# Elemente de programare în limbajul C++.

## Elemente de baza

**Program de calculator**

Set de instructiuni scris de programatori in vederera rularii pe un calculator.

**Limbaj de programare**

Este un set bine definit de expresii si reguli (sau tehnici) valide de formulare a instructiunilor pentru un computer. Un limbaj de programare are definite un set de reguli sintactice si semantice.

**Limbajul de programare C**

Limbajul C a fost creat la începutul anilor '70 de catre Brian W Kernigham si Dennis M Ritchie de la Bell Laboratories New Jersey
Caracteristicile distinctive ale limbajului au fost clar definite de la început, ele pastrîndu-se în toate dezvoltarile ulterioare:

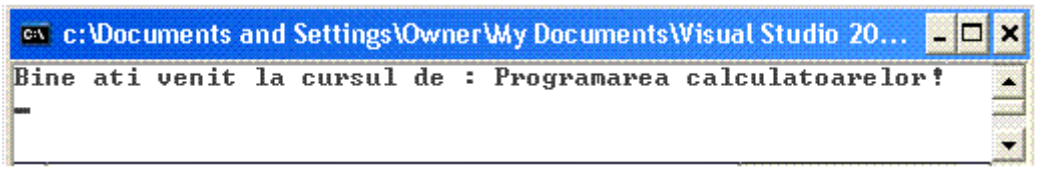
- portabilitate maxima;
- structurare;
- posibilitatea efectuarii operatiilor la nivelul masinii cu pastrarea caracteristicilor unui limbaj evoluat.

Este un limbaj de o importanta cruciala in lumea programarii, drept pentru care exista o serie de variante standardizate. Cel mai important standard ce ofera o variana standardizata a limbajului C este standardul ANSI

## Primul program ANSI C

```
// Primul program scris in C++ Visual Studio 2005 de tipul:ANSI C

#include "stdafx.h"
#include < stdio.h >
int main() {
char c[1];
printf("Bine ati venit la cursul de : Programarea calculatoarelor!");
gets(c);
return 0;
}
```

Acest program afiseaza mesajul: "Bine ati venit la cursul de : Programarea calculatoarelor!".




De obicei un program nu "incepe de la 0", in sensul ca un programator trebuie sa se concentreze asupra problemei de rezolvat si nu asupra lucrurilor de rutina de genul cum sa afisez ceva pe ecran. Exista o serie de "programe" numite functii grupate in biblioteci si care rezolva problemele des intalnite. Astfel prima linie indica faptul ca se folosesc functii de intrare / iesire, iar descrierea modului de utilizare (numele, tipul argumentelor, tipul valorii returnate etc) a acestora se afla în fisierul cu numele stdio.h . Programul scris de programator va fi gazduit tot intr-o functie. Fiind cea mai importanta functie, ea se numeste functia main si va contine instructiunile programului. In cazul programului de sus, cea mai importanta instructiune este un apel al functiei printf care afiseaza un mesaj la terminal. Mesajul este dat între ghilimele si se termina cu un caracter special new-line (\n). Daca programul ar contine numai aceasta linie, s-ar afisa mesajul pe consola standard (ecranul) dupa care fereastra in care se afiseaza acest mesaj dispare si practic nu vedem nimic. Se mai introduce si apelul functiei **gets()** care asteapta introducerea unui text de la dispozitivul satandard de intrare adica tastatura. Textul introdus va fi pastrat in variabila c pe care am definit-o in prima instructiune ca fiind de tip char (un sir de caractere de lungime 1). Functia **gets()** asteapta un text care sa se termine cu **Enter**. Cum in cazul de fata nu astept decat apasarea unei taste, este suficient sa apas tasta **Enter**. Din acest motiv am definit constanta c, de tip caracter de lungime 1. Instructiunea return preda controlul sistemului de operare la terminarea programului si comunica acestuia codul 0 pentru terminare. Prin conventie, aceasta valoare semnifica terminarea normala a programului – adica nu au aparut erori în prelucrarea datelor. Corpul functiei main apare între acolade. Orice program C trebuie sa aiba o functie main.

Functia gets() asteapta introducerea unui text, care in prealabil trebuie definit. Avand in vedere ca in programul de sus nu ne intereseaza textul introdus, am putea folosi o functie care asteapta introducerea unui caracter indiferent care. Functia se numeste getch() si se gaseste in biblioteca "conio.h"

```
#include "stdafx.h"
#include < stdio.h >
#include < conio.h >
int main() {
printf("\n\tS-a utilizat functia getch() pentru a astepta apasarea unui
caracter");
printf("\n\n\n\t\tApasati orce tasta pentru iesire!\a");
getch();
return 0;
}
```

Dupa cum se vede, s-au folosit caracterele speciale : \n;\t;\a; Caracterul special \n determina cursorul sa avanseze pe un rand nou, \t determina cursorul sa avanseze un tab iar \a emite un sunet scurt.

## Programare in C++

Programele C care respecta standardul ANSI C pot fi rulate pe orice mediu de programare C si sub orice sistem de operare.

Din pacate nucleul C standard ANSI C nu este acoperitor pentru diferite implementari mai noi de tip Object-Oriented Programming (OOP) si "Visual".

C++ include noi tehnici procedurale de programare. Daca programarea clasica in C este o programare structurata modulara , programarea C++ include si programarea orientata obiect OOP . Obiectele sunt noi tipuri ce integreaza atat datele cat si metodele asociate crearii, prelucrarii si distrugerii acestor date. Un obiect este definit de o clasa. Clasa reprezinta structura care defineste caracteristicile abstracte ale unui obiect.

O clasa contine functii si date numite functii membru respectiv date membru. Functiile membru se mai numesc si metode. Lansarea unei functii membru se mai numeste si invocarea unei functii membru. Un obiect se obtine prin instantierea unei clase. Prin instantierea unei clase, se obtine deci un obiect sau o instanta. Clasa este un concept de baza al programarii orientate obiect.

Vom folosi in continuare mediului de programare: Visual Studio 2005, fiind un mediu OOP Visual.

Va trebui deci sa analizam si extensiile limbajului C implementate in Visual Studio 2005, mediu ce ne va permite sa utilizam facilitatile OOP si Visual.

Sa luam de exemplu aplicatia de mai jos:

```
// Primul program scris in C++ Visual Studio 2005

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

int main(void)
{
	cout <<" Primul program scris in C++ Visual Studio 2005\n\n
Felicitari!!";
	cin.get();
	return 0;
}
```

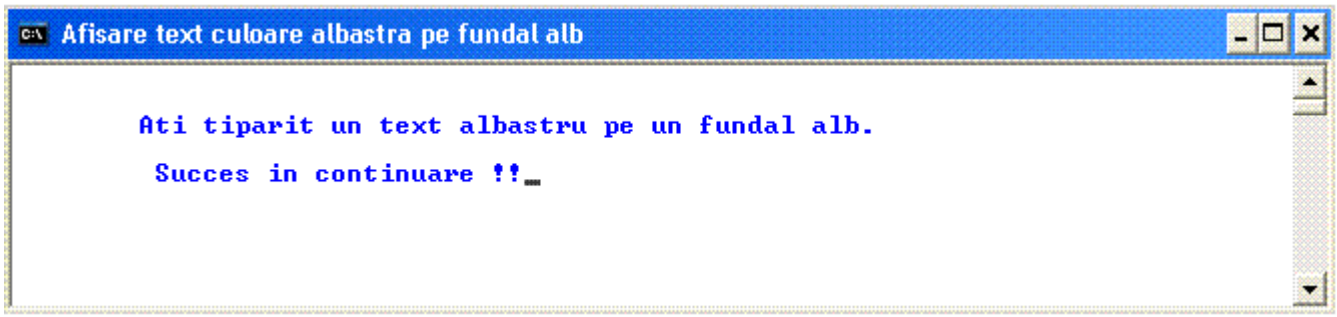
Programul ruleaza in fereastra "Command" pe un fundal negru cu caractere albe. Exista posibilitatea schimbarii atributelor ferestrei "Command" din program. Putem schimba de exemplu culorile, titlul ferestrei, etc. Urmatoarea aplicatie reia aplicatia de sus dar cu cateva modificari de atribute ale ferestrei "Command".

```
// Program scris in C++ Visual Studio 2005
// Se afiseaza un text si se modifica diverse atribute ale ferestrei "Command".
#include "stdafx.h"
#include < iostream >
using namespace std;

int main(void)
{
	system("TITLE Afisare text culoare albastra pe fundal alb ");// Titlul
ferestrei consola
	system("COLOR F9"); // Fundal alb caractere albastre
	cout <<" \n\n\tAti tiparit un text albastru pe un fundal alb. \n\n\t
Succes in continuare !!";
	cin.get();
	return 0;
}
```

In instructiunea: **system("color F9");** , prima cifra reprezinta culoarea fundalului iar cifra a doua reprezinta culoarea textului dupa cum urmeaza:

| | |
|---|---|
| • 0 = Black<br>• 1 = Blue<br>• 2 = Green<br>• 3 = Aqua<br>• 4 = Red<br>• 5 = Purple<br>• 6 = Yellow<br>• 7 = White | • 8 = Gray<br>• 9 = Light Blue<br>• A = Light Green<br>• B = Light Aqua<br>• C = Light Red<br>• D = Light Purple<br>• E = Light Yellow<br>• F = Bright White |

Dupa rularea aplicatiei obtinem:




## Structura unui program in C++

Sa analizam structura programului afisat mai jos:

```
// Primul program scris in C++ Visual Studio 2005 de tipul:CLR (Common Language
Runtime) console application

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

int main(void)
{
        cout <<" Primul program CLR console application\n\n Felicitari!!";
        cin.get();
        return 0;
}
```

**Comentarii**

Sunt precedate de // si pot aparea oriunde in program. Nu sunt executate de calculator, ele fiind destinate celor care scriu sau citesc programele.

Comentariile sunt plasate in vederea explicitarii programelor si pentru intelegerea mai usoara a programelor de catre alti programatori sau chiar de autorul programului. In cazul de fata linia de comentariu :

// Primul program scris in C++ Visual Studio 2005 de tipul:CLR (Common Language Runtime) console application

da informatii despre mediul de programare in care a fost scris programul.

**Directiva #include**

C++ contine un numar de fisiere de biblioteca standard unde sunt incluse functii si obiecte utilizate frecvent. Aceste fisiere sunt sunt grupate in biblioteci
#include < iostream > include biblioteca **iostream** ce contine functii si obiecte pe care le vom folosi in functia **main** , de exemplu **cout >>**

In cazul in care avem mai multe biblioteci de inclus sau daca pentru orce program trebuie neaparat sa includem niste biblioteci, in loc de numele biblotecii se poate indica un fisier ce contine toate numele bibloteciloor. Fisierul are de obicei extensia .h .In cazul primului program scris anterior, #include "stdafx.h" include bibliotecile si directivele de compilare scrise in fisierul stdafx.h

```
// stdafx.h : include file for standard system include files,
// or project specific include files that are used frequently, but
// are changed infrequently
//

#pragma once


#define WIN32_LEAN_AND_MEAN    // Exclude rarely-used stuff from Windows headers
#include < stdio.h>
#include < tchar.h>


// TODO: reference additional headers your program requires here
```

#include < iostream > ar putea fi inclusa in fisierul stdafx.h, in acest caz ne mai fiind nevoie s-o includem in programul principal.

**Functia main**

Functia este un grup de instructiuni scrise in vederea realizarii unei sarcini. Functia este referita prin nume. In cazul primului program scris mai sus, **main** este numele functiei principale. Un program poate avea mai multe functii si de aceeea functia care se lanseaza prima in cadrul executiei programului trebuie sa poarte denumirea **main**

**Spatiul de nume**

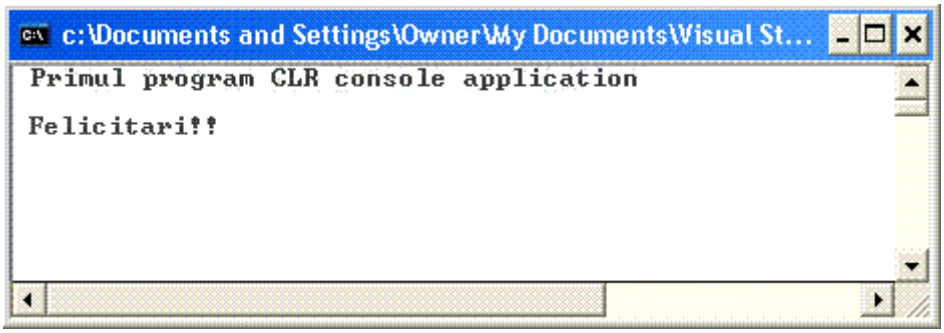
Instructiunea **using namespace std;** este un spatiu de nume. C++ foloseste spatii de nume (namespaces) pentru a organiza clase cu functionalitati inrudite, folosite in cadrul programelor. Si bibliotecile folosesc spatii de nume , astfel clasele din biblioteca sunt referite prin intermediul spatiului de nume.

**Corpul functiei**

Toate instructiunile functiei sunti incluse in corpul functiei. Un corp de functie incepe cu { si se termina cu } De obicei fiecare instructiune se termina cu ; In cazul primului program scris anterior functia **main** contine cele trei instructiuni afisate mai jos. Instructiunile se executa secventiial de sus in jos.

```
cout <<" Primul program CLR console application\n\n Felicitari!!";
cin.get();
return 0;
```

Instructiunea **cout <<** afiseaza pe consola standard ,Textul : "Primul program" pe randul 1 iar pe randul 3 textul: Felicitari. \n comanda trecerea pe randul urmator. La inceput cursorul se afla pe randul 1 unde va scrie textul : "Primul program" . Dupa secventa \n\n cursorul se muta pe randul 3 unde va scrie textul : "Felicitari !!"




Declaratia **cin.get();** reprezinta invocarea metodei get a obiectului cin. Pentru precizarea metodei get au fost deci necesare precizarea tuturor componentelor adica:

*obiect . metoda*

In cazul ca spatiul de nume nu este prezentat in antet,atunci forma generala este:

*spatiu de nume :: obiect . metoda*

Metoda **get();** asteapta apasarea tastei Enter. Invocarea acestei metode este necesara deoarece dupa scrierea textului pe consola, aceasta ar disparea foarte repede fara sa ajungem sa citim textul afisat. Textul afisat ar trebui completat cu mesajul: "Tastati Enter"

Daca nu am fi folosit spatiul de nume **using namespace std;**, instructiunile din corpul functiei **main** ar fi trebuit scrise:

```
        std::cout <<" Primul program CLR console application \n\n Felicitari!!";
        std::cin.get();
        return 0;
```

Practic primul programul, dupa modificarea stadfix.h , poate sa arate astfel:

```
// Primul program scris in C++ Visual Studio 2005 de tipul:CLR console
application

#include "stdafx.h"
#include < iostream >

int main(void)
{
        std::cout <<" Primul program  CLR console application\n\n Felicitari!!\n\
n Tastati Enter";
        std::cin.get();
        return 0;
}
```

Invocarea metodei **get();** are rolul de a astepta apasarea unei taste altfel fereastra consola in care se afiseaza textul dispare fara a avea timp sa citim textul afisat.

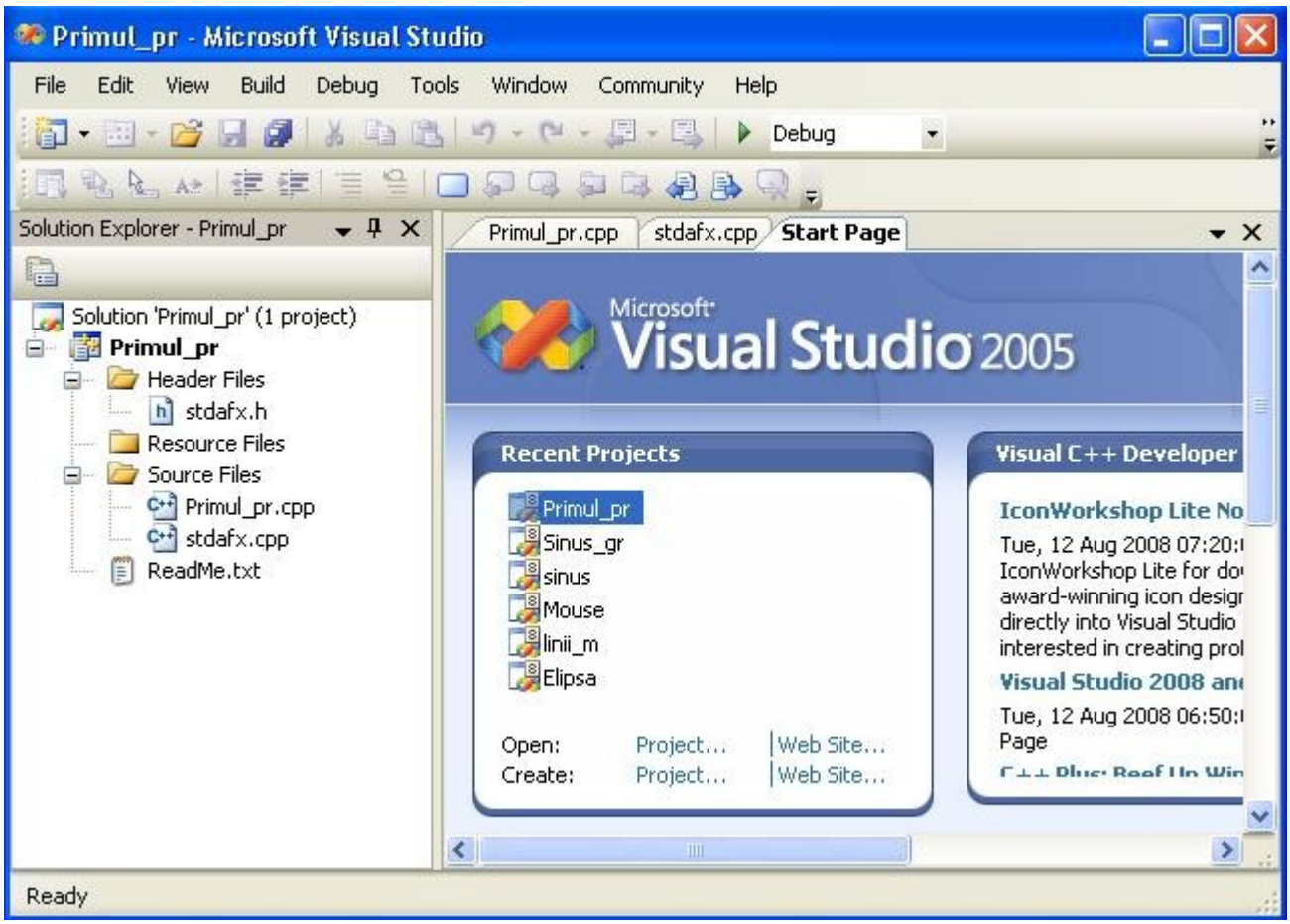
O alta metoda care ne permite sa citim textul afisat, este reprezentata de folosirea functiei sleep() care determina "inghetarea" programului pentru un timp determinat, astfel avem timp sa citim cosola.

```
// Programul  utilizeaza functia sleep din biblioteca

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

int main(void)
{
        cout << " Va urez o zi buna!!";
        Sleep(1000);
        cout << " \n\n\t Salut!!";
        Sleep(500);
        return 0;
}
```

Un alt avantaj il constituie asiatarea permanenta in momentul scrierii liniilor de program care contin denumirea metodelor si claselor din acest spatiu de nume.

# Utilizare Visual Studio 2005 - CLR(Common Language Runtime) Console Application

**IDE**( Integrated Developement Environment) utilizat :Visual studio 2005



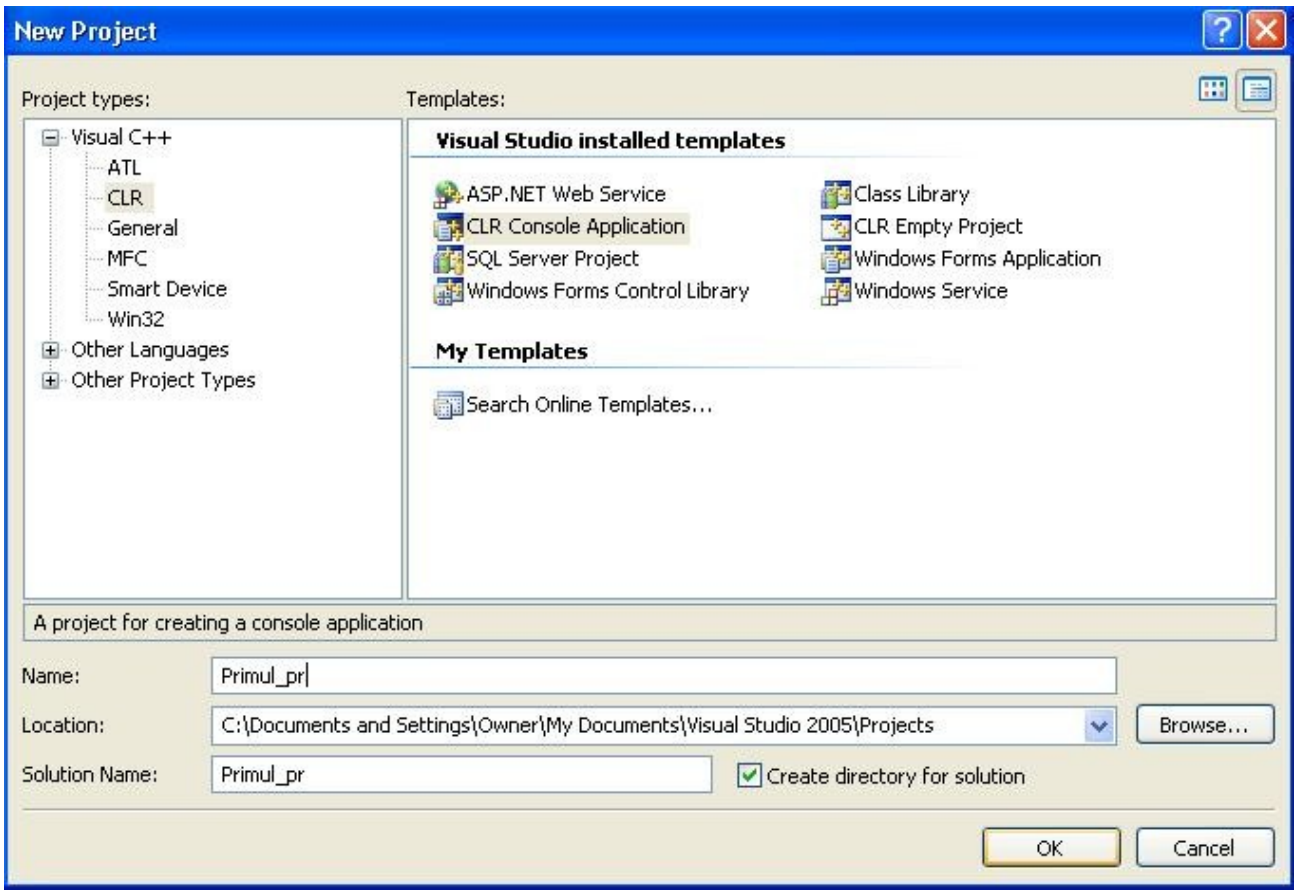

Are o interfata grafica GUI (Graphics User Interface) si contine:

- editor de texte
- preprocesor
- compilator
- link-editor

`IDE-uri similare:Borland C++ Builder, IBM VisualAge, etc.`

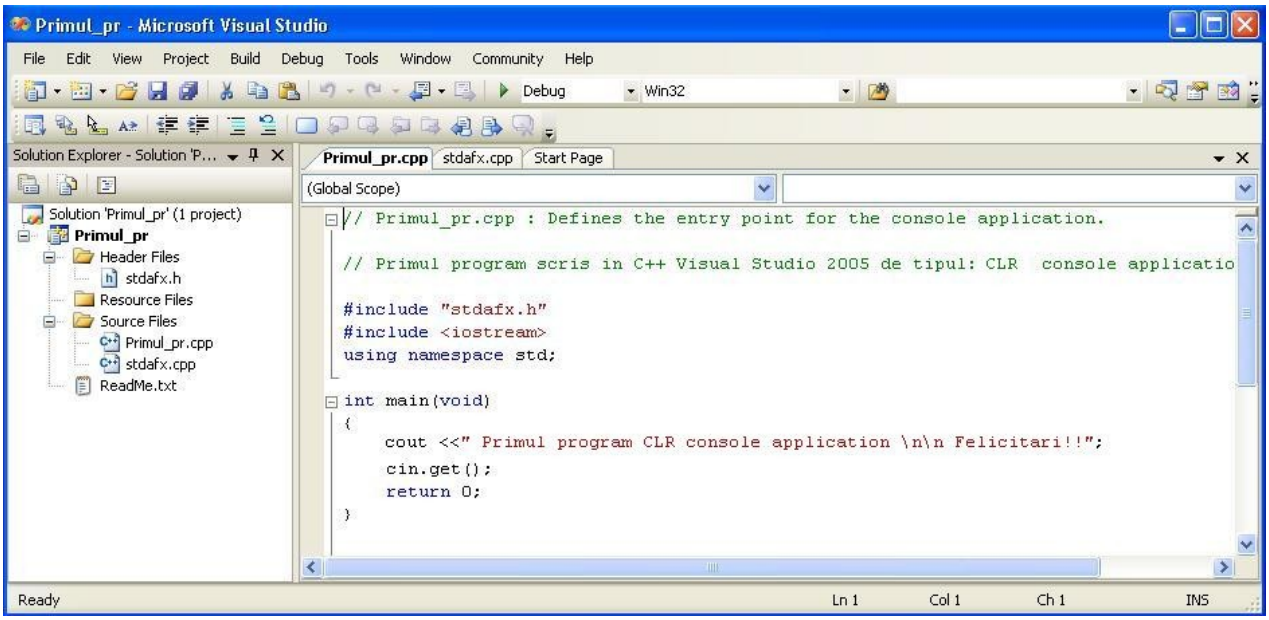
## Crearea proiectelor Visual Studio 2005 de tipul CLR Console Application

- Lansare Visual Studio 2005
- File-New Project
- Se alege - Visual C++ CLR (Common Language Runtime)-CLR Console Application si se completeaza numele proiectului, in cazul de fata Primul_pr




## Scrierea codului sursa

- Se alege fisierul Primul_pr.cpp
- Se scrie codul sursa al programului primul_pr

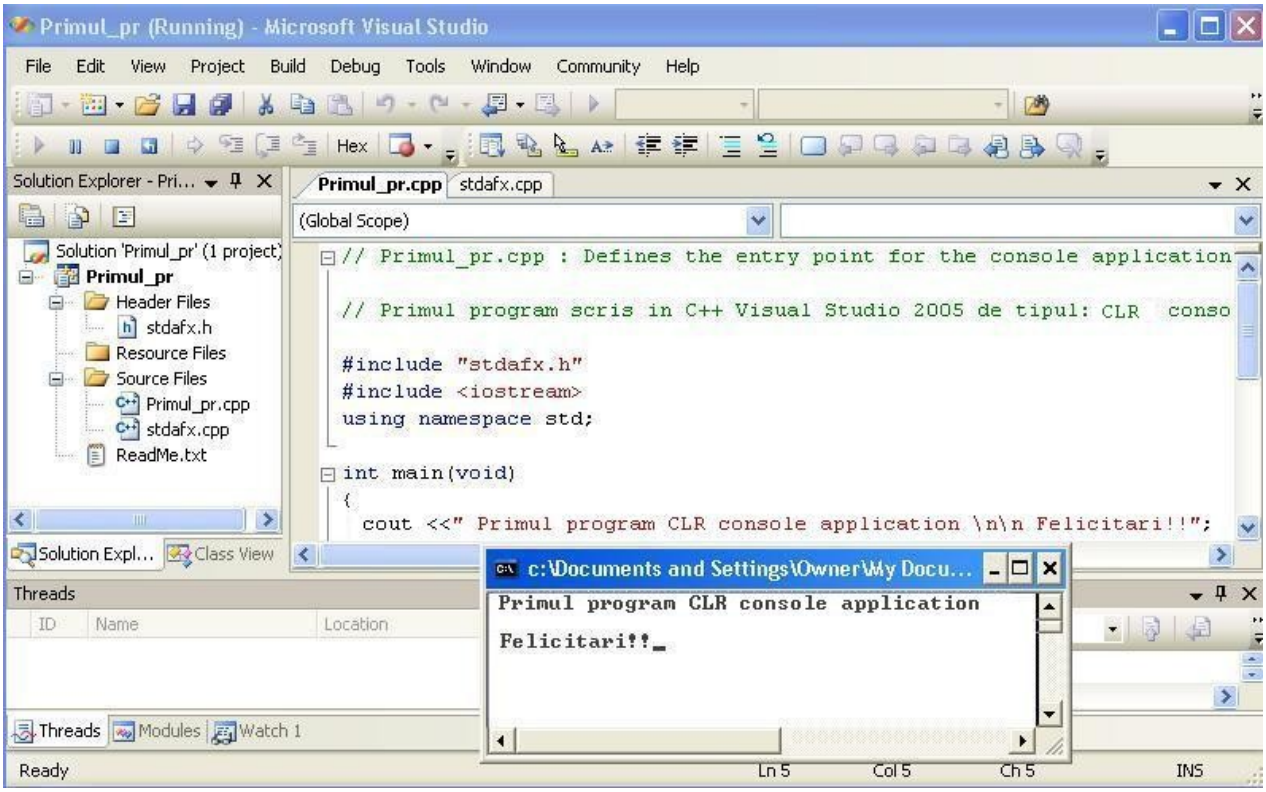


## Constructia proiectului

- Build- Build solution

## Rularea codului

- Debug-Start Debugging




## Spatiul de nume System::

Spatiul de nume **System::** contine clase fundamentale, de baza si clasele frecvent folosite, interfete, metode de manipulare a evenimentelor. Folosind acest spatiu, avem posibilitatea de a utiliza diverse metode pentru a defini si converti diferite tipuri de date necesare in programarea de tip "Visual" pe care o vom utiliza in cadrul modulului Windows Forms Application.

Spatiul de nume **System::** contine si functii pentru consola, putand astfel sa dezvoltam atat aplicatii de tipul CLR Console Application cat si aplicatii de tipul Windows Forms Application. Functiile pentru consola se gasesc in clasa **Console::** Functiile definite in interiorul unei clase poarta numele de metode ale clasei. Apelarea unei metode (functi) definite in cadrul unei clase se face prin precizarea tuturor componentelor si anume:

*spatiul de nume:: clasa:: metoda ( )*

Pentru a apela functia **ReadLine** de exemplu vom utiliza: **System::Console::ReadLine()**

Declaratia de mai sus poate fi citita si astfel: "Se invoca metoda *ReadLine* a clasei *Console* aflata in spatiul de nume *System* .

Pentru a afisa de exemplu textul :" Succes la examene !! " , vom invoca metoda *WriteLine* a clasei *Console* aflata in spatiul de nume *System* .

Mai jos se prezinta un program de tipul CLR Console Application, program ce invoca metode din clasa *Console* din spatiul de nume *System* .

```
// sp_system.cpp : main project file.
// Primul program ce utilizeaza functii consola din spatiul de nume System::

#include "stdafx.h"

using namespace System;

int main(void)
{
        Console::WriteLine("Program ce utilizeaza spatiul de nume System::");
        Console::ReadLine();
    return 0;
}
```

# Utilizare Visual Studio 2005 - Windows Forms Application

Dupa cum am amintit, mediul de programare "Visual Studio 2005" este un mediu visual de programare orientata obiect(OOP), permitand realizarea de aplicatii OOP folosind modul "Visual". Obiectele dorite sunt plasate pe o planseta de design (form) dupa care sunt completate si adaugate diverse secvente de cod pentru a stabili comportarea dorita pentru obiectul plasat.

## Notiuni utilizate in OOP

- **Obiecte** - sunt noi tipuri ce integreaza atat datele cat si metodele (functiile) asociate crearii, prelucrarii si distrugerii acestor date.
- **Clasa** - reprezinta structura care defineste caracteristicile abstracte ale unui obiect.
- **Metode** - functiile membru definite in cadrul unei clase
- **Prporietati** - membrii unei clase care permit accesul controlat la datele membru ale unei clase
- **Evenimente** - membrii unei clase care permit clasei sau obiectelor clasei sa faca notificari, adica sa anunte celelalte obiecte asupra unor schimbari petrecute la nivelul starii lor.

Apelarea unei metode (functi) sau setarea unei proprietati, ale unui obiect se face prin precizarea tuturor componentelor si anume:

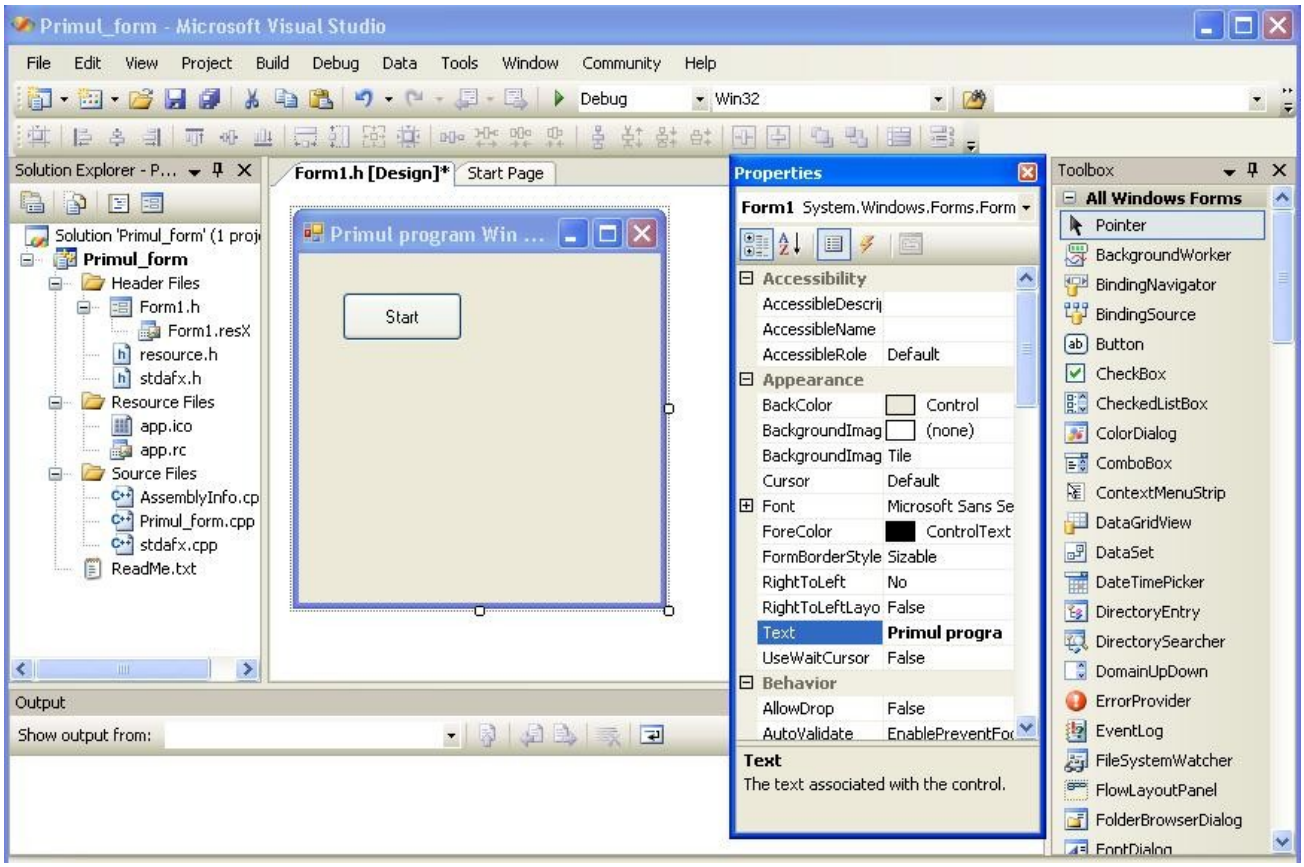
*nume_aplicatie::nume_form::nume_obiect-> metoda sau proprietate*

Componentele **nume_aplicatie::nume_form::** pot fi inlocuite cu **this** deci precizarea unei metode sau a unei proprietati, devine:

*this->nume_obiect-> metoda sau proprietate*

## Realizarea proiectelor Visual Studio 2005 de tipul CLR-Windows Form Application

- Lansare Visual Studio 2005
- File-New Project
- Se alege - Visual C++ CLR (Common Language Runtime)-Windows Form Application si se completeaza numele proiectului, in cazul de fata < b> Primul_form

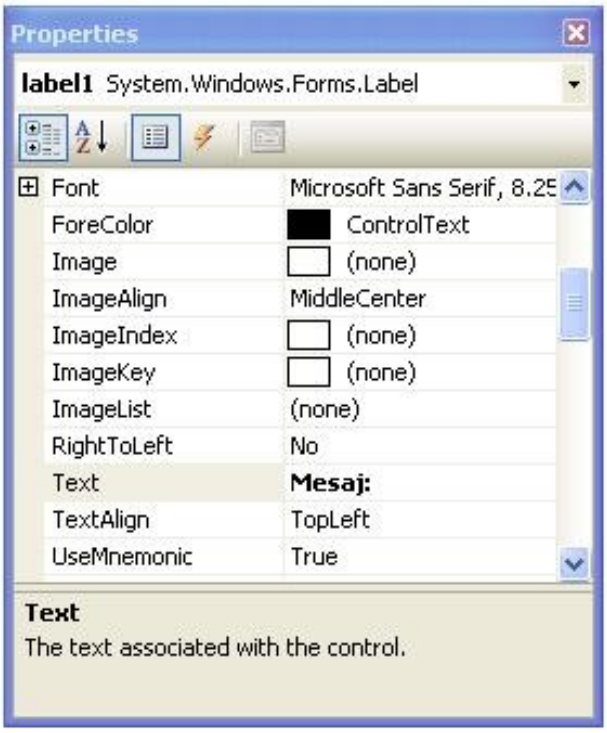



### Plasare buton Start

Din ToolBox se alege All windows Forms--Button si se plaseaza pe Form-ul deschis.
Se selecteaza butonul plasat pe Form cu click dreapta si se alege din meniul deschis optiunea "Properties"
Se selecteaza propietatea "Text" si i se va atribui valoarea "Start". In acest moment pe buton va scrie "Start"
Se selecteaza de aceasta data Formul cu click dreapta, se alege din meniul deschis optiunea "Properties"
Se selecteaza propietatea "Text" si i se va atribui valoarea "Primul program Win Form App". In acest moment titlul ferestrei principale va fi:"Primul program Win Form App"

### Plasare label

Din ToolBox se alege All windows Forms--Label si se plaseaza pe Form-ul deschis.
Se selecteaza label-ul plasat pe Form cu click dreapta si se alege din meniul deschis optiunea "Properties"
Se selecteaza propietatea "Text" si i se va atribui valoarea "Mesaj". In acest moment pe buton va scrie "Mesaj"


**Scrierea codului**

Daca selectam butonl **start** cu click dreapta si se alegem din meniul deschis optiunea "Properties" gasim la proprietatea Name setat numele **button1**, atribuit automat la plasarea butonului. La fel pentru labelul pe care am setat textul mesaj, vom gasi numele **label1**

Ne propunem sa scriem un program care sa afiseze textul:"Primul program Windows Forms Application" la apasarea butonului "Start".
Se selecteaza butonul "Start" cu click dreapta si se alege din meniul deschis optiunea "Properties".
Se apasa butonul "Events".
Se alege Action--Dublu click pe optiunea Click.In acest moment evenimentului "click" al butonului "Start" i se genereaza un schelet de procedura care va trata evenimentul click al butonului button1 numita: **button1_click** pe care trebuie sa-l completam cu liniile de instructiune necesare pentru a trata evenimentul click.In cazull de fata: cu instructiunea:
**this->label1->Text="Primul program Windows Forms Application";**
adica pe formul curent (this) obiectului label1 sa atribuim proprietatii Text valoarea : "Primul program Windows Forms Application"
Procedura care trateaza evenimentul click va fi deci:

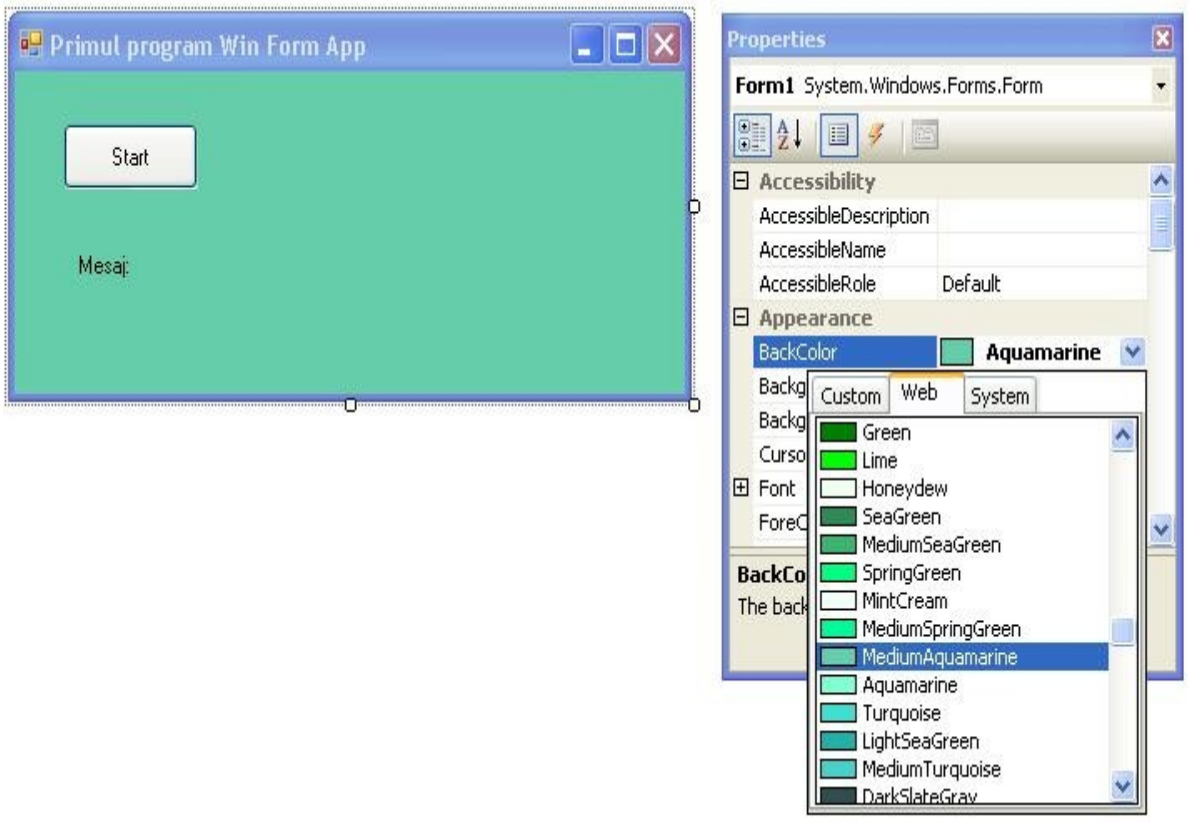
```
private: System::Void button1_Click(System::Object^  sender, System::EventArgs^  e) {

             this->label1->Text="Primul program Windows Forms Application";
         }
```

 **Rulare aplicatie**

In acest moment prin apasarea butonului "Start debug" se lanseaza aplicatia si putem apasa butonul "Start" din aplicatie. Dupa aceste operatii, ecranul aplicatiei arata astfel:


 **Modificare atribute**

Revenim la Form Design si modificam diferite proprietati (atribute) ale obiectelor amplasate pe form .
Sa modificam de exemplu BackColor pentru fereastra principala:



Putem modifica dupa preferinta proprietatile tuturor obiectelor amplasate pe form. Putem modifica de exemplu Fonturile si culoarea mesajului.

- **Varianta finala**

Dupa ce am efectuat toate modificarile si am testat aplicatia, in Folder-ul project gasim Folderul:Primul_form apoi in Folderul debug: gasim executabilul "primul_form.exe" pe care il putem lansa in executie.