ESc 101: Fundamentals of Computing

Lecture 31

Mar 29, 2010

Splitting Code into Files

- When the code is large, it should be split into files.
- Each file typically containing a set of related functions.
- All the constants and functions prototypes should be declared in a header file which is included in all code files.
- The `main()` function should be put in a separate file: this allows the other functions to be used by different programs.

Example

We can use the matrix functions to read a matrix and invert it:

```
#include "matrix.h"

int main()
{
     float A[N][N];
     float invA[N][N];
     float det;
     int n;

     read_matrix(A, &n);

     det = inv_matrix(A, invA, n);
```

Example

```
    if (is_zero(det))
        printf("Matrix not invertible\n");
    else {
        printf("Determinant = %5.2f\n", det);
        output_matrix("Inverse of A =\n", invA, n);
    }
}
```

Examples

With a little more coding, we can also use it to:

- Check linear independence of vectors
- Solve a system of linear equations
- Given a linear subspace, find an orthogonal basis for it.
- And many other matrix operations.

Compiling Using a File

- With lots of files to compile, it is easier to put the compilation command in a file and execute the file.
- For example, we put the following in a file:
  `gcc -o matrix matrix-main.c matrix-inv.c matrix-ops.c matrix-det.c matrix-io.c -std=c99`
- To compile, we can now execute the file using:
  `sh <filename>`

Fibonacci Numbers

Definition

Fibonacci numbers are defined as follows: $F_0 = 1 = F_1$, and $F_n = F_{n-1} + F_{n-2}$ for $n > 1$.

We can write a function to compute Fibonacci numbers in two ways.

Method I: Using Loops

```
int Fib_loop(int n)
{
    int F[N]; // stores Fibonacci sequence

    F[0] = 1; // first two values
    F[1] = 1;

    for (int m = 2; m <= n; m++)
        F[m] = F[m-1] + F[m-2];

    return F[n];
}
```

Method II: Using Recursion

```
int Fib_rec(int n)
{
    if ((n == 0) || (n == 1))
        return 1;

    return Fib_rec(n-1) + Fib_rec(n-2);
}
```

Why is Method I Much Faster?

- The recursive function computes a value multiple times.
- For example, both `Fib_rec(n)` and `Fib_rec(n-1)` compute `Fib_rec(n-2)`.
- This is wasteful!

Improving Method II

```
int F[N]; // stores Fibonacci sequence
int m = 1; // value until which the sequence is computed

int Fib_rec_imp(int n)
{
    if ((n == 0) || (n == 1)) {
        F[n] = 1; // set first two numbers
        return 1;
    }

    if (n > m) { // number not already computed
        F[n] = Fib_rec_imp(n-1) + Fib_rec_imp(n-2); // compute
        m = n; // reset m
    }

    return F[n]; // return the number
}
```