CINT

- A C/C++ interpreter, written in C.
- Written by Masaharu Goto of HP Japan.
- Used in ROOT.
- Freely available for noncommercial use.

- Support K&R-C, ANSI-C, ANSI-C++
  Multiple inheritance, virtual function, function overloading, operator overloading, default parameter, template, etc. Cint is solid enough to interpret its own source code.

- Handling Huge C/C++ source code

- Interpretation and Native Code Execution can be mixed

- Single-Language solution

- Bridge between serious programmers and other professionals
  C++ looks rather easy under the interpreter environment.

- Dynamic C++
  It can process C++ statements from command line ... Extended Run Time Type Identification mechanism ... will allow you to explore unthinkable way of using C++.

- Built-in Debugger, class browser

- Secure C++
  In security mode, unsafe language constructs are prohibited and garbage collection is activated.

- Portability

- CINT users spread world wide
  Many people in U.S, Europe and Asia are using Cint. Number of C/C++ libraries have been encapsulated. CERN, the European High Energy Physics Lab, choose Cint as front-end command processor and script interpreter for "ROOT" extensive Object Oriented Software Framework. The ROOT/CINT framework will be used in Large Hadron Collider(LHC) simulation and research project, at least, up-to 2004. CINT should have support lifetime beyond 2004.

- More applications
  There are unlimited opportunity of CINT integration. Interfaces with Tcl/Tk, VisualBasic, openGL, DirectModel, VRML integration for 3D, Verilog-XL simulator connected by PLI and TCP/IP, Math library + Digital Filter Design tool, Database connection, Win32 API.
  Your contribution will be greatly appreciated.

- Actively being developed/maintained.
- Recent additions:
  - Partial namespace support.
  - Preliminary exception handling support.
  - Improved template support.
- Seems to be relatively easy to port.
- Author is willing to incorporate changes from uses back into the official distribution.
- No cint mailing lists, but author reads the ROOT lists.
- There is a book with "the thoroughest documentation about CINT". (In Japanese.)

Please do not regard those source files as programming example or implementation reference. I started to write Cint when I wasn't knowledgeble to modern programming methodology. I can write much better version if I can start over.

- Not much documentation, either external or internal.
  - (But at least identifiers are usually meaningful.)
- Code difficult to read.
  - MG code management leaves many old versions in the source, disabled by `#ifdef`'s.
- Code is not robust, especially with string handling.
- Standard techniques not used. Lexing and parsing are completely ad hoc, and not entirely consistent.
  - Pretty much impossible to determine what grammar the program actually accepts.
  - Lousy error recovery.
- Over-simplistic implementations make achieving full C++ compliance difficult.
  - Types represented as strings.
  - Template instantiation done by simple text substitution.
  - Namespaces treated like structures.
- To really implement standard C++ would require a complete rewrite.