

IDŹ DO

PRZYKŁADOWY ROZDZIAŁ



SPIS TREŚCI

KATALOG KSIĄŻEK

KATALOG ONLINE

ZAMÓW DRUKOWANY KATALOG

TWÓJ KOSZYK

DODAJ DO KOSZYKA

CENNIK I INFORMACJE

ZAMÓW INFORMACJE
O NOWOŚCIACH

ZAMÓW CENNIK

CZYTELNIA

FRAGMENTY KSIĄŻEK ONLINE

Objective-C. Leksykon kieszonkowy

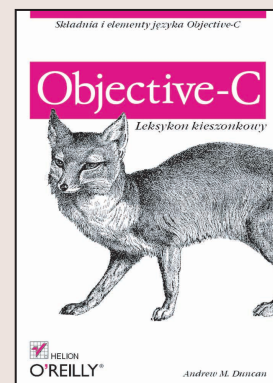
Autor: Andrew Duncan

Tłumaczenie: Leszek Mosingiewicz

ISBN: 83-7361-166-5

Tytuł oryginału: [Objective-C Pocket Reference](#)

Format: B5, stron: 144



Objective C prezentuje ekscytujące i dynamiczne podejście do programowania obiektowego opartego na C. Został ono użyty przez firmę Apple jako podstawa programowania dla systemu operacyjnego Mac OS X, który zdobywa coraz większą popularność wśród programistów. Pomimo że Objective-C jest (pomijając jego podstawę, czyli C) prostym językiem, nie można niedoceniać znaczenia pewnych jego rozszerzeń. Wykorzystanie wszystkich możliwości tego języka jest możliwe dzięki użyciu sprawdzonych wzorców projektowych, uważnej analizie przykładów kodu i prawidłowym wykorzystaniu dostępnych bibliotek. Zamierzeniem tego leksykonu jest dostarczenie szybkiej pomocy na temat składni i elementów języka Objective-C.

Książka zawiera krótkie omówienie podstawowych zagadnień oraz przykłady i definicje. Programiści przechodzący od kodowania w C++ lub Javie znajdą tu informacje pozwalające wykorzystać wszystkich możliwości nowego języka.

Oprócz omówienia składni języka, autor objaśnia też inne zagadnienia związane z językiem: zarządzanie pamięcią, dynamiczne ładowanie, obiekty rozproszone i obsługę wyjątków.



Spis treści

Wstęp	5
Czym jest Objective-C?	7
Dynamiczne wywołania.....	8
Dynamiczne przydzielanie typów	9
Dynamiczne ładowanie	9
Wybór wersji Objective-C	10
Od czego zacząć?	10
Elementy języka	11
Obiekty	12
Klasy.....	13
Dziedziczenie i typy pochodne.....	19
Pola	20
Metody	22
Kategorie	34
Protokoły	37
Deklaracje	40
Predefiniowane typy, stałe i zmienne.....	43
Dyrektywy kompilatora i preprocesora	47
Deklaracje i definicje klasy	47
Deklaracje wyprzedzające.....	47
Dyrektywy rozszerzające	48
Symbole preprocesora	52
Flagi kompilatora	53
Zdalne wiadomości	54
Kwalifikatory parametrów wskaźnikowych	55
Kwalifikatory wartości zwracanych.....	56
Kwalifikatory obiektów	56
Cykl życia obiektu	57
Tworzenie obiektu	58
Kopiowanie obiektu.....	65
Dealokacja obiektu	69

Błędy wykonania	71
Obsługa błędów obiektów	71
Wyjątki w Cocoa	72
Środowisko uruchomieniowe	77
Obiekty klas	78
Obiekty metaklas.....	80
Selektory	82
Obiekty protokołów.....	83
Klasy podstawowe	83
Pola.....	84
Metody	84
Klasa Object	85
Klasa NSObject	95
Przekazywanie wiadomości	106
Przekazywanie wiadomości w klasie Object	107
Przekazywanie wiadomości w klasie NSObject.....	108
Zarządzanie pamięcią	111
Manualne zarządzanie pamięcią	112
Zliczanie odwołań.....	113
Sprzątanie pamięci	119
Archiwizacja obiektów	120
Archiwizowanie potomków klasy Object	120
Archiwizowanie potomków klasy NSObject.....	123
Kodowanie klucz-wartość.....	127
Uprawnienia dostępu	127
Metody kodowania NSKeyValue	128
Obsługa błędów poszukiwania kluczy	132
Optymalizacja wywołań metod	132
Objective-C++	134
Dodatkowe informacje o Objective-C	136
Skorowidz	138

Czym jest Objective-C?

Objective-C jest językiem zorientowanym obiektowo (ang. *object oriented language*). Jako taki wspiera hierarchię typów (klas), przesyłanie wiadomości pomiędzy obiektami oraz wielokrotne wykorzystanie kodu dzięki mechanizmowi dziedziczenia. Objective-C dodaje te cechy do istniejącego języka C.

Objective-C jest rozszerzeniem języka C. Z tego powodu wiele cech programu napisanego w tym języku zależy od właściwości narzędzi programistycznych C. Zaliczyć do nich można:

- rozmiar wartości skalarnych, takich jak liczby całkowite i zmiennoprzecinkowe;
- miejsca umieszczania deklaracji o ograniczonym zasięgu;
- niejawną konwersję typów;
- przechowywanie ciągów znakowych;
- rozszerzenia i makra preprocesora;
- poziomy ostrzeżeń kompilatora;
- optymalizację kodu;
- ścieżki poszukiwania plików dołączanych i linkowanych.

Więcej informacji na temat tych zagadnień można znaleźć w dokumentacji używanych narzędzi programistycznych dla konkretnej platformy systemowej.

Różnica pomiędzy Objective-C a C++, innym obiektowym rozszerzeniem języka C, polega na tym, że decyzje, które w przypadku programów napisanych w C++ są podejmowane w czasie kompilacji, w Objective-C zostają odroczone do fazy wykonania programu. Objective-C można rozpoznać też po poniższych kluczowych cechach:

- dynamiczne wywołania (doręczanie wiadomości);
- dynamiczne przydzielanie typów;
- dynamiczne ładowanie.

Dynamiczne wywołania

W językach zorientowanych obiektowo wywołania funkcji są zastąpione przez **wiadomości** (ang. *messages*). Różnica polega na tym, że ta sama wiadomość może w czasie wykonania uruchomić inny fragment kodu, w zależności od rodzaju odbiorcy wiadomości. W języku Objective-C decyzja taka jest podejmowana dynamicznie w czasie wykonywania programu a realizowana przez odszukiwanie klas, dla których dana wiadomość jest przeznaczona i ich klas bazowych (środowisko uruchomieniowe Objective-C zapewnia przechowanie wyników poszukiwania w pamięci podręcznej a tym samym poprawę wydajności procesu). W C++ tablica wywołań jest tworzona statycznie w fazie kompilacji programu.

Liniowy sposób poszukiwania odbiorcy, wykorzystywany przez Objective-C jest w pełni zgodny z naszym wyobrażeniem dziedziczenia. Z tego względu łatwo jest zrozumieć sposób działania programu utworzonego w Objective-C. Dynamiczne wywołania mogą reagować na zmiany w hierarchii dziedziczenia w czasie wykonania programu. W wypadku obiektów rozproszonych model dynamicznego przesyłania wiadomości stanowi lepsze rozwiązanie niż model oparty na stałej tablicy.

Dynamiczne przydzielanie typów

Dynamiczne przesyłanie wiadomości w Objective-C umożliwia programiście wskazanie jako ich adresatów obiektów, których typ nie został jeszcze zadeklarowany. Środowisko Objective-C rozpo-

znaje dynamicznie, w fazie wykonania, klasę odbiorcy wiadomości i uruchamia odpowiedni kod. Dla porównania, C++ wymaga, by typ odbiorcy został określony statycznie już w czasie kompilacji programu, kiedy tworzona jest tablica wywołań.

Statyczne definiowanie typów umożliwia kompilatorowi wykrycie pewnych błędów w programie. Kontrola typów nie jest jednak jednoznaczna, co oznacza, że nie istnieje algorytm pozwalający na jednoznaczne wskazanie programu z błędami typów. Zatem kompilator musi albo pominąć pewne błędy, albo zabronić wykonania bezpiecznych operacji. Oczywiście w praktyce jest wybierana druga możliwość, co oznacza, że programy, które mogłyby się bezpiecznie wykonać nie zostają skompilowane. Dynamiczna kontrola typów umożliwia uruchamianie programów, których poprawność mogłyby zakwestionować kompilator.

Objective-C umożliwia programistom korzystanie ze statycznej kontroli typów, jeśli tego zechcą, a kiedy jest to potrzebne, mogą oni nadawać typy dynamicznie. Zatem pytanie: *Jaki jest typ odbiorcy w czasie kompilacji?* zostaje zastąpione przez: *Na jakie wiadomości ma odpowiadać obiekt w czasie wykonywania programu?* Jest to bardzo pożyteczna cecha, ponieważ programy działają tylko wtedy, gdy są uruchomione.

Dynamiczne ładowanie

Jako że proces przesyłania wiadomości jest prosty i spójny, łatwo jest przesunąć konsolidację niezależnie skompilowanych modułów do czasu wykonania programu. Programy w Objective-C można podzielić na niezależne elementy, dynamicznie pobierane do pamięci i wykonywane. Możliwość ta ułatwia pracę w sieci, projektowanie aplikacji rozproszonych oraz programów, które mogą być rozwijane przez niezależnych projektantów (firmy).

Wybór wersji Objective-C

Programiści działający w środowisku Unix prawdopodobnie posiadają już kompilator Objective-C — *gcc*, który stanowi część wielu instalacji tego systemu. Jest on dostępny na zasadach licencji publicznej GNU. W niniejszym podręczniku omówiono możliwości języka kompilatora *gcc* w wersji 3.1, dostępnego na wielu różnych platformach sprzętowych i programowych.

Kompilator *gcc* został również zaadaptowany przez Apple Computer dla platformy OS X, opartej na odmianie systemu Unix o nazwie Darwin. Darwin dostarcza własne środowisko uruchomieniowe Objective-C oraz bibliotekę klas o nazwie Cocoa. W tej publikacji zwrócono uwagę na różnice występujące pomiędzy środowiskami uruchomieniowymi Darwin i GNU. Opisano również podstawowe klasy dostępne w GNU i Cocoa.

Istnieje również biblioteka klas o nazwie GNUstep, rozpowszechniana na warunkach licencji GNU Lesser (Library) Public Licence. GNUstep stanowi rozwinięcie kodu, będącego podstawą biblioteki Cocoa i w dużej części jest z nią zgodny. Zatem przedstawiona dalej dyskusja na temat elementów Cocoa, takich jak klasa podstawowa *NSObject*, odnosi się też do biblioteki GNUstep.