

Symfonia C++ standard : programowanie w języku C++ orientowane obiektowo. T. 2 / Jerzy Grębosz. – Wyd. 3 C popr. – Kraków, 2015

Spis treści

20 Struktury, Unie, Pola bitowe	613
20.1 Struktura	613
20.2 Unia	614
20.2.1 Inicjalizacja unii	616
20.2.2 Unia anonimowa	616
20.3 Pola bitowe	618
20.4 Unia i pola bitowe - upraszczają rozpakowanie słów	622
20.5 Ćwiczenia	629
21 Klasa zagnieżdżona lub lokalna	631
21.1 Zagnieżdżona definicja klasy	631
21.2 Praktyczny przykład zagnieżdżenia definicji klasy	635
21.3 Lokalna definicja klasy	647
21.4 Lokalne nazwy typów	650
21.5 Ćwiczenia	650
22 Konstruktory i Destruktory	653
22.1 Konstruktor	653
22.1.1 Przykład programu zawierającego klasę z konstruktorami	654
22.2 Specyfikator (przydomek) <i>explicit</i>	666
22.3 Kiedy i jak wywoływany jest konstruktor	666
22.3.1 Konstruowanie obiektów lokalnych	667
22.3.2 Konstruowanie obiektów globalnych	667
22.3.3 Konstrukcja obiektów tworzonych operatorem <i>new</i>	667
22.3.4 Jawne wywołanie konstruktora	668
22.3.5 Dalsze sytuacje, gdy pracuje konstruktor	671
22.4 Destruktor	671
22.5 Konstruktor domniemany	673
22.6 Lista inicjalizacyjna konstruktora	674
22.7 Konstrukcja obiektu, którego składnikiem jest obiekt innej klasy	677
22.8 Konstruktory nie-publiczne ?	683
22.9 Konstruktor kopiujący (albo inicjalizator kopiujący)	685
22.9.1 Przykład klasy z konstruktorem kopiującym	687
22.9.2 Dlaczego przez referencję?	693
22.9.3 Jak dostać piątkę z C++ ?	694
22.9.4 Konstruktor kopiujący gwarantujący nietykliwość	695
22.9.5 Współodpowiedzialność	696
22.9.6 Konstruktor kopiujący generowany automatycznie	696

22.9.7 Kiedy konstruktor kopiujący jest niezbędny?	697
22.10 Ćwiczenia	702
23 Tablice obiektów	706
23.1 Tablica obiektów definiowana operatorem <i>new</i>	707
23.2 Inicjalizacja tablic obiektów	709
23.2.1 Inicjalizacja tablic obiektów będących agregatami	709
23.2.2 Inicjalizacja tablic nie będących agregatami	713
23.2.3 Inicjalizacja tablic tworzonych w zapasie pamięci	716
23.3 Ćwiczenia	716
24 Wskaźnik do składników klasy	718
24.1 Wskaźniki zwykłe - repetytorium	718
24.2 Wskaźnik do pokazywania na składnik-daną	719
24.2.1 Przykład zastosowania wskaźników do składników klasy	723
24.3 Wskaźnik do funkcji składowej	729
24.3.1 Zastosowanie wskaźników do funkcji składowych	731
24.4 Tablica wskaźników do danych składowych klasy	737
24.5 Tablica wskaźników do funkcji składowych klasy	738
24.6 Wskaźniki do składników statycznych	741
24.7 Ćwiczenia	742
25 Konwersje definiowane przez użytkownika	744
25.1 Sformułowanie problemu	744
25.2 Konstruktory konwertujące	746
25.2.1 Kiedy jawnie, kiedy niejawnie	748
25.2.2 Przykład konwersji konstruktorem	752
25.3 Funkcja konwertująca - operator konwersji	754
25.3.1 Na co konwertować nie można	759
25.4 Który wariant konwersji wybrać?	760
25.5 Sytuacje, w których zachodzi konwersja	762
25.6 Zapis jawnego wywołania konwersji typów	763
25.6.1 Advocatus zapisu przypominającego: „wywołanie funkcji”	764
25.6.2 Advocatus zapisu: „rzutowanie”	764
25.7 Niecałkiem pasujące argumenty, czyli konwersje przy dopasowaniu	765
25.8 Kilka rad dotyczących konwersji	769
25.9 Ćwiczenia	770
26 Przeładowanie operatorów	773
26.1 Przeładowanie operatorów - definicja i trochę teorii	775
26.2 Moje zabawki	779
26.3 Funkcja operatorowa jako funkcja składowa	780
26.4 Funkcja operatorowa nie musi być przyjacielem klasy	783
26.5 Operatory predefiniowane	784
26.6 Argumentowość operatorów	784

26.7 Operatory jednoargumentowe	785
26.8 Operatory dwuargumentowe	787
26.8.1 Przykład na przeładowanie operatora dwuargumentowego	788
26.8.2 Przemienność	789
26.8.3 Choć operatory inne, to nazwę mają tę samą	791
26.9 Przykład zupełnie nie matematyczny	791
26.10 Cztery operatory, które muszą być niestatycznymi funkcjami składowymi	803
26.11 Operator przypisania =	803
26.11.1 Przykład na przeładowanie operatora przypisania	805
26.11.2 Jak konieczność istnienia operatora przypisania - odpowiedź potocznie?	814
26.11.3 Kiedy operator przypisania nie jest generowany automatycznie	815
26.12 Operator []	816
26.13 Operator ()	820
26.14 Operator ->	822
26.14.1 „Sprytny wskaźnik” - wykorzystuje przeładowanie właśnie tego operatora	824
26.15 Operatory <i>new</i> , <i>new[]</i>	830
26.15.1 Przykład przeładowania operatora <i>new</i>	831
26.15.2 Przykład przeładowania operatora <i>new []</i>	833
26.16 Operatory <i>delete</i> , <i>delete []</i>	833
26.16.1 Prosty przykład przeładowania <i>delete</i>	834
26.16.2 Prosty przykład przeładowania <i>delete []</i>	834
26.17 Program przykładowy na zastosowanie operatorów <i>new</i> , <i>delete</i>	835
26.18 Przeładowanie globalnych operatorów <i>new</i> , <i>new[]</i> , <i>delete</i> , <i>delete[]</i>	839
26.19 Operatory postinkrementacji i postdekrementacji, czyli koniec z niesprawiedliwością	840
26.20 Rady praktyczne dotyczące przeładowania	843
26.21 Pojedynek: Operator jako funkcja składowa, czy globalna?	844
26.22 Zasłona spada, czyli tajemnica operatora «	846
26.23 Rzut oka wstecz	852
26.24 Ćwiczenia	853
27 Dziedziczenie klas	857
27.1 Istota dziedziczenia	857
27.2 Dostęp do składników	860
27.2.1 Prywatne składniki klasy podstawowej	860
27.2.2 Nieprywatne składniki klasy podstawowej	862
27.2.3 Klasa pochodna też decyduje	863
27.2.4 Deklaracja dostępu <i>using</i> - czyli udostępnianie wybiórcze	865
27.3 Czego się nie dziedziczy	868
27.3.1 "Nie dziedziczenie" konstruktorów	868
27.3.2 "Nie dziedziczenie" operatora przypisania	869
27.3.3 "Nie dziedziczenie" destruktora	869

27.4	Drzewo genealogiczne	870
27.5	Dziedziczenie - doskonałe narzędzie programowania	871
27.6	Kolejność wywoływania konstruktorów	873
27.7	Przypisanie i inicjalizacja obiektów w warunkach dziedziczenia	879
27.7.1	Klasa pochodna nie definiuje swojego operatora przypisania	879
27.7.2	Klasa pochodna nie definiuje swojego konstruktora kopiującego	880
27.7.3	Inicjalizacja i przypisywanie według obiektu wzorcowego będącego <i>const</i>	881
27.7.4	Definiowanie konstruktora kopiującego i operatora przypisania dla klasy pochodnej	881
27.8	Dziedziczenie od kilku „rodziców” (wielodziedziczenie)	887
27.8.1	Konstruktor klasy pochodnej przy wielodziedziczeniu	889
27.8.2	Ryzyko wieloznaczności przy dziedziczeniu	892
27.8.3	Czy bliższe pokrewieństwo usuwa wieloznaczność?	894
27.8.4	Poszlaki	894
27.9	Pojedynek: Dziedziczenie klasy, contra zawieranie obiektów składowych	895
27.10	Konwersje standardowe przy dziedziczeniu	897
27.10.1	Panorama korzyści	901
27.10.2	Czego robić się nie opłaca	903
27.10.3	Tuzin samochodów nie jest rodzajem tuzina pojazdów	904
27.10.4	Konwersje standardowe wskaźnika do składnika klasy	908
27.11	Wirtualne klasy podstawowe	910
27.11.1	Publiczne i prywatne dziedziczenie tej samej klasy wirtualnej	914
27.11.2	Uwagi o konstrukcji i inicjalizacji w przypadku klas wirtualnych	914
27.11.3	Dominacja klas wirtualnych	918
27.12	Ćwiczenia	919
28	Wirtualne funkcje składowe	925
28.1	Polimorfizm	931
28.2	Typy rezultatów różnych realizacji funkcji wirtualnej	934
28.3	Dalsze szczegóły	937
28.4	Wczesne i późne wiązanie	939
28.5	Kiedy dla wywołań funkcji wirtualnych, mimo wszystko, zachodzi wczesne wiązanie?	941
28.6	Kulisy białej magii, czyli: Jak to jest zrobione?	942
28.7	Funkcja wirtualna, a mimo to <i>inline</i>	943
28.8	Pojedynek - funkcje przeładowane contra funkcje wirtualne	944
28.9	Klasy abstrakcyjne	945
28.10	Destruktor? to najlepiej wirtualny!	952
28.11	Co prawda, konstruktor nie może być wirtualny, ale...	957
28.12	Rzutowanie <i>dynamic_cast</i> jest dla typów polimorficznych	962
28.13	Wszystko co najważniejsze	966
28.14	Finis coronat opus	968
28.15	Ćwiczenia	968

29 Operacje Wejścia/Wyjścia	972
29.1 Biblioteka <code>iostream</code>	973
29.2 Strumień	973
29.3 Strumienie zdefiniowane standardowo	975
29.4 Operatory <code>»</code> i <code>«</code>	976
29.5 Domniemania w pracy strumieni zdefiniowanych standardowo	977
29.6 Uwaga na priorytet	980
29.7 Operatory <code>«</code> oraz <code>»</code> definiowane przez użytkownika	981
29.7.1 Operatorów wstawiania i wyjmowania ze strumienia - nie dziedziczy się	986
29.7.2 Operatory wstawiania i wyjmowania nie mogą być wirtualne. Niestety	987
29.8 Sterowanie formatem	989
29.9 Flagi stanu formatowania	989
29.9.1 Znaczenie poszczególnych flag sterowania formatem	992
29.10 Sposoby zmiany trybu (reguł) formatowania	997
29.10.1 Zmiana sposobu formatowania funkcjami <code>setf</code> , <code>unsetf</code>	999
29.10.2 Dodatkowe funkcje do zmiany parametrów formatowania	1003
29.11 Manipulatory	1009
29.11.1 Manipulatory bezargumentowe	1010
29.11.2 Manipulatory parametryzowane	1015
29.11.3 Definiowanie swoich manipulatorów	1019
29.11.4 Manipulator jako funkcja	1019
29.11.5 Definiowanie manipulatora z parametrem	1021
29.12 Nieformatowane operacje wejścia/wyjścia	1025
29.13 Omówienie funkcji wyjmujących ze strumienia	1027
29.13.1 Funkcje do pracy ze znakami i napisami	1027
29.13.2 Wczytywanie binarne - funkcje <code>read</code> i <code>readsome</code>	1033
29.13.3 Funkcja <code>ignore</code>	1035
29.13.4 Pożyteczne funkcje pomocnicze	1036
29.13.5 Funkcje wstawiające do strumienia	1038
29.14 Strumienie płynące do lub od plików	1040
29.14.1 Otwieranie i zamykanie strumienia	1042
29.15 Błędy w trakcie pracy strumienia	1048
29.15.1 Flagi stanu błędu strumienia	1048
29.15.2 Funkcje do pracy na flagach błędu	1049
29.15.3 Kilka udogodnień dla sprawdzania poprawności	1050
29.15.4 Ustawianie i kasowanie flag błędu strumienia	1051
29.15.5 Trzy plagi - czyli „gotowiec”, jak radzić sobie z błędami	1056
29.16 Przykład programu pracującego na plikach	1059
29.17 Strumienie, a technika rzucania wyjątków	1062
29.18 Wybór miejsca czytania lub pisania w pliku	1066
29.18.1 Funkcje składowe informujące o pozycji wskaźników	1067
29.18.2 Wybrane funkcje składowe do pozycjonowania wskaźników	1067

29.19	Pozycjonowanie w przykładzie większego programu	1071
29.20	Tie - harmonijna praca dwóch strumieni	1077
29.21	Dlaczego tak nie lubimy biblioteki <i>stdio</i> ?	1079
29.22	Synchronizacja biblioteki <i>iostream</i> z biblioteką <i>stdio</i>	1081
29.23	Strumień zapisujący do obiektu klasy <i>string</i>	1082
29.23.1	Program przykładowy ilustrujący użycie klasy <i>ostream</i>	1086
29.24	Strumień czytający z obiektu klasy <i>string</i>	1090
29.24.1	Prosty przykład użycia strumienia <i>istream</i>	1092
29.24.2	Wczytywanie argumentów wywoływania programu	1097
29.25	Ożenek: strumień <i>stringstream</i> - czytający i zapisujący do stringu	1100
29.25.1	Przykładowy program posługujący się klasą <i>stringstream</i>	1101
29.26	Ćwiczenia	1103
30	Projektowanie programów orientowanych obiektowo	1111
30.1	Przegląd kilku technik programowania	1112
30.1.1	Programowanie liniowe	1112
30.1.2	Programowanie proceduralne (czyli "orientowane funkcyjnie")	1112
30.1.3	Programowanie z ukrywaniem danych	1113
30.1.4	Programowanie obiektowe - programowanie „bazujące” na obiektach	1113
30.1.5	Programowanie Obiektowo Orientowane (OO)	1113
30.2	O wyższości programowania obiektowo orientowanego nad Świętami Wielkiej Nocy	1114
30.3	Obiektowo Orientowane: Projektowanie	1116
30.4	Praktyczne wskazówki dotyczące projektowania programu techniką OO	1118
30.4.1	Rekonesans - czyli rozpoznanie zagadnienia	1118
30.4.2	Faza projektowania	1119
30.4.3	Etap 1. Identyfikacja zachowań systemu	1120
30.4.4	Etap 2: Identyfikacja obiektów (klas obiektów)	1120
30.4.5	Etap 3: Usystematyzowanie klas obiektów	1122
30.4.6	Etap 4: Określenie wzajemnych zależności klas	1123
30.4.7	Etap 5: Składanie modelu. Określanie sekwencji działań obiektów i cykli życiowych	1125
30.5	Faza implementacji	1126
30.6	Przykład projektowania	1127
30.7	Faza: Rozpoznanie naszego zagadnienia	1127
30.8	Faza: Projektowanie	1131
30.8.1	Etap 1 - Identyfikacja zachowań naszego systemu	1131
30.8.2	Etap 2 - Identyfikacja klas obiektów, z którymi mamy do czynienia	1132
30.8.3	Etap 3 - Usystematyzowanie klas obiektów z występujących w naszym systemie	1135
30.8.4	Etap 4 - Określenie wzajemnych zależności klas	1136
30.8.5	Etap 5 - Składamy model naszego systemu	1138

30.9 Implementacja modelu naszego systemu	1142
30.10 Symfonia C++, Coda	1149
30.11 Posłowie	1149
A Dodatek: Systemy liczenia	1151
A.1 Dlaczego komputer nie liczy tak jak my?	1151
A.2 System szesnastkowy (heksadecymalny)	1156
A.3 Ćwiczenia	1159
B Dodatek: Sprzężenie zwrotne – podziękowania	1160

oprac. BPK