Part III

Our running example, starting in C

mymatrix.h/main.c/mymatrix.c

```c
#ifndef _MYMATRIXH_
#define _MYMATRIXH_
/* struct to hold a matrix */
struct matrix {
   int rows,cols;     /* matrix dimensions */
   double *elements; /* points to elements */
};
/* initialize matrix (needed before usage) */
void matrix_construct(struct matrix *m,int r,int c);
/* free memory held by matrix (do after last usage) */
void  matrix_destructor(struct matrix *m);
/* access matrix elements (through a pointer) */
double * matrix_element(struct matrix *m,int i,int j);
/* set all matrix elements to value */
void matrix_fill(struct matrix *m,double value);
#endif
```

```c
#include <stdio.h>
#include "mymatrix.h"
void print(struct matrix *m) {
   int i,j;
   for (i=0; i < m->rows; i++) {
      for (j=0; j < m->cols; j++)
         printf("%8.2g ",*matrix_element(m,i,j));
      printf("\n");
   }
}
int main() {
   struct matrix A;
   matrix_construct(&A, 5, 5);
   matrix_fill(&A, 0.0);
   *matrix_element(&A, 0, 0) = 1.3;
   *matrix_element(&A, 4, 3) = -5.2;
   *matrix_element(&A, 1, 3) = -3.3e-4;
   print(&A);
   matrix_free(&A);
}
```

```c
#include <stdlib.h>
#include "mymatrix.h"
void matrix_construct(struct matrix *m,int r,int c) {
   m->elements = (double *)malloc(sizeof(double )*r*c);
   if (m->elements==0) exit(1); /* exit program */
   m->rows = r;
   m->cols = c;
}
void matrix_free(struct matrix *m) {
   free(m->elements);
}
double * matrix_element(struct matrix *m,int i,int j) {
   return  m->elements+i*m->cols+j;
}
void matrix_fill(struct matrix *m, double value) {
   int i,j;
   for (i=0; i < m->rows; i++)
      for (j=0; j < m->cols; j++)
         *matrix_element(m,i,j)=value;
}
```

makefile

```
CC = gcc
CFLAGS = -O3 -march=native
main:  main.o mymatrix.o
    $(CC) main.o mymatrix.o -o main
main.o:  main.c mymatrix.h
mymatrix.o:  mymatrix.h mymatrix.c
```

```
$ make
gcc -O3 -march=native -c -o main.o main.c
gcc -O3 -march=native -c -o mymatrix.o mymatrix.c
gcc main.o mymatrix.o -o main
$ main 1.3 0 0 0 0
0 0 0 -0.00033 0
0 0 0 0 0
0 0 0 0 0
0 0 0 -5.2 0
```