

Bartosz W. Warzocha

PROGRAMOWANIE WIELOPLATFORMOWE z C++ i wxWidgets 3



Programowanie wieloplatformowe z C++ i wxWidgets 3

Bartosz W. Warzocha

Programowanie wieloplatformowe z C++ i wxWidgets 3

Projekt okładki i stron tytułowych **Hubert Zacharski**

Fotografia na okładce **sanchopancho/Adobe Stock**

Wydawca **Edyta Kawala**

Redaktor prowadzący **Jolanta Kowalczuk**

Redaktor **Irena Puchalska**

Koordynator produkcji **Anna Bączkowska**

Skład i łamanie **ALINEA**

Zastrzeżonych nazw firm i produktów użyto w książce
wyłącznie w celu identyfikacji

Książka, którą nabyłeś, jest dziełem twórcy i wydawcy. Prosimy, abyś przestrzegał praw, jakie im przysługują. Jej zawartość możesz udostępnić nieodpłatnie osobom bliskim lub osobiście znanym. Ale nie publikuj jej w internecie. Jeśli cytujesz jej fragmenty, nie zmieniaj ich treści i koniecznie zaznacz, czyje to dzieło. A kopiując jej część, rób to jedynie na użytek osobisty.

Szanujmy cudzą własność i prawo
Więcej na www.legalnakultura.pl
Polska Izba Książki

Copyright © by Wydawnictwo Naukowe PWN
Warszawa 2018

ISBN 978-83-01-19899-2

Wydanie I
Warszawa 2018

Wydawnictwo Naukowe PWN SA
02-460 Warszawa, ul. Gottlieba Daimlera 2
tel. 22 69 54 321, faks 22 69 54 288
infolinia 801 33 33 88
e-mail: pwn@pwn.com.pl, reklama@pwn.pl
www.pwn.pl

Druk i oprawa:

Spis treści

Od autora	XXVII
Wprowadzenie	XXIX
Część I Poznajemy wxWidgets 3	1
1. Biblioteka	3
2. Przygotowanie projektu	34
3. Pierwsze okno	63
4. Obsługa zdarzeń	80
5. Zanim przejdziemy dalej...	116
Część II Zdobywamy obszar klienta	121
6. Wisielec, czyli podstawowe kontrolki i techniki wxWidgets	123
7. Własny edytor C++, czyli bardziej zaawansowane kontrolki i techniki wxWidgets	239
8. Level completed, czyli co jeszcze w trawie piszczy?	459
Część III Potęga dialogu	463
9. Standardowe okna i funkcje dialogowe oraz inne techniki interakcji z użytkownikiem ..	465
10. Walidacja danych wejściowych	545
11. Aplikacja Bibliotekarz i realizacja własnych okien dialogowych	570
Część IV Zaawansowane programowanie wxWidgets	633
12. Tajemnice pracy z ciągami znaków w wxWidgets, czyli klasa wxString i nie tylko... ..	635
13. Data i czas w wxWidgets	662
14. Krótka opowieść o przydatnych szablonach i pseudoszablonach wxWidgets	704
15. Praca z plikami i systemem plików w wxWidgets	719
16. Krótki przegląd niektórych zaawansowanych kontrolki GUI	769
17. Tworzenie własnych generycznych kontrolki GUI	825
18. Techniki i modele realizacji GUI aplikacji	868
19. Drukowanie w wxWidgets od kuchni	914
20. Krótki rozdział o technikach przechowywania konfiguracji programu	924
21. wxWidgets i programowanie współbieżne	933
22. wxWidgets i konsola	947
23. Co jest grane? O logach w wxWidgets	959

24. Nie tylko po polsku, czyli internacjonalizacja aplikacji	966
25. Programowanie sieciowe z wxWidgets	977
26. wxWidgets i nowoczesny OpenGL	1008
27. Zagadnienia programowania bazodanowego z wxWidgets 3+	1016
28. Bonus na koniec, czyli jak wykonać ekran startowy	1042
Zakończenie	1047
Dodatek A Krótki leksykon standardowych elementów GUI wxWidgets	1049
Dodatek B Niektóre ważniejsze rozszerzenia wxWidgets	1087
Dodatek C Licencje	1090
Indeks	1097

Szczegółowy spis treści

Od autora	XXVII
Podziękowania	XXVIII
Wprowadzenie	XXIX
Zakres tematyczny książki	XXIX
Ważna uwaga o konstrukcji książki, czyli zanim kupisz	XXIX
Dla kogo jest ta książka?	XXXI
Konwencje przyjęte w książce	XXXI
Warsztat	XXXIII
MS Windows	XXXIII
Microsoft Visual C++	XXXIII
Code::Blocks	XXXIII
Inne	XXXIV
PoEdit	XXXIV
GIMP	XXXIV
Materiały dodatkowe do książki	XXXIV
Errata i aktualizacje	XXXIV
Notka prawna	XXXV
Część I Poznajemy wxWidgets 3	1
1. Biblioteka	3
1.1. Czym jest wxWidgets?	3
1.2. Pobieranie i instalacja wxWidgets	4
1.2.1. Konwencje numerowania wersji biblioteki	5
1.2.2. Instalacja wxWidgets w systemach MS Windows	5
1.2.2.1. Zmienna systemowa WXWIN	5
1.2.3. Instalacja wxWidgets w systemach Ubuntu i Mint	6
1.2.3.1. Ręczna instalacja biblioteki z archiwum <i>tar</i>	6
1.2.3.2. Instalacja wxWidgets z pakietu Ubuntu/Debian	10
1.3. Foldery i pliki wxWidgets	11
1.4. Kompilacja wxWidgets	13
1.4.1. Plik <i>setup.h</i> i standardowe parametry kompilacji	13
1.4.1.1. Ustawienia globalne (<i>global settings</i>)	13
1.4.1.2. Ustawienia kompatybilności wersji (<i>compatibility settings</i>)	14
1.4.1.3. Ustawienia debugowania (<i>debugging settings</i>)	14
1.4.1.4. Wsparcie standardu Unicode (<i>Unicode support</i>)	14
1.4.1.5. Funkcje globalne (<i>global features</i>)	14
1.4.1.6. Współpraca z biblioteką standardową (<i>Interoperability with the standard library</i>)	15
1.4.1.7. Opcje niezwiązane z GUI (<i>Non GUI features selection</i>)	16
1.4.1.8. Kontrolki GUI (<i>Individual GUI controls</i>)	20
1.4.1.9. Różności GUI (<i>Miscellaneous GUI Stuff</i>)	23
1.4.1.10. Okna dialogowe (<i>common dialogs</i>)	24

1.4.1.11.	Wsparcie metaplików (<i>Metafiles support</i>)	25
1.4.1.12.	Duże kontrolki GUI (<i>Big GUI controls</i>)	26
1.4.1.13.	Przetwarzanie danych (<i>Data transfer</i>)	26
1.4.1.14.	Ustawienia różne (<i>Miscellaneous settings</i>)	27
1.4.1.15.	Klasy wxDC (<i>wxDC classes for various output formats</i>)	27
1.4.1.16.	Wsparcie dla formatów graficznych (<i>image format support</i>)	28
1.4.1.17.	Inne opcje pliku setup.h	28
1.4.1.18.	Opcje zależne od platformy	29
1.4.2.	Kompilacja wxWidgets w systemach MS Windows	29
1.4.3.	Kompilacja wxWidgets w systemach Ubuntu Linux i Mint Linux	29
1.5.	Struktura biblioteki wxWidgets	29
1.6.	Słowo o konwencjach i stylu kodowania w wxWidgets	30
1.6.1.	Pliki	31
1.6.2.	Konwencje nazewnicze	31
1.6.3.	Styl kodowania	31
1.6.4.	Jakość kodu	32
1.6.5.	Ograniczenia C++	32
1.6.6.	Inne	33
2.	Przygotowanie projektu	34
2.1.	Visual C++ (MS Windows)	34
2.1.1.	Utworzenie projektu	35
2.1.2.	Konfiguracja projektu	36
2.1.2.1.	Konfiguracja x86 (Win32)	37
2.1.2.2.	Konfiguracja x64	43
2.1.3.	Dodanie plików źródłowych	50
2.1.4.	Zapisanie szablonu projektu	50
2.2.	Code::Blocks (Ubuntu i Mint)	52
2.2.1.	Utworzenie projektu	52
2.2.2.	Konfiguracja projektu	54
2.2.2.1.	Ogólne opcje projektu	54
2.2.2.2.	Opcje kompilacji i konsolidacji	55
2.2.3.	Dodanie plików źródłowych	56
2.2.4.	Zapisanie szablonu projektu	58
2.2.5.	Rozwiązywanie problemów z kodowaniem plików	58
2.2.6.	Rozwiązywanie problemów z wieloma wersjami wxWidgets	58
2.2.6.1.	Wybór wersji wxWidgets za pomocą wx-config	58
2.2.6.2.	Wybór wersji wxWidgets za pomocą własnego pliku Makefile	59
2.3.	O projektach Code::Blocks i wxWidgets w systemach MS Windows	61
2.4.	Podsumowanie	62
3.	Pierwsze okno	63
3.1.	Ogólna struktura aplikacji	63
3.2.	Szczegóły głównego okna programu	66
3.2.1.	Modyfikowanie konstruktora	68
3.3.	Pasek stanu	69
3.3.1.	Funkcja CreateStatusBar()	69
3.3.2.	Więcej opcji paska stanu	70
3.3.2.1.	Modyfikacja rozmiarów pól paska stanu	70
3.3.2.2.	Modyfikacja liczby pól paska stanu	72
3.3.3.	Zaawansowane tworzenie paska stanu	72
3.4.	Pasek menu	73
3.4.1.	Realizacja paska menu	73
3.4.2.	Sprytne odmiany funkcji Append()	76
3.4.3.	Skróty klawiaturowe menu	78
3.4.4.	Dygresja graficzna, czyli słowo o ikonach w menu	79
3.4.5.	Co dalej?	79

4. Obsługa zdarzeń	80
4.1. Wprowadzenie do obsługi zdarzeń	80
4.1.1. Identyfikatory	80
4.1.2. Funkcje akcji programu	81
4.1.3. Tablica zdarzeń	83
4.1.4. Uzupełniamy program modelowy	84
4.2. Wokół pętli zdarzeń...	86
4.3. Niektóre metody klasy wxCommandEvent	88
4.4. Rozpoznawanie inicjatora zdarzenia za pomocą funkcji GetId()	90
4.4.1. Składanie zdarzeń	90
4.4.2. Zakresowe makra obsługi zdarzeń	91
4.5. Dynamiczna obsługa zdarzeń	92
4.5.1. Odrobina klasyki, czyli funkcje Connect() i Disconnect()	92
4.5.2. Z duchem czasu, czyli funkcje Bind() i Unbind()	94
4.5.3. Standard C++11 i obsługa zdarzeń wxWidgets	96
4.6. Wbudowane identyfikatory wxWidgets	97
4.6.1. Wbudowane identyfikatory wxWidgets i odkrywanie tajemnic menu	98
4.7. Wyjątkowe zdarzenie wxEVT_CLOSE_WINDOW	99
4.7.1. Realizacja pytania o wyjście z programu	99
4.8. Zdarzenia myszy i klawiatury	100
4.8.1. Obsługa zdarzeń myszy	100
4.8.2. Obsługa zdarzeń klawiatury	103
4.8.2.1. Problem globalnej obsługi zdarzeń klawiatury	106
4.9. Inne rodzaje zdarzeń	108
4.10. Pisanie własnych zdarzeń	110
4.10.1. Kiedy i gdzie stosować własne zdarzenia?	110
4.10.2. Realizacja własnych zdarzeń	111
4.10.2.1. Podstawowa realizacja własnych zdarzeń	111
4.10.2.2. Zaawansowana realizacja własnych zdarzeń	112
5. Zanim przejdziemy dalej...	116
5.1. Kod aplikacji wxWidgets w różnych plikach	116
5.2. Podobieństwo konstruktorów i metod, czyli słowo o tym, jak łatwo i szybko opanować wxWidgets	116
5.3. Konstruktor czy Create()?	119
Część II Zdobywamy obszar klienta	121
6. Wisielec, czyli podstawowe kontrolki i techniki wxWidgets	123
6.1. Koncepcja programu	124
6.2. Kilka uwag o menu <i>Wisielca</i>	125
6.3. Klasa wxPanel – podstawowy kontener GUI	126
6.4. Grafika w programie: wxBitmap, wxImage i wxIcon	127
6.4.1. Procedury obsługi grafiki	127
6.4.2. Klasa wxBitmap	128
6.4.2.1. Tworzenie bitmapy i inicjowanie jej danymi	129
6.4.2.2. Tajemniczy format XPM	130
6.4.2.3. Obiekty wxBitmap i grafika wbudowana	131
6.4.2.4. Operacje na wxBitmap	133
6.4.3. Klasa wxImage	135
6.4.3.1. Tworzenie obiektów wxImage	135
6.4.3.2. Operacje na wxImage	136
6.4.3.3. Przekształcanie bitmapy wxBitmap za pomocą wxImage	137
6.4.4. Klasa wxIcon	138
6.5. Lokalizacja plików aplikacji w różnych systemach operacyjnych	139
6.5.4.1. Pomocne funkcje narzędziowe i plik <i>utils.h</i>	142
6.6. Ustawiamy ikonę programu	143

6.7.	Pasek narzędzi wxToolBar	144
6.7.1.	Realizacja paska narzędzi	144
6.7.2.	Zarządzanie paskiem narzędzi	148
6.7.3.	Menu w pasku narzędzi	150
6.7.4.	Kontrolki w pasku narzędzi	151
6.8.	Statyczne elementy GUI	153
6.8.1.	Klasa wxStaticText	153
6.8.2.	Klasa wxStaticBitmap	154
6.8.3.	Klasa wxStaticLine	155
6.8.4.	Klasa wxStaticBox	155
6.9.	Wprowadzenie do programowania GUI opartego na relacjach	156
6.9.1.	Czym są sizery?	157
6.9.2.	Klasa wxBoxSizer	157
6.9.3.	Klasa wxStaticBoxSizer	159
6.9.4.	Analiza przypadków	159
6.9.4.1.	Przypadek 1 – prosty sizer liniowy	159
6.9.4.2.	Przypadek 2 – klasyczne okno dialogowe	162
6.9.4.3.	Przypadek 3 – koncepcja GUI <i>Wisielca</i>	163
6.9.5.	Składamy szkielet GUI <i>Wisielca</i>	165
6.10.	Czcionki i kolory w wxWidgets	169
6.10.1.	Klasa wxFont	169
6.10.1.1.	Modyfikacja utworzonych czcionek	171
6.10.1.2.	Użycie czcionek wxFont	172
6.10.1.3.	Ustawiamy czcionki <i>Wisielca</i>	173
6.10.2.	Klasa wxColour, baza kolorów i inne kolorowe ciekawostki	173
6.10.2.1.	Przetwarzanie danych koloru	176
6.10.2.2.	Słowo o kolorach z <i>wxStockGDI</i>	177
6.10.2.3.	Kolory systemowe	178
6.10.2.4.	Użycie kolorów wxWidgets	179
6.10.2.5.	Ustawiamy kolory GUI <i>Wisielca</i>	180
6.11.	Wstęp do podstawowych kontrolek dynamicznych	181
6.11.1.	Okno tekstowe wxTextCtrl	182
6.11.1.1.	Tworzymy kontrolkę tekstową wxTextCtrl	182
6.11.1.2.	Ważniejsze metody klasy wxTextCtrl	184
6.11.1.3.	Zdarzenia klasy wxTextCtrl i ich obsługa	187
6.11.2.	Świat przycisków wxWidgets	189
6.11.2.1.	Klasa wxButton	189
6.11.2.2.	Klasa wxToggleButton	191
6.11.2.3.	Klasa wxBitmapButton	192
6.11.3.	Lista rozwijana wxComboBox	192
6.11.3.1.	Klasa wxArrayString	193
6.11.3.2.	Realizacja listy rozwijanej	194
6.11.3.3.	Niektóre metody klasy wxComboBox	195
6.11.3.4.	Obsługa zdarzeń listy rozwijanej	196
6.11.4.	Kontrolki numeryczne wxSpinCtrl i wxSpinCtrlDouble	197
6.11.4.1.	Klasa wxSpinCtrl	197
6.11.4.2.	Klasa wxSpinCtrlDouble	198
6.11.4.3.	Style kontrolki numerycznych	199
6.11.4.4.	Ważniejsze metody kontrolki numerycznych	199
6.11.4.5.	O zdarzeniach kontrolki numerycznych	200
6.12.	Inne kontrolki wxWidgets	201
6.12.1.	Jak korzystać z dokumentacji klas wxWidgets	202
6.13.	Kompletujemy GUI <i>Wisielca</i>	203
6.13.1.	Log	203
6.13.2.	Ustawienia	205
6.14.	Rysowanie szubienicy, czyli tajemnice klasy wxDC	207
6.14.1.	Dostępne konteksty urządzeń	207

6.14.2.	Rysowanie z wxDC w praktyce	208
6.14.2.1.	Pióro i pędzel	208
6.14.2.2.	Metody rysujące	210
6.14.3.	Pierwsze rysunki	214
6.14.3.1.	Rysowanie w obszarze klienta (wxClientDC)	214
6.14.3.2.	Rysowanie za pomocą myszy	218
6.14.4.	Znaczenie zdarzenia wxEVT_PAINT i wieloplatformowość	219
6.14.4.1.	Uniwersalny wieloplatformowy blok rysunkowy	220
6.14.5.	Piszemy funkcję rysującą szubienicę	221
6.15.	Wykończyć Wisielca!	224
6.15.1.	Założenia	224
6.15.2.	Niezbędne zmienne i funkcje	225
6.15.3.	Wprowadzanie odgadywanych liter z klawiatury	226
6.15.4.	Obsługa logu	227
6.15.5.	Mierzenie czasu	228
6.15.6.	Ustawienie stanu początkowego gry	229
6.15.7.	Funkcja wyświetlająca hasło	230
6.15.8.	Rozpoczęcie gry	230
6.15.9.	Przebieg rozgrywki	232
6.15.10.	Zakończenie gry	236
6.15.11.	Przerwanie gry	237
6.16.	Podsumowanie	238
7.	Własny edytor C++, czyli bardziej zaawansowane kontrolki i techniki wxWidgets	239
7.1.	Koncepcja programu	240
7.2.	Przygotowanie szkieletu aplikacji	241
7.2.1.	Tworzymy główne okno	241
7.2.2.	Realizujemy pasek menu z ikonami	242
7.2.3.	Pasek narzędzi	247
7.2.4.	Domykamy szkielet aplikacji	248
7.2.5.	Zanim rozpoczniemy pracę nad GUI edytora	251
7.3.	Kontener wxSplitterWindow	251
7.3.1.	Wstawiamy kontener do programu	253
7.3.2.	Wstawiamy kontener prawej kolumny	255
7.3.3.	Pozostałe ważniejsze metody klasy wxSplitterWindow, o których warto wiedzieć	255
7.3.4.	Obsługa zdarzeń kontenera wxSplitterWindow	257
7.3.5.	Edytor C++ – przygotowanie sizerów obszarów GUI	258
7.3.6.	Multi Document Interface (MDI) i wxAUI, czyli inne podejście do organizacji GUI	258
7.4.	Drzewo projektu, czyli klasa wxTreeCtrl	259
7.4.1.	Jak działa drzewo wxTreeCtrl?	260
7.4.2.	Trochę teorii o drzewach wxTreeCtrl	260
7.4.3.	Obsługa zdarzeń klasy wxTreeCtrl	262
7.4.4.	Ikony drzewa i lista wxImageList	262
7.4.5.	Operacje drzewa wxTreeCtrl	265
7.4.5.1.	Dodawanie, wstawianie i usuwanie elementów drzewa	265
7.4.5.2.	Modyfikowanie widoku drzewa	267
7.4.5.3.	Zarządzanie drzewem i jego elementami	268
7.4.6.	Tworzymy drzewo plików projektu edytora C++	279
7.4.6.1.	Przygotowanie struktury danych plików projektu C++	279
7.4.6.2.	A co z plikami spoza projektu?	281
7.4.6.3.	Klasa drzewa projektu, czyli ProjectTreeCtrl	282
7.4.6.4.	Wyszukiwanie plików projektu w drzewie	283
7.4.6.5.	Wyszukiwanie innych elementów drzewa	285
7.4.6.6.	Wstawianie elementów do drzewa	286

	7.4.6.7. Inne przydatne metody	287
	7.4.6.8. Co dalej z drzewem projektu?	287
7.5.	Tworzymy plik projektu, czyli praca z biblioteką wxXML	288
7.5.1.	Błyskawiczne wprowadzenie do języka XML	288
7.5.2.	Struktura XML pliku projektu	289
7.5.3.	Poznajemy bibliotekę składową wxXML i jej klasy	291
7.5.3.1.	Klasa wxXmlDocument	291
7.5.3.2.	Klasa wxXmlNode i znaczniki XML	293
7.5.3.3.	Techniki definiowania znaczników XML z użyciem wxXmlNode	295
7.5.3.4.	Przegląd niektórych przydatnych metod klasy wxXmlNode	298
7.5.3.5.	Odczyt, przetwarzanie i zapis pliku XML	299
7.5.3.6.	Usprawniamy pracę, czyli własna klasa dokumentu XML	305
7.5.4.	Realizacja pliku projektu i jego obsługi w edytorze C++	310
7.5.4.1.	Okno wyjścia konsoli	310
7.5.4.2.	Drzewo i jego ikony	311
7.5.4.3.	Tworzenie nowego pliku projektu	312
7.5.4.4.	Otwieranie istniejącego pliku projektu	319
7.5.4.5.	Dodawanie plików do projektu	324
7.5.4.6.	Zmiana nazwy pliku w projekcie	329
7.5.4.7.	Usuwanie plików z projektu	331
7.5.4.8.	A co z pozostałymi operacjami na plikach?	332
7.5.4.9.	Temat nadobowiązkowy – menu kontekstowe dla plików w drzewie projektu	333
7.6.	wxBookCtrlBase i klasy kontenerów pochodnych	334
7.6.1.	Klasa wxBookCtrlBase	334
7.6.2.	Klasa zdarzeń wxBookCtrlEvent	336
7.6.3.	Przegląd kontenerów opartych na klasie wxBookCtrlBase	337
7.6.3.1.	Klasa wxNotebook	337
7.6.3.2.	Klasa wxChoicebook	339
7.6.3.3.	Klasa wxListbook	340
7.6.3.4.	Pozostałe kontenery wxBookCtrl	341
7.6.3.5.	Sposoby korzystania z kontenerów wxBookCtrl	342
7.6.3.6.	Utrudnienia w korzystaniu z kontenerów wxBookCtrl wraz z elementami języka C++11 i nowszych	347
7.6.3.7.	Dodajemy kontener do edytora C++	347
7.7.	Serce edytora, czyli klasa wxStyledTextCtrl	348
7.7.1.	Przygotowanie biblioteki wxSTC	349
7.7.2.	Tworzenie kontrolki edytora wxStyledTextCtrl i jej konfiguracja	349
7.7.2.1.	Podstawowy konstruktor klasy wxStyledTextCtrl	349
7.7.2.2.	Podstawowa konfiguracja kontrolki wxStyledTextCtrl	350
7.7.3.	Zdarzenia klasy wxStyledTextCtrl i ich obsługa	355
7.7.4.	Przegląd istotnych możliwości klasy wxStyledTextCtrl	358
7.7.4.1.	Słowo o metodach reimplementowanych z wxTextEntry i innych metodach związanych z edycją	358
7.7.4.2.	Składanie kodu (code folding)	361
7.7.4.3.	Autouzupełnianie kodu (<i>code autocompletion</i>)	367
7.7.4.4.	Notatki i komunikaty w kodzie	373
7.7.4.5.	Podświetlanie pasujących nawiasów	378
7.7.5.	Wnioski	381
7.8.	Ten moment, gdy serce zaczyna bić...	381
7.8.1.	Otwieranie plików w edytorze	382
7.8.1.1.	Przygotowania	382
7.8.1.2.	Funkcja obsługująca składanie kodu	386
7.8.1.3.	Funkcja otwierająca dokumenty w nowych oknach edytora	386
7.8.1.4.	Wyświetlanie pliku dodawanego do projektu	387
7.8.1.5.	Wyświetlanie pliku aktywowanego w drzewie projektu	388
7.8.1.6.	Wyświetlanie pliku spoza projektu	390

7.8.2.	Działania związane z edycją plików	390
7.8.2.1.	Wykrywanie edycji i zmiana ikony pliku	390
7.8.2.2.	Metody edycyjne	391
7.8.3.	Wyszukiwanie i zamiana ciągów znaków	392
7.8.3.1.	Wyświetlenie okna dialogowego wyszukiwania i zamiany ciągów znaków	393
7.8.3.2.	Metoda wyszukiwująca	394
7.8.3.3.	Realizacja wyszukiwania ciągów znaków	396
7.8.3.4.	Realizacja zamiany ciągów znaków	398
7.8.3.5.	Właściwe zamykanie okna dialogowego zamiany	399
7.8.4.	Zapis i zamykanie dokumentów	400
7.8.4.1.	Ogólne metody zapisujące	400
7.8.4.2.	Zapis na życzenie użytkownika	402
7.8.4.3.	Zamykanie plików aktywnego projektu	403
7.8.4.4.	Zamykanie aplikacji a kontrola niezapisanych plików	404
7.8.5.	Ostatnie uzupełnienia	404
7.8.6.	Wnioski	405
7.9.	Zgłębiamy tajemnice listy wxListCtrl	406
7.9.1.	Tworzenie listy wxListCtrl i tryby jej pracy	407
7.9.1.1.	Pozostałe style klasy wxListCtrl	408
7.9.1.2.	Ikony listy	408
7.9.1.3.	Elementy listy, czyli klasa wxListItem	409
7.9.2.	Obsługa zdarzeń klasy wxListCtrl	411
7.9.3.	Ważniejsze metody klasy wxListCtrl	412
7.9.3.1.	Zarządzanie elementami listy	412
7.9.3.2.	Dane powiązane z elementem listy	414
7.9.3.3.	Korzystanie z przełączników typu checkbox w obrębie listy	415
7.9.3.4.	Praca z listą w trybie szczegółowym	416
7.9.3.5.	Testowanie elementów listy (na przykładzie listy szczegółowej) ..	419
7.9.3.6.	Szersze spojrzenie na atrybuty elementów listy wxListCtrl i klasa wxListItemAttr	420
7.9.4.	Zaawansowane zabawy z listą wxListCtrl	420
7.9.4.1.	Przemieszczanie wierszy listy w trybie szczegółowym	421
7.9.4.2.	Sortowanie listy wxListCtrl	425
7.9.4.3.	wxListCtrl jako lista rozwijana kontrolki typu 'combobox'	432
7.9.5.	Inne klasy list dostępnych w wxWidgets	436
7.9.6.	Tworzymy listę statystyk plików programu wxC++	440
7.9.6.1.	Tworzenie listy statystyki	441
7.9.6.2.	Aktualizacja danych listy statystyk	441
7.10.	Przełącznik wxCheckBox	443
7.10.1.	Tworzenie i tryby pracy przełączników wxCheckBox	444
7.10.1.1.	Style i obsługa zdarzeń klasy wxCheckBox	444
7.10.1.2.	Ważniejsze metody klasy wxCheckBox	444
7.10.2.	Korzystanie z przełącznika wxCheckBox (na przykładzie aplikacji wxC++) ..	445
7.11.	Szybkie wprowadzenie do drukowania w wxWidgets	446
7.11.1.	Klasa wxHtmlEasyPrinting i inicjowanie środowiska drukowania	447
7.11.2.	Konfiguracja wydruku	448
7.11.3.	Przygotowanie tekstu do drukowania	449
7.11.4.	Wydruk, podgląd wydruku oraz ustawienia strony	450
7.11.5.	<i>Littera scripta manet</i>	452
7.12.	Pozostałe uzupełnienia i funkcje programu wxC++	452
7.12.1.	Implementacja trybu pełnoekranowego	452
7.12.2.	Prosty kreator klas C++	453
7.13.	Aplikacja wxC++ – zakończenie	454
7.13.1.	Propozycje udoskonaleń programu	455
8.	Level completed, czyli co jeszcze w trawie piszczy?	459

Część III Potęga dialogu	463
9. Standardowe okna i funkcje dialogowe oraz inne techniki interakcji z użytkownikiem ..	465
9.1. Okna dialogowe związane z pobieraniem danych od użytkownika	466
9.1.1. Pobieranie tekstu	467
9.1.1.1. Pobieranie zwykłego tekstu	467
9.1.1.2. Pobieranie hasła (tekstu ukrytego)	468
9.1.1.3. Ograniczenia funkcji wxGetTextFromUser() i wxGetPasswordFromUser	468
9.1.1.4. Klasa wxTextEntryDialog	468
9.1.1.5. Klasa wxPasswordEntryDialog	470
9.1.2. Pobieranie wartości numerycznych	470
9.2. Okna dialogowe związane z pobieraniem wyborów użytkownika	471
9.2.1. Wybór kroju czcionki	471
9.2.1.1. Klasa wxFontDialog	473
9.2.2. Wybór znaku	474
9.2.3. Wybór koloru	476
9.2.3.1. Klasa wxColourDialog	477
9.2.4. Wybór jednej lub wielu opcji	478
9.2.4.1. Funkcje dialogowe wyboru pojedynczego	478
9.2.4.2. Sprytnie funkcje wyboru pojedynczego	481
9.2.4.3. Klasa wxSingleChoiceDialog	482
9.2.4.4. Funkcje dialogowe wyboru wielokrotnego	484
9.2.4.5. Klasa wxMultiChoiceDialog	485
9.2.5. Definiowanie sztyku danych	486
9.2.5.1. Klasa wxRearrangeDialog	486
9.2.5.2. Klasach wxRearrangeCtrl i wxRearrangeList	488
9.3. Okna dialogowe systemu plików	488
9.3.1. Okna dialogowe do pracy z katalogami	489
9.3.1.1. Klasa wxDirDialog	489
9.3.2. Okna dialogowe do pracy z plikami	490
9.3.2.1. Tryby pracy dialogów plikowych	490
9.3.2.2. Funkcje dialogowe do pracy z plikami	491
9.3.2.3. Klasa wxFileDialog	494
9.4. Informacyjne okna dialogowe	498
9.4.1. Podstawowe okna informacyjne	498
9.4.1.1. Klasa wxMessageDialog	498
9.4.1.2. Funkcja wxMessageBox()	502
9.4.1.3. Funkcja wxInfoMessageBox()	502
9.4.2. Paski komunikatów	503
9.4.3. Okna informacji o programie	505
9.4.3.1. Funkcja wxAboutBox() i klasa wxAboutDialogInfo	505
9.4.4. Okno komunikatów diagnostycznych, czyli klasa wxLogGui	507
9.4.5. Okna dialogowe i techniki wizualizacji postępu	509
9.4.5.1. Klasy wxGenericProgressDialog i wxProgressDialog	509
9.4.5.2. Klasa wxAppProgressIndicator	512
9.4.6. Okna dialogowe i techniki wizualizowania zajętości	513
9.4.6.1. Wizualizacja zajętości przez wygląd kursora	513
9.4.6.2. Klasa wxBusyInfo	514
9.4.7. Powiadomienia	516
9.4.7.1. Klasa wxNotificationMessage	516
9.4.7.2. Metoda wxTopLevelWindow::RequestUserAttention()	519
9.4.8. Okna dialogowe związane z systemem pomocy i wsparciem	520
9.4.8.1. Porada dnia	520
9.4.8.2. Rozszerzone podpowiedzi	522
9.4.8.3. Okno dialogowe pomocy HTML	523

9.5.	Pozostałe okna dialogowe	527
9.5.1.	Arkusze właściwości	527
9.5.2.	Okno dialogowe wyszukiwania i zamiany ciągu znaków	530
9.5.3.	Okno dialogowe konfiguracji drukowania	531
9.5.4.	Kreator wxWizard	533
9.6.	Kontrolki pochodne od wxPickerBase	538
9.6.1.	Klasa wxColourPickerCtrl	538
9.6.2.	Klasa wxFontPickerCtrl	540
9.6.3.	Klasy wxDirPickerCtrl i wxFilePickerCtrl	541
9.6.4.	Przegląd ważniejszych wspólnych metod klas dziedziczących z wxPickerBase	544
9.7.	Podsumowanie i słowo o dialogowych kontrolkach GUI	544
10.	Walidacja danych wejściowych	545
10.1.	Aplikacja przykładowa i funkcja testująca	545
10.2.	Pasywne testy danych wejściowych	546
10.2.1.	Kontrola pasywna danych wejściowych kontrolki GUI	547
10.2.2.	Kontrola pasywna danych wejściowych w oknach dialogowych	547
10.3.	Aktywne testy danych wejściowych	548
10.3.1.	Kontrola aktywna danych wejściowych kontrolki GUI	548
10.3.2.	Kontrola aktywna danych wejściowych w oknach dialogowych	549
10.4.	Praca z walidatorami wxWidgets	551
10.4.1.	Klasa wxValidator	552
10.4.2.	Walidator tekstowy wxTextValidator	553
10.4.3.	Walidatory numeryczne wxIntegerValidator oraz wxFloatingPointValidator ..	555
10.4.4.	Walidator generyczny wxGenericValidator	557
10.4.5.	Walidacja wyrażeń regularnych	558
10.4.6.	Własny walidator uniwersalny	563
10.4.7.	Nieco więcej o transferze danych	568
10.5.	Podsumowanie rozdziału, czyli jakiej metody walidacji użyć?	568
11.	Aplikacja Bibliotekarz i realizacja własnych okien dialogowych	570
11.1.	Klasa wxDialog i jej cechy	570
11.1.1.	Tworzenie własnych klas okien dialogowych opartych na klasie <i>wxDialog</i> ..	571
11.1.2.	Zdarzenia okna dialogowego wxDialog	572
11.1.3.	Sposoby wyświetlania okien dialogowych wxDialog	572
11.1.3.1.	C++11 i nowe możliwości wyświetlania okien dialogowych	574
11.1.3.2.	Pozostałe metody związane z wyświetlaniem okna dialogowego ..	574
11.1.4.	Metody ułatwiające tworzenie GUI okien dialogowych opartych na klasie wxDialog	575
11.1.4.1.	Tworzenie obszaru przycisków	575
11.1.4.2.	Usprawnianie tworzenia pozostałych obszarów GUI	576
11.1.5.	Adaptacja dużych obszarów zawartości	576
11.1.6.	Pozostałe ważniejsze metody klasy wxDialog	579
11.2.	Przygotowanie aplikacji <i>Bibliotekarz</i>	579
11.2.1.	Główne założenia i przygotowanie do pracy nad programem	579
11.2.1.1.	Przygotowanie trzonu aplikacji	580
11.2.1.2.	Implementacja prostego systemu przechowywania konfiguracji programu	582
11.2.2.	Klasa biblioteki	586
11.2.2.1.	Dodanie biblioteki do programu	594
11.3.	Piszemy pierwsze okno dialogowe	595
11.3.1.	Zanim zaczniemy	596
11.3.1.1.	Horyzontalny sizer z etykietą	596
11.3.2.	Tworzymy klasę własnego okna dialogowego	597
11.3.3.	Piszemy konstruktor klasy BookDialog	599
11.3.4.	Pozostałe metody klasy BookDialog	604

11.3.5.	Korzystanie z okna dialogowego w programie	606
11.3.5.1.	Usuwanie książki z biblioteki	608
11.3.6.	Wnioski	609
11.4.	Współdziałanie własnych okien dialogowych z własnymi walidatorami	609
11.4.1.	Klasa LendBookDialog	609
11.4.2.	Implementacja wypożyczenia i oddawania książki	611
11.5.	Okna dialogowe oparte na klasie wxFrame	612
11.5.1.	Klasa MailConfigDialog	612
11.5.2.	Implementacja mechanizmu modalnego wyświetlania okna	615
11.5.3.	Obsługa standardowych przycisków	617
11.5.4.	Zastosowanie klasy MailConfigDialog w programie	618
11.5.5.	Wnioski	619
11.6.	Ostatnie podejście do pisania własnych okien dialogowych i zmiana zwracanego kodu	619
11.6.1.	Klasa SMTPConfigDialog	619
11.6.2.	Włączenie okna konfiguracji SMTP do aplikacji	622
11.7.	Kończymy aplikację <i>Bibliotekarz</i>	622
11.7.1.	Wybór między wxSocketBase, wxEMail a biblioteką POCO	622
11.7.2.	Pobieranie i kompilacja biblioteki POCO	623
11.7.2.1.	Kompilacja biblioteki POCO w MS Windows	623
11.7.2.2.	Kompilacja i instalacja biblioteki POCO w Linux	625
11.7.2.3.	Wersja binarna bibliotek dla MS Windows	625
11.7.3.	Dodanie biblioteki POCO do projektu	625
11.7.3.1.	MS Windows (Visual Studio)	625
11.7.3.2.	Linux (Code::Blocks)	626
11.7.3.3.	Pliki nagłówkowe	627
11.7.4.	Wysyłanie wiadomości e-mail za pomocą biblioteki POCO	627
11.8.	Podsumowanie i propozycje udoskonalenia aplikacji	631

Część IV Zaawansowane programowanie wxWidgets 633

12.	Tajemnice pracy z ciągami znaków w wxWidgets, czyli klasa wxString i nie tylko...	635
12.1.	Klasa wxString i jej stosowanie	635
12.1.1.	Tworzenie obiektów wxString	636
12.1.2.	Konkatenacja ciągów wxString	638
12.1.2.1.	Operatory konkatenacji	638
12.1.2.2.	Funkcje wstawiające	638
12.1.3.	Dostęp do elementów łańcucha wxString	639
12.1.3.1.	Metody dostępu	639
12.1.3.2.	Dostęp przez iteratory	639
12.1.4.	Konwersja wxString do innych typów i odwrotnie	640
12.1.4.1.	Konwersje między wxString a standardowymi typami tekstowymi oraz literałami łańcuchowymi	640
12.1.4.2.	Kodowanie znaków w konwersji ciągów znaków	642
12.1.4.3.	Konwersje między wxString a typami liczbowymi	644
12.1.5.	Testowanie ciągów wxString	646
12.1.6.	Przetwarzanie i edycja ciągów wxString	647
12.1.6.1.	Wyszukiwanie, zwracanie oraz zamiana ciągów znaków oraz ich fragmentów	647
12.1.6.2.	Zmiana wielkości znaków	649
12.1.6.3.	Usuwanie ciągu znaków i jego części	649
12.1.6.4.	Dzielenie ciągów wxString	650
12.1.7.	wxString i łańcuchy formatowane	652
12.1.7.1.	Formatowane inicjowanie obiektu wxString	652
12.1.7.2.	Formatowane wyjście wxString	652
12.1.7.3.	Ciągi wxString w standardowych funkcjach formatujących	653
12.1.8.	Makra wxT(), _T(), wxT_2(), wxS() oraz _()	653
12.1.9.	wxString i metody STL	653

12.2.	Wyrażenia regularne, czyli klasa wxRegEx	654
12.2.1.	Testowanie łańcuchów znaków	655
12.2.2.	Zmiana ciągów znaków za pomocą wyrażenia regularnego	655
12.3.	Gromadzenie ciągów znaków, czyli klasa wxString	656
12.3.1.	Tworzenie tablic ciągów znaków i zarządzanie ich elementami	656
12.3.2.	Tablice wxString a STL	657
12.3.3.	Testowanie tablic wxString	658
12.3.4.	Tablice wxString a zarządzanie pamięcią	658
12.3.5.	Sortowanie tablic wxString	659
12.4.	Klasa wxSortedArrayString	660
12.5.	Podsumowanie	661
13.	Data i czas w wxWidgets	662
13.1.	Klasa wxDateTime i jej stosowanie	662
13.1.1.	Tworzenie i inicjowanie obiektów wxDateTime	662
13.1.2.	Dostęp do danych daty i czasu	665
13.1.3.	Porównywanie wartości dat i czasu	666
13.1.4.	Arytmetyka dat i czasu, czyli klasy wxDateSpan oraz wxTimeSpan	668
13.1.4.1.	Klasa wxDateSpan i jej właściwości	668
13.1.4.2.	Klasa wxTimeSpan i jej właściwości	670
13.1.4.3.	Wykonywanie działań na obiektach wxDateTime	673
13.1.5.	Parsowanie i formatowanie daty i czasu	677
13.1.5.1.	Formatowanie obiektów daty i czasu do łańcuchów znaków	677
13.1.5.2.	Własne formatowanie daty i czasu	678
13.1.5.3.	Wyświetlanie tekstowych informacji o okresach	686
13.1.5.4.	Parsowanie łańcuchów znaków do obiektów daty i czasu	688
13.1.6.	Strefy czasowe i tzw. czas letni	690
13.1.7.	Pozostałe przydatne metody wxDateTime	692
13.1.7.1.	Aktualny czas i aktualna data	692
13.1.7.2.	Funkcje kalendarzowe	693
13.2.	Pobieranie daty i czasu od użytkownika	693
13.2.1.	Prezentacja klas wxCalendarCtrl oraz wxGenericCalendarCtrl	693
13.2.1.1.	Zdarzenia klas wxCalendarCtrl oraz wxGenericCalendarCtrl i ich obsługa	695
13.2.1.2.	Modyfikacja wyglądu kalendarza generycznego	695
13.2.1.3.	Święta i ich definiowanie	695
13.2.1.4.	Atrybuty dat kalendarza generycznego	696
13.2.1.5.	Inne przydatne metody klas wxCalendarCtrl i wxGenericCalendarCtrl	697
13.2.1.6.	Którą kontrolkę kalendarza wybrać?	698
13.2.1.7.	Przykład zastosowania i porównanie kontrolki wxCalendarCtrl i wxGenericCalendarCtrl w praktyce	698
13.2.2.	Prezentacja klasy wxDatePickerCtrl	700
13.2.2.1.	Przykład stosowania kontrolki wxDatePickerCtrl	701
13.2.3.	Prezentacja klasy wxTimePickerCtrl	701
13.2.3.1.	Przykład stosowania kontrolki wxTimePickerCtrl	702
13.3.	Podsumowanie	703
14.	Krótką opowieść o przydatnych szablonach i pseudoszablonach wxWidgets	704
14.1.	Klasy pojemnikowe	704
14.1.1.	Kontener podstawowy, czyli klasa wxVector<T>	704
14.1.2.	Stos, czyli klasa wxStack<T>	705
14.1.3.	Lista łączona, czyli klasy wxList<T> i wxNode<T>	706
14.1.4.	Kontener asocjacyjny wxHashMap	709
14.1.5.	Tablice wxWidgets, czyli klasa wxArray<T> i jej odmiany	711
14.2.	Sprytne wskaźniki	714
14.2.1.	wxSharedPtr<T>	714
14.2.2.	wxScopedPtr<T>	715

14.2.3.	Klasa wxScopedTiedPtr	716
14.2.4.	wxScopedArray<T>	716
14.2.5.	wxObjectDataPtr<T>	717
14.2.6.	wxWindowPtr<T>	717
14.2.7.	wxWeakRef< T > i wxWeakRefDynamic< T >	717
14.3.	Niektóre przydatne funkcje szablonowe	718
14.4.	Podsumowanie	718
15.	Praca z plikami i systemem plików w wxWidgets	719
15.1.	Niektóre funkcje zarządzania plikami i systemem plików	719
15.1.1.	Funkcje zarządzania katalogami	720
15.1.2.	Funkcje zarządzania plikami	721
15.1.3.	Inne funkcje systemu plików	724
15.2.	Specjalistka od katalogów, czyli klasa wxDir	725
15.2.1.	Tworzenie i inicjowanie obiektów wxDir	725
15.2.2.	Testowanie katalogów za pomocą wxDir	725
15.2.3.	Pobieranie informacji o danej lokalizacji	726
15.2.4.	Techniki listowania plików i katalogów z wxDir	726
15.2.4.1.	Listowanie plików i katalogów w pętli	727
15.2.4.2.	Listowanie plików i katalogów przy wsparciu klasy wxDirTraverser	727
15.2.4.3.	Listowanie plików i katalogów za pomocą metody GetAllFiles() ..	729
15.2.5.	Inne przydatne metody wxDir	729
15.3.	Klasy wxFile oraz wxFFile i podstawowa obsługa plików	730
15.3.1.	Niskopoziomowa obsługa plików, czyli klasa wxFile	730
15.3.1.1.	Tworzenie plików wxFile i zapis danych do pliku	730
15.3.1.2.	Odczyt danych z pliku	732
15.3.1.3.	Deskryptor pliku i pozostałe zagadnienia	733
15.3.2.	Wysokopoziomowa obsługa plików, czyli klasa wxFFile	734
15.3.2.1.	Tworzenie plików wxFFile i zapis danych do pliku	734
15.3.2.2.	Odczyt danych z pliku	735
15.3.2.3.	Pozostałe informacje o wxFFile	736
15.4.	Więcej narzędzi, czyli klasa wxFileName	736
15.4.1.	Tworzenie i inicjowanie obiektów wxFileName	736
15.4.2.	Analiza ścieżki, nazwy i rozszerzenia pliku	737
15.4.3.	Edycja ścieżki, nazwy i rozszerzenia pliku	740
15.4.4.	Ścieżki względne i absolutne	742
15.4.5.	Normalizacja ścieżek (skrótów, zmienne środowiskowe i inne)	743
15.4.6.	Odczyt i edycja czasów pliku	744
15.4.7.	Prawa dostępu do pliku lub katalogu	745
15.4.8.	Badanie i wyświetlanie rozmiaru pliku	746
15.4.9.	Tworzenie i obsługa plików tymczasowych z wxFileName	747
15.4.10.	Niektóre pozostałe testy obiektów wxFileName	749
15.4.11.	Proste operacje na systemie plików	749
15.5.	Wirtualne systemy plików wxWidgets	750
15.6.	Praca z archiwami	752
15.6.1.	wxArchiveFSHandler i odczyt danych z archiwów	753
15.6.1.1.	Listowanie zawartości wybranego archiwum	753
15.6.1.2.	Wyświetlanie zawartości pliku w archiwum	753
15.6.1.3.	Rozpakowanie wybranego archiwum do wskazanej lokalizacji ..	754
15.6.2.	Prezentacja zaawansowanej obsługi archiwów	755
15.6.2.1.	Tworzenie archiwum	756
15.6.2.2.	Rozpakowywanie archiwów	758
15.6.2.3.	Pozostałe operacje i możliwości pracy z archiwami	759
15.7.	Monitorowanie zmian w systemie plików	760
15.8.	Pliki tekstowe wxTextFile	762
15.8.1.	Analiza pliku	763

15.8.2.	Tworzenie i edycja pliku	763
15.8.3.	Odczyt danych pliku	764
15.9.	Pliki tymczasowe wxTempFile	765
15.10.	Ścieżki systemowe, czyli klasa wxStandardPaths	766
15.10.1.	Niektóre przydatne metody wxStandardPaths zależne od platformy	767
15.10.1.1.	Metody dedykowane MS Windows	767
15.10.1.2.	Metody dedykowane Linuksowi	768
15.11.	Podsumowanie	768
16.	Krótki przegląd niektórych zaawansowanych kontroltek GUI	769
16.1.	Prezentacja wxRichTextCtrl	769
16.1.1.	Tworzenie i wstępna konfiguracja obiektów wxRichTextCtrl	770
16.1.1.1.	Tworzenie obiektu edytora	770
16.1.1.2.	Definiowanie arkusza stylów	772
16.1.1.3.	Przygotowanie silnika drukowania i włączenie edytora	774
16.1.2.	Zarządzanie stylami	774
16.1.2.1.	Okno dialogowe wxRichTextStyleOrganiserDialog	774
16.1.2.2.	Kontrolki wyboru stylów	775
16.1.3.	Podstawowe metody edycji tekstu	776
16.1.3.1.	Podstawowe formatowanie tekstu	776
16.1.3.2.	Podstawowe metody formatujące	776
16.1.3.3.	Aktualizacja elementów GUI w zależności od formatowania w pozycji kursora	777
16.1.4.	Rozszerzone formatowanie tekstu	779
16.1.4.1.	Badanie zakresu zaznaczenia tekstu oraz położenia kursora	779
16.1.4.2.	Przykład implementacji efektu znakowego: indeks górny i dolny ..	779
16.1.4.3.	Przykład implementacji efektu akapitu – zmienna interlinia	781
16.1.5.	Szybka zmiana kroju czcionki (i nie tylko)	781
16.1.6.	Listy	782
16.1.6.1.	Lista numerowana	783
16.1.6.2.	Lista wypunktowana	784
16.1.6.3.	Niektóre operacje na listach	785
16.1.7.	Wstawianie nowych elementów do tekstu	786
16.1.7.1.	Wstawianie znaków specjalnych	786
16.1.7.2.	Wstawianie URL	787
16.1.7.3.	Wstawianie grafiki z pliku	787
16.1.7.4.	Wstawianie tabel	788
16.1.8.	Drukowanie z kontrolką wxRichTextCtrl	790
16.1.8.1.	Podstawowe metody obsługi drukowania	790
16.1.8.2.	Nagłówek i stopka	790
16.1.9.	Operacje plikowe	794
16.1.9.1.	Rozpoznanie formatu pliku	794
16.1.9.2.	Otwieranie dokumentów	794
16.1.9.3.	Zapisywanie dokumentów	795
16.1.10.	Wnioski	795
16.2.	Prezentacja wxRibbonBar	796
16.2.1.	Struktura paska wxRibbonBar	797
16.2.2.	Tworzenie i inicjowanie paska wxRibbonBar	797
16.2.2.1.	Metoda Realize()	798
16.2.3.	Dodawanie stron do paska wxRibbonBar	798
16.2.4.	Dodawanie paneli wxRibbonPanel do stron paska narzędziowego	798
16.2.5.	Klasa wxRibbonButtonBar, czyli pasek przycisków	799
16.2.5.1.	Rodzaje przycisków paska wxRibbonBar	799
16.2.5.2.	Dodawanie paska wxRibbonButtonBar do panelu	800
16.2.5.3.	Dodawanie przycisków do wxRibbonButtonBar i ich obsługa	800
16.2.5.4.	Skalowanie paneli i jego wpływ na wyświetlanie przycisków	802

16.2.6.	Pasek narzędzi wxRibbonToolBar	803
16.2.6.1.	Realizacja paska narzędzi wxRibbonToolBar	803
16.2.6.2.	Obsługa przycisków paska narzędzi wxRibbonToolBar	804
16.2.7.	Galeria danych wxRibbonGallery	804
16.2.7.1.	Tworzenie galerii	804
16.2.7.2.	Zapełnianie galerii danymi	805
16.2.7.3.	Obsługa galerii wxRibbonGallery	807
16.2.8.	Dowolne kontrolki GUI w pasku wxRibbonBar	808
16.2.9.	Obsługa przycisku pomocy	808
16.2.10.	Zmiana wyglądu paska wxRibbonBar	808
16.2.11.	Wnioski i jeszcze więcej opcji...	809
16.3.	Prezentacja wxPropertyGrid i wxPropertyGridManager	810
16.3.1.	Tworzenie menedżera właściwości	811
16.3.2.	Hierarchia siatki i rodzaje właściwości	812
16.3.3.	Wskaźnik i nazwa, czyli różne sposoby identyfikacji właściwości	813
16.3.4.	Realizacja przykładowego menedżera	814
16.3.5.	Pobieranie wartości poszczególnych pól siatki	818
16.3.6.	Obsługa zdarzeń i rozpoznawanie modyfikowanych właściwości	819
16.3.7.	Wnioski	820
16.4.	Inne kontrolki zaawansowane	820
16.4.1.	Klasa wxGrid	820
16.4.2.	Klasa wxWebView	821
16.4.3.	Klasa wxMediaCtrl	823
16.5.	Podsumowanie	824
17.	Tworzenie własnych generycznych kontroltek GUI	825
17.1.	Główne zasady tworzenia własnych kontroltek generycznych	825
17.1.1.	Program przykładowy	826
17.2.	Skalowany panel z obrazkowym tłem	827
17.2.1.	Piszemy klasę panelu	828
17.2.2.	Piszemy klasę zdarzenia panelu	829
17.2.3.	Metody podstawowe, czyli konstruktor, destruktor, Init() i Create()	830
17.2.4.	Metoda rysująca kontrolkę	831
17.2.5.	Metody inicjujące rysowanie	832
17.2.6.	Pozostałe metody	832
17.2.7.	Użycie skalowanego panelu graficznego w kodzie programu	833
17.3.	Implementacja przycisku graficznego	834
17.3.1.	Koncepcja i realizacja klasy przycisku graficznego	834
17.3.2.	Piszemy klasę zdarzeń przycisku	835
17.3.3.	Podstawowe funkcje klasy przycisku	836
17.3.4.	Metody związane z rysowaniem przycisku	838
17.3.5.	Metody obsługi i emisji zdarzeń	839
17.3.6.	Użycie przycisku graficznego w aplikacji	840
17.3.6.1.	Obsługa zdarzeń przycisku	841
17.3.7.	Propozycje udoskonaleń kontrolki	842
17.4.	Implementacja graficznego paska postępu	842
17.4.1.	Koncepcja kontrolki i opracowanie klasy	842
17.4.2.	Konstruktor, destruktor oraz funkcje tworzące i inicjujące pasek	844
17.4.3.	Funkcje inicjujące rysowanie kontrolki	845
17.4.4.	Funkcja rysująca pasek	846
17.4.5.	Stosowanie graficznego paska postępu w aplikacji	848
17.4.6.	Propozycje udoskonalenia paska graficznego	849
17.5.	Implementacja prostego kalendarza	849
17.5.1.	Koncepcja i klasa kalendarza, a także klasy pomocnicze	849
17.5.1.1.	Klasa imiennin	849
17.5.1.2.	Klasa dnia miesiąca	851
17.5.1.3.	Okno dialogowe pobierania daty	852
17.5.1.4.	Klasa kalendarza	852

17.5.2.	Klasa zdarzenia kalendarza	854
17.5.3.	Obliczanie danych miesiąca	855
17.5.4.	Obliczanie współrzędnych siatki kalendarza	856
17.5.5.	Metoda Init()	857
17.5.6.	Konstruktor, Create() oraz destruktory	858
17.5.6.1.	Metody związane z rysowaniem kalendarza	859
17.5.6.2.	Metody inicjujące rysowanie	859
17.5.6.3.	Metoda rysująca siatkę kalendarza	859
17.5.6.4.	Metoda rysująca dane poszczególnych dni w miesiącu	860
17.5.7.	Zmiana daty bieżącej	862
17.5.8.	Zdarzenia wewnętrzne i emitowane kalendarza	862
17.5.9.	Metody obsługi zdarzeń menu kontekstowego	864
17.5.10.	Stosowanie klasy MyCalendar w kodzie programu	865
17.5.11.	Propozycje udoskonalenia kalendarza	866
17.5.12.	Tworzenie kontrolki natywnych	866
17.6.	Podsumowanie	867
18.	Techniki i modele realizacji GUI aplikacji	868
18.1.	Sizery wxWidgets	868
18.1.1.	Klasa wxSizer i ogólne aspekty pracy z sizerami	869
18.1.1.1.	Określanie atrybutów elementów sizera	869
18.1.1.2.	Zarządzanie elementami w obrębie sizera	870
18.1.1.3.	Pobieranie wskaźników elementów przypisanych do sizera	872
18.1.1.4.	Wyświetlanie i ukrywanie elementów	872
18.1.1.5.	Pobieranie i dostosowanie rozmiaru oraz pozycji sizera i jego elementów	873
18.1.1.6.	Pozostałe wspólne metody sizerów, dziedziczone z wxSizer	874
18.1.2.	Sizery liniowe i ich stosowanie	874
18.1.2.1.	Klasa wxBoxSizer	875
18.1.2.2.	Klasa wxStaticBoxSizer	876
18.1.2.3.	Klasa wxStdDialogButtonSizer	876
18.1.2.4.	Klasa wxWrapSizer	877
18.1.3.	Sizery tablicowe i ich stosowanie	877
18.1.3.1.	Klasa wxGridSizer	877
18.1.3.2.	Klasa wxFlexGridSizer	879
18.1.3.3.	Klasa wxGridBagSizer	881
18.1.4.	Wnioski	882
18.2.	Multi Document Interface (MDI)	882
18.2.1.	Ogólne zasady tworzenia interfejsu MDI	883
18.2.2.	Automatyczne menu nawigacyjne i menu okien podrzędnych	884
18.2.3.	Poznajemy bliżej klasę wxMDIParentFrame	885
18.2.4.	MDI a model dokument/widok (wxDocument/wxView)	885
18.2.4.1.	Model dokument/widok w wxWidgets	886
18.2.4.2.	Klasy MDI dedykowane modelowi dokument/widok	886
18.2.4.3.	Program przykładowy i koncepcja formatu danych	886
18.2.4.4.	Klasa menedżera dokumentów	887
18.2.4.5.	Klasy dokumentu oraz danych	887
18.2.4.6.	Klasy widoku oraz płótna	889
18.2.4.7.	Wyświetlenie ramek MDI w modelu dokument/widok w programie	891
18.2.5.	Wnioski	893
18.3.	Tworzenie GUI aplikacji przy użyciu wxAUI	893
18.3.1.	Przedstawienie elementów wxAUI	894
18.3.1.1.	Menedżer ramek wxAuiManager	894
18.3.1.2.	Właściwości ramek dokowanych i pływających, czyli klasa wxAuiPaneInfo	896
18.3.1.3.	Kontener wxAuiNotebook	899
18.3.1.4.	Paski narzędzi wxAuiToolBar	901

18.3.2.	Warstwy, wiersze i pozycje, czyli zrozumieć wxAUI	902
18.3.3.	wxAUI w praktyce	903
18.3.3.1.	Menedżer	903
18.3.3.2.	Ramka 1 – edytor (kontener wxAuiNotebook)	903
18.3.3.3.	Ramka 2 – okno konsoli (bezpośrednie dodanie kontrolki)	904
18.3.3.4.	Ramki 3 i 4 – dodanie nowej warstwy GUI	904
18.3.3.5.	Ramka 5 – kontener z sizerem	905
18.3.3.6.	Paski narzędzi	906
18.3.3.7.	Modyfikowanie wyglądu elementów dokowanych	907
18.3.3.8.	To chyba jeszcze nie wszystko!	907
18.3.3.9.	Co z tego wyszło?	907
18.3.4.	Wnioski	908
18.4.	Krótko o pozostałych technikach tworzenia GUI	909
18.4.1.	Single Document Interface (SDI)	909
18.4.2.	wxXRC i zasoby XML	909
18.5.	Słowo o niektórych narzędziach WYSIWYG dedykowanych pracy z wxWidgets	911
18.5.1.	wxFormBuilder	911
18.5.2.	wxSmith	912
18.5.3.	Czy używać narzędzi WYSIWYG?	913
18.6.	Podsumowanie	913
19.	Drukowanie w wxWidgets od kuchni	914
19.1.	Ogólnie o klasie wxPrintout	914
19.1.1.	Tworzenie prostego silnika drukowania	915
19.1.2.	Kustomizacja własnego silnika drukowania	917
19.1.3.	Problemy skalowania wydruku	918
19.1.4.	Źródła danych i kilka silników drukowania w jednym programie	919
19.2.	Konfiguracja wydruku	920
19.2.1.	Klasa wxPrintData	920
19.2.2.	Klasa wxPrintDialogData	921
19.2.3.	Klasa wxPageSetupDialogData	921
19.3.	Klasy wxRichTextBuffer i wxRichTextPrinting oraz drukowanie formatowanego tekstu na skróty	922
19.4.	Podsumowanie	923
20.	Krótki rozdział o technikach przechowywania konfiguracji programu	924
20.1.	Konfiguracja za pomocą klas pochodnych z wxConfigBase	924
20.1.1.	Ogólna koncepcja konfiguracji wxConfigBase	924
20.1.2.	Klasa wxConfigBase	925
20.1.2.1.	Zarządzanie wpisami konfiguracji	925
20.1.2.2.	Odczyt i zapis danych konfiguracyjnych	925
20.1.2.3.	Pozostałe ważniejsze metody wxConfigBase	926
20.1.3.	Plik konfiguracyjny wxFileConfig	926
20.1.3.1.	Struktura pliku konfiguracyjnego	927
20.1.3.2.	Plik konfiguracyjny w praktyce	928
20.1.4.	Konfiguracja w rejestrze systemu, czyli wxRegConfig (MS Windows)	929
20.1.4.1.	Konfiguracja wxRegConfig w praktyce	929
20.1.5.	Konfiguracja domyślna a platforma systemowa	930
20.2.	Zapamiętywanie stanów GUI za pomocą wxPersistenceManager	931
20.3.	Podsumowanie i refleksja na temat innych możliwości zapamiętywania konfiguracji programu	931
21.	wxWidgets i programowanie współbieżne	933
21.1.	Aplikacja wielowątkowa i klasa wxThread	933
21.1.1.	Klasa wxThread	934
21.1.1.1.	Wybrane metody statyczne klasy wxThread	935
21.1.2.	Szkielet aplikacji wielowątkowej	935

21.1.3.	Tworzenie klasy wątku	937
21.2.	Poradnik dla hazardzistów, czyli zabezpieczanie aplikacji przed wyścigiem do danych	939
21.2.1.	Kontrola liczby wątków, czyli klasa wxSemaphore	939
21.2.1.1.	Semafor wxSemaphore w praktyce	940
21.2.2.	Blokowanie dostępu do zasobów, czyli co nieco o tzw. muteksach	941
21.2.2.1.	Korzystanie z wzajemnych wykluczeń w aplikacji	942
21.2.2.2.	Dostęp wątków do biblioteki GUI	942
21.2.3.	Sekcja krytyczna wxCriticalSection	943
21.3.	Wątki i ich zdarzenia	944
21.4.	Podsumowanie	946
22.	wxWidgets i konsola	947
22.1.	Konsolowe aplikacje wxWidgets?	947
22.1.1.	Dostosowanie projektu i makefile	947
22.1.1.1.	MS Windows i Visual Studio	947
22.1.1.2.	Linux i makefile	948
22.1.2.	Szkielet aplikacji konsolowej wxWidgets	948
22.2.	Obsługa parametrów aplikacji z linii komend	949
22.2.1.	Tworzenie list parametrów	949
22.2.1.1.	Sposób 1 – definiowanie parametrów za pomocą metod klasy wxCmdLineParser	950
22.2.1.2.	Sposób 2 – lista parametrów w postaci tablicy wxCmdLineEntryDesc	952
22.2.1.3.	Parsowanie i przetwarzanie parametrów	953
22.2.1.4.	Odczyt i wykorzystanie parametrów w aplikacji	953
22.2.1.5.	Własny tekst w wiadomości pomocy	954
22.3.	Zarządzanie procesami z poziomu aplikacji (wx/utils.h)	955
22.4.	Strumień wyjściowy C++ w kontrolce tekstowej	956
22.5.	Sprawdzanie liczby instancji programu	957
22.6.	Podsumowanie	958
23.	Co jest grane? O logach w wxWidgets	959
23.1.	Klasa wxLog i ogólne informacje o wyświetlaniu logów	959
23.1.1.	Ustawienia ogólnego systemu logowania	959
23.1.2.	Poziome logowanie i odpowiadające im funkcje piszące	960
23.1.3.	Wybór komponentów logujących	961
23.2.	Wyjście logu i logowanie w praktyce	962
23.2.1.	Charakterystyka klas wyjścia logu	962
23.2.2.	Logowanie w praktyce	962
23.2.2.1.	Tworzenie obiektów klas wyjścia logu	963
23.2.2.2.	Zmiana wyjścia logu	963
23.2.2.3.	Sterowanie pracą wyjścia logu	964
23.3.	Własny format logu	964
23.4.	Podsumowanie	965
24.	Nie tylko po polsku, czyli internacjonalizacja aplikacji	966
24.1.	Makro _()	966
24.2.	Tłumaczenia aplikacji	966
24.2.1.	Warsztat tłumacza, czyli oprogramowanie poEdit	967
24.2.2.	Przygotowanie tłumaczeń programu z narzędziem poEdit	967
24.2.3.	Samodzielne przygotowanie plików tłumaczeń	969
24.3.	Implementacja kilku języków w aplikacji	970
24.3.1.	Klasa wxLocale i elementy internacjonalizacji	971
24.3.1.1.	Konstruowanie i inicjowanie obiektów wxLocale	971
24.3.1.2.	Lokalizacja plików tłumaczeń	971
24.3.1.3.	Analizowanie danych wczytanego języka	972
24.3.1.4.	Ważniejsze statyczne metody klasy wxLocale	973

24.3.2.	Internacjonalizacja w praktyce, czyli prosty przykład wykorzystania wxLocale	973
24.4.	Regionalizacja aplikacji	975
24.5.	Podsumowanie	976
25.	Programowanie sieciowe z wxWidgets	977
25.1.	Klasa wxSocketBase i komunikacja serwer/klient	977
25.1.1.	Klasa wxSocketBase	977
25.1.1.1.	Ważniejsze metody klasy wxSocketBase	978
25.1.1.2.	Zdarzenia wxSocketBase i ich obsługa	981
25.1.2.	Piszemy prosty serwer, czyli korzystanie z wxSocketServer	982
25.1.2.1.	Przygotowanie niektórych elementów aplikacji	982
25.1.2.2.	Tworzenie serwera	983
25.1.2.3.	Klasa wątku połączenia	984
25.1.2.4.	Tworzenie nowych wątków połączeń	987
25.1.3.	Piszemy prostego klienta, czyli korzystanie z wxSocketClient	988
25.1.3.1.	Przygotowanie i koncepcja aplikacji	988
25.1.3.2.	Tworzenie i usuwanie obiektu klienta	988
25.1.3.3.	Inicjowanie połączenia z serwerem	989
25.1.3.4.	Nawiązanie połączenia i obsługa zdarzeń gniazda	990
25.1.3.5.	Wysyłanie danych na serwer	991
25.1.3.6.	Rozłączanie klienta	991
25.1.4.	Testowanie komunikacji klient/serwer za pomocą wykonanych aplikacji	992
25.1.5.	Programowanie z protokołem UDP	992
25.2.	Klasa wxProtocol i obsługa podstawowych protokołów sieciowych	992
25.2.1.	Klasa protokołu wxProtocol	993
25.2.2.	Obsługa protokołu HTTP (wxHTTP)	993
25.2.2.1.	Ważniejsze metody klasy wxHTTP	994
25.2.2.2.	Prosty przykład – pobieranie źródła strony do bufora lokalnego	994
25.2.3.	Obsługa protokołu FTP w praktyce, czyli piszemy własnego klienta (wxFTP)	995
25.2.3.1.	Przygotowanie aplikacji	995
25.2.3.2.	Dodajemy obiekt wxFTP	997
25.2.3.3.	Piszemy funkcje narzędziowe	998
25.2.3.4.	Łączenie z serwerem FTP	999
25.2.3.5.	Zamykanie połączenia	1001
25.2.3.6.	Zmiana aktywnego katalogu FTP	1001
25.2.3.7.	Niektóre operacje na zdalnym systemie plików	1002
25.2.3.8.	Klasa wątku pobierania i pobieranie pliku z FTP	1003
25.2.3.9.	Klasa wątku wysyłania i wysyłanie plików na FTP	1005
25.2.3.10.	Testowanie klienta FTP	1007
25.3.	Podsumowanie	1007
26.	wxWidgets i nowoczesny OpenGL	1008
26.1.	Klasa wxGLCanvas	1009
26.2.	Klasa wxGLContext	1011
26.3.	Inicjowanie obsługi nowoczesnego OpenGL w aplikacji wxWidgets	1012
26.4.	Przykładowy program i renderowanie sceny	1013
26.5.	Podsumowanie	1015
27.	Zagadnienia programowania bazodanowego z wxWidgets 3+	1016
27.1.	Przykładowy warsztat	1016
27.1.1.	Biblioteka SOCI	1016
27.1.1.1.	Konfiguracja projektu SOCI w CMake	1017
27.1.2.	Serwer Firebird	1017
27.1.3.	Narzędzie FlameRobin	1017
27.2.	Szybkie wprowadzenie do relacyjnych baz danych i SQL	1018
27.2.1.	Relacyjne bazy danych	1018

27.2.2.	Podstawy języka SQL	1019
27.2.2.1.	Grupa instrukcji Data Definition Language (DDL)	1019
27.2.2.2.	Grupa instrukcji Data Manipulation Language (DML)	1020
27.2.2.3.	Data Query Language (DQL) i instrukcja SELECT	1021
27.2.2.4.	Grupa instrukcji Data Control Language (DCL)	1022
27.2.2.5.	Dla ambitnych, czyli automatyczne nadawanie identyfikatorów ...	1022
27.3.	Prosty przykład, czyli bazy danych i wxWidgets 3+ w praktyce	1023
27.3.1.	Przygotowanie projektu	1023
27.3.2.	Tworzenie nowej bazy danych za pomocą narzędzia FlameRobin	1024
27.3.3.	Zasady kreowania podstawowych zapytań do bazy danych za pomocą SOCI .	1026
27.3.4.	Przykładowy program	1027
27.3.4.1.	Ogólna koncepcja oraz elementy GUI aplikacji ‘Sklep’	1027
27.3.4.2.	Otwieranie i zamykanie sesji z bazą danych	1028
27.3.4.3.	Pobieranie i wyświetlanie informacji statystycznych i list zamówień	1030
27.3.4.4.	Dodawanie towarów do sklepu	1032
27.3.4.5.	Dodawanie klientów sklepu	1034
27.3.4.6.	Tworzenie i zamykanie zamówień	1036
27.3.4.7.	Wysyłanie własnych zapytań do bazy danych	1040
27.4.	Podsumowanie	1040
28.	Bonus na koniec, czyli jak wykonać ekran startowy	1042
28.1.	Ekran startowy wxSplashScreen	1042
28.2.	Wzbogacony ekran startowy	1043
28.3.	To wciąż za mało!	1046
Zakończenie		1047
Dodatek A Krótki leksykon standardowych elementów GUI wxWidgets		1049
Kontrolki statyczne		1049
Kontrolki dynamiczne		1053
Kontenery GUI		1080
Dodatek B Niektóre ważniejsze rozszerzenia wxWidgets		1087
wxPdfDocument		1087
wxMIDI		1087
wxChart		1088
wxThumbnail		1089
Dodatek C Licencje		1090
wxWindows Library Licence, Version 3.1		1090
GNU Library General Public License		1091
Indeks		1097

Od autora

*... dobrze jest koło sterowe mocniej ująć,
największej bowiem trzeba wytrwałości
sternicznej i nie w mapę, lecz w trzewia
słoneczne patrzeć się godzi: nikt bowiem
nie przebył drogi tej tak samo dwa razy.*

Stanisław Lem, *Cyberiada*

Pierwszy program komputerowy napisałem dość dawno temu. Była to bardzo prosta aplikacja graficzna wyświetlająca prostopadłością i kilka dziwacznych figur na ekranie starego lampowego telewizora marki *Elektron*, do którego był podłączony mój pierwszy mikrokomputer – Sinclair ZX Spectrum. Do dziś pamiętam swoje zaskoczenie, gdy ów program zadziałał, szczeciacką satysfakcją i lzy ojca skupionego nad podręcznikiem BASIC'a, którego co rusz wypytywałem o sens i znaczenie różnych egzotycznie wyglądających komend. Była połowa lat osiemdziesiątych zeszłego wieku. Wszelka angielszczyzna kojarzyła się wówczas z „Kinem nocnym” i rodzinnymi serialami, którymi w niedzielne przedpołudnia zachwycał jeden z dwóch dostępnych wówczas kanałów telewizyjnych. Nikt nie przypuszczał, że techniki informatyczne tak się rozwiną – mimo że *Bajtek* wciąż donosił o coraz to bardziej fascynujących odkryciach. Ja też jeszcze nie wiedziałem, że to, czym wtedy żyłem, to początek wieloletniej pasji i wspaniałej przygody z programowaniem.

Prawdziwą rewolucją było dla mnie odkrywanie potęgi języka C++ oraz bibliotek, z którymi i z których mogłem wyczarować to, co chciałem. Z punktu widzenia programowania graficznego interfejsu użytkownika wspomaganego różnymi mechanizmami, jaki powinien mieć nowoczesny program komputerowy, najpotężniejszym z nich zawsze była dla mnie wxWidgets. Pracuję z nią od 2001 r. i po latach programowania stwierdzam, że nawet dzisiaj nie pomyślałbym, aby zwrócić się ku innej bibliotece służącej do wieloplatformowego programowania GUI. Teraz chcę podzielić się z Tobą moją wiedzą, doświadczeniem i pasją.

Na początek chcę Ci wyjaśnić, że to nie jest w pełni formalna, fachowa książka na temat wxWidgets. Takie publikacje już istnieją i temat w pewien sposób wyczerpują. Wszystkie one jednak traktują poszczególne elementy biblioteki w sposób niemal encyklopedyczny i niewiele można się z nich dowiedzieć o tym, jak w praktyce programować z wxWidgets. Ponadto najczęściej dotyczą wersji biblioteki, których świetność dawno już minęła i straciła swą aktualność. Czytelnik tych dzieł jest zmuszony do samodzielnej analizy, poszerzania i aktualizowania zgłębianego materiału, co często skutkuje pominięciem niektórych, bardzo istotnych zagadnień. Dlatego, jako prawdziwy fascynat wxWidgets, postanowiłem wypełnić tę lukę – stąd zrodziła się książka, którą właśnie trzymasz w rękach.

Jest ona rodzajem podręcznika, poradnika, przy lekturze którego – mam nadzieję – nabierzesz sporo nowych umiejętności i przekonasz się, jak łatwe i przyjemne jest programowanie z biblioteką wxWidgets, a także z innymi bibliotekami umożliwiającymi tworzenie profesjonalnych programów komputerowych dla systemów operacyjnych z rodziny MS Windows i Linux. Jeśli jesteś doświadczonym programistą, znajdziesz tu garść usystematyzowanej praktycznej i teoretycznej wiedzy, pogrupowanej według problematyki (choć systematyzowanie i swoista taksonomia wiadomości nie jest najważniejszym celem tej książki), do których poznania lub odświeżenia Cię zachęcam. Niemniej jednak ostrzegam Cię, że książka zawiera również mnóstwo opisów takich elementów programowania, które w oczach doświadczonego programisty mogą wydawać się elementarne. Nie chcę przy tym, abyś wydawał swoje pieniądze na wiedzę, którą już posiadasz, albo co do której masz inne poglądy. Chcę po prostu, aby książka była dostępna dla jak najszerszej grupy odbiorców, także tych, którzy w świecie wieloplatformowego programowania GUI stawiają pierwsze kroki.

Dlaczego akurat wxWidgets? Zdań na temat przydatności i możliwości tej biblioteki jest tyle, ilu jest programistów. Niewątpliwą i pierwszą ze wszystkich zaletą wxWidgets jest jej wieloplatformowość, za którą zmierza także niezwykła łatwość jej użycia. W rozpędzonym świecie coraz bardziej zróżnicowanych usług informatycznych, wykorzystujących różne platformy systemowe, są to cechy kluczowe. Poza tym biblioteka jest nieustannie rozwijana i wzbogacana o atrakcyjne i nowoczesne rozwiązania, które z pewnością spełnią oczekiwania nawet najbardziej wymagającego programisty. W końcu wxWidgets jest też biblioteką opartą na idei *open source*, a dzięki dość liberalnej licencji umożliwia dowolne jej wykorzystanie także w projektach komercyjnych (zob. *Dodatek C*).

Książka, którą właśnie trzymasz w rękach, absolutnie nie wyczerpuje tematu, o którym ma ambicję traktować. Wymagałoby to stworzenia kilkutomowego dzieła, przy którym sienkiewiczowska *Trylogia* byłaby zaledwie broszurą. W związku z tym zmuszony jestem pokazać Ci tę część fascynującego świata wxWidgets, która pozwoli Ci pewnie i zdecydowanie poczuć się w jej meandrach, jednak poszukiwaniem niektórych szczegółów będziesz musiał zająć się sam (będę się starał zostawiać drogowskazy wszędzie tam, gdzie uznam to za konieczne). Twoją uwagę zechcę skierować na zagadnienia, które Ci pomogą poznać wxWidgets na tyle, abyś potrafił tworzyć za jej pomocą złożone aplikacje oraz wiedział, co zrobić dalej, gdy poczujesz potrzebę samodzielnego zgłębienia jej tajemnic.

Moim celem jest ukazanie Ci możliwości projektowania i tworzenia profesjonalnych, otwartych i komercyjnych programów komputerowych, wyłącznie przy użyciu ogólnie dostępnych i bezpłatnych narzędzi (najczęściej *otwartych*), a także pokazanie tego, w jaki sposób jedna wieloplatformowa biblioteka umożliwia tworzenie kodu współpracującego z różnymi systemami operacyjnymi i różnymi kompilatorami.

Jest nim też popularyzacja biblioteki wxWidgets wśród polskich programistów, mam bowiem wrażenie, że w polskich realiach została ona nieco zmarginalizowana, ale nie przez to, że ustępuje innym bibliotekom tego typu (bo takie stwierdzenie jest pochopne i zdecydowanie mija się z prawdą), a przez dotkliwy dla wszystkich brak jakiegokolwiek literatury w języku polskim, która byłaby jej poświęcona i traktowałaby temat na tyle wyczerpująco, aby po jej lekturze dysponować wachlarzem silnych argumentów w zestawianiu jej z innymi bibliotekami. Samodzielna analiza źródeł w języku angielskim może być zniechęcająca, a brak szybkich i widocznych efektów pracy z nowymi zagadnieniami może często prowadzić do zbyt wczesnego odstawienia narzędzi, które w rzeczywistości oferują potężne i ciekawe rozwiązania. Ta książka ma temu zapobiec.

Zapraszam Cię w podróż, z której nie wrócisz już taki sam...

Podziękowania

Nie pisałbym dzisiaj tych słów, gdybym nie spotkał kilku osób, którym zawdzięczam to, że moja fascynacja wxWidgets ma dziś taki właśnie kształt. Dziękuję mojemu ojcu, który sam będąc żadnym wiedzy o rozwijającym się świecie, w odpowiednim momencie spowodował, że w naszym domu pojawił się znienawidzony przez żeńską część rodziny (rozkochaną w serialach na *dwójce*) mikrokomputer, który stanowił iskrę zapalną dla wspaniałej pasji. Dziękuję też Piotrowi Cymbalukowi za to, że umożliwił mi wczesne wejście do krainy C++ i poznanie tego potężnego języka, a także motywował mnie do badań nad różnymi bibliotekami służącymi do kreowania wydajnych i efektywnych GUI aplikacji komputerowych. Dziękuję Marcinowi Łopacińskiemu, który służył mi niestrudzoną pomocą zawsze wtedy, gdy moja znajomość angielskiego okazywała się niewystarczająca do zgłębienia niektórych technicznych i zaawansowanych meandrów programowania, i z którym odbyłem wiele godzin fascynujących i kształcących programistycznych dyskusji. Szczególne podziękowania kieruję do Pani Edyty Kawali z Wydawnictwa Naukowego PWN, która doprowadziła do zmaterializowania niniejszej książki, a także do Pani Ireny Puchalskiej za wzbudzenie semantycznych wątpliwości, a także za to, że redagując kolejne rozdziały książki, niejednokrotnie dokonała cudu. Dziękuję również recenzentom, którzy dostarczyli mi cennych uwag, które niejednokrotnie miały wpływ na ostateczny kształt treści i przykładów zawartych na jej stronach. Przede wszystkim jednak dziękuję Elżbiecie, która dzielnie znosząc moje – chwilami zbyt wielkie – zaangażowanie w pisanie książki, nieustannie motywowała mnie do przelewania każdego słowa „na papier”.

Ta książka nigdy nie powstałaby bez Was i Wam ją dedykuję.

Wprowadzenie

Zakres tematyczny książki

Biblioteka wxWidgets to temat, o którym można mówić bez końca. Niemniej jednak chęć zwięzłego i przejrzystego opowiadania o jej możliwościach zmusza do pewnych kompromisów. W związku z tym wywody, jakie znajdziesz w tej książce, są ograniczone do rozważań nad programowaniem dla platform MS Windows oraz Linux (np. Ubuntu oraz Mint¹). Ponadto skupiłem się na tych elementach biblioteki, które prawdopodobnie bardzo często będziesz wykorzystywał w niemal wszystkich swoich programach, a także takich, których zrozumienie pomoże Ci na efektywne samodzielne korzystanie z biblioteki w przyszłości, czy też rozwijanie własnych, opartych na niej, komponentów programistycznych. Przykładowe programy oraz kody, jakie wspólnie napiszemy i stworzymy, nie tylko nauczą Cię praktycznego stosowania wxWidgets, ale przekonają Cię również o tym, że wxWidgets może z powodzeniem współistnieć i współpracować z innymi komponentami i bibliotekami w kodzie programu, a samo programowanie z nią jest łatwe, przyjemne i pozwala także na tworzenie czytelnego kodu dobrej jakości.

Ponadto celem książki nie jest wchodzenie w analityczne powtórzenia zagadnień, którymi zajęli się już autorzy innych opracowań traktujących o wxWidgets², ani takie, które bezpośrednio wynikają z dokumentacji bibliotek przedstawionych w książce, dlatego możesz przygotować się na odrobinę gawędziarstwa o elementach wxWidgets, ale przez pryzmat empiryzmu i ich przydatności w kontekście omawianych tematów.

Ważna uwaga o konstrukcji książki, czyli zanim kupisz

Podjęcie decyzji o konstrukcji książki opowiadającej o wxWidgets zajęło mi najwięcej czasu, jaki poświęciłem na jej przygotowywanie. Trudność tkwiła nie w doborze zakresu merytorycznego, z którym chcę Cię zapoznać, ale w identyfikacji najbardziej efektywnej oraz przystępnej formy i strategii poznawczej i dydaktycznej. Ostatecznie zdecydowałem się podjąć ryzyko i odejść od encyklopedycznej czy leksykonowej konwencji, jaką stosuje większość autorów, i wpleść poszczególne elementy wxWidgets w pewien ciąg przyczynowo-skutkowy, którego istotą jest wprowadzanie kolejnych zagadnień w miarę pisania i komplikowania kolejnych programów przykładowych. To właśnie praktyką i pisaniem programów będziemy zajmować się przede wszystkim na stronach niniejszej książki (stąd

¹ Wszelkie przykłady i metody zawarte w książce bez problemu działają również w dystrybucji Debian. Nie ukrywam, że zdecydowałem się na wykorzystanie dystrybucji Ubuntu oraz Mint ze względu na ich rosnącą popularność wśród zatwardziałych przeciwników linuksowego pingwina. Powinieneś również pamiętać, że próby uruchomienia przykładów w innych dystrybucjach Linuksa mogą zakończyć się niepowodzeniem, jednak zwykle będzie można to naprawić dzięki kilku prostym zabiegom.

² Wiele przydatnych informacji na temat podstaw programowania z biblioteką wxWidgets możesz znaleźć w książce Janusza Ganczarskiego i Mariusza Owczarka pod tytułem C++. *Wykorzystaj potęgę aplikacji graficznych*, wydanej przez wydawnictwo Helion w 2008 r. Autorzy omówili wiele ciekawych zagadnień, wśród których na szczególną uwagę zasługuje przystępne przedstawienie zdarzeń, przybliżenie wykorzystania standardowych okien dialogowych, przystępne i zwięzłe omówienie podstaw aplikacji sieciowych, obsługi grafiki czy elementów aplikacji wielowątkowych.

też jej „praktyczny” tytuł – *Programowanie...*). Wobec powyższego pozycja ta nie jest typową informacyjną monografią. Jest też zdecydowanie jedną z tych książek, które najlepiej jest przeczytać *od deski do deski*, mimo że poszczególne zagadnienia są łatwe do odszukania dzięki zawartemu na końcu książki indeksowi. Będziemy iść stopniowo od szczegółowej analizy wszystkich pojawiających się zagadnień w początkowych rozdziałach książki, ku konkretyzacji rozważań i skupieniu się wyłącznie na omawianych tematach w rozdziałach ostatnich. Cała książka stanowi jeden ciąg, jeśli zatem przeskoczysz dalej i coś wyda Ci się niezrozumiałe, jest wielce prawdopodobne, że odpowiedź na nurtujące Cię pytania znajdziesz w rozdziałach poprzednich.

Jeśli zdecydujesz się podjąć taką przygodę, jestem przekonany, że czytając ostatnie strony, będziesz umiał programować z wxWidgets. Jest to książka z rodzaju tych, z którymi trzeba po prostu się zaprzyjaźnić, i wierzę, że tak właśnie będzie w Twoim przypadku.

Aby zmierzyć się z niełatwą misją przekazania Ci moich doświadczeń i wiedzy, postanowiłem podzielić książkę na kilka części.

W pierwszej z nich przedstawię Ci bibliotekę wxWidgets 3, a także podpowiem, jak ją skompilować i jak przygotować swoje środowisko pracy do tworzenia nowych projektów. Pokażę Ci również, jak łatwa do opanowania i intuicyjna jest ogólna składnia wxWidgets, a także jak czytelna i przejrzysta jest „mechanika” aplikacji tworzonej z wxWidgets. Zobaczysz też, w jaki sposób zastosować w praktyce podstawowe mechanizmy wxWidgets, w tym – niepotrzebnie budzącą lęk (zwłaszcza dzięki początkującym programistom) obsługę zdarzeń.

Druga część książki pomoże Ci wejść łagodnie w niezwykle świat kontrolki GUI, a także różnych przydatnych klas oraz mechanizmów wxWidgets, z którymi będziesz mógł tworzyć swoje pierwsze aplikacje. Napiszemy w niej razem kilka programów, które pomogą Ci zrozumieć, w jaki sposób działają i współpracują ze sobą różne elementy biblioteki. W tej części książki nie omówiłem szczegółowo wszystkich dostępnych elementów wxWidgets, jednak jej lektura spowoduje, że będziesz czuć się swobodnie, stosując składnię stosowaną w bibliotece, a także będziesz dobrze się orientować w stylu programowania i możliwościach wxWidgets. Druga część jest również w pewien sposób wyjątkowa, gdyż wprowadza kolejne elementy wxWidgets w miarę tworzenia programów przykładowych, których pisaniem zajmiemy się przede wszystkim. Oznacza to, że kolejne tematy pojawiają się w niej w porządku wyznaczonym przez konkretne programistyczne potrzeby, a nie według jakiejś z góry przyjętej, swoistej „taksonomii” i systematyki.

Trzecia część książki wprowadza pewne uszczegółowienie i traktuje o bardzo ważnym zagadnieniu, jakim jest wykorzystanie standardowych okien dialogowych i tworzenie własnych. Okna dialogowe są bowiem podstawą efektywnej interakcji między programem komputerowym a jego użytkownikiem. Nie muszę Cię chyba przekonywać, że jest to lektura obowiązkowa. Ponadto znajdziesz w niej całą masę informacji o technikach kontroli danych wejściowych, jakie za pomocą okien dialogowych trafiają do programu.

W czwartej części książki wejdziemy w zaawansowaną analizę niektórych zagadnień wxWidgets. Między innymi omówimy najważniejsze klasy wxWidgets z punktu widzenia przetwarzania danych. Ponadto zdradzę Ci kilka przydatnych rozwiązań, których nie znajdziesz w dokumentacji biblioteki wxWidgets, a które zrodziły się podczas mojego wieloletniego obcowania z jej subtelnymi zawiłościami. Wierzę, że mogą one wzbogacić Twój warsztat i sprawić, że Twoje programy będą bardziej atrakcyjne i różnorodne. Zmierzymy się także z meandrami konwersji typów wxWidgets, a także z zasadami samodzielnego konstruowania i programowania interfejsu użytkownika. Zdradzę Ci tajniki projektowania i tworzenia własnych kontrolki, a także pokażę Ci wiele innych zaawansowanych technik, dzięki którym będziesz mógł w pełni wykorzystać możliwości wxWidgets i realizować swoje programistyczne pomysły. Między wierszami przemyciłem także trochę informacji o narzędziach, które być może ułatwią Ci pracę.

Na ostatnich stronach książki znajdziesz kilka przydatnych dodatków. Dodatek A zawiera krótki leksykon większości standardowych elementów graficznego interfejsu użytkownika, z jakich możesz korzystać, programując z wxWidgets. Oprócz ogólnych informacji o każdej z kontrolki GUI zawiera także przykładowe kody oraz informacje o najważniejszych metodach poszczególnych klas biblioteki. Dodatek B został poświęcony zwięzłemu przedstawieniu najważniejszych (w mojej subiektywnej

ocenie) rozszerzeń biblioteki, jakie są tworzone przez jej sympatyków i jakie znacznie podnoszą jej wszechstronność. Dodatek C zawiera teksty kluczowych licencji związanych z wxWidgets.

Gdybyś potrzebował dotrzeć do konkretnego tematu, możesz zawsze skorzystać z indeksu, który znajduje się na końcu książki.

Dla kogo jest ta książka?

Odpowiedź na to pytanie jest prosta – dla wszystkich, którzy chcą zapoznać się z wxWidgets lub chcą poszerzyć posiadaną już na jej temat wiedzę. Ponadto książka jest dla tych, których intrygują potężne możliwości wykorzystania darmowych narzędzi do tworzenia nowoczesnych aplikacji komputerowych, bez specjalnych ograniczeń co do platformy systemowej.

Niemniej jednak, mimo pełnej otwartości, zakres tematyczny książki narzuca pewne wymagania wobec jej czytelników. Abyś mógł w pełni wykorzystać treści zawarte w książce, powinieneś zatem:

- znać język programowania C++ (w tym umieć praktycznie stosować elementy biblioteki standardowej STL oraz znać zagadnienia programowania zorientowanego obiektowo, a także orientować się w nowościach wprowadzonych wraz z tzw. 11 wersją języka);
- umieć praktycznie stosować środowiska programistyczne Visual C++ dla systemów z rodziny MS Windows i/lub Code::Blocks dla systemów Linux (lub inne, lecz na stronach niniejszej książki skupimy się jedynie na pracy z wymienionymi narzędziami);
- znać język angielski w stopniu umożliwiającym lekturę dokumentacji bibliotek (biblioteki omówione w książce nie mają zbyt wielu opracowań lub nie mają ich w ogóle, jednak wszystkie mają bardzo dobrze opracowaną dokumentację w języku angielskim) lub posiadać umiejętność szybkiego korzystania z tłumaczy internetowych;
- mieć chęć i cierpliwość do systematycznego zgłębiania nowej wiedzy, jaką jest biblioteka wxWidgets, oraz stosowania jej w praktyce.

Aby tworzyć wspaniałe aplikacje okienkowe z użyciem wxWidgets, nie musisz mieć żadnego doświadczenia w kreowaniu tego typu programów z innymi bibliotekami C++ (Windows API, NET. Framework, GTK+, Qt i tak dalej). Swoją przygodę z wxWidgets możesz zacząć „od zera”.

Konwencje przyjęte w książce

Jeżeli jesteś już przyzwyczajony do literatury informatycznej, nie znajdziesz w tej książce żadnych innowacji dotyczących sposobu prezentowania treści.

Fragmenty kodu i wszelkie listingi są złożone czcionką stałej szerokości. Mimo potrzeby częstego, nienaturalnego łamania wierszy, taki sposób zapisu wydaje mi się bardziej dostępny i intuicyjnie łatwiej rozpoznawalny przez programistów. Nie mogę oprzeć się wrażeniu, że samo spojrzenie na tekst pisany czcionką stałej szerokości przełącza umysł programisty na odpowiedni tryb pracy i analizy, co ułatwia przyswajanie i syntezę elementów prezentowanego kodu. Nawet ojciec języka C++, Bjarne Stroustrup, mimo swoich pierwotnych przekonań o wyższości zapisu kodu za pomocą „zwykłych” krojów czcionek, w końcu przekonał się do monotypowej prezentacji kodu w swoich dziełach³.

Odrębnym zagadnieniem jest konwencja i stylistyka kodu przyjęta na potrzeby programowania z wxWidgets. Poświęcam temu tematowi trochę uwagi w dalszej części książki (zobacz podrozdz. 1.6).

Ponadto wszędzie tam, gdzie moim zamiarem jest przeprowadzenie szczegółowej analizy elementów programu, stosuję numerację wierszy kodu.

Jako że osobiście brzydzę się wszelkimi przejawami notacji węgierskiej (przepraszam wszystkich, którzy są do niej przyzwyczajeni), nie znajdziesz w tej książce nic, co mogłoby mieć z nią cokolwiek wspólnego, chyba że będzie to konieczne wraz z cytowaniem fragmentów kodu biblioteki.

³ Porównaj Bjarne Stroustrup, *Język C++*, wyd. VI, WNT, Warszawa 2002, oraz Bjarne Stroustrup, *Język C++. Kompendium wiedzy*, wyd. IV, Helion, Gliwice 2013.

Wszystkie zagadnienia, którym chciałem nadać większą wagę, w tym również ważne fragmenty kodu, zostały wyróżnione czcionką pogrubioną. Niektóre nazwy, cytaty, słowa kluczowe i inne elementy, w zależności od kontekstu, oznaczyłem kursywą.

Wszystkie nazwy klas, a także schematy i przytoczone deklaracje funkcji, które pojawiają się po raz pierwszy w tekście, wyróżniłem pogrubioną kursywą. W przypadku funkcji typ zwracanej wartości jest jedynie pogrubiony, przy czym zarówno listy parametrów funkcji, jak i informacja o typie zwracanym mogą być całkiem pominięte, zwłaszcza jeśli nie mają znaczenia dla aktualnego wyводу (możesz zawsze je sprawdzić w dokumentacji biblioteki). Jeśli nazwa funkcji jest przywoływana po raz kolejny, używam wobec niej jedynie kursywy, nieraz rezygnując z informacji o typie zwracanych oraz jej liście parametrów.

Choć tym samym stawiam wyzwanie wielu wyznawcom formalnych i nieformalnych konwencji nazewnictwa w obrębie programowania w C++, z troską o stylistykę i jasność wypowiedzi w odniesieniu do funkcji globalnych używam terminu *funkcja globalna* lub po prostu *funkcja*, w odniesieniu zaś do funkcji składowych klas – zamiennie terminów *funkcja*, *funkcja składowa* oraz *metoda*, starając się jednak do maksimum ograniczyć ryzyko semantycznych niejasności. Ponadto, gdy długie listy argumentów zacierają czytelność przykładów i schematów kodu skoncentrowanych na konkretnych zagadnieniach, miejscami stosuję wielokropki w obrębie list argumentów funkcji, co jest po prostu symbolicznym skrótem i nie należy tego mylić ze zmienną listą argumentów.

Wszelkie odniesienia do rozdziałów lub ich części, jakie znajdują się w tekście, opierają się na ich numeracji, zgodnej ze spisem treści – na przykład „zobacz 1.2.2.1”.

Odniesienia do dzieł innych autorów przywołuję głównie w przypisach lub bezpośrednio w treści książki, z uwagi zaś na ogrom zagadnienia, jakim jest programowanie w języku C++, zrezygnowałem z przedstawienia zbiorczej bibliografii przedmiotu, gdyż zakres informacji na ten temat, jakie zawiera niniejsze opracowanie, nie odbiega od zakresu przedstawionego w ogólnodostępnej literaturze dotyczącej C++, która od lat, w przyzwoitej ilości i różnorodności, dostępna jest dla polskiego programisty.

Na koniec dodam, że w trakcie lektury książki możesz spotkać różne ramki, którym nadałem takie oto znaczenie:

Przydatne informacje

W takiej ramce znajdziesz ciekawostki, uwagi oraz wszystkie inne rzeczy, na które warto spojrzeć i które mogą być lub są istotne dla omawianego właśnie tematu.



Uwagi o programowaniu w MS Windows

W takiej ramce będę chciał zwrócić Twoją uwagę na te kwestie programowania z wxWidgets, które są specyficzne dla systemów operacyjnych z rodziny MS Windows.



Uwagi o programowaniu w Linux

W ramce oznakowanej ikoną pingwinka GNU/Linux zamieściłem informacje, które są związane ze specyfiką programowania w systemach Linux.



Zadanie, problem do przemyślenia

Taka ramka zawiera zadania lub problemy do samodzielnego przemyślenia i rozwiązania.



Pomysł

W ramce tego typu zechcę zasugerować Ci pomysły, jakie wydają mi się ciekawe i dotyczą omawianych zagadnień.

Warsztat

Aby zrozumieć i samodzielnie wypróbować wszystko to, z czym zapoznasz się w trakcie lektury, a także byśmy mogli razem stworzyć przykładowe programy, będziesz potrzebować odpowiednio przygotowanego warsztatu. W zależności od systemu operacyjnego, którego używasz, będą to specyficzne dla tych systemów narzędzia. Wykorzystamy również programy dostępne dla obu omawianych platform systemowych.

MS Windows

Microsoft Visual C++

Wśród różnych środowisk programistycznych C++ (IDE) dostępnych dla platformy systemowej MS Windows moim zdecydowanym faworytem jest Visual C++. Oczywiście nie mam zamiaru przekonywać Cię do zmiany Twojego ulubionego IDE, lecz musisz pamiętać, że wszystkie przykłady zawarte w książce zostały przygotowane tak, aby bez problemu można je było kompilować z użyciem środowiska Microsoft Visual C++ pochodzącego z pakietu Microsoft Visual Studio Community lub Microsoft Visual Studio Express for Windows Desktop w wersji 2015, choć nie powinieneś mieć najmniejszych problemów z uruchomieniem ich na nowszych wersjach środowiska. To, z której wersji środowiska będziesz korzystać, jest zatem tylko kwestią Twoich upodobań i nie narzucam tu żadnych ograniczeń.

Visual Studio Community oraz Visual Studio Express for Windows Desktop są darmowymi wersjami środowiska umożliwiającymi (mimo drobnych różnic licencyjnych) wykorzystanie ich zarówno w projektach otwartych, jak i komercyjnych. Ważne jest, że w przypadku tej pierwszej wsparcie narzędzi służących do programowania C++ jest opcjonalne, dlatego trzeba się upewnić, że zostały one poprawnie zainstalowane ze środowiskiem (w rzeczywistości polega to na doinstalowaniu do środowiska narzędzi Visual C++ for Windows Desktop). Niedostępność opcji właściwych dla pełnych i profesjonalnych wersji oprogramowania nie przeszkadza w pełnym wykorzystaniu możliwości biblioteki wxWidgets.

Wybraną wersję środowiska Visual Studio możesz pobrać na stronie internetowej firmy Microsoft.

Wspomniane wyżej środowisko nie jest jedynym dostępnym narzędziem. Inne, jakie są przeznaczone dla systemów operacyjnych z rodziny MS Windows, to Code::Blocks będący doskonałym środowiskiem do programowania z biblioteką wxWidgets, a także Dev-C++⁴, który korzysta m.in. z kompilatora MinGW GCC. Wszędzie tam, gdzie jest to konieczne, na wybrane aspekty pracy z kompilatorem MinGW GCC zwracam uwagę przy okazji omawiania programowania w systemach linuksowych Ubuntu i Mint oraz środowisku Code::Blocks IDE.

⁴ Program ma również swój klon, który jest dedykowany pracy z biblioteką wxWidgets. Jest on upowszechniany pod nazwą wxDevC++. Program, oprócz przygotowanych wstępnie projektów wxWidgets, ma również narzędzie umożliwiające graficzne tworzenie interfejsu użytkownika z użyciem kontrolki dostępnych w wxWidgets.

Linux (Ubuntu i Mint)

Code::Blocks

O systemach operacyjnych Linux można powiedzieć, że są rajem dla programistów, a wszystko to dzięki ogromnej liczbie dostępnych narzędzi i wbudowanych bibliotek. Nie bez znaczenia pozostaje również otwarty kod tych narzędzi i całego systemu operacyjnego. Takie okoliczności aż proszą o niezliczone eksperymenty i testy.

Oprogramowanie Code::Blocks jest dość potężnym kombajnem programistycznym dostępnym zarówno dla systemów operacyjnych Linux, jak i MS Windows, dostarczającym profesjonalnych narzędzi ułatwiających tworzenie i kompilowanie programów pisanych w C++. Będąc użytkownikiem Linuksa, zapewne już dawno odkryłeś swoje ulubione narzędzia, jednak na potrzeby książki wszystkie przykłady i specyficzne problemy, jakie są związane z programowaniem w systemach linuksowych Ubuntu oraz Mint, będę prezentować na podstawie pracy ze środowiskiem Code::Blocks.

Code::Blocks IDE możesz pobrać ze strony projektu (<https://www.codeblocks.org>) lub zainstalować wraz ze wszystkimi wymaganymi komponentami za pomocą standardowego instalatora oprogramowania, w jaki jest wyposażona Twoja edycja Linuksa.

Code::Blocks IDE jest również dostępny w wersji dla systemów operacyjnych z rodziny MS Windows, może być zatem dobrym wyborem, jeżeli zechcesz mieć jak największą gwarancję łatwej przenośności swojego kodu między platformami systemowymi.

Inne

Poniższe narzędzia są dostępne dla wszystkich interesujących nas platform systemowych.

PoEdit

Biblioteka wxWidgets udostępnia czytelny i wydajny system internacjonalizacji aplikacji, dzięki któremu w łatwy sposób możesz dodać do swojego programu tłumaczenia na inne języki. PoEdit jest narzędziem służącym do wygodnego tworzenia i edycji plików tłumaczeń. Jeżeli chcesz programować aplikacje wielojęzyczne z wxWidgets, PoEdit powinien koniecznie znaleźć się w Twoim programistycznym warsztacie.

GIMP

GIMP-a nie trzeba przedstawiać nikomu, kto choć przez chwilę dotknął ogromnego świata wspaniałych narzędzi *open source*. Ten rozbudowany program graficzny wykorzystamy do ubrania naszego programu w piękne ikonki i obrazki. Oczywiście możesz używać swojego ulubionego programu graficznego, jednak przykłady zawarte w książce, a także dołączone do niej materiały dodatkowe, zawierają grafiki utworzone za pomocą programu GIMP.

Materiały dodatkowe do książki

Materiały dodatkowe oraz kody źródłowe programów przedstawionych w książce można pobrać w formie spakowanego archiwum ze strony internetowej:

```
http://it.pwn.pl/Artykuly/Programowanie/  
Programowanie-wieloplatformowe-z-C-i-wxWidgets-3
```

po wpisaniu hasła:

```
w8x18WxWidgets2e
```

Wszystkie zaprezentowane w książce przykłady, jakie znalazły odzwierciedlenie w materiałach dodatkowych do książki, zostały przygotowane do pracy ze środowiskami programistycznymi MS Visual Studio 2015 (Visual C++) oraz Code::Blocks 16.01. Były one uruchamiane i testowane w systemach operacyjnych MS Windows 7, 8.1 oraz 10, a także w systemie operacyjnym Ubuntu Linux w wersjach 16.04, 16.10 oraz 17.04 i systemie operacyjnym Mint Linux w wersjach 17.3 i 18.1.

Głównymi wersjami biblioteki, użytymi podczas kreowania programów przykładowych, były wxWidgets 3.1.0 oraz 3.1.1, choć w większości przypadków możliwe jest również kompilowanie ich z biblioteką w wersjach 3.0.x.

Jeśli podczas lektury książki i testowania przykładów zawartych w materiałach dodatkowych napotkasz wątpliwości i problemy wymagające bardziej szczegółowych objaśnień, możesz napisać do mnie kilka słów na adres bartosz.warzocho@gmail.com. Tak jak każdy w dzisiejszym rozpędzonym świecie, ja również często bywam zajęty, niemniej jednak dołożę wszelkich starań, aby szybko odpowiedzieć na Twoją wiadomość.

Errata i aktualizacje

Kondycja człowieka bezlitośnie kształtuje naturę jego czynów, a te czasami bywają skażone drobnymi pomyłkami. Jako że nie jestem żadnym wyjątkiem od tej reguły, podczas lektury książki możesz sporadycznie spotkać drobne pomyłki literowe, choć bezgranicznie wierzę, że tak nie będzie. Jeśli jednak mimo wszystko tak się stanie, wybac mi proszę i jeśli zechcesz, poświęć chwilę na napisanie do mnie, aby poinformować mnie o odkrytym błędzie. Wszelkie aktualizacje kodu źródłowego materiałów dodatkowych do książki oraz aktualną erratę możesz znaleźć na stronie internetowej Wydawnictwa Naukowego PWN.

Notka prawna

W pracy wykorzystano zmodyfikowaną ikonę Tux, zmodyfikowaną ikonę żarówki oraz zmodyfikowaną ikonę znaku zapytania, których pierwowzory znajdują się w domenie publicznej. Zostały one wykorzystane jako ikony wybranych ramek informacyjnych umieszczonych w tekście (zob. *Konwencje przyjęte w książce*).

Użyta w książce nazwa Linux jest zastrzeżonym znakiem towarowym Linusa Torvaldsa w Stanach Zjednoczonych i innych krajach, natomiast nazwy Microsoft, Windows, Word, Excel, Visual Studio, Visual C++ są zastrzeżonymi znakami towarowymi firmy Microsoft Corporation w Stanach Zjednoczonych i innych krajach. Prawa autorskie dotyczące biblioteki wxWidgets i jej dokumentacji należą do Juliana Smarta, Vadima Zeitlina, Stefana Csomora, Roberta Roebingha, a także innych członków zespołu wxWidgets (wymienionych w punkcie *Copyrights and Licenses* dokumentacji biblioteki).

Wszystkie materiały dodatkowe do książki (z wyjątkiem komponentów trzecich lub jeśli nie zastrzeżono inaczej) są udostępnione na zasadach licencji wxWidgets (zajrzyj do *Dodatku C*). Oznacza to, że możesz je swobodnie modyfikować, rozpowszechniać i zmieniać na warunkach tej licencji. Przede wszystkim bez ograniczeń możesz je wykorzystać we własnych projektach, dostosowując je do własnych potrzeb.