Computer Programming using C
Lecture 7:

Pointers and Functions

Prof. Stephen Smith

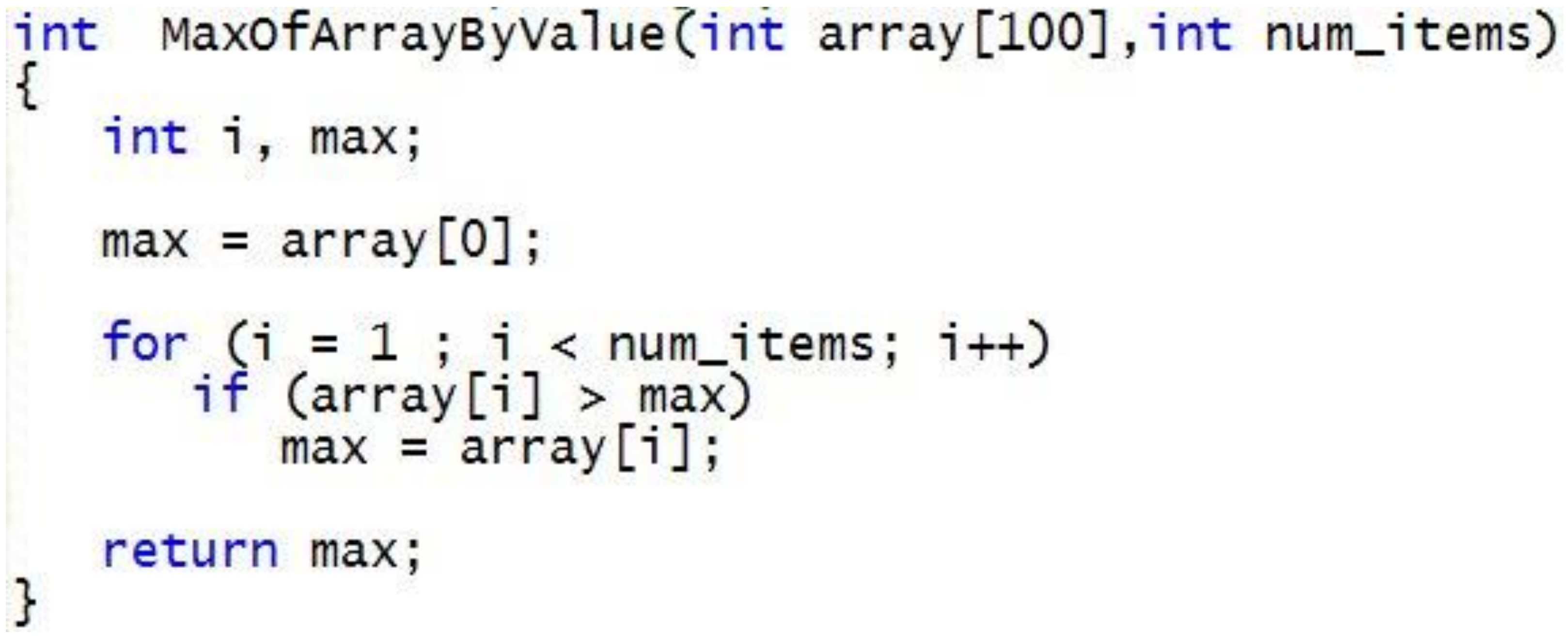
E-mail: stephen.smith@york.ac.uk

Based on lecture notes by Dr Julian Miller

Functions: Input and output of values

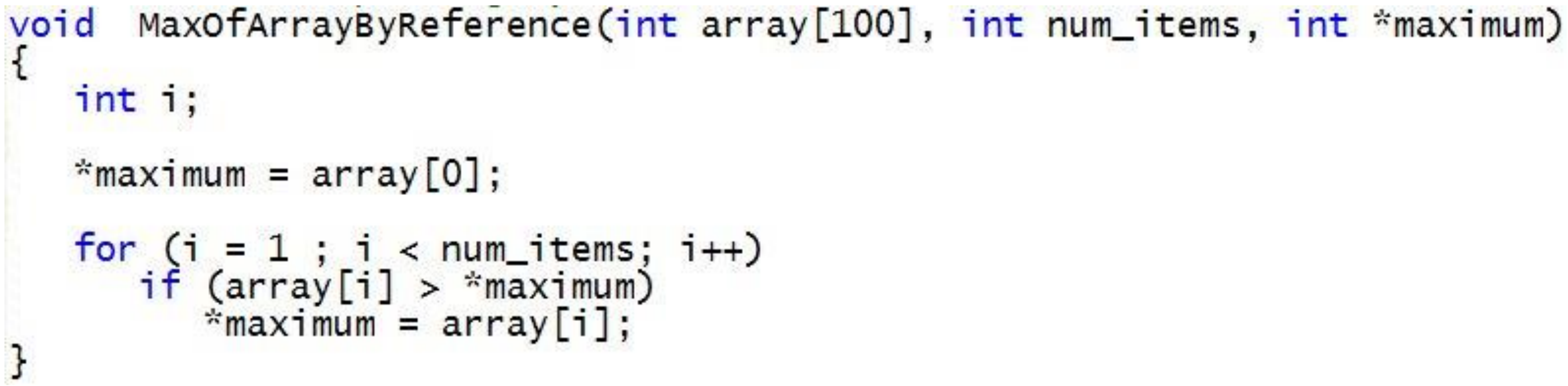
- Inputting a value to a function: passing by value
  - When a variable is passed to a function *a copy is made*
- We have seen how functions return a value
  - Using the **`return`** statement
  - Also passing by value
- When we want to input or return more than one value we have used an array
  - Passing by reference

Example: Passing by Value



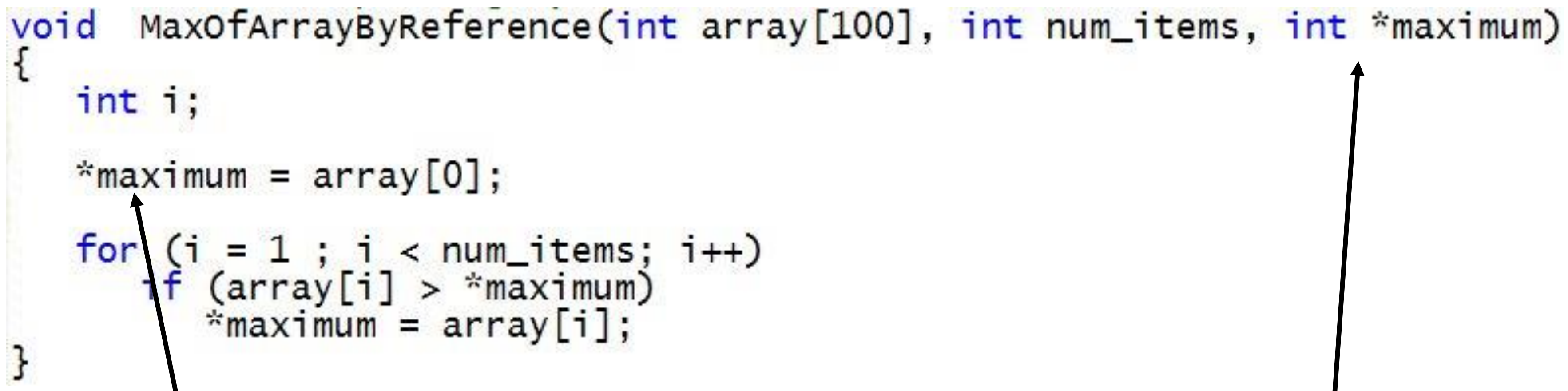
```
int  MaxOfArrayByValue(int array[100],int num_items)
{
   int i, max;

   max = array[0];

   for (i = 1 ; i < num_items; i++)
      if (array[i] > max)
         max = array[i];

   return max;
}
```

Example: Passing by Reference



```
void  MaxOfArrayByReference(int array[100], int num_items, int *maximum)
{
   int i;

   *maximum = array[0];

   for (i = 1 ; i < num_items; i++)
      if (array[i] > *maximum)
         *maximum = array[i];
}
```

Passing by Reference explained

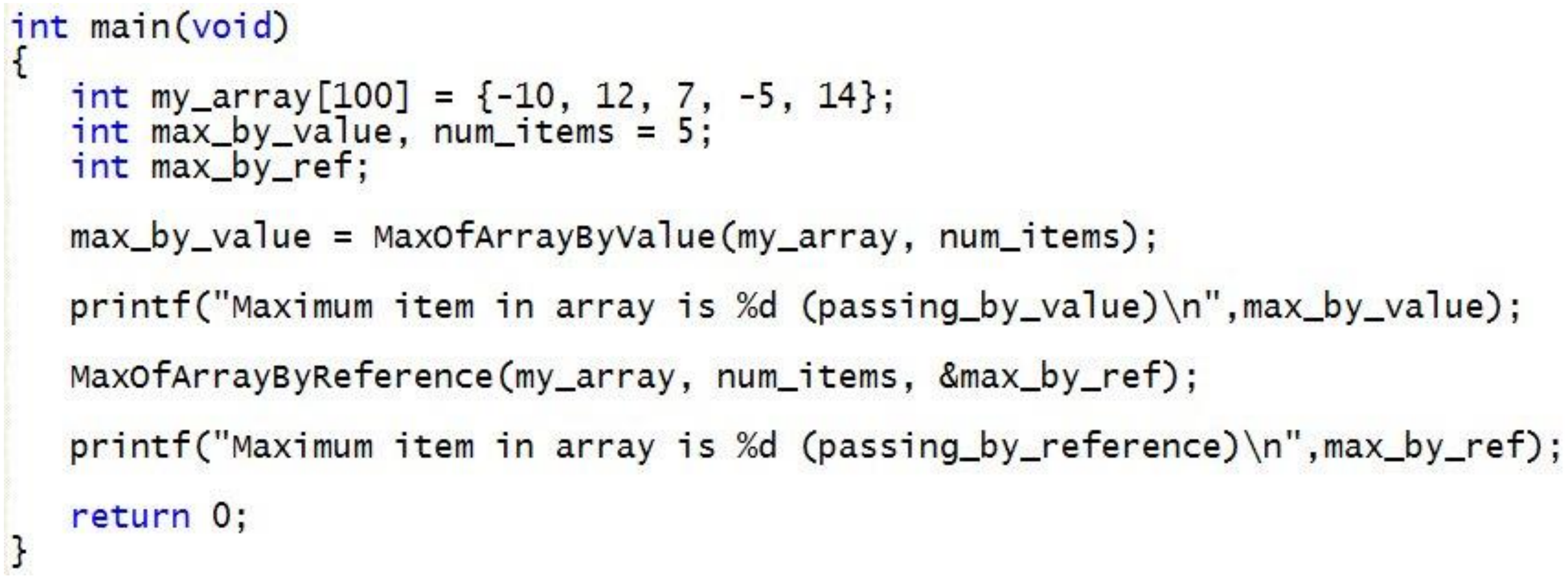


```
void  MaxOfArrayByReference(int array[100], int num_items, int *maximum)
{
    int i;

    *maximum = array[0];

    for (i = 1 ; i < num_items; i++)
        if (array[i] > *maximum)
            *maximum = array[i];
}
```

This says. Set the *contents of the variable whose address is* *maximum to …

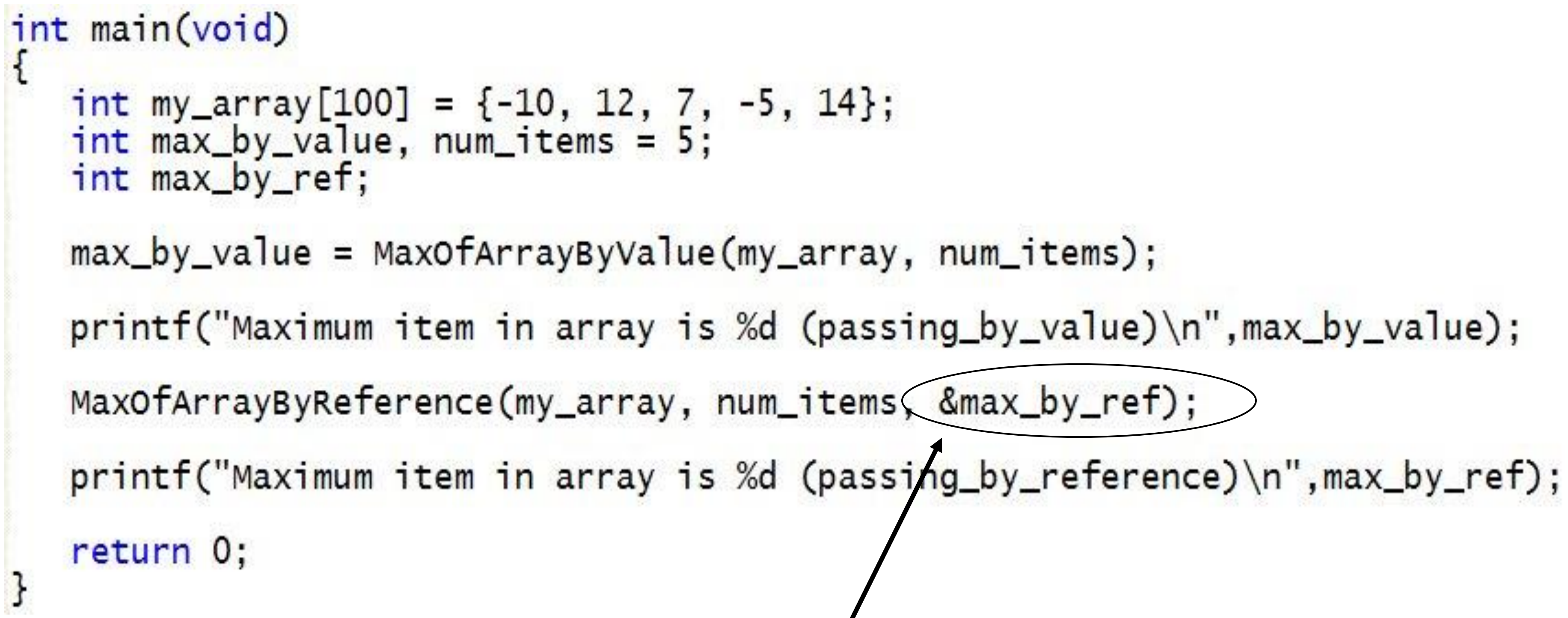
This says. An address of an integer variable called maximum is being passed in

Calling these functions



```c
int main(void)
{
   int my_array[100] = {-10, 12, 7, -5, 14};
   int max_by_value, num_items = 5;
   int max_by_ref;

   max_by_value = MaxOfArrayByValue(my_array, num_items);

   printf("Maximum item in array is %d (passing_by_value)\n",max_by_value);

   MaxOfArrayByReference(my_array, num_items, &max_by_ref);

   printf("Maximum item in array is %d (passing_by_reference)\n",max_by_ref);

   return 0;
}
```

Passing by Reference: The call



```
int main(void)
{
   int my_array[100] = {-10, 12, 7, -5, 14};
   int max_by_value, num_items = 5;
   int max_by_ref;

   max_by_value = MaxOfArrayByValue(my_array, num_items);

   printf("Maximum item in array is %d (passing_by_value)\n",max_by_value);

   MaxOfArrayByReference(my_array, num_items, &max_by_ref);

   printf("Maximum item in array is %d (passing_by_reference)\n",max_by_ref);

   return 0;
}
```

This says. Pass the address in memory of the variable max_by_ref

Pointers

- A *pointer* is a variable that holds a memory address
  - usually the address of another variable

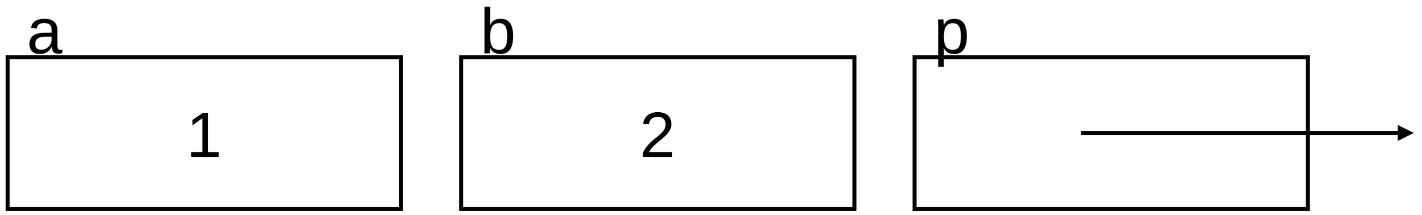
```
type *name;
```

Where

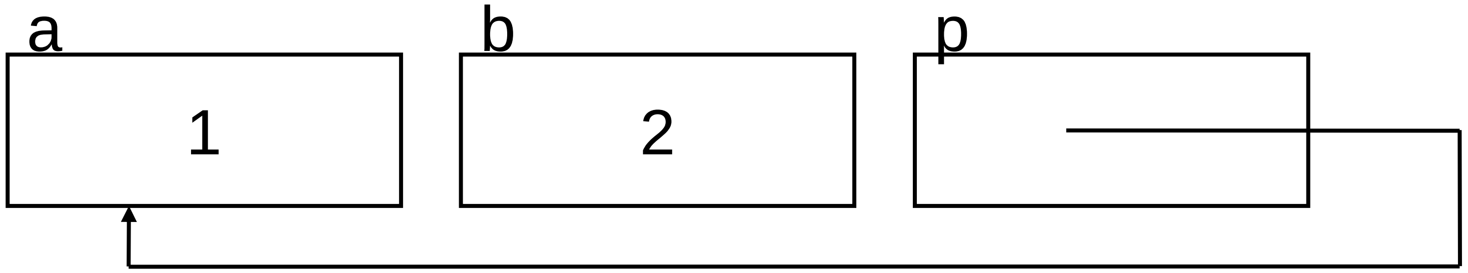
- “type” is any valid C data type
- “name” is the name of the pointer variable.
- “type” defines the data type of variables the pointer can point to

& (address-of) and * (contents-of)

- int a = 1, b = 2, *p;



- p = &a;



- b = *p;
- This is equivalent to
- b = a;

Pointers have types

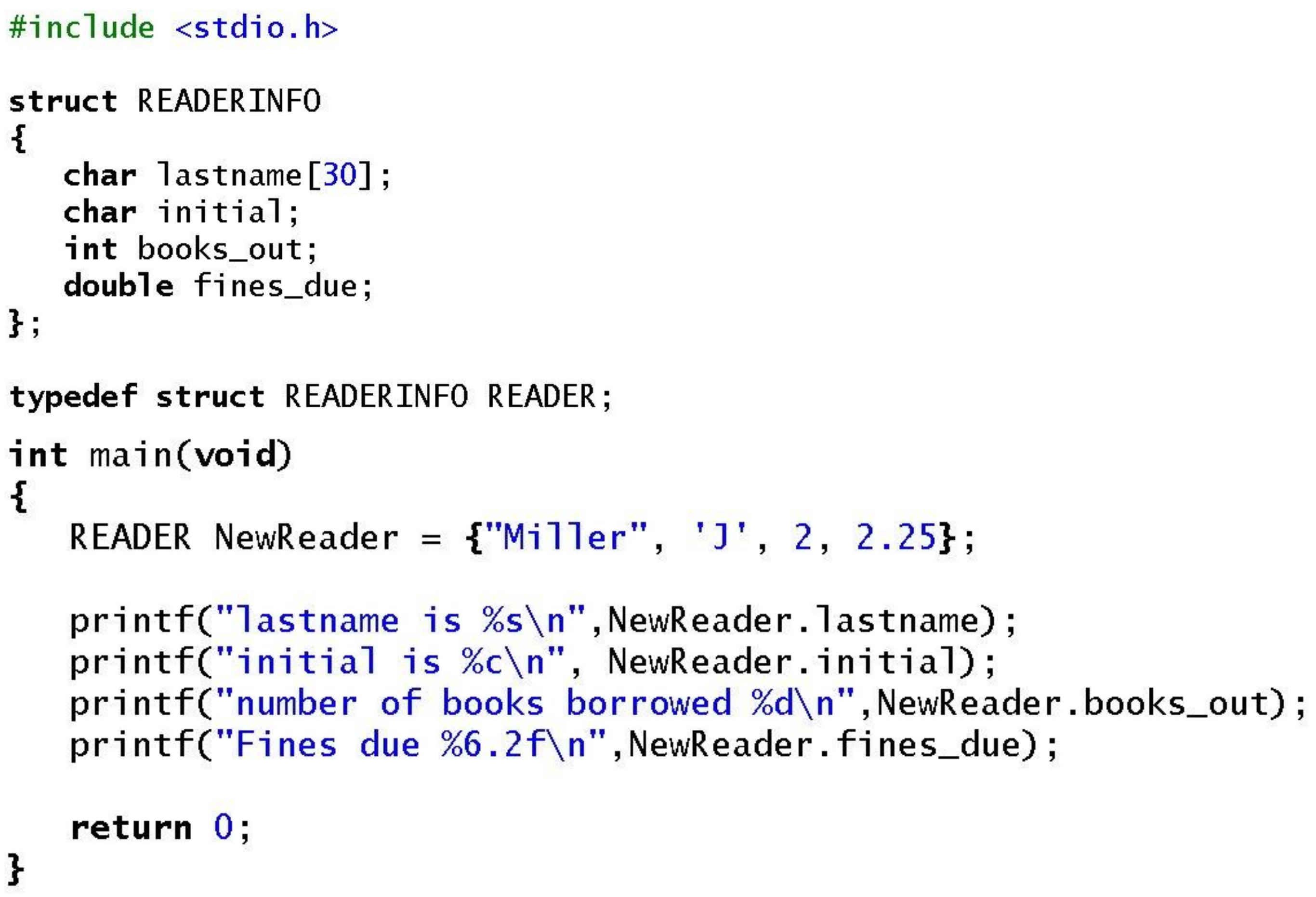
- Pointers point to specific data types

```
int *p; double *x; char *r;
```

```
int *p, fred;
char *r, c;
r  = &c;          /* This is OK */
r  = &fred;    /* This is NOT OK */
p = &fred;   /* This is OK */
```

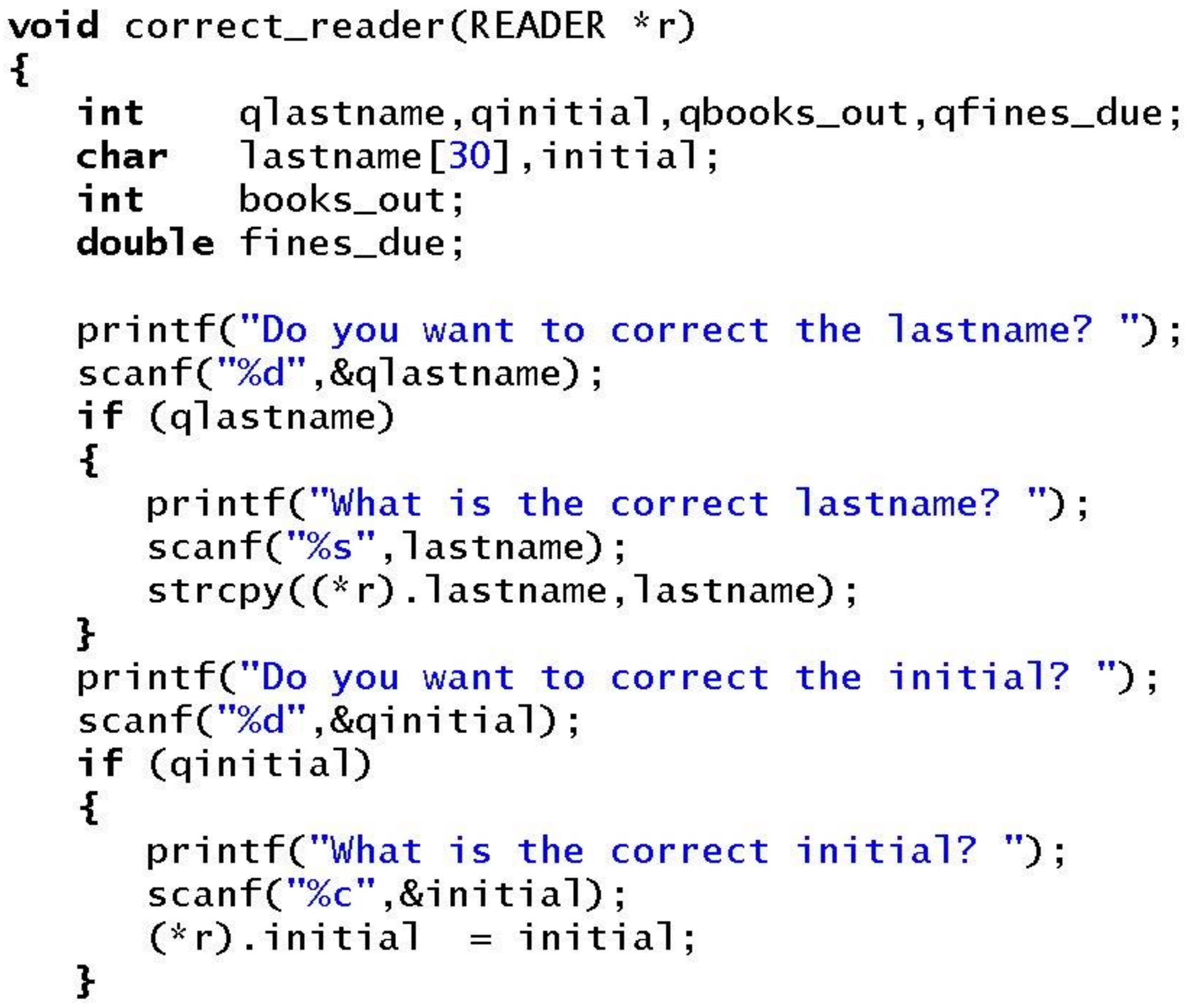
READERINFO example

- Suppose we want to write a function that allows us to correct the contents of NewReader? Using the passing-by-reference idea.



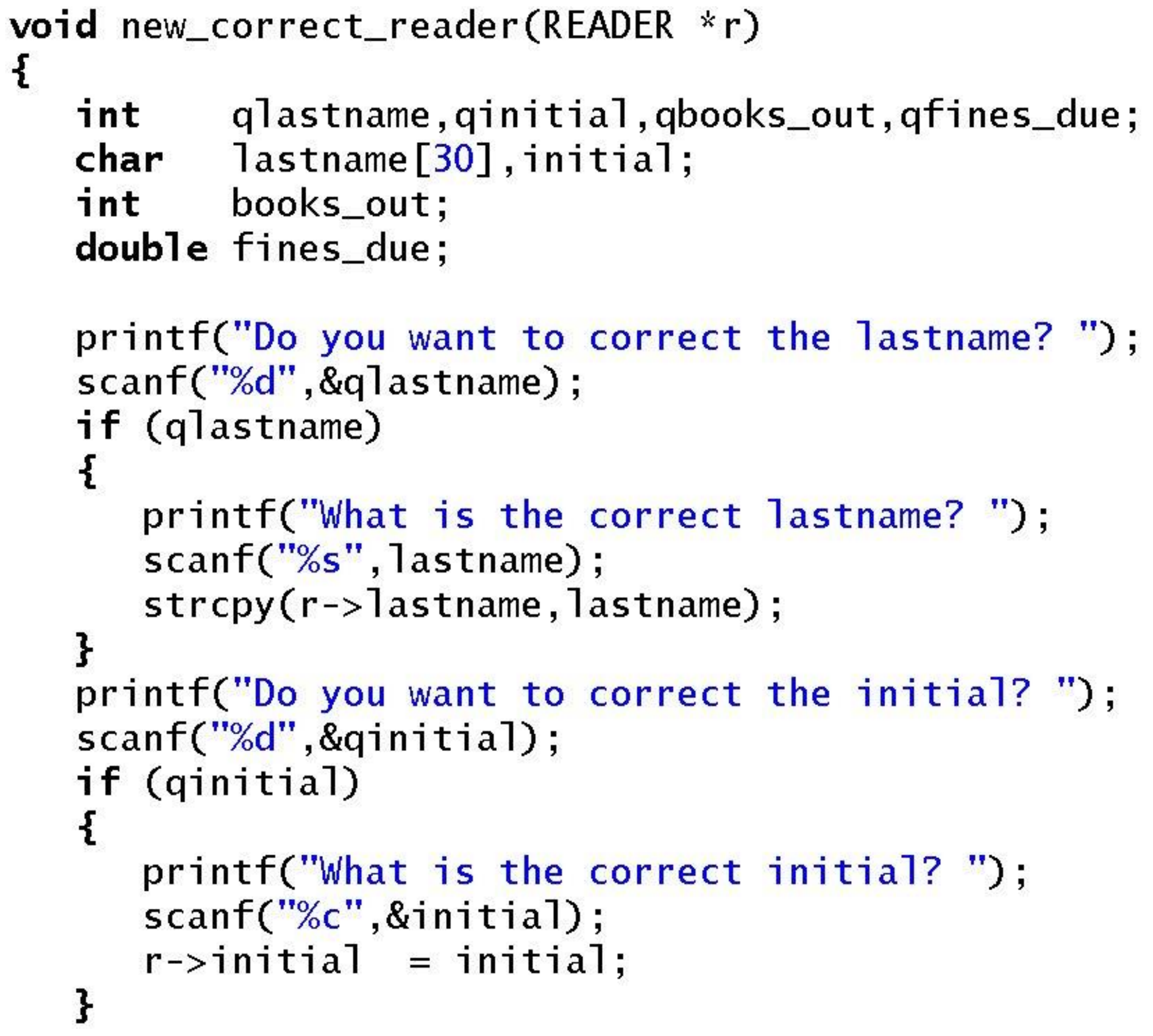
```c
#include <stdio.h>

struct READERINFO
{
   char lastname[30];
   char initial;
   int books_out;
   double fines_due;
};

typedef struct READERINFO READER;

int main(void)
{
   READER NewReader = {"Miller", 'J', 2, 2.25};

   printf("lastname is %s\n",NewReader.lastname);
   printf("initial is %c\n", NewReader.initial);
   printf("number of books borrowed %d\n",NewReader.books_out);
   printf("Fines due %6.2f\n",NewReader.fines_due);

   return 0;
}
```

Passing structures by reference



```c
void correct_reader(READER *r)
{
   int    qlastname,qinitial,qbooks_out,qfines_due;
   char   lastname[30],initial;
   int    books_out;
   double fines_due;

   printf("Do you want to correct the lastname? ");
   scanf("%d",&qlastname);
   if (qlastname)
   {
      printf("What is the correct lastname? ");
      scanf("%s",lastname);
      strcpy((*r).lastname,lastname);
   }
   printf("Do you want to correct the initial? ");
   scanf("%d",&qinitial);
   if (qinitial)
   {
      printf("What is the correct initial? ");
      scanf("%c",&initial);
      (*r).initial  = initial;
   }
```

`(*some).thing` is the same as `some -> thing` when `some` points to a structure



```c
void new_correct_reader(READER *r)
{
   int    qlastname,qinitial,qbooks_out,qfines_due;
   char   lastname[30],initial;
   int    books_out;
   double fines_due;

   printf("Do you want to correct the lastname? ");
   scanf("%d",&qlastname);
   if (qlastname)
   {
      printf("What is the correct lastname? ");
      scanf("%s",lastname);
      strcpy(r->lastname,lastname);
   }
   printf("Do you want to correct the initial? ");
   scanf("%d",&qinitial);
   if (qinitial)
   {
      printf("What is the correct initial? ");
      scanf("%c",&initial);
      r->initial  = initial;
   }
```

Summary

- Passing by reference
  - A new way of communicating with functions
  - Uses addresses of variables
  - Introduced the address-of operator &
  - Introduced the contents-of operator *
- Pointers are variables that hold the address of another variable of a particular type
- Passing structures by reference

  r → fieldname is equivalent to (*r).fieldname
- NEXT WEEK:
  - Arrays and pointers
  - Variable dimension arrays
  - Some other aspects of pointers