Opus Magnum C++11 : programowanie w języku C++. T. 1 / Jerzy Grębosz. – Wydanie II poprawione. – Gliwice, copyright © 2020

Spis treści

0 Proszę tego nie czytać! 1
0.1 Zaprzyjaźnijmy 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 bool w roli warunku 21
2.1.3 Stare dobre sposoby z dawnego C++ 21
2.2 Instrukcja warunkowa if 22
2.3 Pętla while 26
2.4 Pętla do...while 27
2.5 Pętla for 28
2.6 Instrukcja switch 31
2.7 Co wybrać: switch czy if...else? 33
2.8 Instrukcja break 36
2.9 Instrukcja goto 37
2.10 Instrukcja continue 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 Typy 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 bool 63
3.7.2 Stałe będące liczbami całkowitymi 63
3.7.3 Stałe reprezentujące liczby zmiennoprzecinkowe 66
3.7.4 Stała dosłowna nullptr - 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 void 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) const 87
3.13 Specyfikator (przydomek) constexpr 88
3.14 Obiekty register 92
3.15 Specyfikator volatile 92
3.16 using oraz typedef- tworzenie dodatkowej nazwy typu 93
3.17 Typy wyliczeniowe enum 96
3.17.1 Dawne zwykłe enum a nowe zakresowe enum class 103
3.17.2 Kilka uwag dla wtajemniczonych 105
3.18 auto, czyli automatyczne rozpoznawanie typu definiowanego obiektu 106
3.19 decltype - operator do określania typu zadanego wyrażenia 109
3.20 Inicjalizacja z pustą klamrą { }, czyli wartością domniemaną 111
3.21 Przydomek alignas - adresy równe i równiejsze 113
3.22 Ćwiczenia 115

4 Operatory **119**
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 II 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 sizeof 135
4.9 Operator noexcept 137
4.10 Deklaracja static_assert 137
4.11 Operator alignof 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 static_cast 143
4.12.4 Operator const_cast 145
4.12.5 Operator dynamic_cast 146
4.12.6 Operator reinterpret_cast 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 wzmianka 156
5.1 Typ std::string do pracy z tekstami 156
5.2 Typ vector - długi rząd obiektów 161
5.3 Zakresowe for 169
5.4 Ćwiczenia 172

6 Funkcje 174
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 auto, a inicjalizowany rezultatem funkcji 179
6.4.2 O zwracaniu (lub niezwracaniu) rezultatu przez funkcję main 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 inline (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 constexpr 240
6.21.1 Wymogi, które musi spełniać funkcja constexpr (w standardzie C++11) 242
6.21.2 Przykład pokazujący aspekty funkcji COHStexpr 243
6.21.3 Argumenty funkcji COHStexpr, będące referencjami 252
6.22 Definiowanie referencji przy użyciu słowa auto 253
6.22.1 Gdy inicjalizatorem jest wywołanie funkcji zwracającej referencję 260
6.23 Ćwiczenia 263

7 Preprocesor 270
7.1 Dyrektywa pusta # 270
7.2 Dyrektywa #define 270
7.3 Dyrektywa #undef 272
7.4 Makrodefinicje 273
7.5 Sklejacz nazw argumentów, czyli operator ## 275
7.6 Parametr aktualny makrodefinicji - w postaci tekstu 276
7.7 Dyrektywy kompilacji warunkowej 276
7.8 Dyrektywa #error 280
7.9 Dyrektywa #line 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 Tablice 289
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 enum 297
8.6 Tablice znakowe 299
8.7 Ćwiczenia 307

9 Tablice wielowymiarowe 312
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 vector 322
10.2 Jak za pomocą klasy vector 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 resize 329
10.8 Zmiany rozmiaru wektora 2D funkcjami push_back, pop_back 330
10.9 Zmniejszanie rozmiaru wektora dwuwymiarowego funkcją pop_back 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 resize 345
10.16.2 Zmiana rozmiarów wektora 3D funkcjami push_back 347
10.17 Trójwymiarowe wektory 3D - nieprosto padło ścienne 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 auto 360
11.5 Wyrażenie *wskaźnik jest lwartością 361
11.6 Operator rzutowania reinterpret_cast a wskaźniki 361
11.7 Wskaźniki typu void* 364
11.8 Strzał na oślep - wskaźnik zawsze na coś wskazuje 366
11.8.1 Wskaźnik wolno porównać z adresem zero - nullptr 368

11.9 Ćwiczenia 368

12 Cztery domeny zastosowania wskaźników 370
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 const 387
12.3 Zastosowanie wskaźników przy dostępie do konkretnych komórek pamięci 390
12.4 Rezerwacja obszarów pamięci 391
12.4.1 Operatory new i delete albo Oratorium Stworzenie Świata 392
12.4.2 Operator new a słowo kluczowe auto 396
12.4.3 Inicjalizacja obiektu tworzonego operatorem new 396
12.4.4 Operatorem new możemy także tworzyć obiekty stałe 397
12.4.5 Dynamiczna alokacja tablicy 398
12.4.6 Tablice wielowymiarowe tworzone operatorem new 399
12.4.7 Umiejscawiający operator new 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 std::bad_alloc 410
12.4.11 Funkcja set_new_handler 412
12.5 Ćwiczenia 414

13 Wskaźniki – runda trzecia 418
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 auto 421
13.3.1 Symbol zastępczy auto a opuszczanie gwiazdki przy definiowaniu wskaźnika 424
13.4 Sposoby ustawiania wskaźników 426
13.5 Parada kłamców, czyli o rzutowaniu const_cast 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 funkcji 446
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 using i typedef 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 auto lub decltype 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 using lub typedef oraz typy enum 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: T, const T, volatile T 485
15.7.6 Przeładowanie a typy: T*, volatile T*, const T* 486
15.7.7 Przeładowanie a typy: T&, volatile T&, const T& 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 Klasy 504
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? (this) 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 zasłonięcie równocześnie 530
16.13 Nowa klasa? Osobny plik! 530
16.13.1 Poznajmy praktyczną realizację wieloplikowego programu 533
16.13.2 Zasada umieszczania dyrektywy using namespace 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 const oraz volatile 577
16.19.1 Przeładowanie a funkcje składowe const i volatile 581
16.20 Struktura 582
16.21 Klasa będąca agregatem. Klasa bez konstruktora 582
16.22 Funkcje składowe z przydomkiem constexpr 585
16.23 Specyfikator mutable 591
16.24 Bardziej rozbudowany przykład zastosowania klasy 593
16.25 Ćwiczenia 603

oprac. BPK