ISSN 2307-9851

НАУКОВО-МЕТОДИЧНИЙ ЖУРНАЛ

Виходить 8 разів на рік

Видається з лютого 1998 року

Засновники:
Інститут педагогіки НАПН України, Інститут інформаційних технологій і засобів навчання НАПН України, Редакція журналу

Журнал видається за сприяння Міністерства освіти і науки України
Свідоцтво про реєстрацію серія КВ №12217–1101ПР від 17.01.2007

Передплатний індекс 74248

Журнал включено до Переліку наукових фахових видань України у галузі педагогічних наук, Наказ МОН України від 29.09.2014 року №1081

Журнал внесений до наукометричної бази даних РИНЦ

Затверджено Вченою радою Інституту педагогіки НАПН України, протокол №11 від 2 липня 2018 р.

E-mail: csf22101@ukr.net

Офіційний сайт журналу:
www.csf221.wordpress.com

КОМП'ЮТЕР у школі та сім'ї

№4 (148) ◈ 2018

ЗМІСТ

На першій і другій всторінках обкладинки:
« До 45-річчя Луцького міського Центру науково-технічної творчості учнівської молоді»

УДК 37.01:004.4'2

СТВОРЕННЯ ПРОЕКТУ WXWIDGETS ТА ОПИС ПОДІЙ У С++

Рудик Олександр Борисович

кандидат фізико-математичних наук, доцент Інституту післядипломної педагогічної освіти Київського університету імені Бориса Грінченка, rudykob@gmail.com,
ORCID ID 0000-0003-3676-0688

Холодова Олена Станіславівна
учитель спеціалізованої школи I–III ступенів № 120 з поглибленим вивченням предметів природничо-математичного циклу міста Києва, helenholodova@gmail.com

Анотація. Описано створення простого подійно та об'єктно орієнтованого проекту wxWidgets у середовищі CodeBlocls у вигляді опису частини практичної роботи. Її використання як навчального матеріалу не передбачає наявності в учня надійних знань й умінь з програмування і навіть допомоги учителя. Подано перелік найчастіше використовуваних подій з їх тлумаченням.

Ключові слова: подійно та об'єктно орієнтоване програмування, С++, CodeBlocks, wxWidgets, створення проекту.

У публікації [1] проведено аналіз проблем, пов'язаних зі зміною парадигми вивчення інформатики у школі: істотним зростанням кількості навчальних годин на вивчення програмування. Там само обґрунтовано вибір мови С++, середовища програмування CodeBlocks і бібліотеки wxWidgets. Дана робота є безпосереднім продовженням публікації [1] і її написано у вигляді опису частини практичної роботи. Її використання як навчального матеріалу не передбачає надійних знань й умінь учня з програмування і навіть втручання учителя за умови, якщо учень дотримується написаних вказівок або уміє знаходити власні помилки при відхиленні від цих вказівок. Для цього опис детально проілюстровано виглядом вікна програми CodeBlocks або його частин для різних етапів створення проекту. Використати цей опис може користувач, який до цього взагалі не працював із середовищем наочного програмування. Бажано, щоб читач сам пересвідчився у цьому, попередньо встановивши програмне забезпечення згідно зі вказівками роботи [1].

Вікно CodeBlocks, налаштоване на здійснення об'єктно- і подійно-орієнтоване програмування з використанням бібліотеки wxWidgets, має складові, проілюстровані на рис. 1–9.

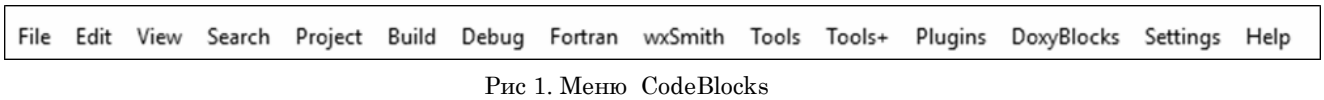


Рис 1. Меню CodeBlocks

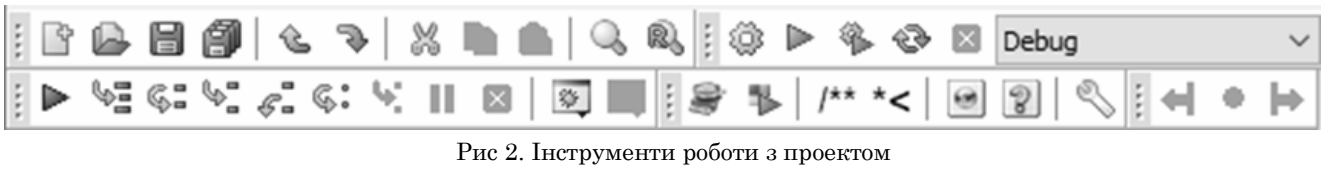


Рис 2. Інструменти роботи з проектом

Для виклику редактор програмного коду потрібно двічі клацнути лівою клавішею миші на екранній формі. Для повернення до роботи з екранною формою у наочному режимі потрібно натиснути на вкладення з назвою проекту *.wxs.

Екранна форма — це вікно, що містить графічні елементи або об'єкти управління (меню, кнопки тощо). Кожний такий об'єкт може мати зв'язаний із ним функціональний програмний код, оформлений у вигляді процедури опрацювання події.

Завдання. Створити проект, в якому користувач, після натиснення на кнопку з надписом Click here, побачить на екрані надпис з привітанням Hello C++!.

Вказівки до виконання (згідно з рекомендаціями [2])

1. Створити теку Ваше прізвище у вказаній вчителем теці.

2. У середовищі CodeBlocks скористатися вказівкою меню New / Project…

3. У вікні діалогу New from template обрати WxWidgets project і натиснути Go.

4. У вікні привітання натиснути Next.

5. У наступному вікні діалогу обрати версію бібліотеки 3.0.x і натиснути кнопку Next.

6. У вікні діалогу wxWidgets project вказати:

- назву проекту tutorial у полі Project title (Назва проекту);

•теку, в якій буде збережено проект, у полі Folder to create project in (Тека, у якій створити проект). Решту полів буде заповнено автоматично. Натиснути кнопку **Next**.

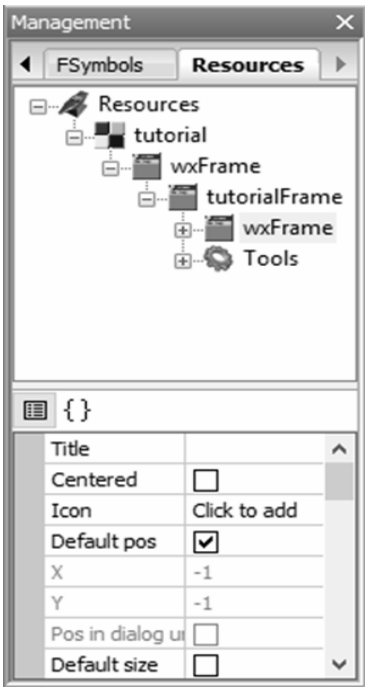


Рис 3. Інспектор об'єктів

7. За бажанням вказати дані про автора проекту:

- у поді Author — ім'я і прізвище;
- у поді Author's e-mail — адресу електронної скриньки;
- у полі Author's webcite — адресу сайту і натиснути кнопку **Next**.

8. Обрати:

- в області GUI Builder — інструмент для графічного користувацького інтерфейсу — wxSmith;
- в області Applicationtype — тип застосунку — Frame Based і натиснути кнопку **Next**.

9. Указати розташування теки (найвищого рівня), де було розпаковано wxWidgets (наприклад, D:\WxWidgets) і натиснути кнопку **Next**.

10. Вказати GNU GCC Compiler — компілятор, який буде використано, і натиснути кнопку **Next**.

11. Вказати параметри конфігурації wxWidgets, виставивши мітки:

- Use wxWidgets DLL;
- wxWidgetsis built as a monolitic library;
- Enable unicode;
- Create and use precompiled header (PCH);
- Configure Advanced Options.

При появі вікна попередження про неможливість знайти конфігурацію відладчика у вказаній теці

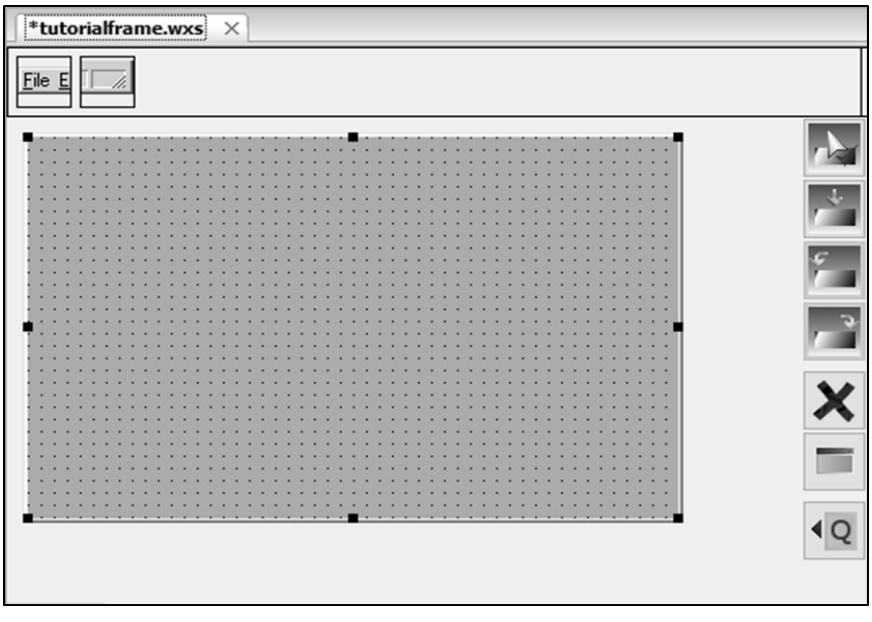


Рис. 4. Вікно екранної форми проекту

WxWidgets, обирати варіант **Так**. Згодом можна буде уточнити шлях до теки з відладчиком.

12. При використанні ОС Windows вказати розширені параметри компілятора, виставивши мітки:

- Use_WXDEBUG_ and Debug wxWidgets lib в області Advanced options (GCC only);
- GUI Mode Application в областях Debug Target і Release Target і настинути кнопку **Finish**. Цей крок відсутній про роботі з ОС Linux.

13. Переконатися у правильності налаштувань компілятора, використавши вказівку меню Settings / Compiler… (Налаштування / Компілятор). У вікні налаштувань компілятора в розділі Global compiler settings (Глобальні параметри компілятора) переконатися, що обрано компілятор GNU GCC Compiler та на вкладенні Compiler Flags (Прапорці компілятора) прапорець

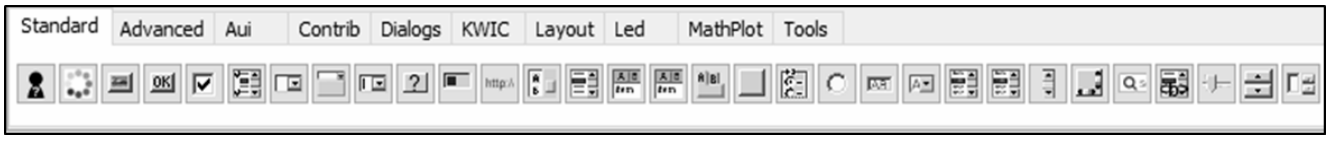


Рис. 5. Панель компонент графічного інтерфейсу

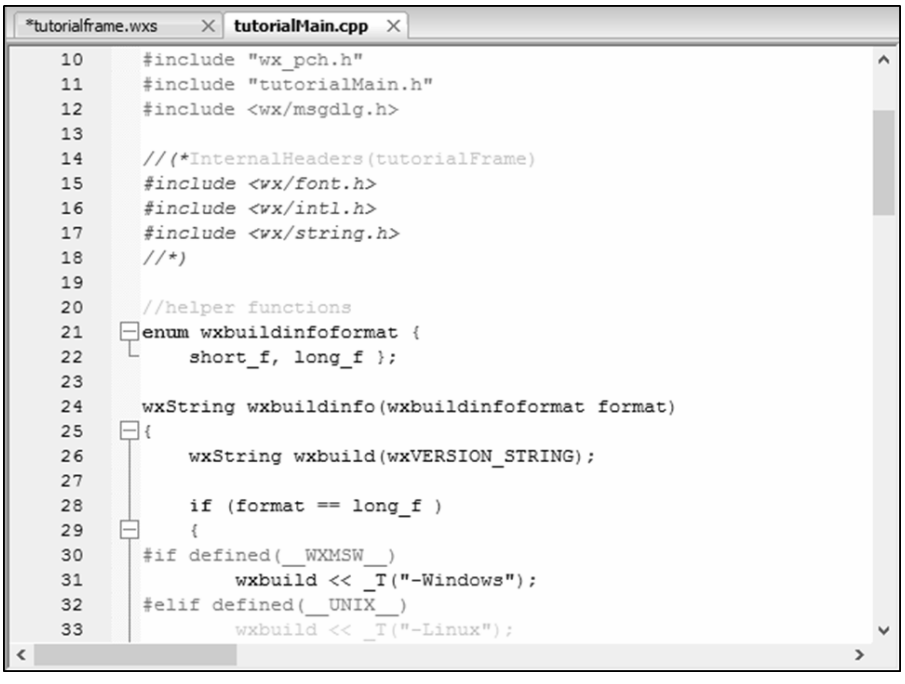


Рис. 6. Убудований редактор програмного коду

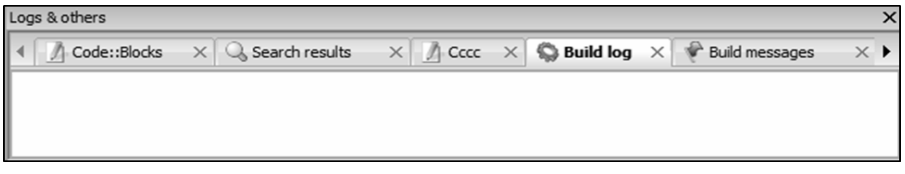


Рис. 7. Вікно повідомлень

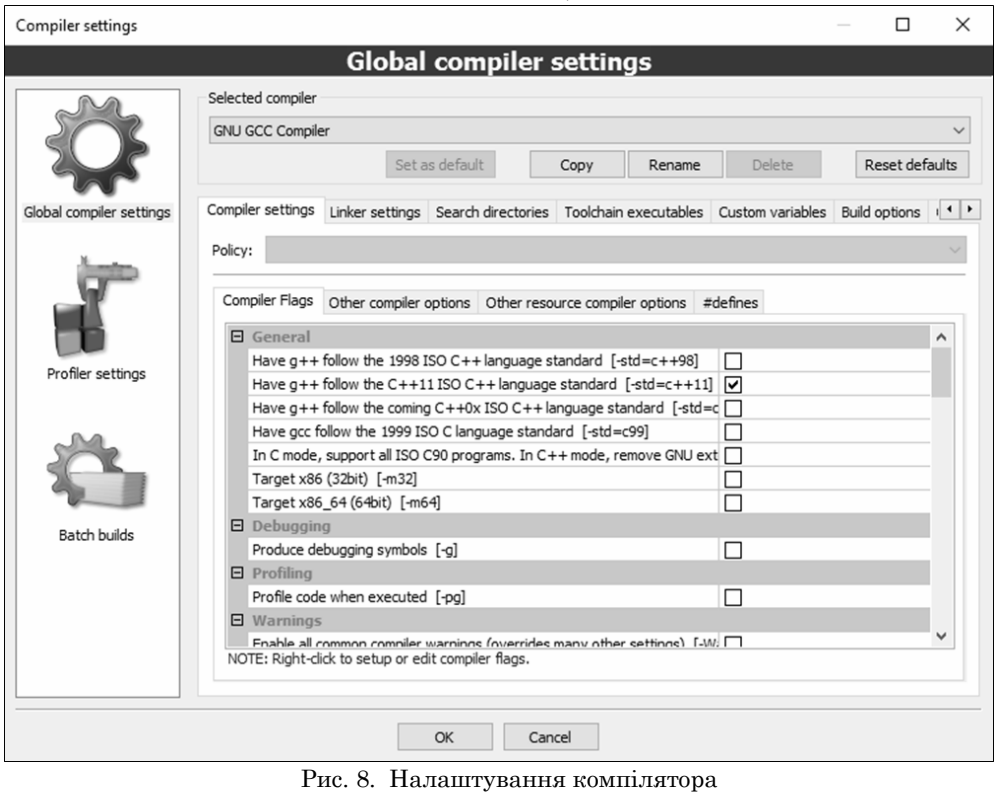


Рис. 8. Налаштування компілятора

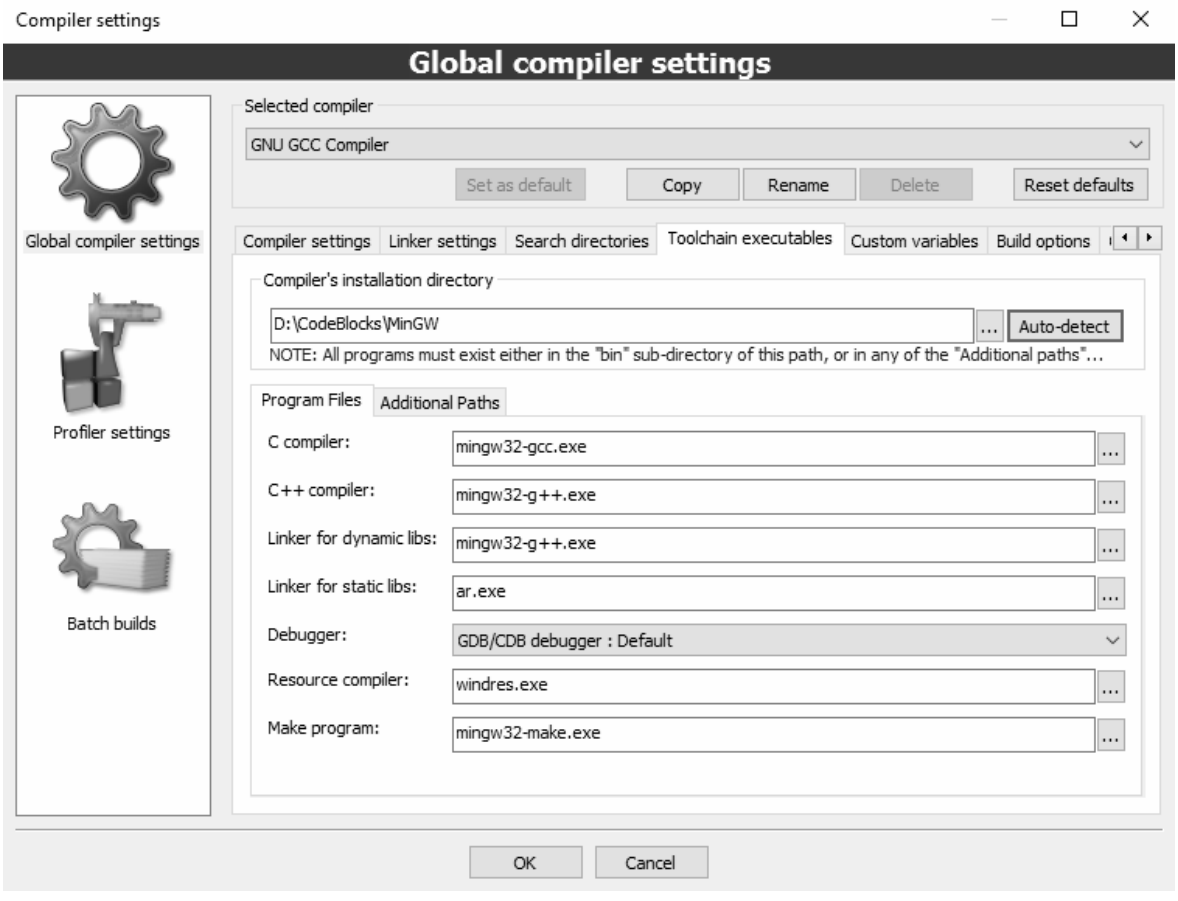
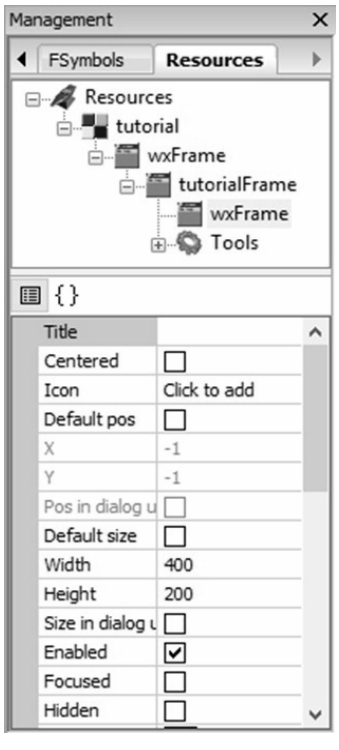


Рис. 9. Уточнення шляху до теки з відладчиком

виставлено лише для такого:



Have g++ follow the C++ 11 ISO C++ language standard [-std=c++11].

Перейти на вкладення *Toolchain executables* (Виконувані файли) і натиснути кнопку *Auto-detect* (Автовизначення) для уточнення шляху до теки з відладчиком. Натиснути OK для завершення налаштувань.

14. У інспекторі об'єктів перейти на вкладення *Resources* (Джерела). Виділити об'єкт wxFrame (Конструкція) та змініть розміри екранної форми, прибравши прапорець навпроти властивості *Default size* (Розміри як усталено), і надати нові значення властивостям:

Height (Висота)= 400,

Width (Ширина) = 200.

15. На панелі об'єктів — компонентів графічного інтерфейсу перейти на вкладення *Layot* (Макет), обрати інструмент wxBoxSizer.

Рис 10. Змінювання розмірів екранної форми

Цей компонент використовують як контейнер для інших компонентів. Розташувати його на екранній формі, клацнувши на ній лівою клавішею миші.

16. На панелі компонентів *Standart* (Стандартна) обрати інструмент wxButton (Кнопка). Розташувати її на екранній формі всередині контейнера wxBoxSizer.

17. Змінити значення властивостей об'єкта wxButton (Кнопка):

- *Label* (Надпис) — *Click here*! (для того щоб текст було подано двома рядками, потрібно викликати редактор тексту, натиснувши на кнопку праворуч від значення *Label*);
- *Font* (Шрифт) — для початку форматування тексту натиснути на кнопку праворуч від значення *Click to edit* (Натисніть для редагування), з'явиться вікно *Font settings* (Налаштування шрифту) в ньому натиснути кнопку *Change* (Змінити).

У вікні налаштування шрифту надати такі значення властивостей:

Шрифт — Arial;
Накреслення — звичайне;
Розмір — 20;
Колір — за власним смаком.

18. Створити програмний код для опрацювання події *OnButtonClick* — натискання лівої кнопки миші. Для цього в інспекторі об'єктів перейти на вкладення {}*Event* (Події). У полі *EVT_Button* випадного списку обрати значення *Add new handler* (Додати новий обро-

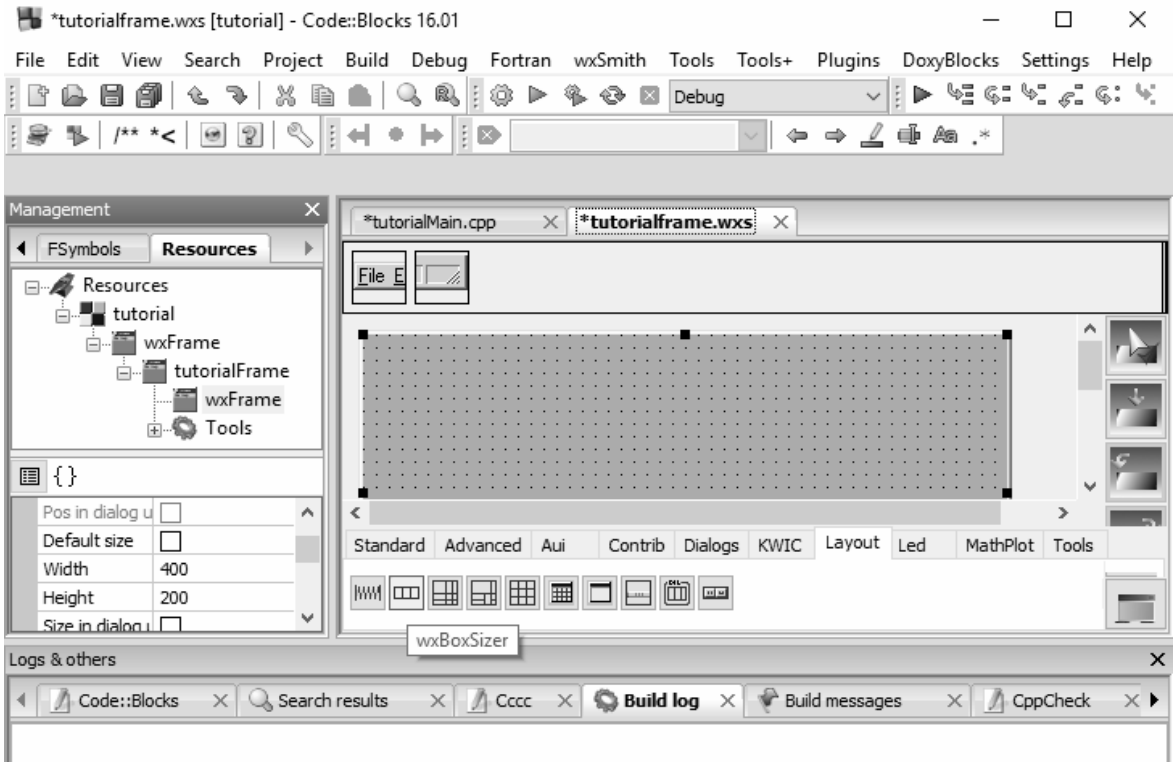


Рис. 11. Розташування контейнера wxBoxSizer на екранній формі

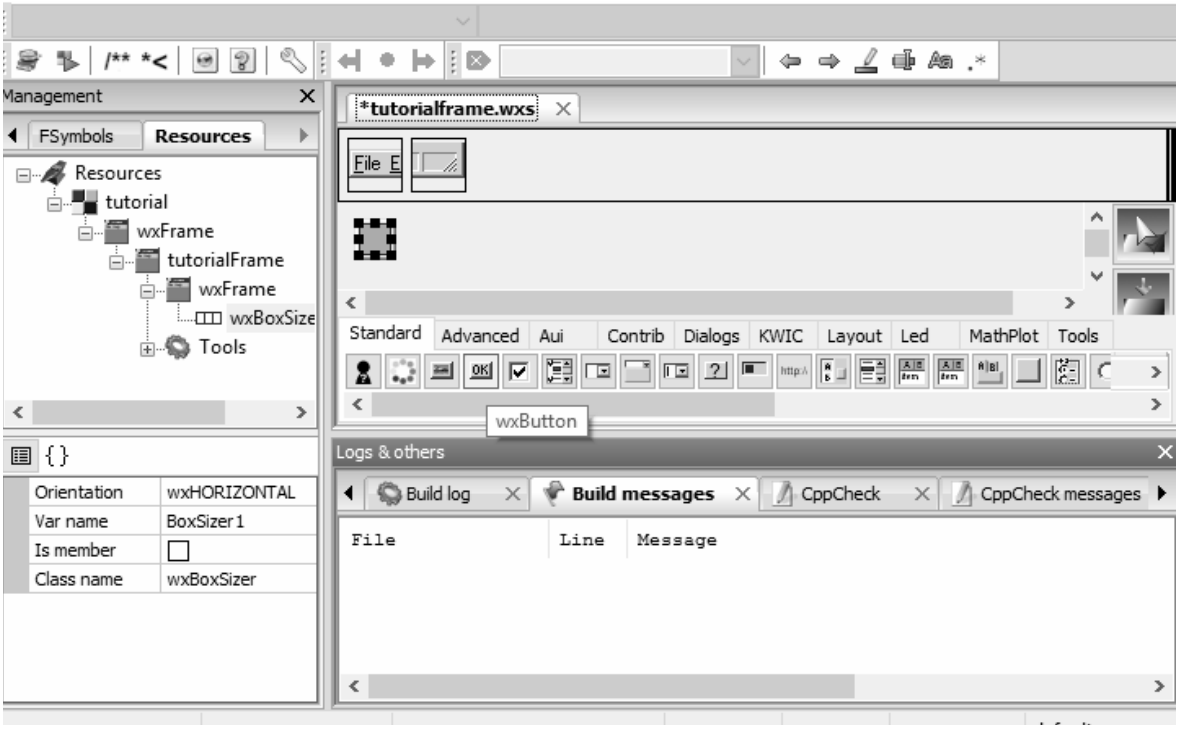


Рис. 12. Розташування *wxButton* (Кнопка) на екранній формі всередині контейнера *wxBoxSizer*

бник).У вікні, що з'явиться, вказати значення події *OnButtonClick* та натиснути кнопку *OK*.

У вікні редактора коду внести код опрацювання події:

```
void tutorialFrame::OnButton1Click
  (wxCommandEvent& event)
  {wxString msg = _T("Hello C++!");
  wxString info = _T("greeting");
  wxMessageBox (msg, info,
    wxOK|wxICON_INFORMATION, this);
  }
```

Проаналізувати поданий вище код.

19. Запустити проект на виконання, натиснувши на панелі інструментів кнопку Build and run (Побудувати та виконати). Перевірити правильність роботи програми. За потреби виправити помилки.

20. Зберегти проект і завершити роботу з проектом і середовищем.

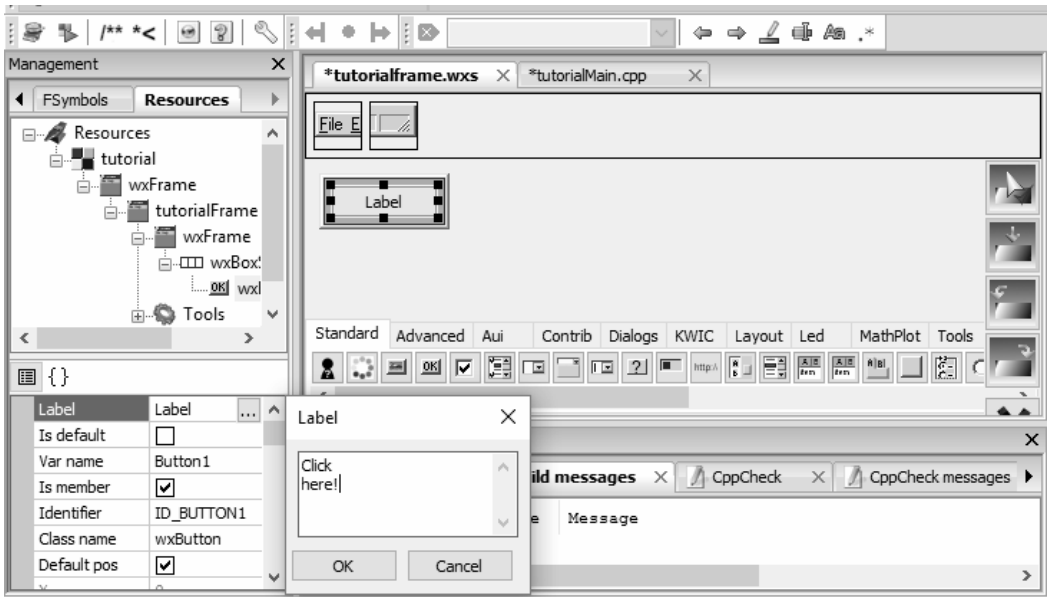


Рис. 13. Змінювання значення властивості *Label* (Надпис) об'єкта *wxButton* (Кнопка)

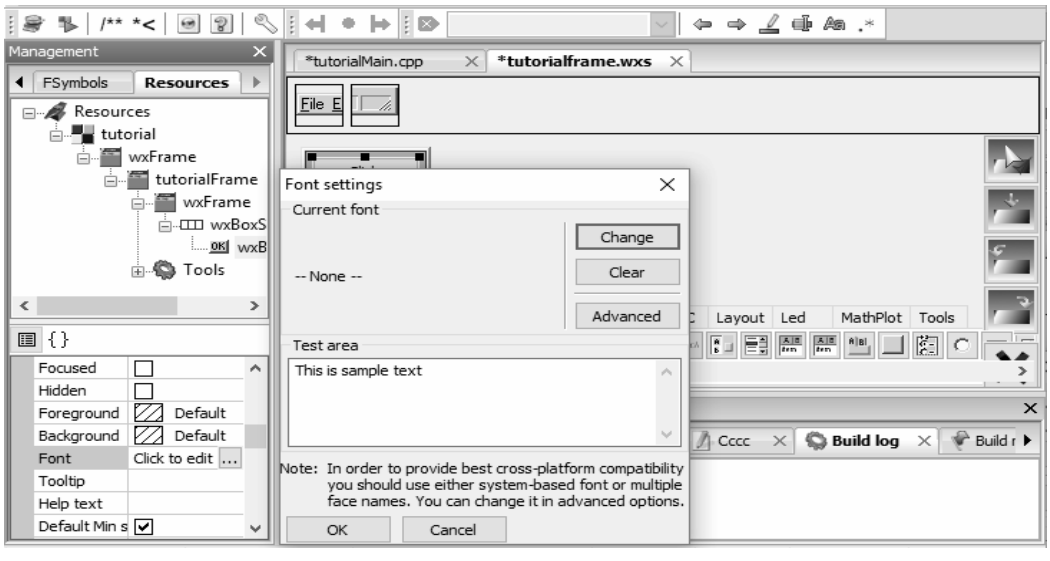


Рис. 14. Змінювання значення властивості *Font* (Шрифт) об'єкта *wxButton* (Кнопка)

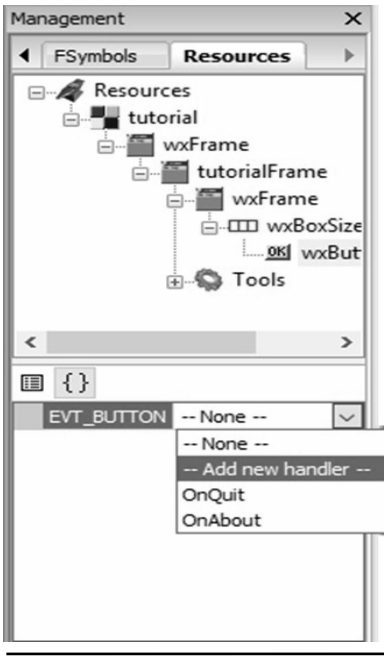


Рис. 15. Додавання нового обробника подій

Для планування створення складних проектів необхідно принаймні ознайомитися з переліком [3] найчастіше використовуваних подій.

●Подія *OnChanging* для класу *TCanvas* (полотно для малювання) відбувається одразу після зміни зображення на полотні.

● Подія *OnChange* для класу *TCanvas* настає після зміни зображення на полотні. При виклику будь-якого методу малювання здійснюється така послідовність операцій:

настає подія *OnChanging*;

викликаний метод полотна *TCanvas* робить зміни в зображенні;

настає подія *OnChange*.

Подія полотна *OnChange* настає при зміні самого зображення, а не властивостей полотна. Такі властивості полотна як об'єкти *Font* (шрифт), *Brush* (пензель) *Pen* (перо) мають свої власні події.

● Подія *OnClick* **для класу** *TControl* настає, якщо користувач натиснув і відпустив основну клавішу миші тоді, коли вказівник миші розташовано на об'єкті. Один обробник події *OnClick* можна використовувати для опрацювання подій з різними об'єктами. Якщо потрібно розрізняти, з яким об'єктом пов'язана подія, що відбулася і передбачити для різних об'єктів різний відгук, використовують параметр `Sender`: `ShowMessage("OnClick у "+((TControl *) Sender)->Name);`

● Подія ***OnCreate*** для класу ***TCustomForm*** на-

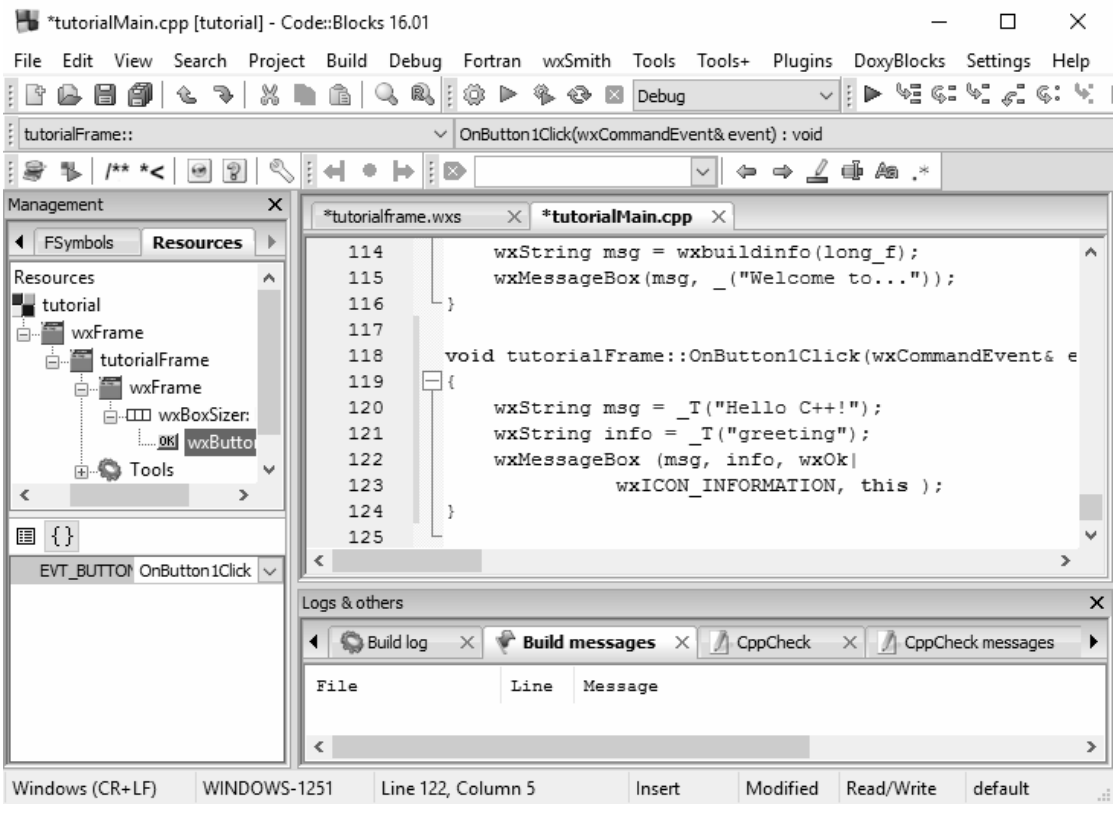


Рис. 16. Додавання коду обробника подій

стає у момент створення форми. Її використовують для виконання процедур налаштувань форми або розташованих на ній об'єктів. Якщо в обробнику цієї події створюються об'єкти, їх потрібно руйнуватися, щоб звільнити пам'ять, в обробнику події *OnDestroy*. Послідовність подій при створенні форми зі значенням true властивості *Visible* така: *OnCreate, OnShow, OnActivate, OnPaint*.

• Подія ***OnDblClick* для класу *TControl*** *настає при подвійному клацанні*. До одного і того самого об'єкту можна написати обробники подій *OnClick* і *OnDblClick*, бо перший з них завжди перехопить перше клацання. Властивість *Sender* містить назву об'єкта, з яким трапилася подія. Її можна використати для диференційованого відгуку на події з різними об'єктами.

• Подія ***OnDragDrop*** для класу ***TControl*** настає у момент відпускання об'єкту, який перетягують, над даним об'єктом. У обробнику події треба описати, що в цей момент має відбутися. Параметр *Source* містить назву об'єкту, який перетягуємо, а параметр *Sender* — об'єкту, над яким здійснено відпущенння. Параметри *X, Y* містять координати розташування вказівника миші над об'єктом у системі координат клієнтської області цього об'єкта.

Нехай на формі є кілька списків типу *TListBox* і дозволено переміщати рядки з одного списку в інший. Це можна зробити таким чином. У всіх списках надають властивості *DragMode* значення *dmAutomatic*. Це забезпечує автоматичний початок перетягування. Далі для одного зі списків пишуть обробник події *OnDragOver*.

```
void __fastcall TForm1::ListBox1DragOver
(TObject *Sender,TObject *Source,
int X, int Y, TDragStatS State,
bool SAccept)
{ Accept = Source->ClassNameIs
("TListBox");
}
```

Цей обробник вказує, що на даний компонент можна перетягувати об'єкти типу *TListBox*. У всіх інших списках на подію *OnDragQver* вказують цей самий обробник. Далі для одного зі списків пишуть обробник події *OnDragDrop*:

```
void __fastcall TForm1::ListBox1DragDrop
(TObject *Sender,TObject *Source,
int X, int Y)
{ TListBox *S = (TListBox *)Source;
((TListBox*)Sender)->Items->
Add(S->Items->Strings[S->ItemIndex]);
S->Items->Delete(S->ItemIndex);
}
```

Перші два оператора обробника додають до списку рядок, виділений у списку-джерелі. Якщо пишуть не універсальний оброблювач, а призначений для для даного об'єкта, то перший оператор можна видалити, а у другому ((TListBox *) Sender) і S потрібно замінити на назву компонента ListBox1. Третій оператор видаляє перенесений рядок з джерела. Якщо потрібно не перенесення, а лише копіювання рядків з одного списку в інший, то цей оператор не потрібен.

У всіх інших списках для події *OnDragDrop* вказують цей самий обробник. Після цього, заповнивши списки, можна запускати застосунок. Користувач зможе перетягувати рядки між будь-якими наявними списками.

• **Подія *OnDragOver* для класу *TControl*** *настає у момент, коли об'єкт, який переміщують, перетнув межу даного об'єкта і опинився всередині його контуру*. Завершується подія, коли переміщуваний об'єкт, залишаючи даний об'єкт, перетнув його межу. Обробник події *OnDragOver* використовують для того, щоб

дати сигнал про готовність прийняти переміщуваний об'єкт у разі, якщо користувач відпустить його над даними об'єктом. Якщо даний об'єкт готовий прийняти переміщуваний об'єкт, в обробнику потрібно надати значення true параметру *Accept*. Але цей параметр за замовчанням має значення true, тому його значення можна і не змінювати, якщо перед цим воно не змінювалося. У граничному випадку обробник може бути порожнім, що позначатиме його готовність прийняти будь-який об'єкт. Але навіть порожній обробник потрібен, бо інакше повідомлення про прийом об'єкта застосунок не отримає. Під час перетягування форма вказівника миші може змінюватися, сигналізуючи користувачеві про готовність прийняти об'єкт. Для цього потрібно до моменту події *OnDragOver* задати відповідне значення властивості об'єкта *DragCursor*. Параметр *Source* визначає переміщуваний об'єкт, параметр *Sender* — об'єкт, на який можна перемістити, параметри *X, Y* — координати точки екрана в пікселах. Параметр *State* типу *TDragState* визначає стан об'єкта, що переміщуємо, по відношенню до інших об'єктів. Можливі наступні значення параметра *State*:

dsDragEnter — вказівник миші входить у межі об'єкта;

dsDragMove — вказівник миші переміщують у межах об'єкта;

dsDragLeave — вказівник миші виходить за межі компонента.

У поданому нижче прикладі обробник події *OnDragOver* сигналізує про те, що компонент готовий прийняти переміщуваний об'єкт, якщо це компонент *ListBox1*.

```
  void __fastcall TForm1::ListBox1DragOver
  (TObject *Sender,TObject *Source,
  int X, int Y, TDragState State,
  bool SAccept)
  {  Accept    =  (((TControl*)Sender)->Name
== "ListBox1");
}
```

● Подія ***OnEndDrag*** для класу ***TControl*** *настає при будь-якому завершенні процесу переміщення:* вдалому (об'єкт переміщено у приймач) або невдалому (об'єкт відпущено над формою або іншим об'єктом, які не можуть його прийняти). Подія стосується переміщуваного об'єкта. Опрацювання цієї події не є необхідним для здійснення самого процесу переміщення.

Обробник необхідно створювати лише за потреби дії або повідомлення — підтвердження результату перетягування.

Параметр *Sender* — це об'єкт переміщення. Параметр *Target* — це приймач, якщо об'єкт був ним прийнятий, або NULL, якщо переміщення закінчилося невдачею. Параметри *X, Y*— координати екрану в пікселях. Подамо приклад відображення повідомлення про результати перетягування: «Перенесення об'єкта *ListBox1* перервано» або «*ListBox1*перенесено в *ListBox2*». Подібним чином можна запрограмувати будь-які дії.

```
  void __fastcall TForm1::ListBox1EndDrag
  (TObject *Sender,TObject *Target, int X,
int Y)
  { if (Target == NULL) ShowMessage
  ("Перенесенння об'єкту " +
  ((TControl*)Sender)->Name + " перервано");
  else                 ShowMessage
  (((TControl*)Sender)->Name +
  " перенесено в " +
  ((TControl*)Target)->Name);
  }
```

● Подія ***OnEnter*** для класу ***TWinControl*** настає у момент отримання елементом фокусу (виділення). Ця подія не настає при перемиканні між формами або між додатками. Під час перемикань між елементами, розташованими в різних контейнерах, наприклад, на різних панелях, подія *OnEnter* спочатку настає для контейнера, а потім для елемента, розташованому в ньому. Нехай форма має кнопку *OK* і групову панель, що включає три радіокнопки. Нехай у початковий момент активна кнопка *OK*. Коли користувач клацне на одній з радіокнопок, для кнопки *OK* настане подія *OnExit*, потім настане подія *OnEnter*групової панелі, і лише потім настане подія *OnEnter* тієї кнопки, по якій клацнули. Якщо після цього користувач клацне на кнопку *OK*, то спочатку настане подія *OnExit*радіокнопки, потім подія *OnExit* групової панелі, а потім подія *OnEnter* кнопки *OK*.

● Подія ***OnExit*** для класу ***TWinControl*** настає у момент втрати елементом фокусу, тобто у момент перемикання з нього на інший елемент. Ця подія не настає при перемиканні між формами або між застосунками. Значення властивості *ActiveControl* змінюється перш, ніж відбувається подія *OnExit*. Під час перемикань між елементами, розташованими в різних контейнерах, наприклад, на різних панелях, подія *OnExit* спочатку настає для елемента, а потім для контейнера, що його містить.

● Подія ***OnKeyDown*** для класу ***TWinControl*** настає, якщо об'єкт перебуває у фокусі, при натисканні користувачем будь-якої клавіші, включаючи функціональні й допоміжні (Shift, Alt, Ctrl). В обробник події передають, крім звичайного параметра *Sender*, що вказує на об'єкт, в якому відбулася подія, також параметри *Key*, *Shift*. Параметр *Key* визначає натиснуту клавішу клавіатури. Для *не* алфавітно-цифрових клавіш використано віртуальний код API Windows. Коди не розрізняють символи у верхньому і нижньому регістрах і не розрізняють літер кирилиці й латиниці. Параметр *Shift* є множиною, яка може бути порожньою або містити такі елементи:

- *ssShift* — натиснуто клавішу *Shift*;
- *ssAlt* — натиснуто клавішу *Alt*;
- *ssCtrl* — натиснуто клавішу *Ctrl*.

Наприклад, для розпізнавання комбінації клавіш *Alt + X* використовують такий оператор:

```
if((Key == 'X') && Shift.Contains(ssAlt))
…
```

А відгук на натиснення клавіші *Enter* можна оформити одним з таких трьох операторів:

- або `if(Key == 13) …;`
- або `if(Key == 0x0D) …;`
- або `if(Key == VK_RETURN) …`

● Подія ***OnKeyPress*** для класу ***TWinControl****настає,* якщо об'єкт перебуває у фокусі, при натис-

канні користувачем клавіші символу. Параметр *Key* в обробнику цієї події має тип *Char* і відповідає символу натиснутої клавіші. При цьому розрізняють символи у верхньому і нижньому регістрах, символи кирилиці й латиниці. Клавіші, що не відображаються в кодах ASCII (функціональні клавіші і такі як *Shift, Alt, Ctrl*), не викликають цієї події. Тому натискання таких комбінацій клавіш, як, наприклад, *Shift + A*, породжує лише одну подію *OnKeyPress*, при якому параметр *Key* дорівнює «A». Для того щоб розпізнавати клавіші, які не відповідають символам, або комбінаціям клавіш, треба використовувати обробники подій *OnKeyDown* і *OnKeyUp*. Зауважимо: подія *OnKeyPress* наступає, якщо натиснуто лише клавішу символу або клавіша символу і клавішу *Shift*. Якщо ж клавіша символу натиснуто одночасно з якоюсь з допоміжних клавіш, то подія *OnKeyPress* може не настати: відбудуться тільки події *OnKevDown* при натисканні й *OnKeyUp* при відпусканні. Або, якщо і настане, то вкаже на хибний символ.

Параметр *Key* передають в обробник за посиланням. Його можна змінити, передаючи для подальшої стандартної обробки інший символ.

Оператор обробника події *OnKeyPress* переводить латинські символи у верхній регістр, незалежно від того, в якому регістрі набрав їх користувач. Цей оператор діє тільки на літери латиниці:

```
Key = UpCase (Key);
```

Аналогічний оператор діє на літери латиниці та літери кирилиці:

```
Key = AnsiOpperCase(Key);
```

● Подія ***OnKeyUp*** для класу ***TWinControl*** настає, якщо компонент перебуває у фокусі, при відпусканні користувачем будь-якої раніше натиснутої клавіші, включаючи функціональні й допоміжні, такі як *Shift, Alt, Ctrl*. В обробник події передають, крім параметра *Sender*, що вказує на об'єкт, у якому відбулася подія, також параметри *Key* і *Shift*. Параметр *Key* визначає клавішу клавіатури, яку щойно відпустили. Для *не* алфавітно-цифрових клавіш використовуються віртуальні коди API Windows. Подія ***OnKeyUp*** не розрізняють коди символів у верхньому й нижньому регістрах, не розрізняють символи кирилиці й латиниці.

Параметр *Shift* є множиною, яка може бути порожньою або містити такі елементи:

- ○ *ssShift* — відпускання клавіші *Shift*;
- ○ *ssAlt* — відпускання клавіші *Alt*;
- ○ *ssCtrl* — відпускання клавіші *Ctrl*.

Значення елементів *Shift*, при відповідних натисканнях кнопок миші, в даній події не використовують. Подія *OnKeyUp* найзручніша для розпізнавання натиснутих клавіш, особливо їх комбінацій. Потрібно лише не забувати, що параметр *Key* має тип word, а не char, тому для розпізнавання потрібно використовувати віртуальні коди. До того ж, треба враховувати, що віртуальний код не розрізняє символи верхнього і нижнього регістру клавіш й не реагує на те, яку розкладку на даний момент увімкнено.

Нехай потрібно написати обробник, який би реагував на натискання клавіш *Shift + Y*. Перевірити *натиснуті* клавіші можна оператором, але він буде реагувати і на «У», і на «у», і навіть на літери кирилиці «Н» і «н». У цій ситуації потрібно використовувати подію *OnKeyPress*:

```
if((Key == 'Y') && Shift.Contains
(ssShift))
```

● Події ***OnMouseDown*** та ***OnMouseUp*** для класу ***TControl*** настають відповідно при натисканні і відпусканні користувачем будь-якої кнопки миші. Обробник події *OnMouseDown* можна використовувати для початку процесу перетягування компонента, якщо потрібно встановити додаткову умову. Наприклад, перевірити умови, за яких можна починати перетягування. У цьому випадку для об'єкта задають значення *dmManual*властивості *DragMode*, що забезпечує керування початком перетягування. Обробник події *OnMouseDown* може мати такий вигляд:

```
void __fastcall TForm1 ::
ListBox1MouseDown (TObject * Sender,
TMouseButton Button, TShiftState Shift,
int X, int Y)
{ if ((Button == mbLeft) &&<умова>)
((TControl *) Sender) -> BeginDrag (false,
5);
}
```

● Подія ***OnMouseEnter*** для класу ***TCustomLabel*** настає, коли вказівник миші входить в область мітки. Обробник події *OnMouseEnter* створюють, якщо потрібно зробити якісь операції при переміщенні вказівника миші над позначкою. Використавши цю подію, можна змінити колір тла або тексту мітки, а потім повернути його до попереднього кольору, коли вказівник покине область мітки і настане подія *OnMouseLeave*. Для цього в обробник події *OnMouseEnter* усіх міток застосунку можна вставити оператор:

```
((TLabel *)Sender)->Font->Color=clBlue;
```

А в обробник події *OnMouseLeave* такий оператор:

```
((TLabel *)Sender)->Font->Color = clBlack;
```

При переміщенні вказівника миші над будь-якою міткою її текст буде перефарбовано у блакитний колір.

● Подія ***OnMouseLeave*** для класу ***TCustomLabel*** настає, коли вказівник миші залишає елемент керування. Її використовують для скасування операції, раніше запроваджених в обробнику події *OnMouseEnter* для змін у мітці часу переміщення над нею вказівника миші.

● Подія ***OnMouseMove*** для класу ***TControl*** настає при переміщенні вказівника миші по елементу керування. Її використовують, якщо потрібно зробити певні операції при переміщенні вказівника миші над компонентом. Параметр *Shift*, що є множиною, містить елементи, що дозволяють визначити, які кнопки миші і які допоміжні клавіші (*Shift, Ctrl, Alt*) натиснуто у поточний момент. Параметри *X, Y* містять координати вказівника миші у клієнтській області об'єкта. Параметр *Sender* (джерело події) — сам об'єкт. Подія *OnMouseMove* виникає незалежно від того, натиснуто якісь кнопки або клавіші чи ні. Наприклад, для виконання певної дії при натиснутій кнопці *Alt* під час переміщення курсора миші над об'єктом, використовують такий умовний оператор: `if(Shift.Contains(ssAlt)) …`

● Подія ***OnPaint*** для класів ***TCustomForm,***

TPaintBox настає, коли приходить повідомлення ОС про необхідність перемалювати зіпсоване зображення. Зображення може зіпсуватися через тимчасове перекриття даного вікна іншим вікном або роботи стороннього застосунку. Оброблювач цієї події повинен перемалювати зображення. При перемальовуванні зображення полотна *Canvas* можна використовувати властивість *ClipRect*, яке вказує область полотна, всередині якої зображення зіпсовано.

Якщо копію зображення, яке з'являється на полотні, розташовано в об'єкті *BitMap,* то обробник події *OnPaint* для форми може мати такий вигляд:

```
Canvas->Draw(0,0,BitMap);
```

Для об'єкта *PaintBox*:

```
PaintBox1->Canvas->Draw(0,0,BitMap);
```

Швидше перемальовування здійснюють при використанні властивості *ClipRect* полотна:

```
Canvas->CopyRect(Canvas->ClipRect,
Bitmap->Canvas, Canvas->ClipRect);
```

● Подія ***OnProgress*** для класів ***TGraphic, TImage, TPicture*** настає під час повільних процесів змін графічного зображення: завантаження, збереження, трансформація. Ці події забезпечують побудувати в застосунку індикатора перебігу процесу, який забезпечує зворотний зв'язок з користувачем. Розробники нових складових можуть породжувати події *OnProgress*, викликаючи захищений метод *Progress*. Параметр *Stage* вказує на стадію процесу: початок, продовження, закінчення. Він може відповідно набувати значень *psStarting, psRunning, psEnding*. Якщо застосунок передбачає індикацію процесу, то можна створювати індикатор при набутті Stage значення psStarting, змінювати його значення, якщо Stage мав значення psRunning і закривати при набутті Stage значення psEnding. Параметр *PercentDone* показує, яка приблизно частина процесу виконана. Цей параметр може використовуватися в індикаторі процесу. Параметр *RedrawNow* вказує, чи може зображення в даний момент бути успішно відображене на екрані. Параметр *R* указує область зображення, яка змінена і потребує перемальовування. Параметр *Msg* містить коротку довідку про перебіг процесу, наприклад: Loading, Storing, Reducing colors. Рядок *Msg* може бути порожнім.

● Подія ***OnStartDrag*** для класу ***TControl*** настає, коли користувач натиснув ліву кнопку миші над об'єктом і, не відпускаючи її, почав зміщати курсор миші, тобто почав перетягування. Обробник події *OnStartDrag* дозволяє описати якісь спеціальні дії, необхідні на початку перетягування. Параметр *Sender* містить назву перетягуваного об'єкту. Змінна *DragObject* як усталено дорівнює NULL. Це означає, що переносити будуть сам компонент або його об'єкт.

Література

1. Рудик О. Налаштування взаємодії wxWidgets і CodeBlocks для забезпечення осучаснення навчання інформатики. — Комп'ютер у школі та сім'ї. — № 3 (147), 2018. — с. 16–20.

2. Ильин Е. Компиляция библиотеки wxWidgets в Code::Blocks. — Режим доступу: https://jenyay.net/Programming/Wx

3. C++ Builder. Інтернет підручник для всіх. — Режим доступу: https://cubook.supernew.org/manual-c/events

References. Translation and transliteration

1.Rudyk A. Setting up wxWidgets and CodeBlocks interactions for providing the studying of informatics training. — Computer at School and Family. — № 3 (147), 2018. — c. 16–20.

2.Ilyin E. Compiling the wxWidgets library in Code::Blocks. — Access code:https://jenyay.net/Programming/Wx

3.C++ Builder. Online tutorial for everyone. — Access code: https://cubook.supernew.org/manual-c/events

CREATION OF THE WXWIDGETS PROJECT AND DESCRIPTION OF EVENTS WITH C++

Rudyk Alexander Borisovich
Ph.D (physics and mathematics), associate professor,
Institute of Postgraduate Pedagogical Education of named Boris Grinchenko Kyiv University,
rudykob@gmail.com

Holodova Olena Stanislavovna
the teacher of the specialized school № 120 of the city of Kyiv,
helenholodova@gmail.com

Annotation. The creation of a simple event and object-oriented wxWidgets project in the CodeBlocls environment is described as part of some of the practical work. Its usage does not need reliable student knowledge and skills of programming and even e teacher intervention. A list of the most frequently used events with their interpretation is given.

***Keywords*:** event oriented and object-oriented programming, C++, CodeBlocks, wxWidgets, project creation.

СОЗДАНИЕ ПРОЕКТА WXWIDGETS И ОПИСАНИЕ СОБЫТИЙ В С ++

Рудик Александр Борисович,
кандидат физико-математических наук, доцент,
Институт последипломного педагогического образования
Киевского университета имени Бориса Гринченко,
rudykob@gmail.com

Холодовая Елена Станиславовна

Учитель,
Специализированная школа I-III ступеней № 120 с углубленным изучением предметов естественно-математического цикла, г. Киев,
helenholodova@gmail.com

Аннотация. Описано создание простого событийно и объектно ориентированного проекта wxWidgets в среде CodeBlocls в виде описания части практической работы. Ее использование как учебного материала не предусматривает добротных знаний и умений ученика по программированию и даже вмешательство учителя. Перечислены наиболее часто используемых события с их толкованием.

***Ключевые слова*:** событийно и объектно ориентированное программирование, С ++, CodeBlocks, wxWidgets, создание проекта.

Продовження серії статей у наступних номерах

* * *

УДК 378.14

ІНФОРМАЦІЙНА СИСТЕМА МОНІТОРИНГУ ПРОЦЕСУ ПІДВИЩЕННЯ КВАЛІФІКАЦІЇ У ЗАКЛАДІ ПІСЛЯДИПЛОМНОЇ ПЕДАГОГІЧНОЇ ОСВІТИ

Грабовський Петро Петрович
кандидат педагогічних наук, старший викладач кафедри педагогіки й андрагогіки, Комунальний заклад "Житомирський обласний інститут післядипломної педагогічної освіти" Житомирської обласної ради ,
GrabovskyP@gmail.com
ORCID ID 0000-0002-2555-3678

Анотація. У статті здійснено концептуальне проектування бази даних, що може слугувати основою розбудови інформаційної системи моніторингу процесу підвищення кваліфікації у закладі післядипломної педагогічної освіти. Реалізація та впровадження цієї бази даних забезпечує автоматизацію обліку слухачів, проведення аналізу ефективності процесу навчання під час підвищення кваліфікації за допомогою методів математичної статистики.

Ключові слова: післядипломна педагогічна освіта; процес підвищення кваліфікації педагогів; інформаційна система; моніторинг; концептуальне проектування бази даних

Постановка проблеми. Освіта – одна із найважливіших сфер життєдіяльності сучасного суспільства. Тому, як і будь-який інший продукт людської діяльності, має свою якість. Від рівня цієї якості залежить сталість розвитку суспільства, гуманізація суспільно-економічних відносин, а також формування нових життєвих орієнтирів особистості. Згідно Національної стратегії розвитку освіти в Україні на період до 2021 року одним із пріоритетних стратегічних напрямів розвитку є забезпечення проведення національного моніторингу цієї системи: якості освіти, якості навчальних досягнень, структури тощо.

Водночас, розвиток інформаційно-комунікаційних технологій (ІКТ) та інформаційної компетентності, як працівників освітньої сфери так і пересічних громадян, обумовило доцільність та необхідність впровадження цих технологій у процес моніторингу для автоматизації рутинних операцій, щодо опрацювання отриманих даних. Зазначене потребує проектування та реалізації відповідних програмних засобів, зокрема інформаційних систем (ІС) для потреб середніх загальноосвітніх закладів, вищих навчальних закладів та закладів післядипломної педагогічної освіти.

Аналіз останніх досліджень і публікацій. Визначеної проблем стосується значна кількість наукових праць, серед яких виділяємо роботи В. Ю. Бикова, Д. М. Бодненка, В. С. Понамаренка, А. В. Співаковського, О. М. Спіріна, М. С. Львова, Л. О . Щоголевої, Г. В. Єльникової.

Науковцями визначено сутність базових понять [1; 2; 3; 4; 7], що знайшли своє відображення у нормативних документах [5;6]; описано структурні елементи, класифікацію та тенденції розвитку інформаційних систем [1]; пропонується організація процесу моніторингу освітньої діяльності навчального закладу[2; 4], результатів науково-педагогічних досліджень, зокрема і з використанням ІС [9; 10]; описано управління або підтримка науково-освітньої діяльності навчального закладу з використання ІС [5;7; 8].

На сьогоднішній день загальнодержавною ІС освітньої галузі є інформаційно-виробнича система «Освіта» [11], що була створена на вимогу постанови Кабінету Міністрів України. Ця система забезпечує побудову єдиного інтегративного інформаційного середовища країни в галузі освіти та являє собою комплекс адміністративних, правових, програмних та апаратно-