Cuprins C 1 - VB

Noțiuni introductive în limbajele de programare

- **Limbaj de programare;**
- **Limbaj mașină;**
- **Compilator;**
- **Interpretor.**

1. Termeni specifici limbajelor de programare

Limbaj de programare

Un limbaj formal utilizat la scrierea programelor pentru calculatoare. Definiția unui limbaj particular se face prin **sintaxă** (modul de combinare a simbolurilor de limbaj - este descrisă prin **gramatica** limbajului) şi **semantică** (semnificația construcțiilor de limbaj). O denumire alternativă pentru limbajul de programare este aceea de **limbaj sursă** sau, mai pe scurt, sursă.

Limbaj mașină

Reprezentarea unui program pentru calculator care poare fi citită şi interpretată de calculator. Este format din secvențe de instrucțiuni mașină. Colecția tuturor instrucțiunilor unui calculator particular se numește "set de instrucțiuni". O denumire alternativă a limbajului mașină este aceea de **cod obiect**.

Program compilator

Un program care convertește un alt program scris într-un limbaj sursă în limbaj maşină.

Program interpretor

Un program care execută alte programe. Un program interpretat se execută mai încet decît unul compilat, dar timpul de interpretare este mai scurt decît cel total de compilare şi de execuție. În faza de prototip şi de testare a codului, din acest motiv, ciclul de editare-interpretare-depanare poate fi mai scurt decît cel de editare-compilare-execuție-depanare. Interpretarea codului este mai înceată decît execuția codului compilat deoarece interpretorul trebuie să analizeze fiecare instrucțiune din program de fiecare dată cînd se execută după care va realiza acțiunea cerută, în timp ce în cazul compilării acțiunea se face o singură dată.

Istoria BASIC-ului

- **Dezvoltat de J. G. Kemeny și Thomas E. Kurz la Colegiul Dartmouth în 1963;**
- **Scris pentru a-i învăța studenți să programeze ușor și repede;**
- **De la BASIC la Visual Basic;**
- **Avantaje și dezavantaje VB.**
- **Unde se folosește.**

2. Istoria BASIC-ului

J. G. Kemeny, T. E. Kurz

La început a fost doar BASIC ...

Inventat de John Kemeny şi Thomas Kurz de la Colegiul Darmouth din California. Scris pentru a-i învăţa uşor pe studenţi fundamentele informaticii şi a programării calculatoarelor. Primul BASIC a rulat pe un IBM 704 la data de 1 mai 1964.

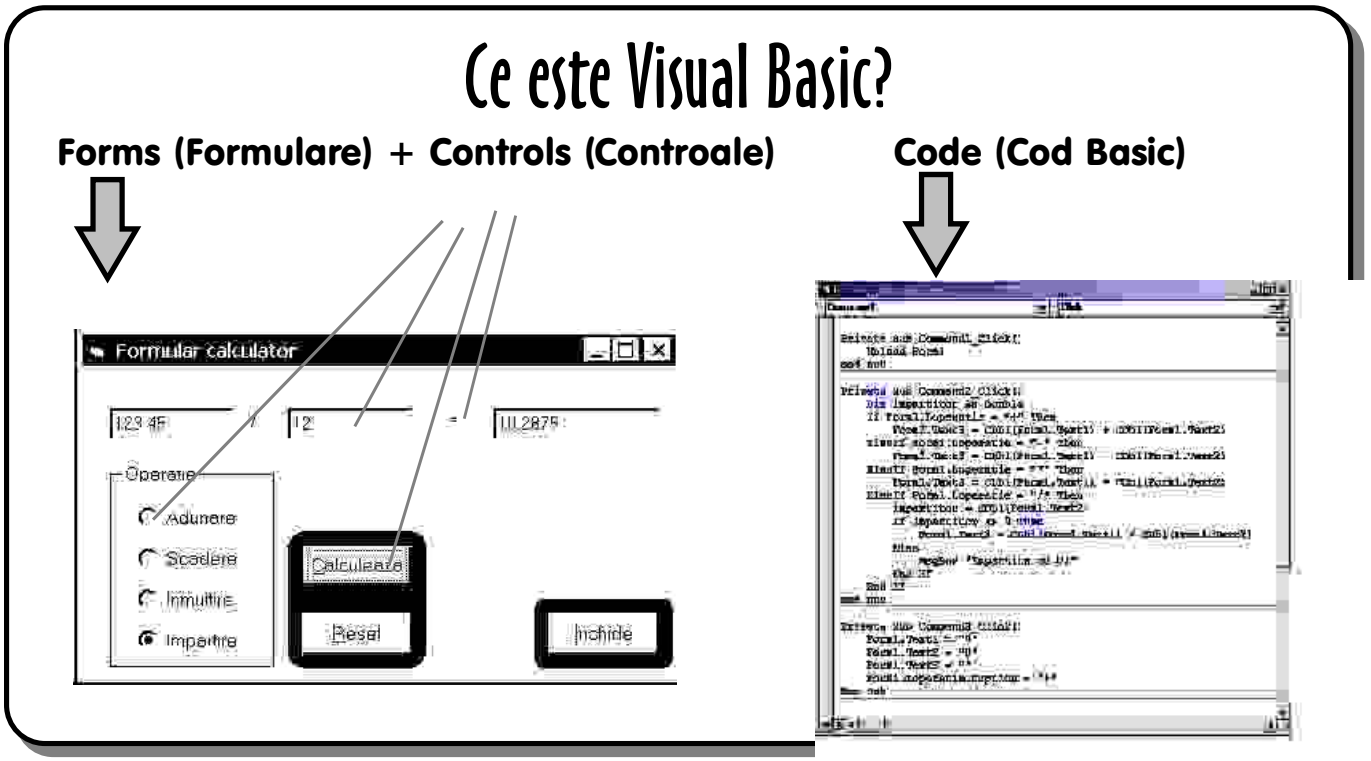
În februarie 1975, Paul Allen şi Bill Gates (fondatorii Microsoft) au scris primul BASIC pe Altair (avea 4K şi permitea scrierea a 50 de linii de cod). Microsoft a continuat să dezvolte limbajul ...

Apariţia şi dezvoltarea lui VB

În 1991, odată cu apariţia lui Windows 3.0, Microsoft lansează produsul MS Visual Basic inspirat din BASIC, dar cu o interfaţă grafică nouă care permitea conceperea unei ferestre Windows în cîteva minute folosind metoda "Drag and Drop".

Pînă la versiunea 5, codul generat de Visual Basic era P-cod (un cod intermediar între BASIC şi limbajul maşină care se interpreta în vederea execuţiei). Din aces motiv VB avea reputaţia unui limbaj încet. Începînd cu VB5 Microsoft a încorporat un compilator în VB pentru a putea crea direct programe executabile în limbaj maşină (dar acestea au nevoie de Visual Basic runtime pentru a putea fi rulate independent de mediul VB). Viteza de execuţie a programelor a crescut din acest motiv de peste 10 ori.

Ultima versiune este VB.NET. VB6 era un limbaj de programare bazat pe obiecte (Object Based), VB.net este un limbaj de programare obiectual (Object Oriented), din acest motiv, şi puţin mai complicat.



3. Ce este Visual Basic

Limbajul Visual Basic este un instrument pent
pe calculator) Windows.

Visual se referă la elementele vizuale ale
reprezintă ferestre (**windows**) prin care utiliza
suprafaţa formularelor se află controale (**cont
(**command button**s), cutiile de text (**text box
utilizatorul poate acţiona pentru a face aplica
pentru a vedea rezultatele acţiunilor lui. Term
se foloseşte pentru descrierea aspectului vi
controalele folosite la scrierea aplicaţiei sînt c
programatorul le manipulează în spaţiul vizu
spaţiul codului prin instrucţiuni VB.

BASIC este acronimul lui Beginer All-pu
Limbajul a progresat mult din anii '60, dar a p
iniţiale şi convenţiile folosite în acele vremuri

Programarea în VB implică două tipuri de ac
şi cea de scriere a codului pentru manipulare
vederea obţinerii rezultatelor. Utilizatorul c
butoanelor, selectarea unor opţiuni sau con
aplicaţia va afişa rezultatele tot prin controal

Pro şi contra VB

Avantaje	Dezavanataje
• **Dezvoltare rapidă** • **Simplu de învăţat** • **Număr mare de controale** • **Bazat pe obiecte** • **Poate crea componente**	• **Performanţe inferioare** • **Cod nestructurat** • **Nu este orientat pe obiecte**

4. Pro şi contra VB

Avantaje Punctul forte al VB este abilitatea dezvoltării rapide a aplicaţiilor. Datorită controalelor predefinite şi cu ajutorul formularelor şi ca urmare a consistenţei secvenţelor de programat, se definesc şabloane pe baza cărora se pot construi, testa şi genera kituri de aplicaţii.

Codul se poate înţelege uşor, motiv pentru care există o bază extinsă de resurse VB pentru dezvoltarea, întreţinerea şi generarea kiturilor de aplicaţii.

Există o mulţime standardizată de controale scrise de firme terţe (3rd party controls) pe care un dezvoltator VB le poate încorpora în aplicaţie.

Limbajul este bazat pe obiecte şi permite crearea de componente pentru alte aplicaţii. Ca urmare apare eficientizarea dezvoltării datorită reutilizării codului componentelor.

Dezavantaje Deşi VB poate construi o aplicaţie foarte repede, aceasta nu este foarte eficientă în termenii execuţiei codului, pentru aceasta limbajele C sau C++ sînt recunoscute ca fiind cele mai bune.

Codul VB poate să fie slab structurat ca urmare a lipsei unor constructii de cod moderne şi datorită păstrării construcţiilor de cod nestructurat de la începuturile BASIC-ului. Ca urmare, întreţinerea programelor vechi poate să fie dificilă.

VB este bazat pe obiecte (Object Based) şi nu orientat pe obiecte (Object Oriented)). Multe dintre conceptele cheie ale limbajelor orientate pe obiecte nu pot fi implementate simplu (de ex. moştenirea - proprietatea unei clase de a moşteni comportamentul unei alte clase). Tehnicile moderne de proiectare presupun existenţa şi necesită folosirea posibilităţilor specifice OO.

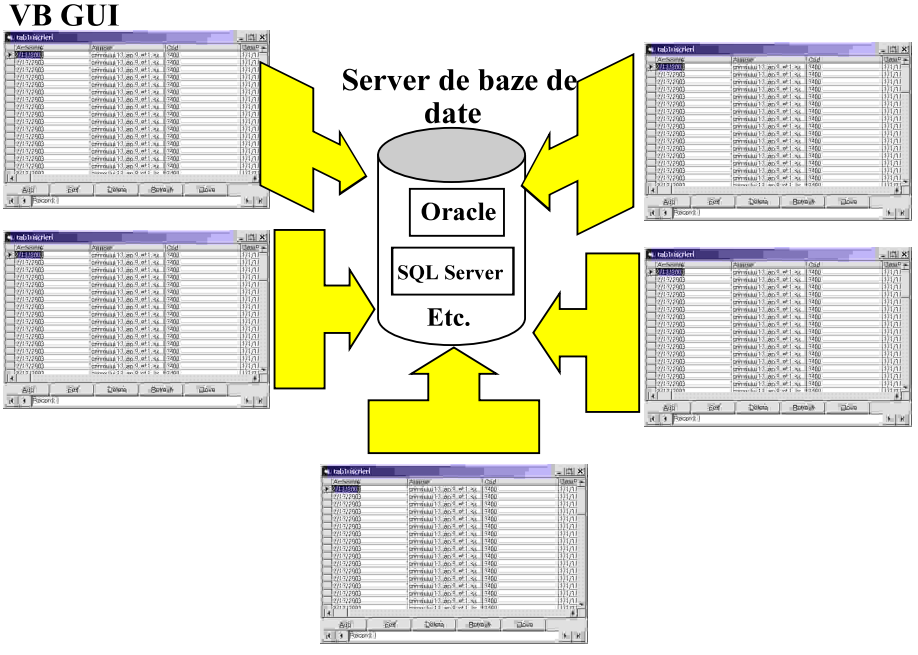
VB în aplicații de sine stătătoare

- **Aplicații de calcul:**
  - **ipotecă, asigurări, vînzări, inginerie;**
  - **înregistrarea vînzărilor, soluții mobile;**
- **Utilitare de PC:**
  - **gestiunea de fișiere;**
  - **gestiunea de conturi.**
- **Colectarea de date:**
  - **interfețe pentru bazele de date;**
  - **aplicații de tipul jurnal de conectare.**
- **Integrare cu aplicațiile:**
  - **Microsoft Office (Word, Excel, Access, PowerPoint, Outlook, ...);**
  - **CATIA, AutoCAD, SolidWorks etc.**

5. Aplicații VB de sine stătătoare

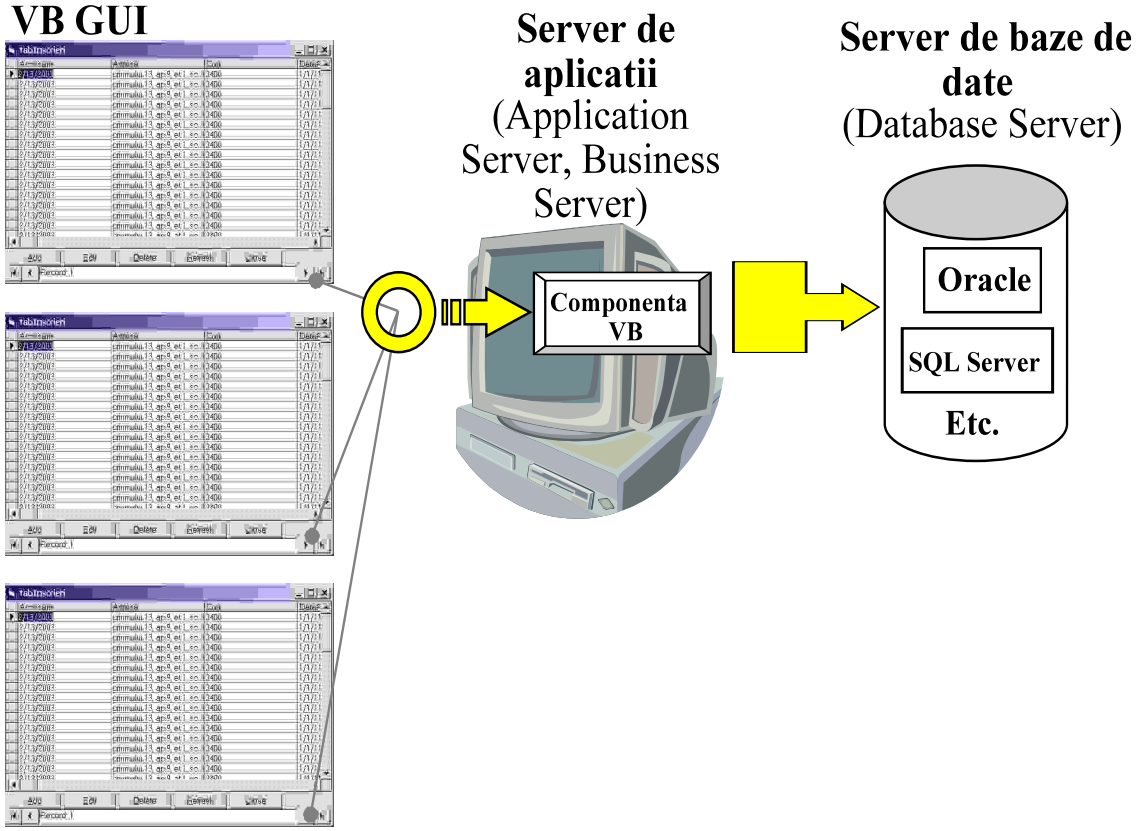
VB poate fi folosit pentru crearea de **aplicații de sine stătătoare** (aplicații care nu necesită alte aplicații pentru a funcționa).

Datorită complexității şi a gradului înalt de control pe care VB îl are asupra controalelor din formulare poate fi folosit ca o **interfață grafică** (GUI) excelentă cu alte aplicații. Cîteva sugetii de aplicații s-au prezentat mai sus.



Una dintre aplicațiile frecvente ale lui VB este GUI-ul pentru aplicații de baze da date mari. Termenul de **2-Tier** se referă la faptul că datele sau prelucrarea codului se realizează în mai

multe spaţii de prelucrare (în cazurile obişnuite, pe mai mult de un singur calculator). În figura anterioară s-a reprezentat un grup de calculatoare client, conectate într-o reţea, ce au interfeţele scrise în VB (front end) prin care se conectează direct la datele unui sistem de gestiune a bazelor de date în scopul manipulării (inserare, actualizare, ştergere) lor. Fiecare program client are o linie directă cu datele atît timp cît poate stabili o conexiune cu server-ul. Soluţiile 2-Tier au însă o problemă mare, ele nu sînt scalabile din punctul de vedere al creşterii numărului de utilizatori. Conexiunile la bazele de date sînt resurse costisitoare motiv pentru care, deseori, trebuie limitate. Ca urmare a limitărilor utilizatorii nu se mai pot conecta în orice moment sau sistemul ajunge să fie saturat de cereri şi performanţele scad dramatic. De asemenea, aplicaţiile client pot încorpora porţiuni de cod specifice afacerii programate. Dacă aceasta (logica aplicaţiei) se modifică trebuie modificate toate aplicaţiile client (care pot ajunge la cîteva sute), administrarea devenind foarte complicată în acest caz.



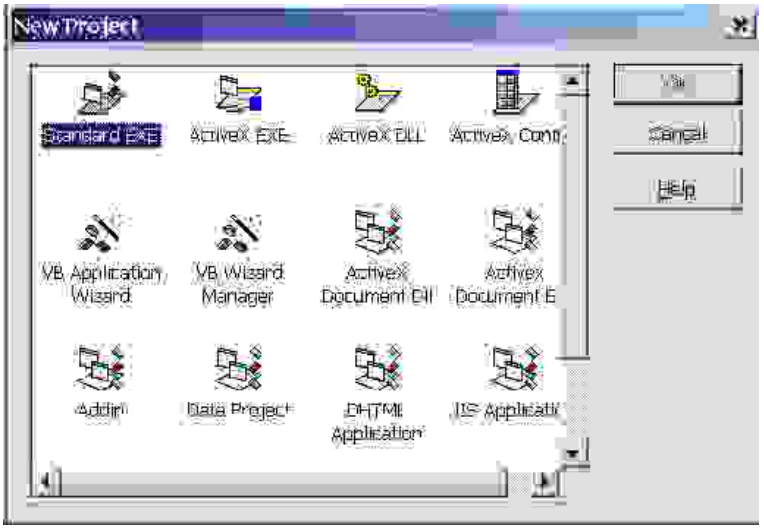
Ca urmare a problemelor de scalabilitate ale aplicaţiilor 2-Tier s-a dezvoltat modelul **3-Tier**. În acest model VB poate crea programele care ţin de GUI, adică va asigura serviciile de prezentare. Aceste programe nu accesează sistemul de gestiune a bazelor de date direct ci folosesc componente externe care realizează interacţiunea cu datele.

Componentele sînt cod pur, nu au elemente vizuale şi se folosesc pentru implementarea regulilor de logică specifice afacerii programate (business logic). Din punct de vedere tehnic componenta este stocată pe nivelul numit "Business Server" (cîte denumiri alternetive sînt Middle Tier, Second Tier), iar VB poate fi folosit pentru crearea acestor componente. În mod frecvent, aceste componente sînt stocate pe unul sau mai multe business server-e pentru a uşura întreţinerea şi distibuţia noilor versiuni.

Tipuri de proiecte VB

- **Standard EXE;**
- **ActiveX EXE;**
- **ActiveX DLL;**
- **ActiveX Control.**

- **Celelalte sînt doar variaţii minore ale primelor 4.**



6. Tipuri de proiecte VB

Un program VB este un grup de fişiere care împreună permit crearea unei singure aplicaţii. Fişierele reprezintă formulare, clase sau module individuale şi definesc ce anume va face aplicaţia. Oricare fişier poate să aparţină la mai multe proiecte astfel, secvenţele de cod pot fi refolosite în mai multe aplicaţii. La pornirea lui VB fereastra de dialog de mai sus prezintă tipurile de proiecte care pot fi dezvoltate. Există numai 4 tipuri de proiecte VB:

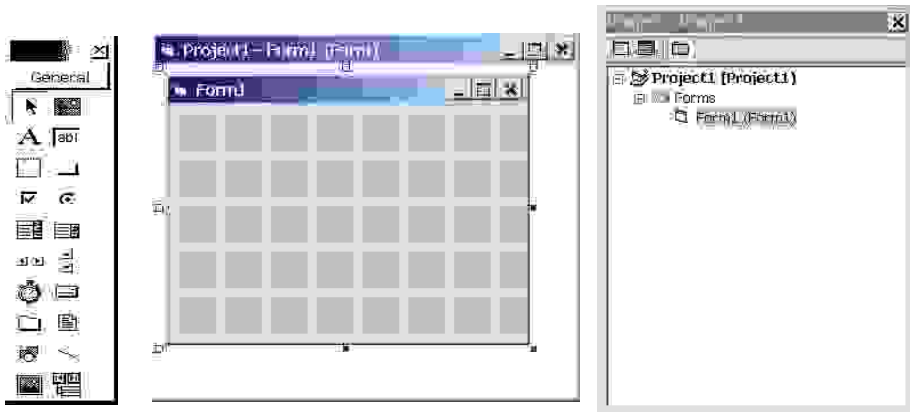
- **Standard EXE** - sînt aplicaţii care interacţionează cu utilizatorul prin controalele de pe suprafaţa formularelor. Pot avea un singur formular sau mai multe care pot folosi controale ale unor firme terţe.
- **ActiveX EXE** - sînt aplicaţii ce pun la dispoziţia altor aplicaţii obiecte sau componente şi au propriul lor spaţiu de adrese. Pot sau nu să aibă interfeţe. De exemplu, Word sau Excel pot fi rulate independnet de un anumit utilizator sau pot fi comandate de alte programe caz în care, deseori, nu vor afişa elemente vizuale.
- **ActiveX DLL** - sînt aplicaţii ce pun la dispoziţia altor aplicaţii obiecte sau componente şi NU au propriul lor spaţiu de adrese. Din acest motiv ele nu pot fi rulate independent de alte aplicaţii. Sînt biblioteci de cod care devin parte ale unor alte aplicaţii.
- **ActiveX Control** - sînt aplicaţii ce creează controale pentru alte aplicaţii. Acestea pot fi folosite în pagini de web sau în orice programe client tradiţionale.

Toate celelalte proiecte trebuie să creeze un proiect bazat pe unul dintre cele 4 discutate. Sînt proiecte preconfigurate care conţin formulare predefinite, clase, generatoare etc. şi, deseori, referinţe la biblioteci externe de cod care sînt necesare pentru îndeplinirea anumitor funcţii ale aplicaţiei. Modalitatea de creare a unui şablon de proiect va fi predată la curs.

Proiectul Standard EXE - Formular Implicit

- **Aplicaţie "Vizuală"**
- **Formular implicit - Form1**

Control Toolbox	**Form Designer**	**Project Explorer**
(ferestra controalelor)	**(fereastra de proiectare a formularului)**	**(navigatorul de proiecte)**



7. Proiectul Standard EXE

După selectarea proiectului **Standard EXE** VB va crea implicit un formular vid (fără controale şi cod) cu numele **Form1.**

Proiectul va primi numele implicit de **Proiect1** şi va fi stocat numai în RAM atît timp cît nu este salvat pe disc (*File➞ Save Project*).

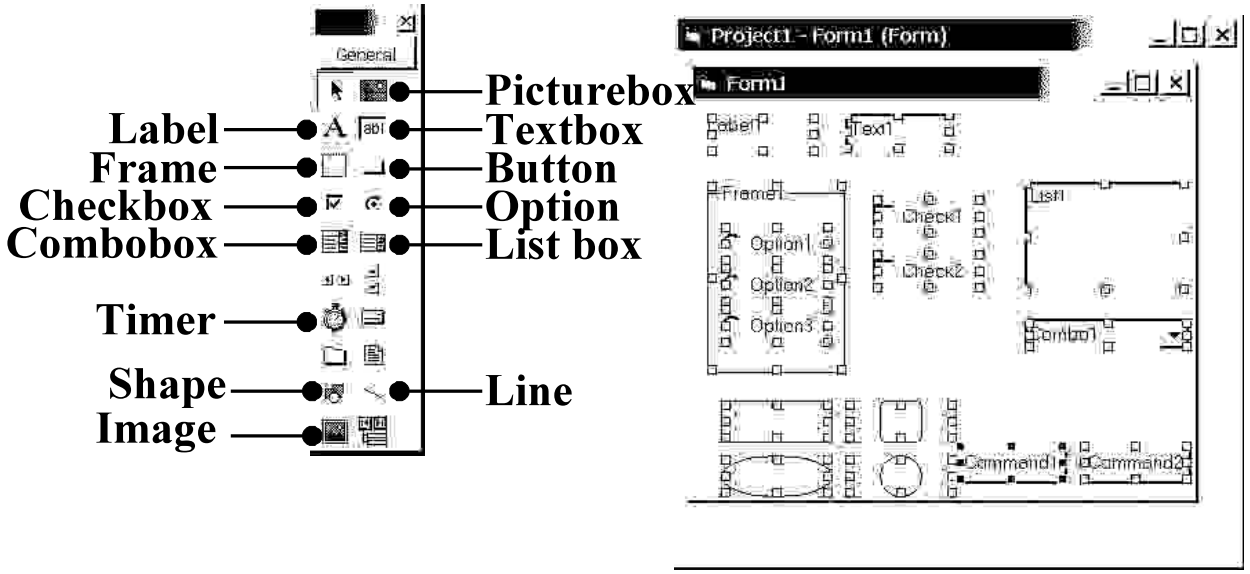
Numele **Project1** şi **Form1** sînt nerelevante şi inutile motiv pentru care trebuie schimbate la începutul unui proiect nou.

Ancorarea ferestrelor (Docking and Undocking Windows)
Toolbox, **Project Explorer** şi alte ferestre ajutătoare pot fi dez-ancorate prin tragerea lor de la marginea ferestrei mediului de proiectare spre centru. Unele dintre ele sînt ancorate între ele, aşa că dez-ancorarea uneia ar putea afecta alte ferestre. Dacă nu vă convine ancorarea ea poate fi stopată prin clic pe butonul drept de mouse, apoi deselectînd opţiunea *Dockable.*

Control Toolbox

Sursa controalelor pentru formulare:

- **se pun pe formular cu drag'n'drop, sau**
- **clic dublu de imaginea din Toolbox.**



8. Control ToolBox

Controalele sînt obiecte care pot fi plasate pe formular şi cărora li se poate asocia cod. Controalele au un comportament consistent şi prin combinarea lor pe suprafaţa formularului conduc la obţinerea comportamentului dorit al aplicaţiei.

Toolbox are un număr de controale standard (acestea sînt o parte intrinsecă a lui VB şi pot fi folosite direct în orice proiect Standard EXE). Menţinerea cursorului asupra imaginii unui control din **Toolbox** duce la afişarea unui *ToolTip* (un mesaj care conţine numele controlului). Cîteva controale standard din **Toolbox** sînt:

Control	Utilizare
Label	Afişează un text informativ, static, în proprietatea **`Caption`** care descrie conţinutul controlului adiacent.
Textbox	Citeşte sau afişează date de la utilizator prin proprietatea **`Text`**.
Command Button	La apăsare, realizează execuţia unei secvenţe de cod definite de utilizator. Textul afişat e din proprietatea **`Caption`**, iar codul se asociază evenimentului **`Click`**.

Option Button	Permite selectarea de către utilizator a unei opţiuni dintr-un grup de variante. Cel mai des se folosesc proprietăţile **`Caption`** şi **`Value`**, eventual evenimentul **`Click`**.
Checkbox	Permite selectarea sau deselectarea mai multor variante. Cel mai des se folosesc proprietăţile **`Caption`** şi **`Value`**, eventual evenimentul **`Click`**.
Frame	Conţine un grup de controale între care este legătură, tipic, acestea sînt **Option Button** sau **Checkbox**. Proprietatea **`Caption`** se foloseşte cel mai des.
Listbox	Afişează o listă fixă de articole între care utilizatorul se poate deplasa în vederea selectării unui articol sau a mai multora, în funcţie de setări.
Combo Box	La fel ca şi **Listbox**, dar lista poate fi editată.
Image	Afişează o imagine statică pe baza unuia dintre următoarele formate de imagine: *.bmp*, *.ico*, *.jpg*, *.wmf*, *.gif* etc.
Picture Box	La fel cu **Image**, dar cu posibilitatea adăugării unor elemente de imagine de tipul linie (**Line**), cerc (**Circle**) etc. şi tipărire (**Print**).
Line	Linie folosită pentru separarea unor grupuri de controale.
Shape	Un obiect grafic de formă predefinită. Proprietatea **`Shape`** permite alegerea formei.
Timer	Se foloseşte la generarea unor evenimente la intervale regulate de timp, de exemplu la un ceas, la afişarea întîrziată a unor mesaje. Proprietatea **`Interval`** defineşte intervalul de timp al generării evenimetului **`Timer`**. Acestui eveniment i se poate asocia o secvenţă de cod.
Data Control	Folosit pentru extragerea datelor ce se vor afişa în controale legate de **Data Control**. Are facilităţi de navigare şi editare a datelor. Se folosesc proprietăţile **`Database`** şi **`Recordsource`**. Este niţel depăşit de controlul **ADO Data Control**.

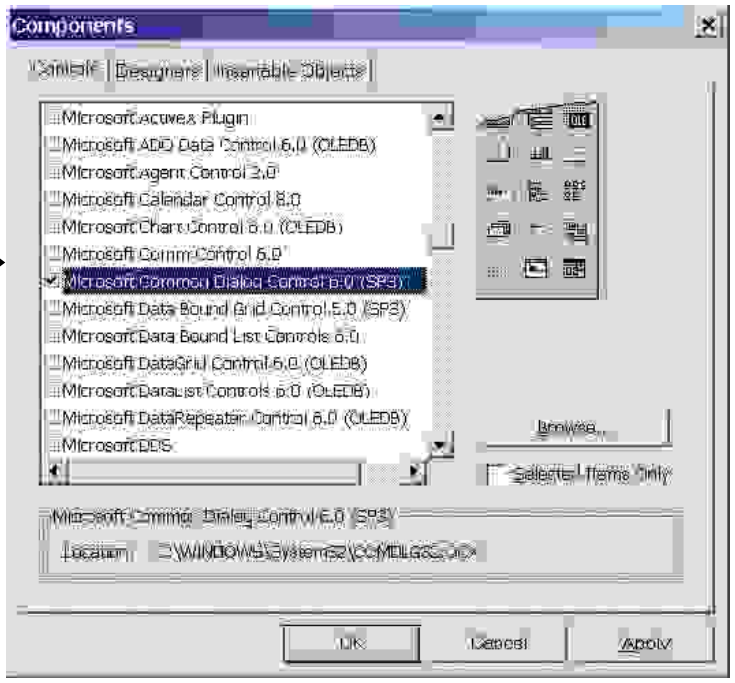
Adăugarea unui control la Toolbox

Toolbox-ul poate să nu conţină toate controalele necesare dezvoltării unui proiect de la început. În acest caz noile controale pot fi adăugate în 3 feluri:

- cu mouse-ul pe suprafaţa Toolbox-ului clic pe butonul din dreapta, apoi selectaţi *Components*;
- din meniul *Project*, selectaţi *Components*;
- apăsaţi simultan tastele **Ctrl** şi **T** ➜ **Ctrl+T**.

Deseori, se adaugă **Microsoft Common Dialog Control** pentru accesul la ferestre Windows de fişiere (File selection), de tipărire (Printer), de culori (Color), de caractere (Fonts) şi **Microsoft Windows Common Controls** care include controalele **Toolbar**, **Listview**, **Slider**.

Clic pe butonul drept.



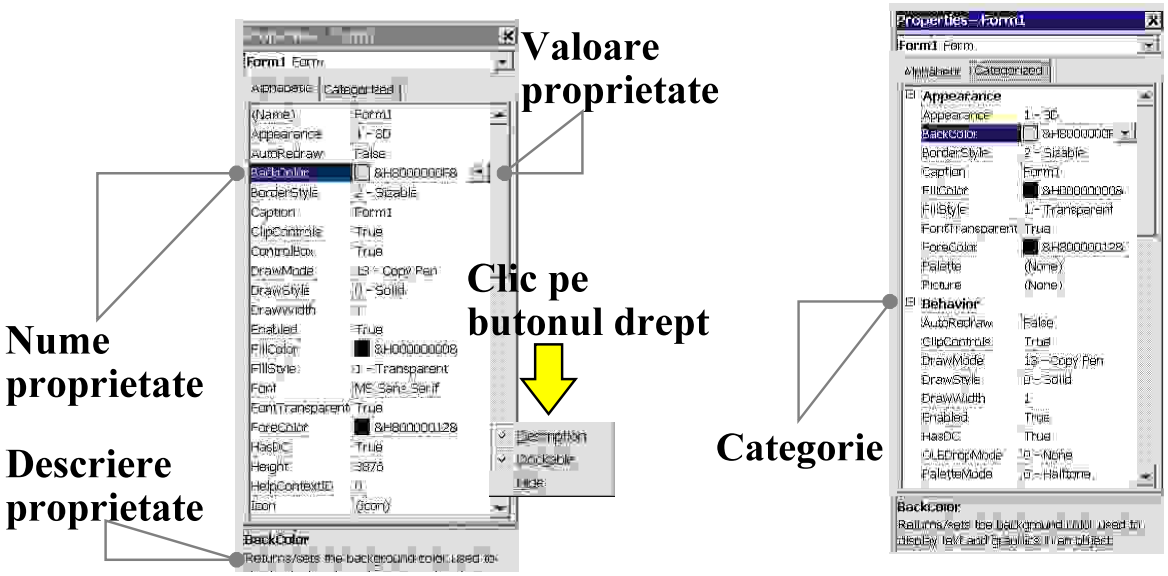
Organizarea controalelor în Toolbox

Deoarece numărul controalelor care pot fi adăugate la Toobox poate să fie foarte mare într-o aplicație complexă opțiunea **Add Tab** permite organizarea controalelor pe categorii, caz în care realizînd clic pe o anumită categorie vor fi afișate numai controalele pe care le-am adăugat noi acolo cu **Add Tab**.

Fereastra proprietăților

Permite accesul la valorile proprietăților după:

- **Nume**
- **Categorie**



9. Fereastra proprietăților

Fereastra proprietăților (**Properties Window**) este necesară la proiectarea formularelor deoarece permite vizualizarea şi modificarea proprietăților obiectelor în timpul desfăşurării proiectării.

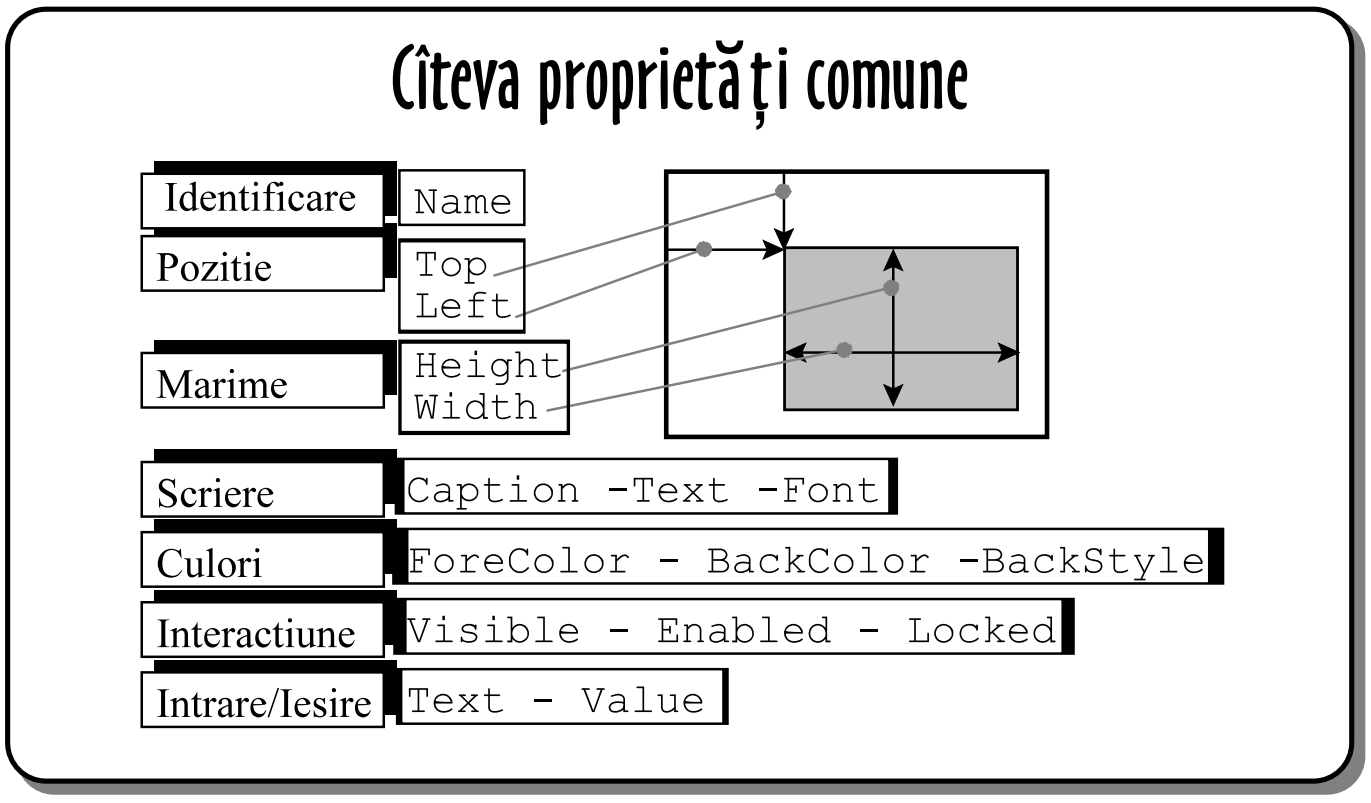
O proprietate poate fi privită ca o valoarea asociată unui aspect particular al unui obiect. Ea reprezintă un element de dată sau o informație care ajută la definirea stării obiectului individual. Există proprietăți care sînt comune majorității obiectelor, iar altele care sînt specifice anumitor obiecte. Cîteva proprietăți comune sînt: **`Height`**, **`Width`**, **`Visible`** sau **`BackColor`**. Pentru modificarea valorii unei proprietăți, scrieți noua valoare dorită pe locul celei vechi sau selectați noua valoare dintr-o listă, eventual dintr-o fereastră de dialog (în cazul culorilor sau a caracterelor).

Caracteristicile ferestrei proprietăților

Proprietățile sînt organizate pe două coloane; prima conține numele proprietății, iar a doua valoarea ei. Majoritatea programatorilor preferă afişarea lor în ordine alfabetică (butonul **Alphabetic** este selectat), dar este posibilă şi afişarea lor pe categorii (butonul **Categories** este selectat).

O scurtă descriere a fiecărei proprietăți apare în partea de jos a ferestrei, mărimea ferestrei se poate modifica ridicînd marginea de sus a ei. Fereastra poate fi închisă prin deselectarea opțiunii *Description* din meniu contextual afişat la clic pe butonul din dreapta.

Tip: **F4** permite deschiderea ferestrei proprietăților, iar **F1** afişează un mesaj de ajutor corespunzător proprietății selectate.



10. Cîteva proprietăți comune

Numărul proprietăților este mare, cele de bază vor fi prezentate acum, unele vor fi descrise, mai tîrziu, pe parcursul cursului, iar altele pot fi găsite în *Help*.

Poate cea mai importantă proprietate este **`Name`** deoarece aceasta permite referirea controlului din codul aplicației. O metodologie de numire a controalelor va fi prezentată după descrierea principalelor proprietăți comune din fereastra proprietăților.

Denumire	Descriere
`Top`, **`Left`**, **`Heigth`**, **`Width`**	Formularele şi majoritatea controalelor de pe suprafața lor au aceste proprietăți ce specifică poziția colțului stînga-sus, lungimea şi înălțimea formularului. Pentru formular ele îi definesc poziția în ecran, pentru controale ele definesc poziția acestora în cadrul formularului (sau într-un control container cum sînt **Picturebox** sau **Frame**). Unitatea de măsură implicită este twip-ul (1440 twips = 1 inch = 2.54 cm).
`Caption`, **`Text`**	**`Caption`** se foloseşte pentru textul static (nu poate fi controlat de utilizator) afişat în bara de titlu a formularului sau scris în **Label**, respectiv alături de **Checkbox**, **Option** sau **Command** Button. **`Text`** se foloseşte pentru editarea sau selectarea conținutului unui control de către utilizator într-un **Textbox**, **Listbox** sau **Comb Box**.
`Font`	Este o proprietate specială deoarece este un obiect care are la rîndul lui mai multe proprietăți. Un font (tip de caracter) are proprietățile **`Name`**, **`Size`**, **`Bold`**, **`Italic`** etc. ce pot fi setate individual prin cod sau printr-o fereastra de dialog specială din fereastra proprietăților.
`BackColor`, **`ForeColor`**, **`BackStyle`**	**`BackColor`** specifică culoarea fondului unui formular sau control. **`ForeColor`** specifică culoarea de scriere şi de tipărire care se va folosi. **`BackStyle`** (stilul fondului) este disponibilă uneori şi poate fi transparentă caz în care **`BackColor`** nu mai are efect. Este utilă la **Labels**.
`Visible`, **`Enabled`**, **`Locked`**	Deoarece aceste proprietăți au valori boleene, ele pot lua numai valorile **True** sau **False**. **`Visible`** permite afişarea controlului, dacă are valoarea **False** controlul este invizibil şi nu se poate interacționa cu el. Dacă **`Enabled`** este **False** utilizatorul vede controlul, dar nu poate interacționa cu el (are culoare lui gri specială). Dacă este disponibilă, **`Locked`** permite interacțiunea, dar fără modificări din partea utilizatorului (de exemplu într-un **Textbox** informația poate fi selectată şi copiată dar nu se poate modifica).

`Value`	`Value` se foloseşte în cazul controalelor **Checkbox** şi **Option Button** pentru a indica starea de selecţie. Valoarea este **True** sau **False** pentru un **Option Button**, dar pentru un **Checkbox** acestea sînt **Checked**, **Unchecked** şi **Grayed** (indică imposibilitatea selectării de către utilizator; mai este folosită şi pentru afişarea în unor stări mixte).

Convenţii de nume

Există convenţii standard pentru numele de:

- **controale;**
- **componente;**
- **variabile.**

Numele trebuie să descrie tipul obiectului + semnificaţia lui în program, de ex.:

- `txtNume` **- Textbox în care se va stoca numele unei persoane;**
- `frmValInit` **- Form care conţine valorile de iniţializare ale aplicaţiei.**

11. Convenţii de nume

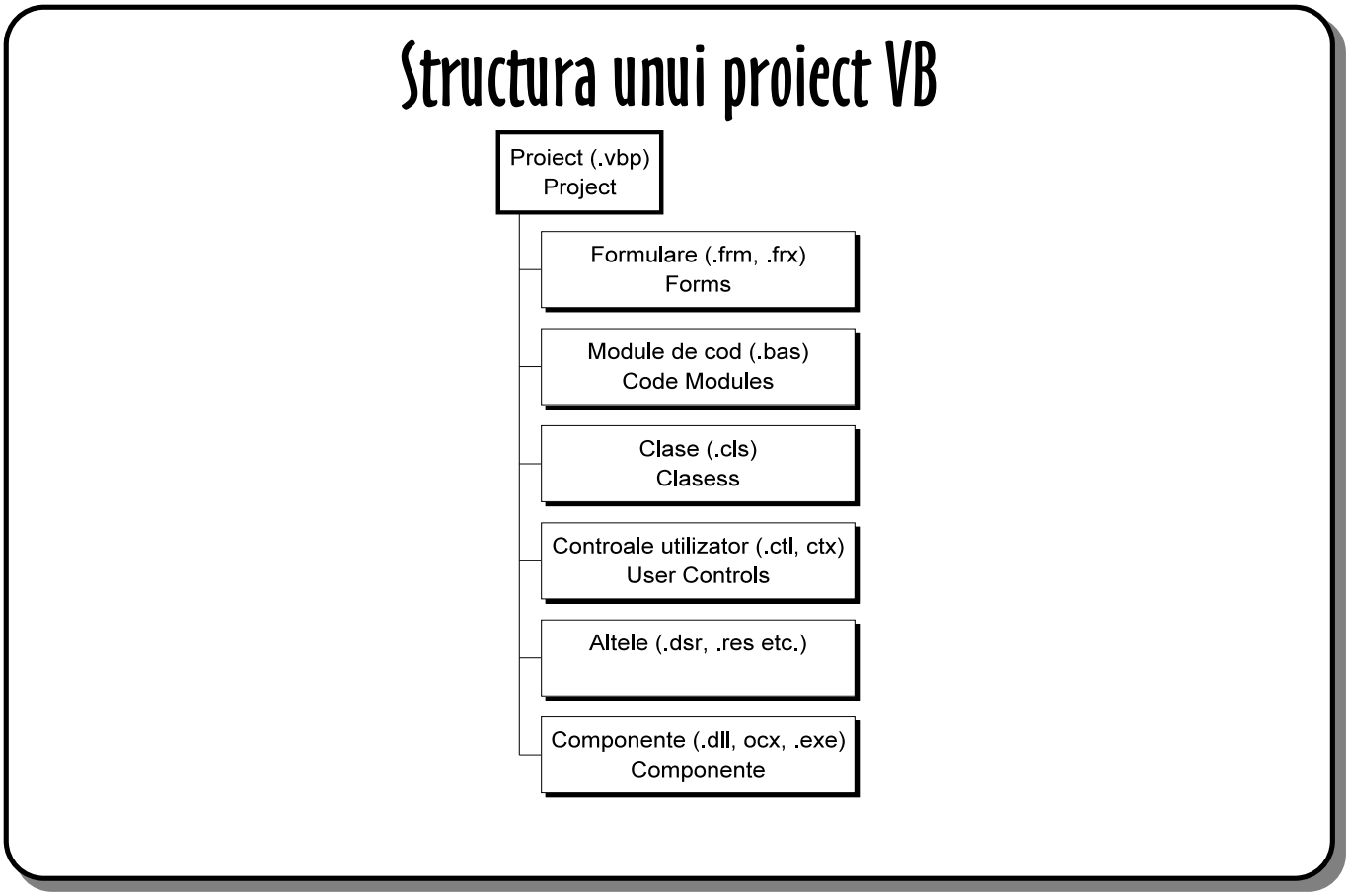
Convenţii implicite

Minimul convenţiilor de numire este cel folost implicit de VB pentru formulare şi controale, de exemplu: `Form1`, `Label7` şi `Text12`. Aceste nume sînt însă mult mai puţin sugestive decît `frmValInit`, `lblAdresa` sau `txtNume`.

Variabilele corespunzătoare tipurilor de date standard pot folosi convenţii de nume care includ vizibilitatea variabilei (acestea se vor discuta în cursul următor).

Cîteva sugestii de nume

Tip control	Prefix	ListBox	lst	Tip Variabilă	Prefix
Checkbox	chk	**Option Button**	opt	**Boolean**	b
Combo Box	cbo	**Picture Box**	pic	**Double**	dbl
Comand Button	cmd	**Timer**	tim	**Integer**	int
Image	img	**Label**	lbl	**Object**	ob
Textbox	txt	**Shape**	shp	**String**	str



12. Structura unui proiect VB

Organizarea obişnuită a unităţii de lucru este proiectul. El este un fişier text ce stochează referinţele la o combinaţie de formulare, clase, controale etc. folosite în proiect. Este bine ca toate fişierele proiectului să fie stocate într-un singur director.

Este posibil ca unele fişiere să fie partajate între mai multe proiecte. În acest caz apar conflicte dacă utilizatori diferiţi încearcă să modifice aceleaşi fişiere. Programe pentru controlul codului sursă pot fi utilizate pentru a preveni aceste accidente. Microsoft Visual SourceSafe este un exemplu de astfel de program care se poate integra direct în VB cu ajutorul unui **Add-Ins**.

Componentele compilate folosite în proiect sînt descrise în fişierul cu extensia .vbp, dar sînt accesate prin informaţiile din registrul Windows-ului fără a fi stocate în directorul care conţine fişierele proiectului.

Folosiţi Notepad-ul pentru a vizualiza conţinutul fişierelor unui proiect, veţi găsi următoarele tipuri de componente VB6:

- **Formulare**, se stochează în fişiere text cu extensia **`.frm`**. Orice conţinut binar va fi salvat sub acelaşi nume de fişier dar cu extensia **`.frx`**;
- **Module de cod**, se stochează ca fişiere text şi au extensia **`.bas`**;
- **Clase**, este stocat şi în format text în fişierele cu extensia **`.cls`**;
- **Controale necompilate**, se salvează ca fişiere text cu extensia **`.ctl`**, iar

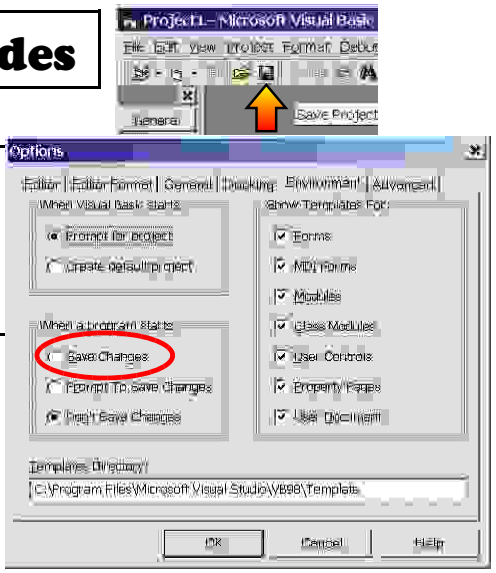
formatul binar sub extensia **.ctx** (la fel ca la formulare);

- **Fişiere de resurse**, folosite pentru internaţionalizarea proiectului, au extensia **.res**.
- **Mediile** de prelucrare a datelor sau unele **instrumente de dezvoltare** folosesc fişiere text auxiliare cu extensia **.dsr** (sau **dsx** dacă sînt compilate).

În final, este posibilă combinarea a 2 sau mai multor proiecte într-un **grup de proiecte** care va avea extensia **.vbg**. Aceasta este o procedură comună folosită la testarea şi depanarea proiectelor ActiveX in-process.

Salvarea proiectului

- **Păstraţi toate componentele într-un singur director.**
- **Salvaţi manual cît mai des**
- **Folosiţi opţiunea de salvare înainte de execuţie**



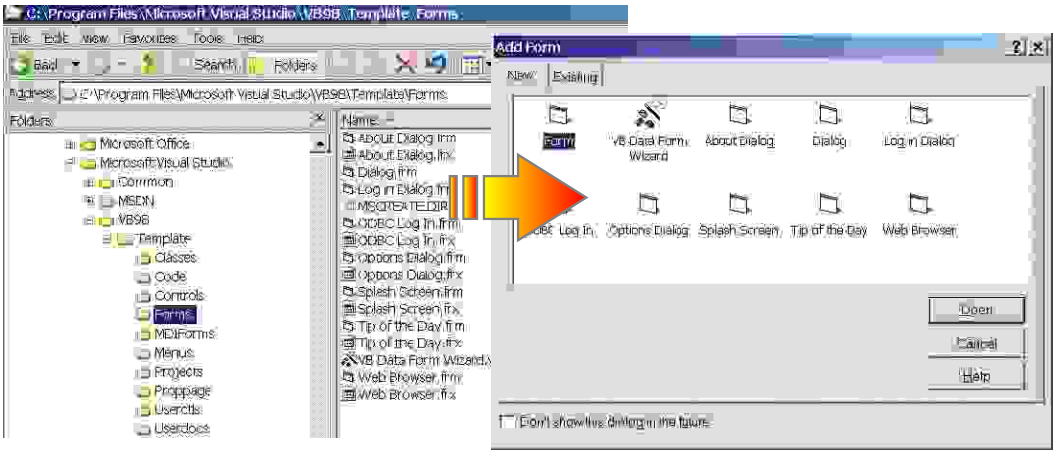
13. Salvarea proiectului

La prima salvare a unui proiect VB vrea să-l salveze în directorul de lucru propriu (**C:\Program Files\Microsoft Visual Studio\VB98**). Aici trebuie ales un alt director pentru salvarea proiectului. O problemă este aceea că după închidere, dacă proiectul este redeschis şi se vor adăuga noi componente, se poate ca VB să încerce să salveze din nou în directorul lui propriu, în locul celui care conţine celelalte fişiere ale proiectului.

Mediul VB are o setare prin care poate salva automat modificările aduse unui program înainte de lansarea lui în execuţie (*Tools* ➜ *Option* ➜ *Environment* ➜ *Save Changes*).

Utilizarea şabloanelor (templates)

Convenabile - Consistente - Sigure



14. Utilizarea şabloanelor (templates)

O parte a lui VB standard este directorul **`VB98`**, care conţine un subdirector **`Templates`**. Acesta, la rîndul lui, conţine subdirectori cu şabloane pentru formulare, clase, controale şi proiecte actuale.

Prin salvarea unui fişier în directorul corespunzător se pot fabrica componente predefinite pentru noile proiecte. În fereastra de dialog *Add Form*, observaţi că la unele icon-uri le corespund 2 fişiere (cu extensiile **`.frm`** şi **`.frx`**) în subdirectorul **`Forms`**.

Tehnologia de lucru cu şabloane permite implementarea textului "Programează o singură dată, foloseşte de cît mai multe ori".