

Spis treści

0. Proszę tego nie czytać! (1)	
0.1. Wyruszamy na kolejną wyprawę! (1)	
1. Szablony o zmiennej liczbie parametrów (3)	
1.1. Szablon funkcji o zmiennej liczbie parametrów (i argumentów) (4)	
1.2. Jak dobrać się do argumentów tkwiących w pakiecie? (11)	
1.2.1. Ciekawe szablony zwracające rezultat (16)	
1.3. Szablon klas o dowolnej (zmiennej) liczbie parametrów (18)	
1.4. Trzy rodzaje pakietów parametrów szablonu (24)	
1.4.1. Pakiet szablonu będący pakietem wartości (25)	
1.5. Argumenty pakietu odbierane przez wartość, referencję, adres (30)	
1.6. Rozwinięcie według wzorca (czyli rozwinięcie "z kontekstem") (31)	
1.7. Rozwinięcie pakietu typów w klamrowej liście inicjalizatorów (34)	
1.7.1. Łatwe narzędzie do wypisania argumentów (34)	
1.8. Rozwinięcie pakietu na liście parametrów aktualnych innego szablonu (36)	
1.9. Gdzie można umieścić wyrażenia rozwijające pakiet parametrów (39)	
1.10. Ćwiczenia (39)	
2. Cechy języka wprowadzone do standardu C++14 (43)	
2.1. Zapis dwójkowy stałych dosłownych (43)	
2.2. Separatory cyfr w stałych dosłownych (44)	
2.2.1. Wypisywanie liczb w postaci binarnej (45)	
2.2.2. Wczytywanie liczb dwójkowych strumieniem wejściowym (48)	
2.3. Kompilator rozpoznaje typ rezultatu funkcji (50)	
2.4. Deklaracja typu rezultatu decltype(auto) (52)	
2.4.1. Przykład zastosowania konstrukcji decltype(auto) w szablonie funkcji (55)	
2.5. Szablon definicji zmiennej (59)	

- 2.5.1. Jak to drzewiej bywało, czyli świat bez szablonów zmiennych (64)
- 2.5.2. Teraz zobaczysz, jak prosto się to robi z C++14 (71)
- 2.5.3. Ciekawe zastosowanie: sprawdzenie cech charakteru danego typu (72)
- 2.5.4. Lubię, nie lubię... (77)
- 2.5.5. Dwa usprawnienia (80)
- 2.5.6. Realizacja tego pomysłu w programie (81)
- 2.6. Przeładowanie globalnych operatorów new, new[], delete i delete[] (86)
- 2.7. Nowości C++14 w wyrażeniach lambda (94)
 - 2.7.1. Przykład uogólnionego wyrażenia lambda (94)
 - 2.7.2. Przykład definicji obiektu na liście wychwytywania i jego inicjalizacja (96)
 - 2.7.3. Przykład wychwycenia na zasadzie przeniesienia (move) (99)
- 2.8. C++14 a funkcje constexpr (104)
 - 2.8.1. Zniesienie wielu ograniczeń w ciele funkcji constexpr (104)
 - 2.8.2. Funkcje składowe constexpr w C++14 nie są już automatycznie const (110)
- 2.9. Atrybuty (111)
 - 2.9.1. Nowy atrybut [[deprecated]] wprowadzony w C++14 (112)
 - 2.9.2. Oznaczenie wybranej funkcji jako przestarzałej (113)
 - 2.9.3. Argument funkcji uznany za przestarzały (114)
 - 2.9.4. Przestarzałe niestaticzne składniki klasy: funkcja składowa i dana składowa (114)
 - 2.9.5. Obiekt oznaczony jako przestarzały (115)
 - 2.9.6. deprecated a zbiorcza definicja kilku zmiennych (z ewentualną inicjalizacją) (116)
 - 2.9.7. Typy, które uznajemy za przestarzałe (116)
 - 2.9.8. Przestarzałe synonimy typów (w instrukcjach typedef i using) (118)
 - 2.9.9. Oznaczanie atrybutem [[deprecated]] specjalizacji szablonu klasy (118)
 - 2.9.10. Oznaczanie atrybutem [[deprecated]] specjalizacji szablonu funkcji (119)
- 2.10. Przewrotu nie było (119)
- 2.11. Ćwiczenia (120)

- 3. Cechy języka wprowadzone do standardu C++17 (123)
 - 3.1. Specyfikacja wyjątków staje się częścią typu funkcji (123)
 - 3.2. Technika "pomijanie kopiowania" bywa teraz obowiązkiem kompilatora (128)
 - 3.3. Przydomek alignas a operatory new i delete (134)
 - 3.3.1. Przeładowanie globalnych new i delete uwzględniające wyrównanie adresów (135)
 - 3.3.2. Jak przeładować wyrównujące operatory new/delete na użytek wybranej klasy (141)
 - 3.4. Porządek obliczania składników w złożonych wyrażeniach - nareszcie ustalony (144)
 - 3.5. Stała znakowa typu u8 (146)
 - 3.6. Szesnastkowy zapis liczb zmiennoprzecinkowych (147)
 - 3.6.1. Wypisywanie i wczytywanie zmiennoprzecinkowych liczb szesnastkowych (149)
 - 3.7. Wyrażenia poskładane w harmonijkę - ułatwienie pracy z pakietem argumentów (152)
 - 3.7.1. Pierwszy przykład użycia wyrażenia harmonijkowego w szablonie (152)
 - 3.7.2. Harmonijka z dodatkowym wyrażeniem początkowym (155)
 - 3.7.3. Cztery formy wyrażenia harmonijkowego (157)
 - 3.7.4. Kontekst wyrażenia harmonijkowego - przykład (159)
 - 3.8. Dozwolone słowo auto w deklaracji template <auto> (161)
 - 3.9. Kompilator rozpoznaje typ parametrów klasy szablonowej (166)
 - 3.9.1. Wektory czego innego niż widać (170)
 - 3.10. Instrukcja if constexpr - prawie jak kompilacja warunkowa (173)
 - 3.10.1. Instrukcja if constexpr rozwiązuje problem "lubianych" i "nie lubianych" typów (176)
 - 3.11. Wyrażenia inicjalizujące w instrukcjach if i switch (179)
 - 3.12. Dowiązania strukturalne, czyli łatwe "sięganie do składników" (182)
 - 3.12.1. Dowiązanie do tablic zbudowanych na bazie klasy std::array (184)
 - 3.12.2. Łatwe sięganie do składników struktur/klas (185)
 - 3.12.3. Przystosowanie naszej klasy do obsługi deklaracji dowiązań (192)
 - 3.12.4. Przystosowanie cudzej klasy do obsługi deklaracji dowiązań (198)
 - 3.13. Operator preprocesora zwany has_include (201)

- 3.14. Nowe atrybuty: maybe_unused, fallthrough i nodiscard (203)
 - 3.14.1. Atrybut [[maybe_unused]] (205)
 - 3.14.2. Atrybut [[fallthrough]] używany w instrukcji switch (208)
 - 3.14.3. Atrybut [[nodiscard]] - nie zlekceważ mnie (210)
- 3.15. Typ std::byte do operacji na surowych blokach pamięci (213)
- 3.16. Modyfikacje istniejących cech języka (222)
- 3.17. Rozluźnienie zasady inicjalizowania typów wyliczeniowych (223)
- 3.18. Modyfikacja deklaracji static_assert (224)
- 3.19. Prostszy sposób zapisu zagnieżdżonych przestrzeni nazw (225)
- 3.20. Dozwolone słowo typename w parametrze określającym inny szablon (229)
- 3.21. Dla zakresowej pętli for funkcje begin i end mogą zwracać odmienne typy (233)
- 3.22. Rozwinięcie pakietu możliwe nawet w deklaracji using (239)
- 3.23. Nowe zasady auto-rozpoznawania typu obiektów mających inicjalizację klamrową (245)
- 3.24. W C++17 agregat może być nawet klasą pochodną (247)
- 3.25. Zmiana typu rezultatu funkcji std::uncaught_exception (250)
- 3.26. Ćwiczenia (254)
- 4. Pośłowie - czyli C++20 ante portas (264)
- Skorowidz (265)