Programarea calculatoarelor
Limbajul C

CURS 11

Operatorii pe biti
Fişiere binare

Operatorii pe biți

- Se aplică fiecărui bit din reprezentarea *operanzilor întregi*
- Restul operatorilor se aplică valorilor operanzilor

- & şi
- | sau
- ^ XOR (sau exclusiv)
- ~ complement față de 1
- << deplasare la stânga
- >> deplasare la dreapta

Exemplu:

- Exemplu:

  7 0000000000000111

  8 0000000000001000

  7 & 8 0000000000000000

  7 & 8 = 0

  7 && 8 = 1 (adevărat şi adevărat → adevărat).

- Operatorul ~ transformă fiecare bit din reprezentarea operandului în complementarul său (biții 1 în 0 şi cei 0 în 1)

Operatorii de deplasare

- Primul operand este cel al cărui biți sunt deplasati
- Al doilea indică numărul de biți cu care se face deplasarea:

  a<<n, a>>n
- La *deplasarea la stânga* cu o poziție, bitul cel mai semnificativ se pierde, iar în dreapta se completează cu bitul 0.
- La *deplasarea la dreapta* cu o poziție, bitul cel mai puțin semnificativ se pierde, iar în stânga se completează cu un bit identic cu cel de semn.
- Cu excepția cazurilor când se produce depăşire, deplasarea la stânga cu n biți echivalează cu înmulțirea cu 2 la puterea n.
- Deplasarea la dreapta cu n biți echivalează cu împărțirea cu 2 la puterea n.

Observație

- Este indicat să se realizeze înmulțirile şi împărțirile cu puteri ale lui 2 prin deplasări (necesită un timp mult mai scurt):

int i;

- i * =8; echivalent cu i<<=3;
- i / =4; echivalent cu i>>=2;
- i * =10; echivalent cu i= i<<3 + i<<1;

Exemple

char x	x dupa fiecare executie	Valoarea lui x
x=7	00000111	7
x=x<<1	00001110	14
x=x<<3	01110000	112
x=x<<2	11000000	192
x=x>>1	01100000	96
x=x>>2	00011000	24

Exemplu

- Se consideră n, p întregi. Să se seteze pe 1 bitul p din reprezentarea lui n, (ceilalți biți rămân nemodificați).

```
#include<stdio.h>
{
  unsigned int n=5, p=1;
  n = n | (1<<p);
  printf("%u\n"n);
 }
```

Exemplu

- Să se scrie o secvență care construieşte o mască ce conține n biți de 1 începând cu poziția p, spre dreapta (bitul cel mai puțin semnificativ se consideră ca fiind de poziție 0)
- int masca=0, unu=1;
  int i, p=5, n=3;

```
unu << = p;  // bitul 1 este pe pozitia p
for ( i=1; i<=n; i++ ) {
        masca | = unu;
        unu >> = 1;
}
```

Exerciții

1. Se consideră n şi p întregi. Să se seteze pe 0 bitul p din reprezentarea lui n.
2. Să se afişeze un număr întreg fără semn în binar folosind extragerea selectivă de biți cu o mască ce conține un singur bit 1.

Rezolvare 1

```
#include<stdio.h>
void main()
{
  unsigned int n=7,p=1;
  n&= ~(1<<p);
  printf(“%u\n”n);
}
```

Rezolvare 2

```
#include<stdio.h>
    void main () {
      int n;
      printf("n= "); scanf("%d",&n);
      unsigned int mask = 1<<31; /* bitul 31 =1, ceilalti zero*/
      while (mask) {
            printf ("%d", (mask & n) ? 1: 0);
            mask >>= 1;      /* deplasare masca la dreapta cu un bit*/
      }
      printf("\n");
    }
```

Funcții de acces la fişiere binare

- Un fişier binar este format în general din articole de lungime fixă, fără separatori între articole.
- Un articol poate conține
  - un singur octet
  - un număr binar (pe 2,4 sau 8 octeți)
  - o structură cu date de diferite tipuri.
- Funcțiile de acces pentru fişiere binare "fread" şi "fwrite" pot citi sau scrie unul sau mai multe articole, la fiecare apelare.
- Transferul între memorie şi suportul extern se face fără conversie sau editare (adăugare de caractere la scriere sau eliminare de caractere la citire).

Funcții intrare/iesire (fişiere binare – b)

- **size_t fread (void *ptr, size_t size, size_t nmemb, FILE *fp)**
- citeşte la adresa **ptr** cel mult **nmemb** elemente de dimensiune **size** din fişierul referit de **fp**:

  int a[10];

  fread (a, sizeof(int), 10, fp);

- **size_t fwrite (void *ptr, size_t size, size_t nmemb, FILE *fp)**
- scrie în fişierul referit de **fp** cel mult **nmemb** elemente de dimensiune **size** de la adresa **ptr**:

  fwrite(a, sizeof(int),10,fp);

- Rezultatul funcțiilor "fread" şi "fwrite" este numărul de articole efectiv citite sau scrise
- Este diferit de argumentul 3 numai la sfârsit de fişier (la citire) sau în caz de eroare de citire/scriere.

Exemplu: operații cu un fişier de elevi (nume şi medie)

```
#include<stdio.h>
#include<stdlib.h>
#include<assert.h>
#include<string.h>
typedef struct {
  char nume[25];
  float medie;
} Elev;
     // creare fişier cu nume dat
void creare(char * numef) {
  FILE * f;  Elev s;
  f=fopen(numef,"wb");
  assert (f != NULL);
  printf ("Nume şi medie ptr. fiecare student :\n");
  while ( scanf ("%s%f", s.nume, &s.medie) !=  EOF)
     fwrite (&s, sizeof(s), 1, f);
  fclose (f);
}
```

Exemplu

```
     // afisare conținut fişier pe ecran
void listare (char* numef) {
 FILE * f;  Elev e;
 f = fopen (numef, "rb");  assert (f != NULL);
 printf ("Nume şi medie: \n");
 while ( fread (&e, sizeof(e), 1, f ) ==1 )
   printf ("%-25s %6.2f \n", e.nume, e.medie);
 fclose (f);
}
       // adaugare articole la sfârşitul unui fişier existent
void adaugare (char * numef) {
  FILE * f;  Elev e;
  f = fopen (numef, "ab");  assert (f != NULL);
  printf ("Adaugare nume şi medie:\n");
  while (scanf ("%s%f", e.nume, &e.medie) != EOF)
     fwrite (&e, sizeof(e), 1, f);
  fclose (f);
}
```

Acces direct la datele dintr-un fişier

- posibilitatea de a citi sau scrie oriunde într-un fişier, printr-o poziționare prealabilă înainte de citire sau scriere
- În C poziționarea se face pe un anumit octet din fişier, iar funcțiile standard permit accesul direct la o anumită adresă de octet din fişier.
- Funcțiile pentru acces direct din <stdio.h> permit operațiile următoare:
  - Pozitionarea pe un anumit octet din fişier ("fseek").
  - Citirea poziției curente din fişier ("ftell").
  - Memorarea poziției curente şi poziționare ("fgetpos", "fsetpos").
- Poziția curentă în fişier este un număr de tip *long*, pentru a permite operații cu fişiere foarte lungi!

Funcții pentru acces direct

- **long int ftell (FILE *fp)**
  - Întoarce valoarea indicatorului de poziție
  - Pentru fişier binar: numărul de octeți de la începutul fişierului
  - Fişier text: o valoare ce poate fi utilizată de fseek pentru a seta indicatorul de poziție în fişier la această poziție.
- **int fseek (FILE *fp, long int offset, int poziție)**

  poziționează indicatorul de poziție la valoarea dată de offset față de poziție:
  - SEEK_SET = 0 - față de începutul fişierului
  - SEEK_CUR = 1 - față de poziția curentă
  - SEEK_END = 2 - față de sfârşitul fişierului
- Intr-un fişier text poziționarea este posibilă numai fată de începutul fişierului, iar poziția se obține printr-un apel al funcției "ftell"!

Exemple

- poziționarea la sfârşitul fişierului: **fseek (fp, 0, SEEK_END)**
- poziționarea la caracterul precedent: **fseek (fp, -1, SEEK_CUR)**
- poziționarea la inceputul fişierului: **fseek (fp, 0, SEEK_SET)**

- Atenție! Poziționarea relativă la sfârşitul unui fişier nu este garantată nici chiar pentru fişiere binare, astfel că ar trebui evitată!
- Ar trebui evitată şi poziționarea față de poziția curentă cu o valoare negativă, care nu funcționează în toate implementările!

Funcții pentru acces direct

- **int fgetpos (FILE *fp, fpos_t *poziție)**

  memorează starea curentă a indicatorului de poziție al fluxului referit de **fp** în **poziție;**

  întoarce 0 dacă operația s-a realizat cu succes!
- **int fsetpos (FILE *fp, const fpos_t *poziție)**

  setează indicatorul de poziție al fluxului referit de **fp** la valoarea data de **poziție**
- **void rewind (FILE *fp)**

  setează indicatorul de poziție al fluxului referit de **fp** la începutul fişierului

Funcție care modifică conținutul mai multor articole din fişierul de elevi creat anterior

```
      // modificare conținut articole, dupa cautarea lor
void modificare (char * numef) {
  FILE * f; Elev e;  char nume[25];
  long pos;  int  ef;
  f = fopen(numef,"rb+");  assert (f != NULL);
  do {
     printf ("Nume cautat: "); scanf ("%s",nume);
        if (strcmp(nume, ".") == 0)  break;     // datele se termină cu un punct
              // cauta "nume" în fişier
        fseek (f, 0, 0);                              // readucere pe inceput de fişier
        while ( (ef=fread (&e, sizeof(e), 1, f)) ==1 )
                 if (strcmp (e.nume, nume)==0) {
                          pos= ftell(f) - sizeof(e);
                          break;
                 }
```

Exemplu

```
      if ( ef < 1) break;
      printf ("noua medie: "); scanf ("%f", &e.medie);
      fseek (f, pos, 0);                  // pe inceput de articol gasit
       fwrite (&e, sizeof(e), 1, f); // rescrie articol modificat
  } while (1);
  fclose (f);
}

int main(){
    char name[]="c:elev.txt“;
    creare (name);
    listare (name);
    adaugare (name);
    modificare (name);
    listare (name);
    system("pause");
    return 0;
}
```

Fişiere predefinite

- Există trei *fluxuri predefinite*, care se deschid automat la lansarea unui program:
  - stdin - fişier de intrare, text, este intrarea standard - tastatura
  - stdout - fişier de ieşire, text, este ieşirea standard - ecranul monitorului.
  - stderr - fişier de iesire, text, este ieşirea standard de erori - ecranul monitorului.
- pot fi folosite în diferite funcții
- practic se folosesc în funcția "fflush" care goleşte zona tampon ("buffer") asociată unui fişier.

Observații

- Nu orice apel al unei funcții de citire sau de scriere are ca efect imediat un transfer de date între exterior şi variabilele din program!
- Citirea efectivă de pe suportul extern se face într-o zonă tampon asociată fişierului, iar numărul de octeți care se citesc depind de suport: o linie de la tastatură, unul sau câteva sectoare disc dintr-un fişier disc, etc.
- Cele mai multe apeluri de funcții de I/E au ca efect un transfer între zona tampon (anonimă) şi variabilele din program.
- Funcția “fflush” are rolul de a goli zona tampon folosită de funcțiile de I/E, zonă altfel inaccesibilă programatorului C.
- Are ca argument variabila pointer asociată unui fişier la deschidere, sau variabilele predefinite “stdin” şi “stdout”.

Redirectarea fişierelor standard

- Prin redirectare, fişierele standard se pot asocia cu alte fişiere.
- *Exemplu:*

  fişier_exe <fişier_1 >fişier_2
- În acest caz, preluarea informațiilor se face din fişier_1, iar afişarea informațiilor de ieşire se face în fişier_2.

Exemplu: copierea conținutului unui fişier în alt fişier utilizand redirectarea

- Folosind redirectarea fişierelor standard, se va lansa printr-o linie de comandă de forma:
  copiere1 <fişier_sursa.dat >fişier_dest.dat

Şi atunci următorul program va avea acelaşi rezultat ca si când s-ar citi din fisier_sursa.dat şi s-ar scrie în fişier_dest.dat:

```
# include <stdio.h>
int main(void){
        char c;
        while ( (c=getchar()) != EOF )
                putchar(c);
        return 0;
}
```