COMP1911 22T2 (https://webcms3.cse.unsw.edu.au/COMP1911/22T2) **Code Examples from Lectures on 8-4_arrays** Introduction to Programming (https://webcms3.cse.unsw.edu.au/COMP1911/22T2)

2dArray2.c (https://cgi.cse.unsw.edu.au/~cs1911/22T2/lec/8-4_arrays/code/2dArray2.c)

An example of initialising a 2d array using scanf and printing out the values

```
#include <stdio.h>
#define NUM_ROWS 3
#define NUM_COLS 5


/*
  col
   0   1   2   3   4
 ______________________
0 |1   2   3   4   5
1 |6   7   8   9   10 row
2 |?   ?   ?   ?   ?

*/
int main(void){
    int matrix[NUM_ROWS][NUM_COLS];
    int col;
    int row;
    printf("Enter %d integers\n", NUM_ROWS*NUM_COLS);

    //Read input from user
    row = 0;
    while(row < NUM_ROWS ){
        col = 0;
        while( col < NUM_COLS ){
            scanf("%d",&matrix[row][col]);
            col = col + 1;
        }
        row = row + 1;
    }

    //Print 2d array
    row = 0;
    while(row < NUM_ROWS ){
        col = 0;
        while( col < NUM_COLS ){
            printf("%d ",matrix[row][col]);
            col = col + 1;
        }
        printf("\n");

        row = row + 1;
    }
    return 0;
}
```

2dArray1.c (https://cgi.cse.unsw.edu.au/~cs1911/22T2/lec/8-4_arrays/code/2dArray1.c)

An example of initialising and accessing values in a 2d array

```c
#include <stdio.h>
#define MAX 3


/*
  col
   0   1   2   3   4   5
 ______________________
0 |1   2   3   4   5   6
1 |7   8   9  10  11  12
2 |99  -1  0  13  15   3

*/
int main(void){
    int matrix[MAX][MAX*2] = { {1,2,3,4,5,6},
                               {7,8,9,10,11,12},
                               {99,-1,0,13,15,3} };

    //This would initialise all values to 0
    //int matrix[MAX][MAX*2] = { {0} };

    //What does this code do?
    printf("%d\n",matrix[0][0]); //1
    printf("%d\n",matrix[1][0]); //7
    printf("%d\n",matrix[0][5]); //6
    printf("%d\n",matrix[2][5]); //3
    //printf("%d ",matrix[3][0]);  //ERROR!
    //printf("%d ",matrix[2][6]);  //NOT GOOD!
    //printf("%d ",matrix[1][6]);  //NOT GOOD!

    return 0;
}
```

2dArray3.c (https://cgi.cse.unsw.edu.au/~cs1911/22T2/lec/8-4_arrays/code/2dArray3.c)

Initialising a 2d array using a while loop

```
#include <stdio.h>

/*
   0   1   2
 ------------
0 |-1  -1  -1
1 |-1  -1  -1
2 |-1  -1  -1
3 |-1  -1  -1
4 |-1  -1  -1

*/

#define ROWS 5
#define COLS 3

int main(void){
    int numbers[ROWS][COLS];
    int row,col;

    //Initialise all elements to -1
    row = 0;
    while ( row < ROWS){
        col = 0;
        while ( col < COLS){
            numbers[row][col] = -1;
            col = col + 1;
        }
        row = row + 1;
    }

    //print the array
    row = 0;
    while(row < ROWS){
        col = 0;
        while(col < COLS){
             printf("%d ",numbers[row][col]);
             col = col + 1;
        }
        printf("\n");
        row = row + 1;
    }

    return 0;
}
```

outOfBounds.c (https://cgi.cse.unsw.edu.au/~cs1911/22T2/lec/8-4_arrays/code/outOfBounds.c)

An example of what can happen when you go beyond the bounds of an array

Hint: bad things can happen!

```
#include <stdio.h>

//See what happens with these different compiler options
//gcc -o
//gcc -Wall -Werror -O -o
//dcc -o
//dcc --valgrind -o

//Enter index 10 (just out of bounds)
//Enter index 100   (way out of bounds)
//Enter index 10000 (way way out of bounds)


int main(void) {
    int index;
    int testArray[10] = {0,1,2,3,4,5,6,7,8,9};
    int var = 99;

    printf("Var = %d\n",var);
    printf ("Enter an array index between 0 and 9\n");
    scanf ("%d", &index);

    printf ("testArray[%d]=%d\n",index, testArray[index]);
    testArray[index] = -1;
    printf("Var = %d\n",var);

    return 0;
}
```

magicSquare.c (https://cgi.cse.unsw.edu.au/~cs1911/22T2/lec/8-4_arrays/code/magicSquare.c)

Not finished: We will continue this in the next lecture....

Read SIZE x SIZE numbers and test if they form a magic square http://en.wikipedia.org/wiki/Magic_square (http://en.wikipedia.org/wiki/Magic_square)

We still need to check columns and diagonals

This will be done as a tutorial exercise (along with breaking it down into more functions).

```
Lo Shu Square
4  9  2   //0
3  5  7
8  1  6

Magic square of primes
17  89  71
113  59   5
47  29 101
```

```c
#include <stdio.h>

#define SIZE 3

void readSquare(int square[SIZE][SIZE]);
void printSquare(int square[SIZE][SIZE]);
int checkMagic(int square[SIZE][SIZE]);

int main(void) {
    int square[SIZE][SIZE];
    int row,col;
    int sumRow0;
    int sumRow;
    int isMagic = 1;

    // read potential magic square
    printf("Enter %d numbers please:\n", SIZE*SIZE);
    readSquare(square);


    // print potential magic square
    printf("Numbers are:\n");
    printSquare(square);

    isMagic = checkMagic(square);

    if(isMagic == 1){
        printf("It is magic\n");
    } else {
        printf("It is NOT magic\n");
    }
    return 0;
}

void readSquare(int square[SIZE][SIZE] ){
    int row,col;
    row = 0;
    while(row < SIZE ){
        col = 0;
        while( col < SIZE ){
            scanf("%d",&square[row][col]);
            col = col + 1;
        }
        row = row + 1;
    }
}

void printSquare(int square[SIZE][SIZE]){
    int row,col;
    row = 0;
    while(row < SIZE ){
        col = 0;
        while( col < SIZE ){
            printf("%d ",square[row][col]);
            col = col + 1;
        }
        printf("\n");

        row = row + 1;
    }
}

//returns 1 if it is a magic square
//returns 0 otherwise
int checkMagic(int square[SIZE][SIZE]){
    int firstRowSum = 0;

    //Find the sum of the first row
    int col = 0;
    while ( col < SIZE){
        firstRowSum = firstRowSum + square[0][col];
        col = col + 1;
    }

    //Check all the rows add up to firstRowSum


    int row = 1;
    int rowSum = 0;
    while (row < SIZE){
        //Find the sum of the row
        col = 0;
        rowSum = 0;
        while( col < SIZE){
```

```
            rowSum = rowSum + square[row][col];
            col = col + 1;
        }
        if( rowSum != firstRowSum){
            return 0;
        }
        row = row + 1;
    }
    // We still need to check columns and diagonals
    // This will be done as a tutorial exercise.
    return 1;
}
```

arrayRevisionExample.c (https://cgi.cse.unsw.edu.au/~cs1911/22T2/lec/8-4_arrays/code/arrayRevisionExample.c)

A revision example. Writing and using a function that takes in an array on ints and fills another array with the odd ints

```
#include <stdio.h>
#include <stdlib.h>

#define MAX 5

int findOdd(int src[],int dest[],int maxSize);

int main(void){
    int nums1[MAX] = {2,9,-4,6,1};
    int nums2[MAX];

    int numOdd = findOdd(nums1,nums2,MAX);

    //Print out the odd numbers we found
    int i  = 0;
    while (i < numOdd){
        printf("%d\n",nums2[i]);
        i = i + 1;
    }
    return EXIT_SUCCESS;
}
//src 0 1 2  3 4
//    2 9 -4 6 1
//              i

//dest 0 1 2 3 4
//     9 1 ? ? ?
//         destIndex

//Takes in an array called src of size maxSize
//and fills the dest array with any odd integers from the src array
//and returns the number of odd integers it found
int findOdd(int src[],int dest[],int maxSize){
    int destIndex = 0;
    int i = 0;
    while( i < maxSize ){
        if(src[i] % 2 != 0){
            dest[destIndex] = src[i];
            destIndex = destIndex + 1;
        }
        i = i + 1;
    }
    return destIndex;
}
```

whatDoesThisDo2.c (https://cgi.cse.unsw.edu.au/~cs1911/22T2/lec/8-4_arrays/code/whatDoesThisDo2.c)

Exercise in reading code

Try to understand and work out what this code would do

Then run to see if you are correct.

```c
#include <stdio.h>

#define SIZE 10

int main(void) {
    int numbers[SIZE];
    int i;

    i = 0;
    while (i < SIZE ) {
       numbers[i] = i * i ;
       i = i + 1;
    }


    i = 0;
    while (i < SIZE ) {
       printf("%d\n",numbers[i]);
       i = i + 1;
    }
    return 0;
}
```

uninitialisedMemory.c (https://cgi.cse.unsw.edu.au/~cs1911/22T2/lec/8-4_arrays/code/uninitialisedMemory.c)

An example of a program using uninitialized values in an array

This will result in using garbage values or if using the --valgrind flag with dcc, a runtime error

```c
#include <stdio.h>

#define N_NUMBERS 10

//Try running after compiling with these different options
//gcc -Wall -Werror -O -o arrayDemo2 arrayDemo2.c
//dcc -o arrayDemo2 arrayDemo2.c
//dcc --valgrind -o arrayDemo2 arrayDemo2.c


int main(void) {
    int x[N_NUMBERS], i, j;

    printf("Enter %d numbers: ", N_NUMBERS);
    i = 0;
    // Only reads 9 numbers instead of 10 so
    // the last one in the array still has garbage in it.
    while (i < N_NUMBERS-1) {
        scanf("%d", &x[i]);
        i = i + 1;
    }

    printf("Numbers reversed are:\n");
    j = N_NUMBERS-1;
    while (j >= 0) {
        printf("%d\n", x[j]);
        j = j - 1;
    }
    return 0;
}
```

reverse5smarter.c (https://cgi.cse.unsw.edu.au/~cs1911/22T2/lec/8-4_arrays/code/reverse5smarter.c)

Read 5 numbers and print them in reverse order

```c
#include <stdio.h>

#define N_NUMBERS 5

int main(void) {
    int x[N_NUMBERS], i, j;

    printf("Enter %d numbers: ", N_NUMBERS);
    i = 0;
    while (i < N_NUMBERS) {
        scanf("%d", &x[i]);
        i = i + 1;
    }
    printf("Numbers reversed are:\n");
    j = N_NUMBERS - 1;
    while (j >= 0) {
        printf("%d\n", x[j]);
        j = j - 1;
    }
    return 0;
}
```

allPositive.c (https://cgi.cse.unsw.edu.au/~cs1911/22T2/lec/8-4_arrays/code/allPositive.c)

An example of using functions to:

Read numbers into an array

Check array and see whether all numbers are +ve

```c
#include <stdio.h>

#define SIZE 10

// A function that returns 1 if all integers in the array are +ve and 0
// if any are negative.
int allPositive(int nums[SIZE]);
//returns the number of valid integers read in.
int readInput(int nums[SIZE]);
void printOutput(int nums[SIZE]);

int main(void) {
    int nums[SIZE];
    int i;
    printf("Enter %d numbers: ",SIZE);

    readInput(nums);
    printOutput(nums);

    if(allPositive(nums) == 1){
        printf("I am positive they are all positive\n");
    } else {
        printf("There is a Negative!\n");
    }

    return 0;
}

int readInput(int nums[SIZE]){
    int counter = 0;
    while (counter < SIZE && scanf("%d",&nums[counter]) == 1){
        counter = counter + 1;
    }
    return counter;
}

void printOutput(int nums[SIZE]){
    int counter = 0;
    while (counter < SIZE){
        printf("%d\n",nums[counter]);
        counter = counter + 1;
    }

}

// A function that returns 1 if all integers in the array are +ve and 0
// if any are negative.
// As soon as we encounter a negative number we return 0
// If we get to the end of the array and did not find any negative numbers,
// we return 0.
int allPositive(int nums[SIZE]){
    int counter = 0;
    while ( counter < SIZE){
        if(nums[counter] < 0){
            return 0;
        }
        counter = counter + 1;
    }
    return 1;
}
```

reverse5stupid.c (https://cgi.cse.unsw.edu.au/~cs1911/22T2/lec/8-4_arrays/code/reverse5stupid.c)

Read 5 numbers and print them in reverse order - the hard way

This approach quickly becomes impractical if you want to read more numbers a much better approach is to use an array

```c
#include <stdio.h>

int main(void) {
    int x0, x1, x2, x3, x4;
    printf("Enter 5 numbers: ");
    scanf("%d", &x0);
    scanf("%d", &x1);
    scanf("%d", &x2);
    scanf("%d", &x3);
    scanf("%d", &x4);
    printf("Numbers reversed are:\n");
    printf("%d\n", x4);
    printf("%d\n", x3);
    printf("%d\n", x2);
    printf("%d\n", x1);
    printf("%d\n", x0);
    return 0;
}
```

passByReference.c (https://cgi.cse.unsw.edu.au/~cs1911/22T2/lec/8-4_arrays/code/passByReference.c)

A comparison between passing primitive values like ints or doubles, which are passed by value/copy and arrays are passed by reference in C

```c
#include <stdio.h>

void f(int nums[], int size);
void f2(int x);

int main(void){
    int nums[5] = {0};
    int x = 3;
    f2(x);
    printf("%d\n",x); //NOT 42 as f gets a copy of x

    f(nums,5);

    int i = 0;
    while(i < 5){
        printf("%d\n",nums[i]); //nums[0] will be 42 as arrays are passed
                                //by reference
        i = i + 1;
    }
    return 0;
}

void f(int nums[], int size){
    nums[0] = 42;
}

void f2(int x){
    x = 42;
}
```

2dArrayFunctions.c (https://cgi.cse.unsw.edu.au/~cs1911/22T2/lec/8-4_arrays/code/2dArrayFunctions.c)

An example of writing functions to initialise and print a 2d array.

```
#include <stdio.h>

/*
   0   1   2
 ------------
0 |-1  -1  -1
1 |-1  -1  -1
2 |-1  -1  -1
3 |-1  -1  -1
4 |-1  -1  -1

*/

#define ROWS 5
#define COLS 3

// Initialise the first 'size' rows of the 2d numbers array
// to the given value
void initialiseArray(int numbers[][COLS],int size, int value);

// print 'size' rows of the 2d array numbers.
void printArray(int numbers[][COLS],int size);

int main(void){
    int numbers[ROWS][COLS];
    int row,col;

    initialiseArray(numbers,ROWS,-1);
    printArray(numbers,ROWS);

    return 0;
}



// Initialise the first 'size' rows of the 2d numbers array
// to the given value
void initialiseArray(int numbers[][COLS],int size, int value){
    int row,col;
    row = 0;
    while ( row < size){
        col = 0;
        while ( col < COLS){
            numbers[row][col] = value;
            col = col + 1;
        }
        row = row + 1;
    }
}

// print 'size' rows of the 2d array numbers.
void printArray(int numbers[][COLS],int size){
    int row,col;
    row = 0;
    while(row < size){
        col = 0;
        while(col < COLS){
             printf("%d ",numbers[row][col]);
             col = col + 1;
        }
        printf("\n");
        row = row + 1;
    }
}
```

returnArray.c (https://cgi.cse.unsw.edu.au/~cs1911/22T2/lec/8-4_arrays/code/returnArray.c)

An example to demonstrate that you can't return an array from a function. Instead you can pass one in and fill it with values!

```c
#include <stdio.h>

#define SIZE 3


//You can't return an array from a function
//int[] foo(void);

//You can instead pass in an array and modify its values!
void foo(int nums[]);

int main(void) {

    int a[SIZE];
    foo(a);
    printf("%d %d %d\n",a[0],a[1],a[2]);

    return 0;
}

//Instead of returning an array
//pass in an array and then fill it with values!
void foo(int nums[]){
    nums[0] = 1;
    nums[1] = 2;
    nums[2] = 3;
}

/*
You can't return an array.
As it is not a copy and the array
does not exist once the function has ended
int[] foo(void){
    int nums[] = {1,2,3};
    return nums;
}
*/
```

plotRainfall0.c (https://cgi.cse.unsw.edu.au/~cs1911/22T2/lec/8-4_arrays/code/plotRainfall0.c)

Read annual rainfall and plot as bar graph

This is a first draft that reads in the values stores them into two arrays and prints out the data

We still need to modify it to print out the rainfall as **** instead of just the raw numbers.

We still need to check users are not going to try to overfill the arrays

We still need to error check

We still need to use functions!

```
How many years of rainfall totals? 10
Enter year: 2005
Enter rainfall(mm): 816
Enter year: 2006
Enter rainfall(mm): 994.0
Enter year: 2007
Enter rainfall(mm): 1499.2
Enter year: 2008
Enter rainfall(mm): 1082.6
Enter year: 2009
Enter rainfall(mm): 956.2
Enter year: 2010
Enter rainfall(mm): 1153.8
Enter year: 2011
Enter rainfall(mm): 1369.2
Enter year: 2012
Enter rainfall(mm): 1213.6
Enter year: 2013
Enter rainfall(mm): 1344.4
Enter year: 2014
Enter rainfall(mm): 893.8

1 asterisk == 100 mm of rainfall
2005 ********
2006 *********
2007 **************
2008 **********
2009 *********
2010 ***********
2011 *************
2012 ************
2013 *************
2014 ********
```

```
#include <stdio.h>

#define N_YEARS  200000
#define SCALE    100

//       0     1     2     3    4   5 .... 199999
//years 2005  2006  2007  2008
//rain  816    994
int main(void) {
    int years[N_YEARS];
    double rainfall[N_YEARS];
    int numYears;
    printf("How many years of rainfall totals? ");
    scanf("%d",&numYears);

    int i = 0;
    while( i < numYears){
        printf("Enter year: ");
        scanf("%d",&years[i]);
        printf("Enter rainfall(mm):");
        scanf("%lf",&rainfall[i]);
        i = i + 1;
    }

    printf("1 asterisk == 100 mm of rainfall\n");
    i = 0;
    while( i < numYears){
        printf("%d: %lf\n",years[i],rainfall[i]);
        i = i + 1;
    }


    return 0;
}
```

arrayEx1.c (https://cgi.cse.unsw.edu.au/~cs1911/22T2/lec/8-4_arrays/code/arrayEx1.c)

Our first example using arrays

```
#include <stdio.h>
#include <stdlib.h>

#define SIZE 5

int main(void){
    int numbers[SIZE];
    int counter;
    int sum = 0;

    printf("Enter %d integers\n",SIZE);

    //Read in array values from user
    counter = 0;
    while (counter < SIZE){
        scanf("%d",&numbers[counter]);
        counter = counter + 1;
    }

    //print out values of the array
    printf("You entered\n");
    counter = 0;
    while (counter < SIZE){
        printf("%d\n",numbers[counter]);
        counter = counter + 1;
    }

    //add up all the values in the array
    counter = 0;
    while (counter < SIZE){
        sum = sum + numbers[counter];
        counter = counter + 1;
    }
    printf("%d\n",sum);

    return EXIT_SUCCESS; //return 0;
}
```

reverseN.c (https://cgi.cse.unsw.edu.au/~cs1911/22T2/lec/8-4_arrays/code/reverseN.c)

Read n numbers and print them in reverse order

Note for simplicity we are assuming scanf succeeds in reading an integer.

A robust program would check that scanf returns 1 to indicate an integer read.

```
#include <stdio.h>

#define MAX_NUMBERS 1000000

int main(void) {
    int x[MAX_NUMBERS], i, j, howMany;
    printf("Read how many numbers? ");
    scanf("%d", &howMany);
    if (howMany > MAX_NUMBERS) {
        printf("I'm sorry. I'm afraid I can't do that.\n");
        printf("Reading %d numbers\n", MAX_NUMBERS);
        howMany = MAX_NUMBERS;
    }
    i = 0;
    while (i < howMany) {
        scanf("%d", &x[i]);
        i = i + 1;
    }
    printf("Numbers reversed are:\n");

    j = howMany - 1;
    while (j >= 0) {
        printf("%d\n", x[j]);
        j = j - 1;
    }
    return 0;
}
```

sum.c (https://cgi.cse.unsw.edu.au/~cs1911/22T2/lec/8-4_arrays/code/sum.c)

An example of different prototypes for simple array based functions

```c
#include <stdio.h>


#define SIZE 10
#define MAX 100


int sum1(int nums[SIZE]);
int sum2(int nums[], int size);

int main(void) {
    int nums[SIZE] = {1, 2, 3};
    int nums2[MAX] = {1,2,3,4,5,6,7,8,9,10,11};
    int nums3[] = {3,1,4,1,8};

    int s1 = sum1(nums);

    //sum2 can be used for arrays of different sizes

    int s2 = sum2(nums,SIZE);
    int s3 = sum2(nums2, MAX);
    int s4 = sum2(nums3,5);

    //Or even to find the sum of say the first 7 items of a larger array
    int sumOfFirst7 = sum2(nums2,7);
    printf("%d\n",s1);
    printf("%d\n",s2);
    printf("%d\n",s3);
    printf("%d\n",s4);
    printf("%d\n",sumOfFirst7);
    return 0;
}

//This can only be used for arrays of size SIZE
int sum1(int nums[SIZE]){
    int sum = 0;
    int i = 0;
    while ( i < SIZE){
        sum = sum + nums[i];
        i = i + 1;
    }
    return sum;
}

//This can be used for any sized array, or to sum
//the first 'size' elements of an array
int sum2(int nums[], int size){
    int sum = 0;
    int i = 0;
    while ( i < size){
        sum = sum + nums[i];
        i = i + 1;
    }
    return sum;
}
```

whatDoesThisDo.c (https://cgi.cse.unsw.edu.au/~cs1911/22T2/lec/8-4_arrays/code/whatDoesThisDo.c)

Exercise in reading code

Try to understand and work out what this code would do

Then run to see if you are correct.

```c
#include <stdio.h>

#define SIZE 10

int main(void) {
    int numbers[] = {1,2,3,4,5,6,7,8,9,10};
    int i;

    i = 0;
    while (i*2 < SIZE ) {
        printf("%d\n",numbers[i*2]);
        i = i + 1;
    }

    return 0;
}
```

reverse5smart.c (https://cgi.cse.unsw.edu.au/~cs1911/22T2/lec/8-4_arrays/code/reverse5smart.c)

Read 5 numbers and print them in reverse order

Note for simplicity we are assuming scanf succeeds in reading an integer.

A robust program would check that scanf returns 1 to indicate an integer read.

The constants 4 & 5 below would be better replaced with a #deine

```
#include <stdio.h>

int main(void) {
    int x[5], i, j;
    printf("Enter 5 numbers: ");
    i = 0;
    while (i < 5) {
        scanf("%d", &x[i]);
        i = i + 1;
    }
    printf("Numbers reversed are:\n");
    j = 4;
    while (j >= 0) {
        printf("%d\n", x[j]);
        j = j - 1;
    }
    return 0;
}
```

arrayCopy.c (https://cgi.cse.unsw.edu.au/~cs1911/22T2/lec/8-4_arrays/code/arrayCopy.c)

Example of how to make a copy of an array

```
#include <stdio.h>
#define SIZE 10

int main(void) {
    int i;

    int numbers[SIZE] = {1,2,3,4,5,6,7,8,9,10};
    int numbers2[SIZE];

    //You can't assign one array to another
    //numbers2 = numbers

    //Copy each int from numbers to numbers2
    //Make sure numbers2 is big enough!
    i = 0;
    while( i < SIZE){
        numbers2[i] = numbers[i];
        i = i + 1;
    }

    //Print out both arrays
    i = 0;
    while( i < SIZE){
        printf("%d %d\n",numbers[i],numbers2[i]);
        i = i + 1;
    }
    printf("\n");


    return 0;
}
```

plotRainfall.c (https://cgi.cse.unsw.edu.au/~cs1911/22T2/lec/8-4_arrays/code/plotRainfall.c)

Read annual rainfall and plot as bar graph

This is a second draft that reads in the values stores them into two arrays and prints out the bar graph

We still need to check users are not going to try to overfill the arrays

We still need to error check

We still need to use functions for reading in data etc!

To feed in data from the file sydney_annual_rainfall.txt run like ./plotRainfall < sydney_annual_rainfall.txt

```
How many years of rainfall totals? 10
Enter year: 2005
Enter rainfall(mm): 816
Enter year: 2006
Enter rainfall(mm): 994.0
Enter year: 2007
Enter rainfall(mm): 1499.2
Enter year: 2008
Enter rainfall(mm): 1082.6
Enter year: 2009
Enter rainfall(mm): 956.2
Enter year: 2010
Enter rainfall(mm): 1153.8
Enter year: 2011
Enter rainfall(mm): 1369.2
Enter year: 2012
Enter rainfall(mm): 1213.6
Enter year: 2013
Enter rainfall(mm): 1344.4
Enter year: 2014
Enter rainfall(mm): 893.8

1 asterisk == 100 mm of rainfall
2005 ********
2006 *********
2007 **************
2008 **********
2009 *********
2010 ***********
2011 *************
2012 ************
2013 *************
2014 ********
```

```c
#include <stdio.h>

#define N_YEARS  200000
#define SCALE    100

void printStars(int n);

//       0     1     2     3    4   5 .... 199999
//years 2005  2006  2007  2008
//rain  816    994
int main(void) {
    int years[N_YEARS];
    double rainfall[N_YEARS];
    int numYears;
    printf("How many years of rainfall totals? ");
    scanf("%d",&numYears);

    int i = 0;
    while( i < numYears){

        printf("Enter year: ");
        scanf("%d",&years[i]);
        printf("Enter rainfall(mm):");
        scanf("%lf",&rainfall[i]);
        i = i + 1;
    }

    printf("1 asterisk == 100 mm of rainfall\n");
    i = 0;
    while( i < numYears){
        int numStars = rainfall[i]/100;
        printf("%d: ",years[i]);
        printStars(numStars);
        i = i + 1;
    }


    return 0;
}

//print n stars
void printStars(int n){
    int j = 0;
    while(j < n){
        printf("*");
        j = j + 1;
    }
    printf("\n");
}
```

arrayEx1Functions.c (https://cgi.cse.unsw.edu.au/~cs1911/22T2/lec/8-4_arrays/code/arrayEx1Functions.c)

This is our arrayEx1.c example restructured to use functions!

And also error check the input

```c
#include <stdio.h>
#include <stdlib.h>

#define SIZE 5


//returns how many valid integes were read in
int readInput(int nums[SIZE]);
void printOutput(int nums[SIZE]);
int findSum(int nums[SIZE]);

// 0  1  2  3  4
//23  1  9 12 -3
int main(void){
    int numbers[SIZE];
    int sum = 0;

    printf("Enter %d integers\n",SIZE);
    int numRead = readInput(numbers);
    if(numRead != SIZE){
        printf("Invalid\n");
        return EXIT_FAILURE;
    }

    printf("You entered\n");
    printOutput(numbers);

    sum = findSum(numbers);
    printf("%d\n",sum);

    return EXIT_SUCCESS; //return 0;
}

int findSum(int nums[SIZE]){
    int counter = 0;
    int sum = 0;
    while (counter < SIZE){
        sum = sum + nums[counter];
        counter = counter + 1;
    }
    return sum;
}

//returns how many valid integes were read in
int readInput(int nums[SIZE]){
    int counter = 0;
    while (counter < SIZE && scanf("%d",&nums[counter]) == 1){
        counter = counter + 1;
    }
    return counter;
}

void printOutput(int nums[SIZE]){
    int counter = 0;
    while (counter < SIZE){
        printf("%d\n",nums[counter]);
        counter = counter + 1;
    }
}
```

reverse5checkScanf.c (https://cgi.cse.unsw.edu.au/~cs1911/22T2/lec/8-4_arrays/code/reverse5checkScanf.c)

Read 5 numbers and print them in reverse order

This version checks that the scanf was able to read the number

```c
#include <stdio.h>

#define N_NUMBERS 5

int main(void) {
    int x[N_NUMBERS], i, j;

    printf("Enter %d numbers: ", N_NUMBERS);
    i = 0;
    while (i < N_NUMBERS && scanf("%d", &x[i]) == 1) {

        i = i + 1;
    }
    if (i < N_NUMBERS) {
        printf("Insufficient numbers read\n");
    } else {
        printf("Numbers reversed are:\n");
        j = N_NUMBERS - 1;
        while (j >= 0) {
            printf("%d\n", x[j]);
            j = j - 1;
        }
    }
    return 0;
}
```

initArrays.c (https://cgi.cse.unsw.edu.au/~cs1911/22T2/lec/8-4_arrays/code/initArrays.c)

Different ways to initialise an array

```c
#include <stdio.h>

#define SIZE_ARRAY 3

//Uncomment out the different lines that initialise the
//nums array and confirm that printing the array
//would give the results you expect
int main(void) {
    //int nums[SIZE_ARRAY];
    //int nums[] = {4,2,5};
    //int nums[SIZE_ARRAY] = {0};
    //int nums[SIZE_ARRAY] = {3,99};
    int i;


    i = 0;
    while(i < SIZE_ARRAY){
       printf("%d\n", nums[i]);
       i = i + 1;
    }

    return 0;
}
```