```
/*
 Print.cpp - Base class that provides print() and println()
 Copyright (c) 2008 David A. Mellis.  All right reserved.

 This library is free software; you can redistribute it and/or
 modify it under the terms of the GNU Lesser General Public
 License as published by the Free Software Foundation; either
 version 2.1 of the License, or (at your option) any later version.

 This library is distributed in the hope that it will be useful,
 but WITHOUT ANY WARRANTY; without even the implied warranty of
 MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
 Lesser General Public License for more details.

 You should have received a copy of the GNU Lesser General Public
 License along with this library; if not, write to the Free Software
 Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA  02110-1301  USA

 Modified 23 November 2006 by David A. Mellis
 */

#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <math.h>
#include "Arduino.h"

#include "Print.h"

// Public Methods //////////////////////////////////////////////////////////////

/* default implementation: may be overridden */
size_t Print::write(const uint8_t *buffer, size_t size)
{
  size_t n = 0;
  while (size--) {
    n += write(*buffer++);
  }
  return n;
}

size_t Print::print(const __FlashStringHelper *ifsh)
{
  PGM_P p = reinterpret_cast<PGM_P>(ifsh);
  size_t n = 0;
  while (1) {
    unsigned char c = pgm_read_byte(p++);
    if (c == 0) break;
    n += write(c);
  }
  return n;
}

size_t Print::print(const String &s)
{
  return write(s.c_str(), s.length());
}

size_t Print::print(const char str[])
{
  return write(str);
}

size_t Print::print(char c)
{
  return write(c);
}
```

```
size_t Print::print(unsigned char b, int base)
{
  return print((unsigned long) b, base);
}

size_t Print::print(int n, int base)
{
  return print((long) n, base);
}

size_t Print::print(unsigned int n, int base)
{
  return print((unsigned long) n, base);
}

size_t Print::print(long n, int base)
{
  if (base == 0) {
    return write(n);
  } else if (base == 10) {
    if (n < 0) {
      int t = print('-');
      n = -n;
      return printNumber(n, 10) + t;
    }
    return printNumber(n, 10);
  } else {
    return printNumber(n, base);
  }
}

size_t Print::print(unsigned long n, int base)
{
  if (base == 0) return write(n);
  else return printNumber(n, base);
}

size_t Print::print(double n, int digits)
{
  return printFloat(n, digits);
}

size_t Print::println(const __FlashStringHelper *ifsh)
{
  size_t n = print(ifsh);
  n += println();
  return n;
}

size_t Print::print(const Printable& x)
{
  return x.printTo(*this);
}

size_t Print::println(void)
{
  size_t n = print('\r');
  n += print('\n');
  return n;
}

size_t Print::println(const String &s)
{
  size_t n = print(s);
  n += println();
  return n;
}

size_t Print::println(const char c[])
{
  size_t n = print(c);
  n += println();
  return n;
}

size_t Print::println(char c)
{
  size_t n = print(c);
  n += println();
  return n;
}

size_t Print::println(unsigned char b, int base)
{
  size_t n = print(b, base);
  n += println();
  return n;
}

size_t Print::println(int num, int base)
{
  size_t n = print(num, base);
  n += println();
  return n;
}

size_t Print::println(unsigned int num, int base)
{
  size_t n = print(num, base);
  n += println();
  return n;
}

size_t Print::println(long num, int base)
{
  size_t n = print(num, base);
  n += println();
  return n;
}

size_t Print::println(unsigned long num, int base)
{
  size_t n = print(num, base);
  n += println();
  return n;
}

size_t Print::println(double num, int digits)
{
  size_t n = print(num, digits);
  n += println();
  return n;
}

size_t Print::println(const Printable& x)
{
  size_t n = print(x);
  n += println();
  return n;
}

// Private Methods /////////////////////////////////////////////////////////////
```

```
size_t Print::printNumber(unsigned long n, uint8_t base) {
  char buf[8 * sizeof(long) + 1]; // Assumes 8-bit chars plus zero byte.
  char *str = &buf[sizeof(buf) - 1];

  *str = '\0';

  // prevent crash if called with base == 1
  if (base < 2) base = 10;

  do {
    unsigned long m = n;
    n /= base;
    char c = m - base * n;
    *--str = c < 10 ? c + '0' : c + 'A' - 10;
  } while(n);

  return write(str);
}

size_t Print::printFloat(double number, uint8_t digits)
{
  size_t n = 0;

  if (isnan(number)) return print("nan");
  if (isinf(number)) return print("inf");
  if (number > 4294967040.0) return print ("ovf");  // constant determined empirically
  if (number <-4294967040.0) return print ("ovf");  // constant determined empirically

  // Handle negative numbers
  if (number < 0.0)
  {
     n += print('-');
     number = -number;
  }

  // Round correctly so that print(1.999, 2) prints as "2.00"
  double rounding = 0.5;
  for (uint8_t i=0; i<digits; ++i)
    rounding /= 10.0;

  number += rounding;

  // Extract the integer part of the number and print it
  unsigned long int_part = (unsigned long)number;
  double remainder = number - (double)int_part;
  n += print(int_part);

  // Print the decimal point, but only if there are digits beyond
  if (digits > 0) {
    n += print(".");
  }

  // Extract digits from the remainder one at a time
  while (digits-- > 0)
  {
    remainder *= 10.0;
    int toPrint = int(remainder);
    n += print(toPrint);
    remainder -= toPrint;
  }

  return n;
}
```