Step 8: Arrays

This assigment is done in class on (monday)03-10-14 AND (friday)02-28-14.

This assignment is due for monday class 03-10-14 AND for the firday class 02-28-14.

Getting Started

By now you should be all logged on. You should still have a cse251 directory in your home directory from last week. Please change into that directory.

You should be able to do a pwd and see that you are in the cse251 directory at this time.

Remember, start gedit with this command line:

```
gedit &
```

Create a new program called array.c with the following code:

```
#include <stdio.h>

/*
 * Name : <Insert name>
 * Program to experiment with arrays
 */

int main()
{

}
```

Be sure it compiles and runs. The commands to compile and run are, of course:

```
gcc -o array array.c
./array
```

> I don't need the -lm here because I'm not using any of the math library functions. If we start to use sin, cos, pow, etc. we will need to add the -lm back on.

1. Introducing Arrays

We are going to create an array. Add this variable declaration to your program:

```
    double gross[5] = {4633070, 3856195, 3171189, 1877685, 3251622};
```

This is a variable declaration just like we have been doing before. The only difference is that this now is an array instead of a scalar variable. This goes in the main function like this:

```
int main()
{
    double gross[5] = {4633070, 3856195, 3171189, 1877685, 3251622};

}
```

 This array contains the box office gross for the top five movies from February 25, 2011. It's much easier to enter this data as an array than it is to have five different variables. It will definitely be easier to process.

Now, we'll write some simple code to add these together:

```
    double sum;

    sum = gross[0] + gross[1] + gross[2] + gross[3] + gross[4];
    printf("Total gross for these films was $%.0f\n", sum);
```

 When you run this you'll get the total gross.

> Notice: In the C programming language an array starts with location zero, not one. So, this array, which has five locations (gross[5]) will have locations from 0 to 4. If you try to access outside the range of the array your program may crash and you definitely will get erroneous results.

It's a pain to add these one at a time and even worse is that this would be hard to adapt to some different number. Fortunately, we can use a for loop to add these together, instead. Replace the line the computes sum with this code:

```
    sum = 0;        /* Initialize to zero */
    for(i = 0;  i<5;  i++)
    {
        sum += gross[i];
    }
```

 Be sure to declare the variable i as an integer. We use integers when indexing into an array. This should work like before.

> The loop will execute the loop body with the numbers 0, 1, 2, 3, and 4. Because I said i < 5, it does not access location 5, which is good because it does not exist.

A problem with a program like this is that it has a bunch of "hard numbers" in it. If I decided I wanted to average 10 movies instead of 5, I might have a lot of those 5's in my program to fix. A good solution for that problem is to use a constant. Add this line of code to your program above the main function:

```
#define NumMovies 5
```

It goes here:

```
#include <stdio.h>

/*
 * Name : <Insert name>
 * Program to experiment with arrays
 */

#define NumMovies 5


int main()
{
```

This is called a "preprocessor directive" and is treated somewhat special in the C language. Note that there is NO ";" on the end of this line. Do not put a semicolon on the end of a #define line. This line says to the compiler: Wherever I see "NumMovies", replace that with a 5. It doesn't care how it is used. It provides a single place to put this number.

Change the line that creates the array to this:

```
    double gross[NumMovies] = {4633070, 3856195, 3171189, 1877685, 3251622};
```

Then change the loop to this:

```
    for(i = 0;  i<NumMovies;  i++)
```

 This should run exactly as before.

I'm going to change my program to have 10 movies instead of 5. Change the line that declares the array to:

```
    double gross[NumMovies] = {4633070, 3856195, 3171189, 1877685, 3251622,
                               3156594, 1629735, 2659234, 2028036, 510768};
```

I now have the box office gross for the top ten movies. Try compiling this. You will get a message like this:

```
cbowen@ubuntu:~/cse251$ gcc array.c
array.c: In function 'main':
array.c:14: warning: excess elements in array initializer
array.c:14: warning: (near initialization for â€˜grossâ€™)
array.c:14: warning: excess elements in array initializer
array.c:14: warning: (near initialization for â€˜grossâ€™)
array.c:14: warning: excess elements in array initializer
array.c:14: warning: (near initialization for â€˜grossâ€™)
```

```
array.c:14: warning: excess elements in array initializer
array.c:14: warning: (near initialization for â€˜grossâ€™)
array.c:14: warning: excess elements in array initializer
array.c:14: warning: (near initialization for â€˜grossâ€™)
```

This is telling you that you have provided too many initializers. The value of NumMovies is 5, so the array has 5 locations. Since we are providing 10 values, it can't put them all into the array.

You can fix this easily by changing the value of NumMovies to 10:

```
#define NumMovies 10
```

 This should run and give you the correct answer.

Suppose I want to display the output. I can add another for loop to do that. Add this code before the loop that computes the sum:

```
    for(i=0;  i<NumMovies; i++)
    {
        printf("Movie %d: %.0f\n", i, gross[i]);
    }
```

 This should output the movie grosses and the total gross.

Now we get into a strange thing in that C requires an array to start at 0, but humans don't expect the first thing to be 0. When you count the petals on a clover, do you go: "Zero, One, Two, Three"? Of course not. So, we'll make our output look like what a human expects. Change that printf statement to this:

```
        printf("Movie %d: %.0f\n", i+1, gross[i]);
```

 This output is probably more like what you would expect.

Up to now we've never had a program that could store or manipulate a name or other string of characters. I want to add the name of the movies to this program. A name of a movie is a bunch of characters (of type "char") in a row in memory. So, we can use a pointer to point to them. The type we'll use is "char *". We'll see today how pointers and arrays are actually related. A character string is an array of characters.

For now, add this line of code to declare an array of the movie names:

```
    char *names[NumMovies] = {"Hall Pass", "Unknown", "I Am Number Four", "The King's Speech",
                        "Just Go With It", "Gnomeo and Juliet", "Drive Angry",
                        "Justin Beiber: Never Say Never", "Big Mommas: Like Father, Like Son",
                        "True Grit"};
```

This is an array. The type of the array is "char *", which is a pointer to a string of characters. The name of the array is names. The initializers are character strings. There are 10 of them and they match the box office grosses in the gross declaration.

Now we can print the names of the movies. The printf format specifier for a string is %s. Change the printf statement to this:

I apologize for the mention of Justin Beiber in this assignment. It's what the data said and I can't help it if a zillion screaming little girls really want to see the guy.

BTW, if you know he cut his hair?

```
        printf("Movie %d %s: %.0f\n", i+1, names[i], gross[i]);
```

 This should run and output the movie names.

I thought the output was rather ugly, so I added some format specifiers to make it look nicer:

```
        printf("Movie %2d %33s: %.0f\n", i+1, names[i], gross[i]);
```

 This should run and output the movie names in a nice column format like this:

```
cbowen@ubuntu:~/cse251$ ./array
Movie  1                         Hall Pass: 4633070
Movie  2                           Unknown: 3856195
Movie  3                  I Am Number Four: 3171189
Movie  4                 The King's Speech: 1877685
Movie  5                   Just Go With It: 3251622
```

```
Movie  6                  Gnomeo and Juliet: 3156594
Movie  7                        Drive Angry: 1629735
Movie  8    Justin Beiber: Never Say Never: 2659234
Movie  9 Big Mommas: Like Father, Like Son: 2028036
Movie 10                         True Grit: 510768
Total gross for these films was $26774128
```

Here's something fun to do. I want to know what movie grossed the most money. Now that happens to be the first one in the list, but maybe it would not be. Here's the code to figure that out. First, add these variables:

```
    int highestGrossIndex = 0;
    double highestGross = 0;
```

Then add this code to find the highest grossing movie:

```
    highestGross = 0;
    for(i=0;  i<NumMovies;  i++)
    {
        if(gross[i] > highestGross)
        {
            highestGrossIndex = i;
            highestGross = gross[highestGrossIndex];
        }
    }

    printf("The highest grossing film is: %s\n", names[highestGrossIndex]);
```

 This should indicate that Hall Pass is the highest grossing movie. Be sure you understand how this works.

Now take a moment and add code to your program that outputs the highest grossing film with a gross less than each of these values:

```
    double maxGross[MaxGrosses] = {500000, 1000000, 2000000, 4000000, 10000000};
```

This output should look something like this:

```
No film made less than 500000
The highest gross less than 1000000 is ??? at ???
The highest gross less than 2000000 is ??? at ???
The highest gross less than 4000000 is ??? at ???
The highest gross less than 10000000 is ??? at ???
```

Turn in array.c via http://www.cse.msu.edu/handin

2. More Array Work

Create a new program called words.c and start with this code:

```
#include <stdio.h>

/*
 * Name : <Insert name>
 * Program to experiment with character arrays
 */

#define MaxWord 20

int main ()
{
    char c;
    char str[MaxWord+1];

    puts ("Enter text. Include a dot ('.') to end a sentence to exit:");
    do
    {
        c=getchar();

    } while (c != '.');

}
```

 Run this to see what it does.

I have introduced a new function here: getchar(). This gets one character from the user. I put that character into the variable c, which is of type char. A char variable holds a single character.

I want to collect up the letters of a word in the character array str. We store character strings in a char array. Right now it has nothing in it at all.

Add this variable to keep track of the length of the word we are currently entering:

```
    int len = 0;
```

When we get a character we will check to see if it is a space or period. If it is not, we add to the word we are entering. If it is, we will output the word using printf. We'll use str to keep track of the letters as we enter them.

Add this code after the line: c=getchar():

```
        if(c != ' ' && c != '.')
        {
            /* This is a character of a word */
            str[len] = c;
            len++;
        }
```

This is collecting up the letters of a word.

Just to see what this is doing, I'm going to print the string after the loop is done. Now, there is something we have to do before we can print the string. In the language C, a string must be *null terminated*, meaning it has to end with a zero. That's how C knows the end of your string. After the end of the do/while loop, add this line of code to null terminate the string:

```
    str[len] = 0;
```

Then use printf to print the string:

```
    printf("%s\n", str);
```

> You may have noticed that I made the size of the character string str MaxWord + 1. The reason for that is that you have to make a character string one longer than the space you need to allow for the null termination zero at the end. So, the string "Hello" is 5 letters and a zero, requiring an array of 6 locations. I want the maximum word size my program will accommodate to be MaxWord, so I need MaxWord + 1 space in the character array.

 Run this and enter:

```
Now is the time.
```

Be sure to end the sentence with a period. The output should look like this:

```
Enter text. Include a dot ('.') to end a sentence to exit:
Now is the time.
Nowisthetime
```

See how the spaces and period were not added to the line?

We want to output the words now. To do that, I will need to add an else to the test for space or period. Add this code:

```
        if(c != ' ' && c != '.')
        {
            /* This is a character of a word */
            str[len] = c;
            len++;
        }
        else
        {
            /* The word is done */
            str[len] = 0;
            printf("%s\n", str);
            len = 0;
        }
```

 Run this and type: Now is the time. You should get something like this:

```
Enter text. Include a dot ('.') to end a sentence to exit:
Now is the time.
Now
is
the
time
```

Isn't this cool?

There is a major problem with this program. What if I enter: My galoopadonkeltykinbarken is dead. Try it and see what happens. You should get some awful message like this:

```
*** stack smashing detected ***: ./words terminated
======= Backtrace: =========
/lib/tls/i686/cmov/libc.so.6(__fortify_fail+0x50)[0x1f2390]
/lib/tls/i686/cmov/libc.so.6(+0xe233a)[0x1f233a]
./a.out[0x8048513]
/lib/tls/i686/cmov/libc.so.6(__libc_start_main+0xe6)[0x126bd6]
./a.out[0x80483e1]
======= Memory map: ========
00110000-00263000 r-xp 00000000 08:01 10376      /lib/tls/i686/cmov/libc-2.11.1.so
00263000-00264000 ---p 00153000 08:01 10376      /lib/tls/i686/cmov/libc-2.11.1.so
00264000-00266000 r--p 00153000 08:01 10376      /lib/tls/i686/cmov/libc-2.11.1.so
00266000-00267000 rw-p 00155000 08:01 10376      /lib/tls/i686/cmov/libc-2.11.1.so
00267000-0026a000 rw-p 00000000 00:00 0
0068f000-006aa000 r-xp 00000000 08:01 10374      /lib/ld-2.11.1.so
006aa000-006ab000 r--p 0001a000 08:01 10374      /lib/ld-2.11.1.so
006ab000-006ac000 rw-p 0001b000 08:01 10374      /lib/ld-2.11.1.so
009a4000-009a5000 r-xp 00000000 00:00 0          [vdso]
009c6000-009e3000 r-xp 00000000 08:01 6668       /lib/libgcc_s.so.1
009e3000-009e4000 r--p 0001c000 08:01 6668       /lib/libgcc_s.so.1
009e4000-009e5000 rw-p 0001d000 08:01 6668       /lib/libgcc_s.so.1
08048000-08049000 r-xp 00000000 08:01 389977     /home/cbowen/cse251/a.out
08049000-0804a000 r--p 00000000 08:01 389977     /home/cbowen/cse251/a.out
0804a000-0804b000 rw-p 00001000 08:01 389977     /home/cbowen/cse251/a.out
09c1e000-09c3f000 rw-p 00000000 00:00 0          [heap]
b76f6000-b76f7000 rw-p 00000000 00:00 0
b7706000-b770a000 rw-p 00000000 00:00 0
bfc2d000-bfc42000 rw-p 00000000 00:00 0          [stack]
Aborted
```

What happened is you tried to write past the end of the array. The word galoopadonkeltykinbarken is longer than 20 characters. So, we need to stop when the string gets to 20 characters. That's easy to do:

```
            /* Only save it if we have space */
            if(len < MaxWord)
            {
                str[len] = c;
                len++;
            }
```

 This should work better.

Final Tasks

Some final tasks to do and this program will be done:

1. Have your program ignore words that are zero length. This happens if you type two spaces in a row. That will take just one simple test.

2. Have your program print the average word length.

3. Have your program print the longest word entered.

To do the last part, you will need to create another array and save the word into it. You can save the word into it by simply copying the characters over into it. Remember, len is how long the word is so far.

Enter this line of text as a test case:

```
The comfort of a knowledge of a rise above the sky
but could never parallel the challenge of an acquisition in the
```

```
here and now.
```

Turn in words.c via http://www.cse.msu.edu/handin

CSE 251