Data Representation and Applets

CSC 1051 – Data Structures and Algorithms I
Dr. Mary-Angela Papalaskari
Department of Computing Sciences
Villanova University

Course website:
http://www.csc.villanova.edu/~map/1051/

Overview

• Binary representation
• Image representation
• Java Applets
• Data types revisited
• Type conversions

Data Representation

• Computers store all information digitally, using binary codes:
  – numbers
  – text
  – images
  – audio
  – video
  – program instructions

A byte is a group of eight bits

Computing devices store & use binary codes to represent data of all kinds
Binary codes

<table>
<thead>
<tr>
<th>1 bit</th>
<th>2 bits</th>
<th>3 bits</th>
<th>4 bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>00</td>
<td>000</td>
<td>0000</td>
</tr>
<tr>
<td>1</td>
<td>01</td>
<td>001</td>
<td>0001</td>
</tr>
<tr>
<td>10</td>
<td>010</td>
<td>010</td>
<td>0100</td>
</tr>
<tr>
<td>11</td>
<td>011</td>
<td>011</td>
<td>0110</td>
</tr>
<tr>
<td>100</td>
<td>0101</td>
<td>100</td>
<td>1001</td>
</tr>
<tr>
<td>101</td>
<td>0110</td>
<td>101</td>
<td>1010</td>
</tr>
<tr>
<td>110</td>
<td>1000</td>
<td>110</td>
<td>1011</td>
</tr>
<tr>
<td>111</td>
<td>1011</td>
<td>111</td>
<td>1100</td>
</tr>
<tr>
<td></td>
<td>1100</td>
<td></td>
<td>1101</td>
</tr>
<tr>
<td></td>
<td>1110</td>
<td></td>
<td>1110</td>
</tr>
<tr>
<td></td>
<td>1111</td>
<td></td>
<td>1111</td>
</tr>
</tbody>
</table>

Each additional bit doubles the number of possible codes.

Storage requirements examples

- If a code requires 5 bits, a document consisting of 4000 such codes will require a total of: 5 x 4000 = 20,000 bits
  - How many **bytes** is that? __________
- If a code requires 8 bits (i.e., a byte), a document consisting of 2000 such codes will require 2000 bytes.
  - How many **bits** is that? __________
- If a code requires 32 bits, a program that needs to store 2000 such codes will require _______ bits or _______ bytes.

Storage Capacity

- Every memory device has a **storage capacity**, indicating the number of bytes it can hold.

<table>
<thead>
<tr>
<th>Unit</th>
<th>Symbol</th>
<th>Number of Bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>kilobyte</td>
<td>KB</td>
<td>$2^{10} = 1024$</td>
</tr>
<tr>
<td>megabyte</td>
<td>MB</td>
<td>$2^{20}$ (over one million)</td>
</tr>
<tr>
<td>gigabyte</td>
<td>GB</td>
<td>$2^{30}$ (over one billion)</td>
</tr>
<tr>
<td>terabyte</td>
<td>TB</td>
<td>$2^{40}$ (over one trillion)</td>
</tr>
<tr>
<td>petabyte</td>
<td>PB</td>
<td>$2^{50}$ (a whole bunch)</td>
</tr>
</tbody>
</table>
Data Representation and Applets

Overview

- Binary representation
- Image representation
- Java Applets
- Data types revisited
- Type conversions

What’s a picture?

- programs represent pictures as grids of picture elements or pixels

Representing Images

- Bitmap: 1 bit
- Grayscale: 8 bits
- RGB Color:
  - 3 colors: red, green, blue
  - 8 bits/color
  - 24 bits

Example: Representing Pixels

- red=116
- green=86
- blue=142
- Color: 10110100 01010110 10001110
Additive/Subtractive Color

We choose 3 primary colors that can be combined to produce almost all visible colors.

**Additive primaries**
- combining light
  - Red
  - Green
  - Blue

**Subtractive primaries**
- combining ink, thus subtracting light
  - Cyan
  - Yellow
  - Magenta

Bitmap image

A 2000 by 1000 bitmap image contains _________ pixels.
Thus, we need _________ bits to represent this image, or _________ bytes.

Grayscale Image

A 2000 by 1000 grayscale image contains _________ pixels.
Thus, we need _________ bytes to represent this image, or _________ bits.

RGB Image

A 2000 by 1000 RGB image contains _________ pixels.
Thus, we need _________ bytes to represent this image, or _________ bits.
Graphics and images in Java

- Images are represented as objects in Java.
- Color components and positions for pixels can be represented as integers (but also in other ways).
- Java libraries allow flexibility, providing many alternative ways of representing and processing images.
- We will be using the `Graphics` class of the `awt` package and `JApplet` from the `swing` package.

Applets

- A Java applet is a program that is intended to be transported over the Web and executed using a web browser.
- An applet doesn't have a `main` method
  - The type of program we have seen so far is a Java `application` - a stand-alone program with a `main` method.

Example: `Einstein.java`
```java
import javax.swing.JApplet;
import java.awt.*;
public class Einstein extends JApplet {
    public void paint(Graphics page) {
        page.drawRect(50, 50, 40, 40); // square
        page.drawRect(60, 80, 225, 30); // rectangle
        page.drawOval(75, 65, 20, 20); // circle
        page.drawLine(35, 60, 100, 120); // line
        page.drawString("Out of clutter, find simplicity.", 110, 70);
        page.drawString("-- Albert Einstein", 130, 100);
    }
}
```

Overview

- Binary representation
- Image representation
- Java Applets
  - Data types revisited
  - Type conversions
Data Representation and Applets

---

**Einstein.java**

```java
import javax.swing.JFrame;
import java.awt.*;

public class Einstein extends JFrame {
    public void paint(Graphics page) {
        page.drawRect(50, 50, 40, 40); // square
        page.drawRect(60, 80, 225, 30); // rectangle
        page.drawOval(75, 65, 20, 20); // circle
        page.drawLine(35, 60, 100, 120); // line
        page.drawString("Out of clutter, find simplicity.", 110, 70);
        page.drawString("-- Albert Einstein", 130, 100);
    }
}
```

---

### Drawing a Line

- Start: 
  - x: 10
  - y: 20
- End: 
  - x: 150
  - y: 45
- Command: `page.drawLine(10, 20, 150, 45);`
  - Or
- Command: `page.drawLine(150, 45, 10, 20);`

---

### Drawing a Rectangle

- Start: 
  - x: 50
  - y: 20
- End: 
  - Width: 100
  - Height: 40
- Command: `page.drawRect(50, 20, 100, 40);`

---

### Drawing an Oval

- Start: 
  - x: 175
  - y: 20
- End: 
  - Width: 50
  - Height: 80
- Command: `page.drawOval(175, 20, 50, 80);`
Drawing an Arc

- An arc is defined by an oval, a start angle, and an arc angle:

\[
\text{drawArc (10, 10, 60, 30, 20, 90)}
\]

Filled vs unfilled shapes

- Instead of using `drawRect()`, `drawOval()` etc, we can use `fillRect()`, `fillOval()` etc
- We can set the color using `setColor()`
  - See `Snowman.java`
  - See also Snowman applet on a webpage

```java
// Snowman.java
// Demonstrates basic drawing methods and the use of color.
import javax.swing.JApplet;
import java.awt.*;
public class Snowman extends JApplet {
    // Draw a snowman.
    public void paint (Graphics page)
    {
        final int MID = 150;
        final int TOP = 50;
        setBackground (Color.cyan);
        page.setColor (Color.blue);
        page.fillRect (0, 175, 300, 50); // ground
        page.setColor (Color.yellow);
        page.fillOval (-40, -40, 80, 80); // sun
        page.setColor (Color.white);
        page.fillOval (MID-20, TOP, 40, 40); // head
        page.fillOval (MID-35, TOP+35, 70, 50); // upper torso
        page.fillOval (MID-50, TOP+80, 100, 60); // lower torso
        page.setColor (Color.black);
        page.fillOval (MID-10, TOP+10, 5, 5); // left eye
        page.fillOval (MID+5, TOP+10, 5, 5); // right eye
        page.drawArc (MID-10, TOP+20, 20, 10, 190, 160); // smile
        page.drawLine (MID-25, TOP+60, MID-50, TOP+40); // left arm
        page.drawLine (MID+25, TOP+60, MID+55, TOP+40); // right arm
        page.drawLine (MID-20, TOP+5, MID+20, TOP+5); // brim of hat
        page.fillRect (MID-15, TOP-20, 30, 25); // top of hat
    }
}
```
### The Java Color Class

- A color in a Java program is represented as an object created from the `Color` class.
- The `Color` class also contains several predefined colors, eg:
  
<table>
<thead>
<tr>
<th>Object</th>
<th>RGB Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Color.black</code></td>
<td>0, 0, 0</td>
</tr>
<tr>
<td><code>Color.blue</code></td>
<td>0, 0, 255</td>
</tr>
<tr>
<td><code>Color.cyan</code></td>
<td>0, 255, 255</td>
</tr>
<tr>
<td><code>Color.orange</code></td>
<td>255, 200, 0</td>
</tr>
<tr>
<td><code>Color.white</code></td>
<td>255, 255, 255</td>
</tr>
<tr>
<td><code>Color.yellow</code></td>
<td>255, 255, 0</td>
</tr>
</tbody>
</table>

- Using a color: `page.setColor(Color.blue);`
- Creating a new color:
  ```java
  Color salmon = new Color(255, 140, 128);
  page.setColor(salmon);
  ```

### Translation of programs into machine code

- **Source code** to **bytecode** is translated by the Java compiler.
- **Bytecode** is then executed by the **bytecode interpreter** which is part of the browser.

### The HTML applet Tag

- An applet is embedded into an HTML file using a tag that references the bytecode file of the applet.
- The bytecode version of the program is transported across the web and executed by a Java interpreter that is part of the browser.

```html
<html>
<head>
<title>The Einstein Applet</title>
</head>
<body>
<applet code="Einstein.class" width=350 height=175>
</applet>
</body>
</html>
```
Overview

- Binary representation
- Image representation
- Java Applets
- Data types revisited
- Type conversions

Variables

- A variable is a name for a location in memory
- A variable must be declared by specifying the variable’s name and the type of information that it will hold

```
int sum;
double milesPerGallon;
String name, petName;
```

Numeric Primitive Data

- The difference between the numeric primitive types is their size and the values they can store:

<table>
<thead>
<tr>
<th>Type</th>
<th>Storage</th>
<th>Min Value</th>
<th>Max Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>byte</td>
<td>8 bits</td>
<td>-128</td>
<td>127</td>
</tr>
<tr>
<td>short</td>
<td>16 bits</td>
<td>-32,768</td>
<td>32,767</td>
</tr>
<tr>
<td>int</td>
<td>32 bits</td>
<td>-2,147,483,648</td>
<td>2,147,483,647</td>
</tr>
<tr>
<td>long</td>
<td>64 bits</td>
<td>&lt; -9 x 10^15</td>
<td>&gt; 9 x 10^15</td>
</tr>
<tr>
<td>float</td>
<td>32 bits</td>
<td>+/- 3.4 x 10^38 with 7 significant digits</td>
<td></td>
</tr>
<tr>
<td>double</td>
<td>64 bits</td>
<td>+/- 1.7 x 10^30 with 15 significant digits</td>
<td></td>
</tr>
</tbody>
</table>

Example: Representing Text

- Characters, including spaces, digits, and punctuation are represented by numeric codes

```
Hi, Heather.
```

<table>
<thead>
<tr>
<th>Character</th>
<th>ASCII Code</th>
<th>Unicode Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hi</td>
<td>72 105</td>
<td>01110100</td>
</tr>
<tr>
<td>,</td>
<td>44</td>
<td>00000000</td>
</tr>
<tr>
<td>heathe</td>
<td>104 114 46</td>
<td>01110100 01110100</td>
</tr>
</tbody>
</table>

The ASCII (American Standard Code for Information Interchange) character set uses eight bits per character, allowing for 256 unique characters.

The Unicode character set extends ASCII to sixteen bits per character, allowing for 65,536 unique characters.
Characters in Java

- A `char` variable stores a single character
- Character literals are delimited by single quotes:
  `'a'   'X'   '7'   '$'   ','   '\n'

```java
char topGrade = 'A';
char terminator = ';', separator = ' ';
char nextToTopGrade = (char)(topGrade + 1);
```

Note the difference between a primitive character variable, which holds only one character, and a `String` object, which can hold multiple characters.

Automatic type conversion

Values of different types can be combined in an assignment or an expression

- Example:
  ```java
  int dollars = 5;
  double money = dollars + 2.50;
  System.out.println(dollars + " dollars");
  ```

  - These are all examples of **widening conversions**, i.e., "smaller" data type \(\rightarrow\) "larger" data type

Overview

- Binary representation
- Image representation
- Java Applets
- Data types revisited
- Type conversions

Converting from one type to another

- **Widening conversions**
  - "small" data type \(\rightarrow\) "larger" one
  - eg: `int` \(\rightarrow\) `double`

- **Narrowing conversions**
  - "large" data type \(\rightarrow\) "smaller" one
  - eg: `double` \(\rightarrow\) `int`

- Narrowing conversions can lose information!
- Narrowing conversions cannot happen automatically (for example, through assignment)
Casting

- *Casting* forces a change of type, even if information is lost
- Can be used for both widening and narrowing conversion
- To cast, put the type in parentheses in front of the value to be converted:

```java
int total = 5;
double result = (double) total / 2;
int answer = (int) result + 4;
double angle = 0; // 0 radians
int x = (int) (Math.cos(angle) * 300);
```

Casting forces a change of type, even if information is lost. It can be used for both widening and narrowing conversion. To cast, put the type in parentheses in front of the value to be converted.

---

Data Conversion

**Widening Conversions**

<table>
<thead>
<tr>
<th>From</th>
<th>To</th>
</tr>
</thead>
<tbody>
<tr>
<td>byte</td>
<td>byte, char, short, int, long, float, or double</td>
</tr>
<tr>
<td>short</td>
<td>byte, char, short, int, long, float, or double</td>
</tr>
<tr>
<td>char</td>
<td>byte, char, short, int, long, float, or double</td>
</tr>
<tr>
<td>int</td>
<td>byte, char, short, int, long, float, or double</td>
</tr>
<tr>
<td>long</td>
<td>Float, double</td>
</tr>
<tr>
<td>Float</td>
<td>Double</td>
</tr>
</tbody>
</table>

**Narrowing Conversions**

<table>
<thead>
<tr>
<th>From</th>
<th>To</th>
</tr>
</thead>
<tbody>
<tr>
<td>byte</td>
<td>char</td>
</tr>
<tr>
<td>short</td>
<td>byte, or char</td>
</tr>
<tr>
<td>char</td>
<td>byte, or short</td>
</tr>
<tr>
<td>int</td>
<td>byte, char, short, or int</td>
</tr>
<tr>
<td>long</td>
<td>Float, byte, or char, or int</td>
</tr>
<tr>
<td>Float</td>
<td>byte, short, or char, int, or long</td>
</tr>
</tbody>
</table>

**How to use cast?**

*Forcing floating point division between int expressions*

```java
int qp = 35;
int credits = 10;
double gpa = (double) qp / credits;
```

gpa should be 3.5

```java
int qp = 35;
int credits = 10;
double gpa = (double) (qp / credits);
```

How to use cast?

*Scaling a double and converting to int*

```java
double gpa = 3.2;
int gpaPercent = (int) (gpa / 4) * 100;
```

gpaPercent should be 80

```java
double gpa = 3.2;
int gpaPercent = (int) ((gpa / 4) * 100);
```

How to use cast?