5.5 — While statements

BY ALEX ON JUNE 22ND, 2007 | LAST MODIFIED BY ALEX ON JANUARY 23RD, 2020

The **while statement** is the simplest of the four loops that C++ provides, and it has a definition very similar to that of an *if statement*:

```
while (expression)
    statement;
```

A while statement is declared using the **while** keyword. When a while statement is executed, the expression is evaluated. If the expression evaluates to true (non-zero), the statement executes.

However, unlike an *if statement*, once the statement has finished executing, control returns to the top of the while statement and the process is repeated.

Let's take a look at a simple while loop. The following program prints all the numbers from 0 to 9:

```
#include <iostream>

int main()
{
    int count = 0;
    while (count < 10)
        {
        std::cout << count << " ";
        ++count;
        }
    std::cout << "done!";

    return 0;
}
```

This outputs:

```
0 1 2 3 4 5 6 7 8 9 done!
```

Let's take a closer look at what this program is doing. First, count is initialized to 0. `0 < 10` evaluates to true, so the statement block executes. The first statement prints 0, and the second increments count to 1. Control then returns back to the top of the while statement. `1 < 10` evaluates to true, so the code block is executed again. The code block will repeatedly execute until count is 10, at which point `10 < 10` will evaluate to false, and the loop will exit.

It is possible that a while statement executes 0 times. Consider the following program:

```
#include <iostream>

int main()
{
    int count = 15;
    while (count < 10)
        {
        std::cout << count << " ";
        ++count;
        }
    std::cout << "done!";

    return 0;
```

```
}
```

The condition `15 < 10` immediately evaluates to false, so the while statement is skipped. The only thing this program prints is done!.

Infinite loops

On the other hand, if the expression always evaluates to true, the while loop will execute forever. This is called an **infinite loop**. Here is an example of an infinite loop:

```
#include <iostream>

int main()
{
    int count = 0;
    while (count < 10) // this condition will never be false
        std::cout << count << " "; // so this line will repeatedly execute

    return 0; // this line will never execute
}
```

Because count is never incremented in this program, `count < 10` will always be true. Consequently, the loop will never terminate, and the program will print "0 0 0 0 0 ..." forever.

We can declare an intentional infinite loop like this:

```
while (1) // or while (true)
{
  // this loop will execute forever
}
```

The only way to exit an infinite loop is through a return statement, a break statement, an exit statement, a goto statement, an exception being thrown, or the user killing the program.

Programs that run until the user decides to stop them sometimes intentionally use an infinite loop along with a return, break, or exit statement to terminate the loop. It is common to see this kind of loop in web server applications that run continuously and service web requests.

Loop variables

Often, we want a loop to execute a certain number of times. To do this, it is common to use a **loop variable**, often called a **counter**. A loop variable is an integer variable that is declared for the sole purpose of counting how many times a loop has executed. In the examples above, the variable *count* is a loop variable.

Loop variables are often given simple names, such as i, j, or k. However, naming variables i, j, or k has one major problem. If you want to know where in your program a loop variable is used, and you use the search function on i, j, or k, the search function will return half your program! Many words have an i, j, or k in them. Consequently, a better idea is to use iii, jjj, or kkk as your loop variable names. Because these names are more unique, this makes searching for loop variables much easier, and helps them stand out as loop variables. An even better idea is to use "real" variable names, such as count, or a name that gives more detail about what you're counting.

It is best practice to use signed integers for loop variables. Using unsigned integers can lead to unexpected issues. Consider the following code:

```
#include <iostream>

int main()
{
    unsigned int count = 10;

    // count from 10 down to 0
```

```
    while (count >= 0)
    {
        if (count == 0)
            std::cout << "blastoff!";
        else
            std::cout << count << " ";
        --count;
    }

    return 0;
}
```

Take a look at the above example and see if you can spot the error. It's not very obvious.

It turns out, this program is an infinite loop. It starts out by printing "10 9 8 7 6 5 4 3 2 1 blastoff!" as desired, but then goes off the rails, and starts counting down from 4294967295. Why? Because the loop condition `count >= 0` will never be false! When count is 0, `0 >= 0` is true. Then `--count` is executed, and count overflows back to 4294967295. And since 4294967295 is >= 0, the program continues. Because count is unsigned, it can never be negative, and because it can never be negative, the loop won't terminate.

Rule: Always use signed integers for your loop variables.

Iteration

Each time a loop executes, it is called an **iteration**.

Because the loop body is typically a block, and because that block is entered and exited with each iteration, any variables declared inside the loop body are created and then destroyed with each iteration. In the following example, variable x will be created and destroyed 5 times:

```
#include <iostream>

int main()
{
    int count = 1;
    int sum = 0; // sum is declared up here because we need it later (beyond the loop)

    while (count <= 5) // iterate 5 times
    {
        int x; // x is created here with each iteration

        std::cout << "Enter integer #" << count << ':';
        std::cin >> x;

        sum += x;

        // increment the loop counter
        ++count;
    } // x is destroyed here with each iteration

    std::cout << "The sum of all numbers entered is: " << sum;

    return 0;
}
```

For fundamental variables, this is fine. For non-fundamental variables (such as structs and classes) this may cause performance issues. Consequently, you may want to consider defining non-fundamental variables before the loop. This is another one of the cases where you might declare a variable well before its first actual use.

Note that variable count is declared outside the loop. This is necessary because we need the value to persist across iterations (not be destroyed with each iteration).

Often, we want to do something every n iterations, such as print a newline. This can easily be done by using the modulus operator on our counter:

```
#include <iostream>

// Iterate through every number between 1 and 50
int main()
{
    int count = 1;
    while (count <= 50)
    {
        // print the number (pad numbers under 10 with a leading 0 for formatting purposes)
        if (count < 10)
            std::cout << "0" << count << " ";
        else
            std::cout << count << " ";

        // if the loop variable is divisible by 10, print a newline
        if (count % 10 == 0)
            std::cout << "\n";

        // increment the loop counter
        ++count;
    }

    return 0;
}
```

This program produces the result:

```
01 02 03 04 05 06 07 08 09 10
11 12 13 14 15 16 17 18 19 20
21 22 23 24 25 26 27 28 29 30
31 32 33 34 35 36 37 38 39 40
41 42 43 44 45 46 47 48 49 50
```

Nested loops

It is also possible to nest loops inside of other loops. In the following example, the inner loop and outer loops each have their own counters. However, note that the loop expression for the inner loop makes use of the outer loop's counter as well!

```
#include <iostream>

// Loop between 1 and 5
int main()
{
    int outer = 1;
    while (outer <= 5)
    {
        // loop between 1 and outer
        int inner = 1;
        while (inner <= outer)
            std::cout << inner++ << " ";

        // print a newline at the end of each row
        std::cout << "\n";
        ++outer;
    }

```

```
19      return 0;
20  }
```

This program prints:

```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```

Quiz

1) In the above program, why is variable inner declared inside the while block instead of immediately following the declaration of outer?

Show Solution

2) Write a program that prints out the letters a through z along with their ASCII codes. Hint: to print characters as integers, you have to use a static_cast.

Show Solution

3) Invert the nested loops example so it prints the following:

```
5 4 3 2 1
4 3 2 1
3 2 1
2 1
1
```

Show Solution

4) Now make the numbers print like this:

```
        1
      2 1
    3 2 1
  4 3 2 1
5 4 3 2 1
```

hint: Figure out how to make it print like this first:

```
X X X X 1
X X X 2 1
X X 3 2 1
X 4 3 2 1
5 4 3 2 1
```

Show Solution


5.6 -- Do while statements


Index


5.4 -- Goto statements

C++ TUTORIAL | PRINT THIS POST

306 comments to 5.5 — While statements

« Older Comments 1 ... 3 4 5

Michael Sneberger
February 6, 2020 at 4:14 pm · Reply

For Quiz #4 I assumed that since we are in the while lesson we should use while loops but no if statements so I came up with this:

```
int main()
{

    int outer = 5;
    while (outer >= 1)
    {
        // loop between 1 and outer
        int inner = 5 - outer + 1;
        while (inner >= 1)
        {
            int spaces = 5 - inner;
            while (spaces > 0 && (inner + outer == 6)) {
                std::cout << "  ";
                --spaces;
            }

```

```
            std::cout << inner-- << " ";
        }

        // print a newline at the end of each row
        std::cout << "\n";
        --outer;
    }
    return 1;
}
```

Fabio Henrique
January 22, 2020 at 6:34 am · Reply

Hi there,
I came out with this solution to the last question, let me know what you guys think please ?!

```
#include <iostream>

int main()
{
    // number of rows to loop
    int rows{ 5 };
    while ( rows > 0 )
    {
        // number of numeric characters to print in every row
        int chars{ 0 };
        while ( chars < 5 )
        {
            // using the current row number to print our spaces
            int spaces{ --rows };
            while ( spaces > 0 )
            {
                std::cout << ' ';
                --spaces;
            }
            // printing our numbers until the end of the line
            int numbers{ ++chars };
            while ( numbers > 0 )
            {
                std::cout << numbers--;
            }
            std::cout << '\n'; //next row
        }
    }
    return 0;
}
```

Thank you guys !

nascardriver
January 22, 2020 at 6:39 am · Reply

You're not using the outer loop, it only ever runs 1 cycle. Otherwise the code is good :)

Fabio Henrique
January 22, 2020 at 7:34 am · Reply

Thank you nascar! And you're right, the rows loop were just running once... a better way to do this would be like this then ?!

```
#include <iostream>

int main()
{
    // number of numeric characters to print in every row
    int chars{ 5 };

    // number of rows to print
    int rows{ 0 };
    while ( rows < 5 )
    {
        // using the current number of characters to print our spaces
        int spaces{ --chars };
        while ( spaces > 0 )
        {
            std::cout << ' ';
            --spaces;
        }
        // using the current row number to print the numeric characters
        int numbers{ ++rows };
        while ( numbers > 0 )
        {
            std::cout << numbers--;
        }
        std::cout << '\n'; //next row
    }
    return 0;
}
```

And thank you and Alex for these tutorial series, they are the best I've found so far !

nascardriver
January 22, 2020 at 7:55 am · Reply

Yep, that's it!
Syntax highlighting works after refreshing the page if you edit your comment, you don't have to delete and re-post it.

Fabio Henrique
January 22, 2020 at 8:59 am · Reply

kkkk thanks for the tip, I was so annoyed that after every word I changed the code formatted text was gone. Lesson learned now XD cheers !!

Hakeem A Thomas
December 28, 2019 at 4:21 pm · Reply

Out of all the quiz question before this chapter, question 2 stumped me. I had to look at the solution. What do you recommend to do better problem solving in this situation. I am trying to become a better programmer and thinker.

nascardriver
December 29, 2019 at 2:58 am · Reply

Figure out where you're stuck, then read up on that topic.

Let's dissect the question
"Write a program that prints out the letters a through z along with their ASCII codes."
What do we need?
"Write a program" -> You know how to compile code and run the program.
"that prints" -> std::cout
"the letters a through z" -> You need a loop, because you want to repeat something. You also need to know about `char`.
"ASCII codes" -> ASCII codes and conversion to int

Out of these things, I assume you got stuck at the loop or the ASCII codes. ASCII and conversion to int was covered before, so you'd have to re-read lesson 4.11. Then there's the new loop, which was covered in this lesson. To make it work in the quiz, you'll have to remember that `char` is an integral type and you can compare them with < = >. If you don't remember that, you could have used the cast to int and compare the ints.

Hakeem
December 30, 2019 at 6:31 pm · Reply

I got stuck figuring out how to inverse the loop. I was trying to figure out how to get inner to equal 5 but display correct output. Turns out I only had to replace inner with a variable that contained 5. I got confused about how I could do that. Then in the solution, I just needed outer to be the variable

nascardriver
December 31, 2019 at 4:11 am · Reply

Oh that's question 3 then. Nested loops can be confusing, don't worry, you'll get the hang of it.

Alixsep
December 18, 2019 at 8:14 am · Reply

Hi nascardriver,
I am learning c++ with your website, this is my 2nd comment here,
I solved this lesson's quizzes at 5 A.M (by myself in 10 minutes).

```
1  #include <iostream>
2
3  int main()
4  {
5      std::cout << "Welcome to this shitty illuminati numbers (LMFAO!!) game v.1 made by Alix
6      start:
7      std::cout << "\nEnter an Integer: ";
8      int c;
9      std::cin >> c;
10     int a{ 1 };
11     while (a <= c)
12     {
13         int b{ c };
14         while (b >= 1)
15         {
16             if (b <= a)
17             {
```

```
                std::cout << b << ' ';
            }
            else
            {
                std::cout << "  ";
            }
            --b;
        }
        std::cout << '\n';
        ++a;
    }
    std::cout << "\nIf you wanna play again, enter 1, otherwise enter 0 to end this game! "
    int chcontinue;
    std::cin >> chcontinue;
    if (chcontinue == 1)
    {
        goto start;
    }

    std::cout << "\nTHANKS FOR PLAYING, BUT HONESTLY GO GET A BETTER GAME... ~ programmer c

    return 0;
}
```

I slept at 5:30 AM and woke up at 6:00 A.M went to college.

my college teacher thought us the for and while loops, and exactly asked the students the quiz 4 and I was able to solve it out of 90 students, very clean and exactly like your solution.

but then he just destroyed my whole character, and he said I am an asshole for solving the question LIKE THAT (TOO CONFUSING), he literally attacked me for straight up 10 minutes, and said that I can never become a programmer. He made me really sad.

He also said that whoever avoiding "using namespace std;" is an asshole.

the other student cannot even say the difference of double and integer, they don't know what is static_cast, function prototypes, enum classes, scopes, and ...

the college is skipping these lessons, and the students also retarded as fuck. Unfortunately I live in Iran, you may not believe but in all of them it is only me who can talk English a little bit.

I love your website, My only hope is ya'll. Thanks for your help, You literally changed my life forever, I hope you bests man. I am really sad, please tell me something that can motivate me to learn programming and make enough money to migrate and get out of Iran, I hate here.

nascardriver
December 18, 2019 at 8:45 am · Reply

That sounds like a horrible teacher. I'm not expecting teachers to be able to write good code, quite the opposite actually. But he shouldn't have demotivated you by telling you how bad your code was, he should've shown you how to improve it.

Since your college covered for-loops, you should've used one, because you knew how many times to loop will run before you start it:

```
#include <iostream>

int main() {

  for (int i{ 0 }; i <= 5; ++i)
  {
    for (int j{ 5 }; j > 0; --j)
```

```
    {
      if (j <= i)
      {
        std::cout << j << ' ';
      }
      else
      {
        std::cout << "  ";
      }
    }

    std::cout << '\n';
  }

  return 0;
}
```

A simple change that makes the code a lot easier. If your teacher wants you to do something, even if you know it's bad practice, just go along with it to get a good grade. This is unfortunate, because all other students will learn it wrong, but it doesn't sound like it's worth trying to convince your teacher otherwise. If you do your homework with friends, show them how it's done right before doing it teacher-style. The next time your teacher complains, your friends will know that he's at fault, not you.

When you're done with college and you're looking for a job, you'll be happy that you learned cpp on your own. If that kind of teacher is common in your college or country, make sure you write projects that you can show companies you apply at as a reference so that they know you're not a student who "learned" c++ in college.

I wish you to best of luck! If you have any coding questions, either from here or your college, don't hesitate to ask.

Alixsep
December 18, 2019 at 8:01 pm · Reply

nascardriver thanks you so much.
yes and i will use your beloving website, i owe you bro, you helped me a lot. is deitel and deitel a good book to read after your website?

nascardriver
December 19, 2019 at 1:51 am · Reply

I don't recommend books at all. They don't get updates, resulting in outdated content. If you want to read a book, choose a topic that doesn't change as quickly as a language, for example networking, threading, graphics, algorithms, etc.
Then read about that topic online in the latest version of C++.

kitabski
December 18, 2019 at 5:50 am · Reply

The final task seems so easy but it took me 3 days to solve it)))
And here is solution. I'll really appriciate if there are any comments.

```
#include <iostream>

int main()
{
```

```
5       using namespace std;
6       int outer {1};
7       while (outer<=5){ // runs outer loop 5 times printing 5 lines of code
8           char ch {' '};
9           int innerInt {1}, innerBlank {5};
10          while (innerBlank>outer){
11              cout << ch << ' ';
12              innerBlank--;
13          }
14          while (innerInt<=outer){
15              cout << innerInt << ' ';
16              innerInt++;
17              }
18          cout << '\n';
19          outer++;
20      }
21  return 0;
22  }
```

nascardriver
December 18, 2019 at 6:52 am · Reply

Declare one variable per line for better readability. Use --prefix and ++prefix unless you need postfix. You can change line 11 to

```
1   std::cout << "  ";
```

and remove `ch`.

kitabski
December 18, 2019 at 5:38 pm · Reply

nascardriver, thank you very much for your comments and suggestions!!!
I want to make a small suggestion myself. I think it would be a good idea if while posting a quiz it will be indicated which operators are allowed to be used. I mean, as in the task 4 of this unit i thought that solution should be based solely on while operator.
Thank you once again!!!!
And happy coming holidays!!!

nascardriver
December 19, 2019 at 1:47 am · Reply

You can use everything that's been covered up to that point, unless the quiz says otherwise.
Happy holidays!

Ryan
December 4, 2019 at 3:27 am · Reply

Is it more preferable to write \n with double speech marks("\n") or single speech marks('\n) or does it not matter

nascardriver
December 4, 2019 at 4:21 am · Reply

\n is a single character, so you should use single quotation marks '\n'.
If you use normal quotation marks, "\n" is treated as a string and the function you pass it to (`std::cout <<`) has to determine its length. This is unnecessarily slow.

Daki
November 13, 2019 at 9:09 am · Reply

Hello!

Firstly, I would like to thank the author for coming up with these tutorials! Hopefully it's normal for me to struggle a little and taking quite some time to understand as someone completely new to programming.

It took me some time to solve quiz 4 and I came up with a different solution. Would really appreciate it if I could receive some feedback on my code.

```
#include <iostream>

int main()
{
        int outer{ 1 };

    while (outer <= 5)
    {
        //Variable for printing numbers
        int inner{ outer };

        //Variables for spacing
        char blank{ };
        int count{ 5 - outer };

        //Print spaces first then numbers
        while (count > 0)
        {
            if (inner < 5)
            {
                std::cout << blank << " ";
                --count;
            }
        }

        //Print numbers after spacings, if they are needed
        while (inner >= 1)
        {
            std::cout << inner-- << " ";
        }

        //Adds 1 to the loop and prints new line
        std::cout << "\n";
        ++outer;
    }

    return 0;
}
```

nascardriver
November 14, 2019 at 1:03 am · Reply

Hello Daki!

> Hopefully it's normal for me to struggle a little and taking quite some time
It is. Once you learned a programming language, it's a lot easier to learn another.

- Inconsistent formatting. Use your editor's auto-formatting feature.
- Use single quotation marks for characters (' ' instead of " ", '\n' instead of "\n").
- Line 17 and 19 should be swapped. `inner` doesn't change inside the loop, so the `if` always evaluates to the same value.
- `inner` should be moved right above the inner loop. You can replace its use in line 19 (Which should be line 17) with `outer`.
- `blank` doesn't do anything. You can do `std::cout << ' ';` in line 21.

Daki
November 16, 2019 at 8:11 am · Reply

Hi nascardriver,

Thank you so much for your feedback! I see where my mistakes are now.
The usage of loops is still a little confusing for me but I think I'll get the hang of it once I have enough practice on them.

JamesWeb
November 4, 2019 at 4:54 pm · Reply

Hello my programm works, but I would like to know if there is any bad practice inside or something I could improve.

```
1  #include <iostream>
2
3  int main()
4  {
5      int outer = 1;
6      while (outer <= 5)
7      {
8          int inner = 5;
9          while (inner >= 1)
10         {
11             if (inner > outer)
12                 std::cout << "  ";
13             else
14                 std::cout << inner << ' ';
15
16             inner--;
17         }
18
19         std::cout << '\n';
20         ++outer;
21     }
22
23     return 0;
24 }
```

nascardriver
November 5, 2019 at 2:13 am · Reply

Hello

- Initialize your variables with brace initializers. Brace initialization provides better type safety.
- Use --prefix unless you need postfix--. Postfix operators are slower for large types.

Ethan.Apollo
August 17, 2019 at 4:49 am · Reply

Wanted to say that, I'm not sure you realise this but you said "a better idea is to use iii, jjj, or kkk" accidental racism? Also thanks for these tutorials there very helpful!

nascardriver
August 18, 2019 at 12:15 am · Reply

Using an acronym isn't racism.

Benur21
August 5, 2019 at 2:24 pm · Reply

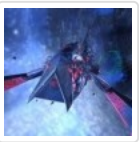
Why in the while syntax there is a ";"? While's don't have ; at the end

potterman28wxcv
August 14, 2019 at 8:41 am · Reply

It's not the while who has the ';', but the statement. I suppose that the actual statement here is "statement;" and not just "statement".

Benur21
August 29, 2019 at 6:36 am · Reply

Usually instead of "statement" we use a block of statements between "{}". How many ';'s are there now?



potterman28wxcv
August 29, 2019 at 6:52 am · Reply

As you just wrote it's not a statement then but a block of statements. So the comment is irrelevant since the author wrote about a statement, not a block of statements..

If they meant a block of statements surely they would have written :

```
while (expression)
    statement-block
```

Or

```
while (expression)
{
    /* statements */
}
```

Parsa
July 28, 2019 at 1:34 pm · Reply

this is irrelevant to this topic, but can you output a function call using cout?

And what does "CL.exe exited with code 2" mean?

And how do I delay how many times the while statements execute every second or something? I know there is a function called sleep but I don't know how it works or how to use it.

nascardriver
July 29, 2019 at 3:50 am · Reply

> can you output a function call using cout

```
1 std::cout << myFunction() << '\n';
```

> And what does "CL.exe exited with code 2" mean?
You'll have to google for that one.

> delay

```
1 #include <chrono>
2 #include <thread>
3
4 // ...
5
6 std::this_thread::sleep_for(std::chrono::milliseconds(123));
```

Don't use `Sleep`, it only exists on Windows.

Parsa
July 29, 2019 at 11:03 am · Reply

Ok, thanks.

Since it is a function it's not possible to use a 'using' statement to assign a different name for it.

Is there a similar thing for functions?

nascardriver
July 29, 2019 at 11:31 am · Reply

What you said doesn't fit to your previous question. Can you show and example of what you want to do?

Parsa
July 29, 2019 at 12:51 pm · Reply

```
1 using sleep = std::this_thread::sleep_for;
```

(except this doesn't work).

nascardriver
July 29, 2019 at 11:24 pm · Reply

There's no real aliasing for functions, you can only copy the function pointer (Covered later).

```
constexpr auto sleep{ static_cast<void (*)(const std::chrono::millisec

sleep(std::chrono::seconds(1));
```

If `std::this_thread::sleep_for` wasn't overloaded (Also covered later), you could do

```
constexpr auto sleep{ std::this_thread::sleep_for };
```

ceeEss
July 25, 2019 at 3:20 am · Reply

here's how I did it.

```
#include <iostream>

int main() {
    int count{ 1 };                          //outer count starts at 1
    while (count <= 5) {
        int iCount(5-count);                //innerCount 5 minus whatever count is currentl

        while (iCount>=1) {                  //when innerCount is greater than 1, print t
            std::cout << "  ";
            --iCount;
        }

        int iCount2{ count };                //innerCount2 = count because count is in asce
        while (iCount2>=1) {
            std::cout << iCount2--<<" ";    //after the first "while" prints all the requir
        }
        std::cout << std::endl;              //end a line after both "while" are done.
        ++count;                            //increase count by one after line ends.
    }
    return 0;
}
```

Westly LaFleur
June 22, 2019 at 12:13 am · Reply

I'm pretty shit at math, so - for the first time ever - I decided to put my humble programming skills to work on a real problem. Insofar as I can tell, the program is working as intended.

So, bearing in mind that I'm working through these lessons sequentially, did I do anything wrong? Could I have written my code more efficiently or used better logic?

Just so you know that I'm paying attention: I realize that goto statements aren't generally advisable, but I REALLY thought it was a good solution here, given the minor scope of the project, and the need to revisit the moment post-seed. I also realize that infinite loops aren't generally advisable, but again, I thought it fit the purpose.

Please let me know if I'm mistaken. Once again, thanks @NascarDriver & @Alex, you guys are doing a service to humanity running this site.

```
#include <iostream>
#include <cmath>
#include <cstdlib>
#include <ctime>

// this program generates multiplication problems to quiz the
// user much like a digital multiplication 'flash card'
```

```
8
9   int main()
10  {
11      int total_correct{};
12
13      while (1) // deliberately infinite loop, although I'd like an 'exit if user presses (es
14      {
15          std::srand(time(nullptr));
16          int x{ std::rand() % static_cast<unsigned int>(9 + 1) };
17          int y{ std::rand() % static_cast<unsigned int>(9 + 1) };
18
19      tryAgain:
20          std::cout << x << " * " << y << ": ? " << std::endl;
21
22          int answer{std::cin >> answer}; // this doesn't feel right, is this incorrect?
23
24          //std::cin >> answer
25
26          if (answer == (x * y))
27          {
28              ++total_correct;
29              std::cout << "Nice job! You've gotten " << total_correct << " Answers right!" <
30          }
31          else
32          {
33              std::cout << "Whoops, that was incorrect. Try again!" << std::endl;
34              goto tryAgain;
35          }
36      }
37      return 0;
38  }
```

nascardriver
June 22, 2019 at 2:05 am · Reply

- Line 13: Conditions are bools, use `while (true)`.
- Don't use `time`, use `std::time`.
- `123` is an int, `123u` is an unsigned int.
- Don't use `goto`. Add another `while (true)` loop and break it if the answer is correct.
- Line 22: No, it's not. It shouldn't even compile. Make sure you followed lesson 0.10 and 0.11. You can't initialize variables with a value read from `std::cin` without writing a wrapper function.
- Line 11, 28: If you're using the initial value of a variable, explicitly initialize the variable to that value (0). This won't change anything in you program, but makes your code easier to follow.
- Seed random number generators only once. If the user solves the quiz within 1 second, the same "random" numbers will be generated.

> Could I have written my code more efficiently
You're calculating `x * y` in every cycle of the `goto`. This is unnecessary, since neither `x` nor `y` changed their values.

Dhananjay Singh
June 1, 2019 at 3:36 pm · Reply

Hi, thanks for your tutorials I have a different solution to your last quiz question
Can you just go through it and tell me your feedbacks.

```
1   #include <iostream>
2
```

```
3  int main()
4  {
5      int outer = 1;
6
7      while(outer<6){
8
9          int inner{outer};
10         int another{5};
11             while(another>inner){
12                 std::cout<<"  ";
13                 --another;}
14         while(inner>0){
15             std::cout<<inner<<" ";
16             --inner;
17
18         }
19         std::cout<<'\n';
20         ++outer;
21     }
22 }
```

thank you both;

nascardriver
June 1, 2019 at 11:41 pm · Reply

* Line 5: Initialize your variables with brace initializers.
* Inconsistent formatting. Use your editor's auto-format feature.
* Use single quotation marks for characters (' ' instead of " ").
* Missing return-statement. Although not required in `main`, it should be added for consistency.

Dhananjay Singh
June 4, 2019 at 6:20 am · Reply

thanks for the feedback and your tutorials

nascardriver
June 4, 2019 at 8:33 am · Reply

You're welcome! The tutorials aren't mine though, all credit goes to @Alex.

Ankit
May 28, 2019 at 2:31 am · Reply

I had a different approach for the last question than the one given in the solution, can anyone tell me if it is a good practice to code like I have done below.

#include<iostream>

int main()
{
  int outer = 1;
  while(outer<=5)
  {
    int inner = 5;
    while(inner>outer)

```
    {
      std::cout<< "  ";
      inner--;
    }
    while(inner>0)
      std::cout<<" "<<inner--;
    std::cout<<"\n";
    outer++;
  }
}
```

nascardriver
May 28, 2019 at 3:14 am · Reply

Hi!

Your solution is better, because it doesn't perform a comparison in ever cycle of the inner loop.

* Initialize your variables with brace initialization.
* Use --prefix unless you need postfix--.
* Use single quotation marks for chars.
* Missing return-statement.

Please use code tags when posting code.

Ankit
May 29, 2019 at 8:18 am · Reply

Thanks a lot for the feedback. Keep up the great work

Napalm
May 13, 2019 at 8:16 am · Reply

I had a feeling this would be wrong because of having those numbers in there. I didn't realise you could increment a char. I'll certainly remember now though so I guess I still learned :)

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

int main()
{
    std::cout << "Letter\tASCII Number\n\n";

    int count{ 97 };
        while (count >= 97 && count <= (97 + 25))
        {
            std::cout << static_cast<char>(count) << "\t" << count << "\n";
            ++count;
        }
        std::cout << "Done!\n";
    return 0;
}
```

Harshitha.R
May 10, 2019 at 2:48 am · Reply

Hello there , could anyone help me as to how this program works?
#include <iostream>

// Loop between 1 and 5
int main()
{
  int outer = 1;
  while (outer <= 5)
  {
    // loop between 1 and outer
    int inner = 1;
    while (inner <= outer)
      std::cout << inner++ << " ";

    // print a newline at the end of each row
    std::cout << "\n";
    ++outer;
  }

  return 0;
}

Miquel
April 23, 2019 at 7:52 am · Reply

Hello again!

I finished the last quiz and had a different solution then the ones I found here.
It works as expected, but I'd love to know if there's I could do better or any potential dangerous habits.

```
1  int main()
2  {
3      int outer{ 1 };
4
5      while (outer <= 5)
6      {
7          int inner{ outer - 5 };
8
9          while (inner <= 0)
10         {
11             std::cout << "  ";
12             ++inner;
13         }
14         while (inner <= outer)
15         {
16             std::cout << inner++ << " ";
17         }
18
19         std::cout << "\n";
20         ++outer;
21     }
22
23     return 0;
24 }
```

Again, thanks Alex and nascardriver for all the help! I'm having a great time learning here.

nascardriver
April 23, 2019 at 8:36 am · Reply

Hello!

If you want to, you can merge line 12 into line 9.
Line 16 and 19 should print chars. This allows @std::cout::operator<< to use an optimized version to print (Since it doesn't have to check the text's length when it's just a char).

Red Lightning
April 22, 2019 at 8:21 am · Reply

```
1  // Thanks to Shiva for this solution
2  #include <iostream>
3
4  int main()
5  {
6      // There are 5 rows, we can loop from 1 to 5
7      int outer = 1;
8
9      while (outer <= 5)
10     {
11         // Row elements appear in descending order, so start from 5 and loop through to 1
12         int inner = 5;
13
14         while (inner >= 1)
15         {
16             // The first number in any row is the same as the row number
17             // So number should be printed only if it is <= the row number, space otherwise
18             if (inner <= outer)
19                 std::cout << inner << " ";
20             else
21                 std::cout << "  "; // extra spaces purely for formatting purpose
22
23             --inner;
24         }
25
26         // A row has been printed, move to the next row
27         std::cout << "\n";
28
29         ++outer;
30     }
31
32     return 0;
33 }
```

--The code in question 4, am I the only one spotting magic numbers (5)?

« Older Comments 1 ... 3 4 5