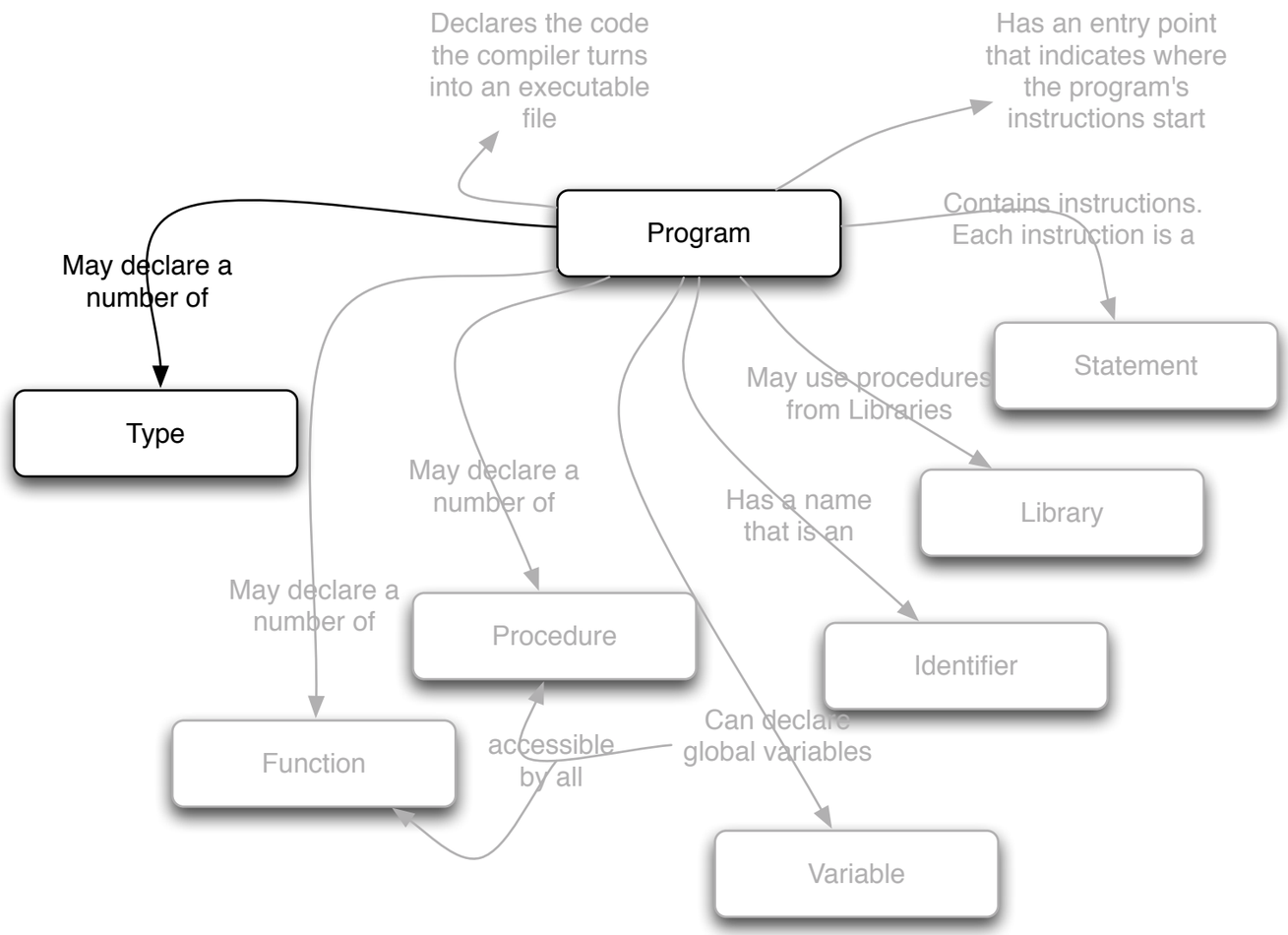
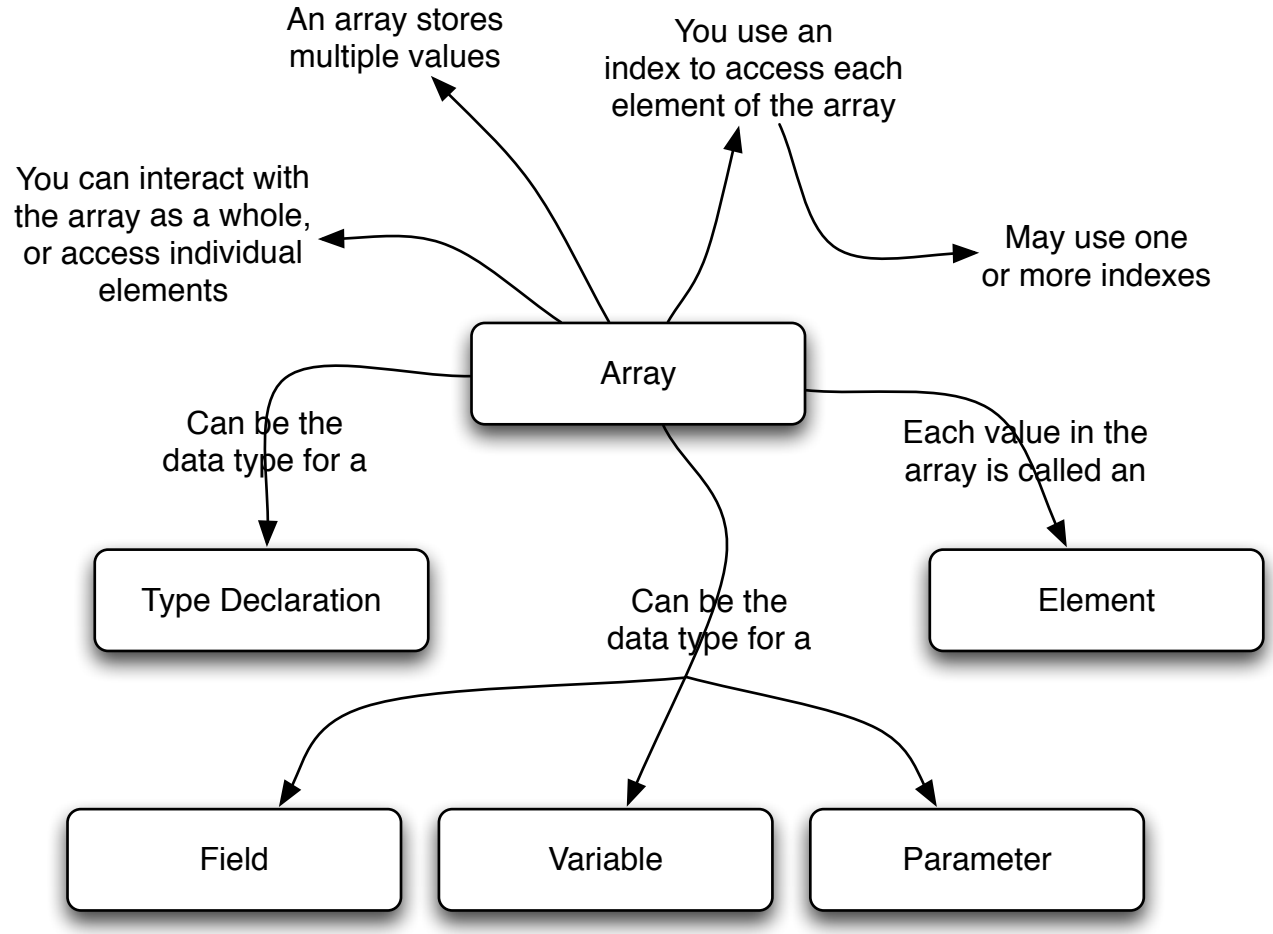
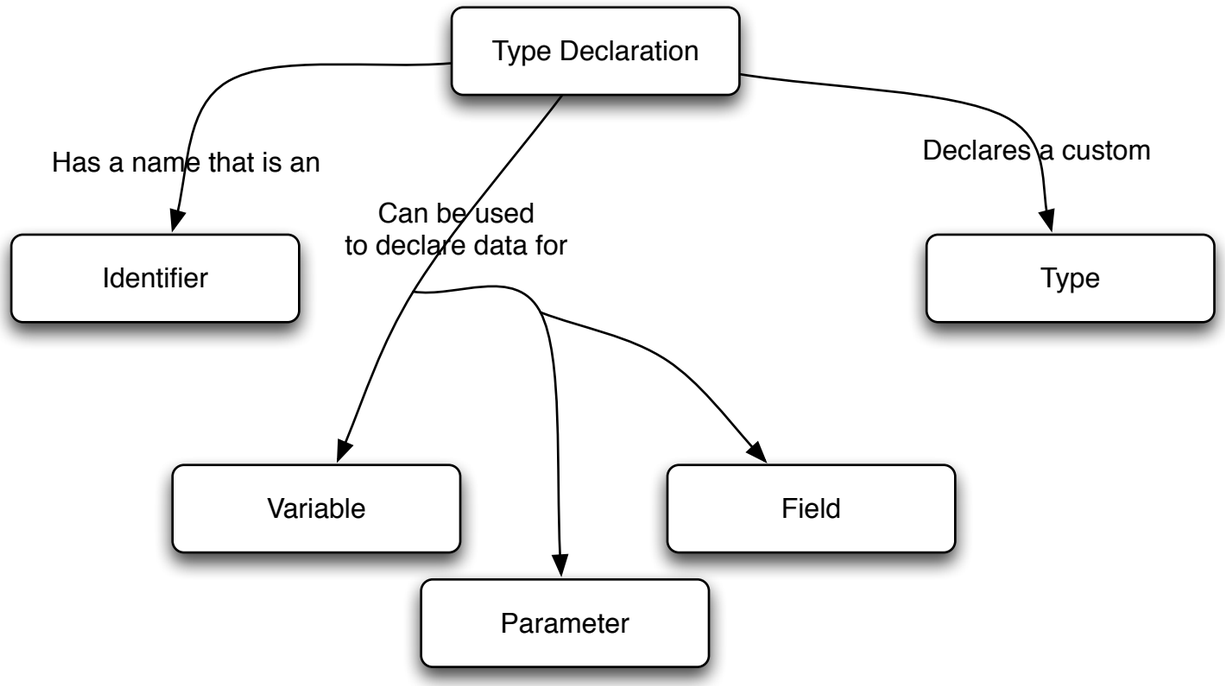
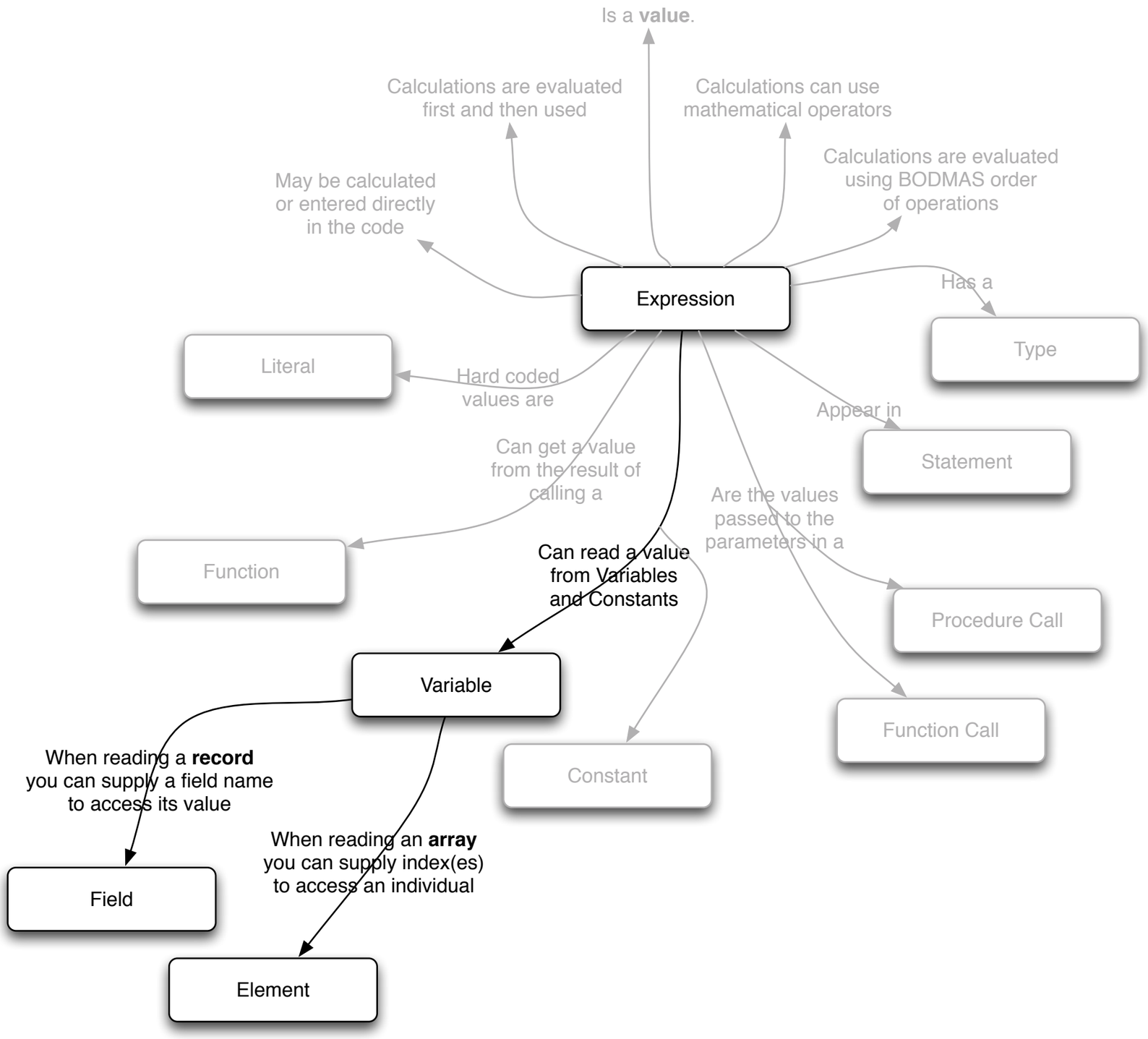
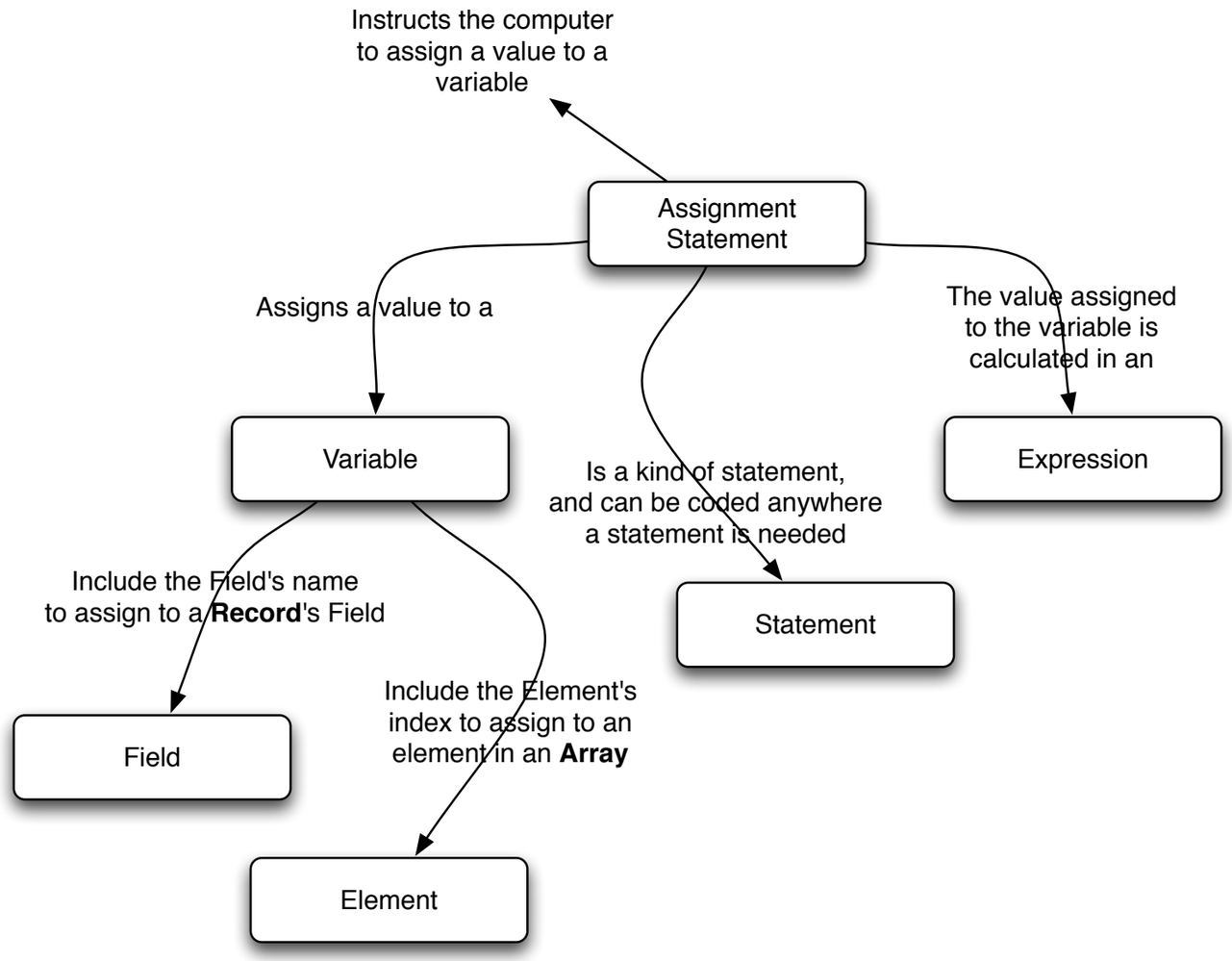
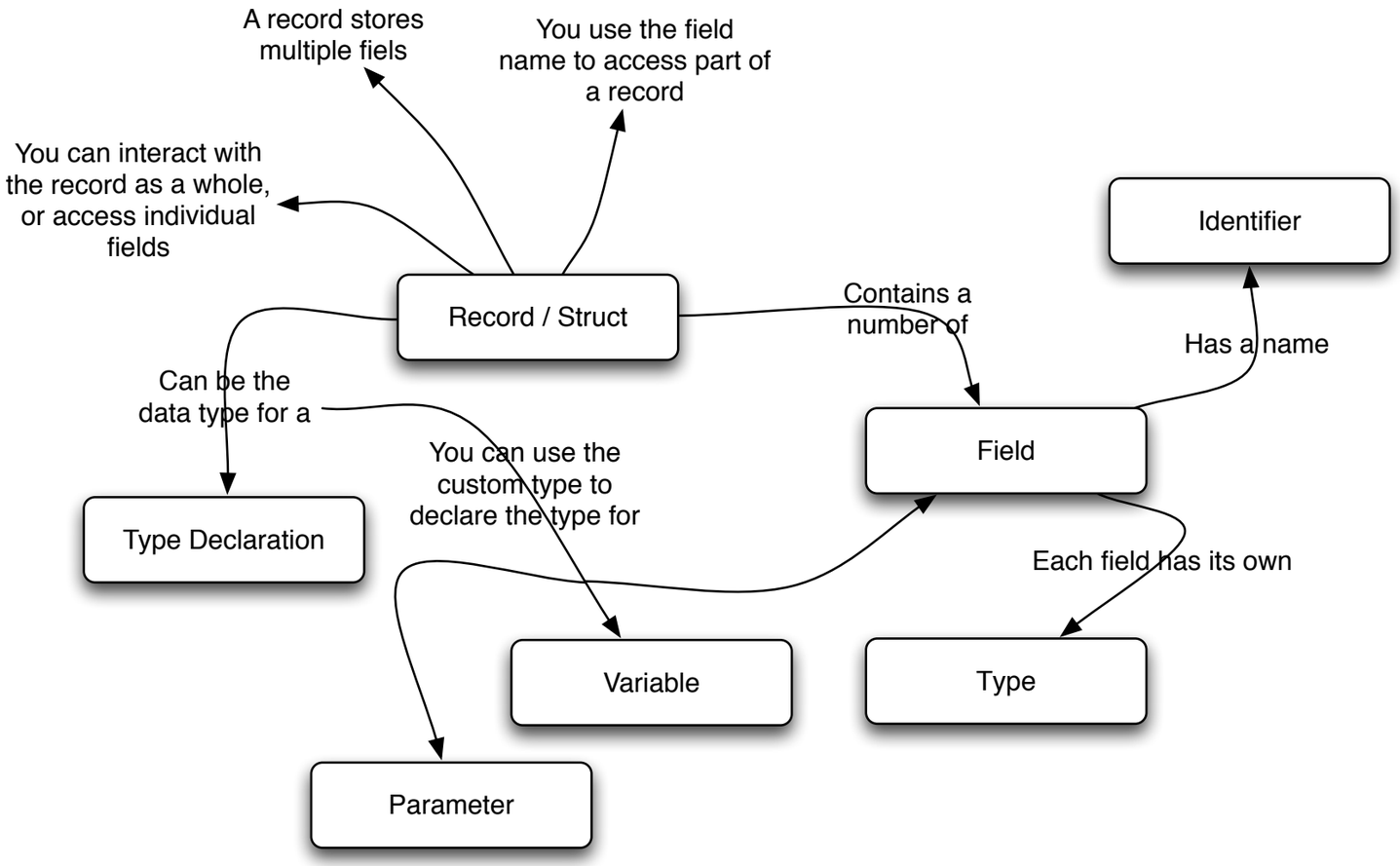

Declares the code the compiler turns into an executable file
Has an entry point that indicates where the program's instructions start
Program
Contains instructions. Each instruction is a
May declare a number of
Type
Statement
May use procedures from Libraries
Library
May declare a number of
Has a name that is an
May declare a number of
Procedure
Identifier
Function
accessible by all
Can declare global variables
Variable

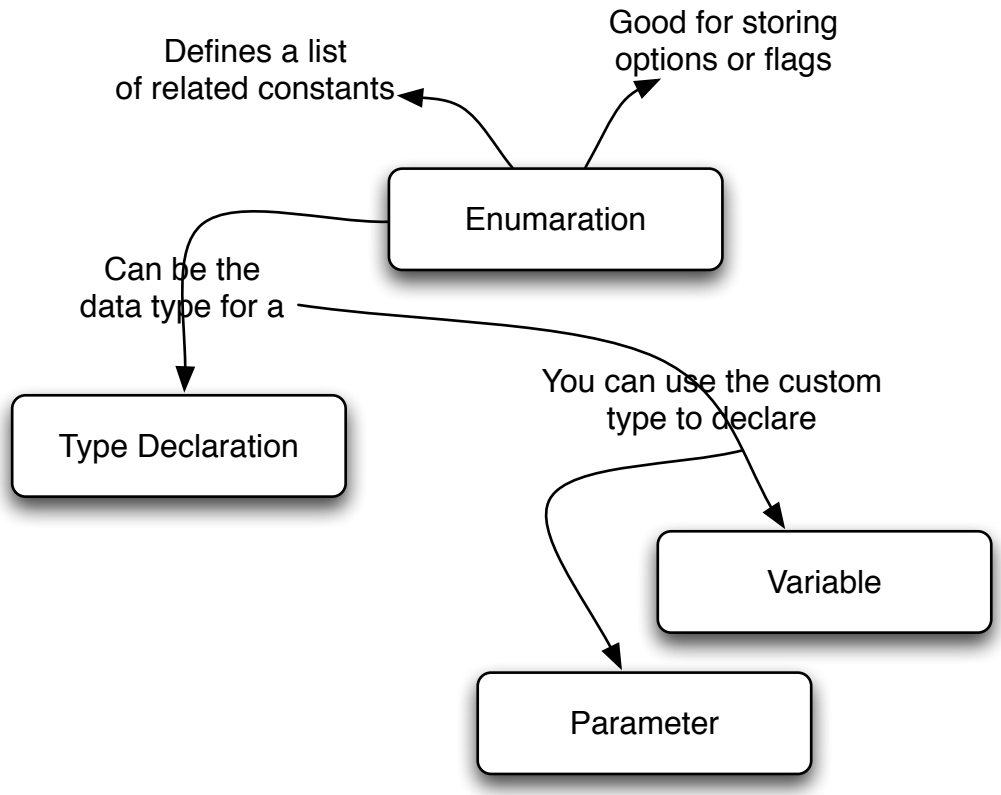

An array stores multiple values
You use an index to access each element of the array
You can interact with the array as a whole, or access individual elements
May use one or more indexes
Array
Can be the data type for a
Each value in the array is called an
Type Declaration
Element
Can be the data type for a
Field
Variable
Parameter

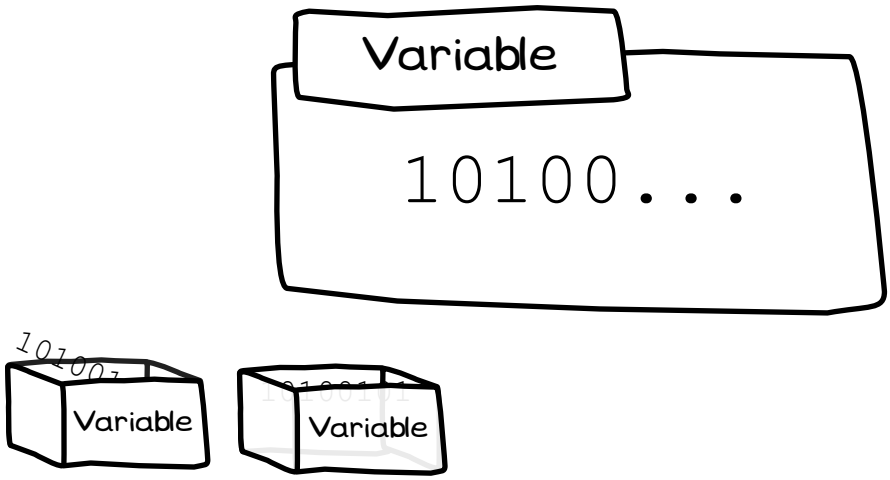

Type Declaration
Has a name that is an
Identifier
Can be used
to declare data for
Declares a custom
Type
Variable
Field
Parameter

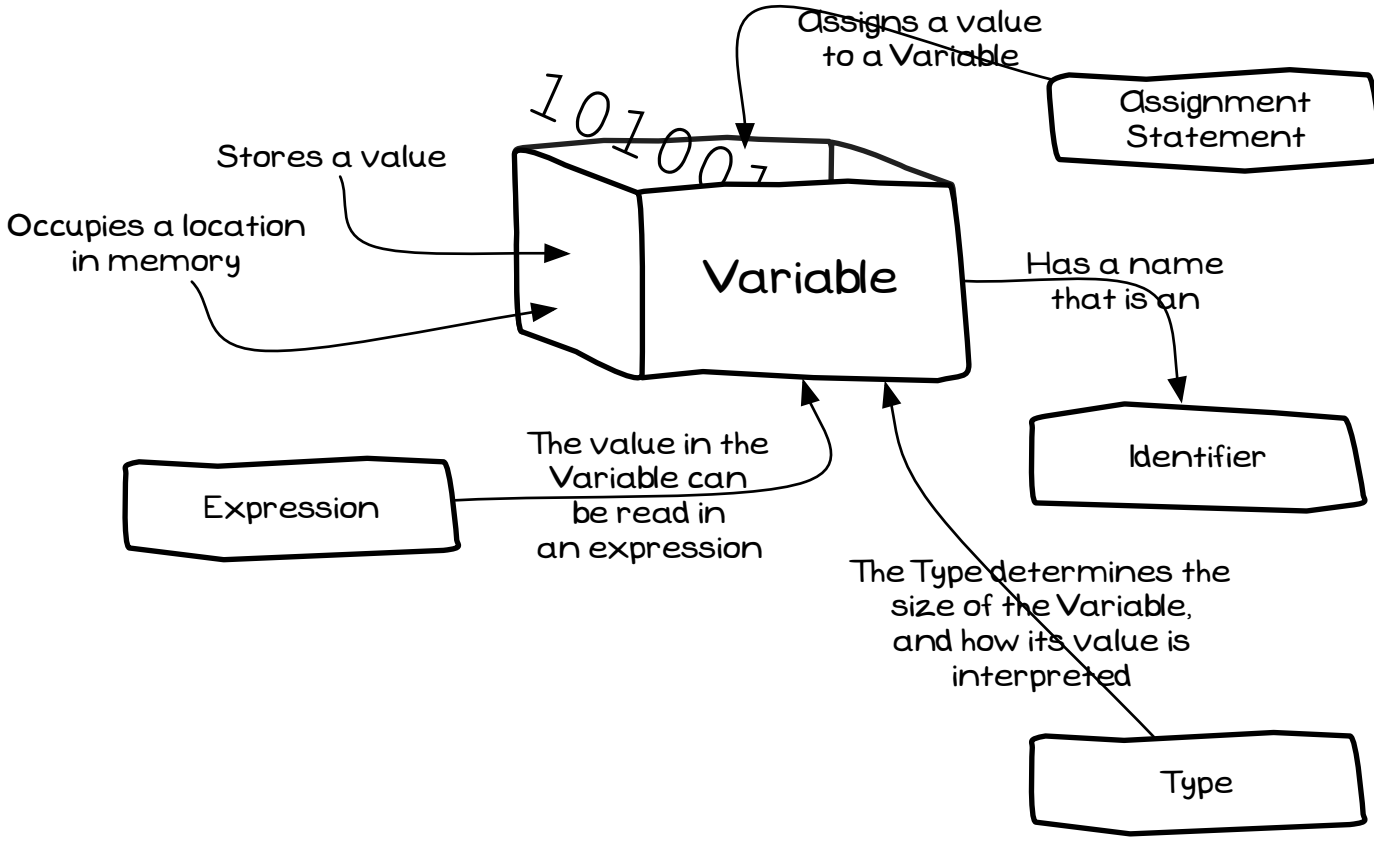

Is a **value**.
Calculations are evaluated first and then used
Calculations can use mathematical operators
May be calculated or entered directly in the code
Calculations are evaluated using BODMAS order of operations
Expression
Has a
Type
Literal
Hard coded values are
Appear in
Statement
Can get a value from the result of calling a
Are the values passed to the parameters in a
Function
Can read a value from Variables and Constants
Procedure Call
Variable
Function Call
When reading a **record** you can supply a field name to access its value
Constant
When reading an **array** you can supply index(es) to access an individual
Field
Element

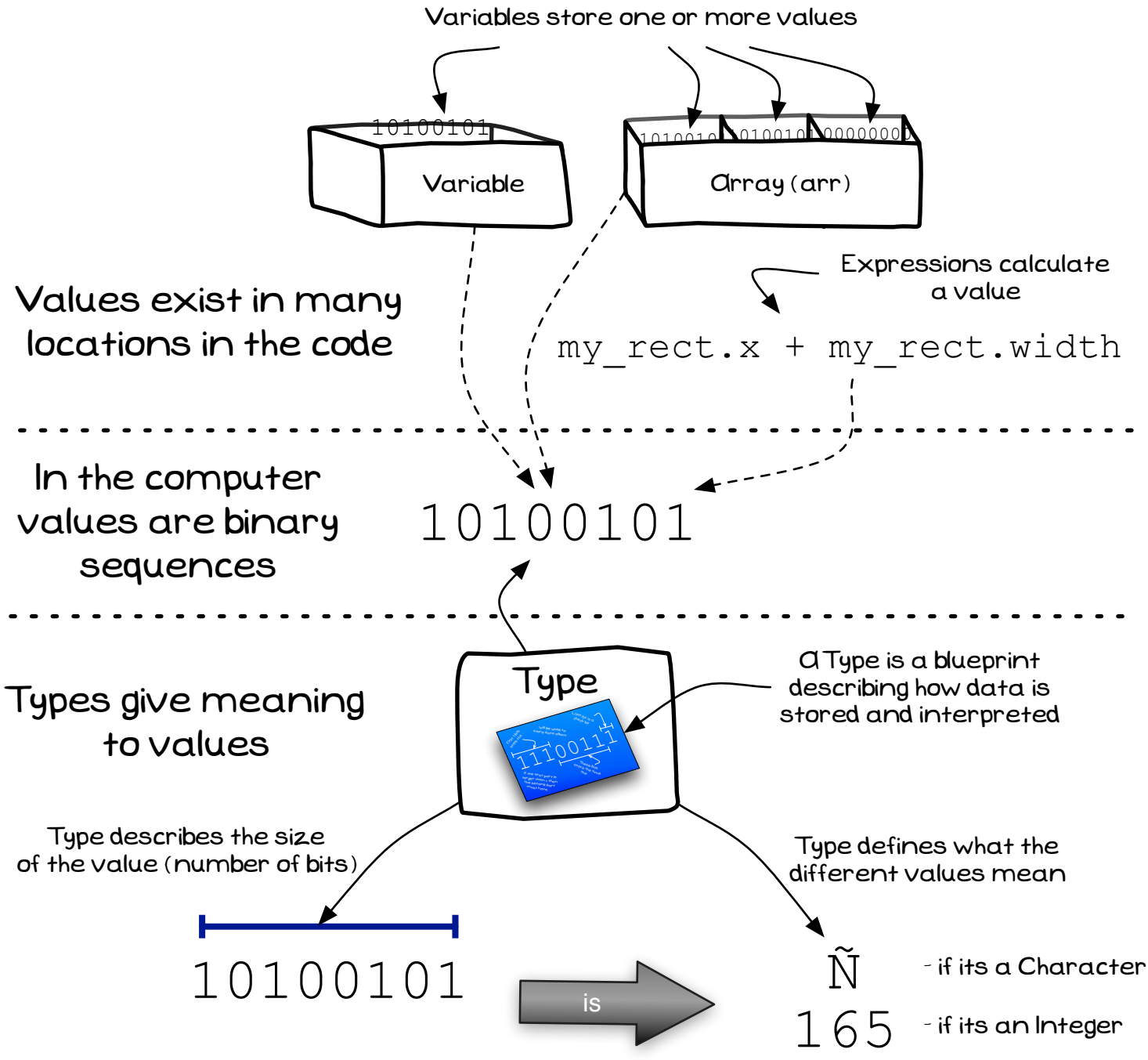

Instructs the computer to assign a value to a variable
Assignment Statement
Assigns a value to a
The value assigned to the variable is calculated in an
Variable
Expression
Is a kind of statement, and can be coded anywhere a statement is needed
Statement
Include the Field's name to assign to a **Record**'s Field
Include the Element's index to assign to an element in an **Array**
Field
Element

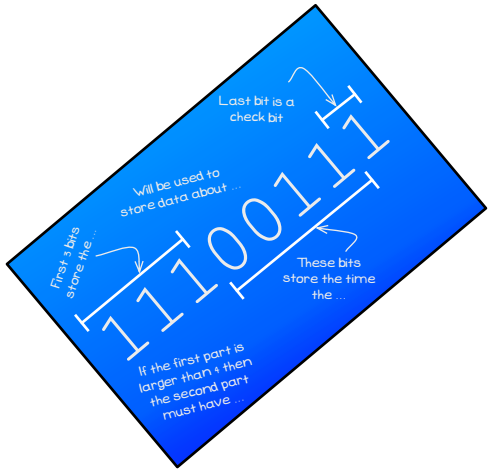

A record stores multiple fiels
You use the field name to access part of a record
You can interact with the record as a whole, or access individual fields
Record / Struct
Contains a number of
Identifier
Has a name
Field
Can be the data type for a
Type Declaration
You can use the custom type to declare the type for
Variable
Parameter
Each field has its own
Type

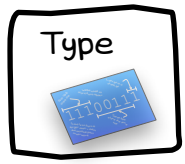

Defines a list
of related constants
Good for storing
options or flags
Enumaration
Can be the
data type for a
Type Declaration
You can use the custom
type to declare
Variable
Parameter

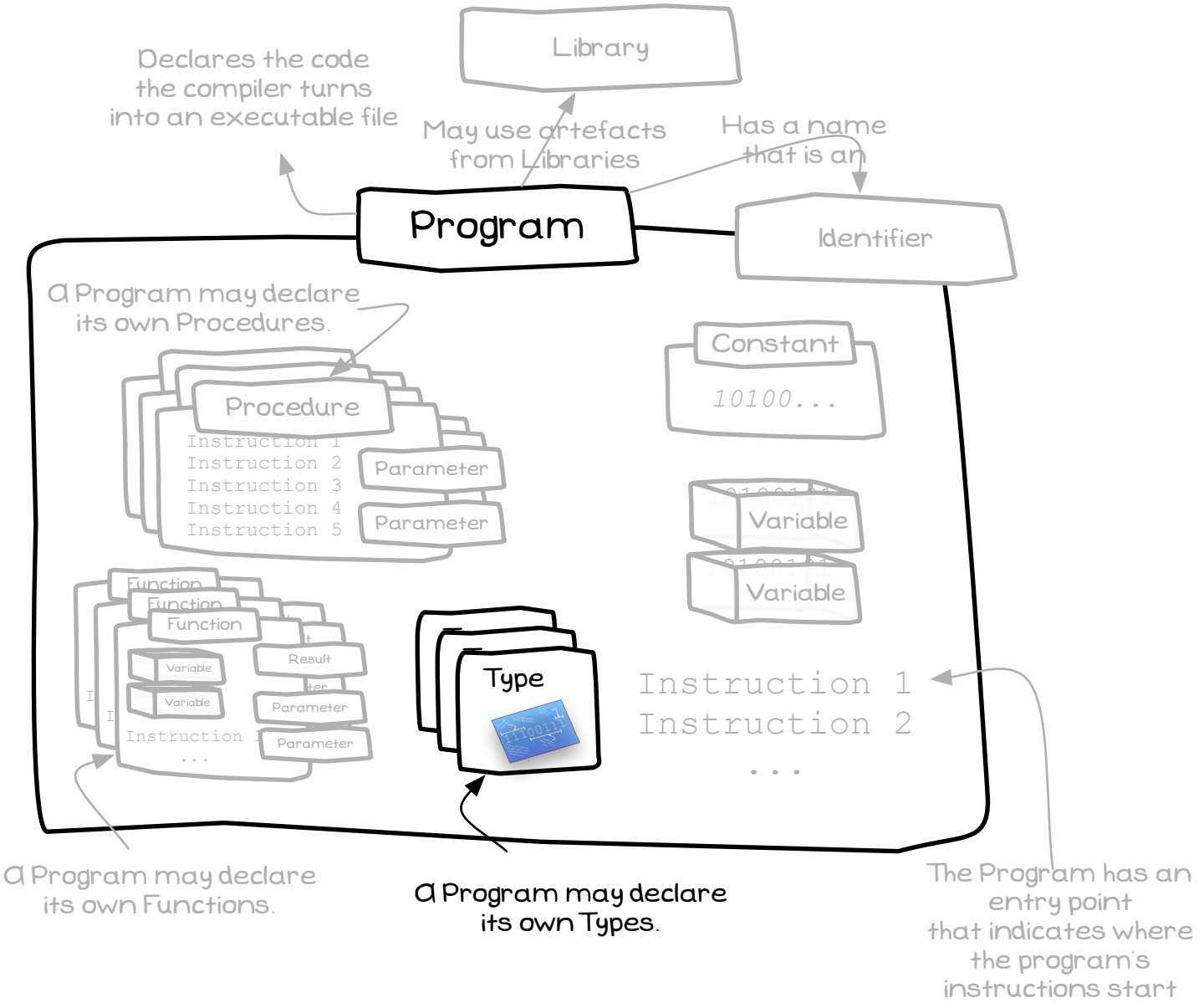

Variable
10100...
Variable
Variable

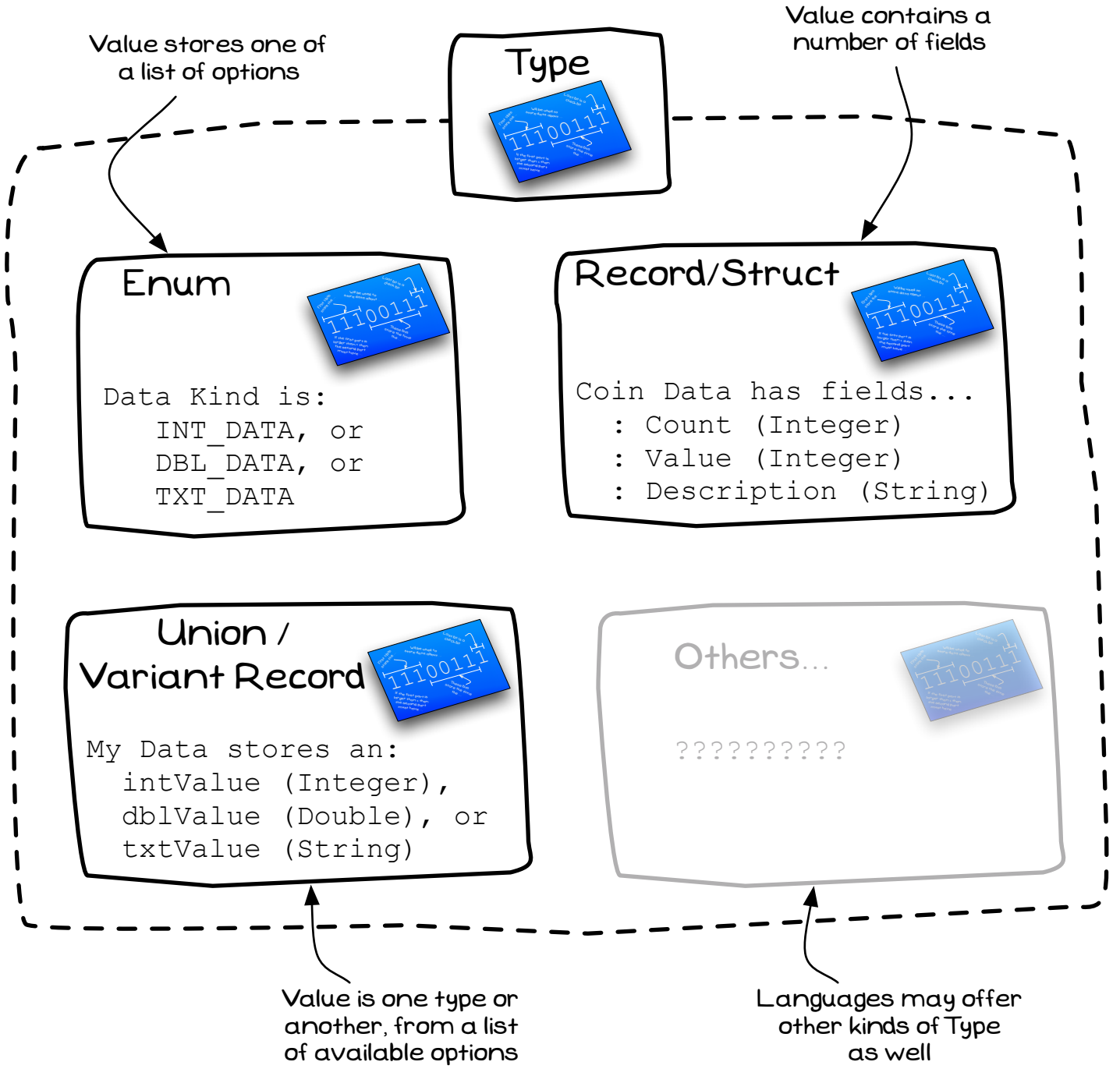

Assigns a value to a Variable
Assignment Statement
Stores a value
Occupies a location in memory
Variable
Has a name that is an
Identifier
Expression
The value in the Variable can be read in an expression
The Type determines the size of the Variable, and how its value is interpreted
Type


Variables store one or more values
10100101
Variable
Array (arr)
Values exist in many locations in the code
Expressions calculate a value
my_rect.x + my_rect.width
In the computer values are binary sequences
10100101
Types give meaning to values
Type
A Type is a blueprint describing how data is stored and interpreted
Type describes the size of the value (number of bits)
Type defines what the different values mean
10100101
is
Ñ
- if its a Character
165
- if its an Integer
A Type is a specification

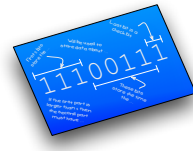

Last bit is a check bit
Will be used to store data about ...
First 3 bits store the ...
11100111
These bits store the time the ...
If the first part is larger than 4 then the second part must have ...


Type

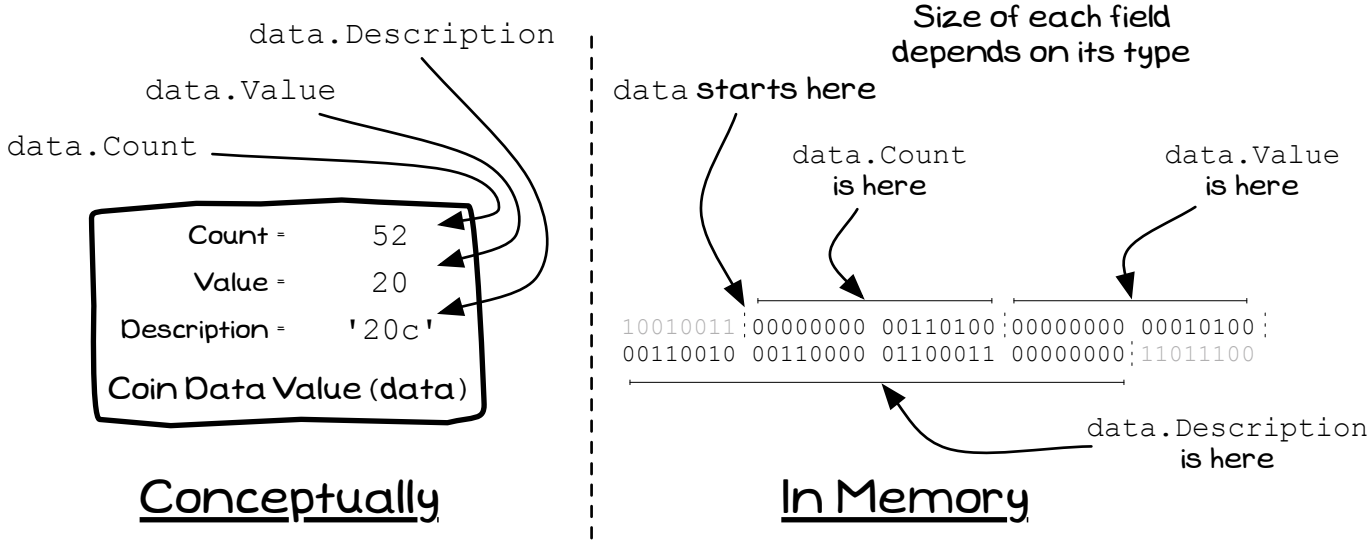

Library
Declares the code the compiler turns into an executable file
May use artefacts from Libraries
Has a name that is an
Program
Identifier
A Program may declare its own Procedures.
Procedure
Instruction 1
Instruction 2
Instruction 3
Instruction 4
Instruction 5
Parameter
Parameter
Constant
10100...
Variable
Variable
Function
Function
Function
Variable
Variable
Result
Parameter
Parameter
Type
Instruction 1
Instruction 2
...
A Program may declare its own Functions.
A Program may declare its own Types.
The Program has an entry point that indicates where the program's instructions start


Value stores one of a list of options
Type
Value contains a number of fields
Enum
Data Kind is:
INT_DATA, or
DBL_DATA, or
TXT_DATA
Record/Struct
Coin Data has fields...
: Count (Integer)
: Value (Integer)
: Description (String)
Union / Variant Record
My Data stores an:
intValue (Integer),
dblValue (Double), or
txtValue (String)
Others...
??????????
Value is one type or another, from a list of available options
Languages may offer other kinds of Type as well

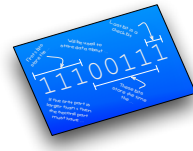
Record/Struct



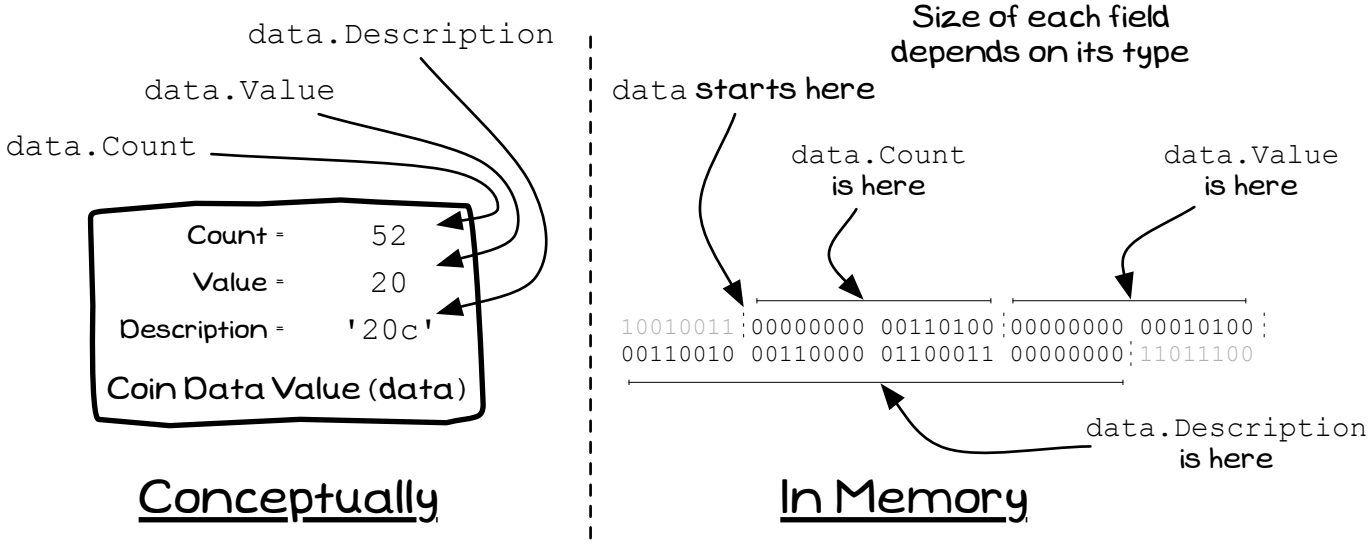

```
Coin Data has fields...
   : Count (Integer)
   : Value (Integer)
   : Description (String)
```



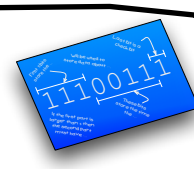
Record/Struct




```
Row has fields...
   : Id (Integer)
   : Kind (Data Kind)
   : Value (Column Value)
```



Enum



```
Data Kind is:
   INT_DATA, or
   DBL_DATA, or
   TXT_DATA
```

Each value is either
`INT_DATA`, `DBL_DATA`, or `TXT_COINS`

`DBL_DATA`

Data Kind

Each option is mapped
to a numeric value.

`INT_DATA` is mapped to `0`

`DBL_DATA` is mapped to `1`

`TXT_DATA` is mapped to `2`

`10010011` `00000001` `11011100`

The first option get `0`,
the second `1`, the third `2`,
and so on...

Conceptually

In Memory

Union / Variant Record

```
My Data stores an:
  intValue (Integer),
  dblValue (Double), or
  txtValue (String of 9 chars)
```

Space taken is as large as the largest type that could be stored

Each value stores either the `Integer intValue`, the `Double dblValue`, or `Coin Data coinValue`

`intValue 165`

My Data (value)

Conceptually

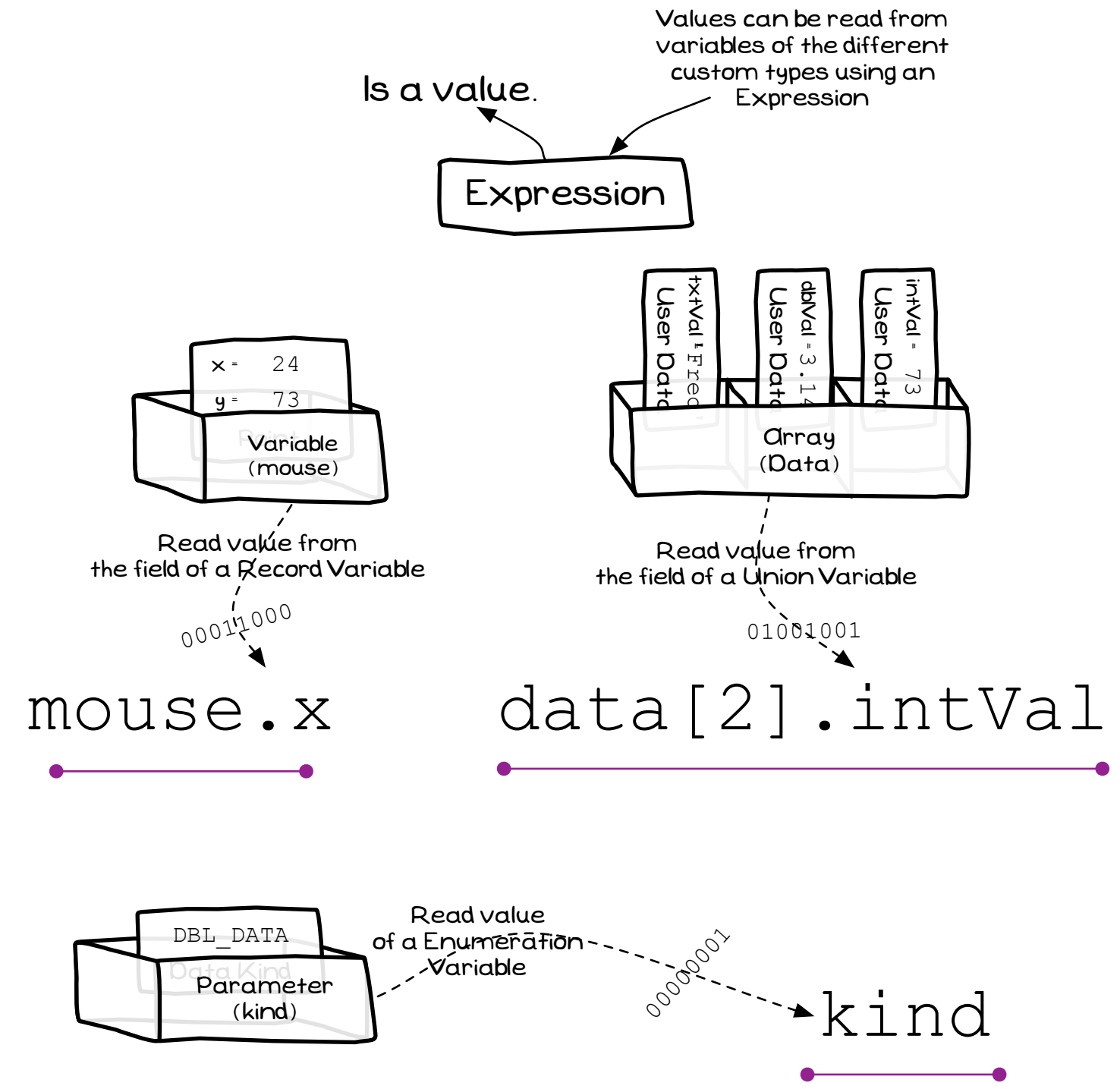
If this stores `intValue` then it only uses the first four bytes (32 bits) (others taken, but not used)

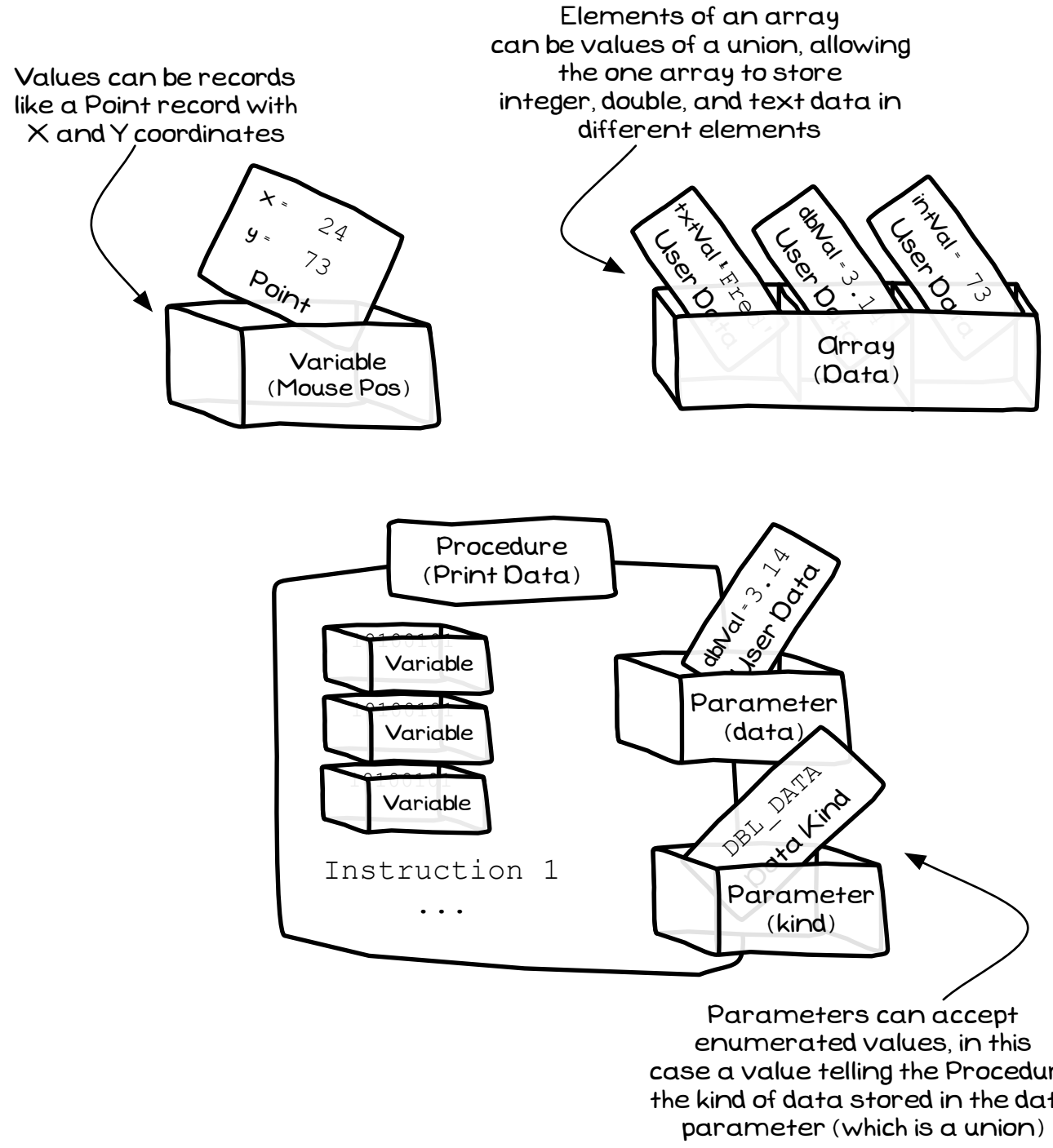
All 9 bytes, 72 bits are used when this stores `txtValue`

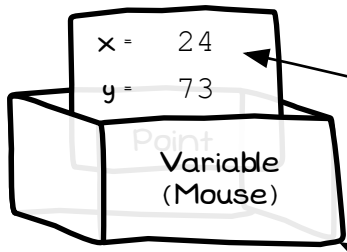
```
10010011 00000000 00000000 00000000 10100101
00110010 00110000 01100011 00000000 11011100
```

8 bytes (64 bits) are used when this stores `dblValue`

In Memory


Values can be read from variables of the different custom types using an Expression
Is a value.
Expression
x = 24
y = 73
Variable (mouse)
Read value from the field of a Record Variable
00011000
mouse.x
txtVal = "Fred"
dblVal = 3.14
intVal = 73
User Data
Array (Data)
Read value from the field of a Union Variable
01001001
data[2].intVal
DBL_DATA
Data Kind
Parameter (kind)
Read value of a Enumeration Variable
00000001
kind


Values can be records like a Point record with X and Y coordinates
x = 24
y = 73
Point
Variable (Mouse Pos)
Elements of an array can be values of a union, allowing the one array to store integer, double, and text data in different elements
txtVal = "Fre
dbVal = 3.1
intVal = 73
Array (Data)
Procedure (Print Data)
Variable
Variable
Variable
Instruction 1
...
dbVal = 3.14
Parameter (data)
DBL_DATA
Parameter (kind)
Parameters can accept enumerated values, in this case a value telling the Procedure the kind of data stored in the data parameter (which is a union)

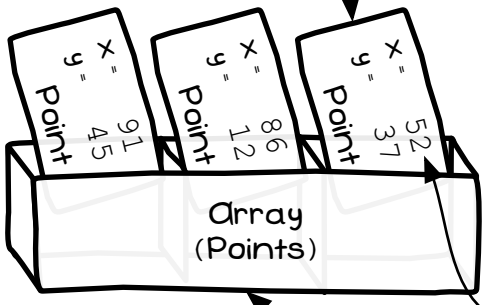


mouse.x

Is an Integer, it can be used anywhere an Integer is required

mouse

Is a Point, it can be used anywhere a Point is required



points[2]

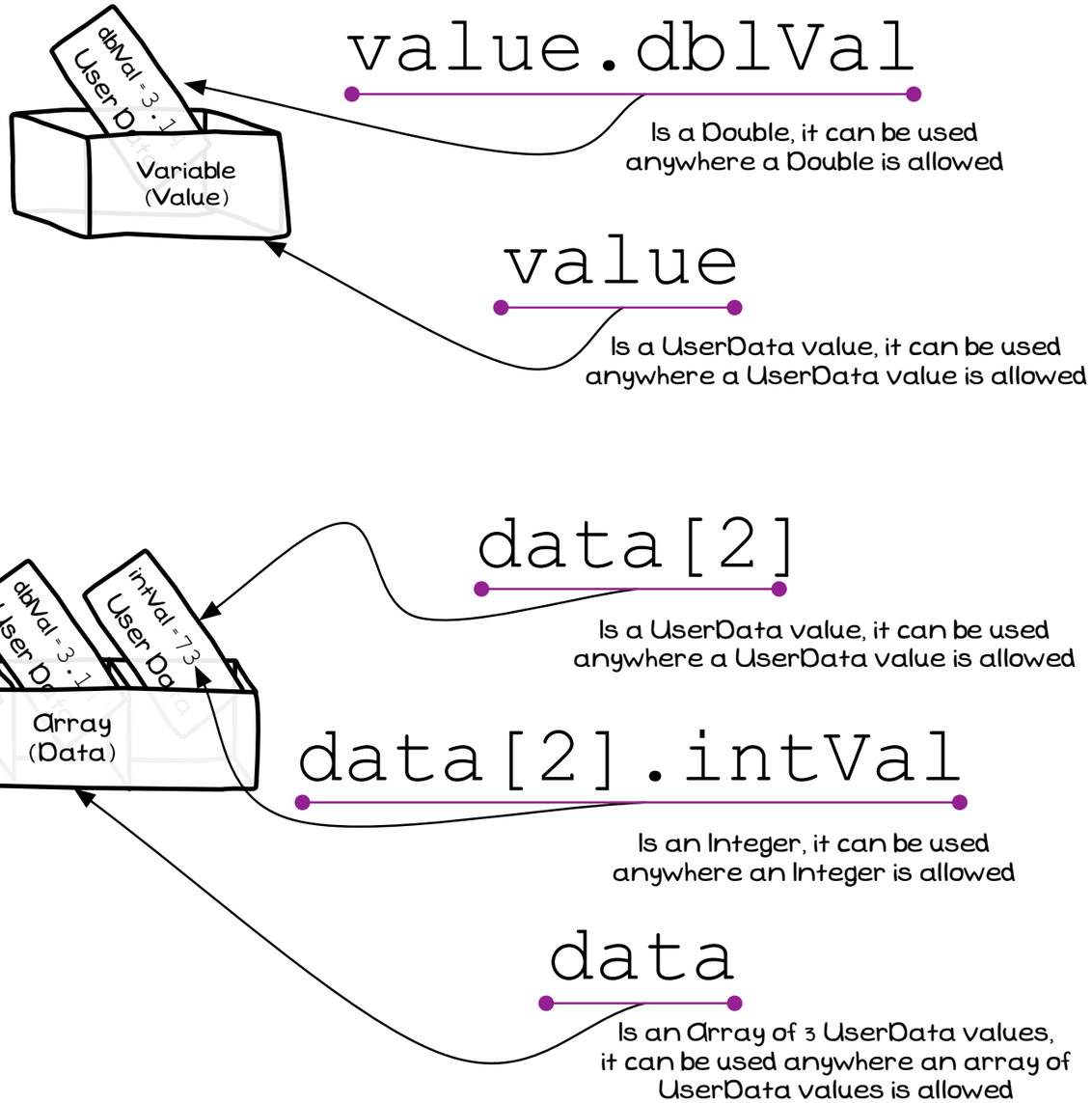
Is a Point, it can be used anywhere a Point is required

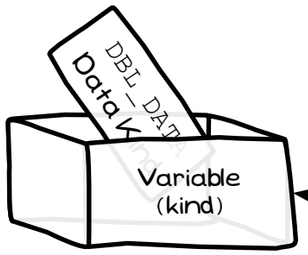
points[2].x

Is an Integer, it can be used anywhere an Integer is required

points

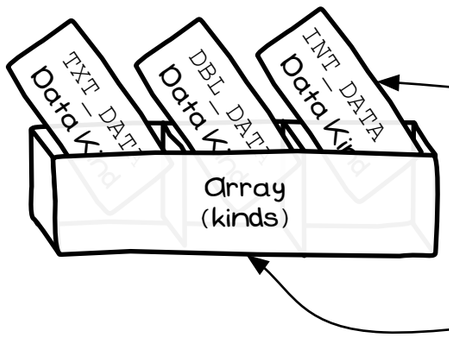
Is an Array of 3 Points, it can be used anywhere an array of Points is required


value.dblVal
Is a Double, it can be used anywhere a Double is allowed
value
Is a UserData value, it can be used anywhere a UserData value is allowed
Variable (Value)
data[2]
Is a UserData value, it can be used anywhere a UserData value is allowed
data[2].intVal
Is an Integer, it can be used anywhere an Integer is allowed
data
Is an Array of 3 UserData values, it can be used anywhere an array of UserData values is allowed
Array (Data)



kind

Is a DataKind value, it can be used
anywhere a DataKind value is allowed

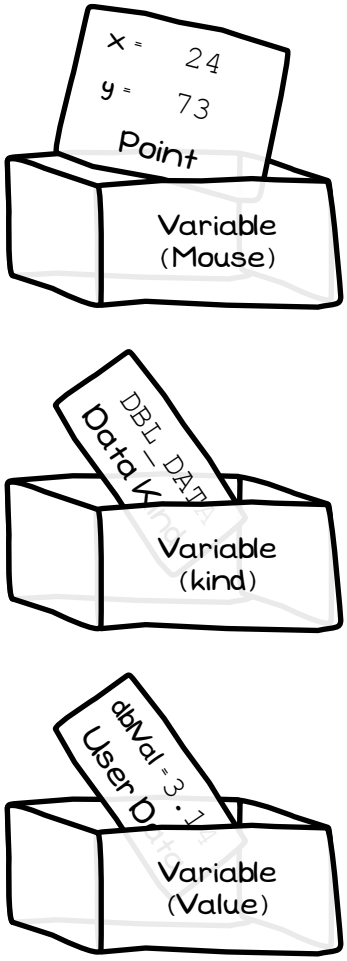


kinds[2]

Is a DataKind value, it can be used
anywhere a DataKind value is allowed

kinds

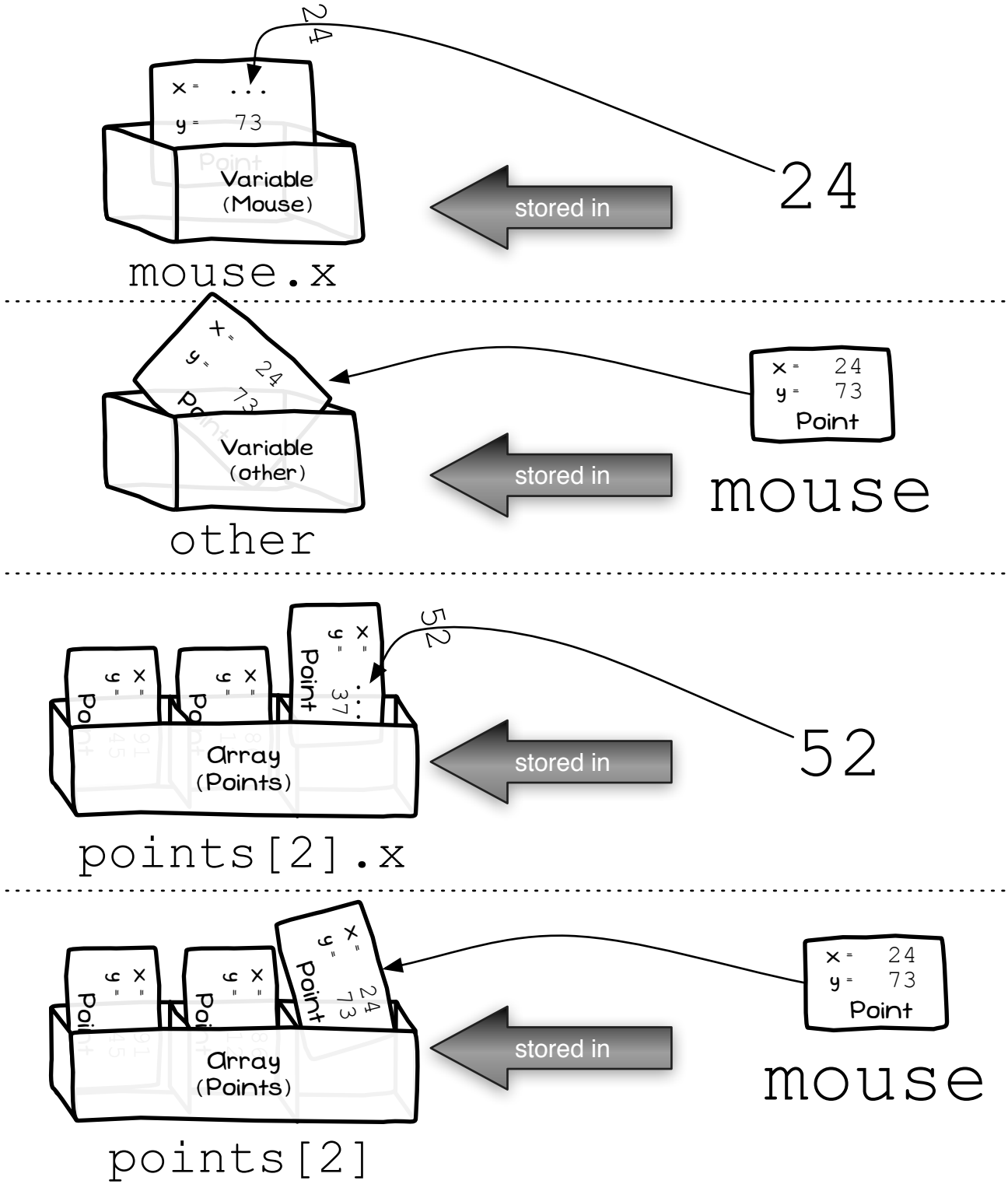
Is an Array of 3 DataKind values,
it can be used anywhere an array of
DataKind values is allowed

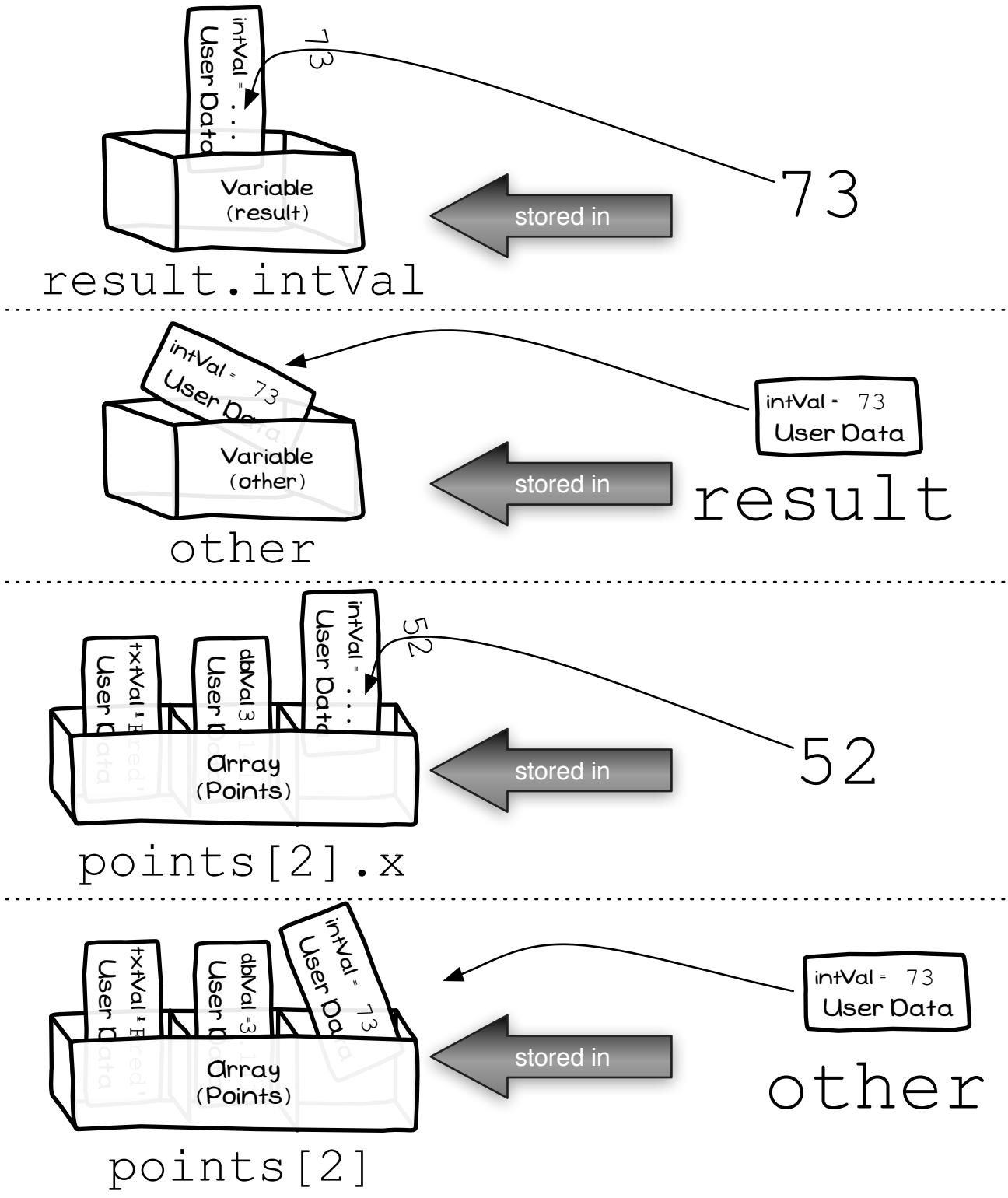


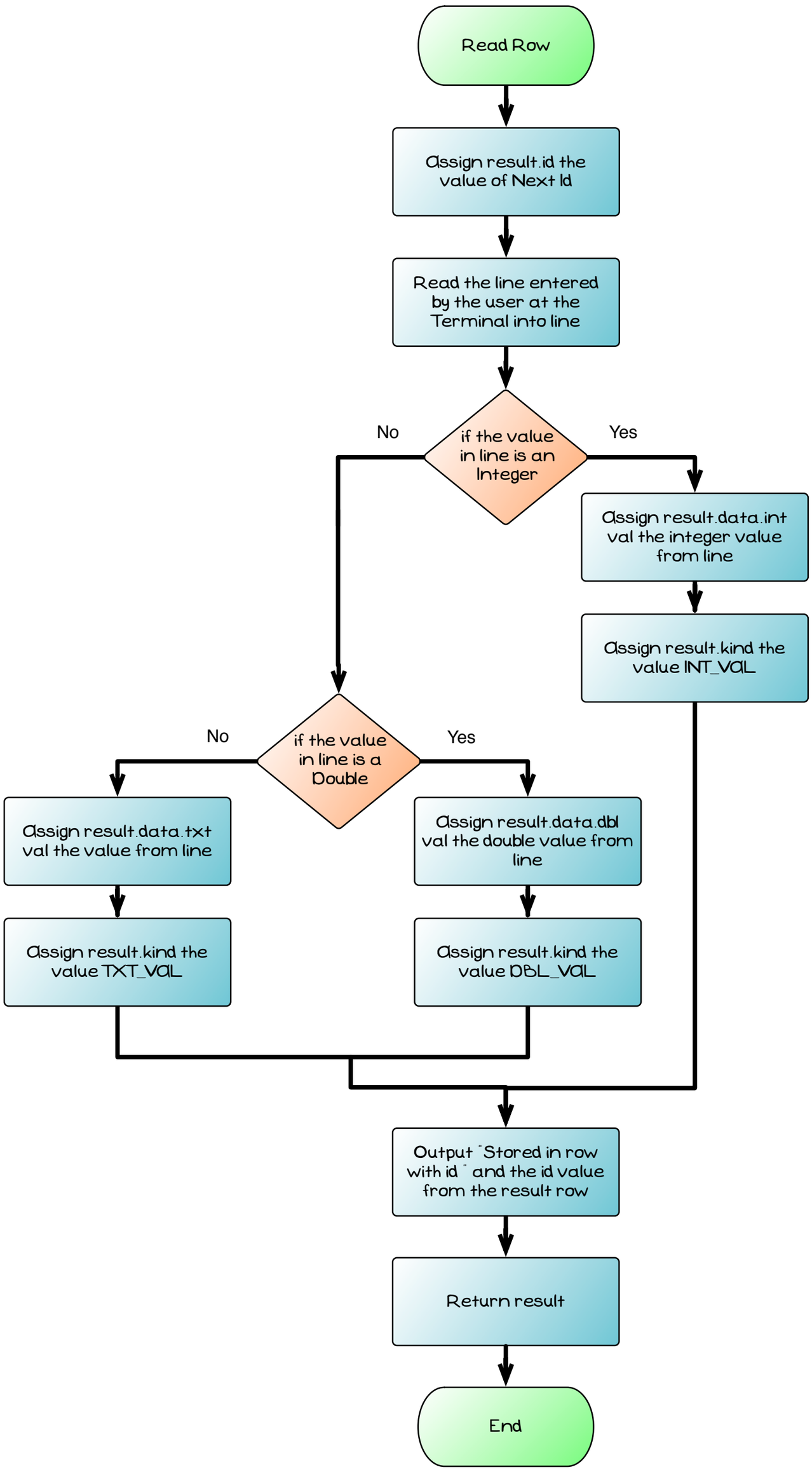
stored in

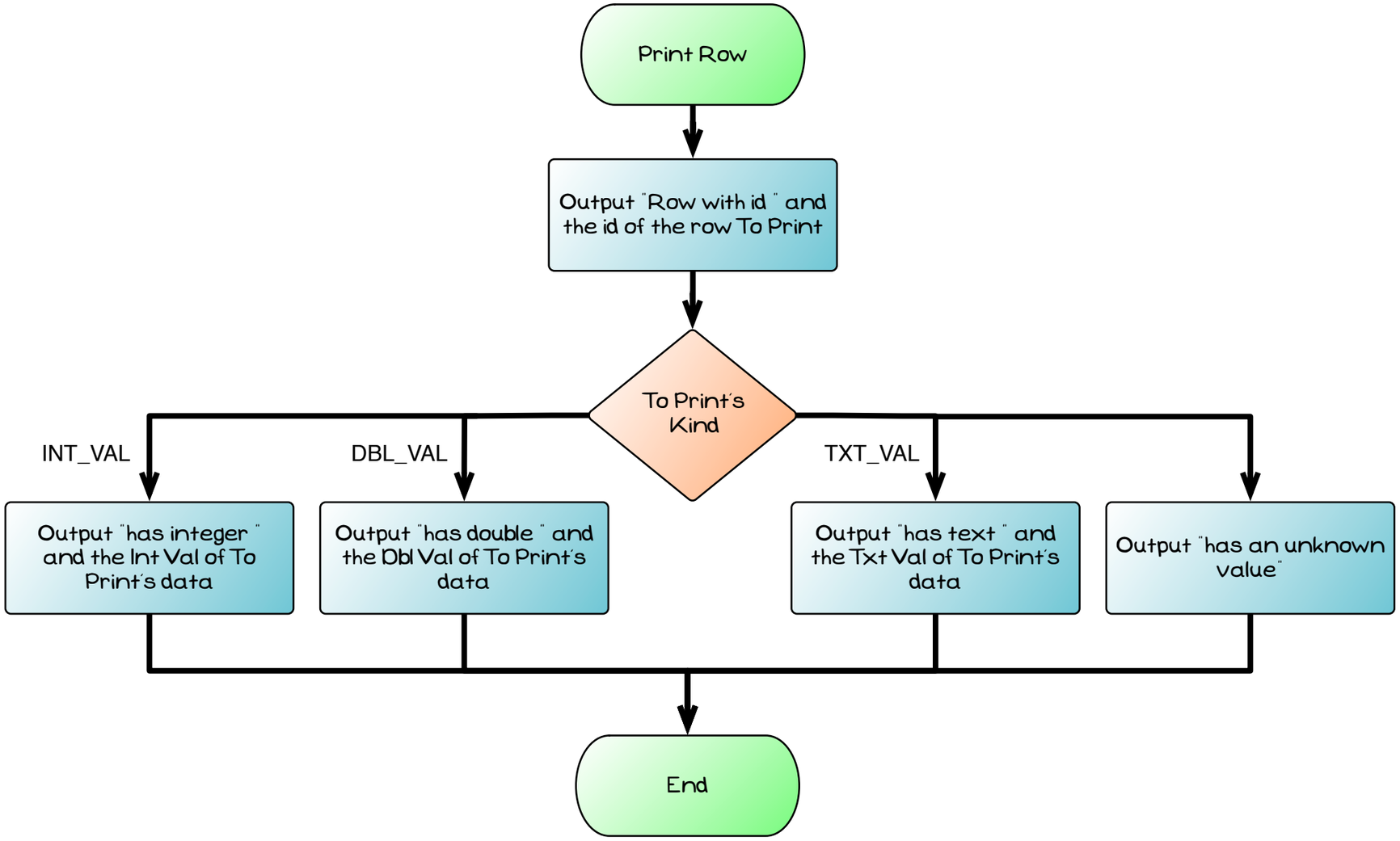
Expression

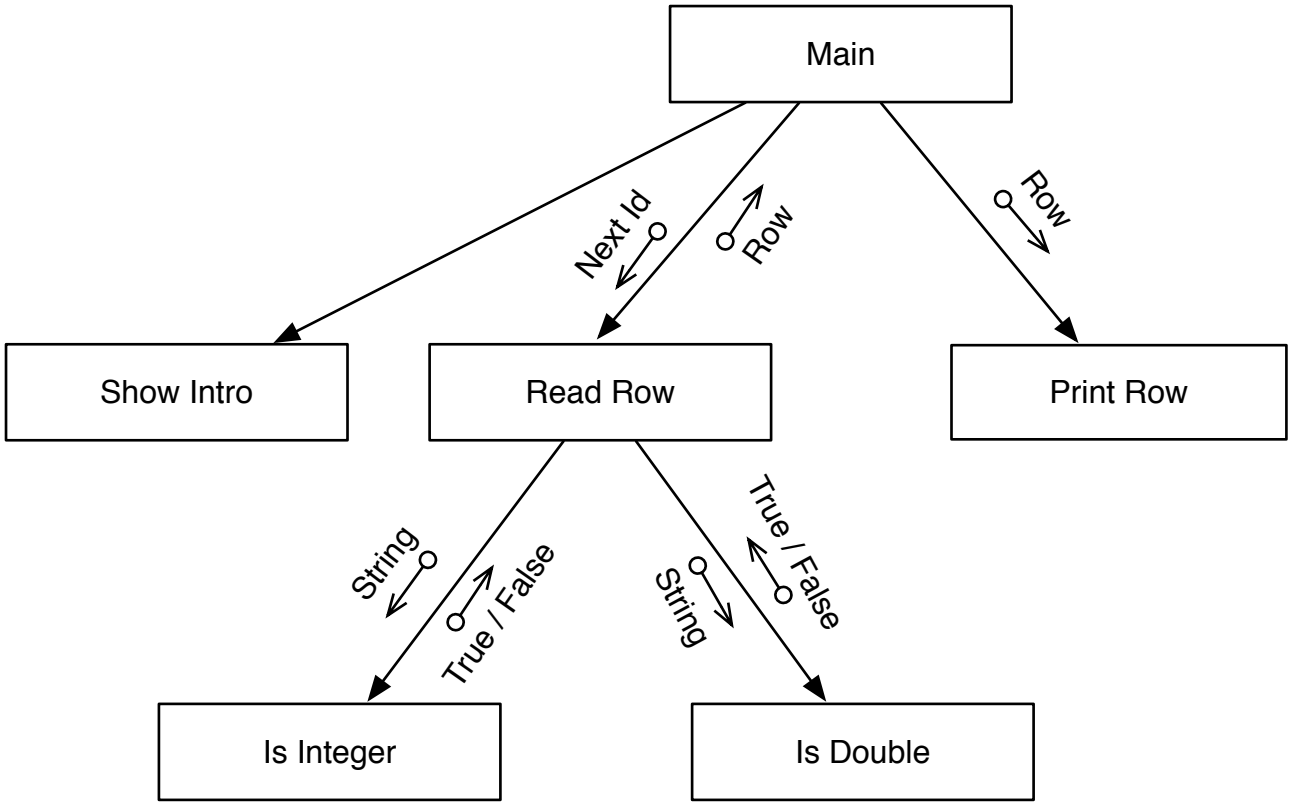
The right hand side of the Assignment is an Expression, the value that is to be stored

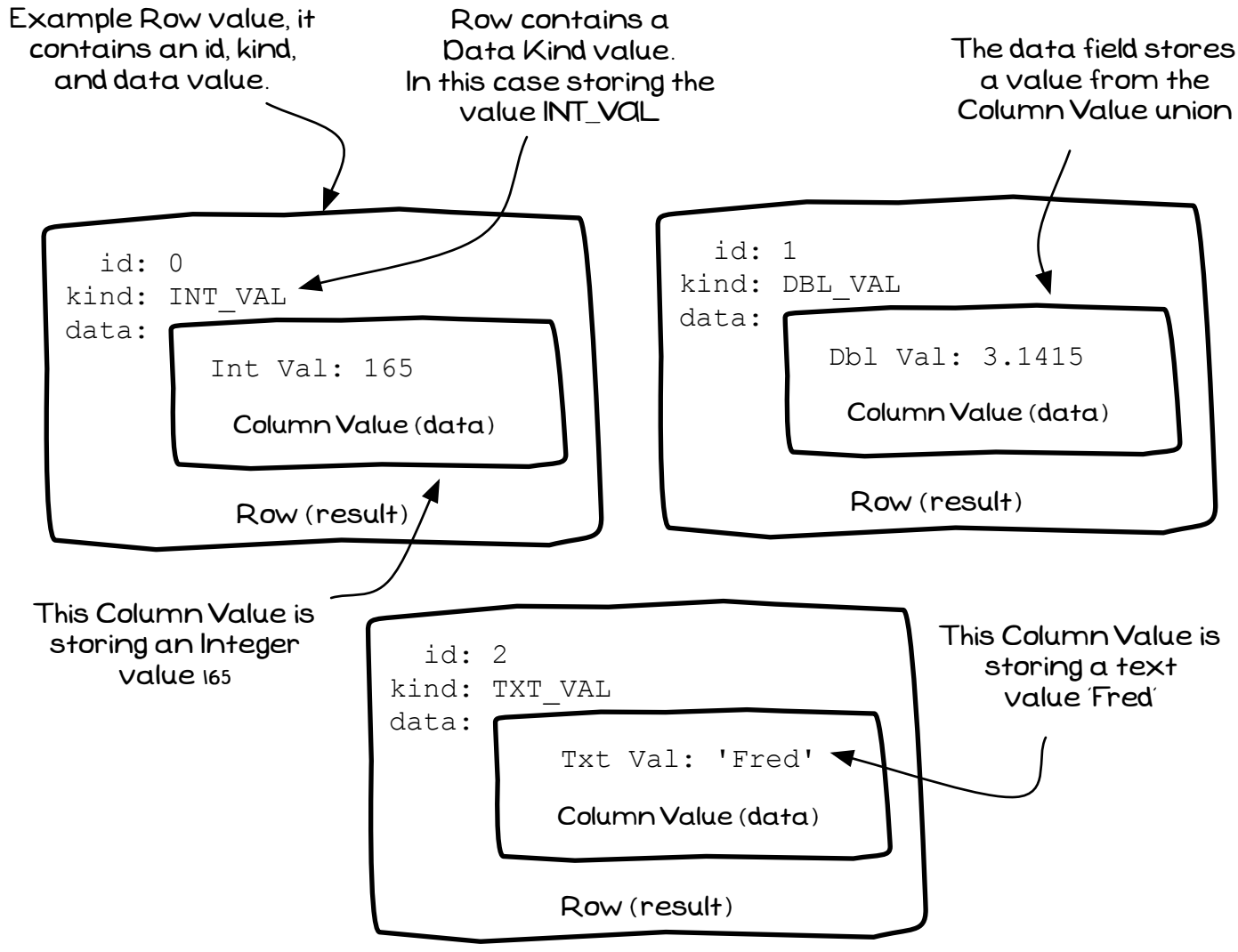

24
x = ...
y = 73
Point
Variable
(Mouse)
stored in
24
mouse.x
+ =
y =
24
73
Point
Variable
(other)
stored in
x = 24
y = 73
Point
mouse
other
52
x = ...
y = 37
Point
Array
(Points)
stored in
52
points[2].x
x = 24
y = 73
Point
Array
(Points)
stored in
x = 24
y = 73
Point
mouse
points[2]

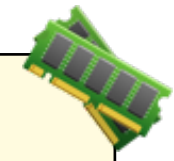

intVal = . . .
User Data
73
Variable
(result)
stored in
73
result.intVal
intVal = 73
User Data
Variable
(other)
stored in
intVal = 73
User Data
result
other
txtVal = "Fred"
User Data
dbVal = 3.1
User Data
intVal = . . .
User Data
52
Array
(Points)
stored in
52
points[2].x
txtVal = "Fred"
User Data
dbVal = 3.1
User Data
intVal = 73
User Data
Array
(Points)
stored in
intVal = 73
User Data
other
points[2]


Read Row
Assign result.id the value of Next Id
Read the line entered by the user at the Terminal into line
if the value in line is an Integer
No
Yes
Assign result.data.int val the integer value from line
Assign result.kind the value INT_VAL
if the value in line is a Double
No
Yes
Assign result.data.txt val the value from line
Assign result.data.dbl val the double value from line
Assign result.kind the value TXT_VAL
Assign result.kind the value DBL_VAL
Output "Stored in row with id " and the id value from the result row
Return result
End


Print Row
Output "Row with id " and the id of the row To Print
To Print's Kind
INT_VAL
DBL_VAL
TXT_VAL
Output "has integer " and the Int Val of To Print's data
Output "has double " and the Dbl Val of To Print's data
Output "has text " and the Txt Val of To Print's data
Output "has an unknown value"
End


Main
Next Id
Row
Row
Show Intro
Read Row
Print Row
String
True / False
String
True / False
Is Integer
Is Double


Example Row value, it contains an id, kind, and data value.
Row contains a Data Kind value. In this case storing the value INT_VAL
The data field stores a value from the Column Value union
id: 0
kind: INT_VAL
data:
Int Val: 165
Column Value (data)
Row (result)
id: 1
kind: DBL_VAL
data:
Dbl Val: 3.1415
Column Value (data)
Row (result)
This Column Value is storing an Integer value 165
id: 2
kind: TXT_VAL
data:
Txt Val: 'Fred'
Column Value (data)
Row (result)
This Column Value is storing a text value 'Fred'



Main is loaded onto the Stack, with its Local Variable

1

Main

i: 79123

db_data[]:

id:	7139
kind:	UNK_VAL
data.int_val:	10723
data.dbl_val:	
data.txt_val:	

id:	-91823
kind:	UNK_VAL
data.int_val:	36126
data.dbl_val:	
data.txt_val:	

id:	0
kind:	UNK_VAL
data.int_val:	-73
data.dbl_val:	
data.txt_val:	

Instruction: Step 1

```
Function: Read Row
---------------------------
Returns: Row - a Row with data read from the user
Parameters:
 1: next id (Integer) - the id of the row to be read
Local Variables:
 *: line (String - 16 characters) - the text read from the user
Steps:
  1: Set result's id to next id
  2: Output 'Enter value: ' to the Terminal
  3: Read text entered by user into line
  4: If line is an integer
  5:      set result's data's Int Val to the integer value in line
  6:      set result's kind to INT_VAL
  7: Else If line is a double
  8:      set result's data's Dbl Val to the double value in line
  9:      set result's kind to DBL_VAL
 10: Else
 11:      set result's data's Txt Val to the text in line
 12:      set result's kind to TXT_VAL
 13: Output "Stored in row with id ", and result's id
 14: Return the result

Procedure: Main
---------------------------
Local Variables:
 *: db_data (array containing 3 Row values)
 *: i (Integer) -
Steps:
  1: for i loops over each element in db_data
  2:      set db_data[i] to result of calling Read Row(i)
  3: ...
```

3

Each Row has an id, kind, and data

4

Each of the Column Value fields stores either int_val, dbl_val or txt_val

2

Each element of the db_data array is a Row

```
Procedure: Print Row
----------------------------
Parameters:
 1: to print (Row) - the row to print
Steps:
  1: Output 'Row with id ', and to print's id to the Terminal
  2: select case from to print's kind
  3:     case is INT_VAL
  4:         Output ' has integer ' and int_val of to print's data
  5:     case is DBL_VAL
  6:         Output ' has double ' and dbl_val of to print's data
  7:     case is TXT_VAL
  8:         Output ' has text ' and txt_val of print's data

Procedure: Main
----------------------------
Local Variables:
 *: db_data (array containing 3 Row values)
 *: i (Integer) -
Steps:
  2: ...
  3: for i loops over each element in db_data
  4:     call Print Row( db_data[i] )
```