

Lucrarea nr. 2

Elemente de grafică în Visual C++

1. Scopul lucrării:

Lucrarea de fata își propune familiarizarea cu lucrul în mod grafic în limbajul Visual C++: fundamentele proiectării grafice GDI, alegerea culorilor, tratarea erorilor, realizarea figurilor grafice de bază. Sunt prezentate modalitățile de dezvoltare a aplicațiilor grafice folosind platforma Microsoft Foundation Classes (MFC).

2. Noțiuni teoretice:

Microsoft Visual Studio .NET – limbajul Visual C++

Introducere

Visual C++ face parte, alături de Java, dintr-o nouă generație de limbi bazate pe paradigma programării orientate obiect. Produsul a devenit mult, mult mai mult decât un simplu compilator. Sunt incluse clasele fundamentale Microsoft (MFC), care simplifică și accelerează dezvoltarea aplicațiilor Windows. Permite programatorului să dezvolte în mod elegant aplicații complexe cu ajutorul unui set puternic de instrumente. În mediul de dezvoltare pentru Visual C++ programatorul își poate alege obiectele care trebuie să facă parte din interfața grafică cu mouse-ul și apoi le poate deplasa în fereastră, astfel încât să extindă interfața în mod vizual și nu numai prin cod. Un instrument pentru depanare integrat perfect va permite inspectarea în detaliu a fiecărui aspect din cadrul programului aflat în execuție. Nucleul acestei tehnologii este programarea bazată pe componente de dimensiuni reduse, ușor de construit și de folosit. Pe lângă faptul că Visual Studio .NET este ideal pentru crearea foarte rapidă a unor aplicații Windows, el permite de asemenea folosirea tuturor trăsăturilor avansate ale limbajului C++, precum și a funcțiilor Windows API. Ca o extensie a Visual Studio 6, este posibilă și scrierea de cod MFC.

Clasele obiectelor din GUI au anumite funcții: meniu, timer, textbox, buton, etc. Ele sunt introduse pur și simplu în proiect și în continuare se folosesc metodele și proprietățile lor, iar comunicarea cu sistemul de operare Windows se realizează transparent pentru programator.

2.1. Componentele mediului de programare Visual C++

- ✓ editor de cod și un gestionar de proiecte interactive. Un proiect reprezintă o colecție de fișiere sursă legate între ele care sunt compilate și link-editate pentru a forma un program executabil. De obicei, fișierele sursă ale unui proiect se află pe disc într-un director separat. Un proiect depinde de mai multe fișiere care nu se află în directorul acestuia și care fac parte dintr-o bibliotecă de fișiere. Visual C++ permite lucrul cu mai multe proiecte în același timp, folosind fișiere de descriere a mediului de lucru (*workspace* - au extensia .dsw) și a fiecărui proiect în parte (fișiere cu extensia .dsp).

- ✓ editor de resurse. Resursele reprezintă elemente de interfață cu utilizatorul, cum ar fi meniurile, cursoarele, cutiile de dialog, *icon-uri*, etc. Fiecare proiect construit în Visual C++ va conține un fișier cu extensia **.rc** ce cuprinde descrierea textuala a tuturor obiectelor de interfață care fac parte din aplicație. Editorul de resurse va genera în acest fișier cod corespunzător fiecărei acțiuni efectuate asupra resurselor. Nu este recomandată editarea manuală a acestui fișier.
- ✓ compilator C/C++. Acest compilator determină limbajul utilizat în fișierele sursă (C sau C++) inspectând extensia acestora (.c sau .cpp). Limbajul utilizat în fișierele sursă trebuie să respecte standardul ANSI plus extensiile introduse de Microsoft. În urma, compilării pentru fiecare fișier sursă este creat un fișier obiect cu extensia **.obj**.
- ✓ link-editor care construiește fișiere "executabile" (având extensia .exe, .dll, .lib, .ocx) folosind fișiere obiect realizate de către compilator pentru fiecare fișier din proiect.
- ✓ compilator de resurse care operează în două moduri: compilează un fișier ASCII de resurse (cu extensia .rc) generat de editorul de resurse într-un fișier binar cu extensia .res, care este apoi atasat unui fișier binar "executabil" (.exe, .lib, .dll, .ocx). Dacă sunt aduse modificări asupra fișierului .res acesta poate fi unit cu fișierul executabil fără a mai fi necesară *link-editarea*.
- ✓ depanator simbolic care este strâns legat de editorul de cod sursa. Printre caracteristicile sale principale amintim aici: posibilitatea inserării de puncte de intrerupere, execuție pas cu pas a programului, posibilitatea vizualizării în timpul execuției a valorilor variabilelor unei aplicații la un moment dat, etc. Pentru a putea folosi depanatorul simbolic asupra unui proiect este necesară setarea corespunzătoare a anumitor opțiuni de compilare și *link-editare*.
- ✓ generator de aplicații-sablon, numit AppWizard, care creează "scheletul" unei aplicații Windows generice. Fișierele și conținutul acestora vor fi generate în urma unui dialog direct cu programatorul. Codul generat de către AppWizard este un cod care ajută programatorul să realizeze rapid o aplicație inițială pe baza căreia va dezvolta proiectul în continuare.
- ✓ instrument care automatizează anumite faze ale procesului de implementare a aplicației, numit ClassWizard. Dacă se dorește crearea unei noi clase sau a unei funcții prin care o anumită clasă să răspundă la apariția unui mesaj, ClassWizard va micșora efortul de programare prin generarea declarației și definiției clasei sau funcției precum și a conexiunii dintre funcție și mesaj.
- ✓ navigator prin sursele unui proiect. Pentru o aplicație implementată de la un capăt la altul de către un singur programator este foarte posibil ca acesta să cunoască bine conținutul tuturor fișierelor sursă, a claselor sau a funcțiilor membru ale acestora. În schimb, dacă se încearcă înțelegerea codului unei aplicații scrise de o altă persoană este nevoie de puțin ajutor. Navigatorul (*browser-ul*) mediului Visual C++ permite examinarea unei aplicații atât prin prisma claselor și membrilor acestora cat și prin prisma fișierelor sursă. Cu acest navigator se poate selecta orice funcție, variabilă, macro sau clasă și apoi se poate vedea unde sunt acestea definite și utilizate în proiect.
- ✓ help online. Întregul conținut al manualelor referitoare la Windows SDK și biblioteca MFC sunt incluse în *help-ul* mediului Visual C++. *Help-ul* mediului Visual C++ rezolvă conflictele între numele identice ale funcțiilor SDK și cele din biblioteca MFC.

- ✓ instrumente de diagnosticare Windows: SPY pentru observarea mesajelor Windows, HEAPWALK pentru examinarea memoriei, HC31 pentru compilarea fișierelor *help* și STRESS pentru memoria disponibilă limitată artificial.

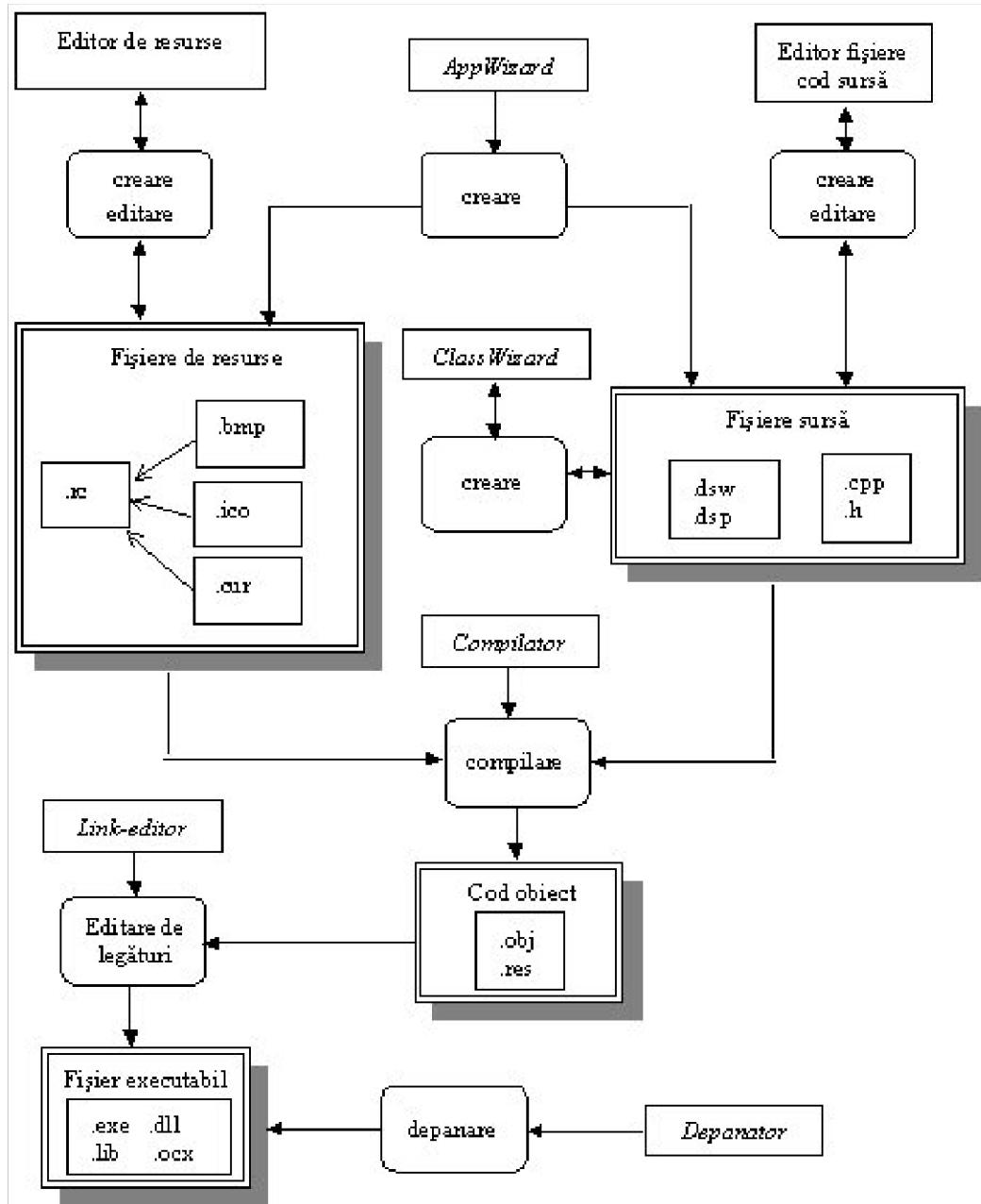


Figura 1. Principalele componente ale mediului Visual C++ și interacțiunea acestora cu fișierele proiectului

În Figura 1. se prezintă într-o manieră grafică relațiile dintre editoarele de cod sursă și resurse, ClassWizard, AppWizard și fișierele sursă conținute într-un proiect.

2.2. Componentele mediului de programare Visual C++

Biblioteca de clase MFC poate fi împărțită pe trei nivele. La primul nivel se află clasele care descriu comportamentul obiectelor grafice și încapsulează funcțiile API

(*Application Program Interface*) Windows. Aceste obiecte (cum ar fi de exemplu ferestrele, butoanele, fonturile, contextul dispozitiv, etc.) sunt create și accesate de către programatorii Windows prin intermediul unor funcții sistem. Prin urmare clasele din MFC de la acest nivel încapsulează practic obiectele și funcțiile care le sunt atașate existente în API Windows, oferind un grad ridicat de abstractizare. La al doilea nivel se află clasele care nu depind de sistemul de operare Windows și care implementează structuri fundamentale de date (liste, tablouri, dicționare, siruri de caractere, dată calendaristică, etc.). Ultimul nivel este reprezentat de clasele care fac parte din mediul de lucru al aplicațiilor (*application framework*), și anume clasele aplicație, view, document, etc. Un mediu de lucru al aplicațiilor este o colecție de componente soft (clase, *macro-uri*, funcții globale) care pot fi utilizate în realizarea unei aplicații generice.

2.3. Lansarea mediului de programare Visual C++ și crearea unui nou proiect

In cele ce urmează prezentăm pașii necesari pentru crearea unui proiect cu **Microsoft Visual Studio .NET**.

- ✓ Pentru a lansa Visual C++, selectați **Microsoft Visual Studio** din cadrul meniului **Start** de pe bara de sarcini. Crearea unui proiect nou se face selectând comanda **New Project** din fereastra principală.

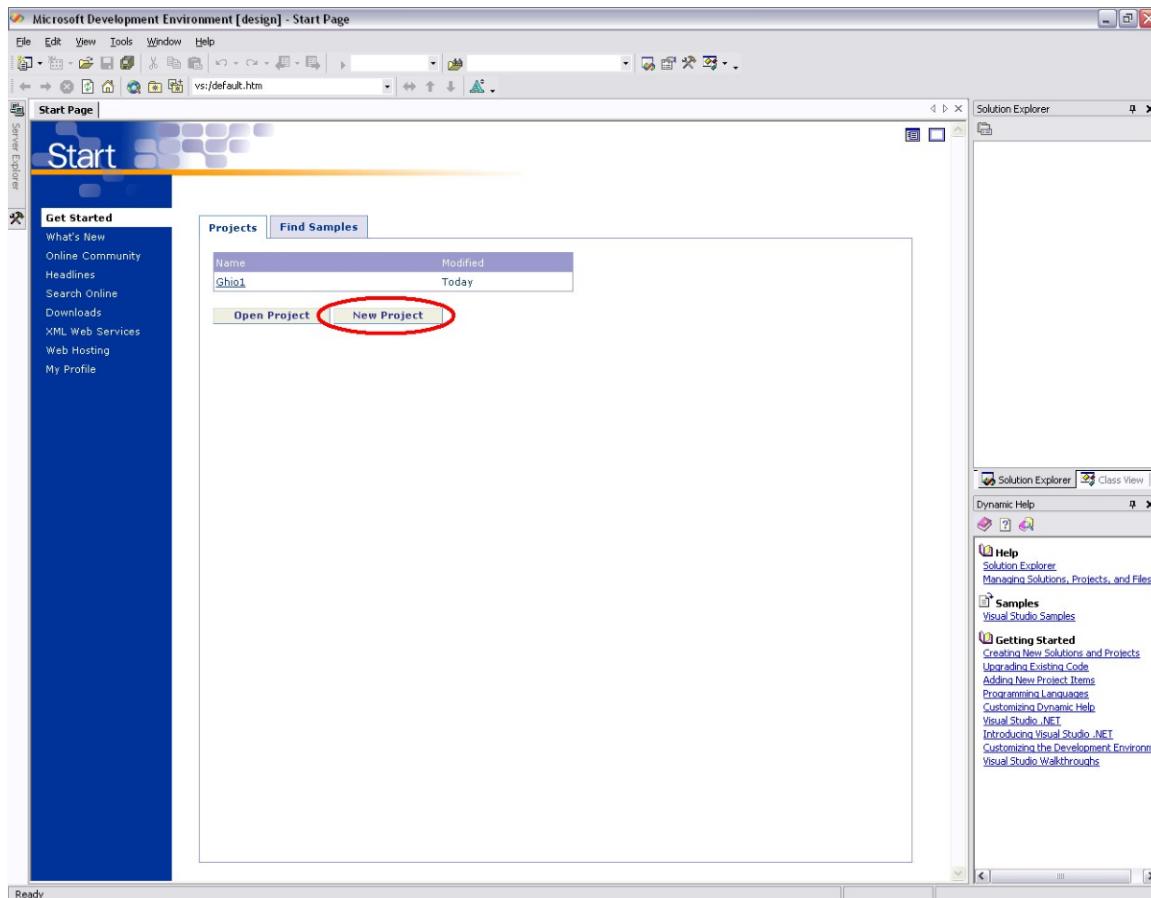


Figura 2. Fereastra principală Microsoft Visual Studio

- ✓ Caseta de dialog **New Project** va fi afișată ca în Figura 3. Selectarea tipului de proiect se face alegând opțiunea **Visual C++ Projects** din arborele din stânga. În partea dreaptă a ferestrei va apărea o listă cu mai multe tipuri de proiecte. Pentru

acest exemplu selectați **MFC Application** din lista cu tipurile posibile de proiecte. Selectarea acestei opțiuni înseamnă ca proiectul va avea ca rezultat un program executabil. Apoi se alege denumirea proiectului și locația în care vor fi salvate fișierele proiectului.

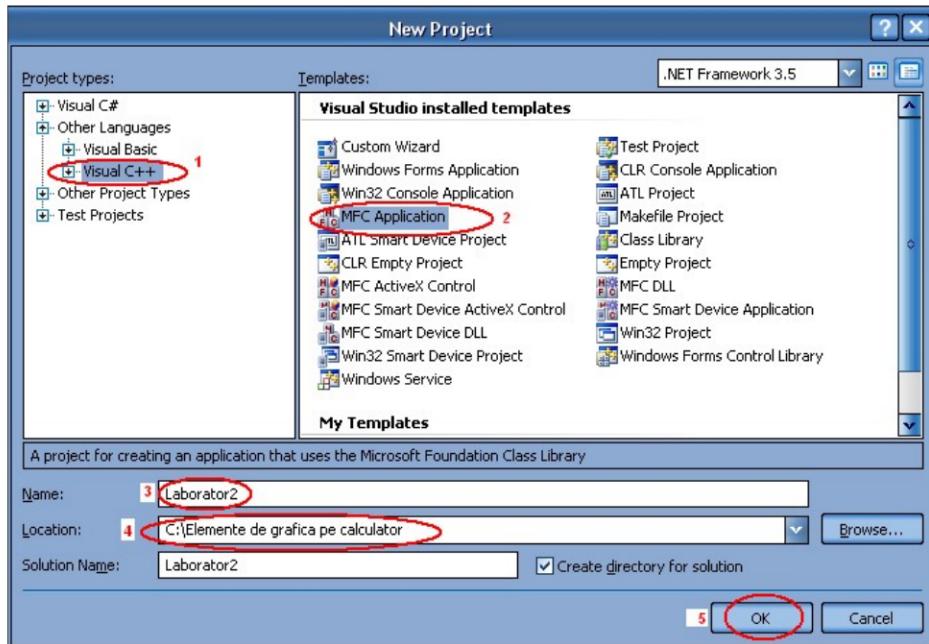


Figura 3. Caseta de dialog New Project

- In caseta de dialog **MFC Application Wizard** selectați **Application Type** și bifați tipul aplicației **Dialog based** (Figura 4).

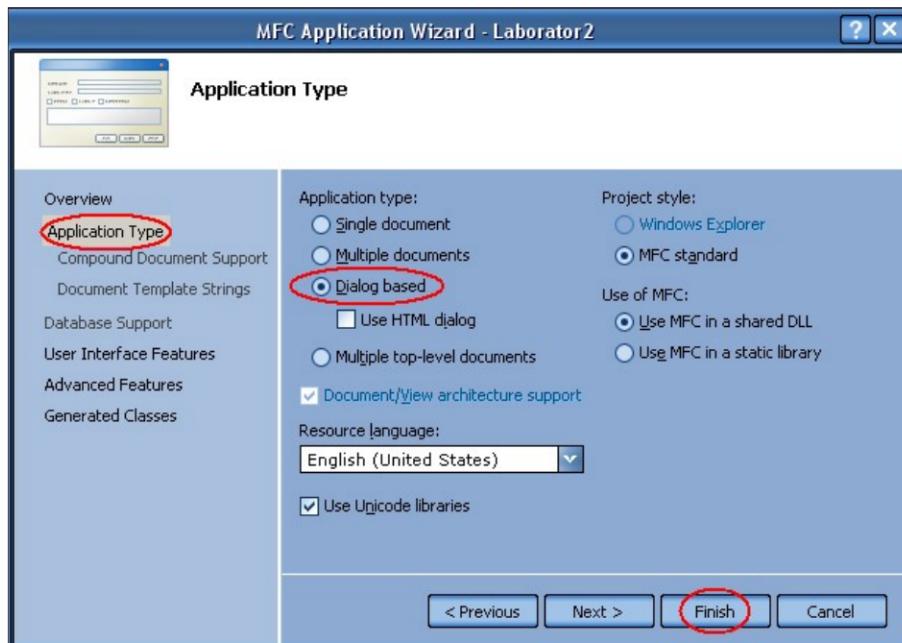


Figura 4. Caseta de dialog MFC Application Wizard

- ✓ Când este creat un nou proiect Visual C++ de tip MFC, apare o fereastră cu toate caracteristicile setate implicit în Windows. În mijlocul ecranului, apare o „formă”

(Form) – fereastra principală a programului, în care se vor adăuga diverse componente de control: butoane, textbox-uri, etc. (vezi Figura 5).

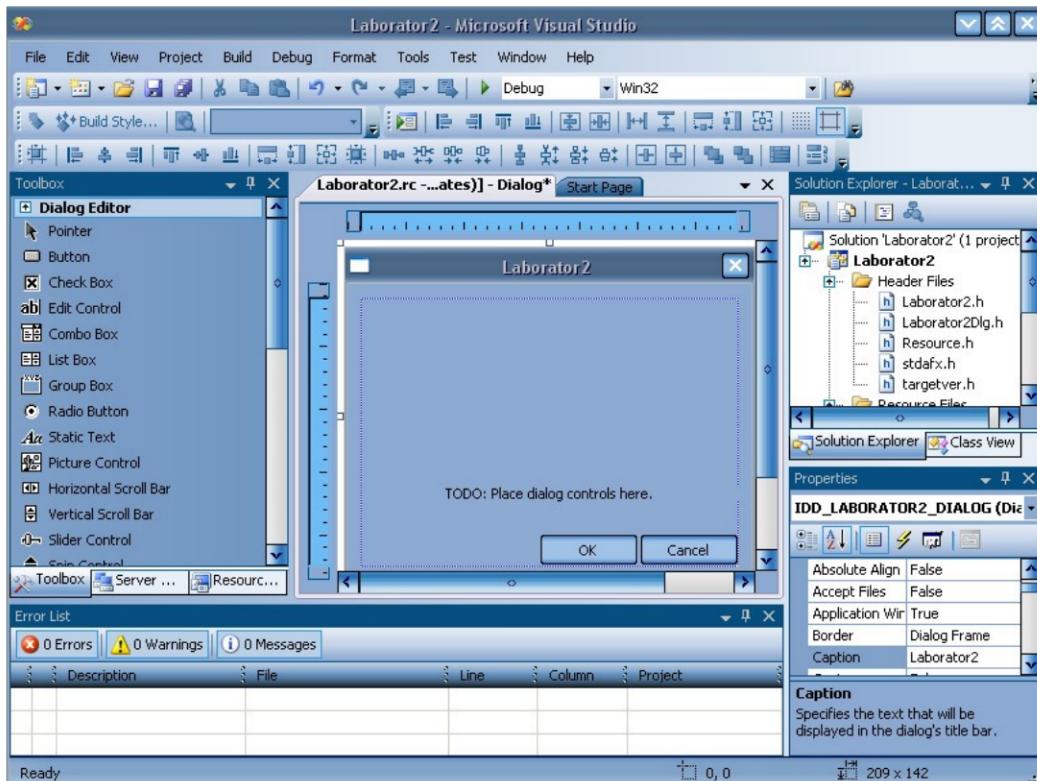


Figura 5. Asamblarea proiectului

În partea din stânga a ecranului există un toolbar (*View→Toolbox*, *Ctrl+Alt+X*) din care se vor alege cu mouse-ul componentele ce vor fi adăugate în fereastră.

Pentru adăugarea unei componente, programatorul va face click cu mouse-ul pe imaginea corespunzătoare din toolbox, apoi va face click în formă, în locul unde dorește să apară componenta respectivă. Odată introduse în fereastră, componentele pot fi mutate, redimensionate, copiate sau șterse. În dreapta este o fereastră de proprietăți (*View→Properties Window*, F4). De aici, fiecărei componente folosite i se pot modifica proprietățile, adică aspectul exterior, aşa cum va apărea în program. De asemenea, se pot selecta evenimentele corespunzătoare componentei care vor fi tratate în program. Secțiunea afișată de către Visual Studio în cele două părți: stânga și dreapta se numește secțiunea *spațiului de lucru*. Se remarcă faptul că, după ce proiectul a fost creat, secțiunea spațiului de lucru oferă trei pagini: **ClassView**, **ResourceView**, și **Solution Explorer**. Aceste pagini permit accesarea oricărei componente a proiectului.

Notă:

AppWizard este utilizat exclusiv pentru crearea noilor proiecte. Nu se poate reveni la casetele cu opțiuni AppWizard în cadrul unui proiect deja existent. Dacă descoperiți ca ați făcut o alegere greșită și doriti să reluați etapele AppWizard, trebuie să înlăturați mai întâi proiectul existent. Pentru aceasta se șterge directorul acestuia.

Grafică în Visual C++

Pentru lucrul în mod grafic, Visual C++ pune la dispoziția programatorului o clasă numită *Graphics* ce permite accesul la proprietățile grafice GDI+ de bază. GDI+ permite utilizarea elementelor sofisticate de grafică pentru jocuri și alte medii grafice. Aici, obiectul *Graphics* oferă metode pentru a desena arce, elipse, linii, puncte și chiar stringuri.

1. *Clase grafice: furnizează metodele pentru desenare pe monitor.*

NUME CLASĂ	SCOP
Bitmap	Încapsulează un bitmap GDI+.
Brush	De la această clasă de bază sunt derivate clasele pentru definirea obiectelor care pot fi folosite pentru a umple formele grafice închise. Formele pot fi elipse, poligoane, felii de plăcintă, dreptunghiuri, etc.
Brushes	Furnizează pensule pentru toate culorile standard.
ColorConverter	Convertește o culoare de la un tip de date la altul.
ColorTranslator	Translatează culorile la sau de la obiectele Color GDI+.
Cursor	Reprezintă imaginea cursorului.
CursorConverter	Convertește cursorul de la un tip de date la altul.
Cursors	Definește cursoarele standard.
Font	Definește proprietățile fonturilor pentru text. Acestea sunt aspectul fontului, dimensiunea și atributele de stil.
FontConverter	Convertește fonturile de la un tip de date la altul.
FontFamily	Definește un grup de fonturi cu design similar și cu variații minore de stil.
Graphics	Încapsulează o suprafață de desen GDI+.
Icon	Reprezintă o pictogramă.
IconConverter	Convertește o pictogramă de la un tip de date la altul.
Image	Furnizează funcționalitate pentru bitmap-uri, cursoare, pictograme și metafișiere.
ImageAnimator	Suportă animația imaginilor.
ImageConverter	Convertește o imagine de la un tip de date la altul.
ImageFormatConverter	Convertește culorile de la un tip de date la altul.
Pen	Definește proprietățile peniței pentru desenarea obiectelor, cum ar fi arce de cerc, linii, dreptunghiuri și multe altele.
Pens	Furnizează penițe în toate culorile standard.
PointConverter	Convertește un punct de la un tip de date la altul.
RectangleConverter	Convertește un dreptunghi de la un tip de date la altul.
Region	Descrie interiorul unei forme grafice compuse din dreptunghiuri și căi.
SizeConverter	Convertește o dimensiune de la un tip de date la altul.
SolidBrush	Definește o pensulă de o singură culoare.
StringFormat	Încapsulează informații despre layout-ul textului, cum ar fi alinierea, spațiul dintre rânduri și alte trăsături.
SystemBrushes	Definește pensule pentru sistemul extins de culori specific Windows.
SystemColors	Oferă acces la culorile sistemului extins specific Windows. SystemPens și SystemBrushes sunt preferate față de SystemColors.
SystemIcons	Definește obiectele pictogramă pentru sistemul extins de

	pictograme specific Windows.
SystemPens	Definește penițe pentru culorile selectate ale sistemului extins specific Windows.
TextureBrush	Încapsulează o pensulă care utilizează o imagine pentru umplerea unei forme închise.
ToolboxBitmapAttribute	Definește imaginile utilizate cu o componentă specificată.

2. Structuri utilizate de clasele grafice:

STRUCTURĂ	SCOP
Color	Definește o culoare ARGB.
Point	Definește o pereche ordonată de coordonate x și y. Aceste coordonate definesc un punct într-un plan bidimensional.
PointF	Definește o pereche ordonată de coordonate x și y în virgulă mobilă. Aceste coordonate definesc, de asemenea, un punct într-un plan bidimensional.
Rectangle	Stochează localizarea și dimensiunile pentru o regiune dreptunghiulară.
RectangleF	Stochează localizarea și dimensiunile pentru o regiune dreptunghiulară, definită prin valori în virgulă mobilă.
Size	Definește dimensiunea unei regiuni dreptunghiulare, cu o pereche ordonată de valori pentru lățime și înălțime.
SizeF	Definește dimensiunea unei regiuni dreptunghiulare, cu o pereche ordonată de valori în virgulă mobilă pentru lățime și înălțime.

3. Enumerările folosite de clasele grafice:

ENUMERAREA	SCOP
BrushStyle	Furnizează o varietate de siluri pentru pensule.
ContentAlignment	Furnizează alinierea conținutului pe o suprafață de desen.
FontStyle	Furnizează informații de stil pentru text.
GraphicsUnit	Furnizează o unitate de măsură pentru date.
KnownColor	Furnizează culorile sistemului.
PenStyle	Definește stilurile penițelor.
PolyFillMode	Identifică modul de umplere a poligoanelor formate din regiunile de suprapunere.
StringAlignment	Furnizează alinierea unui string text în raport cu dreptunghiul de layout.
StringDigitSubstitute	Furnizează informații asupra stilului aplicat la String Digit Substitute.
StringFormatFlags	Furnizează informațiile de afișare și layout pentru stringul text.
StringTrimming	Înlătură caracterele dintr-un string când acesta nu începe în layout.
StringUnit	Furnizează unitățile de măsură pentru un string text.

4. Metode comune disponibile pentru desen și destinația utilizării în cadrul caselor grafice:

METODA	DESTINAȚIA
--------	------------

FromHDC	Furnizează o nouă instanță a clasei grafice, folosind o administrare pentru contextul dispozitivului (HDC – <i>Handle to the Device Context</i>).
FromHWND	Furnizează o nouă instanță a clasei grafice, folosind o administrare pentru fereastră (HWND – <i>Handle to the Window</i>).
FromImage	Generează o instanță a unei clase grafice, folosind o imagine existentă.
GetHalftonePalette	Livrează o paletă în semiton (<i>halftone</i>).

5. *Metode cu instanțe publice. Tabelul conține cea mai mare parte a primitiveelor de desen utilizate pentru crearea arcurilor de cerc, a liniilor, a elipselor, a dreptunghiurilor, și.a.m.d.* Toate formele grafice sunt trasate pornind de la o colecție predefinită de metode numite *primitive de desen*.

METODA	SCOP
AddMetafileComment	Copiază comentariul dintr-un buffer într-un metafișier cu format îmbunătățit.
BeginContainer	Furnizează începutul unui container de metafișier.
Clear	Curăță aria prin umplerea suprafeței de desen cu o culoare dată.
Dispose	Sterge grafica și eliberează memoria alocată.
DrawArc	Trasează un arc de la o specificație de elipsă dată.
DrawBezier	Trasează o curbă cubică Bezier.
DrawBeziers	Trasează o serie de curbe cubice Beziers.
DrawClosedCurve	Trasează o curbă închisă. O matrice de curbe definește curba.
DrawCurve	Trasează o curbă. Curba este definită printr-o matrice de puncte.
DrawEllipse	Trasează o elipsă.
DrawIcon	Trasează o imagine iconică.
DrawIconUnstretched	Trasează o imagine iconică. Această metodă alungește imaginea la dimensiunile date, permitând utilizatorului să furnizeze un dreptunghi drept cadru al imaginii de desenat.
DrawImage	Trasează o imagine.
DrawImageUnscaled	Trasează o imagine. Această metodă alungește imaginea la dimensiunile date, permitând utilizatorului să furnizeze un dreptunghi drept cadru al imaginii de desenat.
DrawLine	Trasează o linie.
DrawLines	Trasează o serie de segmente de dreaptă. Segmentele se conectează într-un tablou de puncte.
DrawPath	Trasează linii și curbe definite printr-un GraphicsPath.
DrawPie	Trasează conturul unei felii de plăcintă (<i>pie section</i>).
DrawPolygon	Trasează conturul unui poligon.
DrawRectangle	Trasează conturul unui dreptunghi.
DrawRectangles	Trasează conturul unei serii de dreptunghi.
DrawString	Trasează un string.
EndContainer	Furnizează sfârșitul unui container de metafișier.
EnumerateMetafile	Enumerează un fișier.
Equals (moștenită de la Object)	Află dacă obiectul utilizat reprezintă aceeași instanță ca și obiectul curent.
ExcludeClip	Specifică regiunea ocupată pentru excludere.

FillClosedCurve	Umple interiorul unei curbe închise specificate printr-un tablou de puncte.
FillEllipse	Umple interiorul unei elipse specificate printr-un dreptunghi mărginit.
FillPath	Umple interiorul unei căi.
FillPie	Umple interiorul unei felii de plăcintă.
FillPolygon	Umple interiorul unui poligon specificat cu un tablou de puncte.
FillRectangle	Umple interiorul unui dreptunghi cu o pensulă.
FillRectangles	Umple interiorul unei serii de dreptunghiuri cu o pensulă.
FillRegion	Umple interiorul unei regiuni.
Flush	Execută toate operațiile aflate curent pe stivă.
GetHashCode	Acționează ca o funcție <i>Hash</i> pentru un tip particular. Poate fi utilizată în <i>hashing</i> de algoritmi ca tabel de <i>hash</i> .
GetHDC	Obține administrarea pentru contextul dispozitivului (DC – <i>Device Context</i>) asociat cu clasa grafică.
GetLifetimeService	Returnează un obiect lifetime service în scopul de a controla gestionarea timpului de funcționare.
GetNearestColor	Găsește culoarea solidă care se potrivește cel mai bine la o culoare logică specificată.
GetType	Furnizează tipul unui obiect dat.
InitializeLifetimeService	Permite obiectelor să furnizeze propria concesiune și, astfel, controlul asupra propriului timp de funcționare.
Intersect Clip	Dă specificațiile pentru intersecția regiunii decupate.
IsVisible	Comunică dacă obiectul sau obiectele selectate sunt vizibile în cadrul panoului.
MeasureString	Returnează o dimensiune în virgulă mobilă pentru stringul măsurat.
MeasureStringRegion	Returnează regiunea stringului măsurat.
MultiplyTransform	Multiplică matricea pentru transformarea universală cu o matrice dată.
ReleaseHDC	Returnează memoria alocată pentru HDC.
ResetClip	Reinițializează regiunea decupată pentru obiect.
ResetTransform	Reinițializează transformarea universală la transformarea identitate.
Restore	Reinițializează starea grafică.
RotateTransform	Rotește matricea de transformare.
Save	Salvează starea grafică.
ScaleTransform	Scalează matricea de transformare.
SetClip	Setează regiunea decupată pentru obiect.
ToString	Returnează un string care reprezintă obiectul curent.
TransformPoints	Specifică punctele de transformare utilizate de către matricea de transformare.
TranslateClip	Furnizează informația de translatăre pentru regiunea decupată a obiectului.
TranslateTransform	Translatează matricea de transformare.

Tabelul 9.9 Metode cu instanțe publice

Proprietățile grafice

Proprietățile grafice se referă la aspecte cum ar fi trasarea și culorile de umplere, stilul liniilor, grosimile liniilor și stilul de umplere. Combinarea corectă de culori și stiluri creează un efect vizual plăcut pentru diagrame și grafice.

Culori: Culorile pentru desen sunt specificate prin crearea penișelor de desen, iar culorile de umplere sunt specificate prin crearea pensulelor. Penișele de desen conturează formele, în timp ce pensulele sunt utilizate pentru a umple cu culoare interiorul unor forme închise. De exemplu, o simplă penită roșie poate fi creată pornind de la clasa CPen cu următoarea sintaxă:

```
CPen red (0,1,RGB(255,0,0));
```

1 of 2 ▾ CPen (int nPenStyle, int nWidth, COLORREF crColor)

Stiluri de trasare a liniilor: Când este creată o nouă penită pornind de la clasa CPen, penită va trasa o linie solidă, de grosime prestabilită sau specificată. Pentru schimbarea stilului penișei se procedează astfel:

```
//trasează cu o liniuță și două puncte  
CPen red (PS_DASHDOTDOT,1,RGB(255,0,0));
```

DENUMIREA STILULUI	DESCRIERE
Custom	Permite un stil de liniuțe personalizat, definit de utilizator.
Dash	Trasează o linie alcătuită dintr-o serie de liniuțe.
DashDot	Trasează o linie alcătuită dintr-o serie de alternanțe liniuță-punct.
DashDotDot	Trasează o linie alcătuită dintr-o serie de alternanțe liniuță-punct-punct.
Dot	Trasează o linie alcătuită dintr-o serie de puncte.
Solid	Trasează o linie plină.

Stiluri de umplere cu pensula: Formele ca dreptunghiuri, cercuri, elipse pot fi umplute cu o culoare și un model. Stilul de umplere poate varia de la o pensulă de culoare solidă la careuri pe diagonală.

DENUMIREA STILULUI	DESCRIEREA UMLERII
BDiagonal	Trasează linii pe diagonală de la dreapta-sus la stânga-jos.
Cross	Trasează linii încrucișate orizontale și verticale.
DiagCross	Trasează linii încrucișate diagonale.
FDiagonal	Trasează linii pe diagonală de la stânga-sus la dreapta-jos.
Horizontal	Trasează linii orizontale.
Vertical	Trasează linii verticale.
Solid	Trasează linii pline.

Ex: Creăm o pensulă hașurată cu un stil de hașurare încrucișat. Se stabilește culoarea de prim plan la negru.

```
CBrush br(HS_CROSS, RGB(0,0,0));  
dc.FillRect(rect,&br);
```

Sistemul de coordonate

Multe limbaje de la început au folosit cele mai mici elementele de pe ecran posibile pentru a descrie coordonatele de ecran și scalele de desen. **ACESTE ELEMENTE, NUMITE PIXELI, SUNT, DE OBICEI, MĂSURATE DE LA PARTEA SUPERIOARĂ ȘI DIN PARTEA STÂNGĂ A ECRANULUI.**

Diferitele standarde grafice au mărit rezoluția sau numărul de pixeli în oricare dintre direcțiile date.

De exemplu, monitoarele **CGA** au avut rezoluția de culoare de 320 x 200 pixeli, care reprezintă 320 de pixeli, începând cu 0 în stânga și terminând cu 319 în dreapta, respectiv 200 de pixeli, pornind cu 0 în partea superioară și terminând cu 199 în partea inferioară (de jos) a ecranului.

VGA a crescut rezoluția la 640 x 480, iar acum **SVGA** și alte standarde au făcut un salt cantitativ. Când Microsoft a proiectat sistemul de operare Windows, a decis să-l facă "independent de dispozitiv". Astă înseamnă că atunci când dorești un cerc cu o rază fixată, el va apărea la fel pe monitoarele CGA și VGA, fără programare suplimentară, care să suprasolicite programatorul. Implicit, sistemul de coordonate utilizat în limbajul Visual C++ pentru grafică este sistemul de coordonate în pixeli descris mai sus.

Pentru investigarea valorilor coordonatelor în diverse locații pe suprafața unui formular creat în Visual C++ se procedează astfel:

- Se creează un formular în Visual C și pe acesta se plasează 2 etichete (*Static Text*), ca în figura 6.

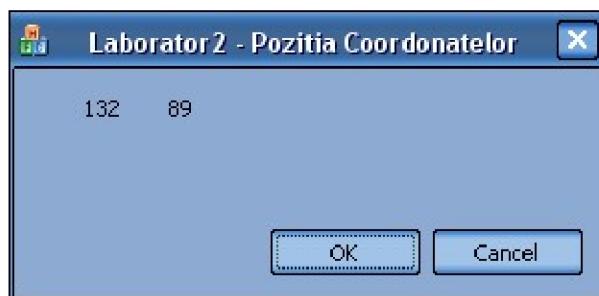


Figura 6. Coordonate mouse

- Se adaugă un eveniment *MouseMove*, scopul fiind acela de a returna coordonatele X și Y ale pozițiilor mouse-ului către cele 2 controale *Static Text*.
- Porțiunea de cod a programului care realizează acest lucru este următoarea:

```
void CLaborator2Dlg::OnMouseMove(UINT nFlags, CPoint point)
{
    // TODO: Add your message handler code here and/or call default

    label_X.Format(L"%d", point.x);
    label_Y.Format(L"%d", point.y);

    //afisare valori memorate în label_X și label_Y pe ecran
    UpdateData(FALSE);
    CDialog::OnMouseMove(nFlags, point);
}
```

Pe măsură ce mouse-ul este mișcat în suprafața client a formularului, valorile X și Y sunt raportate la cele 2 controale Label.

În continuare se pune pe formular un control de tip button cu numele "Afisare Mesaj". Butonul trebuie să prezinte utilizatorului un mesaj atunci când este apăsat (Figura 7).

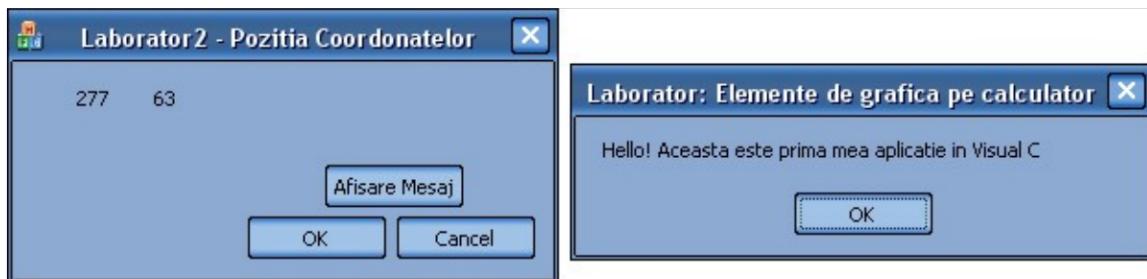


Figura 7. Afisare mesaj

- Se adaugă un eveniment *ClickedButton*, scopul fiind acela de a afișa mesajul în momentul în care s-a apăsat butonul.
- Porțiunea de cod a programului care realizează acest lucru este următoarea:

```
void CLaborator2Dlg::OnBnClickedButton1()
{
    // TODO: Add your control notification handler code here
    MessageBoxA(NULL, "Hello! Aceasta este prima mea aplicatie in
                    Visual C", "Laborator: Elemente de grafica pe
                    calculator", MB_OK);
}
*****
```

În continuare propunem dezvoltarea aplicației astfel încât să urmărească mutarea mouse-ului pe monitor ca în figura 8.



Figura 8. Afisare poziție mouse

Se va afișa poziția mouse-ului, prin aprinderea pixelilor corespunzători, numai dacă este apăsat butonul stânga al acestuia. Deplasarea mouse-ului este adusă la cunoștința aplicațiilor printr-un nou mesaj WM_MOUSEMOVE. Adăugarea unei noi funcții de tratare pentru acest mesaj se face la fel ca la tratarea butoanelor mouse-ului.

Porțiunea de cod a programului care realizează acest lucru este următoarea:

```
// se verifică mai întâi dacă a fost apăsat butonul stâng al mouse-ului
if ((nFlags & MK_LBUTTON) == MK_LBUTTON)
{
    CClientDC dc(this);

    // trasarea unui pixel
    dc.SetPixel(point.x, point.y, RGB(255, 0, 0));
}
```

Așa cum se poate vedea din rularea aplicației de mai sus, pentru a desena o linie continuă trebuie să mutăm mouse-ul foarte lent. Pentru a trasa o linie rapid se procedează astfel:

- ✓ Se selectează clasa CLaborator2Dlg
- ✓ Se execută clic-dreapta pe această clasă și, din meniu, se selectează opțiunea: Add Member Variable,
- ✓ Mai departe se completează ca în figura 9.

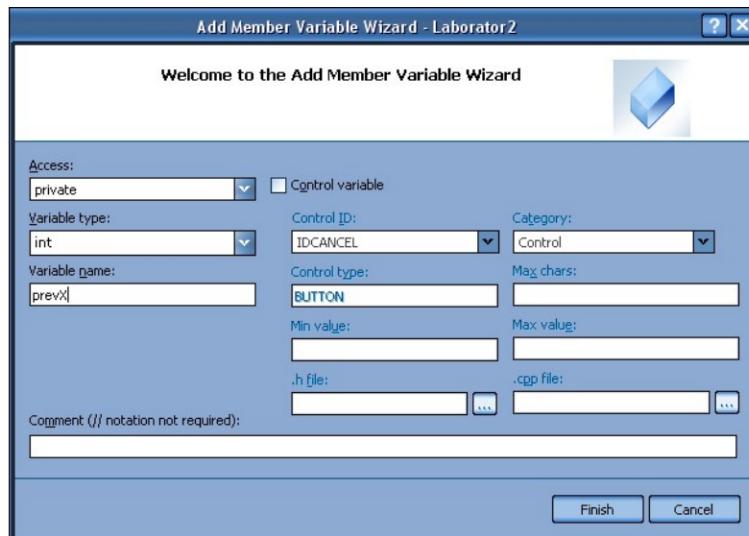


Figura 9. Adăugare variabile

- ✓ Se repetă pașii de mai sus pentru o a doua variabilă numită: prevY.
- ✓ După adăugarea variabilelor se modifică programul ca în listingul de mai jos:

```
//crearea unui nou creion
CPen myPen(PS_SOLID, 5, RGB(255,255,0));

//se verifică dacă butonul stâng al mouse-ului a fos apasat
if ((nFlags & MK_LBUTTON) == MK_LBUTTON)
{
    CClientDC dc(this);

    //utilizare creion
    dc.SelectObject(&myPen);

    // trasarea unei linii de la un punct la altul
    dc.MoveTo(prevX, prevY);
    dc.LineTo(point.x, point.y);

    // salvare
    prevX = point.x;
    prevY = point.y;
}
```

Se observă că desenarea se face începând din margini (vezi figura 10), întotdeauna. Motivul pentru care nu începe desenarea din locul de unde a fost butonul stâng al mouse-ului apăsat îl constituie faptul că nu au fost inițializate variabilele : **prevX, prevY**.



Figura 10. Desenare traseu mouse

Pentru a înlătura acest dezavantaj se parcurge următorii pași:

- ✓ Se adaugă o funcție pentru WM_LBUTTONDOWN prin intermediul ClassWizard,
- ✓ Pentru funcția creată se adaugă codul de mai jos:

```
void CLaborator2Dlg::OnLButtonDown(UINT nFlags, CPoint point)
{
    prevX=point.x;
    prevY=point.y;
    CDialog::OnLButtonDown(nFlags, point);
}
```



Figura 11. Desenare traseu mouse

În continuare ne propunem ca în formular să afișăm un ceas analogic și digital (Figura 12). În proiectul creat anterior, în meniul *Resource View – Dialog – Resource Symbol*, se selectează submeniul *New Symbol* și se adaugă un timer al cărui identificator este: ID_COUNT_TIMER1. Pentru pornirea și oprirea timerului creat se folosesc două butoane "Start Ceas" și "Stop Ceas, iar funcțiile pentru pornire și oprire timer sunt următoarele:

```
//start timer
SetTimer(ID_COUNT_TIMER1, 500, NULL);

//stop timer
KillTimer(ID_COUNT_TIMER1);
```

Se adaugă o variabilă de tip *CRect* numită *rect* ce va menține poziția unde se va afișa ceasul analogic pe formular. În funcția *OnInit()* se initializează variabila astfel:

```
rect.left = 100;
rect.top = 50;
rect.right = 200;
rect.bottom = 150;
```

Tot în funcția *OnInit()* se initializează și variabila *radian*.

```
radian = pi/180;
```

Mai departe în funcția *OnPaint()*, pe ramura de else (deci nu când se desenează formularul) se desenează ceasul analogic:

```
.....
else
{
    CClientDC dc(this);
    CBrush br(HS_CROSS, RGB(0,0,0));
    dc.FillRect(rect,&br);
    dc.Ellipse(rect);

    dc.SetPixel(rect.left+50, (rect.top+rect.bottom)/2,
               RGB(255,0,0));
    CDialog::OnPaint();
}
```

În funcția atașată timer-ului se citește ora curentă și se afișează într-un *textBox* și apoi se calculează pozițiile liniilor de la ore, minute, secunde.

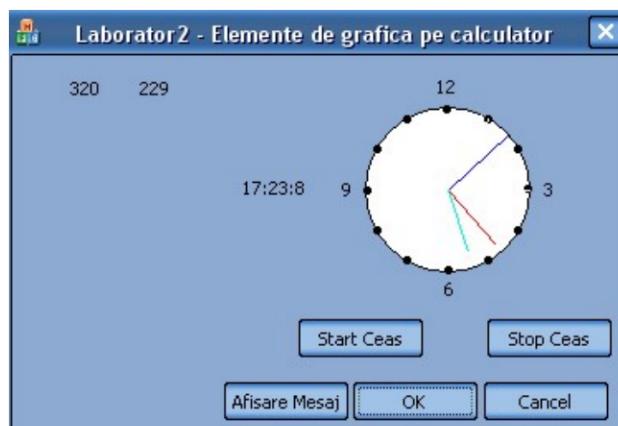


Figura 12. Afisare ceas analogic și digital în Visual C

3. Desfășurarea lucrării:

1. Se va citi breviarul teoretic. Se atrage atenția asupra faptului că aceste informații vor fi folosite și la desfășurarea altor lucrări de laborator.
2. Se vor studia exemplele prezentate pe parcursul acestui laborator urmărind efectul rulării acestora și buna lor execuție.
3. Se va încerca găsirea altor posibilități de soluționare a soluțiilor propuse.
4. Cum se poate modifica caseta de dialog apărută la apăsarea butonului „Afisare Mesaj”, fără a schimba titlul aplicației?
5. Cum se poate modifica numele unui buton?
6. Cum se atașează o funcție la apăsarea unui buton?

7. Cum se mapează o variabilă peste un control din formular?
8. Modificați programul prezentat în laborator astfel încât să se deseneze pe monitor traseul mouse-ului cu culoare roșie atunci când butonul dreapta al acestuia este ținut apăsat.