

Spis treści trzech tomów

Tom 1

0	Proszę tego nie czytać!.....	1
0.1	Zaprzyjajnijmy się!	1
1	Startujemy!.....	8
1.1	Pierwszy program	8
1.2	Drugi program	13
1.3	Ćwiczenia	18
2	Instrukcje sterujące.....	20
2.1	Prawda – fałsz, czyli o warunkach.....	20
2.1.1	Wyrażenie logiczne	20
2.1.2	Zmienna logiczna <i>bool</i> w roli warunku.....	21
2.1.3	Stare dobre sposoby z dawnego C++	21
2.2	Instrukcja warunkowa <i>if</i>	22
2.3	Pętla <i>while</i>	26
2.4	Pętla <i>do...while</i>	27
2.5	Pętla <i>for</i>	28
2.6	Instrukcja <i>switch</i>	31
2.7	Co wybrać: <i>switch</i> czy <i>if...else?</i>	33
2.8	Instrukcja <i>break</i>	36
2.9	Instrukcja <i>goto</i>	37
2.10	Instrukcja <i>continue</i>	39
2.11	Klamry w instrukcjach sterujących.....	40
2.12	Ćwiczenia	41
3	Typy	44
3.1	Deklaracje typu.....	44
3.2	Systematyka typów z języka C++.....	45
3.3	Typy fundamentalne.....	46
3.3.1	Typy przeznaczone do pracy z liczbami całkowitymi.....	46
3.3.2	Typ do przechowywania znaków alfanumerycznych.....	47
3.3.3	Typy reprezentujące liczby zmiennoprzecinkowe	47

3.3.4	bool – typ do reprezentacji obiektów logicznych.....	48
3.3.5	Kwestia dokładności	49
3.3.6	Jak poznać limity (ograniczenia) typów wbudowanych.....	51
3.4	Typy o precyzyjnie żądanej szerokości	55
3.5	Inicjalizacja, czyli nadanie wartości w momencie narodzin	59
3.6	Definiowanie obiektów „w biegu”	60
3.7	Stałe dosłowne.....	62
3.7.1	Stałe dosłowne typu <i>bool</i>	63
3.7.2	Stałe będące liczbami całkowitymi	63
3.7.3	Stałe reprezentujące liczby zmiennoprzecinkowe	66
3.7.4	Stała dosłowna <i>nullptr</i> – dla wskaźników	67
3.7.5	Stałe znakowe	68
3.7.6	Stałe tekstowe, napisy, albo po prostu stringi	71
3.7.7	Surowe stałe tekstowe (napisy, stringi)	73
3.8	Typy złożone	76
3.9	Typ <i>void</i>	77
3.10	Zakres ważności nazwy obiektu a czas życia obiektu	78
3.10.1	Zakres: lokalny	78
3.10.2	Zakres instrukcji.....	79
3.10.3	Zakres: blok funkcji	79
3.10.4	Zakres: obszar pliku	80
3.10.5	Zakres: obszar klasy	80
3.10.6	Zakres określony przez przestrzeń nazw	80
3.11	Zasłanianie nazw	85
3.12	Specyfikator (przydomek) <i>const</i>	87
3.13	Specyfikator (przydomek) <i>constexpr</i>	88
3.14	Obiekty <i>register</i>	92
3.15	Specyfikator <i>volatile</i>	92
3.16	<i>using</i> oraz <i>typedef</i> – tworzenie dodatkowej nazwy typu	93
3.17	Typy wyliczeniowe <i>enum</i>	96
3.17.1	Dawne zwykłe <i>enum</i> a nowe zakresowe <i>enum class</i>	103
3.17.2	Kilka uwag dla wtajemniczonych	105
3.18	<i>auto</i> , czyli automatyczne rozpoznawanie typu definiowanego obiektu	106
3.19	<i>decltype</i> – operator do określania typu zadanego wyrażenia	109
3.20	Inicjalizacja z pustą klamrą { }, czyli wartością domniemaną.....	111
3.21	Przydomek <i>alignas</i> – adresy równe i równiejsze	113
3.22	Ćwiczenia	115

4 Operatory119

4.1	Operatory arytmetyczne.....	119
4.1.1	Operator %, czyli reszta z dzielenia (modulo)	120
4.1.2	Jednoargumentowe operatory + i –	121
4.1.3	Operatory inkrementacji i dekrementacji	121
4.1.4	Operator przypisania =	123
4.2	Operatory logiczne	124
4.2.1	Operatory relacji	124
4.2.2	Operatory sumy logicznej oraz iloczynu logicznego &&	125
4.2.3	Wykrzyknik !, czyli operator negacji.....	126
4.3	Operatory bitowe.....	127
4.3.1	Przesunięcie w lewo <<	128
4.3.2	Przesunięcie w prawo >>	129
4.3.3	Bitowe operatory sumy, iloczynu, negacji, różnicy symetrycznej	130
4.4	Różnica między operatorami logicznymi a operatorami bitowymi	130
4.5	Pozostałe operatory przypisania	132
4.6	Operator uzyskiwania adresu (operator &).....	133

4.7	Wyrażenie warunkowe	134
4.8	Operator <i>sizeof</i>	135
4.9	Operator <i>noexcept</i>	137
4.10	Deklaracja <i>static_assert</i>	137
4.11	Operator <i>alignof</i> informujący o najkorzystniejszym wyrównaniu adresu	139
4.12	Operatory rzutowania	141
4.12.1	Rzutowanie według tradycyjnych (niezalecanych) sposobów	141
4.12.2	Rzutowanie za pomocą nowych operatorów rzutowania	142
4.12.3	Operator <i>static_cast</i>	143
4.12.4	Operator <i>const_cast</i>	145
4.12.5	Operator <i>dynamic_cast</i>	146
4.12.6	Operator <i>reinterpret_cast</i>	147
4.13	Operator: przecinek	148
4.14	Priorytety operatorów	148
4.15	Łączność operatorów	151
4.16	Ćwiczenia	152

5 Typ *string* i typ *vector* – pierwsza wzmianka156

5.1	Typ <i>std::string</i> do pracy z tekstami	156
5.2	Typ <i>vector</i> – długi rząd obiektów	161
5.3	Zakresowe <i>for</i>	169
5.4	Ćwiczenia	172

6 Funkcje174

6.1	Definicja funkcji i jej wywołanie.....	174
6.2	Deklaracja funkcji.....	175
6.3	Funkcja często wywołuje inną funkcję	177
6.4	Zwracanie przez funkcję rezultatu	177
6.4.1	Obiekt tworzony za pomocą <i>auto</i> , a inicjalizowany rezultatem funkcji	179
6.4.2	O zwracaniu (lub niezwracaniu) rezultatu przez funkcję <i>main</i>	180
6.5	Nowy, alternatywny sposób deklaracji funkcji.....	181
6.6	Stos	183
6.7	Przesyłanie argumentów do funkcji przez wartość	184
6.8	Przesyłanie argumentów przez referencję.....	185
6.9	Pożyteczne określenia: lwartość i rwartość.....	188
6.10	Referencje do lwartości i referencje do rwartości jako argumenty funkcji.....	190
6.10.1	Który sposób przesyłania argumentu do funkcji wybrać?.....	197
6.11	Kiedy deklaracja funkcji nie jest konieczna?	198
6.12	Argumenty domniemane	199
6.12.1	Ciekawostki na temat argumentów domniemanych	202
6.13	Nienazwany argument	207
6.14	Funkcje <i>inline</i> (w linii).....	208
6.15	Przypomnienie o zakresie ważności nazw deklarowanych wewnątrz funkcji	212
6.16	Wybór zakresu ważności nazwy i czasu życia obiektu.....	212
6.16.1	Obiekty globalne	212
6.16.2	Obiekty automatyczne.....	213
6.16.3	Obiekty lokalne statyczne	214
6.17	Funkcje w programie składającym się z kilku plików	218
6.17.1	Nazwy statyczne globalne.....	222
6.18	Funkcja zwracająca rezultat będący referencją lwartości	223
6.19	Funkcje rekurencyjne	228
6.20	Funkcje biblioteczne.....	237
6.21	Funkcje <i>constexpr</i>	240
6.21.1	Wymogi, które musi spełniać funkcja <i>constexpr</i> (w standardzie C++11)	242

6.21.2	Przykład pokazujący aspekty funkcji <i>constexpr</i>	243
6.21.3	Argumenty funkcji <i>constexpr</i> , będące referencjami	252
6.22	Definiowanie referencji przy użyciu słowa <i>auto</i>	253
6.22.1	Gdy inicjalizatorem jest wywołanie funkcji zwracającej referencje.....	260
6.23	Ćwiczenia	263

7 Preprocesor270

7.1	Dyrektywa pusta #	270
7.2	Dyrektywa <i>#define</i>	270
7.3	Dyrektywa <i>#undef</i>	272
7.4	Makrodefinicje.....	273
7.5	Sklejacz nazw argumentów, czyli operator <i>##</i>	275
7.6	Parametr aktualny makrodefinicji – w postaci tekstu	276
7.7	Dyrektywy kompilacji warunkowej.....	276
7.8	Dyrektywa <i>#error</i>	280
7.9	Dyrektywa <i>#line</i>	281
7.10	Wstawianie treści innych plików do tekstu kompilowanego właśnie pliku.....	281
7.11	Dyrektywy zależne od implementacji	283
7.12	Nazwy predefiniowane	283
7.13	Ćwiczenia	286

8 Tablice289

8.1	Co to jest tablica	289
8.2	Elementy tablicy	290
8.3	Inicjalizacja tablic	292
8.4	Przekazywanie tablicy do funkcji.....	293
8.5	Przykład z tablicą elementów typu <i>enum</i>	297
8.6	Tablice znakowe.....	299
8.7	Ćwiczenia	307

9 Tablice wielowymiarowe312

9.1	Tablica tablic	312
9.2	Przykład programu pracującego z tablicą dwuwymiarową.....	314
9.3	Gdzie w pamięci jest dany element tablicy.....	316
9.4	Typ wyrażeń związanych z tablicą wielowymiarową.....	316
9.5	Przesyłanie tablic wielowymiarowych do funkcji	318
9.6	Ćwiczenia	320

10 Wektory wielowymiarowe.....322

10.1	Najpierw przypomnienie istotnych tu cech klasy <i>vector</i>	322
10.2	Jak za pomocą klasy <i>vector</i> budować tablice wielowymiarowe	323
10.3	Funkcja pokazująca zawartość wektora dwuwymiarowego	324
10.4	Definicja dwuwymiarowego wektora – pustego.....	326
10.5	Definicja wektora dwuwymiarowego z listą inicjalizatorów	327
10.6	Wektor dwuwymiarowy o żądanych rozmiarach, choć bez inicjalizacji	328
10.7	Zmiana rozmiarów wektora dwuwymiarowego funkcją <i>resize</i>	329
10.8	Zmiany rozmiaru wektora 2D funkcjami <i>push_back</i> , <i>pop_back</i>	330
10.9	Zmniejszanie rozmiaru wektora dwuwymiarowego funkcją <i>pop_back</i>	333
10.10	Funkcje mogące modyfikować treść wektora 2D	333
10.11	Wysłanie rzędu wektora 2D do funkcji pracującej z wektorem 1D.....	335
10.12	Całość przykładu definiującego wektory dwuwymiarowe	336
10.13	Po co są dwuwymiarowe wektory nieprostokątne.....	336

10.14	Wektory trójwymiarowe.....	338
10.15	Sposoby definicji wektora 3D o ustalonych rozmiarach	341
10.16	Nadawanie pustemu wektorowi 3D wymaganych rozmiarów.....	345
10.16.1	Zmiana rozmiarów wektora 3D funkcjami <i>resize</i>	345
10.16.2	Zmiana rozmiarów wektora 3D funkcjami <i>push_back</i>	347
10.17	Trójwymiarowe wektory 3D – nieprostokątne.....	348
10.18	Ćwiczenia	352

11 Wskaźniki – wiadomości wstępne.....354

11.1	Wskaźniki mogą bardzo ułatwić życie	354
11.2	Definiowanie wskaźników.....	356
11.3	Praca ze wskaźnikiem.....	357
11.4	Definiowanie wskaźnika z użyciem <i>auto</i>	360
11.5	Wyrażenie <i>*wskaźnik jest lwartością</i>	361
11.6	Operator rzutowania <i>reinterpret_cast</i> a wskaźniki.....	361
11.7	Wskaźniki typu <i>void*</i>	364
11.8	Strzał na oślep – wskaźnik zawsze na coś wskazuje.....	366
11.8.1	Wskaźnik wolno porównać z adresem zero – <i>nullptr</i>	368
11.9	Ćwiczenia	368

12 Cztery domeny zastosowania wskaźników370

12.1	Zastosowanie wskaźników wobec tablic	370
12.1.1	Ćwiczenia z mechaniki ruchu wskaźnika.....	370
12.1.2	Użycie wskaźnika w pracy z tablicą.....	374
12.1.3	Arytmetyka wskaźników.....	378
12.1.4	Porównywanie wskaźników.....	380
12.2	Zastosowanie wskaźników w argumentach funkcji.....	381
12.2.1	Jeszcze raz o przesyłaniu tablic do funkcji.....	385
12.2.2	Odbieranie tablicy jako wskaźnik	385
12.2.3	Argument formalny będący wskaźnikiem do obiektu <i>const</i>	387
12.3	Zastosowanie wskaźników przy dostępie do konkretnych komórek pamięci.....	390
12.4	Rezerwacja obszarów pamięci.....	391
12.4.1	Operatory <i>new</i> i <i>delete</i> albo Oratorium Stworzenie Świata.....	392
12.4.2	Operator <i>new</i> a słowo kluczowe <i>auto</i>	396
12.4.3	Inicjalizacja obiektu tworzonego operatorem <i>new</i>	396
12.4.4	Operatorem <i>new</i> możemy także tworzyć obiekty stałe.....	397
12.4.5	Dynamiczna alokacja tablicy.....	398
12.4.6	Tablice wielowymiarowe tworzone operatorem <i>new</i>	399
12.4.7	Umiejscawiający operator <i>new</i>	402
12.4.8	„Przychodzimy, odchodzimy – cichuteńko, na...”	407
12.4.9	Zapas pamięci to nie studnia bez dna	409
12.4.10	Nowy sposób powiadomienia: rzucenie wyjątku <i>std::bad_alloc</i>	410
12.4.11	Funkcja <i>set_new_handler</i>	412
12.5	Ćwiczenia	414

13 Wskaźniki – runda trzecia418

13.1	Stałe wskaźniki	418
13.2	Stałe wskaźniki a wskaźniki do stałych.....	419
13.2.1	Wierzch i głębia	420
13.3	Definiowanie wskaźnika z użyciem <i>auto</i>	421
13.3.1	Symbol zastępczy <i>auto</i> a opuszczanie gwiazdki przy definiowaniu wskaźnika.....	424
13.4	Sposoby ustawiania wskaźników	426
13.5	Parada kłamców, czyli o rzutowaniu <i>const_cast</i>	428

13.6	Tablice wskaźników	432
13.7	Wariacje na temat C-stringów	434
13.8	Argumenty z linii wywołania programu	441
13.9	Ćwiczenia	444

14 Wskaźniki do funkcji446

14.1	Wskaźnik, który może wskazywać na funkcję	446
14.2	Ćwiczenia z definiowania wskaźników do funkcji	449
14.3	Wskaźnik do funkcji jako argument innej funkcji	455
14.4	Tablica wskaźników do funkcji	459
14.5	Użycie deklaracji <i>using</i> i <i>typedef</i> w świecie wskaźników	464
14.5.1	Alias przydatny w argumencie funkcji	464
14.5.2	Alias przydatny w definicji tablicy wskaźników do funkcji	465
14.6	Użycie <i>auto</i> lub <i>decltype</i> do automatycznego rozpoznania potrzebnego typu	466
14.7	Ćwiczenia	468

15 Przeładowanie nazwy funkcji.....470

15.1	Co oznacza przeładowanie	470
15.2	Przeładowanie od kuchni	473
15.3	Jak możemy przeładowywać, a jak się nie da?	473
15.4	Czy przeładowanie nazw funkcji jest techniką orientowaną obiektowo?	476
15.5	Linkowanie z modułami z innych języków	477
15.6	Przeładowanie a zakres ważności deklaracji funkcji	478
15.7	Rozważania o identyczności lub odmienności typów argumentów	480
15.7.1	Przeładowanie a typy tworzone z <i>using</i> lub <i>typedef</i> oraz typy <i>enum</i>	481
15.7.2	Tablica a wskaźnik	481
15.7.3	Pewne szczegóły o tablicach wielowymiarowych	482
15.7.4	Przeładowanie a referencja	484
15.7.5	Identyczność typów: <i>T</i> , <i>const T</i> , <i>volatile T</i>	485
15.7.6	Przeładowanie a typy: <i>T*</i> , <i>volatile T*</i> , <i>const T*</i>	486
15.7.7	Przeładowanie a typy: <i>T&</i> , <i>volatile T&</i> , <i>const T&</i>	487
15.8	Adres funkcji przeładowanej	488
15.8.1	Zwrot rezultatu będącego adresem funkcji przeładowanej	490
15.9	Kulisy dopasowywania argumentów do funkcji przeładowanych	492
15.10	Etapy dopasowania	493
15.10.1	Etap 1. Dopasowanie dokładne	493
15.10.2	Etap 1a. Dopasowanie dokładne, ale z tzw. trywialną konwersją	494
15.10.3	Etap 2. Dopasowanie z awansem (z promocją)	495
15.10.4	Etap 3. Próba dopasowania za pomocą konwersji standardowych	497
15.10.5	Etap 4. Dopasowanie z użyciem konwersji zdefiniowanych przez użytkownika	499
15.10.6	Etap 5. Dopasowanie do funkcji z wielokropkiem	499
15.11	Wskaźników nie dopasowuje się inaczej niż dosłownie	499
15.12	Dopasowywanie wywołań z kilkoma argumentami	500
15.13	Ćwiczenia	501

16 Klasy504

16.1	Typy definiowane przez użytkownika	504
16.2	Składniki klasy	506
16.3	Składnik będący obiektem	507
16.4	Kapsułowanie	508
16.5	Ukrywanie informacji	509
16.6	Klasa a obiekt	512
16.7	Wartości wstępne w składnikach nowych obiektów. Inicjalizacja „w klasie”	514
16.8	Funkcje składowe	517

16.8.1	Posługiwanie się funkcjami składowymi	517
16.8.2	Definiowanie funkcji składowych	518
16.9	Jak to właściwie jest? (<i>this</i>).....	523
16.10	Odwołanie się do publicznych danych składowych obiektu	525
16.11	Zasłanianie nazw	526
16.11.1	Nie sięgaj z klasy do obiektów globalnych.....	529
16.12	Przeładowanie i zastąpienie równocześnie	530
16.13	Nowa klasa? Osobny plik!.....	530
16.13.1	Poznajmy praktyczną realizację wieloplikowego programu	533
16.13.2	Zasada umieszczania dyrektywy <i>using namespace</i> w plikach	545
16.14	Przesyłanie do funkcji argumentów będących obiektami.....	545
16.14.1	Przesyłanie obiektu przez wartość	545
16.14.2	Przesyłanie przez referencję	547
16.15	Konstruktor – pierwsza wzmianka	548
16.16	Destruktor – pierwsza wzmianka	553
16.17	Składnik statyczny.....	557
16.17.1	Do czego może się przydać składnik statyczny w klasie?.....	566
16.18	Statyczna funkcja składowa.....	566
16.18.1	Deklaracja składnika statycznego mająca inicjalizację „w klasie”	571
16.19	Funkcje składowe typu <i>const</i> oraz <i>volatile</i>	577
16.19.1	Przeładowanie a funkcje składowe <i>const</i> i <i>volatile</i>	581
16.20	Struktura	582
16.21	Klasa będąca agregatem. Klasa bez konstruktora	582
16.22	Funkcje składowe z przydomkiem <i>constexpr</i>	585
16.23	Specyfikator <i>mutable</i>	591
16.24	Bardziej rozbudowany przykład zastosowania klasy.....	593
16.25	Ćwiczenia	603