Laboratorul 5 – Funcții

```
/// 3.2. Se citeşte gradul unui polinom şi coeficienţii săi, care sunt numere întregi: în care p0 este nenul. Ştiind că polinomul admite numai rădăcini întregi simple, să se găsească rădăcinile polinomului.

#include <stdio.h>
#include <stdlib.h>
#define GRAD_MAX 20

int citpol(int a[]);
int val_polinom(int x, int n, int a[]);
int afisRad(int n, int a[]);

int main(){

    int p[GRAD_MAX+1], grad;

    grad=citPol(p);
    printf("Radacinile intregi: ");
    if(afisRad(grad, p)==0) printf("Nu exista radacini intregi\n");

    return 0;
}

int citPol(int a[]){
    int n, i;

    printf("Gradul (<=%d): ", GRAD_MAX);
    scanf("%d", &n);
    printf("Coeficientii polinomului: \n");
    for(i=n; i>=0; i--){
        printf("a[%d] = ", i);
        scanf("%d", &a[i]);
    }

    return n;
}

int val_polinom(int x, int n, int a[]){
    int i, v=0;
    for(i=n; i>=0; i--) v=v*x+a[i];

    return v;
}

int afisRad(int n, int a[]){
    int nrRad=0, d;

    for(d=1; d<=abs(a[0]); d++)
        if(a[0]%d==0){
            if(val_polinom(d, n, a)==0){
                nrRad++;
                printf("%d ", d);
```

```
            }
        }

    if(val_polinom(-d, n, a)==0){
        nrRad++;
        printf("%d ", -d);
    }

    return nrRad;
}
```

```
/// 3.3. Se citeşte n şi perechile de numere întregi (xi, yi) reprezentând o relaţie
binară R peste mulţimea M. Admiţând că fiecare element din mulţimea M apare în cel puţin
o pereche dintre cele citite, să se determine mulţimea M. Să se verifice dacă relaţia R
este o relaţie de echivalenţă (reflexivă, simetrică şi tranzitivă).

#include <stdio.h>
#include <stdlib.h>
#define N 20

int exista(int x, int a[], int m){
    int i;

    for(i=0; i<m; i++)
        if(a[i]==x) return 1;

    return 0;
}

int exista_in_R(int x, int y, int R[][2], int n){
    int i;
    for(i=0; i<n; i++)
        if(R[i][0]==x && R[i][1]==y) return 1;

    return 0;
}

int citRel(int R[][2]){
    int n, i;
    scanf("%d", &n);
    printf("Elementele relatiei: \n");

    for(i=0; i<n; i++){
        printf("[Perechea %d]: ", i);
        scanf("%d%d", &R[i][0], &R[i][1]);
    }

    return n;
}

int detMult(int R[][2], int n, int M[]){
    int m=0, i, j;

    for(i=0; i<n; i++)
        for(j=0; j<2; j++)
            if(!exista(R[i][j], M, m)) M[m++]=R[i][j];
}
```

```
int refl(int R[][2], int n, int M[], int m){
    int i;

    for(i=0; i<m; i++)
        if(!exista_in_R(M[i], M[i], R, n)) return 0;

    return 1;
}

int sim(int R[][2], int n){
    int i;

    for(i=0; i<n; i++)
        if(!exista_in_R(R[i][1], R[i][0], R, n)) return 0;

    return 1;
}

int tranz(int R[][2], int n){
    int i, j;

    for(i=0; i<n; i++)
        for(j=0; j<n; j++)
            if(R[i][1]==R[j][0])
                if(!exista_in_R(R[i][0], R[j][1], R, n)) return 0;

    return 1;
}

int main()
{

    int R[N][2], M[2*N], n, m;

    n=citRel(R);
    m=detMult(R, n, M);

    if(refl(R, n, M, m)) printf("Este reflexiva\n");
    else printf("Nu este reflexiva\n");

    if(sim(R, n)) printf("Este simetrica\n");
    else printf("Nu este simetrica\n");

    if(tranz(R, n)) printf("Este tranzitiva\n");
    else printf("Nu este tranzitiva\n");

    if(refl(R, n, M, m) && sim(R, n) && tranz(R, n)) printf("Relatie de echivalenta");

    return 0;
}
```

```
/// 3.4. Se dau două şiruri de caractere care reprezintă numere întregi zecimale foarte
mari. Să se scrie un program de efectuare a operaţiilor aritmetice asupra lor.

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define NMAX 100

void Shl(int a[], int k){
  memmove(&a[k+1], &a[1], sizeof(int)*a[0]); /// Shifteaza vectorul cu k pozitii
  memset(&a[1], 0, sizeof(int)*k); /// Umple primele Count pozitii cu 0
  a[0]+=k; /// Incrementeaza numarul de cifre
}

int Sgn(int a[], int b[]) {
  while (a[0] && !a[a[0]]) a[0]--; /// Elimina zero-urile semnificative, daca exista.
  while (b[0] && !b[b[0]]) b[0]--;

  if (a[0]<b[0]) return -1;
  else if(a[0]>b[0]) return 1;

  for(int i=a[0]; i>0; i--) {
    if(a[i]<b[i]) return -1;
    else if (a[i]>b[i]) return 1;
  }

  return 0;
}

void citire(char s[NMAX], int a[NMAX], int k){
    int i, n;

    printf("Introduceti numarul %d: ", k);
    gets(s);

    n=strlen(s);

    for(i=n; i>0; i--) a[i]=(int)(s[n-i]-'0');
    a[0]=n; /// punem numarul de cifre astfel incat sa fie usor de accesat
}

void afisare(int a[NMAX], int n){
    int i;

    for(i=n; i>0; i--) printf("%d", a[i]);
}

void suma(int a[], int b[], int s[]){
    int i, cifra, carry=0, max;
    /// completam numarul cel mai mic cu zero-uri nesemnificative la sfarsit
    if(a[0]<b[0]){
        max=b[0];
        for(i=a[0]+1; i<=b[0]; i++) a[i]=0;
    }
    else {
        max=a[0];
        for(i=b[0]+1; i<=a[0]; i++) b[i]=0;
    }
```

```
    for(i=1; i<=max; i++){
        cifra=a[i]+b[i]+carry;
        s[i]=cifra%10;
        carry=cifra/10;
    }
    if(carry!=0) s[i]=carry; /// aici i-ul va fi trecut deja la urmatoarea pozitie din
stanga numarului
    else i--; /// daca nu exista carry revenim la ultima cifra calculata

    s[0]=i;

}

void diferenta(int a[], int b[], int d[]){
    int i, carry=0;

    if(a[0]<b[0]) diferenta(a, b, d);
    else{
        for(i=1; i<=a[0]; i++){
            d[i]=a[i]-b[i]+carry;
            if(d[i]<0) d[i]+=10, carry=-1;
            else carry=0;
        }
        i--;
        while(i && !d[i]) i--;

        d[0]=i;
    }
}

void produs(int a[], int b[], int p[]){
    int i, j, carry=0, n;

    p[0] = a[0]+b[0];

    for(i=1; i<=p[0];) p[i++]=0;
    for(i=1; i<=a[0]; i++)
        for(j=1; j<=b[0]; j++)
            p[i+j-1]=p[i+j-1]+a[i]*b[j];

    for(i=1; i<=p[0]; i++){
        p[i]=p[i]+carry;
        carry=p[i]/10;
        p[i]%=10;
    }
    if(carry) p[++p[0]]=carry;
}

void impartire(int a[], int b[], int c[], int r[]){
    int i;

    r[0]=0;
    c[0]=a[0];

    for(i=a[0]; i>0; i--){
        Shl(r, 1);
        r[1]=a[i];
        c[i]=0;
        while(Sgn(b, r)!=1){
```

```
            c[i]++;
            diferenta(r, b, r);
        }
    }

    while(!c[c[0]] && c[0]>1) c[0]--;
}

int main()
{

    char charA[NMAX], charB[NMAX];
    int a[NMAX], b[NMAX], s[NMAX], d[NMAX], p[NMAX], r[NMAX], imp[NMAX], k, i, n, max;

    citire(charA, a, 1);
    citire(charB, b, 1);

    suma(a, b, s);
    printf("A + B = "); afisare(s, s[0]);
    printf("\n");

    diferenta(a, b, d);
    printf("A - B = "); afisare(d, d[0]);
    printf("\n");

    produs(a, b, p);
    printf("A * B = "); afisare(p, p[0]);
    printf("\n");

    impartire(a, b, imp, r);
    printf("A : B = "); afisare(imp, imp[0]);
    printf(", rest "); afisare(r, r[0]);

    return 0;
}
```

```c
/// 3.5. Să se scrie funcţiile pentru adunarea, scăderea şi înmulţirea a două matrice şi
apoi să se calculeze A=B*C-2*(B+C), unde B şi C sunt două matrice pătratice de ordinul n.

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

void citeste(int a[100][100], int *n, char c){
    int i, j;

    printf("Citeste matricea %c: \n", c);

    printf("\tNumarul de linii si coloane: ");
    scanf("%d", n);

    for(i=0; i<*n; i++)
        for(j=0; j<*n; j++){
            printf("%c[%d][%d] = ", c, i, j);
            scanf("%d", &a[i][j]);
        }
}

void afiseaza(int a[100][100], int n){
    int i, j;

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

        printf("\n");
    }
}

void adunare(int a[100][100], int b[100][100], int c[100][100], int n){
    int i, j;

    for(i=0; i<n; i++)
        for(j=0; j<n; j++)
            c[i][j]=a[i][j]+b[i][j];
}

void scadere(int a[100][100], int b[100][100], int c[100][100], int n){
    int i, j;

    for(i=0; i<n; i++)
        for(j=0; j<n; j++)
            c[i][j]=a[i][j]-b[i][j];
}

void inmultireScalar(int a[100][100], int n, int nr){
    int i, j;

    for(i=0; i<n; i++)
        for(j=0; j<n; j++)
            a[i][j]=a[i][j]*nr;
}

void inmultire(int a[100][100], int b[100][100], int c[100][100], int n){
    int i, j, k;
```

```
    for(i=0; i<n; i++)
        for(j=0; j<n; j++){
            c[i][j]=0;
            for(k=0; k<n; k++)
                c[i][j]=c[i][j]+a[i][k]*b[k][j];
        }
}

int main()
{
    int a[100][100], b[100][100], c[100][100], *n;

    citeste(b, &n, 'B');
    citeste(c, &n, 'C');

    inmultire(b, c, a, n); /// B*C
    adunare(b, c, b, n); /// B+C
    inmultireScalar(b, n, 2); /// 2*(B+C)
    scadere(a, b, a, n); /// B*C - 2*(B+C)

    afiseaza(a, n);

    return 0;
}
```

```
/// 3.6. Să se scrie funcţia care realizează operaţiile aritmetice asupra a două matrice
rare (matricea rară este o matrice de dimensiune mare, care are majoritatea elementelor
nule).

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

void citeste(int a[100][100], int *n, int *m, char c){
    int i, j;

    printf("Citeste matricea %c: \n", c);

    printf("\tNumarul de linii: ");
    scanf("%d", n);

    printf("\tNumarul de coloane: ");
    scanf("%d", m);

    for(i=0; i<*n; i++)
        for(j=0; j<*m; j++){
            printf("%c[%d][%d] = ", c, i, j);
            scanf("%d", &a[i][j]);
        }
}

void afiseaza(int a[100][100], int n, int m){
    int i, j;

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

        printf("\n");
```

```
    }
}

void adunare(int a[100][100], int b[100][100], int c[100][100], int nA, int mA, int nB,
int mB){
    int i, j;

    if(nA!=nB || mA!=mB){
        printf("Matricele nu sunt patratice");
        exit(1);
    }
    else{
        for(i=0; i<nA; i++)
            for(j=0; j<mA; j++)
                c[i][j]=a[i][j]+b[i][j];
    }
}

void scadere(int a[100][100], int b[100][100], int c[100][100], int nA, int mA, int nB,
int mB){
    int i, j;

    if(nA!=nB || mA!=mB){
        printf("Matricele nu sunt patratice");
        exit(1);
    }
    else{
        for(i=0; i<nA; i++)
            for(j=0; j<mA; j++)
                c[i][j]=a[i][j]-b[i][j];
    }
}

void inmultireScalar(int a[100][100], int nA, int mA, int nr){
    int i, j;

    for(i=0; i<nA; i++)
        for(j=0; j<mA; j++)
            a[i][j]=a[i][j]*nr;
}

void inmultire(int a[100][100], int b[100][100], int c[100][100], int nA, int mA, int nB,
int mB){
    int i, j, k;

    for(i=0; i<nA; i++)
        for(j=0; j<mA; j++){
            c[i][j]=0;
            for(k=0; k<nB; k++)
                c[i][j]=c[i][j]+a[i][k]*b[k][j];
        }
}


int main()
{
    int a[100][100], b[100][100], c[100][100], *nA, *mA, nB, mB;

    citeste(a, &nA, &mA, 'A');
```

```
    citeste(b, &nB, &mB, 'B');

    printf("A + B\n");
    adunare(a, b, c, nA, mA, nB, mB);
    afiseaza(c, nA, mB);

    printf("\nA - B\n");
    scadere(a, b, c, nA, mA, nB, mB);
    afiseaza(c, nA, mB);

    printf("\nA * B\n");
    inmultire(a, b, c, nA, mA, nB, mB);
    afiseaza(c, nA, mB);

    return 0;
}
```

```
/// 3.7. Fiind date anul, luna, ziua, să se scrie o funcţie care să returneze a câtea zi
din an este ziua respectivă şi câte zile au mai rămas din anul respectiv.

#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;
}
```

```
/// 3.8. Să se scrie o funcţie care primind ca parametru un număr roman sub forma unui
şir de caractere, returnează numărul respectiv ca număr arab în baza 10.

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

int convert(const char ch){
    switch(ch){
        case 'M': return 1000;
        case 'D': return 500;
        case 'C': return 100;
        case 'L': return 50;
        case 'X': return 10;
        case 'V': return 5;
        case 'I': return 1;
        default: return 0;
    }
}

int transformare(char c[100]){
    int i, rez=0;

    for(i=0; i<strlen(c); i++)
        if(convert(c[i])>convert(c[i-1])) rez=rez+(convert(c[i])-2*convert(c[i-1]));
        else rez=rez+convert(c[i]);

    return rez;
}

int main(){
    int i;
    char c[100];
    printf("Introduceti numarul in cifre romane: ");
    gets(c);

    printf("%s in litere arabe: %d", c, transformare(c));

    return 0;
}
```

```
/// 3.9. Să se scrie o funcţie care primind ca parametru un număr arab în baza 10,
calculează şirul de caractere ce reprezintă numărul respectiv sub formă romană.

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

void toRoman(int nr){
    int i=0;
    char *rom[]={"M", "CM", "D", "CD", "C", "XC", "L", "XL", "X", "IX", "V", "IV", "I"};
    int arab[14]={1000, 900, 500, 400, 100, 90, 50, 40, 10, 9, 5, 4, 1};

    for(i=0; i<12; i++){
        while(nr>=arab[i]){
            nr-=arab[i];
            printf("%s", rom[i]);
        }
    }
}

int main()
{
    int nr;

    printf("Numar in cifre arabe: ");
    scanf("%d", &nr);

    toRoman(nr);

    return 0;
}
```

```
/// 3.10. Se citeşte un număr întreg, reprezentând o sumă de bani. Să se găsească numărul
minim de bancnote româneşti necesare pentru plata sumei respective.

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

void suma(int n){
    int bancnote[]={1, 5, 10, 50, 100, 200, 500};
    int i;

    for(i=6; i>=0; i--){
        if(n>=bancnote[i]){
            printf("%d bancnote de %d\n", n/bancnote[i], bancnote[i]);
            n=n-(n/bancnote[i])*bancnote[i];
        }
    }
}

int main(){
    int s;
    printf("Introduceti suma: ");
    scanf("%d", &s);

    suma(s);

    return 0;
}
```

```c
/// 3.11. Să se scrie o funcție pentru a verifica dacă un șir de caractere este subșir al altui șir de caractere. În caz afirmativ funcția trebuie să returneze poziția la care începe subșirul, iar în caz negativ trebuie să returneze valoarea -1.

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define N 100

int pozSubsir(const char *s, const char *subsir){
    char *p=strstr(s,subsir);

    if(p!=NULL) return p-s;
    else return -1;
}

int main()
{
    int poz, nrAparitii=0, i=0;
    char s[N], subsir[N];

   printf("Introduceti sirul: ");
   gets(s);

   printf("Introduceti subsirul cautat: ");
   gets(subsir);

   while((poz=pozSubsir(s+i, subsir))>=0)
   {
       nrAparitii++;
       printf("Subsirul apare pe pozitia: %d ", poz+i);
       i+=poz+1;
   }

   if(nrAparitii==0) printf("Subsirul nu apare");

   return 0;
}
```

```c
/// 3.12. Să se scrie o funcție pentru a verifica dacă parametrul ei (un întreg pozitiv) este pătrat perfect. Utilizați funcția pe un șir de întregi pozitivi cu scopul extragerii tuturor pătratelor perfecte într-un alt șir.

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

int patratPerfect(int n){
    if((int)sqrt(n)==sqrt(n)) return 1;
    return 0;
}

int main()
{
    int n;

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

    if(patratPerfect(n)) printf("%d este patrat perfect", n);
    else printf("%d nu este patrat perfect", n);

    return 0;
}
```

```c
/// 3.13. Să se scrie o funcție care verifică dacă parametrul ei (un întreg pozitiv) este număr perfect. Numărul perfect este un număr întreg egal cu suma divizorilor săi, din care se exclude numărul însuși.

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

int sDiv(int n){
    int i, s=1;
    for(i=2; i<=n/2; i++)
        if(n%i==0) s+=i;

    return s;
}

void perfect(int n){
    if(sDiv(n)==n) printf("%d este numar perfect", n);
    else printf("%d nu este numar perfect", n);
}

int main()
{
    int n;
    printf("Introduceti numarul: ");
    scanf("%d", &n);

    perfect(n);

    return 0;
}
```