor reale negative se porneşte de la formula de calcul a valorii numărului reprezentat în C2 asemănătoare cu (2) si prezentata mai jos (2b):

$$R = -a_n b^n + a_{n-1}b^{n-1} + \ldots + a_1 b^1 + a_0 b^0 + a_{-1}b^{-1} + a_{-2}b^{-2} + \ldots + a_{-m}b^{-m} \qquad (2b)$$

Operaţii aritmetice în binar, octal, hexazecimal

Operaţiile aritmetice în alte baze se fac asemănator cu cele din zecimal ţinându-se însă cont de simbolurile şi caracteristicile fiecărei baze în parte.

În cadrul sistemelor de calcul de obicei se preferă tabelarea să zicem a operaţiilor de adunare şi înmulţire pentru bazele:

- binar;
- octal;
- zecimal;
- hexazecimal.

Se poate considera şi un algoritm care se bazează pe următorii paşi:

- Numerele care participă la operaţie se convertesc în baza zece.
- Se efectuează calculele în baza zece.
- Rezultatul se converteşte în baza *b* cerută.

Există şi alte procedee de conversie pentru trecerea din două baze *b* şi *B* diferite de 10. În cadrul domeniului calculatoarelor cel mai adesea se fac conversii pornind de la baza 2 în bazele 8 respectiv 16 şi invers din bazele 16 respectiv 8 în baza 2.

Exemple:

Fie $N = 1011_2$

Conversia în baza 8 se face prin gruparea a câte 3 cifre binare pornind de la dreapta spre stânga, iar în baza 16 prin gruparea a 4 cifre, astfel:

N = 1| 011 =
= 1 3
$= 13_8$

Procesul de conversie directă din bazele 16, respectiv 8 în baza 2 se face reprezentând cifrele hexa respectiv octale pe 4, respectiv 3 biţi astfel:

$AB_{16} = 1010.1011_2$
$341_8 = 011.100.011_2$

Calculul complementului unui număr

Complementul unui număr se poate determina prin diverse metode:

- reprezentare binară şi inversarea stărilor
- sau considerând următorul procedeu care ne permite determinarea diferenţei a două numere *d* şi *s*:

$d - s = d + (n-s) - n$, unde *n* este o valoare arbitrară.

Dacă $n = b^k$, unde *k* e mare (ca şi putere a bazei de numeraţie) atunci *n-s* va fi o valoare pozitivă şi scăderea lui *s* din *d* se va reduce la eliminarea cifrei *1* situată pe primul loc al sumei $d + (n-s)$.

Exemplu:

$157 - 14 = 157 + (10^4 - 14) - 10^4 = 157 + 9986 - 10000$
$= \underline{1}0143 - 10000 = 0143 = 143$

Deci algoritmul de calcul al complementului unui număr *n* dintr-o bază oarecare *b* este următorul :

Considerăm:
b baza sistemului de numeraţie şi
n, o valoare ca şi putere a lui *b*, b^k mai mare decât un număr *s* dat,
$s = s_m s_{m-1} \ldots s_0$, unde s_i sunt cifrele reprezentării lui *s* în baza *b*

Dacă numărul de zerouri în reprezentarea lui *n* este *p*, *p* > *m* atunci *s* se reprezintă sub forma:
$s = 00\ldots0 s_m s_{m-1} \ldots s_0$,
unde numărul de zerouri din faţă este *p* – *m* iar complementul lui *s*, *n-s* se obţine scăzând cifra s_0 din *b* iar restul cifrelor s_i din *b-1*.
Exemplu:

Fie:
s = 14
n = 10000
atunci:
m = 2
p = 4
s = 0014

Pentru a calcula complementul avem:
10 – 4 = 6
9 – 1 = 8
9 – 0 = 9
9 – 0 = 9

şi deci complementul este 9986 care poate fi utilizat la calculul diferenţei 157 – 14.

Algoritmi
Un *algoritm* este o metodă de rezolvare a unui anumit tip de problemă şi desemnează o mulţime finită şi ordonată de operaţii.
Algoritmii lucrează cu *date* (constante, *variabile*, *tablouri* etc.), *operatori*, *expresii*,*structuri*, *clase etc* .
Pseudocodul este un asa zis limbaj folosit în proiectarea şi documentarea programelor.
Pseudocodul, ca orice limbaj, foloseste date, variabile, operatii si instructiuni, insa intr-un mod mai relaxat, mai apropiat de limbajul natural uman. Poate fi considerat şi ca o metodă de documentare folosită pentru a specifica logica de procesare a unui modul.
Enunţurile în pseudocod pot contine:
-*Instructiuni :* de bază - intrare, ieşire, de control – if, while etc.
-*Funcţii, Proceduri,*
-*Operaţii pe fişiere, etc.*
Un exemplu de algoritm pentru determinarea valorii Min/Max dintr-un sir de numere scris în pseudocod :

```
MinMax_Alg:
{
Input X[i], i=0,...,N-1
min = X[0]
max = X[0]
For i=1 To N-1 Do
        If X[i] < min Then
                min = X[i]
        EndIf
        If X[i] > max Then
                max = X[i]
```

```
        EndIf
EndFor
Output min, max
}//end
```

Căutarea unui număr într-un şir de numere ordonat, prin metoda înjumătăţirii intervalului

```
Algoritm de Cautare:
{
        Input X[i], I=0,…,N-1
        Input val
        început = 0
        sfârşit = N-1
        mijloc = (început + sfârşit) / 2
        While ((început <= sfârşit) AND (val <> X[mijloc])) Do
                if val < X[mijloc] then
                        sfârşit = mijloc – 1
                else
                        început = mijloc + 1
                mijloc = (început + sfârşit) / 2
        EndWhile
        If X[mijloc] = val Then
                Output mijloc
        Else
                Output "Nu am găsit valoarea dată"
        EndIf
}
```

Algoritmul lui Euclid pentru determinarea c.m.m.d.c. a două numere întregi

```
Algoritm Euclid:
{
        Input n1, n2
        If n1 > n2 Then
                deimp = n1
                imp = n2
        Else
                deimp = n2
                imp = n1
        EndIf
        rest = imp
        While rest != 0 Do
                rest = deimp modulo imp
                deimp = imp
                imp = rest
        EndWhile
        Output deimp
}
```

Ordonarea unui şir de numere întregi prin metoda selecţiei

Prin această metodă se caută poziţia celui mai mic element, după care se schimbă acesta cu elementul de pe prima poziţie. În continuare, se ia în considerare şirul format din elementele 2,3,…,N şi se caută poziţia celui mai mic element din acest şir ş.a.m.d.

```
Algoritm SortareSelectie:
{
        Input X[i], i=0,…,N-1
```

```
        For i=0 To N-1 Do
                Pozmin = i
                For j=i+1 To N-1 Do
                        If X[pozmin] > X[j] Then
                                Pozmin = j
                        EndIf
                EndFor
                Temp = X[pozmin]
                X[pozmin] = X[i]
                X[i] = Temp
        EndFor
        Output X[i], i=0,…,N-1
}
```

Summary:

The main numeration systems involved in computer science domain are:

- *Binary with 0, 1 symbols;*
- *octal* with *0, 1, …, 7 symbols;*
- *decimal with 0, 1, …, 9 symbols;*
- *hexadecimal with 0, 1… 9, A, B, C, D, E, F symbols.*

An algorithm is a method used to solve a typical problems being composed by a finite and orderly set of operations.

The main elements that an algorithm uses are *data* (constants, *variables*, *arrays*, etc.), *operators*, *expressions*, *structures*, *classes*, *etc.*

Pseudo-code is a language used to design and to document problems specifying the control stream and not the involved data. May be considered as a documentation method used to specify the processing logic of a module.

A pseudo-code may contain:

-*Instructions:* base – input/output, etc., control – if, while, etc.

-*Functions, Procedures,*

-*File Operations, etc.*

Example of an algorithm used to determine the Min/Max value from a string of numbers using a pseudo-code:

```
MinMax_Alg:
{
Input X[i], i=0,…,N-1
min = X[0]
max = X[0]
For i=1 To N-1 Do
        If X[i] < min Then
                min = X[i]
        EndIf
        If X[i] > max Then
                max = X[i]
        EndIf
EndFor
Output min, max
}//end
```

Întrebări:

1. Ce este un algoritm? De ce este necesară folosirea lui?
2. Care este algoritmul de trecere a unui număr din baza 10 în baza 2? (intreg/real)

3. Cum exprimăm în pseudocod situația în care avem un punct de decizie în program ?

Teme:

1. Convertiţi:
- în baza 8 numărul 347;
- în baza 16 numărul 2755;
- în baza 2 numărul 20.

2. Convertiţi în baza 10 numerele: 1000101_2, 357_8, $C7A_{16}$
3. Efectuaţi următoarele operaţii:
   a) $FA_{16}+23_{16}=$
   b) $1101\ 0010_2 + 1110\ 1101_2=$
   c) $1176_8-723_8=$
4. Convertiţi numerele din baza 2 în baza 8 şi apoi 16:
   a) 10101010,110010
   b) 1100010,11101
5. Convertiţi numărul 175.1285_{10} în baza 4.
6. Scrieţi pseudocodul pentru diverse metode de sortare ca, Heap (heap sort), Bulelor (buble sort), Rapidă (quick sort), Radix (radacina), etc.
7. Considerând două numere întregi determinaţi cmmdc folosind algorimul lui Euclid.
8. Descrieţi algoritmii pentru descompunerea unui număr în factori primi, determinarea tuturor numerelor prime până la un n citit etc.
9. Considerând algoritmul de înmulţire chinezească descrieţi în pseudocod paşii necesari pentru a înmulţi două înumere întregi. Ştiţi şi alţi algoritmi interesanţi de înmulţire?

Questions:

1. What do you understand by an algorithm? Why we use algorithms?
2. Which is the algorithm used to convert a value from 10 base to 2 base? (int/real)
3. How do we express in pseudo-code the situation when the program is in a decisive point?

Homework:

1. Convert:
- the number 347 in the corresponding number in base 8;
- the number 2755 in the corresponding number in base 16;
- the number 20 in the corresponding number in base 2.

2. Convert in base 10 the numbers: 1000101_2, 357_8, $C7A_{16}$
3. Calculate:
   a) $FA_{16}+23_{16}=$
   b) $1101\ 0010_2 + 1110\ 1101_2=$
   c) $1176_8-723_8=$
4. Convert the following numbers from base 2 in base 8 and then in base 16:
   a) 10101010,110010
   b) 1100010,11101
5. Convert the number 175.1285_{10} in base 4.
6. Describe in pseudo-code different sort methods as Heap (heap sort), Bubble (bubble sort), Quick (quick sort), Radix (root), etc.
7. Considering two integer numbers determine the greatest common measure using the Euclidian algorithm.

8. Describe the algorithm used to decompose a number in prime factors, determine all the prime numbers to a value n, etc.

9. Considering the chine's multiplying algorithm describe in pseudo-code the steps to multiply 2 int numbers. Do you know others interesting algorithms for multiplying?