



Programare procedurala

- suport de curs -

Dobrovat Anca - Madalina

An universitar 2016 – 2017
Semestrul I

Curs 4



Agenda cursului

1. **Scurta recapitulare a cursului anterior: Fundamentele limbajului C:**
Reprezentarea datelor in memorie (signed vs. unsigned);
Expresii si operatori – evaluare, precedenta;
Operatorii pe biti;
Functii de citire / scriere cu format;
2. **Complexitatea algoritmilor – notiuni introductive (2)**

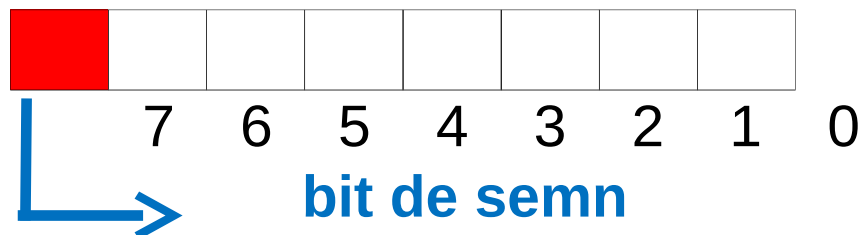


Fundamentele limbajului C

Reprezentarea datelor in memorie

Bitul de semn: - cel mai semnificativ bit – (0 pentru numere pozitive; 1 pentru numere negative)

Exemplificare – reprezentare pe 1 octet ([-128, 127])



1. Numere pozitive

- Transformare in baza 2
- Completare bit de semn
- Completare rest biti cu 0

n = 47

0	0	1	0	1	1	1	1
---	---	---	---	---	---	---	---



Fundamentele limbajului C

Reprezentarea datelor in memorie

2. Numere negative

$n = -47$

- Transformare in baza 2 a valorii absolute

|47|

0	0	1	0	1	1	1	1
----------	----------	----------	----------	----------	----------	----------	----------

- Calculul complementului fata de 1 a reprezentarii obtinute la pasul anterior (bitul 1 devine 0, iar bitul 0 devine 1)

1	1	0	1	0	0	0	0
----------	----------	----------	----------	----------	----------	----------	----------

- se aduna 1 (adunarea se face in baza 2) la valoarea obinuta

1	1	0	1	0	0	0	0
----------	----------	----------	----------	----------	----------	----------	----------

+

1

1	1	0	1	0	0	0	1
----------	----------	----------	----------	----------	----------	----------	----------



Fundamentele limbajului C

Reprezentarea datelor in memorie

Diferenta dintre intregii cu semn si fara semn → interpretarea bitului cu ordinul cel mai mare (indicator de semn).

0 – numere pozitive / 1 – numere negative

Exemplu

int x = 190;

0 1 0 1 1 1 1 1 0

int x = -190;

E de-ajuns sa schimbam valoarea bitului de semn? **NU**

~~1 0 1 0 1 1 1 1 1 0~~



Fundamentele limbajului C

Reprezentarea datelor in memorie

$$190 + (-190) = 0$$

0 1 0 1 1 1 1 1 0

+

1 0 1 0 0 0 0 1 0

=

0 0

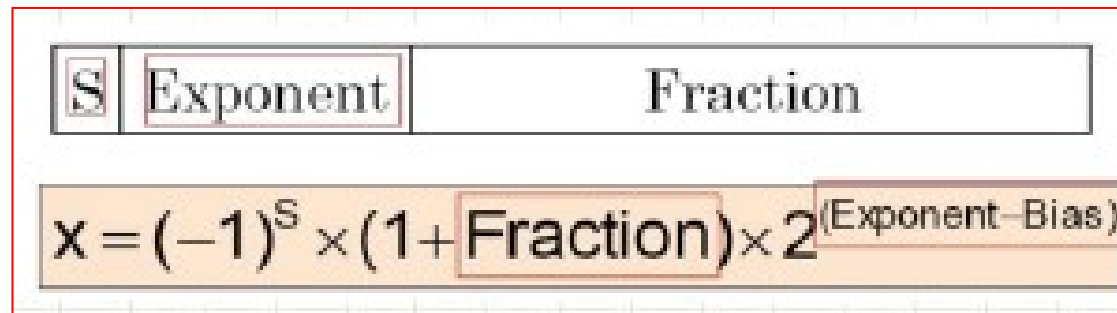


Fundamentele limbajului C

Reprezentarea datelor in memorie

Datele de tip float

Reprezentare pe 32 biti (semn + exponent + mantisa) **(1+8+23)**



Bias = 127 pentru tipul float



Fundamentele limbajului C

Reprezentarea datelor in memorie

Exemplu: float x = 14.0

$$14.0 = 1.75 \times 8$$

$$14.0 = 1.75 \times 2^{(130 - 127)}$$

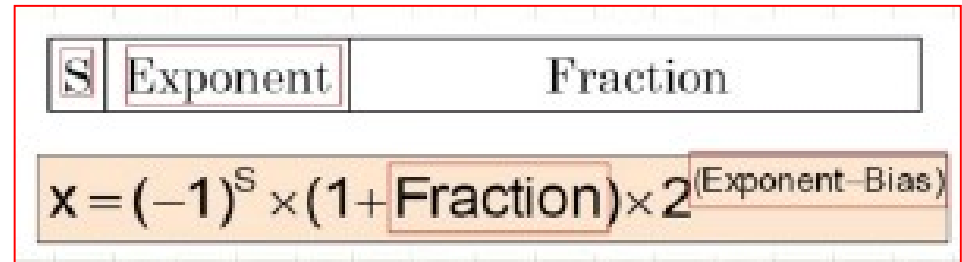
$$14.0 = (-1)^0 \times 1.75 \times 2^{(130 - 127)}$$

$$14.0 = (-1)^0 \times (1 + 0.75) \times 2^{(130 - 127)}$$

$$130 = 2^7 + 2^1$$

$$0.75 = 0.5 + 0.25 = 2^{-1} + 2^{-2}$$

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





Operatii pe biti

1. Negatia \sim (operator unar)

- intoarce numarul intreg a carui reprezentare interna se obtine din reprezentarea interna a numarului initial, prin complementarea fata de 1 a fiecarui bit ($1 \rightarrow 0$ si $0 \rightarrow 1$). Expl. $\sim (-47)$

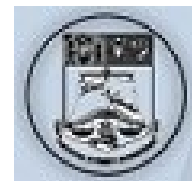
- 47

1	1	0	1	0	0	0	1
---	---	---	---	---	---	---	---

Complementare fata de 1

46

0	0	1	0	1	1	1	0
---	---	---	---	---	---	---	---



Operatii pe biti

2. Conjunctia & (operator binar)

- intoarce numarul întreg a carui reprezentare interna se obtine prin conjunctia bitilor care apar in reprezentarea interna a operanzilor. Conjunctia se face cu toate perechile de biti situati pe aceeasi pozitie.

Expl. **47** & **28** = **12**

47

0	0	1	0	1	1	1	1
----------	----------	----------	----------	----------	----------	----------	----------

28

0	0	0	1	1	1	0	0
----------	----------	----------	----------	----------	----------	----------	----------

12

0	0	0	0	1	1	0	0
----------	----------	----------	----------	----------	----------	----------	----------

Bit 1	Bit 2	Bit 1 & Bit 2
0	0	0
0	1	0
1	0	0



Operatii pe biti

3. Disjunctia | (operator binar)

- intoarce numarul întreg a carui reprezentare interna se obtine prin disjunctia bitilor care apar in reprezentarea interna a operanzilor.
Disjunctia se face cu toate perechile de biti situati pe aceeasi pozitie.

Expl. **47** | **28** = **63**

47

0	0	1	0	1	1	1	1
---	---	---	---	---	---	---	---

28

0	0	0	1	1	1	0	0
---	---	---	---	---	---	---	---

63

0	0	1	1	1	1	1	1
---	---	---	---	---	---	---	---

Bit 1	Bit 2	Bit 1 Bit 2
0	0	0
0	1	1
1	0	1
1	1	1



Fundamentele limbajului C

Operatii pe biti

4. Xor ("sau exclusiv") \wedge (operator binar)

- intoarce numarul întreg a carui reprezentare interna se obtine prin disjunctia bitilor care apar in reprezentarea interna a operanzilor. Disjunctia se face cu toate perechile de biti situati pe aceeasi pozitie.

Expl. $47 \wedge 28 = 51$

47

0	0	1	0	1	1	1	1
---	---	---	---	---	---	---	---

28

0	0	0	1	1	1	0	0
---	---	---	---	---	---	---	---

51

0	0	1	1	0	0	1	1
---	---	---	---	---	---	---	---

Bit 1	Bit 2	Bit 1 \wedge Bit 2
0	0	0
0	1	1
1	0	1
1	1	0



Operatii pe biti

5. shl ("shift left") << (operator binar)

- intoarce numarul intreg a carui reprezentare interna se obtine din reprezentarea interna a primului operand prin deplasare la stanga cu un numar de biti egal cu al doilea operand.

Expl. $10 \ll 3 = 80$ ($10 * 2^3 = 80$)

10

0	0	0	0	1	0	1	0
---	---	---	---	---	---	---	---

80

0	1	0	1	0	0	0	0
---	---	---	---	---	---	---	---



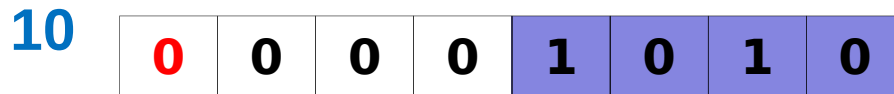
Fundamentele limbajului C

Operatii pe biti

6. shr ("shift right") >> (operator binar)

- intoarce numarul intreg a carui reprezentare interna se obtine din reprezentarea interna a primului operand prin deplasare la dreapta cu un numar de biti egal cu al doilea operand.

Expl. $10 \gg 3 = 1$ ($10 / 2^3 = 1$)



~~0 1 0~~



Fundamentele limbajului C

Operatii pe biti

1. Transformarea unui bit in 1 (X or (1 shl B))

- Cerinta: valorii X sa i se seteze la valoarea 1, bitul B ($0 \leq B \leq 7$).
 - Rezolvare:
 - o **masca logica M** in care doar bitul B are valoarea 1. Valoarea mastii M e data de **1 shl B**
 - operatia OR intre X si M
- Expl. **X = 37, B = 4** (bitului 4 i se va seta valoarea 1)

37

0	0	1	0	0	1	0	1
---	---	---	---	---	---	---	---

 OR

M.

0	0	0	1	0	0	0	0
---	---	---	---	---	---	---	---

Rezultat

0	0	1	1	0	1	0	1
---	---	---	---	---	---	---	---



Fundamentele limbajului C

Operatii pe biti

2. Transformarea unui bit in 0 ($X \text{ and } (255 - (1 \text{ shl } B))$)

- Cerinta: valorii X sa i se seteze la valoarea 0, bitul B ($0 \leq B \leq 7$).
 - Rezolvare:
 - o **masca logica M** in care doar bitul B are valoarea 0. Valoarea mastii M e data de **$255 - (1 \text{ shl } B)$** [sau **$255 \text{ xor } (1 \text{ shl } B)$**]
 - operatia AND intre X si M
- Expl. **$X = 37$** , **$B = 2$** (bitului 2 i se va seta valoarea 1)

37

0	0	1	0	0	1	0	1
---	---	---	---	---	---	---	---

AND

M

1	1	1	1	1	0	1	1
---	---	---	---	---	---	---	---

Rezultat

0	0	1	0	0	0	0	1
---	---	---	---	---	---	---	---



Operatii pe biti

3. Testarea valorii unui bit ($X \& (1 \ll B) \neq 0$)

- Cerinta: ce valori au bitii B1 si B2 ai valorii X?
 - Rezolvare:
 - o **masca logica M1** pentru bitul B1 si o **masca logica M2** pentru bitul B2 cu valoarea 1
 - operatia AND intre X si M1 respectiv inte X si M2
- Expl. $X = 37$, $B1 = 2$

37

0	0	1	0	0	1	0	1
---	---	---	---	---	---	---	---

M1

0	0	0	0	0	1	0	0
---	---	---	---	---	---	---	---

R1

0	0	0	0	0	1	0	0
---	---	---	---	---	---	---	---

AND

$$X \& (1 \ll 2) = 1$$

$$1 \neq 0 \text{ (adevarat)} \rightarrow 1$$



Operatii pe biti

3. Testarea valorii unui bit ($X \& (1 \ll B) \neq 0$)

- Cerinta: ce valori au bitii B1 si B2 ai valorii X?
 - Rezolvare:
 - o **masca logica M1** pentru bitul B1 si o **masca logica M2** pentru bitul B2 cu valoarea 1
 - operatia AND intre X si M1 respectiv inte X si M2
- Expl. $X = 37$, $B2 = 4$

37

0	0	1	0	0	1	0	1
---	---	---	---	---	---	---	---

M2

0	0	0	1	0	0	0	0
---	---	---	---	---	---	---	---

R2

0	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---

AND

$$X \& (1 \ll 4) = 0$$

$$0 \neq 0 \text{ (fals)} \rightarrow 0$$



Fundamentele limbajului C

Operatii pe biti

4. Testarea valorii ultimilor biti ($X \& (1 \ll B) - 1$)

- Cerinta: ce valori au ultimii B biti (egal cu restul impartirii la 2^B) ?

- Rezolvare:

- o **masca logica M** pentru bitii 0, 1, ... B-1 cu valoarea 1
- operatia AND intre X si M1

Expl. $X = 37$, $B = 3$ (ultimii 3 biti = restul impartirii la 8)

37	0	0	1	0	0	1	0	1
----	---	---	---	---	---	---	---	---

M	0	0	0	0	0	1	1	1
---	---	---	---	---	---	---	---	---

Rez	0	0	0	0	0	1	0	1
-----	---	---	---	---	---	---	---	---

AND

$$1 \ll B = 8 \rightarrow 1 \ll B - 1 = 7$$



Operatii pe biti

Aplicatii

1. Testarea paritatii unui numar ($X \& 1$)

$X = \text{par} \rightarrow$ cel mai din dreapta bit este 0

$X = \text{impar} \rightarrow$ cel mai din dreapta bit este 1

$X = 23$

0	0	0	1	0	1	1	1	&
0	0	0	0	0	0	0	1	
0	0	0	0	0	0	0	1	

$X = 44$

0	0	1	0	1	1	0	0	&
0	0	0	0	0	0	0	1	
0	0	0	0	0	0	0	0	



Fundamentele limbajului C

Operatii pe biti

Aplicatii

2. Se considera un numar natural n . Sa se verifice daca n este sau nu o putere a lui 2.

$X = \text{putere a lui } 2 \rightarrow \text{un singur bit nenul}$

Rezolvare fara numararea bitilor ($X \& (X - 1) = 0 \Leftrightarrow X = 2^k$)

$X = 32$

0	0	1	0	0	0	0	0
---	---	---	---	---	---	---	---

 &

$X - 1 = 31$

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

$X = 20$

0	0	0	1	0	1	0	0
---	---	---	---	---	---	---	---

 &

$X - 1 =$

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



Fundamentele limbajului C

Operatii pe biti

Aplicatii

3. Se considera un numar natural nenul n . Sa se determine numarul de cifre de 1 din reprezentarea lui binara.

Solutie 1: parcurgerea secventiala a bitilor

```
int i, n_1=0; // n_1 – contor pentru bitii de valoare 1
for ( i=0; i<32; i++)
if ( $n \& (1 \ll i)$ ) n_1++;
```

$X = 27$

0	0	0	1	1	0	1	1
---	---	---	---	---	---	---	---

32 x 2 operatii ($n \& (1 \ll i)$)



Fundamentele limbajului C

Operatii pe biti

Aplicatii

3. Se considera un numar natural nenul n . Sa se determine numarul de cifre de 1 din reprezentarea lui binara.

Solutie 2: folosirea $n \& (n - 1)$

Operatia anuleaza cel mai nesemnificativ bit de 1 a lui n .

```
int n_1=0;    // n_1 – contor pentru bitii de valoare 1
```

```
do {
```

```
n &= n-1;
```

Obs: se executa un numar de pasi egali cu numarul de cifre de 1 din reprezentare.

```
n_1++;
```

```
}while (n);
```

$X = 27$

0	0	0	1	1	0	1	1
---	---	---	---	---	---	---	---

&

$X - 1 = 26$

0	0	0	1	1	0	1	0
---	---	---	---	---	---	---	---

Rez

0	0	0	1	1	0	1	0
---	---	---	---	---	---	---	---



Fundamentele limbajului C

Operatii pe biti

Aplicatii

4. Se consideră două numere naturale n și i ($0 \leq i < 16$). Să se marcheze cu 1 bitul i al lui n .

Solutie: Setam valoarea 1 la bitul i , indiferent de valoarea memorată anterior (0 sau 1)

$$n \mid (1 \ll i)$$

I

$X = 19$

0	0	0	1	0	0	1	1
---	---	---	---	---	---	---	---

$i = 3 \rightarrow 1 \ll 3 = 8$

0	0	0	0	1	0	0	0
---	---	---	---	---	---	---	---

Rez

0	0	0	1	1	0	1	1
---	---	---	---	---	---	---	---



Expresii si operatori – evaluare, precedenta

Observatii finale

Expresii aritmetice si operatori aritmetici

- Obs:**
1. Se aplica operanzilor de tip intreg (int, char) sau real (float, double);
 2. Operatorul % reprezinta restul impartirii a doi **intregi**;
 3. Operatorul / returneaza catul impartirii (daca cei doi operanzi sunt intregi), sau impartirea reala in caz contrar.
 4. Operatorii / si % nu pot avea operandul din dreapta nul.

Exemple:

int a = +7, b = -3;	// + si – operatori unari
a = b + a – 1;	// + si – operatori binari
b = b * a / 3;	// * si / - operatori binari
int x = a % b;	// % - operator binar



Fundamentele limbajului C

Expresii si operatori – evaluare, precedenta

Observatii finale

Expresii aritmetice si operatori aritmetici

dacă într-o expresie apar mai mulți operatori, atunci evaluarea expresiei respectă **ordinea de precedență** a operatorilor

dacă într-o expresie apar mai mulți operatori de aceeași prioritate, atunci se aplică **regula de asociativitate** a operatorilor

Ordinea de precedenta:

Cei mai prioritari	+	(plus unar)
	-	(minus unar)
	*	(înmulțire)
	/	(împărțire)
	%	(restul împărțirii)
Cei mai puțin prioritari	+	(adunare)
	-	(scădere)



Fundamentele limbajului C

Expresii si operatori – evaluare, precedenta

Observatii finale

Exemple (Ordinea de precedenta a operatorilor aritmetici)

1. $-a + b / c$	\Leftrightarrow	$(-a) + (b/c)$
2. $-a * b - c$	\Leftrightarrow	$((-a) * b) - c$

Asociativitatea operatorilor aritmetici

1. **Unari (asociativi la dreapta)** – Gruparea de la dreapta la stanga
2. **Binari (asociativi la stanga)** – Gruparea de la stanga la dreapta

$- + a$	\Leftrightarrow	$-(+a)$
$+ - b$	\Leftrightarrow	$+(-b)$
$a + b - c + d$	\Leftrightarrow	$((a + b) - c) + d$
$a / b * c$	\Leftrightarrow	$(a / b) * c$



Expresii si operatori – evaluare, precedenta

Observatii finale

Operatori de atribuire

Asociativi la dreapta

$$A = B = C = 7 \quad \Leftrightarrow \quad A = (B = (C = 7))$$

Operatori de atribuire compusi

$+=$, $-=$, $*=$, $/=$, $\%=$, etc. (combinati cu operatorii pe biti)

- **In evaluarea expresiilor conteaza ordinea de precedenta**

$$A *= B + C \quad \Leftrightarrow \quad A = A * (B + C)$$



Expresii si operatori – evaluare, precedenta

Observatii finale

Operatori de incrementare / decrementare

Preincrementarea / Predecrementarea au aceeasi prioritate ca operatorii unari si sunt asociativi la dreapta.

Postincrementarea / Postdecrementarea au prioritate crescuta fata de operatorii unari si sunt asociativi la stanga.

```
int a = 10, b = 6, c;
```

```
c = a - ++b;
```

```
printf("b = %d  c = %d\n", b, c);
```

```
c = ++a + b--;
```

```
printf("a = %d  b = %d  c = %d\n", a, b, c);
```

```
b = 7  c = 3  
a = 11  b = 6  c = 18
```

a = a + 1; c = a + b; b = b - 1



Expresii si operatori – evaluare, precedenta

Observatii finale

Operatori relationali

Asociativi la stanga si au prioritate mai mica decat operatorii aritmetici.

1. $5 < 6$	\Leftrightarrow	returneaza 1
2. $7 \leq 0$	\Leftrightarrow	returneaza 0
3. $a + b \leq c * d$	\Leftrightarrow	$(a + b) \leq (c * d)$

Operatori de egalitate $==$, $!=$

Asociativi la stanga si au prioritate mai mica decat operatorii relationali.

1. $X == 6$	\Leftrightarrow	returneaza 1 daca $X = 6$
2. $A != B$	\Leftrightarrow	returneaza 1 daca A diferit de B
3. $a > b == c \leq d$	\Leftrightarrow	$(a > b) == (c \leq d)$



Expresii si operatori – evaluare, precedenta

Observatii finale

Operatori logici: !, &&, ||

! (NOT) are prioritate egala cu operatorii unari.

&& (AND) si || (OR) au prioritate mai mica decat operatorii relationali si de egalitate.

Obs: intr-o expresie compusa, daca se poate deduce rezultatul global al expresiei din stanga, cea din dreapta nu se mai evalua

1. $X = 6; (X \leq 6) \parallel (A > B) \Leftrightarrow$ returneaza 1 indiferent de valorile lui A si B, pentru ca $(6 \leq 6)$ adevarat.



Fundamentele limbajului C

Expresii si operatori – evaluare, precedenta

Observatii finale

Operatori pe biti

ordinea de precedență - în cadrul acestei categorii	
Prioritate crescută	~ (complement față de unu)
	<< (deplasare stânga)
	>> (deplasare dreapta)
	& (și pe biți)
	^ (sau exclusiv pe biți)
Prioritate scăzută	(sau pe biți)

Sursa: Alexe B. – Programare Procedurala (note de curs – seria 13 – 2016)



Fundamentele limbajului C

Expresii si operatori – evaluare, precedenta

Observatii finale

Operatorul virgula

- evaluarea secvențială a expresiilor (de la stg. la dreapta)
- valoarea ultimei expresii din înlănțuire este valoarea expresiei compuse
- cel mai puțin prioritar din lista de precedență

```
int a = 10, b = 6, c, i, s;  
  
a = a+10, b = b + a, c = (b+a)/2;  
printf("a = %d  b = %d  c = %d\n", a, b, c);  
  
for(i = 1, s = 0; i<=10; i++, s += i*i);  
printf("\n Suma patratelor pana la 10 = %d\n\n", s);
```

```
C:\Users\Ank\Desktop\curs4\bin\Debug\curs4.e  
a = 20  b = 26  c = 23  
  
Suma patratelor pana la 10 = 505
```

ATENTIE LA SEPARAREA INSTRUCIUNILOR PRIN VIRGULA!!!!!!



Fundamentele limbajului C

Funcții de citire / scriere cu format

- Funcțiile **printf** și **scanf** permit controlul formatului în care se scriu respectiv se citesc datele

printf

- afișează un șir de caractere la ieșirea standard – implicit pe ecranul monitorului
- dacă șirul conține specificatori de format, atunci argumentele adiționale (care urmează după șir) **sunt formate în concordanță cu specificatorii de format** (subșir care începe cu caracterul %) și inserate în locul și pe pozițiile acestora din cadrul șirului



Fundamentele limbajului C

Functii de citire / scriere cu format

Exemple: printf

Afisarea caracterelor si a valorilor numerice

Denumire tip	Reprezentare	Interval de valori
int	4 octeți cu semn	$-2^{31} \dots 2^{31} - 1$
unsigned int	4 octeți fără semn	$0 \dots 2^{32} - 1$
long int	4 octeți cu semn	$-2^{31} \dots 2^{31} - 1$
unsigned long int	4 octeți fără semn	$0 \dots 2^{32} - 1$
short int	2 octeți cu semn	$-2^{15} \dots 2^{15} - 1$
unsigned short int	2 octeți fără semn	$0 \dots 2^{16} - 1$
long long int	8 octeți cu semn	$-2^{63} \dots 2^{63} - 1$
unsigned long long int	8 octeți fără semn	$0 \dots 2^{64} - 1$
char	1 octet cu semn	$-2^7 \dots 2^7 - 1$
unsigned char	1 octet fără semn	$0 \dots 2^8 - 1$

Sursa: <http://www.pbinfo.ro/?pagina=articole&subpagina=afisare&id=58>



Fundamentele limbajului C

Functii de citire / scriere cu format

Exemple: printf

Afisarea caracterelor si a valorilor numerice

```
int a = 2140000000;  
unsigned int b = 4000000000;  
long int c = 2140000000;  
unsigned long int d = 4000000000;  
short int e = 32000;  
unsigned short int f = 60000;  
long long int g = 4611686018427387904;  
unsigned long long int h = 9250000000000000000;  
  
printf("\nAfisare corecta %d - %u - %ld - %lu \n",a,b,c,d);  
printf("\nAfisare incorecta %d - %d - %d - %u \n",a,b,c,d);  
  
printf("\nAfisare corecta %d - %u - %lld - %llu \n",e,f,g,h);  
printf("\nAfisare incorecta %d - %d - %lld - %lu \n",e,f,g,h);
```

```
C:\Users\Ank\Desktop\curs4\bin\Debug\curs4.exe  
  
Afisare corecta 2140000000 - 4000000000 - 2140000000 - 4000000000  
Afisare incorecta 2140000000 - -294967296 - 2140000000 - 4000000000  
Afisare corecta 32000 - 60000 - 4611686018427387904 - 9250000000000000000  
Afisare incorecta 32000 - 60000 - 4611686018427387904 - 3428646912
```



Fundamentele limbajului C

Functii de citire / scriere cu format

Exemple: printf

Afisarea caracterelor si a valorilor numerice

```
float i = 10.1231231;  
double j = 10.12312312312312;  
  
printf("\nFloat - %f \n",i);  
printf("\nFloat - %.14f \n",i);  
printf("\nFloat - %e \n",i);  
printf("\nFloat - %g \n",i);  
  
printf("\nDouble - %f \n",j);  
printf("\nDouble - %.14f \n",j);
```

```
C:\Users\Ank\Desktop\curs4\bin\Deb  
Float - 10.123123  
Float - 10.12312316894531  
Float - 1.012312e+001  
Float - 10.1231  
Double - 10.123123  
Double - 10.12312312312312
```

Nu exista specificator special de format pentru “double”!

- valorile de tip intreg se convertesc implicit la int sau unsigned int;
- variabilele cu virgula mobila se convertesc implicit la double;



Fundamentele limbajului C

Functii de citire / scriere cu format

Exemple: printf

Afisarea unui caracter

```
int x = 100;
char k = x;
char l = (char)x;

printf("\nChar - %c \n", k);
printf("\nChar - %c \n", l);
printf("\nCaracterul %% \n");
printf("\nCaracterul \\ \n");
```

```
C:\Users\Ank\De
Char - d
Char - d
Caracterul %
Caracterul \
```

Afisarea unui sir de caractere

```
printf("\nSir de caractere <constant>\n");

char *a;
a = "Exemplificare";
printf("\nSir variabil:   %s\n\n", a);
```

```
C:\Users\Ank\Desktop\curs4\bin\Debug\
Sir de caractere <constant>
Sir variabil:   Exemplificare
```




Fundamentele limbajului C

Functii de citire / scriere cu format

Exemple: printf

Afisarea unui pointer si scrierea in octal si hexazecimal

```
int y = 1500;

printf("\nAdresa lui y - %p \n",&y);
printf("\nOctal - %o \n",y);
printf("\nHexazecimal - %x \n",y);
printf("\nHexazecimal cu majuscule - %X \n",y);
```

```
Adresa lui y - 0028FEE0
Octal - 2734
Hexazecimal - 5dc
Hexazecimal cu majuscule - 5DC
```

Afisarea unui numar cu aliniere la stanga / dreapta si precedat de semnele - / +

```
int z = 75;

printf("\n z cu semn  %+d \n",z);
printf("\n Alinierea la dreapta si afisarea in 5 spatii %5d\n",z);
printf("\n Alinierea la dreapta si afisarea in 5 spatii %+5d\n",z);
printf("\n Alinierea la stanga si afisarea in 5 spatii %-5d\n",z);
```



Fundamentele limbajului C

Funcții de citire / scriere cu format

scanf

- citește date de intrare (implicit de la tastatura) în formatul indicat de șirul de formatare și le salvează la adresele indicate de argumentele adiționale (variabilele de intrare)

Exemple:

```
int a;  
unsigned int b;  
short int e ;  
unsigned long long int h;  
float i;  
double j;  
char k;  
char v[20];  
char *w;
```



Fundamentele limbajului C

Functii de citire / scriere cu format

Exemple: scanf

```
int a;  
unsigned int b;  
short int e ;  
unsigned long long int h;  
float i;  
double j;  
char k;  
char v[20];  
char *w;
```

```
scanf ("%c%s", &k, v);  
printf ("\nCaracter %c - sir de caractere %s\n", k, v);  
w = malloc(20 * sizeof(char));  
scanf ("%s", w);  
printf ("Alt sir de caractere %s\n", w);  
  
scanf ("%d : %u + %hd * %lld", &a, &b, &e, &h);  
printf ("%d \t %u \t %d \t %lld\n\n", a, b, e, h);  
  
scanf ("%f%f", &i, &j);  
printf ("\nFloat si double: %2.3f - %1.4f\n", i, j);
```



Fundamentele limbajului C

Funcții de citire / scriere cu format

Exemple: scanf

```
scanf ("%c%s", &k, v);  
printf ("\nCaracter %c - sir de caractere %s\n", k, v);  
w = malloc(20 * sizeof(char));  
scanf ("%s", w);  
printf ("Alt sir de caractere %s\n", w);  
  
scanf ("%d : %u + %hd * %lld", &a, &b, &e, &h);  
printf ("%d \t %u \t %d \t %lld\n\n", a, b, e, h);  
  
scanf ("%f%f", &i, &j);  
printf ("\nFloat si double: %2.3f - %1.4f\n", i, j);
```

```
aRe  
Caracter a - sir de caractere Re  
Doi  
Alt sir de caractere Doi  
-240 : 4111111111 + 32123 * 8000000000  
-240      4111111111      32123      8000000000
```




Fundamentele limbajului C

Funcții de citire / scriere cu format

Exemple: scanf

Caracteristica scanset

Un specificator scanset se poate crea prin includerea unui listă de caractere în interiorul unor paranteze drepte. exemple: %[ABC], %[A-Z] (domeniu);

```
char a[20];  
printf("Sir exclusiv cu literele XY  
scanf("%[XYZ]", a);  
printf("%s\n",a);
```

```
Sir exclusiv cu literele XYZ  
XYZXYZZZZZYYYYXXXT  
XYZXYZZZZZYYYYXXX
```

```
char d[20];  
printf("\n\nSir exclusiv cu litere intre A si F\n\n");  
scanf("%[a-zA-F]", d);  
printf("%s\n",d);
```

```
Sir exclusiv cu litere intre A si F  
AffbbcdG  
Affbbcd
```



Fundamentele limbajului C

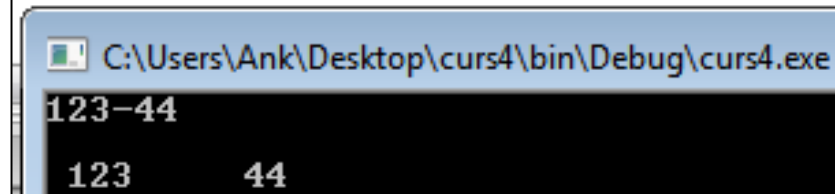
Functii de citire / scriere cu format

Exemple: scanf

Caracteristica **scanset**

- **Semnul *** precede setul care **NU** trebuie luat in considerare

```
int b,c;  
scanf("%d%c%d",&b, &c);  
printf("\n %d \t %d \n",b,c);
```



Orice caracter vine dupa primul numar intreg este ignorat!



Agenda cursului

1. Scurta recapitulare a cursului anterior: Fundamentele limbajului C:
Reprezentarea datelor in memorie (signed vs. unsigned);
Expresii si operatori – evaluare, precedenta;
Operatorii pe biti;
Functii de citire / scriere cu format;

2. Complexitatea algoritmilor – notiuni introductive (2)

- complexitate : timp, spatiu de memorie, notatii asimptotice,
“time.h”



Complexitatea algoritmilor

Analiza complexității unui algoritm => determinarea resurselor de care acesta are nevoie pentru a produce datele de ieșire.

Resurse - timpul de executare
- spatiu de memorie etc.

Obs: Modelul masinii pe care va fi executat algoritmul nu presupune existenta operatiilor paralele (operatiile se executa secvential).

Notatie: $T(n)$ – timp de rulare al unui algoritm (in general masurat in nr. de comparatii sau de mutari)

Cazuri:

- cel mai favorabil
- cel mai nefavorabil
- mediu



Complexitatea algoritmilor

Intuitiv – Numărarea operațiilor elementare realizate de un algoritm
În cazul cel mai defavorabil

Aflarea maximului unui vector de dimensiune n	# operații elementare realizate de algoritm în cazul cel mai defavorabil
<code>maxim = v[0];</code>	→ 2 (o indexare + o atribuire)
<code>for (i=1; i<n;i++)</code>	→ $2n$ (o atribuire + n comparații + $(n-1)$ incrementări)
<code>if (maxim < v[i])</code>	→ $2(n-1)$ ($n-1$ indexări + $n-1$ comparații)
<code>maxim = v[i];</code>	→ $2(n-1)$ ($n-1$ indexări + $n-1$ atribuiri)
<code>return maxim</code>	→ 1 (o întoarcere a rezultatului)
$T(n) = 6n-1$ operații elementare	



Complexitatea algoritmilor

Exemple

1. Produsul a doua numere complexe ([1])

```
int main()
{
float a,b,c,d, p, q;
float t1, t2;
scanf("%f%f%f%f", &a, &b, &c, &d);
t1 = a*c; t2 = b*d; p = t1 - t2;
t1 = a*d; t2 = b*c; q = t1 + t2;
printf("Real = %f, Imaginar = %f", p, q);
}
```

- memorie pentru 8 variabile

Operatii elementare: 4 inmultiri, o adunare
si o scadere

```
int main()
{
float a,b,c,d, p, q;
float t1, t2, t3, t4;
scanf("%f%f%f%f", &a, &b, &c, &d);
t1 = a + b; t2 = t1 * c; t1 = d - c; t3 = a * t1;
q = t2 + t3; t1 = d + c; t4 = b * t1; p = t2 - t4;
printf("Real = %f, Imaginar = %f", p, q);
}
```

- memorie pentru 10 variabile

Operatii elementare: 3 inmultiri, 3 adunari si 2
scaderi

**Operatia de inmultire e mai costisitoare
decat adunarea / scaderea.**



Complexitatea algoritmilor

Exemple

2. Cmmdc a 2 numere

Euclid:

```
int a,b,r;  
scanf("%d%d", &a, &b);  
r = a % b;  
while (r!=0)  
{ a = b;  
  b = r;  
  r = a % b;  
}  
printf("Cmmdc = %d", b);
```

Scaderi repetate:

```
int a,b,r;  
scanf("%d%d", &a, &b);  
while (a != b)  
    if (a > b)  
        a = a - b;  
    else  
        b = b - a;  
printf("Cmmdc = %d", a);
```

Algoritm brut:

```
int a,b,c,i,min;  
scanf("%d%d", &a, &b);  
  
if (a < b) min = a;  
else min = b;  
  
for(i = 1 ; i <= min; i++)  
    if ( %i==0 && b%i == 0)  
        c = i;  
printf("Cmmdc = %d", c);
```

Cate operatii se executa daca se citesc initial numerele 97 si 99?



Complexitatea algoritmilor

Exemple

3. Ordonarea unui sir folosind Interschimbarea directa

```
int main()
{
    int v[100], n, i, j, aux;
    // citire vector
    for (i = 1; i < n; i++)
        for(j = i+1; j <= n; j++)
            if (v[i] > v[j])
                { aux = v[i];
                  v[i] = v[j];
                  v[j] = aux;
                }
    // Afisare vector ordonat
}
```

Caz	Comparatii	Mutari
Cel mai favorabil	$n(n - 1) / 2$	0
Cel mai defavorabil	$n(n - 1) / 2$	$3n(n - 1) / 2$
mediu	$n(n - 1) / 2$	$3n(n - 1) / 4$

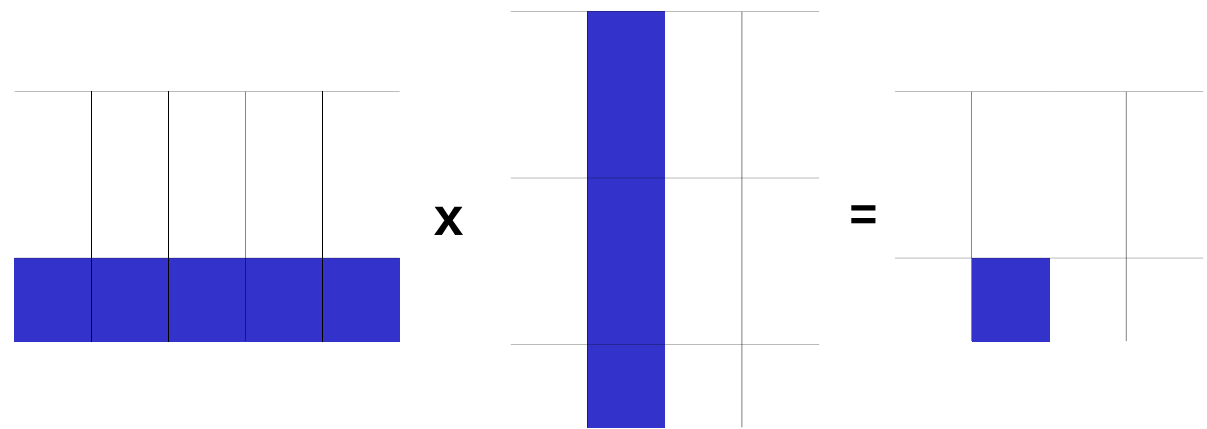


Complexitatea algoritmilor

Exemple

4. Inmultirea a doua matrice

```
int main()
{
    int a[10][20], b[20][30], c[10][30];
    int n, m, p, i, j, k;
    // citire matrice a si b
    for(i=1; i<=n; i++)
        for(k=1; k<=p; k++)
            { c[i][k] = 0;
              for(j=1; j<=m; j++)
                  c[i][k] = c[i][k] + a[i][j] * b[j][k];
            }
    // Afisare matrice produs
}
```



$$A_{n,m} \times B_{m,p} = C_{n,p}$$

$$O(m \cdot n \cdot p)$$



Complexitatea algoritmilor

Exemple

5. Diferența maximă dintre 2 elemente ale unui vector de dimensiune n

```
difMaxima = 0;  
for (i=0; i<n;i++)  
    for (j = i+1; j< n; j++ )  
        if (difMaxima < abs(v[i]-v[j]))  
            difMaxima = abs(v[i]-v[j]);  
return difMaxima
```

$O(n^2)$

Exista un algoritm mai eficient? Ce complexitate are?



Complexitatea algoritmilor

Notatia asimptotica

$T(n)$ – timp de rulare al unui algoritm (comparatii / mutari)

comportarea lui $T(n)$ cind $n \rightarrow \infty$ ([2])

Formal

$f : \mathbb{N} \rightarrow \mathbb{R}_+$ (f asimptotic pozitiva)

$O(g) := \{f \mid \exists c > 0, \exists n_0 \text{ a.i. } 0 \leq f(n) \leq cg(n) \text{ oricare } n \geq n_0\}$

$\Omega(g) := \{f \mid \exists c > 0, \exists n_0 \text{ a.i. } 0 \leq cg(n) \leq f(n) \text{ oricare } n \geq n_0\}$

$\Theta(g) := \{f \mid \exists c_1, c_2 > 0, \exists n_0 \text{ a.i. } 0 \leq c_1g(n) \leq f(n) \leq c_2g(n) \text{ oricare } n \geq n_0\}$

[2] Ceterchi R. – Algoritmi si Structuri de Date (note de curs 2013)



Complexitatea algoritmilor

Notatia asimptotica

$f : \mathbb{N} \rightarrow \mathbb{R}_+$ (f asimptotic pozitiva)

$O(g) := \{f \mid \exists c > 0, \exists n_0 \text{ a.i. } 0 \leq f(n) \leq cg(n) \text{ oricare } n \geq n_0\}$

Exemplu: $f(n) = 2n + 10$ este $O(n)$

$2n + 10 \leq cn$, pentru $n \geq n_0$

$(c - 2)n \geq 10$

$n \geq 10/(c - 2)$

alegem $c = 3$ și $n_0 = 10$

Exemplu: $f(n) = n^2$ nu este $O(n)$

$n^2 \leq cn$, pentru $n \geq n_0$

$n \leq c$

nu putem alege constanta c

n^2 este $O(n^2)$



Complexitatea algoritmilor

Notatia asimptotica

Notatia O oferă o margine superioară a creșterii asimptotice a unei funcții.

Creșterea asimptotică caracterizează comportamentul funcțiilor pentru valori mari ale lui n .

Când spunem “ $f(n)$ este $O(g(n))$ ” înseamnă că rata de creștere a lui $f(n)$ nu este mai mare decât rata de creștere a lui $g(n)$

Putem folosi notația O pentru a ordona funcțiile pe baza ratei lor de creștere asimptotică



Complexitatea algoritmilor

Notatia asimptotica

Ordonati functiile pe baza cresterii lor asimptotice

$$f_1(n) = n + \sin n$$

$$f_2(n) = \ln n$$

$$f_3(n) = n + \sqrt{n}$$

$$f_4(n) = \frac{1}{n}$$

$$f_5(n) = 13 + \frac{1}{n}$$

$$f_6(n) = 13 + n$$

$$f_7(n) = (n + \sin n) \cdot (n^{20} - 5)$$

$$f_8(n) = n^e$$

$$f_9(n) = n^n$$

$$f_{10}(n) = n \cdot \ln n$$

$$f_{11}(n) = n \cdot (\ln n)^2$$

$$f_{12}(n) = \log_2 n$$


$$f_{13}(n) = e^n$$



Complexitatea algoritmilor

Notatia asimptotica

Ordonati functiile pe baza cresterii lor asimptotice


$$f_4(n) = \frac{1}{n}$$

$$f_5(n) = 13 + \frac{1}{n}$$

$$f_2(n) = \ln n \quad f_{12}(n) = \log_2 n$$

$$f_1(n) = n + \sin n \quad f_3(n) = n + \sqrt{n} \quad f_6(n) = 13 + n$$

$$f_{10}(n) = n \cdot \ln n$$

$$f_{11}(n) = n \cdot (\ln n)^2$$

$$f_8(n) = n^e$$

$$f_7(n) = (n + \sin n) \cdot (n^{20} - 5)$$

$$f_{13}(n) = e^n$$

$$f_9(n) = n^n$$



Complexitatea algoritmilor

Notatia asimptotica

Funcții incomparabile

- pentru două funcții $f(n)$ și $g(n)$ nu întotdeauna avem
 $f(n) = O(g(n))$ sau $g(n) = O(f(n))$
- f și g se numesc **asimptotic incomparabile**
- exemplu:
 $f(n) = |n^2 \sin(n)|$ și $g(n) = 5n^{1.5}$



Complexitatea algoritmilor

Fisierul antet time.h ([1])

Inclus pentru masurarea timpului

Contine prototipul functiei **clock()** => nr. de tacte de ceas de timp real scurs de la inceperea programului, pana in punctul in care se plaseaza aceasta functie

clock()/CLK_TCK => timpul in secunde (**Obs:** CLK_TCK este denumirea pentru CLOCKS_PER_SEC in versiunile anterioare Microsoft C.)

Plasand doua asemenea expresii, inaintea si dupa apelul subprogramului aflat sub masurare, diferenta lor da timpul consumat de algoritm.



Complexitatea algoritmilor

Fisierul antet time.h ([1])

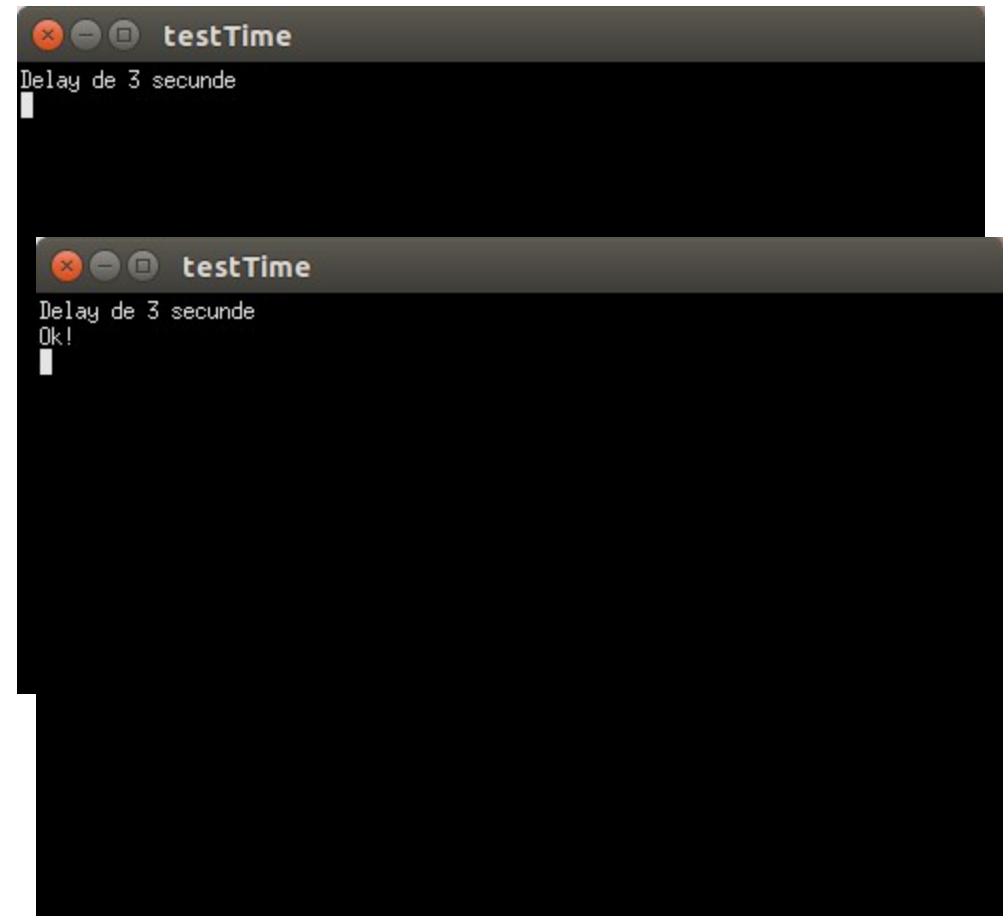
Exemplu

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

// pauza pentru un numar specificat de milisekunde
void sleep(clock_t f)
{
    clock_t g;
    g = f + clock();
    while (g > clock()) ;
}

int main()
{
    long i = 600000000L;
    clock_t start, finish;
    double duration;

    //Delay pentru un timp specificat
    printf("Delay de 3 secunde\n");
    sleep((clock_t)3*CLOCKS_PER_SEC);
    printf("Ok!\n");
}
```





Complexitatea algoritmilor

Fisierul antet time.h ([1])

Exemplu

```
//Masurarea duratei unui eveniment
printf("Timpul de executie a %ld bucle vide este ",i);
start = clock();
while(i--);
finish = clock();
duration = (double)(finish - start)/CLOCKS_PER_SEC;
printf("%.1f secunde \n", duration);
```

```
testTime
Delay de 3 secunde
Ok!
Timpul de executie a 6000000000 bucle vide este 1.3 secunde

Process returned 0 (0x0)   execution time : 4.327 s
Press ENTER to continue.
```




Concluzii

1. S-au recapitulat notiunile de Limbaj C predate in cursul 3:
Precedenta si asociativitatea operatorilor, citirea / scrierea datelor cu format in Limbajul C
2. S-au prezentat legaturile dintre notatia O si cresterea asimptotica a functiilor



Perspective

Cursul 5:

1. Tipuri derivate de date
 - Tablouri. Șiruri de caractere.
 - Structuri, uniuni, câmpuri de biți, enumerări.
 - Pointeri.