Sample Basic C Programs

Program #1: Fahrenheit and Celsius, Version 1
This prints a table with two columns, the left being Fahrenheit degrees, and the right the corresponding Celsius temperatures.

#include <stdio.h>

/*
 * print a table for Fahrenheit to Celsius
 * from 0 F to 300 F
 */
void main(void)
{
    int fahr; /* fahrenheit temperature */
    int celsius ; /* celsius temperature */
    register int lower = 0; /* begin table here */
    register int upper = 300; /* end table here */
    register int step = 20; /* increment */

    /*
    * print out the lines for the table
    */
    fahr = lower;
    while(fahr <= upper){
        /* get corresponding temp in degrees C */
        celsius = 5 * (fahr - 32) / 9;
        /* print it */
        printf("%d\t%d\n", fahr, celsius);
        fahr += step;
    }

    /*
    * say goodbye
    */
    exit(0);
}
Program #2: Fahrenheit and Celsius, Version 2

This does the same thing using different control structures and floating point arithmetic.

```c
#include <stdio.h>
#define LOWER 0 /* begin table here */
#define UPPER 300 /* end table here */
#define STEP 20 /* increment */

/* print a table for Fahrenheit to Celsius
* from 0 F to 300 F
* floating point version
*/
void main(void)
{
    float fahr; /* fahrenheit temperature */

    /* print out the lines for the table */
    for(fahr = LOWER; fahr <= UPPER; fahr += STEP)
        printf("%3.0f	%6.1f\n", fahr, (5.0/9.0) * (fahr - 32));

    /* say goodbye */
    exit(0);
}
```
Program #3: cat(1), Version 1

This short program copies the standard input to the standard output.

```c
#include <stdio.h>

/*
 * copy input to output: short version
 */
void main(void)
{
    int c; /* input character */

    /*
     * copy the input to the output
     * one char at a time
     */
    while ((c = getchar()) != EOF)
        putchar(c);

    /*
     * say goodbye
     */
    exit(0);
}
```
Program #4: cat(1), Version 2

This does the same thing, but uses different control structures. Most C programmers would write this program the previous way.

#include <stdio.h>

/*
 * copy input to output: long version
 */
void main(void)
{
    int c; /* input character */

    /* copy the input to the output
     * one char at a time
     */
    do {
        /* read a char */
        c = getchar();
        /* write a char (unless it's */
        /* the end of file marker) */
        if (c != EOF)
            putchar(c);
    } while (c != EOF);

    /* say goodbye */
    exit(0);
}
Program #5: A Version of wc(1)

This program counts the number of lines, words, and characters in its input, where a “word” is defined as any sequence of non-whitespace characters (blanks, tabs, and newlines).

#include <stdio.h>
#define IN_WORD 1 /* currently inside a word */
#define NOTIN_WORD 0 /* currently not in a word */

/* count the number of lines, words, and chars in the input
   * a word is a maximal sequence of nonspace characters, so
   * the quote "+++ --- hi bye 879+3" has 5 words ("+++", "---",
   * "hi", "bye", and "879+3")
 */
void main(void)
{
    register int c; /* input char */
    register int nl; /* line count */
    register int nw; /* word count */
    register int nc; /* char count */
    register int state; /* in or not in a word? */

    /* initialize */
    nl = nw = nc = 0;
    state = NOTIN_WORD;

    /* handle input a char at a time */
    while((c = getchar()) != EOF){
        /* got another character */
        nc++;
        /* is it a newline? */
        if (c == '
')
            nl++;
        /* is it a word separator? */
        if (c == ' ' || c == '	' || c == '
')
            /* YES -- change state */
            state = NOTIN_WORD;
        else if (state == NOTIN_WORD){
            /* NO -- we're now in a word; update */
            /* the counter and state if need be */
            state = IN_WORD;
            nw++;
        }
    }

    /* announce the results and quit */
}
printf("\%6d\t\%6d\t\%6d\n", nl, nw, nc);
exit(0);
}
Program #6: Digit and Space Counter, Version 1

This is similar to the word counting program, but counts digits, whitespace, and everything else (grouped into the category “other”). Note the array.

```c
#include <stdio.h>

/*
 * count the number of each digit, whitespace,
 * and all other chars
 */
void main(void)
{
    register int c; /* input char */
    register int nwhite = 0; /* whitespace count */
    register int nother = 0; /* other count */
    register int i; /* counter in a for loop */
    int ndigit[10]; /* digit counts */

    /*
     * initialize the ndigit array
     */
    for(i = 0; i < 10; i++)
        ndigit[i] = 0;

    /*
     * handle input a char at a time
     */
    while((c = getchar()) != EOF){
        /* see what it is */
        if (c >= '0' && c <= '9')
            /* it's a digit -- bump the right count */
            ndigit[c - '0']++;
        else if (c == ' ' || c == '	' || c == '
')
            /* it's whitespace */
            nwhite++;
        else
            /* it's neither a digit nor whitespace */
            nother++;
    }

    /*
     * announce the results and quit
     */
    printf("digits: ");
    for(i = 0; i < 10; i++)
    {
        printf("%c %3d	", i + '0', ndigit[i]);
        /* put 5 digits per line, for neat output */
        if (i == 4)
            printf("\n ");
    }
}
```
```c
putchar('\n');
printf("whitespace: %dnother: %dn", nwhite, nother);
exit(0);
```
Program #7: Digit and Space Counter, Version 2

This is also a digit counter, but uses a `switch` statement rather than an `if ... else if ... else` statement.

```c
#include <stdio.h>

/*
 * count the number of each digit, whitespace,
 * and all other chars
 */
void main(void)
{
    register int c;    /* input char */
    register int nwhite = 0;    /* whitespace count */
    register int nother = 0;    /* other count */
    register int i;    /* counter in a for loop */
    int ndigit[10];    /* digit counts */

    /*
    * initialize the ndigit array
    */
    for(i = 0; i < 10; i++)
        ndigit[i] = 0;

    /*
    * handle input a char at a time
    */
    while((c = getchar()) != EOF){
        /* see what it is */
        switch(c){
            case '0': case '1': case '2': case '3':  /* digit */
                case '4': case '5': case '6': case '7':
                case '8': case '9':
                    ndigit[c - '0']++;
                    break;
            case ' ': case '	': case '
':    /* whitespace */
                nwhite++;
                break;
            default:    /* neither a digit nor whitespace */
                nother++;
                break;
        }
    }

    /*
    * announce the results and quit
    */
    printf("digits: ");
    for(i = 0; i < 10; i++){
        printf("%c %3d\t", i + '0', ndigit[i]);
        /* put 5 digits per line, for neat output */
        if (i == 4)
            printf("\n    ");
    }
}
putchar('\\n');
printf("whitespace: %d\nother:      %d\n", nwhite, nother);
exit(0);
}
Program #8a: Powers of 2 and –3

This program prints a table of 2 and -3 raised to the powers 0 to 9 inclusive. It illustrates the use of function calls.

```c
#include <stdio.h>

/*
 * prototype (forward declaration)
 */
int power(int m, int n);

/*
 * generate a table of powers of 2
 */
void main(void)
{
    register int i; /* counter in a for loop */

    /*
     * generate the table
     */
    for(i = 0; i < 10; ++i)
        printf("%3d %6d %6d\n", i, power(2, i), power(-3, i));

    /*
     * bye!
     */
    exit(0);
}
```
Program #8b: Simple Exponentiation Function

This is the function call, from the same program and (as I wrote it) in the same file. Interestingly enough, this program will still work if the two routines (main and pow) are in different files!

```c
/*
 * compute a power
 * arguments: int base what you want to raise
 *            int n non-negative integral power
 *            to raise base to
 * returns:   base ^ n (base ** n to you FORTRANers!)
 * exceptions: none handled; overflow may occur, but there
 *              will be no indication of it
 */
int power(int base, int n)
{
    register int i; /* counter in a for loop */
    register int p; /* resulting power */

    /*
     * do it the obvious, multiplicative, way
     */
    for(i = 1, p = 1; i <= n; i++)
        p *= base;
    return(p);
}
```