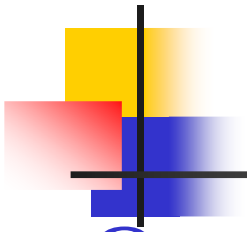

Aplicaţii

1. Operații pe biți
2. Șiruri de caractere
3. Operații cu vectori
4. Operații cu matrici
5. Operații cu structuri
6. Prelucrarea fișierelor
7. Recursivitate. Metode generale de proiectare a algoritmilor

Aplicaţii
Operaţii pe biţi

```
// Pentru o valoare intreaga citita de la tastatura sa se
// tipareasca valoarea corespunzatoare in binar
#include <stdlib.h>
#include <stdio.h>
int main() {
    int x, i, dim=sizeof(i)*8, masca=1;
    printf("Introduceti numarul\n");
    scanf("%d", &x);
    printf("reprezentarea in binar este:\n");
    for(i=dim-1; i>=0; i--)
       putchar(x & masca << i ? '1' : '0');
       //  << are precedenta mai mare decat &
    printf("\n");
    system("pause");
    return 0;
}
```

Aplicaţii
Operaţii pe biţi

```
// Fie x un numar natural.
// Utilizand operatorii pe biti,
// sa se scrie expresiile care efectuaza urmatoarele:
// a. inmulteste valoarea variabilei x cu 2^n
      x = x << n
// b. imparte valoarea x cu 2^n
      x = x >> n
// c. furnizeaza valoarea 1 daca si numai daca x este impar
      x & 1
// d. furnizeaza valoarea 0 daca si numai bitul n din x este 0
      x & (1 << n)
// d. anuleaza bitul n din x
      x = x & ~(1 << n)
// e. seteaza bitul n din x
      x = x | (1 << n)
// f. schimba bitul n din x
      x = x ^ (1 << n)
```

Aplicaţii
Şiruri de caractere

- **Câteva funcţii pentru lucrul cu şiruri de caractere:**

Fişierul antet **<string.h>**

char *strcpy(s, ct) — copiază şirul ct în şirul s, inclusiv caracterul '\0'; returnează s.

char *strncpy(s, ct, n) — copiază cel mult n caractere din şirul ct în şirul s, inclusiv caracterul '\0'; returnează s. Completează cu caractere nule '\0' dacă ct are mai puţin de n caractere.

char *strcat(s, ct) — concatenează şirul ct la sfârşitul s; returnează s.

char *strncat(s, ct, n) — concatenează cel mult n caractere din ct la sfârşitul s; termină şirul s cu '\0' ; returnează s.

Aplicaţii
Şiruri de caractere

int strcmp(cs,ct) compară şirul cs cu şirul ct; returnează o valoare <0 dacă cs<ct, 0 dacă cs==ct sau >0 dacă cs>ct.

int strcncmp(cs,ct,n) compară cel mult n caractere din şirul cs cu şirul ct; returnează o valoare <0 dacă cs<ct, 0 dacă cs==ct sau >0 dacă cs>ct.

char *strchr(cs,c) returnează un pointer spre prima apariție a lui c în cs, sau NULL dacă acesta nu apare.

char *strrchr(cs,c)returnează un pointer spre ultima apariție a lui c în cs, sau NULL dacă acesta nu apare.

size_t strlen(cs) returnează lungimea lui cs.

Aplicaţii
Şiruri de caractere

- Numărarea caracterele dintr-un şir până la caracterul NULL (determinarea lungimii unui şir de caractere).

```
size_t strlen(const char *sir)
{
   int i = 0;
   while (sir[ i ])
       i++;
   return (i);
}
```

Aplicaţii
Şiruri de caractere

- Copierea şirului sursă în şirul destinație.

```
char *strcpy(char *destinatie, const char *sursa)
{
   char *initial = destinatie;
   while (*destinatie++ = *sursa++)
       ;
   return (initial);
}
```

Aplicaţii
Şiruri de caractere

- Adăugarea conținutului unui şir la alt şir (concatenarea şirurilor).

```
char *strcat(char *destinatie, const char *sursa)
{
   char *initial = destinatie;
   while (*destinatie)
      destinatie++;   /* cauta sfarsitul sirului */
   while (*destinatie++ = *sursa++)
      ;
   return (initial);
}
```

Aplicaţii
Operaţii cu vectori

```
// Fie a un vector cu n (n<=50) componente de tip int.
// a. Verificati daca exista in vectorul a un palindrom
// b. Verificati daca toate elementele vectorului sunt nr. prime
#include<stdio.h>
#include <stdlib.h>
#include <math.h>
int main()  {
   int a[50], x, rx, n, i, exista, ok, d, rad;
   printf("n="); scanf("%d", &n);
   printf("Introduceti elementele vectorului\n");
   for(i=0; i<n; i++)
      scanf("%d", &a[i]);
```

Aplicaţii
Operaţii cu vectori

```
for (exista=i=0; !exista && i<n; i++)  {
     // verific daca a[ i ] este palindrom
     x=a[i];
     rx=0;  // “inversul” lui x – in oglinda
     while (x)  {
        rx=rx*10 + x%10;
        x /= 10;   }
     if (rx == a[ i ])  exista=1;
}
   if (exista)
      printf("exista un palindrom in vector \n");
   else
      printf("nu exista un palindrom in vector \n");
```

Aplicaţii
Operaţii cu vectori

```
for (ok=1, i=0; ok && i<n; i++)  {
      // verific daca a[ i ] este prim
      for (d=2, rad=sqrt(a[ i ]); a[ i ]%d && d<=rad; d++);
      if (a[ i ] % d == 0) ok=0;  // a[ i ] nu este prim
}
   if (ok)
      printf("Toate elementele vectorului sunt prime\n");
   else
      printf("Nu toate elementele vectorului sunt
prime\n");
   system("pause");
}
```

Aplicaţii

Operaţii cu matrici

```
// Fie a o matrice de intregi cu n linii si m coloane(n, m<=100)
//  Numarati cate elemente din matrice au ca vecini numai
//  numere pare (doua elemente sunt vecine daca sunt
//  adiacente pe verticala sau orizontala
#include<stdio.h>
#include <stdlib.h>
#define NVecini 4
#define DMAX 102
int a[DMAX][DMAX], n, m;
int dl[ NVecini ]={-1, 0, 1, 0};
int dc[ NVecini ]={0, 1, 0, -1};
int main()
{  int i, j, k, nr, total=0;
   printf("Introduceti nr de linii si coloane\n");
   scanf("%d%d", &n, &m);
```

Aplicaţii

Operaţii cu matrici

```
printf("Introduceti elementele matricii\n");
  for (i=1; i<=n; i++)
    for (j=1; j<=m; j++)  {
        printf("a[%d,%d]=", i, j);
        scanf("%d", &a[ i ][ j ]);  }
  for (i=1; i<=n; i++)
    for (j=1; j<=m; j++)
      {   // parcurg vecinii si-i numar pe cei pari
        for (nr=0, k=0; k<NVecini; k++)
           if (a[i+dl[k]][j+dc[k]] % 2 == 0) nr++;
        // daca toti vecinii sunt pari, numar acest element
        if (nr == NVecini) total++;
        }
  printf("Numarul de elemente cu vecini pari:%d\n", total);
  system("pause");
  return 0;
}
```

Aplicaţii

Operaţii cu structuri

```
// Se citesc de la tastatura numele si cele 4 note obtinute de
// studenti in sesiune
// Sa se calculeze media fiecarui student promovat
// Sa se afiseze, in ordine alfabetica, toti studentii.
// Sa se afiseze studentii nepromovati, tot in ordine alfabetica
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define NMAX 30
struct student{
    char *nume;
    int note[4];
    float media;
} s[NMAX];
int n;
```

Aplicaţii
Operaţii cu structuri

```
void citire()  {
    int i, j, cond;
    char vnume[30];
    printf("Numarul de studenti:"); scanf("%d",&n);
    printf("Introduceti datele celor %d studenti\n",n);
    for (i=0; i<n; i++)  {
        fflush(stdin);  // goleste bufferul de intrare
        printf("\nStudentul %d\n",i+1);
        printf("Numele:"); gets(vnume);
        s[i].nume=(char *)malloc(strlen(vnume)+1);
        strcpy(s[i].nume, vnume);
```

Aplicaţii
Operaţii cu structuri

```
printf("Notele:");
        for (j=0; j<4; j++)
           scanf("%d",&s[ i ].note[ j ]);
        s[ i ].media=0;
        cond=1;
        for (j=0; j<4; j++)
           if(s[ i ].note[ j ]<5) cond=0;
        if (cond)
           for (j=0; j<4; j++)
              s[ i ].media+=s[ i ].note[ j ];
        s[ i ].media /=4.0;
    }
}
```

Aplicaţii
Operaţii cu structuri

```
void afisare()
  {
    int i,j;
    for (i=0; i<n; i++){
       printf("%-30s", s[ i ].nume);
       for (j=0; j<4; j++)
           printf("  %2d",s[ i ].note[ j ]);
       printf("%8.2f\n",s[ i ].media);
    }
  }
```

Aplicaţii
Operaţii cu structuri

```
void sortare_alf()
{ int i,j, ok;
  struct student aux;
  do  {
     for(i=0, ok=1; i<n-1; i++)
        if (strcmp(s[ i ].nume, s[i+1].nume)>0)  {
             aux=s[ i ];
             s[ i ]=s[i+1];
             s[i+1]=aux;
             ok=0;
             }
  } while (!ok);
}
```

Aplicaţii
Operaţii cu structuri

```
void afis_nepromovati()
{
        int i, j;

        for (i=0; i<n; i++){
           if(s[ i ].media<5) {
                printf("%-30s", s[ i ].nume);
                for (j=0; j<4; j++)
                    printf("  %2d",s[ i ].note[ j ]);
                printf("%8.2f\n",s[ i ].media);
           }
        }
}
```

Aplicaţii
Operaţii cu structuri

```
int main()  {
    citire();
    printf("Datele studentilor\n");
    afisare();
    printf("\nStudentii in ordine alfabetica\n");
    sortare_alf();
    afisare();
    printf("\nStudentii nepromovati\n");
    afis_nepromovati();
    system("pause");
    return 0;
}
```

Aplicaţii
Prelucrarea fişierelor

- **Problemă:** Să se scrie un program care afişează lungimea celei mai lungi linii din fişierul text numit TEST.txt.

```
#include <stdio.h>
void main(void) {
  FILE *f;
  int c;
  int lg_max, lg_curenta;
  lg_max=lg_curenta=0;
   if(!(f=fopen(“TEST.txt”,”r”))) {
      puts(“Fisierul TEST nu poate fi deschis”);
      return;  }
```

Aplicaţii
Prelucrarea fişierelor

```
while ((c=getc(f))!=EOF)
    if (c==‘\n’)  {
          if(lg_max < lg_curenta) lg_max=lg_curenta;
          lg_curenta=0;  }
      else
          lg_curenta++;
   fclose(f);
   printf(“\nLinia cea mai lunga are lung. %d”, lg_max);
}
```

Aplicaţii
Prelucrarea fişierelor

- **Problemă**: Să se scrie un program care copiază un fişier binar *sursă* în alt fişier binar *destinație*.

```
#include <stdio.h>
void main(void)
{
   FILE *fs, *fd;
   char c;
   if ((fs=fopen(“sursa”,”rb”))==NULL) {
      fprintf(stderr, “Fisierul sursa nu poate fi deschis\n”);
      return;
   }
```

Aplicaţii
Prelucrarea fişierelor

```
   if ((fd=fopen(“dest”,”wb”))==NULL) {
      fprintf(stderr, “Fisierul dest. nu poate fi deschis\n”);
      return;
   }
   c=getc(fs);
   while (!feof(fs))  {
       putc(c, fd);
       c=getc(fs);  }
   fclose(fs);
   fclose(fd);
}
```

Aplicaţii
Prelucrarea fişierelor

- **Problemă**: Mai multe persoane au participat la un concurs şi au obținut diferite punctaje. Să se realizeze o aplicație care să permită stocarea datelor persoanelor într-o bază de date (un fişier cu structuri), să se poată adăuga noi persoane, să se caute şi să se modifice punctajul unei persoane şi să se ordoneze persoanele, descrescător după punctaj.

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
struct persoana {
    char nume[20];
    int punctaj;  };
```

Aplicaţii
Prelucrarea fişierelor

```
void Citeste (struct persoana *p){
    fflush(stdin);
    printf(“Dati numele: ”);  gets((*p).nume);
    printf(“Dati punctajul lui %s: “, (*p).nume);
    scanf(“%d”, &(*p).punctaj);
    fflush(stdin);
}
```

Aplicaţii
Prelucrarea fişierelor

```
void Adauga (char nume_fis[13]) {
    FILE *f;
    struct persoana p;
    if (!strcmp(nume_fis, “”)) return;
    if ((f = fopen(nume_fis, “ab”)) == NULL) {
      printf(“\nNu se poate deschide fis. %s.\n”,nume_fis);
      return;
    }
    Citeste(&p);
    fwrite(&p, sizeof(p), 1, f);
    fclose(f);
}
```

Aplicaţii
Prelucrarea fişierelor

```
void CautaSiModifica(char nume_fis[13]) {
    FILE *f;
    struct persoana p, q;
    int gasit=0;  long poz;
    if (!strcmp(nume_fis, “”)) return;
    if ((f = fopen(nume_fis, “r+b”)) == NULL)
      { printf(“\nNu se poate deschide %s.\n”,nume_fis);
        return;
        }
    Citeste(&p);
```

Aplicaţii
Prelucrarea fişierelor

```
while ( !feof(f) && ( !gasit))  {
        poz=ftell(f);
        fread(&q, sizeof(struct persoana), 1, f);
        if (!strcmp(p.nume, q.nume))  {
                gasit=1;  fseek(f, poz, SEEK_SET);
                fwrite(&p, sizeof(p), 1, f);   }
    }
    fclose(f);
    if(!gasit)  printf("\nPersoana inexistenta.\n");
    else
        printf("Persoana avea punctajul: %d si acum are:
                %d.\n", q.punctaj, p.punctaj);
}
```

Aplicaţii
Prelucrarea fişierelor

```
long FileSize (FILE *f)  {
    long poz_curenta, lungime;
    poz_curenta = ftell(f);   fseek(f, 0L, SEEK_END);
    lungime = ftell(f);    fseek(f, poz_curenta, SEEK_SET);
    return lungime;  }
void Listeaza (char nume_fis[13])   {
    struct persoana p;
    FILE *f;   long i=0;
    if (!strcmp(nume_fis, “”)) return;
    if ((f = fopen(nume_fis, “rb”)) == NULL)
        { printf(“\nNu se poate deschide fisierul.\n”);   return;}
```

Aplicaţii
Prelucrarea fişierelor

```
while ( !feof(f))  {
   fread(&p, sizeof(struct persoana), 1, f);
   if (!feof(f))printf("%ld, %s->%d\n", ++i, p.nume, p.punctaj);
   if (wherey() == 20)
         {   printf("\nApasati o tasta pentru continuare …\n");
             getch();  clrscr(); }
    }
    printf("\nIn total: %ld persoane.\n",
                FileSize(f)/sizeof(struct persoana));
    fclose(f);
}
```

Aplicaţii
Prelucrarea fişierelor

```
void Sorteaza(char nume_fis[13]){
    FILE *f;
    struct persoana p, q;
    size_t lung_pers = sizeof(struct persoana);
    long poz, i, lungime_fisier;
    int ordonat;
    if (!strcmp(nume_fis, “”)) return;
    if ((f = fopen(nume_fis, “r+b”)) = = NULL)
        { printf(“\nNu se poate deschide fisierul. \n”);
          return;  }
    lungime_fisier = FileSize(f) / lung_pers;
```

Aplicaţii
Prelucrarea fişierelor

```
do   {
     ordonat = 1;   rewind(f);
     for (i=1; i<=lungime_fisier-1; i++)  {
            poz=ftell(f);
            fread(&p, sizeof(struct persoana), 1, f);
            fread(&q, sizeof(struct persoana), 1, f);
            if (p.punctaj < q.puntaj)   {
                 fseek(f, poz, SEEK_SET);  ordonat = 0;
                 fwrite(&q, sizeof(q), 1, f);  fwrite(&p, sizeof(p), 1, f);}
            fseek(f, poz+lung_pers, SEEK_SET);  }
}  while (!ordonat);
fclose(f);     }
```

Aplicaţii
Prelucrarea fişierelor

```
void main(void)  {
    FILE *f;
    char nf[13];   char c;
    strcpy(nf, “”);
    do  {
       clrscr();
       printf(“\nFISIER CONCURS -> %s\n\n”, nf);
       printf(“\n[D]enumire fisier, [A]daugare, [S]orteaza”);
       printf(“\n[C]autare/modificare,[L]istare, [O]prire\n ”);
       c=tolower(getch()); printf(“\n”);
```

Aplicaţii
Prelucrarea fişierelor

```
    switch(c)  {
        case ‘d’: printf(“Dati numele fisierului: “); gets(nf);
                  break;
        case ‘a’: Adauga(nf); getch();  break;
        case ‘c’: CautaSiModifica(nf);  getch();  break;
        case ‘l’: Listeaza(nf);  getch();  break;
        case ‘s’: Sorteaza(nf);  Listeaza(nf);  getch();
    }
  } while (c!=‘o’);
}
```

Aplicaţii
Recursivitate

- **Problemă:** Să se scrie un program care caută un element într-un şir $x=(x_1, x_2, ..., x_n)$ ordonat crescător. Se va folosi metoda **căutării binare**. Obs: vom aplica metoda *divide et impera*.

```
#include <stdio.h>
#include <conio.h>
typedef int vector[20];
void Citeste(vector x, int *n) {
    int p, i;
    printf("dati nr. de elemente:"); scanf("%d", &p);
    *n=p;
    for ( i=0; i<p; i++ )  {
        printf("Dati x[ %d ]=", i); scanf("%d",&x[ i ]); } }
```

Aplicaţii
Recursivitate

```
void Afiseaza (vector x, int n)
{
    printf (“Elementele lui x sunt: “);
    for (int i=0; i<n; i++)
        printf (“%d,”, x[ i ]);
    printf (“\n\n”);
}
```

Se va aplica metoda *divide et impera* în varianta recursivă, prin funcția recursivă CautareBinara. Se începe căutarea cu elementul din mijlocul şirului, continându-se apoi căutarea, dacă este nevoie, cu elementele din jumătatea stângă sau jumătatea dreaptă a şirului.

Aplicaţii
Recursivitate

```
int CautareBinara(vector a, int elem, int inceput, int sfarsit) {
   if (inceput<sfarsit)  {
      int mijloc=(inceput+sfarsit)/2;
      if (elem==a[mijloc])
           return mijloc+1;   // atentie la indici !
      else  if (elem<a[mijloc])
                  return CautareBinara(a, elem, inceput, mijloc-1);
             else
                  return CautareBinara(a, elem, mijloc+1, sfarsit);
    }
    else
        return 0;  }
```

Aplicaţii
Recursivitate

```
void main ( ) {
   int n,p,e;  vector a;
   clrscr( );  Citeste(a, &n);  Afiseaza(a, n);
   printf(“Dati numarul cautat: “); scanf(“%d”, &e);
printf(“\n”);
   if (p=CautareBinara(a, e, 0, n-1))
      printf(“%d exista pe pozitia %d in sir !”, e, p);
   else
      printf(“%d nu exista in sir !”, e);
   getch( );
}
```