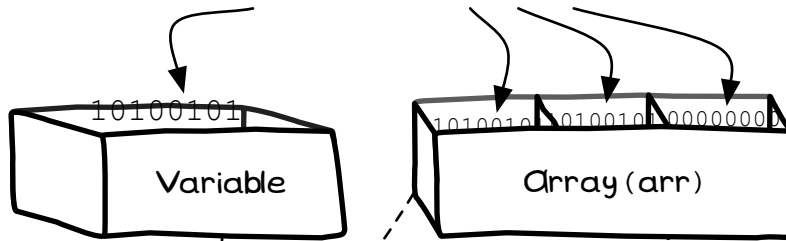




Variables store one or more values



Values exist in many locations in the code

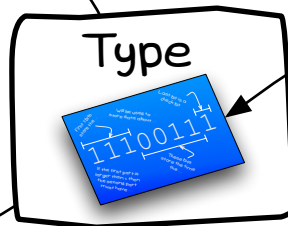
Expressions calculate a value

`my_rect.x + my_rect.width`

In the computer values are binary sequences

10100101

Types give meaning to values

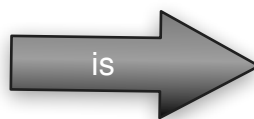


A Type is a blueprint describing how data is stored and interpreted

Type describes the size of the value (number of bits)

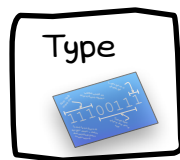
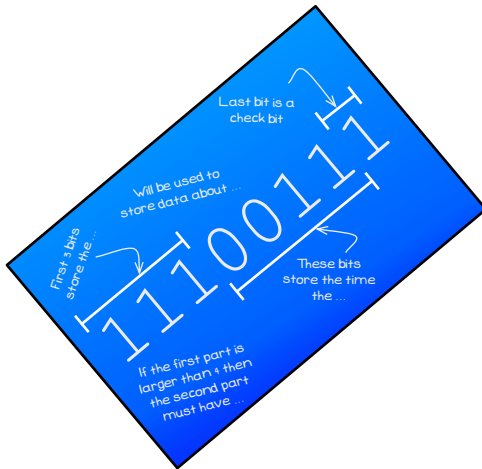
10100101

Type defines what the different values mean



$\tilde{N}$  - if its a Character  
165 - if its an Integer

A Type is a specification



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

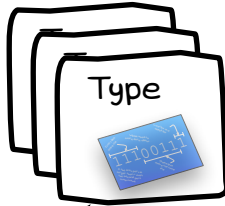
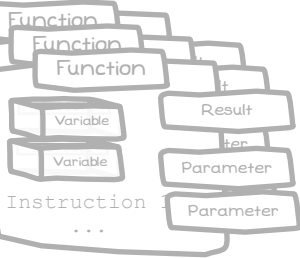
Type

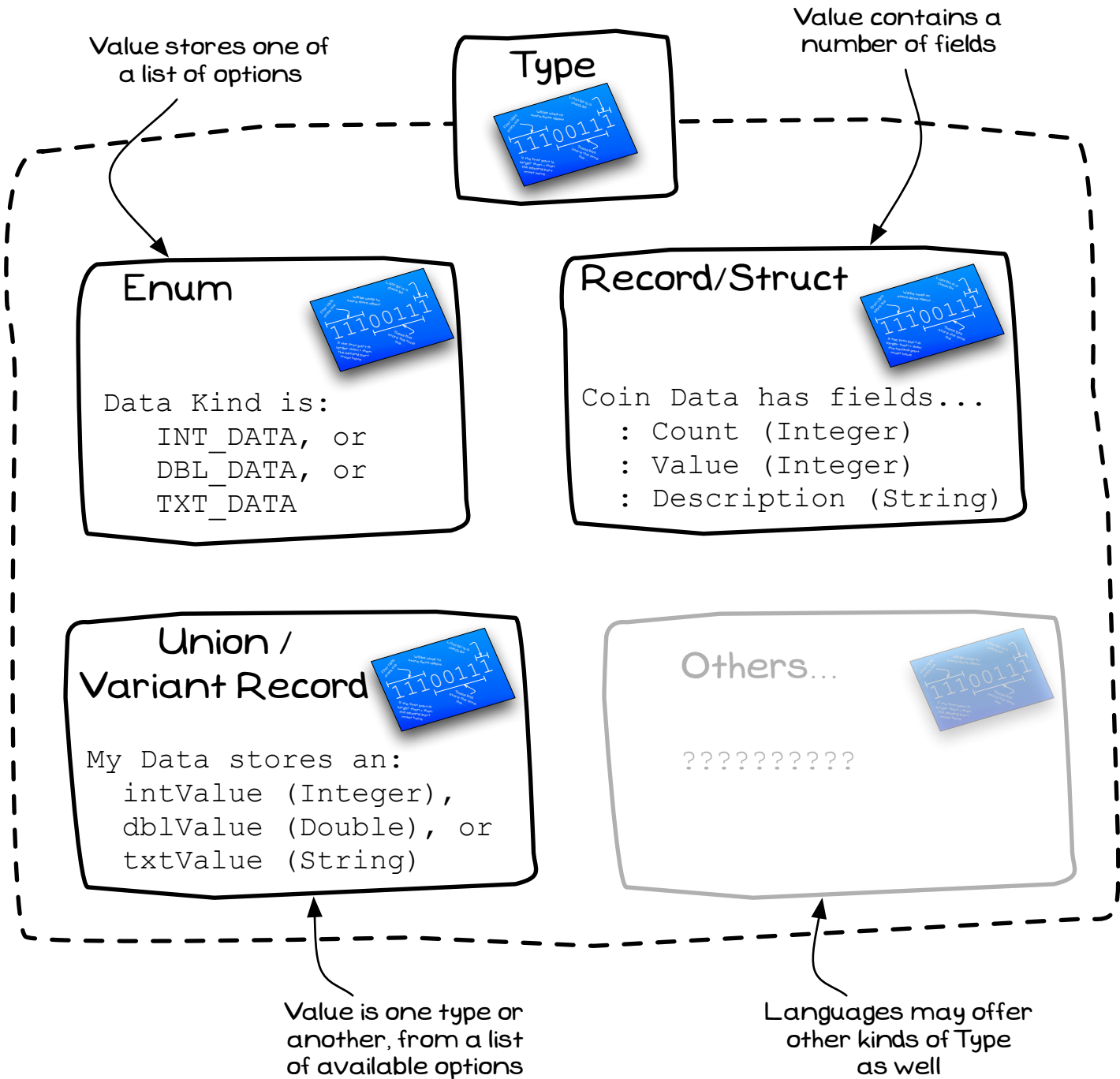
Instruction 1  
Instruction 2  
...

The Program has an entry point that indicates where the program's instructions start

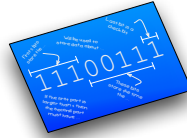
A Program may declare its own Types.

A Program may declare its own Functions.



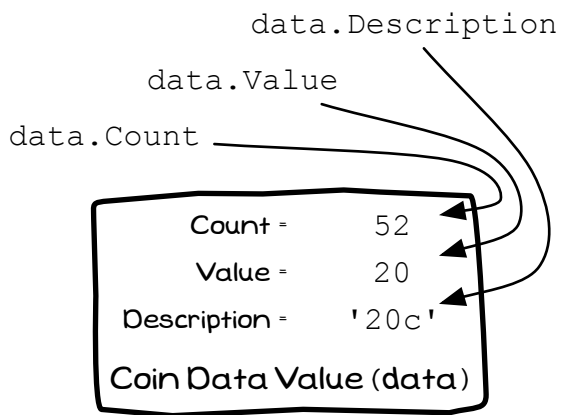


## Record/Struct

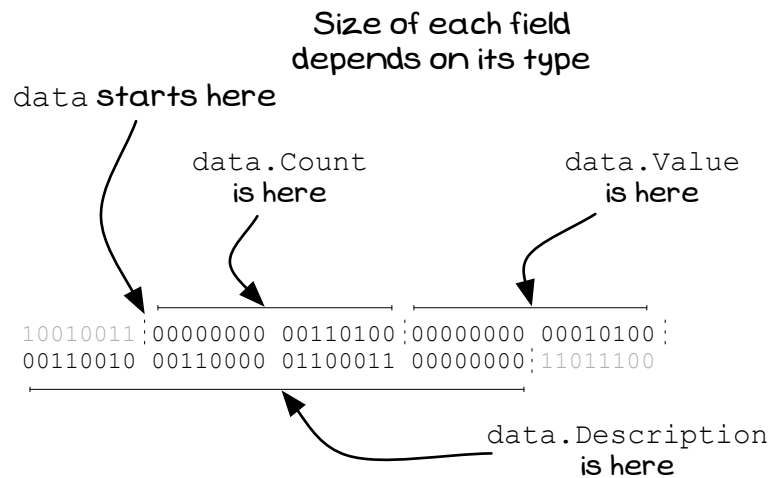


Coin Data has fields...

- : Count (Integer)
- : Value (Integer)
- : Description (String)

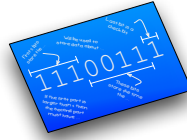


Conceptually



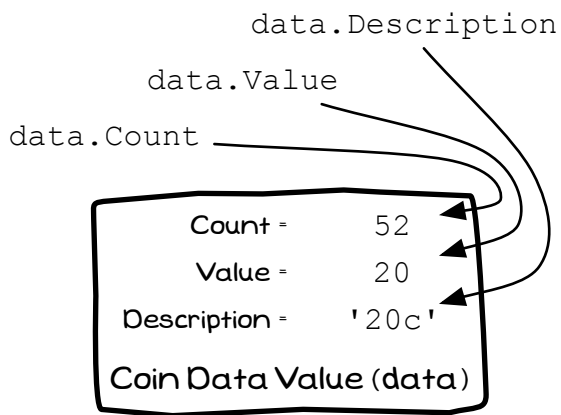
In Memory

## Record/Struct

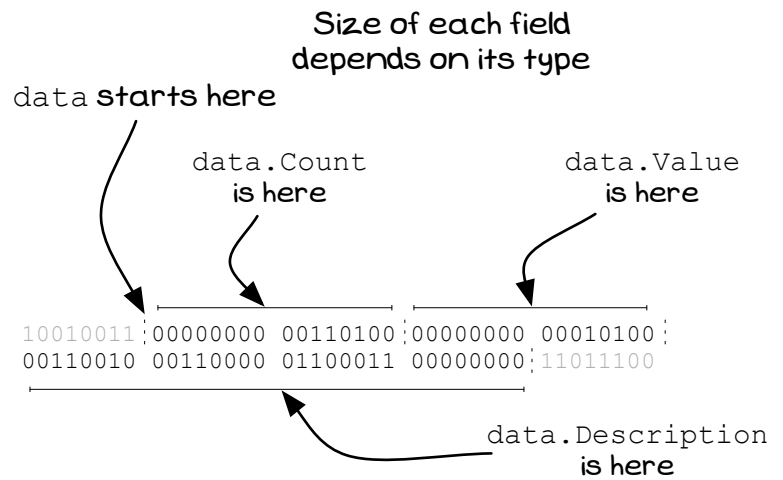


Row has fields...

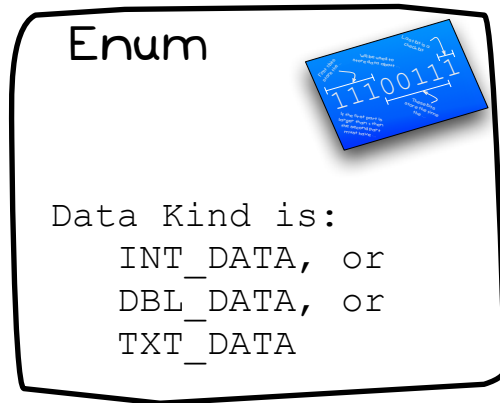
- : Id (Integer)
- : Kind (Data Kind)
- : Value (Column Value)



Conceptually



In Memory



Each value is either  
INT\_DATA, DBL\_DATA, or TXT\_COINS



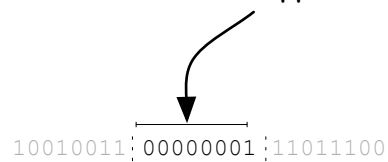
Conceptually

Each option is mapped  
to a numeric value.

INT\_DATA  
is mapped to 0

DBL\_DATA  
is mapped to 1

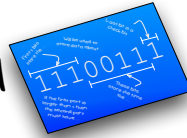
TXT\_DATA  
is mapped to 2



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

In Memory

## Union / Variant Record

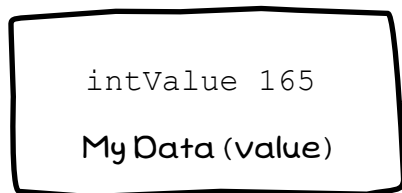


My Data stores an:

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



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

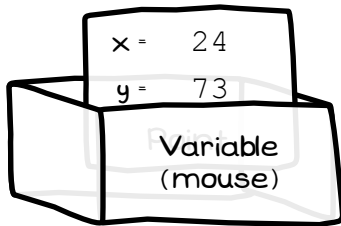
In Memory



Is a value.

Values can be read from variables of the different custom types using an Expression

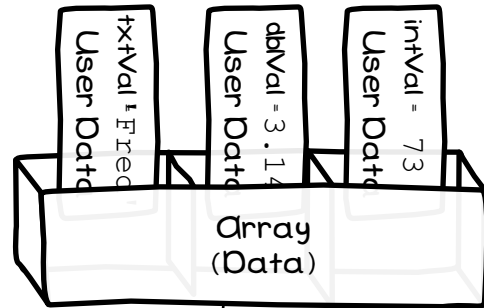
Expression



Read value from the field of a Record Variable

00011000

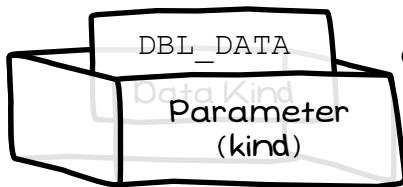
`mouse.x`



Read value from the field of a Union Variable

01001001

`data[2].intVal`

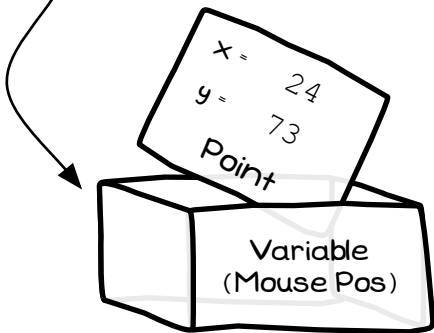


Read value of a Enumeration Variable

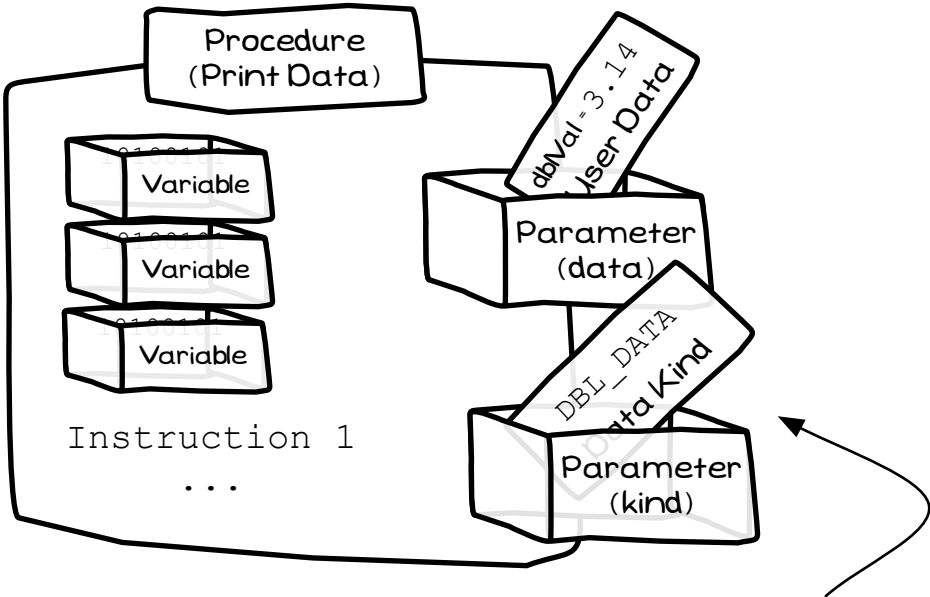
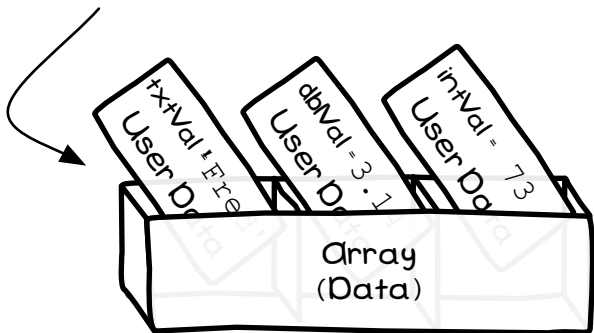
00000001

`kind`

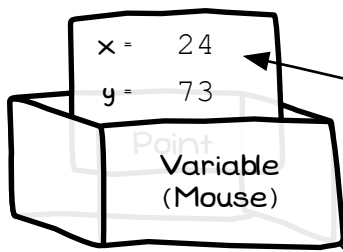
Values can be records like a Point record with X and Y coordinates



Elements of an array can be values of a union, allowing the one array to store integer, double, and text data in different elements



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)

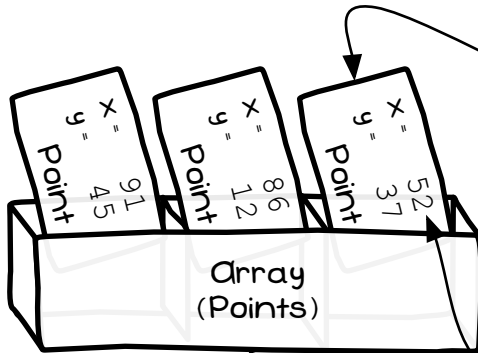


`mouse.x`

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]`

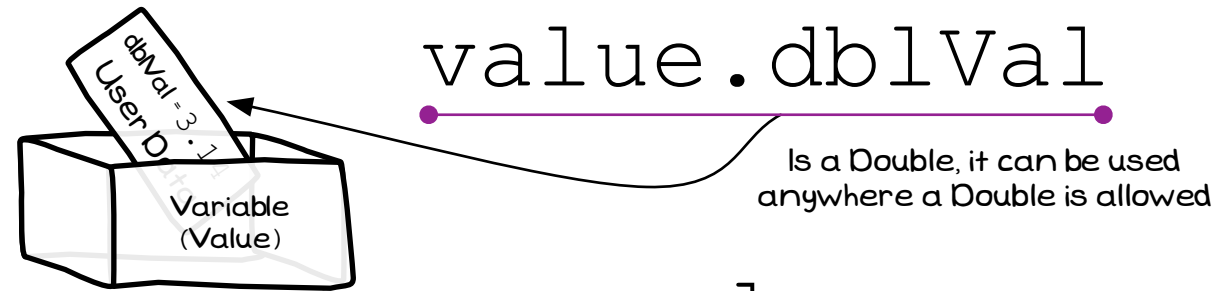
Is a Point, it can be used anywhere a Point is required

`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

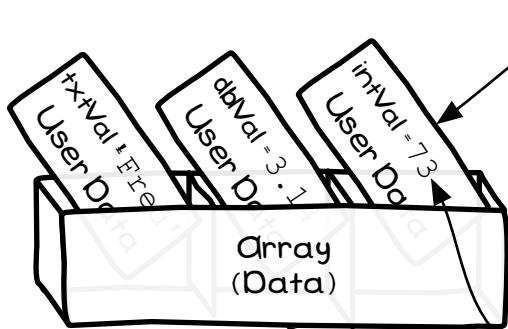


value.dblVal

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



data[2]

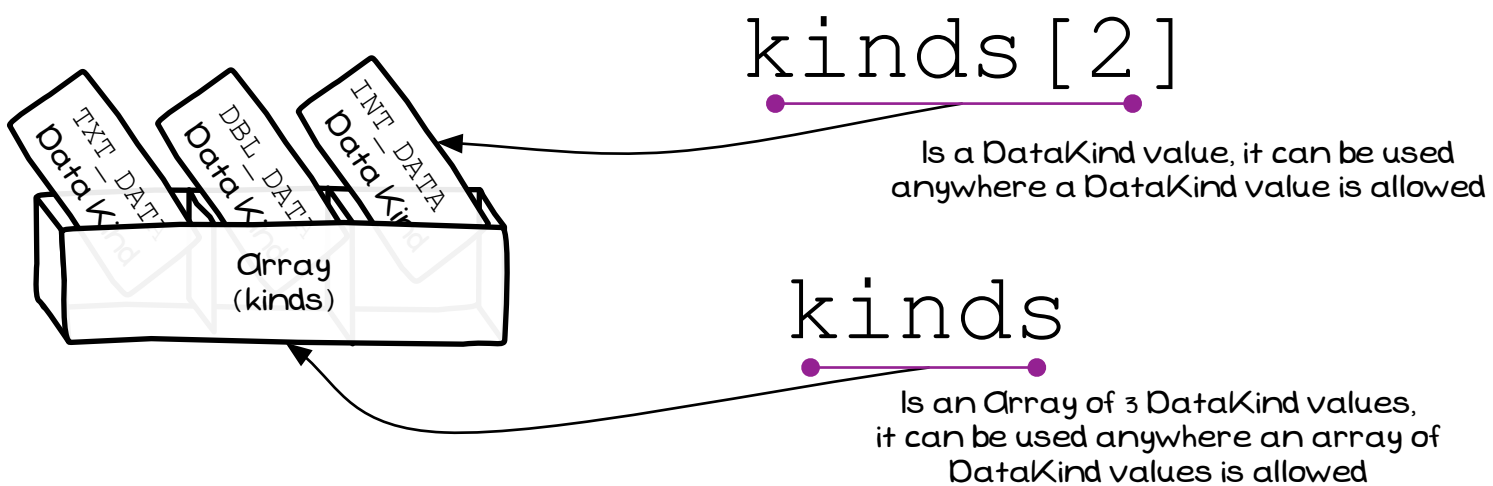
Is a UserData value, it can be used  
anywhere a UserData value is allowed

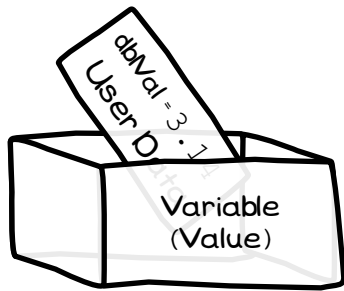
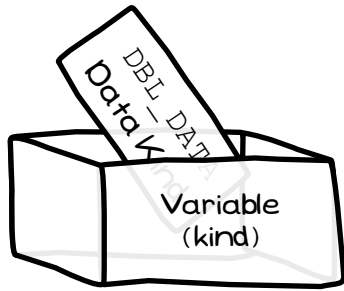
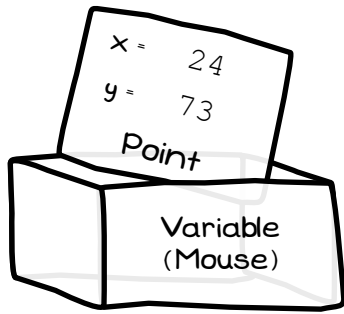
data[2].intVal

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

data

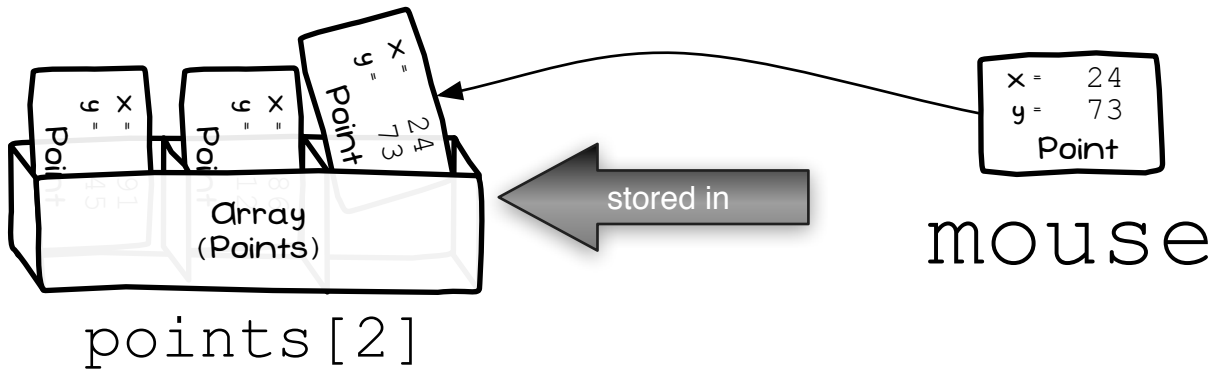
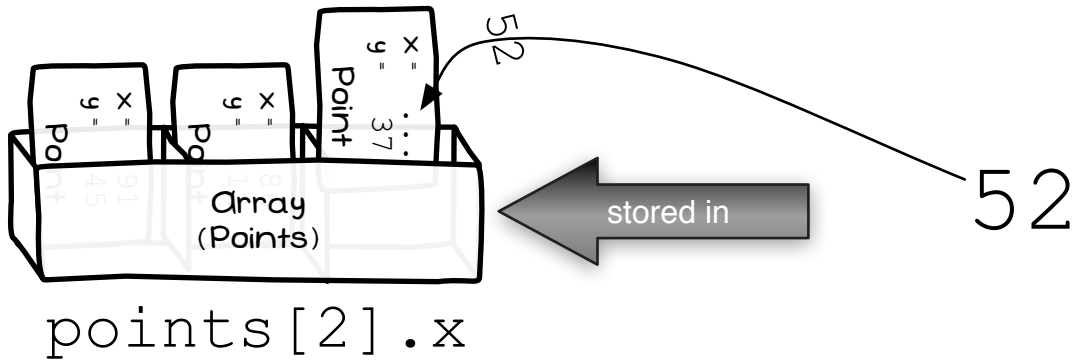
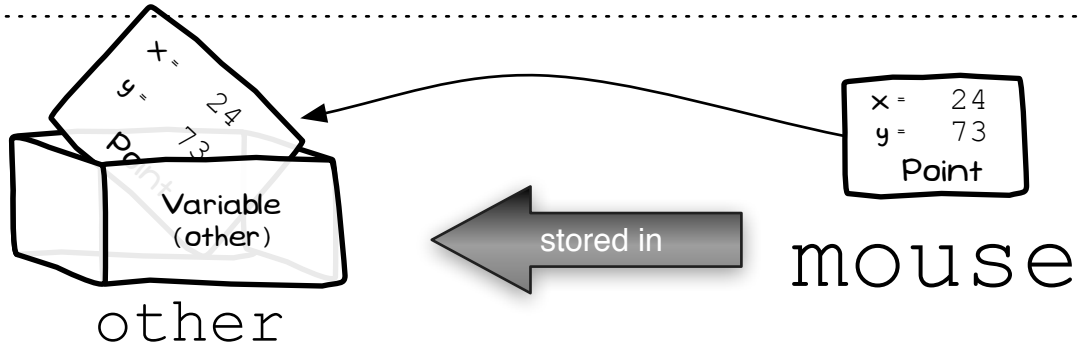
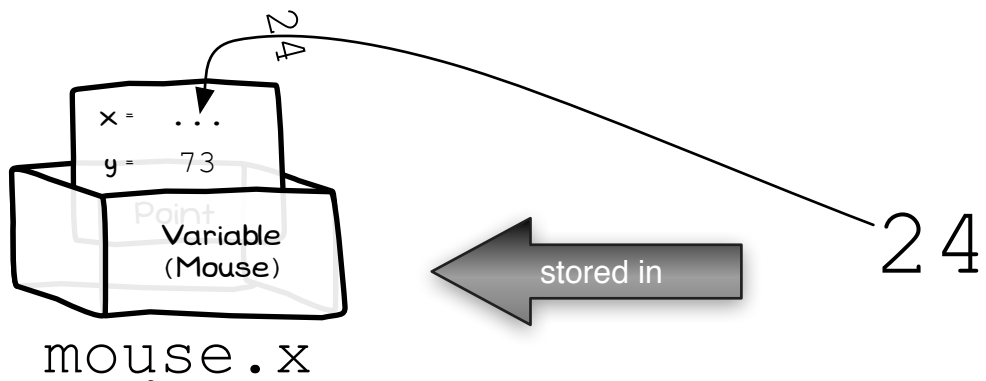
Is an Array of 3 UserData values,  
it can be used anywhere an array of  
UserData values is allowed

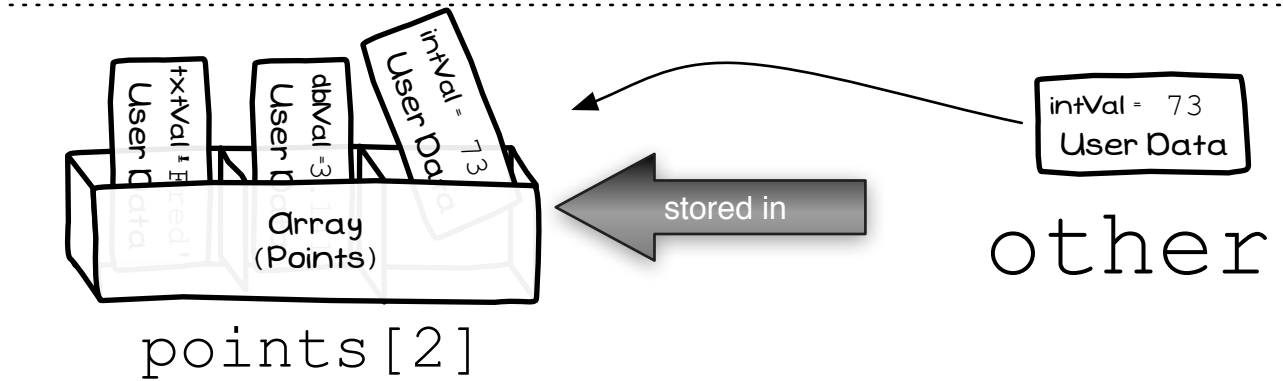
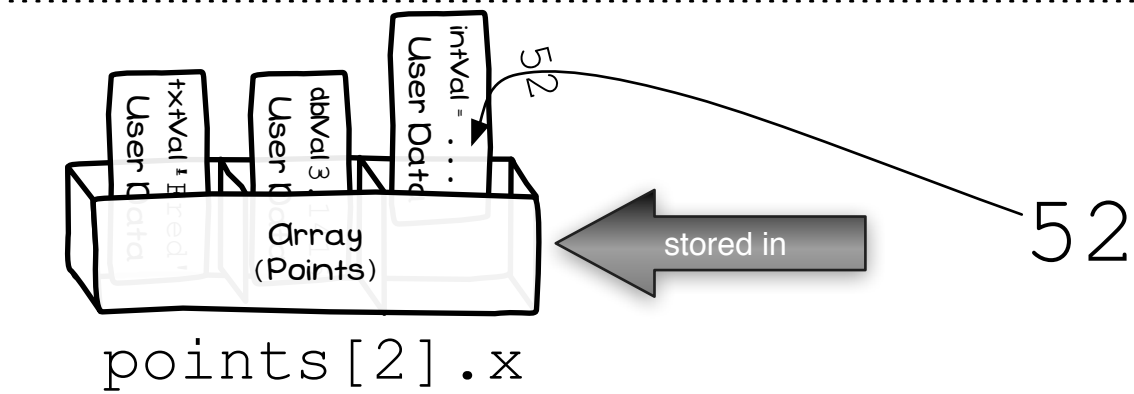
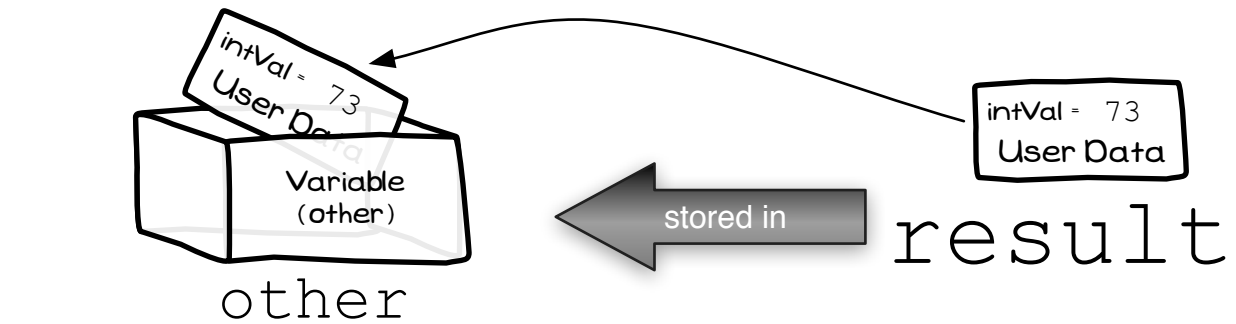
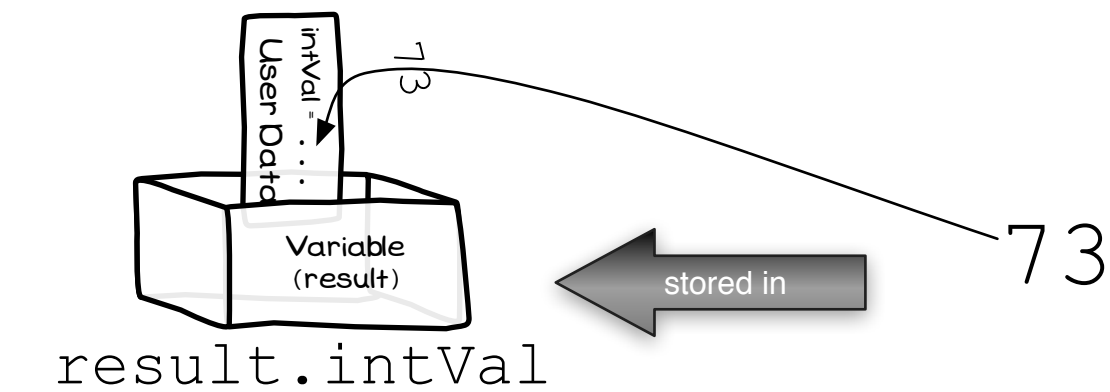




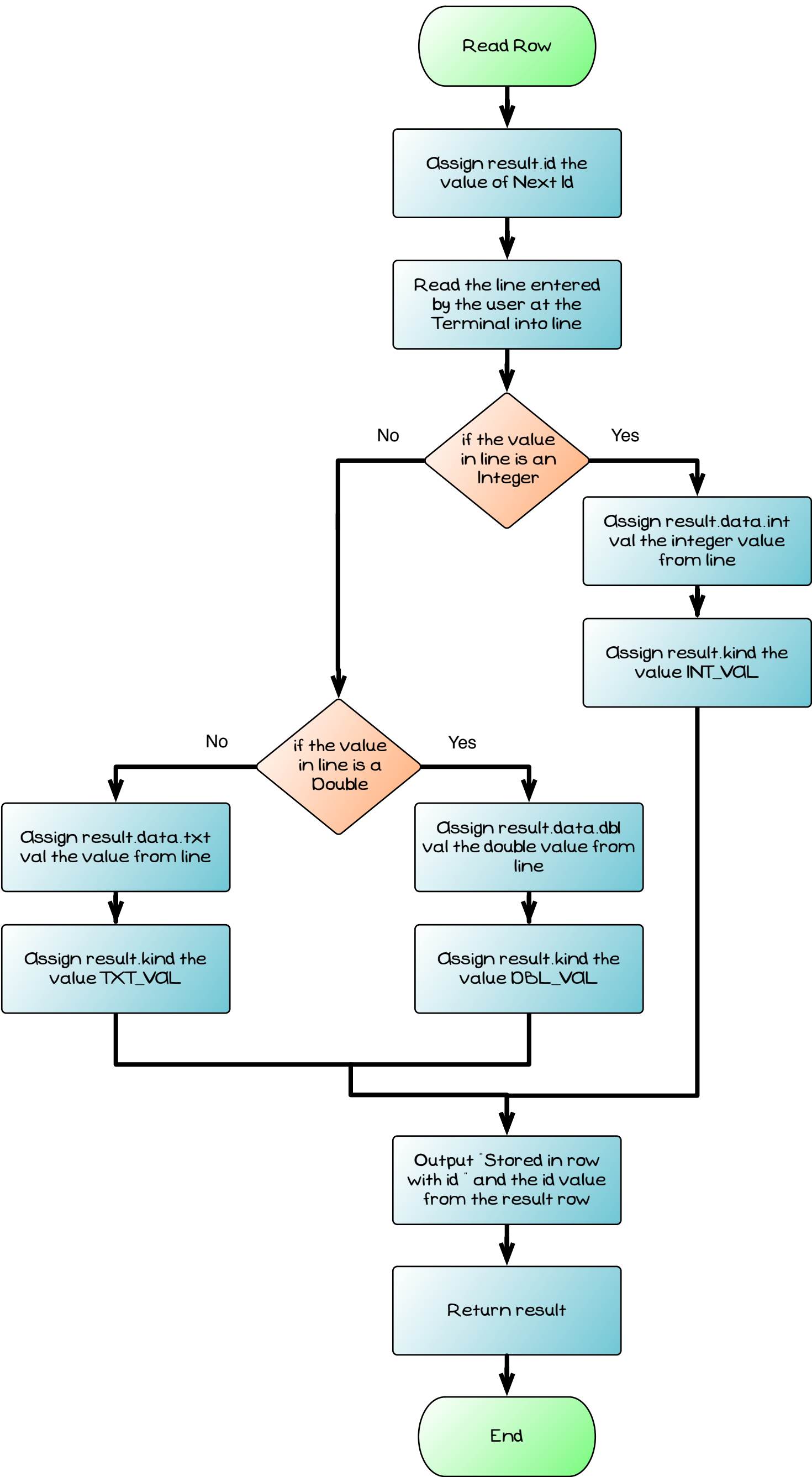
Expression

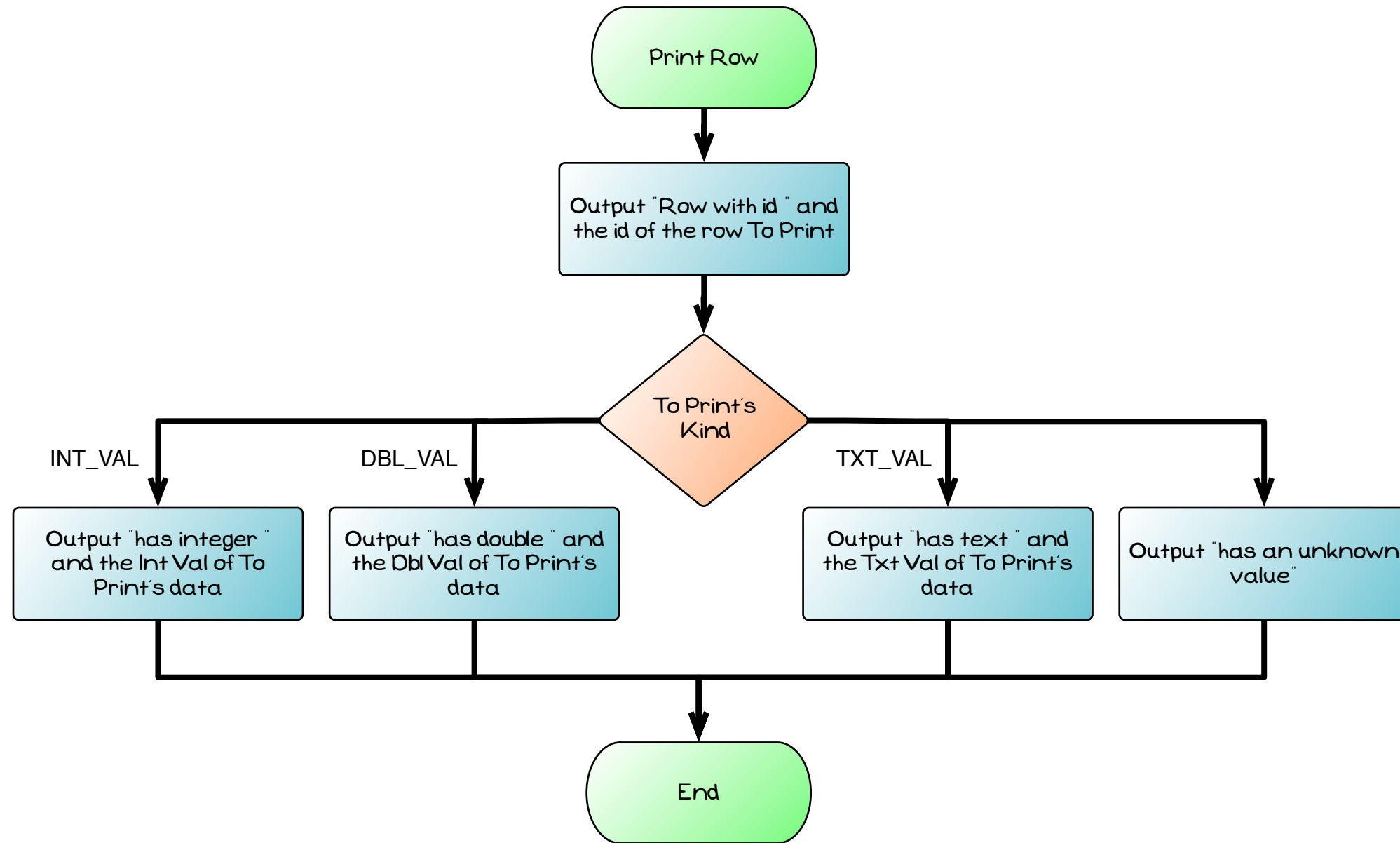
The right hand side of the Assignment is an Expression, the value that is to be stored

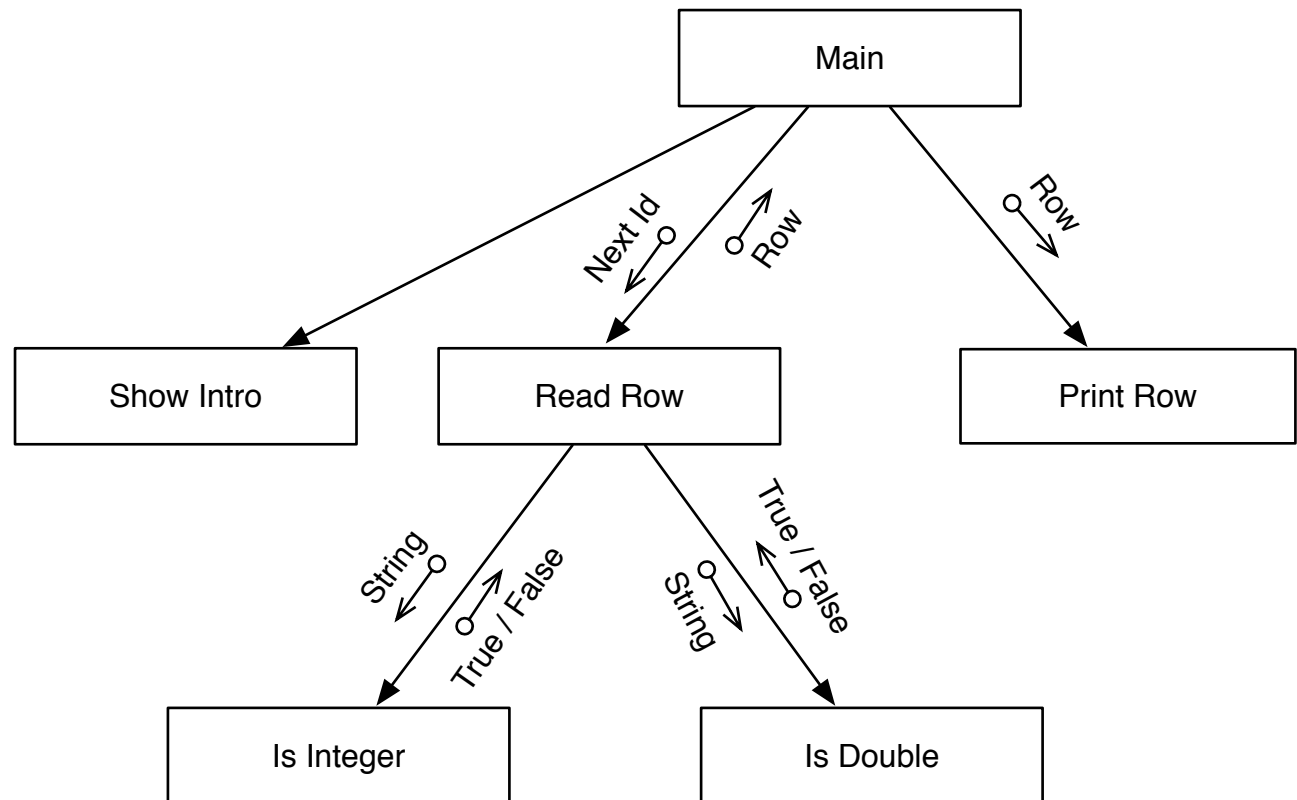








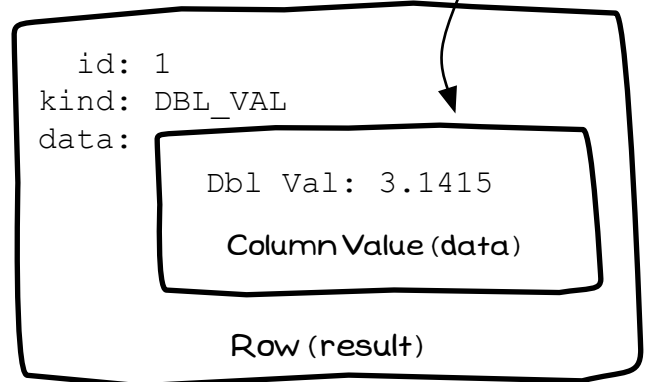
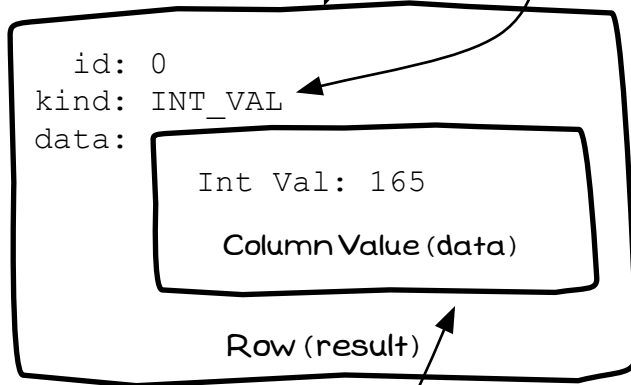




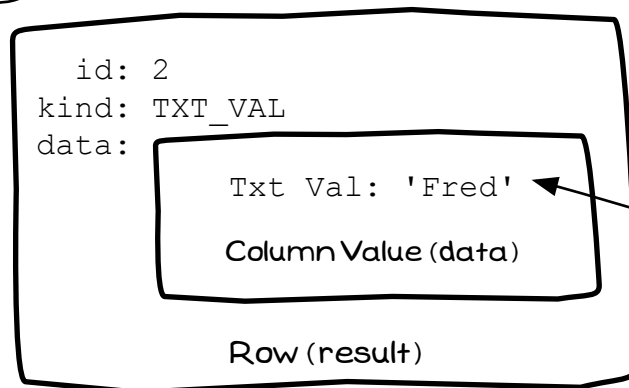
Example Row value, it contains an id, kind, and data value.

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

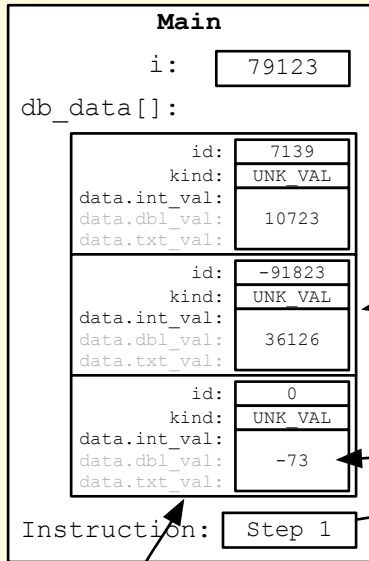


This Column Value is storing an Integer value 165



This Column Value is storing a text value 'Fred'

1 Main is loaded onto the Stack, with its Local Variable



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] )

