UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Department of Computer Science
COS132 - Imperative Programming
Practical 6

1 Introduction

Deadline: 21st of May, 20:30

1.1 Objectives and Outcomes

The objective of this practical is two-fold. The first is to introduce the programming concept of functions, which are vital towards building expansive and reusable programs. The second is to introduce the concept of arrays which are a data structure of great use towards structuring and organising large collections of data.

1.2 Structure of the Practical

This practical will consist of 2 tasks and you will be required to complete all of them as part of this practical. You are also advised to consult the Practical 1 specification for information on aspects of extracting and creating archives as well as compilation if you need it. Also consult the provided material if you require additional clarity on any of the topics covered in this practical.

1.3 Submission

Submit your code to Fitchfork before the closing time. Students are **strongly advised** to submit well before the deadline as **no late submissions will be accepted**. Also note that file names are case sensitive. Failure to name the files as specified will result in no marks being awarded.

1.4 Plagiarism

The Department of Computer Science considers plagiarism as a serious offence. Disciplinary action will be taken against students who commit plagiarism. Plagiarism includes copying someone else's work without consent, copying a friend's work (even with consent) and copying textual material from the Internet. Copying will not be tolerated in this course. For a formal definition of plagiarism, the student is referred to **http://www.ais.up.ac.za/plagiarism/index.htm** (from the main page of the University of Pretoria site, follow the *Library* quick link, and then click the *Plagiarism* link). If you have questions regarding this, please ask one of the lecturers, to avoid any misunderstanding.

1.5 Mark Distribution

Activity	Mark
Arrays	10
Functions Basics	15
Total	**25**

2 Practical Activities

2.1 Task 1: Arrays

For this task, you are required to provide files called **array.cpp** as well as a makefile to compile and run it. In your file, array.cpp, should have a skeleton of a main program as per normal which you will then fill it in as per the following.

The objective of this activity is to demonstrate the basics of array use, in terms of how to use declare and use them for programs.

You will first need to declare an array, called bins. This array should be an integer array and be declared to have a size of 5. During your initialisation, you should initialise each index in the array to 0.

Now, you will need to read in some values from a file called **values.txt**. In this file, there will be a number of lines. Each line will be a comma separated list of integers. The first column, column 0, is associated with index 0 in the bins array and so on for the rest of the columns.

You must total up the columns, and store the result in each array element. So the sum of the first column will be stored in index 0 in the array and so on for the rest of the columns.

The file will have a format as below (for example):

```
1,2,3,4,5
2,4,5,6,7
3,6,7,8,9
```

Once totalled, you then need to print out the results of each index one, per line, with a new line at the end of each output. The result will be:

```
6
12
15
18
21
```

2.2 Task 2: Functions

For this task, you are required to provide files called **fun.cpp** as well as a makefile to compile and run it. In your file, fun.cpp, should have a skeleton of a main program as per normal which you will then fill it in as per the following.

The objective of this activity is to demonstrate the basics of functions and their use. For this activity, you will need to declare your functions alongside your main program skeleton. You will use these functions inside your main program.

You will need to declare a function: **echo**. The echo function returns a string, and receives one as an input. Set the default value of this input variable as "echo". The echo function merely receives its input and returns it.

Once **echo** has been defined, you are going to perform a function overloading of this function. Function overloading lets one reuse a function name, but change the input variables so that the same function can be used under different conditions. Therefore, define another echo function. This is declared as before, but instead of taking a single string variable, this one takes a string variable and an integer variable. This will not have any default values.

In this function, you will still return the input of the string that is passed into the function. If the number passed in is an even number (which includes 0 in this case), then it is simply appended to the string. If it is an odd number, then it is put in front of the passed in string.

The main program of your file, will work as follows. At the start of the program you will read from a file called **values.txt**. This file will have string lines, each on a separate line. You will then pass these strings, one line at a time, into the echo functions and print the results. The original echo function should be called first (and output first) with the second function called immediately after. Additionally, the second echo function should also be passed the line number (starting from 0) that the line they has. Each line should be printed on a new line.

For example, given the following:

```
this
is
a
sentence
```

The output (for the first line) will be:

```
this
this0
```

3 Submission Requirements

Your submission requirements for this task are:

- makefile
- array.cpp
- fun.cpp
- values.txt

The following libraries are recommended:

- fstream
- sstream
- string

By default the use of c++11 functions is not enabled. Please take this into account when designing your programs.

Please note that marks will be subtracted from programs that have too many lines output. You are advised to make use of while (getline (..)) to read your files and use an if statement to check if the files actually exist before attempting to open them.

You will have a maximum of 10 uploads for this practical. One submission will evaluate both tasks.