UNIVERSITEIT•STELLENBOSCH•UNIVERSITY
jou kennisvennoot • your knowledge partner

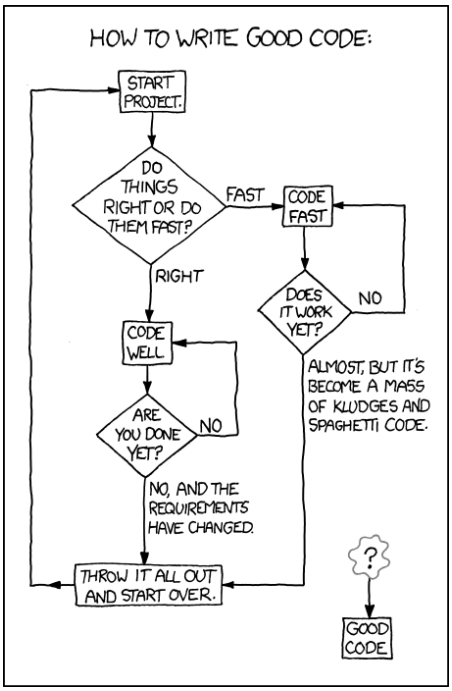
Computer Programming 143

Practical 7

2019

Aim of Practical 7:

- Accessing an array with a pointer
- Learn to use functions with call-by-value and call-by-reference



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 7A

Goal: *Using pointers and arrays together.*

Getting started

1. Create a project named `Assignment7A`. 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 the following variables: an array of 5 floating-points numbers (`double`) and a pointer to a `double`. Refer to chapters 6 and 7 in your textbook and/or your class notes if you do not know how to do this.
2. Assign suitable values to **ALL** the elements in the `double` array.
3. Display the following message:

   `REFERRING TO MEMORY BLOCK USING ARRAY NAME:`
4. Display each array element's value and address in the following format:

   `Value stored at index ?:  ___, Address at index ?:  ___`

   Display the array index instead of the question marks, and the value and address instead of the underscores. Both of these will be specific to your program.
5. Assign the starting address of the array to the `double` pointer.
6. Display the following message:

   `REFERRING TO MEMORY BLOCK BY USING THE POINTER WITH [] NOTATION:`
7. Display each array element's value and address (using the same print format as above) by **using the pointer as if it were an array**. Use subscript notation, e.g. square brackets, to access each element.
8. Display the following message:

   `REFERRING TO MEMORY BLOCK BY USING THE POINTER WITH * NOTATION:`
9. Display each array element's value by **dereferencing the pointer**, and its address using the pointer directly (using the same print format as above). Make use of the `for` loop counter to move to next index.
10. Display the following message:

    `REFERRING TO MEMORY BLOCK BY USING THE POINTER PASSED TO FUNCTION:`

11. Write the function with the following function definition:

```
void passPointer(double *start);
```

    Call this function from the main with the pointer which contains the starting address of the array. This function prints the elements within the array by dereferencing and incrementing the pointer (using the same print format as above).

12. Your program's output should look similar to the following. Note the variable value and address will be specific to your program.

```
REFERRING TO MEMORY BLOCK USING ARRAY NAME:
At index 0: Value = 1.00, address = 2686736
At index 1: Value = 2.20, address = 2686744
At index 2: Value = 3.50, address = 2686752
At index 3: Value = 6.50, address = 2686760
At index 4: Value = 7.80, address = 2686768

REFERRING TO MEMORY BLOCK BY USING THE POINTER WITH [] NOTATION:
At index 0: Value = 1.00, address = 2686736
At index 1: Value = 2.20, address = 2686744
At index 2: Value = 3.50, address = 2686752
At index 3: Value = 6.50, address = 2686760
At index 4: Value = 7.80, address = 2686768

REFERRING TO MEMORY BLOCK BY USING THE POINTER WITH * NOTATION:
At index 0: Value = 1.00, address = 2686736
At index 1: Value = 2.20, address = 2686744
At index 2: Value = 3.50, address = 2686752
At index 3: Value = 6.50, address = 2686760
At index 4: Value = 7.80, address = 2686768

REFERRING TO MEMORY BLOCK BY USING THE POINTER PASSED TO FUNCTION:
At index 0: Value = 1.00, address = 2686736
At index 1: Value = 2.20, address = 2686744
At index 2: Value = 3.50, address = 2686752
At index 3: Value = 6.50, address = 2686760
At index 4: Value = 7.80, address = 2686768
```

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

14. Ensure that you copy the **Assignment7A** project folder to a flash drive as a backup.

Question 7B

Goal: *Functions, pointers and arrays.*

Getting started

1. Create a project named `Assignment7B`. 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

This program calculates the torque due to a certain force at a specific point. Torque is the twist or rotation about an axis that an applied force can have on an object. It can be considered as the resulting moment of force, the result of applying a force at a distance about a pivot point. The formula for torque is given by:

$$\tau = r \times F$$

where
τ = torque vector[N.m],
r = position vector (moment arm)[m]
F = force vector [N].

Note that this is not normal multiplication but a cross product. These terms are **NOT** scalars, they are vectors and thus have a magnitude **AND** a direction (see Figure 1).

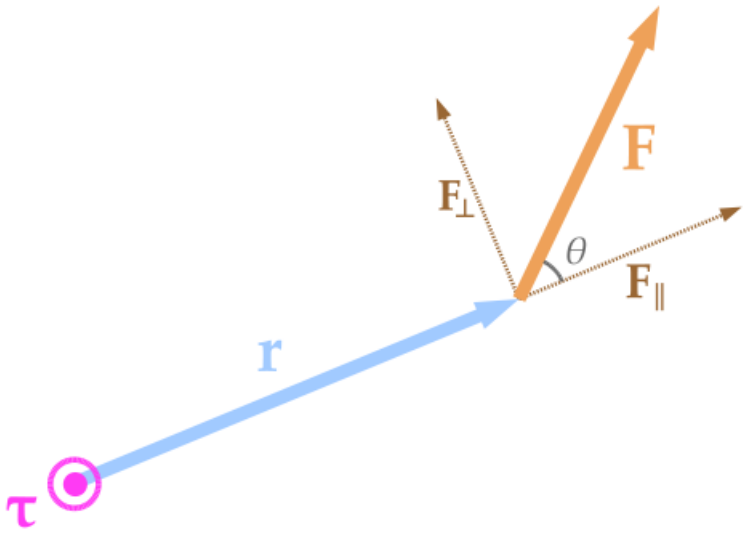


Figure 1: Torque diagram - showing the three vectors involved when calculating torque. Note the directions. The torque's direction is coming out of the page.

The torque vector is determined by taking the cross product of the position vector and the force vector. The resulting torque's **direction** is always perpendicular to the plane that is formed by the other two vectors. The resulting torque's **magnitude** is equivalent to the product of the other two vectors' magnitude AND the sin of the angle that these two form with each other in their plane. In other words only the force component that is perpendicular to the position vector causes torque.

A cross product between two vectors can be expressed in determinant form (see Equation 1). This will help us to retain the direction. We use **i**, **j**, **k** unit vectors to indicate direction, where **i** indicates direction along the x-axis, **j** along the y-axis and **k** along the z-axis.

$$\mathbf{r} \times \mathbf{F} = \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ r_x & r_y & r_z \\ F_x & F_y & F_z \end{vmatrix} = \begin{vmatrix} r_y & r_z \\ F_y & F_z \end{vmatrix} \mathbf{i} - \begin{vmatrix} r_x & r_z \\ F_x & F_z \end{vmatrix} \mathbf{j} + \begin{vmatrix} r_x & r_y \\ F_x & F_y \end{vmatrix} \mathbf{k}$$

In expanded form it is:

$$\mathbf{r} \times \mathbf{F} = (r_y F_z - r_z F_y)\mathbf{i} - (r_x F_z - r_z F_x)\mathbf{j} + (r_x F_y - r_y F_x)\mathbf{k} \tag{1}$$

Write a program to do the following:

1. In your main function, define a 2D array of floating-point values (consisting of 2 rows and 3 columns). The first row holds the components for the position vector and the second row the components for the force vector. Thus the first column contains the x-components, the second column the y-components and the last column the z-components of the position and force vectors. The column position of a value in the array indicates its direction. By using the array like this we have it in the **i**, **j**, **k** format.

2. Write a function that asks the user to enter values for the different components of the position and force vectors (**HINT**: use `char` comp[3]='x','y','z'; and a loop to ask for the user input) and load these values into the 2D array. Use the following prototype:

   ```
   void getPositionForce(float matrix[][3]);
   ```

   The function returns nothing. This function will request a value for each index of the 2D array from the user. Remember the first row must be the position vector values and the second row must be the force vector values. The first column contains the x-component, the second column the y-component and the third column the z-component. Call this function with the correct arguments from your main function.

3. In the main function, define a 1D array of floating-point values of size 3. This array holds the values for the torque vector.

4. Write a function that calculates the torque from the values that have been loaded in the 2D array. Its prototype is:

```
void calcTorque(float posForce[][3], float torqueArr[]);
```

   Note that no sizes are sent through to this function. Can you think why? Use the formula given in (1) to calculate the torque. For each torque component it is required to multiply and subtract the correct elements in the 2D position and force array. The x-component of torque is loaded into the first element of `torqueArr`, the y-component into the second and the z-component into the third.
   Note: make sure the signs are correct! Call this function with the correct arguments from your main function.

5. Write a function that will display the calculated torque. Use the following prototype and call this function with the correct arguments from your main function:

```
void displayTorque(float torque[]);
```

6. Your program's output should look similar to the following. Note the values will be specific to your program.

```
Enter the x, y and z components for the position vector in [m].
Position Vector: Enter the x component:
(it will be stored in element [0][0])
3
Position Vector: Enter the y component:
(it will be stored in element [0][1])
4
Position Vector: Enter the z component:
(it will be stored in element [0][2])
5
Enter the x, y and z components for the force vector in [N].
Force Vector: Enter the x component:
(it will be stored in element [1][0])
1
Force Vector: Enter the y component:
(it will be stored in element [1][1])
2
Force Vector: Enter the z component:
(it will be stored in element [1][2])
3
The resulting torque vector is in [N.m]
Torque Vector: The x component is 2.000000
Torque Vector: The y component is -4.000000
Torque Vector: The z component is 2.000000
```

7. Ensure that your code is indented correctly.

8. Ensure that you copy the **Assignment7B** project folder to a flash drive as a backup.