

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”)	32
1.7	Rozwinięcie pakietu typów w klamrowej liście inicjalizatorów	34
1.7.1	Łatwe narzędzie do wypisania argumentów	35
1.8	Rozwinięcie pakietu na liście parametrów aktualnych innego szablonu	37
1.9	Gdzie można umieścić wyrażenia rozwijające pakiet parametrów	39
1.10	Ćwiczenia	40
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 <i>decltype(auto)</i>	52
2.4.1	Przykład zastosowania konstrukcji <i>decltype(auto)</i> 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...	76
2.6	Przeładowanie globalnych operatorów <i>new</i> , <i>new[]</i> , <i>delete</i> i <i>delete[]</i>	77
2.7	Nowości C++14 w wyrażeniach <i>lambda</i>	84
2.7.1	Przykład uogólnionego wyrażenia <i>lambda</i>	84
2.7.2	Przykład definicji obiektu na liście wychwytywania i jego inicjalizacja	87
2.7.3	Przykład wychwycenia na zasadzie przeniesienia (<i>move</i>)	90
2.8	C++14 a funkcje <i>constexpr</i>	95
2.8.1	Zniesienie wielu ograniczeń w ciele funkcji <i>constexpr</i>	95

2.8.2	Funkcje składowe <i>constexpr</i> w C++14 nie są już automatycznie <i>const</i>	101
2.9	Atrybuty	102
2.9.1	Nowy atrybut <code>[[deprecated]]</code> wprowadzony w C++14	102
2.9.2	Oznaczenie wybranej funkcji jako przestarzałej	104
2.9.3	Argument funkcji uznany za przestarzały	104
2.9.4	Przestarzałe niestatyczne składniki klasy: funkcja składowa i dana składowa.....	105
2.9.5	Obiekt oznaczony jako przestarzały	106
2.9.6	<i>deprecated</i> a zbiorcza definicja kilku zmiennych (z ewentualną inicjalizacją)	107
2.9.7	Typy, które uznajemy za przestarzałe	107
2.9.8	Przestarzałe synonimy typów (w instrukcjach <i>typedef</i> i <i>using</i>)	108
2.9.9	Oznaczenie atrybutem <code>[[deprecated]]</code> specjalizacji szablonu klasy	109
2.9.10	Oznaczenie atrybutem <code>[[deprecated]]</code> specjalizacji szablonu funkcji	109
2.10	Przewrotu nie było.....	110
2.11	Cwiczenia	110

3 Cechy języka wprowadzone do standardu C++17114

3.1	Specyfikacja wyjątków staje się częścią typu funkcji.....	114
3.2	Technika „pomijanie kopiowania” bywa teraz obowiązkiem kompilatora.....	119
3.3	Przydomek <i>alignas</i> a operatory <i>new</i> i <i>delete</i>	125
3.3.1	Przeladowanie globalnych <i>new</i> i <i>delete</i> uwzględniające wyrównanie adresów	126
3.3.2	Jak przeladować wyrównujące operatory <i>new/delete</i> na użytek wybranej klasy.....	132
3.4	Porządek obliczania składników w złożonych wyrażeniach – nareszcie ustalony	135
3.5	Stała znakowa typu <i>u8</i>	137
3.6	Szesnastkowy zapis liczb zmiennoprzecinkowych.....	138
3.6.1	Wypisywanie i wczytywanie zmiennoprzecinkowych liczb szesnastkowych.....	140
3.7	Wyrażenia poskładane w harmonijkę – ułatwienie pracy z pakietem argumentów.....	143
3.7.1	Pierwszy przykład użycia wyrażenia harmonijkowego w szablonie.....	143
3.7.2	Harmonijka z dodatkowym wyrażeniem początkowym	146
3.7.3	Cztery formy wyrażenia harmonijkowego	148
3.7.4	Kontekst wyrażenia harmonijkowego – przykład	150
3.8	Dozwolone słowo <i>auto</i> w deklaracji <i>template <auto></i>	152
3.9	Kompilator rozpoznaje typ parametrów klasy szablonowej	157
3.9.1	Wektory czego innego niż widać	161
3.10	Instrukcja <i>if constexpr</i> – prawie jak kompilacja warunkowa.....	164
3.10.1	Instrukcja <i>if constexpr</i> rozwiązuje problem „lubianych” i „nie lubianych” typów	167
3.11	Wyrażenia inicjalizujące w instrukcjach <i>if</i> i <i>switch</i>	170
3.12	Dowiązanie strukturalne, czyli łatwe „sięganie do składników”	172
3.12.1	Dowiązanie do tablic zbudowanych na bazie klasy <i>std::array</i>	174
3.12.2	Łatwe sięganie do składników struktur/klas.....	175
3.12.3	Przystosowanie naszej klasy do obsługi deklaracji dowiązań	182
3.12.4	Przystosowanie cudzej klasy do obsługi deklaracji dowiązań	189
3.13	Operator preprocesora zwany <i>has include</i>	191
3.14	Nowe atrybuty: <i>maybe_unused</i> , <i>fallthrough</i> i <i>nodiscard</i>	194
3.14.1	Atrybut <code>[[maybe_unused]]</code>	195
3.14.2	Atrybut <code>[[fallthrough]]</code> używany w instrukcji <i>switch</i>	198
3.14.3	Atrybut <code>[[nodiscard]]</code> – nie zlekceważ mnie.....	200
3.15	Typ <i>std::byte</i> do operacji na surowych blokach pamięci	204
3.16	Modyfikacje istniejących cech języka	213
3.17	Rozluźnienie zasady inicjalizowania typów wyliczeniowych	213
3.18	Modyfikacja deklaracji <i>static assert</i>	215
3.19	Prostszy sposób zapisu zagnieżdżonych przestrzeni nazw	215
3.20	Dozwolone słowo <i>typename</i> w parametrze określającym inny szablon	219
3.21	Dla zakresowej pętli <i>for</i> funkcje <i>begin</i> i <i>end</i> mogą zwracać odmienne typy	223
3.22	Rozwinięcie pakietu możliwe nawet w deklaracji <i>using</i>	229

3.23	Nowe zasady <i>auto</i> -rozpoznawania typu obiektów mających inicjalizację klamrową	235
3.24	W C++17 agregat może być nawet klasą pochodną	237
3.25	Zmiana typu rezultatu funkcji <i>std::uncaught_exception</i>	240
3.26	Ćwiczenia	244

4	Posłowie – czyli C++20 <i>ante portas</i>.....	254
----------	---	------------

	Skorowidz.....	255
--	-----------------------	------------