# Lecture 9
# CS 137
# Fall 2014
# by Chantelle Gellert

## Arrays

```
void one23(int a[], int n){
  int i;
  for(i=0; i < n; i++){
        a[i] = i++;
  }
}

int main(void){
  int a[1000];
  one23(a, 1000);

  return 0;
}
```

sizeof:
sizeof(type);
sizeof(variable);
expression: returns the size of the type or the variable usually in bytes)

---

```
void strange(int a[, int n){
 printf("%d\n", sizeof(a));
 printf("%d\n", sizeof(n));
 printf("%d\n", (long int) a); //140733275212096 location in memory
}

int main(void){
 int i; float x; double y; int a[10] = {0};
 printf("%zu\n", sizeof(int)); //4
 printf("%zu\n", sizeof(i)); //4
 printf("%zu\n", sizeof(x)); //4
 printf("%zu\n", sizeof(y)); //8
 printf("%zu\n", sizeof(char)); //1
 printf("%zu\n" sizeof(a)); //40

 strange(a,10); //8 4
 //8 is the size of an address, an int is size 4

 return 0;
}
```

---

An array in C is really just a pointer to an int. int *a;

---

```
void whatever(int *a, int n){
  printf("%d\n", a[n-1]);
  //as a parameter type int *a is equivalent to a[]
}

int main(void){
 int a[5] = {0,1,2,3,4};
 int *b = a;
 printf("%d\n", a[2]);
 b[2] = 100;
 printf("%d\n", a[2]);

 return 0;
}
```

---

# Multidimensional array

```
int a[4][3] = {{11,12,13}, {21, 22,23}, {31, 32, 33}, {41,42,43}};
3 : represents the number of columns
4 : represents the number of rows
```

e.g sum elements in a 4x3 array

```
int sum43(int a[4][3]){ //need at least number of columns for memory management
  int i,j,sum = 0;

  for(i = 0; i < 4; i++){
    for(j = 0; j < 3; j++){
      sum += a[i][j];
    }
  }
  return sum;
}
```