DOWNLOAD PDF 11 PROGRAMMING LANGUAGES 41

Chapter 1 : Programming language - Wikipedia

Programming languages come and go. Here are 11 that have given up the spotlight to more modern options.

Formats of string arguments for the printf functions: Python can be compiled in two modes: Literal Unicode strings are written with the prefix u: Code points can be written as hexadecimal: This coercion was a bad design idea because it was the source of a lot of confusion. At the same time, it was not possible to switch completely to Unicode in It took 8 years to switch completely to Unicode: Python 3 was relased in December Narrow build of Python 2 has a partial support of non-BMP characters. Non-BMP characters are encoded as surrogate pairs. Literal byte strings are written with the b prefix: Code points can be used directly in hexadecimal: Each item of a byte string is an integer in range 0â€” Python 3 has a full support of non-BMP characters, in narrow and wide builds. In Python 3, comparing bytes and str gives False, emits a BytesWarning warning or raises a BytesWarning exception depending of the bytes warning flag -b or -bb option passed to the Python program. UTF-8 decoder of Python 2 accept surrogate characters , even if there are invalid, to keep backward compatibility with Python 2. In Python 3, the UTF-8 decoder is strict: They support a lot of encodings. Python provides also many error handlers used to specify how to handle undecodable byte sequences and unencodable characters: It is required because UTF-8 decoder of Python 3 rejects surrogate characters by default.

Chapter 2 : 11 Most Profitable Programming Languages To Learn in - Insider Monkey

11 IoT Programming Languages Worth Knowing Choosing which language to use for an Internet of Things project can be as big a decision as choosing a hardware platform. Here are 11 options to consider for your next coding project.

Commitment to systems programming. Here are the PYPL best programming languages for StackOverflow Stack Overflow is a question-and-answer platform. It has over 4 million users with 10 million questions on the platform. Based on the number of questions, more people use Javascript compared to any other language. It was amazing to see its scale with more than HackerEarth We at HackerEarth provide tons of coding challenges for our users every month and hundreds of hiring challenges for companies looking for recruitment. With our platform supporting over 30 programming languages, users can happily pick one they are most comfortable with. We could create portable applications in this language. Going by the number of jobs listed with the language keyword, Java ranks first, and it has no close competitor. The surprise contender on the list is R. What do professionals feel about these top programming languages? This has resulted in the evolution of more sophisticated languages which are at more higher level than the previous ones. Glassdoor had been listing developer based job When it comes to the job category, the industry has been looking for Java developers, followed by Javascript. High performance computing that was once restricted to scientists, is now mainstream due to deep learning and AI applications. In , we saw Julia applications for algorithmic trading, self-driving cars, 3D printing, risk management, medical diagnostics, air safety, parallel supercomputing and astrophysics. Ruby is a favorite among the startups and has been for some time; by no means is its popularity expected to diminish in the near future. Top Programming languages to learn will be â€” Javascript â€" Scripting language for the worldwide web. As the web take over our universe, so is the use of Javascript which is predominantly used for a web application. A good front-end tool, which could be used in all environment. From adding effects to creating basic functionality games, javascript is industry favorite and top popular programming language to learn. Java â€" Most common programming language as on the day. Used for Android development and most of the financial systems around the world. Java is prominently preferred for its speed and performance. It would not be inaccurate to say that R could be the most sought after language in if you go by the increasing number of jobs related to R advertised on job portals. SQL â€" As more and more people are getting onboard technology, the database has been increasing exponentially. If you are interested in managing the database, SQL is made for you. Termed Special purpose language, SQL is made for a special purpose, unlike general purpose language. Used almost everywhere where database management is required. SQL is a great skill to learn. Arduino â€" IoT is here to stay! And embedded-based programming will be a key player too. With more embedded chips waiting to be coded, Arduino will be the new skill to pick up in As of now, iOS developers are spinning codes as quickly as others and as they come out in the market use of swift would only expand. If the mobile side appeals to you, then learning Swift should be a part of your resolutions! It just means that the wind could be blowing in another direction altogether. No harm in being prepared, right?

Chapter 3 : Programming languages â€” Programming with Unicode

C (/ s iË• /, as in the letter c) is a general-purpose, imperative computer programming language, supporting structured programming, lexical variable scope and recursion, while a static type system prevents many unintended operations.

Most of its changes are in the implementation of the toolchain, runtime, and libraries. As always, the release maintains the Go 1 promise of compatibility. We expect almost all Go programs to continue to compile and run as before. Changes to the language There are no changes to the language specification. Ports As announced in the Go 1. There are known issues with NetBSD on i hardware. The NetBSD race detector support has known issues. This setting is enabled in default kernel configurations, but is sometimes disabled in stripped-down configurations. Go programs currently compile to one WebAssembly module that includes the Go runtime for goroutine scheduling, garbage collection, maps, etc. As a result, the resulting size is at minimum around 2 MB, or KB compressed. Binary size and interop with other languages has not yet been a priority but may be addressed in future releases. If you have existing filenames matching those patterns, you will need to rename them. More information can be found on the WebAssembly wiki page. Tools Modules, package versioning, and dependency management Go 1. Using modules, developers are no longer confined to working inside GOPATH, version dependency information is explicit yet lightweight, and builds are more reliable and reproducible. Module support is considered experimental. Details are likely to change in response to feedback from Go 1. Although the details of module support may change, projects that convert to modules using Go 1. If you encounter bugs using modules, please file issues so we can fix them. For more information, see the go command documentation. Import path restriction Because Go module support assigns special meaning to the symbol in command line operations, the go command now disallows the use of import paths containing symbols. Such import paths were never allowed by go get, so this restriction can only affect users building custom GOPATH trees by other means. Package loading The new package golang. Because it runs an external query command such as go list to obtain information about Go packages, it enables the construction of analysis tools that work equally well with alternative build systems such as Bazel and Buck. Build cache requirement Go 1. Starting in Go 1. The module and package loading support described above already require that the build cache be enabled. If you have disabled the build cache to avoid problems you encountered, please file an issue to let us know about them. Compiler toolchain More functions are now eligible for inlining by default, including functions that call panic. The compiler toolchain now supports column information in line directives. A new package export data format has been introduced. This should be transparent to end users, except for speeding up build times for large Go projects. The compiler now rejects unused variables declared in a type switch guard, such as x in the following example: Assembler The assembler for amd64 now accepts AVX instructions. Debugging The compiler now produces significantly more accurate debug information for optimized binaries, including variable location information, line numbers, and breakpoint locations. There are still limitations to the quality of the debug information, some of which are fundamental, and some of which will continue to improve with future releases. DWARF sections are now compressed by default because of the expanded and more accurate debug information produced by the compiler. This is useful, for example, to call String methods when paused at a breakpoint. This is currently only supported by Delve version 1. Test Since Go 1. In particular, tests that contain an unused variable inside a closure compiled with Go 1. Previously, a type checking error simply caused a warning to be printed, and vet to exit with status 1. Vet now detects the mistake in this example: Cgo Since Go 1. Code that uses these types may need some updating. See the Go 1. This is useful in certain situations. In future releases, godoc will only be a web server. Users should use go doc for command-line help output instead. The godoc web server now shows which version of Go introduced new API features. The initial Go version of types, funcs, and methods are shown right-aligned. For example, see UserCacheDir , with "1. For struct fields, inline comments are added when the struct field was added in a Go version other than when the type itself was introduced. For a struct field example, see ClientTrace. Gofmt One minor detail of the default formatting of Go source code has changed. When formatting expression lists with inline comments, the comments were aligned according to a heuristic.

However, in some cases the alignment would be split up too easily, or introduce too much whitespace. The heuristic has been changed to behave better for human-written code. Note that these kinds of minor updates to gofmt are expected from time to time. In general, systems that need consistent formatting of Go source code should use a specific version of the gofmt binary. Runtime The runtime now uses a sparse heap layout so there is no longer a limit to the size of the Go heap previously, the limit was GiB. The syscall package still makes direct system calls; fixing this is planned for a future release. Performance As always, the changes are so general and varied that precise statements about performance are difficult to make. Most programs should run a bit faster, due to better generated code and optimizations in the core library. Compiler toolchain The compiler now optimizes map clearing operations of the form: The compiler now performs significantly more aggressive bounds-check and branch elimination. It also understands simple arithmetic such as s[i] and can recognize more inductive cases in loops. Furthermore, the compiler now uses bounds information to more aggressively optimize shift operations. Core library All of the changes to the standard library are minor. Minor changes to the library As always, there are various minor changes and updates to the library, made with the Go 1 promise of compatibility in mind.

Chapter 4 : Concepts of Programming Languages (11th Edition) - Ebook pdf and epub

C languageÂ¶. The C language is a low level language, close to the hardware. It has a builtin character string type (wchar_t), but only few libraries support this type.*

If programming plays a role in your life in any way possible, then it is important that you know where and how to bet on your developmental future. It is crucial in programming world to plan the future and to know what will work best for you in advance. While exactly predicting the future of IT may seem like a stretch, it is definitely possible to make projections based on experience and trends. In keeping with our experience and with the inspiration from our love for programming we have stipulated a list of top 11 languages that we think will dominate the year and will be good to learn for making a career or hobby in coming years. This is one of the most popular and widely applied programming languages that have been around since the s. We believe that this language will be every bit as present and dominant if not more as it has been in the past. And guess we are not alone in this belief. The TIOBE index which measures the popularity of various programming languages iterates the fact that Java is the most popular and a largely preferred programming language. The fact that Java is independent i. In simpler terms this language can run on any device, irrespective. Java is not faded by any sort of scalability issues, as is apparent by the numerous large enterprises that are using Java. It has been used for mere applications games such as Temple Run, Angry Birds etc. The language also garners full points when it comes to the performance and reliability. All in all, it will not at all be a stretch to claim that Java will continue its dominant spree in the year So if you are anybody, who wants to be somebody or wants to try their hand at programming, Java is the way to go. It is a dynamic language which makes possible the interaction between client-side script and the users. A language like JS which has been standardized by ECMA Script cannot be forgotten in the future, at least not for as far ahead as we can peek into. If this does not seem like reason enough to make JavaScript a part of our top programming languages for , listed below are a few advantages of this language. This might be reason enough for some programmers to hang onto JS a little while longer. The fact that the language is executed client-side, gives the developer the ability to run code functions almost immediately, and irrespective of where the JavaScript is hosted. Another added advantage of using JavaScript is that it supports all of the modern browsers while producing same results all across. Many additional tools have been written or developed in addition to the core of the language making it easier, interesting and functionally better to use. Also there is a lot of pre-written JavaScript that has been made available by many programmers; people can just use the pre-existing scripts and plug them into their webpage s. That being said, we cannot deny that there are some security and other concerns with this language; but nothing is as strong an influence to lead to its extinction. Python The Python Software Foundation makes available a language that we think will have some level of dominance in the year Now why do we think Python will be one of the top languages of ? Or rather why should you take a chance of learning this language? Python is definitely a very versatile language. Not only that, Python is completely script friendly and it is possible to run it on all major operating systems. The above two are very good reasons to hang onto this soon to be 28 yrs old language in But for those seeking more, Python will definitely not disappoint. Python helps in solving complex problems comparatively faster and it is more flexible thus allowing coders with greater amount of freedom. Nonetheless, with all its advantages and with constant evolution as hand, Python seems to be one of the languages of the future. It was also developed by Bell Labs just like C but this time in collaboration with Bjarne Stroustrup, a Danish computer scientist. An extension for C, this language first appeared in the year about 11 years after C made its first appearance. But once you get comfortable with it, you will start reaping the benefits. So the language that came as a pre-processor to C eventually evolved and became a competitor to C instead. And that really is something! It has huge scope of scalability. Since the language is static, it is very fast. There is regular addition of new features and additional standards to this language, making evolution one of its strong suits. The robustness, the speed, the classes and many other features are enough for us to believe that this programming language will garner some programmer love in Ruby In the s, Yukihito Matsumoto a Japanese computer scientist was working on developing a programming language that would be dynamic, object oriented and reflective. The results of his toil culminated into a programming language that we all know as- Ruby. The language was first released in And even though it has been around for more than two decades, it existence became more prominent after the release of the Rails framework. The condition at present is that, Ruby is considered a powerful language which we believe will be at par with the other programming languages that dominate the year The Rails framework gives the language an edge and makes it faster; thus also making it more enticing to work with it. Ruby is also considered by many programmers to have a different style which appeals to their programmer-y senses. Other reasons which make Ruby worthy of being on this list are: The fact that Ruby creator was inspired by Perl, Ada, Lisp and other modern languages, has resulted in this languages being more readable and easier for those already familiar with other languages. Many huge corporations and entities are using Ruby. For example, Ruby on Rails was used to develop Twitter. Ruby is also quite popular amongst and is preferred by startups and developers alike. So go ahead and discover this language, and it just might change your programming world forever, who knows? Swift This language appeared only two and a half years ago out of the multinational tech company- Apple Inc. Chris Lattner a software developer at Apple was known as the proud author of LLVM, but in he added the language Swift to his list of creations. Also while Swift was a proprietary language when launched in , by the time of its second upgrade to version 2. It is now available under the Apache License 2. Since becoming open-source there has been a rise in the number of people using it, teaching it, researching it and more. It hence is probably a good idea to jump on the Swift bandwagon and be a part of the evolution of this rather new language. Another reason that led us to place Swift on the list of languages we expect to dominate the year is that it works as a great alternative to Objective-C and employs the modern concepts of programming that remained in theory till date. The language itself is built so as to optimize development and the Swift compiler provides optimization in terms of performance. This combo of qualities offered by Swift render it a suitable and important place in the year Rasmus Lerdorf, the programmer of Greenlandic- Danish origin, is the creator of this language. The language made its first debut in the year and has been developing since. Lerdorf once stated that creating PHP as a new programming language was not his actual intention. He was simply trying to extend the CGI s he wrote to maintain his personal homepage, into something that could also work with web forms etc. But once he started, the language grew organically and he merely added one logical step after another. Yes, we believe that this language is definitely going to be prominent in When it comes to web development, PHP is used widely across the globe. So why would someone think of this language as a thing of the past. Another very good thing about PHP is that it is extremely easy to get started with. It is very easy to scale up with PHP. The shared nothing architect of the language makes it pro-scalability. And we would like to end our argument on the benefits of this language by mentioning that the documentation for PHP is undeniably the best on the web. The C Programming Language One of the oldest programming languages out there, C made its first appearance about 45 years ago in the year The language was developed by Dennis Ritchie at the Bell Labs, and since its inception the language has come a long way. For decades, this language has persisted and stayed as one of the top programming languages, and by our estimations that is not going to change anytime in the near future. This imperative, general purpose computer programming language supports structured programming. The language is as powerful as it was, and is a great solution to when you want to directly influence hardware by bypassing SDKs and APIs. Most programmers work with multiple languages and C is a great stepping stone for progressing in the world of programming languages. The various data types and the operators render the programs coded in C extremely efficient and above par. This is another reason why programmers will not let go of this language and it hence has every possibility of being dominant in as well. Another fact that adds to the popularity of this language is that it is the mostly commonly used language across industry and also in universities and colleges. The language meets all the international standards, falls into all the recommendations and is essential part of programming today. NET framework initiative of Microsoft, the language has since attained the multi-paradigm language status. Its latest version was released in , and its next version is currently under development. There are many advantages and distinguishing features of this language which make it one of the languages to be learned. These advantages and distinguishing features include An automatic garbage collection mechanism The reflection capabilities CIL- the fact that it is complied with intermediate language which helps it and you work independently. The ability to define classes within classes Simplicity of multithreading Delegates- cleaner event management Since it is written in the. NET framework, it can access code written in and inherit classes written in any. NET compliant language It may be considered as rather unrestrictive and liberal in comparison to other languages like Java etc. Indexers Generics The interoperability model of C-sharp makes it easier to work with this language Considering these features and advantages we believe that the language does live up to its name and is definitely sharper than many others in its league. As the general programmer consensus goes- once you have gained some experience with this language, you will actually start having fun. Well, there is only one way to find that out right!! Even decades later, this language persists with its relevance and importance in programming. Yes it is not a programming language per se, but it still chalks a spot here. Not only is HTML widely considered the building blocks for learning any web language, most websites are also built with HTML making it an essential part of programming.

Chapter 5 : Top 7 most popular programming languages to learn | HackerEarth Blog

Chapter Programming Languages and Program Development kareem mohamed. Understand 6 Best Programming Languages You Must Learn - Duration.

The relatively static nature of the top ten languages is interesting, certainly, in a technology landscape that is best characterized as heavily fragmented and growing more so at an accelerating rate. New language entrants are behind from the day they are released, in other words, which makes displacing the most popular languages a significant and uphill battle. Outside of the top tier, however, there are several changes worth discussing. Kotlin -1 , Scala 2 , Clojure 4 , Groovy 2: In general, we caution readers not to assign too much weight to small changes in the rankings; the differences between one spot or another, in general, tend to be slight. This remains true as we look at the case of the JVM languages. Kotlin took a minor step back from 27th to 28th, while its JVM counterparts Scala 12 , Clojure 21 and Groovy 21 all showed bounces â€" some significant. This was notable for Scala in particular, as it had dropped for three consecutive quarters entering this run. A week or so ago, we received the following inquiry from a large vendor: But the inquiry was well timed, because according to our rankings Julia is making slow but steady progress. This quarter Julia jumped three spots to 36, which is its fourth consecutive quarter of growth 36, 39, 40, R has dropped two spots previously, only to bounce back a quarter or two later. Its domain specific nature means that it is never likely to compete for the very top spots on this ranking, but it remains dominant and broadly used within its area of expertise. Having jumped three spots in the last edition of these rankings and being the subject of much discussion in programming language communities, the question for Rust in Q3 was whether it could continue its upward trajectory into the Top For the short term, at least, the answer is no, as the headwinds kept Rust at 23 for the second consecutive quarter. This quantitative stall is to some degree unsurprising, because growth becomes increasingly difficult the higher up a language ascends in the rankings. This has the consequence of boosting Objective C from a tie for 10th to outright 9th place. This may seem consequential, but the reality is that the difference in their collective rankings, while measurable, is slight. If or when it is ever established as a server side language as IBM and others have attempted to push at times, its ceiling is virtually unlimited. Go 2 , TypeScript In keeping with the overall theme this quarter of retrenchment, however, TypeScript dropped for the first time in several quarters. It was a mere two spot slide, and the language remains enormously popular at 16th on our rankings, but it was notable nonetheless. After a one quarter single point drop, Go bounced back, essentially trading places with TypeScript at While the languages are quite distinct in where and how they are used, it will be interesting to see how these languages perform relative to one another moving forward given that they seem to occupy relatively the same spots on the rankings with frequency. My colleague Rachel Stephens evaluated the available options for extracting rankings from GitHub data, and wrote and executed the queries that are responsible for the GitHub axis in these rankings. She is also responsible for the query design and collection for the Stack Overflow data. The original version of this chart had the axes reversed; this has been fixed and the image updated. Thanks to Edgar Arenas for the tip.

Chapter 6 : 11 Programming Languages that will Dominate - GeeksProgramming

What it is: A general-purpose, imperative programming language developed in the early '70s, C is the oldest and most widely used language, providing the building blocks for other popular languages.

Can you release faster without sacrificing quality? DevOps use languages for software development and deployment automation. It is an open-source configuration management tool that uses its own declarative scripting language to build automation and management scripts. Bash It is the most important and frequently used Unix Shell. It provides the command shell and scripting language used to automate processes on tens of thousands of Linux servers around the world. It has become a normal programming language in organizations. PHP is used for all stages of application development. Perl Perl is an interpreted language often used for scripts that hold together bits and pieces of larger applications. Ruby Ruby is an object-oriented, compiled, full-featured programming language with a syntax that should be familiar to anyone who knows Python or Perl. Java It is one of the top languages taught in university computer science programs. JavaScript is becoming more completely functional as time goes on, with server-side implementations and JavaScript in virtual machines. C C is a procedural language. C started life as a tool for automating processes inside long-distance switches. C is the next best thing to assembler if you need to cut through all the APIs and SDKs and directly manipulate the hardware itself. Tcl Tcl is a scripting language that can take you from application deployment automation to industrial Internet of Things IoT embedded control. It is widely used, open source, and applicable to a variety of situations. So does anyone who is working on applications designed to reach back into a database for processing fodder. The ability to drop a line of SQL into an application can save you time. Get your test automation project off to the right start. Download your free test planning template and a day no-obligation trial of Ranorex Studio today! Read More From DZone.

Chapter 7 : Go Release Notes - The Go Programming Language

Objectives. Explain what a programming language is and how it works. (p.) Explain the development of programming languages over the years and the benefits and drawbacks of high-level programming languages.

Mostly, it happens when the new readers quit utilizing the eBooks as they are not able to use all of them with the appropriate and effectual style of reading these books. There present variety of motives behind it due to which the readers stop reading the eBooks at their first most attempt to utilize them. Nevertheless, there exist some techniques that may help the readers to really have a good and effectual reading encounter. A person ought to correct the appropriate brightness of display before reading the eBook. It is a most common issue that the majority of the folks usually endure while using an eBook. As a result of this they have problems with eye sores and head aches. The best solution to overcome this acute difficulty is to decrease the brightness of the displays of eBook by making particular changes in the settings. You can even adjust the brightness of screen determined by the kind of system you are utilizing as there exists bunch of the approaches to adjust the brightness. It is suggested to keep the brightness to potential minimum amount as this can help you to raise the time that you can spend in reading and provide you great comfort onto your eyes while reading. A great eBook reader ought to be installed. You may also use free software that can provide the readers that have many functions to the reader than just an easy platform to read the desirable eBooks. You can also save all your eBooks in the library that is additionally supplied to the user by the software program and have a good display of all your eBooks as well as access them by identifying them from their specific cover. Besides offering a place to save all your precious eBooks, the eBook reader software even offer you a high number of characteristics to be able to enhance your eBook reading experience compared to the conventional paper books. You can also enhance your eBook reading experience with help of alternatives supplied by the software program like the font size, full display mode, the specific variety of pages that need to be exhibited at once and also change the colour of the backdrop. You ought not use the eBook consistently for a lot of hours without breaks. You need to take appropriate rests after specific intervals while reading. Nevertheless, this does not mean that you ought to step away from the computer screen every now and then. Continuous reading your eBook on the computer screen for a long time without taking any break can cause you headache, cause your neck pain and suffer from eye sores and in addition cause night blindness. So, it is essential to provide your eyes rest for a while by taking rests after specific time intervals. This can help you to prevent the troubles that otherwise you may face while reading an eBook continuously. While reading the eBooks, you should prefer to read huge text. Usually, you will note that the text of the eBook tends to be in moderate size. So, boost the size of the text of the eBook while reading it on the screen. It is suggested that never use eBook reader in full screen mode. It is suggested not to go for reading the eBook in fullscreen mode. Though it might appear simple to read with full screen without turning the page of the eBook quite often, it set ton of stress in your eyes while reading in this mode. Always prefer to read the eBook in exactly the same length that would be similar to the printed book. This really is so, because your eyes are used to the span of the printed book and it would be comfy for you to read in the same way. By using different techniques of page turn you can also boost your eBook experience. You can try many ways to turn the pages of eBook to improve your reading experience. Check out whether you can turn the page with some arrow keys or click a certain part of the display, aside from utilizing the mouse to handle everything. Attempt to use the mouse if you are comfy sitting back. Lesser the movement you need to make while reading the eBook better will be your reading experience. This will definitely help to make reading easier. By using each one of these effective techniques, you can definitely boost your eBook reading experience to an excellent extent. These tips will help you not only to prevent particular dangers that you may face while reading eBook consistently but also facilitate you to take pleasure in the reading experience with great comfort. The download link provided above is randomly linked to our ebook promotions or third-party advertisements and not to download the ebook that we reviewed. We recommend to buy the ebook to support the author. Thank you for reading.

Chapter 8 : 10 Programming Languages You Should Learn Right Now

You can use desktop programming languages like C# to support the programming aspects. But Web allows you to be interface-first instead of second. With desktop programming languages you could spend a long long time getting ready to even implement an interface.

The similarity between these two operators assignment and equality may result in the accidental use of one in place of the other, and in many cases, the mistake does not produce an error message although some compilers produce warnings. The program prints "hello, world" to the standard output , which is usually a terminal or screen display. The original version was: This causes the compiler to replace that line with the entire text of the stdio. The angle brackets surrounding stdio. The next line indicates that a function named main is being defined. The main function serves a special purpose in C programs; the run-time environment calls the main function to begin program execution. The type specifier int indicates that the value that is returned to the invoker in this case the run-time environment as a result of evaluating the main function, is an integer. The keyword void as a parameter list indicates that this function takes no arguments. The next line calls diverts execution to a function named printf , which in this case is supplied from a system library. The string literal is an unnamed array with elements of type char, set up automatically by the compiler with a final 0-valued character to mark the end of the array printf needs to know this. The return value of the printf function is of type int, but it is silently discarded since it is not used. A more careful program might test the return value to determine whether or not the printf function succeeded. The semicolon ; terminates the statement. The closing curly brace indicates the end of the code for the main function. Formerly an explicit return 0; statement was required. This is interpreted by the run-time system as an exit code indicating successful execution. Please help improve this article by adding citations to reliable sources. Unsourced material may be challenged and removed. October Learn how and when to remove this template message The type system in C is static and weakly typed , which makes it similar to the type system of ALGOL descendants such as Pascal. Integer type char is often used for single-byte characters. C99 added a boolean datatype. There are also derived types including arrays , pointers , records struct , and unions union. C is often used in low-level systems programming where escapes from the type system may be necessary. The compiler attempts to ensure type correctness of most expressions, but the programmer can override the checks in various ways, either by using a type cast to explicitly convert a value from one type to another, or by using pointers or unions to reinterpret the underlying bits of a data object in some other way. For example, a comparison of signed and unsigned integers of equal width requires a conversion of the signed value to unsigned. This can generate unexpected results if the signed value is negative. Pointers C supports the use of pointers , a type of reference that records the address or location of an object or function in memory. Pointers can be dereferenced to access data stored at the address pointed to, or to invoke a pointed-to function. Pointers can be manipulated using assignment or pointer arithmetic. Pointer arithmetic is automatically scaled by the size of the pointed-to data type. Pointers are used for many purposes in C. Text strings are commonly manipulated using pointers into arrays of characters. Dynamic memory allocation is performed using pointers. Many data types, such as trees , are commonly implemented as dynamically allocated struct objects linked together using pointers. Pointers to functions are useful for passing functions as arguments to higher-order functions such as qsort or bsearch or as callbacks to be invoked by event handlers. Dereferencing a null pointer value is undefined, often resulting in a segmentation fault. Null pointer values are useful for indicating special cases such as no "next" pointer in the final node of a linked list , or as an error indication from functions returning pointers. In appropriate contexts in source code, such as for assigning to a pointer variable, a null pointer constant can be written as 0, with or without explicit casting to a pointer type, or as the NULL macro defined by several standard headers. In conditional contexts, null pointer values evaluate to false, while all other pointer values evaluate to true. Since the size and type of the pointed-to object is not known, void pointers cannot be dereferenced, nor is pointer arithmetic on them allowed, although they can easily be and in many contexts implicitly are converted to and from any other object pointer type. Because they are typically unchecked, a pointer variable can be made to point to any arbitrary location, which can cause undesirable effects. Although properly used pointers point to safe places, they can be made to point to unsafe places by using invalid pointer arithmetic ; the objects they point to may continue to be used after deallocation dangling pointers ; they may be used without having been initialized wild pointers ; or they may be directly assigned an unsafe value using a cast, union, or through another corrupt pointer. In general, C is permissive in allowing manipulation of and conversion between pointer types, although compilers typically provide options for various levels of checking. Some other programming languages address these problems by using more restrictive reference types. C string Array types in C are traditionally of a fixed, static size specified at compile time. The more recent C99 standard also allows a form of variable-length arrays. Since arrays are always accessed in effect via pointers, array accesses are typically not checked against the underlying array size, although some compilers may provide bounds checking as an option. If bounds checking is desired, it must be done manually. C does not have a special provision for declaring multi-dimensional arrays , but rather relies on recursion within the type system to declare arrays of arrays, which effectively accomplishes the same thing. The index values of the resulting "multi-dimensional array" can be thought of as increasing in row-major order. Multi-dimensional arrays are commonly used in numerical algorithms mainly from applied linear algebra to store matrices. The structure of the C array is well suited to this particular task. However, since arrays are passed merely as pointers, the bounds of the array must be known fixed values or else explicitly passed to any subroutine that requires them, and dynamically sized arrays of arrays cannot be accessed using double indexing. A workaround for this is to allocate the array with an additional "row vector" of pointers to the columns. C99 introduced "variable-length arrays" which address some, but not all, of the issues with ordinary C arrays. This implies that an array is never copied as a whole when named as an argument to a function, but rather only the address of its first element is passed. Therefore, although function calls in C use pass-by-value semantics, arrays are in effect passed by reference. The latter only applies to array names: However, arrays created by dynamic allocation are accessed by pointers rather than true array variables, so they suffer from the same sizeof issues as array pointers. Thus, despite this apparent equivalence between array and pointer variables, there is still a distinction to be made between them. Even though the name of an array is, in most expression contexts, converted into a pointer to its first element , this pointer does not itself occupy any storage; the array name is not an l-value , and its address is a constant, unlike a pointer variable. Consequently, what an array "points to" cannot be changed, and it is impossible to assign a new address to an array name. Array contents may be copied, however, by using the memcpy function, or by accessing the individual elements. Memory management One of the most important functions of a programming language is to provide facilities for managing memory and the objects that are stored in memory. C provides three distinct ways to allocate memory for objects: For example, static memory allocation has little allocation overhead, automatic allocation may involve slightly more overhead, and dynamic memory allocation can potentially have a great deal of overhead for both allocation and deallocation. The persistent nature of static objects is useful for maintaining state information across function calls, automatic allocation is easy to use but stack space is typically much more limited and transient than either static memory or heap space, and dynamic memory allocation allows convenient allocation of objects whose size is known only at run-time. Most C programs make extensive use of all three. Where possible, automatic or static allocation is usually simplest because the storage is managed by the compiler, freeing the programmer of the potentially error-prone chore of manually allocating and releasing storage. However, many data structures can change in size at runtime, and since static allocations and automatic allocations before C99 must have a fixed size at compile-time, there are many situations in which dynamic allocation is necessary. See the article on malloc for an example of dynamically allocated arrays. Unlike automatic allocation, which can fail at run time with uncontrolled consequences, the dynamic allocation functions return an indication in the form of a null pointer value when the required storage cannot be allocated. Static allocation that is too large is usually detected by the linker or loader , before the program can even begin execution. Unless otherwise specified, static objects contain zero or null pointer values upon program startup. Automatically and dynamically allocated objects are initialized only if an initial value is explicitly specified; otherwise they initially have indeterminate values typically, whatever bit pattern happens to be present in the storage , which might not even represent a valid value for that type. If the program attempts to access an uninitialized value, the results are undefined. Many modern compilers try to detect and warn about this problem, but both false positives and false negatives can occur. Another issue is that heap memory allocation has to be synchronized with its actual usage in any program in order for it to be reused as much as possible. For example, if the only pointer to a heap memory allocation goes out of scope or has its value overwritten before free is called, then that memory cannot be recovered for later reuse and is essentially lost to the program, a phenomenon known as a memory leak. Conversely, it is possible for memory to be freed but continue to be referenced, leading to unpredictable results. Typically, the symptoms will appear in a portion of the program far removed from the actual error, making it difficult to track down the problem. Such issues are ameliorated in languages with automatic garbage collection. Libraries The C programming language uses libraries as its primary method of extension. In C, a library is a set of functions contained within a single "archive" file. Each library typically has a header file , which contains the prototypes of the functions contained within the library that may be used by a program, and declarations of special data types and macro symbols used with these functions. This library supports stream input and output, memory allocation, mathematics, character strings, and time values. Several separate standard headers for example, stdio. Another common set of C library functions are those used by applications specifically targeted for Unix and Unix-like systems, especially functions which provide an interface to the kernel. Since many programs have been written in C, there are a wide variety of other libraries available. Libraries are often written in C because C compilers generate efficient object code ; programmers then create interfaces to the library so that the routines can be used from higher-level languages like Java , Perl , and Python. July Learn how and when to remove this template message A number of tools have been developed to help C programmers find and fix statements with undefined behavior or possibly erroneous expressions, with greater rigor than that provided by the compiler. The tool lint was the first such, leading to many others. Automated source code checking and auditing are beneficial in any language, and for C many such tools exist, such as Lint. A common practice is to use Lint to detect questionable code when a program is first written. Once a program passes Lint, it is then compiled using the C compiler. Also, many compilers can optionally warn about syntactically valid constructs that are likely to actually be errors. MISRA C is a proprietary set of guidelines to avoid such questionable code, developed for embedded systems.

Chapter 9 : Chapter Programming Languages and Program Development

A programming language is a formal language, which comprises a set of instructions used to produce various kinds of calendrierdelascience.commming languages are used to create programs that implement specific algorithms.

Static semantics[edit] The static semantics defines restrictions on the structure of valid texts that are hard or impossible to express in standard syntactic formalisms. Examples include checking that every identifier is declared before it is used in languages that require such declarations or that the labels on the arms of a case statement are distinct. Other forms of static analyses like data flow analysis may also be part of static semantics. Newer programming languages like Java and C have definite assignment analysis , a form of data flow analysis, as part of their static semantics. Semantics of programming languages Once data has been specified, the machine must be instructed to perform operations on the data. For example, the semantics may define the strategy by which expressions are evaluated to values, or the manner in which control structures conditionally execute statements. The dynamic semantics also known as execution semantics of a language defines how and when the various constructs of a language should produce a program behavior. There are many ways of defining execution semantics. Natural language is often used to specify the execution semantics of languages commonly used in practice. A significant amount of academic research went into formal semantics of programming languages , which allow execution semantics to be specified in a formal manner. Results from this field of research have seen limited application to programming language design and implementation outside academia. Data type , Type system , and Type safety A type system defines how a programming language classifies values and expressions into types, how it can manipulate those types and how they interact. The goal of a type system is to verify and usually enforce a certain level of correctness in programs written in that language by detecting certain incorrect operations. Any decidable type system involves a trade-off: In order to bypass this downside, a number of languages have type loopholes, usually unchecked casts that may be used by the programmer to explicitly allow a normally disallowed operation between different types. In most typed languages, the type system is used only to type check programs, but a number of languages, usually functional ones, infer types , relieving the programmer from the need to write type annotations. The formal design and study of type systems is known as type theory. Typed versus untyped languages[edit] A language is typed if the specification of every operation defines types of data to which the operation is applicable, with the implication that it is not applicable to other types. The invalid operation may be detected when the program is compiled "static" type checking and will be rejected by the compiler with a compilation error message, or it may be detected when the program is run "dynamic" type checking , resulting in a run-time exception. Many languages allow a function called an exception handler to be written to handle this exception and, for example, always return "-1" as the result. A special case of typed languages are the single-type languages. These are often scripting or markup languages, such as REXX or SGML , and have only one data type[dubious â€" discuss]-â€"most commonly character strings which are used for both symbolic and numeric data. In contrast, an untyped language, such as most assembly languages , allows any operation to be performed on any data, which are generally considered to be sequences of bits of various lengths. In practice, while few languages are considered typed from the point of view of type theory verifying or rejecting all operations , most modern languages offer a degree of typing. Static versus dynamic typing[edit] In static typing , all expressions have their types determined prior to when the program is executed, typically at compile-time. In the first case, the programmer must explicitly write types at certain textual positions for example, at variable declarations. In the second case, the compiler infers the types of expressions and declarations based on context. Complete type inference has traditionally been associated with less mainstream languages, such as Haskell and ML. Dynamic typing , also called latent typing, determines the type-safety of operations at run time; in other words, types are associated with run-time values rather than textual expressions. Among other things, this may permit a single variable to refer to values of different types at different points in the program execution. However, type errors cannot be automatically detected until a piece of code is actually executed, potentially making debugging more difficult. Weak and strong typing[edit]

Weak typing allows a value of one type to be treated as another, for example treating a string as a number. Strong typing prevents the above. An attempt to perform an operation on the wrong type of value raises an error. An alternative definition for "weakly typed" refers to languages, such as Perl and JavaScript , which permit a large number of implicit type conversions. Such implicit conversions are often useful, but they can mask programming errors. Strong and static are now generally considered orthogonal concepts, but usage in the literature differs. Some use the term strongly typed to mean strongly, statically typed, or, even more confusingly, to mean simply statically typed. Thus C has been called both strongly typed and weakly, statically typed. This is extremely similar to somehow casting an array of bytes to any kind of datatype in C without using an explicit cast, such as int or char. Standard library and run-time system[edit] Main article: Core libraries typically include definitions for commonly used algorithms, data structures, and mechanisms for input and output. The line between a language and its core library differs from language to language. In some cases, the language designers may treat the library as a separate entity from the language. Indeed, some languages are designed so that the meanings of certain syntactic constructs cannot even be described without referring to the core library. For example, in Java , a string literal is defined as an instance of the java. Conversely, Scheme contains multiple coherent subsets that suffice to construct the rest of the language as library macros, and so the language designers do not even bother to say which portions of the language must be implemented as language constructs, and which must be implemented as parts of a library. Design and implementation[edit] Programming languages share properties with natural languages related to their purpose as vehicles for communication, having a syntactic form separate from its semantics, and showing language families of related languages branching one from another. A significant difference is that a programming language can be fully described and studied in its entirety, since it has a precise and finite definition. While constructed languages are also artificial languages designed from the ground up with a specific purpose, they lack the precise and complete semantic definition that a programming language has. Many programming languages have been designed from scratch, altered to meet new needs, and combined with other languages. Many have eventually fallen into disuse. Although there have been attempts to design one "universal" programming language that serves all purposes, all of them have failed to be generally accepted as filling this role. Programs range from tiny scripts written by individual hobbyists to huge systems written by hundreds of programmers. Programmers range in expertise from novices who need simplicity above all else, to experts who may be comfortable with considerable complexity. Programs must balance speed, size, and simplicity on systems ranging from microcontrollers to supercomputers. Programs may be written once and not change for generations, or they may undergo continual modification. Programmers may simply differ in their tastes: One common trend in the development of programming languages has been to add more ability to solve problems using a higher level of abstraction. The earliest programming languages were tied very closely to the underlying hardware of the computer. As new programming languages have developed, features have been added that let programmers express ideas that are more remote from simple translation into underlying hardware instructions. Because programmers are less tied to the complexity of the computer, their programs can do more computing with less effort from the programmer. This lets them write more functionality per time unit. However, this goal remains distant and its benefits are open to debate. Dijkstra took the position that the use of a formal language is essential to prevent the introduction of meaningless constructs, and dismissed natural language programming as "foolish". The most important of these artifacts are the language specification and implementation. Programming language specification The specification of a programming language is an artifact that the language users and the implementors can use to agree upon whether a piece of source code is a valid program in that language, and if so what its behavior shall be. A programming language specification can take several forms, including the following: An explicit definition of the syntax, static semantics, and execution semantics of the language. While syntax is commonly specified using a formal grammar, semantic definitions may be written in natural language e. A description of the behavior of a translator for the language e. The syntax and semantics of the language have to be inferred from this description, which may be written in natural or a formal language.