21 Structures

It is possible to collect data into packages, called 'structures.' For example, the following declaration would be intended for complex numbers

```
struct {double re, im;} z;
```

The variable z would be stored in 16 bytes, and its two components — which are double-precision numbers — would be referred to as

```
z.re
z.im
```

respectively. This 'struct {etcetera}' is a new kind of **type.** Usually, one introduces the type via **typedef**:

```
typedef struct COMPLEX { double re, im; }
COMPLEX;

  COMPLEX x;
```

Note: the components of a `struct` are usually called *fields*.
We could have functions

```
COMPLEX * make_complex ( double re, double im )
{
  COMPLEX * result = calloc ( 1, sizeof ( COMPLEX ) );

  result->re = re;
  result->im = im;

  return result;
}

COMPLEX * sum ( COMPLEX * a, COMPLEX * b )
{
  return make_complex ( a->re + b->re, a->im + b->im );
}
```

and so on.

21.1 Matrix structures

It is definitely useful to be able to work with matrices of any size:

```
typedef struct MATRIX
{ int height, width; double ** entry; }
MATRIX;


MATRIX * make_zero_matrix ( int height, int width )
{
  MATRIX * mat = (MATRIX *) calloc( 1, sizeof ( MATRIX ) );
  mat->height = height; mat->width = width;
  double * pool = calloc( height * width, sizeof ( double ) );
  mat->entry = (double**) calloc ( height, sizeof ( double * ) );
  int i;
  for (i=0; i<height; ++i)
    mat->entry[i] = pool + i * width;
  return mat;
}

void print_matrix ( MATRIX * a )
{
  int i,j;
  for ( i=0; i<a->height; ++i )
  {
    for ( j=0; j<a->width; ++j )
      printf( " %6g", a -> entry[i][j] );
    printf("\n");
  }
}
```

21.2 Structures and arrays

Compare the following

```
typedef struct MATRIX
{ int height, width; double ** entry; }
MATRIX;

typedef struct OTHER_MATRIX
{ int height, width; double entry[10][10];}
OTHER_MATRIX;
```

The size of a MATRIX structure is 12 or 16 bytes, depending on the architecture. The size of an OTHER_MATRIX structure is 808 bytes. That is, it includes the array. (It is debatable whether it should include a pointer to where the array begins, but apparently it does not.)

If a structure is passed as an argument to a routine or function, the entire structure is copied. This is different from arrays.