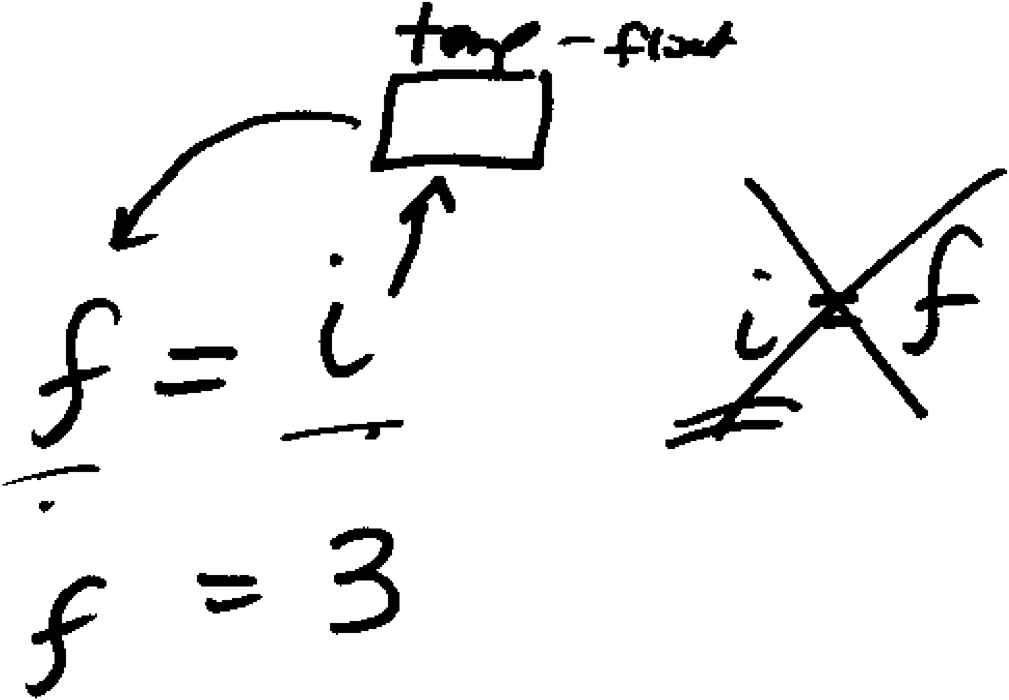

temp-float
f = i
f = 3
i ≠ f

Cast `i = (int)fj;` ©, C#

functional notation `i = int(f);` C#

Cast `ptr = (char*)originptr`

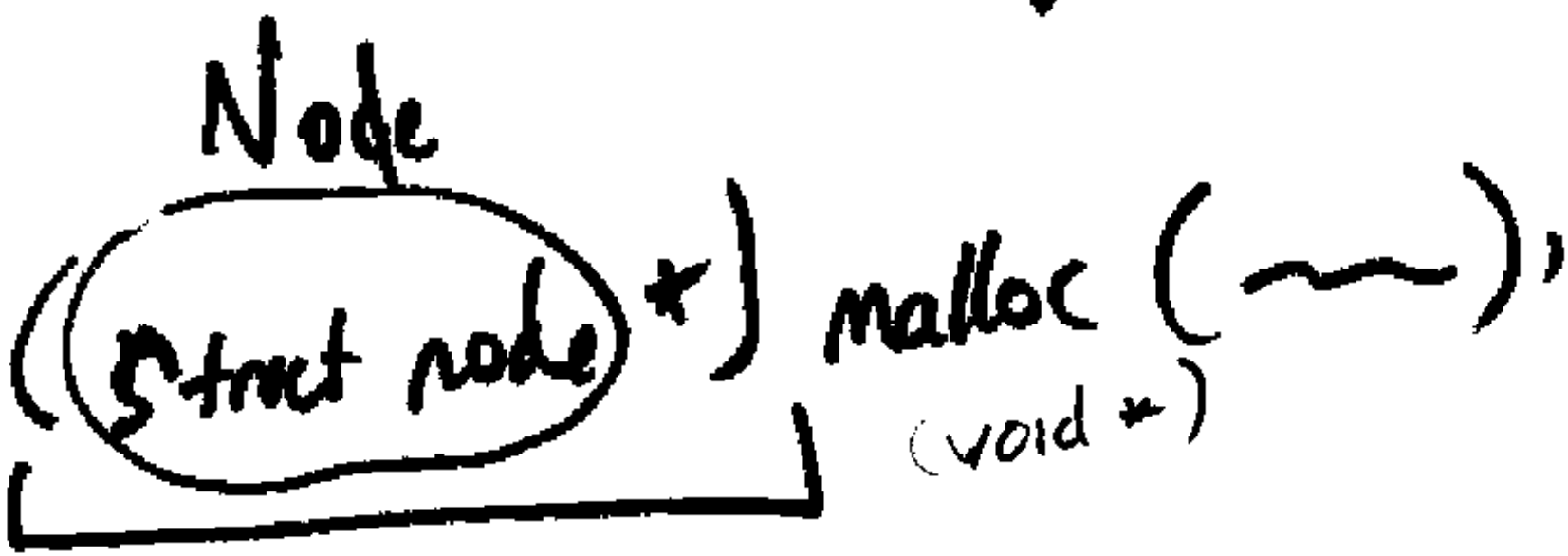
~~`ptr = char*(f)`~~

NO!

Explicit Conversions

cast

```
i = (int) f;
```



Functional Notation

~~char*~~ pchar

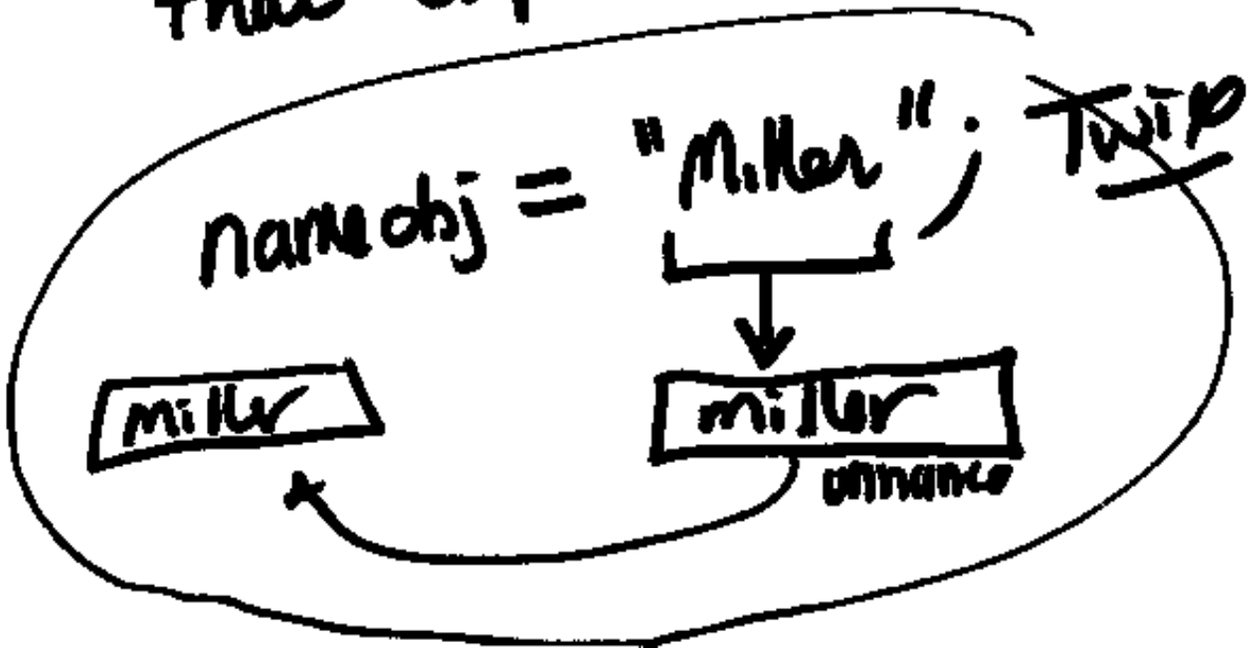
i = int(f);

data

data member → str = new char [strlen(org)+1];
strcpy(str, org);

```
class name
{   public:
        name (char *org);
        name & operator = (const name &);
```

Allow a char * to be used in any context that expects a name obj



Data member char * str

```
class name
{
    public:
        operator Kstring();

name::operator Kstring()
{
    Kstring obj;
    obj.copy(str);  —①
    return obj;     —② cc
}
```

Kstring — class type

int array[5];

array

array[i] = 10; ptr } *(array+i)

int * ptr = array; // ptr=&array[0];

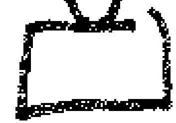
*ptr = 10;

ptr[i] = 10;

*(ptr+i)

&(array + 0)

0 * sizeof element



temp

Pointers to Functions

```
int p1;
int *p2;
int *p3[4];
int *p4();          <- prototype!
int (*p5)();
int *(*p6)();
int (*p7)(float);
```

p5 is a pointer that could point to a function that takes no args and returns an int.

```
int display();
int statement();
int (*p5)();
```

~~p5 = &display();~~

```
p5 = display;
```

```
i = p5();
```

```
try
{
    f();
}
catch(...)
{
}
```