

# 4<sup>th</sup> C Class with Mr. Czik

## Arrays continued

Declare and initialize the array (how to get the values into the array in the first place).

Can enter individually - `int a[0]=0; int a[1]=5;` etc.

Can enter all on one line of C code -

```
int a[10] = {0, 5, 10, 15, 20, 25, 30, 35, 40, 45};
```

0	1	2	3	4	5	6	7	8	9
0	5	10	15	20	25	30	35	40	45

## Functions

Sample function `multbytwo` – multiplies the number passed in to it by two, and passes it back again. Naming a function descriptively helps to make it easy to use and document.

```
int multbytwo(int);           declaration
main ( )                     every C program has a main routine!
{
    -----
    -----
    -----
    y = multbytwo(x);         y will be set equal to returnval
}
```

```
-----
-----
-----
-----
int multbytwo(int x);         passes out int variable, passes in int x
{
    int returnval;           declaration
    returnval = x * 2;        doubles x value
    return returnval;        return statement sends back the value
}
```

## 4<sup>th</sup> C Class with Mr. Czik

The function can be defined elsewhere in the same program, or it can be "included" from separate file of C code. This can be a file that you wrote yourself, or a file of pre-programmed functions coded by someone else.

Example: `#include <piclib.h>` include the PIC function library

Most of our programming will be in writing our own functions. If we were following a line, we could use a function that we named "linefollow". If we lost the line, we could call a function that we named "linefind". Once the line was found, we could call "linefollow" again, etc.

If statements and loops can be nested (placed inside one another), and so can functions. One function can call another function, or many other functions. We will avoid having functions call themselves (recursion), even though it can be a powerful tool, because it is also a powerful way to mess up.

### Pointers

Every location in memory has a unique address. Just as every building has its own unique address - country, state, city, street, street number.

`int i = 5;` declares i as an integer, assigns value of 5 to the variable i

var											i						
val		3				3				3	<b>5</b>						
addr		0				0				0	0						
		x				x				x	x						
		1				4				5	7						
		F				F				F	F						
		6				7				F	0						

`int x, *ip;` declares x as an integer and ip as a pointer

var				x							i				ip		
val		3				3				3	<b>5</b>						
addr		0		0		0				0	0				0		
		x		x		x				x	x				x		
		1		4		4				5	7				D		
		F		3		F				F	F				O		
		6		7		7				F	0				F		

## 4<sup>th</sup> C Class with Mr. Czik

`ip = &i;`                      assigns `ip` to hold the address of the variable `i`

`*ip = 7;`                      assigns the value `7` to the address that `ip` points to (`i`)

var			x						i				ip			
val		3	<b>8</b>		3			3	<b>7</b>				<small>7F0</small>			
addr		0	0		0			0	0				0			
		x	x		x			x	x				x			
		1	4		4			5	7				D			
		F	3		F			F	F				O			
		6	7		7			F	0				F			

`i = 8;`                              assigns the value of `8` to `i`

var			x						i				ip			
val		3			3			3	<b>8</b>				<small>7F0</small>			
addr		0	0		0			0	0				0			
		x	x		x			x	x				x			
		1	4		4			5	7				D			
		F	3		F			F	F				O			
		6	7		7			F	0				F			

`x = *ip;`                      assigns the value that `ip` points to (`i`) to `x`  
 an **indirect** reference

var			x						i				ip			
val		3	<b>8</b>		3			3	<b>8</b>				<small>7F0</small>			
addr		0	0		0			0	0				0			
		x	x		x			x	x				x			
		1	4		4			5	7				D			
		F	3		F			F	F				O			
		6	7		7			F	0				F			

`int a[5] = {2,4,6,8,10};`

`int x;`

`int *aptr;`

0	1	2	3	4
2	4	6	8	10

## 4<sup>th</sup> C Class with Mr. Czik

`aptr = &a[3];` points to the location of array entry `a[3]`  
`x = *aptr;` assigns `x` the value of the location `aptr` points to (8)  
`aptr ++;` increments to next memory location  
`x = *aptr;` assigns `x` the value of the next location (10)  
`*aptr = 12;` now `a[4]=12` (indirect reference)

0	1	2	3	4
2	4	6	8	12

`aptr ++;` increments to next memory location  
`x = *aptr;` assigns `x` the value of the next location  
this goes beyond the defined array, and could be anything (program statement, data, operating system, ?) **buffer overflow**  
`*aptr = 56;` **if we assigned a value to this location, we do not know what would happen!**