5.x — Chapter 5 summary and quiz

BY ALEX ON SEPTEMBER 11TH, 2011 | LAST MODIFIED BY NASCARDRIVER ON DECEMBER 20TH, 2019

Quick review

Always use parentheses to disambiguate the precedence of operators if there is any question or opportunity for confusion.

The arithmetic operators all work like they do in normal mathematics. The Modulus (%) operator returns the remainder from an integer division. Prior to C++11, beware of integer division and modulus using negative numbers.

The increment and decrement operators can be used to easily increment or decrement numbers. Avoid the postfix versions of these operators whenever possible.

Beware of side effects, particularly when it comes to the order that function parameters are evaluated. Do not use a variable that has a side effect applied more than once in a given statement.

The comma operator can compress multiple statements into one. Writing the statements separately is usually better.

The conditional operator is a nice short version of an if-statement, but don't use it as an alternative to an if-statement. Only use the conditional operator if you use its result.

Relational operators can be used to compare floating point numbers. Beware using equality and inequality on floating point numbers.

Logical operators allow us to form compound conditional statements.

Quiz time

Question #1

Evaluate the following:

a) (5 > 3 && 4 < 8)

Show Solution

b) (4 > 6 && true)

Show Solution

c) (3 >= 3 || false)

Show Solution

d) (true || false) ? 4 : 5

Show Solution

Question #2

Evaluate the following:

a) 7 / 4

Show Solution

b) 14 % 5

Show Solution

Question #3

Why should you never do the following:

a) int y = foo(++x, x);

Show Solution

b) double x = 0.1 + 0.1 + 0.1; if (x == 0.3) return true; else return false;

Show Solution

c) int x = 3 / 0;

Show Solution

O.1 -- Bit flags and bit manipulation via std::bitset

Index

5.7 -- Logical operators

C++ TUTORIAL | PRINT THIS POST

39 comments to 5.x — Chapter 5 summary and quiz

koe
December 20, 2019 at 1:10 am · Reply

Hi,

This summary doesn't mention commas and the conditional operator (section 5.5).

nascardriver
December 20, 2019 at 1:31 am · Reply

Well spotted, comma and conditional operator added.

P
October 3, 2019 at 8:13 am · Reply

Hey Alex,

First of all, thank you so much for making this tutorial. It's been so very helpful. I totally appreciate all the work you've put into making this site.

I had a question:

#Question 3 b)

```
  float x = 0.1 + 0.1; if (x == 0.2) return true; else return false;
```

I changed this to

```
    float x = 0.1f + 0.1f; if (x == 0.2f) return true; else return false;
```

It returns true.

I don't understand why it's returning true exactly. Is a implicit conversion to double causing the program to return false? If we used the "f" suffix here, as you cover in a previous chapter, does it eliminate the rounding error? So it should be okay if we compare float numbers as long as we remember the f suffix? Could you please explain this?

I'm sorry if you've talked about this already. I'm kind of new to programming (well, not exactly, but I never took it seriously before).

Thank you in advance for your time!

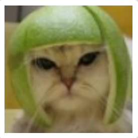
Alex
October 3, 2019 at 1:59 pm · Reply

I updated the quiz question a bit, both to get rid of the mismatch between the float variable and the double literals, and to actually make the statement break as expected.

Strangely enough 0.1 + 0.1 == 0.2 is true, but 0.1 + 0.1 + 0.1 == 0.3 is false, which just goes to highlight the unpredictable nature of floating point precision errors.

Mhz93
September 30, 2019 at 7:56 am · Reply

in the solution for last part of Question #2, we can see: "14 % 5 = 2 remainder 4, so this equals 4." which I guess it should be written: "14 / 5 = 2 remainder 4, so this equals 4."
I mean the operand "%" must be changed to "/"



Alex

September 30, 2019 at 9:18 am · Reply

Fixed. Thanks!

Sam

August 3, 2019 at 3:37 pm · Reply

Here is my updated solution with a struct (Question #2)

```
#include <iostream>
#include <random>
#include <ctime>
#include <limits>

auto streammax { std::numeric_limits <std::streamsize>::max() };

namespace Seed {
  std::mt19937 mersenne(time(nullptr));
}

int randomNumber(int min, int max) {
  std::uniform_int_distribution <> num(min, max);
  return num(Seed::mersenne);
}

struct Variables {
  int correct_value { randomNumber(1, 100) };
  int user_lives { 8 };
  int guess_count { 1 };
  int guess { 0 };
};

char playAgain() {
  char response;

  do {
    std::cout << "\nWould you like to play again? (y/n): ";
    std::cin >> response;

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

  } while ((tolower(response)) != 'y' && (response != 'n'));

  return response;
}

void youLose(int answer) {
  std::cout << "\n\nYou lose! Answer: " << answer << '\n';
}

void checkGuess(int guess, int correct_value) {
  if (guess > correct_value)
```

```
    std::cout << "\nToo high.";
  else if (guess < correct_value)
    std::cout << "\nToo low.";
  else
    std::cout << "\nYou win!";
}

void startScreen(int lives) {
  if (lives > 1) {
    std::cout << "\n\nGuess the correct number." <<
      " You have " << lives << " lives left.\n\n";
  }
  else {
    std::cout << "\n\nGuess the correct number." <<
      " You have 1 life left.\n\n";
  }
}

int main() {
  std::cout << "Welcome to Hi-Lo.";

  Variables game;

  do {
    if (game.user_lives == 0) {
      youLose(game.correct_value);

      char retry = playAgain();

      switch (retry) {
        case 'y': {
          game.correct_value = { randomNumber(1, 100) };
          game.user_lives = { 8 }; // +1 life, or it starts at 6 lives (not sure why)
          game.guess_count = { 1 };
          game.guess = { 0 };

          std::cout << "\nWelcome to Hi-Lo.";
        }
        continue;

        case 'n': {
          std::cout << "\nThank you for playing.";
        }
        break;

        default: {
          std::cout << "An error occurred.";
        }
      }
    }
    else {
      startScreen(game.user_lives);

      std::cout << "Guess #" <<
        game.guess_count++ << ": ";
      std::cin >> game.guess;

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

      checkGuess(game.guess, game.correct_value);
    }
  } while ((game.guess != game.correct_value) && (game.user_lives-- != 0));
}
```

nascardriver
August 4, 2019 at 8:09 am · Reply

"Variables" is a bad name as it doesn't tell anything about the struct's contents.
`guess` shouldn't be stored in the struct, it's a one-time use variable and there's no need to remember what it was after it has been processed.
Packaging the variables is a good start. It's not all to useful with what you know so far, but you'll learn about more ways of fitting things into objects later.

ethano
March 3, 2019 at 2:37 pm · Reply

For the 3rd question in the quiz, clarifying that the binary is unsigned would be nice.

Alex
March 4, 2019 at 9:16 pm · Reply

Good point. Amended. Thanks for pointing that out!

Trevor
January 16, 2018 at 3:41 am · Reply

Quiz 5d - may actually work correctly and return true, even though neither 0.1 nor 0.2 will be represented exactly. If the compiler and hardware are consistent, the only difference between the representations of 0.1 and 0.2 will be a change of 1 in the exponent (indicating a change by a power of 2), and there should not be any additional errors introduced by the addition calculation given that the same number is being added to itself. I suggest replacing "will" with "may".

Alex
January 18, 2018 at 5:37 pm · Reply

Done. Thanks for the feedback.

AMG
June 10, 2017 at 10:26 am · Reply

Alex,
Would advise everyone to execute Quiz 5) on computer. Comments on 5d): In float x = 0.1+0.1;, float x is assigned double 0.1+0.1, and because of double precision if statement is false. However, float x = 0.1f + 0.1f; if (x == 0.2f) -> true.

C++ learner
May 21, 2017 at 2:10 pm · Reply

Hi! Coach. Thak you again for this tutorial.
I have a question. Do we have to go through all the chapters in c++ to be really good at it? Let's say I want to make games or robots. Do I have to do all the chapters or I should focus on specific chapters?
I have an idea and I wanted to share it. You can try to add interactive c++ competitions for fun and people will test their skill at the same time.

Alex
May 21, 2017 at 3:14 pm · Reply

No, you don't have to do all the lessons or chapters to be good. But the more you know, the better you'll be, as a lot of stuff is inter-related, and the more tools you have in your toolbox the more likely it is you'll know which tool you should use for a given task.

Ultimately its up to you to determine whether you're likely to use something. If you think maybe not, spend less time on that lesson and come back later if it turns out not to be the case.

Ilker
August 12, 2016 at 10:23 pm · Reply

For quiz question 5c, the wording of "Why should you never do the following" might be a bit misleading, when

```
if(x % 2 == 0)
```

is valid for even/odd checking even if x is negative.

Alex
August 14, 2016 at 9:25 am · Reply

Fair point, I put added a parenthetical note to the answer.

bert
August 9, 2016 at 3:55 pm · Reply

Did you mean to use "particularly" in the following:

" Beware of side effects, particular when it comes to the order that function parameters are evaluated."

Alex
August 10, 2016 at 7:33 pm · Reply

Yep, fixed. Thanks!

functionLover
August 1, 2016 at 9:19 pm · Reply

alex! you should perhaps consider adding donation links to this page.
this tutorial format is far more helpful than many books out there.

Alex
August 3, 2016 at 1:08 pm · Reply

There are donation links in the **Support** section of the site, for readers who are interested in helping contribute to defray hosting costs.

SyJ
February 28, 2016 at 4:09 am · Reply

Thanks, this really helped me. I didn't move on until I understood this lesson and the one before it. Please note you mention enum (which I don't know yet) I think you previously used an enum but removed it. Please can you clarify this and remove that part in the lesson thank you

Aryan
September 7, 2015 at 7:38 pm · Reply

Hey Alex!
Can you please explain the 1st question d part? I don't know how and where did that bitwise OR operator got in between 4 and 5.

Alex
September 7, 2015 at 8:29 pm · Reply

Just a typo. It's fixed now.

Jarvis Chen
September 3, 2015 at 4:12 pm · Reply

In the comprehensive quiz, there is a line break missing.

...
1) Evaluate the following:

a) (5 > 3 && 4 < 8) b) (4 > 6 && true)
...

Just a small thing, but if you can make something perfect, why not? ;)

EDIT: Awesome website btw!!! I am looking forward to more quizzes!

Alex
September 4, 2015 at 12:33 am · Reply

Thanks! The site sometimes decides to interpret < as an html tag instead of a less than, and that screws up the formatting.

Robert W
November 25, 2017 at 7:07 am · Reply

Just a note, some of the formatting is not consistent in the quiz sections. Some questions have a line before part a) and some don't. Example:
2) Answer the following:

a) 7 / 4
b) 14 % 5

3) Convert the following from binary to decimal:
a) 1101
b) 101110

Alex
November 26, 2017 at 9:30 am · Reply

Fixed, thanks!

nj
August 28, 2015 at 2:14 am · Reply

hii alex !! kudos for this awesome work of yours.

but why don't we have quizzes after chapter 6?
please add them soon.

Alex
August 31, 2015 at 4:01 pm · Reply

I'm working on adding them as I rewrite the lessons but I can only write so fast. :(

nj
September 1, 2015 at 10:10 am · Reply

:(

Bogdan
August 26, 2015 at 7:11 am · Reply

In C++11 the modulo division for negatives is defined now.

Alex
August 26, 2015 at 12:41 pm · Reply

Yes, I noted this in the lesson on modulus and division but forgot to update the quiz. Thanks for the note.

Yongxin
March 14, 2015 at 2:29 am · Reply

It turns out we don't have further quizzes. How to access more?

Alex
March 19, 2015 at 4:31 pm · Reply

I'm intending to add more as I update the articles.

Gocha

February 4, 2015 at 7:27 am · Reply

Great tutorial!
Whether I was learning c++ or other programming language somewhere else, everything seemed much complicated than on this site. I would say that this is the site where everything is detailed and well explained, even through c++ is really hard to learn/teach, I follow this tutorial without any troubles so far, I wish you all the best, my friend!

Speeds03
May 9, 2013 at 9:42 am · Reply

What I don't understand from the quiz is #4; whether or not i'm using 4 bit, 8 bit etc... How would I know that with 15 decimal, I should start of with "15 >= 8" instead of "15 >= 16". The same goes for 53 decimal (or any other number for that matter...

Thomas
October 12, 2013 at 9:36 pm · Reply

You must always think in bytes allocated for the given number, because that is the smallest unit that can be addressed in memory. If you allocated an int for it, that should mean 4 bytes (at least, on 32 bit architectures). In a case like this quiz, you only need to check whether the given number can be stored in 1 byte (8 bits, like a char or bool). If it's too big for that, then you must go up to 2 bytes (like short int). If it's still too big, then 4 bytes (int) and so on.

Alex
March 19, 2015 at 4:31 pm · Reply

When using method 2 of converting a decimal to binary, always start with the largest power of 2 that is smaller than your starting decimal number. So if your number is 15, start with 8 (2^3). If your number is 53, start with 32 (2^5).