Lesson 5

Arrays 1

Class Objectives

In this lesson, you will learn how to:

- Declare arrays of the base types: `char`, `int`, `float`
- Declare an array of character arrays (strings)
- Read values into the elements of arrays
- Write values from the elements of arrays
- How to declare an array of structures

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 it again**

Arrays

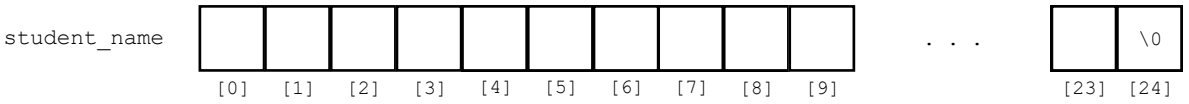
An array is a set of data items, referred to as *elements*, and each of those data items has the same base type, (`int`, `char`, `float` etc.).

The general form of an array declaration is:

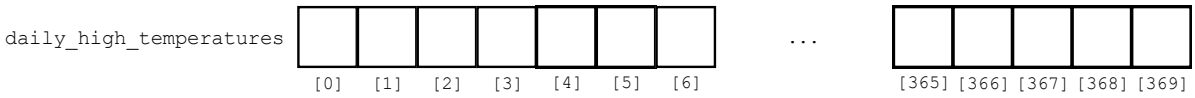
```
<base type> <variable name>[number of elements];
```

Below are example array declarations of base type: `char`, `int`, `float`, with a pictorial representation of each.

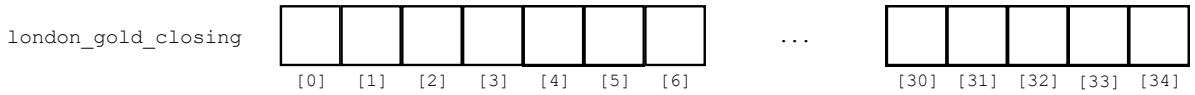
```
char student_name[25];
```



```
int daily_high_temperatures[370];
```



```
float london_gold_closing[35];
```



As you can see from the pictorial representations above, the first element of each array is element[0], and the last element of each array is element[size - 1].

The standard way to access the elements of an array sequentially, is to use a counter variable, in a loop.

Program (temps1.c)

This program is based on the integer array:

```
int daily_high_temperatures[370];
```

```
/*
  Program temps1.c
  written by: Joe Dorward
  Date: 04/09/00

  This program has 7 integers, assigned into a 10 element array.

  It then uses a for() loop to run through the array, and
  print out those values.
*/

#include <stdio.h>

void main(void)
{
int element_number = 0,
    daily_high_temperatures[10] = {65, 64, 67, 68, 70, 75, 72};

  //=====================================================================
  // print out the values in the array

  for (element_number = 0; element_number < 7; element_number++)
  {
    printf("The high temperature for day: %d ",element_number + 1);
    printf("is: %d \n",daily_high_temperatures[element_number]);
  }
}
```

Program (gold1.c)

This program is based around the float array:

```
float london_gold_closing[35];
```

```
/*
  Program gold1.c
  written by: Joe Dorward
  Date: 04/09/00

  This program assigns values into an array for the daily price
  of gold in London.

  Then it asks for a price from the user, and it tells the user
  how many days the price of gold closed in London higher than that.
*/

#include <stdio.h>

void main(void)
{
int element_number = 0,   // loop counter
    day_counter = 0;   // counts the number of days

float entered_price,
      london_gold_closing[35] =

                {288.34f, 289.40f, 292.90f, 289.90f, 288.50f, 0.0f, 0.0f,
                 288.80f, 289.70f, 291.20f, 291.20f, 290.50f, 0.0f, 0.0f,
                 290.25f, 290.65f, 288.75f, 289.15f, 288.50f, 0.0f, 0.0f };

  // ================================================================
  // Ask for a price, and report

  printf(" Please enter the bottom price: ");
  scanf("%f",&entered_price);

  for (element_number = 0 ; element_number <= 20 ; element_number++)
  {
    if (london_gold_closing[element_number] > entered_price)
    {
      day_counter++;
    }
  }

  printf("\n London gold closed above: $%3.2f,",entered_price);
  printf(" %d days this month. \n\n",day_counter);
}
```

Arrays of character arrays

This program uses an array, of character arrays (strings) to store the names of students.

That is a two dimensional grid 11 elements high, by 15 characters wide. A pictorial representation is shown below.

	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]	[12]	[13]	[14]
[0]	E	d	w	a	r	d		H	o	p	p	e	r		
[1]	J	i	m		S	m	i	t	h						
[2]	F	r	a	n	k		J	o	n	e	s				
[3]	J	e	a	n		G	r	a	y						
[4]	P	a	u	l		B	r	o	w	n					
[5]															
[6]															
[7]															
[8]															
[9]															
[10]															

Each row of this array holds a string.

To put a string into this array, we need to use the string copy function: `strcpy()`, just as in:

```
strcpy( name_array[students] , student_name );
```

Where:

`name_array` is the name of the array
`students` is the integer variable holding a value 0 - 11
`student_name` is the string holding a name

Program (student6.c)

```
/*
  Program "student6.c"
  Written by: Joe Dorward
  Date: 03/12/00

  Asks for student names, and puts them in an array
  then run through array printing them out
*/

#include <stdio.h>
#include <string.h>

void main(void)
{
char student_name[15] = "",
     name_array[11][15] = {""};   // an array of character arrays

int students = 0;   // loop counter variable

  printf("Please enter student\'s name: ");
  gets(student_name);

  // ask for, and put names into array
  while(strcmp(student_name,"quit") != 0)   // while not the same
  {
    strcpy( name_array[students] , student_name );
    students++;

    printf("Please enter student\'s name: ");
    gets(student_name);
  }

  printf("\n");

  // ============================================================

  // print out names from array
  for (students = 0; students < 11; students++)
  {
    printf("%2d. %s \n",(students + 1),name_array[students]);
  }
  printf("\n");
}
```

Defining, and declaring structures

We have already seen how to define, and declare a structures. We *defined* a structure in the program `student4.c` with the lines:

```
// define a data structure
struct a_student_record
{
  char student_name_field[25];
  char student_address_field[50];
  char student_date_of_birth_field[10];
  char student_program_field[15];
};
```

Then we *declared* a variable of that type, with the line:

```
// declare a data structure
struct a_student_record first_student_record;
```

But that line only allocates enough memory to store the details of one student. To allocate memory to store the details of two students would require adding another variable of the same type:

```
// declare a data structure
struct a_student_record first_student_record,
                        second_student_record;
```

That's clearly not a great solution for say 300 students. The solution, is to use Arrays of structures.

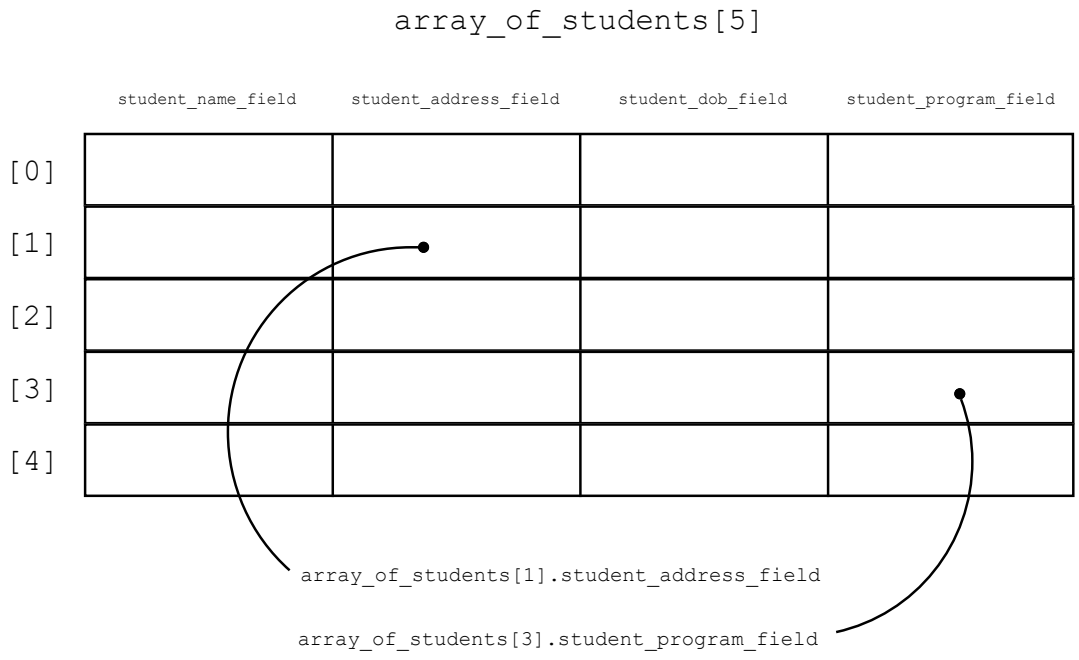
Declaring an array of structures

Declaring an array of the structure type `a_student_record` requires only a variable name, square brackets, and the number of structures to be in the array.

If we wanted an array to hold the details of 5 students, the declaration would be:

```
struct a_student_record array_of_students[5];
```

This could be represented pictorially by:



NOTE:
That referring to any field in the array requires the use of:

- The array name
- The array element number
- The field name

Program (student5.c)

```
/*
  Program "student5.c"
  Written by: Joe Dorward
  Date: 03/11/00

  A small student details database program using an array of structures
*/

#include <stdio.h>
#include <string.h>

void main(void)
{
int entry_number = 0;

char temp_name[25] = "";
char temp_address[50] = "";
char temp_date_of_birth[10] = "";
char temp_program[15] = "";

struct a_student_record   // define the structure
{
  char student_name_field[25];
  char student_address_field[50];
  char student_date_of_birth_field[10];
  char student_program_field[15];
};

struct a_student_record array_of_students[5];   // Declare the array

  // ============================================================
  // get student details
  printf("\n");

  while (entry_number < 5)
  {
    printf(" Enter the student\'s name: ");
    gets(temp_name);
    strcpy(array_of_students[entry_number].student_name_field,temp_name);

    entry_number++;   // increment entry_number
  }
  // ============================================================================
  // display all entries

  for (entry_number = 0; entry_number < 5; entry_number++)
  {
    printf("\n Student %d: is: ",entry_number + 1);
    printf("%s, ",array_of_students[entry_number].student_name_field);
  }
  // ============================================================================
  // Exit program

  printf("\n\n ** Exiting Program ** \n\n");
}
```