

Tipuri de date definite de utilizator: structuri

O structura este un tip de date definit de utilizator cuprinzand variabile multiple, de diferite tipuri. Structurile sunt utile pentru descrierea obiectelor. Spre deosebire de tablouri, care stocheaza mai multe valori de acelasi tip, structurile stocheaza valori de diferite tipuri.

- **Declararea unei structuri**

Structurile descriu obiecte prin caracteristicile acestora. Toate aceste caracteristici sunt deci corelate. Structurile permit in pachetarea variabilelor corelate intr-o singura entitate numita structura. Pentru a putea fi interpretate de compilator, structurile trebuie declarate. Sa presupunem ca vrem sa descriem parametri electrici ai unui aparat electric. Vom realiza o structura care sa cuprinda parametrii electrici care definesc respectivul aparat electric.

Urmatoarele instructiuni declara o structura ce reprezinta un aparat electric cu patru caracteristici: nume, curent , tensiunea nominala si frecventa.

```
struct ap_electric {  
    string nume;  
    int u_n;  
    double i;  
    int frecv;  
};
```

Structura se declara folosind cuvantul rezervat **struct** . Continutul structurii este pus intre acolade. Dupa ultima acolada, este obligatoriu caracterul punct si virgula. Corpul structurii se compune din variabilele corelate numite **variabile membru** .

- **Declararea unei variabile de structura**

O variabila de structura se declara ca o variabila a unui tip de date incorporat. Astfel pentru a declara o variabila a de tip ap_electri, utilizam:

```
ap_electric a;
```

Pot fi declarate de asemenea si variabile multiple:

```
ap_electric a1,a2,a3,a4;
```

Se pot declara tablouri de variabile astfel:

```
ap_electric a[25];
```

- **Accesarea variabilelor membru ale unei structuri**

Dupa declararea unei structuri si a unei variabile de structura, nu se atribuie valori pentru variabilele membru. Pentru a accesa o variabila membru, se utilizeaza operatorul punct. Sa presupunem ca am declarat o variabila de structura cu urmatoarea declaratie:

```
ap_electric a;
```

Pentru a atribui valoarea 220 variabilei membru u_n procedam astfel:

```
a.u_n=220;
```

Pentru a afisa tensiunea nominala vom utiliza

```
cout << a.u_n;
```

Vom realiza in continuare o aplicatie care defineste o structura ap_electric, atribuie valori variabilelor membru dupa care afiseaza aceste valori.

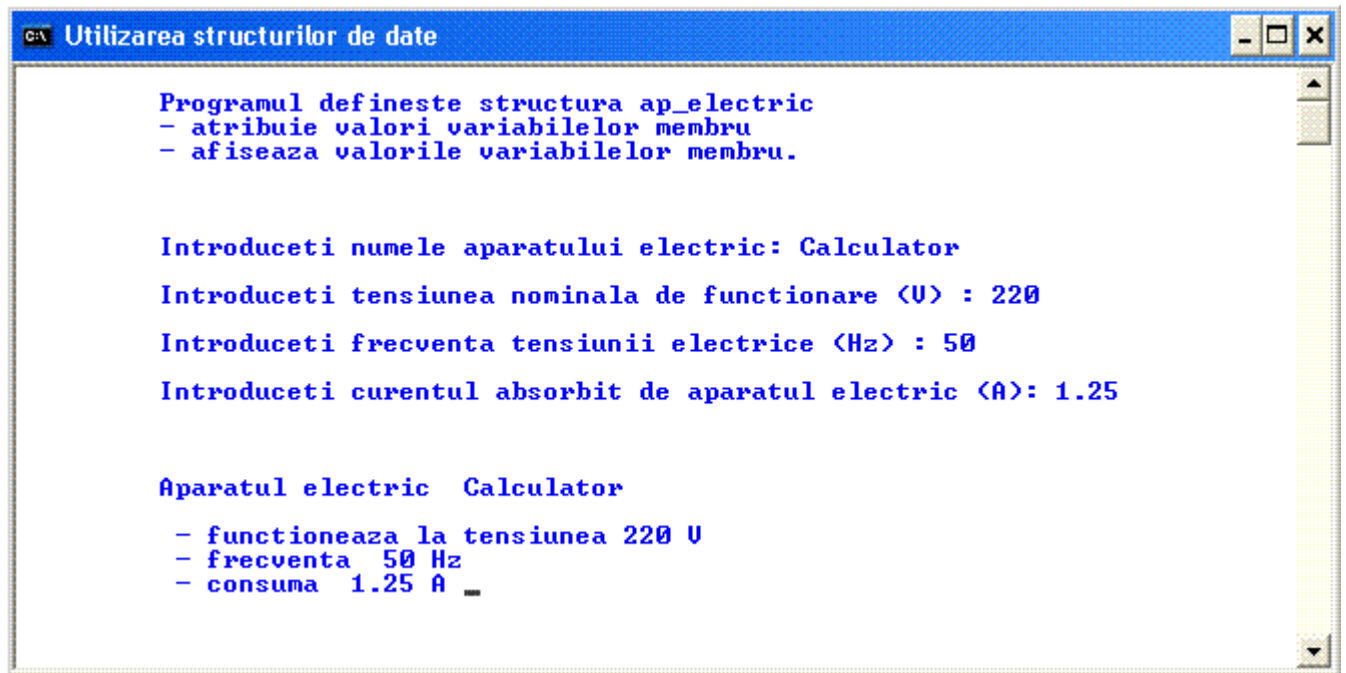
```
// Utilizarea structurilor de date
#include "stdafx.h"
#include < iostream >
#include < string >
using namespace std;
struct ap_electric {
    char nume[25];
    int u_n;
    double i;
    int frecv;
};
int main(void)
{
    system("TITLE Utilizarea structurilor de date ");
    system("COLOR F9");
```

```

cout << "\n\t Programul defineste structura ap_electric";
cout << "\n\t - atribuie valori variabilelor membru";
cout << "\n\t - afiseaza valorile variabilelor membru.\n\n\n";
cout << "\n\t Introduceti numele aparatului electric: ";
ap_electric a;
cin.getline(a.nume,25);
cout << "\n\t Introduceti tensiunea nominala de functionare (V) : ";
cin >> a.u_n;
cout << "\n\t Introduceti frecventa tensiunii electrice (Hz) : ";
cin >> a.frecv;
cout << "\n\t Introduceti curentul absorbit de aparatul electric (A):
";
cin >> a.i;
cout << "\n\n\n\t Aparatul electric " << a.nume ;
cout << "\n\n\t - functioneaza la tensiunea " << a.u_n << " V";
cout << "\n\t - frecventa " << a.frecv << " Hz ";
cout << "\n\t - consuma " << a.i << " A ";
cin.ignore();
cin.get();
return 0;
}

```

Rulam aplicatia, introducem datele cerute si obtinem:



```

C:\ Utilizarea structurilor de date

Programul defineste structura ap_electric
- atribuie valori variabilelor membru
- afiseaza valorile variabilelor membru.

Introduceti numele aparatului electric: Calculator
Introduceti tensiunea nominala de functionare (V) : 220
Introduceti frecventa tensiunii electrice (Hz) : 50
Introduceti curentul absorbit de aparatul electric (A): 1.25

Aparatul electric  Calculator
- functioneaza la tensiunea 220 V
- frecventa 50 Hz
- consuma 1.25 A

```

C#

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace struct_01
{
    class Program
    {
        public struct ap_electric {
            public String nume;
            public int u_n;
            public double i;
            public int frecv;
        };
        static void Main(string[] args)
        {
            String u_n_s, i_s, frecv_s;
            Console.WriteLine("\n\t Programul defineste structura ap_electric");
            Console.WriteLine("\n\t - atribuie valori variabilelor membru");
            Console.WriteLine("\n\t - afiseaza valorile variabilelor
membru.\n\n\n");
            Console.WriteLine("\n\t Introduceti numele aparatului electric: ");
            ap_electric a;
            a.nume = Console.ReadLine();
            Console.WriteLine("\n\t Introduceti tensiunea nominala de functionare
(V) : ");
            u_n_s = Console.ReadLine();
            a.u_n = System.Convert.ToInt16(u_n_s);
            Console.WriteLine("\n\t Introduceti frecventa tensiunii electrice
(Hz) : ");
            frecv_s = Console.ReadLine();
            a.frecv = System.Convert.ToInt16(frecv_s);
            Console.WriteLine("\n\t Introduceti curentul absorbit de aparatul
electric (A) : ");
            i_s = Console.ReadLine();
            a.i = System.Convert.ToDouble(i_s);
            Console.WriteLine("\n\n\n\t Aparatul electric "+a.nume );
            Console.WriteLine("\n\n\t - functioneaza la tensiunea " +a.u_n );
            Console.WriteLine("\n\t - frecventa " + a.frecv);
            Console.WriteLine("\n\t - consuma " + a.i);
            Console.ReadKey();
        }
    }
}

```

Sa utilizam acum o variabila de structura de tip tablou. Vom realua aplicatia anterioara si o vom modifica astfel incat aplicatia sa preia caracteristicile mai multor aparate electrice si apoi sa le afiseze.

```

// Utilizarea structurilor de date
#include "stdafx.h"
#include < iostream >
#include < string >
using namespace std;
struct ap_electric {
    char nume[25];
    int u_n;
    double i;
    int frecv;
};
int main(void)
{
    system("TITLE Utilizarea structurilor de date ");
    system("COLOR F9");
    cout << "\n\t Programul defineste structura ap_electric";
    cout << "\n\t Se declara o variabila de structura de tip tablou\n\n";
    const int nr_ap=3;
    ap_electric a[nr_ap];
    for (int i=0; i < nr_ap ;i++){
        cout << "\n\t Introduceti numele aparatului electric: "<< i+1
<< " : ";
        cin.getline(a[i].nume,25);
        cout << "\t Introduceti tensiunea nominala de functionare (V) : ";
        cin >> a[i].u_n;
        cout << "\t Introduceti frecventa tensiunii electrice (Hz) : ";
        cin >> a[i].frecv;
        cout << "\t Introduceti curentul absorbit de aparatul electric (A): ";
        cin >> a[i].i;
        cin.ignore();
    }
    cout << "\n\n\t Caracteristicile aparatelor electrice\n ";
    for (int i=0; i < nr_ap ;i++){
        cout << "\n\t " << a[i].nume << "\t\tU=" << a[i].u_n << " V " <<
"\tf=";
        cout << a[i].frecv << " Hz " << "\tI=" << a[i].i << " A ";
    }
    cin.get();
    return 0;
}

```

Rulam aplicatia, introducem datele cerute si obtinem:

```
C:\ Utilizarea structurilor de date

Programul defineste structura ap_electric
Se declara o variabila de structura de tip tablou

Introduceti numele aparatului electric: 1 : Fier de calcat
Introduceti tensiunea nominala de functionare (U) : 220
Introduceti frecventa tensiunii electrice (Hz) : 50
Introduceti curentul absorbit de aparatul electric (A): 2.55

Introduceti numele aparatului electric: 2 : Motor trifazic
Introduceti tensiunea nominala de functionare (U) : 380
Introduceti frecventa tensiunii electrice (Hz) : 50
Introduceti curentul absorbit de aparatul electric (A): 7.5

Introduceti numele aparatului electric: 3 : Calculator
Introduceti tensiunea nominala de functionare (U) : 220
Introduceti frecventa tensiunii electrice (Hz) : 50
Introduceti curentul absorbit de aparatul electric (A): 1.2

Caracteristicile aparatelor electrice

Fier de calcat      U=220 V      f=50 Hz      I=2.55 A
Motor trifazic     U=380 V      f=50 Hz      I=7.5 A
Calculator          U=220 V      f=50 Hz      I=1.2 A
```

• Initializarea unei structuri

O structura de date se initializeaza prin utilizarea listei de initializare sau prin utilizarea unui constructor.

Liste de initializare

O structura de date poate fi initializata asemanator unei variabile in momentul declararii variabilei de structura astfel:

```
struct ap_electric {
    char nume[25];
    int u_n;
    double i;
    int freqv;
};
ap_electric a={"Calculator", 220, 2.5 ,50};
```

Constructori

Prin initializarea unei structuri se initializeaza un tip de data conform cu structura declarata. Se poate spune ca s-a creat o instanta a structuri de date.

Constructorul este o functie care este apelata automat la crearea unei instante a unei structuri de date.

In exemplele anterioare nu am utilizat constructori. Daca in aplicatie nu se scrie un constructor, atunci se apeleaza automat constructorul implicit. Constructorul implicit a fost apelat atunci cand s-a facut declaratia:

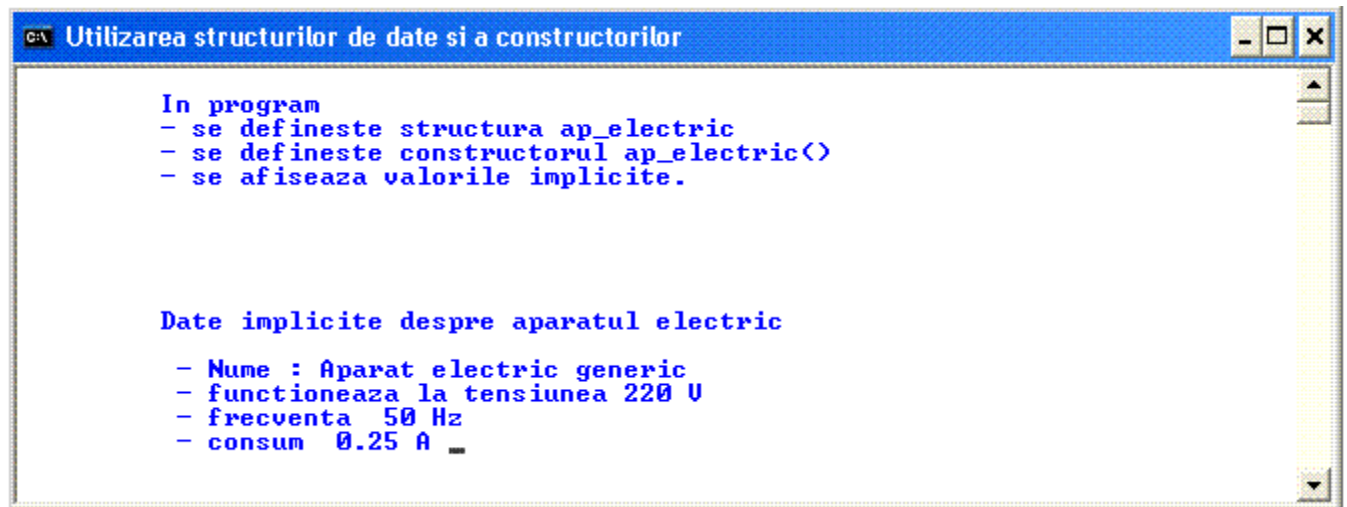
ap_electric a

- **Constructori fara argumente**

Se poate include in cadrul structurii de date un constructor care atribuie valori prestabilite variabilelor membru.

```
// Utilizarea structurilor de date si a constructorilor
#include "stdafx.h"
#include < iostream >
#include < string >
using namespace std;
struct ap_electric {
    string nume;
    int u_n;
    double i;
    int frecv;
    ap_electric(){
        nume="Aparat electric generic";
        u_n=220;
        i=0.25;
        frecv=50;
    }
};
int main(void)
{
    system("TITLE Utilizarea structurilor de date si a constructorilor ");
    system("COLOR F9");
    cout << "\n\t In program";
    cout << "\n\t - se defineste structura ap_electric";
    cout << "\n\t - se defineste constructorul ap_electric()";
    cout << "\n\t - se afiseaza valorile implicite.\n\n\n";
    ap_electric a;
    cout << "\n\n\n\t Date implicite despre aparatul electric ";
    cout << "\n\n\t - Nume : " << a.nume;
    cout << "\n\t - functioneaza la tensiunea " << a.u_n << " V";
    cout << "\n\t - frecventa " << a.frecv << " Hz ";
    cout << "\n\t - consum " << a.i << " A ";
    cin.ignore();
    cin.get();
    return 0;
}
```

Constructorul fara argumente asemenea constructorului implicit se lanseaza in momentul lansarii instructiunii: **ap_electric a**



• Constructori cu argumente

Exista posibilitatea initializarii variabilelor membru cu valorile furnizate in timpul rularii programului. Acest lucru se realizeaza adaugand argumente la functia constructor.

```
// Utilizarea structurilor de date si a constructorilor cu argumente
#include "stdafx.h"
#include <iostream>
#include <string>
using namespace std;
struct ap_electric {
    string nume;
    int u_n;
    double i_n;
    int frecv;
    ap_electric() {
        nume="Neprecizat";
        u_n=220;
        i_n=0;
        frecv=50;
    }
    ap_electric(string n, int u , double i, int f){
        nume=n;
        u_n=u;
        i_n=i;
        frecv=f;
    }
};
int main(void)
{
    string denum;
    int volti;
    double amperi;
```



```

int herti;
system("TITLE Utilizarea structurilor de date si a constructorilor ");
system("COLOR F9");
cout << "\n\t In program";
cout << "\n\t - se defineste structura ap_electric";
cout << "\n\t - se defineste constructorul ap_electric()";
cout << "\n\n\n\t Introduceti denumirea aparatului electric : ";
getline(cin,denum);
cout << "\n\t Introduceti tensiunea nominala de functionare (V) : ";
cin >> volti;
cout << "\n\t Introduceti frecventa tensiunii electrice (Hz) : ";
cin >> herti;
cout << "\n\t Introduceti curentul absorbit de aparatul electric (A) : ";

cin >> amperi;
ap_electric a(denum,volti,amperi,herti);
cout << "\n\n\n\t Aparatul electric " << a.nume ;
cout << "\n\n\n\t - functioneaza la tensiunea " << a.u_n << " V";
cout << "\n\t - frecventa " << a.frecv << " Hz ";
cout << "\n\t - consuma " << a.i_n << " A ";
cin.ignore();
cin.get();
return 0;
}

```

Constructori cu argumente in C#

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace struct_01
{
    class Program
    {
        struct ap_electric
        {
            public string nume;
            public int u_n;
            public double i_n;
            public int frecv;

            public ap_electric(double i)
            {
                nume = "Bec";
                u_n = 220;
                i_n = i;
                frecv = 50;
            }
            public ap_electric(string n)

```

```

        {
            nume = n;
            u_n = 220;
            i_n = 0;
            frecv = 50;
        }
        public ap_electric(string n, int u, double i, int f)
        {
            nume = n;
            u_n = u;
            i_n = i;
            frecv = f;
        }
    }
    static void Main(string[] args)
    {
        Console.WriteLine("\n\t Programul defineste structura ap_electric");
        Console.WriteLine("\n\t - atribuie valori variabilelor membru");
        Console.WriteLine("\n\t - afiseaza valorile variabilelor
membru.\n\n\n");
        Console.WriteLine("\n\t Introduceti numele aparatului electric: ");
        //ap_electric a = new ap_electric();
        ap_electric a = new ap_electric(12.25);
        //ap_electric a = new ap_electric("Calculator");
        //ap_electric a = new ap_electric("Calculator", 220, 1.2, 50);
        Console.WriteLine("\n\n\n\t Aparatul electric  " + a.nume);
        Console.WriteLine("\n\n\t - functioneaza la tensiunea " + a.u_n);
        Console.WriteLine("\n\t - frecventa  " + a.frecv);
        Console.WriteLine("\n\t - consuma  " + a.i_n);
        Console.ReadKey();
    }
}

```

• Prototipul constructorului

Implementarea constructorului poate fi realizata in afara structurii de date cu conditia sa se defineasca un prototip al constructorului. Utilizand un prototip de constructor, aplicatia devine:

```

// Utilizarea structurilor de date si a constructorilor cu prototip
#include "stdafx.h"
#include < iostream >
#include < string >
using namespace std;
struct ap_electric {
    string nume;
    int u_n;
    double i_n;
}

```

```

        int freqv;
        ap_electric();// Prototipul constructorului implicit
        ap_electric(string n, int u , double i, int f); // Prototipul
constructorului
};
ap_electric::ap_electric(){ //Constructorul implicit
        nume="Neprecizat";
        u_n=220;
        i_n=0;
        freqv=50;
    }
ap_electric::ap_electric(string n, int u , double i, int f){ //Constructorul
        nume=n;
        u_n=u;
        i_n=i;
        freqv=f;
    }
int main(void)
{
    string denum;
    int volti;
    double amperi;
    int herti;
    system("TITLE Utilizarea prototipurilor pentru constructori ");
    system("COLOR F9");
    cout << "\n\t In program";
    cout << "\n\t - se defineste structura ap_electric";
    cout << "\n\t - se defineste constructorul ap_electric()";
    cout << "\n\n\t Introduceti denumirea aparatului electric : ";
    getline(cin,denum);
    cout << "\n\t Introduceti tensiunea nominala de functionare (V) : ";
    cin >> volti;
    cout << "\n\t Introduceti frecventa tensiunii electrice (Hz) : ";
    cin >> herti;
    cout << "\n\t Introduceti curentul absorbit de aparatul electric (A): ";
    cin >> amperi;
    ap_electric a(denum,volti,amperi,herti);
    cout << "\n\n\t Aparatul electric " << a.nume ;
    cout << "\n\n\t - functioneaza la tensiunea " << a.u_n << " V";
    cout << "\n\t - frecventa " << a.freqv << " Hz ";
    cout << "\n\t - consuma " << a.i_n << " A ";
    cin.ignore();
    cin.get();
    return 0;
}

```

Dupa cum se observa pentru a defini constructorul s-a folosit operatorul de rezolutie a vizibilitatii (::) pentru a preciza ca faptul ca functia respectiva apartine structurii definite anterior si nu este o functie independenta.

- **Structuri ca argumente de functii**

O structura poate fi folosita ca argument de functie.

Sa reluam o aplicatie anterioara care atribuie valori variabilelor membru al unei structuri dupa care afiseaza aceste valori, folosind de data aceasta doua functii : "ini_val" si "afis_val"

```
// Utilizarea structurilor de date si a functiilor
#include "stdafx.h"
#include < iostream >
#include < string >
using namespace std;
struct ap_electric {
    string nume;
    int u_n;
    double i_n;
    int frecv;
};
void ini_val(ap_electric&);
void afis_val(const ap_electric&);
int main(void)
{
    system("TITLE Utilizarea functiilor cu argument structurii de date ");
    system("COLOR F9");
    cout << "\n\t Programul defineste structura ap_electric";
    cout << "\n\t - atribuie valori variabilelor membru folosind functia
ini_val";
    cout << "\n\t - afiseaza valorile variabilelor membru folosind functia
afis_val .\n\n\n";
    ap_electric a;
    ini_val(a);
    cout << "\n\n\n\t Datele aparatului electric  " ;
    afis_val(a);
    cin.ignore();
    cin.get();
    return 0;
}
void ini_val(ap_electric& ap_e){// ap_e fiind o variabila formala
    cout << "\n\t Introduceti numele aparatului electric: ";
    getline(cin,ap_e.nume);
    cout << "\n\t Introduceti tensiunea nominala de functionare (V) : ";
    cin >> ap_e.u_n;
    cout << "\n\t Introduceti frecventa tensiunii electrice (Hz) : ";
    cin >> ap_e.frecv;
    cout << "\n\t Introduceti curentul absorbit de aparatul electric (A):
";
    cin >> ap_e.i_n;
}
void afis_val(const ap_electric& ap_e){// ap_e fiind o variabila formala
    cout << "\n\n\n\t Denumire aparat:  " << ap_e.nume ;
    cout << "\n\n\t - functioneaza la tensiunea " << ap_e.u_n << " V";
    cout << "\n\t - frecventa  " << ap_e.frecv << " Hz ";
    cout << "\n\t - consuma  " << ap_e.i_n << " A ";
}
```

Dupa cum se observa structura de date este transmisa functiei prin referinta. Functia "ini_val" modifica variabilele membru ale structurii. Functia "afis_val" afiseaza valorile variabilelor membru fara a face modificarea lor. Pentru mai multa siguranta, in cadrul functiei "afis_val" s-a declarat ca argument "const ap_electric" pentru a preveni modificarea accidentala a valorilor variabilelor membru.

- **Imbricarea structurilor de date**

Structurile de date permit imbricarea unei structurii in alte structuri. Sa presupunem ca vrem sa descriem aparatul electric atat din punct de vedere electric cat si mecanic. Vom defini o structura pentru caracteristicile electrice numita "c_electrice" si o structura pentru caracteristicile mecanice numita c_mecanice.

Pentru a defini aparatul electric vom defini o structura care contine numele si cele doua structuri definite anterior.

```
// Utilizarea structurilor de date imbricate
#include "stdafx.h"
#include < iostream >
#include < string >
using namespace std;

struct c_electrice {
    int u_n;
    double i_n;
    int freqv;
};
struct c_mecanice {
    int masa;
    double lung;
    double lat;
    double inalt;
};
struct ap_electric { // imbricarea structurilor c_electrice si c_mecanice
    char nume[25];
    c_electrice electr;
    c_mecanice mec;
};

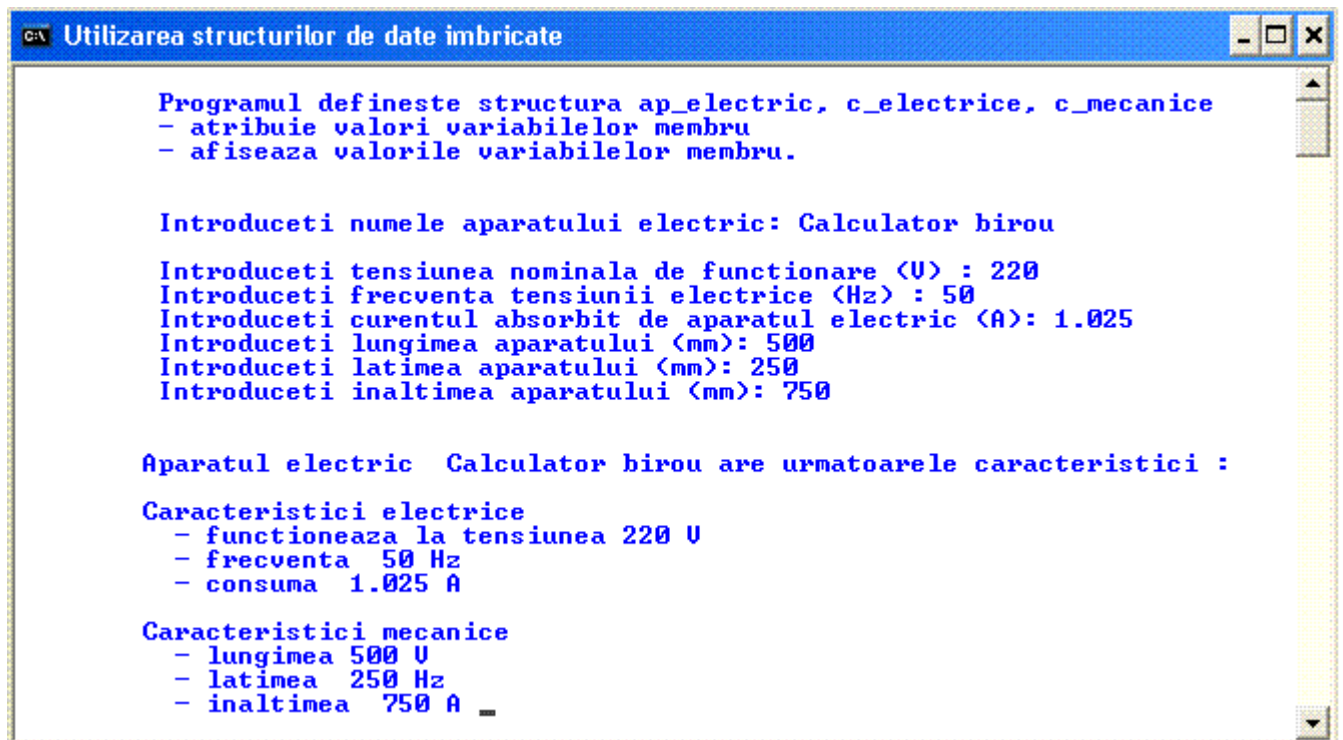
int main(void)
{
    system("TITLE Utilizarea structurilor de date imbricate ");
    system("COLOR F9");
    cout << "\n\t Programul defineste structura ap_electric, c_electrice, c_mecanice";
    cout << "\n\t - atribuie valori variabilelor membru";
    cout << "\n\t - afiseaza valorile variabilelor membru.\n\n";
    cout << "\n\t Introduceti numele aparatului electric: ";
    ap_electric a;
    cin.getline(a.nume,25);
    cout << "\n\t Introduceti tensiunea nominala de functionare (V) : ";
```

```

    cin >> a.electr.u_n;
    cout << "\t Introduceti frecventa tensiunii electrice (Hz) : ";
    cin >> a.electr.frecv;
    cout << "\t Introduceti curentul absorbit de aparatul electric (A): ";
    cin >> a.electr.i_n;
    cout << "\t Introduceti lungimea aparatului (mm): ";
    cin >> a.mec.lung;
    cout << "\t Introduceti latimea aparatului (mm): ";
    cin >> a.mec.lat;
    cout << "\t Introduceti inaltimea aparatului (mm): ";
    cin >> a.mec.inalt;
    cout << "\n\n\tAparatul electric  " << a.nume << " are urmatoarele
caracteristici :";
    cout << "\n\n\tCaracteristici electrice ";
    cout << "\n\t - functioneaza la tensiunea " << a.electr.u_n << " V";
    cout << "\n\t - frecventa  " << a.electr.frecv << " Hz ";
    cout << "\n\t - consuma  " << a.electr.i_n << " A ";
    cout << "\n\n\tCaracteristici mecanice ";
    cout << "\n\t - lungimea  " << a.mec.lung << " V";
    cout << "\n\t - latimea  " << a.mec.lat << " Hz ";
    cout << "\n\t - inaltimea  " << a.mec.inalt << " A ";
    cin.ignore();
    cin.get();
    return 0;
}

```

Dupa rularea aplicatiei si introducerea datelor cerute, obtinem:



```

Utilizarea structurilor de date imbricate

Programul defineste structura ap_electric, c_electrice, c_mecanice
- atribuie valori variabilelor membru
- afiseaza valorile variabilelor membru.

Introduceti numele aparatului electric: Calculator birou

Introduceti tensiunea nominala de functionare (U) : 220
Introduceti frecventa tensiunii electrice (Hz) : 50
Introduceti curentul absorbit de aparatul electric (A): 1.025
Introduceti lungimea aparatului (mm): 500
Introduceti latimea aparatului (mm): 250
Introduceti inaltimea aparatului (mm): 750

Aparatul electric  Calculator birou are urmatoarele caracteristici :

Caracteristici electrice
- functioneaza la tensiunea 220 U
- frecventa  50 Hz
- consuma  1.025 A

Caracteristici mecanice
- lungimea  500 U
- latimea  250 Hz
- inaltimea  750 A

```

Tipuri de date definite de utilizator: clase

O structura este un tip de date definit de utilizator. Structura pentru descrierea obiectelor se numeste clasa.

- **Declararea unei clase**

Urmatoarele instructiuni declara o clasa ce reprezinta un aparat electric cu patru atribute: nume, tensiunea nominala, frecventa si curentul nominal.

```
class ap_electric {
private:
    string nume;
    int u_n;
    int freqv;
    double i_n;
public:
    functie membru_1;
    ...
    functie membru_n
};
```

Dupa cum se vede o clasa se declara similar cu o structura de date, cu deosebirea ca se foloseste cuvantul rezervat **class** .

In cadrul unei clase, variabilele membru sunt de tip "privat" spre deosebire de variabilele membru ale unei structuri care sunt de tip "public". O variabila membru a unei clase poate fi accesata numai de o functie membru a acelei clase.

Variabilele membru sunt deci "ascunse". Procedul de ascundere a informatiilor referitoare la la clasa se numeste **incapsulare** . Prin incapsulare, variabilele membru nu mai pot fi setate direct decat prin intermediul functiilor membru, deci exista posibilitatea validarii datelor ce se atribuie variabilelor membru, de catre functiile membru. Citirea variabilelor membru se face tot prin intermediul functiilor membru, existand posibilitatea deci de a proteja citirea neautorizata a valorilor variabilelor membru de catre utilizatorii care nu au dreptul de a accesa informatiile respective.

Sa realizam o aplicatie care utilizeaza clasa ap_electric. Programul inscrie variabilele membru ale clasei apoi afiseaza valorile inscrise si calculeaza puterea consumata utilizand functii membru ale clasei ap_electric.

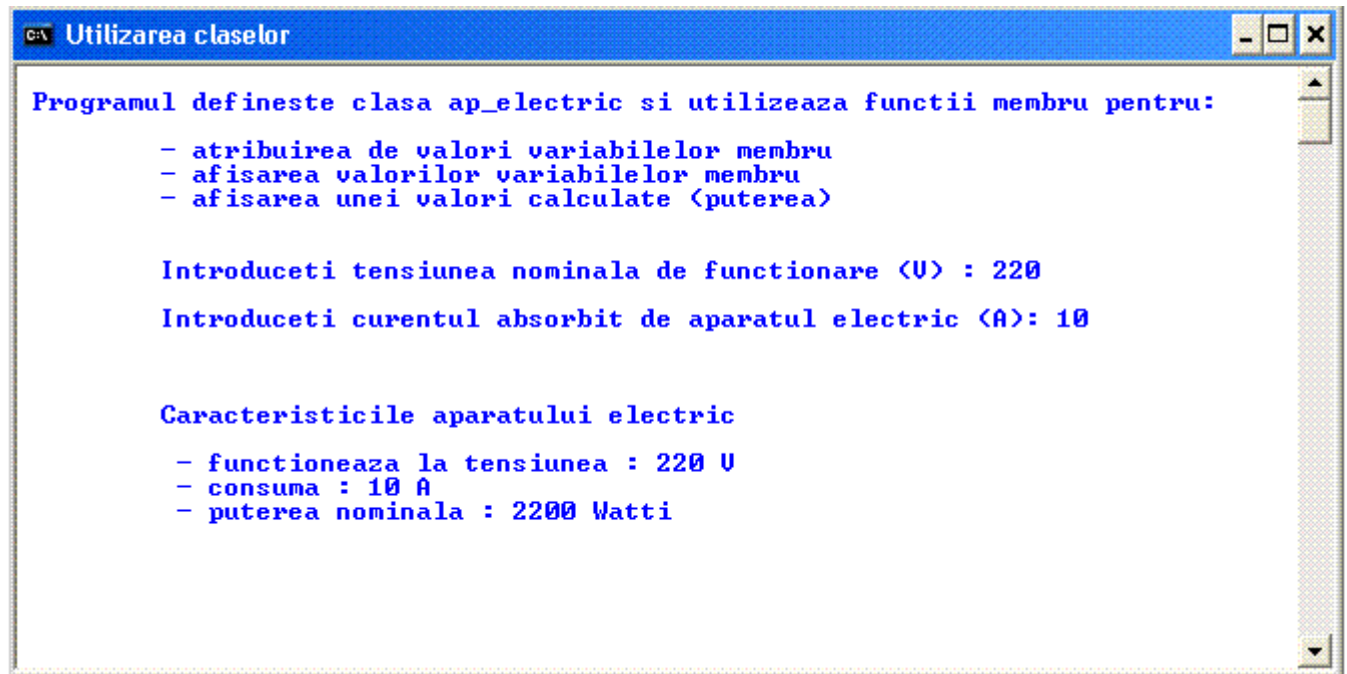
```
// Utilizarea claselor-calculul puterii unui consumator
#include "stdafx.h"
#include < iostream >
#include < string >
```

```

using namespace std;
class ap_electric {
    private:
        double u_n;
        double i_n;
    public:
        void set_u(double);
        int vezi_u() const;
        void set_i(double);
        double vezi_i() const;
        double vezi_p() const;
};

void ap_electric::set_u(double tens) {
    if (tens<< "\n Programul defineste clasa ap_electric si utilizeaza
functii membru pentru:" ;
    cout << "\n\n\t - atribuirea de valori variabilelor membru";
    cout << "\n\t - afisarea valorilor variabilelor membru";
    cout << "\n\t - afisarea unei valori calculate (puterea)\n\n";
    ap_electric a;
    double tens;
    double crnt;
    cout << "\n\t Introduceti tensiunea nominala de functionare (V) : ";
    cin >> tens;
    a.set_u(tens);
    cout << "\n\t Introduceti curentul absorbit de aparatul electric (A):
";
    cin >> crnt;
    a.set_i(crnt);
    cout << "\n\n\n\t Caracteristicile aparatului electric " ;
    cout << "\n\n\t - functioneaza la tensiunea : " << a.vezi_u() << "
V";
    cout << "\n\t - consuma : " << a.vezi_i() << " A ";
    cout << "\n\t - puterea nominala : " << a.vezi_p() << " Watti ";
    cin.ignore();
    cin.get();
    return 0;
}

```

Pentru scrierea functiilor membru s-a utilizat operatorul de vizibilitate(::).

Functiile membru se mai numesc si metode ale clasei. Cu alte cuvinte s-au definit metodele `vezi_u`, `vezi_i`, `vezi_p` etc.

Prin declaratia **ap_electric a** ; s-a creat o instanta a clasei "ap_electric" sau s-a creat obiectul a.

In programul principal s-a afisat de exemplu puterea prin lansarea functiei membru `vezi_p()`. Se spune ca s-a invocat metoda `vezi_p()` a obiectului a care este o instanta a clasei "ap_electric"

C#

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace struct_01
{
    class Program
    {
        class ap_electric {
            private double u_n;
            private double i_n;
            public void set_u(double tens) {
                if (tens
```

- **Clase ca argumente de functii**

La fel ca si structurile, clasele pot fi utilizate ca argumente de functii. Aplicatia anterioara poate fi rescrisa utilizand functiile "set_val" si "vezi_val", functii ce au ca argument clasa "ap_electric"

```
// Utilizarea claselor-calculul puterii unui consumator
#include "stdafx.h"
#include < iostream >
#include < string >
using namespace std;
class ap_electric {
    private:
        double u_n;
        double i_n;
    public:
        void set_u(double);
        int vezi_u() const;
        void set_i(double);
        double vezi_i() const;
        double vezi_p() const;
};

void ap_electric::set_u(double tens) {
    if (tens << "\n\t Programul defineste clasa ap_electric";
    cout << "\n\t - atribuie valori variabilelor membru folosind functia
set_val";
    cout << "\n\t - afiseaza valorile variabilelor membru folosind functia
vezi_val .\n\n\n";
    ap_electric a;
    set_val(a);
    cout << "\n\n\n\t Caracteristicile aparatului electric  " ;
    vezi_val(a);
    cin.ignore();
    cin.get();
    return 0;
}

void set_val(ap_electric& el){// el fiind o variabila formala
    string num;
    double tens;
    double crnt;
    cout << "\n\t Introduceti tensiunea nominala de functionare (V) : ";
    cin >> tens;
    el.set_u(tens);
    cout << "\n\t Introduceti curentul absorbit de aparatul electric (A):
";
    cin >> crnt;
    el.set_i(crnt);
}

void vezi_val(const ap_electric& el){// el fiind o variabila formala
    cout << "\n\n\t - functioneaza la tensiunea " << el.vezi_u() << " V";
    cout << "\n\t - consuma " << el.vezi_i() << " A ";
    cout << "\n\t - puterea " << el.vezi_p() << " Watti ";
}
```

Sa reluam aplicatia anterioara si sa introducem noi attribute ale clasei "ap_electric", cum ar fi numele aparatului, frecventa la care lucreaza etc.

```
// Utilizarea claselor
#include "stdafx.h"
#include < iostream >
#include < string >
using namespace std;
class ap_electric {
    private:
        string nume;
        int u_n;
        double i_n;
        int f;
    public:
        void set_nume(string);
        string vezi_nume() const;
        void set_u(int);
        int vezi_u() const;
        void set_f(int);
        int vezi_f() const;
        void set_i(double);
        double vezi_i() const;
        double vezi_p() const;
};
void ap_electric::set_nume(string s){
    if (s.length() ==0)
        nume="Nu are";
    else
        nume=s;
}
string ap_electric::vezi_nume() const {
    return nume;
}
void ap_electric::set_u(int tens) {
    if (tens<< "\n\t Programul defineste clasa ap_electric";
    cout << "\n\t - atribuie valori variabilelor membru folosind functia
set_val";
    cout << "\n\t - afiseaza valorile variabilelor membru folosind functia
vezi_val .\n\n\n";
    ap_electric a;
    set_val(a);
    cout << "\n\n\n\t Datele aparatului electric  " ;
    vezi_val(a);
    cin.ignore();
    cin.get();
    return 0;
}
void set_val(ap_electric& el){// el fiind o variabila formala
    string num;
    int tens;
    int fr;
```

```

double crnt;
cout << "\n\t Introduceti numele aparatului electric: ";
getline(cin,num);
el.set_nume(num);
cout << "\n\t Introduceti tensiunea nominala de functionare (V) : ";
cin >> tens;
el.set_u(tens);
cout << "\n\t Introduceti frecventa tensiunii electrice (Hz) : ";
cin >> fr;
el.set_f(fr);
cout << "\n\t Introduceti curentul absorbit de aparatul electric (A):
";
    cin >> crnt;
    el.set_i(crnt);
}
void vezi_val(const ap_electric& el){// el fiind o variabila formala
    cout << "\n\n\n\t Denumire aparat: " << el.vezi_nume() ;
    cout << "\n\n\t - functioneaza la tensiunea " << el.vezi_u() << " V";
    cout << "\n\t - frecventa " << el.vezi_f() << " Hz ";
    cout << "\n\t - curentul " << el.vezi_i() << " A ";
    cout << "\n\t - puterea " << el.vezi_p() << " Watti ";
}

```