Анотації

У статті розглядаються питання побудови телекомунікаційної мережі для проведення дистанційного навчання державних службовців. Висвітлюються питання заочного навчання і використання нових інформаційних технологій при проведенні навчання. Як базова технологія передбачається використовувати віртуальні приватні мережі. Описані основні параметри і переваги таких мереж.

В статье рассмотрены вопросы построения телекомуникационной сети для проведения дистанционного обучения государственных служащих. Рассмотрены вопросы заочного обучения и использования новых информационных технологий при проведении обучения. В качестве базовой технологии предполагается использовать виртуальные частные сети. Описаны основные параметры и преимущества таких сетей.

In the article the questions of construction of telecommunication network are considered for the leadthrough of the controlled from distance teaching of civil servants. The questions of the extra-mural teaching and use of new information technologies are considered during the leadthrough of teaching. As base technology it is assumed to utillize virtual private networks. Basic parameters and advantages of such networks are described.

УДК 681.3

А. Л. ЄРОХІН,
доктор технічних наук, професор,
начальник кафедри інформатики
Харківського національного університету внутрішніх справ
В. О. РОМАНОВ,
магістрант
Харківського національного університету радіоелектроніки

РОЗРОБКА ДОДАТКІВ ЗАСОБАМИ БІБЛІОТЕКИ WXWIDGETS

Постановка завдання. При створенні віконних додатків на будь-якій платформі використовується певний інструментарій для зручного створення графічних інтерфейсів. З огляду на вирівнювання популярності різних операційних систем, у розробників виникає потреба в крос-платформному інструменті для розробки додатків, включаючи як графічну складову, так і інші елементи узагальненої взаємодії з операційною системою, що не вимагають від розробника переписування коду під час міграції на іншу платформу. Прикладом таких елементів є керування багатопотоковістю, взаємодія через мережу, мультимовні інтерфейси, а також робота з файловою системою.

У даній статті розглядається бібліотека wxWidgets, основною метою якої є створення крос-платформних графічних інтерфейсів. Бібліотека розповсюджується з вільною ліцензією і набуває популярності як серед невеликих колективів розробників, так і серед великих корпорацій. Використання бібліотеки варіюється від наукових досліджень до клієнтських графічних додатків і візуалізації статистики.

Бібліотека розбита на модулі; деякі з них є обов'язковими для використання в будь-якому додатку, що працює з wxWidgets, інші ж допомагають розробникові в конкретних ситуаціях.

До обов'язкових базових модулів належать wxBase і wxCore.

Додаткові модулі надають інтерфейси для роботи з базами даних, мережами (сокетами), потоками, XML (включаючи можливість динамічного створення інтерфейсів з XML-файлів), візуалізацією HTML-документів, базовою роботою з

медіа-компонентами, звуком, а також 3D-графікою (інтеграція з OpenGL).

Інструментарії створення графічного інтерфейсу дозволяють використовувати такі складні компоненти, як календарі, буферизоване полотно, т. зв. «літаючі» панелі з можливістю їх перетягування, сітки (grids), автоматичне створення «майстрів» та спливаючих екранів. Таким чином, необхідно дослідити особливості бібліотеки wxWidgets та розробити принципи практичного використання компонентів бібліотеки.

1. Дослідження особливостей wxWidgets. Розглянемо основні класи бібліотеки wxWidgets з урахуванням їх взаємодії. На рис.1 зображена схема взаємозв'язку між класами бібліотеки.

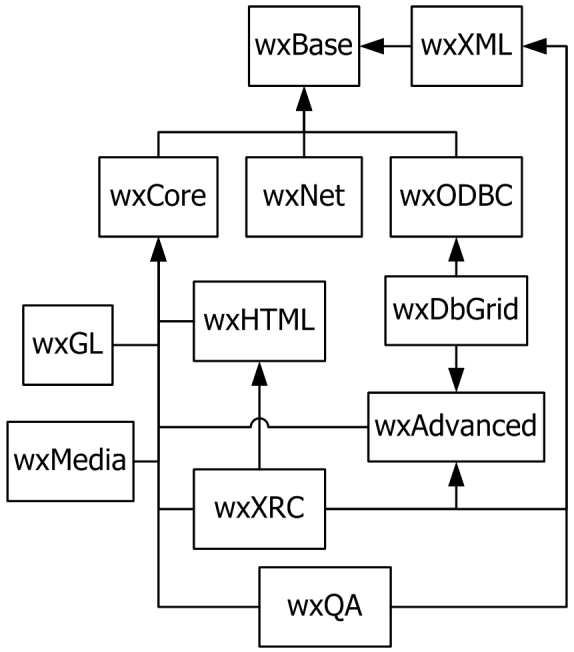

Рис. 1. Схема взаємозв'язку класів бібліотеки wxWidgets

На вершині ієрархії знаходиться бібліотека *wxBase*. Вона містить класи, від яких залежить написаний код, а також абстрактні класи, що відрізняються один від одного залежно від платформи.

У wxWidgets існує бібліотека під назвою *wxCore.* Вона відповідає за GUI, а саме: усі класи й елементи керування, що використовуються в побудові графічних інтерфейсів, успадковані від *wxCore*. Практично всі програми, за винятком консольних додатків, використовують цю бібліотеку.

Бібліотека wxNet містить у собі класи для мережевого доступу:

- *wxSocket* класи (*wxSocketClient*, *wxSocketServer*);
- *wxSocketOutputStream* і *wxSocketInputStream;*
- базово-сокетні IPC класи (*wxTCPServer*, *wxTCPClient* і *wxTCPConnection*);
- *wxURL;*
- *wxInternetFSHandler* (*wxFileSystem* оброблювач).

Бібліотека *wxODBC* містить у собі класи баз даних: *wxDb* і *wxDbTable*. Ця бібліотека прямо зв'язана з бібліотекою *wxBase.*

Бібліотека *wxXML* містить класи для аналізу документів XML. При використанні бібліотеки варто звернути увагу на те, що пізніше вона буде змінена, що породжує проблему несумісності зі старими версіями. Не рекомендується використовувати *wxXML* для створення додатків. Ця бібліотека використовується на рівні аналізу системою XML-ресурсів.

Бібліотека *wxHTML* є HTML–візуалізатором, а також містить набір додаткових класів, таких, як *wxHtmlHelpController, wxBestHelpController* і *wxHtmlListBox.*

Бібліотека *wxQA* орієнтована на зміст додаткових класів для тестування. На даний момент *wxQA* містить лише клас *wxDebugReport* і класи, що його успадковують, але в нових версіях планується розширити можливості *wxQA*.

Бібліотека *wxGl* містить у собі клас *wxGlCanvas* – клас для інтеграції бібліотеки *OpenGL* із бібліотекою *wxWidgets*.

Бібліотека *wxAdvanced* – бібліотека, що містить додаткові або рідко використовувані класи графічних інтерфейсів, такі, як: *wxBufferedDC, wxCalendarCtrl, wxGrid classes, wxLayoutAlgorithm, wxWizard, wxSound, wxSplashScreen, wxTaskBarIcon, wxSashWindow, wxJoystick, wxSashLayoutWindow.*

Бібліотека *wxXRC* містить у собі клас *wxXmlResource,* що забезпечує доступ до XML-файлів у форматі XRC.

Бібліотека *wxMedia* – це множина класів, що належать до мультимедіа. На даний момент ця бібліотека містить тільки клас *wxMediaCtrl*, але в наступних версіях планується розширити можливості бібліотеки.

Бібліотека *wxDbGrid* містить клас *wxDbGridTableBase*, що поєднує *wxGrid* і *wxDbTable*; залежності від *wxODBC* і *wxAdvanced*.

2. Принцип практичної реалізації і застосування wxWidgets. Бібліотека wxWidgets використовує свою ідеологію створення додатків. Принцип створення простого додатка розглянемо на прикладі.

Кожен додаток на wxWidgets визначає свій клас, що походить від *wxApp*. У програмі існує єдиний екземпляр даного класу. Звичайно, цей клас повідомляє функцію *OnInit*, що є вхідною точкою програми (функція *OnInit* є еквівалентом функції *main* мови C++).

Нижче дане мінімально можливе оголошення класу додатка:

```
// повідомляємо клас додатка
classMyApp:publicwxApp
{
public:
// викликається під час старту додатка
virtual bool OnInit();
};
```

Реалізація OnInit створює хоча б одне вікно, зчитує параметри з командного рядка, ініціалізує необхідні для роботи структури даних і виконує інші дії, необхідні для запуску програми. Якщо функція повертає *true*, то wxWidgets запускає петлю повідомлень, якщо це необхідно. Якщо функція повертає *false*, то wxWidgets коректно очищає свої внутрішні структури і завершує роботу програми.

Реалізація функції *OnInit* має такий вигляд:

```
// ініціалізація додатка
bool MyApp::OnInit()
{
// створення головного вікна додатка
MyFrame*frame = newMyFrame(
wx(«MinimalwxWidgetsApp») );
// показ вікна
frame->Show(true);
// запуск петлі повідомлень
return true;
}
```

Представлена реалізація створює екземпляр нового класу вікна MyFrame (клас буде визначений пізніше), показує вікно на екрані і повертає *true*, щоб запустити петлю повідомлень. Головні вікна (такі, як фрейми та діалоги), на відміну від дочірніх, повинні бути показані відразу після створення.

Програмний код, що створює екземпляр класу MyApp, генерується автоматично бібліотекою wxWidgets; слід лише явно вказати тип об'єкта. Тому до реалізації необхідно додати такі рядки:

```
// повідомляє wxWidgets, що потрібно
створити
// об'єкт MyApp
IMPLEMENT_APP(MyApp)
```

Без цього визначення wxWidgets не зможе створити об'єкт додатка. Даний макрос також установлює код, який перевіряє, що об'єкт є додатком і бібліотека була скомпільована з необхідними опціями.

Коли wxWidgets створює об'єкт типу MyApp, результат зберігається в глобальній змінниці wxTheApp. Цю змінну можна використовувати у програмі, однак краще явно не звертатися до неї у своєму коді. Замість цього після оголошення класу додатка варто помістити макрос:

```
// реалізація MyApp& GetApp()
DECLARE_APP(MyApp),
```

після чого можна використовувати в програмі функцію wxGetApp, що повертає посилання на об'єкт додатка типу MyApp.

Тепер розглянемо клас MyFrame. Фрейм – це основне вікно, що містить усі інші вікна.

Приклад оголошення класу для фрейму, що розміщується після оголошення MyApp:

```
// оголошення класу головного вікна
Class MyFrame:public wxFrame
```

```
{
public:
// конструктор
MyFrame(constwxString&title);
// обробник повідомлень
Void OnQuit(wxCommandEvent&event);
private:
// цей макрос перехоплює повідомлення
DECLARE_EVENT_TABLE()
};
```

Цей клас для фрейму містить конструктор, обробник повідомлень і макрос, який повідомляє, що клас містить обробники повідомлень.

Функції обробки повідомлень у MyFrame не повинні бути віртуальними. Тому обробник повідомлень задається в такий спосіб:

```
// таблиця повідомлень для MyFrame
BEGIN_EVENT_TABLE(MyFrame,
wxFrame)
EVT_MENU(wxID_EXIT,
MyFrame::OnQuit)
END_EVENT_TABLE()
```

Таблиця повідомлень, поміщена у файл із реалізацією класу, говорить wxWidgets про те, яким образом пов'язуються повідомлення, що надходять від користувача, і методи класу.

У зазначеній вище таблиці натискання кнопки миші на пункті меню з ідентифікатором wxID_EXIT направляється до функції MyFrame::OnQuit. Макрос EVT_MENU є одним з багатьох можливих макросів таблиці повідомлень, який можна використовувати, щоб указати wxWidgets, які типи повідомлень направляти у функцію. Ідентифікатори, використовувані в нашому прикладі, є визначеними, але найчастіше потрібно уводити власні за допомогою перерахувань (наприклад, enum).

Цей тип таблиці повідомлень називається статичним, тобто він не може бути змінений у процесі виконання програми.

Реалізація функції обробки повідомлення може бути такою:

```
void
MyFrame::OnQuit(wxCommandEvent&event)
{
Close(); // знищення фрейму
}
```

Ця функція викликається тільки тоді, коли функція OnInit повернула *true*.

Тепер можливо створити конструктор для фрейму, що ініціалізує сам фрейм:

```
MyFrame::MyFrame(const
wxString&title)
: wxFrame(NULL, wxID_ANY, title)
{
// встановлення іконки фрейма
SetIcon(wxIcon(mondrian_xpm));

// встановлення тексту заголовка фрейму
SetStatusText(wxT(«WelcometowxWidgets!»));
}
```

Даний конструктор викликає конструктор батьківського класу з указівкою батьківського вікна (NULL означає його відсутність), ідентифікатором цього вікна і заголовком. Замість ідентифікатора в нашому випадку передається значення wxID_ANY, що говорить wxWidgets, що ідентифікатор повинний бути створений бібліотекою самостійно, тим самим конструктор створює дійсне вікно за допомогою перенаправлення запиту на створення до батьківського класу.

Елементи керування можна додати в конструктор:

```
// створення меню
wxMenu*fileMenu=newwxMenu;
fileMenu-
>Append(wxID_EXIT,wxT(«E&xit\tAlt-X»),
wxT(«Quit this program»));
// додавання меню в створений рядок
меню
wxMenuBar*menuBar=newwxMenuBar();
menuBar-
Append(fileMenu,wx(«&File»));
// та додвання до фрейму
SetMenuBar(menuBar);
```

Рис. 2. Базова програма на wxWidgets

Таким чином, створюється основа програми, що має у своєму складі вікно й обробники повідомлень. У конструкторі можна створити інші функції, наприклад, ініціалізацію необхідних елементів керування. Реалізувавши розглянутий код, одержимо мінімальну програму (рис. 2), яку можна розширювати, використовуючи необхідні розробникові класи, наприклад, елементи керування.

Висновки. На основі проведених досліджень бібліотеки wxWidgets було досліджено її структуру, описано галузь використання і визначення кожного з компонентів, їх взаємодію. Запропонована методика створення додатків з використанням даної бібліотеки на прикладі створення простого додатка. На цьому прикладі розглянуто основні принципи реалізації, яких дотримується wxWidgets.

Література

1. Smart J. Cross-Platform GUI Programming with wxWidgets / J. Smart, K. Hock. – USA, 2005. – 744p.
2. Боровский А. wxWidgets: Живая история / Боровский А. – Москва, 2007.
3. Електроний ресурс. – Режим доступу : http://wxwidgets.org/
4. Електроний ресурс.– Режим доступу : http://wxwidgets.info/
5. Build cross-platform GUIs using wxWidgets [Електроний ресурс]. – Режим доступу : http://www.ibm.com/developerworks/library/l-wxwidgets/index.html

Анотації

У статті розглянуто бібліотеку wxWidgets, описано структуру, галузь використання компонентів цієї бібліотеки та їх взаємодія. Також розглянуто методику побудови програм з використанням wxWidgets шляхом створення простої програми, яка демонструє принципи та основні критерії при розробці подібних програм.

В статье рассмотрена библиотека wxWidgets, описана структура, область использования компонентов этой библиотеки и их взаимодействие между собой. Также рассмотрена методика построения приложений средствами wxWidgets путем создания минимального приложения, которое демонстрирует принципы и основные критерии при разработке приложений.

We have observed the wxWidgets application toolkit, its structure, components usage range, their dependencies and iteraction with each other. Also the methodology of program's graphical interface building with wxWidgets library was described through creating of simple application demonstrating principles and ideas of the library.