

Spis treści

| | |
|---|------------|
| 17 Biblioteczna klasa std::string | 609 |
| 17.1 Rozwiązanie przechowywania tekstów musiało się znaleźć | 609 |
| 17.2 Klasa std::string to przecież nasz stary znajomy | 611 |
| 17.3 Definiowanie obiektów klasy string | 612 |
| 17.4 Użycie operatorów =, +, += w pracy ze stringami | 617 |
| 17.5 Pojemność, rozmiar i długość stringu | 618 |
| 17.5.1 Bliźniacze funkcje size() i length() | 618 |
| 17.5.2 Funkcja składowa empty | 619 |
| 17.5.3 Funkcja składowa max_size | 619 |
| 17.5.4 Funkcja składowa capacity | 619 |
| 17.5.5 Funkcje składowe reserve i shrink_to_fit | 621 |
| 17.5.6 resize - zmiana długości stringu „na siłę” | 622 |
| 17.5.7 Funkcja składowa clear | 624 |
| 17.6 Użycie operatora [] oraz funkcji at | 624 |
| 17.6.1 Działanie operatora [] | 625 |
| 17.6.2 Działanie funkcji składowej at | 626 |
| 17.6.3 Przebieganie po wszystkich literach stringu zakresowym for | 629 |
| 17.7 Funkcje składowe front i back | 629 |
| 17.8 Jak umieścić w tekście liczbę? | 630 |
| 17.9 Jak wczytać liczbę ze stringu? | 632 |
| 17.10 Praca z fragmentem stringu, czyli z substringiem | 635 |
| 17.11 Funkcja składowa substr | 636 |
| 17.12 Szukanie zadanego substringu w obiekcie klasy String | |
| - funkcje find | 637 |
| 17.13 Szukanie rozpoczynane od końca stringu | 640 |
| 17.14 Szukanie w stringu jednego ze znaków z zadanego zestawu | 641 |
| 17.15 Usuwanie znaków ze stringu - erase i pop_back | 643 |
| 17.16 Wstawianie znaków do istniejącego stringu - funkcje insert | 644 |
| 17.17 Zamiana części znaków na inne znaki - replace | 646 |
| 17.18 Zagłądanie do wnętrza obiektu klasy string funkcją data | 649 |
| 17.19 Zawartość obiektu klasy string a C-string | 650 |
| 17.20 W porządku alfabetycznym, czyli porównywanie stringów | 653 |
| 17.20.1 Porównywanie stringów za pomocą funkcji compare | 654 |
| 17.20.2 Porównywanie stringów przy użyciu operatorów ==, !=, <, <=, >= | 658 |
| 17.21 Zamiana treści stringu na małe lub wielkie litery | 659 |
| 17.22 Kopiowanie treści obiektu klasy String do tablicy znakowej | |
| - funkcja copy | 662 |
| 17.23 Wzajemna zamiana treści dwóch obiektów klasy string | |
| - funkcja swap | 662 |

| | |
|---|------------|
| 17.24 Wczytywanie z klawiatury stringu o nieznanym wcześniej długości | |
| - getline | 663 |
| 17.24.1 Pułapka, czyli jak getline może Cię zaskoczyć | 666 |
| 17.25 Iteratory stringu | 670 |
| 17.25.1 Iterator do obiektu stałego | 674 |
| 17.25.2 Funkcje składowe klasy String pracujące z iteratorami | 675 |
| 17.26 Klasa String korzysta z techniki przenoszenia | 680 |
| 17.27 Bryk, czyli „pamięć zewnętrzna” programisty | 681 |
| 17.28 Ćwiczenia | 689 |
| 18 Deklaracje przyjaźni | 696 |
| 18.1 Przyjaciele w życiu i w C++ | 696 |
| 18.2 Przykład: dwie klasy deklarują przyjaźń z tą samą funkcją | 698 |
| 18.3 W przyjaźni trzeba pamiętać o kilku sprawach | 700 |
| 18.4 Obdarzenie przyjaźnią funkcji składowej innej klasy | 703 |
| 18.5 Klasy zaprzyjaźnione | 705 |
| 18.6 Konwencja umieszczania deklaracji przyjaźni w klasie | 707 |
| 18.7 Kilka ostrzegających słów na zakończenie | 707 |
| 18.8 Ćwiczenia | 708 |
| 19 Obsługa sytuacji wyjątkowych | 710 |
| 19.1 Jak dać znać, że coś się nie udało? | 710 |
| 19.2 Pierwszy prosty przykład | 712 |
| 19.3 Kolejność bloków catch ma znaczenie | 714 |
| 19.4 Który blok catch nadaje się do złapania lecącego wyjątku? | 715 |
| 19.5 Bloki try mogą być zagnieżdżane | 718 |
| 19.6 Obsługa wyjątków w praktycznym programie | 721 |
| 19.7 Specyfikator noexcept i operator noexcept | 731 |
| 19.8 Ćwiczenia | 734 |
| 20 Klasa-składnik oraz klasa lokalna | 737 |
| 20.1 Klasa-składnik, czyli gdy w klasie jest zagnieżdżona definicja innej klasy | 737 |
| 20.2 Prawdziwy przykład zagnieżdżenia definicji klasy | 744 |
| 20.3 Lokalna definicja klasy | 755 |
| 20.4 Lokalne nazwy typów | 758 |
| 20.5 Ćwiczenia | 759 |
| 21 Konstruktory i destruktory | 761 |
| 21.1 Konstruktor | 761 |
| 21.1.1 Przykład programu zawierającego klasę z konstruktorami | 762 |
| 21.2 Specyfikator (przydomek) explicit | 773 |
| 21.3 Kiedy i jak wywoływany jest konstruktor | 774 |
| 21.3.1 Konstruowanie obiektów lokalnych | 774 |
| 21.3.2 Konstruowanie obiektów globalnych | 775 |
| 21.3.3 Konstrukcja obiektów tworzonych operatorem new | 775 |
| 21.3.4 Jawne wywołanie konstruktora | 776 |

| | |
|--|------------|
| 21.3.5 Dalsze sytuacje, gdy pracuje konstruktor | 779 |
| 21.4 Destruktor | 779 |
| 21.4.1 Jawne wywołanie destruktora (ogromnie rzadka sytuacja) | 781 |
| 21.5 Nie rzucajcie wyjątków z destruktorów | 781 |
| 21.6 Konstruktor domniemany | 783 |
| 21.7 Funkcje składowe z przypiskami = default i = delete | 784 |
| 21.8 Konstruktorowa lista inicjalizacyjna składników klasy | 786 |
| 21.8.1 Dla wtajemniczonych: wyjątki rzucone z konstruktorowej listy inicjalizacyjnej | 793 |
| 21.9 Konstruktor delegujący | 797 |
| 21.10 Pomocnicza klasa std.initializerjst - lista inicjalizatorów | 804 |
| 21.10.1 Zastosowania niekonstruktorowe | 804 |
| 21.10.2 Konfuzja: lista inicjalizatorów a lista inicjalizacyjna | 813 |
| 21.10.3 Konstruktor z argumentem będącym klamrową listą inicjalizatorów | 814 |
| 21.11 Konstrukcja obiektu, którego składnikiem jest obiekt innej klasy | 819 |
| 21.12 Konstruktory niepubliczne? | 826 |
| 21.13 Konstruktory constexpr mogą wytwarzać obiekty constexpr | 828 |
| 21.14 Ćwiczenia | 838 |
| 22 Konstruktory: kopiujący i przenoszący | 841 |
| 22.1 Konstruktor kopiujący (albo inicjalizator kopiujący) | 841 |
| 22.2 Przykład klasy z konstruktorem kopiującym | 842 |
| 22.3 Kompilatorowi wolno pominąć niepotrzebne kopiowanie | 847 |
| 22.4 Dlaczego przez referencję? | 849 |
| 22.5 Konstruktor kopiujący gwarantujący nietykliwość | 850 |
| 22.6 Współodpowiedzialność | 851 |
| 22.7 Konstruktor kopiujący generowany automatycznie | 851 |
| 22.8 Kiedy powinniśmy sami zdefiniować konstruktor kopiujący? | 852 |
| 22.9 Referencja do rwartości daje zezwolenie na recykling | 859 |
| 22.10 Funkcja std::move, która nie przenosi, a tylko rzutuje | 862 |
| 22.11 Odebrana rwartość staje się w ciele funkcji lwartością | 864 |
| 22.12 Konstruktor przenoszący (inicjalizator przenoszący) | 866 |
| 22.12.1 Konstruktor przenoszący generowany przez kompilator | 871 |
| 22.12.2 Inne konstruktory generowane automatycznie | 871 |
| 22.12.3 Zwrot obiektu lokalnego przez wartość? Nie używamy przenoszenia! | 872 |
| 22.13 Tak zwana „semantyka przenoszenia” | 873 |
| 22.14 Nowe pojęcia dla ambitnych: glwartość, xwartość i prwartość | 873 |
| 22.15 dedtype - operator rozpoznawania typu bardzo wyszukanych wyrażeń | 876 |
| 22.16 Ćwiczenia | 881 |
| 23 Tablice obiektów | 883 |
| 23.1 Definiowanie tablic obiektów i praca z nimi | 883 |
| 23.2 Tablica obiektów definiowana operatorem new | 884 |
| 23.3 Inicjalizacja tablic obiektów | 886 |

| | |
|---|------------|
| 23.3.1 Inicjalizacja tablicy, której obiekty są agregatami | 886 |
| 23.3.2 Inicjalizacja tablic, których elementy nie są agregatami | 889 |
| 23.4 Wektory obiektów | 893 |
| 23.4.1 Wektor, którego elementami są obiekty klasy będącej agregatem | 895 |
| 23.4.2 Wektor, którego elementami są obiekty klasy niebędącej agregatem | 897 |
| 23.5 Ćwiczenia | 898 |
| 24 Wskaźnik do składników klasy | 899 |
| 24.1 Wskaźniki zwykłe - repetytorium | 899 |
| 24.2 Wskaźnik do pokazywania na składnik-daną | 900 |
| 24.2.1 Przykład zastosowania wskaźników do składników klasy | 904 |
| 24.3 Wskaźnik do funkcji składowej | 911 |
| 24.3.1 Przykład zastosowania wskaźników do funkcji składowych | 913 |
| 24.4 Tablica wskaźników do danych składowych klasy | 920 |
| 24.5 Tablica wskaźników do funkcji składowych klasy | 921 |
| 24.5.1 Przykład tablicy/wektora wskaźników do funkcji składowych | 922 |
| 24.6 Wskaźniki do składników statycznych są zwykłe | 925 |
| 24.7 Ćwiczenia | 926 |
| 25 Konwersje definiowane przez użytkownika | 928 |
| 25.1 Sformułowanie problemu | 928 |
| 25.2 Konstruktory konwertujące | 930 |
| 25.2.1 Kiedy jawnie, kiedy niejawnie | 931 |
| 25.2.2 Przykład konwersji konstruktorem | 936 |
| 25.3 Funkcja konwertująca - operator konwersji | 938 |
| 25.3.1 Na co funkcja konwertująca zamieniać nie może | 944 |
| 25.4 Który wariant konwersji wybrać? | 945 |
| 25.5 Sytuacje, w których zachodzi konwersja | 947 |
| 25.6 Zapis jawnego wywołania konwersji typów | 948 |
| 25.6.1 Advocatus zapisu przypominającego: „wywołanie funkcji” | 948 |
| 25.6.2 Advocatus zapisu: „rzutowanie” | 949 |
| 25.7 Nie całkiem pasujące argumenty, czyli konwersje kompilatora przy dopasowaniu | 949 |
| 25.8 Kilka rad dotyczących konwersji | 954 |
| 25.9 Ćwiczenia | 955 |
| 26 Przeładowanie operatorów | 957 |
| 26.1 Co to znaczy przeładować operator? | 957 |
| 26.2 Przeładowanie operatorów - definicja i trochę teorii | 959 |
| 26.3 Moje zabawki | 963 |
| 26.4 Funkcja operatorowa jako funkcja składowa | 964 |
| 26.5 Funkcja operatorowa nie musi być przyjacielem klasy | 967 |
| 26.6 Operatory predefiniowane | 967 |
| 26.7 Ile operandów ma mieć ten operator? | 968 |
| 26.8 Operatory jednooperandowe | 968 |
| 26.9 Operatory dwuoperandowe | 971 |

| | | |
|---|--|-------------|
| 26.9.1 | Przykład na przeładowanie operatora dwuoperandowego | 971 |
| 26.9.2 | Przemienność | 973 |
| 26.9.3 | Choć operatory inne, to nazwę mają tę samą | 974 |
| 26.10 | Przykład zupełnie niematematyczny | 974 |
| 26.11 | Operatory postinkrementacji i postdekrementacji - koniec z niesprawiedliwością | 984 |
| 26.12 | Praktyczne rady dotyczące przeładowania | 986 |
| 26.13 | Pojedynek: operator jako funkcja składowa czy globalna? | 988 |
| 26.14 | Zasłona spada, czyli tajemnica operatora << | 989 |
| 26.15 | Stałe dosłowne definiowane przez użytkownika | 995 |
| 26.15.1 | Przykład: stałe dosłowne użytkownika odbierane jako gotowane | 999 |
| 26.15.2 | Przykład: stałe dosłowne użytkownika odbierane na surowo | 1008 |
| 26.16 | Ćwiczenia | 1011 |
| 27 Przeładowanie: =, [], (), -> | | 1015 |
| 27.1 | Cztery operatory, które muszą być niestatycznymi funkcjami składowymi | 1015 |
| 27.2 | Operator przypisania = (wersja kopiująca) | 1015 |
| 27.2.1 | Przykład na przeładowanie (kopiującego) operatora przypisania | 1017 |
| 27.2.2 | Przypisanie „kaskadowe” | 1024 |
| 27.2.3 | Po co i jak zabezpieczamy się przed przypisaniem a = a | 1026 |
| 27.2.4 | Jak opowiedzieć potocznie o konieczności istnienia operatora przypisania? | 1027 |
| 27.2.5 | Kiedy kopiujący operator przypisania nie jest generowany automatycznie | 1029 |
| 27.3 | Przenoszący operator przypisania = | 1029 |
| 27.4 | Specjalne funkcje składowe i nierealna prosta zasada | 1038 |
| 27.5 | Operator [] | 1039 |
| 27.6 | Operator () | 1043 |
| 27.7 | Operator -> | 1049 |
| 27.7.1 | „Sprytny wskaźnik” wykorzystuje przeładowanie właśnie tego operatora | 1051 |
| 27.8 | Ćwiczenia | 1058 |

oprac. BPK