Austin Bumbalough
CPE 325-08
9/3/2019

Lab 2 Solution

Output Screenshots

Part 1

Console

Lab_02:CIO

```
|-------------------------------------------------------------------------------------------
| Data Type              | Size (in bytes) | Minimum              | Maximum               |
-------------------------------------------------------------------------------------------
| char                   | 1               | -128                 | 127                   |
| short int              | 2               | -32768               | 32767                 |
| int                    | 2               | -32768               | 32767                 |
| long int               | 4               | -2147483648          | 2147483647            |
| long long int          | 8               | -9223372036854775808 | 9223372036854775807   |
| unsigned char          | 1               | 0                    | 255                   |
| unsigned short int     | 2               | 0                    | 65535                 |
| unsigned int           | 2               | 0                    | 65535                 |
| unsigned long int      | 4               | 0                    | 4294967295            |
| unsigned long long int | 8               | 0                    | 18446744073709551615  |
| float                  | 4               | 1.175494e-38         | 3.402823e+38          |
| double                 | 8               | 2.225074e-308        | 1.797693e+308         |
-------------------------------------------------------------------------------------------
```

Part 2

Console

Lab_02:CIO

```
Input Array X: [-3 -2 -1 0 1 2 3]
Input Constants m: 1 c: 0
Output Array Y: [-3 -2 -1 0 1 2 3]
```

Austin Bumbalough
CPE 325-08
9/3/2019

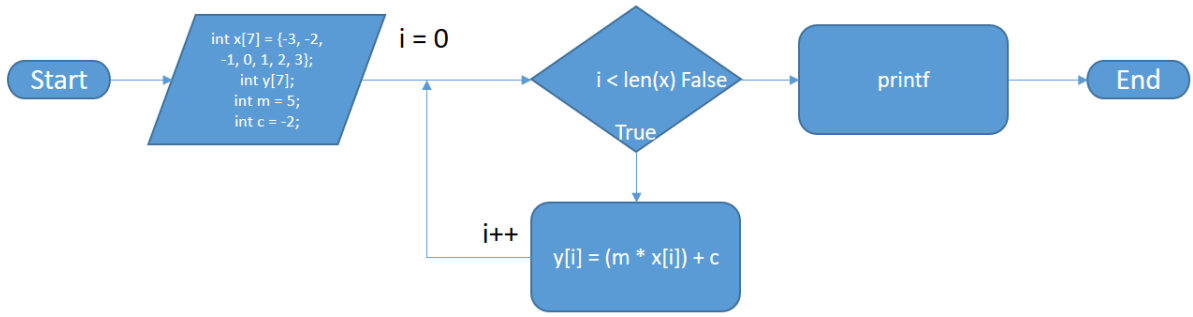
Bonus

```
Input Array X: [
[4 3 3 8 8 9 8 9],
[3 9 3 1 7 7 9 9],
[0 0 1 8 2 4 9 6],
[5 9 4 0 4 0 3 0],
[2 7 6 2 6 5 3 5],
[0 2 0 3 8 4 4 2],
[6 4 5 4 9 7 1 5],
[2 1 4 5 0 6 8 1]
]

Input Array I: [
[2 0 0 0 0 0 0 0],
[0 2 0 0 0 0 0 0],
[0 0 2 0 0 0 0 0],
[0 0 0 2 0 0 0 0],
[0 0 0 0 2 0 0 0],
[0 0 0 0 0 2 0 0],
[0 0 0 0 0 0 2 0],
[0 0 0 0 0 0 0 2]
]

Output Array Y = XI: [
[8 6 6 16 16 18 16 18],
[6 18 6 2 14 14 18 18],
[0 0 2 16 4 8 18 12],
[10 18 8 0 8 0 6 0],
[4 14 12 4 12 10 6 10],
[0 4 0 6 16 8 8 4],
[12 8 10 8 18 14 2 10],
[4 2 8 10 0 12 16 2]
]
```

Flow Diagram



Austin Bumbalough
CPE 325-08
9/3/2019

Source Code

```
#include <msp430.h>
#include <stdio.h>
#include <float.h>
#include <limits.h>

#define TABLE_WIDTH 91
#define NUM_TYPES 12
#define COL_1_WIDTH -23
#define COL_2_WIDTH -16
#define COL_3_WIDTH -20
#define COL_4_WIDTH -23

#define ARR_LEN 7


/*
-------------------------------------------------------------
*
File: Lab_02/main.c
*
Description: Print size and range of char, short int,
             int, long int, long long int, unsigned
             char, unsigned short int, unsigned int,
             unsigned long int, unsigned long long int,
             float, and double data types.
*
Input: N/A
*
Output: Print to stdout
*
Author: Austin Bumbalough
*
Lab Section: 8
*
Date: 8/29/19
*
Notes: Part 1 solution adapted from Stack Overflow user Paul Hankin
https://stackoverflow.com/a/14418659
*
-------------------------------------------------------------
*/


void part_1();

void part_2();

void bonus();

void print_data_type_info(int);

int main(void)
{
```

Austin Bumbalough
CPE 325-08
9/3/2019

```
    WDTCTL = WDTPW | WDTHOLD;   // stop watchdog timer

    part_1();

    part_2();

    bonus();

    return 0;
}

void part_1() {
    // Print table header row
    int i;
    for (i=0;i<TABLE_WIDTH;i++) printf("-");
    printf("\n");
    printf("| %-*s| %-*s| %-*s| %-*s|\n", COL_1_WIDTH, "Data Type",
COL_2_WIDTH, "Size (in bytes)", COL_3_WIDTH, "Minimum", COL_4_WIDTH,
"Maximum");
    for (i=0;i<TABLE_WIDTH;i++) printf("-");
    printf("\n");

    // Print information for each data type
    int j;
    for (j=0;j<NUM_TYPES;j++) {
        print_data_type_info(j);
    }

    // Print last row of table
    for (i=0;i<TABLE_WIDTH;i++) printf("-");
    printf("\n");
}

void part_2() {

    // Declare variables for input array, constants, and output array
    int x[ARR_LEN] = {-3, -2, -1, 0, 1, 2, 3};
    int m = 1;
    int c = 0;
    int y[7];

    // Comput output array using y = mx+c
    int i;
    for (i=0;i<ARR_LEN;i++) {
        y[i] = (m * x[i]) + c;
    }

    // Print each input array element
    printf("Input Array X: [");
    for (i=0;i<ARR_LEN;i++) {
        if (!(i == ARR_LEN -1)) {
            printf("%d ", x[i]);
        } else {
            printf("%d", x[i]);
        }
```

Austin Bumbalough
CPE 325-08
9/3/2019

```
    }
    printf("]\n");

    // Print constants
    printf("Input Constants m: %d c: %d\n", m, c);

    // Print each output array element
    printf("Output Array Y: [");
    for (i=0;i<ARR_LEN;i++) {
        if (!(i == ARR_LEN -1)) {
            printf("%d ", y[i]);
        } else {
            printf("%d", y[i]);
        }
    }
    printf("]\n");
}

void bonus() {
    // Declare input and output matrices and iteration variables
    int x[8][8] = {
        {4, 3, 3, 8, 8, 9, 8, 9},
        {3, 9, 3, 1, 7, 7, 9, 9},
        {0, 0, 1, 8, 2, 4, 9, 6},
        {5, 9, 4, 0, 4, 0, 3, 0},
        {2, 7, 6, 2, 6, 5, 3, 5},
        {0, 2, 0, 3, 8, 4, 4, 2},
        {6, 4, 5, 4, 9, 7, 1, 5},
        {2, 1, 4, 5, 0, 6, 8, 1}
    };

    int i[8][8] = {
        {2, 0, 0, 0, 0, 0, 0, 0},
        {0, 2, 0, 0, 0, 0, 0, 0},
        {0, 0, 2, 0, 0, 0, 0, 0},
        {0, 0, 0, 2, 0, 0, 0, 0},
        {0, 0, 0, 0, 2, 0, 0, 0},
        {0, 0, 0, 0, 0, 2, 0, 0},
        {0, 0, 0, 0, 0, 0, 2, 0},
        {0, 0, 0, 0, 0, 0, 0, 2}
    };

    int y[8][8];

    int j;
    int k;

    for (j=0;j<8;j++) {
        for (k=0;k<8;k++) {
            int sum = 0;
            int m = 0;

            // Multiply Row of X by Column of I and compute the sum
            while (m < 8) {
                sum += x[j][m]*i[m][k];
```

Austin Bumbalough
CPE 325-08
9/3/2019

```
                m++;
            }
            y[j][k] = sum;
        }
    }

    // Print input and output matrices
    printf("\nInput Array X: [\n");
    for (j=0;j<8;j++) {
        printf("[");
        for (k=0;k<8;k++) {
            if (!(k == 7)) {
                printf("%d ", x[j][k]);
            } else {
                printf("%d", x[j][k]);
            }
        }
        if (!(j == 7)) {
            printf("],\n");
        } else {
            printf("]\n");
        }
    }
    printf("]\n");

    printf("\nInput Array I: [\n");
    for (j=0;j<8;j++) {
        printf("[");
        for (k=0;k<8;k++) {
            if (!(k == 7)) {
                printf("%d ", i[j][k]);
            } else {
                printf("%d", i[j][k]);
            }
        }
        if (!(j == 7)) {
            printf("],\n");
        } else {
            printf("]\n");
        }
    }
    printf("]\n");

    printf("\nOutput Array Y = XI: [\n");
    for (j=0;j<8;j++) {
        printf("[");
        for (k=0;k<8;k++) {
            if (!(k == 7)) {
                printf("%d ", y[j][k]);
            } else {
                printf("%d", y[j][k]);
            }
        }
        if (!(j == 7)) {
            printf("],\n");
```

Austin Bumbalough
CPE 325-08
9/3/2019

```
        } else {
            printf("]\n");
        }
    }
    printf("]\n");

}

void print_data_type_info(int i) {

    switch(i) {
        case 0: // Signed char
            printf("| %*s| %*d| %*d| %*d|\n", COL_1_WIDTH, "char",
COL_2_WIDTH, sizeof(char), COL_3_WIDTH, SCHAR_MIN, COL_4_WIDTH, SCHAR_MAX);
            break;
        case 1: // Signed short int
            printf("| %*s| %*d| %*d| %*d|\n", COL_1_WIDTH, "short int",
COL_2_WIDTH, sizeof(short int), COL_3_WIDTH, SHRT_MIN, COL_4_WIDTH,
SHRT_MAX);
            break;
        case 2: // Signed int
            printf("| %*s| %*d| %*d| %*d|\n", COL_1_WIDTH, "int",
COL_2_WIDTH, sizeof(int), COL_3_WIDTH, INT_MIN, COL_4_WIDTH, INT_MAX);
            break;
        case 3: // Signed long int
            printf("| %*s| %*d| %*ld| %*ld|\n", COL_1_WIDTH, "long int",
COL_2_WIDTH, sizeof(long int), COL_3_WIDTH, LONG_MIN, COL_4_WIDTH, LONG_MAX);
            break;
        case 4: // Signed long long int
            printf("| %*s| %*d| %*lld| %*lld|\n", COL_1_WIDTH, "long long
int", COL_2_WIDTH, sizeof(long long int), COL_3_WIDTH, LLONG_MIN,
COL_4_WIDTH, LLONG_MAX);
            break;
        case 5: // Unsigned char
            printf("| %*s| %*d| %*d| %*d|\n", COL_1_WIDTH, "unsigned char",
COL_2_WIDTH, sizeof(unsigned char), COL_3_WIDTH, 0, COL_4_WIDTH, CHAR_MAX);
            break;
        case 6: // Unsigned short int
            printf("| %*s| %*d| %*d| %*u|\n", COL_1_WIDTH, "unsigned short
int", COL_2_WIDTH, sizeof(unsigned short int), COL_3_WIDTH, 0, COL_4_WIDTH,
USHRT_MAX);
            break;
        case 7: // Unsigned int
            printf("| %*s| %*d| %*d| %*u|\n", COL_1_WIDTH, "unsigned int",
COL_2_WIDTH, sizeof(unsigned int), COL_3_WIDTH, 0, COL_4_WIDTH, UINT_MAX);
            break;
        case 8: // Unsigned long int
            printf("| %*s| %*d| %*d| %*lu|\n", COL_1_WIDTH, "unsigned long
int", COL_2_WIDTH, sizeof(unsigned long int), COL_3_WIDTH, 0, COL_4_WIDTH,
ULONG_MAX);
            break;
        case 9: // Unsigned long long int
            printf("| %*s| %*d| %*d| %*llu|\n", COL_1_WIDTH, "unsigned long
long int", COL_2_WIDTH, sizeof(unsigned long long int), COL_3_WIDTH, 0,
COL_4_WIDTH, ULLONG_MAX);
```

```
            break;
        case 10: // Float
            printf("| %*s| %*d| %*e| %*e|\n", COL_1_WIDTH, "float",
COL_2_WIDTH, sizeof(float), COL_3_WIDTH, FLT_MIN, COL_4_WIDTH, FLT_MAX);
            break;
        case 11: // Double
            printf("| %*s| %*d| %*e| %*e|\n", COL_1_WIDTH, "double",
COL_2_WIDTH, sizeof(double), COL_3_WIDTH, DBL_MIN, COL_4_WIDTH, DBL_MAX);
            break;
        default:
            return;
    }
}
```