

Forum:[Hack Forums](#)

Thread: [Read before posting "How to learn C/C++" \[Ask Questions\]](#)

caiveman - December 9, 2016, 6:03 am

Experienced programmers, please provide constructive criticism

My Skype: caiveman.hf@outlook.com
Add for any C programming help

Hello All,

I see many posts especially on this section of HF that go something like "Learning C" or "How to learn C++". Hopefully this post will answer some of your questions that I had when I was in your position and end this spam.

Before we start, I want to make sure nobody has the - "I want to learn C by making my own botnet/crypter" mentality. You are guaranteed to give up at the first difficulty if you have this mindset.

Introduction/Background Information

Firstly, an introduction to C. C was developed by AT&T makes up most of UNIX (and Linus' open source copy, Linux). It therefore is a native language to Windows. Many programmers prefer C because of its speed, portability, and the ability to interface with low-level aspects of the operating system, despite it's steep learning curve.

C is not the easiest language to learn as your first. If you have never programmed before, I would highly recommend starting with Python as it is incredibly powerful yet has a simple syntax, and is very hard to fuck up your PC.

Prerequisites

To learn C, it is essential to know basic computer science, aka the stack, bytes, endian-ness, pointers, memory, etc because of the low-level nature of the language. My recommendation for this is to watch this series of lectures by UNSW which covers the basics of computer science. The ~20 videos are available on youtube to watch for free.

[Video: <https://youtu.be/jR10gWCx3WQ>]

It is absolutely crucial to know this stuff to program in C, especially the concept of pointers and addresses.

Tools neededText editor (Highly recommend [Sublime Text 3](#))

gcc (or [MinGW](#) on Windows)

A brain

Do these things before reading on

Google (in order) to familiarise yourself:"C Hello World"

"C argc and argv meaning"

"Pointer"

"C iterators"

"C conditionals"

"C main return values"

Read <https://msdn.microsoft.com/en-us/library/s3f49ktz.aspx>

[My favourite "Introduction to C Programming"](#)

Actual Programming

Okay, now to the easiest part of learning C - the syntax. A common misconception that beginners hold when learning a new language is that the most important and hardest aspect is the syntax, or actually writing the code. Any good programmer should be able to learn the syntax to a new language in a couple of days, but it can take months to develop an efficient and effective programming style.

I will put "quiz" questions along the way so you can make sure you know the stuff before moving on.

Deconstructing a C program

Lets have a look at a typical C program

```
Code #include <stdlib>
#include <stdio>
```

```
int main(int argc, char* argv[]){
printf("Hello world!\n");
return 0;
}
```

Now, lets go line by line and analyse the code

Variables

Variables store data in memory. In C, you do it like this.

```
Code int year = 2016;
```

```
DWORD dwflashDuration;
```

```
unsigned char = 'f';
```

A constant variable is just a variable that cannot be changed.

```
Code const int encryption_key = 5497512145; OR
```

```
int const encryption_key = 5497512145;
```

the "const" can come before or after data type declaration, pick one and stick to it.

Protip: an "unsigned" variable basically means it can only be positive, doubling the "range" of the variable.

Quiz: How many bytes does a signed int hold?

Global variables vs Local variables

Global variables are variables that are declared outside of any function.

aka:

```
Code #include <stdlib>
```

```
#include <stdio>
```

```
char* name = "caiveman-senpai" <- Global variable, can be used anywhere in the file.
```

```
int main(int argc, char* argv[]){
```

```
    int port = 1337; <- Local variable, can only be used inside the function
```

```
    return 0;
```

```
}
```

Protip: Try to avoid using global variables, if not at all. 99% of the time there is a better way of doing it (especially in c++)

Loops and conditionals

These are the two core principles found in any programming language.

There are a number of loops that you can use in C, the two most common ones being

Code while (condition)and

Code for (variable; condition; iterator)

counting to 100 using each loop

```
Code #include <stdio>
```

```
#include <stdlib>
```

```
int main(int argc, char*argv[]){
```

```
    int i = 0;
```

```
    while (i <= 100){
```

```
        printf("%d\n", i);
```

```
        i++;
```

```
    }
```

```
    for (int i =0; i <= 100; i++){
```

```
        printf("%d\n", i);
```

```
    }
```

```
    return 0;
```

```
}
```

There are more loops, such as the do - while loop, but you don't really need to know these, and it's more "syntactic sugar" to save the programmer a few lines of code.

Input and Output

Input

There are couple of input methods in C, and it is a little bit contentious on which one you should use. Some people use `fgets()` or `getline()` but for the purpose of this tutorial I will use `scanf`.

Example:

Code `#include <stdlib>`

`#include <stdio>` <<<<< Standard input and output library

```
int main(int argc, char* argv[]){
    int sum;
    printf("Enter an int\n");
    scanf("%d", &sum); << Get an input into the address of sum
    printf("%d\n", sum);
    return 0;
}
```

Quiz: Think of how you could get multiple characters of input using loops and `scanf()`...

Output

Use `printf` for output in C

Code `printf("%d\n", integer);`

`printf("%s\n", string);`

`printf("%d and %.6f", integer, float to 6 places);` View the rest of the formatting options [here](#)

Strings

Strings are a tricky one in C, because unlike other languages where they are stored as an object, strings are literally just arrays of characters.

Two ways of making a string

Code `char* str = "this string can be as long as u like, but you can't change it once defined (easily)";`

`char str2[10] = "10 chars";`

All strings end with the null character `'\0'` so the pc knows when to stop reading. There are obvious advantages and disadvantages to both.

In C++, it is much better because there is actually an inbuilt function in the `<string>` header

`std::string str("sick string with lots of functions");`

[Further reading on C++ strings](#)

Moving onto C++

C++ retains 99% of the syntax from C, but adds a crucial new concept. Classes and proper Object Oriented Programming.

I don't have time to explain in OOP detail but it is essentially programming Objects that interact with each other.

Read this -> <https://www.codeproject.com/Tips/679615/OOP-Explained-in-Seconds>

Watch this -> <https://www.youtube.com/watch?v=lbXsrHGhBAU>

This keeps the code nice and organised so other people can understand what you are doing. It's really awesome once you get the hang of it.

Conclusion and further reading

Now if you actually did all those steps and learnt some basic computer science you should have a shallow understanding of the amazing language that is C.

The most helpful piece of advice I can give you is to read lots of good C code, and try to understand it, and replicate it.

Further reading and googling: malloc and it's uses

c allocate and free memory

c++ classes

c++ class inheritance

c multithreading

c linked list

Projects to complete from here: Write a calculator that takes user input

Make a linked list program using malloc

Make a double-linked list using malloc and pointers

Make a minesweeper hack

Write a port scanner in C using winsock

Write a hard hitting self rep decentralized mirai killer botnet with web api

~caiveman

Killer Kamal - December 9, 2016, 6:08 am

Nice HQ thread! Definitely worth reading.

caiveman - December 9, 2016, 6:09 am

This post is reserved for further updating

g4143 - December 9, 2016, 6:28 am

You missed one of the crucial things a rookie programmer should be encouraged to use...constants. It won't be apparent to the rookie why constants are important but they should get into the habit of using them wherever they can. Some say it's better 'for the rookie' to declare or define everything as constants or literals and then change them to immutable variables as needed.

caiveman - December 9, 2016, 6:37 am

(Dec 9, 2016 07:28 AM)g4143 Wrote: [\[CITE\]](#) You missed one of the crucial things a rookie programmer should be encouraged to use...constants. It won't be apparent to the rookie why constants are important but they should get into the habit of using them wherever they can. Some say it's better 'for the rookie' to declare or define everything as constants or literals and then change them to immutable variables as needed.

dude i'm still writing it, and I wouldn't consider constants to be absolutely crucial.

g4143 - December 9, 2016, 6:53 am

(Dec 9, 2016 07:37 AM)caiveman Wrote: [\[CITE\]](#) dude i'm still writing it, and I wouldn't consider constants to be absolutely crucial.

Constants and literals are two of the most underappreciated methods for producing maintainable, sane code.

caiveman - December 9, 2016, 7:34 am

(Dec 9, 2016 07:53 AM)g4143 Wrote: [\[CITE\]](#) Constants and literals are two of the most underappreciated methods for producing maintainable, sane code.

Okay - can you please give me an example?

g4143 - December 9, 2016, 7:55 am

(Dec 9, 2016 08:34 AM)caiveman Wrote: [\[CITE\]](#) Okay - can you please give me an example?

An example of what? Syntax or a case where constants/literals are preferred?

Dostoevsky - December 9, 2016, 3:56 pm

(Dec 9, 2016 08:34 AM)caiveman Wrote: [\[CITE\]](#) Okay - can you please give me an example?

Well, Google have found constants important enough to add it to their coding style guides. When using local constants, you prefix the variable name with "k".

Constants are incredibly useful for showing intention and imposing it (despite the fact you can cast away constness).

Some people also fail to realise where STL things are made constant and then write inefficient code.

E.g. if you tried to create a map declared like:

PHP Code

```
<?php
```

```
std::map<std::string, int> m;
```

and tried to iterate it with:

PHP Code

```
<?php
```

```
for(const std::pair<std::string, int>& i : m)
```

```
{
```

```
    // ...
```

```
}
```

you'd make a copy each time since novices forget that std::map will actually store the pairs as std::pair<const Key, T> where Key is a const type.

Small things like this can demonstrate how understanding the importance of const can stop you

from writing bad code.

The example I described above was an example by Scott Myers regarding how using `auto` for types is a good way to get the desired type rather than assuming the type. Essentially: use `const auto&` if you don't want to make a copy when iterating maps.

Const correctness is vital to writing good C++ code.

caiveman - December 9, 2016, 4:23 pm

(Dec 9, 2016 04:56 PM)Dostoevsky Wrote: [\[CITE\]](#) E.g. if you tried to create a map declared like:

PHP Code

```
<?php
```

```
std::map<std::string, int> m;
```

and tried to iterate it with:

PHP Code

```
<?php
```

```
for(const std::pair<std::string, int>& i : m)
```

```
{
```

```
    // ...
```

```
}
```

Thanks for this example. I'll add something similar to the tutorial.

Infinity. - December 10, 2016, 9:11 pm

<https://www.youtube.com/watch?v=Ps8jOj7diA0&list=PLD28639E2FFC4B86A>

An interesting series for C programming.

caiveman - December 11, 2016, 12:08 am

(Dec 10, 2016 10:11 PM)Infinity. Wrote: [\[CITE\]](#)

<https://www.youtube.com/watch?v=Ps8jOj7diA0&list=PLD28639E2FFC4B86A>

An interesting series for C programming.

Wow, just watched the first lecture and definitely looks interesting.

Thanks!

dendro - December 11, 2016, 5:31 am

(Dec 10, 2016 10:11 PM)Infinity. Wrote: [\[CITE\]](#)

<https://www.youtube.com/watch?v=Ps8jOj7diA0&list=PLD28639E2FFC4B86A>

An interesting series for C programming.

Those are some quality videos, thanks for sharing.

caiveman - January 3, 2017, 8:42 pm

Bump because people can't use the search bar

Sajuuk - January 4, 2017, 12:36 am

Thread is useless to most because one of the prerequisites is a brain.

0xDEAD10CC - January 4, 2017, 6:07 pm

Typical C programs?

```
Code #include <stdlib>
```

```
#include <stdio>
```

^ All of the examples you have fail because they don't have `".h"`

caiveman - January 4, 2017, 10:57 pm

(Jan 4, 2017 07:07 PM)0xDEAD10CC Wrote: [\[CITE\]](#) Typical C programs?

```
Code #include <stdlib>
```

```
#include <stdio>
```

^ All of the examples you have fail because they don't have ".h"

Fuck off

0xDEAD10CC - January 6, 2017, 4:39 pm

(Jan 4, 2017 11:57 PM)caiveman Wrote: [\[CITE\]](#) Fuck off

^

Quote:Code unsigned char = 'f';

^ You're the one making easily avoidable mistakes and recommending horrible MinGw.

<http://www.mingw.org/wiki/c99>

Quote:Because MinGW relies on MSVCRT, it has many of the same limitations and quirks with compatibility as Visual Studio 6. You should assume that MinGW applications cannot rely on C99 behaviour, only on C89. For example, the newer format characters in printf like %a and %ll are not supported, although there exists a workaround for %ll.

caiveman - January 6, 2017, 4:49 pm

(Jan 6, 2017 05:39 PM)0xDEAD10CC Wrote: [\[CITE\]](#) ^

^ You're the one making easily avoidable mistakes and recommending horrible MinGw.

<http://www.mingw.org/wiki/c99>

The point is you need to learn how to compile from the command line

0xDEAD10CC - January 6, 2017, 5:04 pm

(Jan 6, 2017 05:49 PM)caiveman Wrote: [\[CITE\]](#) The point is you need to learn how to compile from the command line

That has nothing to do with lack of MinGw modern ISO C compiler support.. :/ "-std=c(99/11)" still isn't going to get you all the way in a few cases where support for C features is bugged. Use clang, because using cl would practically be the same as using MinGw for the reasons outlined on that page.

RDCA - January 6, 2017, 6:37 pm

(Jan 6, 2017 05:49 PM)caiveman Wrote: [\[CITE\]](#) The point is you need to learn how to compile from the command line

Hypothetically you don't need to learn how to compile from the command line. I literally do not remember the last time I did.

caiveman - January 6, 2017, 7:05 pm

(Jan 6, 2017 07:37 PM)RDCA Wrote: [\[CITE\]](#) Hypothetically you don't need to learn how to compile from the command line. I literally do not remember the last time I did.

The word hypothetically makes hypothetically no sense in this context.