

Spis treści

(obu tomów)

0 Proszę nie czytać tego!	1
1 Startujemy!	6
1.1 Pierwszy program	6
1.2 Drugi program	11
1.3 Ćwiczenia	17
2 Instrukcje sterujące	19
2.1 Prawda – Fałsz, czyli o warunkach	19
2.1.1 Wyrażenie logiczne	19
2.1.2 Zmienne logiczne bool jako warunek	20
2.1.3 Stare dobre sposoby z dawnego C++	20
2.2 Instrukcja warunkowa if	21
2.3 Pętla while	25
2.4 Pętla do...while	26
2.5 Pętla for	27
2.6 Instrukcja switch	29
2.7 Co wybrać: switch czy if...else?	32
2.8 Instrukcja break	34
2.9 Instrukcja goto	36
2.10 Instrukcja continue	37
2.11 Klamry w instrukcjach sterujących	38
2.12 Ćwiczenia	39

3 Typy	42
3.1 Deklaracje typów	42
3.2 Systematyka typów z języka C++	43
3.3 Typy fundamentalne	44
3.3.1 Definiowanie obiektów „w biegu”	48
3.4 Stałe dosłowne	51
3.4.1 Stałe będące liczbami całkowitymi	51
3.4.2 Stałe reprezentujące liczby zmiennoprzecinkowe	53
3.4.3 Stałe znakowe	54
3.4.4 Stałe tekstowe, napisy, albo po prostu stringi	56
3.5 Typy złożone	59
3.6 Typ void	60
3.7 Zakres ważności nazwy obiektu, a czas życia obiektu	60
3.7.1 Zakres: lokalny	61
3.7.2 Zakres: blok funkcji	61
3.7.3 Zakres: obszar pliku	62
3.7.4 Zakres: obszar klasy	62
3.7.5 Zakres określony przez przestrzeń nazw	62
3.8 Zaskanianie nazw	67
3.9 Specyfikator (przydomek) <code>const</code>	69
3.9.1 Pojedynk: <code>const</code> contra <code>#define</code>	70
3.10 Obiekty <code>register</code>	71
3.11 Specyfikator <code>volatile</code>	72
3.12 Instrukcja <code>typedef</code>	73
3.13 Typy wyliczeniowe <code>enum</code>	74
3.14 Ćwiczenia	77
4 Operatory	81
4.1 Operatory arytmetyczne	81
4.1.1 Operator <code>%</code> , czyli reszta z dzielenia (modulo)	82
4.1.2 Jednoargumentowe operatory <code>+</code> i <code>-</code>	83
4.1.3 Operatory inkrementacji i dekrementacji	84
4.1.4 Operator przypisania <code>=</code>	85
4.2 Operatory logiczne	86
4.2.1 Operatory relacji	86
4.2.2 Operatory sumy logicznej <code> </code> i iloczynu logicznego <code>&&</code>	88
4.2.3 Wykrzyknik <code>!</code> czyli operator negacji	89
4.3 Operatory bitowe	90
4.3.1 Przesunięcie w lewo <code><<</code>	91
4.3.2 Przesunięcie w prawo <code>>></code>	91
4.3.3 Bitowe operatory sumy, iloczynu, negacji, różnicy symetrycznej	92
4.4 Różnica między operatorami logicznymi, a operatorami bitowymi	93
4.5 Pozostałe operatory przypisania	95
4.6 Operator uzyskiwania adresu (operator <code>&</code>)	96

4.7	Wyrażenie warunkowe	97
4.8	Operator <code>sizeof</code>	98
4.9	Operatory rzutowania	99
4.9.1	Rzutowanie według tradycyjnych (nie zalecanych) sposobów	99
4.9.2	Rzutowanie za pomocą nowych operatorów rzutowania	101
4.9.3	Operator <code>static_cast</code>	101
4.9.4	Operator <code>const_cast</code>	103
4.9.5	Operator <code>dynamic_cast</code>	105
4.9.6	Operator <code>reinterpret_cast</code>	105
4.10	Operator: przecinek	106
4.11	Priorytety operatorów	107
4.12	Łączność operatorów	110
4.13	Ćwiczenia	110

5 Funkcje 115

5.1	Funkcja często wywołuje inną funkcję	118
5.2	Zwracanie przez funkcję rezultatu	118
5.3	Stos	122
5.4	Przesyłanie argumentów do funkcji przez wartość	122
5.5	Przesyłanie argumentów przez referencję	124
5.6	Kiedy deklaracja funkcji nie jest konieczna?	126
5.7	Argumenty domniemane	128
5.7.1	Ciekawostki na temat argumentów domniemanych	130
5.8	Nienazwany argument	135
5.9	Funkcje <code>inline</code> (w linii)	137
5.10	Przypomnienie o zakresie ważności nazw deklarowanych wewnątrz funkcji	140
5.11	Wybór zakresu ważności nazwy i czasu życia obiektu	141
5.11.1	Obiekty globalne	141
5.11.2	Obiekty automatyczne	142
5.11.3	Obiekty lokalne statyczne	143
5.12	Funkcje w programie składającym się z kilku plików	146
5.12.1	Nazwy statyczne globalne	150
5.13	Funkcje rekurencyjne	152
5.14	Funkcje biblioteczne	161
5.15	Ćwiczenia	164

6 Preprocesor 170

6.1	Na pomoc rodakom	170
6.2	Dyrektywa <code>#define</code>	171
6.3	Dyrektywa <code>#undef</code>	174
6.4	Makrodefinicje	174
6.5	Sklejacz nazw, czyli operator <code>##</code>	177
6.6	Zamiana parametru aktualnego makrodefinicji na string	177

6.7	Dyrektywy kompilacji warunkowej	179
6.8	Dyrektywa <code>#error</code>	183
6.9	Dyrektywa <code>#line</code>	184
6.10	Wstawianie treści innych plików w tekst kompilowanego własnie pliku	184
6.11	Dyrektywa pusta <code>#</code>	186
6.12	Dyrektywy zależne od implementacji	186
6.13	Nazwy predefiniowane	187
6.14	Ćwiczenia	188

7 Tablice.....192

7.1	Elementy tablicy	193
7.2	Inicjalizacja tablic	195
7.3	Przekazywanie tablicy do funkcji	196
7.4	Przykład z tablicą elementów typu <code>enum</code>	200
7.5	Tablice znakowe	202
7.6	Tablice wielowymiarowe	210
7.6.1	Typ wyrażeń związanych z tablicą wielowymiarową	213
7.6.2	Przesyłanie tablic wielowymiarowych do funkcji	215
7.7	Ćwiczenia	216

8 Wskaźniki.....221

8.1	Wskaźniki mogą bardzo ułatwić życie	221
8.2	Definiowanie wskaźników	223
8.3	Praca ze wskaźnikiem	224
8.4	L-wartość	227
8.5	Operator rzutowania <code>reinterpret_cast</code> , a wskaźniki	228
8.6	Wskaźniki typu <code>void</code>	231
8.7	Cztery domeny zastosowania wskaźników	233
8.8	Zastosowanie wskaźników wobec tablic	233
8.8.1	Ćwiczenia z mechaniki ruchu wskaźnika	233
8.8.2	Użycie wskaźnika w pracy z tablicą	237
8.8.3	Arytmetyka wskaźników	241
8.8.4	Porównywanie wskaźników	243
8.8.5	Wskaźnik można porównać z adresem 0 (zero).....	245
8.9	Zastosowanie wskaźników w argumentach funkcji	246
8.9.1	Jeszcze raz o przesyłaniu tablic do funkcji	249
8.9.2	Odbieranie tablicy jako wskaźnika	250
8.9.3	Argument formalny będący wskaźnikiem do obiektu <code>const</code>	251
8.10	Zastosowanie wskaźników przy dostępie do konkretnych komórek pamięci	254
8.11	Rezerwacja obszarów pamięci	255
8.11.1	Operatory <code>new</code> i <code>delete</code> albo Oratorium Stworzenie Świata	257
8.11.2	Dynamiczna alokacja tablicy	260
8.11.3	Tablice wielowymiarowe tworzone operatorem <code>new</code>	261

8.11.4	Umiejscawiający operator <code>new</code>	263
8.11.5	"Przychodzimy, odchodzimy – cichuteczko, na..."	267
8.11.6	Zapas pamięci to nie jest studnia bez dna	270
8.11.7	Funkcja <code>set_new_handler</code>	273
8.11.8	Pojedynek: <code>new</code> contra <code>malloc</code>	274
8.12	Stałe wskaźniki	275
8.13	Stałe wskaźniki, a wskaźniki do stałych	276
8.14	Strzał na oślep – Wskaźnik zawsze pokazuje na coś	277
8.15	Sposoby ustawiania wskaźników	278
8.16	Parada kłamców, czyli o rzutowaniu <code>const_cast</code>	280
8.17	Tablice wskaźników	284
8.18	Wariacje na temat C-stringów	286
8.19	Wskaźniki do funkcji	293
8.19.1	Ćwiczenia z definiowania wskaźników do funkcji	297
8.19.2	Wskaźnik do funkcji jako argument innej funkcji	302
8.19.3	Tablica wskaźników do funkcji	311
8.20	Argumenty z linii wywołania programu	316
8.21	Ćwiczenia	318

9 Przeladowanie nazwy funkcji.....326

9.1	Co to znaczy: przeladowanie	326
9.2	Bliższe szczegóły przeladowania	329
9.3	Czy przeladowanie nazw funkcji jest techniką obiektowo orientowaną?	332
9.4	Linkowanie z modułami z innych języków	333
9.5	Przeladowanie, a zakres ważności deklaracji funkcji	334
9.6	Rozważania o identyczności lub odmienności typów argumentów	336
9.6.1	Przeladowanie, a <code>typedef</code> i <code>enum</code>	336
9.6.2	Tablica, a wskaźnik	336
9.6.3	Pewne szczegóły o tablicach wielowymiarowych	338
9.6.4	Przeladowanie, a referencja	340
9.6.5	Identyczność typów: <code>T</code> , <code>const T</code> , <code>volatile T</code>	340
9.6.6	Przeladowanie – a typy: <code>T*</code> , <code>volatile T*</code> , <code>const T*</code>	341
9.6.7	Przeladowanie – a typy: <code>T&</code> , <code>volatile T&</code> , <code>const T&</code>	343
9.7	Adres funkcji przeladowanej	343
9.7.1	Zwrot rezultatu będącego adresem funkcji przeladowanej	345
9.8	Kulisy dopasowywania argumentów do funkcji przeladowanych	347
9.9	Etapy dopasowania	348
9.9.1	Etap 1. Dopasowanie dokładne	349
9.9.2	Etap 1a. Dopasowanie dokładne, ale z tzw. trywialną konwersją	349
9.9.3	Etap 2. Dopasowanie z awansem (z promocją)	350
9.9.4	Etap 3. Próba dopasowania za pomocą konwersji standardowych	352
9.9.5	Etap 4. Próba dopasowania z użyciem konwersji zdefiniowanych przez użytkownika	354
9.9.6	Etap 5. Próba dopasowania do funkcji z wielokropkiem	354
9.9.7	Wskaźników nie dopasowuje się inaczej niż dosłownie	354
9.10	Dopasowywanie wywołań z kilkoma argumentami	355

9.11	Ćwiczenia	356
10 Klasy		359
10.1	Typy definiowane przez użytkownika	359
10.2	Składniki klasy	361
10.3	Składnik będący obiektem	362
10.4	Enkapsulacja	363
10.5	Ukrywanie informacji	364
10.6	Klasa, a obiekt	367
10.7	Funkcje składowe	369
10.7.1	Posługiwanie się funkcjami składowymi	369
10.7.2	Definiowanie funkcji składowych	370
10.8	Jak to właściwie jest? (<i>this</i>)	376
10.9	Odwwołanie się do publicznych danych składowych	378
10.10	Zasłanianie nazw	379
10.10.1	Nie sięgaj z klasy do obiektów globalnych	382
10.11	Przeladowanie i zasłonięcie równocześnie	382
10.12	Nowa klasa? Osobny plik!	383
10.13	Przesyłanie do funkcji argumentów będących obiektami	394
10.13.1	Przesyłanie obiektu przez wartość	394
10.13.2	Przesyłanie przez referencję	396
10.14	Konstruktor – pierwsza wzmianka	397
10.15	Destruktor – pierwsza wzmianka	402
10.16	Składnik statyczny	405
10.16.1	Deklaracja składnika statycznego połączona z inicjalizacją	410
10.17	Statyczna funkcja składowa	415
10.18	Do czego może nam się przydać składnik statyczny w klasie?	418
10.19	Funkcje składowe typu <i>const</i> oraz <i>volatile</i>	419
10.19.1	Przeladowanie, a funkcje składowe <i>const</i> i <i>volatile</i>	422
10.20	Specyfikator <i>mutable</i>	422
10.21	Ćwiczenia	434
11 Biblioteczna klasa <i>std::string</i> do operacji z tekstami		438
11.1	Przykład programu z użyciem klasy <i>string</i>	440
11.2	Definiowanie obiektów klasy <i>string</i>	445
11.3	Użycie operatorów <i>=</i> , <i>+</i> , <i>+=</i> w pracy ze <i>stringami</i>	449
11.3.1	Jak umieścić w tekście liczbę?	450
11.4	Pojemność, rozmiar i długość <i>stringu</i>	452
11.4.1	Funkcje <i>size()</i> i <i>length()</i>	452
11.4.2	Funkcja składowa <i>empty</i>	453
11.4.3	Funkcja składowa <i>max_size</i>	453
11.4.4	Funkcja składowa <i>capacity</i>	453
11.4.5	Funkcja składowa <i>reserve</i>	455

11.4.6	<i>resize</i> – zmiana długości stringu „na siłę”	456
11.4.7	Funkcja składowa <i>clear</i>	458
11.5	Użycie operatora <i>[]</i> oraz funkcji <i>at</i>	458
11.5.1	Działanie operatora <i>[]</i>	459
11.5.2	Działanie funkcji składowej <i>at</i>	460
11.6	Praca z fragmentem stringu, czyli z sub-stringiem	462
11.7	Funkcja składowa <i>substr</i>	463
11.8	Szukanie zadanego substringu w obiekcie klasy <i>string</i> – funkcja <i>find</i> i jej pokrewne	464
11.9	Szukanie rozpoczynane od końca stringu.....	467
11.10	Szukanie w stringu jednego ze znaków z zadanego zestawu	468
11.11	Usuwanie znaków ze stringu – funkcje <i>erase</i>	471
11.12	Wstawianie znaków do już istniejącego stringu – funkcje <i>insert</i>	472
11.13	Zamiana części znaków na inne znaki – <i>replace</i>	474
11.14	Zamiana zawartości obiektu klasy <i>string</i> na C-string.....	479
11.15	Zagłębienie do wnętrza obiektu klasy <i>string</i> funkcją <i>data</i>	482
11.16	W porządku alfabetycznym – czyli porównywanie stringów	483
11.16.1	Porównywanie stringów funkcjami <i>compare</i>	484
11.16.2	Porównywanie stringów przy użyciu operatorów <i>==</i> , <i>!=</i> , <i><</i> , <i>></i> , <i><=</i> , <i>>=</i>	488
11.17	Zamiana treści stringu na małe (lub wielkie) litery	490
11.18	Kopiowanie treści obiektu klasy <i>string</i> do wybranej tablicy znakowej – funkcja <i>copy</i>	496
11.19	Wzajemna zamiana treści dwóch obiektów klasy <i>string</i> – funkcja <i>swap</i>	496
11.20	Przypisanie do obiektu klasy <i>string</i> , funkcja <i>assign</i>	497
11.21	Dopisywanie do końca stringu za pomocą funkcji <i>append</i>	499
11.22	Wczytywanie z klawiatury długiego stringu o nieznanym wcześniej długości – <i>getline</i>	500
11.22.1	Pułapka – czyli jak <i>getline</i> może Cię zaskoczyć	503
11.23	Iteratory stringu.....	507
11.23.1	Iterator do obiektu stałego	511
11.23.2	Funkcje składowe klasy <i>string</i> pracujące z iteratorami	513
11.24	Bryk – czyli „pamięć zewnętrzna” programisty	519
11.25	Ćwiczenia	528

12Deklaracje przyjaźni.....534

12.0.1	Klasy zaprzyjaźnione	543
12.0.2	Słowo o zakresie	545

Skorowidz.....S1

Tom II

13Struktury, Unie, Pola bitowe.....547

13.1	Struktura.....	547
13.2	Unia.....	548

13.2.1	Inicjalizacja unii	550
13.2.2	Unia anonimowa	550
13.3	Pola bitowe	552
13.4	Unia i pola bitowe – upraszczają rozpakowanie słów	556
13.5	Ćwiczenia	563
14	Klasa zagnieżdżona lub lokalna	565
14.1	Zagnieżdżona definicja klasy	565
14.2	Praktyczny przykład zagnieżdżenia definicji klasy	569
14.3	Lokalna definicja klasy	581
14.4	Lokalne nazwy typów	584
14.5	Ćwiczenia	584
15	Konstruktory i Destructory	587
15.1	Konstruktor	587
15.1.1	Przykład programu zawierającego klasę z konstruktorami	588
15.2	Specyfikator (przydomek) <i>explicit</i>	600
15.3	Kiedy i jak wywoływany jest konstruktor	600
15.3.1	Konstruowanie obiektów lokalnych	601
15.3.2	Konstruowanie obiektów globalnych	601
15.3.3	Konstrukcja obiektów tworzonych operatorem <i>new</i>	601
15.3.4	Jawne wywołanie konstruktora	602
15.3.5	Dalsze sytuacje, gdy pracuje konstruktor	605
15.4	Destructor	605
15.5	Konstruktor domniemany	607
15.6	Lista inicjalizacyjna konstruktora	608
15.7	Konstrukcja obiektu, którego składnikiem jest obiekt innej klasy	611
15.8	Konstruktory nie-publiczne ?	617
15.9	Konstruktor kopiujący (albo inicjalizator kopiujący)	619
15.9.1	Przykład klasy z konstruktorem kopiującym	621
15.9.2	Dlaczego przez referencję?	627
15.9.3	Jak dostać piątkę z C++ ?	628
15.9.4	Konstruktor kopiujący gwarantujący nietykalność	629
15.9.5	Współodpowiedzialność	630
15.9.6	Konstruktor kopiujący generowany automatycznie	630
15.9.7	Kiedy konstruktor kopiujący jest niezbędny?	631
15.10	Ćwiczenia	636
16	Tablice obiektów	640
16.1	Tablica obiektów definiowana operatorem <i>new</i>	641
16.2	Inicjalizacja tablic obiektów	643
16.2.1	Inicjalizacja tablic obiektów będących agregatami	643
16.2.2	Inicjalizacja tablic nie będących agregatami	647
16.2.3	Inicjalizacja tablic tworzonych w zapasie pamięci	650

16.3	Ćwiczenia	651
17 Wskaźnik do składników klasy		652
17.1	Wskaźniki zwykłe – repetytorium	652
17.2	Wskaźnik do pokazywania na składnik-daną	653
17.2.1	Przykład zastosowania wskaźników do składników klasy	657
17.3	Wskaźnik do funkcji składowej	663
17.3.1	Zastosowanie wskaźników do funkcji składowych	665
17.4	Tablica wskaźników do danych składowych klasy	672
17.5	Tablica wskaźników do funkcji składowych klasy	672
17.6	Wskaźniki do składników statycznych	675
17.7	Ćwiczenia	676
18 Konwersje		678
18.1	Sformułowanie problemu	678
18.2	Konstruktory konwertujące	680
18.2.1	Kiedy jawnie, kiedy niejawnie	682
18.2.2	Przykład konwersji konstruktorem	686
18.3	Funkcja konwertująca – operator konwersji	688
18.3.1	Na co konwertować nie można	693
18.4	Który wariant konwersji wybrać?	694
18.5	Sytuacje, w których zachodzi konwersja	696
18.6	Zapis jawnego wywołania konwersji typów	697
18.6.1	Advocatus zapisu przypominającego: „wywołanie funkcji”	698
18.6.2	Advocatus zapisu: „rzutowanie”	698
18.7	Niecałkiem pasujące argumenty, czyli konwersje przy dopasowaniu	699
18.8	Kilka rad dotyczących konwersji	703
18.9	Ćwiczenia	704
19 Przeladowanie operatorów		707
19.1	Przeladowanie operatorów – definicja i trochę teorii	709
19.2	Moje zabawki	713
19.3	Funkcja operatorowa jako funkcja składowa	714
19.4	Funkcja operatorowa nie musi być przyjacielem klasy	717
19.5	Operatory predefiniowane	718
19.6	Argumentowość operatorów	718
19.7	Operatory jednoargumentowe	719
19.8	Operatory dwuargumentowe	721
19.8.1	Przykład na przeladowanie operatora dwuargumentowego	722
19.8.2	Przemienność	723
19.8.3	Choć operatory inne, to nazwę mają tę samą	725
19.9	Przykład zupełnie nie matematyczny	725
19.10	Cztery operatory, które muszą być niestatycznymi funkcjami składowymi	736

19.11	Operator przypisania =	737
19.11.1	Przykład na przeładowanie operatora przypisania	739
19.11.2	Jak konieczność istnienia operatora przypisania – opowiedzieć potocznie?	748
19.11.3	Kiedy operator przypisania nie jest generowany automatycznie	749
19.12	Operator []	750
19.13	Operator ()	754
19.14	Operator ->	756
19.14.1	„Sprytny wskaźnik” – wykorzystuje przeładowanie właśnie tego operatora	758
19.15	Operatory new, new[]	764
19.15.1	Przykład przeładowania operatora new	765
19.15.2	Przykład przeładowania operatora new[]	767
19.16	Operatory delete, delete[]	767
19.16.1	Prosty przykład przeładowania delete	768
19.16.2	Prosty przykład przeładowania delete[]	768
19.17	Program przykładowy na zastosowanie operatorów new, delete	769
19.18	Przeładowanie globalnych operatorów new, new[], delete, delete[]	773
19.19	Operatory postinkrementacji i postdekrementacji, czyli koniec z niesprawiedliwością	774
19.20	Rady praktyczne dotyczące przeładowania	777
19.21	Pojedynek: Operator jako funkcja składowa, czy globalna?	778
19.22	Zasłona spada, czyli tajemnica operatora <<	780
19.23	Rzut oka wstecz	786
19.24	Ćwiczenia	787

20 Dziedziczenie

791

20.1	Istota dziedziczenia	791
20.2	Dostęp do składników	794
20.2.1	Prywatne składniki klasy podstawowej	794
20.2.2	Nieprywatne składniki klasy podstawowej	796
20.2.3	Klasa pochodna też decyduje	797
20.2.4	Deklaracja dostępu using – czyli udostępnianie wybiórcze	799
20.3	Czego się nie dziedziczy	802
20.3.1	„Nie dziedziczenie” konstruktorów	802
20.3.2	„Nie dziedziczenie” operatora przypisania	803
20.3.3	„Nie dziedziczenie” destruktora	803
20.4	Drzewo genealogiczne	804
20.5	Dziedziczenie – doskonałe narzędzie programowania	805
20.6	Kolejność wywoływania konstruktorów	807
20.7	Przypisanie i inicjalizacja obiektów w warunkach dziedziczenia	813
20.7.1	Klasa pochodna nie definiuje swojego operatora przypisania	813
20.7.2	Klasa pochodna nie definiuje swojego konstruktora kopiującego	814
20.7.3	Inicjalizacja i przypisywanie według obiektu wzorcowego będącego const	815
20.7.4	Definiowanie konstruktora kopiującego i operatora przypisania dla klasy pochodnej	815
20.8	Dziedziczenie od kilku „rodziców” (wielodziedziczenie)	821
20.8.1	Konstruktor klasy pochodnej przy wielodziedziczeniu	823

20.8.2	Ryzyko wieloznaczności przy dziedziczeniu	826
20.8.3	Czy bliższe pokrewieństwo usuwa wieloznaczność?	828
20.8.4	Poszlaki	828
20.9	Pojedynek: Dziedziczenie klasy, contra zawieranie obiektów składowych	829
20.10	Konwersje standardowe przy dziedziczeniu	831
20.10.1	Panorama korzyści	835
20.10.2	Czego robić się nie oplaca	837
20.10.3	Tuzin samochodów nie jest rodzajem tuzina pojazdów	838
20.10.4	Konwersje standardowe wskaźnika do składnika klasy	842
20.11	Wirtualne klasy podstawowe	844
20.11.1	Publiczne i prywatne dziedziczenie tej samej klasy wirtualnej	847
20.11.2	Uwagi o konstrukcji i inicjalizacji w przypadku klas wirtualnych	848
20.11.3	Dominacja klas wirtualnych	851
20.12	Ćwiczenia	853

21 Funkcje wirtualne

21.1	Polimorfizm	865
21.2	Typy rezultatów różnych realizacji funkcji wirtualnej	868
21.3	Dalsze szczegóły	871
21.4	Wczesne i późne wiązanie	873
21.5	Kiedy dla wywołań funkcji wirtualnych, mimo wszystko, zachodzi wczesne wiązanie?	875
21.6	Kulisy białej magii, czyli: Jak to jest zrobione?	876
21.7	Funkcja wirtualna, a mimo to inline	877
21.8	Pojedynek – funkcje przeładowane contra funkcje wirtualne	878
21.9	Klasy abstrakcyjne	879
21.10	Destruktor? to najlepiej wirtualny!	885
21.11	Co prawda, konstruktor nie może być wirtualny, ale... ..	890
21.12	Rzutowanie <code>dynamic_cast</code> jest dla typów polimorficznych	896
21.13	Wszystko co najważniejsze	899
21.14	Finis coronat opus	902
21.15	Ćwiczenia	902

22 Operacje Wejścia/Wyjścia

22.1	Biblioteka <code>iostream</code>	906
22.2	Strumień	906
22.3	Strumienie zdefiniowane standardowo	908
22.4	Operatory <code>>></code> i <code><<</code>	909
22.5	Domniemania w pracy strumieni zdefiniowanych standardowo	910
22.6	Uwaga na priorytet	913
22.7	Operatory <code><<</code> oraz <code>>></code> definiowane przez użytkownika	914
22.7.1	Operatorów wstawiania i wyjmowania ze strumienia – nie dziedziczy się	919
22.7.2	Operatory wstawiania i wyjmowania nie mogą być wirtualne. Niestety	920
22.8	Sterowanie formatem	922

22.9	Flagi stanu formatowania	922
22.9.1	Znaczenie poszczególnych flag sterowania formatem	925
22.10	Sposoby zmiany trybu (reguł) formatowania	930
22.10.1	Zmiana sposobu formatowania funkcjami <i>self</i> , <i>unsetf</i>	932
22.10.2	Dodatkowe funkcje do zmiany parametrów formatowania	936
22.11	Manipulatory	942
22.11.1	Manipulatory bezargumentowe	943
22.11.2	Manipulatory parametryzowane	948
22.11.3	Definiowanie swoich manipulatorów	952
22.11.4	Manipulator jako funkcja	952
22.11.5	Definiowanie manipulatora z parametrem	954
22.12	Nieformatowane operacje wejścia/wyjścia	958
22.13	Omówienie funkcji wyjmujących ze strumienia	960
22.13.1	Funkcje do pracy ze znakami i stringami	960
22.13.2	Wczytywanie binarne – funkcje <i>read</i> i <i>readsome</i>	966
22.13.3	Funkcja <i>ignore</i>	968
22.13.4	Pożyteczne funkcje pomocnicze	969
22.13.5	Funkcje wstawiające do strumienia	971
22.14	Strumień płynący do lub od plików	973
22.14.1	Otwieranie i zamykanie strumienia	975
22.15	Błędy w trakcie pracy strumienia	981
22.15.1	Flagi stanu błędu strumienia	981
22.15.2	Funkcje do pracy na flagach błędu	982
22.15.3	Kilka udogodnień	983
22.15.4	Ustawianie i kasowanie flag błędu strumienia	984
22.15.5	Trzy plagi – czyli „gotowiec”, jak radzić sobie z błędami	988
22.16	Przykład programu pracującego na plikach	992
22.17	Strumień, a technika rzucania wyjątków	994
22.18	Wybór miejsca czytania lub pisania w pliku	999
22.18.1	Funkcje składowe informujące o pozycji wskaźników	1000
22.18.2	Wybrane funkcje składowe do pozycjonowania wskaźników	1000
22.19	Pozycjonowanie w przykładzie większego programu	1003
22.20	Tie – harmonijna praca dwóch strumieni	1010
22.21	Dlaczego tak nie lubimy biblioteki <i>stdio</i> ?	1012
22.22	Synchronizacja biblioteki <i>iostream</i> z biblioteką <i>stdio</i>	1014
22.23	Strumień zapisujący do obiektu klasy <i>string</i>	1015
22.23.1	Program przykładowy ilustrujący użycie klasy <i>ostream</i>	1019
22.24	Strumień czytający z obiektu klasy <i>string</i>	1023
22.24.1	Prosty przykład użycia strumienia <i>istream</i>	1025
22.24.2	Wczytywanie argumentów wywołania programu	1030
22.25	Ożenek: strumień <i>stringstream</i> – czytający i zapisujący do stringu	1033
22.25.1	Przykładowy program posługujący się klasą <i>stringstream</i>	1034
22.26	Ćwiczenia	1037

23 Projektowanie programów orientowanych obiektowo	1044
23.1 Przegląd kilku technik programowania	1045
23.1.1 Programowanie liniowe	1045
23.1.2 Programowanie proceduralne (czyli "orientowane funkcyjnie")	1045
23.1.3 Programowanie z ukrywaniem danych	1046
23.1.4 Programowanie obiektowe – programowanie „bazujące” na obiektach	1046
23.1.5 Programowanie Obiektowo Orientowane (OO)	1046
23.2 O wyższości programowania obiektowo orientowanego nad Świątami Wielkiej Nocy	1047
23.3 Obiektowo Orientowane: Projektowanie	1049
23.4 Praktyczne wskazówki dotyczące projektowania programu techniką OO	1051
23.4.1 Rekonesans – czyli rozpoznanie zagadnienia	1051
23.4.2 Faza projektowania	1052
23.4.3 Etap 1: Identyfikacja zachowań systemu	1053
23.4.4 Etap 2: Identyfikacja obiektów (klas obiektów)	1053
23.4.5 Etap 3: Usystematyzowanie klas obiektów	1055
23.4.6 Etap 4: Określenie wzajemnych zależności klas	1056
23.4.7 Etap 5: Składanie modelu. Określanie sekwencji działań obiektów i cykli życiowych	1058
23.5 Faza implementacji	1059
23.6 Przykład projektowania	1060
23.7 Faza: Rozpoznanie naszego zagadnienia	1060
23.8 Faza: Projektowanie	1064
23.8.1 Etap 1 – Identyfikacja zachowań naszego systemu	1064
23.8.2 Etap 2 – Identyfikacja klas obiektów, z którymi mamy do czynienia	1065
23.8.3 Etap 3 – Usystematyzowanie klas obiektów z występujących w naszym systemie	1068
23.8.4 Etap 4 – Określenie wzajemnych zależności klas	1069
23.8.5 Etap 5 – Składamy model naszego systemu	1071
23.9 Implementacja modelu naszego systemu	1075
23.10 Symfonia C++, Coda	1082
23.11 Posłowie	1082
A Dodatek: Systemy liczenia	1084
A.1 Dlaczego komputer nie liczy tak jak my?	1084
A.2 System szesnastkowy (heksadecymalny)	1089
A.3 Ćwiczenia	1092
B Dodatek: Sprzężenie zwrotne	1093
Skorowidz	S1