Tema: Deplasări pe biți (afișarea variabilelor în diferite sisteme de numerotație)

Obiective:

- să cunoască operatorii binari;
- să aplice operatorii logici pe cuvânt;
- să aplice operatori logici pe biţi;
- să propună expresii logice pe biţi.

4.1. Sarcina pentru soluționare

Să se elaboreze două programe, unul dintre care să introducă părțile componente ale structurii datelor prezentate în Tabelul 4.1. şi să formeze arhivarea acestora, iar al doilea să afișeze structura arhivată ca număr hexazecimal şi valorile părților separate din componența ei.

Tabelul 4.1. Informaţia referitoare la erorile critice pe disc are următoarea formă:

Nr. bitului	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
Valoarea	O	A	A	R	R	R	0	1	D	D	D	D	D	D	D	D

unde:
O – tipul operaţiei – citire/scriere (0/1);
AA – codul sectorului de disc;
RRR – acţiunile posibile la eroare;
D..D – numărul discului.

4.2. Exemplu de soluţionare a sarcinii

4.2.1. Programul arhivării

Algoritmul programului include următorii paşi:

1) Pentru fiecare componentă se afișează invitația de introducere şi se solicită valorile.

2) Transferarea *tipul operației* (*o*) în cuvântul de stare a echipamentului necesită evidențierea unui bit inferior. Cu operatorul & pe biți se verifică valoarea setată, apoi valoarea obținuta se deplasează cu 15 biți în stânga.

3) Amplasarea *codului sectorului de disc* (*a*) în cuvântul de stare a echipamentului necesită evidențierea a 2 biți inferiori (constantă binară – 0011, zecimală – 3). Cu operatorul & (și) pe biți se verifică mărimea variabilei. Valoarea obținută se deplasează cu 13 biți în stânga. Pentru a adăuga codul obținut la cuvântul de stare, urmează de îndeplinit aplicat operatorul I (sau) pe biți între valoarea cuvântului de stare a echipamentului precedent şi a valorii variabilei *a*.

4) Pentru a introduce variabila *acțiunile posibile la eroare* (*r*) în cuvântul de stare a echipamentului, se distribuie 3 biți inferiori (constantă binară – 0111, zecimală – 7), respectiv operatorul & (și) se aplică asupra variabilei și constantei 7. Valoarea obținută se deplasează cu 10 biți în stânga şi se adăugă la acel cod, care a fost obținut la pasul precedent.

5) În bitul 9 şi 8 este înscrisă constanta binară 01, care se adaugă în cuvântul de stare a echipamentului utilizând operatorul I (sau) pe biți.

6) Introducerea variabilei *numărul discului* (*d*) în cuvântul de stare a echipamentului necesită distribuirea a 8 biți inferiori (constantă binară – 1111 1111, zecimală – 255, hexazecimală FF). Cu operatorul & (și) pe biți se verifică mărimea variabilei. Valoarea primită se adăugă la codul obținut la pasul precedent.

Menționăm, dacă valorile părților componente ale codului introdus de operator nu depășesc intervalul valorilor admisibile, atunci operația de verificare este în plus. Dar prevedem această operație pentru cazul unei greșeli introduse de operator.

Determinarea variabilelor programului

Algoritmul de lucru prevede patru variabile în care se vor păstra părțile componente ale cuvântului de stare. Pentru păstrarea tuturor părților componente va fi de ajuns tipul de date *int*. Domeniul valorilor admisibile pentru aceste variabile:

o – tipul operaţiei – 0, 1;

a – codul sectorului de disc – 0..3;

r – acţiunile posibile la eroare – 0..7;

d – numărul discului – 0..255.

Mai avem nevoie de o variabilă pentru păstrarea cuvântului de stare a echipamentului, care este o variabilă întreagă.

Elaborarea textului programului

Textul programului începe cu includerea în program a fișierului antet:

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

În continuare urmează antetul funcţiei *main()*:

```
int main() {
```

şi declararea variabilelor:

```
int o, a, r, d, UnitStateWord;
```

Introducerea variabilei *o* se subînţelege din afişarea invitaţiei la ecran:

```
printf("Introduceți tipul operației (0 / 1) >");
```

şi atribuirea valorii respective variabilei *o*:

```
scanf("%d",&o);
```

Astfel de perechi de operatori se repetă pentru introducerea valorilor pentru variabilele *a*, *r*, *d*. Toate valorile introduse sunt citite ca numere întregi, în care scop se utilizează specificatorul de tip *%d*.

În continuare urmează formarea codului arhivat:

```
UnitStateWord = (o&1)<<15;
```

Valoarea variabilei *o* este deplasată cu 15 biți în stânga. Corespunzător sunt arhivate și celelalte variabile.

```
UnitStateWord |= (a&3)<<13;
UnitStateWord |= (r&7)<<10;
UnitStateWord |= 1<<8;
UnitStateWord |= d&0xFF;
```

Afișarea codului arhivat:

```
printf("\n Cuvântul de stare a echipamentului = %04x\n",UnitStateWord);
```

Rezultatul este extras ca un număr hexazecimal din patru cifre.

Textul integral al programului este prezentat în Figura 4.1.

[*] Lab4.c

```c
#include <stdio.h>
int main(){
int a, o, r, d, UnitStateWord;

//codul de arhivare
printf(" \n Setati tipul operatiei (0 / 1) >> ");
scanf("%d", &o);
UnitStateWord = (o&1)<<15;

printf(" \n Setati codul sectorului de disc (0 - 3) >> ");
scanf("%d", &a);
UnitStateWord |= (a&3)<<13;

printf(" \n Setati actiunile posibile in caz de eroare (0 - 7) >> ");
scanf("%d", &r);
UnitStateWord |= (r&7)<<10;

UnitStateWord |= 1<<8;

printf(" \n Setati numarul discului (0 - 255) >> ");
scanf("%d", &d);
UnitStateWord |= d&0xFF;

//afisarea rezultatelor
printf("\n Cuvantul de stare a echipamentului = %04x\n", UnitStateWord);
return 0;
}

```

Figura 4.1. Textul programului de arhivare

Ajustarea programului

Ajustarea programului poate fi efectuată urmărind valorile variabilelor părților componente la intrare şi cuvântul de stare. Pentru ultimul pot apărea incomodități, se recomandă ca valoarea variabilei *UnitStateWord* să fie un număr zecimal. Astfel, pentru urmărire poate fi utilizată funcţia *printf* în locurile corespunzătoare ale textului programului, care va afișa valoarea intermediară a cuvântului de stare în zecimal.

Rezultatul derulării programului

În urma lansării programului la ecran vor fi afişate rezultatele:

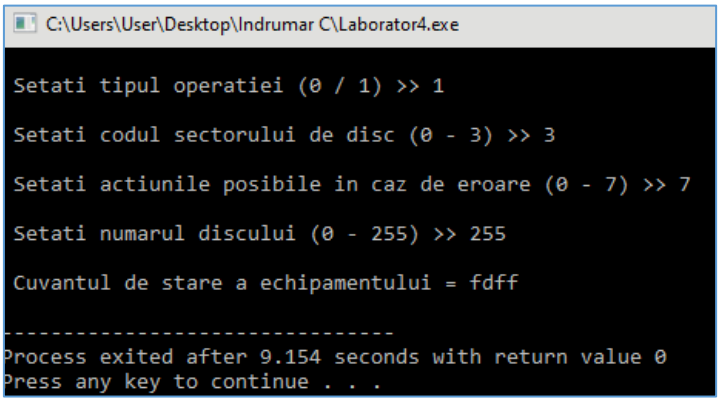

Figura 4.2. Rezultatele furnizate de programul arhivării

4.2.2. Programul de dezarhivare

Elaborarea algoritmului de soluţionare

Algoritmul programului este format din următorii paşi:

1) Introducerea cuvântului de stare a echipamentului.

2) Extragerea valorii variabilei *o* din cuvântul de stare. Codul cuvântului de stare a echipamentului trebuie deplasat cu 15 biți în dreapta şi evidenţiat un bit inferior cu operatorul & (și). Cuvântul de stare a echipamentului rămâne neschimbat, pe când valoarea obținută se înscrie în variabila – tipul operației.

3) Variabila *a* din cuvântul de stare a echipamentului se obține în modul următor: codul cuvântului de stare a echipamentului se deplasează cu 13 biți în dreapta şi se evidențiază 2 biți inferiori cu operatorul & (și). Cuvântul de stare a echipamentului rămâne neschimbat, însă valoarea obținută se înscrie în variabila – codul sectorului de disc.

4) Pentru extragerea variabilei *r* din cuvântul de stare a echipamentului, se deplasează codul cuvântului de stare a echipamentului cu 10 biți în dreapta şi se evidențiază 3 biţi inferiori. Valoarea obținută se înscrie în variabila – acţiuni posibile la eroare.

5) Evidențierea variabilei *d* din cuvântul de stare a echipamentului, se realizează evidențiind 8 biți. Valoarea obținută se înscrie în variabila – numărul discului.

Determinarea variabilelor programului

Pentru lucrul programului, vom avea nevoie de aceleași variabile descrise în pct. 4.2.1.2.

Elaborarea textului programului

Începutul programului – antetul şi determinarea variabilelor sunt descrise în pct. 4.2.2.1. În continuare urmează formarea codului de dezarhivare, care întocmai repetă pașii algoritmului descriși mai sus:

```
o=(UnitStateWord>>15)&1;
a=(UnitStateWord>>13)&3;
r=(UnitStateWord>>10)&7;
d=UnitStateWord&0xFF;
```

Extragerea primului rezultat – tipul operaţiei se îndeplineşte cu operatorul:

```
printf("Tipul operației = %d\n",o);
```

În mod analog se execută extragerea celorlalte rezultate.

Textul integral al programului este prezentat în Figura 4.3.

Lab4_dezarhivare.c

```c
#include <stdio.h>
int main(){
int a, o, r, d, UnitStateWord;

//setarea codului de stare a echipamentului

printf(" \n Setati cuvantul de stare a echipamentului >> ");
printf(" \n un numar hexazecimal de la 0x0 pana la 0xFDFF >> ");
scanf("%x", &UnitStateWord);

//Extragerea partilor componente
o=(UnitStateWord>>15)&1;
a=(UnitStateWord>>13)&3;
r=(UnitStateWord>>10)&7;
d=(UnitStateWord&0xFF);

//Afisarea partilor componente
putchar('\n');
printf(" tipul operatiei = %d\n",o);
printf(" codul sectorului de disc = %d\n",a);
printf(" actiuni posibile in caz de eroare = %d\n",r);
printf(" numarul discului = %d\n",d);

return 0;
}
```

Figura 4.3. Textul programului de dezarhivare

Ajustarea programului

Ajustarea programului se efectuează după metoda descrisă în pct. 4.2.1.4.

Rezultatele derulării programului

După lansarea programului, la ecran vor fi afișate rezultatele:

```
C:\Users\User\Desktop\Indrumar C\laborator4\Lab4_dezarhivare.exe

Setati cuvantul de stare a echipamentului >>
un numar hexazecimal de la 0x0 pana la 0xFDFF >> fdff

tipul operatiei = 1
codul sectorului de disc = 3
actiuni posibile in caz de eroare = 7
numarul discului = 255

--------------------------------
Process exited after 3.338 seconds with return value 0
Press any key to continue . . .
```

Figura 4.4. Rezultatele furnizate de programul dezarhivării

4.3. Subiecte de evaluare

Exerciții:

1. Să se implementeze programele cu exemplele prezentate
2. Identificați valoarea returnată următoarea expresie: `int i=5, j=2; i<j+2`.
3. Fie `int x=2, y=1, z=0; x=x&&y||z;` Stabiliți valoarea variabilei x.
4. Stabiliți valorile afișate de următorul cod de program:

```
#define tipar(a) printf("a = %d\n",a)
main() {
 int x,y,z;
 x = 7; y = 5; z = 3;
 tipar( x | y & z );        //1
 tipar( x | y & ~ z );      //2
 tipar( x ^ y & ~ z );      //3
 tipar( x & y && z );       //4

 x = 1; y = -1;
 tipar( ! x | x );          //5
 tipar( ~ x | x );          //6
 tipar( x ^ x );            //7
 x <<= 3; tipar(x);  //8
 y <<= 3; tipar(y);  //9
 y >>= 3; tipar(y);  //10
}
```

Verificarea cunoștințelor:

1. Enumeraţi operatorii logici în ordinea priorităţii lor.
2. Prezentaţi tabelele de adevăr pentru operatorii logici.
3. Enumeraţi operatorii pe biţi în ordinea priorităţii lor.
4. Prezentaţi tabelele de adevăr pentru operatorii pe biţi.

4.4. Bibliografie

1. Kris Jamsa & Lars Klander, *Totul despre C şi C++*, Editura Teora, Bucureşti, 2006.
2. Herbert Schildt, *C++ manual complet*, Editura Teora, Bucureşti, 1999. Disponibil: https://drive.google.com/file/d/1BrQtITgykcWk03xxtl3Q0-nvcRRFAsy7/view?usp=sharing
3. Tutorials C++ Language. Operators. Disponibil: https://cplusplus.com/doc/tutorial-ro/operators/