

Podziękowania (13)

Przedmowa (15)

1 O książce (17)

- 1.1. Dlaczego powstała ta książka? (17)
- 1.2. Co należy wiedzieć przed przystąpieniem do lektury tej książki? (18)
- 1.3. Styl i struktura książki (18)
- 1.4. Jak czytać tę książkę? (21)
- 1.5. Stan obecny (21)
- 1.6. Przykładowy kod i dodatkowe informacje (22)

2 Wprowadzenie do języka C++ i biblioteki standardowej (23)

- 2.1. Historia (23)
- 2.2. Nowe możliwości języka (25)
 - o 2.2.1. Wzorce (25)
 - o 2.2.2. Jawna inicjalizacja typów podstawowych (30)
 - o 2.2.3. Obsługa wyjątków (30)
 - o 2.2.4. Przestrzenie nazw (32)
 - o 2.2.5. Typ bool (33)
 - o 2.2.6. Słowo kluczowe explicit (34)
 - o 2.2.7. Nowe operatory konwersji typu (35)
 - o 2.2.8. Inicjalizacja stałych składowych statycznych (36)
 - o 2.2.9. Definicja funkcji main() (36)
- 2.3. Złożoność algorytmów a notacja O (37)

3 Pojęcia ogólne (39)

- 3.1. Przestrzeń nazw std (39)
- 3.2. Pliki nagłówkowe (40)
- 3.3. Obsługa błędów i wyjątków (42)
 - o 3.3.1. Standardowe klasy wyjątków (42)
 - o 3.3.2. Składowe klas wyjątków (45)
 - o 3.3.3. Zgłaszanie wyjątków standardowych (46)
 - o 3.3.4. Tworzenie klas pochodnych standardowych klas wyjątków (46)
- 3.4. Alokatory (48)

4 Narzędzia (49)

- 4.1. Pary (49)
 - o 4.1.1. Wygodna funkcja make_pair() (51)
 - o 4.1.2. Przykłady użycia par (53)
- 4.2. Klasa auto_ptr (53)
 - o 4.2.1. Motywacja klasy auto_ptr (53)
 - o 4.2.2. Przenoszenie własności w przypadku klasy auto_ptr (55)
 - o 4.2.3. Wskaźniki auto_ptr jako składowe (59)
 - o 4.2.4. Niewłaściwe użycie wskaźników auto_ptr (61)
 - o 4.2.5. Przykłady zastosowania typu auto_ptr (62)
 - o 4.2.6. Klasa auto_ptr w szczegółach (64)
- 4.3. Ograniczenia liczbowe (71)

- 4.4. Funkcje pomocnicze (77)
 - o 4.4.1. Obliczanie wartości minimalnej oraz maksymalnej (77)
 - o 4.4.2. Zamiana dwóch wartości (78)
- 4.5. Dodatkowe operatory porównania (79)
- 4.6. Pliki nagłówkowe <cstdlib> oraz <cstdliblib> (81)
 - o 4.6.1. Definicje w pliku <cstdlib> (81)
 - o 4.6.2. Definicje w pliku <cstdliblib> (82)

5 Standardowa biblioteka wzorców (STL) (83)

- 5.1. Składniki biblioteki STL (83)
- 5.2. Kontenery (85)
 - o 5.2.1. Kontenery sekwencyjne (86)
 - o 5.2.2. Kontenery asocjacyjne (91)
 - o 5.2.3. Adaptatory kontenerów (92)
- 5.3. Iteratory (93)
 - o 5.3.1. Przykłady użycia kontenerów asocjacyjnych (96)
 - o 5.3.2. Kategorie iteratorów (102)
- 5.4. Algorytmy (103)
 - o 5.4.1. Zakresy (105)
 - o 5.4.2. Obsługa wielu zakresów (110)
- 5.5. Adaptatory iteratorów (112)
 - o 5.5.1. Iteratory wstawiające (112)
 - o 5.5.2. Iteratory strumieniowe (114)
 - o 5.5.3. Iteratory odwrotne (116)
- 5.6. Algorytmy modyfikujące (118)
 - o 5.6.1. Usuwanie elementów (118)
 - o 5.6.2. Algorytmy modyfikujące a kontenery asocjacyjne (121)
 - o 5.6.3. Algorytmy a funkcje składowe (123)
- 5.7. Funkcje ogólne definiowane przez użytkownika (124)
- 5.8. Funkcje jako argumenty algorytmów (125)
 - o 5.8.1. Przykłady użycia funkcji jako argumentów algorytmów (125)
 - o 5.8.2. Predykaty (126)
- 5.9. Obiekty funkcyjne (129)
 - o 5.9.1. Czym są obiekty funkcyjne? (129)
 - o 5.9.2. Predefiniowane obiekty funkcyjne (135)
- 5.10. Elementy kontenerów (138)
 - o 5.10.1. Wymagania wobec elementów kontenerów (138)
 - o 5.10.2. Semantyka wartości a semantyka referencji (139)
- 5.11. Obsługa błędów i wyjątków wewnątrz biblioteki STL (140)
 - o 5.11.1. Obsługa błędów (141)
 - o 5.11.2. Obsługa wyjątków (143)
- 5.12. Rozbudowa biblioteki STL (145)

6 Kontenery STL (147)

- 6.1. Wspólne cechy i operacje kontenerów (148)
 - o 6.1.1. Wspólne cechy kontenerów (148)
 - o 6.1.2. Wspólne operacje kontenerów (148)
- 6.2. Wektory (151)

- o 6.2.1. Możliwości wektorów (152)
 - o 6.2.2. Operacje na wektorach (154)
 - o 6.2.3. Używanie wektorów jako zwykłych tablic (158)
 - o 6.2.4. Obsługa wyjątków (159)
 - o 6.2.5. Przykłady użycia wektorów (160)
 - o 6.2.6. Klasa `vector<bool>` (161)
- 6.3. Kolejki o dwóch końcach (163)
 - o 6.3.1. Możliwości kolejek `deque` (164)
 - o 6.3.2. Operacje na kolejkach `deque` (165)
 - o 6.3.3. Obsługa wyjątków (166)
 - o 6.3.4. Przykłady użycia kolejek `deque` (167)
- 6.4. Listy (168)
 - o 6.4.1. Możliwości list (169)
 - o 6.4.2. Operacje na listach (170)
 - o 6.4.3. Obsługa wyjątków (174)
 - o 6.4.4. Przykłady użycia list (175)
- 6.5. Zbiory i wielozbiory (176)
 - o 6.5.1. Możliwości zbiorów i wielozbiorów (178)
 - o 6.5.2. Operacje na zbiorach i wielozbiorach (179)
 - o 6.5.3. Obsługa wyjątków (187)
 - o 6.5.4. Przykłady użycia zbiorów i wielozbiorów (187)
 - o 6.5.5. Przykład określania kryterium sortowania podczas wykonywania (191)
- 6.6. Mapy oraz multimapy (193)
 - o 6.6.1. Możliwości map oraz multimap (194)
 - o 6.6.2. Operacje na mapach oraz multimapach (195)
 - o 6.6.3. Zastosowanie map jako tablic asocjacyjnych (204)
 - o 6.6.4. Obsługa wyjątków (206)
 - o 6.6.5. Przykłady użycia map i multimap (206)
 - o 6.6.6. Przykład z mapami, łańcuchami oraz definiowaniem kryterium sortowania podczas wykonywania (210)
- 6.7. Inne kontenery STL (212)
 - o 6.7.1. Łańcuchy jako kontenery STL (213)
 - o 6.7.2. Zwykłe tablice jako kontenery STL (214)
 - o 6.7.3. Tablice mieszające (216)
- 6.8. Implementacja semantyki referencji (217)
- 6.9. Kiedy stosować poszczególne kontenery (220)
- 6.10. Typy kontenerowe i ich składowe w szczegółach (223)
 - o 6.10.1. Definicje typów (224)
 - o 6.10.2. Operacje tworzenia, kopiowania i niszczenia (225)
 - o 6.10.3. Operacje niemodyfikujące (227)
 - o 6.10.4. Operacje przypisania (230)
 - o 6.10.5. Bezpośredni dostęp do elementów (231)
 - o 6.10.6. Operacje generujące iteratory (233)
 - o 6.10.7. Wstawianie i usuwanie elementów (234)
 - o 6.10.8. Specjalne funkcje składowe list (239)
 - o 6.10.9. Obsługa alokatorów (241)
 - o 6.10.10. Omówienie obsługi wyjątków w kontenerach STL (242)

7 Iteratory STL (245)

- 7.1. Pliki nagłówkowe iteratorów (245)
- 7.2. Kategorie iteratorów (245)
 - o 7.2.1. Iteratory wejściowe (246)
 - o 7.2.2. Iteratory wyjściowe (247)
 - o 7.2.3. Iteratory postępujące (248)
 - o 7.2.4. Iteratory dwukierunkowe (249)
 - o 7.2.5. Iteratory dostępu swobodnego (249)
 - o 7.2.6. Problem z inkrementacją i dekrementacją iteratorów wektorów (252)
- 7.3. Pomocnicze funkcje iteratorów (253)
 - o 7.3.1. Przesuwanie iteratorów za pomocą funkcji advance() (253)
 - o 7.3.2. Obliczanie odległości pomiędzy iteratorami za pomocą funkcji distance() (254)
 - o 7.3.3. Zamiana wartości iteratorów za pomocą funkcji iter_swap() (256)
- 7.4. Adaptatory iteratorów (257)
 - o 7.4.1. Iteratory odwrotne (257)
 - o 7.4.2. Iteratory wstawiające (262)
 - o 7.4.3. Iteratory strumieniowe (268)
- 7.5. Cechy iteratorów (273)
 - o 7.5.1. Definiowanie funkcji ogólnych dla iteratorów (275)
 - o 7.5.2. Iteratory definiowane przez użytkownika (277)

8 Obiekty funkcyjne STL (281)

- 8.1. Pojęcie obiektów funkcyjnych (281)
 - o 8.1.1. Obiekty funkcyjne jako kryteria sortowania (282)
 - o 8.1.2. Obiekty funkcyjne ze stanem wewnętrznym (283)
 - o 8.1.3. Wartość zwracana algorytmu for_each() (287)
 - o 8.1.4. Predykaty a obiekty funkcyjne (288)
- 8.2. Predefiniowane obiekty funkcyjne (291)
 - o 8.2.1. Adaptatory funkcji (291)
 - o 8.2.2. Adaptatory funkcji składowych (293)
 - o 8.2.3. Adaptatory zwykłych funkcji (295)
 - o 8.2.4. Obiekty funkcyjne definiowane przez użytkownika dostosowane do adaptatorów funkcji (296)
- 8.3. Dodatkowe złożeniowe obiekty funkcyjne (298)
 - o 8.3.1. Jednoargumentowe złożeniowe adaptatory obiektów funkcyjnych (299)
 - o 8.3.2. Dwuargumentowe złożeniowe adaptatory obiektów funkcyjnych (302)

9 Algorytmy STL (305)

- 9.1. Pliki nagłówkowe algorytmów (305)
- 9.2. Przegląd algorytmów (306)
 - o 9.2.1. Krótkie wprowadzenie (306)
 - o 9.2.2. Klasyfikacja algorytmów (307)
- 9.3. Funkcje pomocnicze (316)
- 9.4. Algorytm for_each() (318)
- 9.5. Algorytmy niemodyfikujące (320)
 - o 9.5.1. Zliczanie elementów (321)
 - o 9.5.2. Wartość minimalna i maksymalna (322)
 - o 9.5.3. Wyszukiwanie elementów (324)

- o 9.5.4. Porównywanie zakresów (335)
- 9.6. Algorytmy modyfikujące (340)
 - o 9.6.1. Kopiowanie elementów (341)
 - o 9.6.2. Przekształcenia i kombinacje elementów (344)
 - o 9.6.3. Wymienianie elementów (347)
 - o 9.6.4. Przypisywanie nowych wartości (348)
 - o 9.6.5. Zastępowanie elementów (350)
- 9.7. Algorytmy usuwające (353)
 - o 9.7.1. Usuwanie określonych wartości (353)
 - o 9.7.2. Usuwanie powtórzeń (356)
- 9.8. Algorytmy mutujące (360)
 - o 9.8.1. Odwracanie kolejności elementów (360)
 - o 9.8.2. Przesunięcia cykliczne elementów (361)
 - o 9.8.3. Permutacje elementów (363)
 - o 9.8.4. Tasowanie elementów (365)
 - o 9.8.5. Przenoszenie elementów na początek (367)
- 9.9. Algorytmy sortujące (368)
 - o 9.9.1. Sortowanie wszystkich elementów (369)
 - o 9.9.2. Sortowanie częściowe (371)
 - o 9.9.3. Sortowanie według n-tego elementu (374)
 - o 9.9.4. Algorytmy stogowe (375)
- 9.10. Algorytmy przeznaczone dla zakresów posortowanych (378)
 - o 9.10.1. Wyszukiwanie elementów (379)
 - o 9.10.2. Scalanie elementów (385)
- 9.11. Algorytmy numeryczne (393)
 - o 9.11.1. Obliczanie wartości (393)
 - o 9.11.2. Konwersje wartości względnych i bezwzględnych (397)

10 Kontenery specjalne (401)

- 10.1. Stosy (401)
 - o 10.1.1. Interfejs (403)
 - o 10.1.2. Przykład użycia stosów (403)
 - o 10.1.3. Klasa `stack<>` w szczegółach (404)
 - o 10.1.4. Klasa stosu definiowanego przez użytkownika (406)
- 10.2. Kolejki (408)
 - o 10.2.1. Interfejs (410)
 - o 10.2.2. Przykład użycia kolejek (410)
 - o 10.2.3. Klasa `queue<>` w szczegółach (411)
 - o 10.2.4. Klasa kolejki definiowanej przez użytkownika (413)
- 10.3. Kolejki priorytetowe (416)
 - o 10.3.1. Interfejs (417)
 - o 10.3.2. Przykład użycia kolejek priorytetowych (418)
 - o 10.3.3. Klasa `priority_queue<>` w szczegółach (418)
- 10.4. Kontener bitset (421)
 - o 10.4.1. Przykłady użycia kontenerów bitset (422)
 - o 10.4.2. Szczegółowy opis klasy bitset (424)

11 Łańcuchy (431)

- 11.1. Motywacja (431)
 - o 11.1.1. Przykład pierwszy: Pobieranie tymczasowej nazwy pliku (432)
 - o 11.1.2. Przykład drugi: Pobieranie słów i wypisywanie ich w odwrotnej kolejności (437)
- 11.2. Opis klas reprezentujących łańcuchy znakowe (440)
 - o 11.2.1. Typy łańcuchów znakowych (440)
 - o 11.2.2. Przegląd funkcji składowych (441)
 - o 11.2.3. Konstruktory oraz destruktory (443)
 - o 11.2.4. Łańcuchy znakowe zwykłe oraz języka C (444)
 - o 11.2.5. Rozmiar oraz pojemność (446)
 - o 11.2.6. Dostęp do elementów (448)
 - o 11.2.7. Porównania (449)
 - o 11.2.8. Modyfikatory (450)
 - o 11.2.9. Konkatenacja łańcuchów znakowych oraz ich fragmentów (453)
 - o 11.2.10. Operatory wejścia-wyjścia (454)
 - o 11.2.11. Poszukiwanie oraz odnajdywanie łańcuchów znakowych (455)
 - o 11.2.12. Wartość npos (457)
 - o 11.2.13. Obsługa iteratorów w przypadku łańcuchów znakowych (458)
 - o 11.2.14. Obsługa standardów narodowych (464)
 - o 11.2.15. Wydajność (466)
 - o 11.2.16. Łańcuchy znakowe a wektory (466)
- 11.3. Klasa string w szczegółach (467)
 - o 11.3.1. Definicje typu oraz wartości statyczne (467)
 - o 11.3.2. Funkcje składowe służące do tworzenia, kopiowania oraz usuwania łańcuchów znakowych (469)
 - o 11.3.3. Funkcje dotyczące rozmiaru oraz pojemności (470)
 - o 11.3.4. Porównania (471)
 - o 11.3.5. Dostęp do znaków (473)
 - o 11.3.6. Tworzenie łańcuchów znakowych języka C oraz tablic znaków (474)
 - o 11.3.7. Funkcje do modyfikacji zawartości łańcuchów znakowych (475)
 - o 11.3.8. Poszukiwanie oraz odnajdywanie (482)
 - o 11.3.9. Łączenie łańcuchów znakowych (485)
 - o 11.3.10. Funkcje wejścia-wyjścia (486)
 - o 11.3.11. Funkcje tworzące iteratory (487)
 - o 11.3.12. Obsługa alokatorów (488)

12 Kontenery numeryczne (491)

- 12.1. Liczby zespolone (491)
 - o 12.1.1. Przykład wykorzystania klasy reprezentującej liczby zespolone (492)
 - o 12.1.2. Funkcje operujące na liczbach zespolonych (494)
 - o 12.1.3. Klasa `complex<>` w szczegółach (501)
- 12.2. Klasa `valarray` (506)
 - o 12.2.1. Poznawanie klas `valarray` (507)
 - o 12.2.2. Podzbiory wartości umieszczonych w tablicy `valarray` (512)
 - o 12.2.3. Klasa `valarray` w szczegółach (525)
 - o 12.2.4. Klasy podzbiorów tablic typu `valarray` w szczegółach (530)
- 12.3. Globalne funkcje numeryczne (535)

13 Obsługa wejścia-wyjścia z wykorzystaniem klas strumieniowych (537)

- 13.1. Podstawy strumieni wejścia-wyjścia (538)
 - o 13.1.1. Obiekty strumieni (538)
 - o 13.1.2. Klasy strumieni (539)
 - o 13.1.3. Globalne obiekty strumieni (539)
 - o 13.1.4. Operatory strumieniowe (540)
 - o 13.1.5. Manipulatory (540)
 - o 13.1.6. Prosty przykład (541)
- 13.2. Podstawowe obiekty oraz klasy strumieniowe (542)
 - o 13.2.1. Klasy oraz hierarchia klas (542)
 - o 13.2.2. Globalne obiekty strumieni (545)
 - o 13.2.3. Pliki nagłówkowe (546)
- 13.3. Standardowe operatory strumieniowe << oraz >> (547)
 - o 13.3.1. Operator wyjściowy << (547)
 - o 13.3.2. Operator wejściowy >> (549)
 - o 13.3.3. Operacje wejścia-wyjścia dla specjalnych typów (549)
- 13.4. Stany strumieni (552)
 - o 13.4.1. Stałe służące do określania stanów strumieni (552)
 - o 13.4.2. Funkcje składowe operujące na stanie strumieni (553)
 - o 13.4.3. Warunki wykorzystujące stan strumienia oraz wartości logiczne (555)
 - o 13.4.4. Stan strumienia i wyjątki (557)
- 13.5. Standardowe funkcje wejścia-wyjścia (561)
 - o 13.5.1. Funkcje składowe służące do pobierania danych (562)
 - o 13.5.2. Funkcje składowe służące do wysyłania danych (565)
 - o 13.5.3. Przykład użycia (566)
- 13.6. Manipulatory (567)
 - o 13.6.1. Sposób działania manipulatorów (568)
 - o 13.6.2. Manipulatory definiowane przez użytkownika (569)
- 13.7. Formatowanie (570)
 - o 13.7.1. Znaczniki formatu (570)
 - o 13.7.2. Format wartości logicznych (572)
 - o 13.7.3. Szerokość pola znak wypełnienia oraz wyrównanie (573)
 - o 13.7.4. Znak wartości dodatnich oraz duże litery (576)
 - o 13.7.5. Podstawa numeryczna (576)
 - o 13.7.6. Notacja zapisu liczb zmiennoprzecinkowych (579)
 - o 13.7.7. Ogólne definicje formatujące (581)
- 13.8. Umiędzynarodawianie (582)
- 13.9. Dostęp do plików (583)
 - o 13.9.1. Znaczniki pliku (586)
 - o 13.9.2. Dostęp swobodny (589)
 - o 13.9.3. Deskryptory plików (592)
- 13.10. Łączenie strumieni wejściowych oraz wyjściowych (592)
 - o 13.10.1. Luźne powiązanie przy użyciu tie() (593)
 - o 13.10.2. Ścisłe powiązanie przy użyciu buforów strumieni (594)
 - o 13.10.3. Przekierowywanie strumieni standardowych (596)
 - o 13.10.4. Strumień służący do odczytu oraz zapisu (597)
- 13.11. Klasy strumieni dla łańcuchów znakowych (599)
 - o 13.11.1. Klasy strumieni dla łańcuchów znakowych (600)
 - o 13.11.2. Klasy strumieni dla wartości typu char* (603)
- 13.12. Operatory wejścia-wyjścia dla typów zdefiniowanych przez użytkownika (605)
 - o 13.12.1. Implementacja operatorów wyjściowych (605)

- o 13.12.2. Implementacja operatorów wejściowych (607)
- o 13.12.3. Operacje wejścia-wyjścia przy użyciu funkcji pomocniczych (609)
- o 13.12.4. Operatory definiowane przez użytkownika używające funkcji nieformatujących (610)
- o 13.12.5. Znaczniki formatu definiowane przez użytkownika (612)
- o 13.12.6. Konwencje operatorów wejścia-wyjścia definiowanych przez użytkownika (615)
- 13.13. Klasy bufora strumienia (616)
 - o 13.13.1. Spojrzenie użytkownika na bufor strumienia (616)
 - o 13.13.2. Iteratory wykorzystywane z buforem strumienia (618)
 - o 13.13.3. Bufory strumienia definiowane przez użytkownika (621)
- 13.14. Kwestie wydajności (632)
 - o 13.14.1. Synchronizacja ze standardowymi strumieniami używanymi w języku C (633)
 - o 13.14.2. Buforowanie w buforach strumienia (633)
 - o 13.14.3. Bezpośrednie wykorzystanie buforów strumienia (634)

14 Umieźdzynarodowienie (637)

- 14.1. Różne standardy kodowania znaków (638)
 - o 14.1.1. Reprezentacja za pomocą zestawu znaków szerokiego zakresu lub reprezentacja wielobajtowa (639)
 - o 14.1.2. Cechy znaków (640)
 - o 14.1.3. Umieźdzynarodawianie specjalnych znaków (643)
- 14.2. Pojęcie obiektów ustawień lokalnych (644)
 - o 14.2.1. Wykorzystywanie ustawień lokalnych (646)
 - o 14.2.2. Aspekty ustawień lokalnych (650)
- 14.3. Klasa locale w szczegółach (653)
- 14.4. Klasa facet w szczegółach (656)
 - o 14.4.1. Formatowanie wartości numerycznych (657)
 - o 14.4.2. Formatowanie czasu oraz daty (661)
 - o 14.4.3. Formatowanie wartości walutowych (664)
 - o 14.4.4. Klasyfikacja oraz konwersja znaków (669)
 - o 14.4.5. Sortowanie łańcuchów znakowych (677)
 - o 14.4.6. Lokalizacja komunikatów (679)

15 Alokatory (681)

- 15.1. Wykorzystywanie alokatorów przez programistów aplikacji (681)
- 15.2. Wykorzystywanie alokatorów przez programistów bibliotek (682)
- 15.3. Alokator domyślny (685)
- 15.4. Alokator zdefiniowany przez użytkownika (687)
- 15.5. Alokatory w szczegółach (689)
 - o 15.5.1. Definicje typu (689)
 - o 15.5.2. Funkcje składowe (691)
- 15.6. Funkcje stosowane w przypadku niezainicjalizowanej pamięci (692)

A Bibliografia (695)

B Skorowidz (697)