UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Department of Computer Science
COS132 - Imperative Programming
Practical 5

1 Introduction

Deadline: 7th of May, 20:30

1.1 Objectives and Outcomes

The objective of this practical is to introduce more advanced looping techniques, such as the for loop. The purpose is of course to expand the available techniques to construct more advanced programs. Additionally, the practical introduces files as a source of data for programs and manipulating them.

A for loop is a similar type of loop to the while. It is however more structured and specific in terms of quantifying the number of iterations it is required to perform before time. Whereas the while loop is designed to handle situations where the number of loops is difficult (or impossible) to determine beforehand, the for loop is for those times when the number of iterations is easily available.

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

Task	**Mark**
For Loop	10
File Reading	10
Total	**20**

2 Practical Activities

2.1 Task 1: For Loop

For this task, you are required to create and submit a file called **for.cpp** as well as a makefile that compiles and runs it.

Inside **for.cpp**, outside of the standard shell of a main program, you are going to design a program that uses a for loop to perform some iterations. At the start of your program, prompt the user to enter in a symbol and a number of iterations. The symbol can take the form of a +,- or * . The structure of this input will be comma separated. So the first value will be an symbol, followed by a comma, and then the integer. For example,

```
*,3
```

Your loop must apply the operation provided in the input to a running total that is initialised as the number provided in the original input. At each iteration, apply the operation given to the total, using the original number as the second term. For example, given *,3 you will need to multiply an initial 3 by 3, 3 times. Output this outcome with the message: "Result: X" where X is the outcome of the calculation. For example:

```
Enter a pair: *,3
Result: 81
Enter a pair: 2142,242
Result: 0
```

If the input does not match the format requirements, your result should be 0. You need to prompt the user for two inputs. Process their first input and display the result and then prompt them for a second input.

2.2 Task 2: File Reading

For this task, you are required to create and submit a file called **read.cpp** as well as a makefile to compile and run it. You need not worry about specifying directories. The input file will always be included in the directory where the program is compiled and run.

The objective of this activity is to demonstrate file reading and manipulation.

In this text file you will have many rows of single numbers. You will be reading and processing this information, line by line.

When your program starts, it must start reading the file named **lines.txt**, line by line, from the top to its end. However, if any of the lines have the word "stop" then you should halt in the process and print "File reading stopped". If there are no lines with stop (lower case only), you should read the entire file.

On each line, read in the number on that line, and then add the number that was above it in the row index. The first number is exempt from this. As you process the file, it will print out as many lines of output, as there are rows in the file. Note that the number of rows in the file is not specified beforehand and begins at 0.

After processing the first file, the second file, **lines2.txt**, should then be read and processed by the same process as for the first file.

For example, if the **lines.txt** file looks like this:

```
1
2
3
4
```

then an example execution will look like this:

```
1
3
5
7
```

3 Submission Requirements

Your submission requirements for this task are:

- makefile
- for.cpp
- read.cpp

- lines.txt
- lines2.txt

The following libraries are recommended (but not strictly required):

- fstream
- sstream
- string

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