Adrian Runceanu

Laborator nr. 3
Algoritmi elementari (pseudocod)
Structuri repetitive - Structura repetitivă cu număr cunoscut de paşi

A. Probleme rezolvate:

1) Să se determine toate tripletele de numere a, b, c cu proprietăţile: 1 < a < b < c < 100; a + b + c se divide cu 10.

Solutie:

Pas 1. Datele de intrare: nu se citește nicio valoare

Pas 2. Analiza problemei:
Intr-o prima instructiune repetitiva (instrucțiunea **pentru**), generam toate numerele a între 2 și 98.
In a doua instructiune repetitiva generam toate numerele b între a+1 și 99.
In cea de-a instructiune repetitiva generam toate numerele c între b+1 și 100.
In interiorul acestor 3(trei) instrucțiuni repetitive verificăm dacă suma celor 3 numere se divide exact cu 10 (adică dacă restul împărțirii sumei la 10 este egala cu 0 (zero).

Pas 3. Scrierea algoritmului în pseudocod:

```
întreg a, b, c
pentru a = 2, 98 execută
|   pentru b = b + 1, 99 execută
|   |   pentru a = c + 1, 100 execută
|   |   |   dacă (a + b + c) % 10 = 0 atunci
|   |   |   |   scrie a, ` ’, b, ` ’, c
|   |   |   |■
|   |   |■
|   |■
|■
stop
```

2) Să se genereze primii n termeni ai şirului 1, 1, 2, 2, 2, 3, 3, 3, 3, 4, 4, 4, 4, 4, ….

Solutie:

Pas 1. Datele de intrare: n numar intreg

Pas 2. Analiza problemei:
Vom folosi o variabila pentru o instructiune repetitiva astfel incat sa putem genera, pe rand, valorile cerute.
Mai intai pornim cu o valoare inițială sa zicem a, careia îi dăm valoarea 1. Apoi folosim alta variabila b initializata tot cu valoarea 1, si facem urmatoarea verificare:
daca a <= b+1 atunci afisam valoarea b si apoi marim cu o unitate pe a (a = a+1)
altfel
reinitializam valoarea lui a cu 2 si marim cu o unitate pe b (b=b+1) si apoi afisam valoarea lui b

Pas 3. Scrierea algoritmului în pseudocod:

```
întreg n, a, b, i
citește n
a <- 1
b <-1
pentru i = 1, n execută
|       daca a <= b + 1 atunci
|       |       scrie b, ' '
|       |       a <- a + 1
|       | altfel
|       |       a <- 2
|       |       b <- b + 1
|       |       scrie b, ' '
|       |■
|■
stop
```

3) Să se afişeze primii n termeni ai **şirului lui Fibonacc**i:
0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, ….

Solutie:

Pas 1. Datele de intrare: n numar intreg

Pas 2. Analiza problemei:
Pentru a genera elementele șirului lui Fibonacci, folosim 3(trei) variabile: x, y si z.

Inițial dam valorile 0 lui x si 1 lui y (adică exact primele doua valori din șirul lui Fibonacci). Apoi afișăm cele doua valori x si y.
Intr-o instructiune repetitiva de n - 2 pasi, facem suma celor doua valir x si y si o retinem in cea de-a treia variabila - z.
Afisam pe z.
Modificăm pe x dandu-i valoarea lui y, iar pe y dandu-i valoarea lui z.

Pas 3. Scrierea algoritmului în pseudocod:

```
întreg n, x, y, z
citește n
x <- 0
y <- 1
scrie x, ‘, ’, y
pentru i = 1, n-2 execută
|       z <- x + y
|       scrie z, ‘,’
|       x <- y
|       y <- z
|■
stop
```

4) Divizorii unui număr n. Se citeşte un număr întreg n. Să se scrie un algoritm care să care afiseze toti divizorii numărului n.

Solutie:

Pas 1. Datele de intrare: n numar intreg
Pas 2. Analiza problemei: Se foloseste o structură repetitivă cu număr cunoscut de repetiţii, în care se verifică dacă a este divizibil cu i (dacă restul împărţirii lui a la i = 0), unde i ia toate valorile de la 1 la a.
Pas 3. Scrierea algoritmului în pseudocod:

```
întreg n, i
citește n
pentru i ← 1, a execută
|       dacă (a % i = 0) atunci
|       |  scrie i, ‘ ’
|       |■
|■
stop
```

Exemplu:
Dacă se citeşte pentru n = 12 atunci algoritmul va afişa „**1 2 3 4 6 12**”
Iar în cazul în care se citeşte n = 13 se va afişa „**1 13**”.

5) Divizorii proprii ai unui număr n. Se citeşte un număr întreg n. Să se scrie un algoritm care să afişeze toti divizorii proprii ai numărului n.

Solutie:

Pas 1. Datele de intrare: n intreg

Pas 2. Analiza problemei:
Se cunoaşte faptul că divizorii proprii ai unui număr n se află în intervalul închis [2, [n/2]], unde [n/2] este partea întreagă a lui „n/2".
Se foloseşte o structură repetitivă cu număr cunoscut de repetiţii, în care se verifică dacă n este divizibil cu i (dacă restul împărţirii lui n la i = 0), unde i ia toate valorile de la 2 la [n/2].

Pas 3. Scrierea algoritmului în pseudocod:

```
întreg n, i, div
citește n
div ← 0 // variabilă folosită pentru a reţine dacă am găsit divizori
pentru i ← 2, [a/2] execută
|       dacă (a % i = 0) atunci
|       |       scrie i
|       |       div ← 1 // marchez faptul ca am găsit divizori
|       |■
|■
daca div = 0 atunci
|       scrie „nu există divizori proprii"
|■
stop
```

Exemplu:
Dacă se citeşte pentru n = 12 atunci algoritmul va afişa „**2 3 4 6**"
Iar în cazul în care se citeşte n = 13 se va afişa mesajul „**nu există divizori proprii**".

6) Primalitatea unui număr n. Se citeşte un număr întreg n. Să se scrie un algoritm care să verifice dacă numărul n este prim.

Solutie:

Pas 1. Datele de intrare: n intreg

Pas 2. Analiza problemei:
Se cunoaşte faptul că un număr este prim dacă NU are divizori proprii.

De asemenea se ştie că numărul 1 NU este prim, de aceea vom trata separat cazul a=1.
Algoritmul foloseşte o structură repetitivă cu număr cunoscut de repetiţii, în care se caută divizori proprii.
În caz că se gasesc divizori, numărul nu este prim altfel numărul este prim.
Se foloseste o variabilă semafor numită „prim" care iniţial are valoarea „1" şi se modifică în „0" doar dacă se găsesc divizori.

Pas 3. Scrierea algoritmului în pseudocod:

```
întreg n, i, prim
citește n
dacă n <= 1 atunci
|       scrie „numar NEPRIM"
| altfel
| |     prim ← 1 // presupunem ca numărul este prim
| |     pentru i ← 2, [n / 2] execută
| |     |       dacă (n % i = 0) atunci
| |     |       |       prim ← 0 // am găsit divizori deci numarul nu e prim
| |     |       |■
| |     |■
| |     dacă prim = 1 atunci
| |     |       scrie „numar PRIM"
| |     | altfel
| |     |       scrie „numar NEPRIM"
| |     |■
| |■
|■
stop
```

Exemplu:
Dacă se citeşte pentru n = 20 atunci algoritmul va afişa „**numar NEPRIM**"
Iar în cazul în care se citeşte n = 7 se va afişa mesajul „**numar PRIM**"

7) Numere perfecte. Se citeşte un număr întreg n. Să se scrie un algoritm care să să afiseze toate numerele perfecte mai mici sau egale cu n. Spunem că un număr este **perfect** dacă este **egal cu suma divizorilor săi, fără el însuşi**.

De exemplu dacă se citeşte pentru **n** valoarea 30 atunci algoritmul va afişa **6 28,** deoarece aceste două numere sunt singurele pentru care putem scrie:
6 = 1 + 2 + 3 şi
28 = 1 + 2 + 4 + 7 + 14.

Solutie:

Pas 1. Datele de intrare:

Pas 2. Analiza problemei: Algoritmul verifică fiecare număr cuprins între 1 şi n dacă este număr perfect. Această verificăre se face iniţializând suma cu 0 pentru fiecare număr şi căutând divizorii de la 1 la jumatatea numărului. Divizorii se adaugă la sumă iar la final aceasta este comparată cu numărul testat.

Pas 3. Scrierea algoritmului în pseudocod:

```
întreg n, d, s, i
citește n
pentru i ← 1, n execută
|     s ← 0 // calculam suma divizorilor
|     pentru d ← 1, i/2 execută
|     |     dacă (i % d = 0) atunci
|     |     |       s ← s + d // adaugam divizorul la suma
|     |     |■
|     |■
|      dacă (s = i) atunci
|      |     scrie i, " "
|      |■
|■
stop
```

Exemplu:
Dacă se citeşte pentru n = 10000, atunci se va afisa: 6 28 496 8128

8) Numere prietene. Se citesc două numere întregi a şi b. Să se scrie un algoritm care să care să verifice dacă cele doau numere sunt prietene. Spunem ca două numere sunt prietene dacă suma divizorilor proprii ai unui număr este egală cu celalalt şi invers.

Solutie:

Pas 1. Datele de intrare: a si b numere intregi

Pas 2. Analiza problemei: Algoritmul calculează suma divizorilor lui a în suma_a şi suma divizorilor lui b în suma_b.
Apoi verifică dacă suma_a = b şi suma_b = a. Dacă condiţia este adevarată se afişează „numere prietene” altfel se afişează „Nu sunt numere prietene”.

Pas 3. Scrierea algoritmului în pseudocod:

```
întreg a, b, suma_a, suma_b, i
citește a, b
suma_a ← 0
suma_b ← 0
pentru i ← 2, [a/2] execută
|    dacă (a % i = 0) atunci
|    |  suma_a ← suma_a + i   // suma divizorilor proprii numărului a
|    |■
|■
pentru i ← 2, [b/2] execută
|     dacă (b % i = 0) atunci
|     |  suma_b ← suma_b + i   // suma divizorilor proprii numărului b
|     |■
|■
daca suma_a = b şi suma_b = a atunci
|     scrie „numere prietene”
|altfel
|     scrie „NU sunt numere prietene”
|■
stop
```

Exemplu:
Dacă se citeşte a = 284 şi b = 220 atunci algoritmul va afişa mesajul **„numere prietene”**

9) Factorial. Se citeşte un număr întreg a. Să se scrie un algoritm care să afiseze n! Factorial de n (notat n!) este produsul numerelor de la 1 la n.

Solutie:

Pas 1. Datele de intrare: Numarul a intreg

Pas 2. Analiza problemei: Algoritmul calculează produsul numerelor de la 1 la a în variabila p. Iniţial variabila **p = 1** deoarece produsul se iniţializează cu elementul neutru de la înmultire, adică 1.

Pas 3. Scrierea algoritmului în pseudocod:

```
întreg a, i, p
citește a
p ← 1
pentru i ← 1, a execută
|     p ← p * i // calculăm produsul
|■
scrie p
stop
```

Exemplu:
Dacă se citeşte a = 4 atunci algoritmul va afişa 24, deoarece 4! = 1 * 2 * 3 * 4 = 24.

10) Sirul lui Fibonacci. Se citeşte un număr întreg n (2< n <= 20). Să se scrie un algoritm care să să afişeze al n-lea termen din şirul lui Fibonacci.

Solutie:

Pas 1. Datele de intrare: n numar natural

Pas 2. Analiza problemei: Şirul lui Fibonacci se formează după urmatoarea formulă:

$$Fibo(n) = \begin{cases} 1 & \textit{dacă} \quad n = 1 \text{ sau } n = 2 \\ Fibo(n-1) + Fibo(n-2) & \textit{dacă } n > 2 \end{cases}$$

Algoritmul calculează fiecare termen şi când ajunge la al n-lea se opreşte şi îl afişează. Se foloseşte o structură repetitivă cu număr cunoscut de pasi, unde variabila contor i ia valori de la 3 la n, deoarece primii doi termeni sunt deja calculaţi, atunci trebuie să calculăm termenii începand cu al 3-lea.

Pas 3. Scrierea algoritmului în pseudocod:

```
întreg n, f1, f2, f3, i
citește n
f1 ← 1
f2 ← 2 // inițializarea primilor termeni din șir
pentru i ← 3, n execută
|     f3 ← f2 + f1 // calculul termenului curent din șir
|     f1 ← f2
|     f2 ← f3
|■
scrie f3
stop
```

Exemplu:
Dacă se citeşte n = 8 atunci algoritmul va afişa 21, deoarece al 8-lea termen din şirul lui Fibonacci este 21**.**

B. Probleme propuse spre rezolvare:

L3.1) Să se calculeze suma numerelor naturale cuprinse între două numere date (dintr-un interval).
Exemplu:
Date de intrare: capetele intervalului 3 6
Date de ieşire: suma = 9

L3.2) Să se afişeze toţi divizorii unui număr natural dat.
Exemplu:
Date de intrare: 12
Date de ieşire: 1 2 3 4 6 12

L3.3) Se dă un număr. Să se scrie, dacă se poate, ca sumă de două numere impare.
Exemple:
Date de intrare: 24
Date de ieşire:
24 = 1 + 23
24 = 3 + 21
24 = 5 + 19
24 = 7 + 17
24 = 9 + 15
24 = 11 + 13

Date de intrare: 33
Date de ieşire: Nu se poate

L3.4) Se dă un număr. Să se scrie, dacă este posibil, ca sumă de două numere consecutive.
Exemple:
Date de intrare: 5
Date de ieşire: 5 = 2 + 3

Date de intrare: 6
Date de ieşire: Nu se poate

L3.5) Dându-se un număr natural n, să se găsească toate posibilităţile de scriere a acestui număr ca sumă de numere consecutive.
Exemplu:
Date de intrare: 15
Date de ieşire:
15 = 1 + 2 + 3 + 4 + 5
15 = 4 + 5 + 6
15 = 7 + 8

L3.6) Se dă un număr natural n. Afișați în ordine crescătoare primele n numere naturale nenule.

Date de intrare: Programul citește de la tastatură numărul n.
Date de ieşire: Programul afișează pe ecran în ordine crescătoare primele n numere naturale nenule, separate prin exact un spațiu.

Exemplu
Date de intrare
5
Date de ieșire
1 2 3 4 5

L3.7) Se dă un număr natural n. Afișați pe o linie primele n numere naturale nenule în ordine crescătoare, iar pe linia următoare aceleași numere, dar în ordine descrescătoare.

Date de intrare: Programul citește de la tastatură numărul n.
Date de ieşire: Programul afișează pe ecran, pe linii diferite numere cerute. Numerele de pe aceeași linie sunt separate prin câte un spațiu.

Exemplu
Date de intrare
5
Date de ieșire
1 2 3 4 5
5 4 3 2 1

L3.8) Se dau n numere naturale nenule. Calculaţi suma celor n numere date.

Date de intrare: Programul citește de la tastatură numărul n, iar apoi n numere naturale.
Date de ieşire: Programul afișează pe prima linie a ecranului numărul S, reprezentând suma celor n numere.

Exemplu
Intrare
5
6 2 0 4 1
Ieșire
13

L3.9) Se dă un număr întreg n și alte k numere întregi. Să se afle dacă, adunând toate cele k numere la n se obține o valoare egală cu valoarea inițială a lui n.

Date de intrare: Programul citește de la tastatură numerele n k, iar apoi k numere întregi.
Date de ieșire: Programul va afișa pe ecran textul "DA" dacă numărul final este egal cu cel inițial sau textul "NU" în caz contrar.
Exemplu
Intrare
25
3
16 -9 3
Ieșire
NU
Explicație: 25 + 16 - 9 + 3 = 35, număr diferit de cel inițial (25).

L3.10) Se dau n numere naturale. Determinaţi primul număr par dintre cele n numere.
Date de intrare: Programul citește de la tastatură numărul n, iar apoi n numere naturale, separate prin spaţii.
Date de ieşire: Programul afișează pe ecran numărul P, reprezentând primul număr par dintre cele n numere sau mesajul IMPOSIBIL, dacă printre cele n numere citite nu există numere pare.

Exemplu
Date de intrare
5
7 4 2 5 8
Date de ieșire
4

Bibliografie

[1] **http://www.pbinfo.ro**. Descrierea site-ului: www.pbinfo.ro îți propune să rezolvi probleme de informatică, cu evaluator automat. Știi pe loc dacă soluția ta este corectă sau dacă trebuie să mai lucrezi la ea.
Problemele sunt grupate după programa de informatică pentru liceu. Dar nu trebuie să fii la liceu ca să rezolvi aceste probleme. Poți fi elev de gimnaziu, student, profesor sau pur și simplu pasionat de informatică. De fapt, trebuie doar să vrei!!
[2] **https://adrian.runceanu.ro**
[3] Adrian Runceanu, „*Programarea şi utilizarea calculatoarelor*", Editura Academica Brâncuşi din Târgu-Jiu, 2003, ISBN 973-8436-44-3
[4] Adrian Runceanu, Mihaela Runceanu, „*Noțiuni de programare – limbajul C++*", Editura Academica Brâncuși din Târgu-Jiu, 2012, ISBN 978-973-144-550-2
[5] Adrian Runceanu, Mihaela Runceanu, „*Algoritmi implementati in limbajul C++. Volumul I – Algoritmi elementari*", Editura Academica Brâncuși din Târgu Jiu, 2021, ISBN 978-606-9614-06-8