Pointer Arithmetic

2D Arrays & Pointers

```
#include <stdio.h>
int main() {
    int a[10][10];
    int n = sizeof(a[0])/sizeof(a[0][0]);
    int i, j, *ptr;

    ptr = &a[0][0]; // Could use just a[0]
    for (i = 0; i < n;i++)
        for (j = 0; j < n; j++)
            *(ptr+i*n+j) = i+j; // Size of row is n

    for (i = 0; i < n;i++) {
        for (j = 0; j < n; j++)
            printf("%5d", a[i][j]);
        printf("\n");
    }
}
```

Pointer Arithmetic

2D Arrays & Pointers

- For processing one row, we could simply write `ptr = a[i]` (why?).
  - `a[i]` ≡ `*(a + i)`.
  - So `&a[i][0]` ≡ `&(*(a[i] + 0))` ≡ `&*a[i]`, * (de-reference) and & (reference) cancel each other.
- Eg.
```
for (ptr = a[i]; ptr < a[i] + n; ptr++)
        *ptr = 0; // Clears row i
```

Pointer Arithmetic

2D Arrays & Pointers

```
#include <stdio.h>
int main() {
    int a[10][10];
    int n = sizeof(a[0])/sizeof(a[0][0]);
    int i, j, *ptr;

    for (i = 0; i < n; i++)
        for (j = 0, ptr = a[i]; ptr < a[i] + n; ptr++, j++)
            *ptr = i + j;

    for (i = 0; i < n; i++) {
        for (ptr = a[i]; ptr < a[i] + n; ptr++)
            printf("%5d", *ptr);
        printf("\n");
    }
}
```

Pointer Arithmetic

2D Arrays & Pointers

- Processing column is not easy, because of row major ordering.
- For this we need a pointer to an array of length `n`.
  `int (*p)[n];`
- Parentheses around `*p` are necessary,
- Without them, complier treats `p` as an array of pointers instead of pointer to an array.

Pointer Arithmetic

2D Arrays & Pointers

- In `(*p)[i]`, `*p` represents an entire array.
- So `(*p)[i]` represents element in column `i` of that row.