Errata do książki Jerzy Grębosz "Opus magnum C++11", wydanie pierwsze, Helion 2017

ISBN: 978-83-283-3881-4

Stan z dnia: 23 IV 2020

Uwaga: w kratkach o tym kolorze tła są poprawki, które wydają się zmieniać poprawną polszczyznę na błędną.
Są to jednak uwagi czytelników, którzy dbają o to, by w tekstach programów nie było polskich znaków.

Strona	Wiersz	G-od góry D-od dołu	Jest	Powinno być	Kto zgłosił poprawkę
23	4	G	// definicja obiektu o nazwie i	// definicja obiektu o nazwie n	Mateusz Dobija
29	5	D	pasazerów	pasazerow	Miłosz Dominiak
30	7	G	w trakcie wykonywanie	w trakcie wykonywania	Krzysztof Piech
32	6	D	, str.20)	, str. 37)	Kamil Lepianka
32	6	D	(§2.9, str. 20)	(§2.9, str. 37)	Mateusz Bałdowski
41	9	G	while, for, do nie jest mi straszne	while, for, do while nie jest mi straszne	Krzysztof Piech
43	3	D	w trakcie wykonywana	w trakcie wykonywania	Mateusz Bałdowski
47	15	G	wchar_t nadaje [mało wyraźny odstęp]	wchart_t nadaje	Kamil Lepianka
48	10	G	on nazwiska	od nazwiska	Mateusz Bałdowski
50	2	D	W tej tabeli typ int ma te same możliwe wartości co typ long – bo jest zapisywany za pomocą 32 bitów.	W tej tabeli typ long ma te same możliwe wartości co typ long long – bo jest zapisywany za pomocą 64 bitów.	Krzysztof Piech
50	11	D	long 32 4 unsigned long 32 4 signed long 32 4	long 64 8 signed long 64 8 unsigned long 64 8	Grzegorz Szpetkowski
50	17	G	long int 32 4 -9223372036854775808 do 9223372036854775807	long 32 4 -2147483648 do 2147483647	Oskar Lewandowski
50	19	G	signed long 32 4 -9223372036854775808 do 9223372036854775807	signed long 32 4 -2147483648 do 2147483647	Oskar Lewandowski
50	21	G	unsigned long 32 4 0 do 18446744073709551615	unsigned long 32 4 0 do 4294967295	Oskar Lewandowski
51	16	G	[dodać linijkę]	long long – ma co najmniej 64 bity (8 bajtów)	Grzegorz Szpetkowski
52	2	G	[ewentualnie dodać linijkę]	#include <string>	Grzegorz Szpetkowski
54	4	D	funkcja num	funkcja min	Piotr Andrzejewski
55	2	D	int 32_t dana	int32_t dana	Kamil Lepianka
56	7	G	za błąd.	za zachętę, aby użyć „oszukiwanego” typu składającego się z dwóch słów 32-bitowych.	Grzegorz Szpetkowski Piotr Batko
56	9	D	Dla takiego typ	Dla takiego typu	Mateusz Dobija
56	15	D	(błąd kompilacji).	(ostrzeżenie).	Grzegorz Szpetkowski
56	18	D	błąd kompilacji (bo zbyt duża wartość)	ostrzeżenie kompilatora (że zbyt duża wartość)	Grzegorz Szpetkowski
68	15	D	dopasowanie	dopasowane	Jan Kędzierski
74	12	G	bardziej skomplikowanie	bardziej skomplikowane	Michał Grębosz
75	12	G	endl;;	endl;	Kamil Klecha
75	14	G	\nlina druga	\nlinia druga	Michał Grębosz
75	15	D	lina raw druga	linia raw druga	Krzysztof Piech
75	16	G	lina raw druga	linia raw druga	Michał Grębosz
75	28	G	lina druga	linia druga	Michał Grębosz
76	7	G	< 32	<= 16	Kamil Lepianka
79	5	G	Zakres instrukcji	Zakres: instrukcja	Krzysztof Piech
84	15	G	❸	❷	Kamil Purcel
85	22	G	str. 544	str. 545	Mateusz Dobija
94	4	D	Dla wtajemniczonych	👑 Dla wtajemniczonych	Krzysztof Piech
94	12	G	napięcie	napiecie	Krzysztof Piech
94	16	G	napięcie	napiecie	Krzysztof Piech
98	3	D	przewin_tasme oraz rozladuj_tasme	przewin oraz powrot_do_poczatku	Piotr Andrzejewski
98	14	D	powrót_do_poczatku	powrot_do_poczatku	Krzysztof Piech
98	18	G	powrót_do_poczatku	powrot_do_poczatku	Krzysztof Piech
99	3	D	lista_wyliczeniowa	lista wyliczeniowa	Krzysztof Piech
99	16	G	if(co_robić ==	if(co_robic ==	Krzysztof Piech
103	12	D	lista_wyliczeniowa	lista wyliczeniowa	Krzysztof Piech
107	3	G	przykłady definiowana	przykłady definiowania	Mateusz Bałdowski
111	3	D	oferuje jest jeszcze	oferuje jeszcze	Tomasz Serwański
113	16	G	(typu enum wyliczanka)	(typu enum class wyliczanka)	Krzysztof Piech
114	3	G	jakieś	jakiejś	Krzysztof Piech
115	11	D	istniejącemu).	istniejącemu.	J.G.
117	5	G	Cwiczenie XLIII	[Jest prawie identyczne z ćwiczeniem VI]	Kamil Lepianka
124	6	D	równoczesnie	rownoczesnie	J.G.

124	8	D	if((m > 10) && (k > 0)) // koniunkcja
			{
			cout << "Hurra! m jest wieksze od 10 i… [itd.]\n";
			}
				[Te instrukcje można tutaj pominąć,
				bo i tak będą powtórzone i omówione na następnej stronie]
					Kamil Lepianka

124	12	D	równosc	rownosc	Miłosz Dominiak
124	14	D	równosc	rownosc	Miłosz Dominiak
125	5	D	równe	rowne	Kamil Purcel
127	6	G	równe	rowne	Kamil Purcel
136	11	G	poszczególne typy	poszczegolne typy	Kamil Lepianka
136	22	G	wartości	wartosci	Kamil Lepianka
136	27	G	niż	niz	Kamil Lepianka
139	15	D	lower().	lowest().	Damian Rypel
151	7	D	lewostronnie łącznie,	lewostronnie łączne,	Maciej Kłoda
160	9	G	3', kropka, '1' , '4'.	'3', kropka, '1' , '4' (i ewentualnie jakieś znaki '0').	Damian Rypel
164	21	G	elementów	elementow	Kamil Purcel
165	8	G	jest deklaracja using	jest dyrektywa using	Michał Gręboszz
166	13	G	wektora w	wektora ww	Kamil Lepianka
170	1	G	też	tez	Kamil Purcel
170	8	G	obsłużyć też	obsluzyc tez	Kamil Purcel
170	23	G	też	tez	Kamil Purcel
170	25	G	obsłużyć też	obsluzyc tez	Kamil Purcel
172	11	D	VI Co to znaczy…	VI Dla wtajemniczonych. Co to znaczy…	Miłosz Dominiak
172	14	D	IV Jak zareaguje…	IV Dla wtajemniczonych. Jak zareaguje…	Miłosz Dominiak
183	14	D	, str. 841)	, str. 876)	J.G.
213	23	G	liczba = 10 i = 4 sumka = 14	liczba = 10 n = 4 sumka = 14	Miłosz Dominiak
220	17	G	Europejczyków	Europejczykow	Adam Masiewicz
221	7	G	Europejczyków	Europejczykow	Adam Masiewicz
227	7 14	G D	wywołanie możemy	wywolanie mozemy	Kamil Lepianka
234	9	G	, zatem równie dobrze tej dekrementacji mogło by tu nie być	[usunąć ten fragment zdania]	Tomasz Dąbrowski
240	19	D	obywają się	odbywają się	Adam Masiewicz
242	2	D	std::sting	std::string	Kamil Lepianka
244	14	D	obiektu dystans_mile	obiektu dystans_km	Mateusz Bałdowski
247	6	G	'a' < 'b'	a < b	Mateusz Bałdowski
252	5	D	220	228	Miłosz Dominiak
257	5	D	v1	ref_rd	Piotr Batko
257	7	D	v2	ref_rd	Piotr Batko
257	14	D	użyciu … zwykłych	uzyciu … zwyklych	Kamil Purcel
257	22	D	użyciu … zwykłych	uzyciu … zwyklych	Kamil Purcel
259	1	D	użyciu	uzyciu	Kamil Purcel
260	4	G	użyciu	uzyciu	Kamil Purcel
261	9	D	w wyrażeniu inicjalizującym	w wyrazeniu inicjalizujacym	Kamil Purcel
262	14	G	// double& (30)	// fun. double & (30)	Piotr Batko
262	22	D	w wyrażeniu inicjalizującym	w wyrazeniu inicjalizujacym	Kamil Purcel
273	4	D	Zwracam uwagę, że nie może być spacji ani – ogólniej – białych znaków… [itd. do końca akapitu]	Zwracam uwagę, że nie może być spacji ani – ogólniej – białych znaków między nazwą tej „niby-funkcji” ILOCZYN, a nawiasem otwierającym jej listę argumentów (a, b, c).	Piotr Batko
283	14	D	_Pragma(X) odpowiada temu, co #pragma X.	_Pragma("X") odpowiada temu, co #pragma X.	Damian Rypel
285	5	G	__cplusplus__	__cplusplus	Kamil Lepianka
285	21	G	and __LINE__	i __LINE__	Piotr Batko
287	8	G	procesora	preprocesora	Tomasz Dąbrowski
296	5	G	tablicy typu long	tablicy obiektów typu long	Jakub Mazurek
297	8	G	tablicy typu int	tablicy obiektów typu int	Jakub Mazurek
299	19	G	sceny	scena	Jakub Mazurek
299	22	G	sceny	scena	Jakub Mazurek
303	7	G	Instrukcją ++i	Instrukcją i++	Jakub Mazurek
312	11	D	nawiasów klamrowych	nawiasów kwadratowych	Kamil Lepianka
324	16	D	[dodać linijkę]	#include <string>	Andrzej Rzoska
328	14	G	liczba_kolumn = 6;	liczba_kolumn = 5;	Michał Gręboszz
328	24	G	(6 elementowymi	(5-elementowymi	Michał Gręboszz
330	4	G	numerr_rzędu	numer_rzędu	Michał Gręboszz
332	7 i nast.	G	| 5 | | 105 | 750 | | 205 |	| | 750 | |	Piotr Jastrzębski oraz Andrzej Rzoska
333	6	D	w przypadku tablicy	w przypadku funkcji	Andrzej Rzoska
339	14	D	rezerwacj_miejsc	rezerwacji_miejsc	Piotr Batko
340	10	D	rezerwacj_miejsc	rezerwacji_miejsc	Piotr Batko
347	6	D	3 pięter	2 pięter	Marcin Białek
349	2	D	Ile rzędów jest na danym piętrze…	Ile foteli jest na danym piętrze…	Marcin Białek
350	17	G	… [nr_fotela] = tekst_nikt;	… [nr_fotela] = tekst_nikt; // ❺❻	Michał Gręboszz

367	25	G	Ustawienie wskaźnika na nullptr... [ta i 4 następne linijki]	Ustawienie wskaźnika na nullptr ma tę wspaniałą zaletę, że łatwo potem ten fakt sprawdzić i ewentualne na niego zareagować.	Kamil Lepianka
368	20	G	wskazuje na adres zerowy (nullptr)!";	wskazuje na coś konkretnego !";	Kamil Lepianka
368	21	G	wskazuje na coś konkretnego !";	wskazuje na adres zerowy (nullptr)!";	Kamil Lepianka
368	31	G	if(wsk != nullptr)	if(wsk == nullptr)	J.G.
379	9	D	Jak jest różnica	Jaka jest różnica	Kamil Lepianka
385	15	D	- jako tablicę, - jako wskaźnik.	- stosując zapis tablicowy, - stosując zapis wskaźnikowy.	Piotr Batko
386	11	D	także jako tablica.	jakby jako tablica, czyli przy użyciu zapisu []	Piotr Batko
387	5	G	jako tablicę, czy jako wskaźnik?	stosując zapis tablicowy, czy wskaźnikowy?	Piotr Batko
414	18	G	std::alloc	std::bad_alloc	Piotr Jastrzębski
416	21	D	*[Ćwiczenie XXIII rozpoczyna się od słow "w poprzednim ćwiczeniu"…, podczas gdy naprawdę chodzi mu o ćwiczenie **następne** (XXIV)].*	*[Zamienić miejscami ćwiczenie XXIV z ćwiczeniem XXIII]*	Waldemar Barczyk
417	1	G	256 elementów	512 elementów	Kamil Lepianka
428	14	G	kompilator musi (acz niechętnie) pozwalać	kompilatory czasem pozwalają	Damian Rypel
429	17	D	Możemy jednak go to tego zmusić	Możemy jednak go do tego zmusić	Piotr Batko
431	16	G	wsk_do_zmiennej = -333;	*wsk_do_zmiennej = -333;	Piotr Batko
441	19	G	7+1	6+1	Kamil Lepianka
455	19	D	Ppowiem	Powiem	Piotr Jastrzębski
457	1 8	G G	orzekających	orzekajacych	Kamil Lepianka

461	3
	i nast.
		G	char wzorek[25];
			for(int i = 0 ; i < 500 ; ++i) {
			for(int k = 0 ; k < 24 ; k++) wzorek[k] = ' ';
			wzorek[24] = 0;
				char wzorek[**26**];
				for(int i = 0 ; i < 500 ; ++i) {
				for(int k = 0 ; k < **25** ; k++) wzorek[k] = ' ';
				wzorek[**25**] = 0;
					Piotr Bosowski

465	15	G	przelicz_elementu	przelicz_elementy	Bartłomiej Umiński
467	1	D	void (*twf)[3]() =	void (*twf[3])() =	Łukasz Komsta
480	13	G	funkcji dźwięk(int), ale jest funkcja dźwięk(double).	funkcji dźwięk(double), ale jest funkcja dźwięk(int).	Kamil Lepianka
492	11	G	void f(char);	void f(char*);	Tomasz Dąbrowski
493	3	D	będący tablicą typu int int tablica[10] fun(tablica);	będący typu int int obiekt; fun(obiekt);	Piotr Batko
493	6	D	dokładne	dokładne, bo konwersja niepotrzebna	J.G.
494	1	G	Pasuje	Bez potrzeby konwersji pasuje	J.G.
494	1	D	T[] ---> T*	T[] ---> T* (zapis „wskaźnikowy”)	Piotr Batko
494	2	G	void fun(int ttt[]);	void fun(int);	Piotr Batko
494	3	D	T[] ---> T[]	T[] ---> T[] (zapis „tablicowy”)	Piotr Batko
494	8	G	nie znaleziono w etapie …	[usunąć tę i następną linijkę]	Piotr Batko
494	10	G	natomiast jest funkcja	znaleziono funkcję	Piotr Batko
494	12	G	odebrać albo jako tablicę, albo jako wskaźnik	odebrać jako wskaźnik	Piotr Batko

495	1	G	takie dwie funkcje
			void fun(int ttt[ ]);
			void fun(int *wsktab);
			po prostu nie mogą równocześnie
				takie deklaracje
				void fun(int ttt[ ]);
				void fun(int *wsktab);
				void fun (int ttt[1000]);
				są deklaracjami jednej i tej samej funkcji (a nie trzech konkurujących ze sobą funkcji). Takie trzy funkcje po prostu nie mogłyby
					Piotr Batko

518	6	D	= napis);	= napis;	Damian Rypel
518	23	G	wskaźnik->.funkcja	wskaźnik->funkcja	Piotr Jastrzębski
533	1	G	A po dzielić	A po co dzielić	Piotr Jastrzębski
542	9	G	funkcji zapamietaj	funkcji zmien_rodzaj_pociagu	Damian Rypel
546	12	G	Tosoba kompozytor, autor; Tosoba autor;	Tosoba kompozytor, autor;	Miłosz Dominiak
549	4	G	naszą klasą numer	naszą klasą Tnumer	Michał Gręboszc
556	5	D	(operatorem delete [ ]) powinniśmy zwolnić tę rezerwację.	powinnien on zwolnić tę rezerwację (operatorem delete [ ]).	Kamil Lepianka
565	19	G	Spróbujmy jeszcze raz	㉘ Spróbujmy jeszcze raz	Damian Rypel
568	3	D	obiekt::funkcja	obiekt.funkcja	Damian Rypel
568	10	D	obiekt::funkcja	obiekt.funkcja	Damian Rypel
581	5	D	Natomiast obecność tych	Obecność tych	Damian Rypel Grześ Szpetkowski
581	7	D	[Akapit:] Bo const i volatile…	[Usunąć cały ten akapit]	Damian Rypel Grześ Szpetkowski
581	11	D	[Akapit:] W obrębie klasy...	[Usunąć cały ten akapit]	Damian Rypel Grześ Szpetkowski
586	14	D	(int dx, int dy)	(int dx, int dy) const	Damian Rypel
586	19	G	deklaracje alias (i to takie	deklaracje aliasów (takie	Kamil Lepianka
590	5	G	Funkcja zwracająca rezultat typu const	Funkcja składowa typu const	J.G.

595	9	G	moj_biezacy_czas++;	++moj_biezacy_czas;	Waldemar Barczyk
595	12	G	moj_biezacy_czas++;	++moj_biezacy_czas;	Waldemar Barczyk
597	3	G	GAMMA: 20.002 SEPAR: 20 ALFAP: 19.999	GAMMA: 20.002 s SEPAR: 20 s ALFAP: 19.999 s	Damian Rypel
597	4	G	GAMMA: 40.003 SEPAR: 40 ALFAP: 39.997	GAMMA: 40.003 s SEPAR: 40 s ALFAP: 39.997 s	Damian Rypel
597	5	G	GAMMA: 60.004 SEPAR: 60 ALFAP: 59.995	GAMMA: 60.004 s SEPAR: 60 s ALFAP: 59.995 s	Damian Rypel
597	13	D	GAMMA: 600 SEPAR: 600 ALFAP: 600	GAMMA: 600 s SEPAR: 600 s ALFAP: 600 s	Damian Rypel
635	2	G	Wartość indeksu (8) potwierdza	Wartość indeksu (6) potwierdza	Damian Rypel
651	22	G	Zamienia ona treść tego obiektu na C-string, a następnie ten rezultat staje się argumentem wywołania funkcji biblioteczna.	Zwraca ona adres miejsca w pamięci (adres tablicy), gdzie obiekt string obecnie przechowuje znaki, będące treścią stringu.	Damian Rypel
653	14	G	Tak się nie da! Funkcja c_str sporządza dla Ciebie rezultat w postaci C-stringu w jakimś specjalnie zarezerwowanym (przez nią) na tę okoliczność miejscu. Jest to rezultat ważny tylko w tym momencie historycznym. Jeśli dokonasz jakiejś następnej zmiany w obiekcie klasy string (np. dodasz jakieś znaki), to owa zmiana nie przeniesie się automatycznie do tego sporządzonego wcześniej C-stringu. Gorzej: wówczas obiekt mógłby nawet zwolnić rezerwację pamięci przeznaczoną na ten C-string jako już niepotrzebną.	Tak się nie da! Funkcja c_str udostępnia Ci rezultat w postaci adresu C-stringu, a jest to rezultat ważny tylko w tym momencie historycznym. Jeśli dokonasz jakiejś następnej zmiany w obiekcie klasy string (np. dodasz jakieś znaki), to owa zmiana może sprawić, że obiekt zwolni dotychczasową rezerwację pamięci przeznaczoną na ten tekst i zarezerwuje inną.	Damian Rypel
658	4	G	compare(12, string::npos, music+7, 2000);	compare(12, string::npos, music+7, strlen(music+7));	Damian Rypel
658	14	G	Czwartym, ostatnim argumentem, uczyniłem liczbę 2000, co oczywiście przekracza długość tego konkretnego C-stringu, ale w tym przypadku jest to mój sposób powiedzenia, że chciałbym skorzystać z całości tego C-stringu.	Czwartym, ostatnim argumentem, jest wyrażenie strlen(music+7), które określa liczbę dalszych znaków tego C-stringu.	Damian Rypel
661	1	D	opisu klasy std::string.	opisu klasy std::locale.	Damian Rypel
661	16	G	programu ❶ dyrektywa	programu dyrektywa	Damian Rypel
663	16	D	cin >> miasta;	cin >> miasto;	Kamil Lepianka
668	2	D	w przeciwieństwie do operatora<<	w przeciwieństwie do operatora>>	Michał Gręboszz
675	5	G	który jest stał, -to znaczy, że pokazany mu	który jest stały, to znaczy, że przechowywany w nim	Michał Gręboszz
697	17	G	int x, y;	int kolor, pozycja;	Damian Rypel
698	12	G	dolnego rogu prostokąta	dolnego rogu kwadratu	Kamil Lepianka
706	7	G	składników statycznych.	składników, nawet tych statycznych.	Damian Rypel
709	5	D	ma przydomek mutable. W klasie K chcemy zamieścić deklarację z tą	ma przydomek static. W klasie K chcemy zamieścić deklarację przyjaźni z tą	Damian Rypel
715	3	G	Niekoniecznie. [i dalej do końca paragrafu]	Niekoniecznie. Niebawem dowiesz się, że niektóre typy (klasy) mogą być szczególnym rodzajem innych. Samochód jest przecież szczególnym rodzajem pojazdu. Kolejność, w jakiej ustawia się bloki catch przechwytujące obiekty takich typów, ma wtedy znaczenie.	Damian Rypel
722	22	G	helper	Thelper	Damian Rypel
723	7	G	helper	Thelper	Damian Rypel
726	1	G	helper	Thelper	Damian Rypel
726	11	G	helper	Thelper	Damian Rypel
726	31	G	helper	Thelper	Damian Rypel
730	3	D	helper	Thelper	Damian Rypel
732	9	G	void fun0()	void fun0();	Kamil Lepianka
769	10	G	Tprzyrzad (int, int, char*, char*, int);	Tprzyrzad (int, int, string, string, int);	Kamil Lepianka
769	11	G	Tprzyrzad (int, int, char*, char*);	Tprzyrzad (int, int, string, string);	Kamil Lepianka
771	17	D	(błąd kompilacji).	(ostrzeżenie).	Kamil Lepianka
772	18	G	Vari(2, 6, "Wznoszenie", "stopy/sek."); ㉔	Vari(2, 1, "Predkosc", "wezlow"); ㉓	Damian Rypel
772	22	G	= Tprzyrzad(2, 6, "Wznoszenie", "stopy/sek.");	= Tprzyrzad(2, 1, "Predkosc", "wezlow");	Damian Rypel
776	12	D	out_of_range	bad_alloc	Kamil Lepianka

781	6	D	Oto wywołanie destruktora na rzecz obiektu int:
			int a;
			a.~int();
				Musimy to jednak zrobić posługując się synonimem utworzonym za pomocą instrukcji using lub typedef.
				int a;
				using ints = int;
				a.~ints(); // Wywołanie destruktora
					Damian Rypel

788	14	G	przezwisko{ref}	przezwisko(ref) [nie jest to lepsze, ale tu potrzebujemy tej formy w celach szkoleniowych]	Kamil Lepianka
789	7	D	argumentu ddd.	argumentu dd.	Michał Gręboszz
789	9	D	stala{ddd}	stala{dd}	Michał Gręboszz
791	4	G	stal(dd)	stala(dd)	Damian Rypel

792	7	D	stal	stala	Andrzej Siemion
796	12	D	⓰ To właśnie	■⓰■ To właśnie *[numerek w czarnej ramce]*	Damian Rypel
801	7	D	podając tylko jeden argument będący napisem, to znaczy, że chodzi mu o stworzenie przedziału ograniczonego liczbami [300, brzeg],	podając (oprócz napisu) tylko jeden argument liczbowy (brzeg), to znaczy, że chodzi mu o stworzenie przedziału ograniczonego liczbami 300 oraz brzeg,	Damian Rypel
805	15	G	initializer_list<T	initializer_list<T>	Kamil Lepianka
805	15	G	initializer_list<T	initializer_list<T>	Michał Gręboszc
808	14	G	składnia tablicowa	skladnia tablicowa	Kamil Lepianka
809	4	G	składnia tablicowa	skladnia tablicowa	Kamil Lepianka
812	9	D	od wartości argumentu t	od wartości argumentu pierwsza	Kamil Lepianka
812	15	G	liczba: 511	liczba: 512	Michał Gręboszc
814	19	D	auto b { 1.2, 3.14, 55.5 };	auto b = { 1.2, 3.14, 55.5 };	Damian Rypel
816	24	G	dla argumentów	dla argumentow	Kamil Lepianka
817	13	G	dla argumentów	dla argumentow	Kamil Lepianka
820	21	D	using namespace std;	#include <ctime> using namespace std;	Damian Rypel
825	1	D	Tpanel	TPanel	Kamil Lepianka
825	18	G	ma być w kolumnie 40.	ma być w kolumnie 35.	Kamil Lepianka
826	1	G	* a dopiero poniżej deklaracja składnika poz_x? Wystarczy spojrzeć na tę listę (4) i od razu widać, że do inicjalizacji prędkościomierza potrzebna jest już sensowna wartość składnika poz_x. Tymczasem w nim są jeszcze śmieci, bo jako leżący „poniżej” zostanie zainicjalizowany dopiero za chwilę. W takiej sytuacji troskliwy kompilator zachowa się inteligentnie i zmieni kolejność inicjalizacji, ratując nas przed katastrofą. W dodatku wypisze wtedy informację-ostrzeżenie mówiącą, że w trosce o naszą karierę programisty, zmienił tu kolejność wykonywania inicjalizacji składników klasy.	* a dopiero poniżej deklaracja składnika poz_x, * a co najgorsze: prędkościomierz byłby inicjalizowany wartością poz_x (a nie jak teraz wartością x). Co by wtedy było? Niemal katastrofa, bo w chwili inicjalizacji prędkościomierza, w składniku poz_x byłyby jeszcze śmieci. (Dlatego że składnik poz_x leży w klasie po składniku prędkościomierz, czyli zostanie zainicjalizowany dopiero za chwilę). W takiej sytuacji troskliwy kompilator wypisze wtedy ostrzeżenie mówiące, że w trosce o naszą karierę programisty, powinniśmy zmienić kolejność wykonywania inicjalizacji składników klasy.	Damian Rypel
829	10	D	constexpr int ile_piksel_swieci()	constexpr int ile_piksel_swieci() const	Damian Rypel
829	14	D	constexpr insigned int id()	constexpr insigned int id() const	Damian Rypel
831	26	G	typu typ_pix, czyli unsigned int.	typu typ_pix, czyli unsigned char.	Michał Gręboszc
832	12	G	poza ciałem klasy, to kompilator zaprotestuje, że nie wie (jeszcze), jak to zrobić.	poza ciałem klasy (zapewne w innym pliku), to kompilator, pracując nad definicją jakiegoś obiektu constexpr (tej klasy), może jeszcze nie znać definicji tego konstruktora z tamtego pliku. Zaprotestuje więc, że nie wie jak ten obiekt constexpr zdefiniować.	Damian Rypel
844	18	D	cout << "\nTeraz wywolam funkcje druga, " "a jej rezultat podstawie do innej kalibracji \n"; cout << "Obiekt chwilowy zwrocony jako rezultat" " funkcji \nma opis "; cout << (fun_druga()).opis() << endl; // (12)	cout << "\nTeraz wywolam funkcje druga, " "(ktora zwroci rezultat typu Tkalibracja)\n"; cout << (fun_druga()).opis() << endl; // (12) cout << " ^------To opis obiektu zwroconego " "przez funkcje";	Damian Rypel to jest zmiana kosmetyczna, ale myślę że korzystna
845	5	G	O ktory kanal widma chodzi? : 100	O ktory kanal widma chodzi? : 100 ❽	Damian Rypel
845	7	G	opisanej jako ORYGINALNA KOBALTOWA	opisanej jako ORYGINALNA KOBALTOWA ❾	Damian Rypel
845	11	G	To ja, konstruktor kopiujacy!!! --	To ja, konstruktor kopiujacy!!! -- ❿	Damian Rypel
845	15	G	Teraz wywolam funkcje druga, a jej rezultat podstawie do innej kalibracji W funkcji fun_druga definiuje kalibracje i ma ona opis: WEWNETRZNA Obiekt chwilowy zwrocony jako rezultat funkcji ma opis -- To ja, konstruktor kopiujacy!!! --	Teraz wywolam funkcje druga, (ktora zwroci rezultat typu Tkalibracja) W funkcji fun_druga definiuje kalibracje i ma ona opis: WEWNETRZNA -- To ja, konstruktor kopiujacy!!! -- ^------To opis obiektu zwroconego przez funkcje	Damian Rypel to jest zmiana kosmetyczna, ale myślę że korzystna

848	3	G	Teraz wywolam funkcje druga, a jej rezultat podstawie do innej kalibracji W funkcji fun_druga definiuje kalibracje i ma ona opis: WEWNETRZNA Obiekt chwilowy zwrocony jako rezultat funkcji ma opis WEWNETRZNA	Teraz wywolam funkcje druga, (ktora zwroci rezultat typu Tkalibracja) W funkcji fun_druga definiuje kalibracje i ma ona opis: WEWNETRZNA WEWNETRZNA ^------To opis obiektu zwroconego przez funkcje	Damian Rypel to jest zmiana kosmetyczna, ale myślę że korzystna
856	2	G	mieć swoje obiekty klasy string na pomieszczenie nazwisk i imion	mieć swoją tablicę na pomieszczenie nazwiska	Kamil Lepianka
872	1	G	[dodać akapit]	A jak z konstruktorem kopiującym? Standard mówi, że jeśli nie zdefiniujemy konstruktora kopiującego dla danej klasy, to kompilator zrobi to dla nas, pod warunkiem, że w klasie nie zdefiniowaliśmy konstruktora przenoszącego (lub przenoszącego operatora przypisania)	J.G.
872	8	G	K& K(const K &) = default;	K(const K &) = default;	Maciej Wroński
872	13	G	Nawe jeśli zdefiniujesz … kopiujący. Oczywiście	Jeśli kompilator wygeneruje Ci konstruktor kopiujący, to oczywiście	Miłosz Dominiak
885	4	D	parz	patrz	Kamil Lepianka
886	10	D	nie ma składników danych	nie ma składników-danych	Kamil Lepianka
887	6	G	składników danych	składników-danych	J.G.
891	17	G	Wansee :	Wansee:	Kamil Lepianka
892	27	G	po drugiej stronie	po prawej stronie	Kamil Lepianka
894	6	D	elementów NIEbędących agregatami	elementow NIEbedacych agregatami	Kamil Lepianka
894	25	G	elementów będących agregatami	elementow bedacych agregatami	Kamil Lepianka
898	11	D	że jej elementy będą	że jej obiekty będą	Kamil Lepianka
928	7	G	Rozważmy	Wyobraź sobie	Kamil Lepianka
928	13	D	składową klasa Tzespolona	składową klasy Tzespolona	Kamil Lepianka
930	14	G	przyjmujący jeden argument	przyjmujący argument	J.G.
930	27	G	konwertującym.	konwertującym. Standard C++11 nie wymaga, by ten konstruktor miał tylko jeden argument. Gdy ma więcej (np. w klamrach {…}), to mówimy wówczas, że konstruktor ten dokonuje konwersji z zestawu swoich arguentów → na typ K.	J.G.
933	4	D	2.5 (double).	2.55 (double).	Kamil Lepianka
947	2	D	kompilator	kompiltorowi	Kamil Lepianka
956	16	G	c) static_cast(objK)	c) static_cast<M*>(objK)	J.G.
969	12	G	K::operato@(K)	K::operator@(K)	Kamil Lepianka
977	13	G	pulpit + okienko	okienko + okienko	Tomasz Dąbrowski
1007	14	D	-5_pi ====> -(operator""_pi(15))	-5_pi ====> -(operator""_pi(5))	Kamil Lepianka
1011	12	D	k) -*	k) ->*	Miłosz Dominiak
1011	16	D	m) -	m) ->	J.G.
1015	13	G	będzie na pewno lwartością	będzie na pewno obiektem	Tomasz Dąbrowski
1026	11	D	(this == &wzor)	(this != &wzor)	Piotr Kędra
1044	14	G	char operator() ()	void operator() ()	Kamil Lepianka
1051	8	D	ile razy następuje jest	ile razy następuje	Piotr Kędra
1053	23	D	int ile_wypisac = min(ile_ostatnich, pamietnik.size());	long int ile_wypisac = ile_ostatnich < pamietnik.size() ? ile_ostatnich : pamietnik.size() ;	Kamil Lepianka
1065	23	G	argumenty użytkownika...) ;	argumenty użytkownika...) noexcept;	Michał Gręboszz
1070	20	G	Jej definicja	Jego definicja	J.G.
1134	19 30	G G	wysyłane do funkcji	wysylane do funkcji	Kamil Lepianka
1137	14	D	nonexcept	noexcept	Borys Cywiński
1143	5	G	const.T o sprawi, że	const.To sprawi, że	Michał Gręboszz
1145	25	G	wszystkie publiczne	wszystkie prywatne	Kamil Lepianka
1152	8	D	mocy jest deklaracja	mocy jest dyrektywa	Michał Gręboszz
1166	10	G	<match>	<math>	J.G.
1200	2	G	delete tabl_gps;	delete [ ] tabl_gps;	J.G.
1216	8	D	wystarczyć	wystarczy	Tomasz Dąbrowski
1236	2	G	odebrać jako wskaźnik	odebrać stosując zapis wskaźnikowy	J.G.
1236	4	G	void fun1(K ttt*);	void fun1(K *ttt);	Hanna Feruś
1265	19	D	#include "instrum.h"	#include "Tinstrument.h"	Kamil Lepianka
1272	10	D	kompilacji	kompilacji/linkowania	Tomasz Dąbrowski
1291	16	G	znak hasz (#)	znak procentu (%)	Kamil Lepianka
1293	12	G	pięć zdarzeń	sześć zdarzeń	Kamil Lepianka

1293	21	G	Tkomora wypisuje na ekranie gwiazdkę, funkcja z klasy Tscyntylator – procent, a z klasy Tscyntylator_XY – znaczek hasz	Tkomora wypisuje na ekranie procent, funkcja z klasy Tscyntylator – znaczek hasz, a z klasy Tscyntylator_XY – gwiazdkę.	Kamil Lepianka
1298	12	D	virtual void funkcja	virtual void funkcja()	Michał Gręboszc
1313	2	G	pytamyŁ	pytamy:	Kamil Lepianka
1335	5	G	reinterpret_cast	static_cast	Tomasz Dąbrowski
1335	14	G	reinterpret_cast	static_cast	Tomasz Dąbrowski
1353	13	G	manipulatorem width	manipulatorem setw	Kamil Lepianka
1356	15	G	wykładnik jest większy	wykładnik jest większy lub równy	Tomasz Dąbrowski
1375	6	D	ignore(int, int);	ignore(streamsize = 1, int = EOF);	Tomasz Dąbrowski
1377	5	D	<< (int) znak	<< znak	Tomasz Dąbrowski
1377	7	D	<< znak << ", kod: " << (int) znak	<< (char) znak << ", kod: " << znak	Tomasz Dąbrowski
1377	9	D	char znak;	int znak;	Tomasz Dąbrowski
1377	18	G	EOF zostanie wpisany	EOF nie zostanie wpisany	Tomasz Dąbrowski
1392	19	G	ios::ws i ios::skipws	std::ws i std::skipws	Tomasz Dąbrowski
1392	23	G	ios::scientific	std::scientific	Tomasz Dąbrowski
1406	8	D	clean	clear	Tomasz Dąbrowski
1423	22	G	㊽	㊼	Kamil Lepianka
1423	29	G	㊼	㊽	Kamil Lepianka
1424	22	G	void exceptions(io_state flagi…	void exceptions(iostate flagi…	Tomasz Dąbrowski
1424	35	G	ios_state exceptions() const;	ios_base::iostate exceptions() const;	Tomasz Dąbrowski
1442	14	G	wskaźnik do pisania	wskaźnik od czytania	Andrzej Rzoska
1445	17	G	void str(const string & nowa_treść) const;	void str(const string & nowa_treść);	Andrzej Rzoska
1450	18	G	0.01, 0.02, 0.03	0.00, 0.01, 0.02, 0.03	Tomasz Dąbrowski
1451	6	D	void str(const string & nowa_treść) const;	void str(const string & nowa_treść);	Andrzej Rzoska
1460	5	G	(w ㉑)		Kamil Lepianka
1462	27	G	program 2.27 -port 721	program 6.33 -port 721	Kamil Lepianka
1464	18	G	void str(const string & nowa_treść) const;	void str(const string & nowa_treść);	J.G.
1468	17	D	if(pp >> x) break;	if(!(pp >> x)) break;	Andrzej Rzoska
1509	19	D	while(!plik_wy.fail())	while(!plik_wej.fail())	Tomasz Dąbrowski
1512	4	G	cout << "slot jest zajety\n\aWcisnij ENTER... " ;	cout << "slot jest zajety\n\aWcisnij znak C i ENTER... " ; char znak; cin >> znak; return;	Tomasz Dąbrowski
1540	10	D	SFINEA	SFINAE	Damian Werpachowski
1547	15	G	void f_poza_cialem()	void ladownik<Typ, STALA, ADR_OBJ, WSK, REF, WSKOBJ, ADR_FUN, ADRES_DANEJ_STAT, WSK_SKLADNIKA, SZAB, POJEMNIK>::f_poza_cialem()	J.G.