Computer Programming using C
Lecture 5:
Arrays and Strings

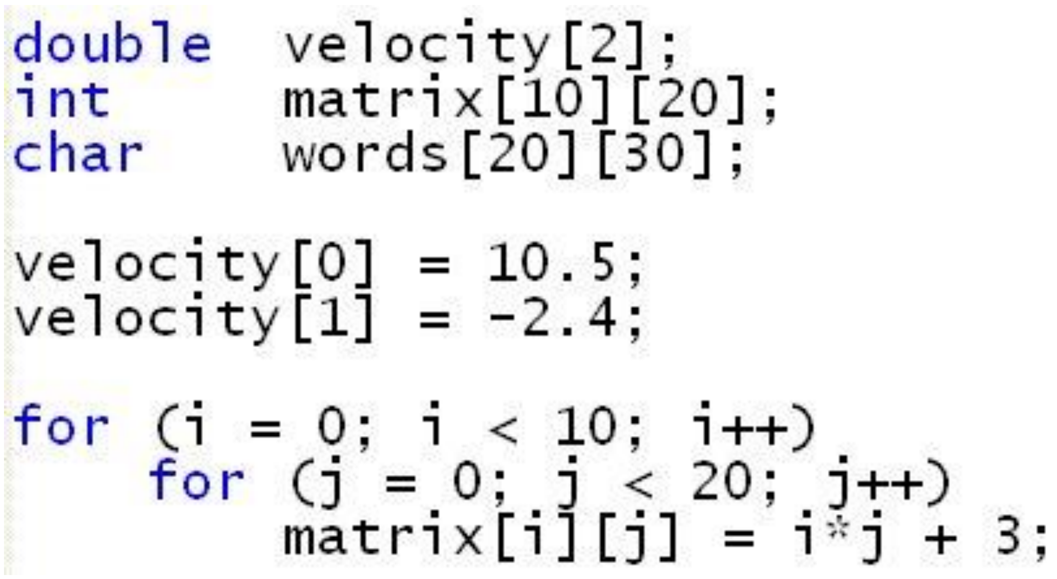
Prof. Stephen Smith

E-mail: stephen.smith@york.ac.uk

Based on lecture notes by Dr Julian Miller

Arrays

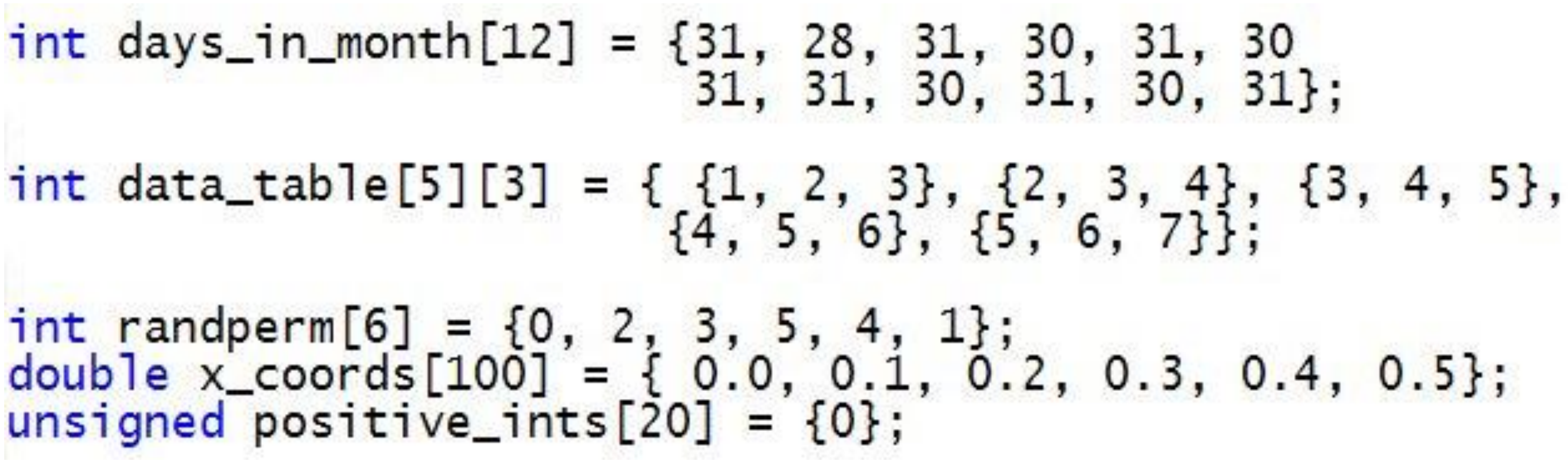
- Arrays are variables which have space for more than one value
- Arrays are assigned values by specifying the index values and assigning a value



```
double  velocity[2];
int     matrix[10][20];
char    words[20][30];

velocity[0] = 10.5;
velocity[1] = -2.4;

for (i = 0; i < 10; i++)
    for (j = 0; j < 20; j++)
        matrix[i][j] = i*j + 3;
```

Initialising arrays

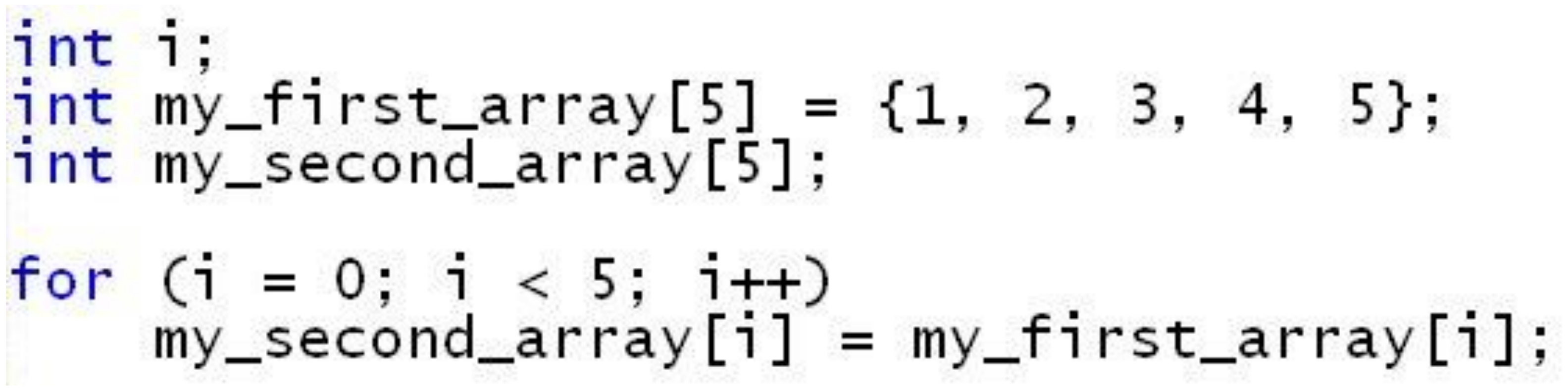
- Arrays can be initialised as in the examples below



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

int data_table[5][3] = { {1, 2, 3}, {2, 3, 4}, {3, 4, 5},
                         {4, 5, 6}, {5, 6, 7}};

int randperm[6] = {0, 2, 3, 5, 4, 1};
double x_coords[100] = { 0.0, 0.1, 0.2, 0.3, 0.4, 0.5};
unsigned positive_ints[20] = {0};
```

Copying arrays: there isn't a fast way!

- If you want to copy the contents of one array into another you have to laboriously copy each element of the first array into the other



```
int i;
int my_first_array[5] = {1, 2, 3, 4, 5};
int my_second_array[5];

for (i = 0; i < 5; i++)
    my_second_array[i] = my_first_array[i];
```

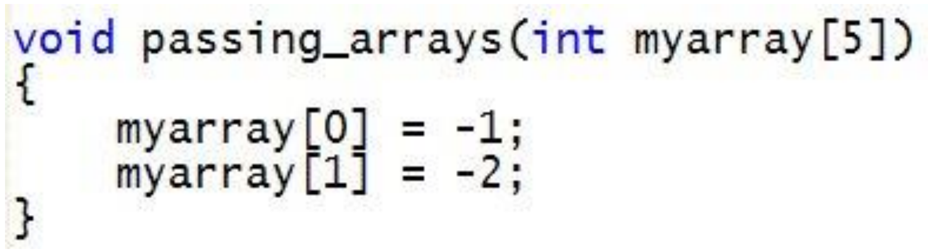
The following will **not** have the same effect – why not?

```
my_second_array = my_first_array;
```

Communicating information to and from functions

- You have found that you can
  - pass information *to* functions using function arguments
  - get information *from* functions via the *return* statement of the function.

- Array can be passed to functions as an *argument*.
  - The contents of the array can be changed inside the function and then the new contents of the array is available to the calling function (no return necessary!). E.g.



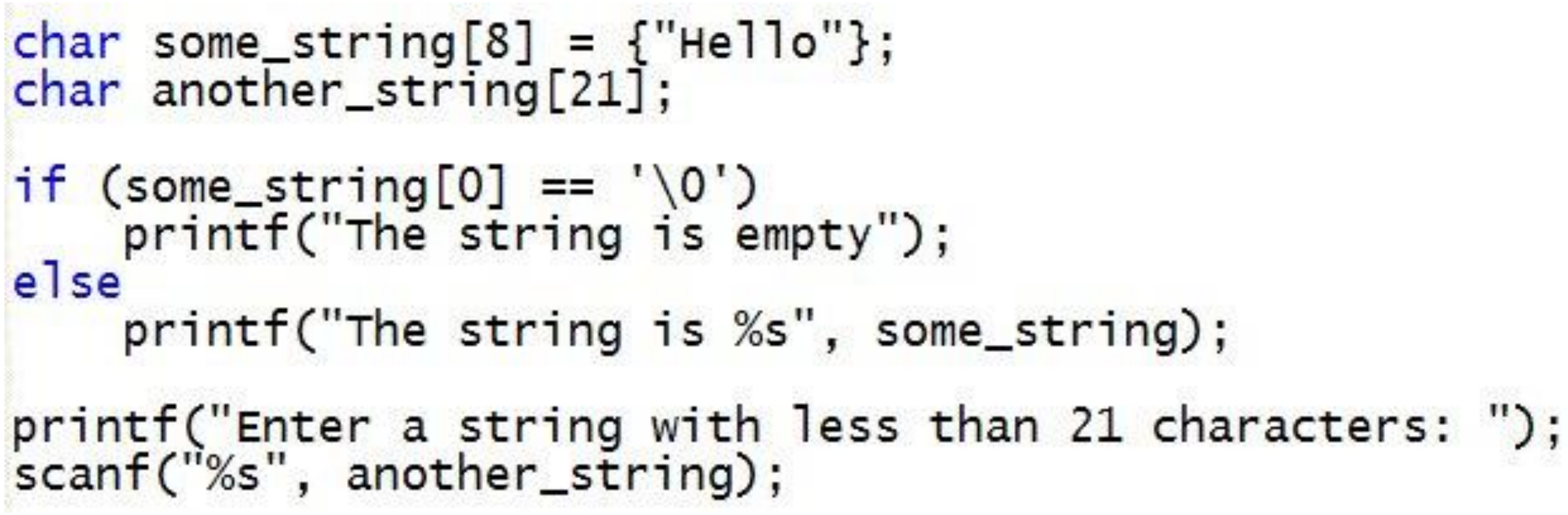
```
void passing_arrays(int myarray[5])
{
    myarray[0] = -1;
    myarray[1] = -2;
}
```

- Calling example:

```
int numbers[5] = {1000, 2, 3, 17, 50};
passing_arrays(numbers);
```

Strings

- Strings are arrays of characters
- The last element in a string is a special symbol ‘\0’.
  - This is referred to as a null terminator

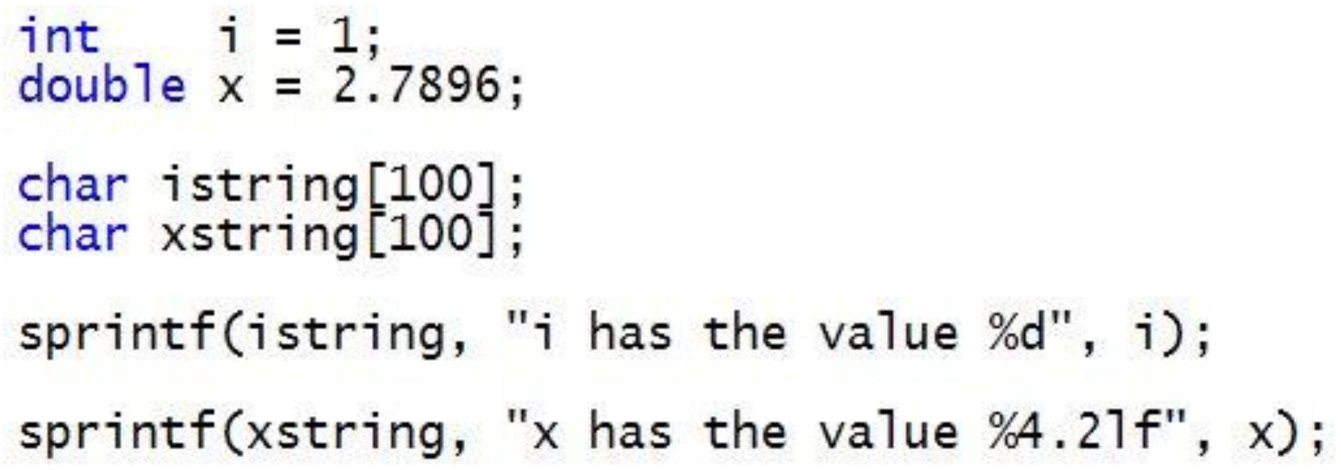


```
char some_string[8] = {"Hello"};
char another_string[21];

if (some_string[0] == '\0')
    printf("The string is empty");
else
    printf("The string is %s", some_string);

printf("Enter a string with less than 21 characters: ");
scanf("%s", another_string);
```

Using **`sprint()`**

- Convert integer entered by user into equivalent string of characters and determine the length of the resulting string using **`sprint()`** function and your own string length function.

- **`sprintf()`** works like **`printf()`** except that instead of writing the output to the screen it writes it into a string variable



```
int    i = 1;
double x = 2.7896;

char istring[100];
char xstring[100];

sprintf(istring, "i has the value %d", i);

sprintf(xstring, "x has the value %4.2lf", x);
```

String handling functions

- In **`string.h`** there are many useful string handling functions. Here are three that are very useful

  `strcpy(string1,string2);`

  - copy **`string2`** into **`string1`**

  `strcat(string1,string2);`

  - concatenate (add to then end of) **`string2`** onto **`string1`**

  `strcmp(string1,string2);`

  - If the strings are identical the value 0 is returned
  - If the first nonmatching character in string1 has a **<u>lower</u>** value than the corresponding character in string2 a **<u>negative</u>** number equal to the difference in these values is returned
  - If the first nonmatching character in string1 has a **<u>higher</u>** value than the corresponding character in string2 a **<u>positive</u>** number equal to the difference in these values is returned

Summary

- Introduced arrays
  - single and multidimensional
- Considered strings
  - arrays of characters