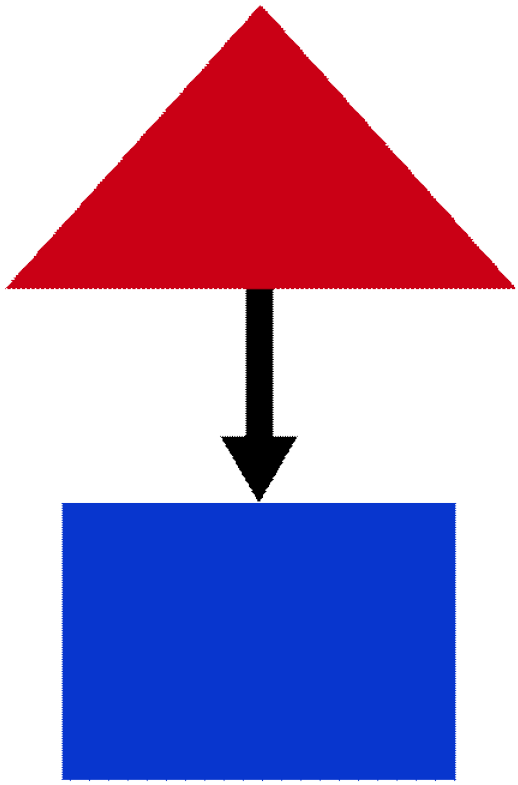


Suum Manual

ver.0.25

Author:Mandru

Table of Contents

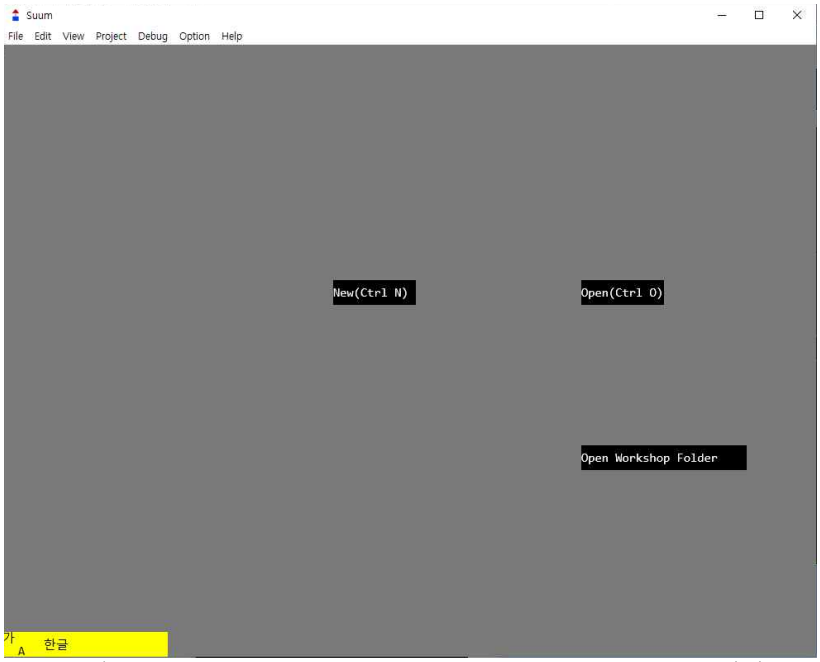
1.What is Suum?
2.Start Screen
3.Class Selection
4.Entry
5.Input and Output
6.Basic data type
7.Variable
8.Syntax and Operator
9.Unary Operator
10.Array
11.Condition
12.Simple Loop
13.Loop
14.Function
15.Pointer
16.Allocator
17.Reference
18.Mixed Refer
19.Class
20.Constructor
21.Self-Send and Refer Function
22.Inheritance and Override
23.Accessor
24.Generic
25.Exception
26.High Order Function
27.Multi Inheritance
28.Type Cast
29.Rendering
30.Input
31.Audio

What is Suum?

Suum is a visual programming language using a flowchart. Visual programming is to develop programs using a visual interface rather than text. It’s easy to read and can only write phrases set on the interface, grammar errors are not common and can be quickly discovered and resolved even if they occur.

However, existing visual programming languages have never been used as mainstream programming languages because of one fatal problem. The problem is that mouse input is forced. You should move your hands between the keyboard and the mouse because entering variable names or numbers eventually requires the use of a keyboard. Also It's cumbersome to drag syntax from interface. To solve this issue, Suum language is designed to allow the creation of syntax using shortcuts, allowing the development of programs with only a keyboard without a mouse. This allows for quick and easy development like a text-based programming language.
In addition, Suum is a programming language with various advantages, such as improving the grammar structure more efficiently and minimizing the possibility of errors that may occur during program execution.

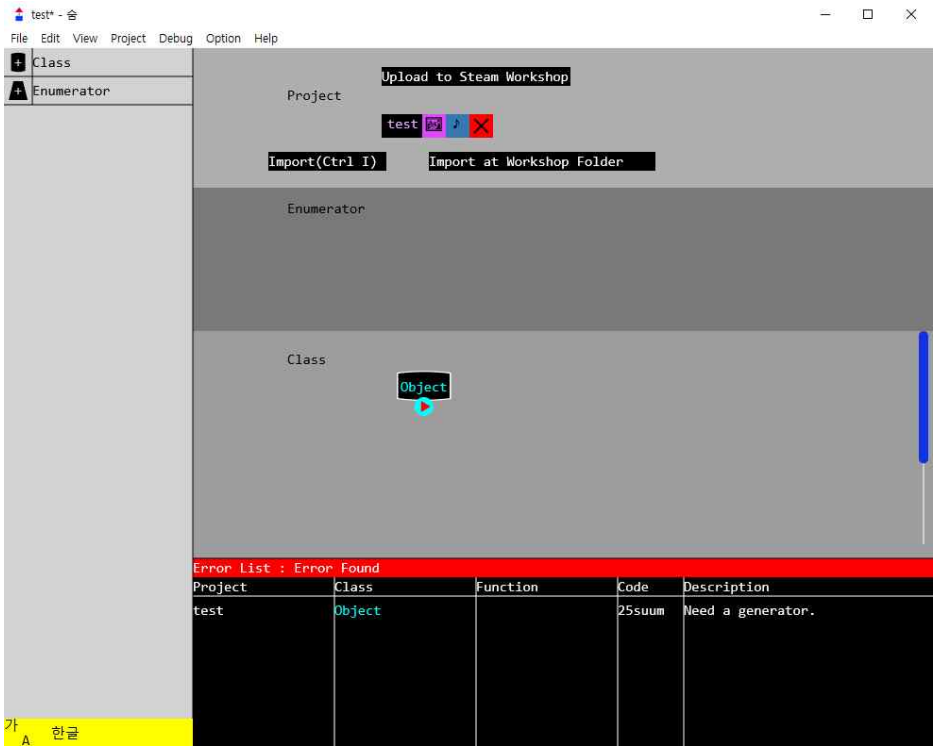
Start Screen



As you run the program, you get a screen like picture. You can change the language settings between Korean and English by pressing the bottom-left button.

You can create a new project by entering a new or Ctrl N shortcut. You can open an existing project file by entering the Open or Ctrl O shortcut, and you can open the folder where the workshop project is located by clicking the Open Workshop Folder button.

Class Selection



The class selection screen appears when you create or open a project file.

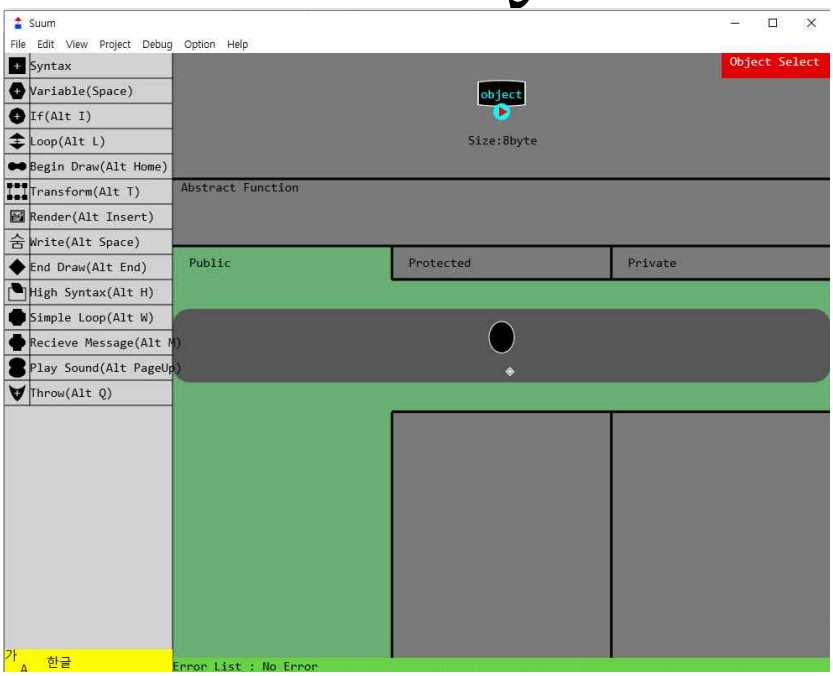
①:This is an interface that displays what you can do at the moment. If you go to a different screen or select a specific target, a different interface will appear.

②: List of projects currently open. You can include other projects with the import button or with the Ctrl I shortcut.

③:List of classes. You can create classes by pressing the Class button in ① or by typing. You can open the object by clicking it once more in the selected state or by entering Enter. Objects in other files cannot be modified.

④:List of syntax errors. Once all errors have been resolved, the program can be executed by pressing the F5 key.

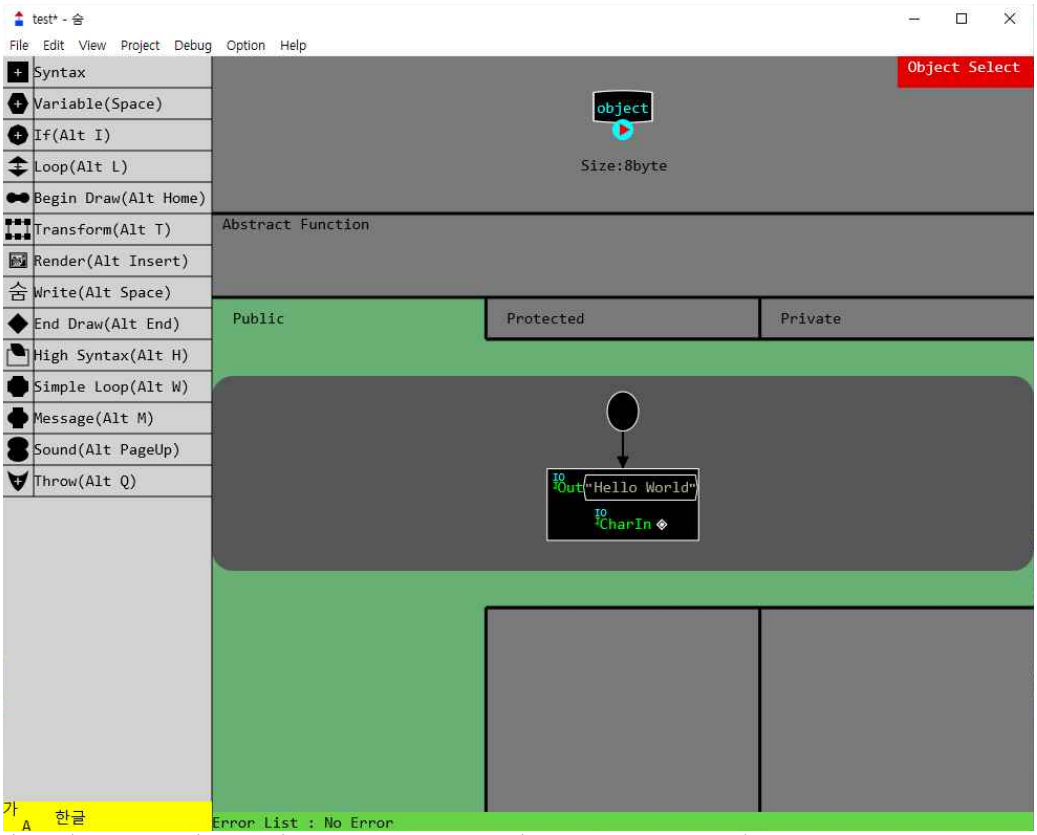
Entry



With the class created and selected, enter the Ctrl E shortcut to display the symbol, which means that it is an entry point class that is created when the program runs.

You can create a function by entering the Space key in one of public, protected, or private areas. Empty the content to make the function a constructor. The constructor of the entry point class is called at the start of the program.

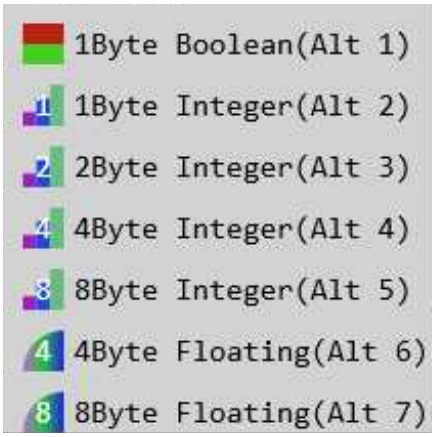
Input and Output



Click the Ctrl I shortcut or the Import button.Open the Library folder, open the English folder, and then select the InOut_eng file.

Press the bottom button 3 times in the function to proceed to the function content. Then enter the ‘IO’ key and enter ; key. Then write ‘Out’, type the space bar, type the " key, write the sentence you want to output, type the right arrow key, and press the Enter key to deselect it. Press F5 to run it.

Basic Data Type

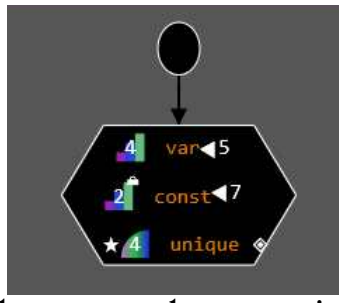


You can apply the basic data type to variables or function’ s return type. You can use shortcuts from Alt 1 to 7, or you can select the default data type from the left interface.

If you enter Alt O, the ⊙ is entered, which means true, with a value of 1.

If you enter Alt X, the ⊗ is entered, which means that it is false, and the value is 0.

Variable



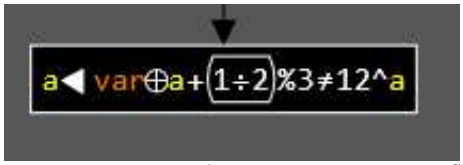
Variables include the member variables that compose the class, the parameters that are entered into the function, and the local variables that are used inside the function. You can create a local variable with a space bar. Enter the data type for the variable on the left, and enter a name on the right. When you enter the : key, the ◀ shape appears, and the value you enter is initialized to the value you enter. If you do not initialize it, it is defaulted to 0.

Except for parameters, you can make a variable static with an Alt U key. A static variable is a variable that has only one object and can be accessed independently of the created object. The order of initialization of a static variable that is member variable starts with the class on the left, followed by a preorder traverse.

If you enter Alt C, the lock will appear and you can specify a constant that cannot be changed since initialization.

Enter the Enter key to select variables or syntax and delete them with the Delete key.

Syntax and Operator



Entering any character in the content of the function creates a syntax. Write down the variables in the execution statement and enter the : key to give the ◀ shape and substitute the value to the right for the variable to the left. You can use the operator in the syntax or initialization. Enter the (key to create a bracket, first perform the operation in bracket, followed by the operation according to priority.

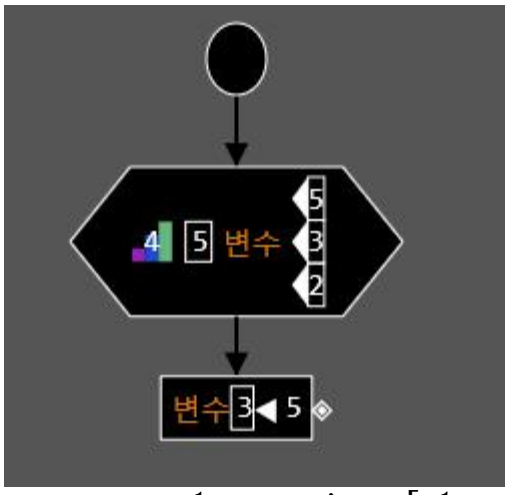
Shape	Input	Priority	Function
^	^	0	Exponent
√	\	0	Root
*	*	1	Multiple
÷	/	1	Divide
%	%	1	Modulo
+	+	2	Sum
-	-	2	Subtract
≪	<<	3	Left Shift
≫	>>	3	Right Shift
⊓	&&	4	Bit And
⊔	||	5	Bit Or
⊙	$$	6	Bit Equivalent
⊕	@@	6	Bit Exclusive Or
=	=	7	Same
≠	!	7	Different
≤	<=	7	Right Bigger or Same
≥	>=	7	Left Bigger or Same
<	<	7	Right Bigger
>	>	7	Left Bigger
⩓	&	8	And
⩔	|	9	Or
≡	$	10	Equivalent
≠	@	10	Exclusive Or

Unary Operator

operator that applies when the identifier is selected and input

Input	Function	Shape
Alt -	Minus	-
~	Bit invert	
`	Not	
,	Logic Cast	

Array

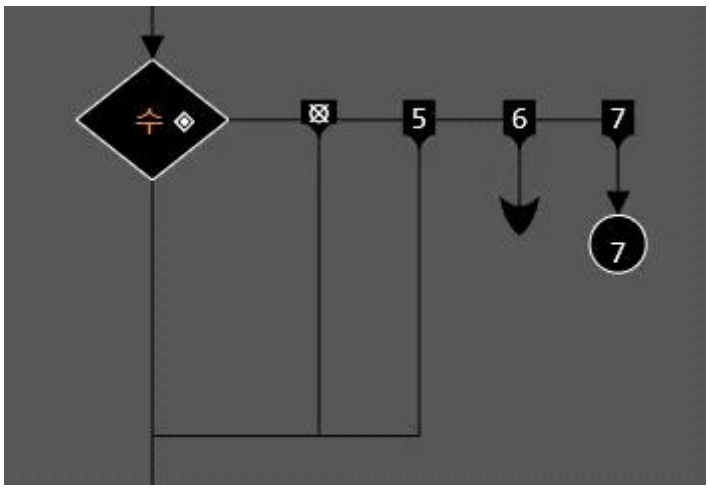


You can make an array by typing [key in a variable. Enter the size of the array in the box. When you initialize an array, you can initialize each element of the array by pressing : key and ESC to move between initializations with the arrow keys closed. If fewer elements are initialized than the array size, the remaining array elements are initialized to the last value.

When in use, you can write down the name of the variable and enter the array index by entering [key. Returns 0 if you add a negative number to the array index or an index larger than the size of the array.

Press # key to get size of an array.

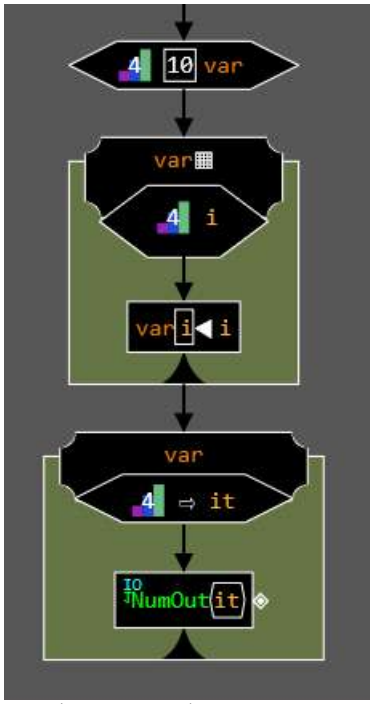
Condition



Generated by the Alt I key. By default, there is a conditional expression and a branch with and a ⊗ on the right. If the value of the branch matches the value in the conditional expression, it proceeds to the branch, and if no branch matches the value in the conditional expression, it proceeds to the below. So by default, it proceeds to the right when the conditional expression is false and downward when it is true. You can move between branches or down to enter the content with the arrow key.

You can create a new branch by entering the space bar from the conditional expression or branch side. You can end the function by entering the Alt Enter shortcut key inside the conditional statement.

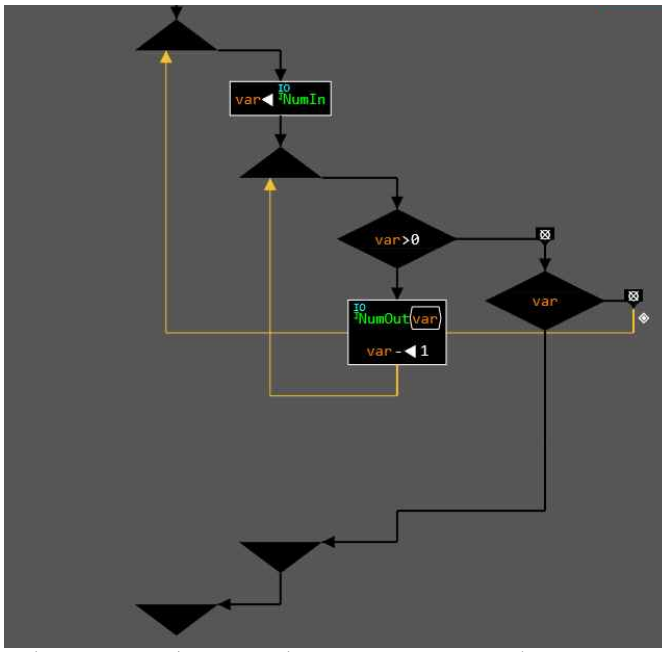
Simple Loop



Generates with the Alt W shortcut. Enter a number in the syntax at the top, and the variable below is inference as an integer and the variable is added 1 by 1 from 0 until the number comes out. You can initialize the variable using : key.

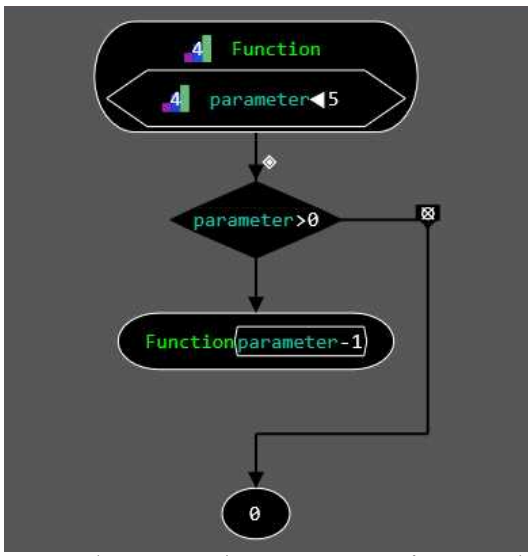
When you enter an array in the syntax at the top, the variables below inference the pointer of the array element and iterate the array to substitute the pointer of each element into the variable.

Loop



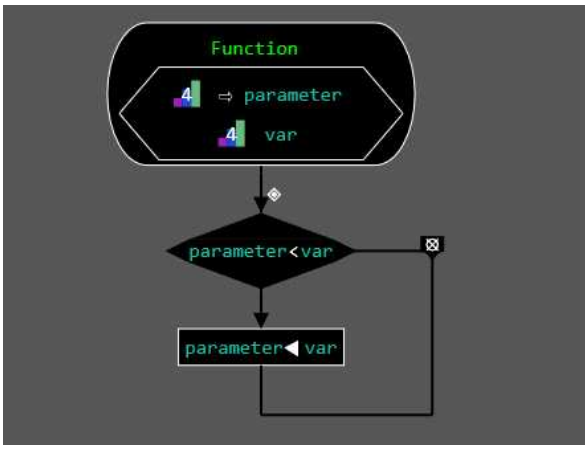
Generates with an Alt L shortcut. It does not function as a loop by itself, but it connects up when you press Alt ↑ in the inner conditional statement, and down when you type Alt ↓. You can connect with a deeper loop with an Alt ← key, or you can disconnect with an Alt → key. When you go to the conditional branch that connects with the loop, you go to the linked loop. When you use it to connect with the top, it runs again from the top, and when you connect with the bottom, it ignore the remaining syntax and go down the loop.

Function



Create with space bar. When it isn’ t selected, only function definitions and parameters are visible, but when selected, function definitions are at the top and the contents are visible. Function definitions can have a return type and a name, such as variables.
In the function definition, you can enter the parameter area by pressing the down arrow key. You can create a parameter by entering the key in the parameter area or by entering the default data type.
Once you have defined a return type, you will see a return syntax at the bottom, and you will also see a return syntax where you end a function in a conditional statement. You can get the value you returned in the return syntax where you used the function.
You can use a function in a syntax, and you can use a function that have a return type instead of a value. You can use a function and then press the space bar to write parameters in parentheses that appear.
If you did not enter any parameters where the function is used, initialize the parameters in the function definition.

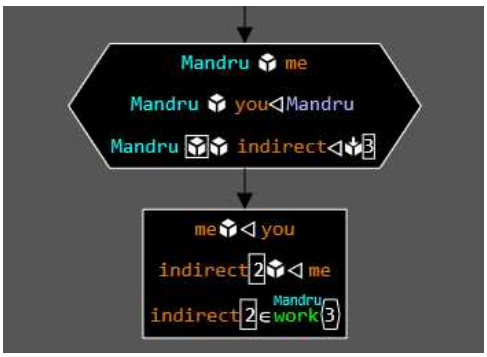
Pointer



Create it by pressing the Alt P shortcut in parameters or local variables. It is a refer type that points to another variable, and modifying the value changes the value of the variable you point to. You cannot point to a variable other than the one you are pointing to. Refer types can be used to point to an array with a [key, return 0 when there is no pointing variable, enter an Alt P shortcut to return 1 when there is a pointing variable, and return 0 when there is no pointing variable.

For refer types, except for the pointer, you can enter an Alt C shortcut to make it a constant so you can't modify it except for initialization, and if you just use it, you can use the values you point to. If you want to refer it to another variable, you can modify it by using the Alt R shortcut in the substitute. You can enter a ⊗ with the Alt X shortcut to empty it. You can set it refer other refer types using multiple times.

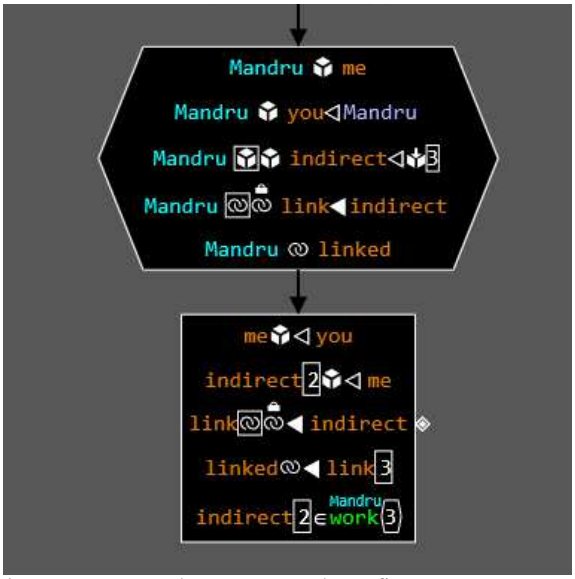
Allocator



A refer type that can be made from a variable to an Alt A shortcut. It’ s a data type that receives dynamic allocation. Dynamic allocation is to request memory from the operating system and using it while the program is running. It can be allocated with an Alt N key, and when another object is substituted, it copies that object. An object can be allocated by calling a constructor without using an Alt N key. The array allocator must specify the array size when allocated and must be indexed when used.

Allocators cannot share values with other allocators; they can only move; if moved, the original allocator will be empty.

Reference

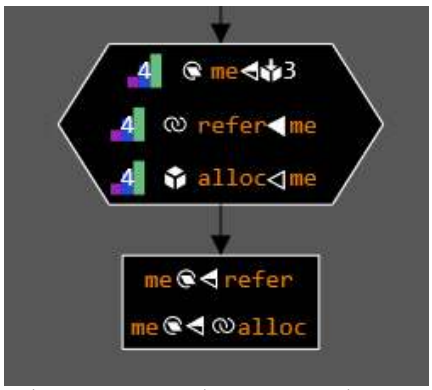


A refer type that can be made from a variable to an Alt R shortcut. A data type that refers to variables that dynamically allocated and static variables. References can be passed from an allocator to a reference, but not from a reference to an allocator. They can also be passed to an pointer, but not from an pointer.

If the allocator which allocated the object that reference refers disappears, the reference is also unavailable.

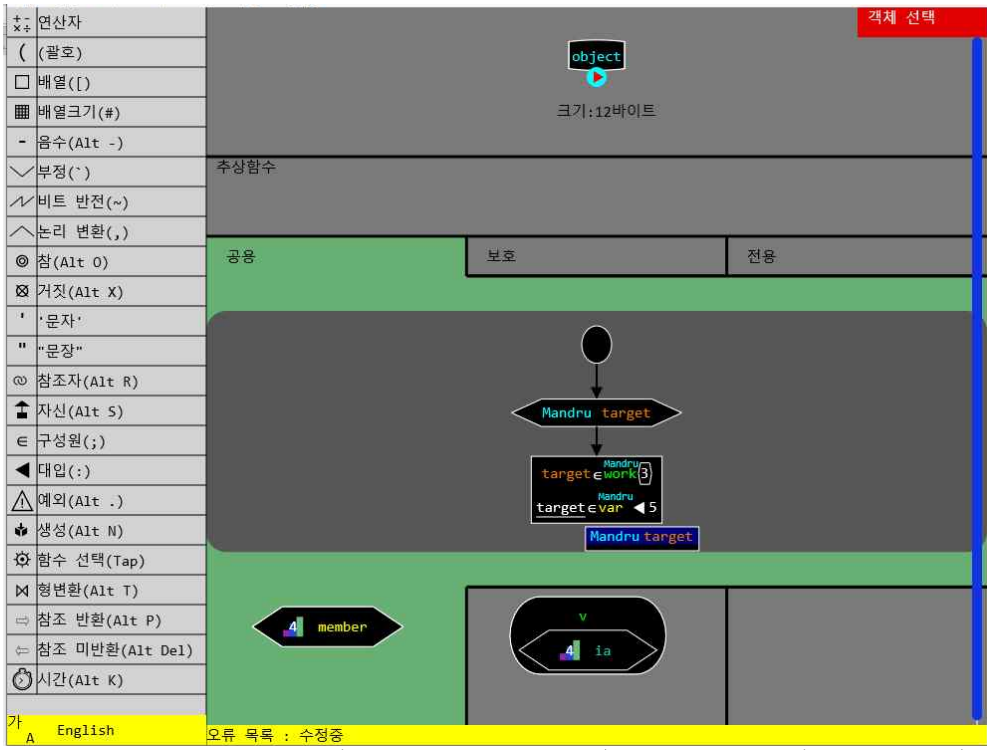
You can treat the allocator as a constant reference and pass it to the constant reference of the reference.

Hybrid Refer



It’s a refer type that can be made with an Alt D shortcut in a variable. It is a data type that can serve as both an allocator and a reference. By default, substituting references is treated as a reference, and substituting allocator is treated as an allocator, but when substituting allocatoror or mixed reference, it can be substituted by treating them as reference with an Alt P shortcut.

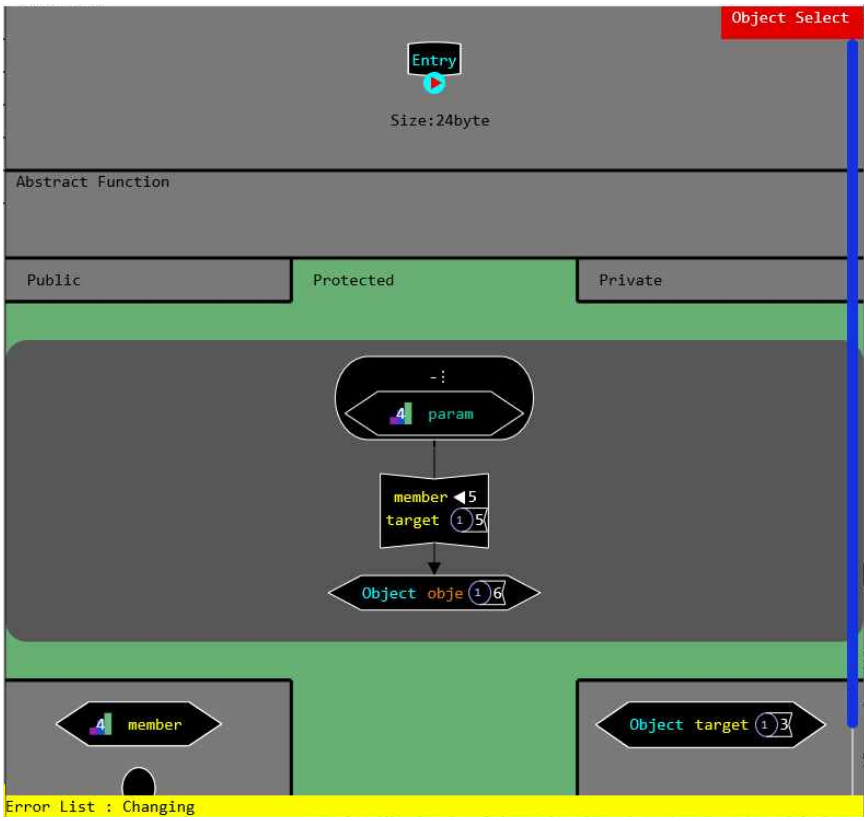
Class



When you open an object, press the space bar in the public, protection, and private area below to create a function, and enter a letter or basic data type shortcut to create a variable. Objects are data types and can be used for variables or function return type.

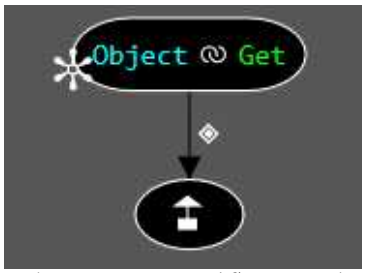
To use functions or variables that compose an class, you must write down the name of the class variable and press the ; key. When you use static variables or static functions, you must write down the class name and press the ; key. Members of an object are marked with the name of the object at the top. Other class can only access members in the public area.

Constructor



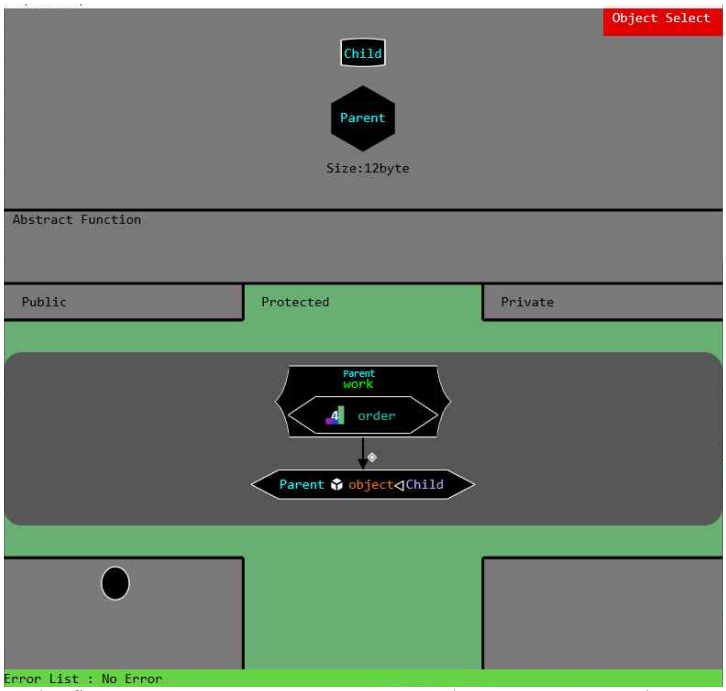
If you do not specify a return type and name in the function definition, it becomes a constructor. The constructor can initialize the member variables before writing the contents of the function where it is down from the parameter. If the constructor does not initialize member variable, use the initialization specified in the variable. By entering a space bar in the variable, the constructor can be selected and used. The first constructor that have no parameter is the default constructor. You can set another constructor as the default constructor by pressing the tab key. If you do not specify a constructor in the object variable initialization, call the default constructor.

Self-Send and Refer Function



You can pass the object itself to the pointer with the Alt S shortcut. To pass the object itself to the reference, you must enter the Alt S shortcut in the function definition to make it a reference function. Refer functions can only be invoked by references, allocators, or mixed references, and constructors, which are refer functions, can only be used when generated by allocators.

Inheritance and Override



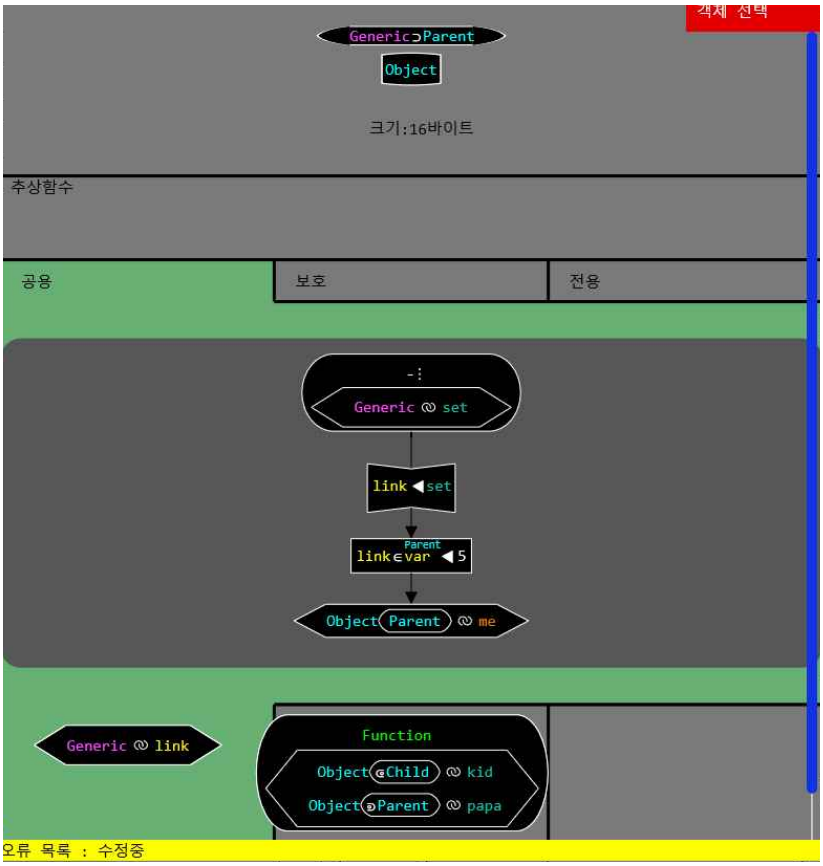
In the class definition, you can inherit another class by pressing the down arrow key. The inherit target class is called the parent class, and the class that receiving inherit is called the child class. You can pass the child class to the parent class' s reference. Functions and variables in the parent object' s protected area are available. You can override the parent class' s function with the Alt V shortcut. If you override a function, calling it from the parent class' s reference will invoke the overridden function. You can prohibit the inheritance of class and the override of functions with the Alt Z shortcut. You can create abstract functions with the Alt B shortcut. Abstract functions cannot create content, and classes with abstract functions cannot be created. If a child class that inherited the class overrides all abstract functions, that child object can be created. When using a function, you can select and call a function implemented on a specific class by pressing the tab key.

Accessor



Create an Alt Y shortcut from a variable in a protected or private area. Even if a variable is in a dedicated or protected area, it can be used from outside the class, depending on the location of the accessor, If you enter Alt C in the accessor to specify it as a constant, you cannot change it if you use a variable through an accessor outside the class. If you use a refer type through accessor, you can use only the variables it is pointing to by default, but you can enter the Alt R shortcut to use the reference.

Generic



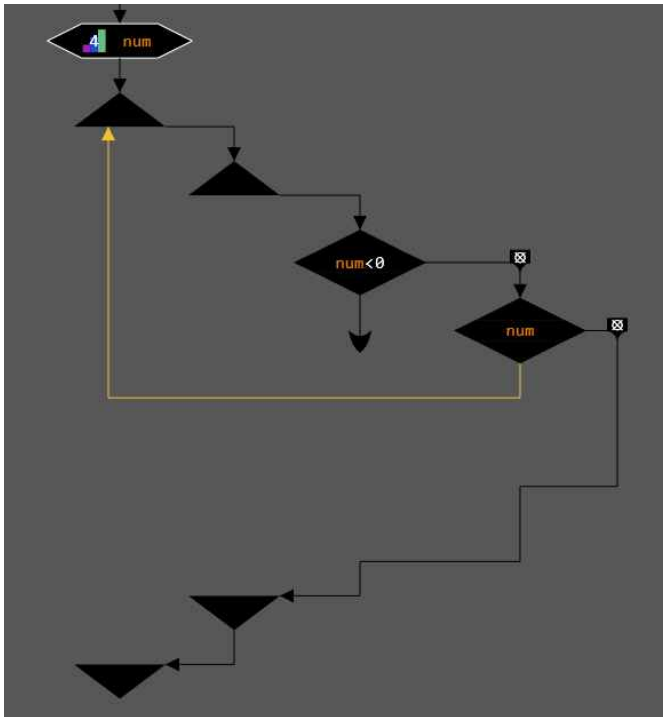
You can create it by pressing the Alt G shortcut in the object definition. It can be used as a data type inside a class, and you must use the refer type. When you use a class which generic is declared, you can put a data type in the generic list. When you use a generic variable or a function that returns a generic outside the class, you access the data type in the generic list that you put when you created the class. You can specify a parent class by pressing the : key. This allows you to create only those parent class and their child objects in the generic list. When you use generic within an class, you can also access the members of the class that you specify as a parent class.

When creating a generic list, press the ; key to make it covariant. The symbol for the covariant is ⊇. The covariant can substitute variables so that the class inherits it. When used, members of the covariant class can be accessed or referenced, but cannot be passed.

When creating a generic list, you can turn it into a contravariant by pressing a key. The symbol on the contravariant is a ⊆. A contravariant can cause its parent class to come. When used, you can pass references to a contravariant class. but you cannot access or receive references to members.

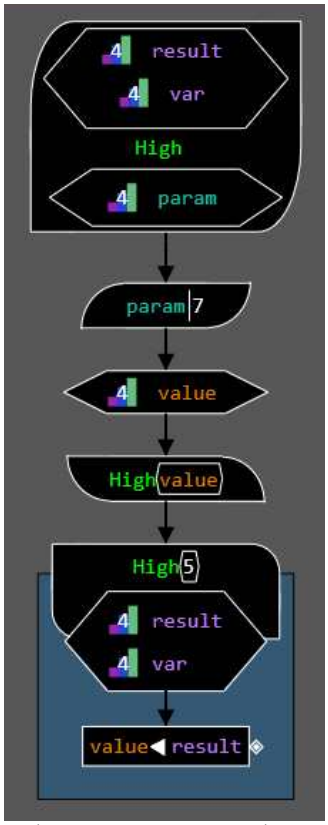
If you empty the generic list, you can substitute references regardless of the generic list, but variables or function return that use blank generic can only be used to receive references or substitute null.

Exception



You can throw an exception with the Alt Q shortcut. If you throw an exception in a conditional branch, you can only throw an exception when it comes to that branch. If an exception is thrown, it ignores the syntax until it is received. You can catch exceptions by entering an Alt Q shortcut in the loop.
There are two types of exceptions that occur without direct throwing: out of memory, stack overflow, and these exceptions change the value of the ⚠ shape variable that can be used as an Alt . shortcut key.
Out of memory is 1 and stack overflow is 2.
Dividing by 0 does not throw an exception, but the value of the variable becomes 3 and does not perform the operation.

High-Order Function

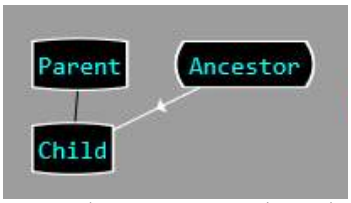


Create using Alt H shortcut in the function definition. When you use it, you must create a higher-order function syntax with the Alt H shortcut in the content of the function and use it within it. Higher-order functions doesn’ t have return type, and you can create parameters for anonymous functions by pressing the up arrow key in the function definition.

In higher-order functions, you can press the Alt E shortcut to call the anonymous function at the bottom of the higher-order function syntax. And you can enter the parameters of the anonymous function in the syntax.

Within higher-order functions, pressing the Alt J shortcut allows you to execute other higher-order functions that match the parameters of the anonymous function, and delivering the anonymous function as it is.

Multi Inheritance



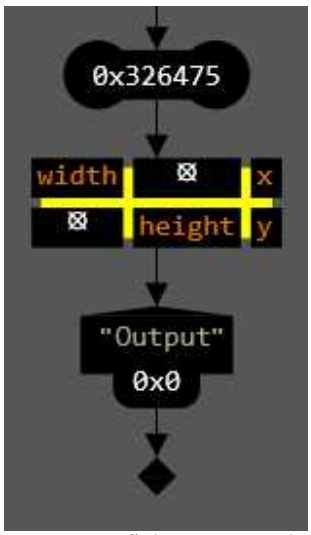
For a single class to inherit multiple classes, you must inherit an interface or ancestor class. An interface is an class that consists of only abstract functions, and an ancestor class is created by pressing a space bar on a class that does not inherit other class.

Type Cast



You can cast the data type of a variable with the Alt T shortcut. You can only cast it from the refer type. If the original data type of the variable you are referring to fits the data type you want to cast, it will be cast, and if it does not, 0 will be returned. For example, in the picture above, the object stored in the created variable is a parent class and will not be cast into a child class, leaving the set variable empty.

Rendering



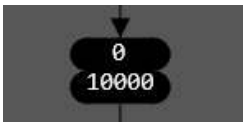
You can include picture files with Ctrl I shortcuts. Start drawing with the Alt Home shortcut and enter the background color code. You can write numbers that begin with 0x to create hexadecimal color codes. Finish drawing with the Alt End shortcut. You can specify a matrix with the Alt T shortcut to determine the size, rotation, and position of a picture. You can draw a picture by pressing the Alt Insert shortcut and entering the number of the picture file in the center. You can crop a picture and display only a portion of it by changing the value of the phrase on all sides. You can print text with the Alt Spacebar shortcut. Place the string you want to print at the top and type the color code at the bottom.

Input



You can receive Windows procedure messages with the Alt M shortcut. You can receive message, wparam, and lparam in order from the top, and you can receive the message by entering a 4-byte integer and an 8-byte integer variable for the rest. If there is no message, the value does not change, so reset the value of the variable that receives the message to 0 and if it is still 0 when you receive the message, there is no message.

Audio



You can include sound files in the wav format with an Alt I shortcut. Create sound output syntax with the Alt PageUp shortcut. Put the number of the sound file at the top and the volume at the bottom. The maximum volume is 10000, and 0 turns off the sound.