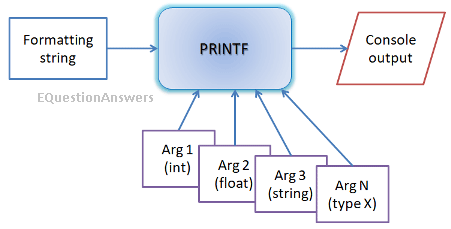
**Printf working principle**

printf or print function in C takes a formatting string and couple of optional variables as input and outputs strings to console while converting input variables to strings.



Printf and scanf takes multiple arguments and these functions are called variable length arguments function or vararg function. Take printf for consideration. User supply a string and input argumnets. Printf creates an internal buffer for constructing output string. Now printf iterates through each characters of user string and copies the character to the output string. Printf only stops at "%". "%" means there is an argument to convert. Arguments are in the form of char, int, long, float, double or string. It converts it to string and appends to output buffer. If the argument is string then it does a string copy. Finally printf may reach at the end of user sting and it copies the entire buffer to the stdout file.

**Implement printf**

Let us implement our own printf function. This is only for the understanding purpose. We name it print(). It has one string argument (str) and rest are variable arguments. Variable arguments are managed by macros like va\_start, va\_arg and va\_end. A temporary buffer (buff) is there to construct the output buffer. A while loop is needed to scan each characters in the input string. Now we iterate character by character in the loop and copy each character to output string. Same time we check for "%". "%" is not copied to output string. Once we found it, we check the next character. This is the formatting character. Formatting character says how to format the argument to visible output string. Printf supports varieties of formatting. C is for character, d for decimal integer, f for floating point, x for hexadecimal and s for strings. We match the formatting and pick the argument variable using va\_arg(). Argument variable is then converted to string format and appends to the output string. character can be copied as it is and Itoa function is used for [integer to string](http://www.equestionanswers.com/c/c-int-to-string.php) conversion.

C itoa() function has been used to convert argument [integer to string](http://www.equestionanswers.com/c/c-int-to-string.php). Integer to String conversion is a process to take each digits and convert those to ASCII format. Visit our topic [integer to string conversion](http://www.equestionanswers.com/c/c-int-to-string.php)for further understanding.

This process of coping characters and conversion of arguments repeats until the string is terminated to last NULL character. For simplicity we have implemented only c, d, x formatting cases. Now at the end we have the output string ready. This is now passed to fwrite() to stdout. Thus the output string prints in the actual console. Printf then returns the number of characters which is printed in the console and exit the function.

/\* Note: this is a minimal printf implementation \*/  
/\* This is for building understanding only \*/  
  
**#include** <stdio.h>  
**#include** <stdlib.h>  
**#include** <stdarg.h>  
**#include** <string.h>  
  
**char** \*\_strrev (**char** \*str)  
{  
  **int** i;  
  **int** len = 0;  
  **char** c;  
  **if** (!str)  
    **return** NULL;  
  **while**(str[len] != '\0'){  
    len++;  
  }  
  **for**(i = 0; i < (len/2); i++)  
  {  
    c = str[i];  
    str [i] = str[len - i - 1];  
    str[len - i - 1] = c;  
  }  
  **return** str;  
}  
  
**char** \* \_itoa(**int** i, **char** \*strout, **int** base)  
{  
  **char** \*str = strout;  
  **int** digit, sign = 0;  
  **if** (i < 0) {  
    sign = 1;  
    i \*= -1;  
  }  
  **while**(i) {  
    digit = i % base;  
    \*str = (digit > 9) ? ('A' + digit - 10) : '0' + digit;  
    i = i / base;  
    str ++;  
  }  
  **if**(sign) {  
  \*str++ = '-';  
  }  
  \*str = '\0';  
  \_strrev(strout);  
  **return** strout;  
}  
  
**int** print (**char** \* str, ...)  
{  
  va\_list vl;  
  **int** i = 0, j=0;  
  **char** buff[100]={0}, tmp[20];  
  **char** \* str\_arg;  
    
  va\_start( vl, str );  
  **while** (str && str[i])  
  {  
    **if**(str[i] == '%'){  
      i++;  
      **switch** (str[i]) {  
        /\* Convert char \*/  
        **case** 'c': {  
          buff[j] = (**char**)va\_arg( vl, **int** );  
          j++;  
          break;  
        }  
        /\* Convert decimal \*/  
        **case** 'd': {  
          \_itoa(va\_arg( vl, **int** ), tmp, 10);  
          strcpy(&buff[j], tmp);  
          j += strlen(tmp);  
          break;  
        }  
        /\* Convert hex \*/  
        **case** 'x': {  
          \_itoa(va\_arg( vl, **int** ), tmp, 16);  
          strcpy(&buff[j], tmp);  
          j += strlen(tmp);  
          break;  
        }  
        /\* Convert octal \*/  
        **case** 'o': {  
          \_itoa(va\_arg( vl, **int** ), tmp, 8);  
          strcpy(&buff[j], tmp);  
          j += strlen(tmp);  
          break;  
        }  
        /\* copy string \*/  
        **case** 's': {  
          str\_arg = va\_arg( vl, **char**\* );  
          strcpy(&buff[j], str\_arg);  
          j += strlen(str\_arg);  
          break;  
        }  
      }  
    } **else** {  
      buff[j] =str[i];  
      j++;  
    }  
    i++;  
  }   
  fwrite(buff, j, 1, stdout);   
  va\_end(vl);  
  **return** j;  
}  
**int** main (**int** argc, **char** \*argv[])  
{  
  **int** ret;  
  ret = print("%c %d %o %x %s\n", 'A', 10, 100, 1000, "Hello from printf!");  
  printf("printf returns %d bytes\n", ret);  
  **return** 0;  
}

**Output**

A 10 144 3E8 Hello from printf!

printf return 32