

Universitatea Tehnică din Cluj-Napoca
Departamentul Calculatoare



- Exempu

- $$157_{(10)} = 10011101_{(2)}$$

- Exemplan

$$10011101_{(2)} = 1 \cdot 2^0 + 0 \cdot 2^1 + 1 \cdot 2^2 + 1 \cdot 2^3 + 1 \cdot 2^4 + 0 \cdot 2^5 + 0 \cdot 2^6 + 1 \cdot 2^7 = 157_{(10)}$$



- Exempu

Diagram illustrating the conversion of binary numbers to decimal:

- Binary 1001 is converted to decimal 9.
- Binary 1101 is converted to decimal D (11).

- | | |
|---|---|
| 1 | 0 |
| 9 | D |

$$9D_{(16)} = 13 \cdot 16^0 + 9 \cdot 16^1 = 157_{(10)}$$



- Exempu

Diagram illustrating the mapping of binary strings to decimal values:

- Binary string **10** maps to decimal value **2**.
- Binary string **011** maps to decimal value **3**.
- Binary string **101** maps to decimal value **5**.

The label **(2)** is shown to the right of the boxes.

- Exempu

2	1	0
2	3	5

$$235_{(8)} = 5 \cdot 8^0 + 3 \cdot 8^1 + 2 \cdot 8^2 = 157_{(10)}$$





- 6



- $$X = -a_{n-1} \cdot 2^{n-1} + \sum_{k=0}^{n-2} a_k \cdot 2^k$$

- | | | | | | | |
|----------------|-----|-----|-----|-----|-----|-------|
| a_{n-1} | ... | ... | ... | ... | ... | a_0 |
| Valoare | | | | | | |

- $$X = \sum_{k=0}^{n-1} a_k \cdot 2^k$$



- | | | |
|-------------|---|---|
| a_{n-1} | $a_{n-2} \quad \dots \quad \dots \quad \dots \quad a_k$ | $a_{k-1} \quad \dots \quad \dots \quad \dots \quad a_0$ |
| Semn | Exponent | Mantisă |

- $$X = (-1)^{semn} \times 2^{exponent-deplasament} \times 1.mantisă$$

- 1 bit *semn*, 8 biți *exponent*, 23 biți *mantisă*
- *deplasament* = 127; în baza 2: 01111111

- 1 bit *semn*, 11 biți *exponent*, 52 biți *mantisă*
- *deplasament* = 1023; în baza 2: 01111111111



- | | | |
|----------|--|---|
| a_{31} | a_{30} a_{23} | a_{22} a_0 |
| 0 | 10000110 | 00001001001000111101100 |

- [illegible]

- 9



Reprezentarea internă a datelor (6)

- Consecințe
 - Numerele întregi
 - Sunt reprezentate exact
 - O valoare întreagă care depășește un capăt al domeniului de reprezentare al numerelor întregi se găsește la extrema cealaltă a domeniului la o distanță egală cu valoarea depășită
 - Numerele reale
 - Nu pot fi întotdeauna reprezentate exact
 - O valoare reală care depășește capătul inferior al domeniului de reprezentare este tratată ca fiind **minus infinit**
 - O valoare reală care depășește capătul superior al domeniului de reprezentare este tratată ca fiind **plus infinit**



Reprezentarea internă a datelor (7)

- Consecințe – exemplu

```
#include <stdio.h>
#include <float.h>
int main() {
    int x = 0x7FFFFFFF, y = x+10;
    printf("%d %d\n", x, y); //2147483647 -2147483639
    double a = 0.1, b = 0.2, c = 0.3;
    printf("%.20f %.20f %.20f\n", a, b, c);
    //0.100000000000000001000 0.200000000000000001000 0.29999999999999999000
    printf("%d %d\n", a+b==c, c==c); // 0 1
    float e = 30000000, f = 1, g = e+f;
    printf("%.15f %.15f %.15f\n", e, f, g);
    //30000000.0000000000000000 1.0000000000000000 30000000.0000000000000000
    double t=DBL_MAX, u=t*1.000001;
    printf("%f", u); //inf
    return 0;
}
```



- 12



- 13





Operatori (2)

- Clase de operatori (continua)
- De conversie de tip **(tip) operand**
- De dimensiune
 - **sizeof(tip)**
 - **sizeof(operand)**
- Operatorul adresă **&operand**
- Operatorul de dereferențiere ***operand**
- Operatorii paranteză **() []**
- Operatorul condițional **?**
 - **expresie ? expresie1 : expresie2**
- Operatorul virgulă **,**
- Operatorii de acces la elementele unei structuri sau uniuni
 - **.** cand operandul este o structură sau uniune
 - **->** cand operandul este pointer la o structură sau uniune (adresa de memorie unde este stocată acea structură sau uniune)



```
int a, b=-5; // b este -5; a este doar declarat
int a=+b;    // a este acum tot -5
```

- ```
int a=13, b=5;
int c=a+b; // c este 18
c=a-b; // c este 8
c=a/b; // c este 2 (câtul împărțirii)
c=a%b; // c este 3 (restul împărțirii)
 // 13/5 = 2 rest 3
float d=a/b; // d este 2
d=a/5.0f; // d este 2.6
```





## Operatori aritmetici (2)

- ```
/* y si z sunt cei din exemplul anterior */
float t=y+z; //t este nan
float w=y*z; //w este -inf
float s=y/z; //s este nan
```



Operatori relaționali și de egalitate

- Operatorii mai mic **<** mai mic sau egal **<=** mai mare **>** mai mare sau egal **>=**
- Operatorii egal **==** diferit **!=**
- Rezultatul expresiei este o valoare logică
 - 0 dacă este fals
 - 1 dacă este adevărat

```
int a=13;
double b=3.5;
int c=a<b; //c este 0
c=(a>=b); //c este 1
c=(a==b); //c este 0
c=(a!=b); //c este 1
c=(a>b>2.7); /* c este 0, a>b se evalueaza mai intai
               la 1 iar apoi se evalueaza expresia
               1>2.7 la fals, adica 0 */
```



- 19



- ```
int a=150;
float b=-2.5f;
int c=!b; //c este 0
int d=a&&c; //d este 0
int e=a||c; //e este 1
int f=a&&d&&(d=b); /* f este 0, d rămâne 0,
 expresia d=b nu s-a
 mai evaluat */
```



- ```
int a=13; // reprezentarea pe biti este 000...01101
int b=10; // reprezentarea pe biti este 000...01010
int c=~a; /* c este in baza 2 : 111...10010
           c este in baza 16: FFFFFFFF2
           c este in baza 10: -14          */
int d=a&b; // d este in baza 2: 000...01000; in baza 10: 8; in baza 16: 8
int e=a|b; // e este in baza 2: 000...01111; in baza 10: 15; in baza 16: F
int f=a^b; // f este in baza 2: 000...00111; in baza 10: 7; in baza 16: 7
```



- Example

```
int 13=a; // nu este permis !!!
int a=13; // corect, a este 13
int b=10; // b este 10
int c=b; // c este 10
int d=c=a+2; //c este 15, d este tot 15
```



- ```
int a=13; // a este 13
a*=10; // echivalent cu a=a*10; a este 130
a<<=1; // echivalent cu a=a<<1; a este 260
a|=5; // echivalent cu a=a|5; a este 261
```







# Operatori de incrementare și decrementare (1)

- Operatorul de incrementare **++**
  - Realizează incrementarea unei variabile cu 1 și returnează valoarea dinainte sau de după incrementare astfel:
  - Forma prefixă **++i**
    - Pre-incrementează variabila i cu 1
    - Returnează valoarea **i+1**, adică valoarea incrementată
  - Forma postfixă **i++**
    - Returnează valoarea **i**, adică valoarea dinainte de incrementare
    - Post-incrementează variabila i cu 1
- Exemple

```
int a=13;
int b=a++; // b este 13, iar a este 14
int c=++a; // c este 15, iar a este tot 15
a++; // a este 16
++a; // a este 17
```



# Operatori de incrementare și decrementare (2)

- Operatorul de decrementare **--**
  - Realizează decrementarea unei variabile cu 1 și returnează valoarea dinainte sau de după decrementare astfel:
  - Forma prefixă **--i**
    - Pre-decrementează variabila i cu 1
    - Returnează valoarea **i-1**, adică valoarea decrementată
  - Forma postfixă **i--**
    - Returnează valoarea **i**, adică valoarea dinainte de decrementare
    - Post-decrementează variabila i cu 1
- Exemple

```
int a=15, b=10, c;
a--; //a este 14
--a; //a este 13;
c = b--; //c este 10, b este 9
c = ++a - b--; //a este 14, c este 5, b este 8
c = a-- + ++b; //b este 9, c este 23, a este 13
```



- (tip) x**

```
int a=27, b=10;
float c = a/b; //c este 2
c = a / (float) b; //c este 2.7
b = c; // conversie implicită la tipul int; b este 2
b = (int) c; // conversie explicită de tip; b este 2
float d = a + c; // d este 29.7
double e = ((double)a) / 2; // e este 13.5
double e = a / 2.0; // e este 13.5
```



# Operatorul de dimensiune

- Operatorul **sizeof**
  - **sizeof(tip)**
    - Returnează numărul de octeți ocupați în memorie de o variabilă de tipul **tip**
  - **sizeof(operand)**
    - Returnează numărul de octeți ocupați în memorie de **operand**
- Exemple

```
int a=27;
float b[10];
double c[2][15];
a = sizeof(a); // a este 4
a = sizeof(char); // a este 1
a = sizeof(float); // a este 4
a = sizeof(double); // a este 8
a = sizeof(b); // a este 40
a = sizeof(c); // a este 240
```





# Operatorii paranteză

- Paranteze rotunde ( )

- Realizează gruparea unor expresii cu scopul de a fi evaluate înaintea altor expresii
- Exemple

```
int a=27, b=3;
float c=10;
float d = a - b / c; // d este 26.7
d = (a - b) / c; // d este 2.4
```

- Paranteze drepte [ ]

- Permit declararea de tablouri și accesul la elementele acestora
- Exemple

```
int a[5]={40,-20};
char b[5][2]={{'A','b'},{'0','a'}};
int c = a[1]; // c este -20
char d = b[0][1]; // d este 'b' sau 98
d = b[1][0]; // d este '0' sau 48
int e = b[1][1]; // e este 'a' sau 97
```



**expresie ? expresie1 : expresie2**

- Example

31



# Operatorul virgulă

- Permite evaluarea secvențială a unei expresii compuse din mai multe expresii separate prin virgule
  - Evaluarea se face de la stânga la dreapta
  - Valoarea ultimei expresii din înlanțuire este valoarea expresiei compuse
- Exemple

```
int a=7;
char b='C';
float c=3.9;
a = (c/2, b++, b-2);
printf("a=%d\nb=%c\nc=%f", a, b, c);
/* se afiseaza:
 a=66
 b=D
 c=3.9 */
```





- 33

34

35



- Valoarea expresiei din dreapta operatorului de atribuire se convertește automat la tipul identificatorului din stânga
  - Conversia de la un tip real la un tip întreg se face prin trunchiere (tăierea părții zecimale de după virgulă) !
  - Se poate întâmpla ca tipul de date la care se face conversia să nu poată reprezenta valoarea convertită !

- Conversia implicită a operanzilor implicați într-o operație se face la tipul de date a celui cu rang mai înalt din ierarhie (care poate reprezenta ambele valori)

- ```
int i=27, j=10;
float u = i / j; // u este 2 de tip float
double v = i / (float) j; // v este 2.7 de tip double
printf("%g, %g", u, v); // 2, 2.7
```



- ```
unsigned int a = 3000000000;
int b = 1;
unsigned int c = a + b;
printf("%u\n", c); // 3000000001
```

- ```
char x=120;
char y=110;
int z = x + y;
printf("%d\n", z); // 230
```