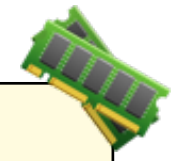


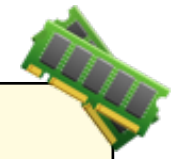
4 Heap

Stack
2

Code Area
1

3 Global Variables

The Program's Memory

Gets access to space into which you can store a value (a record in this case)

| | |
|---|---|
| id: | 2 |
| kind: | TXT_VAL |
| data.int_val: | |
| data.dbl_val: | |
| data.txt_val: | Fred |

Returns the space so that it can be allocated to something else

```
Function: Read Row
-------------------------
Returns: Row - 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: ...
```

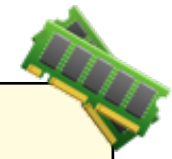
Allocate Space

Free Space

**Func or Proc**

val: 5

Instruction: Step 2

| 3 | . | 1 | 4 | 1 | 5 | □ |
|---|---|---|---|---|---|---|

| | |
|---|---|
| id: | 63 |
| kind: | DBL_VAL |
| data.int_val: | |
| data.dbl_val: | 3.1415□ |
| data.txt_val: | |

| |
|---|
| 10.0 |
| -5 |
| 17.21 |
| 25.1 |

| | |
|---|---|
| id: | 2 |
| kind: | TXT_VAL |
| data.int_val: | |
| data.dbl_val: | Fred□ |
| data.txt_val: | |

?

INT_VAL

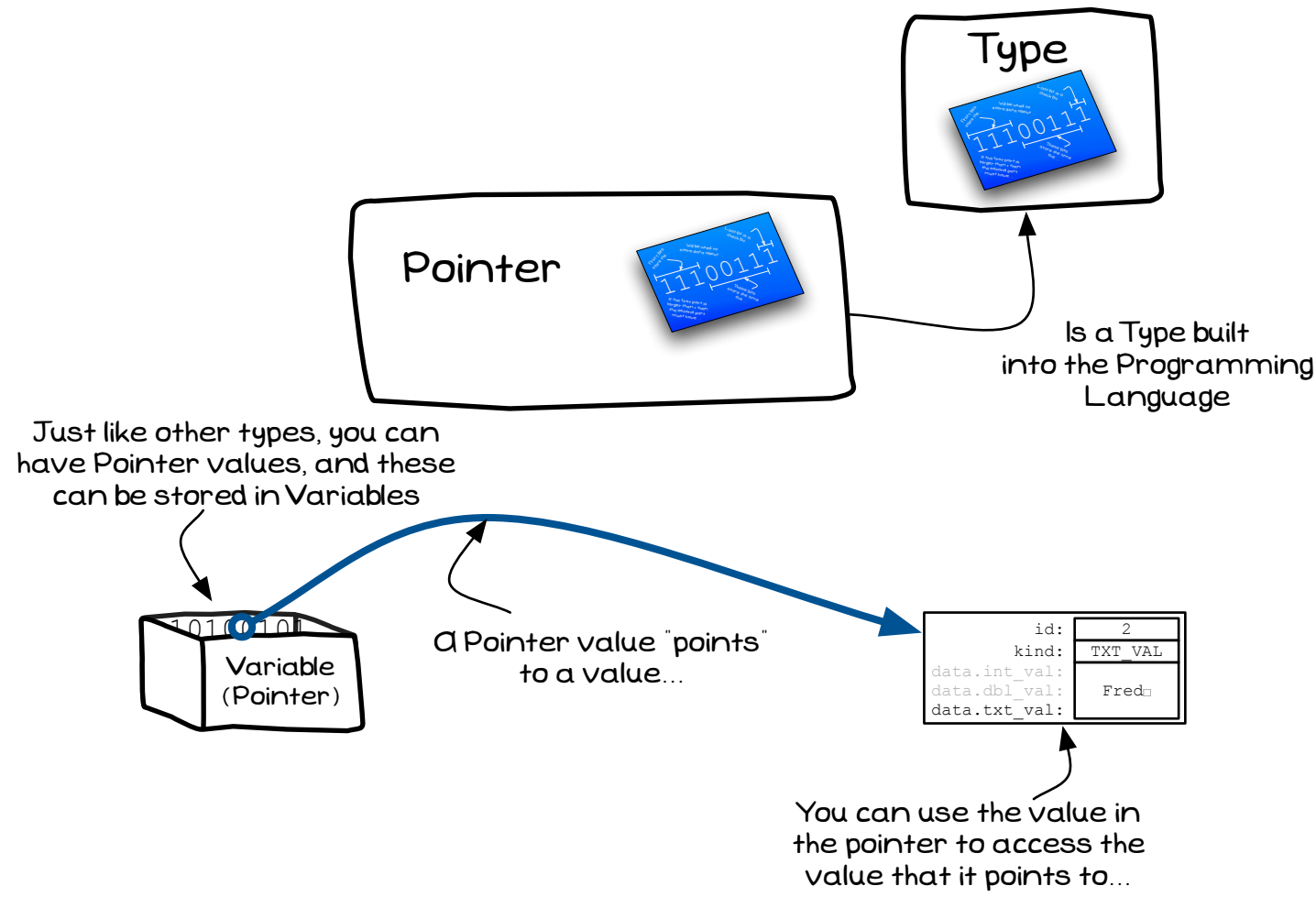
73

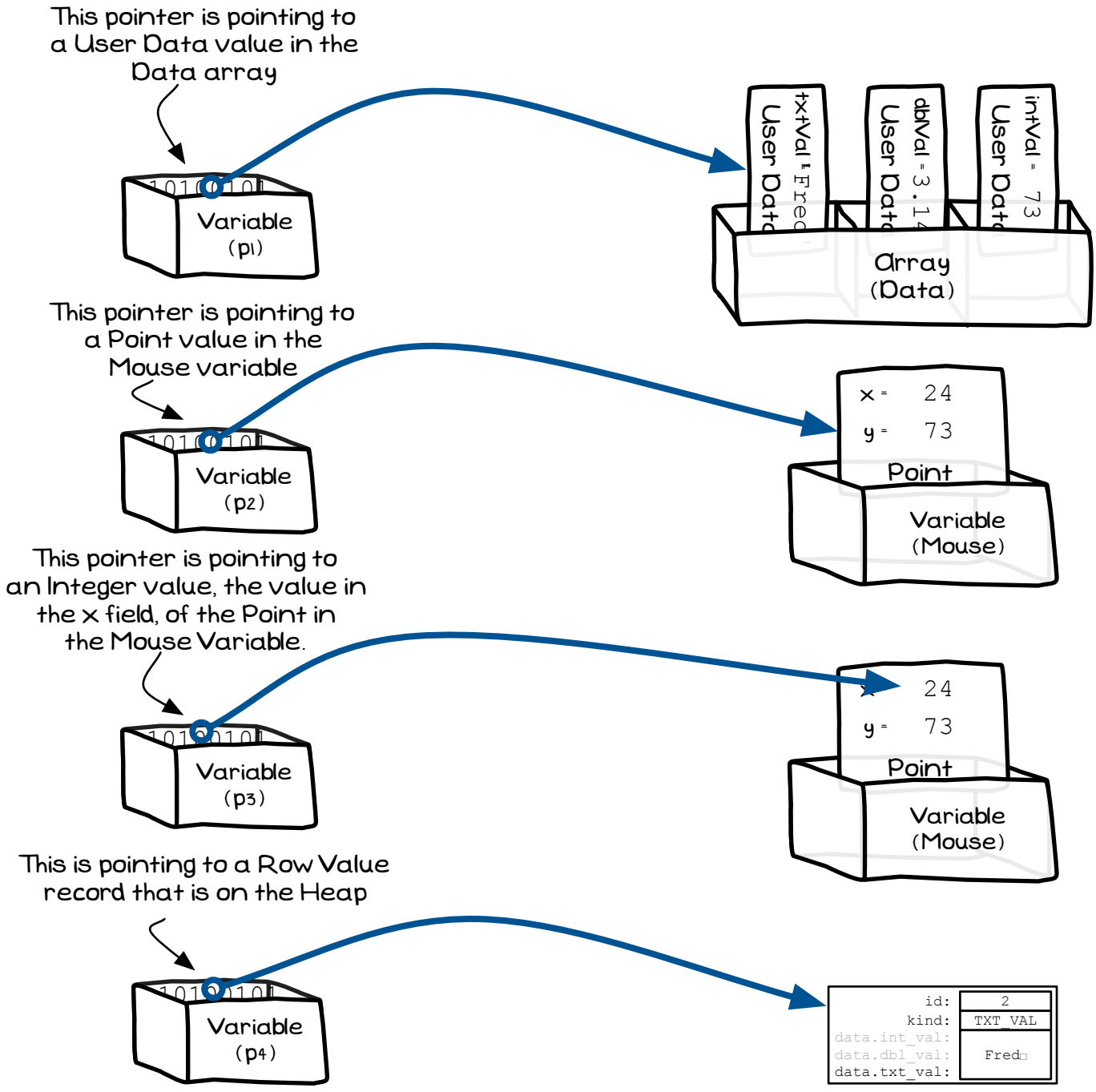
**Func or Proc**

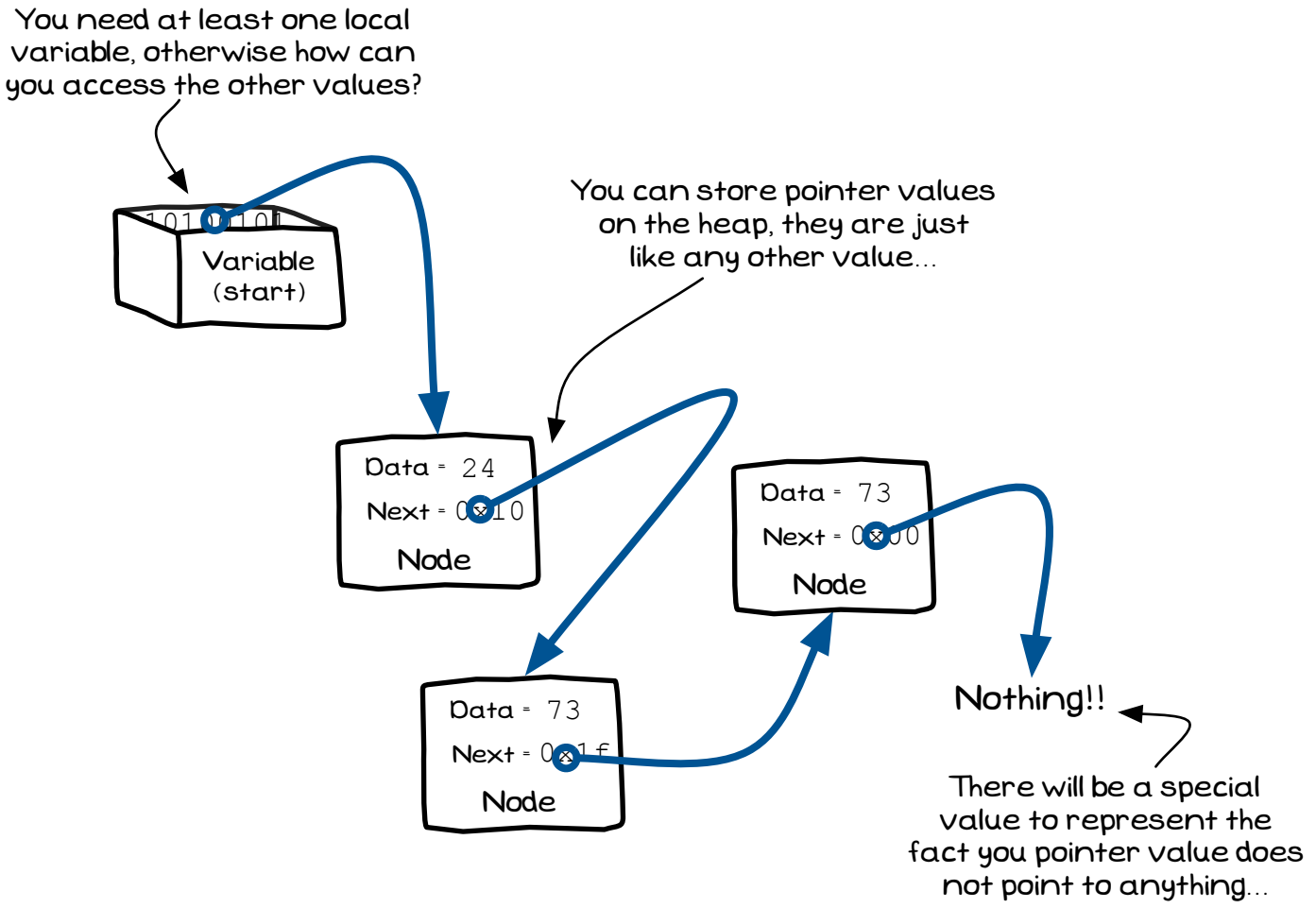
val: 5

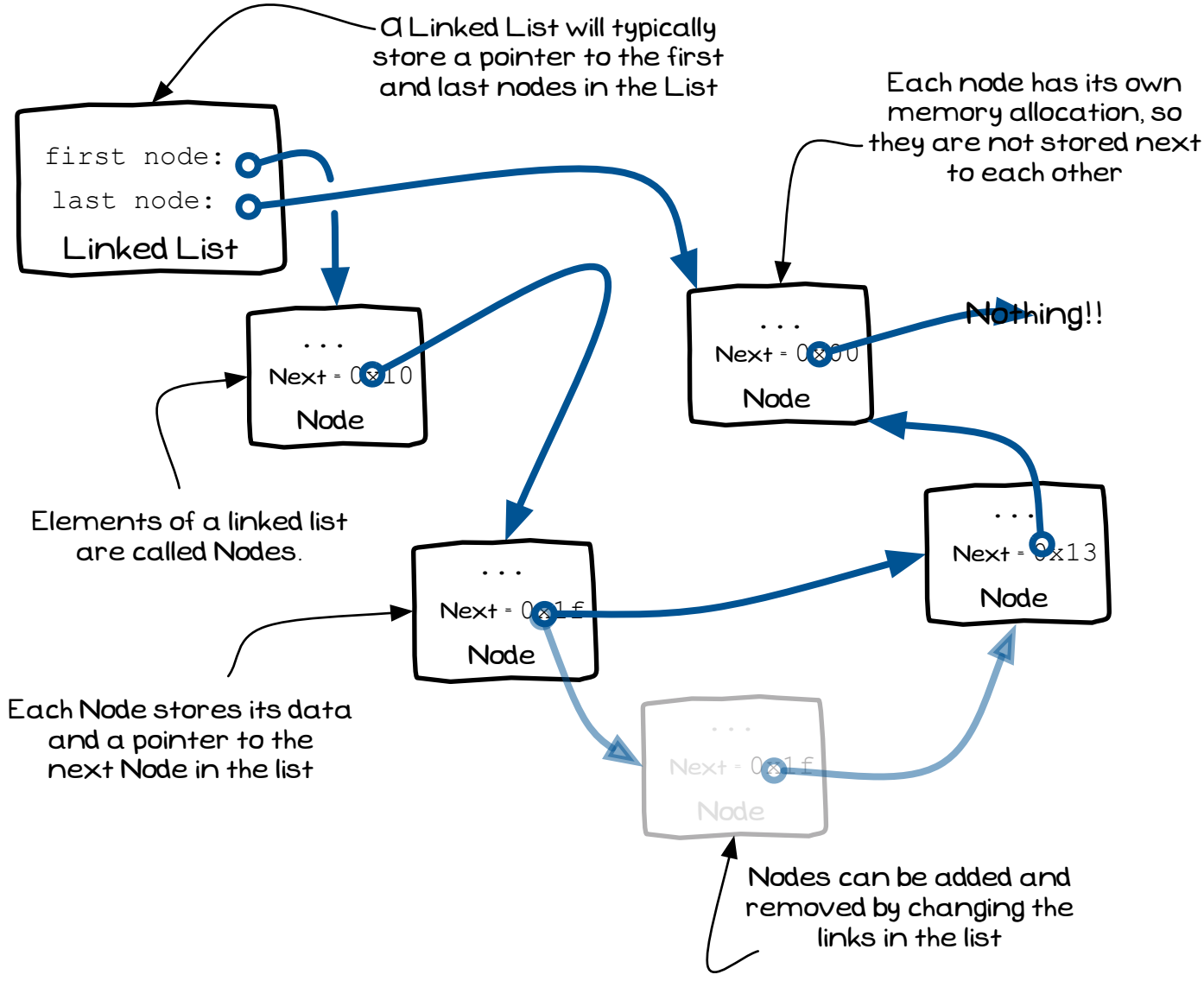
Instruction: Step 2

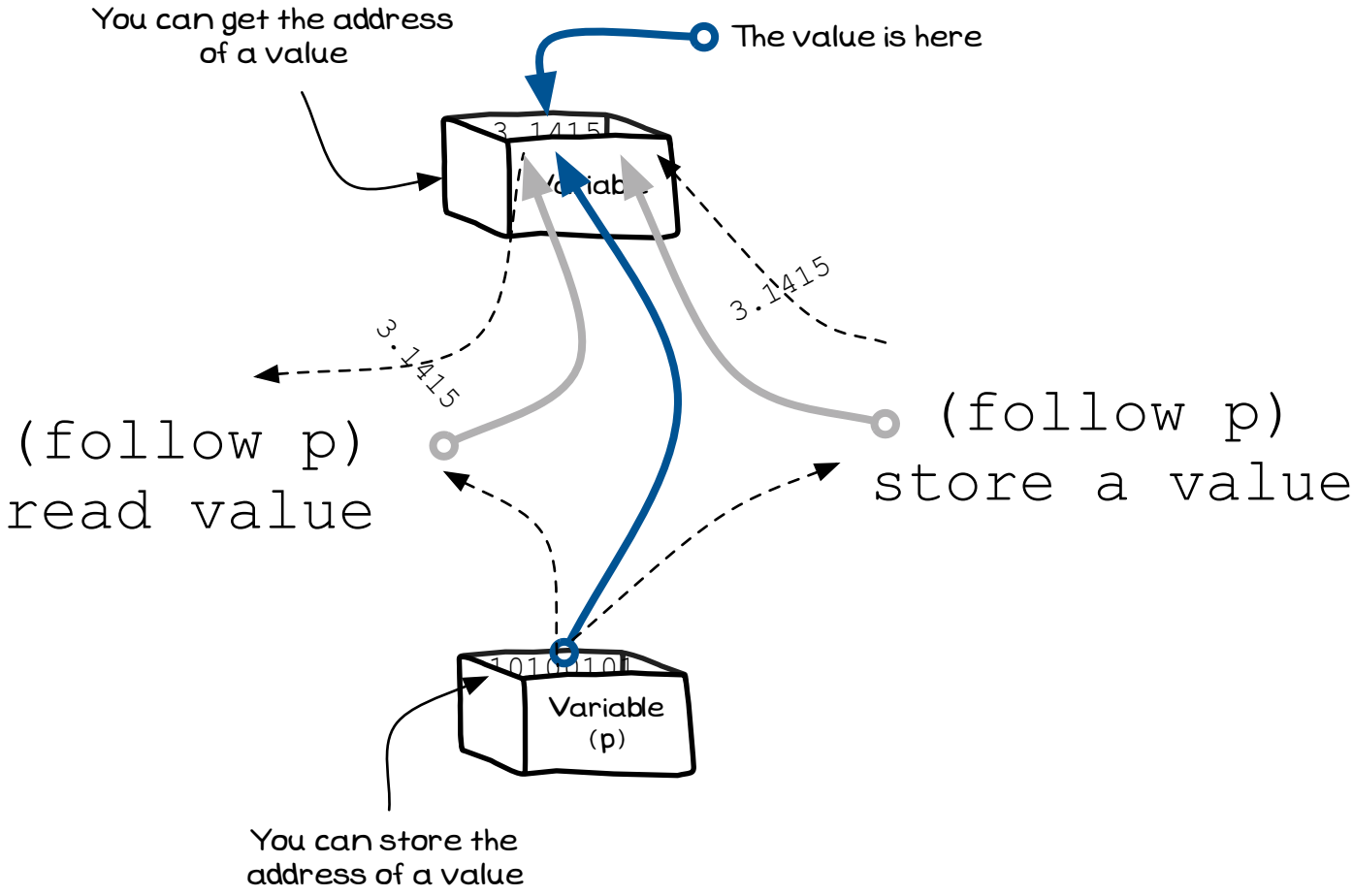
```
Function: Read Row
------------------------------
Returns: Row - a Row with data read from the user
Parameters:
 1: next id (Integer) - the id of the row to be read
Local Variables:
 *: line (String - 16 characters) - the text read from the user
Steps:
  1: Set result's id to next id
  2: Output '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: ...
```

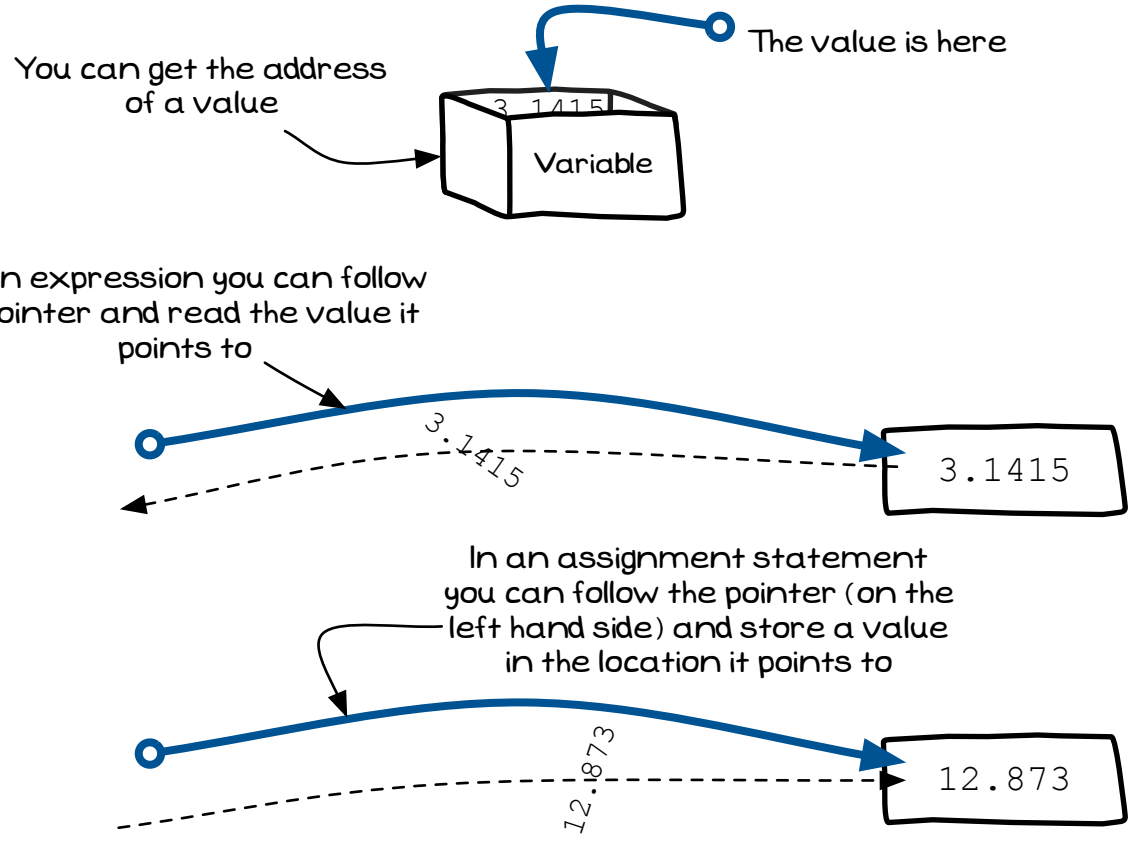


Type
Pointer
Is a Type built into the Programming Language
Just like other types, you can have Pointer values, and these can be stored in Variables
Variable (Pointer)
A Pointer value "points" to a value...
id: 2
kind: TXT_VAL
data.int_val:
data.dbl_val:
data.txt_val: Fred
You can use the value in the pointer to access the value that it points to...

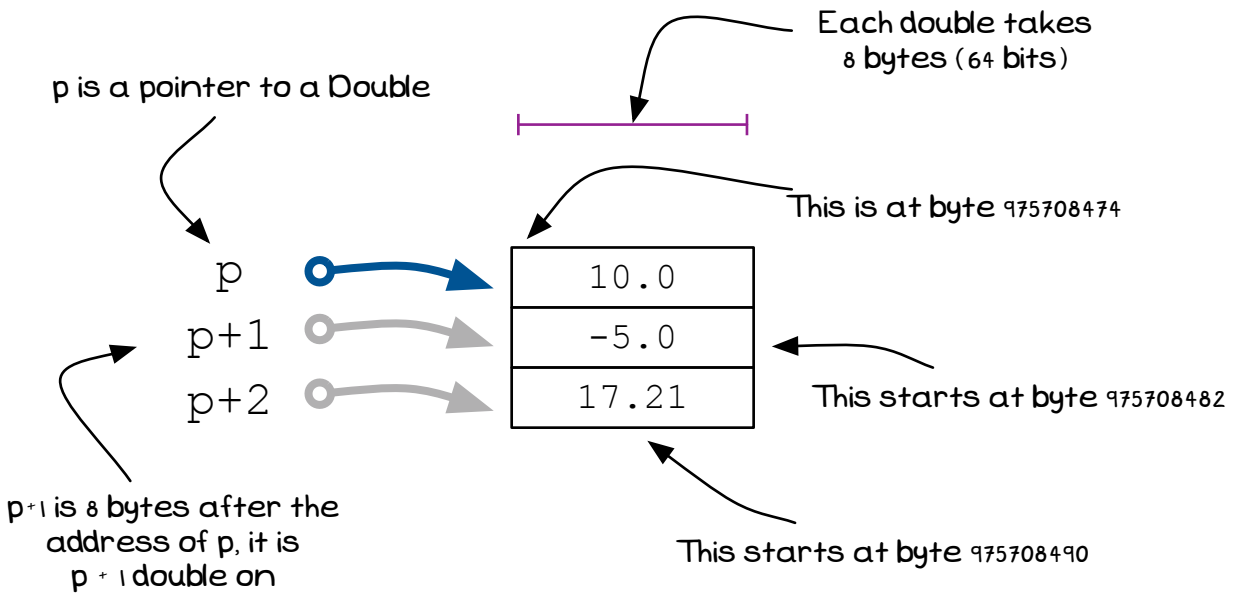

This pointer is pointing to a User Data value in the Data array
Variable (p1)
User Data
User Data
User Data
Array (Data)
This pointer is pointing to a Point value in the Mouse variable
Variable (p2)
x = 24
y = 73
Point
Variable (Mouse)
This pointer is pointing to an Integer value, the value in the x field, of the Point in the Mouse Variable.
Variable (p3)
x = 24
y = 73
Point
Variable (Mouse)
This is pointing to a Row Value record that is on the Heap
Variable (p4)
id: 2
kind: TXT_VAL
data.int_val:
data.dbl_val:
data.txt_val: Fred


You need at least one local variable, otherwise how can you access the other values?
Variable (start)
You can store pointer values on the heap, they are just like any other value...
Data = 24
Next = 0010
Node
Data = 73
Next = 0000
Node
Data = 73
Next = 001f
Node
Nothing!!
There will be a special value to represent the fact you pointer value does not point to anything...

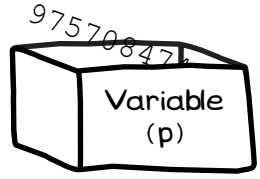

A Linked List will typically store a pointer to the first and last nodes in the List
first node:
last node:
Linked List
Each node has its own memory allocation, so they are not stored next to each other
. . .
Next = 0x10
Node
. . .
Next = 0x00
Node
Nothing!!
Elements of a linked list are called Nodes.
. . .
Next = 0x1f
Node
. . .
Next = 0x13
Node
Each Node stores its data and a pointer to the next Node in the list
. . .
Next = 0x1f
Node
Nodes can be added and removed by changing the links in the list

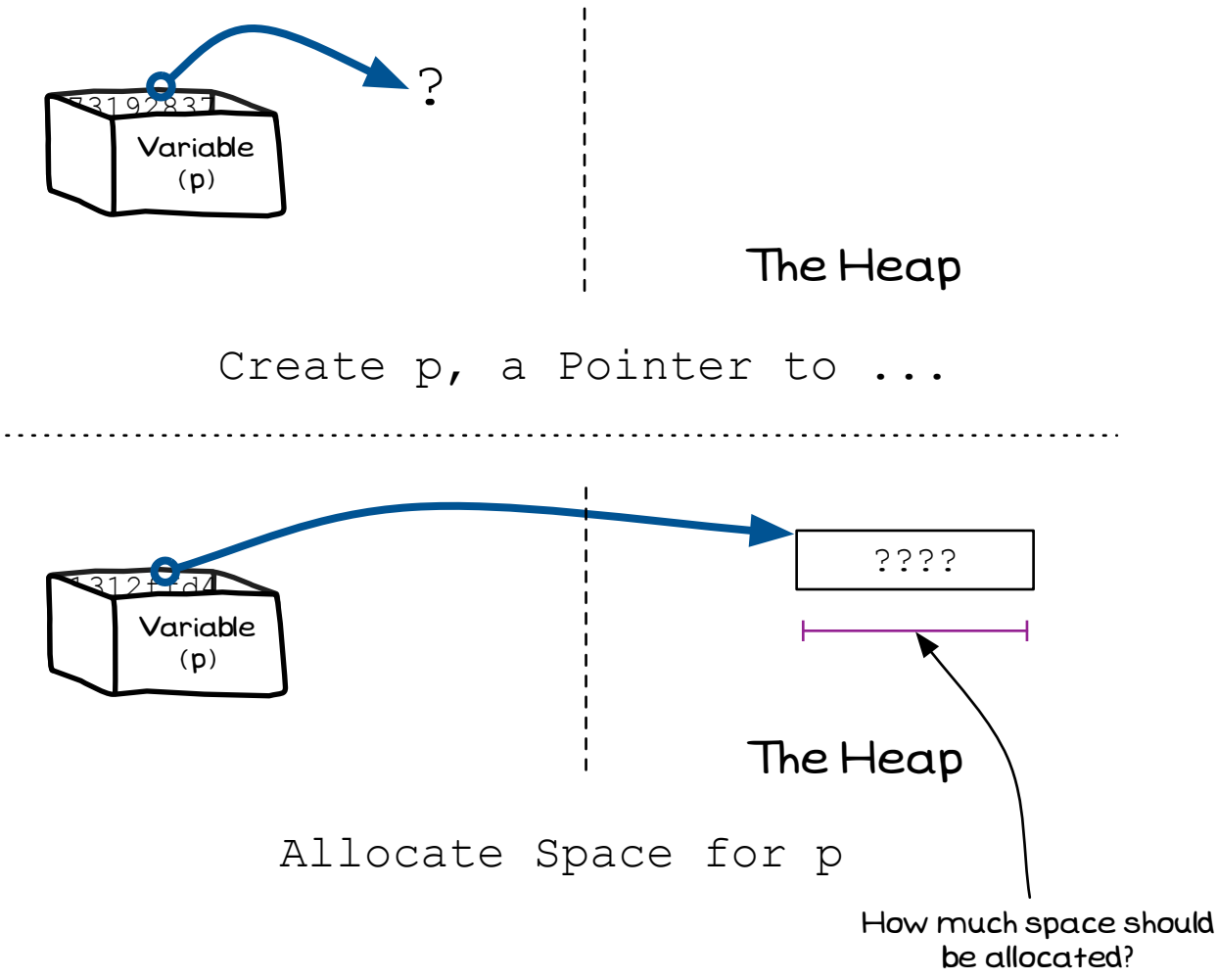

You can get the address of a value
The value is here
3.1415
3.1415
3.1415
(follow p)
read value
(follow p)
store a value
Variable
(p)
You can store the address of a value

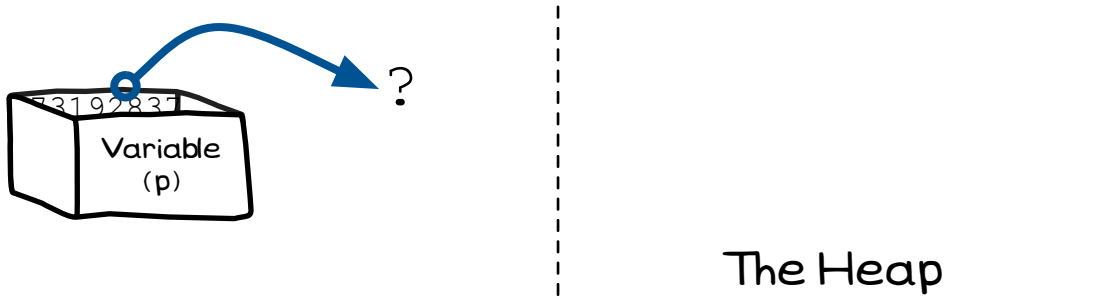

You can get the address of a value
The value is here
3.1415
Variable
In an expression you can follow a pointer and read the value it points to
3.1415
3.1415
In an assignment statement you can follow the pointer (on the left hand side) and store a value in the location it points to
12.873
12.873



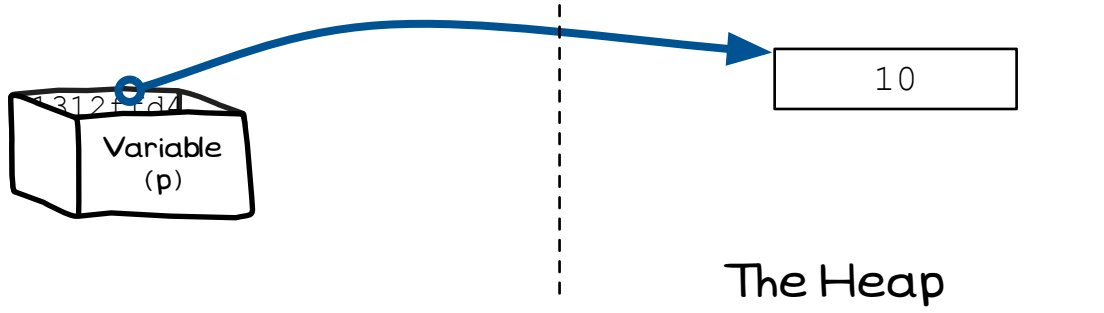

This value stored in p is 975708474,
the address of where the value is stored



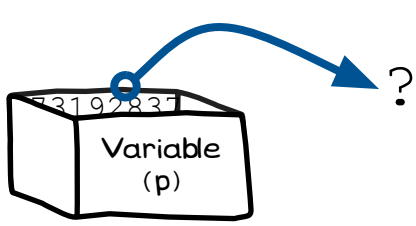

?
Variable
(p)
The Heap
Create p, a Pointer to ...
????
Variable
(p)
The Heap
Allocate Space for p
How much space should
be allocated?




Create p, a Pointer to an Integer

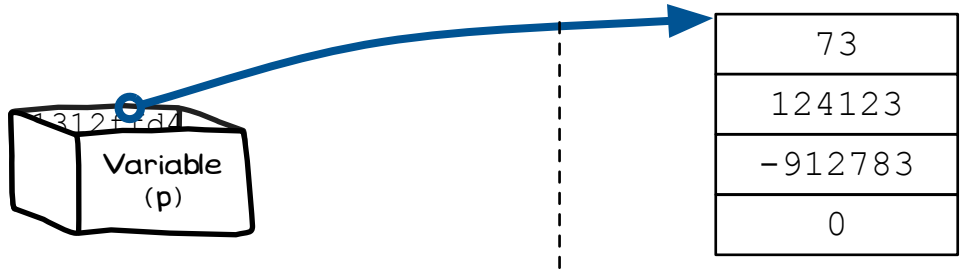



Allocate Space for what p points to (an Integer)



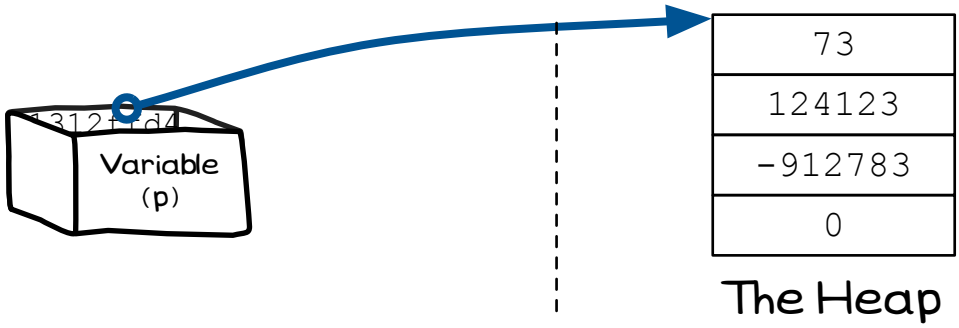

The Heap

Create p, a Pointer to an Integer

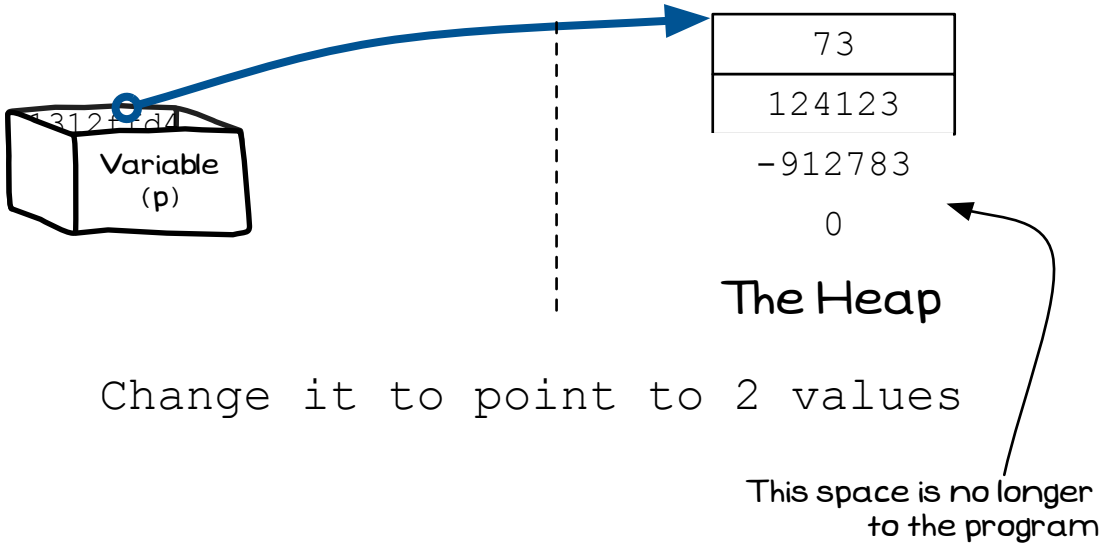



The Heap

Allocate Space for 4 Integer values

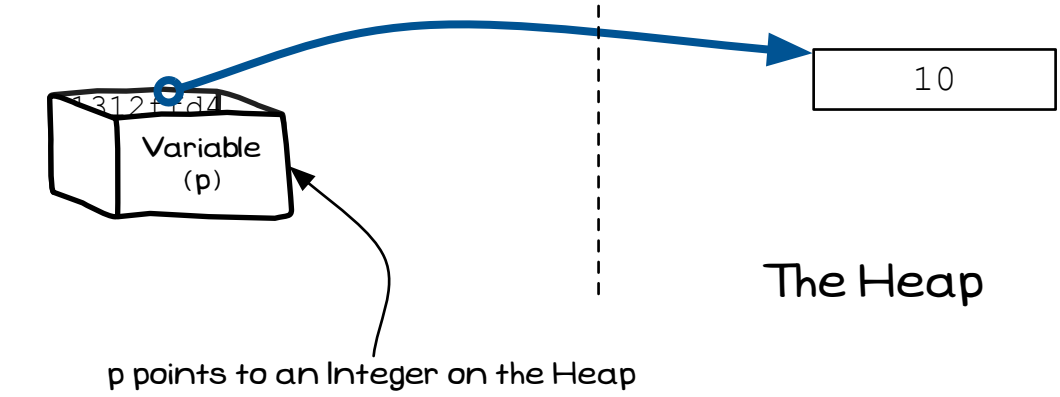


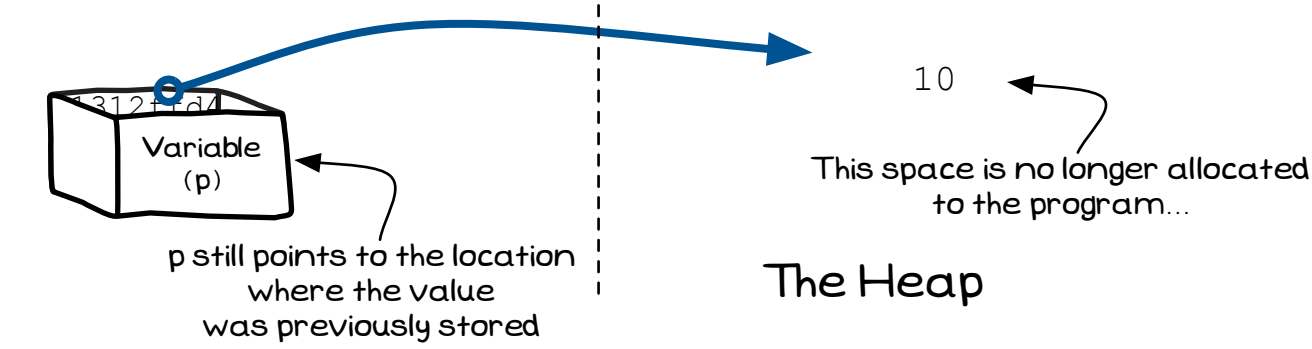

p points to 4 Integer value



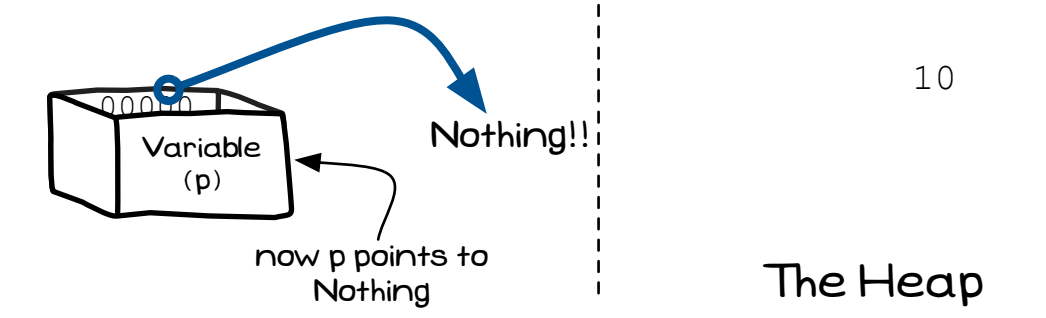

Change it to point to 2 values

This space is no longer allocated to the program...

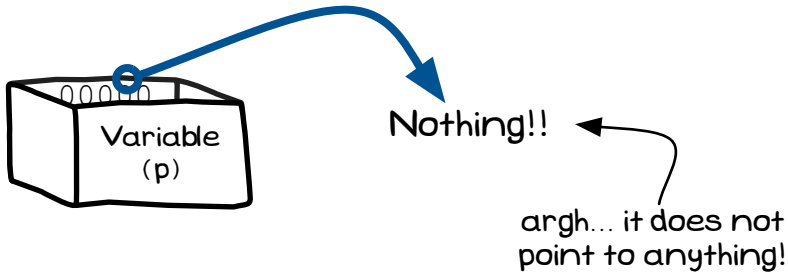


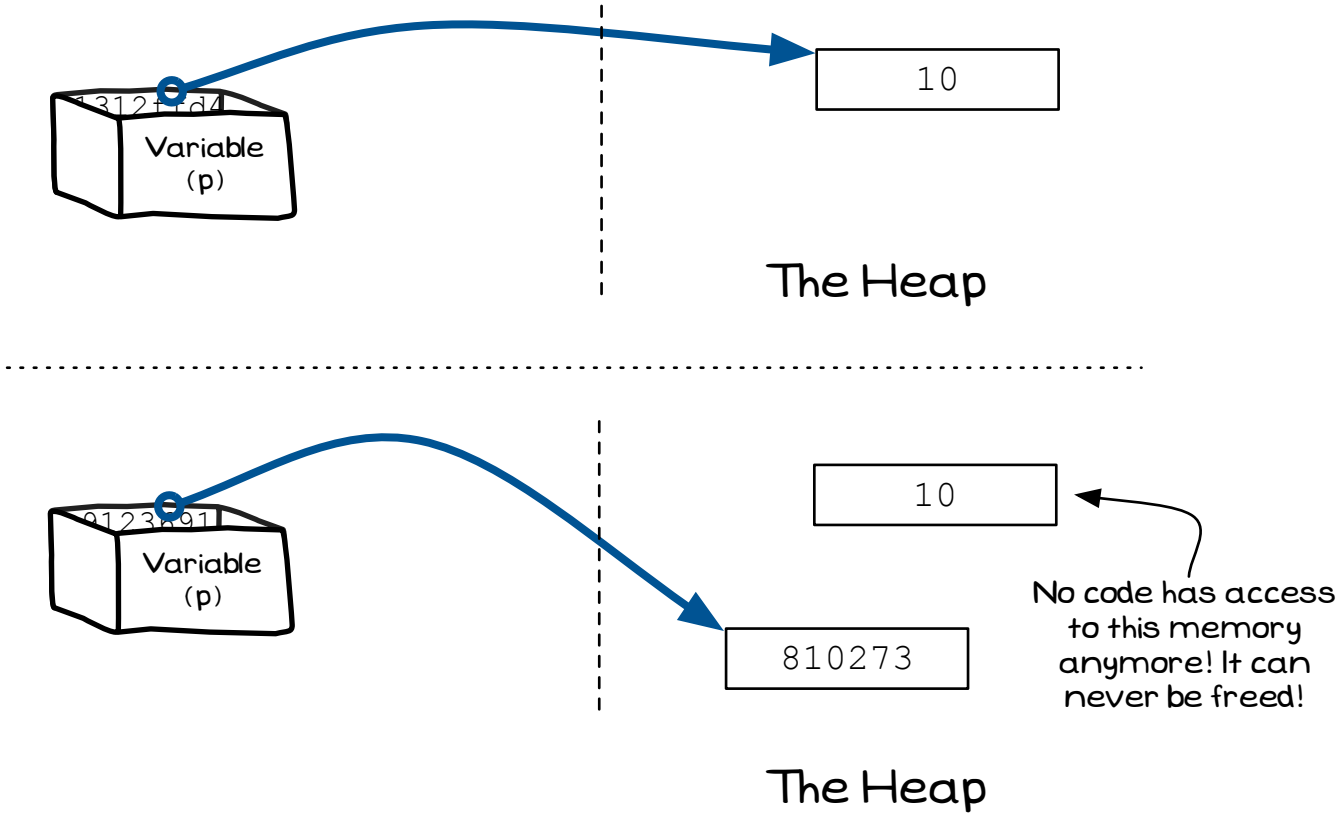

Free the memory referred to by p



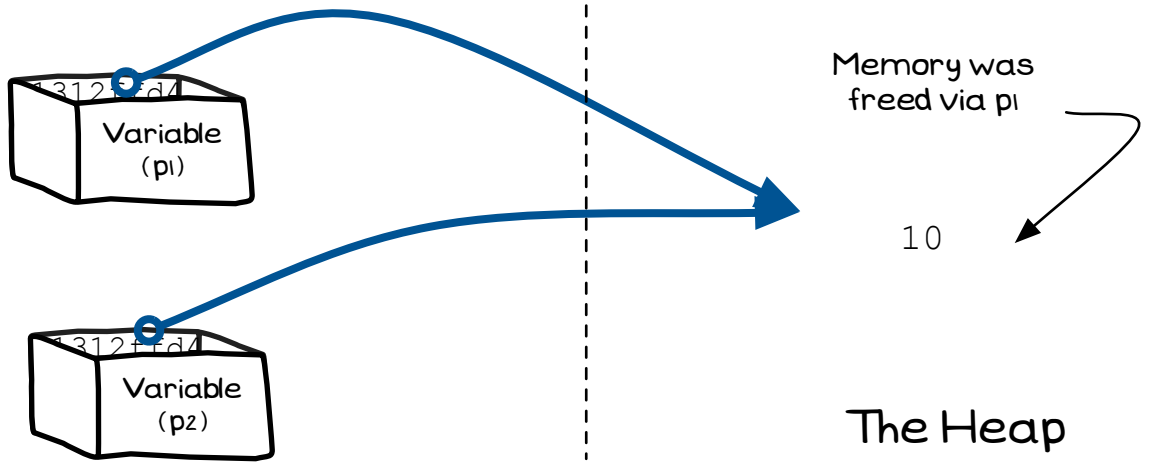

Set p to point to Nothing!



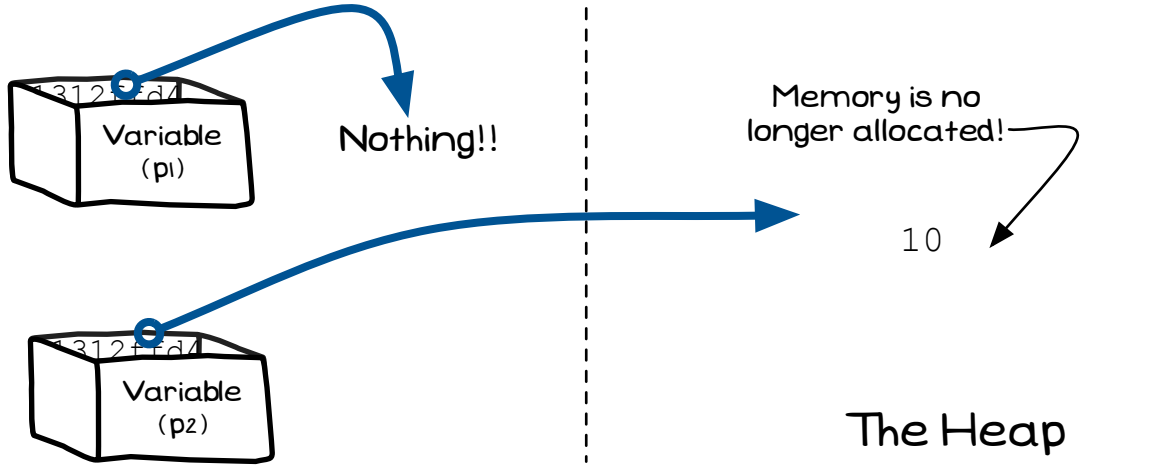

Follow the Pointer p, and ...



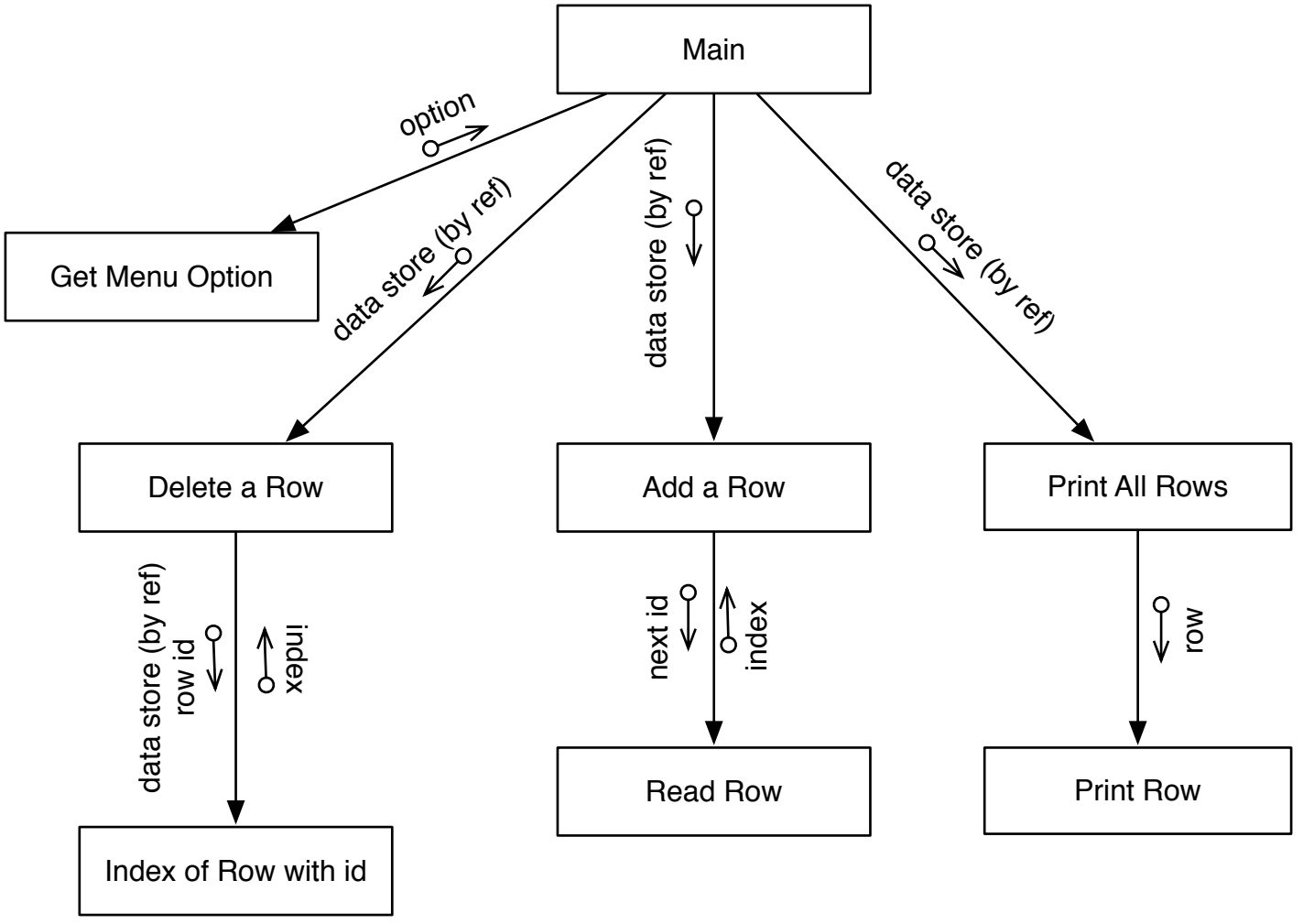

Allocate some memory, and point p at it

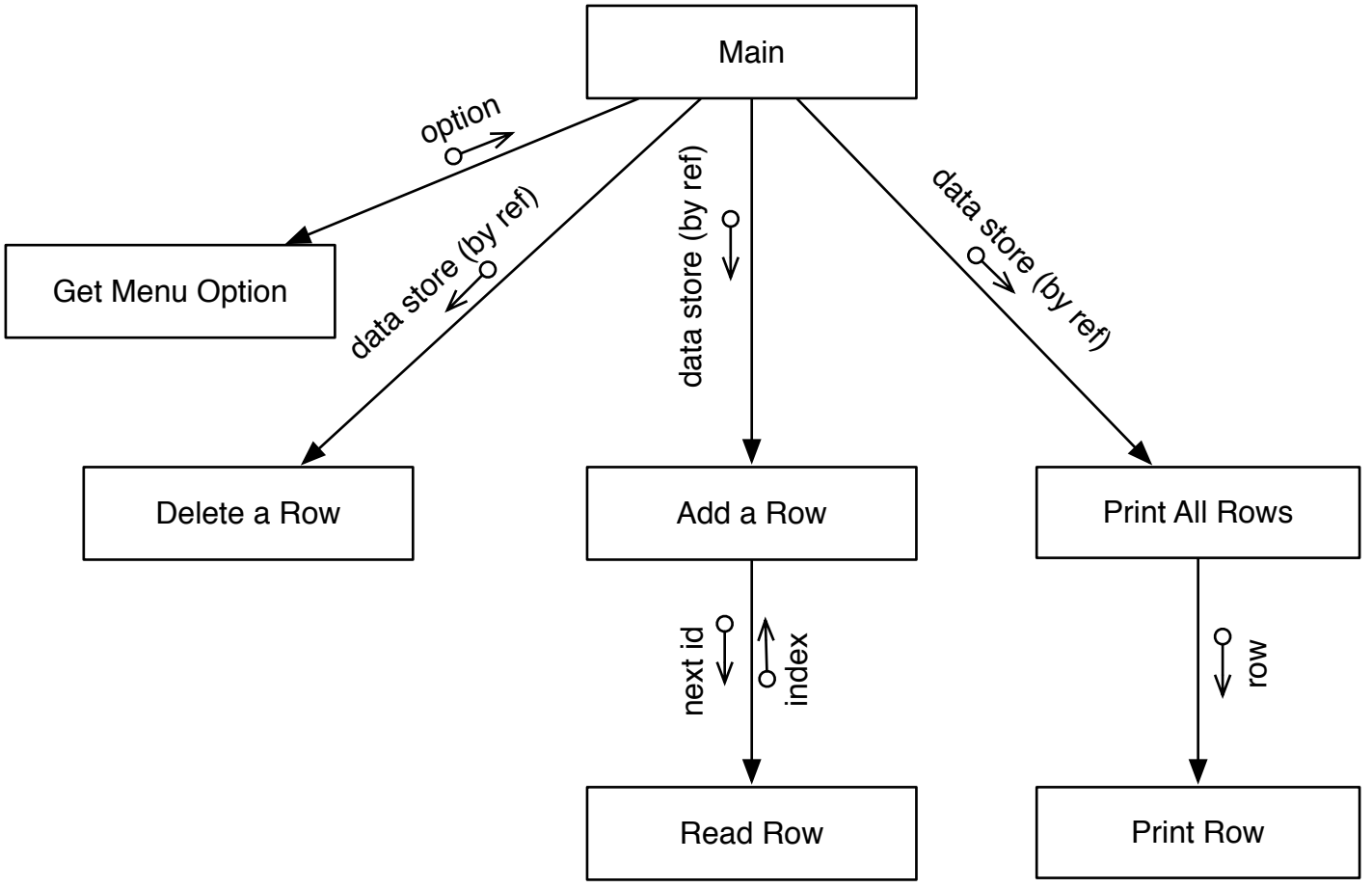


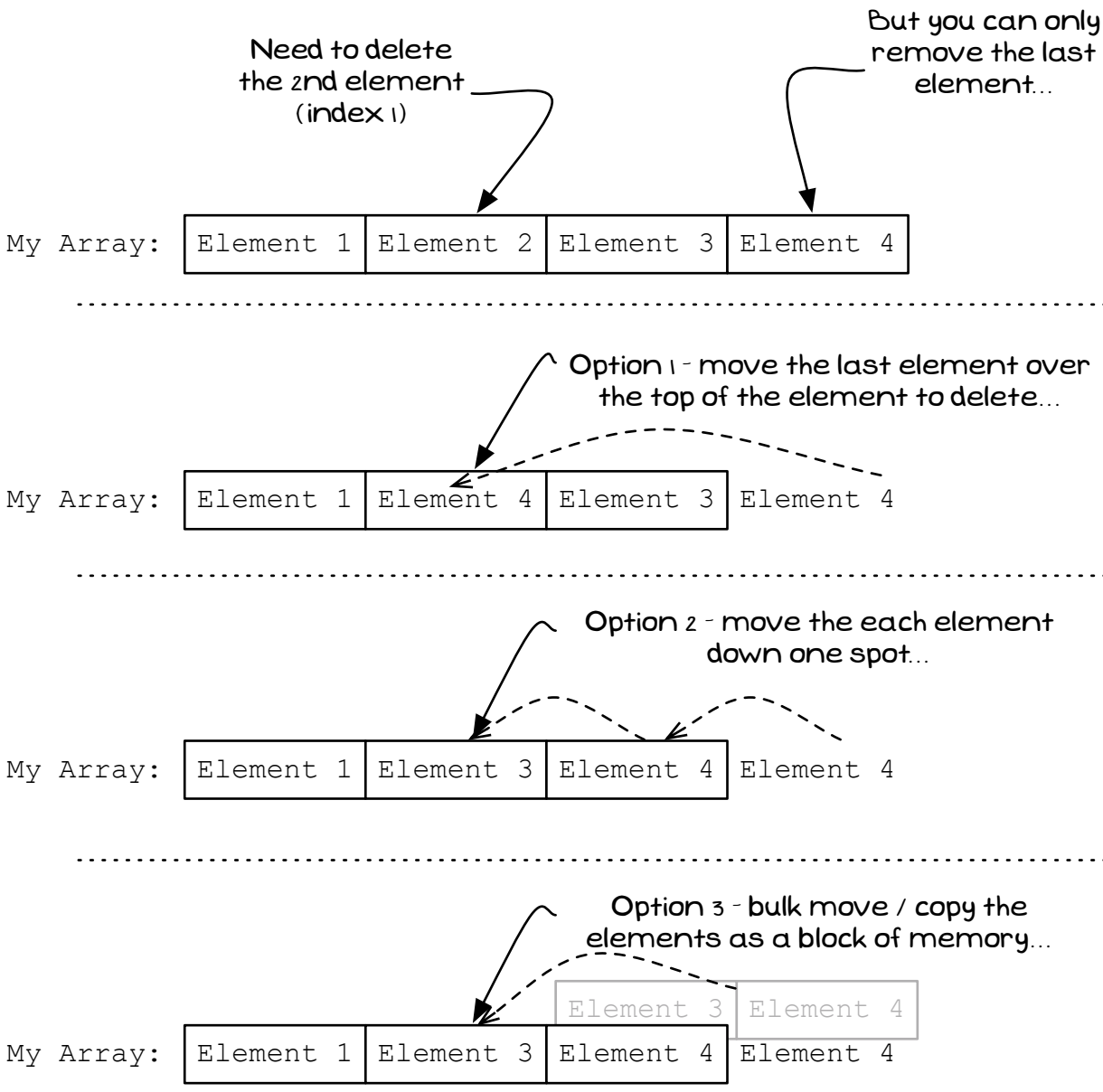

Free memory at p1

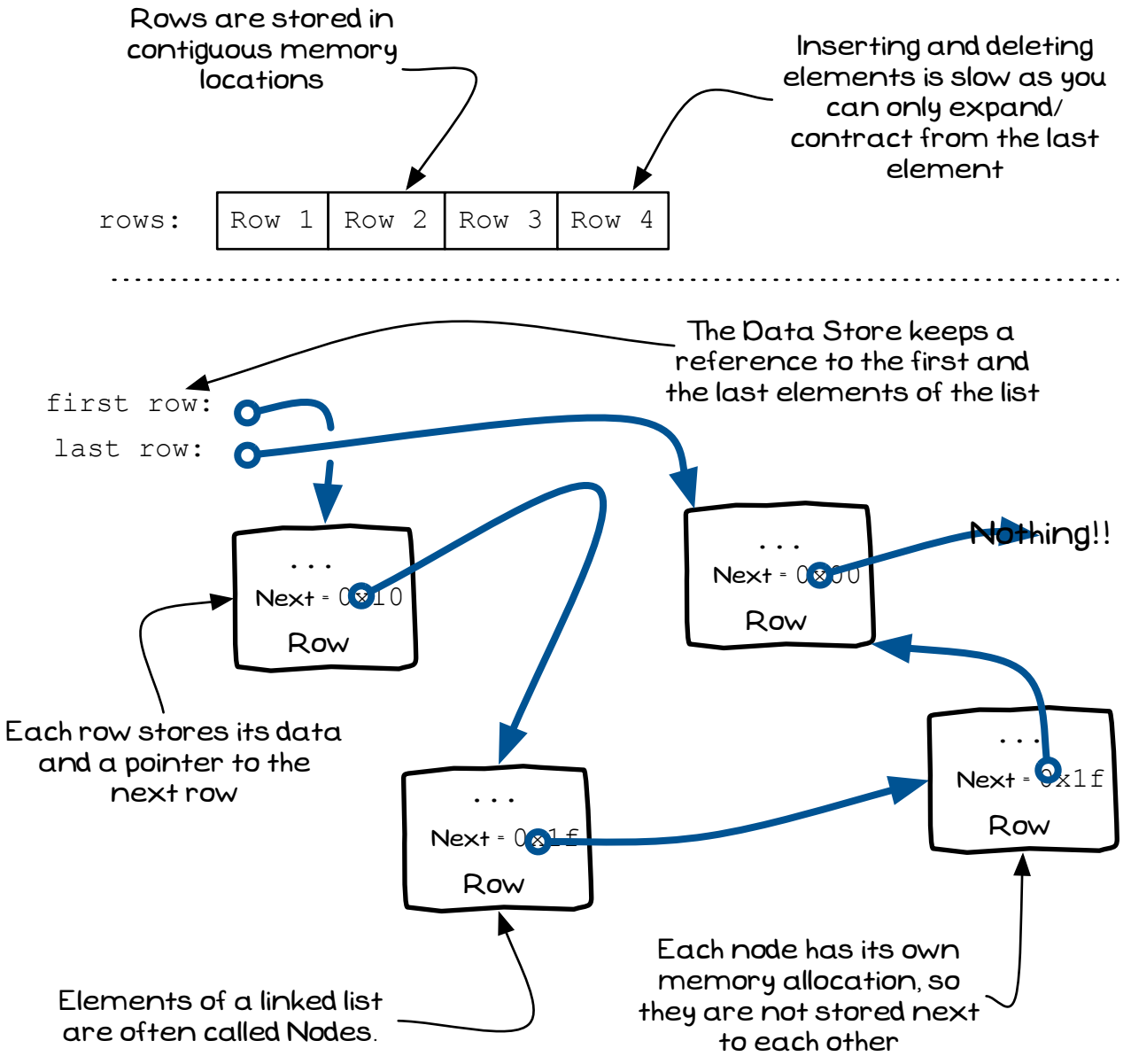


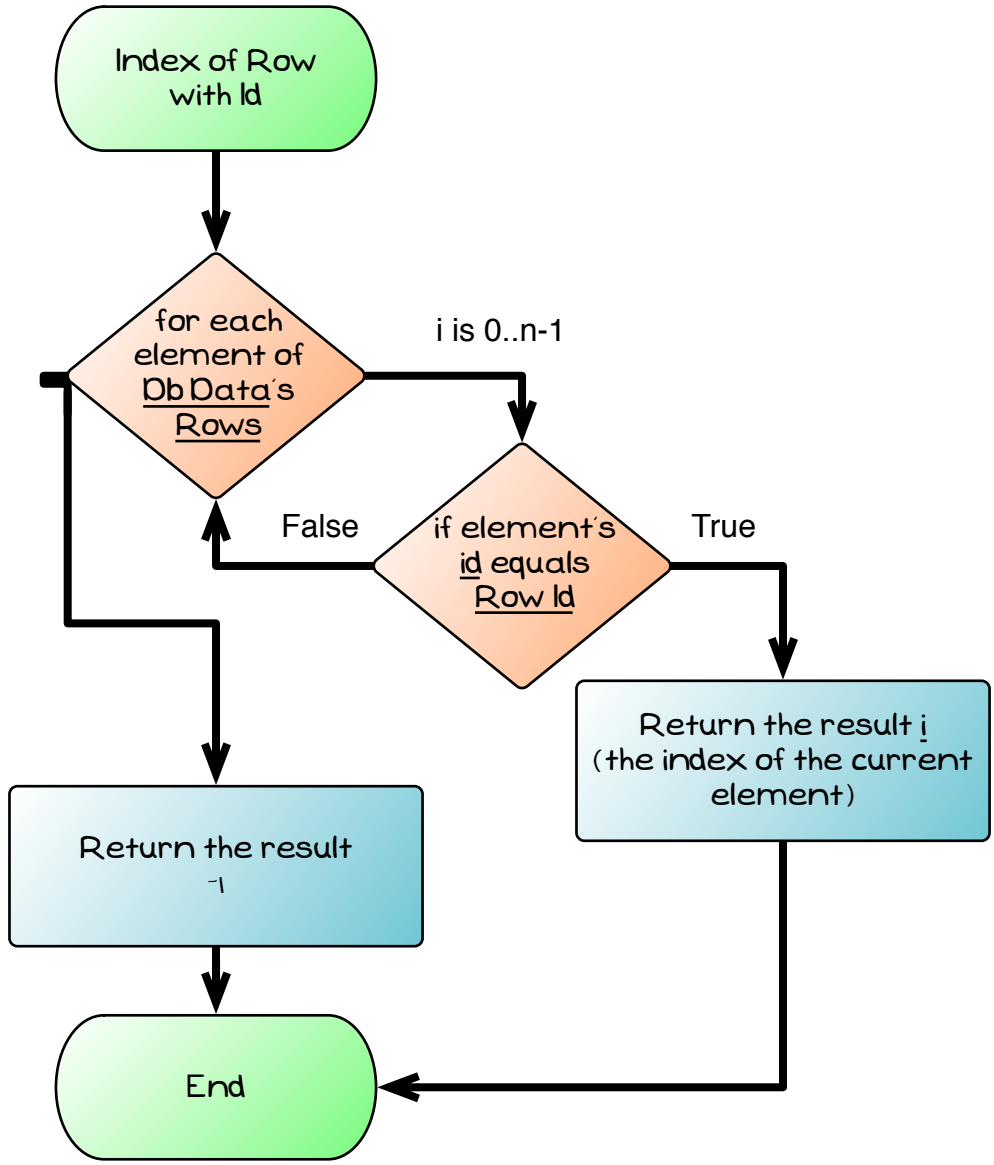

Set p1 to point to Nothing
Read memory at p2 ...

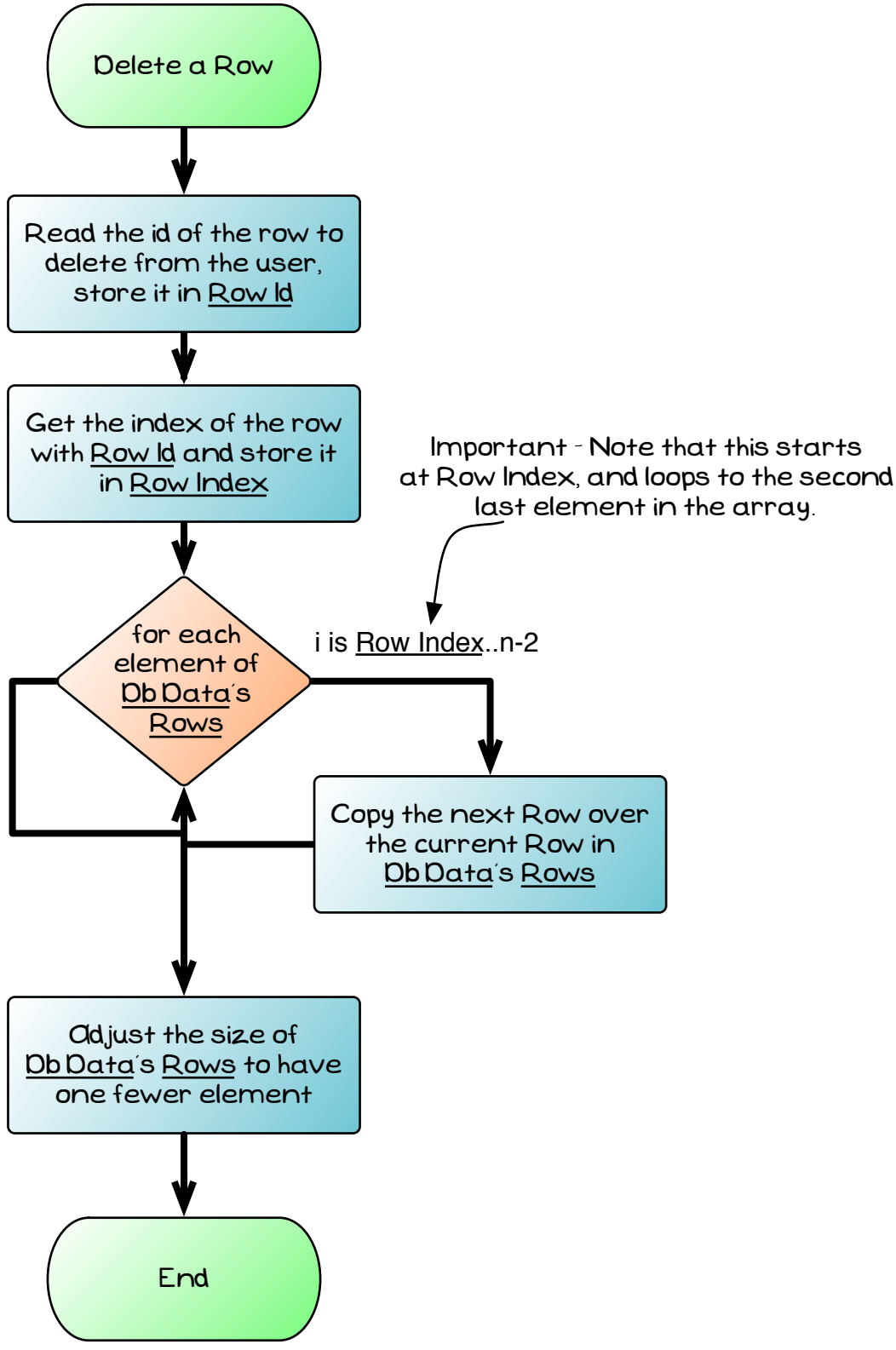

Main
option
Get Menu Option
data store (by ref)
data store (by ref)
data store (by ref)
Delete a Row
Add a Row
Print All Rows
data store (by ref)
row id
index
next id
index
row
Index of Row with id
Read Row
Print Row

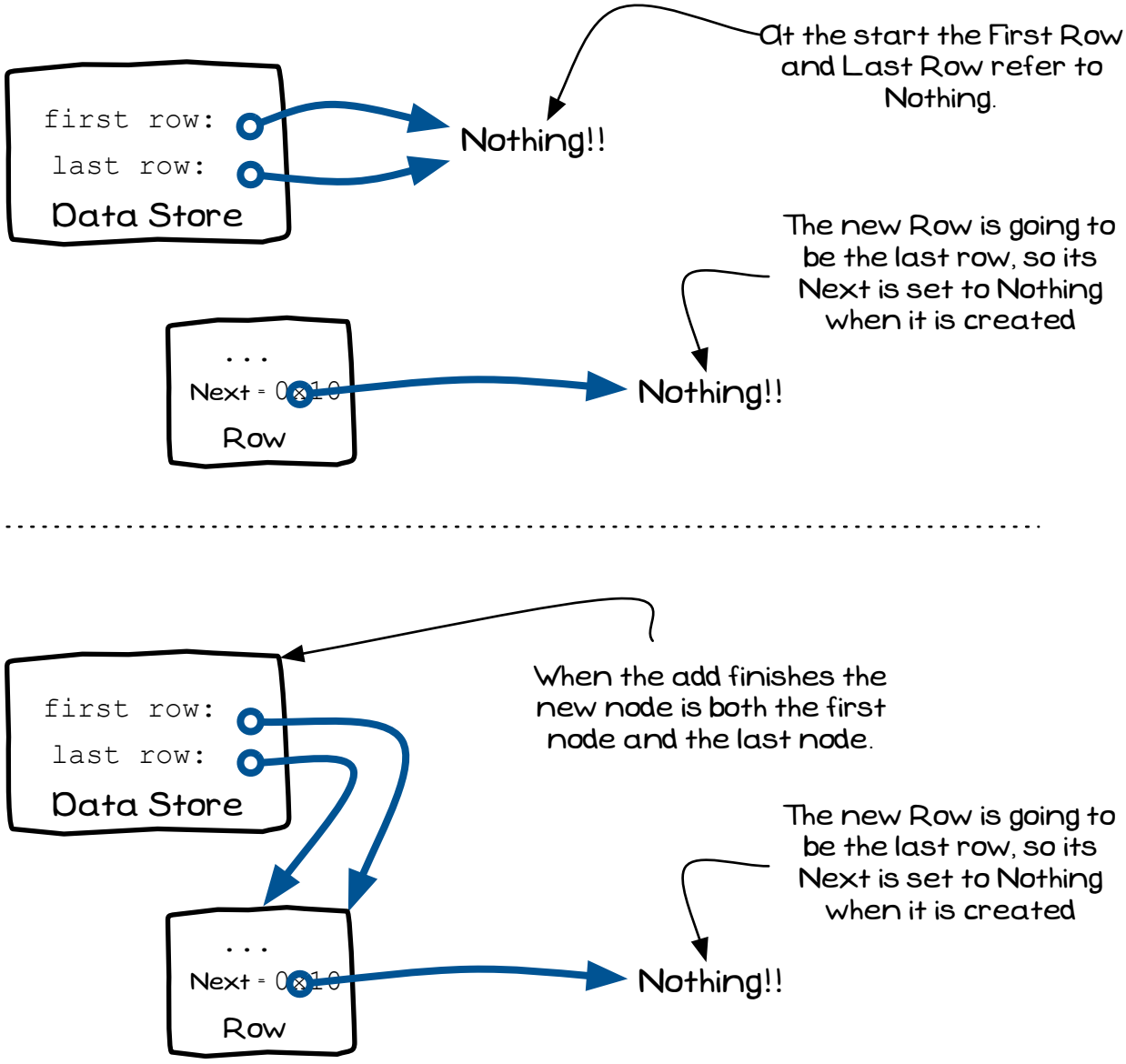

Main
option
Get Menu Option
data store (by ref)
data store (by ref)
data store (by ref)
Delete a Row
Add a Row
Print All Rows
next id
index
row
Read Row
Print Row

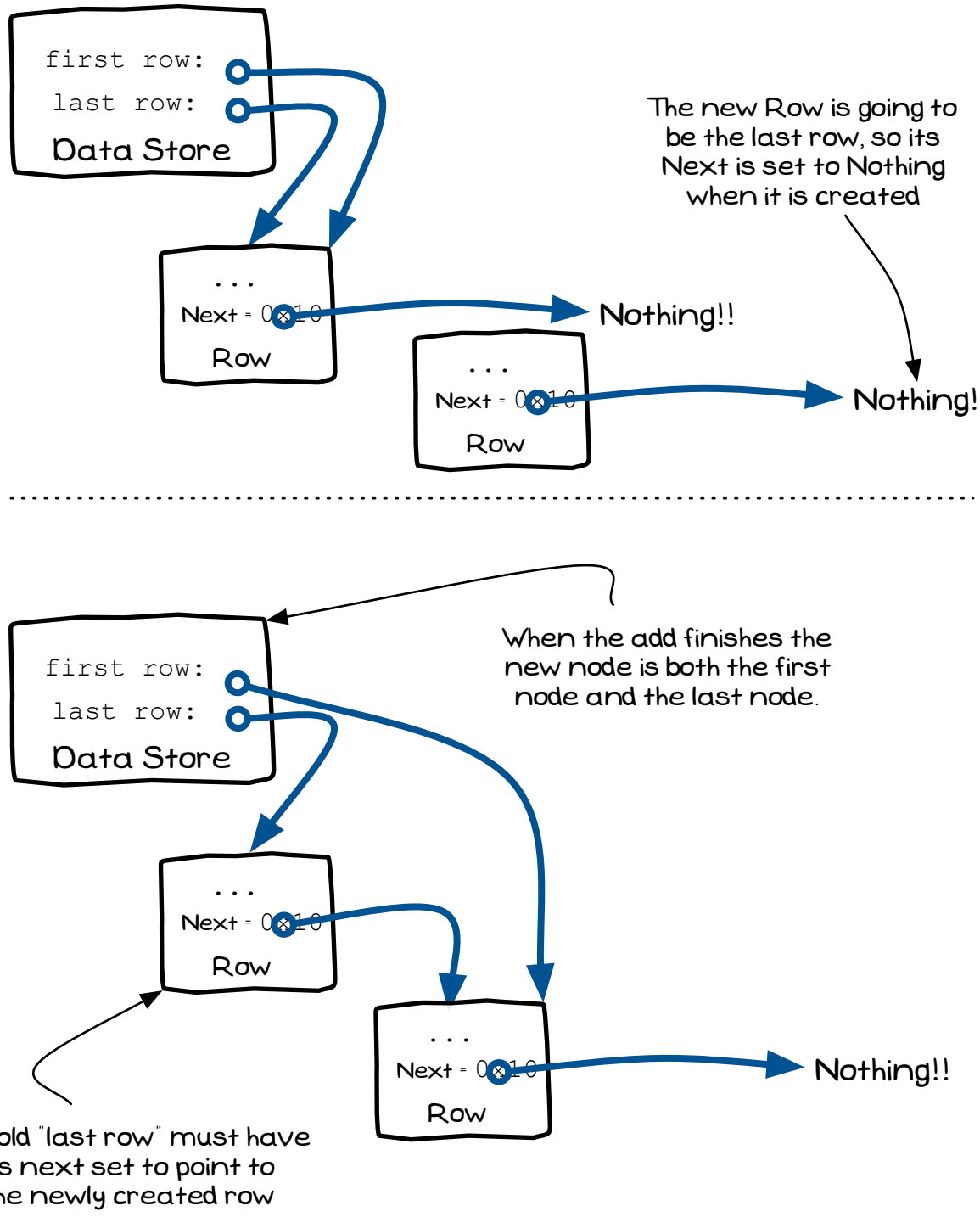

Need to delete the 2nd element (index 1)
But you can only remove the last element...
My Array:
Element 1
Element 2
Element 3
Element 4
Option 1 - move the last element over the top of the element to delete...
My Array:
Element 1
Element 4
Element 3
Element 4
Option 2 - move the each element down one spot...
My Array:
Element 1
Element 3
Element 4
Element 4
Option 3 - bulk move / copy the elements as a block of memory...
Element 3
Element 4
My Array:
Element 1
Element 3
Element 4
Element 4


Rows are stored in contiguous memory locations
Inserting and deleting elements is slow as you can only expand/ contract from the last element
rows:
Row 1
Row 2
Row 3
Row 4
The Data Store keeps a reference to the first and the last elements of the list
first row:
last row:
. . .
Next = 0x10
Row
. . .
Next = 0x00
Row
Nothing!!
Each row stores its data and a pointer to the next row
. . .
Next = 0x1f
Row
. . .
Next = 0x1f
Row
Elements of a linked list are often called Nodes.
Each node has its own memory allocation, so they are not stored next to each other


Index of Row with Id
for each element of Db Data's Rows
i is 0..n-1
if element's id equals Row Id
False
True
Return the result i (the index of the current element)
Return the result -1
End


Delete a Row
Read the id of the row to delete from the user, store it in Row Id
Get the index of the row with Row Id and store it in Row Index
Important - Note that this starts at Row Index, and loops to the second last element in the array.
for each element of Db Data's Rows
i is Row Index..n-2
Copy the next Row over the current Row in Db Data's Rows
Adjust the size of Db Data's Rows to have one fewer element
End


At the start the First Row and Last Row refer to Nothing.
first row:
last row:
Data Store
Nothing!!
The new Row is going to be the last row, so its Next is set to Nothing when it is created
. . .
Next = 0x10
Row
Nothing!!
When the add finishes the new node is both the first node and the last node.
first row:
last row:
Data Store
The new Row is going to be the last row, so its Next is set to Nothing when it is created
. . .
Next = 0x10
Row
Nothing!!


first row:
last row:
Data Store
. . .
Next = 0x10
Row
Nothing!!
. . .
Next = 0x10
Row
Nothing!!
The new Row is going to be the last row, so its Next is set to Nothing when it is created
first row:
last row:
Data Store
When the add finishes the new node is both the first node and the last node.
. . .
Next = 0x10
Row
. . .
Next = 0x10
Row
Nothing!!
The old "last row" must have its next set to point to the newly created row