



Programowanie komputerów I

Podstawowe struktury danych

Willy Picard

Katedra Technologii Informacyjnych
Akademia Ekonomiczna w Poznaniu
<picard@kti.ae.poznan.pl>

Agenda

- ▶ Cel(e) wykładu
- ▶ Logika Boole'a
- ▶ Liczby
- ▶ Znaki
- ▶ Operatory
- ▶ Podsumowanie

Cel(e) wykładu



Przegląd wykładu

Podstawowe pojęcia

- ▶ 1: Wprowadzenie
- ▶ 2: Podstawowe struktury danych & instrukcje
- ▶ 3: Programowanie obiektowe I
- ▶ 4: Programowanie obiektowe II
- ▶ 5: Programowanie obiektowe III
- ▶ 6: Zaawansowane struktury danych
- ▶ 7: Wątki & Wyjątki

Cel dzisiejszego wykładu

- ▶ Wprowadzić logikę Boole'a
- ▶ Przedstawić podstawowe struktury danych

Logika Boole'a



To Be or Not to Be

- ▶ George Boole, 1815-1864
- ▶ Dwie wartości
 - ▶ 0 *fałsz (false)*
 - ▶ 1 *prawda (true)*
- ▶ Pojęcie *bitu*
- ▶ W Javie, typ `boolean`
- ▶ Logika Boole'a

Logika Boole'a

- ▶ Operatory logiczne
 - ▶ AND
 - ▶ OR
 - ▶ NOT
 - ▶ XOR

Operator AND

	TRUE	FALSE
TRUE	TRUE	FALSE
FALSE	FALSE	FALSE

Operator OR

	TRUE	FALSE
TRUE	TRUE	TRUE
FALSE	TRUE	FALSE

Operator NOT

	NOT
TRUE	FALSE
FALSE	TRUE

Operator XOR

	TRUE	FALSE
TRUE	FALSE	TRUE
FALSE	TRUE	FALSE

Podstawowe operatory logiczne

- ▶ Podstawowe operatory
 - ▶ AND
 - ▶ OR
 - ▶ NOT
- ▶ Inne operatory są połączeniem podstawowych
 - ▶ $\text{XOR}(A,B) =$
 $(\text{NOT}(A) \text{ AND } B)$
 OR
 $(A \text{ AND NOT}(B))$

W Javie

	Ewaluacja	Zoptymalizowane
AND	&	&&
OR		
XOR	^	
NOT	!	
EQUALS	==	
DIFFERENT	!=	

Ewaluacja vs zoptymalizowane

- ▶ Ewaluacja

- ▶ Każdy termin jest ewaluowany

```
false & true  
true  | false
```

- ▶ Zoptymalizowane

- ▶ Jeżeli pierwszy termin pozwala ustalić wynik, drugi termin nie jest brany pod uwagi

```
false && true  
true  || false
```



Liczby

Od Bitu do liczby

- ▶ Kodowanie liczb
 - ▶ Dodatnie liczby całkowite
 - ▶ Ujemne liczby całkowite
 - ▶ Liczby rzeczywiste
- ▶ Arytmetyka

Dziesiętny system liczbowy

- ▶ Co oznacza 152?
 - ▶ $1 \times 100 + 5 \times 10 + 2 \times 1$
 - ▶ $1 \times 10^2 + 5 \times 10^1 + 2 \times 10^0$
 - ▶ 1, 5, i 2 są cyframi (*digits*)
- ▶ Dlaczego dziesiętny system?
 - ▶ 10 palców

Siódemkowy system liczbowy

- ▶ Co byliby gdybyśmy mieli 7 placów?
 - ▶ Siódemkowy system liczbowy?
 - ▶ $152_{10} = 147 + 5$
 $= 3 \times 49 + 0 \times 7 + 5$
 $= 3 \times 7^2 + 0 \times 7^1 + 5 \times 7^0$
 $= 305_7$

Popularne systemy liczbowe

- ▶ System binarny
 - ▶ System dwójkowy
 - ▶ „bit” znaczy „Binary digit”
- ▶ System ósemkowy
 - ▶ Cyfry od 0 do 7
- ▶ System szesnastkowy
 - ▶ Cyfry: 0-9, A-F
 - ▶ $A_{16} = 10_{10}$, $B_{16} = 11_{10}$, ... $F_{16} = 15_{10}$

Dodatnie liczby całkowite

- ▶ Zbiór bitów

- ▶ 8 bitów = byte (bajt)
- ▶ 2 byte = word
- ▶ 4 byte = word, doubleword, longword
- ▶ 8 bytes = quadword, longword

- ▶ n bitów

- ▶ $0 < x < 2^n - 1$

- ▶ 1 bajt

- ▶ $0 = 00000000_2 < x < 2^8 - 1 = 255_{10} = 11111111_2$

Działania arytmetyczne

► $101_2 + 110_2 = ?$

► Złe podejście

► $101_2 = 4_{10} + 1_{10} = 5_{10}$

► $110_2 = 4_{10} + 2_{10} = 6_{10}$

► $5_{10} + 6_{10} = 11_{10}$

► $11_{10} = 8_{10} + 2_{10} + 1_{10} = 1011_2$

Działania arytmetyczne

- ▶ $101_2 + 110_2 = ?$
- ▶ Dobre podejście

$$\begin{array}{r} 1 \\ 101 \\ + 110 \\ \hline 1011 \end{array}$$

Liczby całkowite ze znakiem

- ▶ Trzy metody
 - ▶ Znak-moduł (*Sign-and-magnitude*)
 - ▶ Kod uzupełnień do jedności U1 (*One's complement*)
 - ▶ Kod uzupełnień do dwóch U2 (*Two's complement*)
- ▶ Najbardziej popularna
 - ▶ Kod uzupełnień do dwóch

Znak-Moduł

- ▶ 1 bit dla znaku
 - ▶ 0 = dodatnie
 - ▶ 1 = ujemne
- ▶ Inne bity dla modułu
 - ▶ $+5 = 0101$
 - ▶ $-5 = 1101$
- ▶ Skomplikowane działania arytmetyczne

Znak-Moduł

- ▶ Dwa zera
 - ▶ +0: 0000
 - ▶ - 0: 1000
- ▶ n cyfr
 - ▶ Liczby od $-2^{n-1}-1$ do $2^{n-1}-1$
- ▶ 1 bajt
 - ▶ $-127 < n < 127$

Kod uzupełnień do jedności

- ▶ Inwersja
 - ▶ $1 \rightarrow 0$
 - ▶ $0 \rightarrow 1$
- ▶ Liczba ujemna = inwersja liczby dodatniej
 - ▶ $5 = 0101$
 - ▶ $!5 = 1010$
 - ▶ $-5 = 1010$
 - ▶ $5 + (-5) = 1111 = -0$

Kod uzupełnień do jedności

- ▶ Pierwsza cyfra = znak
 - ▶ 0 dla dodatnich, 1 dla ujemnych
- ▶ Dwa zera
 - ▶ +0: 0000, - 0: 1111
- ▶ n cyfr
 - ▶ Liczby od $-2^{n-1}-1$ do $2^{n-1}-1$
- ▶ 1 bajt
 - ▶ $-127 < n < 127$

Kod uzupełnień do dwóch

► Ujemna liczba = inwersja + 1

► 5 = 0101

► !5 = 1010

► !5 + 1 = 1011

► - 5 = 1011

$$\begin{array}{r} 111 \\ 0101 \\ + 1011 \\ \hline \text{X}0000 \end{array}$$

Kod uzupełnień do dwóch

- ▶ Pierwsza cyfra = znak
 - ▶ 0 dla dodatnich
 - ▶ 1 dla ujemnych
- ▶ Jedno zero
 - ▶ +0: 0000
- ▶ n cyfr
 - ▶ Liczby od -2^{n-1} do $2^{n-1}-1$
- ▶ 1 bajt
 - ▶ $-128 < n < 127$

Liczby rzeczywiste

- ▶ Stałoprzecinkowe liczby

- ▶ $0.5 = 1/2 = 00000000.10000000$

- ▶ $1.25 = 1 \frac{1}{4} = 00000001.01000000$

- ▶ $7.375 = 7 \frac{3}{8} = 00000111.01100000$

- ▶ Zmiennoprzecinkowe liczby

- ▶ $znak \times (1 + \text{ułamek mantysy}) \times 2^{\text{wykładnik}}$

- ▶ $sign \times (1 + \text{fractional significand}) \times 2^{\text{exponent}}$

- ▶ Standaryzowany w IEEE 754

Liczby w Javie

Nazwa typu	Rozmiar/Format
<i>Liczby całkowite</i>	
byte	8-bitów, U2
short	16-bitów, U2
int	32-bitów, U2
long	64-bitów, U2
<i>Liczby rzeczywiste</i>	
float	32-bitów, IEEE 754
double	64-bitów, IEEE 754

Liczby w Javie

Typ	Minimalna wartość	Maksymalna wartość
<i>Liczby całkowite</i>		
byte	-128	127
short	-32768	32767
int	-2147483648	2147483647
long	-9223372036854775808	9223372036854775807
<i>Liczby rzeczywiste</i>		
float	-3.402823E+38	3.402823E+38
double	-1.797693134862E+308	1.797693134862E+308

Przykłady liczb w Javie

- ▶ int 178
- ▶ long 8864L
- ▶ double 37.266
- ▶ double 37.266D
- ▶ double 26.77e3
- ▶ float 87.363F

Znaki



Kodowanie znaków

- ▶ Zbiór znaków
 - ▶ *Character set*
- ▶ Kodowanie znaków
 - ▶ *Character encoding*
 - ▶ Znak \leftrightarrow liczba
- ▶ Przykład
 - ▶ Zbiór znaków: [A, B, C, D]
 - ▶ Kodowanie znaków
 - ▶ A \leftrightarrow 1 B \leftrightarrow 2 C \leftrightarrow 3 D \leftrightarrow 4
 - ▶ DAD \leftrightarrow 414

Popularne kodowanie znaków

- ▶ Standard ASCII
 - ▶ American Standard Code for Information Interchange
 - ▶ Alfabet rzymski na 7 bitów
 - ▶ Przykład: A = 64, B = 65, a = 97
- ▶ Standardy ISO-8859-x
 - ▶ 8 bitów: 1 bit dla dodatkowych znaków
- ▶ Windows Cp125x
 - ▶ Niestandaryzowane
 - ▶ 8 bitów

Popularne kodowanie znaków

- ▶ Unicode
 - ▶ Standard mający w zamierzeniu obejmować wszystkie pisma używane na świecie
 - ▶ Różne kodowania
 - ▶ UTF-32, UTF-16, UTF-8, UTF-7
- ▶ W Javie
 - ▶ Podstawowy typ `char`
 - ▶ Wszystkie znaki są znakami Unicode
 - ▶ Kodowanie na 2 bajtach

Przykłady znaków w Javie

- ▶ `'a'` `a`
- ▶ `'\u004A'` `J`
- ▶ `'\112'` `J`
- ▶ `'\\'` `\`
- ▶ `'\t'` znak tabulacji
- ▶ `'\n'` znak nowej linii
- ▶ `'\"'` `'`
- ▶ `'\''` `"`



Operatory

Unary, Binary, a Ternary

- ▶ 1 argument
 - ▶ *Unary*
 - ▶ zapis prefiksowy *operator op1*
 - ▶ zapis postfiksowy *op1 operator*
- ▶ 2 argumenty
 - ▶ *Binary*
 - ▶ zapis infiksowy *op1 operator op2*
- ▶ 3 argumenty

Operatory arytmetyczne

- ▶ 2 argumenty

- ▶ Dodawanie + op1 + op2

- ▶ Odejmowanie - op1 - op2

- ▶ Mnożenie * op1 * op2

- ▶ Dzielenie / op1 / op2

- ▶ Modulo % op1 % op2

- ▶ 1 argument (post- lub prefiksowe)

- ▶ Inkrementacja ++ ++op1

- ▶ Dekrementacja -- op1--

Operatory relacji

- ▶ 2 argumenty (!)

▶ Większe	>	<code>op1 > op2</code>
▶ Większe lub równe	<code>>=</code>	<code>op1 >= op2</code>
▶ Mniejsze	<	<code>op1 < op2</code>
▶ Mniejsze lub równe	<code><=</code>	<code>op1 <= op2</code>
▶ Równe	<code>==</code>	<code>op1 == op2</code>
▶ Różne	<code>!=</code>	<code>op1 != op2</code>

Operator bitowe

- ▶ 1 argument

- ▶ Inwersja $\sim op$ $\sim 5_{10} = \sim 101_2 = 010_2 = 2_{10}$

- ▶ 2 argumenty

- ▶ AND

- $$5_{10} \& 3_{10} = 101_2 \& 011_2 = 001_2 = 1_{10}$$

- ▶ OR

- $$5_{10} | 3_{10} = 101_2 | 011_2 = 111_2 = 7_{10}$$

- ▶ XOR

- $$5_{10} \wedge 3_{10} = 101_2 \wedge 011_2 = 110_2 = 6_{10}$$

Operatory przesunięcia

- ▶ 2 argumenty

- ▶ Na prawo

$$6_{10} >> 2_{10}$$

$$\text{op1} >> \text{op2}$$

$$0110_2 >> 2_{10} = 0001_2 = 1_{10}$$

- ▶ Na prawo bez znaku $\text{op1} >>> \text{op2}$

$$6_{10} >>> 2_{10}$$

$$0110_2 >>> 2_{10} = 0001_2 = 1_{10}$$

- ▶ Na lewo

$$6_{10} << 2_{10}$$

$$\text{op1} << \text{op2}$$

$$0110_2 << 2_{10} = 11000_2 = 24_{10}$$

Operatory przesunięcia a liczby ujemne

► Na prawo

$$-5_{10} \gg 2_{10} \quad 1011_2 \gg 2_{10} = 1110_2 = -2_{10}$$

► Na prawo bez znaku

$$-5_{10} \ggg 2_{10} \quad 1011_2 \ggg 2_{10} = 0010_2 = +2_{10}$$

► Na lewo

$$-5_{10} \ll 2_{10} \quad 1011_2 \ll 2_{10} = 101100_2 = -20_{10}$$

Operatory przypisania

- ▶ 2 argumenty

- ▶ $op1 = op2$

- ▶ $op1 += op2$

$op1 = op1 + op2$

- ▶ $op1 -= op2$

$op1 = op1 - op2$

- ▶ $op1 *= op2$

$op1 = op1 * op2$

- ▶ $op1 /= op2$

- ▶ $op1 \% = op2$

- ▶ $op1 \&= op2$, $op1 |= op2$, $op1 \wedge= op2$

- ▶ $op1 \ll = op2$, $op1 \gg = op2$, $op1 \ggg = op1$

Skrótowa forma operatora `if-else`

- ▶ 3 argumenty
 - ▶ `op1 ? op2 : op3`
 - ▶ Zwraca `op2` jeżeli `op1` jest *true*,
`op3` w przeciwnym wypadku

Podsumowanie



Podsumowanie

- ▶ Logika Boole'a
- ▶ Liczby
 - ▶ Ważność dwójkowego systemu liczbowego
 - ▶ Detale implementacji ukryte w Javie
- ▶ Znaki
 - ▶ od ASCII do Unicode
- ▶ Operatory
 - ▶ Tylko niektóre często używane

Przykład

```
package pl.poznan.ae.compProg;

import java.util.*;

public class Sorter {
    private List _words;

    public void sort(String[] words){
        _words = Arrays.asList(words);
        Collections.sort(_words);
    }
    public String getSortedWords(){
        String sortedString = "";
        for (int i = 0; i< _words.size(); i++){
            sortedString += _words.get(i);
        }
        return sortedString;
    }
    public static void main(String[] args){
        Sorter sorter = new Sorter();
        sorter.sort(args);
        System.out.println(sorter.getSortedWords());
    }
}
```

**Do zobaczenia
za tydzień**

