

Spis treści

O autorze	13
O redaktorze merytorycznym	14
Wprowadzenie	15
Rozdział 1. Nowe funkcje w C++17	21
Wprowadzenie	21
Użycie strukturalnych wiązań do rozpakowania wartości zwrótej	22
Jak to zrobić?	22
Jak to działa?	24
Co dalej?	24
Ograniczanie zasięgu zmiennej do konstrukcji if i switch	26
Jak to zrobić?	26
Jak to działa?	27
Co dalej?	28
Zalety stosowania nowych reguł inicjalizacji z użyciem składni opartej na nawiasach	29
Jak to zrobić?	29
Jak to działa?	30
Umożliwienie konstruktorowi automatycznego określenia typu klasy szablonu	31
Jak to zrobić?	31
Jak to działa?	31
Co dalej?	32
Użycie wyrażenia constexpr-if do uproszczenia decyzji podejmowanych podczas kompilacji	33
Jak to zrobić?	34
Jak to działa?	35
Co dalej?	36
Włączenie bibliotek w postaci samych nagłówków z użyciem osadzonych zmiennych	37
Jak to zrobić?	37
Jak to działa?	38
Co dalej?	39

Implementowanie za pomocą wyrażeń fold przydatnych funkcji pomocniczych	40
Jak to zrobić?	40
Jak to działa?	41
Co dalej?	41
Rozdział 2. Kontenery STL	47
Wprowadzenie	48
Magazyn danych znajdujących się obok siebie	48
Magazyn danych w postaci listy	49
Drzewo wyszukiwania	49
Tabela wartości hash	50
Adapter kontenera	50
Użycie stylu usuń – wymaż w kontenerze <code>std::vector</code>	50
Jak to zrobić?	51
Jak to działa?	52
Co dalej?	54
Usuwanie w czasie $O(1)$ elementów z nieposortowanego kontenera <code>std::vector</code>	54
Jak to zrobić?	55
Jak to działa?	57
Uzyskanie bezpiecznego dostępu do egzemplarzy <code>std::vector</code>	58
Jak to zrobić?	58
Jak to działa?	59
Co dalej?	60
Sortowanie egzemplarzy <code>std::vector</code>	60
Jak to zrobić?	60
Jak to działa?	62
Co dalej?	62
Efektywne i warunkowe wstawianie elementów do kontenera <code>std::map</code>	63
Jak to zrobić?	63
Jak to działa?	65
Co dalej?	66
Stosowanie nowej semantyki podpowiedzi podczas wstawiania elementów za pomocą <code>std::map::insert</code>	66
Jak to zrobić?	66
Jak to działa?	68
Co dalej?	68
Efektywne modyfikowanie kluczy elementów <code>std::map</code>	69
Jak to zrobić?	70
Jak to działa?	72
Co dalej?	72
Użycie kontenera <code>std::unordered_map</code> z niestandardowymi typami danych	73
Jak to zrobić?	73
Jak to działa?	75
Filtrowanie duplikatów w danych wejściowych użytkownika i wyświetlanie ich w kolejności alfabetycznej za pomocą kontenera <code>std::set</code>	76
Jak to zrobić?	76
Jak to działa?	77

Implementowanie za pomocą kontenera std::stack prostego kalkulatora RPN	79
Jak to zrobić?	80
Jak to działa?	82
Co dalej?	84
Implementowanie za pomocą kontenera std::map licznika częstotliwości występowania słów	85
Jak to zrobić?	85
Jak to działa?	87
Implementowanie za pomocą kontenera std::set narzędzia pomocniczego przeznaczonego do wyszukiwania bardzo długich zdań w tekście	88
Jak to zrobić?	89
Jak to działa?	91
Co dalej?	92
Implementowanie za pomocą kontenera std::priority_queue listy rzeczy do zrobienia	93
Jak to zrobić?	93
Jak to działa?	95
Rozdział 3. Iteratory	97
Wprowadzenie	97
Kategorie iteratorów	99
Tworzenie własnego zakresu, który można iterować	101
Jak to zrobić?	101
Jak to działa?	103
Tworzenie własnych iteratorów zgodnych z kategoriami iteratora STL	104
Jak to zrobić?	104
Jak to działa?	106
Co dalej?	107
Użycie adapterów iteratora do wypełniania ogólnych struktur danych	107
Jak to zrobić?	107
Jak to działa?	109
Implementowanie algorytmów w kategoriach iteratorów	110
Jak to zrobić?	111
Co dalej?	113
Iteracja w drugą stronę za pomocą adaptera iteratora odwrotnego	114
Jak to zrobić?	114
Jak to działa?	115
Zakończenie działania iteratora w zakresie za pomocą wartownika iteratora	116
Jak to zrobić?	117
Automatyczne sprawdzanie kodu iteratora	119
Jak to zrobić?	119
Jak to działa?	122
Co dalej?	123
Tworzenie własnego adaptera iteratora łączenia na zakładkę	123
Jak to zrobić?	125
Co dalej?	128

Rozdział 4. Wyrażenia lambda	131
Wprowadzenie	131
Definiowanie funkcji opartej na wyrażeniu lambda	133
Jak to zrobić?	133
Jak to działa?	136
Dodawanie polimorfizmu poprzez opakowanie wyrażenia lambda egzemplarzem <code>std::function</code>	138
Jak to zrobić?	138
Jak to działa?	140
Łączenie funkcji za pomocą konkatenacji	141
Jak to zrobić?	142
Jak to działa?	143
Tworzenie skomplikowanych predykatów z logiczną koniunkcją	144
Jak to zrobić?	145
Co dalej?	146
Wywoływanie wielu funkcji dla tych samych danych wejściowych	146
Jak to zrobić?	147
Jak to działa?	148
Implementowanie funkcji <code>transform_if()</code> za pomocą algorytmu <code>std::accumulate</code> i wyrażeń lambda	150
Jak to zrobić?	150
Jak to działa?	152
Generowanie w trakcie kompilacji iloczynu kartezjańskiego par dla dowolnych danych wejściowych	155
Jak to zrobić?	156
Jak to działa?	158
Rozdział 5. Podstawy algorytmów biblioteki STL	161
Wprowadzenie	161
Kopiowanie elementów między kontenerami	163
Jak to zrobić?	164
Jak to działa?	166
Sortowanie kontenera	167
Jak to zrobić?	167
Jak to działa?	170
Usuwanie określonych elementów z kontenera	171
Jak to zrobić?	171
Jak to działa?	173
Przekształcanie zawartości kontenera	174
Jak to zrobić?	174
Jak to działa?	176
Wyszukiwanie elementów w uporządkowanych i nieuporządkowanych wektorach	176
Jak to zrobić?	177
Jak to działa?	180
Ograniczanie za pomocą <code>std::clamp</code> wartości wektora do określonego zakresu liczbowego	182
Jak to zrobić?	182
Jak to działa?	185

Wyszukiwanie za pomocą std::search wzorca w ciągu tekstowym i wybór optymalnej implementacji	186
Jak to zrobić?	186
Jak to działa?	188
Próbkowanie ogromnego wektora	189
Jak to zrobić?	190
Jak to działa?	192
Generowanie permutacji sekwencji danych wejściowych	193
Jak to zrobić?	193
Jak to działa?	194
Implementowanie narzędzia łączenia słowników	195
Jak to zrobić?	195
Jak to działa?	197
Rozdział 6. Zaawansowane przykłady użycia algorytmów biblioteki STL	199
Wprowadzenie	200
Implementowanie klasy drzewa trie za pomocą algorytmów STL	201
Jak to zrobić?	201
Jak to działa?	204
Implementowanie za pomocą drzewa trie generatora sugestii danych wejściowych używanych podczas wyszukiwania	206
Jak to zrobić?	206
Jak to działa?	210
Co dalej?	210
Implementowanie wzoru przekształcenia Fouriera za pomocą algorytmów STL	211
Jak to zrobić?	212
Jak to działa?	217
Obliczanie błędu sumy dwóch wektorów	218
Jak to zrobić?	218
Jak to działa?	220
Implementowanie procedury generującej dane ASCII dla zbioru Mandelbrota	221
Jak to zrobić?	223
Jak to działa?	226
Opracowanie własnego algorytmu — podział danych	227
Jak to zrobić?	228
Jak to działa?	229
Co dalej?	230
Połączenie użytecznych algorytmów biblioteki STL — zbieranie danych	231
Jak to zrobić?	231
Jak to działa?	233
Usuwanie nadmiarowych białych znaków znajdujących się między słowami	235
Jak to zrobić?	235
Jak to działa?	237
Kompresja i dekompresja ciągów tekstowych	238
Jak to zrobić?	238
Jak to działa?	240
Co dalej?	241

Rozdział 7. Ciągi tekstowe, klasy strumieni i wyrażenia regularne	243
Wprowadzenie	244
Tworzenie, konkatenacja i przekształcanie ciągów tekstowych	245
Jak to zrobić?	246
Jak to działa?	248
Usuwanie białych znaków z początku i końca ciągu tekstowego	248
Jak to zrobić?	249
Jak to działa?	250
Komfortowe użycie klasy <code>std::string</code> bez kosztów związanych z tworzeniem obiektów <code>std::string</code>	251
Jak to zrobić?	252
Jak to działa?	254
Odczyt wartości z danych wejściowych dostarczonych przez użytkownika	254
Jak to zrobić?	255
Jak to działa?	257
Zliczanie wszystkich słów w pliku	258
Jak to zrobić?	258
Jak to działa?	260
Formatowanie danych wyjściowych za pomocą manipulatorów strumienia wejścia – wyjścia	260
Jak to zrobić?	261
Jak to działa?	264
Inicjalizacja skomplikowanych obiektów na podstawie pliku źródłowego	266
Jak to zrobić?	266
Jak to działa?	268
Wypełnianie kontenera za pomocą iteratorów <code>std::istream</code>	269
Jak to zrobić?	269
Jak to działa?	272
Proste wyświetlanie danych za pomocą iteratorów <code>std::ostream</code>	273
Jak to zrobić?	273
Jak to działa?	276
Przekierowywanie sekcji kodu do pliku danych wyjściowych	277
Jak to zrobić?	278
Jak to działa?	280
Tworzenie własnych klas ciągu tekstowego za pomocą dziedziczenia po klasie <code>std::char_traits</code>	281
Jak to zrobić?	282
Jak to działa?	286
Tokenizowanie danych wejściowych za pomocą biblioteki wyrażeń regularnych	287
Jak to zrobić?	287
Jak to działa?	289
Wygodne formatowanie liczb w locie w zależności od kontekstu	291
Jak to zrobić?	291
Przechwytywanie na podstawie błędów <code>std::istream</code> wyjątków możliwych do odczytania	293
Jak to zrobić?	294
Jak to działa?	296

Rozdział 8. Klasy narzędziowe	297
Wprowadzenie	298
Konwertowanie między różnymi jednostkami czasu za pomocą std::ratio	298
Jak to zrobić?	299
Jak to działa?	301
Co dalej?	303
Konwertowanie między bezwzględnymi i względnymi wartościami czasu za pomocą std::chrono	304
Jak to zrobić?	304
Jak to działa?	306
Bezpieczne sygnalizowanie awarii za pomocą typu std::optional	307
Jak to zrobić?	307
Jak to działa?	310
Użycie funkcji wraz z krotkami	311
Jak to zrobić?	311
Jak to działa?	313
Szybkie opracowywanie struktur danych za pomocą std::tuple	313
Jak to zrobić?	314
Jak to działa?	318
Zastąpienie void* przez std::any dla zwiększenia bezpieczeństwa typu	320
Jak to zrobić?	321
Jak to działa?	323
Przechowywanie różnych typów za pomocą std::variant	323
Jak to zrobić?	324
Jak to działa?	327
Automatyczna obsługa zasobów za pomocą std::unique_ptr	329
Jak to zrobić?	329
Jak to działa?	332
Automatyczna obsługa współdzielonej pamięci na sterce za pomocą std::shared_ptr	333
Jak to zrobić?	333
Jak to działa?	336
Co dalej?	337
Praca ze słabymi wskaźnikami do współdzielonych obiektów	338
Jak to zrobić?	339
Jak to działa?	341
Uproszczenie obsługi zasobów przestarzałych API za pomocą sprytnych wskaźników	342
Jak to zrobić?	343
Jak to działa?	344
Współdzielenie różnych wartości składowych tego samego obiektu	345
Jak to zrobić?	346
Jak to działa?	347
Generowanie liczb losowych i wybór odpowiedniego silnika do generowania tego rodzaju liczb	348
Jak to zrobić?	349
Jak to działa?	353
Generowanie liczb losowych i umożliwienie bibliotece STL określenia szczegółów rozkładu	354
Jak to zrobić?	354
Jak to działa?	359

Rozdział 9. Programowanie równoległe i współbieżność **361**

Wprowadzenie	362
Automatyczne stosowanie programowania równoległego w kodzie utworzonego za pomocą standardowych algorytmów	363
Jak to zrobić?	363
Jak to działa?	365
Uśpienie programu na podany okres czasu	369
Jak to zrobić?	369
Jak to działa?	370
Uruchamianie i zatrzymywanie wątków	371
Jak to zrobić?	371
Jak to działa?	373
Przeprowadzanie bezpiecznego pod względem wyjątków nakładania blokady współdzielonej za pomocą std::unique_lock i std::shared_lock	375
Jak to zrobić?	375
Jak to działa?	378
Zapobieganie zakleszczeniom dzięki stosowaniu algorytmu std::scoped_lock	381
Jak to zrobić?	382
Jak to działa?	384
Synchronizacja jednoczesnego użycia algorytmu std::cout	384
Jak to zrobić?	385
Jak to działa?	386
Bezpieczne odkładanie inicjalizacji za pomocą std::call_once	388
Jak to zrobić?	389
Jak to działa?	390
Przesunięcie zadania do wykonywania w tle za pomocą std::async	390
Jak to zrobić?	391
Jak to działa?	393
Co dalej?	395
Implementacja wzorca producent – konsument za pomocą std::condition_variable	395
Jak to zrobić?	396
Jak to działa?	398
Implementacja wzorca producent – konsument za pomocą std::condition_variable	400
Jak to zrobić?	400
Jak to działa?	404
Równoległe generowanie za pomocą std::async danych ASCII dla zbioru Mandelbrota	406
Jak to zrobić?	406
Jak to działa?	409
Implementacja za pomocą std::future niewielkiej biblioteki automatycznej programowania równoległego	410
Jak to zrobić?	411
Jak to działa?	415

Rozdział 10. System plików	419
Wprowadzenie	419
Implementowanie programu przeprowadzającego normalizację ścieżki dostępu	420
Jak to zrobić?	420
Jak to działa?	422
Co dalej?	422
Pobieranie kanonicznej ścieżki dostępu na podstawie względnej ścieżki dostępu	423
Jak to zrobić?	423
Jak to działa?	426
Wyświetlanie wszystkich plików znajdujących się w danym katalogu	426
Jak to zrobić?	427
Jak to działa?	430
Implementowanie programu wyszukującego dane i działającego podobnie jak narzędzie grep	431
Jak to zrobić?	431
Jak to działa?	433
Co dalej?	434
Implementowanie programu automatycznie zmieniającego nazwy plików	434
Jak to zrobić?	435
Implementowanie programu obliczającego wielkość katalogu	437
Jak to zrobić?	437
Jak to działa?	439
Obliczanie danych statystycznych dotyczących typów plików	440
Jak to zrobić?	440
Implementowanie narzędzia zmniejszającego wielkość katalogu poprzez zastąpienie powielonych plików dołączeniami symbolicznymi	442
Jak to zrobić?	443
Jak to działa?	445
Co dalej?	446
Skorowidz	447