hasmd 1.0 Specification:

1. Separation characters
   1. Lines are separated by the ‘;’ character
   2. Tokens are separated by the following characters: NEW LINE – (U+000A), CARRIGE RETURN – (U+000D), HORIZONTAL TAB – (U+0009), SPACE – (U+0020), NO–BREAK SPACE – (U+00A0), EN QUAD – (U+2000), EM QUAD – (U+2001), EN SPACE (nut) – (U+2002), EM SPACE (mutton) – (U+2003), THREE–PER–EM SPACE (thick space) – (U+2004), FOUR–PER–EM SPACE (mid space) – (U+2005), SIX–PER–EM SPACE – (U+2006), FIGURE SPACE – (U+2007), PUNCTUATION SPACE – (U+2008), THIN SPACE – (U+2009), HAIR SPACE – (U+200A), ZERO WIDTH SPACE – (U+200B), NARROW NO–BREAK SPACE – (U+202F), MEDIUM MATHEMATICAL SPACE – (U+205F), IDEOGRAPHIC SPACE – (U+3000), ZERO WIDTH NO-BREAK SPACE – (U+FEFF)
      1. In this document, they will all be separated by the SPACE character
2. Name prefixes:
   1. All variable, custom type, method and label identifiers must fit the regex ‘([a-z]|[A-Z]|[0-9])+’
   2. All library identifiers must fit the regex ‘([a-z]|[A-Z]|\_)(([a-z]|[A-Z]|\_|[0-9])\*\.)\*([a-z]|[A-Z]|\_|[0-9])+|([a-z]|[A-Z]|\_)’
   3. All file identifiers must fit the regex ‘(#)?([a-z]|[A-Z]|\_|[0-9])+:(([a-z]|[A-Z]|\_|[0-9])+\\)\*([a-z]|[A-Z]|\_|[0-9])+(\.([a-z]|[A-Z]|\_|[0-9])+)?’
      1. The hashtag means it is unchangeable (base hasmd file, almost like a library but requires possibly less permissions)
   4. Types are defined in 4
   5. Modifiers are listed in 6
3. Naming rules:
   1. All variable **full name**s are in the format ‘variable\_name’ where ‘name’ is the identifier
   2. All type **full name**s are in the format ‘type\_name’ where ‘name’ is the identifier
   3. All custom type **full name**s are in the format ‘type\_custom\_name’ where ‘name’ is the identifier
   4. All method **full name**s are in the format ‘method\_name’ where ‘name’ is the identifier
   5. All label **full name**s are in the format ‘label\_name’ where ‘name’ is the identifier
   6. All library **full name**s are in the format ‘library\_name’ where ‘name’ is the identifier
   7. All file **full name**s are in the format ‘file\_name”’ where ‘name’ is the identifier
   8. All operator **full name**s are in the format ‘operator\_name’ where ‘name’ is the identifier
   9. All modifier **full name**s are in the format ‘modifier\_name’ where ‘name’ is the identifier
   10. All const **full name**s are in the format ‘const\_name’ where ‘name’ is the identifier
4. Types:
   1. In this document, type = [modifier] type
   2. Numbers
      1. Whole numbers
         1. Integers
            1. In these types the number after the ‘`’ character represents the amount of bits used to store the information
            2. ints:

Value fits regex ‘(-)?[1-9][0-9]\*|0’ and is not bigger than max or smaller than min

Default is 0

type\_int`2

Possible value range (integer, inclusive): -2 to 1

type\_int`4

Possible value range (integer, inclusive): -8 to 7

type\_int`8

Possible value range (integer, inclusive): -128 to 127

type\_int`16

Possible value range (integer, inclusive): -32768 to 32767

type\_int`32

Possible value range (integer, inclusive): -2147483648 to 2147483647

type\_int`64

Possible value range (integer, inclusive): -9223372036854775808 to 9223372036854775807

* + - * 1. units:

Value fits regex ‘?[1-9][0-9]\*|0’ and is not bigger than max or smaller than min

Default is 0

type\_uint`2

Possible value range (integer, inclusive): 0 to 3

type\_uint`4

Possible value range (integer, inclusive): 0 to 15

type\_uint`8

Possible value range (integer, inclusive): 0 to 255

type\_uint`16

Possible value range (integer, inclusive): 0 to 65535

type\_uint`32

Possible value range (integer, inclusive): 0 to 4294967295

type\_uint`64

Possible value range (integer, inclusive): 0 to 18446744073709551615

* + - * 1. type\_int`4, type\_int`8, type\_int`16, type\_int`32, type\_int`64, type\_uint`4, type\_uint`8, type\_uint`16, type\_uint`32, type\_uint`64 can be set to a value of 0x then the hex code with the correct length (bits / 4)
      1. type\_number
         1. A number uses the .NET type System.Numerics.BigInteger in the System.Numerics library and thus has an infinite range (depending on memory available), but you cannot assign it to either inf or -inf because that would require infinite memory
         2. Default is 0
         3. Value fits regex ‘(-)?[1-9][0-9]\*|0’
    1. Decimal numbers
       1. type\_decimal
          1. It is basically a fraction using the .NET type hamarb123.Base.BigRational in the hamarb123.Base library and it cannot be inf or -inf
          2. Default is 0
          3. Value fits regex ‘((-)?[1-9][0-9]\*|0)(/[1-9][0-9]\*)?’
       2. type\_double
          1. Based off the .NET type ‘double’
          2. Default is 0
          3. Value fits regex ‘(\+|-|)([1-9][0-9]\*(\.)?|0\.|(\.)?)([1-9][0-9]\*)+([eE](\+|-|)([1-9][0-9]\*|0)|)’
    2. type\_bool
       1. Based off the .NET type ‘bool’
       2. Default is false
       3. Value fits regex ‘(0|1|[tT]rue|[fF]alse)’
       4. Two states:
          1. true (True, 1)
          2. false (False, 0)
  1. type\_string
     1. Based off the .NET type ‘string’
     2. Default state is “”
     3. Value starts and ends with quotation marks (“)
     4. The actual string must use the following escape sequences because (“) ends the string and (\) escapes the character
        1. \" – double quote
        2. \\ – backslash
        3. \0 – Unicode character 0
        4. \a – Alert (character 7)
        5. \b – Backspace (character 8)
        6. \f – Form feed (character 12)
        7. \n – New line (character 10)
        8. \r – Carriage return (character 13)
        9. \t – Horizontal tab (character 9)
        10. \v – Vertical quote (character 11)
        11. \uxxxx – Unicode escape sequence for character with hex value xxxx
     5. Only those characters are allowed after the (\)
  2. type\_pointer<T> where T is a type
     1. Based off the .NET type hamarb123.Base.Container<T> in the hamarb123.Base library
     2. Default state is new type\_pointer<T>() with the ‘value’ set to the default of T
     3. Contains the methods void SetValue(T) and T GetValue()
  3. type\_nullable<T> where T is a type that is not type\_nullable
     1. Based off of the .NET type ‘Nullable<T>’
     2. Default state is null
     3. Allowed values are: null, allowed values for T
     4. Contains the methods void SetNull() and void SetValue(T) and T GetValue() {will return default if IsNull() returns true} and bool IsNull()
  4. type\_readonly<T> where T is a type
     1. Default state sets the value to the default of T
     2. Contains the method T GetValue()
  5. type\_writeonly<T> where T is a type
     1. Contains the method void SetValue(T)
  6. type\_exception
     1. Default: message = “”, ID = 0, extrainfo=default(object)
     2. Contains the methods string GetExceptionMessage() and number GetID() {IDs in 5} and object GetExtraInfo() {associated with IDs in 5} and number GetX() and number GetY()
  7. type\_array<T> where T is type
     1. Based off the .NET type T[]
     2. Default: 0 length
     3. Contains the methods void SetValue(type\_int`64, T) and T GetValue(type\_int`64) and type\_int`64 GetLength() and int`64 IndexOf(T) and type\_list<T> ToList() and type\_array<T> GetRange(type\_int`64, type\_int`64 {length}) and void SetRange(type\_int`64, type\_array<T> or type\_list<T>
  8. type\_list<T> where T is type
     1. Based off the .NET type List<T>
     2. Default: 0 length
     3. Contains the methods void SetValue(type\_int`64, T) and T GetValue(type\_int`64) and type\_int`64 GetLength() and int`64 IndexOf(T) and type\_array<T> ToArray() and type\_array<T> GetRange(type\_int`64, type\_int`64 {length}) and void SetRange(type\_int`64, type\_array<T> or type\_list<T>) and void SetLength(type\_int`64) {Will shrink or expand, will delete and add elements to and from the end, if some are added they will be default(T)) and void Insert(int`64, T) and void InsertRange(type\_int`64, type\_array<T> or type\_list<T>) and void RemoveAt(int`64) and void RemoveRange(int`64, int`64 {length}) and void Remove(T) and void Clear()
  9. type\_object
     1. ALL TYPES INHERIT THIS TYPE
     2. Default: default (empty), all types can be set to this

1. Exception list
   1. 1 – Access denied
   2. 2 – Variable does not exist
   3. 3 – Label does not exist
   4. 4 – Library access is denied / does not exist
   5. 5 – Method does not exist
   6. 6 – File does not exist
   7. 7 – Invalid argument type / type argument
   8. 8 – Other error from library
   9. 9 – Wrong overall type in a command (ex. set method\_a 3)
   10. 10 – No such constant
   11. 11 – Compilation error
   12. 12 – Lexical error
2. Modifier list, they can only be applied once/target
   1. modifier\_private – Can be used on methods (unseeable to other files but can still be used like a normal method otherwise) and file-wide variables
   2. modifier\_saved – Gets saved by the program running this script
   3. modifier\_upgrading<method\_void> – Used to upgrade an obsolete variable, you will not be able to access it if it has this modifier, the void takes the variable as the input and does what it needs to from there
3. Free libraries
   1. Free libraries will be given only if the file contains the requirelibrary command for that library
   2. The removealllibrarypermissions command will not remove the permission to these libraries, so you do not need to worry about running the addlibrary permission command for these libraries
4. Commands
   1. File-wide commands
      1. The first line must be ‘header hasmd 1.0;’ to specify the language and the language version, NOTE: the whitespace in this line must be exactly 1 SPACE character
      2. The second set of lines are in the format ‘requirelibrary library\_name;’ where ‘name’ is the library’s name
      3. The third set of lines are in the format ‘usingtype value1 value2 value3;’ where ‘value1’ is the file/library that the type is in, ‘value2’ is the type’s actual name and ‘value3’ is the new name
         1. This is not necessary for any types in (4)
         2. value2 and value3 are in the format ‘type\_custom\_identifier’ where ‘identifier’ is the identifier
      4. The fourth set of lines are in the format ‘struct type\_custom\_name(variables methods) [initialization function that’s local {argument to initialization function is the variable}];’ where:
         1. variables = ‘declare type name [; variables]’
         2. methods = ‘method\_nameinstruct method\_localname [; methods]’ where the first argument for method\_localname is the struct type and the rest of the arguments (if any) will be applied to method\_nameinstruct
      5. The fifth set of lines are in the format ‘const type const\_identifier value;’ where ‘identifier’ is the identifier
      6. The sixth set of lines are in the format ‘declare type variable\_identifier [value];’ where ‘identifier’ is the identifier
      7. After this, it is method commands
   2. Method commands
      1. The first line in a method command declares the method: ‘method type [method modifiers] method\_identifier [params]’ where ‘identifier’ is the identifier and:
         1. params = ‘type variable\_identifier [params]’ where ‘identifier’ is the identifier
      2. The lines inside of a method are Internal commands
      3. The method ends when either a new method starts, or the end of the file is reached
   3. Internal commands
      1. These commands are the commands inside the method
      2. List of commands (all end with ‘;’)
         1. declare type\_t variable\_v [value]
         2. set variable\_1 argument | argument is either variable\_2 or value
         3. swap variable\_1 variable \_2
         4. run variable\_v | variable\_v is a string, it can be multiple lines, it just has to end each line with a ‘;’
         5. convert variable\_1 type\_new variable\_2
         6. aliaslocalvariable variable\_local variable\_inmethod
         7. label label\_name
         8. goto label\_name
         9. call [variable\_out] argument method\_name [params] | argument is either library\_x, file\_x, variable\_x or local | params = variable\_forparameter [params]
         10. setreturn variable\_name
         11. clone variable\_in variable\_out
         12. deepclone variable\_in variable\_out
         13. getexception variable\_out
         14. unexception
         15. gettype variable\_in variable\_out
         16. isnull variable\_in variable\_boolout
         17. isnullable variable\_in variable\_boolout
         18. conditionalcall variable\_inbool label\_iftrue [label\_iffalse]
         19. getconstant argument const\_name variable\_out | argument is either library\_x, file\_x or local
         20. getstructvalue variable\_structin variable\_nameinstruct variable\_out
         21. setstructvalue variable\_structin variable\_nameinstruct variable\_in
         22. Library permission modifiers:
             1. removealllibrarypermissions | removes all library permissions until either the method ends or the resetlibrarypermissions command is called
             2. addlibrarypermission library\_lib | adds the specified permission back, if it didn’t have permission in the first place it will throw an exception {5}
             3. resetlibrarypermissions | resets all the library permissions
5. Comments: comments will be enclosed like this /\*comment\*/
6. Example code:

header hasmd 1.0;

requirelibrary library\_hamarb123.Base;

requirelibrary library\_Console;

usingtype library\_hamarb123.Base type\_custom\_PointD type\_custom\_pointd;

struct type\_custom\_2pointds(declare type\_custom\_pointd variable\_pointd1; declare type\_custom\_pointd variable\_pointd1; method method\_setboth method\_2pointdssetboth;) method\_initialize2pointds