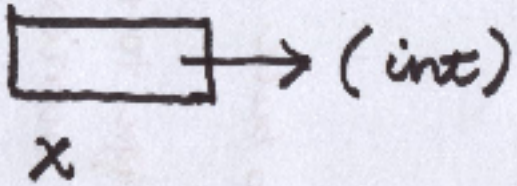
Pointers, arrays, strings.

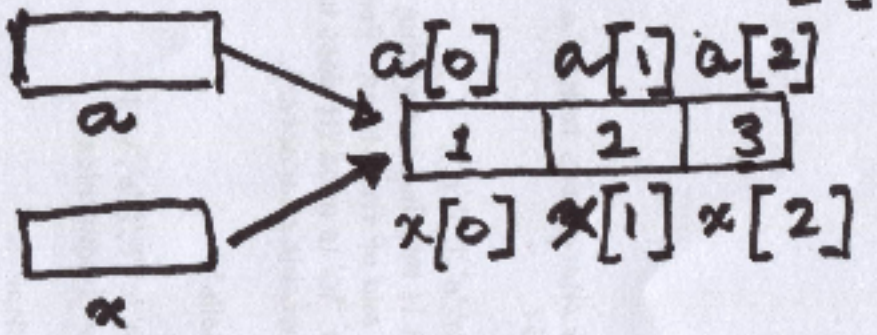
Pointer variable contains address of some item.

Declare: `int *x;`



compatible with 1-dim arrays:

`int a[3] = {1, 2, 3};  x = a;`



ALLOCATION We use only calloc() from stdlib.h

- calloc(p, q) (i) finds an unused block of memory of size p*q
  (ii) clears it to zero (iii) returns its address

Eg char *x = (char *) calloc(100, 1)

double *y = (double*) calloc(100, sizeof(double));

NULL Pointer NULL (in stdio.h)

is never the address of anything.

(actually zero)

FGETS for reading lines of text (stdio.h)

```
char buffer[200];
char * fgets ( char buffer[], int maxlen,
               FILE * input (always stdin) )
```

i reads ~~next~~ chars until ~~end of~~ newline or end of data or up to maxlen - 1.

ii appends null char in buffer

iii returns value of buffer or NULL if end of data.

· STYLE

```
while (fgets (buffer, 200, stdin) != NULL)
{ ... }
```

ALLOCATING 1-dim, 2-dim arrays

1-dim: easy

```
double *a = (double *) calloc (n,
                    sizeof (double));
```

2-dim: hard

```
double ** quasi_2d (int m, int n)
{ double ** mat = (double **)
      calloc(m, sizeof (double *));
  int i;
  for(i=0; i<m; ++i)
  { mat [i] = (double *) calloc (n,
                    sizeof (double));
  }
  return mat;
}
```