

Computer Science & Engineering 150A

Problem Solving Using Computers

Mini Lecture – Dynamic Memory Review

Christopher M. Bourke

Spring 2009

Recall that *pointers* are references to memory addresses.

```
1 int a;  
2 int *b;  
3 int **c;  
4 int ***d;
```

- `a` is an integer
- `b` is a *pointer* to an integer
- `c` is a pointer to a pointer to an integer
- `d` is a pointer to a pointer to a pointer to an integer
- Note: `%p` is the place holder for printing a pointer in `printf`

```
1  #include<stdio.h>
2  #include<stdlib.h>
3
4  int main(void)
5  {
6      int A = 10;
7      int *P = NULL;
8
9      printf("After Declaration: \n");
10     printf("A = %d\n",A);
11     printf("P = %p\n",P);
12
13     /* now we make P point to A */
14     P = &A;
15
16     printf("After assignment: \n");
17     printf("A = %d\n",A);
18     printf("P = %p\n",P);
19
20     printf("To verify: \n");
21     printf("Memory address of A is %p\n", &A);
22     printf("Value Stored at the memory address pointed to by P is %d\n", *P);
23     return;
24 }
```

```
1 After Declaration:
2 A = 10
3 P = (nil)
4 After assignment:
5 A = 10
6 P = 0x7fffc64ad4b4
7 To verify:
8 Memory address of A is 0x7fffc64ad4b4
9 Value Stored at the memory address pointed to by P is 10
```

Factoid: only lower 8 hexadecimal are used for memory addressing, thus $\log_2(16^8) = 32$ bit virtual memory addressing.

```
1 int i, n = 10;
2 int **myMatrix = NULL;
3 myMatrix = (int **)malloc(n * sizeof(int*));
4 for(i=0; i<n; i++)
5     myMatrix[i] = (int *)malloc(n * sizeof(int));
```

```
1  int **myMatrix = NULL; ← declaration
2  myMatrix = (int **)malloc(n * sizeof(int*));
3  for(i=0; i<n; i++)
4      myMatrix[i] = (int *)malloc(n * sizeof(int));
```

`**myMatrix`

```
1  int **myMatrix = NULL; ← declaration
2  myMatrix = (int **)malloc(n * sizeof(int*)); ← initialize pointer to pointers
3  for(i=0; i<n; i++)
4      myMatrix[i] = (int *)malloc(n * sizeof(int));
```

`**myMatrix`

`*myMatrix[0]`

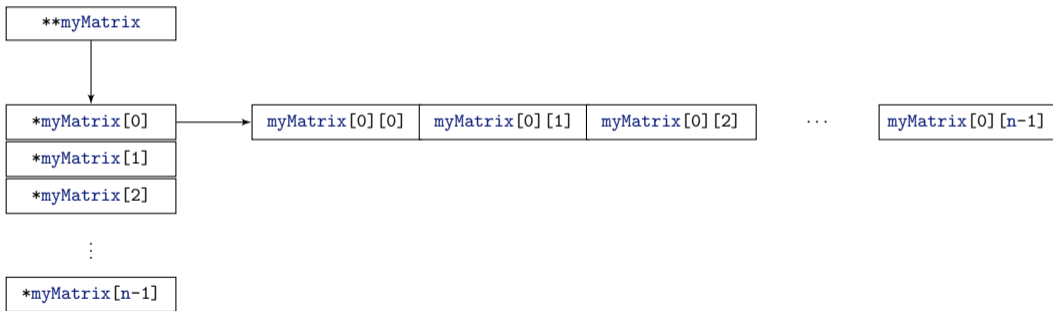
`*myMatrix[1]`

`*myMatrix[2]`

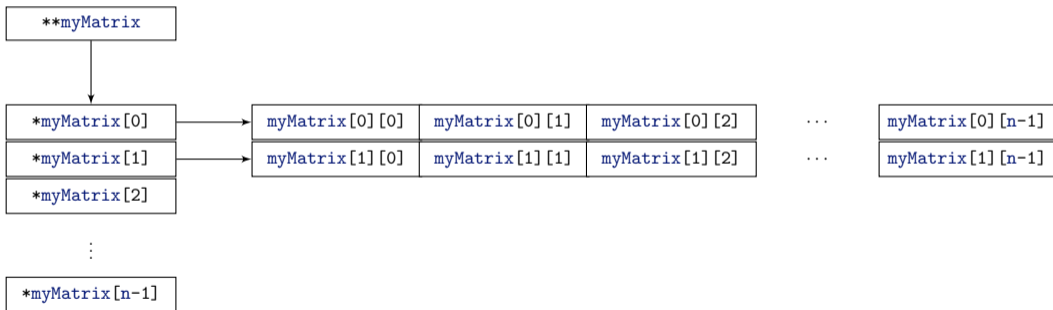
⋮

`*myMatrix[n-1]`

```
1  int **myMatrix = NULL; ← declaration
2  myMatrix = (int **)malloc(n * sizeof(int*)); ← initialize pointer to pointers
3  for(i=0; i<n; i++)
4      myMatrix[i] = (int *)malloc(n * sizeof(int)); ← initialize array for i = 0,1,2,...,n-1
```



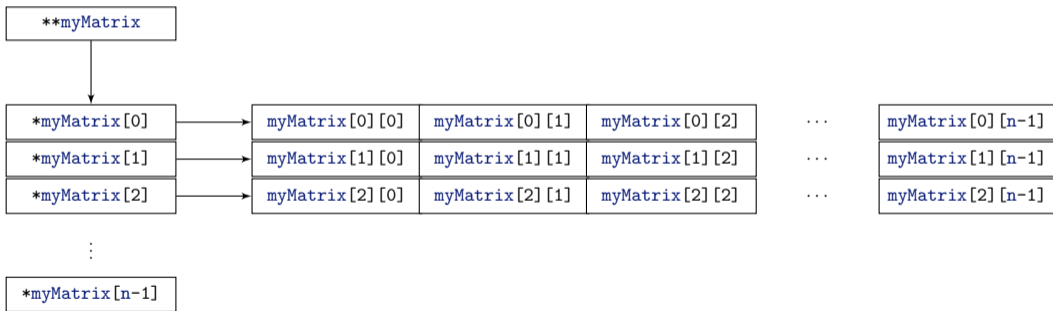
```
1  int **myMatrix = NULL; ← declaration
2  myMatrix = (int **)malloc(n * sizeof(int*)); ← initialize pointer to pointers
3  for(i=0; i<n; i++)
4      myMatrix[i] = (int *)malloc(n * sizeof(int)); ← initialize array for i = 0,1,2,...,n-1
```



```

1  int **myMatrix = NULL; ← declaration
2  myMatrix = (int **)malloc(n * sizeof(int*)); ← initialize pointer to pointers
3  for(i=0; i<n; i++)
4      myMatrix[i] = (int *)malloc(n * sizeof(int)); ← initialize array for i = 0,1,2,...,n-1

```



```

1  int **myMatrix = NULL; ← declaration
2  myMatrix = (int **)malloc(n * sizeof(int*)); ← initialize pointer to pointers
3  for(i=0; i<n; i++)
4      myMatrix[i] = (int *)malloc(n * sizeof(int)); ← initialize array for i = 0,1,2,...,n-1

```

