12. Tajemnice pracy z ciągami znaków w wxWidgets, czyli klasa wxString i nie tylko...

Najpotężniejszą grupą danych przetwarzanych niemal w każdej aplikacji komputerowej są ciągi znaków. Wraz z pojawieniem się języka C++ i jego biblioteki standardowej światło dzienne ujrzał nowy typ danych – *std::string*. Jego wprowadzenie wiązało się z koniecznością opracowania mechanizmów, jakie usprawniłyby przetwarzanie łańcuchów znaków w stosunku do mechanizmów języka C, gdzie wszelkie operacje na tego typu danych były wykonywane na niewygodnych buforach czy tablicach znaków wymagających zdyscyplinowanego podejścia do alokacji i zwalniania pamięci. Prawdziwą rewolucją, jaka wiązała się z typem *std::string*, było rozwiązanie właśnie tego problemu, a wszystko dzięki konstruktorowi kopiującemu oraz operatorowi przypisania. Należy również wspomnieć o potężnym zestawie narzędzi, w jaki ta nowa klasa została wyposażona. Składa się on z dziesiątków różnych metod realizujących różne zadania, jakie z przetwarzaniem ciągów znakowych mogą być kojarzone.

Mówiąc ogólnie, odzwierciedleniem typu *std::string* w świecie wxWidgets jest klasa *wxString*, lecz nie jest ona jedyną klasą związaną z przetwarzaniem łańcuchów znaków. Rolą niniejszego rozdziału jest przybliżenie różnych zagadnień związanych z przechowywaniem oraz przetwarzaniem danych tekstowych w aplikacjach wxWidgets.

12.1. Klasa wxString i jej stosowanie

Tworząc choćby najmniejszą z możliwych aplikację wxWidgets, nie sposób uniknąć stosowania klasy *wxString*. Dzieje się tak, ponieważ jest ona jedną z najważniejszych i najczęściej występujących klas wxWidgets. Z tego punktu dowiesz się niemal wszystkiego, co każdy programista wxWidgets powinien wiedzieć na jej temat.

Po pierwsze, klasa *wxString* jest podstawowym typem wxWidgets służącym do przechowywania i przetwarzania łańcuchów znaków. Stanowi swego rodzaju warstwę pośrednią między wxWidgets a standardowymi typami podstawowymi C++. W znakomitej większości przypadków argumenty wszelkich funkcji wxWidgets właśnie tego typu używają do przechwytywania łańcuchów znaków. Podobnie jest z wartościami zwracanymi przez większość funkcji przetwarzających teksty.

Po drugie, klasa *wxString* jest jak światło, lecz nie dlatego, że rozjaśnia wszystkie ciemności, które utrudniają pracę z ciągami znaków, ale dlatego, że tak jak ono, ma dwoistą naturę. Istotą tej dwoistości jest to, że klasa ta jest specyficznym kontenerem mogącym przechowywać zarówno zestawy znaków *char*, jak i zestawy unikodowych znaków szerokich *wxchar_t* (*wide char*), z czym wiążą się przeróżne konsekwencje, na które zechcę zwrócić Twoją uwagę w dalszej części.

Po trzecie, klasa *wxString* może funkcjonować samodzielnie jako jedyny typ tekstowy w całej aplikacji wxWidgets (jak to miało miejsce w niemal wszystkich aplikacjach przykładowych i treningowych, jakie wspólnie stworzyliśmy). Niemniej jednak musisz też wiedzieć, że twórcy biblioteki zalecają, aby stosować ją oszczędnie i tylko w kontekście, o jakim wspomniałem, tam zaś, gdzie jest to możliwe, wykorzystywać typy podstawowe (choć w praktyce jest to zwykle bardzo trudne, a czasami wręcz kompletnie zbędne, chyba że aplikacja, którą piszemy, musi wykonywać więcej konwersji między tekstowymi typami wxWidgets a podobnymi typami właściwymi dla innych bibliotek).

Po czwarte, ogromna część zestawu narzędzi, jaki ma klasa *wxString* pokrywa się z katalogiem składników klas standardowych, a więc *std::string* czy *std::wstring*, dlatego nie powinieneś mieć najmniejszych problemów z szybkim jego opanowaniem, a także rozpoznaniem jego poszczególnych elementów. Z tego też względu nie będziemy tu przytaczać wszystkich metod klasy *wxString*, a skupimy się na najbardziej charakterystycznych. Cały katalog metod klasy *wxString* został szczegółowo przedstawiony w dokumentacji biblioteki. My spróbujemy przyjrzeć się klasie *wxString* pod kątem zadaniowym.

12.1.1. Tworzenie obiektów wxString

Obiekty *wxString* można tworzyć na kilkanaście sposobów, a wszystko za sprawą pokaźnego zestawu konstruktorów, jakimi dysponuje omawiana klasa. Bliższe ich zbadanie w pełni pokazuje wspomnianą pośredniczącą rolę *wxString* w komunikacji biblioteki z typami podstawowymi, gdyż ich znakomita większość właśnie w typach podstawowych C++ ma swoje argumenty. Nie ma sensu, abyśmy analizowali konstrukcję wszystkich konstruktorów *wxString*, dlatego do zobrazowania powyższych słów posłużymy się kilkoma przykładami:

```
//-------------------------------------------------------------------------
// Użycie konstruktora kopiującego
wxString s1 = wxT("Uwielbiam programować w wxWidgets.");
wxString s2(a); // Zawiera ciąg a

//-------------------------------------------------------------------------
// Użycie standardowej tablicy znaków const char*
const char* a = "Najpotężniejszym językiem programowania jest C++.";
wxString s1(a); // Zawiera ciąg a

const char* b = "1234567890";
wxString s2(b, 5); // Zawiera ciąg od pierwszego do piątego znaku b, czyli "12345"

wxString s3(a, wxConvAuto(wxFONTENCODING_ISO8859_2)); // Ciąg z określonym kodowaniem

wxString s4(b, wxConvAuto(wxFONTENCODING_ISO8859_2), 5);
   // Jak s2 z określonym kodowaniem

wxString s5("Warto również znać inne języki programowania..."); // Zwykłe wpisanie ciągu

//-------------------------------------------------------------------------
// Użycie znaków char
char c1 = 'A';
wxString a(c1); // String zawiera tylko literę A

char c2 = 0x24;
wxString b(c2, 100); // String składa się ze 100 powtórzeń znaku $

//-------------------------------------------------------------------------
// Użycie ciągu std::string
std::string str = "Heleno, wobec Twej urody stoję jak dawni Niceanie...";
wxString s1(str);

wxString s2(std::string("Widzący sponad wonnej wody. Ojczyste brzegi i ogrody."));
wxString s3(std::string("Po wielu dniach na oceanie.");

//-------------------------------------------------------------------------
// Użycie standardowej tablicy znaków wchar_t
const wchar_t* a = "Najpotężniejszym językiem programowania jest C++.";
wxString s1(a); // Zawiera ciąg a

const wchar_t* b = "1234567890";
wxString s2(b, 5); // Zawiera ciąg od pierwszego do piątego znaku b, czyli "12345"
```