Pointers and Arrays

Strong relationship between pointers and arrays in C.

Declaration of an array with N elements:
- Element type [array name][N];
- E.g., int a[10];

```
int a[10];
int *pa;
p = a;

pa = &a[0];
p = a;  // x = pa

*(pa+1) = 7;  // *pa
pa = x+2; <= CONT As this
```

*a[i] <= *(a+i)
&a[i] <= a+i

```
i = 13
a[6] = 5;
p = a;
p++;   // Legal
a = pa;  // Not Legal
a++;```

2D arrays are stored in row-major order.

Index (2nd) varies fastest as you step through memory.

```c
int C[3][2];
```

This extends to more than 2 dimensions. First array index varies slowest, last index varies fastest.

```c
int xp;
p = C;
p = p+4;
xp = 5;
C[?][C[?]] = 5
C[2][0] = 5
```

Arrays of arrays:

```c
typedef int int32_t on 32bit x86
typedef short int int32_t on 64bit x86
```

```c
int32_t c;
for (i = 0; i < 24; i++) {
    c[i] = i;
}
```

```c
typedef int2 complex [2];
```

It's complex's user-defined type:

```c
array of 2 ints x1
```

```c
complex D[3];
```

```c
int C[3][2];
```
structs

struct cplx {  // Define cplx as:
    int real;
    int imag;
};

struct cplx var;

access fields via . notation

Var. real = 3;
Var. imag = 1;

struct cplx E(x2);  // E: array of 3 structs
E(x2). real = 7;
E(x2). imag = 3;

typedefs commonly used with structs

typedef struct cplx_s {  // char label;
    int real;
    int imag;
} Complex;

Complex x;
Complex y;

void sumComplex (Complex a, Complex b, Complex *x) {  // *x
    (*x). label = 'x';
    (*x). real = a.real + b.real;
    (*x). imag = a.imag + b.imag;
    return;
}
No. 1

sum complex 1 (Complex a, Complex b, Complex c)

1

s := label := 1;
2
s := real := a.real + b.real;
3
s := imag := a.imag + b.imag;

return;

No. 2

sum complex 2 (Complex a, Complex b, Complex c)

1

s := label := 1;
2
s := real := a.real + b.real;
3
s := imag := a.imag + b.imag;

x := complex x, y, z;

usage
sum complex (x, y, z), sum complex (b, x, y);