Practical 2

COS132

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA
Denkleiers • Leading Minds • Dikgopolo tša Dihlalefi

Department of Computer Science

Official Deadline: 08/03/2024 at 17:30:00

Extended Deadline: 10/03/2024 at 20:00:00

Marks: 25

1 General instructions:

- This practical should be completed individually; no group effort is allowed.
- Be ready to upload your practical well before the deadline, as no extension will be granted.
- **The extended deadline has been put in place in case of loadshedding or other unforeseen circumstances. No further extension will be granted.**
- You may not import any libraries that have not been imported for you.
- If your code does not compile, you will be awarded a zero mark. Only the output of your program will be considered for marks, but your code may be inspected for the presence or absence of certain prescribed features.
- All submissions will be checked for plagiarism.
- Read the entire practical before you start coding.
- You will be afforded 15 upload opportunities.
- **You have to use C++98 in order to get marks**

2 Plagiarism

The Department of Computer Science considers plagiarism 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 permission) and copying 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.library.up.ac.za/plagiarism/index.htm. **If you have any questions regarding this, please ask one of the lecturers to avoid misunderstanding.**

3 Outcomes

The objective of this practical is to introduce the basic elements that comprise programs. Variables, of various data types, arithmetic operations, mathematical expressions are all the basic elements from which all programs are constructed. This practical consists of 2 tasks. You are 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

4 Structure

This practical consists of two tasks. Each task is self-contained and all of the code you will require to complete it will be provided in the appropriate Task folder. Each task will require you to submit a separate archive to an individual upload slot. That is, each separate task will require its own answer archive upload. You will upload Task 1 to Practical 2 Task 1 and so on.

5 Mark Distribution

Activity	**Mark**
Task 1 - Faulty Code	14
Task 2 - Data Types and Operations	16
Total	**25**

Table 1: Mark Distribution

6 Resources

Here are some additional resources you can use to help you complete the practical

- Data types in C++: `https://www.w3schools.com/cpp/cpp_data_types.asp`
- Constant variables in C++: `https://www.w3schools.com/cpp/cpp_variables_constants.asp`
- Basic operators, including Arithmetic Operators: `https://www.w3schools.com/cpp/cpp_operators.asp`
- Something More Advanced - Integer Division: `https://linuxhint.com/integer-division-cpp/`

7 Tasks

7.1 Task 1

Your task is to fix the code given to you in the file called task1.cpp, in the task1 folder. The code is given below for your perusal. Pay special attention to the instructions below on how to "fix" the code: running code does not guarantee full marks. **Do not modify the cout statements or any code above line 19**

Listing 1: Task 1 - Faulty Code

```
#include <iostream>
#include <string>
// DO NOT MODIFY THE CODE BEFORE LINE 15
void checkType(int) {
    std::cout << "int" << std::endl;
}
void checkType(float) {
    std::cout << "double" << std::endl;
}
void checkType(double) {
    std::cout << "double" << std::endl;
}
void printSection() {
    std::cout << "###" << std::endl;
}

int main() {
//DO NOT CHANGE ANY CODE ABOVE THIS LINE
        printSection();
        int a = 2.5;
        std::cout << a << std::endl;
        checkType(a);

        printSection();

```

```
        int b = 3.7;
        std::cout << b << std::endl;
        checkType(b);

        printSection();

        std::string str = 'AssignMe';
        std::cout << str << std::endl;

        printSection();

        int additionResult = 2 - 2;
        std::cout << additionResult << std::endl;

        printSection();

        int denominator = 1;
        denominator--;
        int divisionResult = 2 / denominator;
        std::cout<<divisionResult<<std::endl;

        printSection();

        float numerator = 2.5;
        int faultyDivision = numerator / 1.6;
        std::cout<<faultyDivision<<std::endl;
        checkType(faultyDivision);

        printSection();

        int numerator2 = 5;
        int denominator2 = 2;
        float divisionResultFloat = numerator2 / denominator2;
        std::cout << divisionResultFloat << std::endl;
        checkType(divisionResultFloat);

        printSection();

}
```

Listing 1: Task 1 - Faulty Code

Instructions to fix faulty code:

- **Do not remove the printSection() lines**.
- Variable a: Modify the value so that it fits appropriately with the variable type (do not modify the variable type)
- Variable b. Modify the variable type so that it fits appropriately with the value (do not modify the value)

- Variable str: Fix the syntax error
- Addition Calculation: Fix the code such that the addition operation is used
- Division Calculation: The goal is to have the division result in 1 (devisionResult should be 1). The current operation on denominator leads to a division by zero. Adjust the operation on denominator (line 9: denominator–) to achieve the desired division result.
- Faulty Division: Modify the variable type for variable faultyDivision so that it prints out the decimal values
- The division numerator2 / denominator2 results in integer division, which means the fractional part of the result is discarded. The goal is to perform a floating-point division so that the result includes the fraction. Adjust the code to correctly reflect floating-point division. (See Integer Division under Resources)

7.2 Task 2

You are a software engineer tasked with setting up a basic inventory management system for a local café named "Code Brew". The café wants to digitalize their inventory using C++ to keep track of their supplies and sales.

7.2.1 Part 1: Setting Up the Inventory

NB: Do not change any cout (printing) statements in the first part of this practical. Work in the denoted portions of the task2.cpp file.

Inventory Initialization: Initialize the following variables to simulate the café's inventory system.

- A string variable named bestSellingItem storing the name of the café's best-selling item: "Caramel Macchiato".
- A char variable named cupSize representing the size of cups ('L' for large).
- Two int variables for tracking the number of coffee beans bags (50) and milk cartons (30) in stock, named beanStock and milkStock respectively.
- Two bool variables, the first indicating if the café is open (true), and the second indicating if a restock is needed (false), named cafeOpen and restockNeeded respectively.
- A const float variable representing the café's daily sales in rands, set to 1234.50, named sales.

Once you have declared and initialised the variables, run your code along with the given cout statements to make sure that everything is working. Do not make changes to the cout statements. The output of this should be:

```
###
Best-selling item: Caramel Macchiato
Cup size for best-seller: L
Number of coffee bean bags in stock: 50
Number of milk cartons in stock: 30
Cafe open: 1
Restock needed: 0
Sales in rands: R1234.5
###
```

7.2.2 Part 2: Daily Inventory Calculation

Inventory Log: During the day, inventory figures need to be updated. The café starts with initial inventory that needs to be updated as orders are placed

- At the start of the day, the café always has enough beans for 10 espresso shots. Declare this in an integer variable called espressoShots.
- An espresso shot has a cost price of R5.50. Declare this in a float value called espressoCost.

- One part milk has a cost price of R3. Declare this in an integer value called milkCost.
- A cappuccino will always take 2 espresso shots and 3 parts milk. Declare a constant integer variable called cappuccinoShots initialised to 2, and a constant integer variable called cappuccinoMilk initialised to 3.
- Calculate the cost price of a cappuccino's total espresso shots and save this to a variable called cappuccinoShotCost (Hint: use multiplication and the variables cappucinoShots and espressoCost.) *Think carefully about whether cappuccinoShotCost should be an integer or a float.*
- Calculate the cost price of a cappuccino's total milk parts and save this to a variable called cappuccinoMilkCost (use multiplication and the variables cappuccinoMilk and milkCost)
- Calculate the total cost price of a cappuccino (assuming it only needs the espresso and milk) and save it to a variable named cappuccinoTotalCost. (Hint: use addition and the variables cappuccinoShotCost and cappuccinoMilkCost) *Think carefully about whether cappuccinoTotal-Cost should be an integer or a float.*
- Calculate the number of espresso shots remaining if 3 cappuccinos were ordered. (Hint: use the variables cappuccinoShots and espressoShots and the subtraction and multiplication operators) Save the answer to espressoShots.
- The cafe spent R100 worth of inventory. How many cappuccinos did they sell? Save the answer to a variable called totalSales. (Hint: Use cappuccinoTotalCost and the division operator)
- Print out the value of espressoShots on a new line
- Print out the value of espressoCost on a new line
- Print out the value of milkCost on a new line
- Print out the value of cappuccinoShots on a new line
- Print out the value of cappuccinoMilk on a new line
- Print out the value of cappuccinoShotCost on a new line
- Print out the value of cappuccinoMilkCost on a new line
- Print out the value of cappuccinoTotalCost on a new line
- Print out the value of totalSales on a new line
- There should be a newline after each printed value including totalSales

8 Submission checklist

For Task 1:

- Archive (zip) task1.cpp and rename the archive uXXXXXXXX.zip where XXXXXXX is your student number
- Upload the archive to Fitchfork Practical 2 Task 1 before the deadline
- **Ensure that if you download your code from the online compiler that you rename the file to task1.cpp before archiving and uploading**

For Task 2:

- Archive (zip) task2.cpp and rename the archive uXXXXXXXX.zip where XXXXXXX is your student number
- Upload the archive to Fitchfork Practical 2 Task 2 before the deadline
- **Ensure that if you download your code from the online compiler that you rename the file to task2.cpp before archiving and uploading**