Laplace Project

Overview
Array class
OptionList class

Distribution
- The project distribution (laplace.tgz) contains:
  - laplace.C (main)
  - Array class
  - OptionList class

Getting Started
- Step 0:
  - look at the Makefile
  - compile the distributed program, make sure it runs
- Step 1:
  - make sure you can compile and run an OpenMP program
  - write a parallel "hello, world" program

C++ Arrays
- 1D arrays (vectors) in C and C++ are pointers to contiguous sections of memory
  ```
  double v[100];
  ```
- The size does not have to be a constant
  ```
  double *makeVector(int n) {
    return new double[n];
  }
  ...
  double *v = makeVector(100);
  ```
C++ Arrays (cont’d)

- 2D arrays are vectors of pointers to vectors

```c
double a[100][100];
```

When (and only when) the compiler knows the size of the first dimension you can use a double-index expression:

```c
x = a[i][j];
```

breaking news: C++ seems to have figured this out (g++, CC)...

C++ arrays are arranged in row-major order

- `a[i][j+1]` is in the memory location immediately after `a[i][j]`

Fortran Arrays

- 2D arrays in Fortran are
  - laid out in an N X M block of memory
  - arranged in column-major order

```fortran
REAL A(8,8)
```

Column- vs Row-Major

- Why does it matter that you know whether an array is column- or row-major?
  - answer: cache performance

  When the CPU reads the first array element from memory, it also fetches the next few elements
  - example: 32-byte cache lines, double-precision elements (8 bytes each)
  - what happens when a program first reads from an array?
  - C++: fetch `a[0][0]` through `a[0][3]`
  - FORTRAN: fetch `A(0,0)` through `A(3,0)`
Column- vs Row-Major

- What this means for your program:
  - If you use C++ arrays, the inner loop should iterate over columns for best cache performance:
    ```cpp
    for (int i = 0; i < N; i++)
      for (int j = 0; i < M; j++)
        a[i][j] = f(i,j);
    ```
  - If you use Fortran arrays, the inner loop should iterate over rows:
    ```fortran
    DO J = 1, M
      DO I = 1, N
        A(I,J) = F(I,J)
      END DO
    END DO
    ```

Array Class

- The Array class distributed with the project implements Fortran-style arrays
  - contiguous N X M (or N x M x L) block of doubles
  - column-major order
  - overloads the () operator to allow easy reference to array elements
    ```cpp
    for (int j = 0; j < M; j++)
      for (int i = 0; i < N; i++)
        A(i,j) = i*j;
    ```
- Why bother?
  - flexibility (written before C++ figured out dynamic 2D arrays)
  - compatibility with BLAS subroutine library
    - hand-coded, written for Fortran, can be linked with C/C++

OptionList Class

- The OptionList class provides an interface to command-line parameters
- Example:
  ```cpp
  void init(int argc, char *argv[]) {
    OptionList opt(argc,argv);
    nx = opt.getInt("nx",200);
    p = opt.getFlag("p",false);
    f = opt.getDouble("f",1.0);
  }
  ```

Output

- Your program should write the final computed temperatures in a form that can be visualized
- Matlab, R, and other packages have ways of displaying “images”
  - 2D array of integers
  - each integer is an index into a color map
  - color map entries are (R,G,B) values
- Matlab has several pre-defined color maps, including one for temperature values
  - white = very hot
  - red, yellow, etc are decreasing temperatures
  - black = very cold
Suggestions

- Test the program on a small (10 x 10) grid
- When the sequential program is working, compute the solution for a 100 x 100 grid and plot the result
- Time the sequential program on bigger grids (200 x 200, 400 x 400, etc)
- Add OpenMP pragmas to parallelize the program
- Report timings for a large (1000 x 1000) grid, using up to 8 processors