15 Initialising arrays

15.1 Days of week and arithmetic modulo 7

Arithmetic modulo 7 is useful for converting dates to day-of-week. *For days falling in this century*, given a date $dd/mm/yy$,

$$yy + yy/4 + \widehat{mm} + dd + 6 \quad +6 \text{ in one case}$$
$$\text{MODULO } 7$$

Where $\widehat{mm}$ is

$$0, 3, 3, 6, 1, 4, 6, 2, 5, 0, 3, 5$$

depending on mm (1,2,3, ... 12).

For example, $1/2/17$:

$$17 + 17/4 + 3 + 1 + 6 = 31 \mapsto 3$$

0 is Sunday; this is Wednesday.

Or for $1/5/17$:

$$17 + 17/4 + 1 + 1 + 6 = 29 \mapsto 1.$$

Monday.

The special case: in January and February of a leap year, 6 should be added (or 1 subtracted) because the extra day is in February.

15.2 Initialising arrays

In C, it is possible to 'initialise' variables when they are declared.

It is not always a good idea to initialise variables in this way. It is most useful for creating tables of data.

For example, one can create an array giving the number of days in each month (not a leap-year) as follows

```
int month_length[12]
  = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
```

The curly braces **and the semicolon** are required.

In fact, when array initialisation is used, it is not necessary to give the size of the array:

```
int month_length[]
  = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
```

This is dangerous because the array size is not immediately obvious.

Similarly

```
char hello[] = {'H','e','l','l','o','\0'};
```

defines a character string ”Hello”. One can also write

```
char hello[] = "Hello";
or
char hello[6] = "Hello";
```

Again, one can define abbreviated names for the days of the week:

```
char * weekday[7] = {"Mon","Tue","Wed","Thu","Fri","Sat","Sun"};
```

This notation is like the `*argv[]` notation.

```
or ...
char * weekday [7] =
{"Sunday","Monday","Tuesday","Wednesday","Thursday","Friday","Saturday"};

char * month[12] =
{"January","February","March","April","May","June",
 "July","August","September","October","November","December"};

int month_len[12] =
{31,28,31,30,31,30,31,31,30,31,30,31};

int month_offset[12] =
{ 0, 3, 3, 6, 1, 4, 6, 2, 5, 0, 3, 5 }
```

15.3 Day of week code

```
#include <stdio.h>
#include <stdlib.h>

main( int argc, char * argv[] )
{
  char * weekday [7] =
  {"Sunday","Monday","Tuesday","Wednesday","Thursday",
        "Friday","Saturday"};

  int month_offset[12] =
  { 0, 3, 3, 6, 1, 4, 6, 2, 5, 0, 3, 5 };

  int dd = atoi ( argv[1] );
  int mm = atoi ( argv[2] );
  int yy = atoi ( argv[3] );
        // assumed 0 <= yy <= 99, in this century.
```

```
  int leap_year = yy % 4 == 0; // good enough for 21st century

  int wd = ( yy/4 + yy + month_offset[mm-1] + dd + 6 ) % 7;

  if ( leap_year && mm <= 2 )
    wd = (wd + 6) % 7;

  printf("%s, %d/%d/%d\n", weekday[wd], dd,mm,yy);
}
```