UNIVERSITEIT•STELLENBOSCH•UNIVERSITY
jou kennisvennoot • your knowledge partner

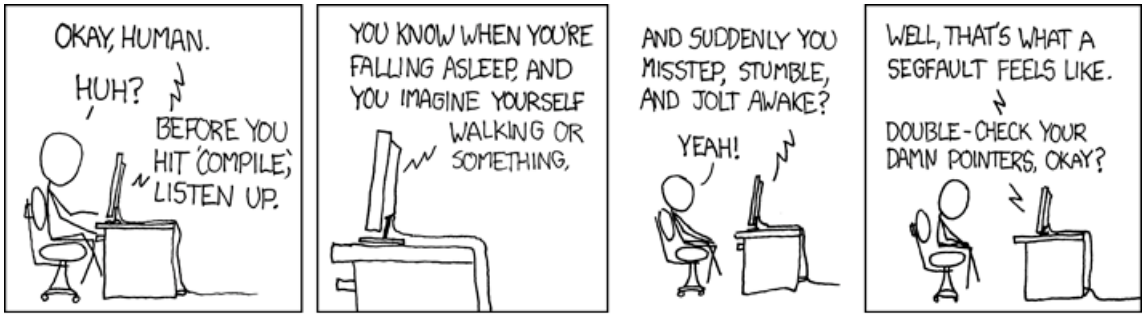
Computer Programming 143

Practical 6

2019

Aim of Practical 6:

- Introduction to pointers
- 2D Arrays
- Multiple files



Instructions

1. Attendance is **compulsory** for all the practical sessions of your assigned group. See the study guide for more details.
2. The last section (usually the last 30 minutes) of the practical will be used for a test.
3. If more than two tests have been missed for what ever reason, you will receive an **incomplete** for the subject. See the study guide for more details.
4. You must do all assignments **on your own**. Students are encouraged to help each other **understand** the problems and solutions, but each should write his/her own code. By simply copying someone else's code or solutions, you will not build an understanding of the work.
5. You are responsible for your own progress. Ensure that you understand the practical work. Check your work against the memorandum that will be posted on Wednesday afternoons on learn.sun.ac.za.
6. Use H:\CP143 as your Code::Blocks workspace folder for all projects. But it is highly suggested that you also use a **flash drive to backup** all your work.
7. Create a new project **for each assignment**. See *Creating a Code::Blocks Project* in Practical 0 for instructions on how to do this.
8. Include a comment block at the top of each source file according to the format given. It must include the correct filename and date, your name and student number, the copying declaration, and the title of the source file.
9. **Indent your code correctly.** Making your code readable is not beautification, it is a time and life saving habit. Adhere to the standards (refer to the documents on SUNLearn).
10. Comment your code sufficiently well. It is required for you and others to understand what you have done.

Question A

Goal: *Introduction to pointers.*

Getting started

1. Create a project named `Assignment6A`. Make sure that this is the active project in the workspace before compiling the program.
2. Include the **standard comment block** above your main function. Also, comment your whole program appropriately.

Program description

1. Declare an integer called **i**. Assign the value 5 to **i**.
   - Display the value of this integer variable as follows:
     `The value stored in the memory location labelled i is ?.`
     Note: The question mark indicates the variable value, which should be 5.
   - Display the address (as a decimal integer) of **i** as follows:
     `Address of memory location labelled i is ?.`
     Note: The question mark indicates the variable address, which will be specific to your program. Note that addresses are normally referred to in the hexadecimal number format, you should therefore cast the address to preferably `long` to display it in decimal integer format.
2. Display the following message:
   `REFERRING TO i THROUGH INDIRECT MEANS:`
   - Declare an integer pointer called **iPtr**. Refer to chapter 7 in your textbook and/or your class notes if you do not know how to do this. Assign the address of **i** to the pointer, **iPtr**.
   - Display the **value** of the integer variable by using the pointer, again with an appropriate message.
   - Display the **address** of the integer variable by using the pointer, again with an appropriate message.
3. Display the following message:
   `CHANGING i THROUGH INDIRECT MEANS:`
   - Now change the value stored at the memory location to which the pointer points. In point 1 you used the variable name to do that, now use the pointer.
   - Display the value stored at this memory location by using the variable name (the label). Use the message:
     `Value in memory labelled i has changed to ?  accessed with label.`
   - Display the value stored at this memory location by using the pointer. Use the message:
     `Value in memory labelled i has changed to ?  accessed with pointer.`

4. Display the following message:
   `CALCULATIONS WITH i THROUGH INDIRECT MEANS:`

   - Multiply the value stored at the memory location to which the pointer points by ten. Use the pointer to do this.
   - Display the value of the memory by using the variable name (the label). Use the message:
     `Value in memory labelled i has been multiplied by 10 to give ? (accessed with label).`
   - Display the value of the memory by using the pointer. Use the message:
     `Value in memory labelled i has been multiplied by 10 to give ? (accessed with pointer).`

5. Your program's output should look similar to the following listing. Note the question marks indicate the variable value or variable address, all of which will be specific to your program.

```
The value stored in the memory location labelled i is 5
The address of memory location labelled i is 2686776

============================================================

REFERRING TO i THROUGH INDIRECT MEANS:
The value stored in the memory location labelled i is 5
The address of memory location labelled i is 2686776

============================================================

CHANGING i THROUGH INDIRECT MEANS
Value in memory labelled i has changed to 100
accessed with label
Value in memory labelled i has changed to 100
accessed with pointer

============================================================

CALCULATIONS WITH i THROUGH INDIRECT MEANS:
Value in memory labelled i has been multiplied by 10 to give 1000
(accessed with label)
Value in memory labelled i has been multiplied by 10 to give 1000
(accessed with pointer)
```

6. Ensure that your code is indented correctly and that the {} braces are on the correct lines. Use the prescribed textbook as guideline.

7. Ensure that you copy the **Assignment6A** project folder to a flash drive as a backup.

Question B

Goal: *Consolidate various techniques to create a usable program.*

Getting started - different than before!

Before, you only had one file per question - you will now have several.

1. Create a new project named `Assignment6B`.
2. Make sure this is the active project in the workspace.
3. Click on **File → New → File...**. Select **C/C++ source**. Click **Next**, select **C** and click **Next**.
4. Under **Filename with full path**, select the directory where your project is stored and then type the filename **`Assignment6B.c`**. Select both **Debug** and **Release**. Click **Finish**.
5. There should now be a **main.c** as well as a **`Assignment6B.c`** file under **Sources** in your project workspace.
6. Repeat exactly the same process, but now create a header file named **`Assignment6B.h`**. To do this, select **C/C++ header** when the **New from template** window pops up.
7. There should now also be a header file **`Assignment6B.h`** under **Headers** in your project workspace.
8. Add the standard comment block to all these files you have created, with the correct file name and date, your name and student number, and the copying declaration. The comment block can be copied from your code for a previous assignment.

Program Overview

For this question you must write a program that will calculate and display the trajectory of a projectile, given the angle and speed at which it was launched. When dealing with larger projects it is good coding practice to divide the work into two or more files. In this question you will have to do the same. The `main.c` source file will only contain the `int main()` function, all other function bodies (function definitions) will be contained in `Assignment6B.c`. The header file `Assignment6B.h` must contain the function prototypes for all the functions written in `Assignment6B.c` and by including this header file in `main.c` you gain access to all of them.

Function descriptions

The following functions are needed in `main.c` (You must choose suitable names for them yourself). Write the prototypes in `Assignment6B.h` and the function bodies in `Assignment6B.c`:

1. **Calculate Trajectory:** A function that uses the initial speed and launch angle to calculate all the [x, y] points that the projectile moves though. Assume that the horizontal axis is [x], the vertical axis is [y] and that 1 character is equivalent to 1 meter for both. This function should receive as arguments the following: Speed,

Launch Angle and an array (call-by-reference) to store the calculated [y] data. Use 300 as an initial length of the data array. See "**Deriving equations for the trajectory**" below for more details on how to calculate the trajectory.

2. **Clear Display Data:** This function receives a 2D `char` array (that represents the "canvas" you will use to draw the trajectory on), clear it by setting all elements equal to [space] and then draw the borders. You my use the characters '-', '|' and '+' for the borders if you like.

3. **Update Display Data:** This function "draws" the trajectory data (calculated in the first function) on the 2D `char` array by marking all the points with 'X'. You may use another character if you wish.

4. **Print Data:** A function that outputs the data to the screen. Remember that you cannot use `printf()` to directly print the 2D array. Use nested `for`-loops to run through the data and print each character individually.

Main Function

The `main.c` file contains only 1 function: `int main()`. Your main function must do the following:

1. Ask the user to enter the initial speed and the launch angle
2. Clear the Display Data
3. Calculate Trajectory Data
4. Update the Display Data
5. Print the Display Data
6. Ask if the user wants to do this again, repeat steps 1 to 5 if yes.

`main.c` is also where you must declare the arrays that contain your Display Data and Trajectory Data

Regarding .h Files

When working with large programs, it becomes clumsy to store all your functions in one file. To prevent clutter, we store our functions in **libraries**. A library consists of a `.c` and a `.h` file of the same name. The **function prototypes** (**only** the function prototypes – for now) are stored in the `.h` file. The **function bodies** are stored in the `.c` files. Any other `.c` file that **includes the `.h` file** can now use the functions in that library. **Never `#include` a `.c` file!** By including the `.h` file, the `.c` file of the same name is automatically compiled.

If you created the header file correctly it will contain pre-processor commands. For example: `Assignment6B.h` should contain the following instructions:

```c
#ifndef ASSIGNMENT6B_H_INCLUDED
#define ASSIGNMENT6B_H_INCLUDED



#endif // ASSIGNMENT6B_H_INCLUDED
```

Your function prototypes must go into the space between the **`#define`** and the **`#endif`**. Remember to include this header file in `Assignment6B.c` and `main.c`. It would also be a good idea to define the WIDTH and HEIGHT of your Display Data array here. 80 for WIDTH and 40 for HEIGHT should work, but you are free to adjust this for the screen size you are working with.

Deriving equations for the trajectory

You can derive the formulas needed to calculate the trajectory yourself by using Newton's Laws of Motion. Assume that the only force exerted on the projectile is the Earth's gravitational pull. The two resultant dynamic equations can be integrated and combined with the initial release velocity to obtain equations describing the x and y positions of the projectile as a function of time t.

At any time t the projectile's horizontal and vertical displacement can be determined by

$$x = v_0 t \cos(\theta) \tag{1}$$

and

$$y = v_0 t \sin(\theta) - \frac{1}{2} g t^2, \tag{2}$$

with v_0 the projectile's initial velocity, θ the launch angle and g the standard gravitational acceleration (assume to be 9.81 $\mathrm{m \cdot s^{-2}}$). The projectile is assumed to be moving until it hits the ground (when $y = 0$ m at $t > 0$ s then $\dot{x} = \dot{y} = 0\ \mathrm{m \cdot s^{-1}}$).

Example output: