Spis treści

0 Proszę nie czytać tego ! 1

1 Startujemy! 7
1.1 Pierwszy program 7
1.2 Drugi program 13
1.3 Ćwiczenia 19

2 Instrukcje sterujące 22
2.1 Prawda – Fałsz, czyli o warunkach 22
2.1.1 Wyrażenie logiczne 22
2.1.2 Zmienne logiczne `bool` jako warunek 23
2.1.3 Stare dobre sposoby z dawnego C++ 24
2.2 Instrukcja warunkowa `if` 25
2.3 Pętla `while` 29
2.4 Pętla `do...while...` 30
2.5 Pętla `for` 31
2.6 Instrukcja `switch` 34
2.7 Co wybrać: `switch` czy `if...else`? 37
2.8 Instrukcja `break` 40
2.9 Instrukcja `goto` 41
2.10 Instrukcja `continue` 43
2.11 Klamry w instrukcjach sterujących 44
2.12 Ćwiczenia 45

3 Typy 48
3.1 Deklaracje typów 48
3.2 Systematyka typów z języka C++ 50
3.3 Typy fundamentalne 50
3.3.1 Definiowanie obiektów „w biegu“. 56
3.4 Stałe dosłowne 58

SYNTEKA C++ STANDARD

3.4.1 Stałe będące liczbami całkowitymi 59
3.4.2 Stałe reprezentujące liczby zmiennoprzecinkowe 61
3.4.3 Stałe znakowe 62
3.4.4 Stałe tekstowe, napisy, albo po prostu stringi 65
3.5 Typy złożone 68
3.6 Typ `void` 70
3.7 Zakres ważności nazwy obiektu, a czas życia obiektu 70
3.7.1 Zakres: lokalny 71
3.7.2 Zakres: blok funkcji 71
3.7.3 Zakres: obszar pliku 72
3.7.4 Zakres: obszar klasy 72
3.7.5 Zakres określony przez przestrzeń nazw 72
3.8 Zasłanianie nazw 78
3.9 Specyfikator (przydomek) `const` 80
3.9.1 Pojedynek: `const` contra `#define` 82
3.10 Obiekty `register` 84
3.11 Specyfikator `volatile` 84
3.12 Instrukcja `typedef` 85
3.13 Typy wyliczeniowe `enum` 87
3.14 Ćwiczenia 91

4 Operatory 95

4.1 Operatory arytmetyczne 95
4.1.1 Operator %, czyli reszta z dzielenia (modulo) 96
4.1.2 Jednoargumentowe operatory + i – 98
4.1.3 Operatory inkrementacji i dekrementacji 98
4.1.4 Operator przypisania = ...100
4.2 Operatory logiczne 101
4.2.1 Operatory relacji 101
4.2.2 Operatory sumy logicznej `||` i iloczynu logicznego `&&` 103
4.2.3 Wykrzyknik ! – czyli operator negacji 104
4.3 Operatory bitowe 105
4.3.1 Przesunięcie w lewo << 106
4.3.2 Przesunięcie w prawo >> 107
4.3.3 Bitowe operatory sumy, iloczynu, negacji, różnicy symetrycznej 108
4.4 Różnica między operatorami logicznymi, a operatorami bitowymi 109
4.5 Pozostałe operatory przypisania 110
4.6 Wyrażenie warunkowe 111
4.7 Operator `sizeof` 113
4.8 Operatory rzutowania 114
4.8.1 Rzutowanie według tradycyjnych (nie zalecanych) sposobów 115
4.8.2 Rzutowanie za pomocą nowych operatorów rzutowania 116
4.8.3 Operator `static_cast` 117
4.8.4 Operator `const_cast` 119
4.8.5 Operator `dynamic_cast` 121
4.8.6 Operator `reinterpret_cast` 121

SYMFONIA C++ STANDARD

4.9 Operator: przecinek 123
4.10 Priorytety operatorów 123
4.11 Łączność operatorów 126
4.12 Ćwiczenia 127

5 Funkcje 132
5.1 Funkcja często wywołuje inną funkcję 135
5.2 Zwracanie rezultatu przez funkcję 136
5.3 Stos 140
5.4 Przesyłanie argumentów do funkcji przez wartość 140
5.5 Przesyłanie argumentów przez referencję 142
5.6 Kiedy deklaracja funkcji nie jest konieczna? 145
5.7 Argumenty domniemane 147
5.7.1 Ciekawostki na temat argumentów domniemanych 150
5.8 Nienazwany argument 156
5.9 Funkcje `inline` (w linii) 157
5.10 Przypomnienie o zakresie ważności nazw deklarowanych wewnątrz funkcji 161
5.11 Wybór zakresu ważności nazwy i czasu życia obiektu 162
5.11.1 Obiekty globalne 162
5.11.2 Obiekty automatyczne 163
5.11.3 Obiekty lokalne statyczne 164
5.12 Funkcje w programie składającym się z kilku plików 168
5.12.1 Nazwy statyczne globalne 173
5.13 Funkcje rekurencyjne 174
5.14 Funkcje biblioteczne 186
5.15 Ćwiczenia 189

6 Preprocesor 195
6.1 Na pomoc rodakom 195
6.2 Dyrektywa `#define` 197
6.3 Dyrektywa `#undef` 200
6.4 Makrodefinicje 200
6.5 Sklejacz nazw, czyli operator ## 203
6.6 Zamiana parametru aktualnego makrodefinicji na string 204
6.7 Dyrektywy kompilacji warunkowej 205
6.8 Dyrektywa `#error` 210
6.9 Dyrektywa `#line` 210
6.10 Wstawianie treści innych plików w tekst kompilowanego właśnie pliku 211
6.11 Dyrektywa pusta 213
6.12 Dyrektywy zależne od implementacji 213
6.13 Nazwy predefiniowane 214
6.14 Ćwiczenia 216

7 Tablice 219
7.1 Elementy tablicy 220
7.2 Inicjalizacja tablic 223
7.3 Przekazywanie tablicy do funkcji 224

SYMFONIA C++ STANDARD

7.4 Przykład z tablicą elementów typu `enum` ... 228
7.5 Tablice znakowe ... 230
7.6 Tablice wielowymiarowe ... 240
7.6.1 Typ wyrażeń związanych z tablicą wielowymiarową ... 243
7.6.2 Przesyłanie tablic wielowymiarowych do funkcji ... 245
7.7 Ćwiczenia ... 247

8 Wskaźniki ... 253
8.1 Wskaźniki mogą bardzo ułatwić życie ... 253
8.2 Definiowanie wskaźników ... 255
8.3 Praca ze wskaźnikiem ... 257
8.4 L–wartość ... 260
8.5 Operator rzutowania `reinterpret_cast`, a wskaźniki ... 261
8.6 Wskaźniki typu `void` ... 264
8.7 Cztery domeny zastosowania wskaźników ... 267
8.8 Zastosowanie wskaźników wobec tablic ... 267
8.8.1 Ćwiczenia z mechaniki ruchu wskaźnika ... 267
8.8.2 Użycie wskaźnika w pracy z tablicą ... 272
8.8.3 Arytmetyka wskaźników ... 276
8.8.4 Porównywanie wskaźników ... 279
8.8.5 Wskaźnik można porównać z adresem 0 ... 281
8.9 Zastosowanie wskaźników w argumentach funkcji ... 282
8.9.1 Jeszcze raz o przesyłaniu tablic do funkcji ... 286
8.9.2 Odbieranie tablicy jako wskaźnika ... 286
8.9.3 Argument formalny będący wskaźnikiem do obiektu `const` ... 288
8.10 Zastosowanie wskaźników przy dostępie do konkretnych komórek pamięci ... 292
8.11 Rezerwacja obszarów pamięci ... 293
8.11.1 Operatory `new` i `delete` albo Oratorium Stworzenie Świata ... 294
8.11.2 Dynamiczna alokacja tablicy ... 298
8.11.3 Tablice wielowymiarowe tworzone operatorem `new` ... 299
8.11.4 Umiejscawiający operator `new` ... 301
8.11.5 "Przychodzimy, odchodzimy – cichuteńko, na..." ... 306
8.11.6 Zapas pamięci to nie jest studnia bez dna ... 309
8.11.7 Funkcja `set_new_handler` ... 313
8.11.8 Pojedynek: `new` contra `malloc` ... 315
8.12 Stałe wskaźniki ... 315
8.13 Stałe wskaźniki, a wskaźniki do stałych ... 316
8.14 Strzał na oślep – Wskaźnik zawsze pokazuje na coś ... 318
8.15 Sposoby ustawiania wskaźników ... 319
8.16 Parada kłamców, czyli o rzutowaniu `const_cast` ... 321
8.17 Tablice wskaźników ... 326
8.18 Wariacje na temat C-stringów ... 328
8.19 Wskaźniki do funkcji ... 337
8.19.1 Ćwiczenia z definiowania wskaźników do funkcji ... 341
8.19.2 Wskaźnik do funkcji jako argument innej funkcji ... 347
8.19.3 Tablica wskaźników do funkcji ... 358

SZKOŁA C++ STANDARD

8.20 Argumenty z linii wywołania programu 363
8.21 Ćwiczenia....366

9 Przeładowanie nazwy funkcji 374
9.1 Co to znaczy: przeładowanie 374
9.2 Bliższe szczegóły przeładowania 378
9.3 Czy przeładowanie nazw funkcji jest techniką obiektowo orientowaną? 381
9.4 Linkowanie z modułami z innych języków 382
9.5 Przeładowanie, a zakres ważności deklaracji funkcji 383
9.6 Rozważania o identyczności lub odmienności typów argumentów 386
9.6.1 Przeładowanie, a `typedef` i `enum` 386
9.6.2 Tablica, a wskaźnik....386
9.6.3 Pewne szczegóły o tablicach wielowymiarowych 388
9.6.4 Przeładowanie, a referencja 390
9.6.5 Identyczność typów: T, const T, volatile T....391
9.6.6 Przeładowanie – a typy: T*, volatile T*, const T* 392
9.6.7 Przeładowanie – a typy: T&, volatile T&, const T&....394
9.7 Adres funkcji przeładowanej 394
9.7.1 Zwrot rezultatu będącego adresem funkcji przeładowanej....397
9.8 Kulisy dopasowywania argumentów do funkcji przeładowanych 398
9.9 Etapy dopasowania 400
9.9.1 Etap 1. Dopasowanie dokładne....401
9.9.2 Etap 1a. Dopasowanie dokładne, ale z tzw. trywialną konwersją 401
9.9.3 Etap 2. Dopasowanie z awansem (z promocją) 402
9.9.4 Etap 3. Próba dopasowania za pomocą konwersji standardowych 405
9.9.5 Etap 4. Próba dopasowania z użyciem konwersji zdefiniowanych przez użytkownika. 406
9.9.6 Etap 5. Próba dopasowania do funkcji z wielokropkiem 407
9.9.7 Wskaźników nie dopasowuje się inaczej niż dosłownie 407
9.10 Dopasowywanie wywołań z kilkoma argumentami....408
9.11 Ćwiczenia....409

10 Klasy....412
10.1 Typy definiowane przez użytkownika 412
10.2 Składniki klasy 414
10.3 Składnik będący obiektem 416
10.4 Enkapsulacja 416
10.5 Ukrywanie informacji 418
10.6 Klasa, a obiekt 421
10.7 Funkcje składowe 424
10.7.1 Posługiwanie się funkcjami składowymi....424
10.7.2 Definiowanie funkcji składowych 425
10.8 Jak to właściwie jest ? (`this`) 431
10.9 Odwołanie się do publicznych danych składowych 434
10.10 Zasłanianie nazw 435
10.10.1 Nie sięgaj z klasy do obiektów globalnych....439

SYMFONIA C++ STANDARD

10.11 Przeładowanie i zasłonięcie równocześnie 439
10.12 Nowa klasa? Osobny plik! 440
10.13 Przesyłanie do funkcji argumentów będących obiektami 452
10.13.1 Przesyłanie obiektu przez wartość 453
10.13.2 Przesyłanie przez referencję 455
10.14 Konstruktor – pierwsza wzmianka 457
10.15 Destruktor – pierwsza wzmianka 462
10.16 Składnik statyczny 465
10.16.1 Deklaracja składnika statycznego połączona z inicjalizacją 470
10.17 Statyczna funkcja składowa 476
10.18 Do czego może nam się przydać składnik statyczny w klasie? 480
10.19 Funkcje składowe typu `const` oraz `volatile` 480
10.19.1 Przeładowanie, a funkcje składowe `const` i `volatile` 485
10.20 Specyfikator `mutable` 485
10.21 Ćwiczenia 499

11 Biblioteczna klasa `std::string` do operacji z tekstami 503
11.1 Przykład programu z użyciem klasy `string` 505
11.2 Definiowanie obiektów klasy `string` 511
11.3 Użycie operatorów =, +, +=, w pracy ze stringami 516
11.3.1 Jak umieścić w tekście liczbę? 517
11.4 Pojemność, rozmiar i długość stringu 519
11.4.1 Funkcje `size()` i `length()` 519
11.4.2 Funkcja składowa `empty` 520
11.4.3 Funkcja składowa `max_size` 521
11.4.4 Funkcja składowa `capacity` 521
11.4.5 Funkcja składowa `reserve` 523
11.4.6 `resize` – zmiana długości stringu „na siłę” 524
11.4.7 Funkcja składowa `clear` 527
11.5 Użycie operatora `[]` oraz funkcji `at` 527
11.5.1 Działanie operatora `[]` 528
11.5.2 Działanie funkcji składowej `at` 529
11.6 Praca z fragmentem stringu, czyli z sub–stringiem 532
11.7 Funkcja składowa `substr` 533
11.8 Szukanie zadanego substringu w obiekcie klasy `string` – funkcja `find` i jej pokrewne 534
11.9 Szukanie rozpoczynane od końca stringu 538
11.10 Szukanie w stringu jednego ze znaków z zadanego zestawu 539
11.11 Usuwanie znaków ze stringu – funkcje `erase` 542
11.12 Wstawianie znaków do już istniejącego stringu – funkcje `insert` 544
11.13 Zamiana części znaków na inne znaki – `replace` 547
11.14 Zamiana zawartości obiektu klasy `string` na C-string 552
11.15 Zaglądanie do wnętrza obiektu klasy `string` funkcją `data` 555
11.16 W porządku alfabetycznym – czyli porównywanie stringów 557
11.16.1 Porównywanie stringów funkcjami `compare` 557
11.16.2 Porównywanie stringów przy użyciu operatorów ==, !=, <, >, <=, >=. 563

SYMFONIA C++ STANDARD

11.17 Zamiana treści stringu na małe (lub wielkie) litery....565
11.18 Kopiowanie treści obiektu klasy `string` do wybranej tablicy znakowej – funkcja `copy` 572
11.19 Wzajemna zamiana treści dwóch obiektów klasy `string` – funkcja `swap`....573
11.20 Przypisanie do obiektu klasy `string`, funkcja `assign`....573
11.21 Dopisywanie do końca stringu za pomocą funkcji `append`....576
11.22 Wczytywanie z klawiatury długiego stringu o nieznanej wcześniej długości – `getline`....577
11.22.1 Pułapka – czyli jak `getline` może Cię zaskoczyć....581
11.23 Iteratory stringu....586
11.23.1 Iterator do obiektu stałego....590
11.23.2 Funkcje składowe klasy `string` pracujące z iteratorami....592
11.24 Bryk – czyli "pamięć zewnętrzna" programisty....599
11.25 Ćwiczenia....608

12 Deklaracje przyjaźni....615
12.0.1 Klasy zaprzyjaźnione....625
12.0.2 Słowo o zakresie....628

13 Struktury, Unie, Pola bitowe....629
13.1 Struktura....629
13.2 Unia....630
13.2.1 Inicjalizacja unii....632
13.2.2 Unia anonimowa....633
13.3 Pola bitowe....634
13.4 Unia i pola bitowe – upraszczają rozpakowanie słów....639
13.5 Ćwiczenia....647

14 Klasa zagnieżdżona lub lokalna....649
14.1 Zagnieżdżona definicja klasy....649
14.2 Lokalna definicja klasy....654
14.3 Lokalne nazwy typów....657
14.4 Ćwiczenia....658

15 Konstruktory i Destruktory....660
15.1 Konstruktor....660
15.1.1 Przykład programu zawierającego klasę z konstruktorami....661
15.2 Specyfikator (przydomek) `explicit`....675
15.3 Kiedy i jak wywoływany jest konstruktor....675
15.3.1 Konstruowanie obiektów lokalnych....675
15.3.2 Konstruowanie obiektów globalnych....676
15.3.3 Konstrukcja obiektów tworzonych operatorem `new`....677
15.3.4 Jawne wywołanie konstruktora....678
15.3.5 Dalsze sytuacje, gdy pracuje konstruktor....680
15.4 Destruktor....681
15.5 Konstruktor domniemany....683
15.6 Lista inicjalizacyjna konstruktora....684
15.7 Konstrukcja obiektu, którego składnikiem jest obiekt innej klasy....688

SYMFONIA C++ STANDARD

15.8 Konstruktory nie-publiczne ? 695
15.9 Konstruktor kopiujący (albo inicjalizator kopiujący) 697
15.9.1 Przykład klasy z konstruktorem kopiującym 699
15.9.2 Dlaczego przez referencję? 706
15.9.3 Jak dostać piątkę z C++ ? 707
15.9.4 Konstruktor kopiujący gwarantujący nietykalność 709
15.9.5 Współodpowiedzialność 710
15.9.6 Konstruktor kopiujący generowany automatycznie 710
15.9.7 Kiedy konstruktor kopiujący jest niezbędny? 711
15.10 Ćwiczenia 717

16 Tablice obiektów 720
16.1 Tablica obiektów definiowana operatorem `new` 722
16.2 Inicjalizacja tablic obiektów 724
16.2.1 Inicjalizacja tablic obiektów będących agregatami 724
16.2.2 Inicjalizacja tablic nie będących agregatami 728
16.2.3 Inicjalizacja tablic tworzonych w zapasie pamięci 732
16.3 Ćwiczenia 733

17 Wskaźnik do składników klasy 734
17.1 Wskaźniki zwykłe – repetytorium 734
17.2 Wskaźnik do pokazywania na składnik-daną 736
17.2.1 Przykład zastosowania wskaźników do składników klasy 740
17.3 Wskaźnik do funkcji składowej 747
17.3.1 Zastosowanie wskaźników do funkcji składowych 749
17.4 Tablica wskaźników do danych składowych klasy 756
17.5 Tablica wskaźników do funkcji składowych klasy 757
17.6 Wskaźniki do składników statycznych 758
17.7 Ćwiczenia 759

18 Konwersje 760
18.1 Sformułowanie problemu 760
18.2 Konstruktory konwertujące 763
18.2.1 Kiedy jawnie, kiedy niejawnie 765
18.2.2 Przykład konwersji konstruktorem 770
18.3 Funkcja konwertująca – operator konwersji 772
18.3.1 Na co konwertować nie można 778
18.4 Który wariant konwersji wybrać ? 779
18.5 Sytuacje, w których zachodzi konwersja 781
18.6 Zapis jawnego wywołania konwersji typów 782
18.6.1 Advocatus zapisu przypominającego: „wywołanie funkcji“ 783
18.6.2 Advocatus zapisu: „rzutowanie“ 783
18.7 Niecałkiem pasujące argumenty, czyli konwersje przy dopasowaniu 784
18.8 Kilka rad dotyczących konwersji 789
18.9 Ćwiczenia 791

SYMFONIA C++ STANDARD

19 Przeładowanie operatorów 793
19.1 Przeładowanie operatorów – definicja i trochę teorii 796
19.2 Moje zabawki 800
19.3 Funkcja operatorowa jako funkcja składowa 801
19.4 Funkcja operatorowa nie musi być przyjacielem klasy 805
19.5 Operatory predefiniowane 806
19.6 Argumentowość operatorów 807
19.7 Operatory jednoargumentowe 807
19.8 Operatory dwuargumentowe 810
19.8.1 Przykład na przeładowanie operatora dwuargumentowego 811
19.8.2 Przemienność 812
19.8.3 Choć operatory inne, to nazwę mają tę samą 814
19.9 Przykład zupełnie nie matematyczny 814
19.10 Cztery operatory, które muszą być niestatycznymi funkcjami składowymi 827
19.11 Operator przypisania = ... 827
19.11.1 Przykład na przeładowanie operatora przypisania 830
19.11.2 Jak konieczność istnienia operatora przypisania – opowiedzieć potocznie? 840
19.11.3 Kiedy operator przypisania nie jest generowany automatycznie 842
19.12 Operator [ ] 843
19.13 Operator ( ) 848
19.14 Operator –> 850
19.14.1 „Zręczny wskaźnik" – wykorzystuje przeładowanie właśnie tego operatora 852
19.15 Operatory `new`, `new[]` 859
19.15.1 Przykład przeładowania operatora `new` 860
19.15.2 Przykład przeładowania operatora `new[]` 862
19.16 Operatory `delete`, `delete[]` 863
19.16.1 Prosty przykład przeładowania `delete` 864
19.16.2 Prosty przykład przeładowania `delete[]` 865
19.17 Program przykładowy na zastosowanie operatorów `new`, `delete` 865
19.18 Przeładowanie globalnych operatorów `new`, `new[]`, `delete`, `delete[]` 869
19.19 Operatory postinkrementacji i postdekrementacji, czyli koniec z niesprawiedliwością 871
19.20 Rady praktyczne dotyczące przeładowania 874
19.21 Pojedynek: Operator jako funkcja składowa, czy globalna 876
19.22 Zasłona spada, czyli tajemnica operatora << 878
19.23 Rzut oka wstecz 884
19.24 Ćwiczenia 886

20 Dziedziczenie 890
20.1 Istota dziedziczenia 890
20.2 Dostęp do składników 894
20.2.1 Prywatne składniki klasy podstawowej 894
20.2.2 Nieprywatne składniki klasy podstawowej 896
20.2.3 Klasa pochodna też decyduje 897
20.2.4 Deklaracja dostępu `using` – czyli udostępnianie wybiórcze 899
20.3 Czego się nie dziedziczy 903

SYMFONIA C++ STANDARD

20.3.1 "Nie dziedziczenie" konstruktorów 903
20.3.2 "Nie dziedziczenie" operatora przypisania 904
20.3.3 "Nie dziedziczenie" destruktora 904
20.4 Drzewo genealogiczne 905
20.5 Dziedziczenie – doskonałe narzędzie programowania 906
20.6 Kolejność wywoływania konstruktorów 909
20.7 Przypisanie i inicjalizacja obiektów w warunkach dziedziczenia 915
20.7.1 Klasa pochodna nie definiuje swojego operatora przypisania 915
20.7.2 Klasa pochodna nie definiuje swojego konstruktora kopiującego 917
20.7.3 Inicjalizacja i przypisywanie według obiektu wzorcowego będącego `const` ..917
20.7.4 Definiowanie konstruktora kopiującego i operatora przypisania dla klasy pochodnej 918
20.8 Dziedziczenie od kilku "rodziców" (czyli wielokrotne) 925
20.8.1 Konstruktor klasy pochodnej przy wielokrotnym dziedziczeniu 927
20.8.2 Ryzyko wieloznaczności przy dziedziczeniu 929
20.8.3 Bliższe pokrewieństwo usuwa wieloznaczność 932
20.8.4 Poszlaki 932
20.9 Pojedynek: Dziedziczenie klasy, contra zawieranie obiektów składowych 933
20.10 Konwersje standardowe przy dziedziczeniu 935
20.10.1 Panorama korzyści 940
20.10.2 Czego robić się nie opłaca 943
20.10.3 Tuzin samochodów nie jest rodzajem tuzina pojazdów 944
20.10.4 Konwersje standardowe wskaźnika do składnika klasy 949
20.11 Wirtualne klasy podstawowe 951
20.11.1 Publiczne i prywatne dziedziczenie tej samej klasy wirtualnej 955
20.11.2 Uwagi o konstrukcji i inicjalizacji w przypadku klas wirtualnych 956
20.11.3 Dominacja klas wirtualnych 960
20.12 Ćwiczenia 961

21 Funkcje wirtualne 968

21.1 Polimorfizm 975
21.2 Typy rezultatów różnych realizacji funkcji wirtualnej 979
21.3 Dalsze szczegóły 982
21.4 Wczesne i późne wiązanie 984
21.5 Kiedy dla wywołań funkcji wirtualnych, mimo wszystko, zachodzi wczesne wiązanie? 986
21.6 Kulisy białej magii, czyli: Jak to jest zrobione ? 988
21.7 Funkcja wirtualna, a mimo to `inline` 989
21.8 Pojedynek – funkcje przeładowane contra funkcje wirtualne 990
21.9 Klasy abstrakcyjne 992
21.10 Destruktor? to najlepiej wirtualny! 999
21.11 Co prawda, konstruktor nie może być wirtualny, ale... 1004
21.12 Rzutowanie `dynamic_cast` jest dla typów polimorficznych 1011
21.13 Wszystko, co najważniejsze 1015
21.14 Finis coronat opus 1017
21.15 Ćwiczenia 1018

SYMFONIA C++ STANDARD

22 Operacje Wejścia/Wyjścia 1021
22.1 Biblioteka iostream 1022
22.2 Strumień 1023
22.3 Strumienie zdefiniowane standardowo 1025
22.4 Operatory >> i << 1026
22.5 Domniemania w pracy strumieni zdefiniowanych standardowo 1027
22.6 Uwaga na priorytet 1031
22.7 Operatory << oraz >> definiowane przez użytkownika 1032
22.7.1 Operatorów wstawiania i wyjmowania ze strumienia – nie dziedziczy się 1038
22.7.2 Operatory wstawiania i wyjmowania nie mogą być wirtualne. Niestety. 1039
22.8 Sterowanie formatem 1041
22.9 Flagi stanu formatowania 1041
22.9.1 Znaczenie poszczególnych flag sterowania formatem 1044
22.10 Sposoby zmiany trybu (reguł) formatowania 1050
22.10.1 Zmiana sposobu formatowania funkcjami `setf`, `unsetf` 1052
22.10.2 Dodatkowe funkcje do zmiany parametrów formatowania 1057
22.11 Manipulatory 1065
22.11.1 Manipulatory bezargumentowe 1066
22.11.2 Manipulatory parametryzowane 1072
22.11.3 Definiowanie swoich manipulatorów 1076
22.11.4 Manipulator jako funkcja 1077
22.11.5 Definiowane manipulatora z parametrem 1079
22.12 Nieformatowane operacje wejścia/wyjścia 1083
22.13 Omówienie funkcji wyjmujących ze strumienia 1086
22.13.1 Funkcje do pracy ze znakami i stringami 1086
22.13.2 Wczytywanie binarne – funkcje `read` i `readsome` 1093
22.13.3 Funkcja `ignore` 1095
22.13.4 Pożyteczne funkcje pomocnicze 1096
22.13.5 Funkcje wstawiające do strumienia 1099
22.14 Strumienie płynące do lub od plików 1102
22.14.1 Otwieranie i zamykanie strumienia 1104
22.15 Błędy w trakcie pracy strumienia 1110
22.15.1 Flagi stanu błędu strumienia 1111
22.15.2 Funkcje do pracy na flagach błędu 1112
22.15.3 Kilka udogodnień 1113
22.15.4 Ustawianie i kasowanie flag błędu strumienia 1115
22.15.5 Trzy plagi – czyli „gotowiec", jak radzić sobie z błędami 1119
22.16 Przykład programu pracującego na plikach 1123
22.17 Strumienie, a technika rzucania wyjątków 1126
22.18 Wybór miejsca czytania lub pisania w pliku 1130
22.18.1 Funkcje składowe informujące o pozycji wskaźników 1132
22.18.2 Wybrane funkcje składowe do pozycjonowania wskaźników 1132
22.19 Pozycjonowanie w przykładzie większego programu 1136
22.20 Tie – harmonijna praca dwóch strumieni 1143
22.21 Dlaczego tak nie lubimy biblioteki `stdio`? 1145

SYMFONIA C++ STANDARD

22.22 Synchronizacja biblioteki `iostream` z biblioteką `stdio` 1148
22.23 Strumień zapisujący do obiektu klasy `string` 1150
22.23.1 Program przykładowy ilustrujący użycie klasy `ostringstream` 1155
22.24 Strumień czytający z obiektu klasy `string` 1159
22.24.1 Prosty przykład użycia strumienia istringstream 1161
22.24.2 Wczytywanie argumentów wywoływania programu 1167
22.25 Ożenek: strumień `stringstream` – czytający i zapisujący do stringu 1170
22.25.1 Przykładowy program posługujący się klasą `stringstream` 1172
22.26 Ćwiczenia 1174

23 Projektowanie programów orientowanych obiektowo 1182
23.1 Przegląd kilku technik programowania 1183
23.1.1 Programowanie liniowe 1183
23.1.2 Programowanie proceduralne (czyli "orientowane funkcyjnie") 1183
23.1.3 Programowanie z ukrywaniem danych 1184
23.1.4 Programowanie obiektowe – programowanie „bazujące“ na obiektach 1184
23.1.5 Programowanie Obiektowo Orientowane (OO) 1185
23.2 O wyższości programowania obiektowo orientowanego nad Świętami Wielkiej Nocy 1186
23.3 Obiektowo Orientowane: Projektowanie 1189
23.4 Praktyczne wskazówki dotyczące projektowania programu techniką OO 1190
23.4.1 Rekonesans – czyli rozpoznanie zagadnienia 1191
23.4.2 Faza projektowania 1192
23.4.3 Etap 1: Identyfikacja zachowań systemu 1193
23.4.4 Etap 2: Identyfikacja obiektów (klas obiektów) 1193
23.4.5 Etap 3: Usystematyzowanie klas obiektów 1195
23.4.6 Etap 4: Określenie wzajemnych zależności klas 1197
23.4.7 Etap 5: Składanie modelu. Określanie sekwencji działań obiektów i cykli życiowych 1199
23.5 Faza implementacji 1200
23.6 Przykład projektowania 1201
23.7 Faza: Rozpoznanie naszego zagadnienia 1201
23.8 Faza: Projektowanie 1205
23.8.1 Etap 1 – Identyfikacja zachowań naszego systemu 1205
23.8.2 Etap 2 – Identyfikacja klas obiektów, z którymi mamy do czynienia 1206
23.8.3 Etap 3 – Usystematyzowanie klas obiektów z występujących w naszym systemie 1210
23.8.4 Etap 4 – Określenie wzajemnych zależności klas 1211
23.8.5 Etap 5 – Składamy model naszego systemu 1214
23.9 Implementacja modelu naszego systemu 1218
23.10 Symfonia C++, Coda 1225
23.11 Posłowie 1226

A Dodatek: Systemy liczenia 1228
A.1 Dlaczego komputer nie liczy tak jak my? 1228
A.2 System szesnastkowy (heksadecymalny) 1234

SYMFONIA C++ STANDARD

A.3 Ćwiczenia 1237
Skorowidz 1239

SYMFONIA C++ STANDARD