C programming with Microcontrollers

C programming with microcontrollers can be divided into two parts. The first part is the standard C programming methods that can be learned or reviewed in the following references:

Essential C http://cslibrary.stanford.edu/101/EssentialC.pdf
(Good free C introduction)

Embedded C Programming and the Atmel AVR, Barnett, Cox and O'Cull, Thomson
(Best AVR programming book, but it is very expensive)

The above references cover the basic syntax in C, for loops, while loops, and if then else statements.

The second part of C programming with microcontrollers is the programming methods necessary to interact with the microcontroller. The following very expensive reference covers this topic:

Embedded C Programming and the Atmel AVR, Barnett, Cox and O'Cull, Thomson
(Best AVR programming book, but it is very expensive)

Communication with the microcontroller is accomplished by interacting with registers and ports on the microcontroller. These registers and ports are defined for each microcontroller in an '.h' file that is included automatically by the AVR Studio when you specify the particular chip you are using. The registers and ports are distinguished from other C variables by being all capitals.

For example in the Blink2 program below The F_CPU variable sets the clock speed of the processor. The DDRC register is the Data Direction Register for port C. The data direction register defines whether each pin on a 8 bit port is either an input (0) or an output (1). The register is usually specified by a binary constant and the ports are numbered from right to left (pin 0 to pin 7) in the binary constant. The variable PORT refers to port c on the microcontroller and the defined sbi and cbi macros set and cleat a pin on the specified port.

```c
// Blink Version 2
#include <avr/io.h>
#define F_CPU 1000000UL // 1 MHz CPU Clock Frequency
#include <util/delay.h> // Include the built in Delay Function

#define sbi(var, mask) ((var) |= (uint8_t)(1 << mask))
#define cbi(var, mask) ((var) &= (uint8_t)~(1 << mask))

int main (void)
{
	//1 = output, 0 = input
	DDRC = 0b00100000; // Set Pin 5 as Output
	while(1)
	{
		sbi(PORTC, 5); // Set pin 5 on PORTC
		_delay_ms(500); // Delay 500 ms
		cbi(PORTC, 5); // Clear pin 5 on PORTC
		_delay_ms(500); // Delay 500 ms
	}
	return(0);
}
```