I'm not a robot

C LanguageC is a procedural programming language. It was developed by Dennis Ritchie as a system programming language to write an operating system. The key features of the C language include low-level access to memory, a simple set of keywords, and a clean style. These features make the C language suitable for system programs like operating systems or compiler development. C is a general-purpose, high-level, compiler-based, machine-independent structure language that is extensively used in various applications.Now lets talk about which kind of programs or applications we can design with the C language.> operating system [e.g.: dos, win, Unix.]> Editor [e.g.: notepad, WordPad]> Commercial applications> Database> Translator> Device driverAnd many moreIs it worth learning the C programming language in 2030???absolutely Yes Remember one thing every programming language has borrowed syntax/features directly or indirectly from the C language. Only the C language can clear the programming logic you need for your whole life. If you learn the C language you can easily switch to another language and that will take truly little time to master that language.Basically, C is a modular/procedural language. When you have the concept of C structure you can easily relate the topic to the class that is involved in object-oriented programming.Lastly, one more thing that is to clear your thoughts about whether is it worth it or not?If you want to study computer science then first start with C programming otherwise if you just love coding but are not in the CS subject then you can go for a project base programming language like python or something else whatever you love to do, that will be helpful because remember that the basic of Computer Science student is to learn how to solve problems. Here project is not the priority.I hope now you have no doubt about this topic.Now, how can you start with the C language? You can follow the below sequence that I follow to learn. But before starting your very first programming language, make sure you have some knowledge about the number system like converting numbers from one to another. that will help you to understand some programming topics more.Note: One thing that will help you to be a good programmer is to keep a notebook and pen with you when you learn about a new topic and write down the syntaxBasics:C Language IntroductionC Programming Language Standardint (1 sign bit + 31 data bits) keyword in CIs it fine to write void main() or main() in C/C++?Difference between int main() and int main(void) in C/C++?Macros and Preprocessors in CCompiling a C program: Behind the ScenesBenefits of C over other languagesProgram error signalsEscape sequences in CLine Slicing in CC/ C++ TokensVariables and Keywords in CHow are variables scoped in C Static or Dynamic?Scope rules in CHow Linkers Resolve Multiple Defined Global Symbols?Quiz on Variable Declaration and ScopeComplicated declarations in CRedeclaration of a global variable in CInternal and External LinkageDeclare a variable as constantWhy variables dont start with numbersRedeclaration of global variablesInitialization of global and static variablesFunctions in CImportance of function prototype in CFunctions that are executed before and after main() in Creturn statement vs exit() in main()How to Count Variable Numbers of Arguments in C?,What is the evaluation order of function parameters in C?Does C support function overloading?How can we return multiple values from a function?What is the purpose of a function prototype?Static functions in Cexit(), abort() and assert()Implicit return type int in CWhat happens when a function is called before its declaration in C?_Noreturn function specifier in Cexit() vs _Exit()__func__ identifier in CCallback function in CNested functions in CParameter Passing Techniquespow() function in Ctolower() function in Ctime() function in CData Types in CUse of bool in CInteger Promotions in CQuiz on Data Types in CComparison of a float with a value in CNeed for a long data type in Csize_t data type in CInteresting facts about data types and modifiersFloat and Double in CCharacter arithmetic in CTypecasting in CIntroduction to pointers in C and C++Double Pointer (Pointer to Pointer) in CWhy does C treat array parameters as pointers?The output of the program | Dereference, Reference, Dereference, ReferenceDangling, Void, Null, and Wild PointersAn Uncommon representation of the array of elementsHow to declare a pointer to a function?Pointer vs Array in Cvoid pointer in CNULL pointer in C!Function Pointer in Cnear, far and huge pointersGeneric Linked List in Crestrict keyword in Cconst char *p, char * const p, and const char * const pPointer to an ArrayStorage Classes in CStatic Variables in CUnderstanding the extern keyword in CWhat are the default values of static variables in C?Understanding the volatile qualifier in CConst Qualifier in CInitialization of static variables in CUnderstanding the register keyword in CQuiz on Storage Classes in CThe understanding of volatile qualifiers in C- Set 1Returned values of printf() and scanf()What is return type of getchar(), fgetc() and getc()?Scansets in Cputs() vs printf() for printing a stringWhat is use of %n in printf()?How to print % using printf()?Quiz on Input-Output in CDifference between printf, sprintf and fprintf?Difference between getc(), getchar(), getch() and getche()Difference between %d and %i format specifierUse of fflush(stdin) in CClearing input buffer in C/C++scanf() and fscanf() in Cgetchar_unlocked() in CProblem with scanf()Differentiate printable and control characterrand() and srand() in C/C++Introduction to operators in C and Arithmetic OperatorsRelational and Logical Operators in CBitwise Operators in COperator Precedence and Associativity in CThe evaluation order of operandsComma in C and C++sizeof operator in COperands for sizeof operatorA comma operator questionResult of comma operator as l-value in C and C++Order of operands for logical operatorsIncrement (Decrement) operators require L-value ExpressionPrecedence of postfix ++ and prefix ++ in C/C++Modulus on Negative NumbersC/C++ Ternary Operator Some Interesting ObservationsPre-increment (or pre-decrement) in C++Difference between ++*p, *p++ and *++pResults of comparison operations in C and C++To find the sum of two numbers without using any operatorSequence Points in CExecution of printf with ++ operatorsAnything written in sizeof() is never executedDifference between strlen() and sizeof()Stringizing and Token-pasting operatorWrite a C macro PRINT(x) which prints xVariable-length arguments for MacrosMultiline macros in CCRASH() macro interpretationThe offsetof() macro,Branch prediction macros in GCCWhat is the difference between #define and const in C?A-C Programming Language PuzzleWhats the difference between header files stdio.h and stdlib.h ?How to print a variable name in C?Constants in CHow a Preprocessor worksPreprocessor Directives: Behind the scenes-1Preprocessor Directives: Behind the scenes-2isgraph() library functionwrite your own header file in Cdifftime() library functiontmpnam() library function_Generic keyword in Cmath.h library functionstypedef versus #define in Cstrftime() library functionexec family of functionsIntroduction to Arrays in C LanguageStrings in C LanguageProperties of the array in C LanguageDo not use sizeof for array parametersInitialization of variables sized arrays in CAre array members deeply copied?What is the difference between single-quoted and double-quoted declaration of char array?Initialization of multidimensional arrays in C/C++Write one-line functions for strcat() and strcmp()Whats the difference between char s[] and char *s in Cgets() is risky to use!C function to Swap strings,Storage for Strings in CDifference between array and pointerHow to dynamically allocate a 2D array in C?How to pass a 2D array as a parameter in C?How to write long strings in Multi-lines C/C++?What are the data types for which it is not possible to create an array?Variable Length Arrays in C and C++Shorthand array notationAccessing an array out of boundsstrcpy() in C/C++strcmp() in C/C++strdup() and strdndup() in C/C++Pass an array by valueReverse a string in C/C++strpbrk() in Cstrcoll() in C/C++ispunct() in Cstrspn() in Cisalpha() and isdigit() in C/C++What should be the data type of case labels of switch statement in C?For Versus WhileA nested loop puzzlesswitch statement in CDifference between while(1) and while(0)goto statementContinuing StatementBreak StatementUsing range in switch caseEnum in CStructures in CUnion in CStruct HackStructure Member Alignment, Padding, and Data PackingOperations on struct variables in CBit Fields in CStructure Sorting (By Multiple Rules) in C++Flexible array of members in the structureDifference between Structure and UnionDifference between C structures and C++ structuresAnonymous Union and Structure in CCompound Literals in CMemory Layout of C Programs,How to deallocate memory without using free() in C?calloc() versus malloc()How does free() know the size of memory to be deallocated?Use of realloc()What is Memory Leak? How can we avoid it?fseek() vs rewind() in CEOF, getc() and feof() in Cfopen() for an existing file in write modeRead/Write structure to a filefgets() and gets() in CBasics of File Handlingfsetpos() in Crename function in C/C++tmpfile() function in Cfgetc() and fputc() in Cfseek() in C/C++ftell() in Clseek() in C/C++remove function in C/C++Merge contents of two files into a third filePrint contents of file in CC Program to print numbers from 1 to N without using semicolon?How to find the sum of two numbers without using any operatorHow will you show the memory representation of C variables?Condition To Print HelloWordChange/add only one character and print * exactly 20 timesHow can we sum the digits of a given number in a single statement?What is the best way in C to convert a number to a string?Calculate Logn in one linePrint Even or Odd without using a Conditional statementHow will you print numbers from 1 to 100 without using a loop?How can we sum the digits of a given number in a single statement?How will you print Geeks for Geeks without using a semicolon?Write a one line C function to round floating point numbersHow will implement Your Own sizeofHow to count set bits in a floating-point number in C?How to change the output of printf() in main() ?How to find the length of a string without string.h and loop in C?Implement your own itoa()Write a C program that does not terminate when Ctrl+C is pressedHow to measure the time taken by a function in C?Print a long int in C using putchar() onlyConvert a floating-point number to string in CHow to write a running C code without main()?Write your own memcpy()C program to print characters without using format specifiersC program to print a string without any quote (singe or double) in the programExecute both if and else statements simultaneouslyPrint Hello World without using any header fileQuine A self-reproducing programComplicated declarations in CUse of bool in CSequence Points in COptimization Techniques (swapping),ASCII NUL, ASCII 0 (0?) and Numeric literal 0Little- and Big-Endian Mystery,Comparator function of qsort() in CProgram to validate an IP addressMultithreading in CAssertions in C/C++fork() in CInteresting Facts in C ProgrammingPrecision of floating-point numbers in C++ (floor(), ceil(), trunc(), round() and setprecision())setjump() and longjump() in Cnextafter() and nexttoward() in C/C++pthread_cancel() in Cpthread_equal() in Cpthread_self() in CLocal Labels in Clvalue and rvalue in CGet and set the stack size of the thread attributeDifference between fork() and exec()Errors in C/C++Why is C considered faster than other languages?Incompatibilities between C and C++Convert C/C++ code to assembly languageError Handling in CExecuting main() in C/C++ : Behind the sceneHygienic Macros in CCommand-line arguments in C/C++scanf(), fscanf(), sscanf(), scanf_s(), fscanf_s(), sscanf_s()Some Interesting Facts in C ProgrammingDatabase Connectivity using C/C++Function Interposition in CMacros vs FunctionsWrite your own memcpy() and memmove() Java code The C programming language has been alive and kicking since 1972, and it still reigns as one of the essential building blocks of our software-studded world. But what about the dozens of newer languages that have emerged over the last few decades? Some were explicitly designed to challenge Cs dominance, while others chip away at it as a byproduct of their own popularity. Its hard to beat C for performance, bare-metal compatibility, and ubiquity. Still, its worth seeing how it stacks up against some of the big-name language competition. C is frequently compared to C++, the language thatas the name suggestswas created as an extension of C. The differences between C++ and C could be characterized as extensive, or as excessive, depending on whom you ask. While still being C-like in its syntax and approach, C++ provides many genuinely useful features that arent available natively in C: namespaces, templates, exceptions, automatic memory management, and so on. Projects that demand top-tier performancelike databases and machine learning systemsare frequently written in C++, using those features to wring every drop of performance out of the system. Further, C++ continues to expand far more aggressively than C. C++ 23 brings even more to the table including modules, coroutines, and a modularized standard library for faster compilation. By contrast, the latest version of the C standard, C23, adds little and focuses on retaining backward compatibility. The thing is, all of the pluses in C++ can also work as minuses. Big ones. The more C++ features you use, the more complexity you introduce and the more difficult it becomes to tame the results. Developers who confine themselves to a subset of C++ can avoid many of its worst pitfalls. But some shops want to guard against that complexity altogether. The Linux kernel development team, for instance, eschews C++, and while the team is eyeing Rust as a language for future kernel additions, the majority of Linux will still be written in C. Picking C over C++ is a way for developers and those who maintain their code to embrace enforced minimalism and avoid tangling with the excesses of C++. Of course, C++ has a rich set of high-level features for good reason. But if minimalism is a better fit for current and future projectsand project teamsthen C makes more sense. After decades, Java remains a staple of enterprise software developmentand a staple of development generally, with many improvements along the way. Java syntax borrows heavily from C and C++. Unlike C, though, Java doesnt by default compile to native code. Instead, Javas JIT (just-in-time) compiler compiles Java code to run in the target environment. The JIT engine optimizes routines at runtime based on program behavior, allowing for many classes of optimization that arent possible with ahead-of-time compiled C. Under the right circumstances, JIT-compiled Java code can approach or even exceed the performance of C. And, while the Java runtime automates memory management, its possible to work around that. For example, Apache Spark optimizes in-memory processing in part by using unsafe parts of the Java runtime to directly allocate and manage memory and avoid the overhead of the JVMs garbage collection system. Javas write once, run anywhere philosophy also makes it possible for Java programs to run with relatively little tweaking for a target architecture. By contrast, although C has been ported to a great many architectures, any given C program may still require customization to run properly on, say, Windows versus Linux. This combination of portability and strong performance, along with a massive ecosystem of software libraries and frameworks, makes Java a go-to language and runtime for building enterprise applications. Where it falls short of C is an area where the language was never meant to compete: running close to the metal, or working directly with hardware. C code is compiled into machine code, which is executed by the process directly. Java is compiled into bytecode, which is intermediate code that the JVM interpreter then converts to machine code. Further, although Javas automatic memory management is a blessing in most circumstances, C is better suited for programs that must make optimal use of limited memory resources, because of its small initial footprint. Nearly two decades after their introduction, C# and .NET remain major parts of the enterprise software world. Its been said that C# and .NET were Microsofts response to Javaa managed code compiler system and universal runtimeand so many comparisons between C and Java also hold up for C and C#/.NET. Like Java (and to some extent Python), .NET offers portability across a variety of platforms and a vast ecosystem of integrated software. These are no small advantages given how much enterprise-oriented development takes place in the .NET world. When you develop a program in C#, or any other .NET language, you can draw on a universe of tools and libraries written for the .NET runtime. Another Java-like .NET advantage is JIT optimization. C# and .NET programs can be compiled ahead of time as per C, but theyre mainly just-in-time compiled by the .NET runtime and optimized with runtime information. JIT compilation allows all sorts of in-place optimizations for a running .NET program that cant be done in C. Like C (and Java, to a degree), C# and .NET provide various mechanisms for accessing memory directly. Heap, stack, and unmanaged system memory are all accessible via .NET APIs and objects. And, developers can use the unsafe mode in .NET to achieve even greater performance. None of this comes for free, though. Managed objects and unsafe objects cannot be arbitrarily exchanged, and marshaling between them incurs a performance cost. Therefore, maximizing the performance of .NET applications means keeping movement between managed and unmanaged objects to a minimum. When you cant afford to pay the penalty for managed versus unmanaged memory, or when the .NET runtime is a poor choice for the target environment (e.g., kernel space) or may not be available at all, then C is what you need. And unlike C# and .NET, C unlocks direct memory access by default. Readable code was one of Gos guiding design goals: Make it easy for developers to get up to speed with any Go project and become proficient with the codebase in short order. C codebases can be hard to grok, as they are prone to turning into a rats nest of macros and #ifdefs specific to both a project and a given team. Gos syntax, and its built-in code formatting and project management tools, are meant to keep those kinds of institutional problems at bay. Go syntax owes much to Ccurly braces as delimiters and statements terminated with semicolons are just two examples. Developers proficient in C can typically leap right into Go without much difficulty, even taking into account Go-specific features. Gos distinct features include goroutines and channels, which are language-level tools for handling concurrency and message-passing between components. C would require such things to be hand-rolled or supplied by an external library, but Go provides them out-of-the-box, making it far easier to construct software that needs them. Where Go differs most from C under the hood is in memory management. Go objects are automatically managed and garbage-collected by default. For most programming jobs, this is tremendously convenient. But it also means that any program that requires deterministic memory handling will be harder to write. Go does include the unsafe package for circumventing some of Gos type-handling safeties, such as reading and writing arbitrary memory with a Pointer type. But unsafe comes with a warning that programs written with it may be non-portable and are not protected by the Go 1 compatibility guidelines. Go is well-suited for building programs like command-line utilities and network services, because they rarely need such fine-grained manipulations. The languages developers are also mulling how Go can be a better choice for machine learning, AI, and more demanding workloads. But low-level device drivers, kernel-space operating system components, and other tasks that demand exacting control over memory layout and management are best created in C. In some ways, Rust is a response to the memory management conundrums created by C and C++, and to many other shortcomings of these languages, as well. Rust compiles to native machine code, so its considered on par with C as far as performance. However, memory safety by default is Rusts main selling point. Rusts syntax and compilation rules help developers avoid common memory management blunders. If a program has a memory management issue that can be trapped at compile time, it simply wont compile. Newcomers to the languageespecially coming from a language like C, which provides plenty of room for such bugsspend the first phase of their Rust education learning how to appease the compiler. But Rust proponents argue that this near-term pain has a long-term payoff: safer code that doesnt sacrifice execution speed. Rusts tooling also improves on C. Project and component management are part of the toolchain supplied with Rust by default, which is the same as with Go. There is a default, recommended way to manage packages, organize project folders, and handle a great many other things that in C are ad-hoc at best, with each project and team handling them differently. (C doesnt even have a standard package manager, for instance.) Still, what is touted as an advantage in Rust may not seem like one to a C developer. Rusts compile-time safety features cant be disabled, so even the most trivial Rust program must conform to Rusts memory safety strictures. C may be less safe by default, but it is much more flexible and forgiving when necessary. Another possible drawback is the size of the Rust language. C has relatively few features, even when taking into account the standard library. The Rust feature set is sprawling and continues to grow. As with C++, the larger feature set means more power, but also more complexity. C is a smaller language, but much easier to model mentally, so its perhaps better suited to projects where Rust would be too much. These days, whenever the talk is about software development, Python always seems to enter the conversation. After all, Python is the second best language for everything, and unquestionably one of the most versatile, with thousands of third-party libraries available. Its explosive popularity has made it one of the most, if not the most, widely used programming language in the world. What Python emphasizes, and where it differs most from C, is favoring speed of development over speed of execution. A program that might take an hour to put together in another languagelike Cmight be assembled in Python in minutes. On the flipside, that program might take seconds to execute in C, but a minute to run in Python. (As a good rule of thumb, Python programs generally run an order of magnitude slower than their C counterparts.) But for many jobs on modern hardware, Python is fast enough, and that has been key to its uptake. Another major difference is memory management. Python programs are fully memory-managed by the Python runtime, so developers dont have to worry about the nitty-gritty of allocating and freeing memory. But here again, developer ease comes at the cost of runtime performance. Writing C programs requires scrupulous attention to memory management, but the resulting programs are often the gold standard for pure machine speed. Under the hood, though, Python and C share a deep connection: the reference Python runtime is written in C. This allows Python programs to wrap libraries written in C and C++. Significant chunks of the Python ecosystem of third-party libraries, such as for machine learning, have C code at their core. In many cases, it isnt a question of C versus Python, but more a question of which parts of your application should be written in C and which in Python. If speed of development matters more than speed of execution, and if most of the performant parts of the program can be isolated into standalone components (as opposed to being spread throughout the code), either pure Python or a mix of Python and C libraries make a better choice than C alone. Otherwise, C still rules. Another recent possible contender for both C and C++ is Carbon, a newer language that is currently being actively developed. Carbons goal is to be a modern alternative to C and C++, with a straightforward syntax, modern tooling and code-organization techniques, and solutions to problems C and C++ programmers have long faced. Its also meant to provide interoperation with C++ codebases, so existing code can be migrated incrementally. All this is a welcome effort, given C and C++ have historically had primitive tooling and processes compared to more recently developed languages. So whats the downside? Right now Carbon is an experimental project, not remotely ready for production use. There isnt even a working compiler; just an online code explorer. Its going to be a while before Carbon becomes a practical alternative to C or C++, if it ever does.Photo by Fahim Muntashir on UnsplashC is a general-purpose, high-level, compiler-based, machine-independent structure language that is extensively used in various applications.A general-purpose language means capable of creating all/various types of programs.Now lets talk about which kind of programs or applications we can design with C language.> operating system [e.g : dos, win, Unix..]> Editor [e.g: notepad, wordpad]> Commercial applications> Database> Translator> Device driverAnd many more.Is it worth learning the C programming language in 2030??? absolutely Yes Remember one thing every programming language has borrowed syntax/features directly or indirectly from the C language. Only the C language can clear the programming logic you need for the whole life. If you learn the C language you can easily switch to another language and that will take very little time to master that language.Basically, C is a modular/procedural language. When you have the concept of C structure you can easily relate the topic about class that is involved in object-oriented programming.Lastly, one more thing that is to clear your thought about is it worth or not?If you want to study computer science then first start with C programming otherwise if you just love coding but not in the CS subject then you can go for a project base programming language like python or something else whatever you love to do, that will be helpful because remember that The basic of Computer Science student is to learn how to solve problems. Here project is not the first priority.I hope now you have no doubt about this topic.Now, how you can start with the C language. You can follow the below sequence that I follow to learn. But before starting your very first programming languages, make sure you have some knowledge about the number system like converting numbers from one to another . that will help you more to understand some programming topics.Note: One thing that will help you to be a good programmer is to keep a notebook and pen with you when you learn about the new topic and write down the syntax.Here I will just show topic by topic that you need to cover for clear the concept of C language but not showing the description of the topic. So let's start.> Introduction to C programming.History of CImportance of CC Programming Language StandardBasic Skeleton of Cetc..Remember, this Introduction is not necessary to study but this will help you to keep your interest on your journey.> The basic structure of C programming> How C programming works?> How a C compiler works?> What is a linker and how it works.> What happens in computer memory?> What is an executable file?> Using terminal to run your very first program.> What is variable and how to declare variables and naming conventions.> Keyword and identifiers> Static variable vs global variable> ASCII Table [remember the term which values the letter start and finish and the same as the numbers.] before covering it, you have to know about the basic number system.> Basic input-output functions printf and scanf.> Escape Sequence and their effect> Placeholder / conversion specifier.> Formatted input and output> Fundamental of data types and valid range of data types.> size_t data type in C> How to typecast in C> Introduction to OperatorsArithmetic OperatorsRelational OperatorsLogical OperatorsAssignment OperatorsIncrement and Decrement operatorsConditional operators***Bitwise OperatorsSpecial operators.> Precedence and Associativity of operators> Software development method(SDM)Specification of needsProblem AnalysisDesign and Algorithms representationImplementationTesting and VerificationDocumentation> Selection Structure / Conditional Structure [if, if else, else if , nested if]> Programming style/ Indentation style> Short Circuit Evaluation> Multiple Selection style [switch case]> Data Representation using the signed magnitude> Repetition Structure / Loops[while, do while, for, infinite loop]> Loop Control statement [break , continue]> Nested loop### Solve at least 20 pyramids of stars program using loop. That will help you to understand how the loop works.> Introduction to Arrays> Definition, Declaration, Initialization and Accessing Arrays elements.> Designated initialization of arrays> Learn about Multi-dimensional arrays.### Solve at least 30 problems using 1d array and solve at least matrix adding and multiplying problems using 2d arrays.> Searching and Sorting in Arrays [Bubble sort, Selection sort, Insertion sort, Linear search, Binary search] these basic algorithms will help to build your algorithms concept.> Introduction to Character array and strings> Declare and initialize string variables> Some useful string handling library Introduction to Function> Function Declaration and Definition> Difference between Parameters and Arguments.> Call by value and call by reference> Static and Dynamic scoping> What the terms- heap, stack does.> Introduction to Recursion> Types of Recursion [direct, indirect, tail and nor-tail]> Advantages and Disadvantages of recursion.### Solve minimum 10 problems using recursion> Passing Arrays to functions> Passing String to functions> The scope, visibility and lifetime of variables> automatic variables, external variables, static variables, register variables> Introduction to Pointers.> Definition, Declaration, Initialization and Accessing pointers.> Value of operator in pointers.> Arrays and Pointers> Pointers and character strings> Arrays of pointers> Pointer as function arguments> Pointer to functions> Benefit of pointers> Pointer and functions> Using pointers to 2d Array> String literals.> Dynamic Memory Allocation[malloc() , calloc() , realloc() , free()].> Concept of linked lists> Application of linked lists> Linked list vs array> Structures and Unions in c.> Declaring and Accessing structure members> Array of structures> Structures and functions> Introduction to the Preprocessor> Macro substitution, file inclusion, compiler control directives> File management in C> File and Stream> Define, open and closing a file> Input-output operations on files> Error handling during I/O operations> Command line arguments> Common programming errors> Program testing and debugging> Projects:So get started with the C language and keep learning. Hope to see you all in the next one! Until then Stay Tuned and Happy Coding!!!Thank You! Short Bytes: The first thing that comes to our mind for the term programming language is C. It is a decade old general-purpose high-level programming language which has defied all norms of popularity. The language has been given the name C because it succeeds another language called B.C is one of the most popular computer programming languages which has existed since the last 44 years. It was created by the famous American programmer Dennis Ritchie with the help of Ken Thompson while working at Bell Labs. Most of us have this curiosity that why is the programming language named after a single alphabet. And why it is C, not any other letter. Anyways, we would have questioned the name of the programming language even if a different letter was used. So, lets focus on the story instead.The naming of C doesnt involve any crunchy fiction story, like in the case the Linux Penguin Tux. It is said Linus Torvalds, the creator of Linux, contracted penguinitis after a man-eating penguin running at 100-miles per hour attacked him.The reason why the language was named C by its creator was that it came after B language. Back then, Bell Labs already had a programming language called B at their disposal.The B language, a high-level language created by Thompson, was itself an implementation of the programming language BCPL designed by Martin Richards of Cambridge University.The Unix operating system was originally created at Bell Labs by Ken Thompson, Dennis Ritchie, and others. It was implemented on the PDP-7 (manufactured by Digital Equipment Corporation in 1965) computer using the assembly language. Later, Unix was ported to PDP-11 using assembly language but the developers thought about rewriting it using B language.The B language fellshort of abilities to take benefit of the features offered by the newer member of the PDP series, the PDP-11. Thats when Dennis Ritchie started working on a new high-level programming language taking clues from B language. C had a great influence on the Unix operating system. In fact, most of the Unix system including a big part of its kernel is now available in C language.C has been improved from time to time since its initial release. Thats the reason its still a part of the curriculumin many schools around the world and it is the 2nd popular programming language. The language has been an inspiration to many other programming languages like C++, C#, Java, PHP, etc.If you have something to add, tell us in the comments below.Also Read:Where Did Microsoft Windows Get Its Name From? Aditya Tiwari Aditya likes to cover topics related to Microsoft, Windows 10, Apple Watch, and interesting gadgets. But when he is not working, you can find him binge-watching random videos on YouTube (after he has wasted an hour on Netflix trying to find a good show).Reach out at Ask the publishers to restore access to 500,000+ books. You cant perform that action at this time. This is the code repository for Learn C Programming, published by Packt.A beginner's guide to learning C programming the easy and disciplined way C is a powerful general-purpose programming language that is excellent for beginners to learn. This book will introduce you to computer programming and software development using C. If you're an experienced developer, this book will help you become familiar with the C programming language.This book covers the following exciting features:Understand fundamental programming concepts and implement them in CWrite working programs with an emphasis on code indentation and readabilityBreak existing programs intentionally and learn how to debug codeAdopt good coding practices and develop a clean coding styleExplore general programming concepts that are applicable to more advanced projectsDiscover how you can use building blocks to make more complex and interesting programsUse C Standard Library functions and understand why this is desirableIf you feel this book is for you, get your copy today! All of the code is organized into folders. For example, Chapter02.The code will look like the following:#include int main(){ printf("Hello, world!"); return 0;}Section Memory Leaks on page 410: The sentence "First, the free() function is called on pThing1, which is the pointer element of pThing2. Then, and only then, can we release the memory of pThing1." must read as "First, the free() function is called on pThing1->pThing2, which is the pointer element to pThing2. Then, and only then, can we release the memory of pThing1."Page no. 20: The sentence "The way in which one v sends text to the console is likely different from any other OS, even if they both run on the same hardware." must read as "The way in which one sends text to the console is likely different from any other OS, even if they both run on the same hardware."Page no. 42: It is mentioned that "Four keywords we have already encountered are include, int, main, and return." but main() is not a keyword; it is a token.Page no. 51: The sentence "Your program can never call main itself; main can only be called by the system." must read as "Your program can call main() again itself but this is a very unwise practice."Page no. 55: The sentence "A function can be values given as input to the function." must read as "A function can have values given as input to the function."Page no. 121: Page 121 example codeIt is: printf(" Final: %" , aValue);... printf(" Final: %" , aValue);Should be:printf(" Final: %d" , aValue);...printf(" Final: %d" , aValue);Page no. 121: Page 121 prefixpostfix.c running resultIt is:++aValue (alone) == 7aValue++ (alone) == 8Should be:++aValue (alone) == 6aValue++ (alone) == 6Page no. 121: The sample program is given as // A more predictable result: increment in isolation. ++aValue; printf("++aValue (alone) == %d" , aValue); aValue++; printf("aValue++ (alone) == %d" , aValue);but this should be // A more predictable result: increment in isolation. aValue = 5; ++aValue; printf("++aValue (alone) == %d" , aValue); aValue = 5; aValue++; printf("aValue++ (alone) == %d" , aValue); const int size1D = 5;const int size2D = 4;const int size3D = 3;const int size4D = 7;The above four statements could and should be written as enums, as follows: enum { size1D = 5, size2D = 4, size3D = 3, size4D = 7 };It could also have been declared #define size1D 5 #define size2D 4 #define size3D 3 #define size4D 7Page 192: The code block after "Alternatively, we could write a function that adds two cards, as follows:" is:int sumCards(struct Card c1 , struct Card c2) { int faceValue = c1.faceFalue + c2.faceValue; return faceValue;}It should be:int sumCards(struct Card c1 , struct Card c2) { int faceValue = c1.faceValue + c2.faceValue; return faceValue;}Page 56: Chapter 2 Code Block: hello6.c, line 27Page 63 and 59It is: It Should be: Following is what you need for this book:This book is written for two very diverse audiences.If you're an absolute beginner who only has basic familiarity with operating a computer, this book will help you learn the most fundamental concepts and practices you need to know to become a successful C programmer.If you're an experienced programmer, you'll find the full range of C syntax as well as common C idioms. You can skim through the explanations and focus primarily on the source code provided.With the following software and hardware list you can run all code files present in the book (Chapter 1-25). ChapterSoftware requiredOS required1-25GCC/Clang/Cygwin/MinGWWindows, Mac OS X, and Linux (Any)We also provide a PDF file that has color images of the screenshots/diagrams used in this book. Click here to download it. Practical C Programming [Packt] [Amazon]Extreme C [Packt] [Amazon] Jeff Szuhayis the principal developer at QuarterTil2 which specializes in graphics-rich software chronographs for desktop environments. In his software career of over 35 years, he has engaged in the full range of development activities from systems analysis and systems performance tuning to application design, from initial development through full testing and final delivery.Throughout that time, he has taught computer applications and programming languages at various educational levels from elementary school students to university students, as well as developed and presented professional, on-site training. Click here if you have any feedback or suggestions. If you have already purchased a print or Kindle version of this book, you can get a DRM-free PDF version at no cost.Simply click on the link to claim your free PDF.

How to design a syllabus pdf. Types of syllabus design. How to design syllabus of the course. Steps of syllabus design pdf. Steps of syllabus construction.