
Skuteczny nowoczesny C++

Scott Meyers

przekład: Maria Chaniewska

APN Promise
Warszawa 2015

O'REILLY®

Spis treści

<i>Podziękowania</i>	ix
<i>Wprowadzenie</i>	1
Terminologia i konwencje.....	2
Raportowanie błędów i sugerowanie poprawek	8
1 Dedukcja typów	9
Punkt 1: Dedukcja typów w szablonach.....	9
Przypadek 1: <i>ParamType</i> to odwołanie lub wskaźnik, ale nie odwołanie uniwersalne	11
Przypadek 2: <i>ParamType</i> jest odwołaniem uniwersalnym	13
Przypadek 3: <i>ParamType</i> nie jest ani wskaźnikiem, ani odwołaniem ..	14
Argumenty tablicowe	16
Argumenty funkcyjne	18
Punkt 2: Dedukcja typu <code>auto</code>	20
Punkt 3: <code>decltype</code>	26
Punkt 4: Jak wyświetlać wydedukowane typy	34
Edytory IDE	34
Diagnostyka kompilatora.....	34
Wyniki czasu wykonania	35
2 <code>auto</code>	41
Punkt 5: Preferuj deklarację <code>auto</code> zamiast jawnych deklaracji typów....	41
Punkt 6: Stosuj idiom jawnego inicjatora typu, gdy deklaracja <code>auto</code> powoduje dedukcję niepożądanych typów	48
3 Droga do nowoczesnego języka C++	55
Punkt 7: Rozróżniaj między <code>()</code> a <code>{}</code> podczas tworzenia obiektów	55
Punkt 8: Preferuj <code>nullptr</code> zamiast <code>0</code> i <code>NULL</code>	67

Punkt 9:	Preferuj deklaracje aliasów zamiast typedef.	72
Punkt 10:	Preferuj wyliczenia enum z zasięgiem zamiast bez zasięgu	77
Punkt 11:	Preferuj funkcje usunięte zamiast prywatnych niezdefiniowanych.	85
Punkt 12:	Deklaruj funkcje nadpisujące jako override.	91
Punkt 13:	Preferuj iteratory const_iterator zamiast iterator	99
Punkt 14:	Deklaruj funkcje jako noexcept, jeśli nie zgłaszają wyjątków.	104
Punkt 15:	Stosuj constexpr, kiedy to tylko możliwe	112
Punkt 16:	Dbaj o bezpieczeństwo wątkowe funkcji składowych const	120
Punkt 17:	Generowanie specjalnych funkcji składowych	127
4	Wskaźniki inteligentne.	137
Punkt 18:	Stosuj wskaźniki std::unique_ptr do zarządzania zasobami posiadanymi wyłącznie	139
Punkt 19:	Stosuj wskaźniki std::shared_ptr w przypadku zarządzania zasobami o współdzielonym posiadaniu	146
Punkt 20:	Stosuj typ std::weak_ptr dla wskaźników przypominających std::shared_ptr, które mogą zawisnąć	157
Punkt 21:	Preferuj funkcje std::make_unique i std::make_shared zamiast bezpośredniego używania instrukcji new	163
Punkt 22:	Podczas używania idiomu Pimpl definiuj specjalne funkcje składowe w pliku implementacji	173
5	Odwołania do r-wartości, semantyka przenoszenia i przekazywanie doskonałe.	185
Punkt 23:	std::move i std::forward	186
Punkt 24:	Odróżniaj odwołania uniwersalne od odwołań do r-wartości	193
Punkt 25:	Stosuj std::move w przypadku odwołań do r-wartości, a std::forward w przypadku odwołań uniwersalnych	199
Punkt 26:	Unikaj przeciążania w przypadku odwołań uniwersalnych.	209
Punkt 27:	Zapoznaj się z alternatywami przeciążania odwołań uniwersalnych	217
	Rezygnacja z przeciążania	218
	Przekazywanie przez const T&	218
	Przekazywanie przez wartość	218
	Technika tag dispatch.	219
	Ograniczanie szablonów przyjmujących odwołania uniwersalne	223

Kompromisy	231
Punkt 28: Zwijanie odwołań.....	233
Punkt 29: Załóż, że operacje przenoszenia nie są ani obecne, ani tanie, ani używane	241
Punkt 30: Zapoznaj się z przypadkami niepowodzeń przenoszenia doskonałego	245
Inicjatory klamrowe	247
0 lub NULL jako wskaźniki null	249
Sama deklaracja całkowitoliczbowych danych składowych o kwalifikatorach <code>static const</code>	249
Przeciążone nazwy funkcji i nazwy szablonów.....	251
Poła bitowe	253
Podsumowanie	255
6 Wyrażenia lambda	257
Punkt 31: Unikaj domyślnych trybów przechwytywania.....	259
Punkt 32: Stosuj przechwytywanie inicjujące do przenoszenia obiektów do domknięć	268
Punkt 33: Stosuj <code>decltype</code> do parametrów <code>auto&&</code> , aby je przekazywać za pomocą <code>std::forward</code>	275
Punkt 34: Preferuj wyrażenia lambda zamiast <code>std::bind</code>	278
7 Interfejs API współbieżności.....	289
Punkt 35: Preferuj programowanie oparte na zadaniach zamiast opartego na wątkach	289
Punkt 36: Określ zasadę <code>std::launch::async</code> , jeśli asynchroniczność jest istotna.....	294
Punkt 37: Doprowadź wątki <code>std::thread</code> do stanu nieprzyłączalnego na wszystkich ścieżkach	300
Punkt 38: Uważaj na różnorodne działanie destruktorów uchwytów wątków	309
Punkt 39: Rozważ obiekty <code>future</code> typu <code>void</code> do komunikacji zdarzeń jednorazowych	314
Punkt 40: Stosuj <code>std::atomic</code> dla współbieżności, a <code>volatile</code> dla pamięci specjalnej	325

8 Szlify.....	335
Punkt 41: Rozważ przekazywanie przez wartość parametrów, które można kopiować i tanio przenosić – o ile są zawsze kopiowane.....	335
Punkt 42: Rozważ umieszczanie zamiast wstawiania.....	348
<i>Indeks</i>	367
<i>O autorze</i>	361