

Seminar 1

Biği

Baze de numerație

- ▶ Baza $b < 10 \rightarrow$ cifrele $\{0, 1, \dots, b-1\}$
- ▶ Pentru exemplificare – baza $b=2$ (folosita și pentru reprezentarea internă a numerelor)

Baze de numerație

► Conversie baza 10 \rightarrow baza 2

Se dă un număr natural x în baza 10. Să se determine a_1, \dots, a_n cifrele lui x în scrierea în baza 2:

$$x = \overline{a_1 \dots a_n}_{(2)}$$

Baze de numerație

► Conversie baza 10 \rightarrow baza 2

Se dă un număr natural x în baza 10. Să se determine a_1, \dots, a_n cifrele lui x în scrierea în baza 2:

$$x = \overline{a_1 \dots a_n}_{(2)}$$

$$x = a_n + a_{n-1} \cdot 2^1 + \dots + a_1 \cdot 2^{n-1} \Rightarrow$$

Baze de numerație

► Conversie baza 10 \rightarrow baza 2

Se dă un număr natural x în baza 10. Să se determine a_1, \dots, a_n cifrele lui x în scrierea în baza 2:

$$x = \overline{a_1 \dots a_n}_{(2)}$$

$$x = a_n + a_{n-1} \cdot 2^1 + \dots + a_1 \cdot 2^{n-1} \Rightarrow$$

$$a_n = x \bmod 2$$

$$a_{n-1} + a_{n-2} \cdot 2 + \dots + a_1 \cdot 2^{n-2} = x \operatorname{div} 2 \Rightarrow$$

$$\Rightarrow a_{n-1} = x \bmod 2 \text{ etc}$$

Baze de numerație

► Conversie baza 10 \rightarrow baza 2

Se dă un număr natural x în baza 10. Să se determine a_1, \dots, a_n cifrele lui x în scrierea în baza 2:

$$x = \overline{a_1 \dots a_n}_{(2)}$$

Idee algoritm: Împărțim succesiv la 2 și păstrăm resturile (obținem cifrele din reprezentarea în baza 2 de la dreapta la stânga)

Baze de numerație

- ▶ Conversie baza 10 \rightarrow baza 2

Exemplu $x = 86$

Conversie baza 10 -> baza 2

```
#include<iostream>
using namespace std;

int f10to2(unsigned int x) {
    int rez=0,p10=1,a;

    while(x>0){
        a=x%2;
        x=x/2;

        rez=rez+a*p10;
        p10*=10;
    }
    return rez;
}

//avem main
int main(){
    cout<<f10to2(235)<<endl;
}
```


Conversie baza 10 -> baza 2

```
#include<iostream>
using namespace std;

int f10to2(unsigned int x) {
    int rez=0,p10=1,a;

    while(x>0){
        a=x%2;
        x=x/2;

        rez=rez+a*p10;
        p10*=10;
    }
    return rez;
}

//avem main
int main(){
    cout<<f10to2(235)<<endl;
}
```

#nu includem/importam nimic

```
def f10to2(x):
    rez=0
    p10=1
```

Conversie baza 10 -> baza 2

```
#include<iostream>
using namespace std;

int f10to2(unsigned int x) {
    int rez=0,p10=1,a;

    while(x>0){
        a=x%2;
        x=x/2;

        rez=rez+a*p10;
        p10*=10;
    }
    return rez;
}

//avem main
int main(){
    cout<<f10to2(235)<<endl;
}
```

```
#nu includem/importam nimic

def f10to2(x):
    rez=0
    p10=1
    while x>0:
        a=x%2
        x=x//2

        rez=rez+a*p10
        p10*=10

    return rez
```

Conversie baza 10 -> baza 2

```
#include<iostream>
using namespace std;

int f10to2(unsigned int x) {
    int rez=0,p10=1,a;

    while(x>0){
        a=x%2;
        x=x/2;

        rez=rez+a*p10;
        p10*=10;
    }
    return rez;
}

//avem main
int main(){
    cout<<f10to2(235)<<endl;
}
```

```
#nu includem/importam nimic

def f10to2(x):
    rez=0
    p10=1
    while x>0:
        a=x%2
        x=x//2

        rez=rez+a*p10
        p10*=10

    return rez

#nu avem main
print(f10to2(235))
```

Baze de numerație

- ▶ Temă: Scrieți un program Python pentru conversia din baza 2 în 10

Reprezentarea numerelor întregi în calculator

- ▶ naturale – baza 2 $x = \overline{a_1 \dots a_n}_{(2)} \Rightarrow 00\dots 0a_1\dots a_n$
- ▶ negative – complement față de 2, un bit pentru semn

$$x + (-x) = 0$$

$$\text{repr}(x) + \text{repr}(-x) = 2^n = \underbrace{10 \dots 0}_{n \text{ biți}}_{(2)}$$

Reprezentarea numerelor întregi în calculator

- ▶ naturale – baza 2 $x = \overline{a_1 \dots a_n}_{(2)} \Rightarrow 00\dots 0a_1\dots a_n$
- ▶ negative – complement față de 2, un bit pentru semn

$$x + (-x) = 0$$

$$\text{repr}(x) + \text{repr}(-x) = 2^n = \underbrace{10 \dots 0}_{n \text{ biți}}_{(2)}$$

$$\text{not}(\text{repr}(x)) = \text{complementul lui repr}(x)$$

$$\text{repr}(x) + \text{not}(\text{repr}(x)) = 1 \dots 1_{(2)} = 2^n - 1$$

Reprezentarea numerelor întregi în calculator

- ▶ naturale – baza 2 $x = \overline{a_1 \dots a_n}_{(2)} \Rightarrow 00\dots 0a_1\dots a_n$
- ▶ negative – complement față de 2, un bit pentru semn

$$x + (-x) = 0$$

$$\text{repr}(x) + \text{repr}(-x) = 2^n = \underbrace{10 \dots 0}_{n \text{ biți}}_{(2)}$$

$\text{not}(\text{repr}(x)) = \text{complementul lui repr}(x)$

$$\text{repr}(x) + \text{not}(\text{repr}(x)) = 1 \dots 1_{(2)} = 2^n - 1$$

$$\begin{aligned} \text{Avem atunci } \text{repr}(-x) &= \text{not}(\text{repr}(x)) + 1 = \\ &= \text{not}(\text{repr}(x-1)) \end{aligned}$$

Reprezentarea numerelor întregi în calculator

- ▶ naturale – baza 2

$$x = \overline{a_1 \dots a_n}_{(2)} \Rightarrow \text{repr}(x) = 00\dots 0a_1\dots a_n$$

- ▶ negative – complement față de 2, un bit pentru semn

$$\text{repr}(-x) = \text{not}(\text{repr}(x)) + 1$$

Reprezentarea numerelor întregi

- ▶ $\text{repr}(-x) = \text{not repr}(x) + 1$
- ▶ **Exemplu:** pentru $n=8$ biți reprezentarea lui -11 se obține astfel:

$$\text{repr}(11) = 00001011$$

$$\text{not}(\text{repr}(11)) = 11110100$$

$$\text{not}(\text{repr}(11)) + 1 = 11110101 = \text{repr}(-11)$$

Operatori

- Operatori pe biți

–rapizi, asupra reprezentării interne

$\sim x$	complement față de 1
$x \& y$	și pe biți
$x y$	sau pe biți
$x \wedge y$	sau exclusiv pe biți
$x \gg k$	deplasare la dreapta cu k biți
$x \ll k$	deplasare la stânga cu k biți


\wedge	0	1
0	0	1
1	1	0

Operatori pe biți

$$11 \ \& \ 86 = ?$$

$$11 \ \wedge \ 86 = ?$$

11:	0	0	0	0	1	0	1	1
86:	0	1	0	1	0	1	1	0
11 & 86:	0	0	0	0	0	0	1	0




Operatori pe biți

$$11 \ \& \ 86 = ?$$

$$11 \ \wedge \ 86 = ?$$

11:	0	0	0	0	1	0	1	1
86:	0	1	0	1	0	1	1	0
11 & 86:	0	0	0	0	0	0	1	0




2

Operatori pe biți

$$11 \ \& \ 86 = ?$$

$$11 \ \wedge \ 86 = ?$$

11:	0	0	0	0	1	0	1	1
86:	0	1	0	1	0	1	1	0
11 \wedge 86:	0	1	0	1	1	1	0	1



Operatori pe biți

$$11 \ \& \ 86 = ?$$

$$11 \ \wedge \ 86 = ?$$

11:	0	0	0	0	1	0	1	1
86:	0	1	0	1	0	1	1	0
11 \wedge 86:	0	1	0	1	1	1	0	1

→ 93

Operatori pe biți

$\sim 11 = ?$

$\sim 0 = ?$

11:

$y = \sim 11$:

0	0	0	0	1	0	1	1
1	1	1	1	0	1	0	0

Negativ; aflăm $|y|$



Operatori pe biți

$\sim 11 = ?$

$\sim 0 = ?$

11:

$y = \sim 11$:

Scădem 1

Trecem la complement

0	0	0	0	1	0	1	1
1	1	1	1	0	1	0	0

Negativ; aflăm $|y|$



Operatori pe biți

$\sim 11 = ?$

$\sim 0 = ?$

11:

$y = \sim 11$:

Scădem 1

Trecem la complement

0	0	0	0	1	0	1	1
1	1	1	1	0	1	0	0
1	1	1	1	0	0	1	1

Negativ; aflăm $|y|$



Operatori pe biți

$$\sim 11 = ?$$

$$\sim 0 = ?$$

11:

$y = \sim 11$:

Scădem 1

Trecem la complement

0	0	0	0	1	0	1	1
1	1	1	1	0	1	0	0
1	1	1	1	0	0	1	1
0	0	0	0	1	1	0	0

Negativ; aflăm $|y|$

$|y| = 12$

$$\sim 11 = -12$$

(avem $\text{repr}(-x) = \text{not}(\text{repr}(x-1)) \Rightarrow \text{repr}(-12) = \text{not}(\text{repr}(11))$)

Operatori pe biți

`11 >> 1 = ?`

`11 << 1 = ?`

Operatori pe biți

`11 >> 1 = 5`

`11 << 1 = 22`

`x >> k` = parte întreagă inferioară `x / 2k` (`x div 2k`)

`x << k` = `x * 2k`

Probleme

- ▶ Se dă un număr natural x . Să se calculeze $x \div 4$ și $x * 6$ folosind operatori pe biți

Probleme

- ▶ Se dă un număr natural x . Să se calculeze $x \div 4$ și $x * 6$ folosind operatori pe biți

```
print(x>>2)
```

```
print((x<<2) + (x<<1))
```

Probleme

- ▶ Se dă un număr întreg x . Să se determine dacă acesta este par sau impar folosind operatori pe biți.

Probleme

- ▶ Se dă un număr întreg x . Să se determine dacă acesta este par sau impar folosind operatori pe biți.

$$x = \overline{a_1 \dots a_n}_{(2)} \text{ par} \Leftrightarrow x \bmod 2 = 0 \Leftrightarrow a_n = 0$$

- ⇒ Trebuie să testăm dacă ultimul bit din reprezentarea lui x este 0

Probleme

- ▶ Se dă un număr întreg x . Să se determine dacă acesta este par sau impar folosind operatori pe biți.

$$x = \overline{a_1 \dots a_n}_{(2)} \text{ par} \Leftrightarrow x \bmod 2 = 0 \Leftrightarrow a_n = 0$$

⇒ Trebuie să testăm dacă ultimul bit din reprezentarea lui x este 0:

$$x \& 1 = a_n \& 1 = a_n = \text{ultimul bit}$$

$$x = a_1 \dots a_{n-1} a_n$$

$$1 = 0 \dots 0 \ 1$$

$$x \& 1 = 0 \dots 0 \ a_n = a_n$$

Probleme

- ▶ Se dă un număr întreg x . Să se determine dacă acesta este par sau impar folosind operatori pe biți.

```
#testam daca ultimul bit din reprezentarea binara este 0 sau 1
if x&1 == 0:
    print("par")
else:
    print("impar")
```

Probleme

- ▶ Se dau 2 numere naturale x și k . Să se afișeze al k -lea bit din dreapta din reprezentarea binară a lui x

Probleme

- ▶ Se dau 2 numere naturale x și k . Să se afișeze al k -lea bit din dreapta din reprezentarea binară a lui x

```
print( (x >> (k-1)) & 1)
```

Probleme

- ▶ Se citesc 2 numere întregi x și k . Să se afișeze numărul obținut din x astfel:
 - se setează bitul numărul k la valoarea 1
 - se setează bitul k la valoarea 0
 - se complementează bitul k

Probleme

- ▶ Se citesc 2 numere întregi x și k . Să se afișeze numărul obținut din x astfel:
 - se setează bitul numărul k la valoarea 1

Probleme

- ▶ Se citesc 2 numere întregi x și k . Să se afișeze numărul obținut din x astfel:
 - se setează bitul numărul k la valoarea 1

$$x = a_1 \dots a_{n-k} a_{n-(k-1)} a_{n-(k-2)} \dots a_n$$

$$m =$$

$x \text{ op } m$



Probleme

- ▶ Se citesc 2 numere întregi x și k . Să se afișeze numărul obținut din x astfel:
 - se setează bitul numărul k la valoarea 1

$$x = a_1 \dots a_{n-k} a_{n-(k-1)} a_{n-(k-2)} \dots a_n$$

$$m = 0 \dots 0 \quad 1 \quad 0 \dots 0$$

$$x|m = a_1 \dots a_{n-k} \quad 1 \quad a_{n-(k-2)} \dots a_n$$



$m = ?$

Probleme

- ▶ Se citesc 2 numere întregi x și k . Să se afișeze numărul obținut din x astfel:
 - se setează bitul numărul k la valoarea 1

$$x = a_1 \dots a_{n-k} a_{n-(k-1)} a_{n-(k-2)} \dots a_n$$

$$m = 0 \dots 0 \quad 1 \quad 0 \dots 0$$

$$x|m = a_1 \dots a_{n-k} \quad 1 \quad a_{n-(k-2)} \dots a_n$$

$$m = 2^{k-1} = (1 \ll (k-1))$$

Probleme

- ▶ Se citesc 2 numere întregi x și k . Să se afișeze numărul obținut din x astfel:
 - se setează bitul numărul k la valoarea 1

```
print(x | (1 << (k-1)))
```

Probleme

- ▶ Se citesc 2 numere întregi x și k . Să se afișeze numărul obținut din x astfel:
 - se setează bitul numărul k la valoarea 0

$$x = a_1 \dots a_{n-k} a_{n-(k-1)} a_{n-(k-2)} \dots a_n$$

$$m =$$

Probleme

- ▶ Se citesc 2 numere întregi x și k . Să se afișeze numărul obținut din x astfel:
 - se setează bitul numărul k la valoarea 0

$$x = a_1 \dots a_{n-k} a_{n-(k-1)} a_{n-(k-2)} \dots a_n$$

$$m = 1 \dots 1 \quad 0 \quad 1 \dots 1$$

$$x \& m = a_1 \dots a_{n-k} \quad 0 \quad a_{n-(k-2)} \dots a_n$$



$m = ?$

Probleme

- ▶ Se citesc 2 numere întregi x și k . Să se afișeze numărul obținut din x astfel:
 - se setează bitul numărul k la valoarea 0

$$x = a_1 \dots a_{n-k} a_{n-(k-1)} a_{n-(k-2)} \dots a_n$$

$$m = 1 \dots 1 \quad 0 \quad 1 \dots 1$$

$$x \& m = a_1 \dots a_{n-k} \quad 0 \quad a_{n-(k-2)} \dots a_n$$

$$m = \sim 2^{k-1} = \sim (1 \ll (k-1))$$

Probleme

- ▶ Se citesc 2 numere întregi x și k . Să se afișeze numărul obținut din x astfel:

- se setează bitul numărul k la valoarea 0

`print(x & (~ (1 << (k-1))))`

Probleme

- ▶ Se citesc 2 numere întregi x și k . Să se afișeze numărul obținut din x astfel:
 - se completează bitul k

Probleme

- ▶ Se citesc 2 numere întregi x și k . Să se afișeze numărul obținut din x astfel:
 - se complementează bitul k

$$x = a_1 \dots a_{n-k} a_{n-(k-1)} a_{n-(k-2)} \dots a_n$$

$$m = 0 \dots 0 \quad 1 \quad 0 \dots 0$$

$$x \wedge m = a_1 \dots a_{n-k} \sim a_{n-(k-1)} a_{n-(k-2)} \dots a_n$$

Probleme

- ▶ Se citesc 2 numere întregi x și k . Să se afișeze numărul obținut din x astfel:

- se setează bitul numărul k la valoarea 1

`print(x | (1 << (k-1)))`

- se setează bitul k la valoarea 0

`print(x & (~ (1 << (k-1))))`

- se complementează bitul k

`print(x ^ (1 << (k-1)))`

Probleme

- ▶ Să se verifice dacă un număr natural x este de forma 2^k

Probleme

- ▶ Să se verifice dacă un număr natural x este de forma 2^k

$$x = a_1 \dots a_{k-1} 1 0 0 \dots 0_{(2)}$$

Probleme

- ▶ Să se verifice dacă un număr natural x este de forma 2^k

$$x = a_1 \dots a_{k-1} 1 0 0 \dots 0_{(2)}$$

$$x - 1 =$$

Probleme

- ▶ Să se verifice dacă un număr natural x este de forma 2^k

$$x = a_1 \dots a_{k-1} 1 0 0 \dots 0_{(2)}$$

$$x - 1 = a_1 \dots a_{k-1} 0 1 1 \dots 1_{(2)}$$

$$x \& (x - 1) =$$

Probleme

- ▶ Să se verifice dacă un număr natural x este de forma 2^k

$$x = a_1 \dots a_{k-1} 1 0 0 \dots 0_{(2)}$$

$$x - 1 = a_1 \dots a_{k-1} 0 1 1 \dots 1_{(2)}$$

$$x \& (x - 1) = a_1 \dots a_{k-1} 0 0 0 \dots 0_{(2)}$$

Probleme

- ▶ Să se verifice dacă un număr natural x este de forma 2^k

$$x = a_1 \dots a_{k-1} 1 0 0 \dots 0_{(2)}$$

$$x - 1 = a_1 \dots a_{k-1} 0 1 1 \dots 1_{(2)}$$

$$x \& (x - 1) = a_1 \dots a_{k-1} 0 0 0 \dots 0_{(2)}$$

$$x \& (x - 1) = 0 \Leftrightarrow x = 1 0 0 \dots 0_{(2)} \Leftrightarrow x \text{ este putere a lui } 2$$

Probleme

- ▶ Să se verifice dacă un număr natural x este de forma 2^k

```
print( x & (x-1) == 0 )
```


Operatorul xor

- ▶ asociativ, comutativ
- ▶ $a \wedge 0 = a$ pentru $a \in \{0, 1\}$
- ▶ $a \wedge 1 = \sim a$ pentru $a \in \{0, 1\}$
- ▶ $a \wedge b = 1 \Leftrightarrow a \neq b$ pentru $a, b \in \{0, 1\}$
- ▶ $x \wedge x = 0$

Operatorul xor

- ▶ Să se interschimbe conținutul a două variabile x și y folosind xor

Operatorul xor

- ▶ Să se interschimbe conținutul a două variabile x și y folosind xor:

Avem

$$(x \oplus y) \oplus y = x$$

$$(x \oplus y) \oplus x = y$$

Operatorul xor

- ▶ Să se interschimbe conținutul a două variabile x și y folosind xor:

Avem

$$(x \oplus y) \oplus y = x$$

$$(x \oplus y) \oplus x = y$$

Notam $aux = x \oplus y$

Atunci intershimbarea se poate face astfel:

$$y = aux \oplus y$$

$$x = aux \oplus x$$

Operatorul xor

- ▶ Să se interschimbe conținutul a două variabile x și y folosind xor:

Avem

```
aux = x ^ y
```

```
y = aux ^ y  $\Rightarrow$  y devine x
```

```
x = aux ^ x
```



Putem renunța la aux?

Operatorul xor

- ▶ Să se interschimbe conținutul a două variabile x și y folosind xor:

Avem

$\text{aux} = x \oplus y$

$y = \text{aux} \oplus y \Rightarrow y \text{ devine } x$

$x = \text{aux} \oplus x$

Putem renunța la aux? – Da, îl folosim pe x

$x = x \oplus y$

$y = x \oplus y \Rightarrow y \text{ devine vechiul } x$

$x = x \oplus y$

 y este vechiul x

Operatorul xor

- ▶ Să se interschimbe conținutul a două variabile x și y folosind xor:

Avem

$\text{aux} = x \oplus y$

$y = \text{aux} \oplus y \Rightarrow y \text{ devine } x$

$x = \text{aux} \oplus x$

Putem renunța la aux? – Da, îl folosim pe x

$x = x \oplus y$

$y = x \oplus y$

$x = x \oplus y$

Exemplu

$x = 75, y = 20$

$x = x \oplus y = 75 \oplus 20$

$y = x \oplus y = (75 \oplus 20) \oplus 20 = 75$

$x = x \oplus y = (75 \oplus 20) \oplus 75 = 20$

y este vechiul x

y este egal cu valoarea initiala a lui x

Probleme

- ▶ Se dau două numere naturale x și y . Calculați numărul biților din reprezentarea binară internă a numărului x a căror valoare trebuie comutată pentru a obține numărul y .

Probleme

- ▶ Se dau două numere naturale x și y . Calculați numărul biților din reprezentarea binară internă a numărului x a căror valoare trebuie comutată pentru a obține numărul y .



Se calculează numărul biților nenuli din $x \wedge y$ –
implementare la laborator

Probleme

- ▶ Se citește un șir format din numere naturale cu proprietatea că fiecare valoare distinctă apare de exact două ori în șir, mai puțin una care apare o singură dată. Să se afișeze valoarea care apare o singură dată în șir.

Probleme

- ▶ Se citește un șir format din numere naturale cu proprietatea că fiecare valoare distinctă apare de exact două ori în șir, mai puțin una care apare o singură dată. Să se afișeze valoarea care apare o singură dată în șir.



Fie x_1, \dots, x_n numerele din șir

Fie $v = x_1 \wedge \dots \wedge x_n$

Probleme

- ▶ Se citește un șir format din numere naturale cu proprietatea că fiecare valoare distinctă apare de exact două ori în șir, mai puțin una care apare o singură dată. Să se afișeze valoarea care apare o singură dată în șir.



Fie x_1, \dots, x_n numerele din șir

Fie $v = x_1 \wedge \dots \wedge x_n$

Deoarece \wedge este asociativ și comutativ și $x \wedge x = 0$, xor pentru valorile care apar de 2 ori este 0, deci v este egal cu numărul care apare o singură dată

Probleme – TEMA

- ▶ Să se calculeze numărul obținut prin aplicarea operatorului XOR între toate elementele tuturor submulțimilor unei mulțimi nevide de numere naturale $A = \{a_1, \dots, a_n\}$ mai puțin mulțimea vidă. De exemplu, pentru mulțimea $A = \{2, 7, 4\}$ trebuie afișată valoarea

$$v = (2) \wedge (7) \wedge (4) \wedge (2 \wedge 7) \wedge (2 \wedge 4) \wedge (7 \wedge 4) \wedge (2 \wedge 7 \wedge 4) = 0$$

Probleme

- ▶ Să se calculeze numărul obținut prin aplicarea operatorului XOR între toate elementele tuturor submulțimilor unei mulțimi nevide de numere naturale $A = \{a_1, \dots, a_n\}$ mai puțin mulțimea vidă. De exemplu, pentru mulțimea $A = \{2, 7, 4\}$ trebuie afișată valoarea

$$v = (2) \wedge (7) \wedge (4) \wedge (2 \wedge 7) \wedge (2 \wedge 4) \wedge (7 \wedge 4) \wedge (2 \wedge 7 \wedge 4) = 0$$



• De câte ori apare o valoare în v ?

Probleme

- ▶ Să se calculeze numărul obținut prin aplicarea operatorului XOR între toate elementele tuturor submulțimilor unei mulțimi nevide de numere naturale $A = \{a_1, \dots, a_n\}$ mai puțin mulțimea vidă. De exemplu, pentru mulțimea $A = \{2, 7, 4\}$ trebuie afișată valoarea

$$v = (2) \wedge (7) \wedge (4) \wedge (2 \wedge 7) \wedge (2 \wedge 4) \wedge (7 \wedge 4) \wedge (2 \wedge 7 \wedge 4) = 0$$



- De 2^{n-1} ori \Rightarrow pentru $n > 1$ apare de un număr par de ori $\Rightarrow v = 0$
- Pentru $n = 1$??

Probleme – laborator

- ▶ Scrieți un program care testează dacă un număr natural n citit de la tastatură este de forma 2^k și, în caz afirmativ, afișează k
- ▶ Scrieți un program care determină numărul de biți egali cu 1 din reprezentarea binară a unui număr natural n citit de la tastatură