1.x — Chapter 1 summary and quiz

BY ALEX ON JUNE 3RD, 2007 | LAST MODIFIED BY ALEX ON APRIL 10TH, 2019

Quick Summary

A **statement** is a type of instruction that causes the program to perform some action. Statements are often terminated by a semicolon.

A **function** is a collection of statements that execute sequentially. Every C++ program must include a special function named *main*. When you run your program, execution starts at the top of the *main* function.

Preprocessor directives tell the compiler to perform a special task. In this chapter, we use them to #include <iostream>, which allows us to access the input/output routines in the standard library.

The rules that govern how elements of the C++ language are constructed is called a **syntax**. A **syntax error** occurs when you violate the grammatical rules of the language.

Comments allow the programmer to leave notes in the code. C++ supports two types of comments. Line comments start with a // and run to the end of the line. Block comments start with a /* and go to the paired */ symbol. Don't nest comments.

You can use comments to temporarily disable lines or sections of code. This is called commenting out your code.

Data is any sequence of symbols that can be interpreted to mean something. A single piece of data, stored somewhere in memory is called a **value**.

A variable is a named piece of memory that we can use to store values. A variable's name is called an **identifier**. In order to create a variable, we use a statement called a **definition statement**. When the program is run, each defined variable is **instantiated**, which means it is assigned a memory address.

A **data type** tells the compiler how to interpret a piece of data into a meaningful value. An **integer** is a number that can be written without a fractional component, such as 4, 27, 0, -2, or -12.

Copy assignment (via operator=) can be used to assign an already created variable a value.

Initialization can be used to give a variable a value at the point of creation. C++ supports 3 types of initialization: copy initialization, direct initialization, and uniform initialization.

You should prefer uniform initialization over the other initialization forms, and prefer initialization over assignment.

Although you can define multiple variables in a single statement, it's better to define and initialize each variable on its own line, in a separate statement.

std::cout and operator<< allow us to output an expression to the console as text. **std::endl** outputs a new line character, forcing the console cursor to move to the next line. **std::cin** and operator>> allow us to get a value from the keyboard.

A variable that has not been given a value is called an **uninitialized variable**. Trying to get the value of an uninitialized variable will result in **undefined behavior**, which can manifest in any number of ways.

C++ reserves a set of names called **keywords**. These have special meaning within the language and may not be used as variable names.

A **literal constant** is a fixed value inserted directly into the source code. Examples are 5 and "Hello world!".

An **operation** is a mathematical calculation involving zero or more input values, called **operands**. The specific operation to be performed is denoted by the provided **operator**. The result of an operation produces an output value.

Unary operators take one operand. **Binary** operators take two operands, often called left and right. **Ternary** operators take three operands.

An **expression** is a combination of literals, variables, operators, and function calls that are evaluated to produce a single output value. The calculation of this output value is called **evaluation**. The value produced is the **result** of the expression.

An **expression statement** is an expression that has been turned into a statement by placing a semicolon at the end of the expression.

Programming is hard, and your programs will rarely come out perfect (or close to it) the first time. Get your programs working first, then refine them into something great.

Quiz time

Question #1

What is the difference between initialization and assignment?

Show Solution

Question #2

When does undefined behavior occur? What are the consequences of undefined behavior?

Show Solution

Question #3

Write a program that asks the user to enter a number, and then enter a second number. The program should tell the user what the result of adding and subtracting the two numbers are.

The output of the program should match the following (assuming inputs of 6 and 4):

```
Enter an integer: 6
Enter another integer: 4
6 + 4 is 10.
6 - 4 is 2.
```

Show Solution

2.1 -- Introduction to functions


Index


1.10 -- Developing your first program

C++ TUTORIAL | PRINT THIS POST

578 comments to 1.x — Chapter 1 summary and quiz

« Older Comments 1 ... 6 7 8


Vitaliy Sh.
January 25, 2020 at 7:03 pm · Reply

Hi Sires!

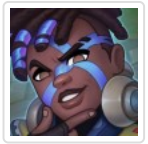
```
1  // There was suggestion somewhere (in comments or lesson)...
2  //          ``the block comments. The // inside of /**/ are fine.``
3  "Don't nest comments."
4
5  // ('\n' also can be used directly with std::cout),
6  "std::endl outputs a new line character,"
7
8  // (if a value gets assigned to an uninitialized variable, the variable stops being uniniti
9  //                              known value upon creation
10 "A variable that has not been given a value is called an uninitialized variable."
11
12 //             to enter two numbers in turn (x, y).
13 "asks the user to enter a number, and then enter a second number."
```


Kringle
January 4, 2020 at 7:44 pm · Reply

It was really cool experimenting with how certain code works, with learning the correct placement of "\n" will make the end result easily readable from each other as the spacing makes it way better

in my opinion. I love reading and learning from this tutorial so far, and I hope in few weeks time I can make something like a simple game of snake. Thank you for making this tutorial <3.

```
#include <iostream>

int main()
{
    int x{}, y{};

    std::cout << "Enter an interger value: ";
        std::cin >> x;
    std::cout << "\n"<< "Enter a second interger value: ";
        std::cin >> y;

        //There is a choice between numerical based answers and texted based answers.

        std::cout << "\n" << x << "/" << y << " = " << x / y << "\n" << "\n"; //Numerical B

        std::cout << x << '*' << y << " = " << x * y << "\n" << "\n"; //Numerical Based Ans

        std::cout << "This is the answer if x and y are divided: " << x / y << "\n" << "\n"


        std::cout << "This is the answer if x and y are multiplied: " << x * y << "\n" << "

        return 0;
}
```



VAC
January 30, 2020 at 11:42 pm · Reply

When not inside an actual sentence, you should use '\n' instead of "\n", the first will be highlighted (just more readable, "\n" works fine btw).

```
std::cout << "Just writing something. \n" // That's just fine because it excludes the
std::cout << '\n'                         // That way your IDE will highlight it for u
std::cout << "\n"                         // This one here will be like any other text
```

When you have a lot of integers that's (personally) not a good way to organize, so, I don't really recommend you become used to it.

```
int x{}, y{};
```

Liked that 2 lines of yours (cin as a "child" to cout).

```
std::cout << "Enter an integer value: ";
    std::cin >> x;
```

nascardriver
January 31, 2020 at 12:59 am · Reply

Highlighting aside, "\n" is slower for any character. The function you're calling treats it as a string and has to run a loop. If you want to use a single character, use single quotation marks.

gunslinger
December 28, 2019 at 4:17 pm · Reply

this chapter was great and i'd like to thank you for all your work. when do i know that it's absolutely important to initialise a variable and when can i just declare and leave it be?

nascardriver
December 29, 2019 at 2:48 am · Reply

When you read from a variable before writing to it, you have to initialize it.
In all other cases, you can omit the initialization. When you pass a variable to a function and want to know if the function reads from the variable, you'll have to read the documentation and possibly implementation.
Omitting an initialization can improve performance, but can decrease maintainability of your code.
Unless you're creating a container (containing a lot of elements), it's good to initialize everything, even if you override it.

James
December 22, 2019 at 3:34 am · Reply

This was my solution;

```
1  /*
2      Program that takes 2 user input integers and outputs
3      the sum and difference of both numbers.
4
5      By James Vorenkamp
6      22/12/2019
7  */
8
9  #include<iostream> // For std::cout and std::cin.
10
11 int main()
12 {
13     int num1{ 0 }; // Uniform initialize variable num1 for storing the first user input.
14     int num2{ 0 }; // Uniform initialize variable num2 for storing the second user input.
15
16     std::cout << "Enter an integer: "; // Ask the user to enter an integer.
17     std::cin >> num1; // Get keyboard input from user and store in variable num1.
18
19     std::cout << "Enter another integer: "; // Ask the user to enter another integer.
20     std::cin >> num2; // Get keyboard input from user and store in variable num2.
21
22     std::cout << num1 << " + " << num2 << " is " << num1 + num2 << '\n'; // Evaluate and pr
23     std::cout << num1 << " - " << num2 << " is " << num1 - num2 << '\n'; // Evaluate and pr
24
25     return 0;
26 }
```

I chose to initialise my variables right at the top of main and was just wondering if there was any tangible difference between doing what I've done and initialising variables at the beginning of each user input as is done in the solution? Cheers.

nascardriver
December 22, 2019 at 4:17 am · Reply

It's fine here. In more complex functions, it's good to declare variables as close to their first use as possible to prevent unnecessary variable creations.

Brian
November 22, 2019 at 7:19 am · Reply

```
#include <iostream>

int main() {
    int x{}, y{};
    std::cout << "Enter an integer: ";
    std::cin >> x;
    std::cout << "Enter another integer: ";
    std::cin >> y;
    std::cout << x << " + " << y << " is " << x+y << '.\n';
    std::cout << x << " - " << y << " is " << x-y << '.\n';

    return 0;
}
```

I wrote this but in the x+y I get a random number such as 10117864 and with x-y I get -211786.

I got my x values and my y values to accept input and to show the correct values but when it does the addition or subtraction, I get very strange numbers.

nascardriver
November 22, 2019 at 7:41 am · Reply

'.\n' are 2 characters. Your compiler should've warned you about that. Make sure you enabled compiler warnings and read them. If you want to print more than 1 character, you need to use quotation marks ".\n"

Brian
November 22, 2019 at 8:55 am · Reply

Ah thank you. Can't believe I didn't see that.

Logan
November 16, 2019 at 3:03 pm · Reply

I made mine like this. I know I should have used comments. just wanted other opinions on it

```
#include <iostream>

int main()
{
    std::cout << "Please enter your first number here :";
    int x{ 0 };
    std::cin >> x;
    std::cout << "Please enter second number here :";
    int y{ 0 };
    std::cin >> y;
    std::cout << "Here are your two numbers added together :" << x + y;
}
```

nascardriver

November 17, 2019 at 1:58 am · Reply

Hey Logan,

if you program prints anything, the last thing it prints should be a line feed (\n) to prevent output of programs getting mixed.
Explicitly initializing `x` and `y` to `0` can be confusing, as the reader might think you're using the value for something. Empty curly braces are fine here.

amine
February 3, 2020 at 4:02 am · Reply

hey man what do you mean by initializing x and y can be confusing
as the reader might think you are using the value for something

nascardriver
February 4, 2020 at 8:20 am · Reply

If I see that a variable is explicitly initialized to a specific value, I assume that this value will be used for something, why else would it have that value.

Hector
October 16, 2019 at 2:00 pm · Reply

This is my solution.

```
// Ask the user for two integers                //
// then shows the addition and substrabction    //
// of both integers.                            //

#include <iostream>


int main() {

    std::cout << "Enter first integer: ";
    int a{};
    std::cin >> a;

    std::cout << "Enter second integer: ";
    int b{};
    std::cin >> b;

    std::cout << a << " + " << b << " is " << a + b << '\n';
    std::cout << a << " - " << b << " is " << a - b << '\n';

    return 0;
}
```

Thanks for share your knowledge with us! :)

nascardriver
October 17, 2019 at 1:01 am · Reply

Looks good :)

Wanderlust814
October 4, 2019 at 10:00 am · Reply

I feel my code is lacking the experimentation of more...advanced...beginners? I tried to get as close to lesson as possible. Still works great though.

```
#include<iostream>

int main()
{
    std::cout << "Enter an integer:  ";

    int y = { 0 };
    //define variable "y" as an integer variable
    std::cin >> y;
    // user input equals y
    std::cout << "Enter a 2nd integer:  ";

    int z = { 0 };
    std::cin >> z;
    // user input equals z

    std::cout << y << " + " << z << " is "
        << y + z << '\n';

    std::cout << y << " - " << z << " is "
        << y - z << '\n';

    return 0;
}
```

nascardriver
October 6, 2019 at 2:29 am · Reply

Initialize your variables with brace initializers. Brace initialization doesn't use `=`.

```
int y{ 0 };
// Or, if you don't use the initial value of @y
int y{}; // This is also 0
```

Looks good otherwise.

danik123
October 9, 2019 at 5:49 am · Reply

Just to add up to nascardrivers answerd. The '=' can be there and is not mistake, for compiler it should be the same. If it helps you to have more intuitive grasp on the initialization, you can use it.

Shubham Maske
September 21, 2019 at 7:08 am · Reply

Here is My answer :

```
 #include <iostream>
using namespace std;

```

```
4  int main(){
5      int x{0}, y{0};
6
7      cout<< "Enter a number :";
8      cin>> x;
9      cout<<"Enter another number :";
10     cin>> y;
11
12    cout<< "addition " << "is :" << x+ y << '\n';
13    cout <<"substraction" << "is :"<< x - y ;
14
15    return  0;
16
17 }
```

nascardriver
September 21, 2019 at 7:33 am · Reply

- Don't use `using namespace`.
- Declare one variable per line.
- Inconsistent formatting. Use your editor's auto-formatting feature.
- If your program prints anything, the last thing it prints should be a line feed ('\n').

Jake
October 2, 2019 at 1:26 pm · Reply

Hi nascardriver.

You admonished Shubham to not use "using namespace". I just came off a different beginner's "course" (Bucky's 9-hour YouTube video) and it uses CodeBlocks. CodeBlocks inserts the "using namespace" directive (pragma? I don't know what I'm talking about yet :-) One rationale for not "using" is that we don't know what in &!@%3$ a namespace is. On the other hand, taking it as an arbitrary artifact of the language, I see it as a minor typing saver: I don't have to use the std:: prefix for cout and cin. So please explain to us, we who have just enough knowledge to be dangerous, why I should not use a shortcut I already happen to know. (Will anyone confess to remembering PL/1 and PROC OPTIONS(MAIN) ? ;-)

nascardriver
October 2, 2019 at 11:43 pm · Reply

It can cause name conflicts, eg. you want to use a function but it resolves to a different function or multiple functions. This causes undesired behavior and a compilation error.
There's a lesson about this that goes into more detail. You'll see it along your journey.

4STR4L_3N3MY
September 11, 2019 at 9:35 am · Reply

I did it like this!
i know, i need to add comments.

#include <iostream>

int main()
{

```
    int inP1;
    int inP2;

    std::cout << "Enter your first #: ";
    std::cin >> inP1;

    std::cout << "\n" << "Enter your second #: ";
    std::cin >> inP2;

    std::cout << "\n" << inP1 << " + " << inP2 << " = " << (inP1 + inP2);
    std::cout << "\n" << inP1 << " - " << inP2 << " = " << (inP1 - inP2);
    std::cout << "\n" << inP1 << " * " << inP2 << " = " << (inP1 * inP2);
    std::cout << "\n" << inP1 << " / " << inP2 << " = " << (inP1 / inP2);
}
```

When dividing, how do i program the higher # to be divided by the lower # - i dunno how to do that yet

nascardriver
September 11, 2019 at 11:51 pm · Reply

Please use code tags when posting code (Yellow message below to comment box).
- Print the line feed at the end of lines. Printing them at the beginning is likely to mess up output.
- Use single quotation marks for characters ('\n' instead of "\n").

You can use `std::swap` to swap the values of 2 variables.

```
1 if (inP1 < inP2)
2 {
3   std::swap(inP1, inP2);
4 }
```

This will only work if the division is the last thing you do, as the variables stay swapped for all further operations.

Nico
July 29, 2019 at 3:36 am · Reply

I have done like this:

```
#include <iostream>

int main()
{
    std::cout << "Enter an Integer: ";

    int numOne { 0 };
    std::cin >> numOne;

    std::cout << "Enter another Integer: ";

    int numTwo { 0 };
    std::cin >> numTwo;

    std::cout << numOne << " + " << numTwo << " is: " << numOne + numTwo << '\n';
    std::cout << numOne << " - " << numTwo << " is: " << numOne - numTwo << '\n';

    return 0;
}
```

nascardriver
July 29, 2019 at 4:27 am · Reply

Hi Nico!

Looks good. You don't need the explicit initialization to 0, but you can keep it if it makes the code clearer for you.
Please use code tags when posting code.

senko
July 19, 2019 at 3:08 am · Reply

what is the purpose of writing:

```
int x{ 0 };
```

if you can write:

```
int x = 0;
```

Am I missing something important? I didn't even know you could write it like that.

nascardriver
July 19, 2019 at 4:14 am · Reply

It's part of lesson 1.4.

Jim
July 8, 2019 at 5:03 am · Reply

```
#include <iostream>

int main()
{
    std::cout << "Enter an integer: " << '\n';
    int numOne; //this is the first value entered
        std::cin >> numOne;
        std::cout << "Enter another integer: " << '\n';
        int numTwo; //this is the second integer entered
        std::cin>>numTwo;
        std::cout << "Integer one plus integer two is: " << numOne + numTwo << '\n';
        std::cout << "integer one times integer two is: " << numOne * numTwo;

        return 0;
]
```

This code is very different from what you posted and it gives the right answer. Is it still good?

nascardriver
July 8, 2019 at 5:08 am · Reply

Hi Jim!

It's very similar to Alex' solution. You used multiplication instead of subtraction, that's fine, experimenting speeds up the learning process.

- Inconsistent formatting. Use your editor's auto-formatting feature.
- If you program prints anything, the last character it prints should be a line feed ('\n').

prasanth guntupalli
June 4, 2019 at 6:57 pm · Reply

#include<iostream>
using namespace std;
int main()
{
cout<<"declare any number"<<endl;
cin>>a;
cout<<"declare another number"<<endl;
cin>>b;
cout<<"result after addition is"<<endl;
cout<<a+b<<endl;
return 0;
}

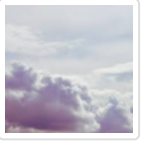
zim
May 24, 2019 at 10:45 am · Reply

Hi, here is my code is it ok to initializing variable first or do you need to do this before you use them ? I just find it much neater to have them all above the main program first. thanks

```
#include <iostream>

int main()
{
int addA{};
int addB{};

  std::cout << "enter a number 1 : ";
  std::cin >> addA;
  std::cout << "enter a number 2 : ";
  std::cin >> addB;
  std::cout << addA << " + " << addB << " = " <<  addA + addB << '\n';
  std::cout << addA << " - " << addB << " = " << addA - addB  << '\n';
  return 0;
}
```

nascardriver
May 25, 2019 at 12:43 am · Reply

You should declare variables as close to their first use as possible. It won't make a difference yet, but once you get to classes, this will improve memory usage.



Chayim
June 6, 2019 at 3:43 pm · Reply

Your code should be like this:

```
#include <iostream>
using namespace std;
```

```
int main()

{
int addA{};
int addB{};

  std::cout << "enter 2 numbers: ";
  std::cin >> addA;
  std::cin >> addB;
  std::cout << addA << " + " << addB << " = " <<  addA + addB << '\n';
  std::cout << addA << " - " << addB << " = " << addA - addB  << '\n';

  return 0;
}
```

Chayim
June 6, 2019 at 10:41 pm · Reply

Better like this:

```
#include <iostream>
using namespace std;

int main()

{
int addA{};
int addB{};

  std::cout << "enter 2 numbers: \n";
  std::cin >> addA;
  std::cin >> addB;
  std::cout << addA << " + " << addB << " = " <<  addA + addB << '\n';
  std::cout << addA << " - " << addB << " = " << addA - addB  << '\n';

  return 0;
}
```

Logan
May 1, 2019 at 6:14 am · Reply

Just one point, I did the question above and rather than declare the variable under two lines, I used one for num1 and num2; Is this okay? And if so, is there also a more concise way of doing ,"cin" in one line?

```
#include "stdafx.h"
#include <iostream>
int main()
{
    int num1,num2{ 0 };
    //int num1{};
    //int num2{};
    std::cout << " Enter a number: \n";
    std::cin >> num1;
    std::cout << "Enter a second number: \n";
    std::cin >> num2;
    std::cout << "The result of " << num1 << " + " << num2 << " = " << num1 + num2 << '\n';
    std::cout << "The result of " << num1 << " - " << num2 << " = " << num1 - num2 << '\n';
```

```
14
15      return 0;
16  }
```

The result below:
Enter a number:
4
Enter a second number:
8
The result of 4 + 8 = 12
The result of 4 - 8 = -4
Press any key to continue . . .

nascardriver
May 1, 2019 at 6:36 am · Reply

> I used one for num1 and num2; Is this okay?
No, this makes your code harder to read and can lead to mistakes. @num1 is uninitialized. The curly braces only apply to @num2.

> is there also a more concise way of doing ,"cin" in one line?
You'll learn about functions later. They can be used to wrap a call to @std::cin.operator>> so you can do this

```
1  int num1{ getIntFromUser() };
```

Logan
May 1, 2019 at 3:39 pm · Reply

Thank you, yeah, I ran it separate and got the uninitialized message. Slight moment of madness. I think my crazy thinking was it would default both to zero. I tested it by then adding its own curly braces with amount inside, least that then worked, but as you say it looks messy.

As for the cin, I am curious to what you mentioned up there, as opposed to using a Loop. To see if data can be placed into multiple variables without using a Loop. Just something I was wondering about. Thanks again!

Prateek
April 24, 2019 at 6:06 pm · Reply

I am trying to find area of a circle.

```
#include <iostream>
int main()
{
std::cout<<"Enter the radius of the circle : ";

int radius{};                // I just want user to enter an integral value of radius.

std::cin>>radius;
std::cout<<"Area of circle is "<< "π"<< "*"<< radius<<" *"<<radius<< "="<< π*radius*radius<<'\n';
return 0;
}
```

I am getting a compilation error.
What exactly is wrong in my program?

Nice
April 24, 2019 at 7:29 pm · Reply

Probably the fact that your compiler thinks π is an undefined identifier and not a number that can be multiplied.

Prateek
April 24, 2019 at 7:56 pm · Reply

I tried defining it but didn't get the issue resolved.

no Name
April 25, 2019 at 9:31 am · Reply

try this bro

```
#include <iostream>
#include <math.h>
int main()
{
std::cout<<"Enter the radius of the circle : ";
double radius{};
double pi {16 * atan2(1.0, 5.0) - 4 * atan2(1.0, 239.0)};
std::cin>>radius;
std::cout<<"Area of circle with radius " << radius<< ' ' << "is " << pi*pow(radius,2)<<'\n';
return 0;
}
```

no Name
April 25, 2019 at 9:37 am · Reply

wtf is this
std::cout<<"Area of circle is "<< "π"<< "*"<< radius<<" *"<<radius<< "="<<
π*radius*radius<<'\n';

try to make like this

```
#include <iostream>
int main()
{
std::cout<<"Enter the radius of the circle : ";

int radius{};                // I just want user to enter an integral value of radius.
double pi {3.1415926535897}; //pi value
std::cin>>radius;
std::cout<<"Area of circle with radius "<< radius << ' ' <<  "is: " << pi*radius*radius<<'\n';
return 0;
}
```

Prateek

April 25, 2019 at 8:31 pm · Reply

My problem was indeed silly,as I am just a beginner, and was trying to figure out a random simple question that came to my mind.
I appreciate your efforts.
Thank you

nascardriver
April 26, 2019 at 2:45 am · Reply

Source code should never contain non-ascii characters. Following this rule guarantees compatibility with all editors and prevents misconceptions like this.

Alex
April 28, 2019 at 3:50 pm · Reply

This program has several things wrong with it. Namely, π hasn't been defined. Generally, you shouldn't use non-ASCII characters for variable names. So you should rename this to pi, and then define pi with a value. Or replace it with the literal value 3.14159.

Bruhdeel
June 28, 2019 at 9:33 pm · Reply

#include <iostream>
int main()
{
   std::cout << "Enter the radius of the circle : ";

   int radius{};

   std::cin >> radius;
   std::cout << "Area of the circle is " << "pi " << " * " << radius << " * " << radius << " = " << 3.1415926535897 * radius * radius << '\n';
   return 0;
}
I changed it up a bit and this compiled with no errors. Don't use non-ascii characters.

Austin
April 17, 2019 at 5:50 pm · Reply

Hi thank you so much for the time and effort you put into these lessons! I've really enjoyed following along.

I have a question about defining a variable and assigning a variable. I noticed at question 3 you defined the variable, used std::cout, then assigned the variable through user input.

I was under the impression that it's better to assign a value to a variable maybe right after declaring it. Maybe this is inconsequential, but I was wondering what was considered "best practice". I think I remember it being said in the lessons that this structure is debated but that you recommended a certain way of doing it.

I'd be happy to clarify myself if necessary. Thank you!

Here is how I wrote it:

```
1 #include "pch.h"
```

```cpp
#include <iostream>

int main()
{
    std::cout << "Enter an integer: ";
    int num1; //first user input
    std::cin >> num1;

    std::cout << "Enter another integer: ";
    int num2; //second user input
    std::cin >> num2;

    std::cout << num1 << " + " << num2 << " is " << num1 + num2 << ".\n";
    std::cout << num1 << " - " << num2 << " is " << num1 - num2 << ".\n";

    return 0;

}
```

Edward
April 18, 2019 at 10:37 pm · Reply

WHY WE HAVE INCLUDED "pch.h"??

Alex
April 22, 2019 at 2:19 pm · Reply

pch.h is a header that is required by modern versions of Visual Studio if you have precompiled headers turned on.

Alex
April 22, 2019 at 10:00 am · Reply

> I was under the impression that it's better to assign a value to a variable maybe right after declaring it

Yes. If you look closely at the code, you'll see this:

```cpp
int x{};
```

This is the equivalent of:

```cpp
int x{ 0 };
```

So we are initializing the variable with a value.

Austin
April 24, 2019 at 7:57 am · Reply

Thank you so much for the response and that makes perfect sense I totally missed that!

ptitBarbu
April 14, 2019 at 2:28 am · Reply

Great course Alex, very well explained !
Thank you for that.

« Older Comments 1 ... 6 7 8