

Spis treści

Przedmowa	13
1. Konfigurowanie środowiska Go	17
Instalowanie narzędzi Go	17
Środowisko robocze Go	18
Polecenie go	19
Polecenia go run i go build	19
Instalowanie pomocniczych narzędzi Go	20
Formatowanie kodu	21
Analiza składowa i weryfikacja kodu	23
Dobór narzędzi	25
Visual Studio Code	25
GoLand	25
Go Playground	27
Pliki reguł	28
Aktualizacja	30
Podsumowanie	31
2. Typy podstawowe i deklaracje	32
Typy wbudowane	32
Wartość zerowa	32
Literały	32
Zmienne boolowskie	34
Typy numeryczne	34
Przedsmak informacji o łańcuchach i runach	40
Jawna konwersja typów	41
Słowo kluczowe var a operator :=	42
Zastosowanie słowa kluczowego const	44
Stałe typowane i nietypowane	45

Niewykorzystane zmienne	46
Nazywanie zmiennych i stałych	47
Podsumowanie	48
3. Typy złożone	49
Tablice — za mało elastyczne, by używać ich bezpośrednio	49
Wycinki	51
Funkcja len	52
Funkcja append	52
Pojemność	53
Funkcja make	54
Deklarowanie wycinka	55
Wycinki wycinków	57
Przekształcanie tablic w wycinki	59
Funkcja copy	59
Łańcuchy, runy i bajty	61
Mapy	63
Mapy — czytanie i zapisywanie danych	66
Idiom „comma ok”	66
Usuwanie elementów z map	67
Używanie map jako zbiorów	67
Struktury	68
Struktury anonimowe	70
Porównywanie i konwertowanie struktur	71
Podsumowanie	72
4. Bloki, przesłanianie oraz struktury sterujące	73
Bloki	73
Przesłanianie zmiennych	74
Wykrywanie przesłoniętych zmiennych	75
Instrukcja if	76
Cztery wersje pętli for	78
Pełna pętla for	78
Warunkowa pętla for	79
Nieskończona pętla for	79
Instrukcje break i continue	80
Instrukcja for-range	81
Oznaczanie instrukcji etykietami	85
Wybór właściwej wersji pętli for	86
Instrukcja switch	88
Puste instrukcje switch	90

Wybór pomiędzy instrukcjami if a switch	92
Instrukcja goto... tak, goto!	92
Podsumowanie	94
5. Funkcje	95
Deklarowanie i wywoływanie funkcji	95
Symulowanie parametrów nazwanych i opcjonalnych	96
Wycinki a zmienna liczba parametrów wejściowych	97
Zwracanie wielu wartości	98
Jeśli funkcja zwraca wiele wartości, należy je traktować osobno	98
Ignorowanie zwróconych wartości	99
Nazywanie zwracanych wartości	99
Pusta instrukcja return — stosowanie surowo wzbronione!	101
Funkcje to wartości	102
Deklaracja typu funkcyjnego	103
Funkcje anonimowe	104
Domknięcia	105
Przekazywanie funkcji jako argumentów	105
Zwracanie funkcji przez inne funkcje	106
Instrukcja defer	107
W Go obowiązuje wywołanie przez wartość	111
Podsumowanie	113
6. Wskaźniki	114
Krótkie wprowadzenie do wskaźników	114
Nie bój się wskaźników	117
Wskaźniki oznaczają argumenty modyfikowalne	119
Wskaźniki są ostatnią deską ratunku	121
Wydajność przekazywania wskaźników	124
Wartość zerowa a brak wartości	124
Różnica między mapami a wycinkami	125
Wycinki w charakterze buforów	128
Jak ułatwić pracę mechanizmowi garbage collector	129
Podsumowanie	132
7. Typy, metody i interfejsy	133
Typy w języku Go	133
Metody	134
Odbiorcy wskaźników i odbiorcy wartości	135
Programowanie metod pod kątem instancji nil	137
Metody także są funkcjami	138

Funkcje kontra metody	139
Deklaracje typów nie oznaczają dziedziczenia	139
Typy są elementem dokumentacji programu	140
Iota służy do tworzenia typów wyliczeniowych... czasami	140
Używaj osadzania w przypadku kompozycji	142
Osadzanie nie jest dziedziczeniem	143
Krótki przewodnik po interfejsach	145
Interfejsy są bezpieczne pod względem typów (duck typing)	145
Osadzanie a interfejsy	148
Przyjmuj interfejsy, zwracaj struktury	149
Interfejsy a nil	150
Pusty interfejs nie informuje o niczym	151
Asercja i przełączniki typów	152
Asercje i przełączniki typów warto stosować oszczędnie	154
Typy funkcyjne są pomostem dla interfejsów	156
Niejawna implementacja interfejsu ułatwia wstrzykiwanie zależności	157
Wire	161
Go nie jest typowym językiem zorientowanym obiektowo (i bardzo dobrze)	161
Podsumowanie	162
8. Błędy	163
Obsługa błędów — podstawy	163
W przypadku prostych błędów używaj łańcuchów znaków	165
Błędy typu sentinel	165
Błędy to wartości	167
Opakowywanie błędów	169
Funkcje Is i As	171
Opakowywanie błędów przy użyciu instrukcji defer	174
Funkcje panic i recover	175
Tworzenie rzutu stosu w przypadku błędu	177
Podsumowanie	177
9. Moduły, pakiety i importowanie	178
Repozytoria, moduły i pakiety	178
Plik go.mod	179
Tworzenie pakietów	179
Importowanie i eksportowanie	179
Tworzenie pakietu i uzyskiwanie dostępu do niego	180
Nazewnictwo pakietów	182
Struktura modułu	182
Zastępowanie nazwy pakietu	183

Komentarze dotyczące pakietów i format godoc	184
Pakiet internal	185
Funkcja init — unikaj jej, jeśli to możliwe	186
Zależności cykliczne	187
Eleganckie podejście do zmiany nazw i porządkowania API	188
Obsługa modułów	190
Importowanie kodu	190
Obsługa wersji	192
Wybór wersji minimalnej	194
Aktualizowanie zgodnych wersji	195
Aktualizowanie niezgodnych wersji	195
Vendoring	197
Serwis pkg.go.dev	197
Dodatkowe informacje	198
Publikowanie modułu	198
Wersjonowanie modułu	198
Serwery proxy modułów	199
Wybór serwera proxy	200
Repozytoria prywatne	200
Podsumowanie	201
10. Przetwarzanie współbieżne w Go	202
Kiedy warto używać przetwarzania współbieżnego	202
Goprocudury	203
Kanały	205
Czytanie, zapisywanie i buforowanie	205
Pętla for-range i kanały	206
Zamykanie kanału	206
Zachowania kanałów	207
Instrukcja select	208
Zalecane rozwiązania i wzorce dotyczące współbieżności	210
API powinno być pozbawione współbieżności	211
Goprocudury, pętle for i zmieniające się... zmienne	211
Pamiętaj o „sprzątanu” po goprocudurach	212
Wzorec z kanałem „done”	213
Zastosowanie funkcji anulującej do wstrzymania goprocudury	214
Kiedy używać kanałów buforowanych, a kiedy niebuforowanych	214
Mechanizm backpressure	215
Wyłączanie klauzuli case w instrukcji select	216
Określanie limitu czasu	217
Zastosowanie struktury WaitGroup	218

Uruchamianie kodu dokładnie raz	220
Łączenie różnych aspektów przetwarzania współbieżnego	221
Kiedy używać muteksów zamiast kanałów	224
Operacje atomowe — raczej Ci się nie przydadzą	227
Gdzie szukać dodatkowych informacji o współbieżności?	228
Podsumowanie	228
11. Biblioteka standardowa	229
Pakiet io i przyjaciele	229
Pakiet time	234
Czas od uruchomienia systemu	235
Liczniki czasu i limit czasu	236
Pakiet encoding/json	236
Zastosowanie znaczników struktur w celu dodania metadanych	236
Unmarshaling i marshaling	238
JSON i operacje odczytywania i zapisywania	238
Kodowanie i dekodowanie strumieni JSON	240
Niestandardowe przetwarzanie obiektów JSON	241
Pakiet net/http	242
Klient	242
Serwer	244
Podsumowanie	248
12. Kontekst	249
Czym jest kontekst?	249
Anulowanie	252
Liczniki czasu	255
Anulowanie kontekstu we własnym kodzie	257
Wartości	258
Podsumowanie	263
13. Pisanie testów	264
Podstawy testowania	264
Zgłaszanie błędów w testach	265
Konfigurowanie i demontowanie	266
Przechowywanie prostych danych testowych	268
Przechowywanie wyników testowania w pamięci podręcznej	268
Testowanie publicznego API	268
Porównywanie wyników testów przy użyciu modułu go-cmp	269
Testy tablicowe	271

Sprawdzanie pokrycia kodu	273
Testowanie wydajności	275
Załączki w języku Go	278
Pakiet httptest	282
Testy integracyjne i znaczniki kompilacji	284
Wyszukiwanie problemów ze współbieżnością przy użyciu narzędzia race checker	286
Podsumowanie	287
14. W krainie smoków: mechanizm refleksji oraz pakiety unsafe i cgo	288
Mechanizm refleksji umożliwia manipulowanie typami podczas działania programu	289
Typy, rodzaje i wartości	290
Tworzenie nowych wartości	294
Zastosowanie refleksji do sprawdzenia, czy wartość interfejsu wynosi nil	295
Tworzenie marshalera danych przy użyciu mechanizmu refleksji	296
Zastosowanie refleksji do tworzenia funkcji automatyzujących powtarzalne zadania	300
Przy użyciu refleksji da się tworzyć struktury, ale... nie rób tego	301
Przy użyciu refleksji nie da się tworzyć metod	301
Używaj refleksji tylko wtedy, gdy to się opłaca	301
Pakiet unsafe rzeczywiście jest niebezpieczny	303
Zastosowanie pakietu unsafe do konwersji zewnętrznych danych binarnych	303
Pakiet unsafe a łańcuchy i wycinki	306
Narzędzia związane z pakietem unsafe	307
W pakiecie cgo chodzi o integrację, nie o wydajność	308
Podsumowanie	311
15. Spojrzenie w przyszłość — typy sparametryzowane w Go	312
Typy sparametryzowane ograniczają redundantność kodu i zwiększają bezpieczeństwo typowania	312
Wstęp do typów sparametryzowanych w Go	315
Zastosowanie list typów do określania operatorów	319
Funkcje sparametryzowane i algorytmy abstrakcyjne	319
Listy typów ograniczają możliwość przypisywania stałych oraz implementacji	321
Pominięte kwestie	323
Typy sparametryzowane a idiomatyczny kod w Go	325
Co niesie przyszłość?	326
Podsumowanie	326