Lesson 7

Pointers

Class Objectives

In this lesson, you will learn:

- What pointers are
- How to declare pointers
- How to change the memory address that a pointer points at
- How to use an array of pointers

Important Notes:

- **You should type in all the programs in this handout, and run them more than once with different data**
- **You should read, and understand everything in this handout, the material in it forms the basis of the quizzes**
- **If you don't understand something; ask me to explain.**

Memory addresses

When we declare variables in a program, such as:

```
int first_number,
    second_number;
```

we are telling the compiler three things:

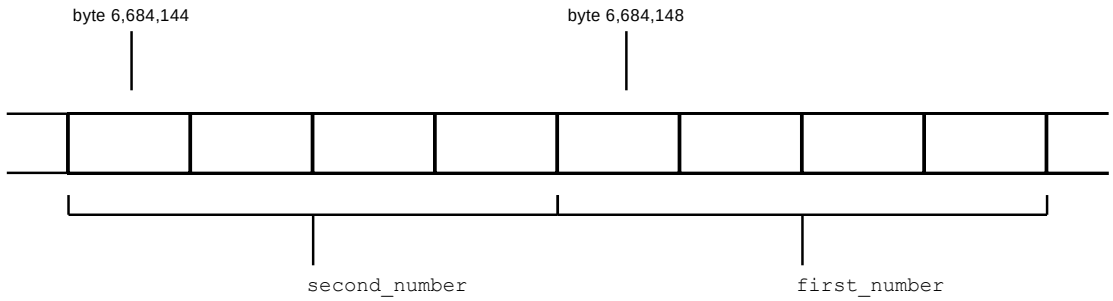
1. That the program needs two storage areas in memory
2. That these storage areas will store integers
3. And the names we will use to refer to these two storage areas

The “address of” (&) operator

```
scanf("%d”,&first_number);
```

Printing the “values of”, and the “addresses of”

```
printf("\n The first value, is: %d \n",first_number);

printf("\n The first address, is: %d \n",&first_number);
```



Program (point1.c)

```
/*
  Program "point1.c"
  written by: Joe Dorward
  Date: 05/03/00

  This program prints the value of two integer variables,
  and their address in memory
*/

#include <stdio.h>

void main(void)
{
int first_number = 3,
    second_number = 7;


  /* print the vaues of the variables */
  printf("\n The first value, is: %d \n",first_number);
  printf(" The second value, is: %d \n\n",second_number);


  /* print the memeory addresses of the variables */
  printf("\n The first address, is: %d \n",&first_number);
  printf(" The second address, is: %d \n\n",&second_number);
}
```

If you want to know how many bytes the computer uses to store an integer, you can subtract the lowest address from the highest address. You should get a difference of 4 bytes.

Pointers, and addresses

When we looked at files, we used the line of code:

```
FILE *pointer_to_file;
```

That line, has three important parts:

1. The variable name `pointer_to_file`
2. The “points to” (`*`) operator
3. The the kind of thing (its base type) the pointer can point to: `FILE`

So what is a pointer?

Pointers are variables that store the memory address of data items.

The base type of a pointer doesn’t tell you the kind of value that it can store, only the type of the variable it can point to.

We can declare an integer with:

```
int first_number;
```

and a pointer to an integer with:

```
int *pointer_to_integers;
```

Pointers; accessing addresses, and values

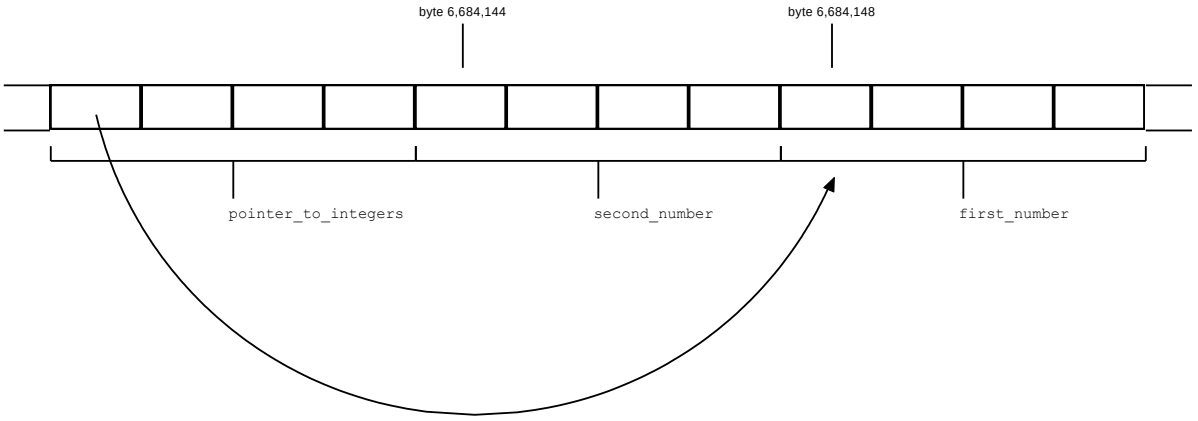
When using pointers, it’s important that you remember:

- That the “address of” (`&`) operator is used when you refer to the memory addresses of variables
- That only the variable name is used when you refer to the value stored in that variable
- When you refer to the value that a pointer points to, you must use the “points to” (`*`) operator

Program (point2.c)

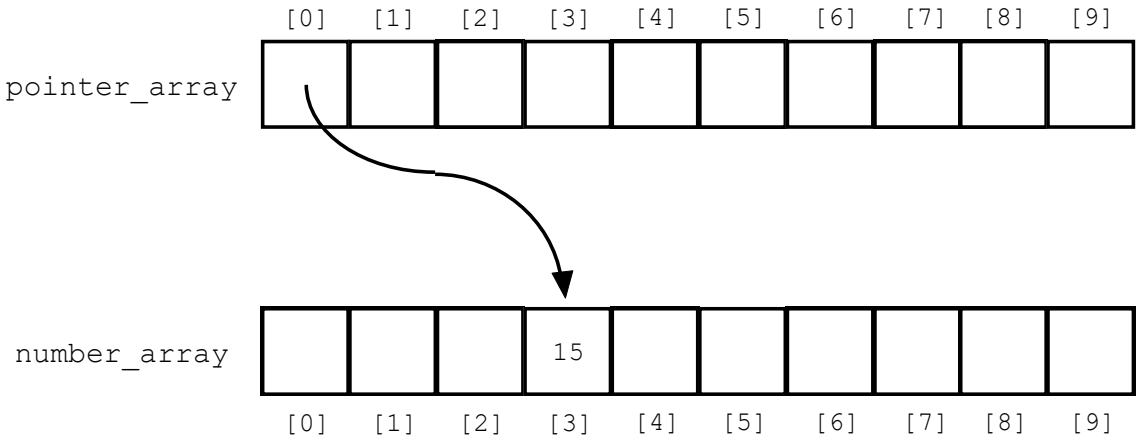
```
/*
  Program "point2.c"
  written by: Joe Dorward
  Date: 05/03/00

  This program prints the value of two integer variables,
  and their address in memory
*/

#include <stdio.h>

void main(void)
{
int first_number = 3,
    second_number = 7,
    *pointer_to_integers;


  /* print the vaues of the variables */
  printf("\n The first value, is: %d \n",first_number);
  printf(" The second value, is: %d \n\n",second_number);

  /* assign the address of first number to the pointer */
  pointer_to_integers = &first_number;

  /* print the memory addresses of the variables */
  printf("\n The first address, is: %d \n",&first_number);
  printf(" The second address, is: %d \n\n",&second_number);

  /* print out the value pointed to by the pointer */
  printf("\n The value pointed to, is: %d \n",*pointer_to_integers);

  /* print out the address pointed to by the pointer */
  printf(" The address pointed to, is: %d \n\n",pointer_to_integers);
}
```

A pictorial representation of how this program uses memory is below.



Here you can see that both the variable name `first_number`, and the pointer `pointer_to_integers` refer to the four bytes beginning at byte 6,684,148.

This next program, gets a little more complicated. This program uses both an array of integers, and an array of integer pointers. A pictorial representation of this program is below, and shows you what happens if you enter 15 for the integer value, and 3 for the array element.



It is the line of code:

```
pointer_array[0] = &number_array[array_element];
```

that sets the pointer in element 0 of the pointer array, to point to element 3 of the integer array.

Program (point3.c)

```
/*
  Program point3.c
  written by: Joe Dorward
  Date: 05/04/00

  This program uses an array of integers to store values, and uses
  an array of pointers to keep track of where those values are stored
*/

#include <stdio.h>

void main(void)
{
int the_number = 0,       /* an integer variable */
    array_element,        /* for indicating the array element */
    number_array[10],     /* an integer array */
    *pointer_array[10];   /* an array of pointers to integers */

  /* ask for an integer value */
  printf(" Please enter an integer: ");

  /* put the value into the variable "the_number" */
  scanf("%d",&the_number);

  /* ask for the array element */
  printf(" Which array element should I put it in (0 - 9): ");

  /* put the value into the variable "array_element" */
  scanf("%d",&array_element);

  /* put the value in "the_number" into the array element */
  number_array[array_element] = the_number;

  /* set the first pointer in the pointer array,
     to point to that element in the integer array */
  pointer_array[0] = &number_array[array_element];

/* ========================================================================= */

  /* print to the screen, the value stored in the integer array */
  printf("\n The value stored in the integer array at element: ");
  printf("%d",array_element);
  printf(", is the value: %d",number_array[array_element]);

  /* print to the screen, the value that the first pointer in the
     pointer array points to */
  printf("\n The value pointed to, by the first pointer in the ");
  printf("pointer array is: %d \n\n",*pointer_array[0]);

}
```