Laborator 3 – Limbaje de Programare

1 Recapitulare şi exemple

1.1 Iteraţia

Să ne amintim câteva exemple cu cicluri de la curs:

1. Funcţia factorial definită în mod iterativ, cu un ciclu **while**:

```
unsigned fact_it(unsigned n){ //factorial iterativ
    unsigned r=1;
    while(n>0){
        r=r*n;
        n=n-1;
    }
    return r;
}
```

   Ca să tipărim rezultatul evaluării funcţiei noastre iterative într-un punct, întreg programul nostru va fi cel de mai jos.

```
#include <stdio.h>

unsigned fact_it (unsigned n) {
  unsigned r = 1;
  while (n > 0) {
      r = r * n;
      n = n - 1;
  }
  return r;
}

int main(void) {
   printf("%d", fact_it(5));
   return 0;
 }
```

2. Citirea unui numar natural de la tastatură, cifră cu cifră, în mod iterativ, cu un ciclu **while**:

```
#include <stdio.h>
#include <ctype.h>

unsigned readnat (void) {
   int c; unsigned r = 0;

   while (isdigit(c = getchar()))  // cat timp e cifra
      r = 10*r + c - '0';          // compune numarul

   ungetc(c, stdin); // pune inapoi ce nu-i cifra
   return r;
}
```

```
int main(void){
    unsigned int nrcitit = readnat();

    printf("%d", nrcitit);
    return 0;
  }
```

3. Suma a maxim n cifre citite de la intrare, implementată cu un ciclu **for**:

```
unsigned sumancif(unsigned n){
   int i=0;
   unsigned r = 0;

   for(i=0; i<n; i++){ // de la 0 la n-1 (n ori)
        int c = getchar(); //citeste caract.

        if(c != EOF && isdigit(c)){ // e cifra?
          r = r + (c-'0'); //aduna la rezultat
           }
   }
   return r;
}
```

1.2 Reprezentare internă. Operaţii pe biţi

ATENŢIE: Operatorii pe biţi se pot folosi **doar** pentru operanzi de **tip întreg**.

Operator	**Semnificaţie**
&	ŞI pe biţi: 1 & 1 e 1, altfel 0
|	SAU pe biţi: 0 | 0 e 0, altfel 1
∧	SAU EXCLUSIV pe biţi: 1 dacă biţi diferiti, altfel 0
∼	COMPLEMENT pt biţi: 1 devine 0, 0 devine 1
<<	SHIFT (deplasare) la stânga: introduce biţi de 0
>>	SHIFT (deplasare) la dreapta: introduce biţi de semn

1.2.1 Exemple operaţii pe biţi:

ŞI	SAU	SAU EXCLUSIV
10110101 &	10110101 |	10110101^
01110001	01110001	01110001
= 00110001	= 11110101	= 11000100

Complement pe biţi:
∼ 10110101 = 01001010 ∼ 01110001 = 10001110

Deplasare la stânga:
10110101 << 1 = 01101010 10110101 << 2 = 11010100
10110101 << 3 = 10101000 10110101 << 6 = 01000000

Deplasare la dreapta:
10110101 >> 1 = 01011010 10110101 >> 2 = 00101101
10110101 >> 3 = 00010110 10110101 >> 6 = 00000010

1.2.2 Exemple de creare şi utilizare de tipare pe biţi

$n << k$ e echivalent cu $n \cdot 2^k$
$n >> k$ e echivalent cu $n/2^k$ (pt. n unsigned!!)

$1 << k$ e 2^k, are doar bitul k pe 1, pt. k < 8 * sizeof(int)
$\sim (1 << k)$ are doar bitul k pe 0, restul pe 1
~ 0 are toţi biţii pe 1 (iar 0 toţi pe 0)
$\sim 0 << k$ are k biţi din dreapta 0, restul pe 1
$\sim (\sim 0 << k)$ are k biţi din dreapta 1, restul pe 0
$\sim (\sim 0 << k) << p$ are k biţi pe 1, începând de la bitul p, restul 0

$b \,\&\, 1$ păstrează valoarea bitului b, pt. orice bit b
$b \,\&\, 0$ e 0, pt. orice bit b
$n \,\&\, (1 << k)$ e $!= 0$ dacă bitul k al lui n e 1, altfel e $== 0$
$n \,\&\, \sim (1 << k)$ va avea 0 pe bitul k al lui n

$b \mid 1$ e 1, pt. orice bit b
$b \mid 0$ păstrează valoarea bitului b, pt. orice bit b
$n \mid (1 << k)$ va avea 1 pe bitul k al lui n

1.3 Exemple reprezentare internă şi operatori pe biţi

1. Tipărirea unui număr în format binar ĉu ajutorul unei functii recursive, luand resturile impartirii la 2 in ordine inversa (pentru a se tipări întâi cifra cea mai semnificativă – cea de pe prima poziţie binară din stânga, apoi următoarea, etc., terminând cu ultima cifră binară, cea mai din dreapta).

```
#include<stdio.h>

void bindigitsrec(unsigned n){
   if (n > 0){
      bindigitsrec(n/2); //apelul recursiv
    }

    char bcif = n % 2 + '0'; //restul impartirii la 2 a lui n
    //se transforma in cifra prin adunare cu '0'

    putchar(bcif); //scrie cifra binara
}

int main(int argc, char ** args){
  unsigned numar;
  scanf("%u", &numar); //citim un numar intreg fara semn

  bindigitsrec(numar);
  return 0;
}
```

2. Tipărirea unui număr în format binar, folosind un ciclu **do while** şi operaţii pe biţi (se tipareste intreaga configuratie binara a numarului):

```
#include<stdio.h>

void bindigits (unsigned n){

 int k = sizeof(unsigned)*8 - 1; //nr. de bi\c ti ai tipului
 //se incepe cu cifra cea mai semnificativa, prima cifra de la stanga

 do{
    unsigned bit = n & (1 << k); //extrage bitul de ordin k
```

```
    // bit va fi 0, daca pe pozitia k n are 0
    // bit va fi 2^k daca n are 1 pe pozitia k

    bit = bit >> k;  //deplasare la stanga cu k poziti binare
    //pt a avea bitul k pe ultima pozitie, deci 0 sau 1 (nu 2^k)

    char bcif = bit + '0';  // obtine cifra corespunzatoare
    putchar(bcif);  //tipareste cifra

    k--;   // urmatoarea cifra
  } while (k >= 0);
}

int main(int argc, char ** args){
  unsigned numar;
  scanf("%u", &numar); // citim un numar intreg fara semn

  bindigits(numar);
  return 0;
}
```

2 Lucrarea 3

Să se implementeze, să se apeleze în **main** şi să se tipărească rezultatele obţinute:

1. Funcţii cu cicluri:
   - Scrieţi o funcţie care citeşte caractere de la intrare pâna la EOF şi scrie la ieşire numai literele mari.
   - Scrieţi o funcţie care calculează în mod iterativ cmmdc (cel mai mare divizor comun) a două numere (prin metoda împărţirilor repetate, sau a scăderilor repetate, după preferinţe)
   - Scrieţi o funcţie care determină daca un număr natural n primit ca parametru e prim (Hint: încercaţi să îi găsiţi divizori între 2 şi $n/2$)
   - Scrieţi o funcţie care tipăreste pe ecran, în ordine inversă, cifrele unui număr natural primit ca parametru.

2. Funcţii cu operaţii pe biţi (ca să vă verificaţi, puteţi folosi, pentru tipărirea configuraţiilor pe biţi, oricare dintre cele 2 funcţii date ca exemplu mai sus):
   - Scrieţi o funcţie care să returneze adevărat dacă bitul 3 al unui număr natural n (primit ca parametru) este 1.

     Generalizare: scrieţi o funcţie care să returneze adevărat dacă bitul k al unui număr natural n este 1, şi fals altfel (k şi n se dau ca parametri).

   - Scrieţi o funcţie care să seteze pe 1 bitul 3 al unui număr natural n primit ca parametru

     Generalizare: scrieţi o funcţie care să seteze pe 1 bitul k al unui număr natural n (k şi n se dau ca parametri)

   - Scrieţi o funcţie care să pună pe 0 bitul 3 al unui număr natural n primit ca parametru

     Generalizare: scrieţi o funcţie care să pună pe 0 bitul k al unui număr natural n (k şi n se dau ca parametri)

   - Scrieţi o funcţie care să schimbe valoarea bitului 3 (din 0 în 1, respectiv din 1 în 0) al unui număr natural n primit ca parametru

     Generalizare: scrieţi o funcţie care să schimbe valoarea bitului k (din 0 în 1, respectiv din 1 în 0) al unui număr natural n (k şi n se dau ca parametri)