Modeling With Data: The Cheat Sheet

Basic C

Types
int Integer
double Real Number
char character

Creating a structure:
typedef struct{
  char* name;
  int  number;
} person;

Declare & initialize:
int i = 8;
int an_array[] = {2, 4, 6, 8}
char string[50] = "Hi there."
int *array_ptr = malloc(sizeof(int)*20)

The three laws of pointers
(1) To declare a pointer to an integer, use int *k.
(2) After its declaration, to refer to the int being pointed to, use *k.
(3) After its declaration, to refer to the pointer itself, use k.

For example, if p1 is an existing pointer, and p2 is being declared, set p2’s address equal to p1’s with int *p1 = p2, because p1 abides by rule 1 and p2 abides by rule 3.

Comparisons
(a > b)  a is greater than b
(a < b)  a is less than b
(a >= b) a is greater than or equal to b
(a <= b) a is less than or equal to b
(a == b) a equals b
(a != b) a is not equal to b
(a && b) a and b
(a || b) a or b
(!a)  not a

A minimal program
#include <stdio.h>
int main(){
  printf("Hi \n");
}

If you haven’t built a makefile yet, name this hello.c and compile with

gcc -g -Wall -std=gnu99 hello.c

Flow

Conditional evaluation:
if (condition){
  do_if_true;
} else {
  do_if_false;
}

Three types of loop: while, do-while, and for:
while (condition){
  do {
    do_while_true;
  } while (condition);
}
for (pre-loop; condition; at_each_loop){
  do_while_true;
}

For example, the standard means of operating on every element of an array declared with array[n]:
for (int i=0; i<n; i++){
  operation(array[i]);
}

Printf family format specifiers
%i integer
%g general real number
%s string
%f float
%f%lf double (formerly long float)
%-left-justify
%* allow n spaces for the number
%n allow n spaces after the decimal
\ begin a new line
\t tab
\" quotation mark
\(newline) continue string on the next line

Matrices and data

Per-element operations

gsl_matrix *a, *b;
gsl_vector *av, *bv;
double x;

gsl_matrix_add (a,b);
gsl_matrix_sub (a,b);
gsl_matrix_mul_elements (a,b);
gsl_matrix_div_elements (a,b);
gsl_vector_add (av,bv);
gsl_vector_sub (av,bv);
gsl_vector_mul (av,bv);
gsl_vector_div (av,bv);
gsl_matrix_scale (a,x);
gsl_matrix_add_constant (av,x);
gsl_vector_scale (av,x);
gsl_matrix_add_constant (av,x);
apop_vector_log(a);
apop_vector_exp(a);

Linear algebra
int *gsl_blas_ddot(gsl_vector *l, gsl_vector *r, double *out);
apop_data *apop dot(apop_data *l, apop_data *r, ...);
double apop_matrix_determinant(gsl_matrix *in);
gsl_matrix * apop_matrix_inverse(gsl_matrix *in);
The GSL provides simple vector and matrix structures.

The usual conditions:
- All four of these must go between two queries (that return the same
  number of columns), e.g.:
  ```sql
  select * from t1
  union
  select * from t2
  ```

**Comment Corner**
- C, one line //two slashes
- SQL, one line --two dashes
- Gnuplot & shell languages #number sign
- C & SQL, multiline /* star-slash
  to slash-star*/