Computer Programming

Basic Data Types

Willy Picard
Department of Information Technology
The Poznan University of Economics
<*picard@kti.ae.poznan.pl*>

Agenda

- Lecture Goal(s)
- Boolean logic
- Numbers
- Characters
- Operators
- Conclusions

Lecture Goal(s)

Lectures Overview

Fundamental Concepts

- 1: Introduction
- 2: Basic data structures & Statements
- 3: Object-oriented programming I
- 4: Object-oriented programming II
- 5: Object-oriented programming III
- 6: Complex data structures
- 7: Threads & Exception handling

LECTURE GOALS

Today's Goal

- To introduce boolean logic
- To provide programming knowledge about basic data types

Boolean Logic

To Be or Not to Be

- George Boole, 1815-1864
- Based on two values
  - 0 *false*
  - 1 *true*
- Concept of *bit*
- In Java, `boolean` type
- Boolean logic

Boolean Logic

- Logical operators
  - AND
  - OR
  - NOT
  - XOR

AND Operator

	TRUE	FALSE
TRUE	TRUE	FALSE
FALSE	FALSE	FALSE

OR Operator

	TRUE	FALSE
TRUE	TRUE	TRUE
FALSE	TRUE	FALSE

NOT Operator

	NOT
TRUE	FALSE
FALSE	TRUE

XOR Operator

	TRUE	FALSE
TRUE	FALSE	TRUE
FALSE	TRUE	FALSE

Basic Logical Operators

- Basic operators
  - AND
  - OR
  - NOT
- Other operators are combinations of basic ones
  - XOR(A,B) =

    (NOT(A) AND B)
    OR
    (A AND NOT(B))

The Java Style

	Evaluation	Optimized
AND	&	&&
OR	|	||
XOR	^	
NOT	!	
EQUALS	==	
DIFFERENT	!=	

Evaluation vs Optimized

- Evaluation
  - Both terms are evaluated

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

- Optimized
  - If the first terms allows to compute the result, the second term is not evaluated

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

Numbers

From Bit to Numbers

- Number representation
  - From bit to positive integers
  - Negative integers
  - Decimal numbers
- Arithmetics

Base-10 Number System

- What means 152?
  - 1x100 + 5x10 + 2x1
  - $1x10^2 + 5x10^1 + 2x10^0$
  - 1, 5, and 2 are *digits*
- Why the base-10 number system?
  - 10 fingers

Base-7 Number System

- What would happen if we had 7 fingers?
  - Base-7 number system?
  - $152_{10} = 147 + 5$
    $= 3x49 + 0x7 + 5$
    $= 3x7^2 + 0x7^1 + 5x7^0$
    $= 305_7$

Popular Number Systems

- Binary system
  - Base-2 number system
  - A bit is a Binary digit
- Octal system
  - Base-8 number system
- Hexadecimal
  - Base-16 number system
  - Symbols: 0-9, A-F
  - $A_{16}=10_{10}$, $B_{16}=11_{10}$, ... $F_{16}=15_{10}$

Natural Numbers

- Sets of bits
  - 8 bits = a byte
  - 2 bytes = a word
  - 4 bytes = a word, a doubleword, a longword
  - 8 bytes = a quadword, a longword
- With n bits
  - $0 < x < 2^n-1$
- With a byte
  - $0 = 00000000^2 < x < 2^8-1 = 255_{10} = 11111111_2$

Binary Arithmetical Operations

- $101_2 + 110_2 = ?$
- Bad approach
  - $101_2 = 4_{10} + 1_{10} = 5_{10}$
  - $110_2 = 4_{10} + 2_{10} = 6_{10}$
  - $5_{10} + 6_{10} = 11_{10}$
  - $11_{10} = 8_{10} + 2_{10} + 1_{10} = 1011_2$

Binary Arithmetical Operations

- $101_2 + 110_2 = ?$
- Good approach

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

Signed Integers

- Three main techniques
  - Sign-and-magnitude
  - One's complement
  - Two's complement
- Most used
  - Two's complement

Sign-and-Magnitude

- 1 bit for sign
  - 0 = positive
  - 1 = negative
- Other bits for amplitude
  - +5 = 0101
  - -5 = 1101
- Difficult additions

Sign-and-Magnitude

- Two zeros
  - +0: 0000
  - - 0: 1000
- With n digits,
  - numbers from $-2^{n-1}-1$ to $2^{n-1}-1$
- For a byte,
  - -127<n<127

One's Complement

- Complement
  - 1 → 0
  - 0 → 1
- Negative number = complement of positive number
  - 5 = 0101
  - !5 = 1010
  - -5 = 1010
  - 5 + (-5) = 1111 = -0

One's Complement

- First digit = sign
  - 0 for positive integers
  - 1 for negative integers
- Two zeros
  - +0: 0000
  - - 0: 1111
- With n digits, numbers from $-2^{n-1}-1$ to $2^{n-1}-1$
  - For a byte, -127<n<127

Two's Complement

- Negative number = complement + 1
  - 5 = 0101
  - !5 = 1010
  - !5 + 1 = 1011
  - - 5 = 1011

```
  111
  0101
+ 1011
------
 X0000
```

Two's Complement

- First digit = sign
  - 0 for positive integers
  - 1 for negative integers
- One zero
  - +0: 0000
- With n digits
  - numbers from -2^{n-1} to $2^{n-1}-1$
- For a byte
  - $-128<n<127$

Real Numbers

- Fixed-point numbers
  - 0.5 = 1/2 = 00000000.10000000
  - 1.25 = 1 1/4 = 00000001.01000000
  - 7.375 = 7 3/8 = 00000111.01100000
- Floating-point numbers
  - *sign* × (1 + *fractional significand*) × $2^{exponent}$
  - Standardized in IEEE 754

Numbers in Java

Keyword	Description	Size/Format
Integers		
`byte`	Byte-length integer	8-bit two's complement
`short`	Short integer	16-bit two's complement
`int`	Integer	32-bit two's complement
`long`	Long integer	64-bit two's complement
Real numbers		
`float`	Single-precision floating point	32-bit IEEE 754
`double`	Double-precision floating point	64-bit IEEE 754

Number Ranges in Java

Keyword	Min. Value	Max. Value
Integers		
`byte`	-128	127
`short`	-32768	32767
`int`	-2147483648	2147483647
`long`	-9223372036854775808	9223372036854775807
Real numbers		
`float`	-3.402823E+38	3.402823E+38
`double`	-1.797693134862E+308	1.797693134862E+308

Examples of Numbers in Java

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

Characters

Character Encoding

- Character set
  - A group of characters
- Character encoding
  - A character ↔ a number
- Example
  - Character set: [A, B, C, D]
  - Character encoding:
    - A ↔ 1 B ↔ 2 C ↔ 3 D ↔ 4
    - DAD ↔ 414

Popular Character Encodings

- Standard ASCII
  - American Standard Code for Information Interchange
  - Roman alphabet on 7 bits
  - Example: A = 64, B = 65, a = 97
- Standardized ISO-8859-x
  - 8 bits: 1 bit for additional characters
- Windows Cp125x
  - Not standardized
  - 8 bits

Popular Character Encodings

- Unicode
  - Standard aiming at supporting all existing characters
  - Various encoding
    - UTF-32, UTF-16, UTF-8, UTF-7
- In Java
  - Primitive `char`
  - All characters are Unicode characters
  - Encoding on 2 bytes

Examples of Characters in Java

- 'a' a
- '\u004A' J
- '\112' J
- '\\' \
- '\t' tabulation
- '\n' new line
- '\'' '
- '\"' “

Operators

Unary, Binary, and Ternary

- Unary
  - one operand
  - prefix notation: `operator op1`
  - postfix notation: `op1 operator`
- Binary
  - two operands
  - infix notation: `op1 operator op2`
- Ternary
  - three operands

Arithmetic Operators

- Binary
  - Addition `+` `op1 + op2`
  - Subtraction `-` `op1 - op2`
  - Multiplication `*` `op1 * op2`
  - Division `/` `op1 / op2`
  - Modulo `%` `op1 % op2`
- Unary
  - Pre-addition `++` `++op1`
  - Post-addition `--` `op1--`

Relational Operators

- Binary (!)
  - Greater `>` `op1 > op2`
  - Greater or equal `>=` `op1 >= op2`
  - Less `<` `op1 < op2`
  - Less or equal `<=` `op1 <= op2`
  - Equal `==` `op1 == op2`
  - Not equal `!=` `op1 != op2`

Bitwise Operators

- Unary
  - Complement ~op $\sim 5_{10} = \sim 101_2 = 010_2 = 2_{10}$
- Binary
  - AND op1 & op2

    $5_{10} \& 3_{10} = 101_2 \& 011_2 = 001_2 = 1_{10}$
  - OR op1 | op2

    $5_{10} | 3_{10} = 101_2 | 011_2 = 111_2 = 7_{10}$
  - XOR op1 ^ op2

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

Shift Operators

- Binary
  - Right shift op1 >> op2

    $6_{10}>>2_{10}$ $0110_2>>2_{10} = 0001_2 = 1_{10}$
  - Right shift unsigned op1 >>>op2

    $6_{10}>>>2_{10}$ $0110_2>>>2_{10} = 0001_2 = 1_{10}$
  - Left shift op1 << op2

    $6_{10}<<2_{10}$ $0110_2<<2_{10} = 11000_2 = 24_{10}$

Shift Operators on Negative Integers

- Right shift

  $-5_{10} >> 2_{10}$ $1011_2 >> 2_{10} = 1110_2 = -2_{10}$

- Right shift unsigned

  $-5_{10} >>> 2_{10}$ $1011_2 >>> 2_{10} = 0010_2 = +2_{10}$

- Left shift

  $-5_{10} << 2_{10}$ $1011_2 << 2_{10} = 101100_2 = -20_{10}$

Assignment Operators

- Binary
  - 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 ^= op2
  - op1 <<=op2, op1 >>= op2, op1 >>>= op1

Short-cut `if-else` Operator

- Ternary
  - op1 ? op2 : op3
  - if op1 is *true*, then op2, else op3

Conclusions

Conclusions

- Boolean logic
- Numbers
  - importance of the base-2 number system
  - implementation details hidden in Java
- Characters
  - from ASCII to Unicode
- Operators
  - Only a few often used

Example

```
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());
 }
}
```

See you next week