

Spis treści

	O autorze	9
	O recenzencie technicznym	11
	Podziękowania	13
Rozdział 1	Wprowadzenie	15
	Entropia oprogramowania	16
	Czysty kod	17
	Dlaczego C++?	18
	C++11 — początek nowej ery	18
	Dla kogo przeznaczona jest ta książka?	19
	Konwencje stosowane w tej książce	19
	Ramki	20
	Uwagi, wskazówki i ostrzeżenia	20
	Przykładowy kod	20
	Witryna książki i repozytorium z kodem źródłowym	21
	Diagramy UML-a	21
Rozdział 2	Tworzenie siatki bezpieczeństwa	23
	Konieczność przeprowadzania testów	23
	Wprowadzenie do testów	25
	Testy jednostkowe	26
	A co z kontrolą jakości?	28
	Reguły tworzenia dobrych testów jednostkowych	29
	Jakość kodu testów	29
	Nazwy testów jednostkowych	29
	Niezależność testów jednostkowych	31
	Jedna asercja na test	31
	Niezależne inicjowanie środowisk testów jednostkowych	32
	Pomijanie testów getterów i setterów	32
	Pomijanie testów kodu innych programistów	32
	Pomijanie testów zewnętrznych systemów	33
	A co zrobić z bazą danych?	33

	Nie łącz kodu testów z kodem produkcyjnym	33
	Testy muszą działać szybko	36
	Zaśleпки	36
Rozdział 3	Postępuj zgodnie z zasadami	39
	Czym są zasady?	39
	Zachowaj prostotę, głupku (KISS)	40
	Nie będziesz tego potrzebować (YAGNI)	40
	Nie powtarzaj się (DRY)	41
	Ukrywanie informacji	41
	Wysoka spójność	44
	Luźne powiązanie	46
	Nie przesadzaj z optymalizacją	49
	Zasada minimalizowania zaskoczenia	50
	Reguła harcerza	50
Rozdział 4	Podstawy czystego C++	53
	Dobre nazwy	54
	Nazwy powinny być czytywne	55
	Stosuj nazwy z dziedziny	56
	Dobieraj nazwy na odpowiednim poziomie abstrakcji	57
	Unikaj nadmiarowości, gdy wymyślasz nazwę	58
	Unikaj zagadkowych skrótów	58
	Unikaj notacji węgierskiej i przedrostków	59
	Unikaj używania tej samej nazwy do różnych celów	60
	Komentarze	60
	Niech kod opowiada historię	60
	Nie komentuj oczywistych rzeczy	61
	Nie dezaktywuj kodu za pomocą komentarzy	61
	Nie pisz komentarzy blokowych	62
	Rzadkie scenariusze, w których komentarze są przydatne	64
	Funkcje	67
	Jedna rzecz — nie więcej!	70
	Twórz małe funkcje	70
	Nazwy funkcji	71
	Stosuj nazwy opisujące intencje	72
	Argumenty i zwracane wartości	72
	Liczba argumentów	73
	Projekty C++ w dawnym stylu specyficznym dla C	82
	Przedkładaj łańcuchy znaków i strumienie z C++ nad dawne łańcuchy char* w stylu języka C	82
	Unikaj instrukcji printf(), sprintf(), gets() itd.	84
	Przedkładaj kontenery z biblioteki standardowej nad proste tablice w stylu języka C	87
	Używanie rzutowania z języka C++ zamiast dawnego rzutowania w stylu języka C	89
	Unikaj makr	90
Rozdział 5	Zaawansowane aspekty współczesnego C++	93
	Zarządzanie zasobami	93
	Idiom RAII	95

Inteligentne wskaźniki	95
Unikanie bezpośrednich wywołań new i delete	100
Zarządzanie niezależnymi zasobami	101
Warto się czasem gdzieś przenieść	102
Czym jest semantyka przenoszenia?	102
Czym są l-wartości i r-wartości?	103
Referencje do r-wartości	104
Nie wymuszaj wszędzie semantyki przenoszenia	106
Reguła zera	106
Kompilator to Twój współpracownik	110
Automatyczna dedukcja typów	110
Obliczenia na etapie kompilacji	113
Szablony zmiennych	115
Nie dopuszczaj do niezdefiniowanych skutków	116
Programowanie z użyciem typów semantycznych	117
Poznaj używane biblioteki	123
Korzystaj z pliku nagłówkowego <algorithm>	123
Korzystaj z biblioteki Boost	128
Inne biblioteki, które powinienś znać	129
Prawidłowa obsługa wyjątków i błędów	130
Lepiej zapobiegać niż leczyć	130
Wyjątek jest wyjątkiem — dosłownie	134
Jeśli nie możesz przywrócić stanu, szybko zamknij program	135
Definiuj specyficzne typy wyjątków	135
Zgłaszanie przez wartość i przechwytywanie za pomocą stałej referencji	137
Zwracaj uwagę na właściwą kolejność klauzul catch	137

Rozdział 6	Podejście obiektowe	139
	Myślenie obiektowe	140
	Abstrakcja — klucz do opanowania złożoności	141
	Zasady poprawnego projektowania klas	141
	Twórz niewielkie klasy	141
	Zasada jednej odpowiedzialności	142
	Zasada otwarte – zamknięte	143
	Zasada podstawiania Liskov	144
	Zasada podziału interfejsu	154
	Zasada zależności acyklicznych	156
	Zasada odwracania zależności	158
	Nie rozmawiaj z nieznanymi (prawo Demeter)	162
	Unikaj „anemicznych” klas	166
	Mów zamiast pytać	167
	Unikaj statycznych składowych klasy	169
Rozdział 7	Programowanie funkcyjne	171
	Czym jest programowanie funkcyjne?	172
	Czym jest funkcja?	173
	Funkcje czyste i „nieczyste”	174

Programowanie funkcyjne w nowoczesnym C++	175
Programowanie funkcyjne z użyciem szablonów języka C++	175
Obiekty podobne do funkcji (funktory)	177
Mechanizm wiązania i nakładki na funkcje	183
Wyrażenia lambda	185
Generyczne wyrażenia lambda (C++14)	187
Funkcje wyższego poziomu	187
Mapowanie, filtrowanie i redukcja	189
Czysty kod w programowaniu funkcyjnym	192
Rozdział 8 Programowanie sterowane testami	195
Wady zwykłych dawnych testów jednostkowych	196
Podejście TDD jako rewolucja	197
Proces pracy w TDD	197
TDD na przykładzie — kata dotyczące liczb rzymskich	200
Zalety TDD	216
Kiedy nie stosować TDD?	217
Rozdział 9 Wzorce projektowe i idiomy	219
Zasady projektowe a wzorce projektowe	220
Wybrane wzorce i sytuacje, w których warto je stosować	220
Wstrzykiwanie zależności	221
Adapter	231
Strategia	233
Polecenie	237
Procesor poleceń	240
Kompozyt	242
Obserwator	245
Fabryka	250
Fasada	252
Klasa Money	253
Obiekt reprezentujący specjalny przypadek (obiekt NULL)	256
Czym jest idiom?	260
Przydatne idiomy języka C++	260
Dodatek A Krótki przewodnik po UML-u	271
Diagramy klas	271
Klasa	271
Interfejs	273
Asocjacja	275
Generalizacja	277
Zależność	278
Komponenty	279
Stereotypy	279
Bibliografia	281
Skorowidz	285