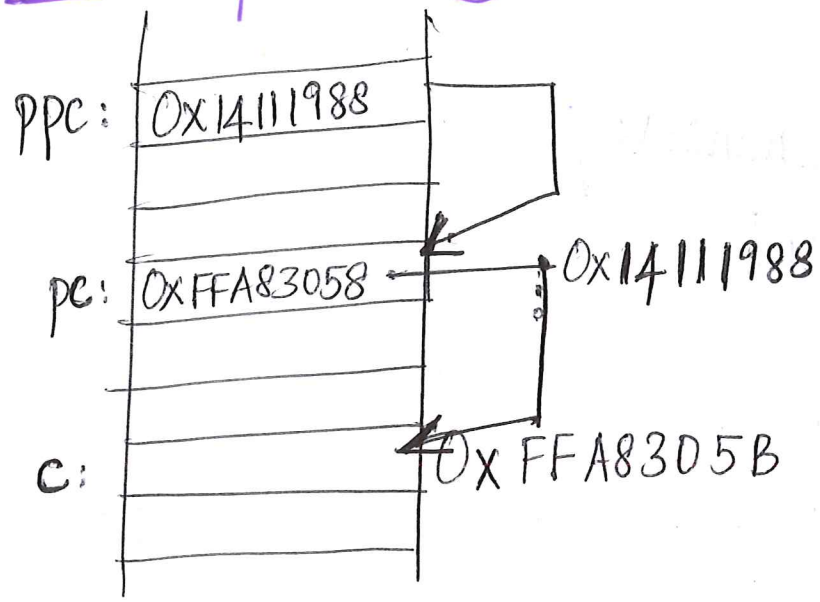
Pointers to pointers

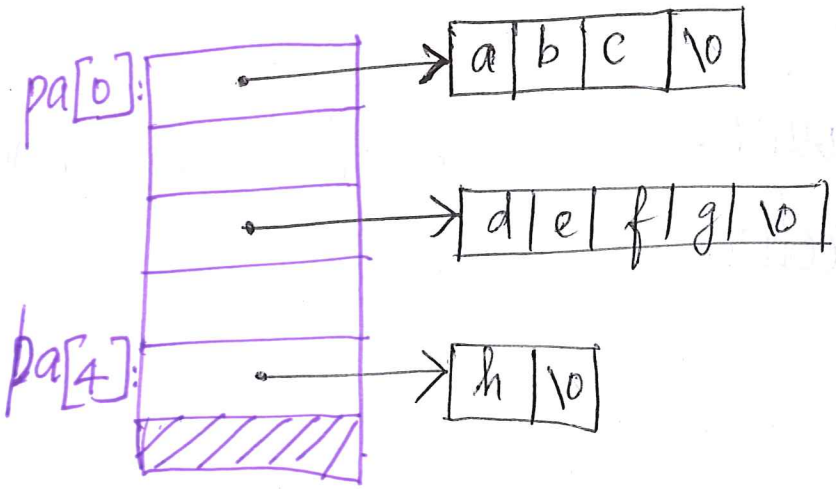
NULL pointer




```
char c;
char *pc;
char **ppc;
c = 'A';
pc = &c;
ppc = &pc;
```

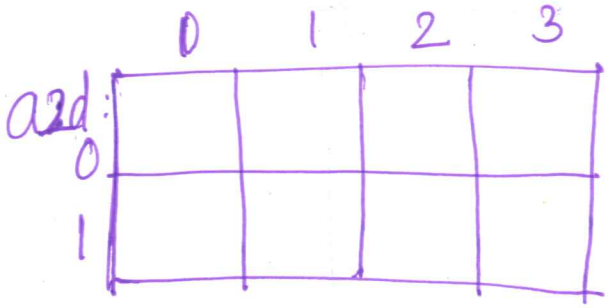
Pointer arrays

```
char *pa[5];
```



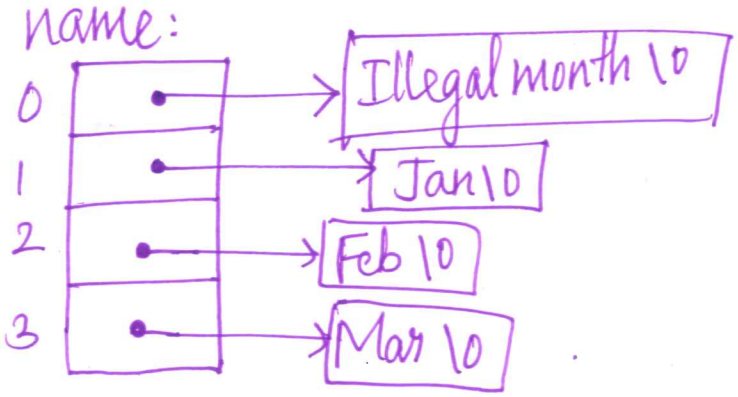
Two dimensional arrays

```
int a2d[2][4];
```



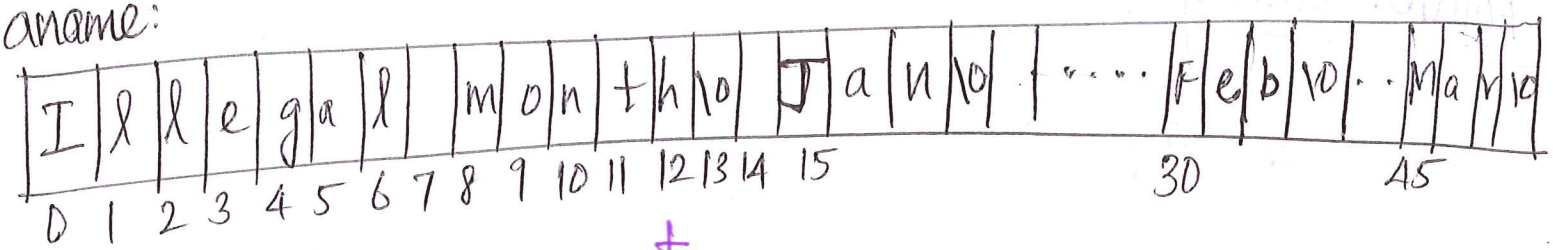
2

```
char *name[] = {"Illegal month", "Jan", "Feb", "Mar"};
```




```
char aname[][15] = {"Illegal month", "Jan", "Feb", "Mar"};
```

aname:

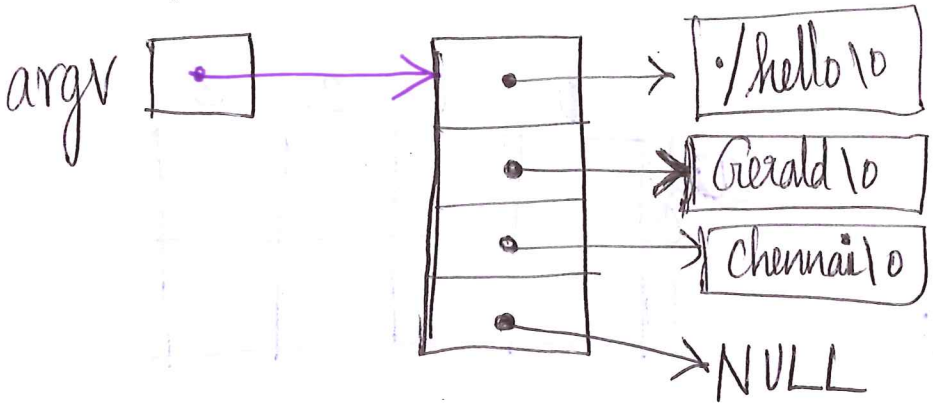


Command line arguments

argc – argument count.

argv – argument vector.

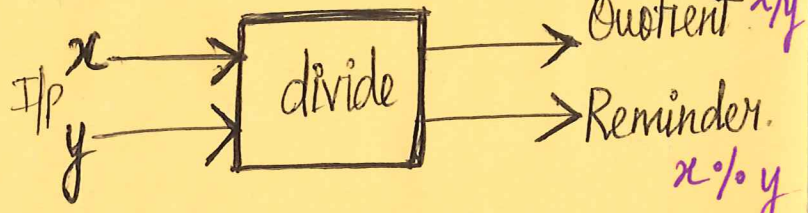
$ ./hello Gerald Chennai



```
int main(int argc, ~~int~~ char *argv[])
{
    ____________
    ____________

}
```

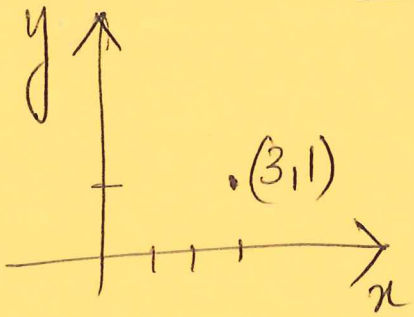
division function



e.g.

Design the function prototype for divide ~~method~~ function.

Structures




```
struct point {
    int x;
    int y;
};
```

x, y – members of the structure.

point – structure tag / name.

a struct declaration – defines a type.

eg. struct point p1, p2;

III^r to int x1, x2;

Accessing members

struct point pt;

pt.x
pt.y
} used to access x and y.

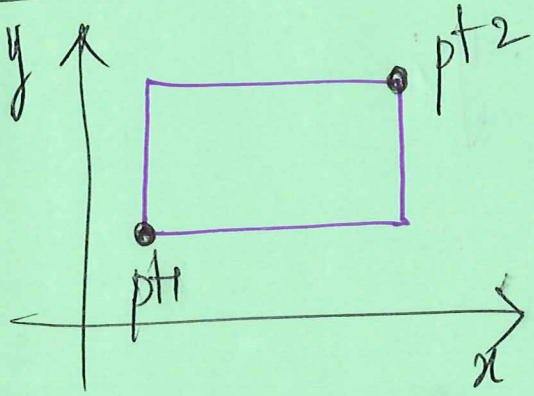
. ⇒ structure member operator.

```
printf("%d, %d", pt.x, pt.y);
```

Distance of (x,y) from $(0,0)$

```
double dist, sqrt(double);
dist = sqrt((double)pt.x * pt.x +
            (double)pt.y * pt.y);
```

Nested Structures




```
struct rect {
    struct point pt1;
    struct point pt2;
};
```

```
struct rect screen;
screen.pt1.x = 1;
```

Structures and Functions

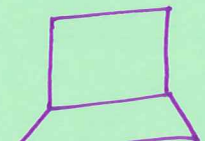
Legal operations on structs:

1. copying a struct
2. assigning to it as a unit
3. taking the address of a struct using &
4. accessing its members.

```
struct point addpoint(struct point p1, struct point p2)
{
    struct point temp;
    temp.x = p1.x + p2.x;
    temp.y = p1.y + p2.y;
    return temp;
}
```

Pointers to structures

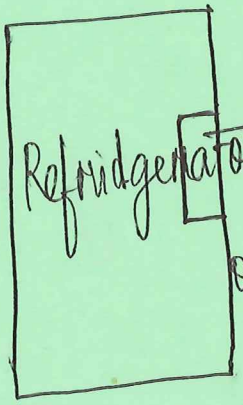
Analogy:



if laptop in repair

take laptop

OK
(∵ small object)



give your address

service provider comes to your house.

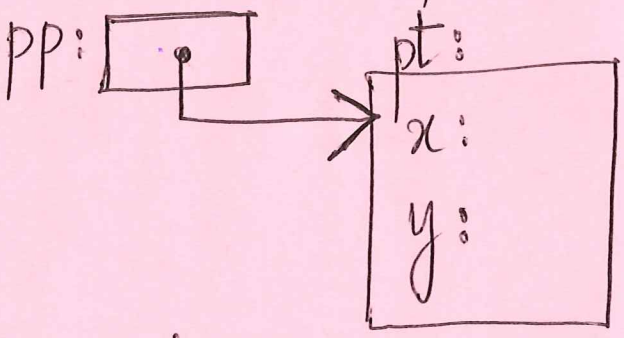
Ill'y OK to copy small structures between functions.

NOT OK to copy Large structures.
USE pointers for doing this!
↓
address of large structure.

Syntax

```
struct point *pp;
struct point pt;
```

pp = &pt;



*pp ⇒ structure (pt)

(*pp).x ⇒ pt.x

(*pp).y ⇒ pt.y

Better syntax

pp→x ⇒ pt.x

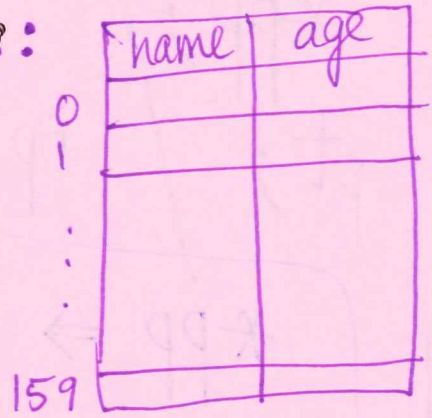
pp→y ⇒ pt.y

Arrays of structures

```
struct studinfo {
    char *name;
    unsigned int age;   // unsigned short int - better
};
```

```
struct studinfo student[160];
```

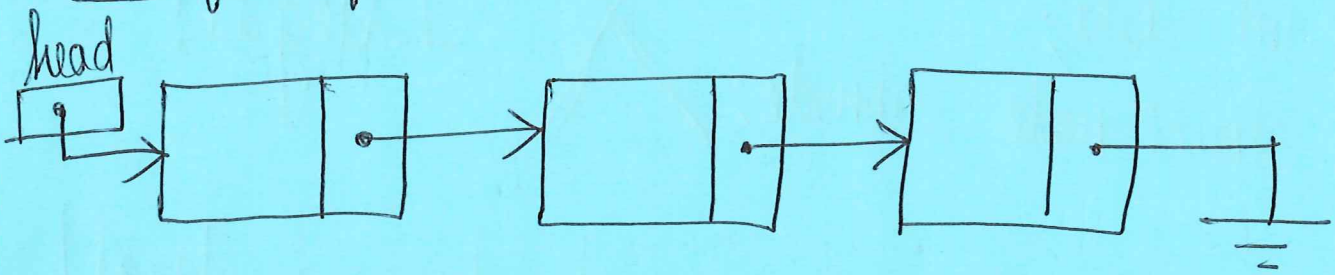
student:



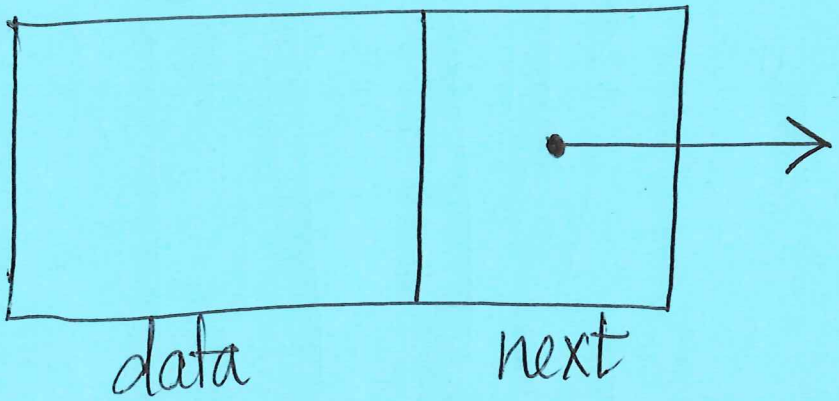
160 students in CS 354 section 2.

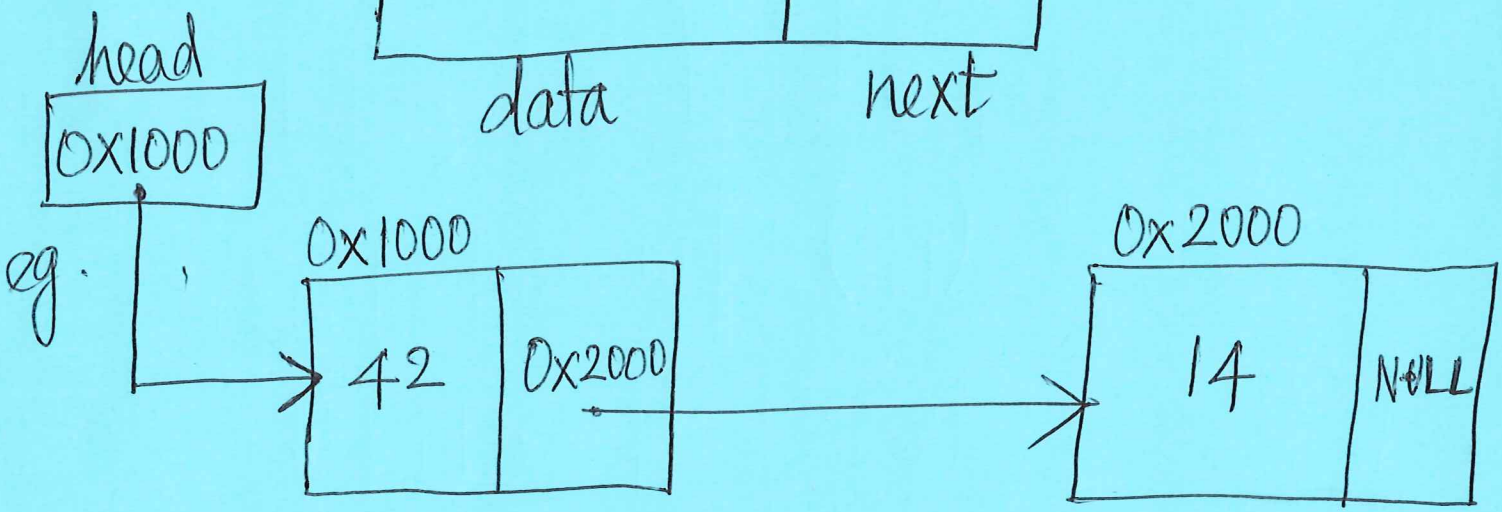
```
student[i].name
student[i].age
```

Self-referential structures

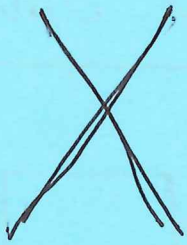



```
struct node {
    int data;
    struct node *next;
};
```





```
struct node {
    int data;
    struct node subnode;
};
```



Illegal!

But pointer to the same structure is legal!