FUNDAMENTELE PROGRAMĂRII

1. Scrieți un program care să afișeze toate numerele prime mai mici sau egale decât un număr natural n citit de la tastatură. De exemplu, pentru $n = 20$ se vor afișa numerele $2, 3, 5, 7, 11, 13, 17$ și 19.

Rezolvare:
Testăm fiecare număr cuprins între 2 și n dacă este prim sau nu, iar în caz afirmativ îl afișăm.

```
#include <stdio.h>

int main()
{
    int n, i, d;

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

    printf("Numerele prime mai mici sau egale cu %d sunt:\n", n);
    for(i = 2; i <= n; i++)
    {
        for(d = 2; d * d <= i; d++)
            if(i%d==0)
                break;

        if(d*d > i)
            printf("%d\n", i);
    }

    return 0;
}
```

2. Scrieți un program care să citească de la tastatură un număr natural n și apoi să afișeze primele n numere prime. De exemplu, pentru $n = 20$ se vor afișa numerele $2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67$ și 71.

Rezolvare:
Testăm numere naturale, începând de la 2, până când găsim n numere prime.

```
#include <stdio.h>

int main()
{
    int n, i = 2, d, c;

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

    printf("Primele %d numerele prime sunt:\n", n);

    c = 0;
    while(c != n)
    {
        for(d = 2; d * d <= i; d++)
            if(i % d == 0)
                break;

        if(d * d > i)
        {
            printf("%d\n", i);
            c++;
        }

        i++;
    }

    return 0;
}
```

3. Scrieți un program care să citească de la tastatură un număr natural nenul k și apoi să afișeze cel mai mic număr natural n cu exact k divizori. De exemplu, pentru $k = 5$ se va afișa numărul 16, care are exact 5 divizori (1, 2, 4, 8 și 16), iar numerele naturale mai mici decât el au alt număr de divizori.

Rezolvare:
Testăm numere naturale, începând de la 1, până când îl găsim pe primul care are exact k divizori.

```c
#include <stdio.h>

int main()
{
    int k, n = 1, k_vf, d, nr;

    printf("Citim numarul k: ");
    scanf_s("%d", &k);

    nr = 1;
    while (1) // ciclare infinita
    {
        k_vf = 2;
        for (d = 2; d <= nr/2; d++)
            if (nr % d == 0)
                k_vf++;

        if (k == k_vf)
        {
            printf("Numarul este: %d\n", nr);
            break;
        }

        nr++;
    }

    return 0;
}
```

4. Scrieți un program care să valideze un cod numeric personal (CNP) citit de la tastatură (vezi pagina https://ro.wikipedia.org/wiki/Cod_numeric_personal_(Rom%C3%A2nia)).

Rezolvare:

```
  CNP =   2   9   9   1   1   1   0   0   3   8   6   5  0
CONST =   2   7   9   1   4   6   3   5   8   2   7   9
SUMA  = 2*2+7*9+9*9+1*1+1*4+1*6+0*3+0*5+3*8+2*8+6*7+5*9 =
      = 4+63+81+1+4+6+0+0+24+16+42+45 = 286:11 = 26, rest 0
```

```c
#include <stdio.h>

//CNP corect: 2991110038650

int main()
{

    unsigned long long cnp, masca = 279146358279, sum = 0;

    printf("CNP: ");
```

```
    scanf("%llu", &cnp);


    unsigned long long aux = cnp;
    unsigned int control = cnp % 10;
    cnp /= 10;

    while(cnp != 0)
    {
        sum += (masca % 10) * (cnp % 10);
        masca /= 10;
        cnp /= 10;
    }

    unsigned int rest = sum % 11;

    if(rest == 10)
        rest = 1;

    if(rest == control)
        printf("CNP-ul %llu este corect.", aux);
    else
        printf("CNP-ul %llu nu este corect.", aux);

    return 0;
}
```

5. Scrieți un program care afișează descompunerea în factori primi a unui număr natural nenul.

Rezolvare:

n = 2040

Numărul	Factor prim	Exponent	Descompunerea
2040	2	1	2^3
1020	2	2	
510	2	3	
255	3	1	3^1
85	4	0	—
85	5	1	5^1
17	6	0	—
17	7	0	—
...	...	...	...
17	17	1	17^1
1			

```
#include <stdio.h>

int main()
{
    int numar, factor, exponent;

    printf("Numarul: ");
    scanf("%d", &numar);

    printf("Descompunerea in factori primi a numarului %d:\n", numar);

    factor = 2;
    while(numar != 1)
    {
        exponent = 0;
        while(numar % factor == 0)
        {
            exponent++;
            numar = numar / factor;
        }

        if(exponent > 0)
            printf("%d^%d * ", factor, exponent);

        factor++;
    }

    //stergerea ultimului spatiu si a ultimului *
    //\b = backspace = sterge ultimul character afisat
    printf("\b\b \n");

    return 0;
}
```

6. Să se determine în câte cifre egale cu 0 se termină produsul a n numere naturale citite de la tastatură, fără a calcula produsul lor. De exemplu, produsul numerelor 130, 75, 244 și 2040 este egal cu 4853160000, deci se termină în 4 cifre egale cu 0.

Rezolvare:
O cifră egală cu 0 apare la sfârșitul produsului numerelor când se înmulțește un 2 cu un 5 => numărul cifrelor egale cu 0 de la sfârșitul produsului este egal cu minimul dintre exponenții la care apar 2 și 5 în descompunerea în factori primi a produsului.

$130 = 2^1 * 5^1 * 13^1$
$75 = 3^1 * 5^2$
$244 = 2^2 * 61^1$
$2040 = 2^3 * 3^1 * 5^1 * 17^1$

$130 * 75 * 244 * 2020 = 2^6 * 3^2 * 5^4 * 13^1 * 17^1 * 61^1$ => produsul se termină în $min\{6, 4\} = 4$ cifre egale cu 0

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

int main()
{
    int n,i, doi, cinci,num;

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

    doi = cinci = 0;
    for(i = 0; i < n; i++)
    {
        printf("Numar: ");
        scanf("%d",&num);

        if(num == 0)
        {
            doi = cinci = 1;
            break;
        }

        while(num % 2 == 0)
        {
            doi++;
            num = num/2;
        }

        while(num % 5 == 0)
        {
            cinci++;
            num = num/5;
        }
    }

    if(cinci <= doi)
        printf("Produsul se termina cu %d de zero la final", cinci);
    else
        printf("Produsul se termina cu %d de zero la final", doi);

    return 0;
}
```