GNU Algol 68 Coding Guidelines

For GCC version 17.0.0 (pre-release)

(GCC)

Jose E. Marchesi

Short Contents

Table of Contents

1 Stropping

The GNU Algol 68 compiler supports two stropping regimes:

- The classic *UPPER stropping*, which is one of the standard stropping regimes defined in the Standard Hardware Representation for Algol 68. This regime uses upper-case letters to encode bold letters and lower-case letters to encode non-bold letters.
- The modern *SUPPER stropping*, which is a GNU extension. This is the standard stropping regime in GCC, and its rules are similar to the naming conventions widely used in many modern programming languages. The resulting programs have a very modern feeling.

In GCC we use SUPPER stropping only. The only instance of UPPER stropping are in test cases. Some of the guidelines and considerations in this document may also be useful in programs using UPPER stropping.

2 Formatting

The placement of spaces and empty lines in the program text plays an important role when it comes to readability.

2.1 Empty lines

Empty lines are often used in programs to separate logical parts in a sequence of statements or expressions. This avoids the code to look like walls of text, which are somewhat difficult to read. This of course also applies to Algol 68, but to a much lesser degree due to the exceptionally clean syntax of the language. Therefore we favor a compact formatting to a reasonable extent.

Please be frugal with empty lines, especially within enclosed clauses.

It is not necessary nor advisable to have separated "declaration parts" in serial clauses, because declarations can appear anywhere. However empty lines may still be useful to group related declarations together.

It is often better to use an explanatory comment rather than an empty line, again especially within enclosed clauses.

2.2 Spaces before parentheses

Do not put spaces before open-parentheses in routine calls.

But make sure to always put a space between `union` or `struct` and the open parenthesis that follows in declarers.

Likewise, please put a space before the open parenthesis when the enclosed clause in a cast is a closed clause.

Examples:

```
{ No space before open-parentheses in calls }
puts(fixed(count,0) + "'n");

{ Space between `union' or `struct' and `(' }
mode Number = union (int,long int,real,long real)

{ Space before '(' in casts }
ref JSON_Fld (fields) := field;
```

2.3 Spaces after parentheses

When writing routine texts always place a space between the formal parameters pack and the mode of the value yielded by the routine.

When writing operator and procedure declarators do not put a space between the parameter modes pack and the mode of the yielded value.

Also in declarers, do not put a space after `op` or `proc` and the parameter modes pack.

```
{ Space after formal parameters pack in routine text }
json_foreach_elem(a, (ref JSON_Val v) void: len +:= 1)
```

```
{ No space after parameters pack in procedure and operator
  declarators }
proc(string)void error;
```

2.4 Spaces within packs

With "pack" we refer to the following source constructs which are collections of other constructs enclosed between (and) symbols:

- The actual parameters in a call.
- The formal parameters in a routine text.
- The fields in a struct mode declarator.
- The modes of the united modes in an union mode declarator.
- The modes of the parameters in an operator or procedure declarator.

Spaces are optional after commas in packs when both the preceding and following symbols are tags.

Put a space after commas in packs when the next construct is not a tag, but only if the preceding construct is a tag.

Do not put spaces before commas in packs.

Examples:

```
{ Spaces are optional around commas surrounded by tags }
process(socket, resp, fragmented);
process(socket,resp,fragmented);
op E = (Symbol a,b) bool: a = b;
op E = (Symbol a, b) bool: a = b;

{ Space after commas separating a non tag and a tag }
op E = (Symbol s, Word w) bool: s E w;
mode M = struct (int i, real r);

{ No spaces in commas separating non tags }
proc(int,string,[]real)int callback;
op(int,int)int handler;
mode Data = union (void,bool,int)
```

2.5 Spaces in row displays

Within row displays spaces are optional after commas, but please never put spaces before commas.

```
{ Spaces are ok after commas in row-displays }
[]int a = (1,2,3);
[]int b = (1, 2, 3);
[]string names = ("jemarch",
                  "mnabipoor",
                  "pietr0");
```

2.6 Spaces in formulas

Do not put spaces after monadic operators whose representation is not a bold word.

However, if the monadic operator is represented by a bold word, always put a space between the operator and the operand, even when the operand starts with a parenthesis.

Always put spaces before and after dyadic operators if the formula is not parenthesized. Spaces are optional if the formula is parenthesized, provided the operator is not represented by a bold word. Note however that if long tags are involved extra spaces may be advisable even for parenthesized formulas.

Examples:

```
{ No space after non-bold monadic operators }
int i = -10;

{ Always a space after bold monadic operator }
int i = ABS (base + offset)

{ Spaces in dyadic operator }
total := a + b;
index := cnt +:= 1;
total := (a + b);
index := (cnt +:= 1);
total := (a+b);
index := (cnt+:=1)
```

2.7 Spaces in declarers

Do not put spaces after ']' in row mode declarers.

Do not put spaces after the bounds of a declarer.

Also, do not put spaces directly within the bounds of a declarer, unless for indentation purposes. Since bounds can contain any unit, the general rules apply within these.

Examples:

```
{ No space after `]' in declarers }
[]int a = (1,2,3);

{ No spaces after bounds in declarers }
mode List = [10]int,
     MatrixList = [10][3,3]int,
     Numbers = []union (int,real);

{ No spaces directly within bounds in declarers }
mode MyString = [1:10@]MyChar,
     DynamicTable = [read_int(10, 20),
                     read_int(10, 20)]char;
```

2.8 Spaces in indexers and trimmers

While indexing and trimming a multiple, never put a space between the indexed tertiary and the SUB symbol.

Do not put spaces directly within indexers and trimmers, unless for indentation purposes. As an exception to this rule, you can put a single space before the "at" operator `@` if desired.

Examples:

```
{ No space between tertiary and '[' }
int i = a[i];
int i = a[10:20]

{ No direct spaces within trimmers and indexers, but before  }
[]int a = b[2:5@10];
[]int c = d[10:20 @1];
```

2.9 Spaces in brief clause forms

It is generally a good idea to have spaces around `|` and `|:` within the brief forms of conditional clauses, case clauses and conformity clauses.

When the brief forms are very short and the units are number denotations, it may be more clear to not use spaces, especially when the form is an operand in a formula.

Examples:

```
{ Space around | and |: in brief forms }
(v | (void): "empty", (bool b): (b | "true" | "false"))

{ No spaces may be more readable sometimes }
int n = 2 + (c>3|10|20);
```

3 Comments

Use `"foo"` to refer to formal parameters when documenting procedures or operators.

Use `` `whatever' `` to refer to any other source construct that is not a formal parameter.

Use `{{` and `}}` delimiters for commenting out code. Remember comments in modern Algol 68 are nestable.

Examples:

```
int error_hash = 0;

{ Return a hash code for the string "s", or `error_hash' if the string
  is too long.  }

proc hash_string (string s)

{{ int no_longer_needed; }}
```

4 Syntactic Conventions

4.1 Closed clauses

Algol 68 allows using `(` and `)` instead of `begin` and `end` to delimit closed clauses. In fact, parenthesized expression in other programming languages are realized in Algol 68 with closed clauses, in a very orthogonal way. Both forms are useful and can generally be used according to the programmer's taste. However, this section contains a few guidelines and recommendations on this regard.

Use parentheses for closed clauses that span a single line, regardless of the context. Having `begin` and `end` symbols in the same line looks weird and confusing.

As a general rule, always use parentheses in closed clauses that are operands in a formula. Exceptionally, using `begin` and `end` in formula operands may be preferable if the operand contains many declarations and units, and only if it spans for more than one line. In this case, however, please consider factoring the code in the operand into a routine and replace it with a procedure call.

The preferred indentation for a closed clause whose contents span more than one line, and that uses `begin` and `end` symbols as delimiters, is to indent the contents right at the right of the `begin` symbol. The `end` symbol shall then be placed in its own line, with the same indentation level than the opening symbol.

If the closed clause contains empty lines, or if the line preceding the closed clause is so long that it would "hide" the first line in the closed clause, then it is ok to put the first unit or declaration in the line after `begin`. This usually happens when the closed clause is the body of a long routine text.

The preferred indentation for a closed clause whose contents span more than one line, and that uses `(` and `)` symbols as delimiters, is to indent the contents right at the right of the `(` symbol. The `)` symbol finishing the closed clause shall not be placed in its own line.

```
{ No `begin' and `end' in the same line }
int i = 2 + (3+4);
int i = 2 + (int i = random(); i % 10 );
bool test = case v in (string): (puts (s); true) out false esac;

{ Closed clauses as formula operands }
int i = 2 + (int cnt := 0;
             to UPB data[@1] do cnt +:= 1 od;
             cnt)
int j = 2 + begin int cnt := 0;
                  to UPB data[@1] do cnt +:= 1 od;
                  cnt
                  end

{ Closed clause with no empty lines }
begin int fd = fopen ("data', file_o_rdonly")
      puts ("first line: " + fgets (fd, 0));
      fclose (fd)
```

```
end;

{ Closed clause with no empty lines but with preceding
  line of similar length }
proc parse_number = int:
begin
      int num := 0;
      while num := num * 10 + ABS ch - ABS "0";
            isdigi(getachar)
      do ~ od;
      ungeachar(ch);
      num
end;

{ Closed clause with empty lines }
proc main_proc = int:
begin
      { Auxiliary procs }
      proc aux1 = int: ...;
      proc aux2 = int: ...;

      { Computation }
      aux1;
      aux2;

      { Result }
      aux1 + aux2
end

{ Indentation of closed clauses using `(' and `)' delimiters }
proc parse_number = int:
(int num := 0;
 while num := num * 10 + ABS ch - ABS "0";
       isdigi(getachar)
 do ~ od;
 ungeachar(ch);
 num)
```

4.2 Indexers and trimmers

Algol 68 allows using `(` and `)` instead of `[` and `]` in bounds and slices to represent the SUB and BUS symbols. This is supported by ga68 via the `-fbrackets` command-line option in order to ease the porting of old code, and it is disabled by default. Please always use square brackets for indexing in new code.

4.3 Conditional clauses

If a conditional clause is still clear and not of excessive length when written on a single line, just do it.

Start the enquiry clause in the if-part of a conditional clause right after the `if` symbol, not in the next line.

The serial clauses in the then- and else-parts of the conditional clause shall be indented five positions right, which is the length of both the `then` and `else` symbols plus one.

The first declaration or unit in the then- and else-parts shall be placed in the same line than the `then` and `else` symbols, respectively.

Place the `fi` closing symbol in its own line, with the same indentation level than the matching `if`. The exception to this rule is when the conditional clause has no else-part and the then-part spans for a single line that is not too long. In that case, place the `fi` in the same line than `then`.

Examples:

```
{ Very small conditional clause in a single line }
if idx < 0 then fatal("invalid idx") fi

{ Short conditional-clause with `fi' in the same line
  than `then' }
if argc /= 3
then puts("expected two arguments'n") fi

{ A conditional-clause that spans several lines }
if a > 10
then puts("truncating");
     a := 10
fi
```

4.4 Loop clauses

If a loop clause is small enough to fit in a single line without occupying most of it, just do it.

If a loop clause spans two lines, and the second line is not too long, you can put `od` in the same line than `do`.

If a loop clause spans several lines, please put the `do` symbol in its own line, indented to the same level than the clause's frobyts.

Examples:

```
{ Very short loop-clause in a single line.  }
for a to argc do puts ("arg: " + argv[a]) od;

{ Short loop-clause with `od' in the same line than `do' }
for i from LWB a to UPB a
do total +:= a[i] od
```

```
{ A loop-clause that spans several lines }
while NOT exit
do string cmd = get_command;
   process_command(cmd)
od
```

4.5 Case and conformity clauses

Do not write a case or conformity clause in a single line, unless you are using the brief form. Unlike conditional and loop clauses, these are difficult to read.

Please put the `in`, `out` and `ease` symbols in their own lines, with the same indentation level than the matching `case`.

Start the choices right after the `in` symbol, in the same line. All the choices may fit in a single line. If they don't, please put each choice in its own line.

```
{ Short case clause }
case i
in 100, 200, 300 out 0 esac;

{ Long case clause }
case i
in 100,
   200,
   300
ouse i % 100
in 100,
   200,
   300
esac;

{ Long conformity clause }
case v
in (void): "empty",
   (bool b): (b | "true" | "false"),
   (string s): s
esac
```

4.6 Procedure and operator declarations

In procedure and operator declarations, if the body of a routine text starts with 'begin', put it at the same indentation than the `pub`, `proc` or `op`. Otherwise, indent it three spaces to the right relative to the `pub`, `proc` or `op`.

Examples:

```
{ Body of routine is a `begin'..`end' closed clause }
proc checked_div = (int a,b) int:
begin
      if b = 0 then fatal ("div by zero") fi;
```

```
      a % b
end;

{ Body of routine does not start with `begin' }
proc checked_div = (int a,b) int:
   (b = 0 | fatal ("div by zero"); skip | a % b);

{ Body of routine is not a closed clause }
proc checked_div = (int a,b) int:
   if b = 0
   then fatal ("div by zero"); skip
   else a % b
   fi;
```

4.7 Contracted declarations

Please don't be shy to use contracted forms of declarations. They can make the program much more readable and they make it easier to add new declarations, because they prevent writing the same text again and again.

However, care should be taken when declaring operators and procedures. In these cases, contracted declarations should only be used when declaring very short, one or two lines long routines. The last routine in the list of joined declarations can be a bit longer.

Examples:

```
{ Contracted declarations lead to compact and very
  readable code }
int disconnected = 0, connected = 0, unknown = 2;
pub ref JSON_Val json_no_val = nil,
    ref JSON_Elm json_no_elm = nil;
    ref JSON_Fld json_no_fld = nil;

{ Use contracted declarations for short routines }
op + = (States ss, State s) States: MoreStates (heap States := ss, s),
   + = (Transitions ts, Transition t) Transitions:
         MoreTransitions (heap Transitions := ts, t)
```

4.8 Brief clause forms

The obvious context where to use the brief forms of conditional, case and conformity clauses is when these clauses appear as operands in formulas. They match well with parenthesized closed clauses.

It is also ok to use brief forms of clauses out of formulas, especially inside case and conformiy clauses. But please be careful, as brief forms may be confused sometimes.

Examples:

```
{ Brief forms in formulas }
int res = (a=0 | fatal("div by zero"); skip | den/a);
```

```
{ Brief forms out of formulas }
for i to ELEMS str
do char newline = REPR 10, tab = REPR 9, c = str[i];
   (c = "\" | res +:= "\\"
    |: c = newline | res +:= "\n"
    |: c = tab | res +:= "\t"
    |  res +:= c)
od
```

5 Naming

Unlike most other programming languages, which are not stropped, in Algol 68 it is possible to have tags with the same name as reserved words, by appending an underscore character to the tag. For example, a tag `if` can be written as `if_`. It is important to note that the trailing underscore is not part of the tag: it is just a stropping artifact. This is always better than contriving artificious synonyms that are often confusing or too long. A copying routine has arguments "from" and "to"? Call them `from_` and `to_`. A struct mode has fields "in" and "out"? Call them `in_` and `out_`.

Please use fully upper-case bold words for operator indicants. This makes it easier for text editors to highlight them in a different style than mode indicants, and look more symmetrical in case of dyadic operators. For example, use `ABS` and not `Abs`.

6 Programming Style

This section contains some recommendation on the usage of the facilities of the language.

6.1 Writing routines

Use routines liberally! Routines are cheap, very easy to write thanks to the excellent Algol 68 syntax for routine texts, and first-class citizens in the language. They also have access to the lexical environment. So if you find yourself wanting to write a macro in order to repeat some little calculation, just write a small procedure or operator instead.

Choose identifiers that are expressive of meaning in order to clarify both the intent of the procedure or operator and the code written that uses it.

Consider using overloaded operators in preference to procedures with united mode parameters to encourage users of the operator to create new versions of the operator for different parameter types, rather than leaving the users trying to figure out how to hack the united mode definition.

The high level of orthogonaliy of Algol 68 combined with the structural type equivalence and the nice compact syntax of declarers makes mode names way less relevant than in many other programming languages. In particular if a routine takes a parameter that is an united mode, and that particular united mode is either not used anywhere else or very short, just write the declarer, you don't have to name it first.

Please make good use of the lexical block structure of the programming language: is is there to be used. In little local auxiliary routines, do not add arguments just to pass a value that is in the environment; rather, place the declaration of the auxiliary routine near the declarations of the values it accesses. If this is not possible, then it may be wise to judiciously pass those non-nearby values as arguments in order that the programmer be aware that non-nearby values are being accessed or possibly altered.

6.2 Enquiry clauses

Enquiry clauses are not just "boolean expressions" with a fancy name. They can be thought as a sort of specialized serial clauses, and as such they can contain any number of declarations and units while being required to always yield a value of mode `bool`.

Avoid polluting the lexical environment with declarations that are intended to be used exclusively within conditional and loop clauses: use a declaration in the enquiry-clause of the clause instead. This also makes it easier to later refactor code.

Examples:

```
{ The range of `c' is the entire loop clause. }
while char c = getchar;
      c /= char_eof
do
   ... use c ...
od
```

6.3 Nihils

Never use `nil` directly in identity relations; it is very error prone. It is better to define *nihils* for all reference modes that are likely to appear in one.

Examples:

```
mode Node = ...;
ref Node no_node = nil;
while n :/=: no_node do ... od
```

GNU Free Documentation License

Version 1.3, 3 November 2008

0. PREAMBLE

   The purpose of this License is to make a manual, textbook, or other functional and useful document *free* in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

   This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

   We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

   This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

   A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

   A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

   The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

The "publisher" means any person or entity that distributes copies of the Document to the public.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
D. Preserve all the copyright notices of the Document.
E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
H. Include an unaltered copy of this License.
I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.
N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.
O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

   You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

   The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

   In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements."

6. COLLECTIONS OF DOCUMENTS

   You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

   You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

   A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

   If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

   Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

   If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

   You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

   However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

   Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

   Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE

    The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See `https://www.gnu.org/copyleft/`.

    Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

    "Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

    "CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

    "Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.

    An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

    The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

```
Copyright (C)  year  your name.
Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.3
or any later version published by the Free Software Foundation;
with no Invariant Sections, no Front-Cover Texts, and no Back-Cover
Texts.  A copy of the license is included in the section entitled ``GNU
Free Documentation License''.
```

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with...Texts." line with this:

```
with the Invariant Sections being list their titles, with
the Front-Cover Texts being list, and with the Back-Cover Texts
being list.
```

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

Index