

## Reprezentare binara, hexazecimala

In sistemele de calcul informatiile sunt codificate binar. Aceste informatii pot fi reprezentate binar, zecimal, hexazecimal sau sub diferite alte formate.

Vom analiza in continuare diferite aplicatii pentru conversia datelor si afisarea lor sub diferite formate.

- **Afisarea unui int sub forma binara**

Prin impartiri repetate cu 2 obtinem digitii ce reprezinta valoarea in binar.

```
// Programul afiseaza valoarea binara a unui intreg
#include "stdafx.h"
#include < iostream >
using namespace std;

int main(void)
{
    system("TITLE Afisare int sub forma binar");
    system("COLOR F9");
    int n,i; // n:numarul de convertit
    cout << "\n\tProgramul afiseaza valoarea binara a unui intreg " ;
    cout << "\n\n\tIntroduceti un numar intreg: " ;
    cin >> n;
    cout << " \n\tValoarea binara afisata in ordine inversa este:\n\n\t";
    if (n > 0) {
        for(i=0; i < 32 ; i++) {
            cout << n%2;
            n = n/2;
        }
    }
    else {
        cout << "\n\n\tIntroduceti un numar pozitiv\n" << endl;
    }
    cin.ignore();
    cin.get();
    return 0;
}
```

Pentru a afisa bitii in ordine directa, va trebui sa memoram acesti biti pentru a fi afisati la sfarsit.

```
// Programul afiseaza valoarea binara a unui intreg
```

```

#include "stdafx.h"
#include < iostream >
using namespace std;

int main(void)
{
    system("TITLE Afisare int sub forma binar");
    system("COLOR F9");
    int n,i;  // number to convert to binary
    char val_b[32];

    cout << "\n\tProgramul afiseaza valoarea binara a unui intreg " ;
    cout << "\n\n\tIntroduceti un numar intreg: " ;
    cin >> n ;
    cin.ignore();

    for (i=0; i < 31; i++){
        if (n%2==0)
            val_b[30-i]='0';
        else
            val_b[30-i]='1';
        n=n/2;
    }

    cout << "\n\t Valoarea binara este:";
    for (i=0; i < 31; i++)
        cout << val_b[i] ;
    cin.get();
    return 0;
}

```

- **Afisarea unui int sub forma hexa**

Pentru a afisa sub forma hexa zecimala, vom utiliza operatorul << **hex** care inclus in cadrul instructiunii **cout** << va forta afisarea valorilor in format hexa.

```

// Programul afiseaza valoarea hexa a unui intreg
#include "stdafx.h"
#include < iostream >
using namespace std;

int main(void)
{
    system("TITLE Afisare int sub forma hexa");
    system("COLOR F9");
    int n;
    cout << "\n\tProgramul afiseaza valoarea hexa a unui intreg " ;
    cout << "\n\n\tIntroduceti un numar intreg: " ;
    cin >> n;
    cout << " \n\tValoarea hexa este:" << hex << n;
}

```

```

        cin.ignore();
        cin.get();
        return 0;
    }

```

Dupa utilizarea operatorului << **hex** trebuie utilizat operatorul << **dec** pentru a reveni la afisarea sub forma zecimala.

```

// Programul afiseaza valoarea hexa a unui intreg
#include "stdafx.h"
#include < iostream >
using namespace std;

int main(void)
{
    system("TITLE Afisare int sub forma hexa");
    system("COLOR F9");
    int n;
    cout << "\n\tProgramul afiseaza valoarea hexa a unui intreg " ;
    cout << "\n\n\tIntroduceti un numar intreg: " ;
    cin >> n;
    cout << " \n\tValoarea hexa este: "<< hex << n;
    cout << " \n\tValoarea lui n este: "<< n;
    cout << " \n\tValoarea zecimala este:"<< dec << n;
    cin.ignore();
    cin.get();
    return 0;
}

```

Afisarea in format zecimal, binar, hexa sau in orice alta baza se poate face si prin utilizarea functiei **itoa**

```

// Conversia unui intreg intr-un sir reprezentand valoarea in diferite baze
de numeratie
#include "stdafx.h"
#include < iostream >
using namespace std;

int main(void)
{
    system("TITLE Afisarea in diferite baze de numeratie");
    system("COLOR F9");
    int n;
    char buffer [33];
    cout << "\n\n\tIntroduceti un numar intreg: " ;
    cin >> n;
}

```

```

        itoa(n,buffer,10);//functia pentru conversia unui intreg intr-un sir
        cout << "\n\n\tValoarea zecimala a numarului este: " << buffer;
        itoa(n,buffer,2);
        cout << "\n\n\tValoarea binara a numarului este: " << buffer;
        itoa(n,buffer,16);
        cout << "\n\n\tValoarea hexa a numarului este: " << buffer;
        cin.ignore();
        cin.get();
    return 0;
}

```

## Operatori binari

Deseori este nevoie sa folosim valori binare sau e nevoie sa convertim in format binar diferite date de diverse tipuri. C++ permite diverse operatii binare, operatii ce le vom folosi in continuare.

- **AND logic -- operatorul &**

Operatia AND este deseori folosita pentru a realizeaza o masca pentru extragerea anumitor biti dintr-un operand. Operatorul folosit este &

```

// Programul realizeaza operatia AND
// Se calculeaza 0xffff AND 0x5555 adica 1111111111111111 &
0101010101010101
// Rezultatul este 0x5555 adica 0101010101010101
// Operatia realizeaza o masca pentru extragerea anumitor biti dintr-un
operand
#include "stdafx.h"
#include < iostream >
#include < string >
using namespace std;

int main(void)
{
    system("TITLE AND logic ");
    system("COLOR F9");
    unsigned short a = 0xFFFF; // = 1111111111111111
    unsigned short b = 0x5555; // = 0101010101010101
    cout << "\n\n\tProgramul calculeaza 0xffff AND 0x5555";
    cout << "\n\n\t0xffff & 0x5555=" << hex << ( a & b ); // rezultat
"5555" adica 0101010101010101
    cin.ignore();
    cin.get();
}

```

Operatia AND este deseori folosita pentru a realizeaza o masca pentru extragerea anumitor biti

dintr-un operand. Operatorul folosit este **&**

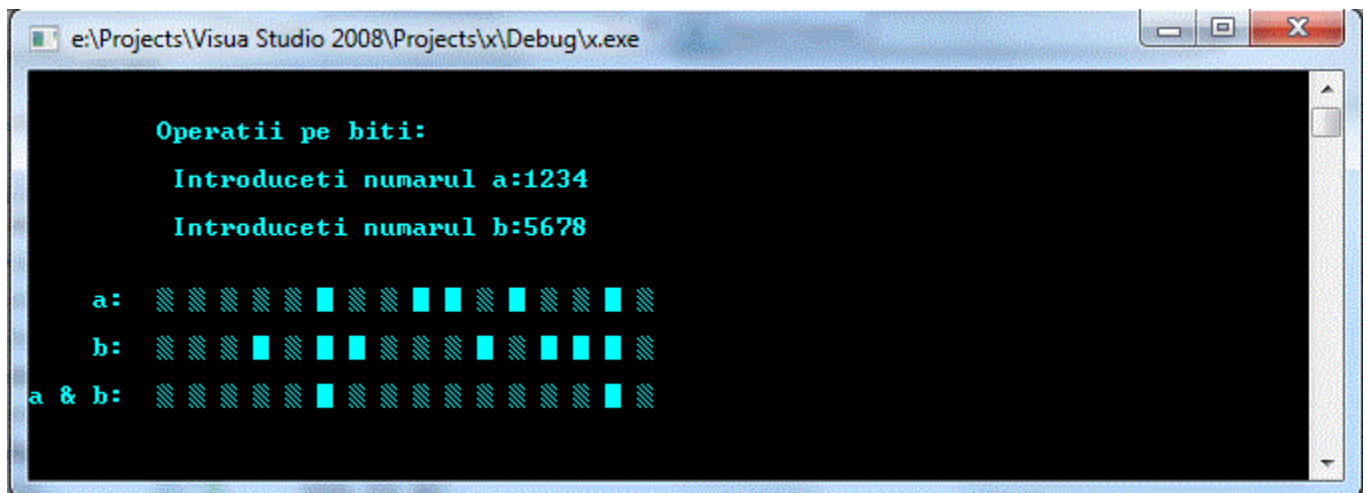
Pentru exemplificare, vom realiza in continuare o aplicatie care realizeaza functia logica **&** intre doi operanzi si sub forma binara atat operanzii cat si rezultatul. Aplicatia se bazeaza pe functiile **afis\_binar()** respectiv **afis\_bin()** functii care se presupune ca sunt definite in fisierul **stadfx.h**

Programul principal fiind:

```
#include "stdafx.h"
#include < iostream >
using namespace std;

int main(void)
{
    unsigned int a,b;
    cout << " \n\n\tOperatii pe biti: ";
    cout <<"\n\n\t Introduceti numarul a:";
    cin >> a;
    cout <<"\n\t Introduceti numarul b:";
    cin >> b;
    cout<<"\n\n    a:\t";
    afis_bin(a,16);
    cout<<"\n\n    b:\t";
    afis_bin(b,16);
    cout<<"\n\na & b:\t";
    afis_bin(a&b,16);
    cin.ignore();
    cin.get();
    return 0;
}
```

Dupa rularea programului in "Command Prompt" se afiseaza:



```
e:\Projects\Visua Studio 2008\Projects\x\Debug\x.exe

Operatii pe biti:
Introduceti numarul a:1234
Introduceti numarul b:5678

a:  0000000000000000
b:  0000000000000000
a & b: 0000000000000000
```

Daca se doreste introducerea operanzilor in format hexa, programul devine:

```
#include "stdafx.h"
#include < iostream >
using namespace std;

int main(void)
{
    unsigned int a,b;
    cout << " \n\n\tOperatii pe biti: ";
    cout <<"\n\n\t Introduceti numarul a:";
    cin >> hex >> a;
    cout <<"\n\t Introduceti numarul b:";
    cin >> hex >> b;
    cout<<"\n\n    a:\t";
    afis_bin(a,16);
    cout<<"\n\n    b:\t";
    afis_bin(b,16);
    cout<<"\n\na & b:\t";
    afis_bin(a&b,16);
    cin.ignore();
    cin.get();
    return 0;
}
```

## • SAU logic -- operatorul |

Operatia SAU este deseori folosita pentru a realizeaza o setare a anumitor biti dintr-un operand. Operatorul folosit este |

```
// Programul realizeaza operatia OR
// Se calculeaza 0xaaaa OR 0x5555  adica  1010101010101010 &
0101010101010101
// Rezultatul este 0xffff adica 1111111111111111
// Operatia realizeaza o setare anumitor biti dintr-un operand
#include "stdafx.h"
#include < iostream >
#include < string >
using namespace std;

int main(void)
{
    system("TITLE SAU logic ");
    system("COLOR F9");
    unsigned short a = 0xaaaa;          // = 1010101010101010
```

```

        unsigned short b = 0x5555;          // = 0101010101010101
        cout << "\n\n\tProgramul calculeaza 0xaaaa AND 0x5555";
        cout << "\n\n\t0xaaaa & 0x5555=" << hex << ( a | b ); // rezultat
"ffff" adica 1111111111111111
        cin.ignore();
        cin.get();
    }

```

- **SAU exclusiv - operatorul ^**

Operatia SAU exclusiv este deseori folosita pentru complementarea anumitor biti dintr-un operand. Operatorul folosit este ^

```

// Programul realizeaza operatia SAU EXCLUSIV
// Se calculeaza 0x5555 SAU EXCLUSIV 0xffff adica 0101010101010101 ^
1111111111111111
// Rezultatul este 0xaaaa adica 1010101010101010
// Operatia complementeaza bitii primului operand
#include "stdafx.h"
#include < iostream >
#include < string >
using namespace std;

int main(void)
{
    system("TITLE SAU EXCLUSIV "); // Titlul ferestrei consola
    system("COLOR F9");           // Fundal alb caractere albastre
    unsigned short a = 0x5555;    // = 0101010101010101
    unsigned short b = 0xFFFF;    // = 1111111111111111
    cout << "\n\n\tProgramul calculeaza 0x5555 SAU EXCLUSIV 0xffff";
    cout << "\n\n\t0x5555 ^ 0xffff=" << hex << ( a ^ b ); // rezultat
"aaaa" adica 1010101010101010
    cin.ignore();
    cin.get();
}

```

- **NOT - operatorul ~**

Operatia NOT realizeaza complementarea bitilor dintr-un operand. Operatorul folosit este ~

```

// Programul realizeaza operatia NOT
// Se calculeaza NOT 0xaaaa adica NOT 1010101010101010
// Rezultatul este 0x5555 adica 0101010101010101
// Operatia realizeaza complementarea bitilor dintr-un operand
#include "stdafx.h"

```

```

#include < iostream >
#include < string >
using namespace std;

int main(void)
{
    system("TITLE NOT logic ");
    system("COLOR F9");
    unsigned short a = 0xaaaa;          // = 1010101010101010
    cout << "\n\n\tProgramul calculeaza NOT 0xaaaa ";
    cout << "\n\n\tNOT 0xaaaa= " << hex << ~a ;    // rezultat "5555"
adica 0101010101010101
    cin.ignore();
    cin.get();
}

```

## • Delasare dreapta - operatorul >>

**n >> p;**

Deplaseaza spre dreapta cu p pozitii a bitilor ce compun in binar valoarea n. Pe pozitia cea mai semnificativa se pune 0. O deplasare spre dreapta cu 1 pozitie este echivalenta cu o impartire cu 2 cu 2. Astfel  $24 \ll 3 = 3$

Sa realizam o aplicatie care deplaseaza dreapta cu doua pozitii valoarea 64. Dupa rularea programului ar trebui sa obtinem valoarea 16.

```

// Deplasare dreapta
#include "stdafx.h"
#include < iostream >
using namespace std;

int main(void)
{
    system("TITLE Deplasare dreapta");
    system("COLOR F9");
    cout << "\n\n\tDeplasare dreapta";
    unsigned short int n=64;
    int i;
    cout << "\n\n\tValoarea initiala a lui n: "<< n;
        n = n >> 2 ;
    cout << "\n\n\tValoarea lui n dupa deplasarea cu doua pozitii
dreapta: "<< n;
    cin.ignore();
    cin.get();

    return 0;
}

```

Afisarea sub forma binara este mult mai simpla daca se utilizeaza operatorul de siftare dreapta



```

// Programul afiseaza valoarea binara a unui intreg
#include "stdafx.h"
#include < iostream >
using namespace std;

int main(void)
{
    system("TITLE Afisare int sub forma binar");
    system("COLOR F9");
    int n;
    cout << "\n\tProgramul afiseaza valoarea binara a unui intreg " ;

    cout << "\n\n\tIntroduceti un numar intreg: " ;
    cin >> n ;
    cin.ignore();
    // print binary with leading zeros
    cout << "\n\tValoarea binara este: : ";
    for (int i=31; i>=0; i--) {
        int bit = ((n>>i) & 1);
        cout << bit;
    }
    cin.get();
    return 0;
}

```

- **Delasare stanga - operatorul <<**

**n << p ;**

Deplaseaza spre stanga cu p pozitii a bitilor ce compun in binar valoarea n. Pe pozitia cea mai nesemnificativa se pune 0. O deplasare spre stanga cu 1 pozitie este echivalenta cu o inmultire cu 2. Astfel  $3 \ll 2 = 12$

Sa realizam o aplicatie care deplaseaza stanga cu doua pozitii valoarea 7. Dupa rularea programului ar trebui sa obtinem valoarea 28.

```

// Deplasare stanga
#include "stdafx.h"
#include < iostream >
using namespace std;

int main(void)
{
    system("TITLE Deplasare stanga");
    system("COLOR F9");
    cout << "\n\n\tDeplasare stanga";
    unsigned short int n=7;
    int i;
    cout << "\n\n\tValoarea initiala a lui n: "<< n;
}

```

```

        n = n << 2 ;
        cout << "\n\n\tValoarea lui n dupa deplasarea cu doua pozitii stanga:
"<< n;
        cin.ignore();
        cin.get();

        return 0;
}

```

Sa utilizam operatorul de siftare stanga pentru a deplasa stanga valoarea 1 de 7 ori si sa afisam in binar valoarea obtinuta dupa fiecare deplasare.

```

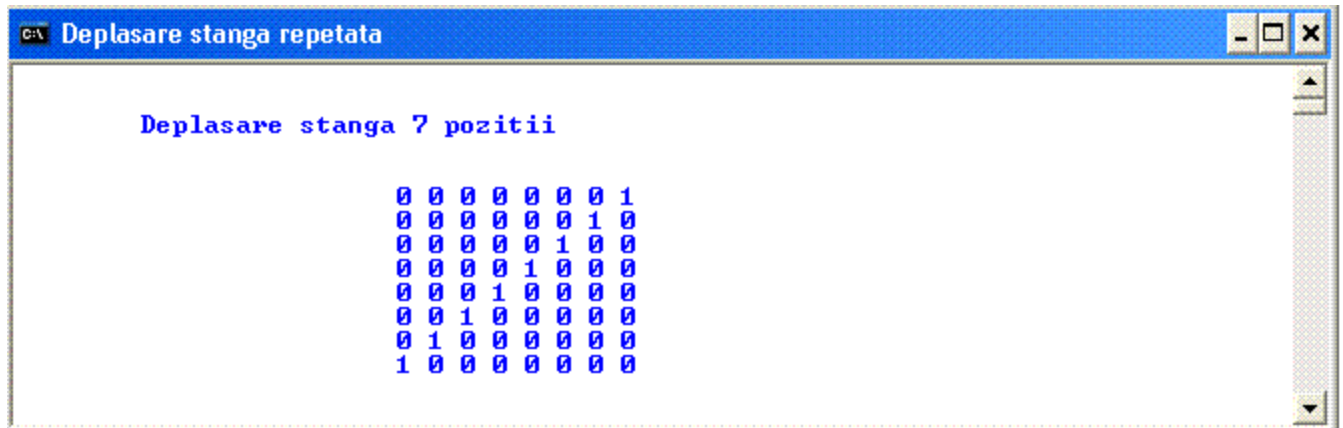
// Deplasare stanga repetata
#include "stdafx.h"
#include < iostream >
using namespace std;
void af_binar(unsigned int );

int main(void)
{
    system("TITLE Deplasare stanga repetata");
    system("COLOR F9");
    cout << "\n\n\tDeplasare stanga 7 pozitii \n\n\n\t\t\t";
    unsigned short int n=1;
    int i;
    for(i=0; i < 8; i++) {
        af_binar(n);
        n = n << 1 ;
    }
    cin.ignore();
    cin.get();

    return 0;
}
// Afisarea bitilor ce corespund valorii parametrului u
void af_binar(unsigned int u)
{
    int j;
    for (int j=7; j>=0; j--) {
        int bit = ((u >> j) & 1);
        cout << bit<<" ";
    }
    cout << "\n\t\t\t";
}

```

Rulam programul si obtinem:



Utilizand operatiile de siftare precum si operatiile logice, vom realiza in continuare afisarea unui int sub forma hexa fara a folosi operatorul << **hex**

Pentru a afisa sub forma hexa zecimala, vom face siftari dreapta cu cate 4 biti si vom interpreta ultimii patru biti utilizand masca 0xF adica 00001111.

```
// Programul afiseaza valoarea hexa a unui intreg
#include "stdafx.h"
#include < iostream >
using namespace std;

int main(void)
{
    system("TITLE Afisare int sub forma hexa");
    system("COLOR F9");
    int n;
    char* c_hex="0123456789ABCDEF";
    cout << "\n\tProgramul afiseaza valoarea hexa a unui intreg " ;
    cout << "\n\n\tIntroduceti un numar intreg: " ;
    cin >> n;
    cout << " \n\tValoarea hexa este:\n\n\t";
    for (int i=2*sizeof(int) - 1; i>=0; i--) {
        cout << c_hex[((n >> i*4) & 0xF)];
    }
    cin.ignore();
    cin.get();
    return 0;
}
```

- **Coduri ASCII**

Functia **char(i)** converteste o valoare intreaga intre 0 si 255 intr-un caracter ASCII.

Urmatoarea aplicatie foloseste functia **char(i)** si afiseaza setul extins de caractere ASCII

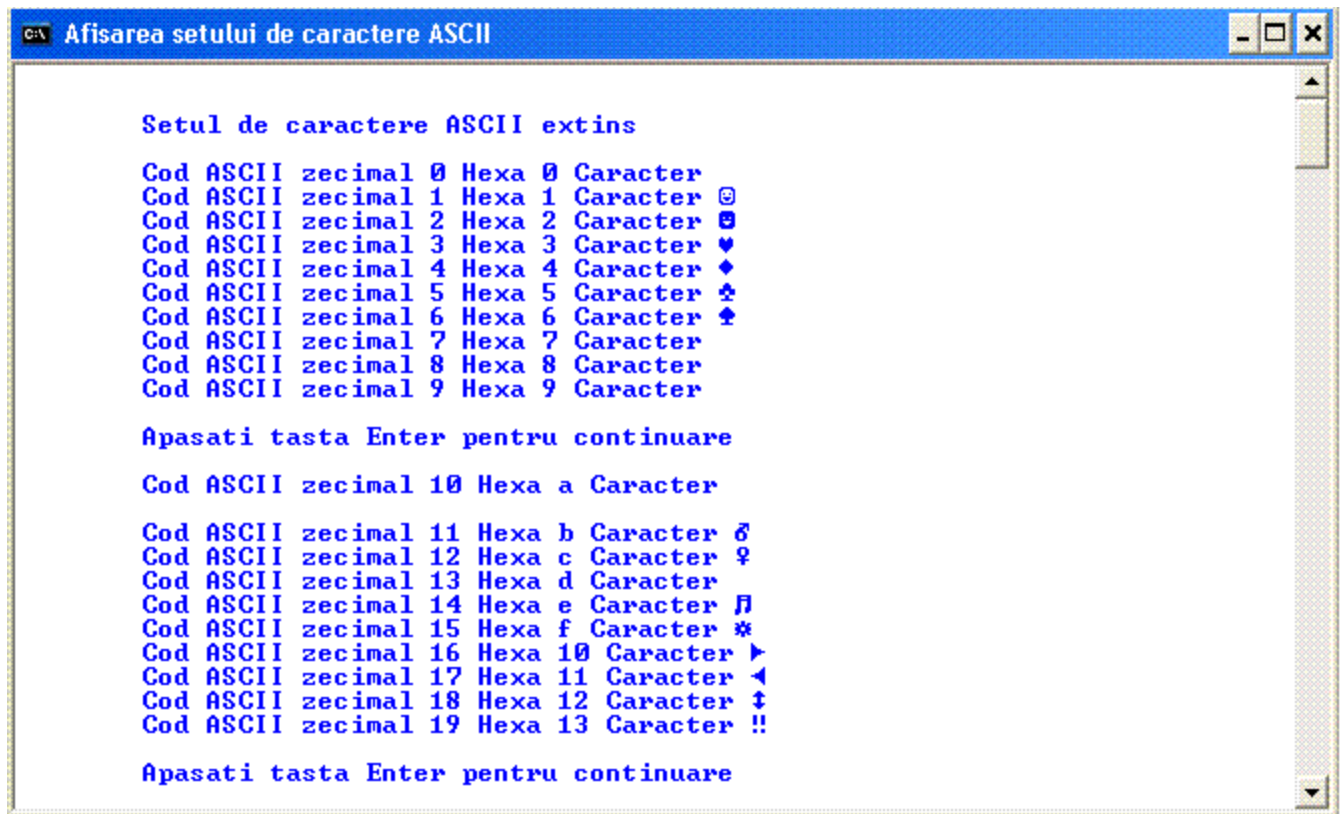
```

// Afisarea setului de caractere ASCII
#include "stdafx.h"
#include <iostream>
using namespace std;

int main(void)
{
    system("TITLE Afisarea setului de caractere ASCII");
    system("COLOR F9");
    int i ;
    cout << "\n\n\tSetul de caractere ASCII extins\n";
    for(i= 0;i < 256; i++){
        if ((i >0) && (i % 10 == 0)){
            cout << "\n\n\tApasati tasta Enter pentru continuare
";
            cin.get();
        }
        cout << "\n\tCod ASCII zecimal " << dec << i << " Hexa " << hex
<< i << " Caracter " << char(i);
    }
    cout << "\n\n\tPentru a termina apasati tasta Enter";
    cin.get();
    return 0;
}

```

Dupa rularea aplicatiei, obtinem :



```
C:\ Afisarea setului de caractere ASCII

Setul de caractere ASCII extins

Cod ASCII zecimal 0 Hexa 0 Caracter
Cod ASCII zecimal 1 Hexa 1 Caracter @
Cod ASCII zecimal 2 Hexa 2 Caracter ¢
Cod ASCII zecimal 3 Hexa 3 Caracter ♥
Cod ASCII zecimal 4 Hexa 4 Caracter ♦
Cod ASCII zecimal 5 Hexa 5 Caracter ♠
Cod ASCII zecimal 6 Hexa 6 Caracter ♣
Cod ASCII zecimal 7 Hexa 7 Caracter
Cod ASCII zecimal 8 Hexa 8 Caracter
Cod ASCII zecimal 9 Hexa 9 Caracter

Apasati tasta Enter pentru continuare

Cod ASCII zecimal 10 Hexa a Caracter
Cod ASCII zecimal 11 Hexa b Caracter ♂
Cod ASCII zecimal 12 Hexa c Caracter ♀
Cod ASCII zecimal 13 Hexa d Caracter
Cod ASCII zecimal 14 Hexa e Caracter ß
Cod ASCII zecimal 15 Hexa f Caracter *
Cod ASCII zecimal 16 Hexa 10 Caracter ►
Cod ASCII zecimal 17 Hexa 11 Caracter ◀
Cod ASCII zecimal 18 Hexa 12 Caracter ‡
Cod ASCII zecimal 19 Hexa 13 Caracter !!

Apasati tasta Enter pentru continuare
```

Apasand tasta Enter in continuare obtinem intreg setul extins de caractere ASCII

```
C:\ Afisarea setului de caractere ASCII

Cod ASCII zecimal 43 Hexa 2b Caracter +
Cod ASCII zecimal 44 Hexa 2c Caracter ,
Cod ASCII zecimal 45 Hexa 2d Caracter -
Cod ASCII zecimal 46 Hexa 2e Caracter .
Cod ASCII zecimal 47 Hexa 2f Caracter /
Cod ASCII zecimal 48 Hexa 30 Caracter 0
Cod ASCII zecimal 49 Hexa 31 Caracter 1

Apasati tasta Enter pentru continuare

Cod ASCII zecimal 50 Hexa 32 Caracter 2
Cod ASCII zecimal 51 Hexa 33 Caracter 3
Cod ASCII zecimal 52 Hexa 34 Caracter 4
Cod ASCII zecimal 53 Hexa 35 Caracter 5
Cod ASCII zecimal 54 Hexa 36 Caracter 6
Cod ASCII zecimal 55 Hexa 37 Caracter 7
Cod ASCII zecimal 56 Hexa 38 Caracter 8
Cod ASCII zecimal 57 Hexa 39 Caracter 9
Cod ASCII zecimal 58 Hexa 3a Caracter :
Cod ASCII zecimal 59 Hexa 3b Caracter ;

Apasati tasta Enter pentru continuare

Cod ASCII zecimal 60 Hexa 3c Caracter <
Cod ASCII zecimal 61 Hexa 3d Caracter =
Cod ASCII zecimal 62 Hexa 3e Caracter >
Cod ASCII zecimal 63 Hexa 3f Caracter ?
Cod ASCII zecimal 64 Hexa 40 Caracter @
Cod ASCII zecimal 65 Hexa 41 Caracter A
Cod ASCII zecimal 66 Hexa 42 Caracter B
Cod ASCII zecimal 67 Hexa 43 Caracter C
Cod ASCII zecimal 68 Hexa 44 Caracter D
Cod ASCII zecimal 69 Hexa 45 Caracter E
```

Pentru a face conversia inversa din caracter ASCII-int se procedeaza astfel:

```
// Conversia unui caracter ASCII intr-o valoare de tip int
#include "stdafx.h"
#include <iostream>
using namespace std;

int main(void)
{
    system("TITLE Conversia ASCII int ");
    system("COLOR F9");
    char car;
    int val_ascii;
    cout << "\n\n\tIntroduceti un caracter :";
    cin.get(car);
    val_ascii = static_cast<int>(car);
    // Se poate atribui lui val_ascii direct car
    // fara a fi nevoie de conversiastatic_cast(car) astfel:
    // val_ascii = car;
    cout << "\n\n\tValoarea ascii in zecimal a caracterului este: "<<
    val_ascii;
```

```

        cout << "\n\n\tValoarea ascii in hexa a caracterului este: "<< hex <<
val_ascii;
        cin.ignore();
        cin.get();
        return 0;
}

```

Cunoscand modul de codificare al caracterelor (codul ASCII) sa realizam un program care un sir de caractere ce contine litere mari si mici, dupa care converteste literele mari in litere mici si afiseaza sirul rezultat.

```

// Programul cere un sir de caractere care contine litere mari si mici.
// Converteste literele mari in litere mici si afiseaza sirul.
#include "stdafx.h"
#include < iostream >
#include < string >
using namespace std;
void conv_lit(char s[]);

int main(void)
{
    system("TITLE Conversie litere mari in litere mici");
    system("COLOR F9");
    char s[25];
    int l;
    cout<<"\n\tProgramul cere un sir de caractere cu litere mari si mici";
    cout<<"\n\tSe va afisa sirul convertit in litere mici";
    cout<<"\n\n\tIntroduceti un sir de caractere cu \n\tformat din litere mari
si mici : ";cin >> s;
    conv_lit(s);
    cout << "\n\n\tSirul transformat in litere mici este : " << s;
    cin.ignore();
    cin.get();
    return 0;
}

void conv_lit(char s[25])
{
    int i;
    i=0;
    while(s[i]!=0)
    {
        if( (s[i] >= 65) && (s[i] <= 90) ) s[i]=s[i]+32;
        i++;
    }
    return;
}

```

## Operatii binare in spatiul system

- **Afisare grafica valori binare**

Bazandu-ne pe operatorii binari studiati si aplicatiile realizate vom realiza simularea grafica a operatiilor binare. Vom afisa bitii ce reprezinta valoarea unei variabile prin mici dreptunghiuri. Pentru inceput vom realiza o aplicatie de tipul CLR Windows Forms Application care afiseaza secvente de dreptunghiuri pe ecran.

Deschidem un nou proiect Windows Forms Application intitulat "**secv\_dr**" pe care plasam un obiect de tip button numit button1 caruia ii schimbam atributul text in "Start" si un obiect timer pe caruia ii setam intervalul la 50. De data aceasta nu setam proprietatea "Enabled" la "true", o lasam "false".

Initializam variabilele i,h,w, obiectele "Desen" si "Pensula" in zona "#pragma region" pentru a asigura domeniul de vizibilitate in proceduri le stasate diferitelor evenimente.

Completam deci #pragma region cu :

```
static int i, w, h; // h, v dimensiunile unui dreptunghi
static System::Drawing::Graphics^ Desen;
static System::Drawing::SolidBrush^ Pensula;
```

Pe evenimentul click al obiectului button1 punem procedura:

```
Desen = this->CreateGraphics();
Pensula=gcnew
System::Drawing::SolidBrush(System::Drawing::Color::Blue);
Desen->Clear(System::Drawing::Color(this->BackColor));
w=this->Size.Width/60;
h=this->Size.Height/20;
i=2*w;
this->timer1->Enabled=true;
```

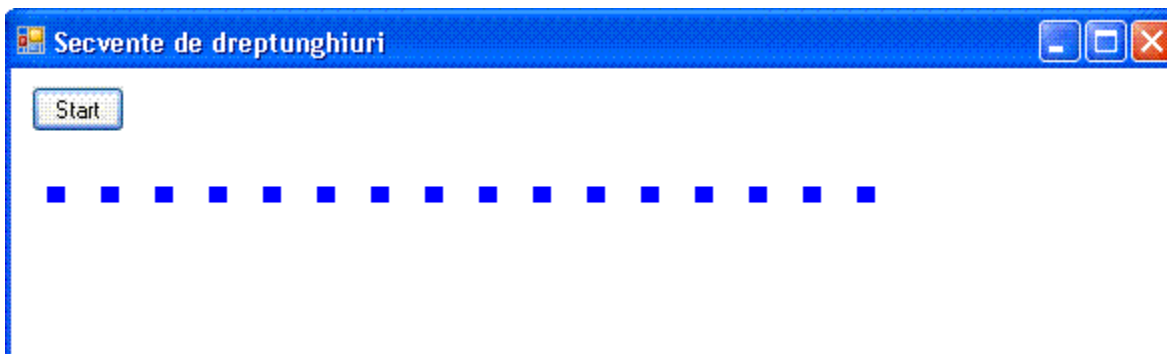
Se observa ca validarea "timer-ului" se face pe evenimentul click al obiectului button1

Completam procedura deschisa pe evenimentul Tick al obiectului timer1 cu :

```
Desen->FillRectangle(Pensula, i, this->Size.Height/3, w,h);
if (i <= this->Size.Width-3*w)
    i+=3*w;
else {
    i=2*w;
    Desen->Clear(System::Drawing::Color(this->BackColor));
}
```



Dupa rularea aplicatiei, obtinem secvente dinamice de dreptunghiuri:



Urmatoarea aplicatie isi propune sa converteasca o valoare numerica in binar dupa care sa afiseze aceasta valoare sub forma de dreptunghiuri pline pentru "1" si dreptunghiuri goale pentru "0"

Deschidem un nou proiect Windows Forms Application intitulat "**binar**" pe care plasam un obiect de tip button numit button1 caruia ii schimbam atributul text in "Start".

Completam deci #pragma region cu :

```
static unsigned int n; // numarul ce va fi convertit in binar si
afisat grafic
static int i, w, h; // h, v dimensiunile unui dreptunghi
static System::Drawing::Graphics^ Desen;
static System::Drawing::SolidBrush^ Pensula;
static System::Drawing::Pen^ Creion_blu ;
```

Pe evenimentul click al obiectului button1 punem procedura:

```
Desen = this->CreateGraphics();
Pensula=gcnnew
System::Drawing::SolidBrush(System::Drawing::Color::Blue);
Creion_blu=gcnnew System::Drawing::Pen(System::Drawing::Color::Blue);
Desen->Clear(System::Drawing::Color(this->BackColor));
w=this->Size.Width/96;
h=this->Size.Height/20;
i=2*w;
n=0xaaaaaaaa;
int val_b[32];
for (i=0; i < 32; i++){
    if (n%2==0)
        val_b[31-i]=0;
```

```

        else
            val_b[31-i]=1;
        n=n/2;
    }
    int x=2*w;
    for (i=0;i < 32;i++){
        if (val_b[i]==1)
            Desen->FillRectangle(Pensula, x, this->Size.Height/3, w,h);
        else
            Desen->DrawRectangle(Creion_blu, x, this->Size.Height/3,
w,h);
        x+=3*w;
    }

```

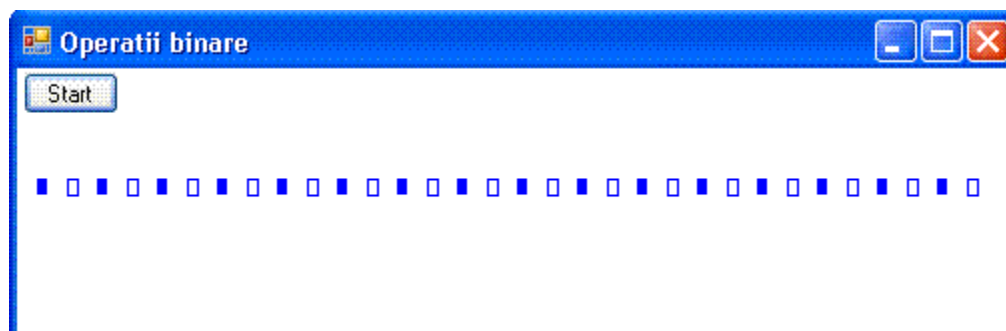
Aplicatia poate fi simplificata, folosind operatorul de siftare. Pe evenimentul click al obiectului button1 vom plasa noua procedura :

```

Desen = this->CreateGraphics();
Pensula=gcnw
System::Drawing::SolidBrush(System::Drawing::Color::Blue);
Creion_blu=gcnw System::Drawing::Pen(System::Drawing::Color::Blue);
Desen->Clear(System::Drawing::Color(this->BackColor));
w=this->Size.Width/96;
h=this->Size.Height/20;
int x=this->Size.Width-10*w;
n=0x00aa;
for (i=31; i >= 0; i--){
    int bit=((n >>(31-i)) & 1);
    if (bit==1)
        Desen->FillRectangle(Pensula, x, this->Size.Height/3, w,h);
    else
        Desen->DrawRectangle(Creion_blu, x, this->Size.Height/3,
w,h);
    x-=3*w;
}

```

Dupa rularea aplicatiei, obtinem valoarea in binar afisata grafica a numarului 0xaaaaaaaa:



Daca realizam un nou proiect "**binar\_v1**" si inlocuim numai procedura de pe evenimentul click al obiectului button1 aplicatia are acelasi rezultat. Observam ca nu a mai fost necesar sa introducem un vector care sa pastreze bitii ce compun numarul afisat.

In C# aplicatia devine::

```
namespace binar
{
    public partial class Form1 : Form
    {
        static System.UInt32 n; // numarul ce va fi convertit in binar si
        afisat grafic
        static int i, w, h; // h, v dimensiunile unui dreptunghi
        static System.Drawing.Graphics Desen;
        static System.Drawing.SolidBrush Pensula;
        static System.Drawing.Pen Creion_blu;
        public Form1()
        {
            InitializeComponent();

            private void button1_Click(object sender, EventArgs e)
            {
                Desen = this.CreateGraphics();
                Pensula=new
                System.Drawing.SolidBrush(System.Drawing.Color.Blue);
                Creion_blu=new System.Drawing.Pen(System.Drawing.Color.Blue);
                Desen.Clear(this.BackColor);
                w=this.Size.Width/96;
                h=this.Size.Height/20;
                int x=this.Size.Width-10*w;
                n=0xaaaaaaaa;
                for (i=31; i >= 0; i--){
                    System.UInt32 bit = ((n >> (31 - i)) & 1);
                    if (bit==1)
                        Desen.FillRectangle(Pensula, x,
                this.Size.Height/3, w,h);
                    else
                        Desen.DrawRectangle(Creion_blu, x,
                this.Size.Height/3, w,h);
                    x-=3*w;
                }
            }
        }
    }
}
```

- **Afisare grafica valori ASCII**

Ne propunem sa realizam o aplicatie care asteapta tiparirea unui caracter, il converteste in cod

ASCII si il afiseaza sub forma grafica.

Deschidem un nou proiect Windows Forms Application intitulat "**binar\_v2**" pe care plasam un obiect de tip textBox numit textBox1 si doua obiecte de tip label : label1 si label2 .

Completam #pragma region cu :

```
static unsigned short int n; // numarul ce va fi convertit in binar
si afisat grafic
static int i, w, h; // h, v dimensiunile unui dreptunghi
static System::Drawing::Graphics^ Desen;
static System::Drawing::SolidBrush^ Pensula;
static System::Drawing::Pen^ Creion_blu ;
static System::String^ txt;
```

Completam procedura deschisa pe evenimentul "TextChanged" al butonului textBox1 cu :

```
Desen = this->CreateGraphics();
Pensula=gcnew
System::Drawing::SolidBrush(System::Drawing::Color::Blue);
Creion_blu=gcnew System::Drawing::Pen(System::Drawing::Color::Blue);
Desen->Clear(System::Drawing::Color(this->BackColor));
w=this->Size.Width/30;
h=this->Size.Height/20;
if (System::String::Compare(this->textBox1->Text
, System::String::Empty)) {
    txt=this->textBox1->Text;
    char c=txt[0];
    c=System::Convert::ToChar(c);
    n=System::Convert::ToByte(c);
    this->label2->Text=System::Convert::ToString(n);
    int x=this->Size.Width-6*w;
    for (i=7; i >= 0; i--){
        int bit=((n >> (7-i)) & 1);
        if (bit==1)
            Desen->FillRectangle(Pensula, x, this-
>Size.Height/3, w,h);
        else
            Desen->DrawRectangle(Creion_blu, x, this-
>Size.Height/3, w,h);
        x-=3*w;
    }
}
```

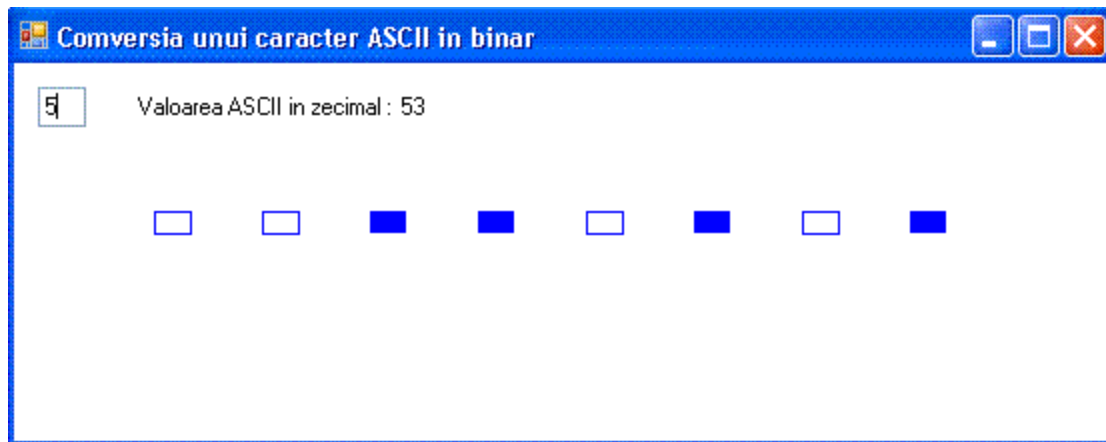
C#

```

namespace ascii
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        static int n; // numarul ce va fi convertit in binar si afisat
        static int i, w, h; // h, v dimensiunile unui dreptunghi
        static System.Drawing.Graphics Desen;
        static System.Drawing.SolidBrush Pensula;
        static System.Drawing.Pen Creion_blu ;
        static System.String txt;
        private void textBox1_TextChanged(object sender, EventArgs e)
        {
            Desen = this.CreateGraphics();
            Pensula=new
            System.Drawing.SolidBrush(System.Drawing.Color.Blue);
            Creion_blu=new System.Drawing.Pen(System.Drawing.Color.Blue);
            Desen.Clear(this.BackColor);
            w=this.Size.Width/30;
            h=this.Size.Height/20;
            if (this.textBox1.Text.Length>0){
                txt=this.textBox1.Text;
                char c=txt[0];
                c=System.Convert.ToChar(c);
                n=System.Convert.ToByte(c);
                this.label2.Text=System.Convert.ToString(n);
                int x=this.Size.Width-6*w;
                for (i=7; i >= 0; i--){
                    int bit=((n >> (7-i)) & 1);
                    if (bit==1)
                        Desen.FillRectangle(Pensula, x,
this.Size.Height/3, w,h);
                    else
                        Desen.DrawRectangle(Creion_blu, x,
this.Size.Height/3, w,h);
                    x-=3*w;
                }
            }
        }
    }
}

```

Rulam aplicatia , tastam cifra 5 si obtinem:



## • Afisare grafica shift

Folosind operatiile de deplasare stanga respectiv dreapta ,ne propunem sa realizam o aplicatie care simuleaza un joc de lumini.

Deschidem un nou proiect Windows Forms Application intitulat "**binar\_v3**". Plasam numai un singur obiect de tip timer numit timer1.

Completam #pragma region cu :

```
static unsigned short int n=0x5555; // numarul ce va fi deplasat
stanga dreapta si afisat
static int i, x, w, h; // h, v dimensiunile unui dreptunghi
static int sem=0;
static System::Drawing::Graphics^ Desen;
static System::Drawing::SolidBrush^ Pensula;
static System::Drawing::Pen^ Creion_blu ;
```

Completam procedura deschisa pe evenimentul "Paint" al form-ului cu :

```
Desen = this->CreateGraphics();
Pensula=gcnw
System::Drawing::SolidBrush(System::Drawing::Color::Blue);
Creion_blu=gcnw System::Drawing::Pen(System::Drawing::Color::Blue);
Desen->Clear(System::Drawing::Color(this->BackColor));
w=this->Size.Width/50;
h=this->Size.Height/20;
```

Completam procedura deschisa pe evenimentul "Tick" al timer-ului cu :

```

Desen->Clear(System::Drawing::Color(this->BackColor));
x=this->Size.Width-6*w;
if (sem==0){
    n=n << 1;
    sem=1;
}else{
    n=n >> 1;
    sem=0;
}
for (i=15; i >= 0; i--){
    int bit=((n >> (15-i)) & 1);
    if (bit==1)
        Desen->FillRectangle(Pensula, x, this->Size.Height/3, w,h);
    else
        Desen->DrawRectangle(Creion_blu, x, this->Size.Height/3,
w,h);
    x-=3*w;
}

```

Rulam aplicatia, obtinem:

