7. GRAFICĂ SUB WINDOWS

Prin *program windows* înțelegem un program care a fost realizat pentru a se executa sub sistemul de operare **Windows**. Avem în vedere sistemele de operare **Windows 95, 98**, **NT, Windows 2000** sau **Windows XP**.

Aceste programe pot fi realizate folosind compilatoarele firmei **Borland** începând cu versiunea 3.1 sau ale firmei **Microsoft**. În cele ce urmează, vom avea în vedere compilatoarele firmei **Borland**, versiunea 3.1, precum şi cele ale firmei Microsoft şi anume **Microsoft Visual C++** versiunile 5.0 şi 6.0.

La scrierea *programelor windows* se pot utiliza o serie de biblioteci.

O bibliotecă utilizată frecvent este biblioteca **Application Programming Interface** (API).

În cazul compilatoarelor Microsoft Visual C++ vom avea în vedere, pe lângă biblioteca API, şi biblioteca **Microsoft Foundation Class Library** (MFC).

7.1. Introducere în programarea windows

Prin *programare windows* înțelegem scrierea de *programe windows***,** adică programe care se execută sub sistemul de *operare windows*.

Programele windows se mai numesc şi *aplicații windows* sau mai scurt *aplicații*.

Într-o primă fază ne vom ocupa de scrierea programelor windows utilizând biblioteca API. Ulterior vom avea în vedere şi biblioteca MFC.

7.1.1. Structura unui program windows

În principiu, un program windows conține două funcții. Una dintre acestea este *funcția principală*. Ea are numele *WinMain* şi are prototipul:

```
int PASCAL WinMain(HINSTANCE hInstance,
                   HINSTANCE hPrevInstance,
                   LPSTR lpszCmdLine,int iCmdShow);
```

Menționăm că acest prototip este valabil atât pentru compilatoarele Borland cât şi pentru compilatoarele Visual C++ până la versiunea 4.0 inclusiv.

În cazul compilatoarelor Visual C++ versiunile 5.0 şi 6.0, în loc de modificatorul PASCAL, de obicei, se utilizează modificatorul WINAPI, deci, prototipul funcției principale ar putea fi următorul:

```
int WINAPI WinMain(HINSTANCE hInstance,
                   HINSTANCE hPrevInstance,
                   LPSTR lpszCmdLine,int iCmdShow);
```

Cealaltă funcție are prototipul:

```
LRESULT CALLBACK WndProc(HWND,UINT,WPARAM,LPARAM);
```

Acest prototip este comun atât compilatoarelor Borland cât şi compilatoarelor Microsoft.

Pentru a utiliza funcțiile prezente în biblioteca API vom include fişierul *windows.h*.

Programele windows au ataşate câte o *fereastră*, la lansarea lor în execuție.

Crearea ferestrei aplicației se realizează în prima parte a funcției principale. O serie de date cu privire la fereastră se înregistrează în windows. În acest scop, se definesc componentele unei structuri de tip WNDCLASS. În cazul compilatoarelor Visual C++ 5.0 şi 6.0, în loc de tipul WNDCLASS se poate utiliza tipul extins, WNDCLASSEX.

După definirea componentelor acestei structuri, se înregistrează structura respectivă apelând funcția *RegisterClass*.

Aceste date se completează apelând o altă funcție şi anume *CreateWindow*. Prin intermediul parametrilor acestei funcții se poate defini textul de pe bara de titlu a ferestrei, dimensiunea ei, etc.

Această funcție returnează *handle* - ul ferestrei create prin înregistrarea datelor membru ale structurii WNDCLASS şi prin apelul funcției *CreateWindow*.

Prin *handle* înțelegem o variabilă care identifică un obiect. În cazul de față, funcția *CreateWindow* returnează un identificator pentru fereastra aplicației. Acest *handle* va fi folosit în continuare pentru a ne referi la fereastra aplicației.

În continuare, se afişează fereastra aplicației. În acest scop, se apelează funcțiile *ShowWindow* şi *UpdateWindow*. Prima funcție setează modul de afişare pentru fereastra aplicației. Ea are doi parametri. Primul este handle - ul ferestrei (returnat de funcția *CreateWindow*), iar cel de al doilea defineşte modul de afişare al ferestrei.

La primul apel al funcției *ShowWindow*, acest parametru este chiar ultimul parametru al funcției *WinMain*, adică în cazul de față ar fi *iCmdShow*. Alte valori ale acestui parametru ar putea fi: SW_HIDE - ascunde fereastra, SW_MAXIMIZE - maximizează fereastra, SW_MINIMIZE - minimizează fereastra, SW_SHOWNORMAL - activează şi afişează fereastra la dimensiunea şi poziția inițială, etc.

Funcția *UpdateWindow* are un singur parametru, handle - ul ferestrei. Această funcție pune la zi zona client a ferestrei aplicației generând un mesaj WM_PAINT. Prin apelul ei se afișează fereastra aplicației.

După afișarea ferestrei aplicației, aplicația va intra într-o stare de "aşteptare de eveniment". Prin *evenimente* înțelegem diferite acțiuni care apar ca răspuns la acționarea unei taste, acționarea unui buton al mouse - ului, scurgerea unui interval de timp, etc. Evenimentele sunt preluate de *windows* şi, în principiu, ele sunt puse într-o *coadă*, într-o formă codificată, numite *mesaje*. Mesajele pot fi generate şi prin intermediul unei funcții.

Aplicațiile intră în aşteptare de mesaje executând un ciclu, numit *bucla de mesaje*. Acest ciclu, apelează funcția *GetMessage* care preia un mesaj din *coada de mesaje* şi îl plasează într-o structură de tip MSG. Această funcție returnează o valoare diferită de zero, exceptând cazul în care a întâlnit mesajul WM_QUIT.

De obicei, bucla de mesaje este de forma:

```
while(GetMessage( ... ))
{
...
}
```

Se observă că instrucțiunea *while* se execută atât timp cât nu se întâlneşte mesajul WM_QUIT în coada de mesaje. Mesajul WM_QUIT apare ca urmare a închiderii ferestrei aplicației.

În corpul lui *while* se pot apela diferite funcții. Dintre acestea, două funcții se apelează frecvent şi anume *TranslateMessage* şi *DispatchMessage*.

Dacă funcția *GetMessage* a plasat mesajul preluat din coada de mesaje în structura *msg* de tip MSG, atunci aceste funcții se apelează având ca parametru adresa acestei structuri, adică:

```
while(GetMessage( ... ))
{
        TranslateMessage(&msg);
        DispatchMessage(&msg);
}
```

Funcția *TranslateMessage* traduce mesajele cheilor virtuale în mesaje caracter pentru acele chei care se mapează în caractere ASCII.

Funcția *DispatchMessage* trimite mesajul tradus prin *TranslateMessage* la cea de a doua funcție a aplicației, funcție care are rolul de a prelucra mesaje. La ieşirea din funcția *DispatchMessage* se revine în Windows şi acesta apelează funcția de prelucrare a mesajului. Această funcție, de obicei, se numeşte *procedura ferestrei* aplicației (*window procedure*).

După tratarea unui mesaj prin procedura ferestrei, se revine iarăşi în *windows*.

Să observăm că procedura ferestrei nu se apelează direct de către programator. Ea este apelată de către *windows*.

V-200
Limbajele C şi C++ pentru începători
volumul IV Probleme de Optimizare şi Grafică
Liviu Negrescu, Lavinia Negrescu

PARTEA I-a PROBLEME DE OPTIMIZARE

PARTEA II-a PROBLEME DE GRAFICĂ

5. GRAFICĂ ÎN PLAN

5.1. Axe de coordonate
5.2. Afişarea punctelor prin coordonate
5.3. Afişarea segmentelor de dreaptă
5.3.1. Puncte care împart un segment dat într-un raport dat
5.3.2. Locuri geometrice
5.3.2.1. Mediatoarea unui segment
5.3.2.2. Bisectoarea unui unghi
5.4. Graficul funcţiilor continue de o variabilă
5.5. Poligoane regulate

6. GRAFICĂ ÎN SPAŢIU

6.1. Axe de coordonate
6.2. Conversia coordonatelor din 3 dimensiuni în sistemul implicit
6.3. Reprezentarea punctelor în spaţiul cu 3 dimensiuni
6.4. Afişarea segmentelor de dreaptă
6.5. Afişarea unei prisme
6.6. Afişarea unei piramide
6.7. Afişarea unui trunchi de piramidă
6.8. Afişarea unui cilindru
6.9. Afişarea unui con
6.10. Afişarea unui trunchi de con
6.11. Afişarea unei sfere

7. GRAFICĂ SUB WINDOWS

7.1. Introducere în programarea windows
7.1.1. Structura unui program windows
7.1.2. Notaţii
7.2. Lansarea compilatoarelor
7.2.1. Lansarea compilatorului Microsoft Visual C++, versiunea 5.0
7.2.2. Lansarea compilatorului Microsoft Visual C++, versiunea 6.0
7.2.3. Lansarea compilatorului firmei Borland, versiunea 3.1
7.3. Programe windows care utilizează biblioteca API
7.3.1. Afişarea unui text într-o fereastră Windows
7.3.1.1. Crearea funcţiei principale folosind compilatoarele firmei Borland
7.3.1.2. Crearea funcţiei principale folosind compilatoarele Visual C++, versiunile 5.0 şi 6.0
7.3.1.3. Crearea funcţiei procedura fereastră
7.3.2. Afişarea de puncte, segmente, poligoane şi elipse
7.3.2.1. Afişarea de puncte
7.3.2.2. Afişarea segmentelor de dreaptă
7.3.2.3. Afişarea poligoanelor
7.3.2.4. Afişarea dreptunghiurilor

7.3.2.5. Afişarea elipselor
7.3.3. Colorarea figurilor folosind funcţii din biblioteca API
7.3.4. Axe de coordonate
7.3.5. Afişarea punctelor prin coordonate
7.3.6. Afişarea graficelor funcţiilor continue de o variabilă
7.3.7. Axe de coordonate în plan cu intervale de valori date
7.3.8. Utilizarea axelor de coordonate cu intervale de valori date
7.4. Programe windows care utilizează biblioteca Microsoft Foundation Class library (MFC)
7.4.1. Crearea aplicaţiilor de tip SDI folosind instrumentul AppWizard
7.4.2. Afişarea unui text într-o fereastră Windows folosind biblioteca MFC
7.4.3. Afişarea de puncte, segmente, poligoane şi elipse
7.4.4. Sisteme de coordonate
7.4.5. Grafice de funcţii plane
7.4.6. Utilizarea modului de mapare MM_ISOTROPIC
7.4.7. Utilizarea modului de mapare MM_ANISOTROPIC
7.4.8. Utilizarea modului de mapare MM_TWIPS