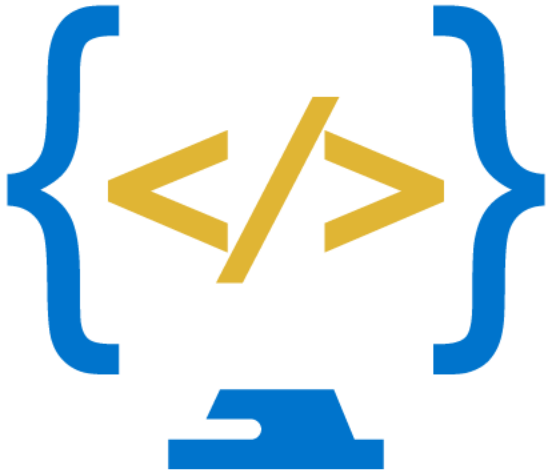


PROGRAMARE PENTRU TOTI

BUSINESS FOR YOU

LSAC

Lectia 3

Introducere in limbajul C

Cuprins

- Operatii pe biti
- Coding style
- Tehnici de debugging
- Exercitii

PROGRAMARE PENTRU TOTI

Operatii pe biti

PROGRAMARE PENTRU TOTI

Operatiile pe biti sunt foarte importante pentru ca pot creste performanta programelor.

De ce?

Procesoarele lucreaza in binar => operatiile in binar sunt mult mai rapide decat oricare altele.

Operatii pe biti

- **NOT: ~**

x	~x
0	1
1	0

- **AND: &**

x	y	x & y
0	0	0
0	1	0
1	0	0
1	1	1

- **OR: |**

x	y	x | y
0	0	0
0	1	1
1	0	1
1	1	1

PROGRAMARE PENTRU TOTI

Operatii pe biti

- **XOR: ^**

x	y	x ^ y
0	0	0
0	1	1
1	0	1
1	1	0

- **Left Shift: <<**
  - Echivalentul inmultirii cu 2 cat timp avem biti liberi in stanga

	B2	B1	B3
x = 3	0	1	1
x « 1 = 6	1	1	0
x « 2 = 8	1	0	0

- **Right Shift: >>**
  - Echivalentul impartirii (naturale) cu 2

PROGRAMARE PENTRU TOTI

Operatii pe biti

PROGRAMARE PENTRU TOTI

- **Lucrul cu masti - 1**
  - Pentru a modifica un bit sau mai multi se folosesc masti
  - Daca se doreste verificarea valorii unui bit de pe o pozitie data se face **si logic** cu o masca

    x = x **&** mask; mask = (1 << i); i – pozitia bitului cautat
  - Pentru setarea unui bit la 1 se foloseste **sau logic** cu o masca

    x = x **|** mask; mask = (1 << i); i – pozitia bitului de setat

Operatii pe biti

PROGRAMARE PENTRU TOTI

- **Lucrul cu masti - 2**
  - Pentru setarea unui bit la 0 se foloseste **si logic** cu o masca inversata

    x = x **&** mask; mask = ~(1 << i); i – pozitia bitului de setat

  - Pentru inversarea unui bit (toggle) se foloseste **xor** cu o masca

    x = x **^** mask; mask = (1 << i); i – pozitia bitului de schimbat

Coding Style

- **Spatiere**
  - La operatorii binari se pune spatiu si inainte si dupa (ex: c = a + b; a / b; a == c;)
  - La operatorii unari nu se pune spatiu (ex: a++; --c;)
  - La if, for, while se respecta urmatoarea structura:

```
<instructiune><spatiu>(<conditii>)<spatiu>{
        instr1;
        instr2;
        ...
}
```

  - Nu se lasa spatiu inainte de “;” sau “,”, se lasa doar dupa

PROGRAMARE PENTRU TOTI

Coding Style

- **Alte reguli**
  - Variabilele si functiile trebuie sa aiba nume suggestive
    (NU folosim f(), int a, b, c; etc.)
  - In C cuvintele din numele variabilelor se separa prin "_" si se scriu cu litere mici
    (ex: functie_cu_nume_sugestiv)
  - Folosim acolade chiar daca blocul de instructiuni are o singura instructiune
  - Acoladele se deschid pe linia cu functia si se inchid la acelasi nivel cu prima litera din antetul functiei:

```
int functie_cu_nume_sugestiv() {
        instr1...
}
```

  - O functie nu trebuie sa aiba foarte multe linii
  - O linie nu trebuie sa depaseasca 80-100 de caractere
  - Comentariile se scriu pe o alta linie (in general deasupra) decat linie sau functia pentru care sunt scrise

PROGRAMARE PENTRU TOTI

Tehnici de debugging

- Erorile de compilare apar în general datorita greșelilor de sintaxă.
  - Variabile nedeclarate
  - Parantezele sau acoladele deschise/închise necorespunzător
  - Lipsesc operatori sau bucăți din cod (Ex: x = y + 5; => x = y 5;)

```
erori.c:8:5: error: expected ']' before 'int'
     int a = 10;
     ^~~
erori.c:21:1: error: expected declaration or statement at end of input
 }
 ^


erori.c:8:5: error: expected '=', ',', ';', 'asm' or '__attribute__' before 'int'
     int a = 10;
     ^~~
```

PROGRAMARE PENTRU TOTI

Tehnici de debugging

- Erorile la runtime apar atunci când în timpul programului se încearcă executarea unei operații nepermise:
  - Accesarea unei zone nealocate de memorie
  - Împărțirea la zero
  - Depășirea memoriei disponibile pe stivă
  - **OBS! Informațiile întoarse de C pentru erorile la runtime sunt succinte și necesită o reevaluare a codului pentru a determina sursa lor.**

PROGRAMARE PENTRU TOTI

Tehnici de debugging

- Warningurile, de obicei, anunță posibilitatea apariției erorilor la runtime:

```
erori.c: In function 'main':
erori.c:5:19: warning: integer constant is too large for its type
     int big_array[40000000000000000000000000] = { 0 };
                   ^~~~~~~~~~~~~~~~~~~~~~~~~~
```

```
erori.c: In function 'main':
erori.c:11:11: warning: division by zero [-Wdiv-by-zero]
     a = a / 0;
           ^
```

PROGRAMARE PENTRU TOTI

Tehnici de debugging

- Cel mai rapid debugging se face cu ajutorul afisarilor pe ecran, fie cu functia printf(), fie cu perror() din biblioteca stdio.h
- OBS: Cand folositi printf() pentru debugging, nu uitati ca la final sa adaugati caracterul linie noua – “\n”.
- Ex: printf(“Ajunge aici!\n”);

PROGRAMARE PENTRU TOTI

Exemple de probleme

- Inversarea bitilor dintr-o variabila. Ex: 11001100->00110011.
- Inmultirea unui numar dat cu 3.5 folosind operatii pe biti.

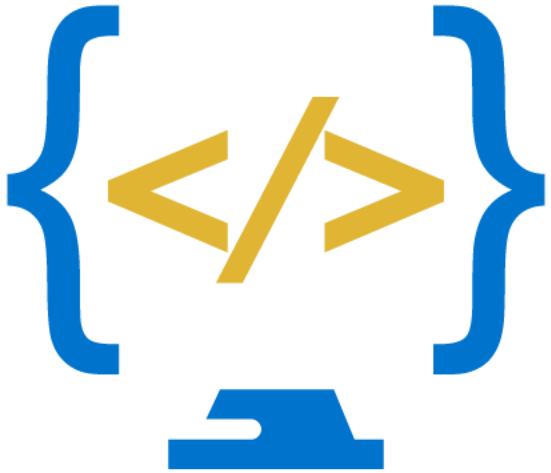
Link util cu probleme simple:

https://codeforwin.org/2016/01/bitwise-operator-programming-exercises-and-solutions-in-c.html

GitHub PPT:

https://github.com/georgescubogdan/programare-pentru-toti

PROGRAMARE PENTRU TOTI



PROGRAMARE PENTRU TOTI

BUSINESS FOR YOU

LSAC

Multumim!

Va asteptam la lectia urmatoare, marti, 29/10, de la ora 20, in PR001!