CONVERTING FORMULAS TO C++

Variables, Operators, and Functions

Delroy A. Brinkerhoff

VARIABLES

- Variables must be defined and initialized before they may be used
- All of the following examples assume that the variables are defined and, where necessary, are initialized
- Variable names must be unique within a scope
- Variables in formulas may have subscripts but variables in C++ may not
  - m_0 may be converted to `m0;`
  - $F_n = F_{n-1} + F_{n-2}$ may be converted to `Fn = Fn1 + Fn2;` or `f = f1 + f2;`

MULTIPLICATION

- Formulas denote multiplication by placing variables next to each other: PV
- C++ requires an explicit operator: *
- The formula T = PV is translated into C++ as
  - T = P * V
  - Temperature = Pressure * Volume

DIVISION

- Formulas denote division in two ways:
  - $v = x/t$
  - $v = \frac{x}{t}$
  - The second way can *imply* grouping: $\frac{P}{T_2 - T_1}$, T_2 - T_1 must be done *before* the division
  - C++: P / (T2 – T1)

INTEGER DIVISION

- Integer division can cause unexpected results
    - $c = \frac{5}{9}(\mathrm{f} - 32)$
    - c = 5 / 9 * (f – 32), *always* produces a 0
  - Problem is easily corrected
    - c = 5.0 / 9.0 * (f – 32)
    - c = 5 * (f – 32) / 9
    - c = (f – 32) * 5 / 9

UNARY MINUS

- C++ has both a unary minus and a unary plus (plus isn't really useful)
- Both convert from formulas straight to C++
  - +N
  - -N
  - -N can be implemented as -1 * N but this looks cluttered and amateurish

$$payment = \frac{PR}{1 - (1 + R)^{-N}}$$

```
double payment = P * R / (1 - pow(1 + r, -N));
```

EXPONENTIATION

- Like Java, C++ does not have an exponentiation operator
- When squaring or even cubing an integer, it's just about as fast and easy to multiply the number by itself
  - x^2 `= x * x;` x^3 `= x * x * x;`
- For higher powers, or variable, negative or factional exponents, use the pow function
  - `y =` $x^{p/q}$ `y =` x^{-n}`;` `y = pow(x, p/q);`
  - The arguments form a group and so don't require parentheses
  - Remember that pow returns a double
  - The return value is a single value that doesn't require parentheses

SQUARE ROOTS

- The sqrt function calculates and returns a square root
- Everything under the radical is part of the function's non-negative argument (i.e., the argument is self-grouping)
- The return value also acts as a group

$$m = \frac{m_0}{\sqrt{1 - \frac{v^2}{c^2}}}$$

```
m = m0 / sqrt(1 - v*v / (c*c));
m = m0 / sqrt(1 - pow(v, 2) / pow(c,2));
```

SYMBOLS OF INCLUSION

- Mathematical formulas can use ( ), [ ], and { } for grouping
- C++ can only use ( )
- You can always use parentheses even when precedence and associativity resolve all ambiguity
  - No magically correct number of parentheses
  - Too many parentheses make the statement harder to read and increase the likelihood of mismatched or unbalanced parentheses

$$P = F\left[\frac{r}{(1+r)^n - 1}\right]\left[\frac{1}{(1+r)}\right]$$

```
P = F * (r / (pow(1 + r, n) - 1)) * (1 / (1 + r));
```