
Spis treści

| | |
|---|-----------|
| Wprowadzenie | 15 |
| 1. Mechanizmy języka C++ | 19 |
| 1.1. Struktura programu – jednostki translacji | 21 |
| 1.1.1. Składnia tekstu źródłowego | 22 |
| 1.1.2. Preprocesor | 25 |
| 1.1.3. Składniki jednostki translacji | 26 |
| 1.2. System typów | 28 |
| 1.2.1. void | 29 |
| 1.2.2. bool | 29 |
| 1.2.3. Typy liczbowe | 30 |
| 1.2.4. Wskazania i tablice | 31 |
| 1.2.5. Enumeracje | 33 |
| 1.2.6. Typ referencyjny | 34 |
| 1.3. Deklarowanie i definiowanie funkcji | 37 |
| 1.3.1. main() | 38 |
| 1.3.2. Przeciążanie funkcji i operatorów; specyfikacja wiązania | 38 |
| 1.3.3. Parametry funkcji | 40 |
| 1.3.4. Funkcje z parametrami dodatkowymi | 41 |
| 1.3.5. Funkcje z parametrami domniemanymi | 43 |
| 1.3.6. Funkcje rozwijane | 45 |
| 1.4. Zarządzanie nazwami | 47 |
| 1.4.1. Zasięgi | 47 |
| 1.4.2. Przestrzenie nazw | 49 |
| 1.4.3. Deklaracje i dyrektywy using | 51 |

| | | |
|---------|---|-----|
| 1.4.4. | Anonimowe przestrzenie nazw | 52 |
| 1.4.5. | Wiązania nazw | 53 |
| 1.5. | Zarządzanie obiektami | 54 |
| 1.5.1. | Klasy pamięci | 55 |
| 1.5.2. | Definiowanie i deklarowanie obiektów | 56 |
| 1.5.3. | Zarządzanie pamięcią dynamiczną | 57 |
| 1.5.4. | Obiekty ustalone i ulotne | 60 |
| 1.6. | Zarządzanie operacjami | 62 |
| 1.6.1. | Wyrażenia | 62 |
| 1.6.2. | Priorytety operatorów i reguły przeciążania | 63 |
| 1.6.3. | Instrukcje | 66 |
| 1.7. | Klasy autonomiczne | 68 |
| 1.7.1. | Definiowanie klas autonomicznych | 68 |
| 1.7.2. | Składowe klasy | 69 |
| 1.7.3. | Konstruktory i destruktory | 71 |
| 1.7.4. | Projektowanie klas autonomicznych | 72 |
| 1.7.5. | Funkcje składowe | 74 |
| 1.7.6. | Przeciążanie operatorów | 75 |
| 1.7.7. | <code>this</code> | 77 |
| 1.7.8. | Funkcje i klasy zaprzyjaźnione | 78 |
| 1.7.9. | <code>Fraction</code> | 81 |
| 1.7.10. | Składowe statyczne | 83 |
| 1.7.11. | Efekty uboczne konstruktorów i destruktorów | 86 |
| 1.7.12. | Klasy ze zmienną strukturą wewnętrzną | 86 |
| 1.7.13. | Klasy zagnieżdżone | 91 |
| 1.7.14. | Listy inicjacyjne | 93 |
| 1.7.15. | Klasy lokalne | 95 |
| 1.7.16. | Wskazania na składowe klasy | 97 |
| 1.8. | Dziedziczenie i polimorfizm | 98 |
| 1.8.1. | Klasy pochodne: składnia i terminologia | 99 |
| 1.8.2. | Klasy bazowe wirtualne | 100 |
| 1.8.3. | Dostęp do składowych | 101 |
| 1.8.4. | Dziedziczenie i funkcje przeciążone | 103 |

| | | |
|-----------|---|------------|
| 1.8.5. | Dziedziczenie a zawieranie | 104 |
| 1.8.6. | Wskazania i referencje | 105 |
| 1.8.7. | Funkcje wirtualne i polimorfizm | 106 |
| 1.8.8. | Destruktory wirtualne | 107 |
| 1.8.9. | Funkcje wirtualne czyste i klasy abstrakcyjne | 108 |
| 1.8.10. | Realizacja funkcji wirtualnych | 110 |
| 1.8.11. | Identyfikacja typów w czasie wykonania | 111 |
| 1.9. | Szablony | 113 |
| 1.9.1. | <code>Vect</code> | 115 |
| 1.9.2. | Schemat składniowy szablonów | 116 |
| 1.9.3. | <code>Matrix</code> | 119 |
| 1.9.4. | Reguły konkretyzacji szablonów | 121 |
| 1.9.5. | Specjalizacje szablonów | 124 |
| 1.9.6. | Szablony w szablonach | 126 |
| 1.9.7. | Szablony i zaprzyjaźnienia | 127 |
| 1.9.8. | Szablony i dziedziczenie | 128 |
| 1.9.9. | Nazwy zależne | 130 |
| 1.10. | Obsługa sytuacji wyjątkowych | 132 |
| 1.10.1. | Wyjątki i sterowanie nielokalnie | 133 |
| 1.10.2. | Identyfikowanie wyjątków | 135 |
| 1.10.3. | Mechanizm obsługi wyjątków w C++ | 136 |
| 1.10.4. | Przekazywanie wyjątku do bloku obsługi | 138 |
| 1.10.5. | Wybór bloku obsługi | 138 |
| 1.10.6. | Ponowne zgłoszenie wyjątku | 139 |
| 1.10.7. | Specyfikacja wyjątków | 140 |
| 1.10.8. | Akcesoria standardowe | 141 |
| 1.10.9. | Blok funkcyjny <code>try</code> | 142 |
| 1.11. | Podsumowanie | 144 |
| 2. | Techniki stosowane w programowaniu generycznym | 145 |
| 2.1. | Klasy cech (trejty) | 145 |
| 2.1.1. | Wybór wartości zależnych od danego typu | 146 |
| 2.1.2. | Wybór algorytmu w czasie kompilacji | 147 |
| 2.1.3. | Standardowe klasy cech | 150 |

| | | |
|-----------|--|------------|
| 2.1.4. | Wybór typu w zależności od klasy cech | 152 |
| 2.1.5. | Minimalizacja wielkości kodu wynikowego | 153 |
| 2.2. | Klasy wytycznych | 154 |
| 2.2.1. | Szablon jako parametr szablonu | 154 |
| 2.2.2. | Tworzenie algorytmów z możliwością wyboru wariantu | 156 |
| 2.2.3. | Różnice pomiędzy klasami cech a klasami wytycznych | 160 |
| 2.3. | Metaprogramowanie | 160 |
| 2.3.1. | Metaprogramy zwiększające czytelność kodu | 160 |
| 2.3.2. | Obliczenia realizowane w czasie kompilacji | 161 |
| 2.3.3. | Kolekcje typów | 163 |
| 2.3.4. | Biblioteka <code>boost::mpl</code> | 165 |
| 2.4. | Statyczne asercje i klasy wymagań | 167 |
| 2.4.1. | Asercja czasu kompilacji | 168 |
| 2.4.2. | Klasy wymagań | 169 |
| 2.5. | Podsumowanie | 171 |
| 2.6. | Ćwiczenia | 172 |
| 3. | Uchwyty do obiektów | 175 |
| 3.1. | Sprytne wskaźniki | 176 |
| 3.1.1. | Wskaźnik <code>boost::scoped_ptr</code> | 177 |
| 3.1.2. | Wzorzec ukrywania implementacji (pimpl) | 179 |
| 3.1.3. | Kopiowanie jako przekazywanie własności, <code>std::auto_ptr</code> | 180 |
| 3.1.4. | Zliczanie odniesień: <code>boost::shared_ptr</code> i <code>boost::weak_ptr</code> | 182 |
| 3.1.5. | Wskaźniki narzucające interfejs, <code>boost::intrusive_ptr</code> | 189 |
| 3.1.6. | Porównanie | 191 |
| 3.2. | Opóźnione (leniwe) tworzenie i kopiowanie obiektów | 191 |
| 3.2.1. | Opóźnione (leniwe) tworzenie | 191 |
| 3.2.2. | Wartości opcjonalne | 194 |
| 3.2.3. | Opóźnione kopiowanie | 195 |
| 3.2.4. | Usunięcie obiektu tymczasowego | 198 |
| 3.3. | Iteratory | 200 |
| 3.3.1. | Wzorzec iteratora | 200 |
| 3.3.2. | Iteratory z biblioteki standardowej | 202 |
| 3.4. | Funktory | 203 |

| | | |
|-----------|---|------------|
| 3.4.1. | Obiekty typu <code>boost::function</code> | 203 |
| 3.4.2. | Wiązanie argumentów | 205 |
| 3.5. | Adaptory | 209 |
| 3.5.1. | Adapter obiektów | 209 |
| 3.5.2. | Adaptory klas | 210 |
| 3.5.3. | Adaptory redukujące kod generowany przez szablony | 210 |
| 3.6. | Podsumowanie | 212 |
| 3.7. | Ćwiczenia | 212 |
| 4. | Tworzenie obiektów | 217 |
| 4.1. | Fabryka obiektów | 218 |
| 4.2. | Prototyp | 222 |
| 4.3. | Singleton | 223 |
| 4.4. | Fabryka abstrakcyjna | 225 |
| 4.5. | Zarządzanie sterłą | 228 |
| 4.6. | Mechanizmy refleksji | 231 |
| 4.7. | Podsumowanie | 233 |
| 4.8. | Ćwiczenia | 233 |
| 5. | Współpraca pomiędzy obiektami | 235 |
| 5.1. | Polimorfizm | 235 |
| 5.1.1. | Interfejs bez funkcji wirtualnych | 236 |
| 5.1.2. | Wybór metody w czasie kompilacji | 236 |
| 5.1.3. | Własny mechanizm późnego wiązania | 238 |
| 5.2. | Odwiedzający (wizytator) | 239 |
| 5.2.1. | Odwroćenie zależności | 240 |
| 5.2.2. | Wizytator cykliczny | 241 |
| 5.2.3. | Wersja generyczna wizytatora | 243 |
| 5.2.4. | Wizytator acykliczny | 244 |
| 5.3. | Wielometody | 245 |
| 5.3.1. | Wielometody wykorzystujące rzutowanie dynamiczne | 246 |
| 5.3.2. | Wielometody wykorzystujące wizytator | 247 |
| 5.3.3. | Późne wiązanie zależne od wielu parametrów | 251 |
| 5.4. | Komenda | 252 |

| | | |
|-----------|---|------------|
| 5.4.1. | Reprezentacja komend | 253 |
| 5.4.2. | Ponawianie i wycofywanie akcji | 254 |
| 5.4.3. | Komendy złożone | 255 |
| 5.4.4. | Inne zastosowania komend | 256 |
| 5.5. | Obserwator | 256 |
| 5.5.1. | Opis wzorca | 257 |
| 5.5.2. | <code>boost::signals</code> oraz <code>boost::signals2</code> | 259 |
| 5.5.3. | Inne zastosowania | 262 |
| 5.6. | Stałość fizyczna i logiczna | 262 |
| 5.7. | Podsumowanie | 266 |
| 5.8. | Ćwiczenia | 266 |
| 6. | Złożone struktury obiektów | 269 |
| 6.1. | Kompozyt | 269 |
| 6.2. | Dekorator | 271 |
| 6.3. | Rekordy | 274 |
| 6.3.1. | Struktury | 274 |
| 6.3.2. | Warianty | 275 |
| 6.3.3. | Wartości opcjonalne | 277 |
| 6.3.4. | Obiekt dowolnego typu | 277 |
| 6.4. | Kolekcje jednowymiarowe | 278 |
| 6.4.1. | <code>std::vector</code> | 279 |
| 6.4.2. | <code>std::list</code> | 280 |
| 6.4.3. | <code>std::deque</code> | 281 |
| 6.4.4. | <code>std::set</code> | 282 |
| 6.4.5. | <code>std::multiset</code> | 283 |
| 6.4.6. | <code>std::map</code> i <code>std::multimap</code> | 283 |
| 6.4.7. | <code>std::tr1::unordered_set</code> i inne kontenery haszujące | 284 |
| 6.4.8. | <code>std::basic_string</code> | 285 |
| 6.4.9. | <code>boost::array</code> | 285 |
| 6.4.10. | Porównanie kolekcji | 286 |
| 6.4.11. | Operacje na kolekcjach | 287 |
| 6.4.12. | Wykorzystanie nienazwanych obiektów funkcyjnych | 293 |
| 6.4.13. | Standardowa biblioteka szablonów | 294 |

| | |
|--|------------|
| 6.5. Tablice wielowymiarowe | 295 |
| 6.6. Grafy, <code>boost::graph</code> | 296 |
| 6.6.1. Reprezentacja grafów | 296 |
| 6.6.2. Algorytmy grafowe | 301 |
| 6.7. Podsumowanie | 304 |
| 6.8. Ćwiczenia | 304 |
| 7. Dostęp do mechanizmów systemu operacyjnego | 309 |
| 7.1. Obsługa czasu i daty | 309 |
| 7.2. Wątki | 311 |
| 7.2.1. Współbieżność w C++ | 312 |
| 7.2.2. Zarządzanie wątkami | 313 |
| 7.2.3. Sekcje krytyczne | 315 |
| 7.2.4. Wzorzec projektowy monitora (pasywny obiekt) | 319 |
| 7.2.5. Wątki i implementacja potoków | 320 |
| 7.2.6. Blokada <code>shared_mutex</code> | 322 |
| 7.2.7. Zmienne warunkowe | 323 |
| 7.2.8. Unikanie niepotrzebnych blokad | 324 |
| 7.2.9. Aktywny obiekt | 325 |
| 7.2.10. Wątki – uwagi końcowe | 329 |
| 7.3. Strumienie | 329 |
| 7.3.1. Zapis i odczyt strumienia z pominięciem konwersji | 331 |
| 7.3.2. Formatowanie za pomocą strumieni | 332 |
| 7.3.3. Strumienie plikowe i napisowe | 334 |
| 7.3.4. Iteratory dla strumieni | 336 |
| 7.3.5. Bufory | 336 |
| 7.3.6. Filtry dla strumieni | 338 |
| 7.3.7. Strumienie błędów i komunikatów | 340 |
| 7.4. Asynchroniczna obsługa wejścia i wyjścia | 341 |
| 7.4.1. Generatory zdarzeń | 342 |
| 7.4.2. Obsługa protokołów sieciowych | 345 |
| 7.4.3. Wykorzystanie puli wątków | 347 |
| 7.5. Podsumowanie | 348 |
| 7.6. Ćwiczenia | 348 |

| | |
|---|------------|
| 8. Przetwarzanie tekstu | 353 |
| 8.1. Reprezentacja znaków i napisów, lokalizm | 353 |
| 8.1.1. Napisy | 353 |
| 8.1.2. Znaki | 354 |
| 8.1.3. Lokalizm | 355 |
| 8.2. Algorytmy dla napisów | 358 |
| 8.3. Wyrażenia regularne i gramatyki bezkontekstowe | 359 |
| 8.3.1. <code>boost::regex</code> – wyrażenia interpretowane w czasie działania . . | 359 |
| 8.3.2. <code>boost::xpressive</code> – wyrażenia tworzone w czasie kompilacji . . . | 363 |
| 8.3.3. Gramatyki bezkontekstowe, <code>boost::spirit</code> | 365 |
| 8.4. Podsumowanie | 370 |
| 9. Łączenie C++ z innymi językami programowania | 371 |
| 9.1. Łączenie C i C++ | 372 |
| 9.2. Biblioteki ładowane dynamicznie | 374 |
| 9.3. C++ i Python | 378 |
| 9.3.1. Rozszerzanie Pythona w C++ | 379 |
| 9.3.2. Osadzanie Pythona w C++ | 381 |
| 9.4. C++ i Java | 381 |
| 9.5. Podsumowanie | 382 |
| Odpowiedzi do ćwiczeń | 383 |
| Indeks | 386 |
| Bibliografia | 386 |