# *Laboratorul 8 – Tipurile de date structură, uniune și enumerare*

```
/// 3.1. Folosind tipul structură pentru o dată curentă an, lună, zi, să se scrie un
program pentru a afișa a câtea zi din an este ziua respectivă și câte zile au mai rămas
până la sfârșitul anului.

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

typedef struct{ unsigned short zi, luna, an; } DataCalendaristica;

int eBisect(int an){
    if((an%4==0 && an%100!=0) || an%400==0) return 1;
    return 0;
}

int nrZileLuna(unsigned short l, unsigned short a){
    switch(l){
        case 2: return 28+eBisect(a);
        case 4: case 6: case 9: case 11: return 30;
    }
    return 31;
}

int aCataZi(DataCalendaristica d){
    int n=0;
    unsigned short l;
    for(l=1; l<d.luna; l++) n+=nrZileLuna(l, d.an);

    return n+d.zi;
}

int eCorecta(DataCalendaristica d){
    if(d.an<1600 || d.an>4900) return 0;
    if(d.luna<1 || d.luna>12) return 0;
    if(d.zi<1 || d.zi>nrZileLuna(d.luna, d.an)) return 0;

    return 1;
}

DataCalendaristica citesteData(){
    DataCalendaristica d;
    do{
        printf("Data (zz/ll/aaaa): ");
        scanf("%2hu/%2hu/%4hu", &d.zi, &d.luna, &d.an);
        fflush(stdin);
    } while(!eCorecta(d));

    return d;
}

int main()
{
    DataCalendaristica d;
```

```
    d=citesteData();

    int n=aCataZi(d);
    printf("Este a %d-a zi\n", n);
    printf("Au ramas %d zile\n", 365+eBisect(d.an)-n);

    return 0;
}
```

---

```
/// 3.2. Folosind tipul structură pentru data de naştere a D-voastră şi ştiind că în anul
curent vă aniversaţi ziua de naştere în ziua de x [luni, marţi, ..., duminică], scrieţi
un program pentru a afişa ziua (din săptămână) în care v-aţi născut.
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

typedef struct{ unsigned short zi, luna, an; } DataCalendaristica;

int eBisect(int an){
    if((an%4==0 && an%100!=0) || an%400==0) return 1;
    return 0;
}

int nrZileLuna(unsigned short l, unsigned short a){
    switch(l){
        case 2: return 28+eBisect(a);
        case 4: case 6: case 9: case 11: return 30;
    }
    return 31;
}

int aCataZi(DataCalendaristica d){
    int n=0;
    unsigned short l;
    for(l=1; l<d.luna; l++) n+=nrZileLuna(l, d.an);

    return n+d.zi;
}

int ceSecol(DataCalendaristica d){
    return d.an/100 + 1;
}

int eCorecta(DataCalendaristica d){
    if(d.an<1600 || d.an>4900) return 0;
    if(d.luna<1 || d.luna>12) return 0;
    if(d.zi<1 || d.zi>nrZileLuna(d.luna, d.an)) return 0;

    return 1;
}

DataCalendaristica citesteData(){
    DataCalendaristica d;
    do{
        printf("Data (zz/ll/aaaa): ");
        scanf("%2hu/%2hu/%4hu", &d.zi, &d.luna, &d.an);
```

```
        fflush(stdin);
    } while(!eCorecta(d));

    return d;
}

void ceZi(DataCalendaristica d){
    int zi, secol=ceSecol(d);

    zi=(int)(d.zi+floor(2.6*d.luna-0.2)-2*secol + d.an + floor(d.an/4) +
floor(secol/4))%7; /// formula de pe net

    switch(zi){
        case 0: printf("Duminica"); break;
        case 1: printf("Luni"); break;
        case 2: printf("Marti"); break;
        case 3: printf("Miercuri"); break;
        case 4: printf("Joi"); break;
        case 5: printf("Vineri"); break;
        case 6: printf("Sambata"); break;
    }
}

int main()
{
    DataCalendaristica d;
    d=citesteData();

    ceZi(d);

    return 0;
}
```

```c
/// 3.3. Să se scrie un program modularizat care citeşte datele legate de studenţii unei grupe: nume, data naşterii, adresa şi îi afisează în ordine crescătoare lexicografică.

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

typedef struct {char nume[30], adresa[60], data_nastere[10];} Student;

void bubble(Student *s, int n){
    int i, sortat=0;
    Student aux;

    while(!sortat){
        sortat=1;
        for(i=1; i<n; i++)
            if(strcmp(s[i-1].nume,s[i].nume)>0){
                aux=s[i-1];
                s[i-1]=s[i]; /// *(a+i-1)=*(a+i)
                s[i]=aux;
                sortat=0;
            }
        n--;
    }
}

Student citeste(int id){
    Student s;

    printf("\nStudentul %d: \n", id);

    printf("\tNume: ");
    gets(s.nume);

    printf("\tAdresa: ");
    gets(s.adresa);

    printf("\tData nasterii: ");
    gets(s.data_nastere);

    return s;
}

int main(){
    Student s[100];
    int n, i;

    printf("Introduceti numarul de elevi: ");
    scanf("%d%*c", &n);

    for(i=0; i<n; i++) s[i]=citeste(i);

    bubble(s, n);

    for(i=0; i<n; i++) printf("#%d \t%s\t\t%s\t\t%s\t\n", i, s[i].nume, s[i].adresa,
s[i].data_nastere);

    return 0;
}
```

```c
/// 3.4. Să se scrie un program pentru calculul valorii unui polinom de gradul n cu coeficienţi complecşi pentru o valoare complexă. Calculul se va face cu ajutorul unei funcţii.

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

int main()
{

    return 0;
}
```

```c
/// 3.5. Să se introducă tipul RAŢIONAL ca o structură formata din numărător şi numitor.
Să se scrie funcţii de simplificare, adunare, scădere, înmulţire, împărţire, ridicare la
putere.

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

typedef struct{ int numar, numit; } Rational;

int cmmdc(int a, int b){
    while(a!=b){
        if(a>b) a=a-b;
        else b=b-a;
    }
    return a;
}

void afis(Rational f){
    if(f.numar==0) printf("0");
    else if(f.numit==1) printf("%d", f.numar);
    else printf("%d/%d", f.numar, f.numit);
}

void simplifica(Rational *pf){
    int c=cmmdc(pf->numar, pf->numit); /// = (*pf).numar
    pf->numar/=c;
    pf->numit/=c;
}

Rational aduna(Rational f1, Rational f2){
    Rational q;
    q.numar=f1.numar*f2.numit + f2.numar*f1.numit;
    q.numit=f1.numit*f2.numit;

    simplifica(&q);

    return q;
}

Rational citesteFr(char nume){
    Rational f;
    do{
        printf("Fractia %c (a/b):", nume);
        scanf("%d/%d", &f.numar, &f.numit);
        fflush(stdin);
    } while(f.numit==0);

    if(f.numit<0){
        f.numar=-f.numar;
        f.numit=-f.numit;
    }
    simplifica(&f);

    return f;
}

int main()
{
    Rational p, q, r;
```

```
    p=citesteFr('p');
    q=citesteFr('q');
    r=aduna(p, q);
    afis(p);
    printf(" + ");
    afis(q);
    printf(" = ");
    afis(r);
    printf("\n");

    return 0;
}
```

---

```
/// 3.6. Folosind tipul uniune, care conţine câmpurile necesare pentru a putea reprezenta
un cerc, un dreptunghi, un pătrat, un triunghi, să se scrie o funcţie pentru a calcula
aria figurii respective.

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

typedef union { float l1, l2, l3, r, h; } Forma;

Forma citesteForma(char forma[20]){
    Forma f;

    if(strcmp(forma, "patrat")==0){
        printf("Introduceti latura: ");
        scanf("%f", &f.l1);
    }
    if(strcmp(forma, "dreptunghi")==0){
        printf("Introduceti latura 1: ");
        scanf("%f", &f.l1);

        printf("Introduceti latura 2: ");
        scanf("%f", &f.l2);
    }
    if(strcmp(forma, "triunghi")==0){
        printf("Introduceti latura 1: ");
        scanf("%f", &f.l1);

        printf("Introduceti latura 2: ");
        scanf("%f", &f.l2);

        printf("Introduceti latura 3: ");
        scanf("%f", &f.l3);

        printf("Introduceti inaltime: ");
        scanf("%f", &f.h);
    }
    if(strcmp(forma, "cerc")==0){
        printf("Introduceti raza: ");
        scanf("%f", &f.r);
    }

    return f;
```

```
}

float calculeazaArie(char forma[20], Forma f){
    if(strcmp(forma, "patrat")==0) return f.l1*f.l1;
    if(strcmp(forma, "dreptunghi")==0) return f.l1*f.l2;
    if(strcmp(forma, "triunghi")==0) return f.l3*f.h;
    if(strcmp(forma, "cerc")==0) return M_PI*f.r*f.r;
}


int main()
{
    char forma[20];
    Forma f;

    printf("Ce forma doriti sa cititi? (patrat, dreptunghi, triunghi sau cerc): ");
    gets(forma);

    f=citesteForma(forma);

    printf("%f", calculeazaArie(forma, f));

    return 0;
}
```

```c
/// 3.7. Folosind tipul enumerare, să se introducă tipul boolean. Să se scrie o funcţie de ordonare crescătoare a unui şir folosind metoda bulelor şi un semafor de tipul boolean.

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

typedef enum {false, true} bool;

void bubble(int *a, int n){
    int i, aux;
    bool sortat=false;

    while(!sortat){
        sortat=true;
        for(i=1; i<n; i++)
            if(a[i-1]>a[i]){
                aux=a[i-1];
                a[i-1]=a[i]; /// *(a+i-1)=*(a+i)
                a[i]=aux;
                sortat=false;
            }
        n--;
    }
}

int main()
{
    int a[100], n, i;

    printf("Introduceti n: ");
    scanf("%d", &n);

    for(i=0; i<n; i++) scanf("%d", &a[i]);

    bubble(a, n);

    for(i=0; i<n; i++) printf("%d ", a[i]);

    return 0;
}
```

```c
/// 3.8. Se citeşte un şir de caractere format din litere şi cifre. Să se indice
frecvenţa de apariţie a caracterelor întâlnite în şir folosind o listă ordonată alfabetic
(nodul conţine caracterul, frecvenţa şi adresa următorului nod).

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

typedef struct { char caracter; int aparitii; } Litera;

int main()
{
    char c[1000];
    Litera l[2000];
    int i;

    printf("Introduceti sirul de caractere: ");
    gets(c);

    for(i='0'; i<='z'; i++) l[c[i]].aparitii=0;

    for(i=0; i<strlen(c); i++) l[(int)c[i]].aparitii++;

    for(i='0'; i<='z'; i++)
        if((isalpha((char)i) || isdigit((char)i)) && l[i].aparitii!=0) printf("%c apare
de %d ori\n", i, l[i].aparitii);

    return 0;
}
```