4.x — Chapter 4 summary and quiz

BY ALEX ON JUNE 11TH, 2007 | LAST MODIFIED BY NASCARDRIVER ON FEBRUARY 1ST, 2020

Quick Review

The smallest unit of memory is a **binary digit**, also called a **bit**. The smallest unit amount of memory that can be addressed directly is a **byte**. The modern standard is that a byte equals 8 bits.

A **data type** tells the compiler how to interpret the contents of memory in some meaningful way.

C++ comes with support for many fundamental data types, including floating point numbers, integers, boolean, chars, null pointers, and void.

Void is used to indicate no type. It is primarily used to indicate that a function does not return a value.

Different types take different amounts of memory, and the amount of memory used may vary by machine. See **4.3 -- Object sizes and the sizeof operator** for a table indicating the minimum size for each fundamental type.

The **sizeof** operator can be used to return the size of a type in bytes.

Signed integers are used for holding positive and negative whole numbers, including 0. The set of values that a specific data type can hold is called its **range**. When using integers, keep an eye out for overflow and integer division problems.

Unsigned integers only hold positive numbers, and should generally be avoided unless you're doing bit-level manipulation.

Fixed-width integers exist to define integer types with guaranteed sizes. Favor the std::int_fast#_t and std::int_least#_t integers when you need a fixed size guaranteed to be at least a certain size. std::int8_t and std::uint8_t should generally be avoided, as they tend to behave like chars instead of integers.

size_t is an unsigned integral type that is used represent the size of length of objects.

Scientific notation is a shorthand way of writing lengthy numbers. C++ supports scientific notation in conjunction with floating point numbers. The digits in the significand (the part before the e) are called the **significant digits**.

Floating point is a set of types designed to hold real numbers (including those with a fractional component). The **precision** of a number defines how many significant digits it can represent without information loss. A **rounding error** can occur when too many significant digits are stored in a floating point number that can't hold that much precision. Rounding errors happen all the time, even with simple numbers such as 0.1. Because of this, you shouldn't compare floating point numbers directly.

The **boolean** type is used to store a true or false value.

If statements allow us to execute one or more lines of code if some condition is true. Multiple statements can be executed if they are put inside a **block** (inside curly braces). The conditional expression of an *if statement* is interpreted as a boolean value.

Char is used to store values that are interpreted as an ASCII character. When using chars, be careful not to mix up ASCII code values and numbers. Printing a char as an integer value requires use of static_cast.

Angled brackets are typically used in C++ to represent something that needs a parameterizable type. This is used with static_cast to determine what data type the argument should be converted to (e.g. `static_cast<int>(x)` will convert *x* to an int).

A **constant** is a fixed value that may not be changed. C++ supports two types of constants: literal constants, and symbolic constants.

Literals are values inserted directly into the code. Literals have types, and literal suffixes can be used to change the type of a literal from default.

Const variables are variables that can't be changed after being initialized. Const variables can be either runtime or compile-time constants. **constexpr** variables must be compile-time constants.

Don't use magic numbers in your code. Instead, use symbolic constants.

Quiz time

Question #1

Why are symbolic constants usually a better choice than literal constants? Why are const/constexpr symbolic constants usually a better choice than #defined symbolic constants?

Show Solution

Question #2

Pick the appropriate data type for a variable in each of the following situations. Be as specific as possible. If the answer is an integer, pick either int, long, or a specific fixed-width integer type (e.g. int16_t) based on range. If the variable should be const, say so.

a) The age of the user (in years)

Show Solution

b) Whether the user wants color or not

Show Solution

c) pi (3.14159265)

Show Solution

d) The number of pages in a textbook (assume size is important)

Show Solution

e) The length of a couch in feet, to 2 decimal places

Show Solution

f) How many times you've blinked since you were born (note: answer is in the millions)

Show Solution

g) A user selecting an option from a menu by letter

Show Solution

h) The year someone was born (assuming size is important)

Show Solution

Question #3

Author's note

The quizzes get more challenging starting here. These quizzes that ask you to write a program are designed to ensure you can integrate multiple concepts that have been presented throughout the lessons. You should be prepared to spend some time with these problems. If you're new to programming, you shouldn't expect to be able to answer these immediately.

Remember, the goal here is to help you pinpoint what you know, and which concepts you may need to spend additional time on. If you find yourself struggling a bit, that's okay.

Here are some tips:

- Don't try to write the whole solution at once. Write one function, then test it to make sure it works as expected. Then proceed.
- Use your debugger to help figure out where things are going wrong.
- Go back and review the answers to quizzes from prior lessons in the chapter, as they'll often contain similar concepts

If you are truly stuck, feel free to look at the solution, but take the time to make sure you understand what each line does before proceeding. As long as you leave understanding the concepts, it doesn't matter so much whether you were able to get it yourself, or had to look at the solution before proceeding.

Write the following program: The user is asked to enter 2 floating point numbers (use doubles). The user is then asked to enter one of the following mathematical symbols: +, -, *, or /. The program computes the answer on the two numbers the user entered and prints the results. If the user enters an invalid symbol, the program should print nothing.

Example of program:

```
Enter a double value: 6.2
Enter a double value: 5
Enter one of the following: +, -, *, or /: *
6.2 * 5 is 31
```

Show Hint

Show Hint

Show Solution

Question #4

Extra credit: This one is a little more challenging.

Write a short program to simulate a ball being dropped off of a tower. To start, the user should be asked for the height of the tower in meters. Assume normal gravity (9.8 m/s^2), and that the ball has no initial velocity (the ball is not moving to start). Have the program output the height of the ball above the ground after 0, 1, 2, 3, 4, and 5 seconds. The ball should not go underneath the ground (height 0).

Your program should include a header file named constants.h that contains a symbolic constant to hold the value of gravity (9.8).

Use a function to calculate the height of the ball after x seconds. The function can calculate how far the ball has fallen after x seconds using the following formula: distance fallen = gravity_constant * $x\_seconds^2$ / 2

Sample output:

```
Enter the height of the tower in meters: 100
At 0 seconds, the ball is at height: 100 meters
At 1 seconds, the ball is at height: 95.1 meters
At 2 seconds, the ball is at height: 80.4 meters
At 3 seconds, the ball is at height: 55.9 meters
At 4 seconds, the ball is at height: 21.6 meters
At 5 seconds, the ball is on the ground.
```

Note: Depending on the height of the tower, the ball may not reach the ground in 5 seconds -- that's okay. We'll improve this program once we've covered loops.
Note: The ^ symbol isn't an exponent in C++. Implement the formula using multiplication instead of exponentiation.

Show Solution

Question #5

Find 2 issues (affecting 3 lines) in the following code.

```
#include <cstdint>
#include <iostream>

int main()
{
  std::cout << "How old are you?\n";

  std::uint8_t age{};
  std::cin >> age;

  std::cout << "Allowed to drive a car in Texas [";

  if (age >= 16)
    std::cout << "x";
  else
    std::cout << " ";

  std::cout << "]\n";

  return 0;
}
```

Sample output

```
How old are you?
6
Allowed to drive a car in Texas [ ]


How old are you?
19
Allowed to drive a car in Texas [x]
```

Show Solution

5.1 -- Operator precedence and associativity

Index

4.13 -- Const, constexpr, and symbolic constants

C++ TUTORIAL | PRINT THIS POST

836 comments to 4.x — Chapter 4 summary and quiz

« Older Comments 1 ... 9 10 11

vova220v
February 8, 2020 at 12:37 pm · Reply

```
#include "constants.h"
#include <iostream>

double getTowerHeight(){
    double towerHeight{0};
    std::cout << "Enter the height of the tower in meters: ";
    std::cin >> towerHeight;

    return towerHeight;
}

int getSeconds(){
```

```
    int seconds{0};
    std::cout << "Enter seconds: ";
    std::cin >> seconds;

    return seconds;
}

void calcDistanceToBall(double towerHeight, int seconds){

    std::cout << "At 0 seconds, the ball is at height: " << towerHeight << "meters.\n";

    double distanceToBall{0};

    for(int i=1; i<=seconds; i++){

        distanceToBall = towerHeight - (constants::gravity * static_cast<double>(i*i) / 2.0

        if(distanceToBall > 0){
            std::cout << "At " << i << "seconds, the ball is at height: " << distanceToBall
        }else{
            std::cout << "At " << i << "seconds, the ball is on the ground.\n";
            i = seconds;    //For example towerHeight - 100 and seconds - 10, than "the bal
                            //will be printing 5 times. To avoiding this, i'm using this li
        }
    }

}

int main(){

    double towerHeight{getTowerHeight()};
    int seconds{getSeconds()};
    calcDistanceToBall(towerHeight, seconds);

    return 0;
}
```

Am
February 8, 2020 at 12:40 am · Reply

I suppose I went a little "extra" on this, any optimizations to reccomend given what's currently been taught, or optimizations that also haven't been taught too I guess that I missed?

Thanks.

```
#ifndef PCH_H
#define PCH_H

namespace constant {
    constexpr double gravityConstant{ 9.8 };
}

#endif
```

```
#include "pch.h"
#include <iostream>

double distanceInput() {
    double x{  };
    std::cout << "\nInput desired starting falling height in meters: ";
    std::cin >> x;
```

```
    return x;
}

double chooseTime() {
    double time{};
    std::cout << "\nInput desired falling time in seconds: ";
    std::cin >> time;
    return time;
}

double calcFallDistance(double time, double x) {
    double fallDistance{ constant::gravityConstant * (time * time) / 2 };
    if (fallDistance >= x)
        return x;
    else
        return fallDistance;
}

double calcCurrentHeight(double fallDistance, double x) {
    double currentHeight{ x - fallDistance };
    if (currentHeight <= 0)
        return 0;
    else
        return currentHeight;
}

void getValues(double x) {
    double time{ chooseTime() };
    double fallDistance{ calcFallDistance(time, x) };
    double currentHeight{ calcCurrentHeight(fallDistance, x) };

    std::cout << "\nAt " << time << " seconds, the ball is has fallen " << fallDistance <<
}

int main() {
    std::cout << "This is a test program to calculate the current height of a\n";
    std::cout << "ball being dropped at user inputted height and 'Five' user\n";
    std::cout << "inputted time variables.\n";

    double x{ distanceInput() };
    getValues(x);
    getValues(x);
    getValues(x);
    getValues(x);
    getValues(x);
    getValues(x);

    return 0;
}
```

nascardriver
February 8, 2020 at 3:38 am · Reply

pch.h is a file of MSVC, don't name your own files that.

- Use double literals for doubles (2.0, 0.0)
- Name variables descriptively. "x" has no meaning.

If you want code to repeat, you can use loops, eg.

```
while (true)
```

```
{
  getValues(x);
}
```

will keep calling `getValues` until you terminate the process.
Loops are covered later.

Tom
February 7, 2020 at 10:14 am · Reply

Here's my solution to Quiz #4. It looks quite different from the one given as the solution.

```
#include <iostream>
#include "constants.h"

double getHeight()
{
    std::cout << "Enter the height of the tower in meters: ";
    double height{};
    std::cin >> height;
    return height;
}

int calcHeight(double startHeight)
{
    double newHeight{ startHeight };
    for (double sec{ 0.0 }; sec < 6.0; ++sec)
    {
        newHeight = { startHeight - ((gravity * (sec * sec)) / 2.0) };
        if (newHeight <= 0)
        {
            std::cout << "\nAt " << sec << " seconds, the ball is on the ground.";
            return 0;
        }
        else
        {
            std::cout << "\nAt " << sec << " seconds, the ball is at height: " <<
                newHeight << " meters";
        }

    }
    return 0;
}

int main()
{
    double startHeight{ getHeight() };
    calcHeight(startHeight);
}
```

I had to consult the given solution for one thing, the equation I used had one mistake. I kept using newHeight as the foundation for height variable in new distance equation (line 17). This meant at 0 (obviously) and 1 seconds, my equation was correct but once newHeight was being updated, the subsequent seconds were off. Once I made the connection that we need to continue using the initial height for the new height equation, then it all fell into place and I am quite happy with my results.

One thing I saw was that VS2019 kept giving me a warning about a potential stack overflow in variable "sec". I was initially making it an int since I figured we're dealing with whole seconds and it's such a small loop that it shouldn't matter, and indeed didn't matter despite the warning, but I don't like seeing squiggly lines under my code so I went with a double to placate the compiler.

RJ
February 7, 2020 at 2:12 pm · Reply

I noticed (on line 14) that you initialized 'newHeight' with the variable 'startHeight', i don't believe that's necessary because the function you made has it's own scope of the variable startHeight so your not really changing the value of the variable you started with on line 35. Instead you could have made startHeight (on line 35) into a Run-time constant,

```
1 const double startHeight{getHeight()}
```

It's almost like you created a shield for the variable to be used in the function calcHeight(). (I just did this Quiz today as well so i just as new to this as you are, so correct me if i'm wrong :))

Tom
February 7, 2020 at 3:39 pm · Reply

Hmmmm thinking about the logic I probably could avoid creating the variable newHeight, and simply put it in the equation and not save the result, just let it print out. I think it would force changes to the loop conditions and statements though. How did you end up solving the quiz? I'd love to see your results.

RJ
February 7, 2020 at 10:45 pm · Reply

Here you go.

calculations.h

```
1  #ifndef CALCULATIONS_H
2  #define CALCULATIONS_H
3
4
5   namespace grav
6   {
7       constexpr float gravity(9.8f);
8   }
9
10
11  int GetHeight();
12  double HeightCalc(double height,int sec);
13  void PrintResult(double results, int sec);
14
15 #endif // CALCULATION_H
```

calculations.cpp

```
1  #include <iostream>
2  #include <cmath>
3  #include "calculations.h"
4
5
6   double HeightCalc(double height,int sec)
7   {
8       height = height - ( grav::gravity * (pow(sec,2) / 2 )) ;
9
10     return height;
11  }
12
```

```
void PrintResult(double results, int sec)
{
    if(results < 0)
       std::cout<< "The ball is on the ground\n";
    else
    std::cout<< "The ball is at height: " << results << " at " << sec << " s
}

int GetHeight()
{
 std::cout<< "Enter the height of the tower in meters:\n ";

   int height{};

   std::cin>> height;

   return height;
}
```

main.cpp

```
#include <iostream>
#include "calculations.h"


int main()
{
   const int height {GetHeight()};

     PrintResult( HeightCalc(height,0), 0 );
     PrintResult( HeightCalc(height,1), 1 );
     PrintResult( HeightCalc(height,2), 2 );
     PrintResult( HeightCalc(height,3), 3 );
     PrintResult( HeightCalc(height,4), 4 );
     PrintResult( HeightCalc(height,5), 5 );

    return 0;
}
```

Tom
February 6, 2020 at 4:49 pm · Reply

Here's what I ended up with for Quiz #3:

```
#include <iostream>

double getValue()
{
    std::cout << "Enter a number: ";
    double num{};
    std::cin >> num;
    std::cout << '\n';
    return num;
}

char getSymbol()
{
    char symbol{};
AskForSymbol:
    std::cout << "Enter an operation symbol (+, -, * or /): ";
    std::cin >> symbol;
```

```
18      if (symbol == '+') return symbol;
19      else if (symbol == '-') return symbol;
20      else if (symbol == '*') return symbol;
21      else if (symbol == '/') return symbol;
22      else
23      {
24          std::cout << "\nYou have entered an invalid symbol, please try again.\n\n";
25          goto AskForSymbol;
26      }
27  }
28
29  double calculateValue(double x, double y, char operation)
30  {
31      double answer{};
32      if (operation == '+') { answer = (x + y); return answer; }
33      else if (operation == '-') { answer = (x - y); return answer; }
34      else if (operation == '*') { answer = (x * y); return answer; }
35      else { answer = (x / y); return answer; }
36  }
37
38  int main()
39  {
40      double value1 = { getValue() };
41      double value2 = { getValue() };
42      char symbol = { getSymbol() };
43      std::cout << '\n' << value1 << " " << symbol << " " << value2 <<
44          " = " << calculateValue(value1, value2, symbol) << '\n';
45  }
```

nascardriver
February 7, 2020 at 8:50 am · Reply

Congrats on solving the quiz!

- Line 40-42: Initialize with brace initialization, no `=`
- You don't need `answer`, you can return immediately.
- Use single quotes for characters (" " vs ' ').

Tom
February 7, 2020 at 9:26 am · Reply

Thanks for the reply and the tutorials, they've been super helpful.

For the single quotes, I take it you mean the ones on line 43? I wasn't sure if single quotes could process a space or not. Guess I could have tried and seen. I see what you mean about the other two corrections, thanks for checking it out and replying!

nascardriver
February 8, 2020 at 1:45 am · Reply

Yes. Single quotes can hold any ASCII character.

Patryk_again
January 27, 2020 at 2:05 am · Reply

Hi again. Could you please tell me what do you think of the code now? I promise I won't bother you in the future :D
(I think my wypiszWynik void does too many tasks...?)

```
#include <iostream>
#include "constants.h"//const gravity

double wysokoscWiezy()
{
    std::cout << "Wpisz wysokosc wiezy w metrach: ";
    double wysokoscWiezy{ 0.0 };
    std::cin >> wysokoscWiezy;

    return wysokoscWiezy;
}


void wypiszWynik(double wysokosc, int sekundy)
{
    int sekundyKwadrat{ 0 };
    sekundyKwadrat = sekundy * sekundy;

    double aktualnaWysokosc{ 0.0 };
    aktualnaWysokosc = wysokosc - gravity * sekundyKwadrat / 2.0;
    if (aktualnaWysokosc > 0.0)
        std::cout <<sekundy<<". sekunda "<< aktualnaWysokosc <<" m"<< '\n';
    else
        std::cout << "W "<<sekundy<<". sekundzie pileczka uderzyla w ziemie.";
}



int main()
{
    double wysokosc{ wysokoscWiezy() };
    wypiszWynik(wysokosc, 0);
    wypiszWynik(wysokosc, 1);
    wypiszWynik(wysokosc, 2);
    wypiszWynik(wysokosc, 3);
    wypiszWynik(wysokosc, 4);
    wypiszWynik(wysokosc, 5);


    return 0;
}
```

nascardriver
January 27, 2020 at 3:59 am · Reply

Hello again,

your code is better. Suggestions now:
- If you know a variables initial value, initialize it instead of assigning (Line 16,17 and 19, 20)
- Programming in english will get you a lot more help if you ever have a problem.
- Your code is inconsistently formatted. Your editor should have an auto-formatting feature.

Papi
January 27, 2020 at 9:07 am · Reply

Thank you for the answer. See you in next chapter

Patryk
January 26, 2020 at 6:23 am · Reply

Hello all. First I want to say thank you for this site, its really cool. I think the code I wrote first time is very bad, but could you actually confirm how bad is it, please? Have a nice day:)
gravity is included in constants.h as constexpr double, obviously.

```
#include <iostream>
#include "constants.h"

double tower()
{
    std::cout << "enter height in meters: ";
    double meters{};
    std::cin >> meters;
    return meters;
}

void falling(double x)
{
    if (x > 0)
    {
        std::cout << " 0 seconds: " << x << '\n';
        std::cout << " 1. second: " << x - (gravity * 1 / 2)<<'\n';
        std::cout << " 2. seconds: " << x - (gravity * (2 * 2) / 2) << '\n';
        std::cout << " 3. seconds: " << x - (gravity * (3 * 3) / 2) << '\n';
        std::cout << " 4. seconds: " << x - (gravity * (4 * 4) / 2) << '\n';
        std::cout << " 5. seconds: " << x - (gravity * (5 * 5) / 2) << '\n';
    }
    else
        std::cout << "wrong height entered.";
}

int main()
{

    double meters{ tower() };
    falling(meters);

    return 0;
}
```

nascardriver
January 26, 2020 at 6:47 am · Reply

Hi!

You have duplicate code and magic numbers. Add a function to calculate the height after a certain amount of seconds. Also, use double literals for doubles (2.0 instead of 2, etc.).

Christopher Springer
January 29, 2020 at 7:25 am · Reply

One additional thing, per Best Practices listed in section 2.11, be sure to list your custom header files first in your include directives, followed by 3rd party libraries, and finally standard library header files.

Tony98
January 25, 2020 at 3:53 pm · Reply

Hey again you all! I'm here to see whether what I did can be called correct once again, sorry for the work!
I've done the last exercise quite differently in the .cpp file. Is this ok, or is there anything I could do better?
Thank you as always!!

```
#include "constants.h"
#include <iostream>



double towerHeigth()
{
    std::cout << "Please enter initial tower heigth: ";
    double heigth{};
    std::cin >> heigth;

    return heigth;
}

double distanceFallen(double userHeigth, int seconds)
{
    double updatedHeigth{ userHeigth - constants::gravity * (seconds * seconds) / 2.0 };
    return updatedHeigth;
}

void printResult(int seconds, double result)
{
    if (result > 0.0)
        std::cout << "At " << seconds << " seconds, the ball is at height: " << result << '
    else
        std::cout << "At " << seconds << " seconds, the ball is on the ground.\n";

}




int main()
{
    double userHeigth{ towerHeigth() };
    printResult(0, distanceFallen(userHeigth, 0));
    printResult(1, distanceFallen(userHeigth, 1));
    printResult(2, distanceFallen(userHeigth, 2));
    printResult(3, distanceFallen(userHeigth, 3));
    printResult(4, distanceFallen(userHeigth, 4));
    printResult(5, distanceFallen(userHeigth, 5));
    return 0;
}
```

nascardriver
January 26, 2020 at 1:57 am · Reply

Hello again!

Your code looks good :-)
In line 35-40 you're repeating the seconds, that can lead to problems when you update them. Adding a combined function to calculate and prints would solve this.

Tony98
January 26, 2020 at 5:14 am · Reply

Thanks nascar for both replies!
I really appreciate your work and your help, and I'm sorry if I'll keep asking questions!
:P

Michael
January 18, 2020 at 8:18 am · Reply

I wrote the same program, only with loops.

```
#include <iostream>
#include "constants.h"

void print(double b, int c)
{
    if (b < 0) {
        std::cout << "В " << c << " секунд мяч находится на земле." << std::endl;
    }
    else {
        std::cout << "В " << c << " секунд мяч находится на высоте: " << b << std::endl;
    }
}

double calculation(const double a)
{
    int second{ -1 };
    double current{ a };
    while (current > 0) {
        ++second;
        double fall{ gravity_constant * (second * second) / 2.0 };
        current = a - fall;
        print(current, second);
    }
    return 0;
}

double heigth()
{
    std::cout << "Введите высоту: ";
    double heigth_v{ 0.0 };
    std::cin >> heigth_v;
    return heigth_v;
}

int main()
{
    setlocale(LC_ALL, "rus");
    const double a{ heigth() };
    calculation(a);
    return 0;
}
```

nascardriver
January 19, 2020 at 1:27 am · Reply

This quiz is repeated in the next chapter with loops. If you know them already, that's fine.

Suggestions:
- Name variables descriptively, avoid abbreviations. "a", "b", "c" are useless names.
- Use '\n' unless you need `std::endl`. `std::endl` is slower.
- `calculation`'s return value is unused. It shouldn't call `print`, but return the new height instead.

Michael
January 19, 2020 at 5:14 am · Reply

Thank you for your reply!

HolzstockG
January 4, 2020 at 4:59 am · Reply

After a long time finally I have produced these 2 programs. I want to share with you with the second one since it is tougher to do. I had several problems with it but I came with my own solution.

For some reason this program produces different results than program made by an author (after long time thinking about what can cause it I looked at solution and I see that there shouldn't be such an cause to produce these results).

If anyone sees the reason why it happens then please respond.

Here are the files I used and contents inside them.

[b] constants.h [/b]

```
#ifndef CONSTANTS_H
#define CONSTANTS_H

constexpr double gravitationalAccelerationEarth{ 9.8 };

#endif
```

[b] allNeededFunctions.h [/b]

```
#ifndef ALLNEEDEDFUNCTIONS_H
#define ALLNEEDEDFUNCTIONS_H

double getUsersNumber();
void displayResult(double height, int second);
double calculateHeight(double towerHeight, int second);
bool checkIfBelowGround(double height, int second);

#endif
```

[b] allNeededFunctions.cpp [/b]

```
#include "constants.h"
#include <iostream>


double getUsersNumber()
{
```

```
    double usersNumber{};
    std::cin >> usersNumber;

    return usersNumber;
}

void displayResult(double height, int second)
{
    std::cout << "At " << second << " seconds, the ball is at height: " << height << '\n';
}

double calculateHeight(double towerHeight, int second)
{
    return towerHeight - ((gravitationalAccelerationEarth * (second * second)) / 2);
}

bool checkIfBelowGround(double height, int second)
{
    if (height < 0)
    {
        std::cout << "At " << second << " second, the ball is on the ground." << '\n';
        return true;
    }
    else
    {
        displayResult(height, second);
        return false;
    }
}
```

[b] main.cpp [/b]

```
#include "allNeededFunctions.h"
#include <iostream>


int main()
{
    std::cout << "Enter the height of the tower in meters: ";
    double towerHeight{ getUsersNumber() };

    std::cout << '\n';

    towerHeight = calculateHeight(towerHeight, 0);
    if (checkIfBelowGround(towerHeight, 0))
        return 0;

    //-------------------------------------------------------------------------

    towerHeight = calculateHeight(towerHeight, 1);
    if (checkIfBelowGround(towerHeight, 1))
        return 0;

    //-------------------------------------------------------------------------

    towerHeight = calculateHeight(towerHeight, 2);
    if (checkIfBelowGround(towerHeight, 2))
        return 0;

    //-------------------------------------------------------------------------

    towerHeight = calculateHeight(towerHeight, 3);
    if (checkIfBelowGround(towerHeight, 3))
```

```
32          return 0;
33
34      //-----------------------------------------------------------------------
35
36      towerHeight = calculateHeight(towerHeight, 4);
37      if (checkIfBelowGround(towerHeight, 4))
38          return 0;
39
40      //-----------------------------------------------------------------------
41
42      towerHeight = calculateHeight(towerHeight, 5);
43      if (checkIfBelowGround(towerHeight, 5))
44          return 0;
45
46
47      return 0;
48  }
```

Results of this program are:
Enter the height of the tower in meters: 100

At 0 seconds, the ball is at height: 100
At 1 seconds, the ball is at height: 95.1
At 2 seconds, the ball is at height: 75.5
At 3 seconds, the ball is at height: 31.4
At 4 second, the ball is on the ground.

Limerk
January 4, 2020 at 9:33 am · Reply

Long story short, every time you use this line: "towerHeight = calculateHeight(towerHeight, numOfSecons);" you reassign the value of towerHeight.

The formula for finding the height of the ball is a Physics formula from kinematics which is
x = vt + 1/2*a*t^2 where (x - displacement, v - initial velocity, t - time in seconds, a - acceleration due to gravity). Since our initial velocity is 0 we can rearrange this formula to
x = 1/2*a*t^2

So:
t = 0 => x = 1/2*9.8*0^2;
t = 1 => x = 1/2*9.8*1^2;
t = 2 => x = 1/2*9.8*2^2, and so on.

Your "towerHeight = calculateHeight(towerHeight, numOfSecons);" reassigns initial value of the height every time it calculates distance.

So:
t = 0; => towerHeight = towerHeight - 1/2*9.8*0^2; (which is 100);
t = 1; => towerHeight = 100 - 1/2*9.8*0^2; (which is 95.1);
t = 2; => towerHeight = 95.1 - 1/2*9.8*0^2; (which is 75.5), and so on.

Instead of reassigning your towerHeight you should subtract displacement from you INITIAL towerHeight value. In order to do this, you need to assign the result of your subtraction to the another variable, for example: ballHeight = towerHeight - 1/2*9.8*t^2, and then use your ballHeight in your check function;

I hope I explained it well enough, I'm sorry if I made grammatical mistakes and etc. English isn't my native language and I'm only learning it =)

HolzstockG
January 5, 2020 at 4:17 am · Reply

No worries, I am not native speaker too so I do not judge because of some gramatical mistakes. The best of the best make them :)

Anyway I think I understand it. It is nothing hard to change in a code but I needed to think for a while why it has to be like that.

```
#include "allNeededFunctions.h"
#include <iostream>


int main()
{
    std::cout << "Enter the height of the tower in meters: ";
    const double towerHeight{ getUsersNumber() };
    double ballHeight{ towerHeight };

    std::cout << '\n';

    ballHeight = calculateHeight(towerHeight, 0);
    if (checkIfBelowGround(ballHeight, 0))
        return 0;

    //-------------------------------------------------------------------------

    ballHeight = calculateHeight(towerHeight, 1);
    if (checkIfBelowGround(ballHeight, 1))
        return 0;

    //-------------------------------------------------------------------------

    ballHeight = calculateHeight(towerHeight, 2);
    if (checkIfBelowGround(ballHeight, 2))
        return 0;

    //-------------------------------------------------------------------------

    ballHeight = calculateHeight(towerHeight, 3);
    if (checkIfBelowGround(ballHeight, 3))
        return 0;

    //-------------------------------------------------------------------------

    ballHeight = calculateHeight(towerHeight, 4);
    if (checkIfBelowGround(ballHeight, 4))
        return 0;

    //-------------------------------------------------------------------------

    ballHeight = calculateHeight(towerHeight, 5);
    if (checkIfBelowGround(ballHeight, 5))
        return 0;


    return 0;
}
```

It works as it should now

Enter the height of the tower in meters: 100

```
At 0 seconds, the ball is at height: 100
At 1 seconds, the ball is at height: 95.1
At 2 seconds, the ball is at height: 80.4
At 3 seconds, the ball is at height: 55.9
At 4 seconds, the ball is at height: 21.6
At 5 second, the ball is on the ground.
```

koe
December 17, 2019 at 11:06 pm · Reply

Typo

"A rounding error can occur when too many significant digits stored in a floating point number that can’t hold that much precision."

Should be

"A rounding error can occur when too many significant digits are stored in a floating point number that can’t hold that much precision."

Another typo (missing space)

"Const variables can be either runtime or compile-time constants.constexpr variables must be compile-time constants."

nascardriver
December 18, 2019 at 5:21 am · Reply

Lesson updated, thanks!

chai
December 3, 2019 at 1:31 pm · Reply

Namespace was discussed in 2:9 but creating one was never discussed. I think an update is needed.

nascardriver
December 4, 2019 at 4:08 am · Reply

Lesson updated, now without a namespace, thanks for pointing out the error!

Chandler
December 2, 2019 at 6:24 pm · Reply

Learning C++ is on my bucket list, and I have to say that this is the most amazing programming guide, for any language, that I've ever seen. You guys, Alex and nascardriver, are doing a superb job. Just incredible work!

However, I digress. I have a question. I added to the last challenge and found something I don't understand. This will probably be answered further down the line in the tutorial. But if I'm not asking too much, maybe you guys can answer me before we get there. Thanks.

Here's my main.cpp code

```
1 #include <iostream>
```

```
#include "constants.h"

//double disatnceFallen{ (myConstants::gravity*(seconds * seconds))/2 };
//double currentHeight{ towerHeight - disatnceFallen };

double getHeight()
{
  std::cout << "Please enter starting height for ball drop: ";
  double height{};
  std::cin >> height;

  return height;
}

int getSeconds()
{
  std::cout << "Enter the number of seconds you think it will take to hit the floor: ";
  int seconds{0};
  std::cin >> seconds;

  return seconds;
}

void dropTheBall(double towerHeight, int seconds)
{
//  double disatnceFallen{0};
//  double currentHeight{0};
  double ballHeight{towerHeight};

  for(int time{0}; time < seconds; ++time)
    {
      double disatnceFallen{ (myConstants::gravity*(time * time))/2 };
      double currentHeight{ ballHeight - disatnceFallen };
      ballHeight = currentHeight;

      if(ballHeight < 0.0)
        {
          std::cout << "\nWith " << seconds - time << " seconds (and some change) to spare,
          break;
        }
      std::cout << "At " << time << " seconds, the ball is at " << ballHeight << " meters.\
    }
}

int main()
{
  double towerHeight{ getHeight() };
  int seconds{ getSeconds() };

  dropTheBall(towerHeight, seconds);

  return 0;
}
```

See the two lines commented out in the function dropTheBall()? I did that because I couldn't leave them and have the lines in the for loop like:

```
disatnceFallen{ (myConstants::gravity*(time * time))/2 };
currentHeight{ ballHeight - disatnceFallen };
```

That gave me several errors. So the only way to fix things was to comment out the top two and add double to them in the for loop. Did I do something wrong? Or is that how it's suppose to be done?

Chandler
December 2, 2019 at 10:28 pm · Reply

Had I waited, Lesson 5.5 would've provided me the answer. Under the topic "The conditional operator evaluates as an expression", right after the second code example, you offer this:

"...variables defined inside an if or else statement die at the end of the if or else statement."

And since I was able to use ballHeight (defined outside the for loop), I've since uncommented CurrentHeight and now use it inside the for loop like this:

```
1 currentHeight = ballHeight - disatnceFallen;
```

But I had to leave distanceFallen as is.

nascardriver
December 3, 2019 at 4:36 am · Reply

Hi!

You don't need `disatnceFallen` and `currentHeight` outside of the loop, so they should be declared inside.
You can declare them outside and inside, but the outer variables won't be used, because the inner variables shadow them (They have the same name).

Anderson
December 2, 2019 at 4:22 am · Reply

If anyone has a minute,

Hi,
I attempted question 4 the right way with loops, seems easy enough, and it works just fine. But, I was wondering if there was a better way to write the code perhaps, maybe for the sake of optimizing for performance or reducing redundancies.

Also, don't feel discouraged, writing this particular program took lots of trials and errors from my side so you know that I'm not perfect at this sort of thing.

P.S. :-- If you are a new programmer(you've never learned any other programming language before), you need not worry about understanding everything in the code below. You will all learn loops later.

Main.cpp

```
1  #include <iostream>
2  #include "MyConstants.h"
3
4  void printFallProgress (double height) {
5
6      double distanceFallen{}; //The distance elapsed by the ball from the top of the tower.
7      double ballHeight { height }; //The height of the ball from the ground relative to the
8      int time { 0 };
9
10     while (ballHeight > 0.0) {
11         distanceFallen = (constant::gravity * time * time) / 2;
12         ballHeight = height - distanceFallen;
13
14         if (ballHeight > 0.0) {
15             std::cout << "At " << time << " seconds," << " the ball is at height: " << ball
```

```
            time++;
        }
        else {
            break;
        }


    }
    std::cout << "At " << time << " seconds, the ball is on the ground.";
}

int main()
{
    //distance fallen = (gravity_constant * x_seconds2) / 2

    std::cout << "What is the height of the tower ?";
    double height{}; //The height of the tower
    std::cin >> height;

    printFallProgress(height);

}
```

MyConstants.h

```
#ifndef MYCONSTANTS_H
#define MYCONSTANTS_H

namespace constant {
    constexpr double gravity{9.8};
}

#endif
```

nascardriver
December 2, 2019 at 4:41 am · Reply

Hi!

This quiz is repeated later in a later lesson with loops, but there's no reason not to use loops if you already know them.

- If your programs prints anything, the last thing it prints should be a line feed. Otherwise output of different programs might get mixed.
- Use double literals for doubles (2.0 instead of 2). Otherwise you might accidentally do integer arithmetic.
- `distanceFallen` should be declared inside of the loop. This won't change to produced binary, but decreases `distanceFallen`'s scope. Creation of fundamental types is pretty much free. You can start worrying about where to create a variable when types get bigger and more complex.
- Use ++prefix instead of postfix++. Postfix++ has to create a copy of the variable, but you don't need a copy.
- Line 10, 14, 19: Duplicate comparison, you can change the loop to `while (true)` and surround it with an `if`.

```
if (ballHeight > 0.0)
{
  while (true)
  {
    // ...
  }
```

```
7 }
```

Anderson
December 2, 2019 at 6:11 am · Reply

Hi nascardriver,

I just read through your reply and I, almost immediately, came up with some questions.

You mentioned that something like:
variable++;

is different from

++variable;

in the sense that variable++ creates one more copy of 'variable' while ++variable increments the numerical (integer or double) value. But I would appreciate it if you could tell me if variable++ in addition to creating a copy of 'variable' also increments the numerical value because the program runs just fine from my end ?
Funny enough, I learned to use the style variable++ in Java, but thank you for enlightening me on that.

Also, I agree with you when you say that 'distanceFallen' should be within the loop since it is dependent on 'time' and the gravity constant and hence there is no need for it to be updated regularly.

Finally, I agree with you when you mentioned the duplicate comparison, a huge redundancy on my end.
I figured that if I changed this part of the program :

```
while (ballHeight > 0.0) {
        distanceFallen = (constant::gravity * time * time) / 2;
        ballHeight = height - distanceFallen;

        if (ballHeight > 0.0)
        {
         //code...
        } else {
                    break;
                        }
}
```

to :

```
while (true) {
        distanceFallen = (constant::gravity * time * time) / 2;
        ballHeight = height - distanceFallen;

        if (ballHeight > 0.0)
        {
         //code...
        } else {
                    break;
                        }
}
```

it would remove the duplicate comparison. Thank you for pointing that out.

Thank you once more, nascardriver, for reviewing the code and maybe I should have posted this in the loops quiz instead so I won't confuse new programmers that might be here.
:)

nascardriver
December 2, 2019 at 6:16 am · Reply

++prefix and postfix++ both increment the variable. The difference is that postfix++ returns the value that the variable had before incrementing it, whereas ++prefix returns the new value. This is not different in Java.

Your suggested update to the loop will cause the loop to be entered even if the ball is on the ground from the beginning. For a simple formula like here, this isn't a huge problem, but if the loop did expensive work, you should consider checking `ballHeight` once before the loop runs.

Ikenna
December 2, 2019 at 7:53 am · Reply

Hi,

You mentioned that the 'while' loop in function printFallProgress() will be entered even if the ball is on the ground (when ballHeight < 0.0).This is true but if you notice in the loop there is an if statement and an else statement.

The if statement is executed if the ballHeight is greater than 0.0 and a statement is printed out

If, however, ballHeight is less than 0.0 then a break statement is executed causing the program to exit the loop.

Maybe you should copy, paste and run the program it works just fine. Thank you

nascardriver
December 2, 2019 at 8:08 am · Reply

It works just fine, but the program calculates `distanceFallen` and `ballHeight` even though could already know that the ball is on the ground (because `height` was `<= 0.0`). As I said, this calculation is easy, so it's not a big deal to lose some time. If the loops performed a more expensive operation, we should check `height` first.

Anderson
December 2, 2019 at 12:40 pm · Reply

Noted. Thanks!! :)

giang
January 11, 2020 at 6:46 pm · Reply

Hi, I did know about the loops so I tried some code to solve the Quiz above:

```
#include<iostream>
#include"constants.h"
using namespace std;

float d_fallen(int sec){
    return gra*sec*sec/2;
```

```
}

int main(){
    cout<<"Enter the height of the tower in meters: ";
    float h;
    cin>>h;

    int i=0;
    while (i<=5){
        if (h-d_fallen(i)>0)
            cout<<"At "<<i<<" seconds, the ball is at height
        else cout<<"At "<<i<<" seconds, the ball is on the g
    i++;
    }
    return 0;
}
```

It's just my idea so I don't know whether it's a good program or not. Can anyone give your opinions?

nascardriver
January 12, 2020 at 1:54 am · Reply

- Unless you have a reason to use a `float`, use a `double`.
- Use floating point literals in floating point calculations (2.0 or 2.0f instead of 2, same for 0).
- Name variables descriptively, avoid abbreviations.
- Inconsistent formatting. Use your editor's auto-formatting feature.
- Initialize variables with brace initialization for better type safety.

Gustav
November 9, 2019 at 9:23 pm · Reply

Hi Alex! Awesome tutorials, I'm having a blast doing these challenges.

Could you please evaluate my code so I can improve it more, thanks!

```
#include <iostream>
#include "constants.h"

using namespace std;

double getuserinput(){
    double x;
    cin >> x;
    return x;
}

void physicsFormula(double x,double seconds){
    double fallen{};
    for(int i = 1; i <= seconds; ++i){
    fallen = (physics::gravityconst * (i * i))/2;
    double delta{x - fallen};

    if(delta > 0){ // prints only absolute numbers

    cout << "@ " << i << " Seconds, the ball is at Height: " << delta << '\n';
    }else{
```

```
22
23      cout << "@ " << i << " Seconds, the ball is at the ground" << '\n';
24          }
25      }
26  }
27
28
29  void calculateHeight(){
30      cout << "Enter the height of the tower in meters: ";
31      double x{getuserinput()};
32      cout << "Enter the seconds: ";
33      double seconds{getuserinput()};
34      physicsFormula(x,seconds);
35
36  }
37
38
39
40  int main()
41  {
42      calculateHeight();
43      return 0;
44  }
```

Unto the next chapters.
~Cheers

nascardriver
November 10, 2019 at 2:45 am · Reply

Hi Gustav!

- Initialize your variables with brace initialization (Line 14).
- Use double literals for doubles (2.0 instead of 2, 0.0 instead of 0).
- "calculateHeight" is a poor name for what the function does. If also reads and prints.
- Inconsistent formatting. Use your editor's auto-formatting feature.
- `fallen` should be declared inside of the loop. There's no extra cost to this, but it limits `fallen`'s scope.
- Name your variables descriptively. `x` doesn't mean anything.

Vitaliy Sh.
November 24, 2019 at 7:41 am · Reply

Hi sir! Mr. Gustav at least is learning. While me stucked playng:

```
1   #include <iostream>
2   #include <array>
3   #include <limits>
4
5   constexpr auto streamMax {std::numeric_limits  <std::streamsize>  ::max()};
6   // XD ;)
7   constexpr std::array xd {32, 32, 32, 77, 65, 75, 69, 32, 76, 79, 86, 69, 32, 44,
8   // Public Place
9   static size_t pp{0};
10
11
12  auto hippi(char& ref_2_arg)
13  {
14      while (ref_2_arg != xd[pp])
15      {
```

```
        if(ref_2_arg > xd[pp]) ref_2_arg--;
        if(ref_2_arg < xd[pp]) ref_2_arg++;
    }
    std::array  s  {ref_2_arg, ' ', 'i', 's', ' '};

    std::cout  << s.begin();
    std::cout  << xd[pp] << '\n';
    pp++;
}

auto troll(char& ref_2_arg)
{
    constexpr int evilLimit {8};
    constexpr int arrayBeginning {0};

    char my_8_bytes[][evilLimit] {"", "World", "!"};

        // codes not tested outside Linux
    std::cout << '\x1B' << '7' << '\n';

    std::cin.getline(my_8_bytes[arrayBeginning],
            // Less is >
            (sizeof(my_8_bytes) / evilLimit),
            '\n');

    if (std::cin.fail())
    {
        std::cin.clear();
        std::cin.ignore(streamMax, '\n');
    }

    ref_2_arg++;
    for (char* i : my_8_bytes)
        std::cout << i << '\n';
}

auto main()->int
{
    enum thingy
    {
        hippies,
        trolles,
        etAll
    };

    char main_scope    {'A'};
    char troll_counter {'0'};
    std::array trollLounge  {*hippi, *troll};

    trollLounge.at(trolles)(troll_counter);

    if (troll_counter > '0')
    {
        for (int_fast16_t i{}; i < 26; i++)
            trollLounge.at(hippies)(main_scope);
    }

    // man terminfo, infocmp.
    std::cout << '\x1B' << '8' << '\x1B' << "[0;1;5;7m" << "Me there!\n";

    return EXIT_SUCCESS;
}
```

Jokes aside -- it hard to get serious and concentrated near 4.x;
In:

```
fallen = gravity_constant * x_seconds2 / 2
```

Is "/ 2" simulating air pressure et all? I'm tryed to parse the Wikipedia, but without succes.

nascardriver
November 24, 2019 at 7:48 am · Reply

I can't explain the formula, but Wikipedia says it ignores air resistance.

First equation here
https://en.wikipedia.org/wiki/Equations_for_a_falling_body#Equations

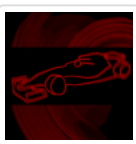
Says "air resistance is neglected" here
https://en.wikipedia.org/wiki/Equations_for_a_falling_body#Overview

Strato
November 6, 2019 at 8:58 pm · Reply

Hi, thank you so much for this fantastic tutorial, here's my code for review. I didn't figure it out how to exit the main function if the ball get to the ground before the 5 seconds.

```
#include "constants.h"
#include <iostream>

double heightOfBall (double h, int t) {
    return (h - (constants::gravity * t * t /2));
}

void printH (double h, int t) {
    if (heightOfBall(h, t) > 0)
     std::cout << "At " << t << " seconds, the ball is at height: " << heightOfBall(h, t) <
    else
     std::cout << "At " << t << " seconds, the ball is on the ground. \n";
}

int main () {
    std::cout << "Please input the height of the tower (in meters): ";
    double h {};
    std::cin >> h;

    printH(h, 0);
    printH(h, 1);
    printH(h, 2);
    printH(h, 3);
    printH(h, 4);
    printH(h, 5);

    return 0;
}

//constants.h

#ifndef CONSTANTS_H
#define CONSTANTS_H

/**
```

```
 *
 */
namespace constants
{
    inline constexpr double gravity { 9.8 };
}

#endif // CONSTANTS_H
```



nascardriver
November 7, 2019 at 2:49 am · Reply

Hi!

- Avoid abbreviations
- `constexpr` variables are always `inline`, you don't need to do it manually.
- Use double literals for doubles (0.0 instead of 0, 2.0 instead of 2).
- Line 9, 10: Duplicate call.

> I didn't figure it out how to exit the main function if the ball get to the ground before the 5 seconds
You could've used `if` and `return`, but that wouldn't look good. This quiz will come up later in the lessons about loops. You'll learn how to do it properly there.

knight
November 5, 2019 at 1:41 am · Reply

Hi, Alex Thanks your tutorial, I understand your most lessons but when I encounter question 3 and 4 which I do not know how to code it. I don't remember many concepts. Could you add more questions or exercises at every lesson? That may help newer to enhance the concept.

Vitaliy Sh.
November 24, 2019 at 8:01 am · Reply

Do you know how to touch-type? Me personally played a game to learn basics of it (Ktouch). Then i'm come there and start touch-typing whole lessons. Only then me noticed ("ninja style") that in 4.10 Quiz, Q#1: that range of problem is only a "single digit integer" *laughter.

Elis
November 4, 2019 at 6:06 am · Reply

Hi!

Managed to get a program working for Q#4 using a for loop, any feedback?
On line 19 and 21 I got a warning for inserting an int with a smaller width into a larger one and was recommended to cast it, is this a reasonable way to do it?

Thank you!

main.cpp:

```
#include <iostream>
#include "myConstants.h"

using namespace std;
using namespace myConstants;

```

```
int getInput()
{
    cout << "Enter the height of the tower in meters: ";
    int_least16_t height{};
    cin >> height;
    return height;
}

void calculatePrint(int height)
{
    for (int seconds = 0; seconds <= 5; seconds = seconds + 1)
    {
        if (height - (gravity * (static_cast<long>(seconds)* seconds) / 2) > 0)
        {
            double calculatedHeight{height - gravity * (static_cast<long>(seconds)* seconds
            cout << "After" << seconds << " seconds, the height of the ball is: " << calcul
        }
        else
        {
            cout << "After" << seconds << " seconds, the ball has hit the ground."; break;
        }

    }
}

int main()
{
    int height{ getInput() };
    calculatePrint(height);
}
```

myConstants.cpp:

```
#ifndef CONSTANTS_H
#define CONSTANTS_H

namespace myConstants
{
    constexpr double gravity { 9.8 };
}

#endif
```

nascardriver
November 4, 2019 at 6:16 am · Reply

Hi Elis!

- Initialize your variables with brace initializers.
- Limit your lines to 80 characters in length for better readability.
- Line 10: There's no reason to use an int16.
- Use double literals for doubles (2.0 instead of 2).
- Line 17: `++seconds`.

> On line 19 and 21 I got a warning
The appropriate cast is to type `double`, as that's what you're using.

Alex
October 11, 2019 at 12:34 pm · Reply

This program is able to display the fly of the ball from whatever height till it reaches the ground using functions only.

```
#include <iostream>
#include "Constants.h"

int getUserinput()
{
    std::cout << "Enter the height in meters: ";
    int x; std::cin >> x;
    return x;
}

double calculateHeight(int x, int y)
{
    double distance_fallen;
    distance_fallen = y - (constants::gravity * x * x / 2);
    return distance_fallen;
}

void elseFunction(int a, int s)
{
    void output(int a, int s);
    std::cout << "At " << s + 1 << " seconds the ball is at " << calculateHeight(s + 1, a)
    s = s + 1;
    output(a, s);
}

void startOutput(int a)
{
    void output(int a, int s);
    std::cout << "At 0 seconds the ball is at " << a << " meters\n";
    output(a, 0);
}

void output(int a, int s)
{
    if ((calculateHeight(s + 1, a)) <= 0)
        std::cout << "At "<< s+1 <<" seconds the ball has reached the ground.";
    else
        elseFunction(a, s);
}


int main()
{
    int a{ getUserinput() };
    startOutput(a);
}
```

Constants.h:

```
#ifndef _CONSTANTS_
#define _CONSTANTS_

namespace constants
{
    constexpr double gravity{ 9.8 };
}

#endif
```

nascardriver
October 12, 2019 at 2:44 am · Reply

Hi!

- Name your variables and function descriptively, `a`, `elseFunction`, `s` are meaningless.
- Initialize your variables instead of assigning to them.
- Use double literals for doubles (2.0 instead of 2). Line 14 performs integer arithmetic.
- `startOutput` and `elseFunction` are almost identical, they can be merged.
- Inconsistent formatting. Use your editor's auto-formatting feature.

You solved this quiz by using recursion (A function calling itself). This is discouraged, as it produces a big overhead (Wasted resources). You'll learn more about this and alternatives later. I think this quiz appears again in a different form, you'll have a head start :)

Adam
October 10, 2019 at 7:04 am · Reply

```
1  #include <iostream>
2  #include "constants.h"
3  double getHeight()
4  {
5      std::cout << "Enter the desired height of the tower: ";
6      double towerheight{};
7      std::cin >> towerheight;
8      return towerheight;
9  }
10
11 int main()
12 {
13     double towerheight{ getHeight() };
14     int time{ 0 };
15     double distancetoground{ towerheight };
16     while(distancetoground > 0)
17     {
18         std::cout << "At " << time << " seconds, the ball is at " << distancetoground << "
19         distancetoground = distancetoground - (gravity * (time+1*time+1)/2);
20         time++;
21     }
22     std::cout << "At " << time << " seconds, the ball is on the ground" << '\n';
23     return 0;
24 }
```

nascardriver
October 10, 2019 at 7:26 am · Reply

- Use ++prefix unless you need postfix++.
- Use double literals for doubles (0.0 instead of 0, etc.).
- Line 19 is missing parentheses. Your calculating `(time + time + 1)`.

Line 19 can use `distancetoground -= /* ... */` to prevent the repetition of `distancetoground`.

Adam
October 10, 2019 at 6:22 am · Reply

How is that?

```
#include <iostream>

double getUserInput()
{
    std::cout << "Enter a double value: ";
    double doublevalue{};
    std::cin >> doublevalue;
    return doublevalue;
}
char getOperator()
{
    std::cout << "Enter one of the following: +, -, *, or /: ";
    char math{};
    std::cin >> math;
    return math;
}
double calculateResult(double value1, char math,double value2)
{
    if (math == '+')
    {
        return value1 + value2;
    }
    else if (math == '-')
    {
        return value1 - value2;
    }
    else if (math == '*')
    {
        return value1 * value2;
    }
    else if (math == '/')
    {
        return value1 / value2;
    }
}

int main()
{
    while (true)
    {
        double value1{ getUserInput() };
        char math{ getOperator() };
        while ((math != '+') && (math != '-') && (math != '*') && (math != '/'))
        {
            math = getOperator();
        }
        double value2{ getUserInput() };
        std::cout << "The answer is: " << calculateResult(value1, math, value2) << '\n';
    }
    return 0;
}
```

nascardriver
October 10, 2019 at 6:26 am · Reply

- Inconsistent formatting. Use your editor's auto-formatting feature.
- Line 42-46 should be a do-while-loop to prevent the duplicate call to `getOperator`.

Looks nice otherwise, keep it up :)

Mike
October 2, 2019 at 7:41 am · Reply

Quick suggestion, apologize if its already been suggested.

Any chance of adding the chapter links from the main page to the side columns for all the pages? That way you can easily jump to a chapter for a quick refresher, which as a beginner happens too many times. You could easily click the chapter link, then just press back to return to where you were. Maybe even add the Previous, Index, and Next page to the top and/or very bottom of the page. This way you can just press Home or End to find them quickly.

The search helps, though having a visual of the chapters only a click away would certainly facilitate user experience.

Alex
October 3, 2019 at 1:42 pm · Reply

I've gotten a similar request several times. Are you envisioning just a list of lesson numbers (0.1, 0.2, 0.3) that link to the respective lessons? Part of the challenge I have is that the sidebar isn't very wide, so the lesson names won't fit.

Mike
October 3, 2019 at 2:02 pm · Reply

Using my 17" and 22" monitors, it just seemed like there is a lot of empty space on both sides that could be put to good use. I don't think the lesson numbers by them self would help much as they would definitely need a description of some sort. I'm no web designer for sure, but what if you combine the blanks sides to just one side, moving the content of the page to the left or right? And when you say the lesson names won't fit, will they not wrap to the next line or is that just bad web design?

Alex
October 4, 2019 at 1:57 pm · Reply

I'm thinking maybe there's some overlay solution that might work (e.g. something you hover over that opens up a menu). I'll look into it, but probably not soon, as my focus is more on updating content right now.

Thanks for the feedback!

Sri
September 18, 2019 at 9:48 pm · Reply

Please check any review my example:

```
1 #include<iostream>
2 #include "constants.h"
3
4 // get the input from user
```

```
5  double getTowerHeight( ){
6
7      std::cout << "Enter the height of the tower in meters : " ; //get the first input from
8      double input;
9      std::cin >> input;
10     return input;
11 }
12
13 //calculate and get the current height
14 double getCurrentHeight (const int sec, const double totalHeight){
15
16     double towerHeight =  totalHeight - (constants_s::gravity * ((sec*sec))/2);
17
18     return towerHeight;
19 }
20
21 // Print the output everysecs till the ball reached on ground
22 void displayHeight(int secs, double height){
23
24     if(height > 0){
25         std::cout << "At " << secs << " seconds, the ball is at height: " <<  height << " m
26     }
27     else
28     {
29         /* code */
30         std::cout << "At " << secs << " seconds, the ball is on the ground " << '\n';
31     }
32
33
34 }
35
36 int main ()
37 {
38
39     double towerHeight{ getTowerHeight()};
40
41     displayHeight(0,getCurrentHeight(0,towerHeight));
42     displayHeight(1,getCurrentHeight(1,towerHeight));
43     displayHeight(2,getCurrentHeight(2,towerHeight));
44     displayHeight(3,getCurrentHeight(3,towerHeight));
45     displayHeight(4,getCurrentHeight(4,towerHeight));
46     displayHeight(5,getCurrentHeight(5,towerHeight));
47
48 }
49
50 //constants.h
51 #ifndef CONSTANTS_H
52 #define CONSTANTS_H
53
54 namespace constants_s{
55
56     constexpr double gravity = {9.8};
57 }
58
59 #endif
```

nascardriver
September 19, 2019 at 12:37 am · Reply

- Initialize your variables with brace initializers.
- Limit your lines to 80 characters in length for better readability.

- Use double literals for doubles `2.0` instead of `2`. Same for `0`.
- `main`: Missing `return`-statement.
- You're using the same name style for variables and functions, this can lead to confusion.

Adriano
August 16, 2019 at 5:34 am · Reply

Hi !
First thank you so much for the time you spent on this tutorial, it's awesome.
Second, i'd like to understeand something, i was looking at your file costant.h in the 4th quiz about the physic quest:

```
#ifndef CONSTANTS_H
#define CONSTANTS_H

namespace myConstants
{
    constexpr double gravity { 9.8 }; // in meters/second squared
}
#endif
```

i'm sorry but i don't understeand why all this , couldn't i declare constexpr double gravity in my "main file"?
Wouldn't be easier and ok ?
i know you asked it as a part of the quiz to test the knowledges of your students, but could i do that?
And i'm sorry :
#ifndef CONSTANTS_H
#define CONSTANTS_H
why do i need those delcarations?
thank you so much sorry for my tricky question .

nascardriver
August 16, 2019 at 6:52 am · Reply

> couldn't i declare constexpr double gravity in my "main file"?
You you could.

> why do i need those delcarations?
They're header guards. See lesson 2.12.

Jose
August 5, 2019 at 4:01 pm · Reply

Hi, I wrote this code for quiz 3 but I don´t know if it is possible to perform the calculation the way it is right now, is there a way to convert the operator character into a real operator in the "printResult()" function?

```
#include <iostream>

double userValue() {
    double x{};
    std::cout << "Enter a floating point number(0.0) ";
    std::cin >> x;
    return x;
}

```

```
char userSign() {
    char x{'+'};
    std::cout << "Enter an operator (+,-,*,/) ";
    std::cin >> x;
    if (x == '+')
        return x;
    else if (x == '-')
        return x;
    else if (x == '*')
        return x;
    else if (x == '/')
        return x;
    else
        return false;
}

void printResult(double x,double y,char z) {
    if (z == false)
        std::cout << "\nYou did not enter a valid operator\n";
    else
        std::cout << "\n" << x << " " << z << " " << y << " = " << "???\n"; //Can this be s
}

int main() {
    double x{ userValue() };
    double y{ userValue() };
    char z{ userSign() };
    printResult(x,y,z);
    return 0;
}
```

nascardriver
August 6, 2019 at 3:00 am · Reply

- Limit your lines to 80 characters in length for better readability on small displays.
- Don't compare booleans to false/true.
- `userSign` returns a `char`. `false` is not a `char`.

> convert the operator character into a real operator
You can't do that. Write a function that checks what the operator is (`if` `else if`) and performs the appropriate calculation.

Jose
August 6, 2019 at 6:09 am · Reply

- Don't compare booleans to false/true.

What do you mean?

nascardriver
August 6, 2019 at 6:10 am · Reply

Line 27

```
if (z == false)
// should be
if (!z)

```

```
// or
if (z)
// then swap the cases
```

Jose
August 6, 2019 at 6:23 am · Reply

I see, thank you.

« Older Comments 1 ... 9 10 11