Project 1

Programming and Algorithms II CSCI 211

Objectives

- Practice text input and output
- Practice arrays
- Practice loops
- Practice functions
- Write a program from scratch

Overview

You will write a program named `chart` to read a set of integers from standard input and draw a bar chart to standard output using asterisks and spaces (‘*’ and ‘ ’).

The user specifies a set of up to 100 integers greater than 0. When the user enters a 0, the program will stop reading standard input and print the chart.

For example, if the user entered the sequence

1 2 3 4 3 2 1 6 0

then your program must print:

```
       *
       *
   *   *
  ***  *
 ***** *
********
```

Note: There are no spaces before the first bar, and there are no spaces after the last bar.

Program Requirements

The user might input numbers on separate lines. Read the users input using ”cin >>” so you don’t have to think about lines, spaces, or tabs, they will just be ignored as “white space”.

Your program should read up to 100 positive integers (greater than 0) from standard input. You must store these integers in a single array. Do not use a vector.

You must use functions to compartmentalize subtasks. Your main function should be the “driver” that calls these functions. There are three subtasks in this program, so you should create at least 3 additional functions.

Your functions should appear before / above your main function.

Formatting and Style

Your program must be neatly organized and consistently indented. You must have informative comments throughout your program. Comment your code! The first lines of all your files (.h and .cpp) must contain the following comments:

```
//  filename
//  last name, first name
//  ecst_username
```

Example Execution

```
$ ./chart

1 4 2 3 0

 *
 * *
 ***
****
$
```

At the UNIX prompt $ the user typed `chart`, the program's executable, and hit enter. Then the user typed five integers separated by spaces. The zero value at the end tells indicates the end of input. The program then prints out the asterisks and spaces to form the chart.

Make sure you output what is expected, nothing more and nothing less.

Additional Requirements

Use Constants

The max allowed number of integers that can be input is 100, so you may statically create an array of 100 integers. It is good programming practice to place the “100” in a single place so that it is easy to change:

```
const int MAX = 100;
int values[MAX];
...
for (int i = 0; i < MAX; i++)
{
    ...
}
```

the above is better than

```
int values[100];
...
for (int i = 0; i < 100; i++)
{
    ...
}
```

Finding the Largest Number

You will be drawing a rectangular grid with a width equal to the number of bars (values entered) and a height equal to the height of the tallest bar. At each spot in the grid you must draw a space or an asterisk.

Since you need to know the height of the tallest bar before you start drawing the bars, you will need a function that finds the largest integer in an array of integers, like the one shown below:

```
// return the largest value in the given array of integers
int find_largest(int values[], int size) {
    int largest = 0;
    /* code to find the largest integer goes here */
    return largest;
}
```

You would call the function above like this:

```
int largest = find_largest(values, size);  // Notice that there are no []
```

In C/C++ you can pass an array without specifying the size of the array. This is because you are passing the pointer to the start of the array. We will discuss this in later lectures.

Compile & Test

Use the run_tests program to validate that all tests are passing locally before submitting your final solution to TurnIn. See **Guides > Advanced Local Testing** if you don't know how to use the run_tests program.

Make sure you output an "end of line" (using '<< endl' or '\n') with your last line of output. If you forget this, you will not pass the project 1 test set.

Submission

Submit your completed program to https://turnin.ecst.csuchico.edu/ under Project 1. Note: Submit only your C++ source file (chart.cpp). Do not submit your object file, Makefile or your executable program.

Each student will complete and submit this assignment individually. Do not work with anyone else on this or any other CSCI 211 project. An anti-plagiarism tool will be used to ensure that all student submissions are unique. DO YOUR OWN WORK. DO NOT ALLOW ANYONE ELSE TO SEE OR USE YOUR SOLUTION. Submit your work before the due date to receive maximum credit, or at least before the late cut-off date to receive partial credit (see Course Content > Syllabus for info on the CSCI 211 lateness policy).