Custom Computer Services, Inc.

CCS C Compiler IDE Statistics: Code Metrics for Project Management

Code Metric results are often used to assist embedded system programmers and designers with a deeper understanding of their code efficiencies. The CCS C Compiler IDE measures and calculates the most common metrics used in C Code. These metrics include: Functions, Statements and Comments; Halstead's Complexity Metrics; Cyclomatic Complexity; and Maintainability Index. Software project managers can use these results to identify problems in source code, program efficiencies, and determine test procedures. When used carefully the code metrics provided by the CCS C IDE can speed up decision making processes, provide justification for software decisions, increase system resources and ultimately reduce costs.

Functions, Statements, and Comments

The basic level of metrics that the IDE provides are the number of Functions, Statements, and Comments in a program. These basic measurements are the basis for the other metrics CCS IDE provides, and are still used by many embedded programmers.

Using Functions, Statements, and Comments

A common goal of system programmers is to keep the number of functions low. There are many conflicting ideas about how many statements an ideal function should have; however, it's key to note that keeping average function length as short as possible makes it more efficient to find and fix bugs.

Another measure of a program's efficiency is through number of statements in the program's code. It is generally agreed that there is not much value in measuring code this way, but it can provide a guide for for how long a project will take, if the length of the projects source code can be estimated. Number of comments provides for the total number of comments found in the source code. These metrics offer insight into measuring some of the complexities of the code, and offer basic metrics of the code, from which the programmer can systemically make decisions.

Halstead Metrics

Halstead Metrics were first published in 1977, and are a well established way of determining the size and complexity of source code. The CCS C IDE reports Halstead metrics Volume (V) and Difficulty (D) for each function in a project, and reports Volume (V), Difficulty (D) and several other derived metrics for the project as a whole. Volume (V) provides a more sophisticated measure of source code size versus number of lines of code. Difficulty (D) attempts to measure the difficulty of writing, or programming, the code.

For the entire project the CCS C IDE also reports Effort to Implement (E), Time to Implement (T), and Estimated Delivered Bugs (B). Both Time to Implement (T) and Estimated Delivered Bugs (B) take into account some assumptions of the source code, but provides reasonable results for these measurements. Effort to Implement (E) is related to both the Volume (V) and Difficulty (D) of a function or project. The Time to Implement (T) and the Estimated Delivered Bugs (B) are generated from the Effort to Implement (E). (Virtual Machinery, 2011, Halstead, para. 7-9)

Using Halstead Metrics

A high Difficulty (D) result for an entire project offers the programmer justification for spending more time on a project than another project of similar length or Volume (V). The Volume (V) for an entire project is a better measure of project size than number of lines of code when trying to determine how long it would take to read through and understand the project. When comparing two programs, that accomplish the same task, these metrics can be compared between programs providing information for technical and non-technical managers. Volume (V) and Difficulty (D) provide objective results respectively; however, they do not capture the subjective manners in which two programs may differ.

The Volume (V) and Difficulty (D) per Function metrics are more valuable for embedded programmers. A high Volume (V) may mean that a function should be split into smaller pieces. A high Difficulty (D) indicates that a variety of different operations are performed, and the purpose of the function may need to be streamlined. These measurements work best when they are used to quickly find problematic areas in a

large code-base. It is the nature of some tasks that they lead to a high Difficulty (D) or Volume (V), but softer rules like "All Volumes above 3000 must include justification for their size in their comments" allow for this. (McCabe, 1996, 2.5 Limiting cyclomatic complexity to 10, para. 1)

Since all metrics are calculated when the project is finished the purpose of Time to Implement (T) could be thought of as the time to re-implement, if the project needed to be rewritten in a different language. Time to Implement (T) is the most controversial of Halstead's Metrics and should be taken with a grain of salt. (Virtual Machinery, 2011, Halstead , para. 14)

Cyclomatic Complexity

Cyclomatic Complexity is a measure of the number of independent paths through a unit of code. It can be used as a measure of difficulty to understanding similar Halstead's Difficulty (D), but it is also useful in determining the necessary number of test cases for the unit of code. Cyclomatic Complexity was developed in 1976 and is a well established technique. (Karakaþ, 1998, McCabe's Cyclomatic Number, para. 1) The CCS C IDE reports Cyclomatic Complexity only for the entire program, not individual functions.

Using Cyclomatic Complexity

One common programming structure that most C programmers find clear and easy to understand is the switch statement. The switch statement causes large cyclomatic complexities and so, as with Halstead's Metrics, Cyclomatic Complexity is useful when applied carefully. It may be poor form to avoid the commonly used and often efficient switch statement because of the large increase in Cyclomatic Complexity it causes. (McCabe, 1996, 2.5 Limiting cyclomatic complexity to 10, para. 1-2)

With these caveats in mind Cyclomatic Complexity can be used to judge program length, and difficulty, but Cyclomatic Complexity also has another use. Baseline Method is a software testing method which uses the theory behind Cyclomatic Complexity. The Baseline Method tests each possible branch in the program. In the Baseline Method the minimum amount of tests necessary is equal to the Cyclomatic Complexity. Therefore Cyclomatic Complexity can be used as the minimum amount of tests for any testing method. (McCabe, 1996, 6 The Baseline Method, para. 1-2)

Maintainability Index

Maintainability Index is a more modern measurement which was designed in 1991. Maintainability Index takes into account the Volume (V), Cyclomatic Complexity, and other measurements from the code whereby measuring the ease with which code can be maintained. Maintainability Index is applied only to the program as a whole, and a larger number is better. There are several ways to report the Maintainability Index. CCS C IDE provides a maximum number of 171 and can provide a negative metric as well. (Virtual Machinery, 2011, MI and MINC, para. 1)

Using Maintainability Index

There is strong evidence that Maintainability Index works in determining how maintainable a program will be over time. It takes into account many factors, and like the other metrics cannot be the only judge of a program. It can be used as a way to encourage good programming practice, by making it a goal to increase the Maintainability Index of the program in each subsequent release. Ways to increase the Maintainability Index include reducing the Cyclomatic Complexity, reducing the Volume (V), and reducing the number of lines of code.

Companies such as Hewlett-Packard have used the Maintainability Index to make large purchasing decisions, and Maintainability Index is a way for non-technical management to evaluate the cost of good coding practices. Purchasing decisions can be evaluated, but so can the companies own software decisions. A decision where the quality of the product could be improved if it is delayed two weeks could be aided by describing the software quality in terms of its Maintainability Index. Comparing Maintainability Indexes between different programs is valid if they perform the same task. However, keep in mind, certain tasks will inherently have lower Maintainability Indexes than others. (Virtual Machinery, 2011, MI and MINC, para. 8)

Conclusion

The program metric tools provided by the CCS C IDE can be used to identify problematic areas in programs, judge quality of code, and improve testing procedures. It provides well known measures of program complexity, size and maintainability. When the the number of Functions, Statements, and Comments, the Halstead Metrics, the Cyclomatic Complexity, and the Maintainability Index are used correctly they can provide an additional tool to help software project managers to make more informed decisions, increase system resources, and ultimately reduce costs.

References

Karakaþ, Ümit & Sultanoðlu, Sencer. (1998). Complexity Metrics and Models.
Retrieved from http://yunus.hacettepe.edu.tr/~sencer/complexity.htm

McCabe, Thomas and Watson, Arthur H. (1996). Structured Testing: A Testing Methodology Using the Cyclomatic Complexity Metric.
Retrieved from http://hissa.nist.gov/HHRFdata/Artifacts/ITLdoc/235/title.htm

Virtual Machinery. (2011). The Halstead Metrics.
Retrieved from http://www.virtualmachinery.com/sidebar2.htm

Virtual Machinery. (2011). MI and MINC - Maintainability Index.
Retrieved from http://www.virtualmachinery.com/sidebar4.htm

About CCS

Established in 1996, CCS is a leading worldwide supplier of embedded software, and hardware development tools, that enable companies to develop premium products based on Microchip PIC®MCU and dsPIC® DSC devices. CCS C Compilers are the most advanced, highly developed and most widely used compiler in the industry. These compilers include a generous library of built-in functions, pre-processor commands, and ready-to-run example programs to quickly jump-start any project. CCS IDE C compiler products provide a unique Profiler Tool to track time and usage information for use on functions, code blocks, as well as receiving live data from running programs. Complete proven tool chains include a full line of programmers and debuggers, application specific hardware prototyping boards, and software development kits. CCS is also a leading provider of electronic engineering services for embedded software development, R&D support, hardware design, and custom electronic products that adhere to our client's high-quality standards.

Learn more by visiting **www.ccsinfo.com**