

# Lab 3 - Work Harder, Not Smarter

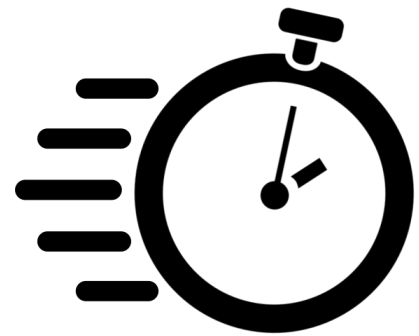
## Introduction

Frederick Taylor is widely considered to be the founder of what we call today *scientific management*. According to Taylor himself,



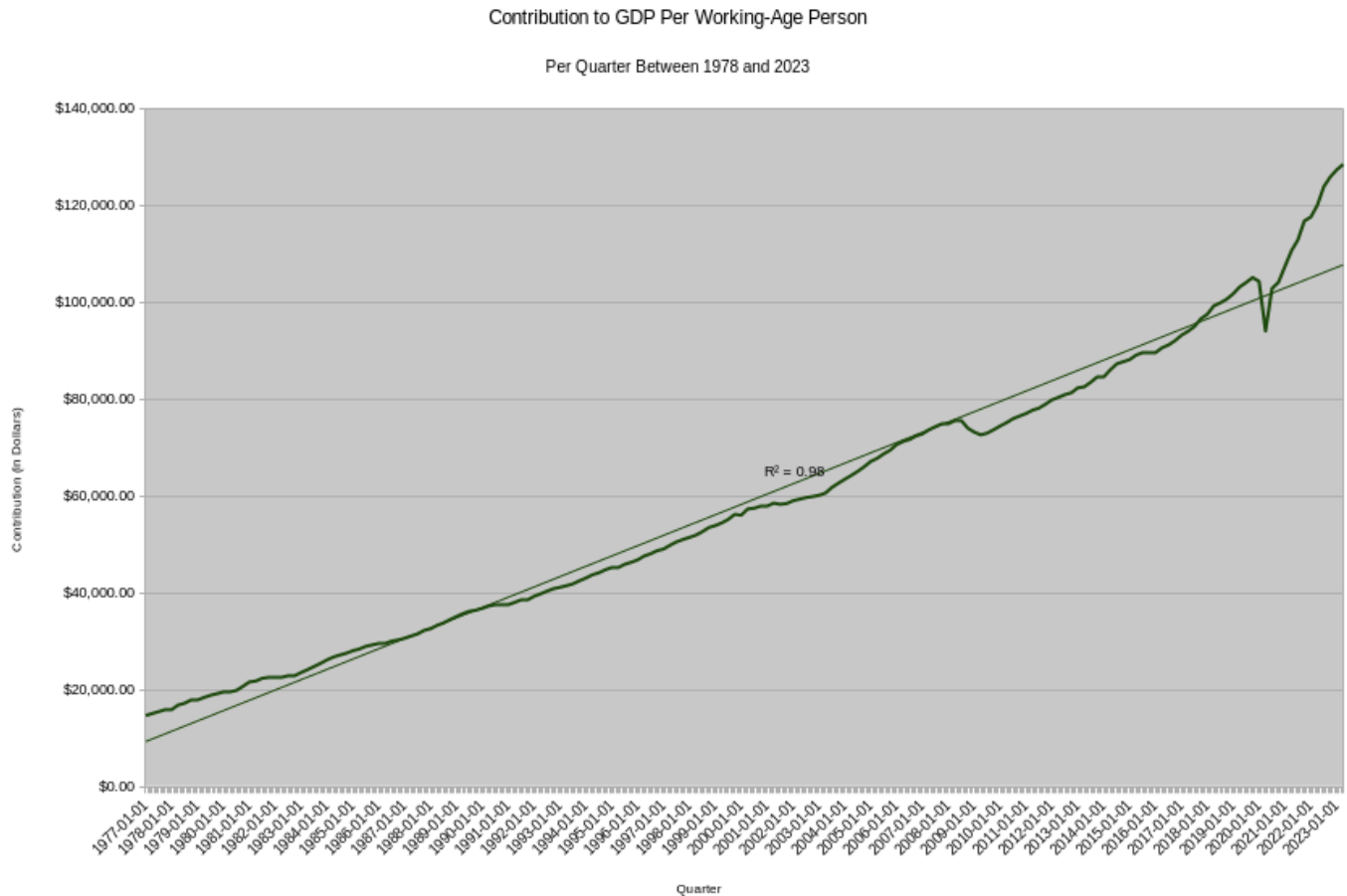
Scientific management ... has for its very foundation the firm conviction that the true interests of the [employer and employee] are one and the same; that prosperity for the employer cannot exist through a long term of years unless it is accompanied by prosperity for the employee, and vice versa; and that it is possible to give the [worker] what [they] most want[] -- high wages -- and the employer what [they] want[] -- a low labor cost -- for [their] manufactures. (*Primer of Scientific Management*, Gilbreth, Frank. p. 1)

Taylor spent his *time* painstakingly recording the minutes and seconds that it took employees to perform every part of their task. Despite the fact that his work led to processes that most modern-day management theorists take for granted (just-in-time supply chains, Six Sigma, Agile Development, etc), he was clearly not the most popular person at dinner parties. In fact, there were probably several wealthy industrialists that were sick of having him around. Railroad barons in the early 20th century attempted to push through increases in the rates that they charged their shippers, obviously hoping to increase their profit. Unfortunately for the railroad owners, the shippers were represented by **future Supreme Court Justice Louis Brandeis** [↗ \(https://www.newyorker.com/magazine/2009/10/12/not-so-fast\)](https://www.newyorker.com/magazine/2009/10/12/not-so-fast) who called on Taylor to testify to Congress that the railroads were inefficiently managed and did not need to charge more. In the era before Netflix and Hulu, his testimony **riveted** [↗ \(https://hdl.handle.net/2027/mdp.39015046810142\)](https://hdl.handle.net/2027/mdp.39015046810142) the nation and his time on the witness stand popularized his method, Taylorism. The rest, as they say, is history.



In whatever way they work -- fast or slow; mechanically or artistically -- *people* are what drive the economy. In particular, those who are of *working age* (generally considered between the ages of 15 and 64) are the real drivers. Modern economies are measured in terms of their Gross Domestic Product (GDP) and The Federal Reserve **has published** [↗ \(https://fred.stlouisfed.org/series/GDP\)](https://fred.stlouisfed.org/series/GDP) our GDP once a quarter since 1977. Data on the size of the working-age population during that same period is also **available** [↗](#)

(<https://fred.stlouisfed.org/series/LFWA64TTUSM647S>). If you divide the GDP by the size of the working-age population at any given moment, you calculate the (working-age) per-capita GDP.



From the looks of that graph (and an [equivalent one](https://fred.stlouisfed.org/graph/?g=mJvB) [from the Fed itself](https://fred.stlouisfed.org/graph/?g=mJvB)), it seems like we are all getting way more productive every year! If you think that this analysis is reductive, please tell me how to improve it! Economists [have](https://www.wsj.com/us-news/this-stat-could-transform-how-you-view-economic-growth-266b1ec0?st=evmpzoagaucf3rb&reflink=desktopwebshare_permalink) [debated](https://www.imf.org/external/pubs/ft/fandd/2017/03/lee.htm) [the extent](https://openknowledge.worldbank.org/bitstream/handle/10986/25050/On0the0impact00savings00and0poverty.pdf;sequence=1) [to which the size of the working-age population determines society's economic robustness](https://www.rand.org/content/dam/rand/pubs/working_papers/WR1000/WR1063-1/RAND_WR1063-1.pdf). If past is prologue, just how much will each working-age person contribute to the GDP in the first quarter of 2025? Or the third quarter of 2052? You are about to write an application to answer *that very question!*

*Prolet* is an application that will calculate for the user the expected (working-age) per-capita GDP in a given quarter of a given year. *Prolet* will prompt the user for a quarter (first, second, third or fourth) and a year (e.g., 2025). Based on that input, *Prolet* will make some calculations and produce an estimate of the (working-age) per-capita GDP in the user's chosen quarter and year. Before you know it, your name will be alongside [John Nash in Sweden](https://en.wikipedia.org/wiki/Nobel_Memorial_Prize_in_Economic_Sciences)!

The correlation between (working-age) per-capita GDP and a given quarter in a year is based on how much time has passed since the 1st quarter of 1977. Let's call the 1st quarter of 1977 the *epoch*. As the developer of *Prolet* you have access to a library (in other words, a *function*) that will perform the mathematical calculations to produce the estimate. To, uhm, function, this function needs to know the number of quarters between the date that user entered and the epoch. Just how would such a calculation work?

First, consider that there are four quarters per year and calculate the *epoch* as the number of quarters since the year zero:

$$epochq = 1977 \cdot 4$$

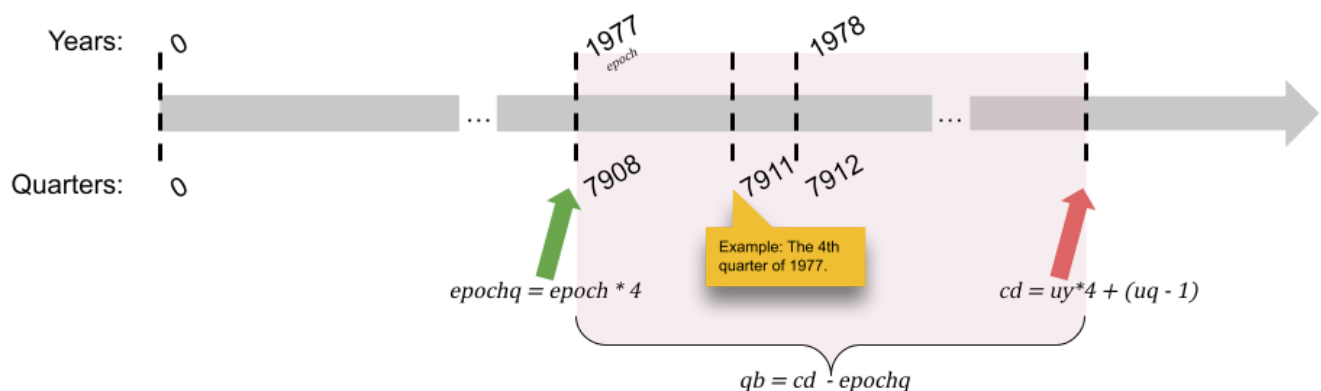
Let's assume that the quarter the user entered is stored in a variable named *uq* (note: if you use *uq* as a variable in your code you will *not* receive full credit -- be more descriptive!). Further, let's assume that the year the user entered is stored in a variable named *uy*. Again, there are four quarters in a year and let's use that fact to calculate the number of quarters between year zero and the user's chosen date:

$$cd = uy \cdot 4 + (uq - 1)$$

That's the hard work! The easy part remains:

$$qb = cd - epochq$$

*qb* is the number of quarters between the 1st quarter 1977 and the date for which the user wants to calculate the expected (working-age) per-capita GDP.



With that laborious calculation out of the way, you can now take advantage of the very helpful provided function to complete the *Prolet* application.

Just don't forget that those who control the means of production ... hmm ... I forget how it ends.

## Program Design Task

As my dad always said, "If you cut multiple times, make sure it's not to spite your face." Before you start writing code, please write the pseudocode for your implementation of the *Prolet* application.

## Program Design Requirements

The pseudocode is a tool for you to organize your thoughts and gather all the necessary elements before beginning to code. You may write your pseudocode as a series of steps or you may draw a flowchart. In either case, your pseudocode must describe the process you will use to

- 1. Prompt the user for input;
- 2. Gather the user's input;
- 3. Calculate the number of quarters between the user's chosen date and the *epoch*; and
- 4. Produce the properly-formatted output.

## Programming Task

*Prolet* will politely ask the user to enter their inputs with following prompts:

```
Please enter the year for the calculation:
Please enter the quarter for the calculation:
```

In response to the first prompt, the user will enter a four-digit year. In response to the second prompt, the user will enter either 1, 2, 3 or 4. Upon successful calculation, *Prolet* will succinctly display the results:

```
In the <first, second, third or fourth> quarter of <year>, the projected contribution to GDP
per working-age person is $<x...x.xx>.
```

For the purpose of writing *Prolet*, you may make the following assumptions:

- 1. The user will always provide input in the proper format.
- 2. The user will always provide a quarter/year that is after the epoch.
- 3. The (working-age) per-capita GDP will always be positive.

Example program input/output:

```
Please enter the year for the calculation: 2070
Please enter the quarter for the calculation: 3
In the third quarter of 2070, the projected contribution to GDP per working-age person is $20
8479.16.
```

**Note:** Your program will be tested against other test cases. Your program must compute properly in all cases in order to receive full points! Check the output of the autograder to make sure that your program behaves as expected. Read the autograder's output carefully!

There are several functions that you might find useful when programming *Prolet*:

Function Name	Parameters:	Result:
<code>get_int</code>	None	The <code>int</code> eger that the user typed.
<code>format_money</code>	An amount (as a <code>double</code> ) to be formatted as a monetary amount.	A <code>std::string</code> variable holding the given amount formatted in dollars and cents.
<code>quarter_to_ordinal</code>	A quarter (as an <code>int</code> ) to be converted to its proper	The ordinal (as a <code>std::string</code> ) matching the

ordinal term.

input. E.g., the result of calling

`quarter_to_ordinal(4)` will be`fourth`.`calculate_per_capita_gdp_estimate`

The number of quarters (as an `int`) in the future (relative to the epoch) for which to calculate a (working-age) per-capita GDP contribution estimate.

The estimated (working-age) per-capita GDP contribution estimate for the given number of months in the future (as a `double`).

You know what makes me work harder *and* smarter? Caffeine. If you are the first person to read this note, see me for some access to free coffee!!

## Programming Requirements

If you are a Windows-based C++ developer, start with this **skeleton**

(<https://uc.instructure.com/courses/1657742/files/176523739?wrap=1>) [↓](#)

([https://uc.instructure.com/courses/1657742/files/176523739/download?download\\_frd=1](https://uc.instructure.com/courses/1657742/files/176523739/download?download_frd=1)) . If you are a macOS-based C++ developer, start with this **skeleton**

(<https://uc.instructure.com/courses/1657742/files/176523989?wrap=1>) [↓](#)

([https://uc.instructure.com/courses/1657742/files/176523989/download?download\\_frd=1](https://uc.instructure.com/courses/1657742/files/176523989/download?download_frd=1)) . If you are a Linux-based developer, start with this **skeleton**

(<https://uc.instructure.com/courses/1657742/files/176540007?wrap=1>) [↓](#)

([https://uc.instructure.com/courses/1657742/files/176540007/download?download\\_frd=1](https://uc.instructure.com/courses/1657742/files/176540007/download?download_frd=1)) . The skeletons provide functions that you will need to successfully complete this lab.


1. Your program must prompt the user for input using exactly the specified format.
2. Your program must display output using exactly the specified format. In particular, the the estimated (working-age) per-capita GDP must be formatted like a true monetary amount, with precisely two digits after the dot.
3. Your program must use named constants to hold
  1. the number of quarters per year;
  2. the epoch year;
  3. and the epoch quarter.
4. Your program must use intermediate variables (of the appropriate type) to store the results of its calculations before printing them to the console.
5. Your program must use *meaningful* variable names.
6. Your program must call the provided function (`get_int`) to gather user input.
7. Your program must use the provided function (`calculate_per_capita_gdp_estimate`) to generate the (working-age) per-capita GDP estimate.
8. Your program must contain a multi-line comment before the `main` function describing the purpose of the program. The text in the comment must be grammatically correct and accurately describe the program's *purpose*, but *not* its operation.

## Critical Thinking Task

In class we talked about how functions are a means of abstraction, of hiding details. The user of the function only cares what the function does and not how it does it. Programmers can read the declaration of a function and know how to call it. This concept has real-world parallels. Consider a restaurant where the kitchen is a function. The menu is like a function declaration. It has a list of all the items that a diner can order from the kitchen (the function names) and the ways that they can customize those dishes (parameters). When the diner tell the waitstaff that they would like their fajitas to have chicken rather than steak, they are calling a function with arguments. Later, the sizzling fajitas come from the kitchen to their table (the return value). As a customer, neither do they see the kitchen nor do they care how the food is prepared. All they care about is that it tastes good!

Your critical thinking task is to come up with a physical analogy (like the one above) for functions.

## Critical Thinking Requirements

Your analogy must incorporate the concepts of function declaration, parameters/arguments, call, and return! Please use no more than 200 words. If you'd like, you may use external resources but you must provide proper documentation. The choice of formatting for external references is up to you, but you may find it helpful to consult the Purdue OWL for [help](https://owl.purdue.edu/owl/research_and_citation/apa_style/apa_style_introduction.html) .

([https://owl.purdue.edu/owl/research\\_and\\_citation/apa\\_style/apa\\_style\\_introduction.html](https://owl.purdue.edu/owl/research_and_citation/apa_style/apa_style_introduction.html)). The

Purdue OWL also has extensive information on ways to [avoid plagiarism](https://owl.purdue.edu/owl/avoiding_plagiarism/index.html) .

([https://owl.purdue.edu/owl/avoiding\\_plagiarism/index.html](https://owl.purdue.edu/owl/avoiding_plagiarism/index.html)).

## Question Task:

For this lab you deployed your knowledge of using functions, saw the definition of abstraction actually play out and worked with the difference between `int`eger and floating-point (`double`) division. Was there anything surprising? Something that doesn't make sense? Please feel free to ask anything related to anything that we've learned so far (or will learn)!

## Question Requirement:

Submit a file named `question.pdf` which contains your question. Take particular notice of the fact that the question does not have to be subtle, or deep or philosophical for you to receive full points. The goal of the question is for me to be able to get a sense whether there are common questions about the material which would indicate where I failed to teach the material appropriately.

## Deliverables

1. The pseudocode describing the algorithm of your *Prolet* program in PDF format (named `design.pdf`).
2. The C++ source code for your *Prolet* application (named `Prolet.cpp`).
3. A written response to the Critical Thinking Task prompt in PDF format (named `abstraction.pdf`).
4. A written question about the material covered so far in this class in PDF format (name the file `question.pdf`).

# Rubric

Points	Task	Criteria
15	Design	Pseudocode uses algorithmic thinking to express a high-level description of how to implement the <i>Prolet</i> application and meets the specified criteria.
15	Critical Thinking	Analogy incorporates the concepts of function declaration, parameters/arguments, call, and return.
20	Programming	Program prompts for user input according to the specified format.
20	Programming	Program correctly calculates its outputs for all test cases.
5	Programming	Program uses appropriately typed and named variables to store the intermediate result of calculations.
10	Programming	The program uses appropriately named constants to hold the values of the number of quarters per year, the epoch year and the epoch quarter.
5	Programming	Program uses provided functions for gathering user input.
5	Programming	Program uses provided function for formatting the output.
5	Programming	<code>main</code> function's purpose is described by a grammatically correct multi-line comment.



## Related Learning Objectives

1. Students will understand abstraction.
2. Students will use named constants effectively to increase code readability.
3. Students will be able to navigate the vagaries of using `int`egers and `double`s in arithmetic in C++.
4. Students will be able to call a function and use its result.
5. Students will be able to use compound assignment operators.

## Attributions

Stopwatch by Ilsur Aptukov from Noun Project