Laborator 8 – Utilizarea Mediului C++ Builder

Tema 8.1

Creați un proiect în C++ Builder, îl salvați și apoi îl rulați.

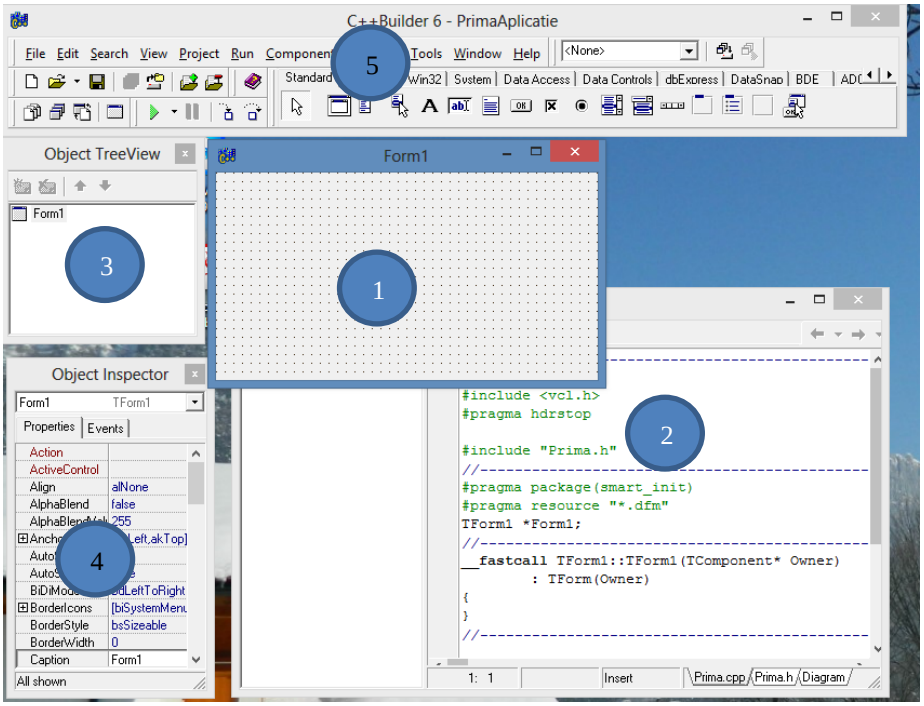
Consideraţii teoretice 8.1

C++ Builder este un mediu (o unealtă) care ajută la dezvoltarea de aplicații proprii în limbajul C++ putându-se crea astfel ușor aplicații cu interfață pentru sistemul de operare WINDOWS. Pentru a putea crea o aplicație windows trebuie creat prima dată un proiect. Proiectele sunt colecții de fișiere sursă (cpp) care sunt compilate pe rând și din care este realizat un singur fișier executabil. De aceea în cadrul unui proiect trebuie să existe o singură funcție *main* (sau *WinMain* pentru windows).

Pentru crearea unei aplicații pentru windows cu interfață în C++ Builder se intră în meniul ”*File->New->Application*”. Se va crea automat o aplicație care conține o fereastră (în c++ Builder se numește FORM) goală perfect funcțională. Pentru a putea fi rulată această aplicație trebuie în prealabil să fie salvată.

Pentru salvarea întregului proiect se folosește comanda „*File->Save Project As*”. Pentru un proiect nou se recomandă prima dată salvarea întregului proiect. La execuția acestei comenzi se cere prima dată calea și un nume pentru un fișier cu extensia cpp (Unit1.cpp). Acest fișier conține codul scris deja pentru forma noastră. Se va specifica un nume pentru acest fișier și se va selecta exact calea unde se dorește salvat proiectul (Pentru această temă recomand numele „Prima.cpp”). În folderul specificat vor fi salvate 2 fișiere cu numele specificat, unul cu extensia „.cpp” și unul cu extensia „.h”. După salvarea formei se va cere specificarea numelui proiectului. Pentru salvarea proiectului în folderul specificat vor fi create alte două fișiere unul cu extensia .cpp și unul cu extensia .bpr. Pentru tema aceasta recomand numele „PrimaAplicație.bpr”.

ATENȚIE: Nu specificați pentru proiect același nume cu numele specificat pentru formă deoarece se va rescrie fișierul cu extensia .cpp de la forma fără nici un avertisment. După salvarea aplicației fereastra ar trebui să fie asemănătoare cu cea din figura 1.1.



Figură 1.1 - Interfața C++ Builder

Câteva informații despre ferestrele care au apărut în aplicația c++ Builder creată.

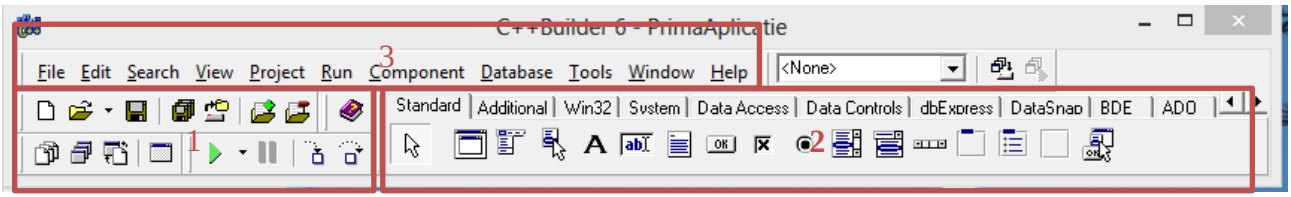
Fereastra (1) este fereastra aplicației create. Pe această fereastră se va dezvolta aplicația care se dorește realizată prin plasarea și configurarea diferitelor componente.

Fereastra (2) este fereastra în care se găsește codul generat deja de C++ Builder pentru aplicația noastră. În această fereastră noi vom continua scrierea de cod pentru dezvoltarea aplicației. În paralel cu noi C++ Builder va continua generarea de cod în aceste fișiere de aceea este recomandat să se păstreze o logică și o structură cât mai bună în fișierele de cod. Orice forma din proiect va avea 2 fișiere de cod atașate, în cazul aplicației noastre „Prima.cpp” și „Prima.h”. Pentru trecerea de la fișierul .cpp la fișierul .h se pot folosi butoanele din partea de jos-centru a acestei ferestre. ATENȚIE: Dacă se închide această fereastră se va închide tot proiectul.

Fereastra (3) numită și „Object TreeView” este fereastra în care se vor găsi toate componentele care le-am folosit în aplicația noastră. Deocamdată în aplicație avem o singură componentă – forma numită automat de C++ Builder „Form1”. Toate componentele existente in C++ Builder și care pot fi utilizate în aplicația noastră sunt afișate în fereastra 5 din figura 1.1. Câteva din aceste componente vor fi prezentate succint în acest laborator. Pentru informații detaliate despre componente se poate consulta help-ul de la C++ Builder.

Fereastra (4) din figura 1.1 numită și „Object Inspector” este una din cele mai importante ferestre de pe ecran. Ea conține toate trăsăturile pentru componenta curent selectată. În acest moment în această fereastră se găsesc trăsăturile (proprietățile) formei curente (deoarece forma este componenta curent selectată momentan). Majoritatea proprietăților unei componente au nume intuitiv. Pe parcursul laboratorului vom detalia rolul unora dintre aceste proprietăți. Anumite proprietăți diferă de la o componentă la alta. De exemplu proprietatea „Caption” pentru o formă se referă la numele care va apărea pe formă în partea de sus a acesteia.

Fereastra (5) din figura 1.1 este fereastra care ține de C++ Builder. Este fereastra care conține comenzile de la C++Builder pentru aplicația noastră. Această fereastră conține câteva butoane pentru comenzile standard cele mai uzuale. Această fereastră are 3 părți importante prezentate în figura 1.2.



Figură 1.2 – Meniul C++ Builder

În partea (1) din figura 1.2 sunt butoane (shortcut-uri) pentru comenzile standard cele mai utilizate în C++ Builder. Pe prima linie sunt butoane standard din windows iar butoanele de pe a doua-a linie au următoarele funcții:

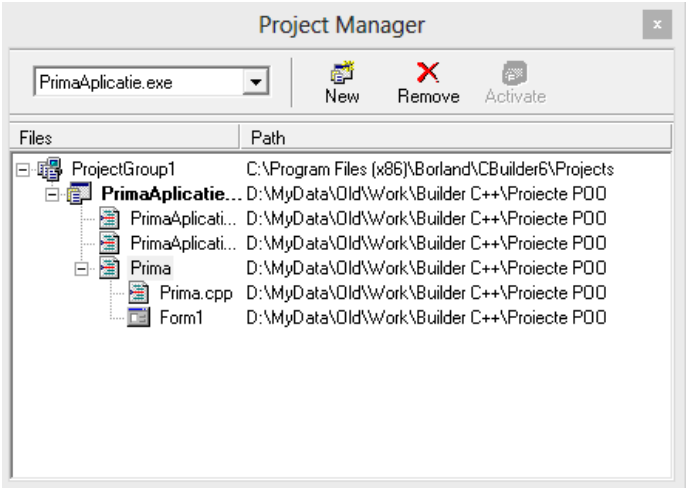
- Butonul  permite afișarea unei ferestre în care se găsește o listă cu toate fișierele cpp din cadrul proiectului – putând fi astfel deschis în fereastra de editare orice fișier din cadrul proiectului. Pe perioada dezvoltării aplicației nu trebuie să fie deschise toate fișierele din proiect.
- Butonul  – va afișa o fereastră cu toate fomele (ferestrele) existente în cadrul proiectului. Se poate deschide astfel orice formă din proiect. Pe perioada dezvoltării aplicației nu trebuie să fie deschise toate formele din proiect.

- Butonul  - permite trecerea de la fereastra de cod la forma aferentă codului sau invers.
- Butonul  – permite inserarea unei noi forme în cadrul proiectului. La apăsarea acestui buton se va crea o formă nou și se va introduce în cadrul proiectului fișierele cpp și h aferente acestei forme. Atenție că ștergerea unei forme din cadrul proiectului este mai dificilă. Prin simpla închidere a formei sau a fișierului cpp corespunzător, forma respectivă nu se șterge din proiect. Pentru ștergerea unei forme din proiect se selectează meniul *Project->Remove from Project…* și se selectează din listă forma care se dorește a fi ștearsă.
- Butonul  – permite rularea aplicației curente. Dacă aplicația nu a fost compilată în prealabil se va compila aplicația și apoi se va rula. Înainte de rularea unei aplicații nu uitați să salvați aplicația respectivă folosind butonul de *Save* sau de *Save All*.
- Butonul  – activ în timpul rulării aplicației, permite oprirea aplicație la linia curentă și continuarea execuției ei pas cu pas (în limbaj de asamblare). Acest buton nu închide aplicația curentă ci doar comută aplicația în modul debug. Pentru oprirea aplicație se pot folosi comenzile standard din windows care opresc o aplicație (de exemplu butonul cu simbolul  din dreapta sus de la fereastră) sau dacă aceasta s-a blocat se poate folosi comanda din meniul de la c++ Builder „*Run->ProgramReset*” sau prescurtat „CTRL+F2”. Nu recomand închiderea aplicației folosind comanda Ctrl+Alt+Del deoarece se va închide și C++ Builder fără salvarea aplicației.

În partea (2) din figura 1.2 se găsește Toolbar-ul cu toate componentele din C++ Builder care pot fi folosite pentru dezvoltarea aplicației. De exemplu pentru a scrie un text pe ecran se poate folosi componenta „Label” din tabul *Standard*, pentru a adăuga un buton pe forma noastră se poate folosi componenta „Button” din tabul *Standard*. Dacă se stă puțin cu mouse-ul deasupra unei componente se va afișa numele acesteia. De asemenea vom detalia mai jos o serie dintre aceste componente.

În partea (3) din figura 1.2 este meniul cu toate comenzile din c++ Builder. Majoritatea sunt meniuri standard windows, nu vom discuta aici despre toate doar câteva considerate mai importante.

Meniul „*View->Project manager*” – comandă care permite vizualizarea tuturor fișierelor incluse în proiectul curent. În urma acestei comenzi va apărea o fereastră de forma celei din figura 1.3.



Figură 1.3 – Project Manager

Fișierele conținute într-un proiect sunt următoarele:

- „Prima.cpp” conține codul sursă pentru fereastra (Forma) din aplicație. Fișierul h conține clasa nou definită pentru fereastra noastră.

- Fișierul „PrimaAplicație.cpp” – conține funcția „WinMain” pentru aplicația curentă. Acest fișier este generat și actualizat automat de C++ Builder de acea nu se recomandă modificarea acestui fișier decât în caz de strictă necesitate.

Nu uitați să rulați aplicația creată și vedeți ce facilități oferă deja forma existentă.

Indicaţii 8.1

- ⇨ Atenție la salvarea aplicației, la numele fișierului .CPP și numele proiectului. Pentru numele proiectului nu specificați același nume ca și pentru numele formei. Se va rescrie fișierul formei fără să ne atenționeze că se pierde definitiv acel fișier.
- ⇨ Pentru oprirea unei aplicații în cazul în care aceasta s-a blocat se poate folosi comanda din meniul „*Run->Program Reset*” sau shortcut-ul „*CTRL+F2*”.

Tema 8.2

Analizați codul scris deja de C++ Builder din toate fișierele din cadrul proiectului.

Considerații teoretice 8.2

În fișierul „Prima.h” se găsește definită o clasă care va conține forma din aplicația noastră. Clasa „*TForm1*” este clasa noastră și moștenește public toate proprietățile clasei „*TForm*” care este o clasă din C++ Builder. În interiorul clasei avem definit doar constructorul clasei. De obicei acest fișier este completat automat de C++ Builder pe parcursul utilizării de noi componente. Dacă se dorește introducerea de noi membrii în clasă se recomandă scrierea acestora în zonele recomandate prin comentariul „*User declaration*”.

În fișierul „Prima.cpp” se găsește codul clasei respective. Deocamdată avem implementat constructorul pentru clasa noastră care nu face altceva decât apelează constructorul cu un parametru din clasa de bază – necesar pentru inițializarea formei.

Deschideți și fișierul „PrimaAplicație.cpp” (vezi tema anterioară). Acesta conține implementarea funcției „*WinMain*” în contextul celor discutate la Tema1.1. Codul aferent conține cele trei etape din cadrul unei aplicații windows – partea de inițializare, partea de creare interfață și partea de așteptare/executare comenzi.

Structura „*try/catch*” este obligatorie a se folosi în orice aplicație windows. Ideea este că îi spunem sistemului de operare să încerce să execute instrucțiunile din blocul „*try*”. Dacă pe parcursul execuției aplicației una dintre instrucțiuni creează probleme și se blochează aplicația atunci sistemul de operare va opri execuția instrucțiunilor din blocul „*try*” și va continua aplicația prin executarea instrucțiunilor din blocul „*catch*”. Această structură „*try/catch*” este obligatorie în toate aplicațiile windows de aceea pentru a rezolva simplu problema C++ Builder a inclus din start toată aplicația noastră într-o astfel de structură. Tratarea este standard, indiferent de tipul problemei apărute se închide aplicația și se afișează unui mesaj cu eroarea apărută. Dacă se dorește, se poate trata și local excepțiile care apar pentru a nu duce la închiderea bruscă a aplicației prin folosirea de blocuri try/catch local acolo unde se dorește.

Tema 8.3

Modificați proprietățile formei astfel încât această și aibă textul din partea de sus „Prima Fereastră” iar numele variabilei folosite în aplicație să fie „fMain”.

Considerații teoretice 8.3

În „*Object Inspector*” se găsesc caracteristicile componentei curent selectate.

Caracteristica „*Caption*” se referă la textul afișat de componentă, text care nu este editabil pe parcursul rulării aplicației ci doar în partea de design (dezvoltare).

Caracteristica „*Name*” reprezintă numele cu care vom folosi variabila respectivă în cadrul aplicației atunci când scriem cod. În momentul în care modificăm această caracteristică C++ Builder va modifica automat numele în toate locațiile în care l-a scris el. Dacă am scris și noi cod în care am folosit numele vechi al variabilei în aceste locuri numele nu va fi modificat automat. De aceea se recomandă schimbarea numelui unei componente (variabile) imediat după ce am inserat componenta (variabila) în aplicația noastră. Pentru o mai bună înțelegere a variabilelor se recomandă ca atunci când se redenumește variabila să se specifice la începutul numelui acesteia câteva caractere care să sugereze tipul componentei.

Tema 8.4

Să se introducă pe formă un buton pe care să scrie textul ”Exit”, să aibă numele ”btnExit” și la click pe acest buton să se închidă aplicația.

Considerații teoretice 8.4

Pentru a insera un buton se folosește toolbar-ul de componente în tabul „*Standard*”. Se face clic pe componenta dorită din toolbar și apoi se face clic pe fereastră în locul unde se dorește inserat butonul. Acesta poate fi modificat și repoziționat prin drag-and-drop.

Pentru atașarea unui eveniment de clic pe buton se merge în „*Object inspector*” și se selectează tabul „*Events*” (atenție să fie selectat butonul pentru că altfel selectați evenimentele pentru Formă). Acolo va apărea o listă cu toate evenimentele care pot fi activate pentru componenta curent selectată. Pe noi ne interesează evenimentul „*OnClick*”. Se va face dublu-click în dreptul acestui eveniment în locația goală. Automat C++ Builder va insera o metodă pe care o va atașa de acest eveniment. Deci de câte ori noi vom facem click pe acel buton se va apela automat codul acestei metode (funcții). Urmează ca în această metodă să scriem funcția standard din c++ care ne închide o aplicație (care este funcția?). O altă modalitatea de a crea (sau ajunge) la funcția creată pentru evenimentul onClick pe un buton este să facem dublu-click pe butonul respectiv. Rulați aplicația să vedeți dacă și funcționează.

Indicații 8.4

- ⇨ Pentru modificarea proprietăților butonului a se vedea tema 1.4
- ⇨ Funcția pentru închiderea aplicației în c++ este „exit(0)”.

Tema 8.5

Introduceți pe fereastră o componentă de tip „*Label*” în care scrieți textul „semigrupa” și variabila atașată va avea numele „lText” și încă un buton pe care scrieți textul „Unu” și numele variabilei va fi „btnUnu”. Pentru buton evenimentul de click pe acesta va produce schimbarea textului afișat de componenta Label cu textul afișat de buton.

Considerații teoretice 8.5

Pentru a accesa în cod o anumită proprietate a unei componente trebuie să amintim că aceste componente nu sunt altceva decât obiecte de tipuri mai complexe, deci se accesează ca și un membru dintr-o clasă(structură). Dacă ne uităm în fișierul „Prima.cpp” vom vedea că toate obiectele care le-am inserat până acum sunt definite de tip pointer, deci membri acestora se vor accesa corespunzător.

De exemplu pentru a modifica textul afișat de componenta noastră „lText” în timpul execuției aplicației se poate scrie codul:

```
fMain->lText->Caption = ”Modificat”;
```

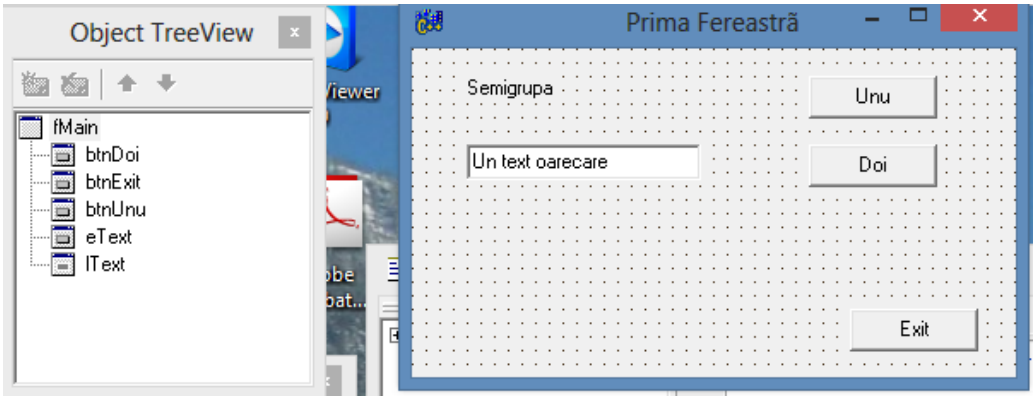
În cadrul unei metode specificarea obiectului de tipul clasei din care face parte metoda (*fMain*) este opțională acesta fiind specificat automat de compilator prin cuvântul cheie *this*.

Tema 8.6

Introduceți pe fereastră o componentă de tip buton cu numele „btnDoi” și textul afișat să fie „DOI” și o componentă de tip Edit cu numele „eText” în care utilizatorul să poată scrie orice text dorește în timpul rulării aplicație. La apăsarea butonului „Doi” textul scris în componenta „eText” să fie copiat în componenta „lText”.

Considerații teoretice 8.6

La final interfața aplicației ar fi de felul următor, cuprinzând și fereastra Object TreeView în care se văd definite toate numele de variabile care sunt de tip componente.



Figură 1.4 – Aplicația realizată