Data types and their representation

Jordi Cortadella
Department of Computer Science
Outline

• Data representation

• Boolean expressions

• Real numbers
The memory

```
Address
1036 0 0 1 0 0 1 0 0 1 1 1 0 0 0 0 1 0 1 0 1 0 0 1 0 0 1 0 1 0 0 0 1 0
1040 1 0 0 1 0 0 1 0 0 1 1 1 0 0 0 0 0 1 1 0 1 0 0 1 0 1 0 1 0 0 0 0
1044 0 1 0 0 1 0 1 0 1 0 0 0 0 0 1 0 0 0 0 1 0 1 0 0 0 0 0 0 0 1 0 0
1048 0 0 1 0 0 1 0 1 1 1 0 0 0 0 0 1 0 0 0 0 1 0 0 1 0 1 0 0 0 0 0 0 1
1052 0 0 1 0 0 1 0 1 1 1 0 0 0 0 0 1 0 0 0 0 1 0 0 1 0 1 0 0 0 0 0 0 1
1056 0 0 1 0 0 1 0 1 1 1 0 0 0 0 0 1 0 0 0 0 1 0 0 1 0 1 0 0 0 0 0 0 1
1060 H e l l o w o r l d
1064 1 0 0 1 0 1 0 1 1 0
1068 0 1 0 1 0 1 0 1 1 0
1072 0 1 1 1 0 0 1 1 0 1 0 0 1 0 0 0 1 0 0 0 1 0 1 0 1 0 1 0 0 0 0 0 0
1076 0
1080 0 1 0 1 1 0 1 0 0 0 0 0 1 0 0 0 0 1 1 1 1 1 0 0 1 0 0 0 1 0 0 0 1
1084 0 0 1 0 0 0 1 0 0 1 0 0 0 0 0 1 1 0 0 1 0 0 0 0 0 1 0 0 0 0 1
```

int

byte

String

("Hello world")
How much memory do we have?

Our laptop has few Gbytes of memory:

\[
\begin{align*}
1 \text{ Kilo (K)} &= 2^{10} &= 1,024 \\
1 \text{ Mega (M)} &= 2^{20} &= 1,048,576 \\
1 \text{ Giga (G)} &= 2^{30} &= 1,073,741,824 \\
1 \text{ Tera (T)} &= 2^{40} &= 1,099,511,627,776
\end{align*}
\]
Number representation

100 10 1
2 1 7

base 10

125 25 5 1
1 3 3 2

base 5

81 27 9 3 1
2 2 0 0 1

base 3

128 64 32 16 8 4 2 1
1 1 0 1 1 0 0 1

base 2

CCXVII

Roman
Write the binary representation

Design a procedure that, given a number \( n \), writes its binary representation (in reverse order).

```c
void base2(int n);
```

// Pre: \( n \geq 0 \)
// Writes the binary representation of \( n \)
// in reverse order.
Write the binary representation

// Pre: n ≥ 0
// Writes the binary representation of n
// in reverse order.

void base2(int n) {
    while (n > 1) {
        cout << n%2;
        n = n/2;
    }
    cout << n << endl;
}
Design a procedure that, given a number \( n \) and a base \( b \), writes the representation of \( n \) in base \( b \) (in reverse order).

```
// Pre: \( n \geq 0, \ 2 \leq b \leq 9 \).
// Writes the binary representation of \( n \)
// in base \( b \) (in reverse order).
void base(int n, int b);
```
Boolean expressions
Boolean Algebra

George Boole, 1815-1864
Maximum of three numbers

// Returns max(a, b, c)
int max3(int a, int b, int c);

```plaintext
// a > b 
and 
// a > c
// b > c
else

// a >= b 
and 
// a >= c
// b >= c
else
```

Introduction to Programming © Dept. CS, UPC 11
// Returns max(a, b, c)
int max3(int a, int b, int c) {
    if (a > b and a > c) {
        return a;
    } else {
        if (b > c) {
            return b;
        } else {
            return c;
        }
    }
}
Maximum of three numbers

// Returns max(a, b, c)
int max3(int a, int b, int c) {
    if (a > b and a > c) return a;
    if (b > c) return b;
    return c;
}
Boolean operators

if (a > b and a > c) ...

while (i >= 10 or c == 0) ...

if (not (a < b and a < c)) ...
### Truth Tables

**& (AND)**

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>a \textbf{and} b</th>
</tr>
</thead>
<tbody>
<tr>
<td>false</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
</tbody>
</table>

**\lor (OR)**

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>a \textbf{or} b</th>
</tr>
</thead>
<tbody>
<tr>
<td>false</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>true</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
</tbody>
</table>

**\neg (NOT)**

<table>
<thead>
<tr>
<th>a</th>
<th>\textbf{not} a</th>
</tr>
</thead>
<tbody>
<tr>
<td>false</td>
<td>true</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
</tr>
</tbody>
</table>
Boolean expressions

\[ x > 2 \text{ and } x < 7 \equiv x \geq 3 \text{ and } x \leq 6 \]

\[ x \leq 2 \text{ or } x > 7 \]

\[ x \leq 2 \text{ and not } (x \leq 2) \equiv x > 2 \]

\[ x = 5 \]
Boolean expressions

\[ x > 2 \quad \text{and} \quad x < 7 \equiv x \geq 3 \quad \text{and} \quad x \leq 6 \]

\[ x \leq 2 \quad \text{or} \quad x > 7 \]

\[ x \leq 2 \quad \text{not} \quad (x \leq 2) \equiv x > 2 \]

\[ \text{not} \quad (x == 5) \equiv x != 5 \]
\[ x = 0 \text{ or } x = 6 \]
Exercise

\[ x \equiv 0 \text{ or } x > 5 \]
Exercise

\[ x \% 2 == 0 \]
Exercise

(x > -1 and x < 3) or
(x >= 6 and x < 11)
Complement of and/or

\[(x > 2 \text{ and } x < 7)\]

\[\text{not} \ (x > 2 \text{ and } x < 7)\]

\[\text{not} \ x \leq 2 \text{ or } x \geq 7\]

De Morgan’s law
De Morgan’s law

\[ \neg (e_1 \text{ and } e_2) \equiv \neg e_1 \text{ or } \neg e_2 \]
\[ \neg (e_1 \text{ or } e_2) \equiv \neg e_1 \text{ and } \neg e_2 \]

Exercise

Simplify: \[ \neg (x \geq y \text{ or } y \% 2 = 0) \]

\[ x < y \text{ and } y \% 2 \neq 0 \]
Operator precedence

\[ a + b \times c \]

(a + b) \times c

\[ a \text{ or } b \text{ and } c \]

(a or b) and c
Real numbers
Intersection of circles

• Write a program that reads the center and the radius of two circles and prints “yes” or “no” depending on whether they intersect or not.

• Example:

\[
\begin{array}{c}
x_1, y_1, r_1 \\
2, 5.3, 1.34
\end{array} \quad \begin{array}{c}
x_2, y_2, r_2 \\
0, 0, 2
\end{array}
\]

no

\[
\begin{array}{c}
1.5, 2.5, 10 \\
0.5, 3.6, 4.3
\end{array}
\]

yes
Circles intersect if and only if the distance between centers is smaller than or equal to the sum of radii.
Intersection of circles

\[ \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \leq r_1 + r_2 \]

\[ (x_1 - x_2)^2 + (y_1 - y_2)^2 \leq (r_1 + r_2)^2 \]
// Reads the center and radius of two circles
// and prints whether they intersect or not.
int main() {
    double x1, y1, r1, x2, y2, r2; // Real numbers
    cin >> x1 >> y1 >> r1 >> x2 >> y2 >> r2;

double dx = x1 - x2;
dx = dx*dx; // (x1 - x2)^2

double dy = y1 - y2;
dy = dy*dy; // (y1 - y2)^2

double r = r1 + r2;
r = r*r; // (r1 + r2)^2

bool intersect = dx + dy <= r; // true or false

if (intersect) cout << "yes" << endl;
else cout << "no" << endl;
}
Boolean variables

```cpp
bool b = false;
b = i <= n; // true if i <= n, false if i > n
b = x + y > z;
b = (i < 10) and (j >= 20);
b = (large and not found) or (x > y);
b = true;
```

```cpp
if (b) { ... } // if (b == true) { ... }
```

```cpp
while (not b) {...} // while (b == false) { ... }
```
Real numbers

• Two types:
  – float (single precision, 32 bits)
  – double (double precision, 64 bits)

• Arithmetic operations: \(+\) \(-\) \(*\) \(/\) (no remainder)

• Real constants:
  \(2\) \(-5.003\) \(3.1416\) \(1.4e9\) \(0.6E-15\)

\(1.4 \cdot 10^9\) \(0.6 \cdot 10^{-15}\)
Type conversion

• Arithmetic operations between integer and real values usually imply an implicit type conversion.

• Be careful:

```c
int i=3, j=2;
double x;
x = i/j; // x = 1.0
x = i/double(j); // x = 1.5
x = double(i)/j; // x = 1.5
x = double(i/j); // x = 1.0
x = i/2; // x = 1.0
x = i/2.0; // x = 1.5

i = x; // i = 1
j = 3.14159265; // j = 3
```
// Returns $x^2$
double sq(double x) {
    return x*x;
}

// Reads the center and radius of two circles
// and prints whether they intersect or not.
int main() {
    double x1, y1, r1, x2, y2, r2;
    cin >> x1 >> y1 >> r1 >> x2 >> y2 >> r2;

    if (sq(x1 - x2) + sq(y1 - y2) <= sq(r1 + r2)) cout << "yes";
    else cout << "no";

    cout << endl;
}
Variables are stored in memory locations and internally represented as bit vectors.

Boolean expressions and variables:
- Can take two values: true or false.
- Use negative thinking when simpler than positive thinking (apply De Morgan’s law).

Real values:
- Can be represented with a limited precision.
- Be careful with int ↔ double conversions.