EECS168/169-Lab6

2-Dimensional Arrays

University of Kansas

2-Dimensional Arrays

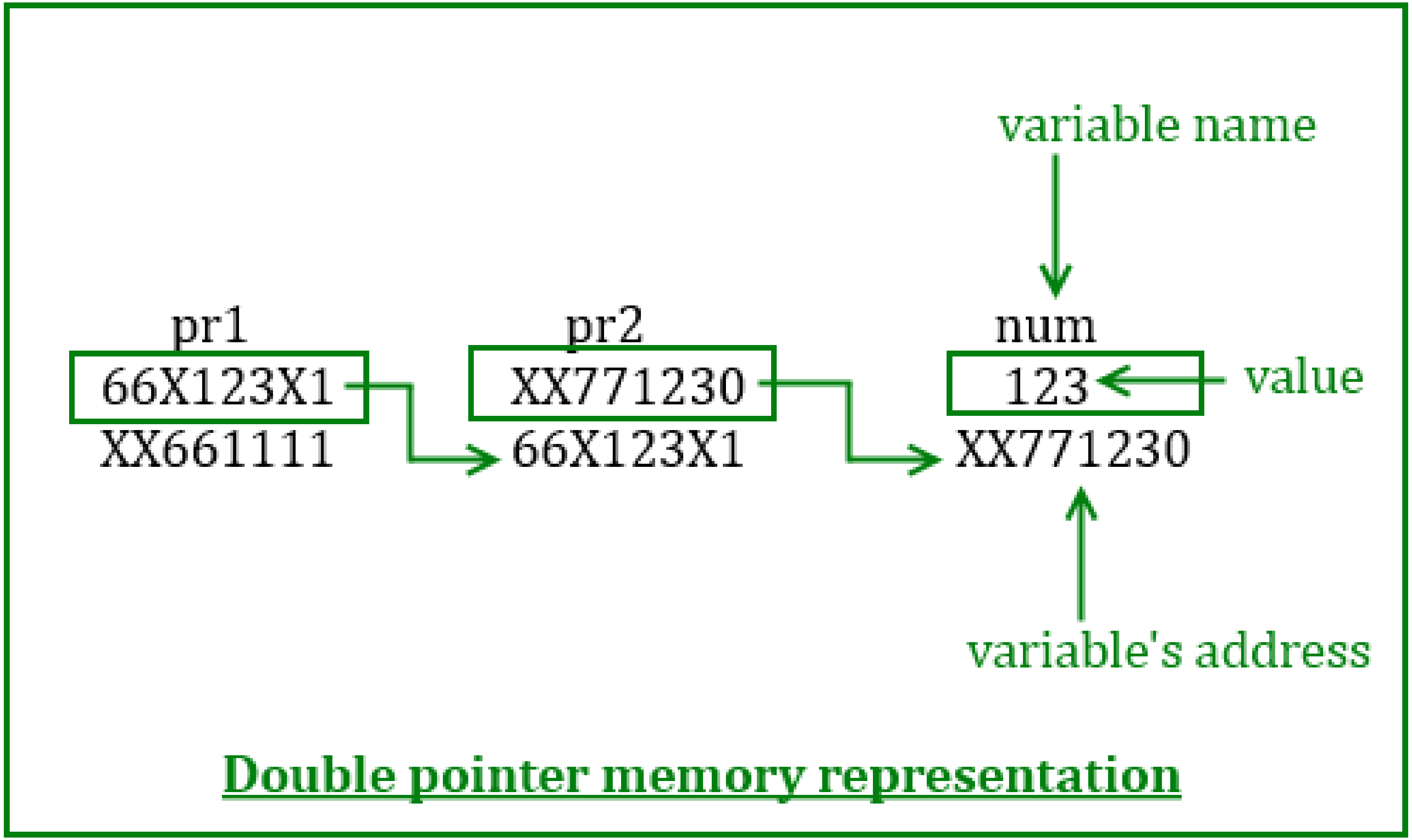
- 2 Dimensional array is “array of many 1 dimensional arrays”
- Create 1-D array holding integer pointers ie “int *”
- int **ar2d = new *int[rows];
- Create 1-D array holding integers ie “int” for each ar2d[i]
- for (int i = 0; i < rows; i++)
-     ard2d[i] = new int[columns]; //for all i

2D Arrays (cont.)

- `new operator returns an address.`
- `int *ptr = new int;`
- `int *ptr = 123; //Error. Cannot initialize pointer variable to anything but address`

- `int **ar2d = new int *[rows];`
- `int **ar2d = new int [rows]; //Error`

Double Pointer



Sample output - EECS168

- Obtain a file name from the user, which will contain data pertaining to a 2D array
- Create a file for each of the following:
  - averages.txt : contains the overall average of the entire array, then the average of each row
  - reverse.txt : contains the original values but each row is reversed
  - flipped.txt : contains the original values but is flipped top to bottom (first row is now the last row etc.)
  - If the dimensions of array are symmetric (NxN), create a diagonal.txt: contains the array mirrored on the diagonal

Examples

The input file will be formatted in the following way:

```
<num rows> <num cols>
<values>
```

Sample file called "input.txt"

```
4 4
1.0 2.0 3.0 4.0
5.0 6.0 7.0 8.0
9.0 10.0 11.0 12.0
13.0 14.0 15.0 16.0
```

Output to averages.txt

```
Total average: 8.5
Row 1 average: 2.5
Row 2 average: 6.5
Row 3 average: 10.5
Row 4 average: 14.5
```

Output to reverse.txt

```
4.0 3.0 2.0 1.0
8.0 7.0 6.0 5.0
12.0 11.0 10.0 9.0
16.0 15.0 14.0 13.0
```

Output to flipped.txt

```
13.0 14.0 15.0 16.0
9.0 10.0 11.0 12.0
5.0 6.0 7.0 8.0
1.0 2.0 3.0 4.0
```

Since it's symmetric we also get diagonal.txt

```
1.0 2.0 3.0 13.0
5.0 6.0 10.0 14.0
9.0 7.0 11.0 15.0
4.0 8.0 12.0 16.0
```

For EECS169

Example of a transpose:

10 20 30

40 50 60

70 80 90

//The transpose would be

10 40 70

20 50 80

30 60 90

```
array[i][j] = array[j][i] except when i = j
Transpose possible if rows = cols for an array
```

Checking for Memory Leaks

- An easy way to check for memory leaks is to use a program called valgrind. It's a program that you can hand your lab off to and it will run your lab and tell if you have any memory leaks.

  - valgrind --leak-check=full --show-leak-kinds=all ./YourProgram

- Do not copy other’s code!
- All GTAs have been instructed to be vigilant about academic misconduct.