

Przedmowa (11)

Wprowadzenie (15)

- W.1. Od C do C++ (15)
- W.2. Jak używać C++ do tworzenia dużych projektów (16)
 - o W.2.1. Zależności cykliczne (16)
 - o W.2.2. Nadmierne zależności na etapie konsolidacji (18)
 - o W.2.3. Nadmierne zależności na etapie kompilacji (20)
 - o W.2.4. Globalna przestrzeń nazw (22)
 - o W.2.5. Projekt logiczny i fizyczny (23)
- W.3. Ponowne użycie (25)
- W.4. Jakość (25)
 - o W.4.1. Kontrola jakości (27)
 - o W.4.2. Zapewnienie jakości (27)
- W.5. Narzędzia (27)
- W.6. Podsumowanie (28)

Część I Podstawy (29)

Rozdział 1. Wstęp (31)

- 1.1. Programy wieloplikowe w C++ (31)
 - o 1.1.1. Deklaracja a definicja (31)
 - o 1.1.2. Konsolidacja (łączenie) wewnętrzna a zewnętrzna (33)
 - o 1.1.3. Pliki nagłówkowe (.h) (36)
 - o 1.1.4. Pliki implementacji (.c) (37)
- 1.2. Deklaracje typedef (38)
- 1.3. Instrukcje sprawdzające (39)
- 1.4. Kilka słów na temat stylu (40)
 - o 1.4.1. Identyfikatory (41)
 - 1.4.1.1. Nazwy typów (41)
 - 1.4.1.2. Nazwy identyfikatorów składające się z wielu słów (42)
 - 1.4.1.3. Nazwy składowych klas (42)
 - o 1.4.2. Kolejność ułożenia składowych w klasie (44)
- 1.5. Iteratory (46)
- 1.6. Logiczna notacja projektu (52)
 - o 1.6.1. Relacja Jest (53)
 - o 1.6.2. Relacja Używa-W-Interfejsie (54)
 - o 1.6.3. Relacja Używa-W-Implementacji (56)
 - 1.6.3.1. Relacja Używa (58)
 - 1.6.3.2. Relacje Ma i Trzyma (58)
 - 1.6.3.3. Relacja Był (59)
- 1.7. Dziedziczenie i warstwy (60)
- 1.8. Minimalizacja (61)
- 1.9. Podsumowanie (62)

Rozdział 2. Podstawowe reguły (65)

- 2.1. Przegląd (65)
- 2.2. Dostęp do pól klasy (66)
- 2.3. Globalna przestrzeń nazw (70)

- o 2.3.1. Dane globalne (70)
- o 2.3.2. Wolne funkcje (72)
- o 2.3.3. Wyliczenia, stałe i deklaracje typedef (73)
- o 2.3.4. Makra preprocesora (74)
- o 2.3.5. Nazwy w plikach nagłówkowych (75)
- 2.4. Kontrola dołączeń (77)
- 2.5. Dodatkowa kontrola dołączeń (79)
- 2.6. Dokumentacja (84)
- 2.7. Sposoby nazywania identyfikatorów (86)
- 2.8. Podsumowanie (87)

Część II Projekt fizyczny (91)

Rozdział 3. Komponenty (93)

- 3.1. Komponenty a klasy (93)
- 3.2. Reguły projektów fizycznych (100)
- 3.3. Relacja ZależyOd (108)
- 3.4. Zależności implikowane (112)
- 3.5. Wydobywanie rzeczywistych zależności (117)
- 3.6. Przyjaźń (119)
 - o 3.6.1. Przyjaźń na odległość i zależności implikowane (122)
 - o 3.6.2. Przyjaźń i oszustwo (124)
- 3.7. Podsumowanie (126)

Rozdział 4. Hierarchia fizyczna (129)

- 4.1. Metafora dla testowania oprogramowania (129)
- 4.2. Złożony podsystem (130)
- 4.3. Problemy z testowaniem "dobrych" interfejsów (134)
- 4.4. Projektowanie zorientowane na testowalność (136)
- 4.5. Testowanie pojedynczych modułów (138)
- 4.6. Acykliczne zależności fizyczne (140)
- 4.7. Numery poziomów (142)
 - o 4.7.1. Źródła numeracji poziomów (142)
 - o 4.7.2. Używanie numerów poziomów w oprogramowaniu (144)
- 4.8. Testowanie hierarchiczne i przyrostowe (147)
- 4.9. Testowanie złożonych podsystemów (153)
- 4.10. Testowalność kontra testowanie (154)
- 4.11. Cykliczne zależności fizyczne (155)
- 4.12. Suma zależności komponentów (156)
- 4.13. Jakość projektu fizycznego (161)
- 4.14. Podsumowanie (167)

Rozdział 5. Podział na poziomy (169)

- 5.1. Niektóre przyczyny cyklicznych zależności fizycznych (169)
 - o 5.1.1. Rozszerzenie (170)
 - o 5.1.2. Wygoda (172)
 - o 5.1.3. Wewnętrzna zależność (176)
- 5.2. Wyniesienie (178)

- 5.3. Obniżenie (187)
- 5.4. Nieprzezroczyste wskaźniki (199)
- 5.5. Głupie dane (206)
- 5.6. Redundancja (216)
- 5.7. Wywołania zwrotne (219)
- 5.8. Klasa-menedżer (231)
- 5.9. Faktoring (235)
- 5.10. Wynoszenie enkapsulacji (249)
- 5.11. Podsumowanie (260)

Rozdział 6. Izolacja (261)

- 6.1. Od enkapsulacji do izolacji (262)
 - o 6.1.1. Koszty powiązań na etapie kompilacji (266)
- 6.2. Konstrukcje w języku C++ a zależności na etapie kompilacji (267)
 - o 6.2.1. Dziedziczenie (Jest) a zależności na etapie kompilacji (268)
 - o 6.2.2. Podział na warstwy (Ma/Trzyma) a zależności na etapie kompilacji (269)
 - o 6.2.3. Funkcje inline a zależności na etapie kompilacji (270)
 - o 6.2.4. Składowe prywatne a zależności na etapie kompilacji (271)
 - o 6.2.5. Składowe chronione a zależności na etapie kompilacji (273)
 - o 6.2.6. Funkcje składowe generowane przez kompilator a zależności na etapie kompilacji (274)
 - o 6.2.7. Dyrektywy include a zależności na etapie kompilacji (275)
 - o 6.2.8. Argumenty domyślne a zależności na etapie kompilacji (277)
 - o 6.2.9. Wyliczenia a zależności na etapie kompilacji (277)
- 6.3. Techniki częściowej izolacji (279)
 - o 6.3.1. Rezygnacja z dziedziczenia prywatnego (279)
 - o 6.3.2. Usuwanie osadzonych danych składowych (281)
 - o 6.3.3. Usuwanie prywatnych funkcji składowych (282)
 - o 6.3.4. Usuwanie składowych chronionych (290)
 - o 6.3.5. Usuwanie prywatnych danych składowych (299)
 - o 6.3.6. Usuwanie funkcji generowanych przez kompilator (302)
 - o 6.3.7. Usuwanie dyrektyw include (302)
 - o 6.3.8. Usuwanie argumentów domyślnych (303)
 - o 6.3.9. Usuwanie wyliczeń (305)
- 6.4. Techniki całkowitej izolacji (307)
 - o 6.4.1. Klasa protokołu (308)
 - o 6.4.2. W pełni izolująca klasa konkretna (317)
 - o 6.4.3. Izolujące komponenty otaczające (322)
 - 6.4.3.1. Pojedyncze komponenty otaczające (323)
 - 6.4.3.2. Wielokomponentowe warstwy otaczające (331)
- 6.5. Interfejs proceduralny (338)
 - o 6.5.1. Architektura interfejsu proceduralnego (339)
 - o 6.5.2. Tworzenie i usuwanie nieprzezroczystych obiektów (341)
 - o 6.5.3. Uchwyty (342)
 - o 6.5.4. Uzyskiwanie dostępu do nieprzezroczystych obiektów i manipulowanie nimi (346)
 - o 6.5.5. Dziedziczenie a nieprzezroczyste obiekty (351)
- 6.6. Izolować czy nie izolować (354)

- o 6.6.1. Koszt izolacji (354)
- o 6.6.2. Kiedy nie należy izolować (356)
- o 6.6.3. Jak izolować (360)
- o 6.6.4. Do jakiego stopnia należy izolować (366)
- 6.7. Podsumowanie (373)

Rozdział 7. Pakiety (377)

- 7.1. Od komponentów do pakietów (378)
- 7.2. Zarejestrowane prefiksy pakietów (385)
 - o 7.2.1. Potrzeba stosowania prefiksów (385)
 - o 7.2.2. Przestrzenie nazw (387)
 - o 7.2.3. Zachowanie integralności pakietu (391)
- 7.3. Podział pakietów na poziomy (393)
 - o 7.3.1. Znaczenie podziału pakietów na poziomy (393)
 - o 7.3.2. Techniki podziału pakietów na poziomy (394)
 - o 7.3.3. Podział systemu (396)
 - o 7.3.4. Wytwarzanie oprogramowania w wielu ośrodkach (398)
- 7.4. Izolacja pakietów (399)
- 7.5. Grupy pakietów (402)
- 7.6. Proces wydawania oprogramowania (406)
 - o 7.6.1. Struktura wydania (408)
 - o 7.6.2. Łaty (413)
- 7.7. Program main (415)
- 7.8. Faza startu (421)
 - o 7.8.1. Strategie inicjalizacji (423)
 - 7.8.1.1. Technika "przebudzenia" w stanie zainicjowania (424)
 - 7.8.1.2. Technika jawnego wywoływania funkcji init (424)
 - 7.8.1.3. Technika wykorzystania specjalnego licznika (426)
 - 7.8.1.4. Technika sprawdzania za każdym razem (431)
 - o 7.8.2. Porządkowanie (432)
 - o 7.8.3. Przegląd (433)
- 7.9. Podsumowanie (434)

Część III Zagadnienia dotyczące projektu logicznego (437)

Rozdział 8. Projektowanie komponentów (439)

- 8.1. Abstrakcje i komponenty (439)
- 8.2. Projekt interfejsu komponentu (440)
- 8.3. Poziomy enkapsulacji (444)
- 8.4. Pomocnicze klasy implementacyjne (454)
- 8.5. Podsumowanie (459)

Rozdział 9. Projektowanie funkcji (461)

- 9.1. Specyfikacja interfejsu funkcji (462)
 - o 9.1.1. Operator czy metoda? (462)
 - o 9.1.2. Wolny operator czy składowa klasy? (468)
 - o 9.1.3. Metoda wirtualna czy niewirtualna? (472)
 - o 9.1.4. Metoda czysto wirtualna czy nie czysto wirtualna? (476)

- o 9.1.5. Metoda statyczna czy niestyczna? (477)
- o 9.1.6. Metody stałe czy modyfikowalne? (478)
- o 9.1.7. Metody publiczne, chronione czy prywatne? (483)
- o 9.1.8. Zwracanie wyniku przez wartość, referencję czy wskaźnik? (484)
- o 9.1.9. Zwracanie wartości typu const czy nie-const? (487)
- o 9.1.10. Argument opcjonalny czy obowiązkowy? (488)
- o 9.1.11. Przekazywanie argumentów przez wartość, referencję lub wskaźnik (490)
- o 9.1.12. Przekazywanie argumentów jako const lub nie-const (495)
- o 9.1.13. Funkcja zaprzeczona czy niezaprzeczona? (496)
- o 9.1.14. Funkcja inline czy nie inline? (497)
- 9.2. Typy podstawowe użyte w interfejsie (498)
 - o 9.2.1. Użycie typu short w interfejsie (498)
 - o 9.2.2. Użycie kwalifikatora unsigned w interfejsie (501)
 - o 9.2.3. Zastosowanie typu long w interfejsie (505)
 - o 9.2.4. Zastosowanie typów float, double oraz long double w interfejsie (507)
- 9.3. Funkcje specjalnego przeznaczenia (508)
 - o 9.3.1. Operatory konwersji (508)
 - o 9.3.2. Semantyka wartości generowanych przez kompilator (512)
 - o 9.3.3. Destruktor (513)
- 9.4. Podsumowanie (515)

Rozdział 10. Implementowanie obiektów (521)

- 10.1. Pola (521)
 - o 10.1.1. Wyrównanie naturalne (522)
 - o 10.1.2. Użycie typów podstawowych w implementacji (524)
 - o 10.1.3. Użycie konstrukcji typedef w implementacji (526)
- 10.2. Definicje funkcji (527)
 - o 10.2.1. Samokontrola (527)
 - o 10.2.2. Unikanie przypadków szczególnych (528)
 - o 10.2.3. Podział zamiast powielania (530)
 - o 10.2.4. Zbytня przebiegłość nie popłaca (533)
- 10.3. Zarządzanie pamięcią (533)
 - o 10.3.1. Wartości stanu logicznego i fizycznego (538)
 - o 10.3.2. Parametry fizyczne (541)
 - o 10.3.3. Systemy przydziału pamięci (545)
 - o 10.3.4. Zarządzanie pamięcią na poziomie klasy (551)
 - 10.3.4.1. Dodawanie własnych mechanizmów zarządzania pamięcią (555)
 - 10.3.4.2. Zablokowana pamięć (558)
 - o 10.3.5. Zarządzanie pamięcią na poziomie obiektu (562)
- 10.4. Użycie szablonów w dużych projektach (567)
 - o 10.4.1. Implementacje szablonów (567)
 - o 10.4.2. Zarządzanie pamięcią w szablonach (568)
 - o 10.4.3. Wzorce a szablony (577)
- 10.5. Podsumowanie (579)

Dodatki (583)

Dodatek A Wzorzec projektowy - Protocol Hierarchy (585)

- Protocol Hierarchy - struktury klas (585)
 - o Cel (585)
 - o Znany też jako (585)
 - o Motywacja (585)
 - o Zakres zastosowania (589)
 - o Struktura (590)
 - o Elementy składowe (590)
 - o Współpraca (591)
 - o Konsekwencje (591)
 - o Implementacja (592)
 - o Przykładowy kod (603)
 - o Znane zastosowania (607)
 - o Pokrewne wzorce (608)

Dodatek B Implementowanie interfejsu C++ zgodnego ze standardem ANSI C (611)

- B.1. Wykrywanie błędu alokacji pamięci (611)
- B.2. Definiowanie procedury main (tylko ANSI C) (618)

Dodatek C Pakiet określania i analizowania zależności (621)

- C.1. Korzystanie z poleceń adep, cdep i ldep (622)
- C.2. Dokumentacja wiersza polecenia (633)
- C.3. Architektura pakietu idep (643)
- C.4. Kod źródłowy (646)

Dodatek D Leksykon (647)

- D.1. Definicje (647)
- D.2. Główne reguły projektowe (652)
- D.3. Poboczne reguły projektowe (653)
- D.4. Wskazówki (654)
- D.5. Zasady (657)

Bibliografia (667)

Skorowidz (671)