Laborator 7 – Exerciții rezolvate

Problema 1. Se citește un număr întreg pozitiv n. Folosind operatori pe biți să se afișeze reprezentarea în baza 2 a numărului n.

Rezolvare: Ne bazăm pe faptul că în memorie n este deja reprezentat în baza 2, deci trebuie să-i afișăm biții de la stânga la dreapta. Presupunând că n este reprezentat pe ***8*sizeof(int)*** (32) de biți, pe aceștia îi numerotăm de la dreapta la stânga cu numere de la ***0*** la ***8*sizeof(int) - 1***. Pentru a obține bitul de pe poziția i (0 <= i <= 31), utilizăm expresia ***(n >> i) & 1***. Nu rămâne decât să utilizăm expresia pentru fiecare i între 0 și ***8*sizeof(int) - 1*** (31).

```
#include <stdio.h>

int main(void) {
  int i;
  unsigned n;
  printf("n=");scanf("%u",&n);

  for (i = 8*sizeof(n)-1; i >= 0; i--)
    //deplasam bitul de pe pozitia i, i pozitii spre dreapta pentru a
    // ajunge pe prima pozitie
    // facem o conjunctie cu masca 000...0001 (numarul 1) pentru a obtine
    valoarea bitului (0 sau 1)
    printf("%d", (n >> i)& 1);
}
```

Rulare:

```
n=7
00000000000000000000000000000111
```

Dacă dorim să eliminăm zerourile nesemnificative codul devine:

```
#include <stdio.h>

int main(void) {
  int i, cif_semn=0;
 //cif_semn indica daca am intalnit prima cifra semnificativa (un 1)
  unsigned n;

  printf("n=");scanf("%u",&n);

  for (i = 8*sizeof(n)-1; i >= 0; i--)
  {
    // facem o disjunctie cu cifra de pozitia i pentru a vedea daca este egala cu 1
    cif_semn |= (n >> i)& 1;//cif_semn = cif_semn | ((n >> i)& 1)
```

```
    //deplasam bitul de pe pozitia i, i pozitii spre dreapta pentru a ajunge pe
    prima pozitie
    //facem o conjunctie cu masca 000...0001 (numarul 1) pentru a obtine valoarea
    bitului (0 sau 1)
    if (cif_semn) printf("%d", (n >> i)& 1);
    //tiparim cifra doar daca este semnificativa
  }
}
```

Rulare:

```
n=9
1001
```

Problema 2. Se citește un număr întreg n reprezentând numărul de elemente dintr-o mulțime și un șir de n valori întregi reprezentând elementele acestei mulțimi. Folosind operatori pe biți să se afișeze toate submulțimile mulțimii date.

Rezolvare:

Exemplu. Fie mulţimea $A = \{x, y, z\}$, submuţimile sale sunt:

- $\{\}$ mulţimea vidă (notată $\emptyset$ sau $\{\varnothing\}$)
- $\{x\}$
- $\{y\}$
- $\{z\}$
- $\{x, y\}$
- $\{x, z\}$
- $\{y, z\}$
- $\{x, y, z\}$

Acestea pot fi construite folosind cele $2^{card(A)}$ reprezentări binare (secvenţe de biţi) ale numerelor $\left\{0, 1, \ldots, 2^{card(A)} - 1\right\}$:

- $000_{(2)} = 0 \rightarrow \{\}$
- $100_{(2)} = 4 \rightarrow \{x\}$
- $010_{(2)} = 2 \rightarrow \{y\}$
- $001_{(2)} = 1 \rightarrow \{z\}$
- $110_{(2)} = 6 \rightarrow \{x, y\}$
- $101_{(2)} = 5 \rightarrow \{x, z\}$
- $011_{(2)} = 3 \rightarrow \{y, z\}$
- $111_{(2)} = 7 \rightarrow \{x, y, z\}$

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

void printSubmultimi(int multime[], int nr_elem)
```

```
{

    //nr_submultimi este egal cu (2**nr_elem - 1)
    unsigned int nr_submultimi = pow(2, nr_elem);//sau nr_submultimi = 1<<nr_elem
    int contor, j;

    //contorul merge de la 0(000..0) la nr_submultimi-1(111..1)
    for(contor = 0; contor < nr_submultimi; contor++)
    {
      printf("{ ");
      for(j = 0; j < nr_elem; j++)
      {
         /* Verificam daca al j-lea bit din contor are valoarea 1
         Daca este 1 atunci tiparim multime[j] ca parte din submutimea  curenta */
         if(contor & (1<<j))
            printf("%d ", multime[j]);
       }
       printf("}\n");
    }
}

int main() {
    int multime[] = {1,2,3};
    printSubmultimi(multime, 3);
    return 0;
}
```

Rulare:

```
{ }
{ 1 }
{ 2 }
{ 1 2 }
{ 3 }
{ 1 3 }
{ 2 3 }
{ 1 2 3 }
```

Problema 3.Temă: Se citește un număr întreg n reprezentând numărul de elemente dintr-o mulțime și un șir de n valori întregi reprezentând elementele acestei mulțimi. Se citește un număr întreg k<n. Se să afișeze toate submulțimile de k elemente ale mulțimii date.

Rezolvare: Pornind de la problema anterioara vom afișa doar acele submulțimi care au exact k elemente (pentru care reprezentarea binara a contorului are exact k valori de unu, deci pentru care suma cifrelor binare ale contorului este egala cu k).

```c
#include <stdio.h>
#include <math.h>

void printSubmultimi(int multime[], int nr_elem, int k)
{
    //nr_submultimi este egal cu (2**n - 1)
    unsigned int nr_submultimi = pow(2, nr_elem);// sau = 1<<nr_elem
    int contor, j, i, suma_contor;

    //contorul merge de la 0(000..0) la nr_submultimi - 1(111..1)
    for(contor = 0; contor < nr_submultimi; contor++)
    {
      //caculam suma cifrelor binare ale contorului
      for (i = 8*sizeof(contor)-1,suma_contor=0; i >= 0; i--)
        suma_contor += (contor >> i)& 1;

      //submultimea asociata contorului are exact k elemente o vom tiparii
      if(suma_contor == k){
        printf("{ ");
        for(j = 0; j < nr_elem; j++)
        {
         /* Verificam daca al j-lea bit din contor are valoarea 1
         Daca este 1 atunci tiparim multime[j] ca parte din submutimea curenta */
            if(contor & (1<<j))
              printf("%d ", multime[j]);
        }
        printf("}\n");
      }
    }
}

int main()
{
    int multimi[100], n, k, i;
    printf("n=");scanf("%d",&n);
    printf("k=");scanf("%d",&k);

    printf("Introduceti elementele multimii:\n");
    for(i=0;i<n;i++){
        printf("a[%i]=",i);
        scanf("%d",&multimi[i]);
    }
```

```
    printf("Submultimile de cate %d elementele ale multimii sunt:\n", k);
    printSubmultimi(multimi, n, k);
    return 0;
}
```

Rulare:

```
n=5
k=3

Introduceți elementele mulțimii:
a[0]=1
a[1]=2
a[2]=3
a[3]=4
a[4]=5

Submulțimile de cate 3 elementele ale mulțimii sunt:
{ 1 2 3 }
{ 1 2 4 }
{ 1 3 4 }
{ 2 3 4 }
{ 1 2 5 }
{ 1 3 5 }
{ 2 3 5 }
{ 1 4 5 }
{ 2 4 5 }
{ 3 4 5 }
```