

Spis treści

Wstęp 19

Słowo do studentów 21

Słowo do nauczycieli 22

Standard ISO C++ 23

Pomoc 23

Podziękowania 24

Uwagi do czytelnika 25

0.1. Struktura książki 26

0.1.1. Informacje ogólne 27

0.1.2. Ćwiczenia, praca domowa itp. 28

0.1.3. Po przeczytaniu tej książki 29

0.2. Filozofia nauczania i uczenia się 29

0.2.1. Kolejność tematów 32

0.2.2. Programowanie a język programowania 34

0.2.3. Przenośność 34

0.3. Programowanie a informatyka 35

0.4. Kreatywność i rozwiązywanie problemów 35

0.5. Uwagi i komentarze czytelników 35

0.6. Bibliografia 36

0.7. Noty biograficzne 36

Bjarne Stroustrup 36

Lawrence "Pete" Petersen 37

Rozdział 1. Komputery, ludzie i programowanie 39

1.1. Wstęp 40

- 1.2. Oprogramowanie 40
- 1.3. Ludzie 42
- 1.4. Informatyka 45
- 1.5. Komputery są wszędzie 46
 - 1.5.1. Komputery z ekranem i bez 46
 - 1.5.2. Transport 47
 - 1.5.3. Telekomunikacja 48
 - 1.5.4. Medycyna 50
 - 1.5.5. Informacja 51
 - 1.5.6. Sięgamy w kosmos 53
 - 1.5.7. I co z tego 54
- 1.6. Ideały dla programistów 54

CZĘŚĆ I. PODSTAWY 61

Rozdział 2. Witaj, świecie! 63

- 2.1. Programy 64
- 2.2. Klasyczny pierwszy program 64
- 2.3. Kompilacja 67
- 2.4. Łączenie 69
- 2.5. Środowiska programistyczne 70

Rozdział 3. Obiekty, typy i wartości 77

- 3.1. Dane wejściowe 78
- 3.2. Zmienne 80
- 3.3. Typy danych wejściowych 81
- 3.4. Operacje i operatory 82
- 3.5. Przypisanie i inicjalizacja 85
 - 3.5.1. Przykład wykrywania powtarzających się słów 87
- 3.6. Złożone operatory przypisania 89

3.6.1. Przykład zliczania powtarzających się słów 89

3.7. Nazwy 90

3.8. Typy i obiekty 92

3.9. Kontrola typów 94

3.9.1. Konwersje bezpieczne dla typów 95

3.9.2. Konwersje niebezpieczne dla typów 96

Rozdział 4. Wykonywanie obliczeń 103

4.1. Wykonywanie obliczeń 104

4.2. Cele i narzędzia 105

4.3. Wyrażenia 107

4.3.1. Wyrażenia stałe 108

4.3.2. Operatory 110

4.3.3. Konwersje 111

4.4. Instrukcje 112

4.4.1. Selekcja 113

4.4.2. Iteracja 119

4.5. Funkcje 123

4.5.1. Po co zaprzętać sobie głowę funkcjami 124

4.5.2. Deklarowanie funkcji 126

4.6. Wektor 126

4.6.1. Przeglądanie zawartości wektora 128

4.6.2. Powiększanie wektora 128

4.6.3. Przykład wczytywania liczb do programu 129

4.6.4. Przykład z użyciem tekstu 131

4.7. Właściwości języka 133

Rozdział 5. Błędy 139

5.1. Wstęp 140

| | |
|--|-----|
| 5.2. Źródła błędów | 141 |
| 5.3. Błędy kompilacji | 142 |
| 5.3.1. Błędy składni | 142 |
| 5.3.2. Błędy typów | 143 |
| 5.3.3. Niebłędy | 144 |
| 5.4. Błędy konsolidacji | 145 |
| 5.5. Błędy czasu wykonania | 146 |
| 5.5.1. Rozwiązywanie problemu przez wywołującego | 147 |
| 5.5.2. Rozwiązywanie problemu przez wywoływane | 148 |
| 5.5.3. Raportowanie błędów | 149 |
| 5.6. Wyjątki | 151 |
| 5.6.1. Nieprawidłowe argumenty | 151 |
| 5.6.2. Błędy zakresu | 152 |
| 5.6.3. Nieprawidłowe dane wejściowe | 154 |
| 5.6.4. Błędy zawężania zakresu | 156 |
| 5.7. Błędy logiczne | 157 |
| 5.8. Szacowanie | 159 |
| 5.9. Debugowanie | 161 |
| 5.9.1. Praktyczna rada dotycząca debugowania | 162 |
| 5.10. Warunki wstępne i końcowe | 165 |
| 5.10.1. Warunki końcowe | 167 |
| 5.11. Testowanie | 168 |
| Rozdział 6. Pisanie programu | 175 |
| 6.1. Problem | 176 |
| 6.2. Przemyślenie problemu | 176 |
| 6.2.1. Etapy rozwoju oprogramowania | 177 |
| 6.2.2. Strategia | 177 |

| | |
|---|-----|
| 6.3. Wracając do kalkulatora | 179 |
| 6.3.1. Pierwsza próba | 180 |
| 6.3.2. Tokeny | 182 |
| 6.3.3. Implementowanie tokenów | 183 |
| 6.3.4. Używanie tokenów | 185 |
| 6.3.5. Powrót do tablicy | 186 |
| 6.4. Gramatyki | 188 |
| 6.4.1. Dygresja - gramatyka języka angielskiego | 192 |
| 6.4.2. Pisanie gramatyki | 193 |
| 6.5. Zamiana gramatyki w kod | 194 |
| 6.5.1. Implementowanie zasad gramatyki | 194 |
| 6.5.2. Wyrażenia | 195 |
| 6.5.3. Składniki | 198 |
| 6.5.4. Podstawowe elementy wyrażeń | 200 |
| 6.6. Wypróbowywanie pierwszej wersji | 200 |
| 6.7. Wypróbowywanie drugiej wersji | 204 |
| 6.8. Strumienie tokenów | 205 |
| 6.8.1. Implementacja typu Token_stream | 207 |
| 6.8.2. Wczytywanie tokenów | 208 |
| 6.8.3. Wczytywanie liczb | 210 |
| 6.9. Struktura programu | 210 |
| Rozdział 7. Kończenie programu | 217 |
| 7.1. Wprowadzenie | 218 |
| 7.2. Wejście i wyjście | 218 |
| 7.3. Obsługa błędów | 220 |
| 7.4. Liczby ujemne | 224 |
| 7.5. Reszta z dzielenia | 225 |

| | |
|--|-----|
| 7.6. Oczyszczanie kodu | 226 |
| 7.6.1. Stałe symboliczne | 227 |
| 7.6.2. Użycie funkcji | 229 |
| 7.6.3. Układ kodu | 230 |
| 7.6.4. Komentarze | 231 |
| 7.7. Odzyskiwanie sprawności po wystąpieniu błędu | 232 |
| 7.8. Zmienne | 235 |
| 7.8.1. Zmienne i definicje | 235 |
| 7.8.2. Wprowadzanie nazw | 239 |
| 7.8.3. Nazwy predefiniowane | 242 |
| 7.8.4. Czy to już koniec? | 242 |
| Rozdział 8. Szczegóły techniczne - funkcje itp. | 247 |
| 8.1. Szczegóły techniczne | 248 |
| 8.2. Deklaracje i definicje | 249 |
| 8.2.1. Rodzaje deklaracji | 252 |
| 8.2.2. Deklaracje stałych i zmiennych | 252 |
| 8.2.3. Domyślna inicjalizacja | 254 |
| 8.3. Pliki nagłówkowe | 254 |
| 8.4. Zakres | 256 |
| 8.5. Wywoływanie i wartość zwrotna funkcji | 261 |
| 8.5.1. Deklarowanie argumentów i typu zwróconego | 261 |
| 8.5.2. Zwracanie wartości | 263 |
| 8.5.3. Przekazywanie przez wartość | 264 |
| 8.5.4. Przekazywanie argumentów przez stałą referencję | 265 |
| 8.5.5. Przekazywanie przez referencję | 267 |
| 8.5.6. Przekazywanie przez wartość a przez referencję | 269 |
| 8.5.7. Sprawdzanie argumentów i konwersja | 271 |

8.5.8. Implementacja wywołań funkcji 272

8.5.9. Funkcje constexpr 276

8.6. Porządek wykonywania instrukcji 277

8.6.1. Wartościowanie wyrażeń 278

8.6.2. Globalna inicjalizacja 279

8.7. Przestrzenie nazw 280

8.7.1. Dyrektywy i deklaracje using 281

Rozdział 9. Szczegóły techniczne - klasy itp. 287

9.1. Typy zdefiniowane przez użytkownika 288

9.2. Klasy i składowe klas 289

9.3. Interfejs i implementacja 289

9.4. Tworzenie klas 291

9.4.1. Struktury i funkcje 291

9.4.2. Funkcje składowe i konstruktory 293

9.4.3. Ukrywanie szczegółów 295

9.4.4. Definiowanie funkcji składowych 296

9.4.5. Odwoływanie się do bieżącego obiektu 298

9.4.6. Raportowanie błędów 299

9.5. Wyliczenia 300

9.5.1. "Zwykłe" wyliczenia 301

9.6. Przeciążanie operatorów 302

9.7. Interfejsy klas 303

9.7.1. Typy argumentów 304

9.7.2. Kopiowanie 306

9.7.3. Konstruktory domyślne 307

9.7.4. Stałe funkcje składowe 310

9.7.5. Składowe i funkcje pomocnicze 311

9.8. Klasa Date 313

CZĘŚĆ II. WEJŚCIE I WYJŚCIE 321

Rozdział 10. Strumienie wejścia i wyjścia 323

10.1. Wejście i wyjście 324

10.2. Model strumieni wejścia i wyjścia 325

10.3. Pliki 327

10.4. Otwieranie pliku 328

10.5. Odczytywanie i zapisywanie plików 330

10.6. Obsługa błędów wejścia i wyjścia 332

10.7. Wczytywanie pojedynczej wartości 334

10.7.1. Rozłożenie problemu na mniejsze części 336

10.7.2. Oddzielenie warstwy komunikacyjnej od funkcji 339

10.8. Definiowanie operatorów wyjściowych 340

10.9. Definiowanie operatorów wejściowych 341

10.10. Standardowa pętla wejściowa 342

10.11. Wczytywanie pliku strukturalnego 343

10.11.1. Reprezentacja danych w pamięci 344

10.11.2. Odczytywanie struktur wartości 345

10.11.3. Zmianianie reprezentacji 349

Rozdział 11. Indywidualizacja operacji wejścia i wyjścia 353

11.1. Regularność i nieregularność 354

11.2. Formatowanie danych wyjściowych 354

11.2.1. Wysyłanie na wyjście liczb całkowitych 355

11.2.2. Przyjmowanie na wejściu liczb całkowitych 356

11.2.3. Wysyłanie na wyjście liczb zmiennoprzecinkowych 357

11.2.4. Precyzja 358

11.2.5. Pola 359

| | |
|---|-----|
| 11.3. Otwieranie plików i pozycjonowanie | 361 |
| 11.3.1. Tryby otwierania plików | 361 |
| 11.3.2. Pliki binarne | 362 |
| 11.3.3. Pozycjonowanie w plikach | 365 |
| 11.4. Strumienie łańcuchowe | 365 |
| 11.5. Wprowadzanie danych wierszami | 367 |
| 11.6. Klasyfikowanie znaków | 368 |
| 11.7. Stosowanie niestandardowych separatorów | 370 |
| 11.8. Zostało jeszcze tyle do poznania | 376 |
| Rozdział 12. Projektowanie klas graficznych | 381 |
| 12.1. Czemu grafika? | 382 |
| 12.2. Model graficzny | 383 |
| 12.3. Pierwszy przykład | 384 |
| 12.4. Biblioteka GUI | 387 |
| 12.5. Współrzędne | 388 |
| 12.6. Figury geometryczne | 389 |
| 12.7. Używanie klas figur geometrycznych | 389 |
| 12.7.1. Nagłówki graficzne i funkcja main | 390 |
| 12.7.2. Prawie puste okno | 390 |
| 12.7.3. Klasa Axis | 392 |
| 12.7.4. Rysowanie wykresu funkcji | 394 |
| 12.7.5. Wielokąty | 394 |
| 12.7.6. Prostokąty | 395 |
| 12.7.7. Wypełnianie kolorem | 397 |
| 12.7.8. Tekst | 398 |
| 12.7.9. Obrazy | 399 |
| 12.7.10. Jeszcze więcej grafik | 400 |

| | |
|---|-----|
| 12.8. Uruchamianie programu | 401 |
| 12.8.1. Pliki źródłowe | 402 |
| Rozdział 13. Klasy graficzne | 407 |
| 13.1. Przegląd klas graficznych | 408 |
| 13.2. Klasy Point i Line | 410 |
| 13.3. Klasa Lines | 412 |
| 13.4. Klasa Color | 415 |
| 13.5. Typ Line_style | 416 |
| 13.6. Typ Open_polyline | 419 |
| 13.7. Typ Closed_polyline | 420 |
| 13.8. Typ Polygon | 421 |
| 13.9. Typ Rectangle | 423 |
| 13.10. Wykorzystywanie obiektów bez nazw | 427 |
| 13.11. Typ Text | 429 |
| 13.12. Typ Circle | 431 |
| 13.13. Typ Ellipse | 432 |
| 13.14. Typ Marked_polyline | 434 |
| 13.15. Typ Marks | 435 |
| 13.16. Typ Mark | 436 |
| 13.17. Typ Image | 437 |
| Rozdział 14. Projektowanie klas graficznych | 445 |
| 14.1. Zasady projektowania | 446 |
| 14.1.1. Typy | 446 |
| 14.1.2. Operacje | 447 |
| 14.1.3. Nazewnictwo | 448 |
| 14.1.4. Zmienność | 450 |
| 14.2. Klasa Shape | 450 |

| | |
|--|-----|
| 14.2.1. Klasa abstrakcyjna | 452 |
| 14.2.2. Kontrola dostępu | 453 |
| 14.2.3. Rysowanie figur | 456 |
| 14.2.4. Kopiowanie i zmienność | 458 |
| 14.3. Klasy bazowe i pochodne | 460 |
| 14.3.1. Układ obiektu | 461 |
| 14.3.2. Tworzenie podklas i definiowanie funkcji wirtualnych | 463 |
| 14.3.3. Przesłanianie | 463 |
| 14.3.4. Dostęp | 465 |
| 14.3.5. Czyste funkcje wirtualne | 466 |
| 14.4. Zalety programowania obiektowego | 467 |
| Rozdział 15. Graficzne przedstawienie funkcji i danych | 473 |
| 15.1. Wprowadzenie | 474 |
| 15.2. Rysowanie wykresów prostych funkcji | 474 |
| 15.3. Typ Function | 478 |
| 15.3.1. Argumenty domyślne | 479 |
| 15.3.2. Więcej przykładów | 480 |
| 15.3.3. Wyrażenia lambda | 481 |
| 15.4. Typ Axis | 482 |
| 15.5. Wartość przybliżona funkcji wykładniczej | 484 |
| 15.6. Przedstawianie danych na wykresach | 488 |
| 15.6.1. Odczyt danych z pliku | 490 |
| 15.6.2. Układ ogólny | 491 |
| 15.6.3. Skalowanie danych | 492 |
| 15.6.4. Budowanie wykresu | 493 |
| Rozdział 16. Graficzne interfejsy użytkownika | 499 |

| | |
|--|-----|
| 16.1. Różne rodzaje interfejsów użytkownika | 500 |
| 16.2. Przycisk Next | 501 |
| 16.3. Proste okno | 502 |
| 16.3.1. Funkcje zwrotne | 503 |
| 16.3.2. Pętla oczekująca | 506 |
| 16.3.3. Wyrażenie lambda jako wywołanie zwrotne | 507 |
| 16.4. Typ Button i inne pochodne typu Widget | 507 |
| 16.4.1. Widżety | 508 |
| 16.4.2. Przyciski | 509 |
| 16.4.3. Widżety In_box i Out_box | 510 |
| 16.4.4. Menu | 510 |
| 16.5. Przykład | 511 |
| 16.6. Inwersja kontroli | 514 |
| 16.7. Dodawanie menu | 515 |
| 16.8. Debugowanie kodu GUI | 519 |
| CZĘŚĆ III. DANE I ALGORYTMY | 525 |
| Rozdział 17. Wektory i pamięć wolna | 527 |
| 17.1. Wprowadzenie | 528 |
| 17.2. Podstawowe wiadomości na temat typu vector | 529 |
| 17.3. Pamięć, adresy i wskaźniki | 531 |
| 17.3.1. Operator sizeof | 533 |
| 17.4. Pamięć wolna a wskaźniki | 534 |
| 17.4.1. Alokacja obiektów w pamięci wolnej | 535 |
| 17.4.2. Dostęp poprzez wskaźniki | 536 |
| 17.4.3. Zakresy | 537 |
| 17.4.4. Inicjalizacja | 538 |
| 17.4.5. Wskaźnik zerowy | 539 |

| | |
|--|-----|
| 17.4.6. Dealokacja pamięci wolnej | 540 |
| 17.5. Destruktory | 542 |
| 17.5.1. Generowanie destruktorów | 544 |
| 17.5.2. Destruktory a pamięć wolna | 544 |
| 17.6. Dostęp do elementów | 546 |
| 17.7. Wskaźniki na obiekty klas | 547 |
| 17.8. Babranie się w typach - void* i rzutowanie | 548 |
| 17.9. Wskaźniki i referencje | 550 |
| 17.9.1. Wskaźniki i referencje jako parametry | 551 |
| 17.9.2. Wskaźniki, referencje i dziedziczenie | 552 |
| 17.9.3. Przykład - listy | 553 |
| 17.9.4. Operacje na listach | 554 |
| 17.9.5. Zastosowania list | 556 |
| 17.10. Wskaźnik this | 557 |
| 17.10.1. Więcej przykładów użycia typu Link | 559 |
| Rozdział 18. Wektory i tablice | 565 |
| 18.1. Wprowadzenie | 566 |
| 18.2. Inicjalizacja | 566 |
| 18.3. Kopiowanie | 568 |
| 18.3.1. Konstruktory kopiujące | 569 |
| 18.3.2. Przypisywanie z kopiowaniem | 571 |
| 18.3.3. Terminologia związana z kopiowaniem | 573 |
| 18.3.4. Przenoszenie | 574 |
| 18.4. Podstawowe operacje | 576 |
| 18.4.1. Konstruktory jawne | 578 |
| 18.4.2. Debugowanie konstruktorów i destruktorów | 579 |
| 18.5. Uzyskiwanie dostępu do elementów wektora | 581 |

- 18.5.1. Problem stałych wektorów 582
- 18.6. Tablice 583
 - 18.6.1. Wskaźniki na elementy tablicy 584
 - 18.6.2. Wskaźniki i tablice 586
 - 18.6.3. Inicjalizowanie tablic 588
 - 18.6.4. Problemy ze wskaźnikami 589
- 18.7. Przykłady - palindrom 592
 - 18.7.1. Wykorzystanie łańcuchów 592
 - 18.7.2. Wykorzystanie tablic 593
 - 18.7.3. Wykorzystanie wskaźników 594
- Rozdział 19. Wektory, szablony i wyjątki 599
 - 19.1. Analiza problemów 600
 - 19.2. Zmienianie rozmiaru 602
 - 19.2.1. Reprezentacja 603
 - 19.2.2. Rezerwacja pamięci i pojemność kontenera 604
 - 19.2.3. Zmienianie rozmiaru 605
 - 19.2.4. Funkcja `push_back()` 605
 - 19.2.5. Przypisywanie 606
 - 19.2.6. Podsumowanie dotychczasowej pracy nad typem `vector` 608
 - 19.3. Szablony 608
 - 19.3.1. Typy jako parametry szablonów 609
 - 19.3.2. Programowanie ogólne 611
 - 19.3.3. Koncepcje 613
 - 19.3.4. Kontenery a dziedziczenie 615
 - 19.3.5. Liczby całkowite jako parametry szablonów 616
 - 19.3.6. Dedukcja argumentów szablonu 618
 - 19.3.7. Uogólnianie wektora 618

| | |
|--|-----|
| 19.4. Sprawdzanie zakresu i wyjątki | 621 |
| 19.4.1. Dygresja - uwagi projektowe | 622 |
| 19.4.2. Wyznanie na temat makr | 624 |
| 19.5. Zasoby i wyjątki | 625 |
| 19.5.1. Potencjalne problemy z zarządzaniem zasobami | 626 |
| 19.5.2. Zajmowanie zasobów jest inicjalizacją | 628 |
| 19.5.3. Gwarancje | 628 |
| 19.5.4. Obiekt unique_ptr | 630 |
| 19.5.5. Zwrot przez przeniesienie | 631 |
| 19.5.6. Technika RAII dla wektora | 631 |
| Rozdział 20. Kontenery i iteratory | 637 |
| 20.1. Przechowywanie i przetwarzanie danych | 638 |
| 20.1.1. Praca na danych | 638 |
| 20.1.2. Uogólnianie kodu | 639 |
| 20.2. Ideały twórcy biblioteki STL | 642 |
| 20.3. Sekwencje i iteratory | 645 |
| 20.3.1. Powrót do przykładu | 647 |
| 20.4. Listy powiązane | 649 |
| 20.4.1. Operacje list | 650 |
| 20.4.2. Iteracja | 651 |
| 20.5. Jeszcze raz uogólnianie wektora | 653 |
| 20.5.1. Przeglądanie kontenera | 655 |
| 20.5.2. Słowo kluczowe auto | 656 |
| 20.6. Przykład - prosty edytor tekstu | 657 |
| 20.6.1. Wiersze | 659 |
| 20.6.2. Iteracja | 660 |
| 20.7. Typy vector, list oraz string | 663 |

- 20.7.1. Funkcje insert() i erase() 664
- 20.8. Dostosowanie wektora do biblioteki STL 666
- 20.9. Dostosowywanie wbudowanych tablic do STL 668
- 20.10. Przegląd kontenerów 670
 - 20.10.1. Kategorie iteratorów 672
- Rozdział 21. Algorytmy i słowniki 677
 - 21.1. Algorytmy biblioteki standardowej 678
 - 21.2. Najprostszy algorytm - find() 679
 - 21.2.1. Kilka przykładów z programowania ogólnego 681
 - 21.3. Ogólny algorytm wyszukiwania - find_if() 682
 - 21.4. Obiekty funkcyjne 683
 - 21.4.1. Abstrakcyjne spojrzenie na obiekty funkcyjne 684
 - 21.4.2. Predykaty składowych klas 686
 - 21.4.3. Wyrażenia lambda 687
 - 21.5. Algorytmy numeryczne 688
 - 21.5.1. Akumulacja 688
 - 21.5.2. Uogólnianie funkcji accumulate() 689
 - 21.5.3. Iloczyn skalarny 691
 - 21.5.4. Uogólnianie funkcji inner_product() 692
 - 21.6. Kontenery asocjacyjne 693
 - 21.6.1. Słowniki 693
 - 21.6.2. Opis ogólny kontenera map 695
 - 21.6.3. Jeszcze jeden przykład zastosowania słownika 699
 - 21.6.4. Kontener unordered_map 701
 - 21.6.5. Zbiory 703
 - 21.7. Kopiowanie 704
 - 21.7.1. Funkcja copy() 705

21.7.2. Iteratory strumieni 705

21.7.3. Utrzymywanie porządku przy użyciu kontenera set 708

21.7.4. Funkcja copy_if() 708

21.8. Sortowanie i wyszukiwanie 709

21.9. Algorytmy kontenerowe 711

CZĘŚĆ IV. POSZERZANIE HORYZONTÓW 717

Rozdział 22. Ideały i historia 719

22.1. Historia, ideały i profesjonalizm 720

22.1.1. Cele i filozofie języków programowania 720

22.1.2. Ideały programistyczne 722

22.1.3. Style i paradygmaty 728

22.2. Krótka historia języków programowania 731

22.2.1. Pierwsze języki 732

22.2.2. Korzenie nowoczesnych języków programowania 733

22.2.3. Rodzina Algol 738

22.2.4. Simula 745

22.2.5. C 747

22.2.6. C++ 750

22.2.7. Dziś 752

22.2.8. Źródła informacji 753

Rozdział 23. Przetwarzanie tekstu 757

23.1. Tekst 758

23.2. Łańcuchy 758

23.3. Strumienie wejścia i wyjścia 762

23.4. Słowniki 762

23.4.1. Szczegóły implementacyjne 767

23.5. Problem 769

- 23.6. Wyrażenia regularne 771
 - 23.6.1. Surowe literały łańcuchowe 773
- 23.7. Wyszukiwanie przy użyciu wyrażeń regularnych 774
- 23.8. Składnia wyrażeń regularnych 776
 - 23.8.1. Znaki i znaki specjalne 776
 - 23.8.2. Rodzaje znaków 777
 - 23.8.3. Powtórzenia 778
 - 23.8.4. Grupowanie 779
 - 23.8.5. Alternatywa 779
 - 23.8.6. Zbiory i przedziały znaków 780
 - 23.8.7. Błędy w wyrażeniach regularnych 781
- 23.9. Dopasowywanie przy użyciu wyrażeń regularnych 783
- 23.10. Źródła 787
- Rozdział 24. Działania na liczbach 791
 - 24.1. Wprowadzenie 792
 - 24.2. Rozmiar, precyzja i przekroczenie zakresu 792
 - 24.2.1. Ograniczenia typów liczbowych 795
 - 24.3. Tablice 796
 - 24.4. Tablice wielowymiarowe w stylu języka C 797
 - 24.5. Biblioteka Matrix 798
 - 24.5.1. Wymiary i dostęp 799
 - 24.5.2. Macierze jednowymiarowe 801
 - 24.5.3. Macierze dwuwymiarowe 804
 - 24.5.4. Wejście i wyjście macierzy 806
 - 24.5.5. Macierze trójwymiarowe 807
 - 24.6. Przykład - rozwiązywanie równań liniowych 808
 - 24.6.1. Klasyczna eliminacja Gaussa 809

24.6.2. Wybór elementu centralnego 810

24.6.3. Testowanie 811

24.7. Liczby losowe 812

24.8. Standardowe funkcje matematyczne 815

24.9. Liczby zespolone 816

24.10. Źródła 818

Rozdział 25. Programowanie systemów wbudowanych 823

25.1. Systemy wbudowane 824

25.2. Podstawy 827

25.2.1. Przewidywalność 829

25.2.2. Ideały 829

25.2.3. Życie z awarią 830

25.3. Zarządzanie pamięcią 832

25.3.1. Problemy z pamięcią wolną 833

25.3.2. Alternatywy dla ogólnej pamięci wolnej 836

25.3.3. Przykład zastosowania puli 837

25.3.4. Przykład użycia stosu 838

25.4. Adresy, wskaźniki i tablice 839

25.4.1. Niekontrolowane konwersje 839

25.4.2. Problem - źle działające interfejsy 840

25.4.3. Rozwiązanie - klasa interfejsu 843

25.4.4. Dziedziczenie a kontenery 846

25.5. Bity, bajty i słowa 848

25.5.1. Bity i operacje na bitach 849

25.5.2. Klasa bitset 853

25.5.3. Liczby ze znakiem i bez znaku 854

25.5.4. Manipulowanie bitami 858

25.5.5. Pola bitowe 860

25.5.6. Przykład - proste szyfrowanie 861

25.6. Standardy pisania kodu 865

25.6.1. Jaki powinien być standard kodowania 866

25.6.2. Przykładowe zasady 868

25.6.3. Prawdziwe standardy kodowania 873

Rozdział 26. Testowanie 879

26.1. Czego chcemy 880

26.1.1. Zastrzeżenie 881

26.2. Dowody 881

26.3. Testowanie 881

26.3.1. Testowanie regresyjne 882

26.3.2. Testowanie jednostkowe 883

26.3.3. Algorytmy i niealgorytmy 889

26.3.4. Testy systemowe 896

26.3.5. Znajdowanie założeń, które się nie potwierdzają 897

26.4. Projektowanie pod kątem testowania 898

26.5. Debugowanie 899

26.6. Wydajność 899

26.6.1. Kontrolowanie czasu 901

26.7. Źródła 903

Rozdział 27. Język C 907

27.1. C i C++ to rodzeństwo 908

27.1.1. Zgodność języków C i C++ 909

27.1.2. Co jest w języku C++, czego nie ma w C 911

27.1.3. Biblioteka standardowa języka C 912

27.2. Funkcje 913

- 27.2.1. Brak możliwości przeciążania nazw funkcji 913
- 27.2.2. Sprawdzanie typów argumentów funkcji 914
- 27.2.3. Definicje funkcji 915
- 27.2.4. Wywoływanie C z poziomu C++ i C++ z poziomu C 917
- 27.2.5. Wskaźniki na funkcje 919
- 27.3. Mniej ważne różnice między językami 920
 - 27.3.1. Przestrzeń znaczników struktur 920
 - 27.3.2. Słowa kluczowe 921
 - 27.3.3. Definicje 922
 - 27.3.4. Rzutowanie w stylu języka C 923
 - 27.3.5. Konwersja typu void* 924
 - 27.3.6. Typ enum 925
 - 27.3.7. Przestrzenie nazw 925
- 27.4. Pamięć wolna 926
- 27.5. Łańcuchy w stylu języka C 927
 - 27.5.1. Łańcuchy w stylu języka C i const 930
 - 27.5.2. Operacje na bajtach 930
 - 27.5.3. Przykład - funkcja strcpy() 931
 - 27.5.4. Kwestia stylu 931
- 27.6. Wejście i wyjście - nagłówek stdio 932
 - 27.6.1. Wyjście 932
 - 27.6.2. Wejście 933
 - 27.6.3. Pliki 935
- 27.7. Stałe i makra 935
- 27.8. Makra 936
 - 27.8.1. Makra podobne do funkcji 937
 - 27.8.2. Makra składniowe 938

| | |
|---|------|
| 27.8.3. Kompilacja warunkowa | 939 |
| 27.9. Przykład - kontenery intruzyjne | 940 |
| DODATKI | 949 |
| Dodatek A. Zestawienie własności języka | 951 |
| A.1. Opis ogólny | 952 |
| A.2. Literały | 954 |
| A.3. Identyfikatory | 957 |
| A.4. Zakres, pamięć oraz czas trwania | 958 |
| A.5. Wyrażenia | 961 |
| A.6. Instrukcje | 970 |
| A.7. Deklaracje | 972 |
| A.8. Typy wbudowane | 973 |
| A.9. Funkcje | 976 |
| A.10. Typy zdefiniowane przez użytkownika | 980 |
| A.11. Wyliczenia | 980 |
| A.12. Klasy | 981 |
| A.13. Szablony | 992 |
| A.14. Wyjątki | 995 |
| A.15. Przestrzenie nazw | 997 |
| A.16. Aliasy | 997 |
| A.17. Dyrektywy preprocesora | 998 |
| Dodatek B. Biblioteka standardowa | 1001 |
| B.1. Przegląd | 1002 |
| B.2. Obsługa błędów | 1005 |
| B.3. Iteratory | 1007 |
| B.4. Kontenery | 1011 |
| B.5. Algorytmy | 1018 |

B.6. Biblioteka STL 1026

B.7. Strumienie wejścia i wyjścia 1032

B.8. Przetwarzanie łańcuchów 1037

B.9. Obliczenia 1041

B.10. Czas 1045

B.11. Funkcje biblioteki standardowej C 1046

B.12. Inne biblioteki 1054

Dodatek C. Podstawy środowiska Visual Studio 1055

C.1. Uruchamianie programu 1056

C.2. Instalowanie środowiska Visual Studio 1056

C.3. Tworzenie i uruchamianie programu 1056

C.4. Później 1058

Dodatek D. Instalowanie biblioteki FLTK 1059

D.1. Wprowadzenie 1060

D.2. Pobieranie biblioteki FLTK z internetu 1060

D.3. Instalowanie biblioteki FLTK 1060

D.4. Korzystanie z biblioteki FLTK w Visual Studio 1061

D.5. Sprawdzanie, czy wszystko działa 1062

Dodatek E. Implementacja GUI 1063

E.1. Implementacja wywołań zwrotnych 1064

E.2. Implementacja klasy Widget 1065

E.3. Implementacja klasy Window 1066

E.4. Klasa Vector_ref 1067

E.5. Przykład - widgety 1068

Słowniczek 1071

Bibliografia 1077

Zdjęcia 1081