Example: Numerical Evaluation of Polynomials

\[ p(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{n-1} x^{n-1} + a_n x^n = \sum_{i=0}^{n} a_i x^i \]
“Pseudocode” in Cheney and Kincaid

\begin{align*}
\text{integer } i, n; \text{ real } p, x; \text{ real array } (a_i)_{0:n} \\
p &\leftarrow 0.0 \\
\text{for } i = 0 \text{ to } n \text{ do} \\
&\quad p \leftarrow p + a_i x^i \\
\text{end for}
\end{align*}

\begin{itemize}
\item Declare variables and their types
\item $n, x$ and $a_i$ are assumed to have assigned values
\item Code statements to execute in sequence
\item Array notation means $n+1$ real values indexed as $a_0$ through $a_n$
\item Iterative 'for' loop: body of loop is executed iteratively $n$ times with the variable $i$ assigned the values 0, 1, 2, ..., $n-2$, $n-1$, $n$ in sequence
\item $\leftarrow$ is the 'assignment' operator:
\item the value of the expression on the right is assigned to the variable on the left.
\item The value that is accumulated in the variable $p$ after $n+1$ iterations will be the value of the polynomial $p(x)$
\end{itemize}
Image for Algorithmic Variables
Expanding the for Loop

initialize: $p = 0$
$i = 0: p = a_0$
$i = 1: p = a_0 + a_1 x$
$i = 2: p = a_0 + a_1 x + a_2 x^2$
$i = 3: p = a_0 + a_1 x + a_2 x^2 + a_3 x^3$
$\vdots$

Result: $p = \sum_{i=0}^{n} a_i x^i$
More Detailed Pseudocode

Now include detailed pseudocode for evaluating $x'$

```plaintext
integer $i, j, n$; real $p, q, x$; real array $(a_i)_{0:n}$
p ← 0.0
for $i = 0$ to $n$ do
  $q ← a_i$
  for $j = 0$ to $i - 1$ do
    $q ← q \ast x$
  end for
  $p ← p + q$
end for
```

Outer 'for' loop

Inner 'for' loop
Nested Evaluation of Polynomials

From section 1.1 of Cheney and Kincaid

\[ p(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{n-1} x^{n-1} + a_n x^n \]

is much more efficiently computed as

\[ p(x) = a_0 + x \left( a_1 + x \left( a_2 + \cdots + x \left( a_{n-1} + x \left( a_n \right) \right) \cdots \right) \right) \]
Pseudocode for Nested Algorithm

```
integer i, n; real p, x; real array \((a_i)_{0:n}\)

\(p \leftarrow a_n\)

for \(i = n - 1\) to 0 do
    \(p \leftarrow a_i + xp\)
end for
```
Expanding the for Loop

initialize: \( p = a_n \)
\( i=n-1: p=a_{n-1} + a_n x \)
\( i=n-2: p=a_{n-2} + a_{n-1} x + a_n x^2 \)
\( i=n-3: p=a_{n-3} + a_{n-2} x + a_{n-1} x^2 + a_n x^3 \)
\( i=n-4: p=a_{n-4} + a_{n-3} x + a_{n-2} x^2 + a_{n-1} x^3 + a_n x^4 \)
\[ \vdots \]

Result: \( p = \sum_{i=0}^{n} a_i x^i \)
Synthetic Division of Polynomials

Let \( p(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{n-1} x^{n-1} + a_n x^n \)

Find \( q(x) = b_0 + b_1 x + b_2 x^2 + \cdots + b_{n-2} x^{n-2} + b_{n-1} x^{n-1} \)

such that \( p(x) = (x - r) q(x) + p(r) \) for some \( r \)

Equate coefficients of \( x^i \) in both expressions of \( p(x) \)

The algorithm for finding \( b_i \) recursively from \( b_{i+1} \) and \( a_{i+1} \) appears

It is the same algorithm as nested evaluation of the polynomial \( p(x) \) except that intermediate terms are stored in an array and are the \( b_i \) coefficients
Pseudocode for Synthetic Division Algorithm

```plaintext
integer i, n; real p, r; real array (a_i)_{0:n}, (b_i)_{0:n-1}
b_{n-1} ← a_n
for i = n−1 to 0 do
    b_{i-1} ← a_i + rb_i
end for

After n iterations
b_i contain coefficients of q(x)
b_{-1} contains p(r)
```

See
http://en.wikipedia.org/wiki/Horner%27s_method
http://en.wikipedia.org/wiki/Polynomial_remainder_theorem
Model of Operating System Processes

Your user programs

Terminal emulator and shell

Operating system kernel

Memory

Disk Drive

External Devices
Infants progress through a “pointing development” stage, but must then learn language and speech to successfully interact with a complex world.
"Point and Click" Versus Command Language

```c
i = 0;
while (i < n) {
    if (i != k) {
        j = k + 1;
        while (j < n) {
            coeff[i][j] -= coeff[i][k] * coeff[k][j];
            j++;
        }
        j = 0;
        while (j < n) {
            invert[i][j] -= coeff[i][k] * invert[k][j];
            j++;
        }
    }
    i++;
}

gcc rlc_integrate.c -o rlc_integrate -lm
./rlc_integrate 6.2832 10 0 0 0 10 0.01 >| rlc_integrate.dat
less rlc_integrate.dat

plot 'rlc_integrate.dat' using 1:2 with lines, '' using 1:3 with lines
```

Language is the only mechanism to effectively interact with a processor to solve complex numerical problems
Reference Resources on Class Web Page

Basic Linux commands:
http://www.physics.smu.edu/fattarus/LinuxCommands.html

bash shell key commands:
http://www.physics.smu.edu/fattarus/BashCommands.html

less pager key commands:
http://www.physics.smu.edu/fattarus/LessCommands.html

gnuplot commands:
http://www.physics.smu.edu/fattarus/GnuplotCommands.html

Table of hexadecimal digits:
http://www.physics.smu.edu/fattarus/HexDigits.html

Linux command tutorials:
http://www.physics.smu.edu/fattarus/LinuxTutorial1.html
http://www.physics.smu.edu/fattarus/LinuxTutorial2.html
A Good On-Line Tutorial

http://www.ee.surrey.ac.uk/Teaching/Unix/index.html

In addition,
http://www.tuxfiles.org/linuxhelp/linuxcommands.html
displays a nice cheat-sheet
A “current working directory” is always defined. When you first log in it is set to your home directory, but may be changed at will with commands.
Referring to Files

Absolute, or “fully-qualified” file names:
/home/my_name/larry
/home/my_name/mysubdir/moe

Note the / characters as directory separators

For files in or below the current working directory:
larry
mysubdir/moe

Use the pseudo-files present in every directory:
. refers to current directory  .. refers to one directory level above
to form relative file names:
../subdir1/larry
../../subdir1/subdir2/moe
./curly
../subdir1/../../subdir2/../../../subdir1/subdir2/../../../subdir1/shemp

Use the special character '~' as an alias for your home directory:
~/subdir1/larry
General Command Syntax

```
command_name -command_options file_name_1 file_name_2 ...
```

- **command_name**: Name of command
- **command_options**: Options for running command, if any, usually starting with `-`
- **file_name_1 file_name_2 ...**: File(s) or directories that command should operate on, if any

Examples:

- `cp -f larry moe`
  Copy file 'larry' into file 'moe', with a 'force' option (overwrite file 'moe' if it exists)

- `ls -l curly`
  List files in directory 'curly' in long format

- `rm -i shemp`
  Remove file 'shemp' using interactive mode (verify before file is deleted)
Default Connections for Standard Input and Output

Process running any command

standard input

standard output

standard error

[Diagram of standard input, output, and error connections]
Redirecting Standard Input from a Disk File

Process running any command

standard input

standard output

standard error

- Redirecting Standard Input from a Disk File

- Process running any command

- standard input

- standard output

- standard error
Redirecting Standard Output to a Disk File

Process running any command

standard input → standard output

standard error
Redirecting output and input on the Command Line

ls > file_list.txt  to write results of ls command to file
ls >> file_list.txt to append results of ls command to contents already in file
ls >| file_list.txt to overwrite results of ls command to existing file

command < data.txt to feed contents of a file into standard input of a command