Becoming Bilingual

Hey everyone! I have recently started learning to program in the C language and expand my technical versatility by being able to learn more programming languages and being able to utilize them as needed! So this begs the question, what are the pros and cons of programming in C?

To start with, C is classified as a mid-level programming language, meaning it is not optimal to learn the basics and fundamentals of programming at the start of someone’s programming journey and is more suited for people that already know a language (or at least understand how it works). This is mainly due to a more simplistic and powerful syntax. To a seasoned programmer’s eyes, the C language is very easy to understand and interpret. As you will see, some of the syntax used in Java is made redundant and simpler in the C language. C also is an excellent building block in learning many other languages and is considered the most fundamental language to learn, since other languages like Python, C++, and Java are literally built off it.

Because C is so efficient and powerful, the runtimes and performance of C are efficient to say the least. This is due to the large amounts of data types and operators available within C, as well as the large number of built-in functions. The functions provided in C are a very useful, but not as useful as the user-defined functions, which allow users to create functions tailored to their specific needs. Since functions can be defined and reused by users, much like Java, C is able to extend itself and grow as a language. These are just a few of the advantages to using C, but what are the disadvantages?

While C is very powerful and simplistic, one of the drawbacks is a lack of Object-Oriented essentials. Fantastic tools like Inheritance, Polymorphism, Encapsulation, and Abstraction are all missing in C. Code is very hard to reuse and share among other classes. As many of you are aware from our Java classes, multiple instances of a class is a beautiful thing, so you will have to stay away from C to enjoy that. C also suffers from inefficient memory management and a lack of garbage collection. C likes to give much control and power to the user, but sometimes having things autonomously is a gift to cherish since the users are most likely trying to create products that do things autonomously. This is directly related to the fact that C does not clean up unused objects and code, instead leaving that to the user to do manually. With inefficient memory management and manual garbage collection, memory space issues are just along the horizon. The last downside that caught my eye, is a lack of exception handling. The ability to tell Java “if a file is missing, maybe skip that and run this instead” is non-existent in C. Users will have to manually catch these errors and rectifying them without the ease of being able to call a system functionality to do it for them.

Here is a simple calculator written in C:

int main() {

char op;

double first, second;

printf("Enter an operator (+, -, \*, /): ");

scanf("%c", &op);

printf("Enter two operands: ");

scanf("%lf %lf", &first, &second);

switch (op) {

case '+':

printf("%.1lf + %.1lf = %.1lf", first, second, first + second);

break;

case '-':

printf("%.1lf - %.1lf = %.1lf", first, second, first - second);

break;

case '\*':

printf("%.1lf \* %.1lf = %.1lf", first, second, first \* second);

break;

case '/':

printf("%.1lf / %.1lf = %.1lf", first, second, first / second);

break;

// operator doesn't match any case constant

default:

printf("Error! operator is not correct");

}

return 0;

}

The simplicity and interpretation of the code should be abundantly clear, and hopeful illustrate just how powerful C and be at first glance. Even without much learning or practice with C, it looks very easy to use!

It is amazing learning a new programming language after already knowing one! The similarities and fundamentals remain the same among both languages, but C seems to be a bit more intuitive about what the coder is trying to do. I have watched some videos covering the fundamentals and basics of coding in the C programming language (including one 4 hour-long video, yikes) and I found myself understanding most of the subjects just by the title, and knowing what the creator is going to say before they say it. Unlike java however, C seems to have more intuitive objects or functions that don’t need to be created or instantiated before they can be used. With the decrease in redundant code, learning the new syntax is still challenging. Simple things like declaring a string are slightly different in the C language. For instance, a string is actually declared as a “char”. Simple things like that can be hard to remember. I am curious to discover if generics and method calls work the same way in the C language, since they are the things, I enjoyed most working in Java, but every language has to be different! I really enjoy the more straightforward approach of C programming. The no-nonsense, easy to read style of the C language is a lot easier on the eyes and takes a lot less time to interpret the function of the code. However, some things are kind of odd. As we are familiar with in Java, “objects” are called “structs” in the C language and ignore camelCase for a more proper noun structure in naming. “Struct” is short for “data structure” but essentially mimics a real-world object, you know, like an object.

The main theme of the C language is pointers. Pointers allow you to store the physical memory location of data into their own variables. While pointers is a highly nuanced topic in the C language, it is simply just another data type, physical memory address. While I have yet to learn how pointers are used productively, they can seem a bit overwhelming and confusing to use, but nonetheless remain one of the key components to programming in the C language. For the time being, all the thoughts I have on pointers make them seem not-so-intuitive and outdated, but I am more than certain I will be wrong.

I am not looking forward to learning how to use them nor to having to code without the use of inheritance or multiple instances of objects, as that alone seems daunting and unintuitive. The loss of using code over and over to instantiate multiple objects at once to complete multiple tasks seems like a huge tool to kick out of your toolbelt, but hey, C has to make up for it somehow right?

I am looking forward to transferring my code from my third semester Java course into the C language and comparing the similarities and differences side by side. The project I am thinking of translating to the C language is the XML Parser I developed for the data structures assignment, which used generics, custom structures, and rigorous error checking for making sure the inputted xml file is in the correct format and all opening tags are closed with no overlapping issues. I assume translating it will be very easy since the differences in syntax and structure will not be a lot to overcome, and it also never hurts to already have the correct code next to you to copy from! I might try and develop a custom data structure for this project but given that I have not done that even in Java, it might be a bit more of a challenge (and good practice) to give it the stressful time required to figure it out. If I discover this project is too easy, I may very well go hunt down a new project that helps me test my knowledge and what I have learned so far in C programming. I really do aim to be proficient in programming in C, as having multiple languages that I am able to code in would be a huge bonus to my skill set, along with having the experience of learning an entirely new language altogether.

<https://unstop.com/blog/advantages-and-disadvantages-of-c-programming-language>

https://www.includehelp.com/c/advantages-and-disadvantages-of-c-programming-language.aspx