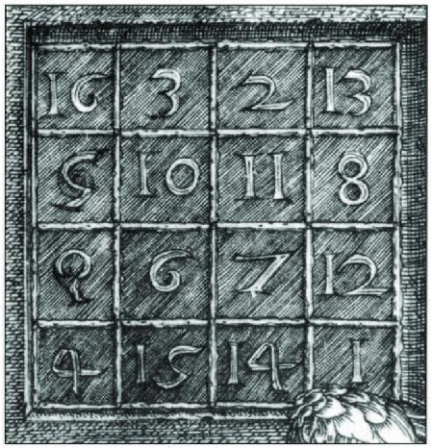
Bine ați venit la cursul
Logică și Structuri Discrete -LSD



Cursul 1

Ș.l. dr. ing. Cătălin Iapă

catalin.iapa@cs.upt.ro

Să facem cunoștiință:

Curs:

Cătălin Iapă

Laboratoare:

Adelina Stana

Alex Grosu

Raul Brumar

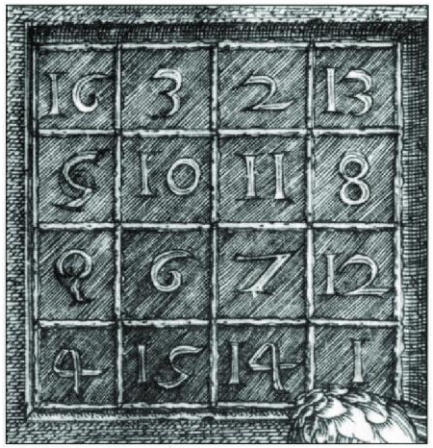
Cătălin Botean

Să facem cunoștiință

- Moldova-Nouă, Caraș-Severin
- Liceul Grigore Moisil Timișoara
- Facultatea de Automatică și Calculatoare, UPT
  - Calculatoare și Tehnologia Informației
- Doctorat
  - Identificarea utilizatorului unui calculator în funcție de modul în care scrie la tastatură
- Liga AC, Casa Tineretului, Primăria Timișora, Ministerul Dezvoltării, Autoritatea pentru Digitalizarea României
- Programarea Calculatoarelor, Tehnici de Programare, Managementul Proiectelor Software

Detalii administrative curs

- Semestrul 1: 14 săptămâni
  - 2 ore de curs/ săptămână
  - 2 ore de laborator/ săptămână
- Prezența la curs și laborator e obligatorie
  - Pentru toate prezențele la curs veți primi 1p bonus la examen
- Veți primi 2 note, media lor este nota finală la LSD
  - 1 notă la examen, în sesiune, după cele 14 săptămâni
  - 1 notă pentru activitatea de la laborator
- Pentru a promova materia e nevoie să luați cel puțin nota 5, atât la examen cât și la laborator
- Examenul se poate da de 3 ori



Ce facem la LSD

Demonstrații

Mulțimi

Funcții

Proprietăți ale funcțiilor

Funcțiile în programare

Despre LSD

- Logică și Structuri Discrete
- Ce vom face?
  - LOGICĂ,
  - MATEMATICĂ și *Ce fel de Matematică?*
  - PROGRAMARE *Ce fel de Programare?*
- De ce ai nevoie înainte să începi acest curs?
  - Noțiuni fundamentale de matematică
  - Curiozitate

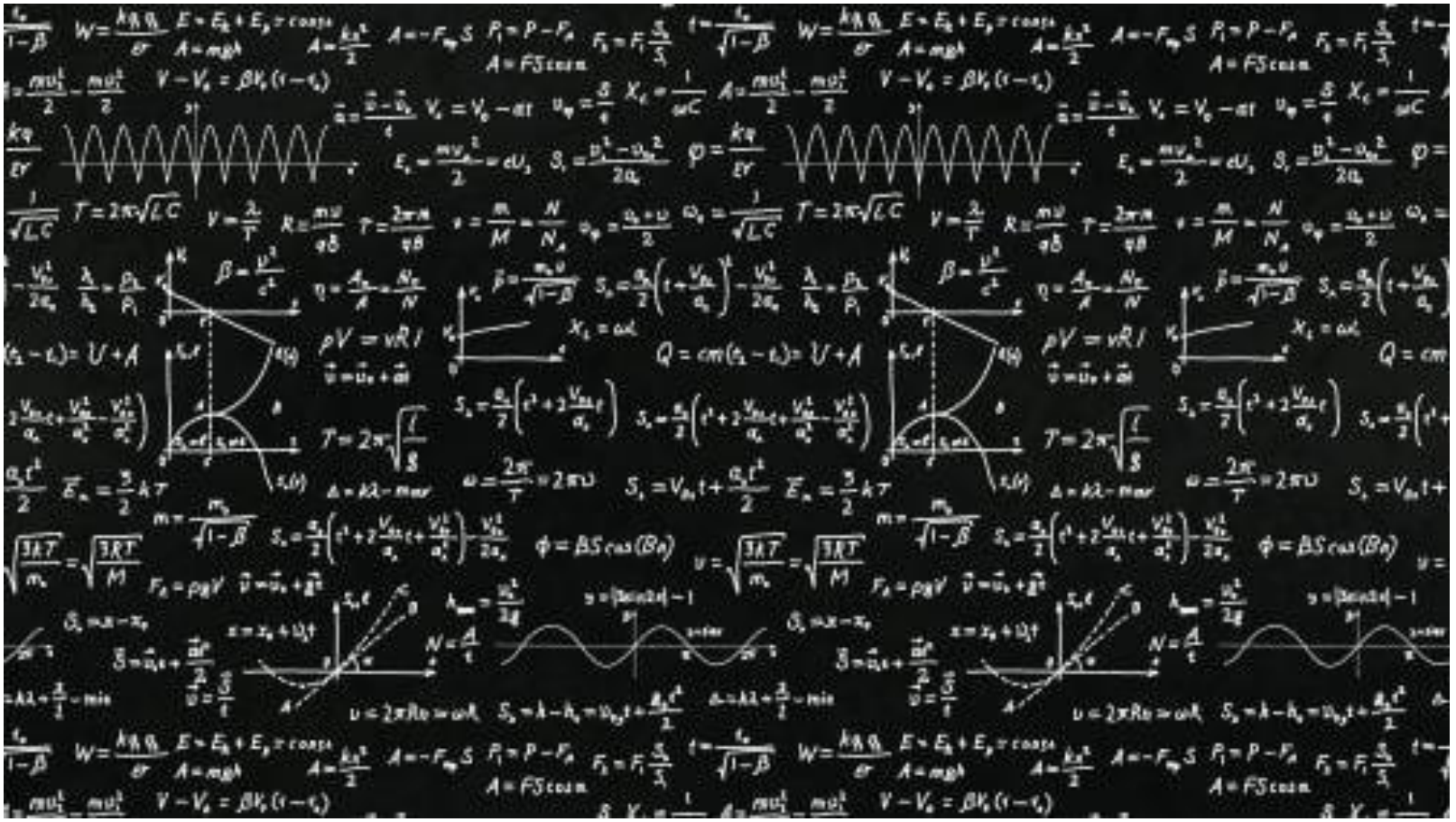
Despre LSD

LOGICĂ MATEMATICĂ

- cum exprimăm precis afirmații
  - pentru definiții riguroase, specificații în software, ...
- cum demonstrăm afirmații
  - pentru a arăta că un algoritm e corect
- cum prelucrăm formule logice
  - pentru a găsi soluții la probleme

Despre LSD

MATEMATICĂ DISCRETĂ



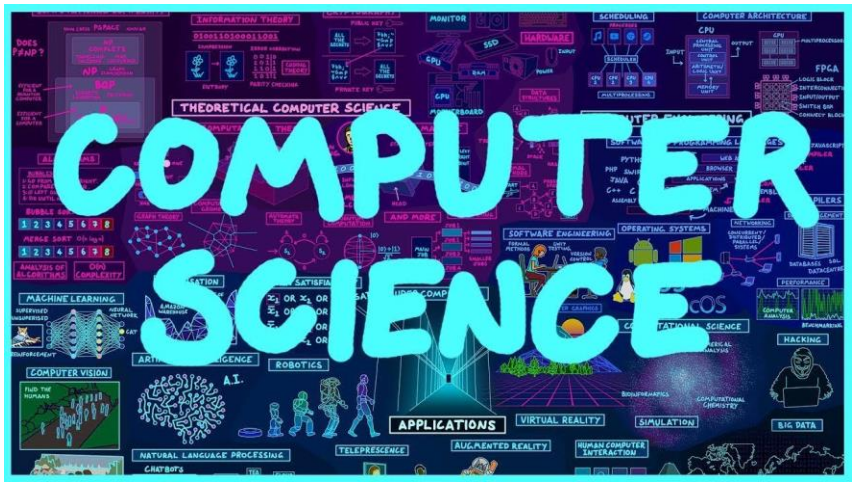
Sursă imagine: https://engineering.jhu.edu/ams/wp-content/uploads/2021/06/hero-image-research-500x282.jpeg

Despre LSD

MATEMATICĂ DISCRETĂ

Este limbajul de bază al științei calculatoarelor

- *algorithms*
- *bioinformatics*
- *computer graphics*
- *data science*
- *machine learning*
- *software engineering etc.*



Sursă imagine: https://img.youtube.com/vi/Jv34MWng28o/maxresdefault.jpg

Despre LSD

MATEMATICĂ DISCRETĂ

- Ce studiem?

studiem noțiuni/obiecte care iau valori distincte, discrete

- întregi, valori logice, relații, arbori, grafuri, etc.

- Ce nu studiem?

nu studiem domeniul continuu

- numere reale, infinitezimale, limite, ecuații diferențiale vezi: analiză matematică

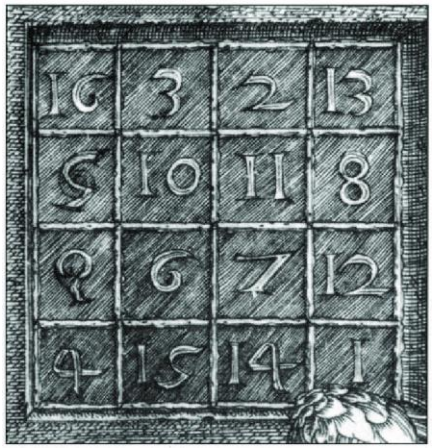
Despre LSD

PROGRAMARE în PYTHON

Vom aplica conceptele parcurse la curs pe calcultor, folosind limbajul de programare PYTHON

Limbaj la nivel înalt

- Este deosebit de ușor să începeți să utilizați Python (chiar dacă nu ați programat înainte)
- Sintaxa este prietenoasă cu cititorul (și aproape de un limbaj natural)
- Codul este compact
- Biblioteca standard Python oferă o gamă largă de facilități și multe biblioteci externe sunt, de asemenea, disponibile
- Este unul dintre cele mai utilizate limbaje de programare la ora actuală



Ce facem la LSD

Demonstrații

Mulțimi

Funcții

Proprietăți ale funcțiilor

Funcțiile în programare

Să începem

- DEMONSTRAȚII

- Ce e demonstrația?

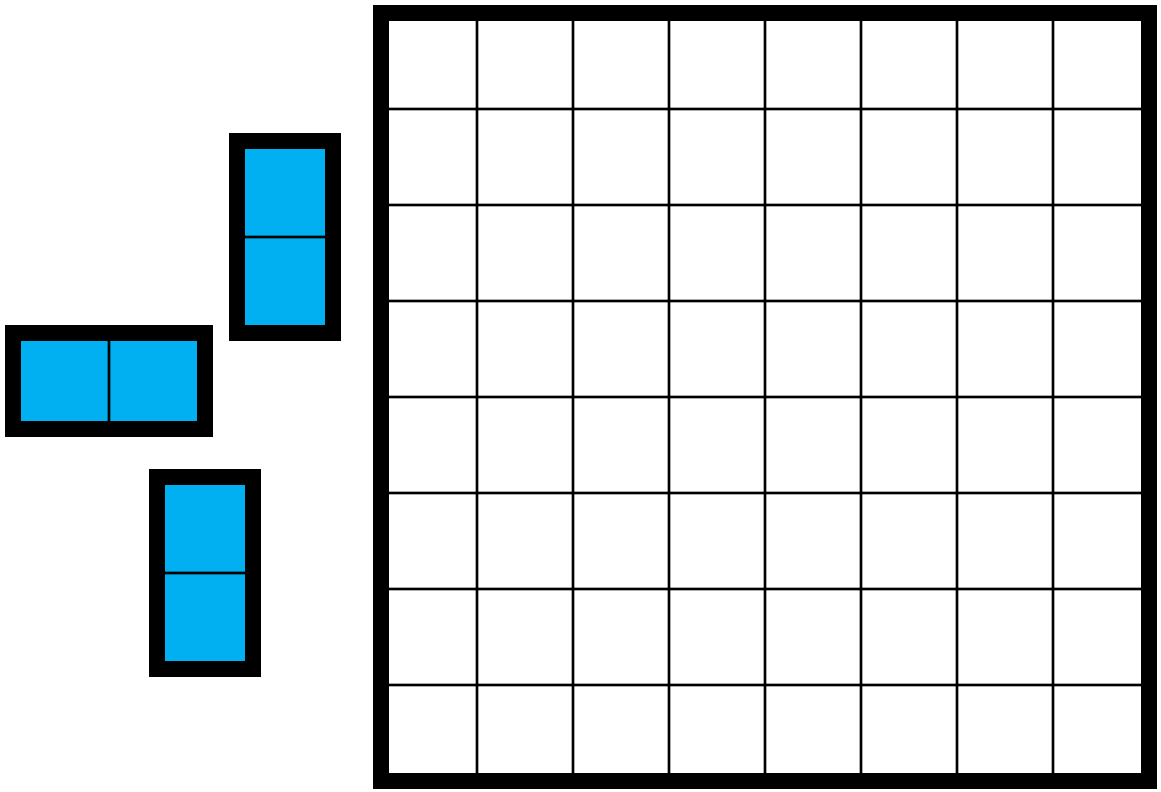
Să începem

- DEMONSTRAȚII

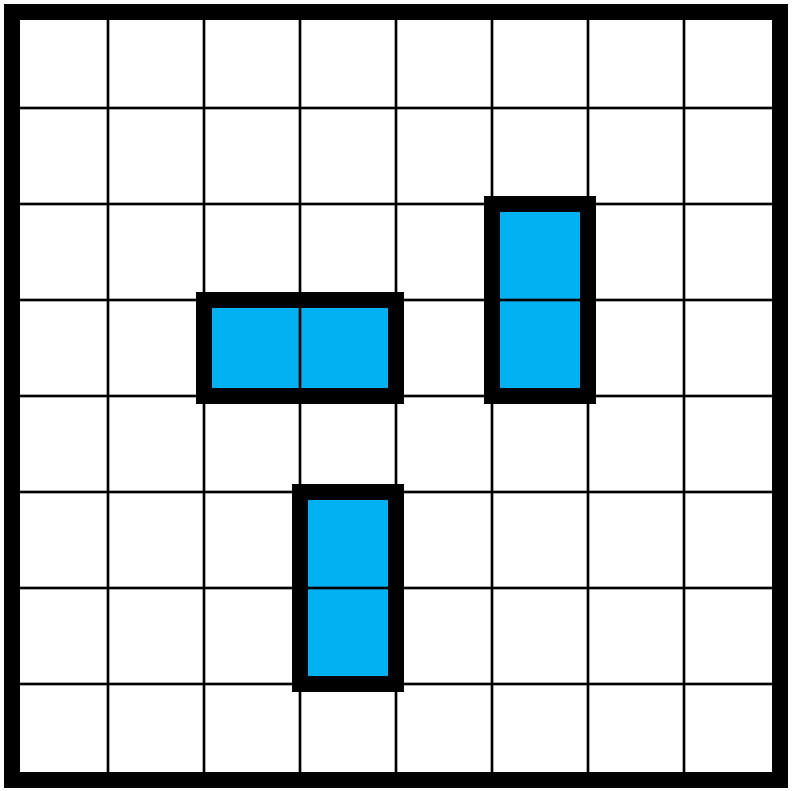
- Ce e demonstrația?
  - Un argument care este așa de convingător astfel încât poți să îl folosești pentru a-i convinge și pe ceilalți
  - Este un semn de înțelegere

DEMONSTRAȚII

Un prim exemplu simplu: Putem umple (fără a lăsa spații goale) o tablă de șah de dimensiunea 8 x 8 cu piese de domino de dimensiunea 1 x 2?

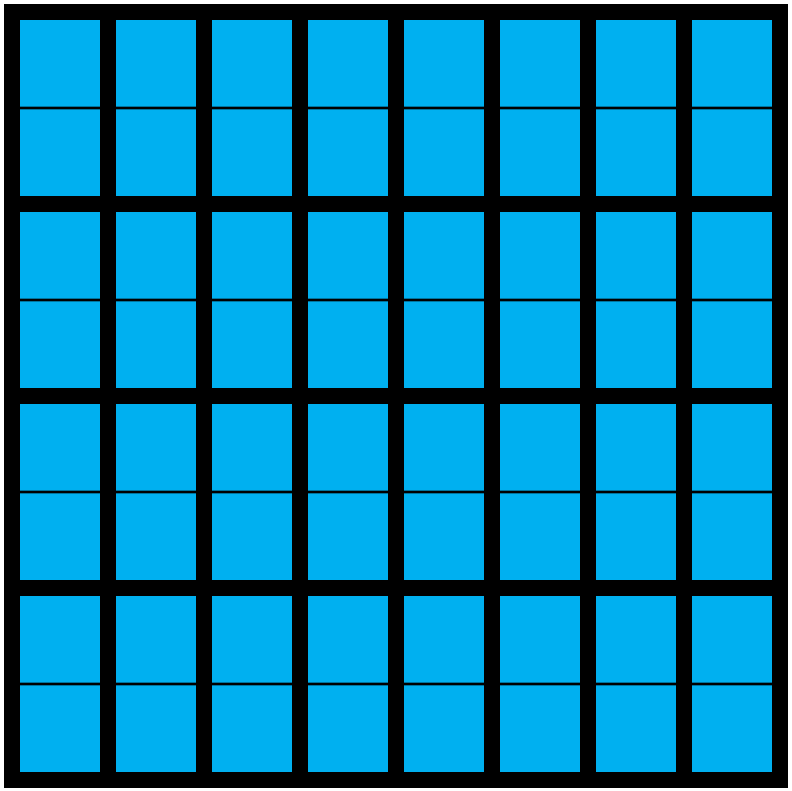


Putem sa o umplem sau nu?



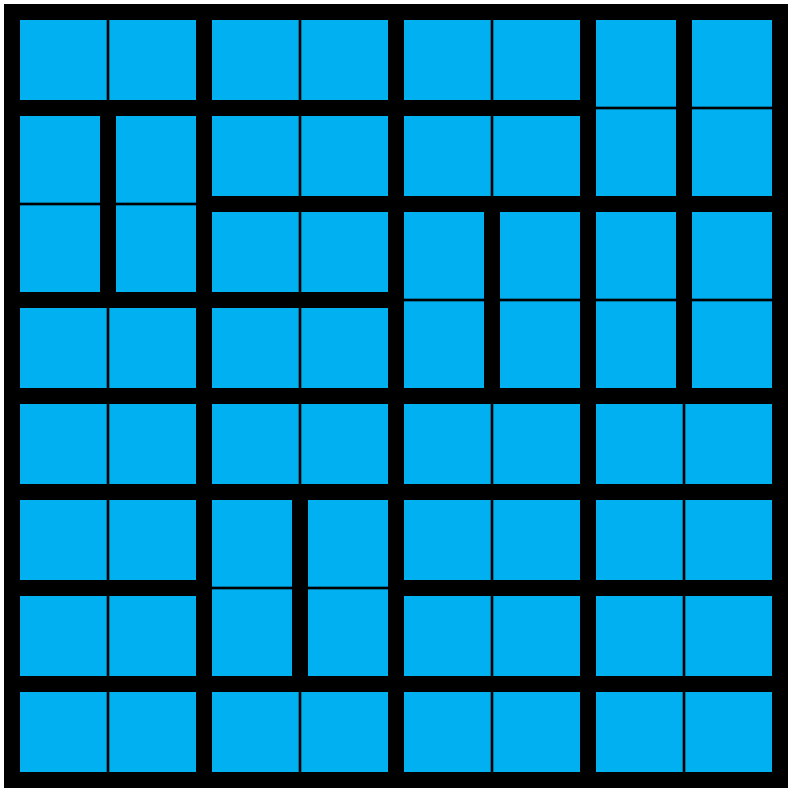
DEMONSTRAȚIA PRIN EXEMPLU

Răspunsul e evident, DA. Și cred că fiecare dintre voi poate demonstra asta. E de ajuns să arătăm un exemplu și am demonstrat:



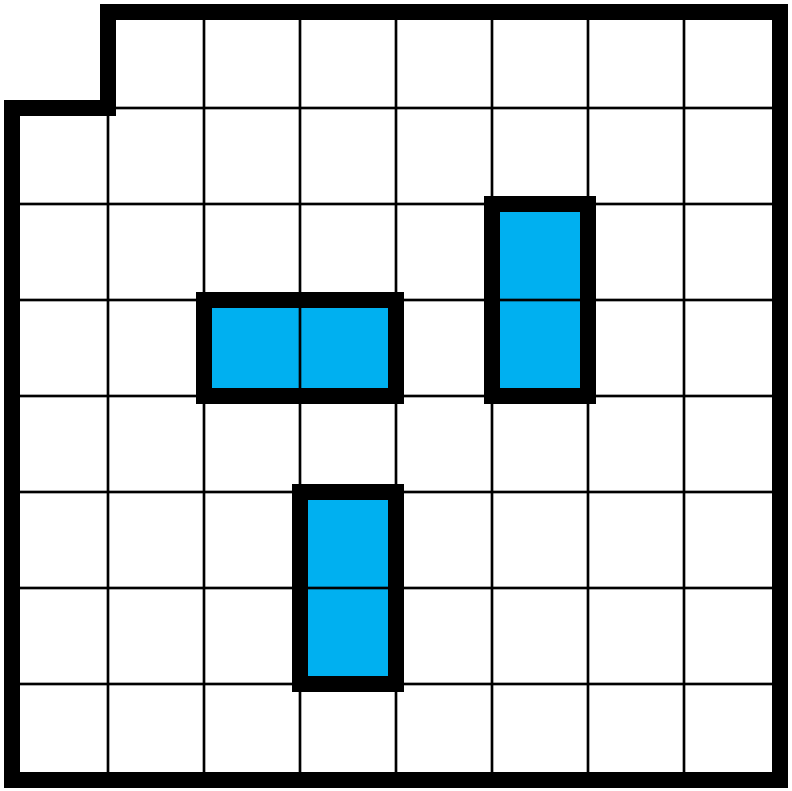
DEMONSTRAȚIA PRIN EXEMPLU

Răspunsul e evident, DA. Și cred că fiecare dintre voi poate demonstra asta. E de ajuns să arătăm un exemplu și am demonstrat:



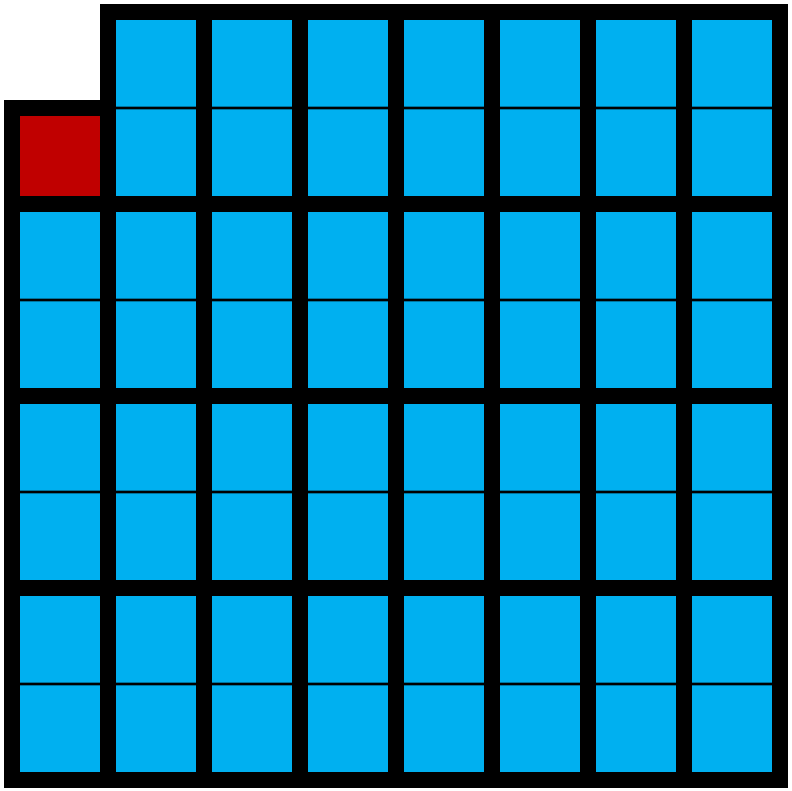
Putem sa o umplem sau nu?

Dar dacă complicăm puțin problema? Dacă eliminăm un pătrat de pe tablă? Mai putem să umplem tabla și de data asta?



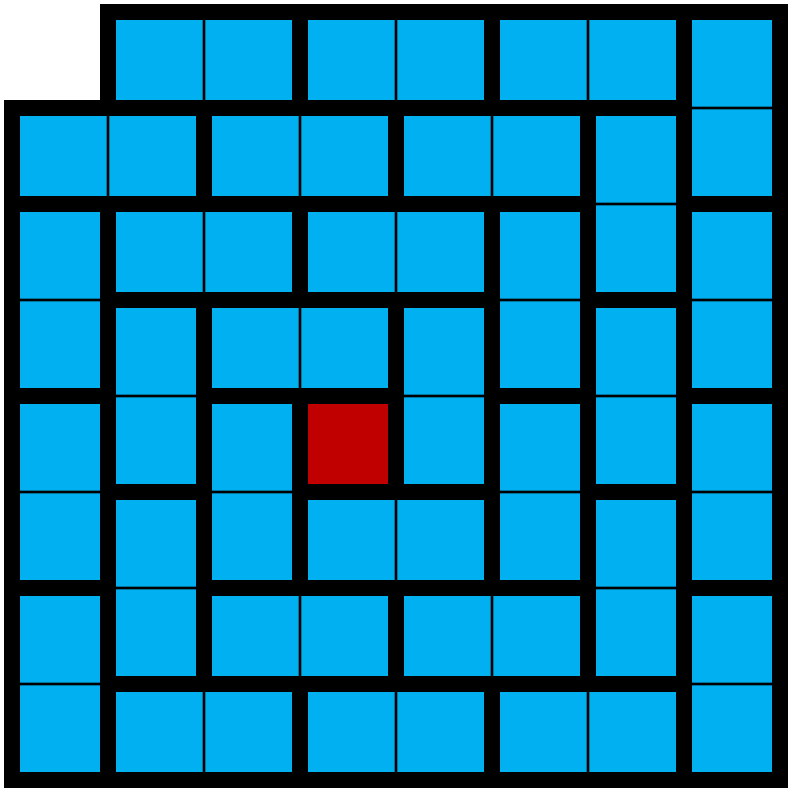
Putem sa o umplem sau nu?

Am încercat în 2 moduri mai jos, dar în ambele ne rămâne un pătrat neumplut. De data asta nu putem demonstra prin exemplu. Dar putem demonstra că nu există nici o combinație posibilă, oare?



Putem sa o umplem sau nu?

Am încercat în 2 moduri mai jos, dar în ambele ne rămâne un pătrat neumplut. De data asta nu putem demonstra prin exemplu. Dar putem demonstra că nu există nici o combinație posibilă, oare?



Putem sa o umplem sau nu?

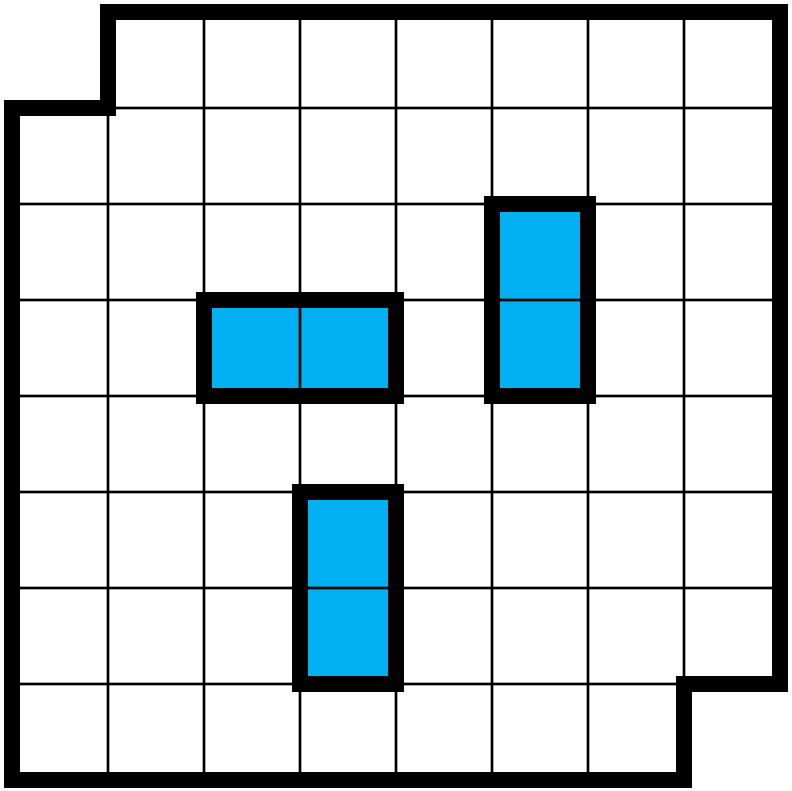
Putem demonstra astfel:

Pe tabla de șah sunt în total 64 de pătrate (8 x 8). Eliminând unul, rămânem doar cu 63 (8 x 8 – 1).

Putem umple cu piese de domino (1 x 2) doar un număr par de pătrate. 63 fiind număr impar, nu poate fi acoperit cu piesele disponibile.

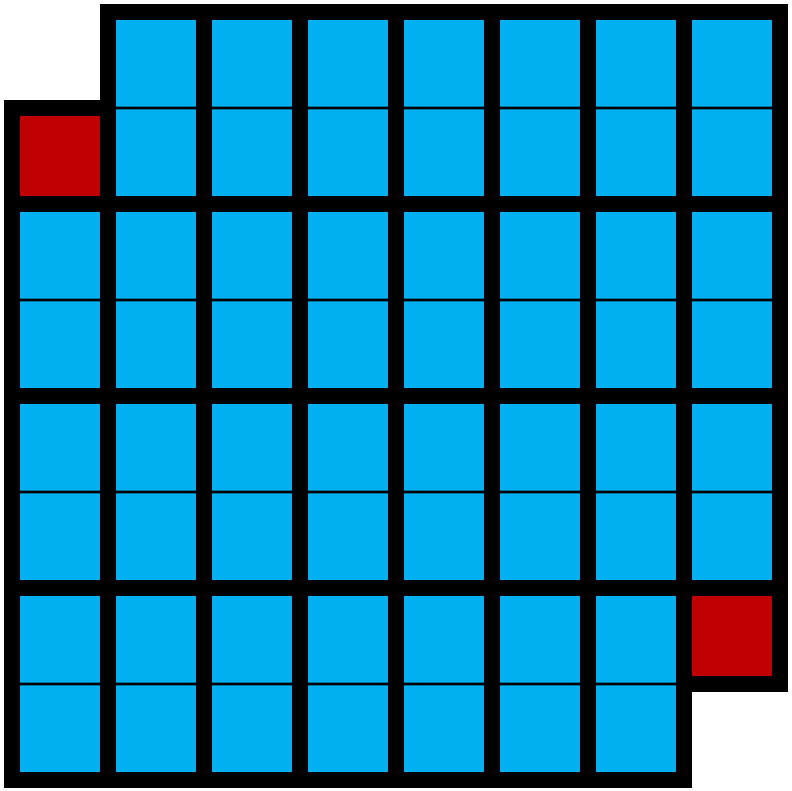
QED

Dar dacă complicăm încă puțin problema? Dacă eliminăm două pătrate de pe tablă? De data asta putem să umplem tabla? Ne rămâne să umplem doar 62 (8 x 8 -2) de pătrate, deci am putea să folosim 31 de piese de domino, nu?



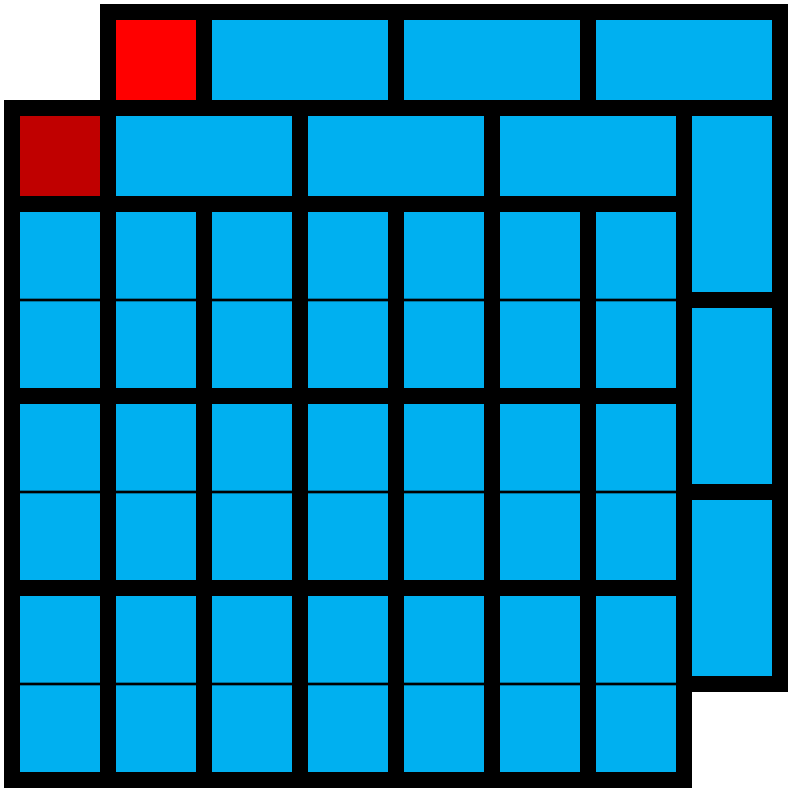
Putem sa o umplem sau nu?

Am încercat în 2 moduri mai jos, dar în ambele ne rămân câte 2 pătrate neumplute. Nici de data asta nu putem demonstra prin exemplu. Putem demonstra, din nou, că nu există nici o combinație posibilă?

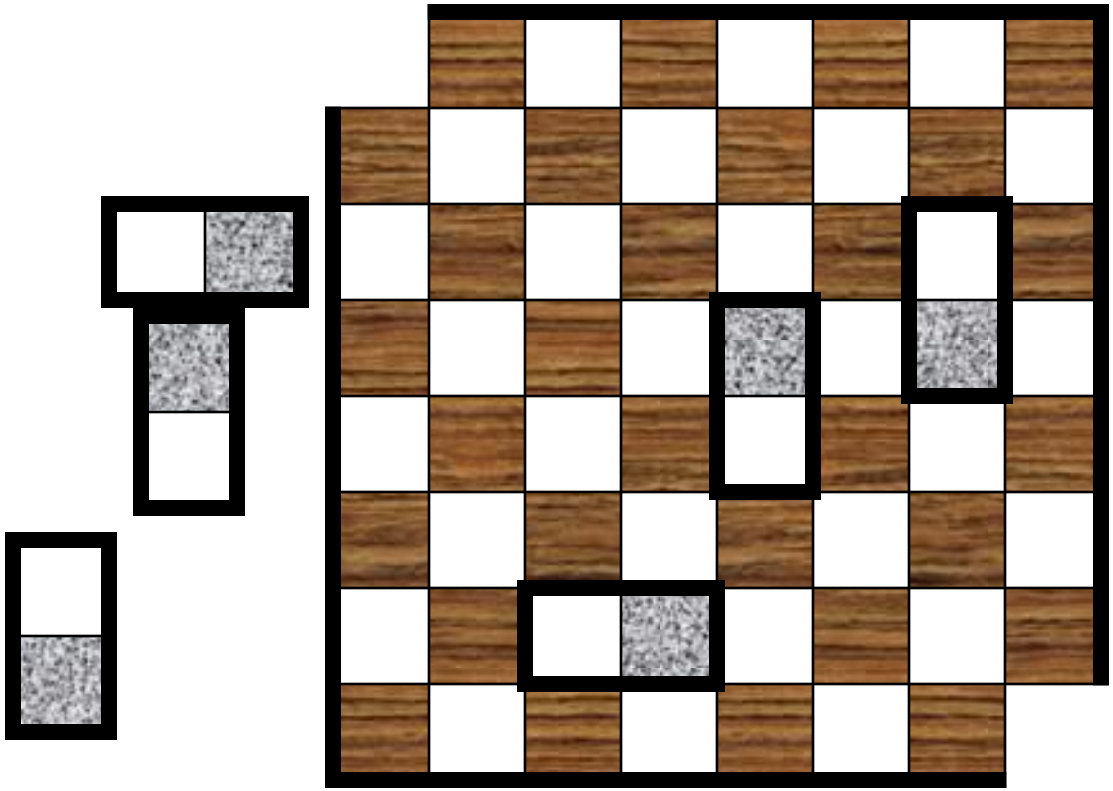


Putem sa o umplem sau nu?

Am încercat în 2 moduri mai jos, dar în ambele ne rămân câte 2 pătrate neumplute. Nici de data asta nu putem demonstra prin exemplu. Putem demonstra, din nou, că nu există nici o combinație posibilă?



Pentru a ne simplifica demonstrația ne vom folosi de cele 2 culori ale tablei de șah. De fiecare dată vor fi 2 culori diferite acoperite. Noi am eliminat 2 patrate albe, deci au mai rămas pe tablă 32 de pătrate maro și doar 30 albe, suprafață imposibil de acoperit, pentru că tot timpul vor rămâne nacoperite 2 pătrate colorate neacoperite.



- V-a convins această demonstrație?

- Putem merge mai departe cu exemplele. Dacă eliminăm 2 pătrate de culori diferite, putem să acoperim restul tablei cu piese de domino?
- Putem demonstra că putem acoperi restul tablei, oricare ar fi cele 2 pătrate de culori diferite eliminate?

Ce am aflat până acum despre demonstrații:

- Ca să demonstrăm că ceva există e de ajuns să oferim un exemplu
- Pentru a demonstra că ceva nu există avem de urmat un raționament logic

Vom continua cu un exemplu mai aproape de matematică:

Există un număr?

- Există un număr de 2 cifre care, dacă devine de 7 ori mai mic, atunci prima sa cifră dispare?

Există un număr?

- Există un număr de 2 cifre care, dacă devine de 7 ori mai mic, atunci prima sa cifră dispare?
- Da, 35 / 7 = 5
- Nu este greu de găsit, numerele de 2 cifere divizibile cu 7 sunt doar câteva: 14, 21, 28, 35, 42, 49, 56, 63
- Dar putem să complicăm problema: Există un număr care, dacă devine de 57 de ori mai mic, atunci prima cifră dispare?

Există un număr?

- Există un număr de 2 cifre care, dacă devine de 7 ori mai mic, atunci prima sa cifră dispare?
- Da, 35 / 7 = 5
- Nu este greu de găsit, numerele de 2 cifere divizibile cu 7 sunt doar câteva: 14, 21, 28, 35, 42, 49, 56, 63
- Dar putem să complicăm problema: Există un număr care, dacă devine de 57 de ori mai mic, atunci prima cifră dispare?
- 7125 / 57 = 125

- Există un număr care, dacă devine de 57 de ori mai mic, atunci prima cifră dispare?
- 7125 / 57 = 125
- Cum îl putem găsi matematic:

- Există un număr care, dacă devine de 57 de ori mai mic, atunci prima cifră dispare?
- 7125 / 57 = 125
- Cum îl putem găsi matematic:
  - Notăm numărul căutat cu ab...z, unde fiecare literă e o cifră
  - ab...z = 57 * b...z
  - X = b...z considerăm că are k cifre

- Există un număr care, dacă devine de 57 de ori mai mic, atunci prima cifră dispare?
- 7125 / 57 = 125
- Cum îl putem găsi matematic:
  - Notăm numărul căutat cu ab...z, unde fiecare literă e o cifră
  - ab...z = 57 * b...z
  - X = b...z considerăm că are k cifre
  - a * 10^k + X = 57 * X
  - a * 10^k = 57 * X – X = 56* X = 7 * 8 * X
  - a * $5^k * 2^k$ = 7 * 8 * X

- Există un număr care, dacă devine de 57 de ori mai mic, atunci prima cifră dispare?
- 7125 / 57 = 125
- Cum îl putem găsi matematic:
  - Notăm numărul căutat cu ab...z, unde fiecare literă e o cifră
  - ab...z = 57 * b...z
  - X = b...z considerăm că are k cifre
  - a * 10^k + X = 57 * X
  - a * 10^k = 57 * X – X = 56* X = 7 * 8 * X
  - a * $5^k * 2^k$ = 7 * 8 * X
  - a trebuie să fie divizibil cu 7, a e 1 cifră, deci a = 7
  - 7 * $5^k * 2^k$ = 7 * 8 * X
  - $5^k * 2^k$ = 8 * X
  - $5^k * 2^k$ = 2^3 * X

- Există un număr care, dacă devine de 57 de ori mai mic, atunci prima cifră dispare?
- 7125 / 57 = 125
- Cum îl putem găsi matematic:
  - Notăm numărul căutat cu ab...z, unde fiecare literă e o cifră
  - ab...z = 57 * b...z
  - X = b...z considerăm că are k cifre
  - a * 10^k + X = 57 * X
  - a * 10^k = 57 * X – X = 56* X = 7 * 8 * X
  - a * $5^k * 2^k$ = 7 * 8 * X
  - a trebuie să fie divizibil cu 7, a e 1 cifră, deci a = 7
  - 7 * $5^k * 2^k$ = 7 * 8 * X
  - $5^k * 2^k$ = 8 * X
  - $5^k * 2^k$ = 2^3 * X
  - Dacă k = 3 avem soluția X = 125
  - Deci am găsit un număr dintre cele căutate 7125 = 57*125

Tipuri de demonstrații

- Prin exemplu – am vazut până acum
- Prin reducere la absurd – vom parcurge imediat
- Prin inducție matematică – vom parcurge imediat

Demonstrația prin reducere la absurd

Contrapozitiva unei afirmații

- Exemplu:
  - Propoziția ”Dacă plouă implică că îmi iau umbrela.”
    este echivalentă cu contrapozitiva ei:
  - ”Dacă nu îmi iau umbrela implică că nu plouă.”

Demonstrația prin reducere la absurd

Contrapozitiva unei afirmații

- Exemplu:
  - Propoziția ”Dacă plouă implică că îmi iau umbrela.”
    este echivalentă cu contrapozitiva ei:
  - ”Dacă nu îmi iau umbrela implică că nu plouă.”

- În logică, exemplul de mai sus se transpune formal astfel:
  - Notăm cu P propoziția ”Plouă”
  - Notăm cu Q propoziția ”Îmi iau umbrela”
  - ”Dacă plouă implică că îmi iau umbrela.” se transcrie $P \Rightarrow Q$, unde P e premiza, iar Q e concluzia
  - $\neg$P și $\neg Q$ sunt negatele celor 2 propoziții
  - o afirmație e echivalentă cu contrapozitiva ei:

$P \Rightarrow Q$	$\Leftrightarrow$	$\neg Q \Rightarrow \neg$P
afirmația		contrapozitiva

Demonstrația prin reducere la absurd

- Demonstrația prin reducere la absurd se folosește de echivalența dintre o afirmație și contrapozitiva ei:

$$P \Rightarrow Q \quad \Leftrightarrow \quad \neg Q \Rightarrow \neg \mathrm{P}$$

afirmația — contrapozitiva

  - presupunem concluzia falsa ($\neg Q$)
  - arătăm că atunci premisa e falsă($\neg$P), ceea ce e absurd, știind că premisa e adevărată (P)
  - deci concluzia nu poate fi falsă ⇒ concluzia e adevarată

Demonstrația prin reducere la absurd

- Exemplu: Suma a trei numere naturale e 139. Demonstrați că cel puțin unul dintre ele e mai mare decât 46.

Demonstrația prin reducere la absurd

- Exemplu: Suma a trei numere naturale e 139. Demonstrați că cel puțin unul dintre ele e mai mare decât 46.

  - *P: ”a+b+c = 139, unde a, b, c sunt numere naturale”*
  - *Q: ”a > 46 sau b > 46 sau c > 46”*

Demonstrația prin reducere la absurd

- Exemplu: Suma a trei numere naturale e 139. Demonstrați că cel puțin unul dintre ele e mai mare decât 46.

  - *P: "a+b+c = 139, unde a, b, c sunt numere naturale"*
  - *Q: "a > 46 sau b > 46 sau c > 46"*
  - *Construim* $\neg Q$: "*a ≤ 46 și b ≤ 46 și c ≤ 46*"
  - $\neg Q \Rightarrow$ "*a+b+c ≤ 46+46+46 ≤ 138*" e în contradicție cu P care spune că suma este 139 $\Rightarrow \neg$P

Demonstrația prin reducere la absurd

- Exemplu: Suma a trei numere naturale e 139. Demonstrați că cel puțin unul dintre ele e mai mare decât 46.

  - *P: "a+b+c = 139, unde a, b, c sunt numere naturale"*
  - *Q: "a > 46 sau b > 46 sau c > 46"*
  - *Construim* $\neg Q$: "*a ≤ 46 și b ≤ 46 și c ≤ 46*"
  - $\neg Q$ ⇒"*a+b+c ≤ 46+46+46 ≤ 138*" e în contradicție cu P care spune că suma este 139 ⇒ ¬P
  - Prin urmare P ⇒Q e adevarată
  - QED

Demonstrația prin inducție matematică

- Daca o propoziție P(n) depinde de un număr natural n și:

  1) cazul de baza : P(1) e adevarată

  2) pasul inductiv : pentru orice n≥1

  $$P(n) \Rightarrow P(n + 1)$$

- atunci P(n) e adevarată pentru orice n.

Demonstrația prin inducție matematică

- Exemplu: Să se demonstreze că

$$9^n - 1 \vdots 8, \forall n \in \mathbb{N}*$$

Demonstrația prin inducție matematică

- Exemplu: Să se demonstreze că

$$9^n - 1 \vdots 8, \forall n \in \mathbb{N}*$$

1) calculăm P(1)= $9^1 - 1 = 8 \vdots 8$ - adevărat

2) presupunem P(n) adevărat

$$P(n) = 9^n - 1 \vdots 8$$

Calculăm $P(n+1) = 9^{n+1} - 1$

Demonstrația prin inducție matematică

- Exemplu: Să se demonstreze că

$$9^n - 1 \vdots 8, \forall n \in \mathbb{N}*$$

1) calculăm P(1)= $9^1 - 1 = 8 \vdots 8$ - adevărat

2) presupunem P(n) adevărat

$$P(n) = 9^n - 1 \vdots 8$$

Calculăm $P(n+1) = 9^{n+1} - 1$

$$= 9^n * 9 - 1$$
$$= 9^n * 9 - (9 - 8)$$
$$= 9 * (9^n - 1) - 8$$
$$= 9 * P(n) - 8$$

Demonstrația prin inducție matematică

- Exemplu: Să se demonstreze că

$$9^n - 1 \vdots 8, \forall n \in \mathbb{N}*$$

1) calculăm P(1)= $9^1 - 1 = 8 \vdots 8$ - adevărat

2) presupunem P(n) adevărat

$$P(n) = 9^n - 1 \vdots 8$$

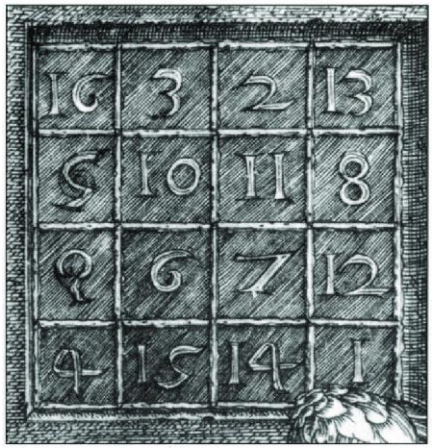
Calculăm $P(n+1) = 9^{n+1} - 1$

$$= 9^n * 9 - 1$$
$$= 9^n * 9 - (9 - 8)$$
$$= 9 * (9^n - 1) - 8$$
$$= 9 * P(n) - 8$$

$P(n) \vdots 8$ *și* $8 \vdots 8 \Rightarrow P(n+1) = 9 * P(n) - 8 \vdots 8$ – *adevărat*

$9^n - 1 \vdots 8, \forall n \in \mathbb{N}*$

QED



Ce facem la LSD

Demonstrații

Mulțimi

Funcții

Proprietăți ale funcțiilor

Funcțiile în programare

Noțiuni introductive despre MULȚIMI

B426

B528a

B528b

B418a

Ce sunt mulțimile?

Defniție:

O mulțime este o colecție de obiecte numite elementele mulțimii.

Avem două noțiuni distincte: elemente și mulțime

- $x \in S$: elementul x aparține mulțimii S
- $y \notin S$: elementul y nu aparține mulțimii S

Ordinea elementelor nu conteaza {1, 2, 3} = {2, 1, 3}

Un element nu apare de mai multe ori {1, 2, 3, 2}

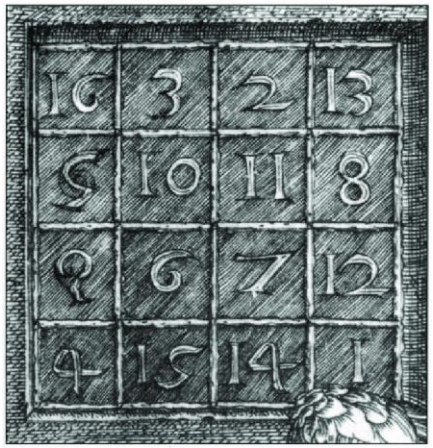
Submulțimi

A e o *submulțime* a lui B: $A \subseteq B$

dacă fiecare element al lui A e și un element al lui B.

Ca să demonstrăm $A \not\subseteq B$ e suficient să găsim un element $x \in A$ pentru care $x \notin B$.

- Dacă $A \subseteq B$ și $B \subseteq A$, atunci $A = B$ (mulțimile sunt egale)



Ce facem la LSD

Demonstrații

Mulțimi

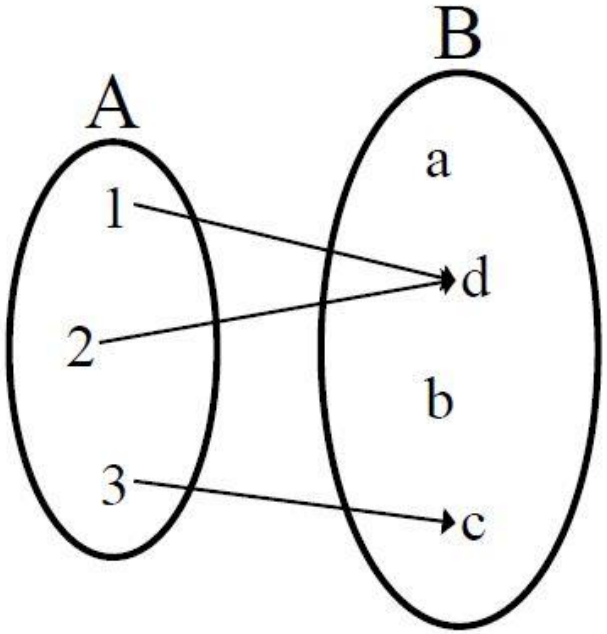
Funcții

Proprietăți ale funcțiilor

Funcțiile în programare

FUNCȚII

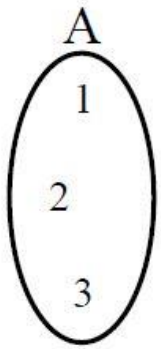
Fiind date mulțimile *A* și *B*, o *funcție* $f : A \to B$ e o asociere prin care *fiecărui* element din *A* îi corespunde *un singur* element din *B*.



Imagine: http://en.wikipedia.org/wiki/File:Total_function.svg

O funcție e definită prin trei componente

1. *domeniul de definiție*

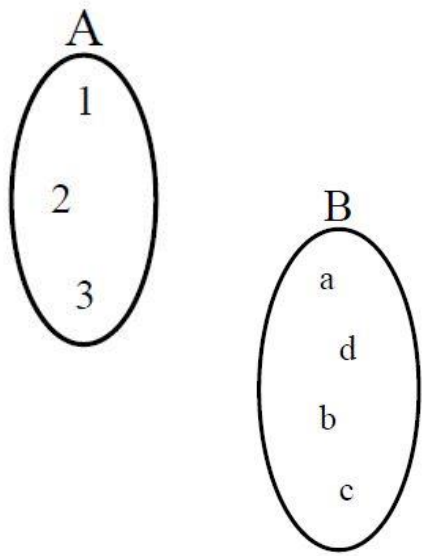


Imagine: http://en.wikipedia.org/wiki/File:Total_function.svg

O funcție e definită prin trei componente

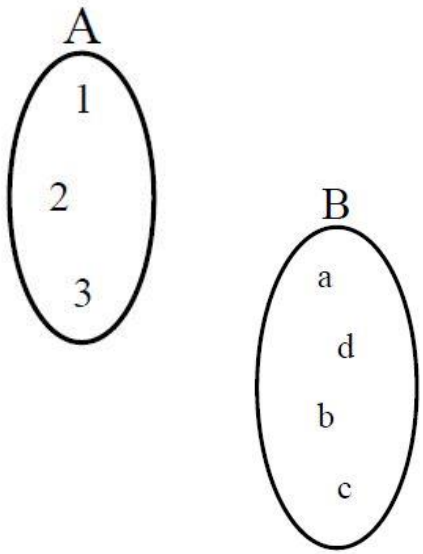
1. *domeniul de definiție*

2. *domeniul de* valori (codomeniul)



Imagine: http://en.wikipedia.org/wiki/File:Total_function.svg

O funcție e definită prin trei componente



1. *domeniul de definiție*

2. *domeniul de* valori (codomeniul)

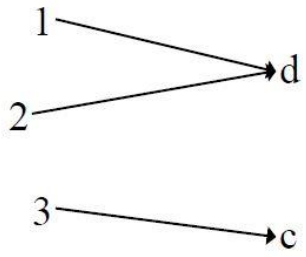
3. *asocierea propriu-zisă*

(legea, regula de asociere, corespondența)

$f : Z \to Z,\ f(x) = x + 1$ și

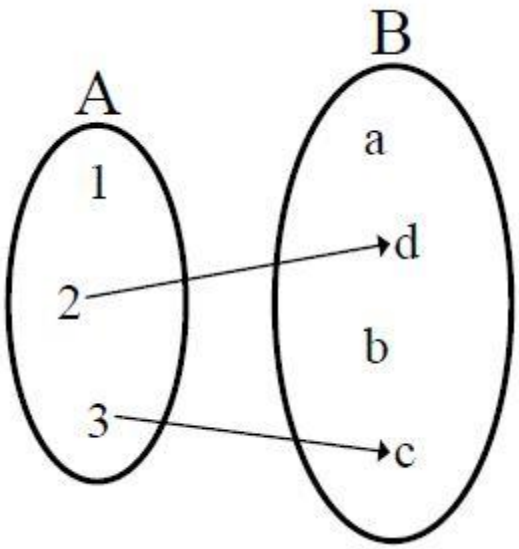
$f : R \to R, f(x) = x + 1$

sunt funcții distincte!

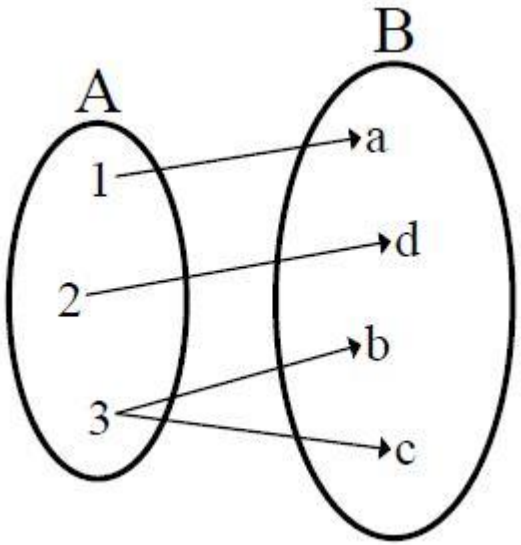


Imagine: http://en.wikipedia.org/wiki/File:Total_function.svg

Exemple care nu sunt funcții



nu asociază o valoare fiecărui element

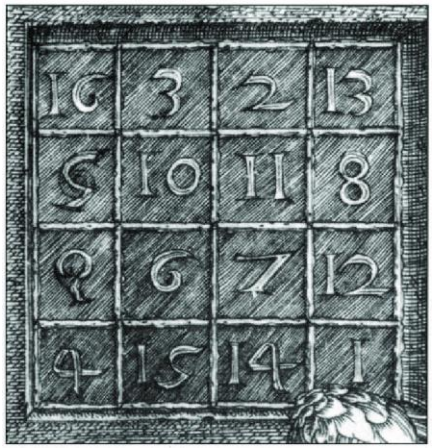


asociaza mai multe valori unui element

Imagini: http://en.wikipedia.org/wiki/File:Partial_function.svg
http://en.wikipedia.org/wiki/File:Multivalued_function.svg

Definiție alternativă a funcțiilor

O funcție $f : A \to B$ este o *mulțime* $f \subseteq A \times B$ a. î. pentru *fiecare* element $a \in A$ există un *unic* element $b \in B$ a. î. $(a, b) \in f$. Notăm această alegere unică a lui b cu $f(a)$.



Ce facem la LSD

Demonstrații

Mulțimi

Funcții

Proprietăți ale funcțiilor

Funcțiile în programare

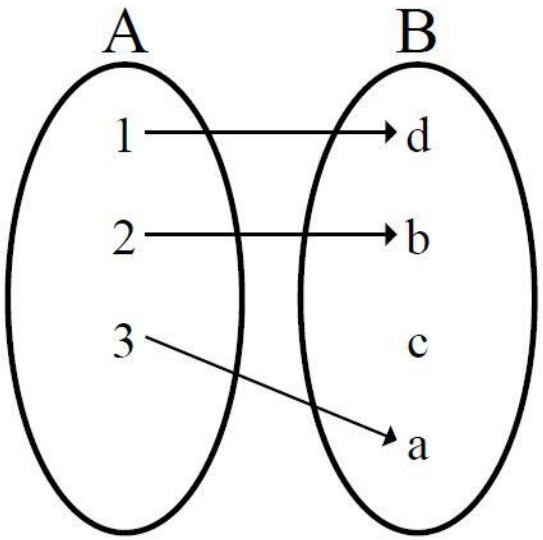
Funcții injective

O funcție $f : A \to B$ e *injectivă* dacă pentru orice

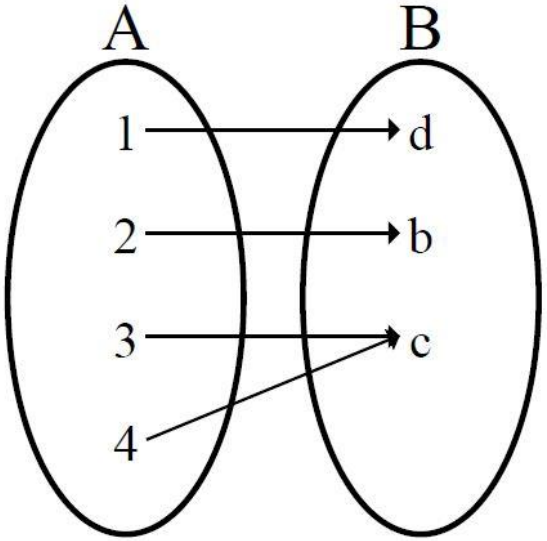
$x_1, x_2 \in A, x_1 \neq x_2 \Rightarrow f(x_1) \neq f(x_2)$

(asociază *valori diferite* la *argumente diferite*)

funcție injectivă



funcție neinjectivă



Imagine: http://en.wikipedia.org/wiki/File:Injection.svg
http://en.wikipedia.org/wiki/File:Surjection.svg

Funcții injective

În locul condiției $x_1, x_2 \in A, x_1 \neq x_2 \Rightarrow f(x_1) \neq f(x_2)$
putem scrie echivalent:

$$f(x_1) = f(x_2) \Rightarrow x_1 = x_2$$

(dacă valorile sunt egale, atunci argumentele sunt egale)

E totuna cu $x_1, x_2 \in A, x_1 = x_2 \Rightarrow f(x_1) = f(x_2)$?

Funcții injective

În locul condiției $x_1, x_2 \in A, x_1 \neq x_2 \Rightarrow f(x_1) \neq f(x_2)$ putem scrie echivalent:

$$f(x_1) = f(x_2) \Rightarrow x_1 = x_2$$

(dacă valorile sunt egale, atunci argumentele sunt egale)

E totuna cu $x_1, x_2 \in A, x_1 = x_2 \Rightarrow f(x_1) = f(x_2)$?

Nu! *Orice funcție* ia aceeași valoare pentru argumente egale! (e o proprietate de bază a egalității și substituției)

Funcții injective

Proprietate a funcțiilor injective:

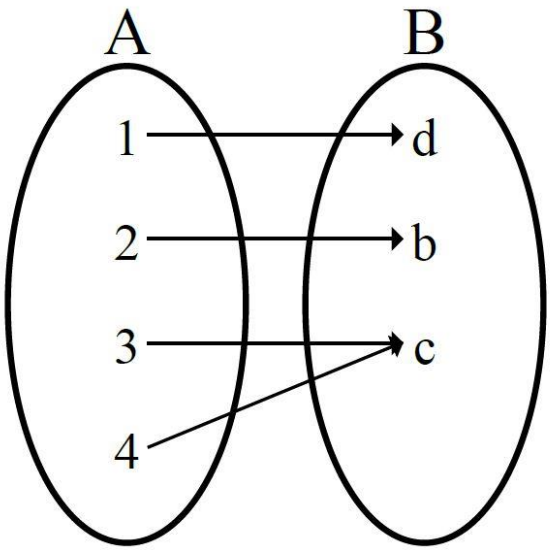
Dacă $f : A \to B$ și f e injectivă, atunci $|A| \leq |B|$.

Nu și invers!

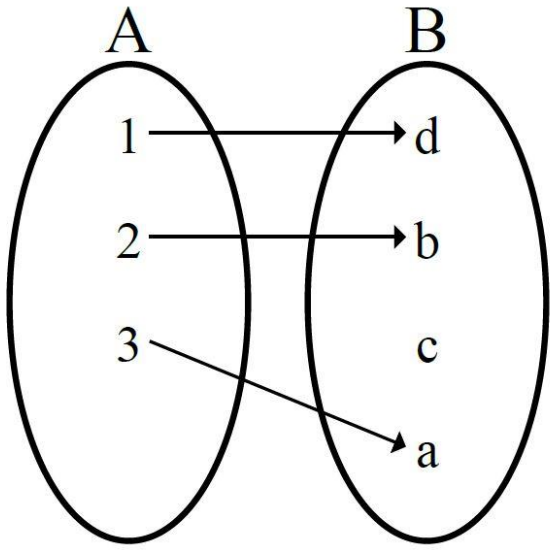
Pentru orice mulțime A a.î. $|A| > 1$ putem construi f să ducă două elemente din A în aceeași valoare din B

Funcții surjective

O funcție $f : A \to B$ e *surjectivă* dacă pentru fiecare $y \in B$ există un $x \in A$ cu $f(x) = y$.



funcție surjectivă



funcție nesurjectivă

Imagine: http://en.wikipedia.org/wiki/File:Surjection.svg
Imagine: http://en.wikipedia.org/wiki/File:Injection.svg

Funcții surjective

Proprietate a funcțiilor surjective:

Dacă $f : A \to B$ și f e surjectivă, atunci $|A| \geq |B|$.

Nu și invers! Putem construi f a. î. să nu ia ca valoare un element anume din B, dacă $|B| > 1$.

Funcții surjective

Proprietate a funcțiilor surjective:

Dacă $f : A \to B$ și f e surjectivă, atunci $|A| \geq |B|$.

Nu și invers! Putem construi f a. î. să nu ia ca valoare un element anume din B, dacă $|B| > 1$.

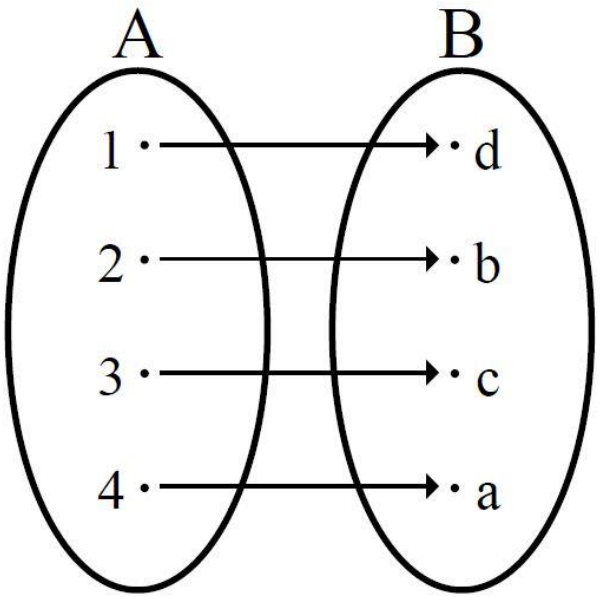
Putem transforma o funcție nesurjectivă într-una surjectivă prin *restrângerea* domeniului de valori:

- $f_1 : \mathrm{R} \to \mathrm{R}$, $f_1(x) = x^2$ nu e surjectivă,
- dar $f_2 : \mathrm{R} \to [0, \infty)$, $f_2(x) = x^2$ (restrânsă la valori nenegative) este surjectivă.

Funcții bijective

O funcție care e injectivă și surjectivă se numește *bijectivă*.

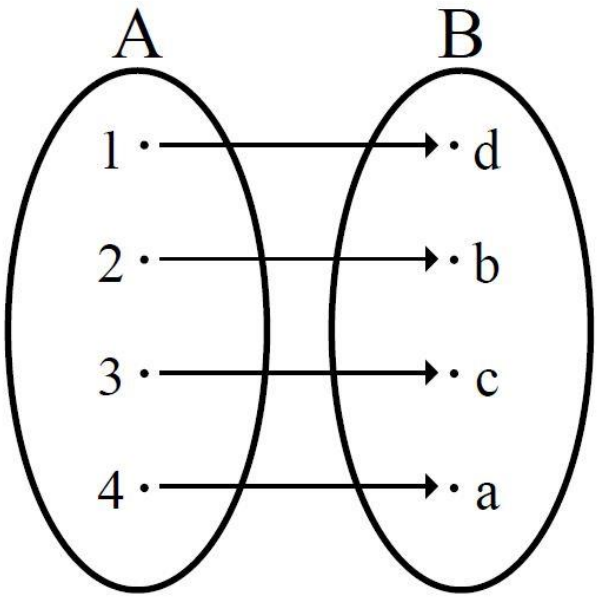
O funcție bijectivă $f : A \rightarrow B$ pune în corespondență *unu la unu* elementele lui A cu cele ale lui B.



Imagine: http://en.wikipedia.org/wiki/File:Bijection.svg

Funcții bijective

Pentru *orice* funcție, din definiție, la fiecare $x \in A$ corespunde un *unic* $y \in B$ cu $f(x) = y$.
Pentru o funcție *bijectivă*, și invers: la fiecare $y \in B$ corespunde un *unic*
$x \in A$ cu $f(x) = y$.

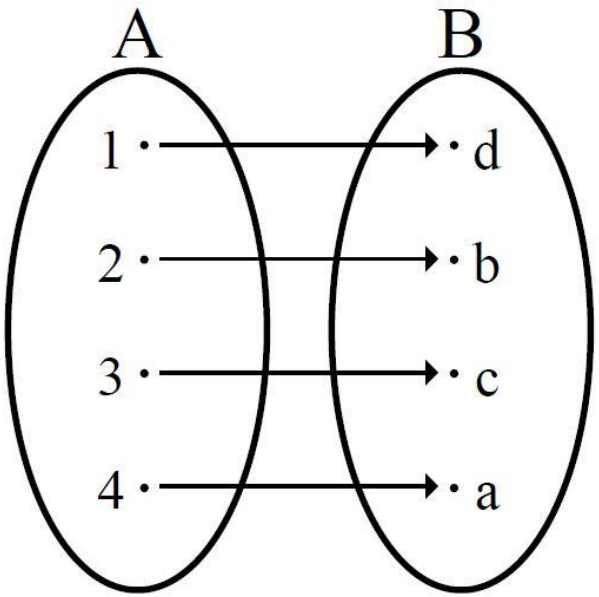


Imagine: http://en.wikipedia.org/wiki/File:Bijection.svg

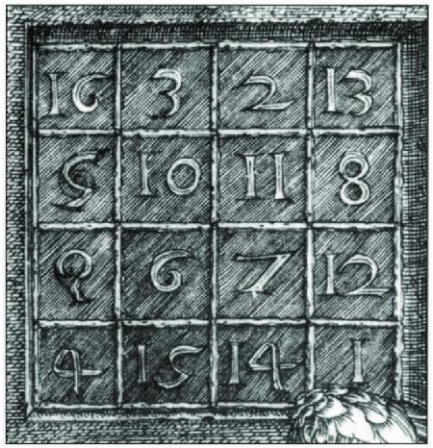
Funcții bijective

Pentru *orice* funcție, din definiție, la fiecare $x \in A$ corespunde un *unic* $y \in B$ cu $f(x) = y$.
Pentru o funcție *bijectivă*, și invers: la fiecare $y \in B$ corespunde un *unic*
$x \in A$ cu $f(x) = y$.

Dacă există $f : A \to B$ și
f e bijectivă, atunci $|A| = |B|$.



Imagine: http://en.wikipedia.org/wiki/File:Bijection.svg



Ce facem la LSD

Demonstrații

Mulțimi

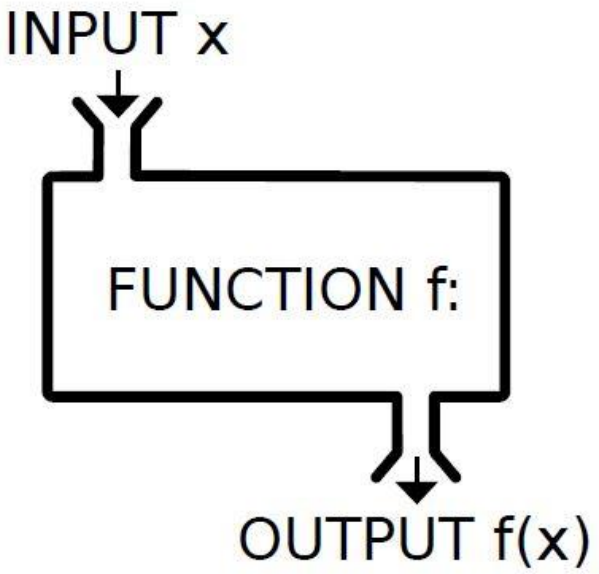
Funcții

Proprietăți ale funcțiilor

Funcțiile în programare

Funcția în programare

În limbajele de programare, o funcție exprimă un *calcul*: primește o *valoare* (*argumentul*) și produce ca *rezultat* altă *valoare*



Imagine: http://en.wikipedia.org/wiki/File:Function_machine2.svg

Funcții în Python

Funcțiile se definesc simplu, cu o sintaxă (regulă de scriere) similară cu alte limbaje de programare. Astfel, funcția

$f : \mathbf{Z} \rightarrow \mathbf{Z}, f(x) = x + 3$

Funcții în Python

Funcțiile se definesc simplu, cu o sintaxă (regulă de scriere) similară cu alte limbaje de programare. Astfel, funcția

$f : \mathbf{Z} \to \mathbf{Z}, f(x) = x + 3$ se scrie în Python:

```
def f(x):
        return x + 3
```

Cuvântul cheie def introduce o *definiție*, aici, pentru identificatorul f.
Dupa numele funcției, se introduc parametrii acesteia între paranteze (*x*), urmați de semnul **:** (două puncte).

Funcții în Python

```
def f(x):
    return x + 3
```

În Python, *indentarea* este foarte importantă în scrierea codului.

Spre deosebire de alte limbaje de programare ce folosesc acolade { }, în Python indentarea este folosită pentru blocurile de cod.

Indentarea este întotdeauna precedată de semnul **:** (două puncte).

Apelarea funcțiilor în Python

Odată definită funcția, aceasta se apelează în felul următor:

```
>>> f(1)
4
```

Când apelăm o funcție putem să specificăm și numele parametrului la apel:

```
>>> f(x=5)
8
```

Putem da funcției și o expresie complexă ca parametru:

```
>>> f(2*3)
9
```

Funcții anonime în Python

Notația lambda *argument* : *expresie* definește în Python o funcție anonimă (funcție lambda).

Aceasta este o *expresie* de tip funcție și poate fi folosită în alte expresii. Putem evalua direct:

```
>>> (lambda x : x + 3)(2)
5
```

fără a fi nevoie să dăm întâi un nume funcției.

Acest exemplu simplu ilustrează că în limbajul Python, o funcție poate fi folosită la fel de simplu ca și orice altă valoare.

Funcții anonime în Python

Folosind notația def, este echivalent să definim:

```
>>> def f(x):
...         return x + 3
```

sau

```
>>> def f(x):
...         return (lambda x : x + 3)
```

Funcții cu mai multe argumente în Python

Fie funcția din matematică:

$$suma: \mathbf{Z} \times \mathbf{Z} \rightarrow \mathbf{Z},\ suma(x, y) = x + y$$

Funcții cu mai multe argumente în Python

Fie funcția din matematică:

$$suma: \mathbf{Z} \times \mathbf{Z} \rightarrow \mathbf{Z},\ suma(x, y) = x + y$$

Varianta cea mai uzuală de a transcrie această funcție într-un program Python este:

```
>>> def suma(x, y):
...         return x + y
```

Funcțiile sunt valori în Python

O funcție e și ea o valoare (ca întregii, realii, etc.) și poate fi folosită la fel ca orice valoare (data ca parametru, returnata, etc.):

Funcțiile sunt valori în Python

O funcție e și ea o valoare (ca întregii, realii, etc.) și poate fi folosită la fel ca orice valoare (data ca parametru, returnata, etc.):

```
>>> def g(f,x):
...         return f(x) + x
```

Funcții definite pe cazuri în Python

Fie $abs : Z \rightarrow Z$, $abs(x) = \begin{cases} x, \text{ dacă } x \geq 0 \\ -x, \text{ altfel } (x < 0) \end{cases}$

Valoarea funcției nu e dată de o singură expresie,
ci de una din două expresii diferite (x sau -x),
depinzând de o condiție ($x \geq 0$).

Funcții definite pe cazuri în Python

Fie $abs : Z \to Z$, $abs(x) = \begin{cases} x, \text{ dacă } x \geq 0 \\ -x, \text{ altfel } (x < 0) \end{cases}$

Valoarea funcției nu e dată de o singură expresie,
ci de una din două expresii diferite (x sau -x),
depinzând de o condiție ($x \geq 0$).

În Python:

```
def abs(x):
        if (x > 0):
                return x
        else:
                return -x
```

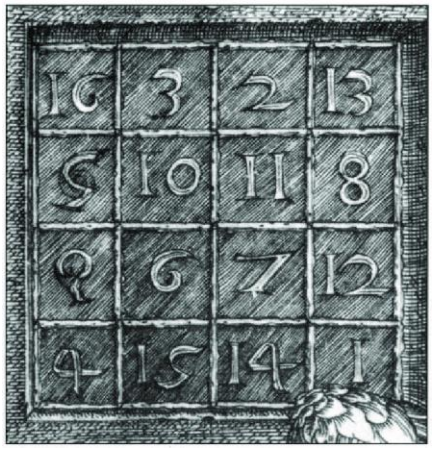
Funcții definite pe cazuri în Python

Cazul general al instrucțiunii *if este:*

```
if (expr1):
        expr2
else:
        expr3
```

Instrucțiunea if se evaluează astfel:

- Daca evaluarea lui expr1 da valoarea true (adevarat), valoarea expresiei e valoarea lui expr2,
- altfel e valoarea lui expr3.



O zi bună în continuare!

Bibliografie

- Exemplele cu tabla de șah și piesele de domino au fost inspirate din cursul ***Mathematical Thinking in Computer Science*** de la University of California San Diego (de pe https://www.coursera.org/)
- Conținutul cursului are punctul de plecare de la materialele de anii trecuți de la cursul de LSD, predat de conf. dr. ing.Marius Minea și ș.l. dr. ing. Casandra Holotescu (http://staff.cs.upt.ro/~marius/curs/lsd/index.html)