Department of Computer Science

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

COS132 2015
Practical 2

Ms Mary Adedayo

Contents

1 Introduction **2**
1.1 Submission . . . 2
1.2 A Serious Warning . . . 2

2 Getting started **2**
2.1 Download and extract . . . 2
2.2 Compiling Program and Creating tarballs to upload . . . 3

3 Programming Tasks **3**
3.1 Task 1: Basics and Syntax Errors [6] . . . 3
3.2 Task 2: Binary to Decimal Converter [12] . . . 4
3.3 ASCII Converter [6] . . . 4
3.4 Task 4: Income Tax Calculator [18] . . . 5
3.5 Task 5: Formatting Output [8] . . . 6
3.6 Task 6: Challenge Activity - Complex Binary Converter . . . 6

4 Finishing Off **7**

5 Mark Distribution **7**

1 Introduction

1.1 Submission

The submission deadline for this practical is **23 February 2015 at 07:30 AM**. Make sure that you have submitted your code to Fitchfork by then. **No late submissions will be accepted**. Remember that Fitchfork marks very strictly and is reliant on your output to comply with the expected output to a high degree. Follow the instructions in this document **precisely**. Also remember that names of source files and executable files are case sensitive.

1.2 A Serious Warning

It is in your own interest that you, at all times, act responsible and ethically. As with any work done for the purpose of your university degree, remember that the University of Pretoria will not tolerate plagiarism. Do not copy a friend's assignment or allow a friend to copy yours. Doing so constitutes plagiarism, and apart from not gaining the experience intended, you may face disciplinary action as a result. For more on the University of Pretoria's plagiarism policy, you may visit the following `http://www.library.up.ac.za/plagiarism/index.htm`

2 Getting started

2.1 Download and extract

1. If you do not already have a sub-directory in your `LinuxShare` directory called `COS132`, create such directory.
2. Download `Practical2.tar.gz` from the CS website and save it in the `COS132` directory.
3. Open a terminal and navigate (using the cd command) to the `COS132` directory or the directory containing the archive.
4. Extract the content of the `Practical2.tar.gz` archive with the following command.
   `tar -f Practical2.tar.gz --extract`
5. After extracting this archive your LinuxShare directory should contain (possibly among some other files and directories) the files and directories shown in the following hierarchical structure. You are advised to create directories for other tasks.

```
LinuxShare
└── COS132
    ├── Practical2.tar.gz
    └── Practical2
        └── Task1
            └── basics.cpp
```

2.2 Compiling Program and Creating tarballs to upload

You are required to upload your solutions to the programming tasks to fitchfork. In each case you will have to create a tarball containing the appropriate .cpp file. Follow these steps in the command terminal to create the tarball and upload your solution for Task1:

- Verify that your working directory is `Task1`: Your prompt should end in `Task1$`
- Verify that `basics.cpp` is in the directory: The file name should appear if you issue the `ls` command.
- Verify that the program compiles. You can do this using the command:
  `g++ basics.cpp -o prac2task1.out`
- Verify that the correct executable was created; if you type `./prac2task1.out` the expected output should appear.
- Create the `tarball` by typing the following command:
  `tar -cvz basics.cpp -f Prac2Task1.tar.gz`
- Upload the created file called `Prac2Task1.tar.gz` on the CS website.

NOTE: If you make changes to the .cpp for subsequent tries, you should save the .cpp file and create a new tarball for the new upload for the changes to take effect. It is advisable to give the new tarball a new name (for example `Prac2Task1Try2.tar.gz`) to make it easier to pick the latest version when uploading. It also keeps your previous version intact if, for some reason, you would like to revert to it.

3 Programming Tasks

3.1 Task 1: Basics and Syntax Errors [6]

You are given the file: `basics.cpp` in the archive that you extracted. You are expected to edit basics.cpp and fix all of the errors. The file should compile without any errors. Refer to section 2.2 above if you are unsure of how to do this. If your program has compiled successfully, run your compiled program using the `./prac2task1.out` command to make sure that it and executes successfully. The output will look similar to this:

```
--------------------------------------------------
This program divides the first number entered by the user with the second one.
Enter an integer number:  3
Enter another integer number:  4
--------------------------------------------------
The first number is an integer and the value is 3
The second number is an integer and the value is 4
The result of 3/4 is 0.75
I fixed the problem with the char and the value is U
I fixed all the errors!
```

When you have finished, create a tar archive containing basics.cpp (refer to section 2.2 for how to do this) and upload it to Fitchfork on the CS website, using the Practical 2 Task 1 upload link. You will have 10 uploads for this task. Test your work on your lab PC and upload it only when it compiles and runs the way it should before uploading to Fitchfork. Refer to Section 2.2 for more information about tarballs and uploading.

3.2 Task 2: Binary to Decimal Converter [12]

Write a program that converts a binary number[1] consisting of five digits to the corresponding decimal number. The program should prompt the user to enter the five digits (0 or 1), one at a time. The values entered must be stored so that the program can use them to calculate the decimal value. The corresponding decimal value should be outputted in a full sentence.

To calculate the decimal number, if we assume that the five digits entered by the user are b_4, b_3, b_2, b_1, b_0, representing the binary number $b_4b_3b_2b_1b_0$, the corresponding decimal number d is given by,

$$d = b_4 \times 2^4 + b_3 \times 2^3 + b_2 \times 2^2 + b_1 \times 2^1 + b_0 \times 2^0$$

The digits of the binary number (0 or 1) must be entered with five separate prompts. The output should contain the complete information. Note that the prompts for the user should end with a colon and a space. The user input has to appear next to the prompt and not below it.

Below is a sample run of the program. User inputs are in boldface:

```
Enter first digit of the binary number:  0
Enter second digit of the binary number:  1
Enter third digit of the binary number:  0
Enter fourth digit of the binary number:  1
Enter fifth digit of the binary number:  1
The binary number is:  01011
The corresponding decimal number is:  11
```

Your source code file must be called **binaryConverter.cpp**. Compile and test that your program works as expected. When you have finished, create a tar archive containing your source code file and upload it to Fitchfork on the CS website, using the Practical 2 Task 2 upload link. You have 10 uploads for this task.

3.3 ASCII Converter [6]

Write a program to ask the user to enter a character and to output the ASCII value of the character in a full sentence. A sample run of the program is shown below with the user inputs in bold:

[1] A binary number is a sequence of one or more binary digits. A binary digit or bit can take a value of either 0 or 1 only.

```
Enter a character:  C
The ASCII value of C is 67
```

Note that the prompt for the user should end with a colon and a space. The user input has to appear next to the prompt and not below it. You should test your program with different input values.

Your source code file must be called **asciiConverter.cpp**. Compile and test that your program works as expected.

Create a tar archive containing your program. Upload this tarball to Fitchfork on the CS website, using the Practical 2 Task 3 upload link. You have 10 uploads for this task.

3.4 Task 4: Income Tax Calculator [18]

Assume that salary earners pay income tax on 80% of their gross income, which is the taxable income. Tax is calculated as 18% of the taxable income. The tax value is deducted from the gross income to get the net income of a person. For example, if someone has gross income of R10000, the taxable income is R8000. The tax paid is R1440 and the net income of the person is R8560. Write a program that can be used to calculate the net income of a person. Your program should declare a constant defining the income tax as 18%. The program should use a descriptive prompt to ask for the name of the person and the gross income (with separate prompts). Assume that the name of a person include at least the first name and the last name, separated by a space but entered on the same line – i.e. the name should be entered in a single variable and its value may contain spaces and punctuation. Further assume the currency is in Rand. All monetary values must be formatted to 2 decimal places.

The program must produce the following output each on a separate line and in sentences:

- a line of **exactly** 51 dashes directly after the prompts (Hint: use setfill).
- the name of the person.
- the gross income (provided by the user).
- the taxable income (80% of the gross income).
- the tax paid (18% of the taxable income).
- the net income (amount left after tax is removed from the gross income).

Your source code file must be called **taxCalculator.cpp**. Compile and test that your program works as expected.

Create a tar archive containing your source code and upload it to Fitchfork on the CS website, using the Practical 2 Task 4 upload link. You will have 10 uploads for this task. The maximum mark assigned by fitchfork is 18.

3.5 Task 5: Formatting Output [8]

Write a program that prompts the user to input a decimal number and outputs the number in the following ways, each on a separate line.

- the number to exactly two decimal places.
- the number to the nearest integer.
- the integer part of the number in a right justified field of 12 spaces.

The prompt should be clear and the output should contain complete information. Fitchfork expects four lines of output. The first contains the prompt and the others contain the mentioned outputs as listed above.

Your source code file must be called **formatting.cpp**. Compile and test that your program works as expected.

Create a tar archive containing the file containing your source code and upload it to Fitchfork on the CS website, using the Practical 2 Task 5 upload link. You have 10 uploads for this task.

3.6 Task 6: Challenge Activity - Complex Binary Converter

> ***Challenge***
>
> Note that this task is **NOT** part of the assignment. It is given as a challenge to those who find our usual assignments too trivial or have completed the required part of the assignment. It does not contribute to your marks. It is not expected from the Teaching Assistants to assist you to complete the assignment. It is intended for you too figure it out on your own.

Write a program that converts a binary number consisting of any number of digits to the corresponding decimal number. The program should prompt the user for a binary number and the corresponding decimal value should be outputted in a full sentence. Your source code file must be called **complexBinaryConverter.cpp**. Compile and test that your program works as expected.

Create a tar archive containing your source code and upload it to Fitchfork on the CS website, using the Practical 2 Challenge upload link. There is no marks assigned for completing this task.

Hint: To calculate the decimal number, if we assume that there are n digits in the binary number entered by the user then the corresponding decimal number d is given by,

$$d = b_{n-1} \times 2^{n-1} + b_{n-2} \times 2^{n-2} + \ldots + b_2 \times 2^2 + b_1 \times 2^1 + b_0 \times 2^0$$

where b_{n-1} is the first digit from the left of the binary number and b_0 is the last.

4 Finishing Off

To avoid abuse of your account, you are advised to log off before leaving the lab. Since other people using the computer afterwards will probably not use Linux, it is always a good idea to restart the computer. It will then automatically boot Windows.

Activity

1. Make sure you completed all activities.
2. Log off.

5 Mark Distribution

Activity	Mark
Task 1	6
Task 2	12
Task 3	6
Task 4	18
Task 5	8
Total	**50**