Podstawy Programowania

Wykład nr 1: Wprowadzenie

dr hab. inż. Dariusz Dereniowski

Katedra Algorytmów i Modelowania Systemów
Wydział ETI, Politechnika Gdańska

Podziękowania

Materiały do niniejszego wykładu zostały opracowane z wykorzystaniem fragmentów wykładów udostępnionych przez dr. hab. inż. Adriana Kosowskiego, dr. Krzysztofa Manuszewskiego i dr Olgi Choreń.

Zasady zaliczenia

Realizacja przedmiotu:

- wykład: 30 godzin (do połowy semestru)
- laboratorium zamknięte (LZ): 15 godzin (do połowy semestru)
- laboratorium otwarte (LO): praca indywidualna

Zaliczenie wykładu:

- w ramach zaliczenia wykładu odbędzie się kolokwium podstawowe (procentowy wynik punktowy oznaczmy przez W_1) oraz kolokwium poprawkowe (W_2)
- wynik z części wykładowej jest równy $W = W_1$ jeśli student nie pisał kolokwium poprawkowego oraz $W = W_2$ jeśli student pisał kolokwium poprawkowe
- wykład jest uznany za zaliczony jeśli $W \geq 50\%$

Zasady zaliczenia c.d.

Zaliczenie laboratorium zamkniętego:

- Laboratorium zamknięte jest uznane za zaliczone jeśli procentowy wynik, oznaczany przez LZ, wynosi co najmniej 50%.

Zaliczenie laboratorium otwartego:

- Laboratorium otwarte jest uznane za zaliczone jeśli procentowy wynik, oznaczany przez LO, wynosi co najmniej 50%.

Zaliczenie przedmiotu:

- Jeśli $W < 50\%$ lub $LO < 50\%$ lub $LZ < 50\%$, to ocena końcowa z przedmiotu wynosi 2.0.
- Jeśli $W \geq 50\%$, $LO \geq 50\%$ i $LZ \geq 50\%$, to obliczamy procentowy wynik z przedmiotu wg wzoru: $PP = 0.4 \cdot W + 0.3 \cdot LO + 0.3 \cdot LZ$ a następnie ocenę końcową z przedmiotu wyznaczamy następująco:
  - $PP \geq 90\% \Rightarrow$ ocena $= 5.0$
  - $80\% \leq PP < 90\% \Rightarrow$ ocena $= 4.5$
  - $70\% \leq PP < 80\% \Rightarrow$ ocena $= 4.0$
  - $60\% \leq PP < 70\% \Rightarrow$ ocena $= 3.5$
  - $50\% \leq PP < 60\% \Rightarrow$ ocena $= 3.0$
  - $PP < 50\% \Rightarrow$ ocena $= 2.0$

Cel przedmiotu i literatura

Cel przedmiotu:

- nauka podstaw programowania strukturalnego
- wprowadzenie do algorytmów i najprostszych struktur danych
- nauka języka C (z drobnymi 'zapożyczeniami' z języka C++)

Literatura:

- J. Grębosz: Symfonia C++. Programowanie w języku C++ orientowane obiektowo, wyd. Kallimach, Kraków
- B.W. Kernighan, D. M. Ritchie, Język ANSI C, Wydawnictwa Naukowo-Techniczne, Warszawa

Cykl wytwarzania oprogramowania

Na nasze potrzeby przytaczamy prosty model:

1. Projekt i analiza:
   - opis i sformułowanie problemu
   - opracowanie algorytmu
2. Implementacja.
   - programowanie
3. Testowanie i wdrożenie.
   - kompilacja
   - uruchamianie
   - testowanie

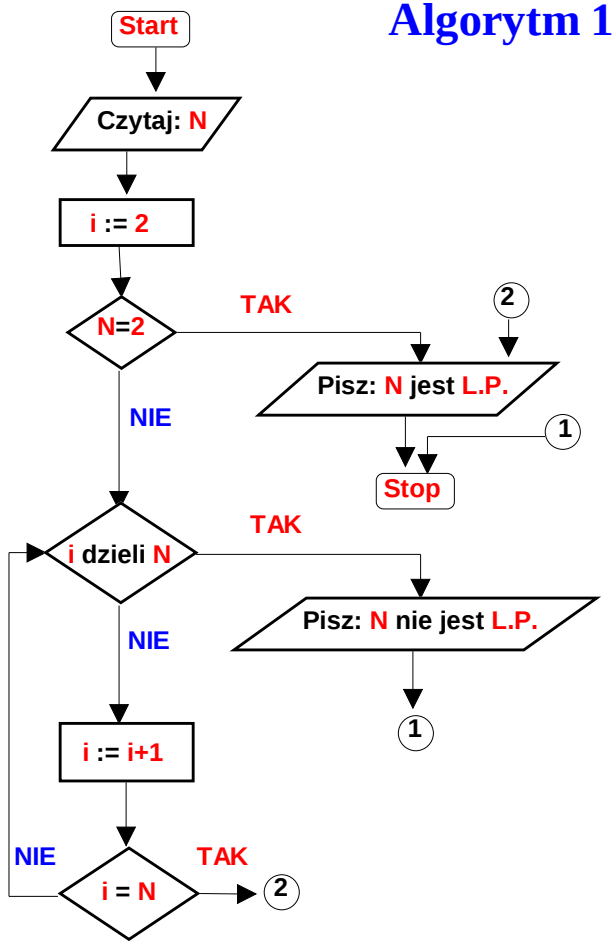
Algorytm — definicje

- Algorytm to skończony ciąg sekwencji/reguł, które aplikuje się do skończonej ilości danych, pozwalający rozwiązywać zbliżone do siebie klasy problemów.
- Algorytm to zespół reguł charakterystycznych dla pewnych obliczeń lub czynności informatycznych.

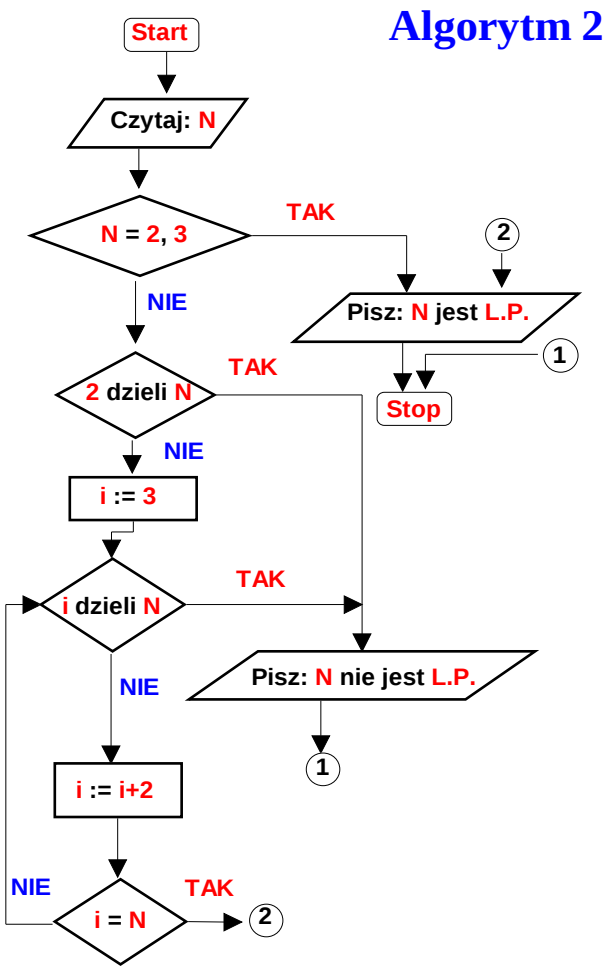
Algorytm:

- przyjmuje dane wejściowe (w ilości większej lub równej zero), pochodzące z dobrze zdefiniowanego zbioru
- produkuje wynik, niekoniecznie numeryczny,
- POWINIEN BYĆ precyzyjnie zdefiniowany – każdy krok algorytmu musi być jednoznacznie określony, deterministyczny
- POWINIEN BYĆ skończony – wynik algorytmu musi być 'kiedyś' dostarczony, algorytm ma własność stopu

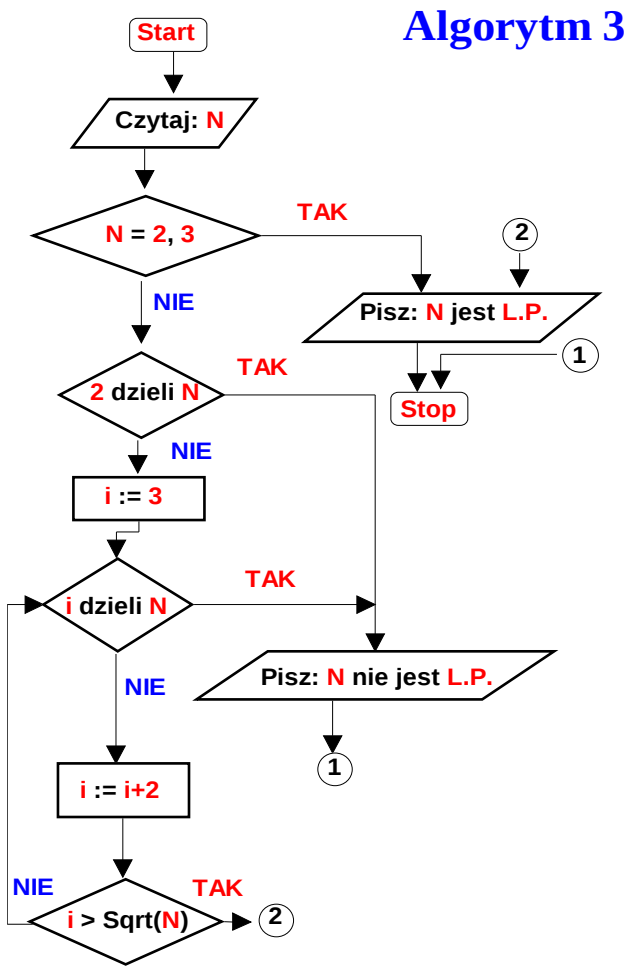
Różne sposoby rozwiązania problemu



Różne sposoby rozwiązania problemu



Różne sposoby rozwiązania problemu



Różne sposoby rozwiązania problemu

- Algorytm 1 sprawdza wszystkie liczby i począwszy od 2 is skończywszy na $N-1$
- Algorytm 2 sprawdza liczbę 2 oraz wszystkie liczby nieparzyste mniejsze niż N
- Algorytm 3 sprawdza liczbę 2 oraz wszystkie liczby nieparzyste mniejsze lub równe $\sqrt{N}$

N	Algorytm 1	Algorytm 2	Algorytm 3
10	8	5	2
100	98	50	5
1000	998	500	16

Uwaga: liczby iteracji podane w tabeli nie odzwierciedlają faktycznej liczby iteracji dla $N = 10, 100, 1000$, lecz stanowią poglądową ilustrację pesymistycznej liczby iteracji, która ma miejsce w sytuacji, gdy N jest liczbą pierwszą.

Zalecenia

- wybór algorytmu poprzedza kodowanie
- wybór algorytmu wpływa na: łatwość kodowania, szybkość działania i czytelność końcowego programu
- dbaj o uniwersalność programu (przenośność)
- dobry algorytm jest warunkiem konieczny ale nie dostateczny stworzenia dobrego programu

Język C

Cechy C:

- kompilacja do kodu maszynowego
- pre-procesor
- biblioteki standardowe (wejście wyjście, matematyczna, i inne)
- niewielka liczba słów kluczowych
- struktury, wskaźniki

Czego w C nie ma:

- wyjątków
- automatycznego odzyskiwania pamięci (tzw. garbage collector)
- elementów charakterystycznych dla programowania obiektowego

Język C

Rys historyczny

- 1972 – język C – Dennis Ritchie, Bell Labs (rozwinięty na potrzeby systemu Unix)
- 1978 – Brian Kernighan i Dennis Ritchie opracowują nieformalną specyfikację języka
- standaryzacja: 1989/1990 (ANSI/ISO), 1999 ("C99"), 2011 ("C11")
- rozwinięcia: C++, Objective C
- języki wzorujące się: Java, C#, JavaScript, Perl, PHP, ...

Główne zastosowania

- programowanie systemowe (Unix/Widnows), programowanie mikrokontrolerów, urządzeń osadzonych,...
- programowanie wydajnych aplikacji do przetwarzania danych, obliczeń numerycznych, aplikacji graficznych (C/C++)
- aplikacje ogólnego przeznaczenia

Podstawowe paradygmaty w języku C

Programowanie imperatywne: programista opisuje w jasny sposób ciąg instrukcji przetwarzania i sterujących (skoków, warunków, pętli, wywołań podprogramów) do wykonania:

- właściwy opis: silnia(n): wynik = 1, w pętli dla i od 1 do n wykonuj: wynik = i * wynik, zwróć wynik
- niewłaściwy opis: silnia(1) = 1, silnia(n) = silnia(n-1)*n;

Programowanie strukturalne:

- program podzielony jest na funkcje/procedury, rozwiązujące konkretne podproblemy
- podział na funkcje, opcjonalnie grupowane w moduły, opisuje logiczną strukturę programu

Pierwszy program

```
#include <iostream>

int main() {
  std::cout << "Hello world!";
  return 0;
}
```

```
#include <iostream>
using namespace std;

int main() {
  cout << "Hello world!";
  return 0;
}
```

- oba powyższe programy są równoważne, tzn. efekt ich wykonania jest identyczny
- dyrektywa #include poleca dołączenie do programu odpowiedniej biblioteki (dołączenie pliku nagłówkowego związanego ze strumieniami C++)
- 'using namespace std' wskazuje na przestrzeń nazw, w której znajduje się cout
- w niniejszym kursie będziemy korzystali z wejścia/wyjścia C++ (będzie to jedyny wyjątek)
- każdy program zawiera funkcję o nazwie main
- wykonanie programu zaczyna się od uruchomienia funkcji main
- instrukcje należące do funkcji ujęte są w nawiasy { }
- do wypisania tekstu na ekran używamy strumieni C++

Styl programowania

- programy mają być czytane przez ludzi
- stosuj komentarze wstępne
- stosuj komentarze wyjaśniające
- komentarz - to nie parafraza instrukcji
- stosuj odstępy do poprawienia czytelności
- używaj dobrych nazw mnemonicznych
- wystarczy jedna instrukcja w wierszu
- stosuj wcięcia do uwidocznienia struktury programu
- PODPROGRAMY!!