

0 Proszę nie czytać tego!

1 Startujemy!

- 1.1 Pierwszy program
- 1.2 Drugi program
- 1.3 Ćwiczenia

2 Instrukcje sterujące

- 2.1 Prawda . Fałsz, czyli o warunkach
 - 2.1.1 Wyrażenie logiczne
 - 2.1.2 Zmienne logiczne bool jako warunek
 - 2.1.3 Stare dobre sposoby z dawnego C++
- 2.2 Instrukcja warunkowa if
- 2.3 Pętla while
- 2.4 Pętla do...while...
- 2.5 Pętla for
- 2.6 Instrukcja switch
- 2.7 Co wybrać: switch czy if...else?
- 2.8 Instrukcja break
- 2.9 Instrukcja goto
- 2.10 Instrukcja continue
- 2.11 Klamry w instrukcjach sterujących
- 2.12 Ćwiczenia

3 Typy

- 3.1 Deklaracje typów
- 3.2 Systematyka typów z języka C++
- 3.3 Typy fundamentalne
 - 3.3.1 Definiowanie obiektów "w biegu".
- 3.4 Stałe dosłowne
 - 3.4.1 Stałe będące liczbami całkowitymi
 - 3.4.2 Stałe reprezentujące liczby zmiennoprzecinkowe
 - 3.4.3 Stałe znakowe
 - 3.4.4 Stałe tekstowe, napisy, albo po prostu stringi
- 3.5 Typy złożone
- 3.6 Typ void
- 3.7 Zakres ważności nazwy obiektu, a czas życia obiektu
 - 3.7.1 Zakres: lokalny
 - 3.7.2 Zakres: blok funkcji
 - 3.7.3 Zakres: obszar pliku
 - 3.7.4 Zakres: obszar klasy
 - 3.7.5 Zakres określony przez przestrzeń nazw
- 3.8 Zastąpienie nazw

- 3.9 Specyfikator (przydomek) const
- 3.9.1 Pojedynyk: const con tra #define
- 3.10 Obiekty register
- 3.11 Specyfikator volatile
- 3.12 Instrukcja typedef
- 3.13 Typy wyliczeniowe enum
- 3.14 Ćwiczenia

4 Operatory

- 4.1 Operatory arytmetyczne
- 4.1.1 Operator %, czyli reszta z dzielenia (modulo)
- 4.1.2 Jednoargumentowe operatory + i -
- 4.1.3 Operatory inkrementacji i dekrementacji
- 4.1.4 Operator przypisania =
- 4.2 Operatory logiczne
- 4.2.1 Operatory relacji
- 4.2.2 Operatory sumy logicznej || i iloczynu logicznego &&
- 4.2.3 Wykrzyknik ! . czyli operator negacji
- 4.3 Operatory bitowe
- 4.3.1 Przesunięcie w lewo <<
- 4.3.2 Przesunięcie w prawo >>
- 4.3.3 Bitowe operatory sumy, iloczynu, negacji, różnicy symetrycznej
- 4.4 Różnica między operatorami logicznymi, a operatorami bitowymi
- 4.5 Pozostałe operatory przypisania
- 4.6 Wyrażenie warunkowe
- 4.7 Operator sizeof
- 4.8 Operatory rzutowania
- 4.8.1 Rzutowanie według tradycyjnych (niezalecanych) sposobów
- 4.8.2 Rzutowanie za pomocą nowych operatorów rzutowania
- 4.8.3 Operator static_cast
- 4.8.4 Operator const_cast
- 4.8.5 Operator dynamic_cast
- 4.8.6 Operator reinterpret_cast
- 4.9 Operator: przecinek
- 4.10 Priorytety operatorów
- 4.11 Łączność operatorów
- 4.12 Ćwiczenia

5 Funkcje

- 5.1 Funkcja często wywołuje inną funkcję
- 5.2 Zwracanie rezultatu przez funkcję
- 5.3 Stos
- 5.4 Przesyłanie argumentów do funkcji przez wartość
- 5.5 Przesyłanie argumentów przez referencję

- 5.6 Kiedy deklaracja funkcji nie jest konieczna?
- 5.7 Argumenty domniemane
- 5.7.1 Ciekawostki na temat argumentów domniemanych
- 5.8 Nienazwany argument
- 5.9 Funkcje inline (w linii)
- 5.10 Przypomnienie o zakresie ważności nazw deklarowanych wewnątrz funkcji
- 5.11 Wybór zakresu ważności nazwy i czasu życia obiektu
- 5.11.1 Obiekty globalne
- 5.11.2 Obiekty automatyczne
- 5.11.3 Obiekty lokalne statyczne
- 5.12 Funkcje w programie składającym się z kilku plików
- 5.12.1 Nazwy statyczne globalne
- 5.13 Funkcje rekurencyjne
- 5.14 Funkcje biblioteczne
- 5.15 Ćwiczenia

6 Preprocesor

- 6.1 Na pomoc rodakom
- 6.2 Dyrektywa #define
- 6.3 Dyrektywa #undef
- 6.4 Makrodefinicje
- 6.5 Sklejacz nazw, czyli operator ##
- 6.6 Zamiana parametru aktualnego makrodefinicji na string
- 6.7 Dyrektywy kompilacji warunkowej
- 6.8 Dyrektywa #error
- 6.9 Dyrektywa #line
- 6.10 Wstawianie treści innych plików w tekst kompilowanego właśnie pliku
- 6.11 Dyrektywa pusta
- 6.12 Dyrektywy zależne od implementacji
- 6.13 Nazwy predefiniowane
- 6.14 Ćwiczenia

7 Tablice

- 7.1 Elementy tablicy
- 7.2 Inicjalizacja tablic
- 7.3 Przekazywanie tablicy do funkcji
- 7.4 Przykład z tablicą elementów typu enum
- 7.5 Tablice znakowe
- 7.6 Tablice wielowymiarowe
- 7.6.1 Typ wyrażeń związanych z tablicą wielowymiarową
- 7.6.2 Przesyłanie tablic wielowymiarowych do funkcji
- 7.7 Ćwiczenia

8 Wskaźniki

- 8.1 Wskaźniki mogą bardzo ułatwić życie
- 8.2 Definiowanie wskaźników
- 8.3 Praca ze wskaźnikiem
- 8.4 L.wartość
- 8.5 Operator rzutowania reinterpret_cast, a wskaźniki
- 8.6 Wskaźniki typu void
- 8.7 Cztery domeny zastosowania wskaźników
- 8.8 Zastosowanie wskaźników wobec tablic
- 8.8.1 Ćwiczenia z mechaniki ruchu wskaźnika
- 8.8.2 Użycie wskaźnika w pracy z tablicą
- 8.8.3 Arytmetyka wskaźników
- 8.8.4 Porównywanie wskaźników
- 8.8.5 Wskaźnik można porównać z adresem 0
- 8.9 Zastosowanie wskaźników w argumentach funkcji
- 8.9.1 Jeszcze raz o przesyłaniu tablic do funkcji
- 8.9.2 Odbieranie tablicy jako wskaźnika
- 8.9.3 Argument formalny będący wskaźnikiem do obiektu const
- 8.10 Zastosowanie wskaźników przy dostępie do konkretnych komórek pamięci
- 8.11 Rezerwacja obszarów pamięci
- 8.11.1 Operatory new i delete albo Oratorium Stworzenie Świata
- 8.11.2 Dynamiczna alokacja tablicy
- 8.11.3 Tablice wielowymiarowe tworzone operatorem new
- 8.11.4 Umiejscawiający operator new
- 8.11.5 "Przychodzimy, odchodzimy . cichuteńko, na..."
- 8.11.6 Zapas pamięci to nie jest studnia bez dna
- 8.11.7 Funkcja set_new_handler
- 8.11.8 Pojedynek: new con tra malloc
- 8.12 Stałe wskaźniki
- 8.13 Stałe wskaźniki, a wskaźniki do stałych
- 8.14 Strzał na ośle . Wskaźnik zawsze pokazuje na coś
- 8.15 Sposoby ustawiania wskaźników
- 8.16 Parada kłanców, czyli o rzutowaniu const_cast
- 8.17 Tablice wskaźników
- 8.18 Wariacje na temat C-stringów
- 8.19 Wskaźniki do funkcji
- 8.19.1 Ćwiczenia z definiowania wskaźników do funkcji
- 8.19.2 Wskaźnik do funkcji jako argument innej funkcji
- 8.19.3 Tablica wskaźników do funkcji
- 8.20 Argumenty z linii wywołania programu
- 8.21 Ćwiczenia

9 Przetładowanie nazwy funkcji

- 9.1 Co to znaczy: przetładowanie
- 9.2 Bliższe szczegóły przetładowania
- 9.3 Czy przetładowanie nazw funkcji jest techniką obiektowo orientowaną?
- 9.4 Linkowanie z modułami z innych języków

- 9.5 Przeładowanie, a zakres ważności deklaracji funkcji
- 9.6 Rozważania o identyczności lub odmienności typów argumentów
- 9.6.1 Przeładowanie, a typedef i enum
- 9.6.2 Tablica, a wskaźnik
- 9.6.3 Pewne szczegóły o tablicach wielowymiarowych
- 9.6.4 Przeładowanie, a referencja
- 9.6.5 Identyfikacja typów: T, const T, vol a tile T
- 9.6.6 Przeładowanie . a typy: T*, vol a tile T*, const T*
- 9.6.7 Przeładowanie . a typy: T&, vol a tile T&, const T&
- 9.7 Adres funkcji przeładowanej
- 9.7.1 Zwrot rezultatu będącego adresem funkcji przeładowanej
- 9.8 Kulisy dopasowywania argumentów do funkcji przeładowanych
- 9.9 Etapy dopasowania
- 9.9.1 Etap 1. Dopasowanie dokładne
- 9.9.2 Etap 1a. Dopasowanie dokładne, ale z tzw. trywialną konwersją
- 9.9.3 Etap 2. Dopasowanie z awansem (z promocją)
- 9.9.4 Etap 3. Próba dopasowania za pomocą konwersji standardowych
- 9.9.5 Etap 4. Próba dopasowania z użyciem konwersji zdefiniowanych przez użytkownika.
- 9.9.6 Etap 5. Próba dopasowania do funkcji z wielokropkiem
- 9.9.7 Wskaźników nie dopasowuje się inaczej niż dosłownie
- 9.10 Dopasowywanie wywołań z kilkoma argumentami
- 9.11 Ćwiczenia

10 Klasy

- 10.1 Typy definiowane przez użytkownika
- 10.2 Składniki klasy
- 10.3 Składnik będący obiektem
- 10.4 Enkapsulacja
- 10.5 Ukrywanie informacji
- 10.6 Klasa, a obiekt
- 10.7 Funkcje składowe
- 10.7.1 Posługiwanie się funkcjami składowymi
- 10.7.2 Definiowanie funkcji składowych
- 10.8 Jak to właściwie jest? (this)
- 10.9 Odwołanie się do publicznych danych składowych
- 10.10 Zastępowanie nazw
- 10.10.1 Nie sięgaj z klasy do obiektów globalnych
- 10.11 Przeładowanie i zastąpienie równocześnie
- 10.12 Nowa klasa? Osobny plik!
- 10.13 Przesyłanie do funkcji argumentów będących obiektami
- 10.13.1 Przesyłanie obiektu przez wartość
- 10.13.2 Przesyłanie przez referencję
- 10.14 Konstruktor . pierwsza wzmianka
- 10.15 Destruktor . pierwsza wzmianka
- 10.16 Składnik statyczny

- 10.16.1 Deklaracja składnika statycznego połączona z inicjalizacją
- 10.17 Statyczna funkcja składowa
- 10.18 Do czego może nam się przydać składnik statyczny w klasie?
- 10.19 Funkcje składowe typu const oraz volatile
- 10.19.1 Przeładowanie, a funkcje składowe const i volatile
- 10.20 Specyfikator mutable
- 10.21 Ćwiczenia

11 Biblioteczna klasa `std::string` do operacji z tekstami

- 11.1 Przykład programu z użyciem klasy `string`
- 11.2 Definiowanie obiektów klasy `string`
- 11.3 Użycie operatorów `=`, `+`, `+=`, w pracy ze `stringami`
- 11.3.1 Jak umieścić w tekście liczbę?
- 11.4 Pojemność, rozmiar i długość `stringu`
- 11.4.1 Funkcje `size()` i `length()`
- 11.4.2 Funkcja składowa `empty`
- 11.4.3 Funkcja składowa `max_size`
- 11.4.4 Funkcja składowa `capacity`
- 11.4.5 Funkcja składowa `reserve`
- 11.4.6 `resize` . zmiana długości `stringu` "na siłę"
- 11.4.7 Funkcja składowa `clear`
- 11.5 Użycie operatora `[]` oraz funkcji `at`
- 11.5.1 Działanie operatora `[]`
- 11.5.2 Działanie funkcji składowej `at`
- 11.6 Praca z fragmentem `stringu`, czyli z `sub.stringiem`
- 11.7 Funkcja składowa `substr`
- 11.8 Szukanie zadanego `substringu` w obiekcie klasy `string` . funkcja `find` i jej pokrewne
- 11.9 Szukanie rozpoczynane od końca `stringu`
- 11.10 Szukanie w `stringu` jednego ze znaków z zadanego zestawu
- 11.11 Usuwanie znaków ze `stringu` . funkcje `erase`
- 11.12 Wstawianie znaków do już istniejącego `stringu` . funkcje `insert`
- 11.13 Zamiana części znaków na inne znaki . `replace`
- 11.14 Zamiana zawartości obiektu klasy `string` na `C-string`
- 11.15 Zglądanie do wnętrza obiektu klasy `string` funkcją `data`
- 11.16 W porządku alfabetycznym . czyli porównywanie `stringów`
- 11.16.1 Porównywanie `stringów` funkcjami `compare`
- 11.16.2 Porównywanie `stringów` przy użyciu operatorów `==`, `!=`, `<`, `>`, `<=`, `>=`.
- 11.17 Zamiana treści `stringu` na małe (lub wielkie) litery
- 11.18 Kopiowanie treści obiektu klasy `string` do wybranej tablicy znakowej . funkcja `copy`
- 11.19 Wzajemna zamiana treści dwóch obiektów klasy `string` . funkcja `swap`
- 11.20 Przypisanie do obiektu klasy `string`, funkcja `assign`
- 11.21 Dopisywanie do końca `stringu` za pomocą funkcji `append`
- 11.22 Wczytywanie z klawiatury długiego `stringu` o nieznanym wcześniej długości . `getline`

- 11.22.1 Pułapka . czyli jak getline może Cię zaskoczyć
- 11.23 Iteratory stringu
- 11.23.1 Iterator do obiektu stałego
- 11.23.2 Funkcje składowe klasy string pracujące z iteratorami
- 11.24 Bryk . czyli "pamięć zewnętrzna" programisty
- 11.25 Ćwiczenia

12 Deklaracje przyjaźni

- 12.1 Klasy zaprzyjaźnione
- 12.2 Słowo o zakresie

13 Struktury, Unie, Pola bitowe

- 13.1 Struktura
- 13.2 Unia
- 13.2.1 Inicjalizacja unii
- 13.2.2 Unia anonimowa
- 13.3 Pola bitowe
- 13.4 Unia i pola bitowe . upraszczają rozpakowanie słów
- 13.5 Ćwiczenia

14 Klasa zagnieżdżona lub lokalna

- 14.1 Zagnieżdżona definicja klasy
- 14.2 Lokalna definicja klasy
- 14.3 Lokalne nazwy typów
- 14.4 Ćwiczenia

15 Konstruktory i Destruktory

- 15.1 Konstruktor
- 15.1.1 Przykład programu zawierającego klasę z konstruktorami
- 15.2 Specyfikator (przydomek) explicit
- 15.3 Kiedy i jak wywoływany jest konstruktor
- 15.3.1 Konstruowanie obiektów lokalnych
- 15.3.2 Konstruowanie obiektów globalnych
- 15.3.3 Konstrukcja obiektów tworzonych operatorem new
- 15.3.4 Jawne wywołanie konstruktora
- 15.3.5 Dalsze sytuacje, gdy pracuje konstruktor
- 15.4 Destruktor
- 15.5 Konstruktor domniemany
- 15.6 Lista inicjalizacyjna konstruktora
- 15.7 Konstrukcja obiektu, którego składnikiem jest obiekt innej klasy
- 15.8 Konstruktory nie-publiczne
- 15.9 Konstruktor kopiujący (albo inicjalizator kopiujący)

- 15.9.1 Przykład klasy z konstruktorem kopiującym
- 15.9.2 Dlaczego przez referencję?
- 15.9.3 Jak dostać piątkę z C++?
- 15.9.4 Konstruktor kopiujący gwarantujący nietykalność
- 15.9.5 Współodpowiedzialność
- 15.9.6 Konstruktor kopiujący generowany automatycznie
- 15.9.7 Kiedy konstruktor kopiujący jest niezbędny?
- 15.10 Ćwiczenia

16 Tablice obiektów

- 16.1 Tablica obiektów definiowana operatorem new
- 16.2 Inicjalizacja tablic obiektów
- 16.2.1 Inicjalizacja tablic obiektów będących agregatami
- 16.2.2 Inicjalizacja tablic nie będących agregatami
- 16.2.3 Inicjalizacja tablic tworzonych w zapasie pamięci
- 16.3 Ćwiczenia

17 Wskaźnik do składników klasy

- 17.1 Wskaźniki zwykłe . repetytorium
- 17.2 Wskaźnik do pokazywania na składnik-daną
- 17.2.1 Przykład zastosowania wskaźników do składników klasy
- 17.3 Wskaźnik do funkcji składowej
- 17.3.1 Zastosowanie wskaźników do funkcji składowych
- 17.4 Tablica wskaźników do danych składowych klasy
- 17.5 Tablica wskaźników do funkcji składowych klasy
- 17.6 Wskaźniki do składników statycznych
- 17.7 Ćwiczenia

18 Konwersje

- 18.1 Sformułowanie problemu
- 18.2 Konstruktory konwertujące
- 18.2.1 Kiedy jawnie, kiedy niejawnie
- 18.2.2 Przykład konwersji konstruktorem
- 18.3 Funkcja konwertująca . operator konwersji
- 18.3.1 Na co konwertować nie można
- 18.4 Który wariant konwersji wybrać?
- 18.5 Sytuacje, w których zachodzi konwersja
- 18.6 Zapis jawnego wywołania konwersji typów
- 18.6.1 Advocatus zapisu przypominającego: "wywołanie funkcji"
- 18.6.2 Advocatus zapisu: "rzutowanie"
- 18.7 Niecałkiem pasujące argumenty, czyli konwersje przy dopasowaniu
- 18.8 Kilka rad dotyczących konwersji
- 18.9 Ćwiczenia

19 Przeładowanie operatorów

- 19.1 Przeładowanie operatorów . definicja i trochę teorii
- 19.2 Moje zabawki
- 19.3 Funkcja operatorowa jako funkcja składowa
- 19.4 Funkcja operatorowa nie musi być przyjacielem klasy
- 19.5 Operatory predefiniowane
- 19.6 Argumentowość operatorów
- 19.7 Operatory jednoargumentowe
- 19.8 Operatory dwuargumentowe
- 19.8.1 Przykład na przeładowanie operatora dwuargumentowego
- 19.8.2 Przemienność
- 19.8.3 Choć operatory inne, to nazwę mają tę samą
- 19.9 Przykład zupełnie nie matematyczny
- 19.10 Cztery operatory, które muszą być niestandardowymi funkcjami składowymi
- 19.11 Operator przypisania =
- 19.11.1 Przykład na przeładowanie operatora przypisania
- 19.11.2 Jak konieczność istnienia operatora przypisania . opowiedzieć potocznie?
- 19.11.3 Kiedy operator przypisania nie jest generowany automatycznie
- 19.12 Operator []
- 19.13 Operator ()
- 19.14 Operator .>
- 19.14.1 "Zręczny wskaźnik" . wykorzystuje przeładowanie właśnie tego operatora
- 19.15 Operatory new, new[]
- 19.15.1 Przykład przeładowania operatora new
- 19.15.2 Przykład przeładowania operatora new[]
- 19.16 Operatory delete, delete[]
- 19.16.1 Prosty przykład przeładowania delete
- 19.16.2 Prosty przykład przeładowania delete[]
- 19.17 Program przykładowy na zastosowanie operatorów new, delete
- 19.18 Przeładowanie globalnych operatorów new, new[], delete, delete[]
- 19.19 Operatory postinkrementacji i postdekrementacji, czyli koniec z niesprawiedliwością
- 19.20 Rady praktyczne dotyczące przeładowania
- 19.21 Pojedynek: Operator jako funkcja składowa, czy globalna
- 19.22 Zasłona spada, czyli tajemnica operatora <<
- 19.23 Rzut oka wstecz
- 19.24 Ćwiczenia

20 Dziedziczenie

- 20.1 Istota dziedziczenia
- 20.2 Dostęp do składników
- 20.2.1 Prywatne składniki klasy podstawowej
- 20.2.2 Nieprywatne składniki klasy podstawowej
- 20.2.3 Klasa pochodna też decyduje

- 20.2.4 Deklaracja dostępu using . czyli udostępnianie wybiórcze
- 20.3 Czego się nie dziedziczy
- 20.3.1 "Nie dziedziczenie" konstruktorów
- 20.3.2 "Nie dziedziczenie" operatora przypisania
- 20.3.3 "Nie dziedziczenie" destruktora
- 20.4 Drzewo genealogiczne
- 20.5 Dziedziczenie . doskonałe narzędzie programowania
- 20.6 Kolejność wywoływania konstruktorów
- 20.7 Przypisanie i inicjalizacja obiektów w warunkach dziedziczenia
- 20.7.1 Klasa pochodna nie definiuje swojego operatora przypisania
- 20.7.2 Klasa pochodna nie definiuje swojego konstruktora kopiującego
- 20.7.3 Inicjalizacja i przypisywanie według obiektu wzorcowego będącego const
- 20.7.4 Definiowanie konstruktora kopiującego i operatora przypisania dla klasy pochodnej
- 20.8 Dziedziczenie od kilku "rodziców" (czyli wielokrotne)
- 20.8.1 Konstruktor klasy pochodnej przy wielokrotnym dziedziczeniu
- 20.8.2 Ryzyko wieloznaczności przy dziedziczeniu
- 20.8.3 Bliższe pokrewieństwo usuwa wieloznaczność
- 20.8.4 Poszlaki
- 20.9 Pojedynek: Dziedziczenie klasy, contra zawieranie obiektów składowych
- 20.10 Konwersje standardowe przy dziedziczeniu
- 20.10.1 Pan orama korzyści
- 20.10.2 Czego robić się nie opłaca
- 20.10.3 Tuzin samochodów nie jest rodzajem tuzina pojazdów
- 20.10.4 Konwersje standardowe wskaźnika do składnika klasy
- 20.11 Wirtualne klasy podstawowe
- 20.11.1 Publiczne i prywatne dziedziczenie tej samej klasy wirtualnej
- 20.11.2 Uwagi o konstrukcji i inicjalizacji w przypadku klas wirtualnych
- 20.11.3 Dominacja klas wirtualnych
- 20.12 Ćwiczenia

21 Funkcje wirtualne

- 21.1 Polimorfizm
- 21.2 Typy rezultatów różnych realizacji funkcji wirtualnej
- 21.3 Dalsze szczegóły
- 21.4 Wczesne i późne wiązanie
- 21.5 Kiedy dla wywołań funkcji wirtualnych, mimo wszystko, zachodzi wczesne wiązanie?
- 21.6 Kulisy białej magii, czyli: jak to jest zrobione?
- 21.7 Funkcja wirtualna, a mimo to inline
- 21.8 Pojedynek . funkcje przeładowane contra funkcje wirtualne
- 21.9 Klasy abstrakcyjne
- 21.10 Destruktor? to najlepiej wirtualny!
- 21.11 Co prawda, konstruktor nie może być wirtualny, ale...
- 21.12 Rzutowanie dynamic_cast jest dla typów polimorficznych
- 21.13 Wszystko, co najważniejsze

- 21.14 Finis coronat opus
- 21.15 Ćwiczenia

22 Operacje Wejścia/Wyjścia

- 22.1 Biblioteka iostream
- 22.2 Strumień
- 22.3 Strumienie zdefiniowane standardowo
- 22.4 Operatory >> i <<
- 22.5 Domniemania w pracy strumieni zdefiniowanych standardowo
- 22.6 Uwaga na priorytet
- 22.7 Operatory << oraz >> definiowane przez użytkownika
- 22.7.1 Operatorów wstawiania i wyjmowania ze strumienia . nie dziedziczy się
- 22.7.2 Operatory wstawiania i wyjmowania nie mogą być wirtualne. Niestety.
- 22.8 Sterowanie formatem
- 22.9 Flagi stanu formatowania
- 22.9.1 Znaczenie poszczególnych flag sterowania formatem
- 22.10 Sposoby zmiany trybu (reguł) formatowania
- 22.10.1 Zmiana sposobu formatowania funkcjami setf, unsetf
- 22.10.2 Dodatkowe funkcje do zmiany parametrów formatowania
- 22.11 Manipulatory
- 22.11.1 Manipulatory bezargumentowe
- 22.11.2 Manipulatory parametryzowane
- 22.11.3 Definiowanie swoich manipulatorów
- 22.11.4 Manipulator jako funkcja
- 22.11.5 Definiowane manipulatora z parametrem
- 22.12 Nieformatowane operacje wejścia/wyjścia
- 22.13 Omówienie funkcji wyjmujących ze strumienia
- 22.13.1 Funkcje do pracy ze znakami i stringami
- 22.13.2 Wczytywanie binarne . funkcje read i readsome
- 22.13.3 Funkcja ignore
- 22.13.4 Pożyteczne funkcje pomocnicze
- 22.13.5 Funkcje wstawiające do strumienia
- 22.14 Strumienie płynące do lub od plików
- 22.14.1 Otwieranie i zamykanie strumienia
- 22.15 Błędy w trakcie pracy strumienia
- 22.15.1 Flagi stanu błędu strumienia
- 22.15.2 Funkcje do pracy na flagach błędu
- 22.15.3 Kilka udogodnień
- 22.15.4 Ustawianie i kasowanie flag błędu strumienia
- 22.15.5 Trzy plagi . czyli "gotowiec", jak radzić sobie z błędami
- 22.16 Przykład programu pracującego na plikach
- 22.17 Strumienie, a technika rzucania wyjątków
- 22.18 Wybór miejsca czytania lub pisania w pliku
- 22.18.1 Funkcje składowe informujące o pozycji wskaźników
- 22.18.2 Wybrane funkcje składowe do pozycjonowania wskaźników
- 22.19 Pozycjonowanie w przykładzie większego programu

- 22.20 Tie . harmonijna praca dwóch strumieni
- 22.21 Dlaczego tak nie lubimy biblioteki stdio?
- 22.22 Synchronizacja biblioteki iostream z biblioteką stdio
- 22.23 Strumień zapisujący do obiektu klasy string
- 22.23.1 Program przykładowy ilustrujący użycie klasy ostream
- 22.24 Strumień czytający z obiektu klasy string
- 22.24.1 Prosty przykład użycia strumienia istream
- 22.24.2 Wczytywanie argumentów wywoływania programu
- 22.25 Ożenek: strumień stringstream . czytający i zapisujący do stringu
- 22.25.1 Przykładowy program posługujący się klasą stringstream
- 22.26 Ćwiczenia

23 Projektowanie programów orientowanych obiektowo

- 23.1 Przegląd kilku technik programowania
- 23.1.1 Programowanie liniowe
- 23.1.2 Programowanie proceduralne (czyli "orientowane funkcyjnie")
- 23.1.3 Programowanie z ukrywaniem danych
- 23.1.4 Programowanie obiektowe . programowanie "bazujące" na obiektach
- 23.1.5 Programowanie Obiektowo Orientowane (OO)
- 23.2 O wyższości programowania obiektowo orientowanego nad Świątami Wielkiej Nocy
- 23.3 Obiektowo Orientowane: Projektowanie
- 23.4 Praktyczne wskazówki dotyczące projektowania programu techniką OO
- 23.4.1 Rekonesans . czyli rozpoznanie zagadnienia
- 23.4.2 Faza projektowania
- 23.4.3 Etap 1: Identyfikacja zachowań systemu
- 23.4.4 Etap 2: Identyfikacja obiektów (klas obiektów)
- 23.4.5 Etap 3: Usystematyzowanie klas obiektów
- 23.4.6 Etap 4: Określenie wzajemnych zależności klas
- 23.4.7 Etap 5: Składanie modelu. Określanie sekwencji działań obiektów i cykli życiowych
- 23.5 Faza implementacji
- 23.6 Przykład projektowania
- 23.7 Faza: Rozpoznanie naszego zagadnienia
- 23.8 Faza: Projektowanie
- 23.8.1 Etap 1 . Identyfikacja zachowań naszego systemu
- 23.8.2 Etap 2 . Identyfikacja klas obiektów, z którymi mamy do czynienia
- 23.8.3 Etap 3 . Usystematyzowanie klas obiektów z występujących w naszym systemie
- 23.8.4 Etap 4 . Określenie wzajemnych zależności klas
- 23.8.5 Etap 5 . Składamy model naszego systemu
- 23.9 Implementacja modelu naszego systemu
- 23.10 Symfonia C++, Coda
- 23.11 Postówie

A Dodatek: Systemy liczenia

- A.1 Dlaczego komputer nie liczy tak jak my?
- A.2 System szesnastkowy (heksadecymalny)
- A.3 Ćwiczenia