

Spis treści (obu tomów)

Tom I

0	Proszę nie czytać tego!	19
1	Startujemy!	24
1.1	Pierwszy program	24
1.2	Drugi program	29
1.3	Ćwiczenia	35
2	Instrukcje sterujące	37
2.1	Prawda – Fałsz, czyli o warunkach	37
2.1.1	Wyrażenie logiczne	37
2.1.2	Zmienne logiczne <code>bool</code> jako warunek	38
2.1.3	Stare dobre sposoby z dawnego C++	38
2.2	Instrukcja warunkowa <code>if</code>	39
2.3	Pętla <code>while</code>	43
2.4	Pętla <code>do...while</code>	44
2.5	Pętla <code>for</code>	45
2.6	Instrukcja <code>switch</code>	47
2.7	Co wybrać: <code>switch</code> czy <code>if...else?</code>	50
2.8	Instrukcja <code>break</code>	52
2.9	Instrukcja <code>goto</code>	54
2.10	Instrukcja <code>continue</code>	55
2.11	Klamry w instrukcjach sterujących	56
2.12	Ćwiczenia	57

3	Typy	60
3.1	Deklaracje typów	60
3.2	Systematyka typów z języka C++	61
3.3	Typy fundamentalne	62
3.3.1	Definiowanie obiektów „w biegu”	66
3.4	Stałe dosłowne	69
3.4.1	Stałe będące liczbami całkowitymi	69
3.4.2	Stałe reprezentujące liczby zmiennoprzecinkowe	71
3.4.3	Stałe znakowe	72
3.4.4	Stałe tekstowe, napisy, albo po prostu stringi	74
3.5	Typy złożone	77
3.6	Typ <i>void</i>	78
3.7	Zakres ważności nazwy obiektu, a czas życia obiektu	78
3.7.1	Zakres: lokalny	79
3.7.2	Zakres: blok funkcji	79
3.7.3	Zakres: obszar pliku	80
3.7.4	Zakres: obszar klasy	80
3.7.5	Zakres określony przez przestrzeń nazw	80
3.8	Zasłanianie nazw	85
3.9	Specyfikator (przydomek) <i>const</i>	87
3.9.1	Pojedynek: <i>const</i> contra <i>#define</i>	88
3.10	Obiekty <i>register</i>	89
3.11	Specyfikator <i>volatile</i>	90
3.12	Instrukcja <i>typedef</i>	91
3.13	Typy wyliczeniowe <i>enum</i>	92
3.14	Ćwiczenia	95
4	Operatory	99
4.1	Operatory arytmetyczne	99
4.1.1	Operator %, czyli reszta z dzielenia (modulo)	100
4.1.2	Jednoargumentowe operatory + i -	101
4.1.3	Operatory inkrementacji i dekrementacji	102
4.1.4	Operator przypisania =	103
4.2	Operatory logiczne	104
4.2.1	Operatory relacji	104
4.2.2	Operatory sumy logicznej i iloczynu logicznego &&	106
4.2.3	Wykrzyknik ! czyli operator negacji	107
4.3	Operatory bitowe	108
4.3.1	Przesunięcie w lewo <<	109
4.3.2	Przesunięcie w prawo >>	109
4.3.3	Bitowe operatory sumy, iloczynu, negacji, różnicy symetrycznej	110
4.4	Różnica między operatorami logicznymi, a operatorami bitowymi	111
4.5	Pozostałe operatory przypisania	113
4.6	Operator uzyskiwania adresu (operator &)	114

4.7	Wyrażenie warunkowe	115
4.8	Operator <i>sizeof</i>	116
4.9	Operatory rzutowania	117
4.9.1	Rzutowanie według tradycyjnych (nie zalecanych) sposobów	117
4.9.2	Rzutowanie za pomocą nowych operatorów rzutowania	119
4.9.3	Operator <i>static_cast</i>	119
4.9.4	Operator <i>const_cast</i>	121
4.9.5	Operator <i>dynamic_cast</i>	123
4.9.6	Operator <i>reinterpret_cast</i>	123
4.10	Operator: przecinek	124
4.11	Priorytety operatorów	125
4.12	Łączność operatorów	128
4.13	Ćwiczenia	128

5 Funkcje133

5.1	Funkcja często wywołuje inną funkcję	136
5.2	Zwracanie przez funkcję rezultatu	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?	144
5.7	Argumenty domniemane	146
5.7.1	Ciekawostki na temat argumentów domniemanych	148
5.8	Nienazwany argument	153
5.9	Funkcje <i>inline</i> (w linii)	155
5.10	Przypomnienie o zakresie ważności nazw deklarowanych wewnątrz funkcji	158
5.11	Wybór zakresu ważności nazwy i czasu życia obiektu	159
5.11.1	Obiekty globalne	159
5.11.2	Obiekty automatyczne	160
5.11.3	Obiekty lokalne statyczne	161
5.12	Funkcje w programie składającym się z kilku plików	164
5.12.1	Nazwy statyczne globalne	168
5.13	Funkcje rekurencyjne	170
5.14	Funkcje biblioteczne	179
5.15	Ćwiczenia	182

6 Preprocesor188

6.1	Na pomoc rodakom	188
6.2	Dyrektywa <i>#define</i>	189
6.3	Dyrektywa <i>#undef</i>	192
6.4	Makrodefinicje	192
6.5	Sklejacz nazw, czyli operator <i>##</i>	195
6.6	Zamiana parametru aktualnego makrodefinicji na string	195

6.7	Dyrektywy kompilacji warunkowej	197
6.8	Dyrektywa <code>#error</code>	201
6.9	Dyrektywa <code>#line</code>	202
6.10	Wstawianie treści innych plików w tekst kompilowanego własnie pliku	202
6.11	Dyrektywa pusta <code>#</code>	204
6.12	Dyrektywy zależne od implementacji	204
6.13	Nazwy predefiniowane	205
6.14	Ćwiczenia	206

7 Tablice210

7.1	Elementy tablicy	211
7.2	Inicjalizacja tablic	213
7.3	Przekazywanie tablicy do funkcji	214
7.4	Przykład z tablicą elementów typu <code>enum</code>	218
7.5	Tablice znakowe	220
7.6	Tablice wielowymiarowe	228
7.6.1	Typ wyrażeń związanych z tablicą wielowymiarową	231
7.6.2	Przesyłanie tablic wielowymiarowych do funkcji	233
7.7	Ćwiczenia	234

8 Wskaźniki239

8.1	Wskaźniki mogą bardzo ułatwić życie	239
8.2	Definiowanie wskaźników	241
8.3	Praca ze wskaźnikiem	242
8.4	L-wartość	245
8.5	Operator rzutowania <code>reinterpret_cast</code> , a wskaźniki	246
8.6	Wskaźniki typu <code>void</code>	249
8.7	Cztery domeny zastosowania wskaźników	251
8.8	Zastosowanie wskaźników wobec tablic	251
8.8.1	Ćwiczenia z mechaniki ruchu wskaźnika	251
8.8.2	Użycie wskaźnika w pracy z tablicą	255
8.8.3	Arytmetyka wskaźników	259
8.8.4	Porównywanie wskaźników	261
8.8.5	Wskaźnik można porównać z adresem 0 (zero)	263
8.9	Zastosowanie wskaźników w argumentach funkcji	264
8.9.1	Jeszcze raz o przesyłaniu tablic do funkcji	267
8.9.2	Odbieranie tablicy jako wskaźnika	268
8.9.3	Argument formalny będący wskaźnikiem do obiektu <code>const</code>	269
8.10	Zastosowanie wskaźników przy dostępie do konkretnych komórek pamięci	272
8.11	Rezerwacja obszarów pamięci	273
8.11.1	Operatory <code>new</code> i <code>delete</code> albo Oratorium Stworzenie Świata	274
8.11.2	Dynamiczna alokacja tablicy	277
8.11.3	Tablice wielowymiarowe tworzone operatorem <code>new</code>	278

8.11.4	Umiejscawiający operator <i>new</i>	280
8.11.5	"Przychodzimy, odchodzimy – cichuteńko, na..."	285
8.11.6	Zapas pamięci to nie jest studnia bez dna	287
8.11.7	Funkcja <i>set_new_handler</i>	291
8.11.8	Pojedynek: <i>new</i> contra <i>malloc</i>	292
8.12	Stałe wskaźniki	293
8.13	Stałe wskaźniki, a wskaźniki do stałych	294
8.14	Strzał na ośle – Wskaźnik zawsze pokazuje na coś	295
8.15	Sposoby ustawiania wskaźników	296
8.16	Parada kłamców, czyli o rzutowaniu <i>const_cast</i>	298
8.17	Tablice wskaźników	302
8.18	Wariacje na temat C-stringów	304
8.19	Wskaźniki do funkcji	311
8.19.1	Ćwiczenia z definiowania wskaźników do funkcji	315
8.19.2	Wskaźnik do funkcji jako argument innej funkcji	320
8.19.3	Tablica wskaźników do funkcji	329
8.20	Argumenty z linii wywołania programu	334
8.21	Ćwiczenia	336

9 Przeladowanie nazwy funkcji344

9.1	Co to znaczy: przeladowanie	344
9.2	Blizsze szczegóły przeladowania	347
9.3	Czy przeladowanie nazw funkcji jest techniką obiektowo orientowaną?	350
9.4	Linkowanie z modułami z innych języków	351
9.5	Przeladowanie, a zakres ważności deklaracji funkcji	352
9.6	Rozważania o identyczności lub odmienności typów argumentów	354
9.6.1	Przeladowanie, a <i>typedef</i> i <i>enum</i>	354
9.6.2	Tablica, a wskaźnik	354
9.6.3	Pewne szczegóły o tablicach wielowymiarowych	356
9.6.4	Przeladowanie, a referencja	358
9.6.5	Identyczność typów: <i>T</i> , <i>const T</i> , <i>volatile T</i>	358
9.6.6	Przeladowanie – a typy: <i>T*</i> , <i>volatile T*</i> , <i>const T*</i>	359
9.6.7	Przeladowanie – a typy: <i>T&</i> , <i>volatile T&</i> , <i>const T&</i>	361
9.7	Adres funkcji przeladowanej	361
9.7.1	Zwrot rezultatu będącego adresem funkcji przeladowanej	363
9.8	Kulisy dopasowywania argumentów do funkcji przeladowanych	365
9.9	Etapy dopasowania	366
9.9.1	Etap 1. Dopasowanie dokładne	367
9.9.2	Etap 1a. Dopasowanie dokładne, ale z tzw. trywialną konwersją	367
9.9.3	Etap 2. Dopasowanie z awansem (z promocją)	368
9.9.4	Etap 3. Próba dopasowania za pomocą konwersji standardowych	370
9.9.5	Etap 4. Próba dopasowania z użyciem konwersji zdefiniowanych przez użytkownika	372
9.9.6	Etap 5. Próba dopasowania do funkcji z wielokropkiem	372
9.9.7	Wskaźników nie dopasowuje się inaczej niż dosłownie	372

9.10	Dopasowywanie wywołań z kilkoma argumentami	373
9.11	Ćwiczenia	374

10 Klasy377

10.1	Typy definiowane przez użytkownika	377
10.2	Składniki klasy	379
10.3	Składnik będący obiektem	380
10.4	Kapsułowanie	381
10.5	Ukrywanie informacji	382
10.6	Klasa, a obiekt	385
10.7	Funkcje składowe	387
10.7.1	Posługiwanie się funkcjami składowymi	387
10.7.2	Definiowanie funkcji składowych	388
10.8	Jak to właściwie jest? (<i>this</i>)	394
10.9	Odwołanie się do publicznych danych składowych	396
10.10	Zasłanianie nazw	397
10.10.1	Nie sięgaj z klasy do obiektów globalnych	400
10.11	Przeładowanie i zasłonięcie równocześnie	401
10.12	Nowa klasa? Osobny plik!	401
10.13	Przesyłanie do funkcji argumentów będących obiektami	412
10.13.1	Przesyłanie obiektu przez wartość	412
10.13.2	Przesyłanie przez referencję	415
10.14	Konstruktor – pierwsza wzmianka	415
10.15	Destruktor – pierwsza wzmianka	420
10.16	Składnik statyczny	423
10.16.1	Deklaracja składnika statycznego połączona z inicjalizacją	428
10.17	Statyczna funkcja składowa	432
10.18	Do czego może nam się przydać składnik statyczny w klasie?	436
10.19	Funkcje składowe typu <i>const</i> oraz <i>volatile</i>	436
10.19.1	Przeładowanie, a funkcje składowe <i>const</i> i <i>volatile</i>	440
10.20	Specyfikator <i>mutable</i>	440
10.21	Ćwiczenia	452

11 Biblioteczna klasa *std::string* do operacji z tekstami.....456

11.1	Przykład programu z użyciem klasy <i>string</i>	458
11.2	Definiowanie obiektów klasy <i>string</i>	463
11.3	Użycie operatorów <i>=</i> , <i>+</i> , <i>+=</i> , w pracy ze <i>stringami</i>	467
11.3.1	Jak umieścić w tekście liczbę?	468
11.4	Pojemność, rozmiar i długość <i>stringu</i>	469
11.4.1	Funkcje <i>size()</i> i <i>length()</i>	470
11.4.2	Funkcja składowa <i>empty</i>	470
11.4.3	Funkcja składowa <i>max_size</i>	471
11.4.4	Funkcja składowa <i>capacity</i>	471

11.4.5	Funkcja składowa <i>reserve</i>	473
11.4.6	<i>resize</i> – zmiana długości stringu „na siłę”	473
11.4.7	Funkcja składowa <i>clear</i>	475
11.5	Użycie operatora <i>[]</i> oraz funkcji <i>at</i>	476
11.5.1	Działanie operatora <i>[]</i>	477
11.5.2	Działanie funkcji składowej <i>at</i>	477
11.6	Praca z fragmentem stringu, czyli z sub-stringiem	480
11.7	Funkcja składowa <i>substr</i>	481
11.8	Szukanie zadanego substringu w obiekcie klasy <i>string</i> – funkcje <i>find</i>	481
11.9	Szukanie rozpoczynane od końca stringu.....	485
11.10	Szukanie w stringu jednego ze znaków z zadanego zestawu	486
11.11	Usuwanie znaków ze stringu – funkcje <i>erase</i>	488
11.12	Wstawianie znaków do już istniejącego stringu – funkcje <i>insert</i>	490
11.13	Zamiana części znaków na inne znaki – <i>replace</i>	492
11.14	Zamiana zawartości obiektu klasy <i>string</i> na C-string.....	496
11.15	Zagłębienie do wnętrza obiektu klasy <i>string</i> funkcją <i>data</i>	499
11.16	W porządku alfabetycznym – czyli porównywanie stringów	501
11.16.1	Porównywanie stringów funkcjami <i>compare</i>	501
11.16.2	Porównywanie stringów przy użyciu operatorów <i>==</i> , <i>!=</i> , <i><</i> , <i>></i> , <i><=</i> , <i>>=</i>	506
11.17	Zamiana treści stringu na małe (lub wielkie) litery	508
11.18	Kopiowanie treści obiektu klasy <i>string</i> do wybranej tablicy znakowej – funkcja <i>copy</i>	513
11.19	Wzajemna zamiana treści dwóch obiektów klasy <i>string</i> – funkcja <i>swap</i>	514
11.20	Przypisanie do obiektu klasy <i>string</i> , funkcja <i>assign</i>	515
11.21	Dopisywanie do końca stringu za pomocą funkcji <i>append</i>	517
11.22	Wczytywanie z klawiatury długiego stringu o nieznanym wcześniej długości – <i>getline</i>	518
11.22.1	Pułapka – czyli jak <i>getline</i> może Cię zaskoczyć	521
11.23	Iteratory stringu.....	525
11.23.1	Iterator do obiektu stałego	529
11.23.2	Funkcje składowe klasy <i>string</i> pracujące z iteratorami	530
11.24	Bryk – czyli "pamięć zewnętrzna" programisty	537
11.25	Ćwiczenia	545

12 Deklaracje przyjaźni

551

12.0.1	Klasy zaprzyjaźnione	560
12.0.2	Słowo o zakresie	562