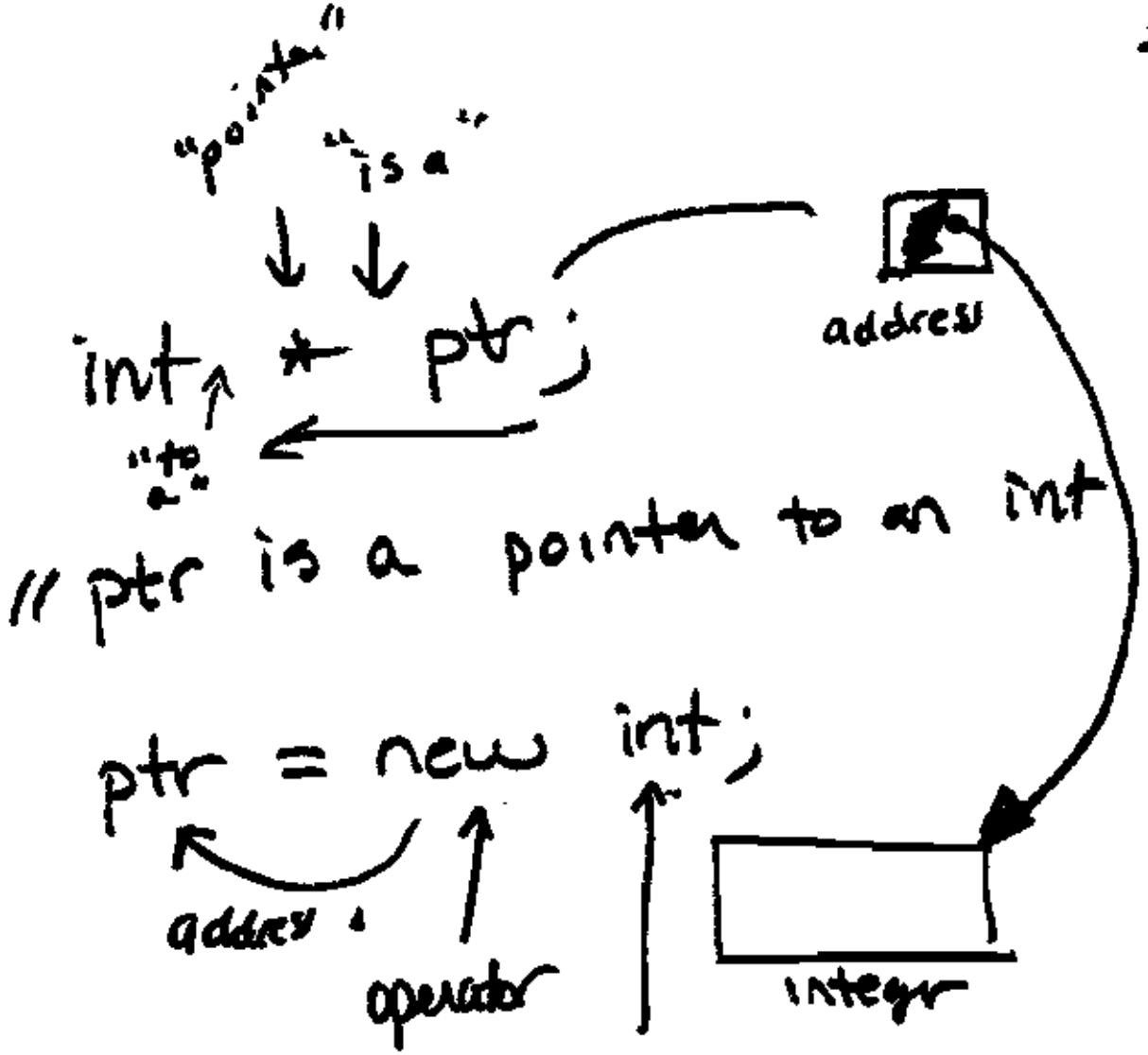
Pointers

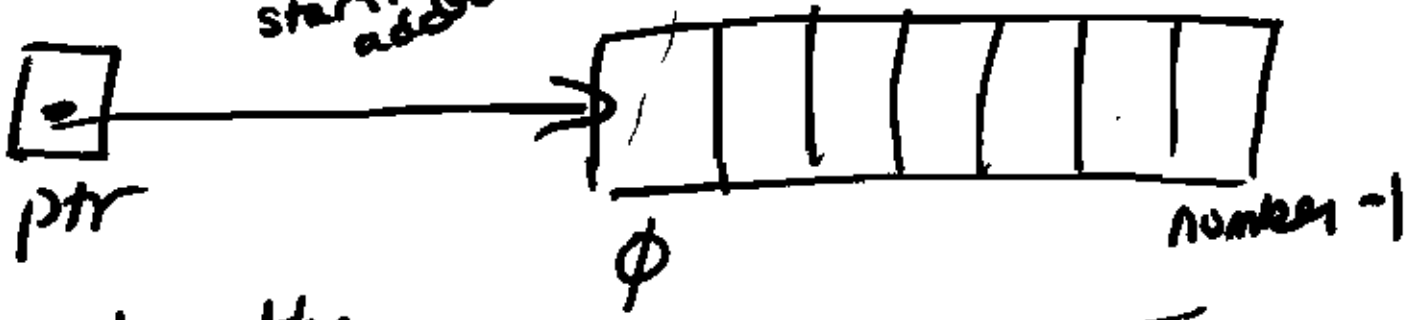

"pointer"
"is a"
int * ptr;
"to a"
address
// ptr is a pointer to an int
ptr = new int;
address
operator
data type
integer
ptr = new int [number];

```
int * ptr = NULL;
or  int * ptr = ∅;
or  int * ptr(∅);
```

int *ptr

ptr = new int[number];

starting address

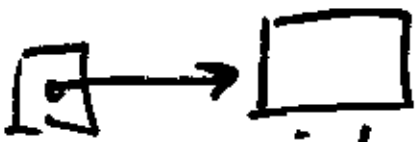


ptr has the address of an integer - the first

number

int number;
:
cin >> number;
:

ptr = new int;



ptr has the address of an integer

- A pointer can point to one item or the first of many

ptr[0] = 10;
ptr[1] = 20;

ptr = new int;

ptr

int 101

ptr = 101;

*ptr = 101;

Dereference

"Go To" the memory address stored in the pointer variable

Statically Allocated Arrays

"at compile time"

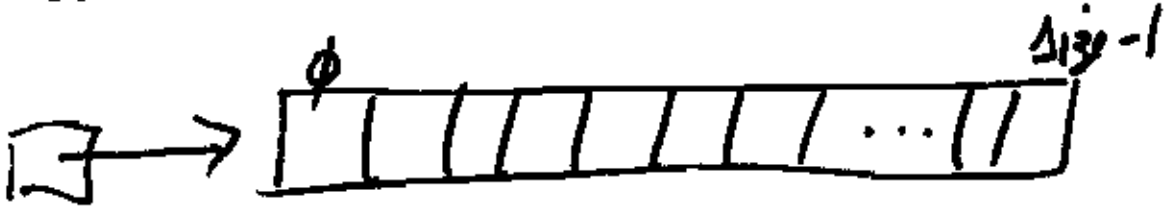
```
char name[21];
```

constant or literal

name ← constant pointer to the first element

Dynamically Allocated Arrays

```
char * array;
array = new char [some_size];
```



array

Decision is made at run time for the size

name[ϕ] = 'a';

cin >> name;

Statically

---

array[ϕ] = 'a';

cin >> array;

dynamically

name [i] === array[i]

*(name + i)

pointer to first element "address" → name

index "integer" → i

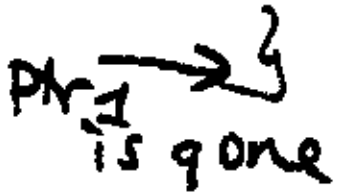
offset → index

i * sizeof(char) → "integer"

address of element in question

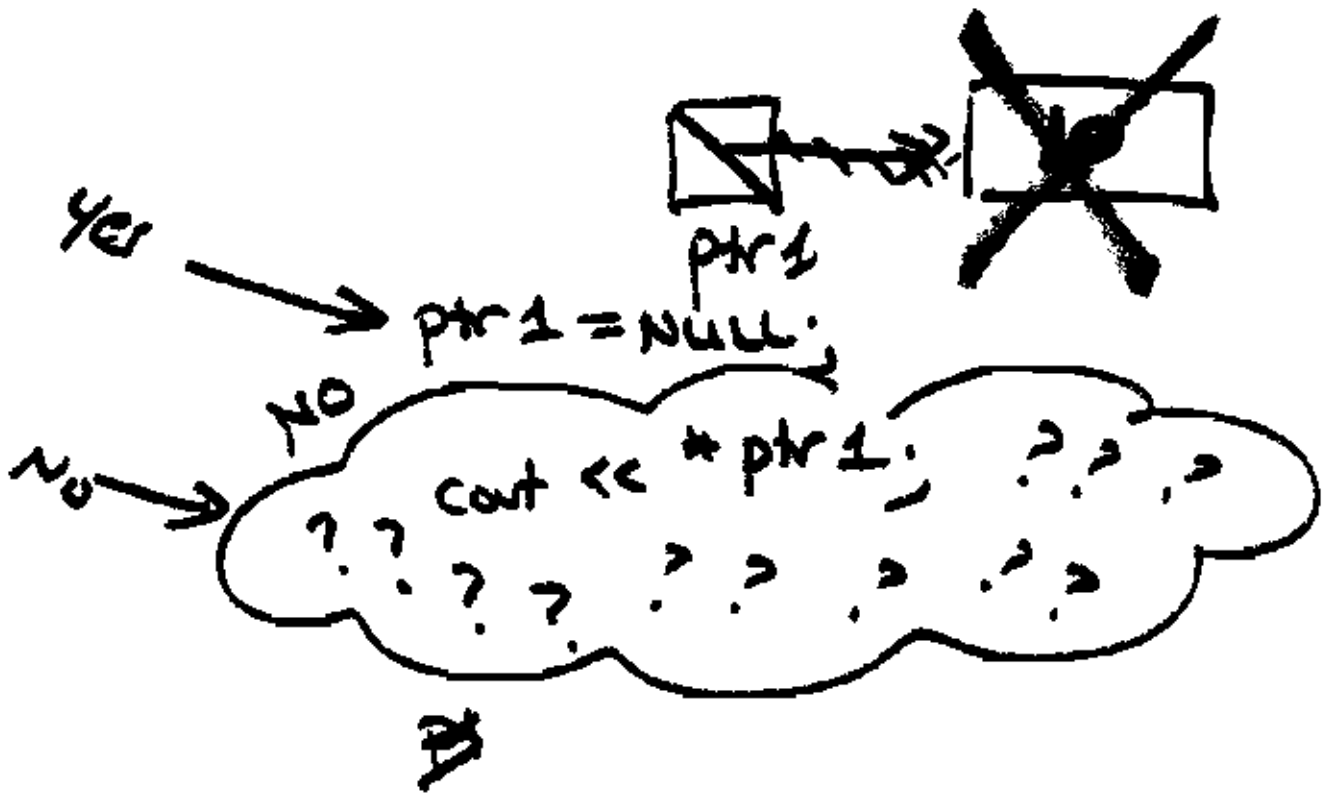
Go There

```
{
   int num;
   int *ptr1; //Local variable
   ptr1 = new int;
   if (ptr1 != NULL) // if (ptr1)
    {    *ptr1 = 10;
```



delete ptr1;

release the memory that ptr1 is pointing at



Dynamically Allocated Arrays

```
char temp[100];
char * name;
   }
cin.get(temp, 100); cin.ignore(100,'\n');
   }
name = new char [strlen(temp) + 1];
strcpy (name, temp);
   }
   }
   }
delete [] name;
```

hw1

```
struct time
{   int hour;
    int minute;
};
```

1)

```
struct course
{
    char name[41];
    char number[8];
    char day;
    time start;
    time end;
    char description[101];
};
```

2) course cs162;

3)
```
cin.get (cs162.name, 41);
cin.ignore (100, '\n');
cin.get (cs162.description, 101);
cin.ignore (100, '\n');
```
↑ size of array

4) `int count_students (course &);`

variable name here OK

5) `int function (course [], int num);`

" (course array [], int num)

"

6) void display-names (course array[], int num)

```
{
    for(int i = 0; i< num ; ++i)
    {
        cout << array[i].name
             << endl;
    }
}
```

opt hard!

7) cin.get();

1 byte or 1 char

cin.get(array, $size);

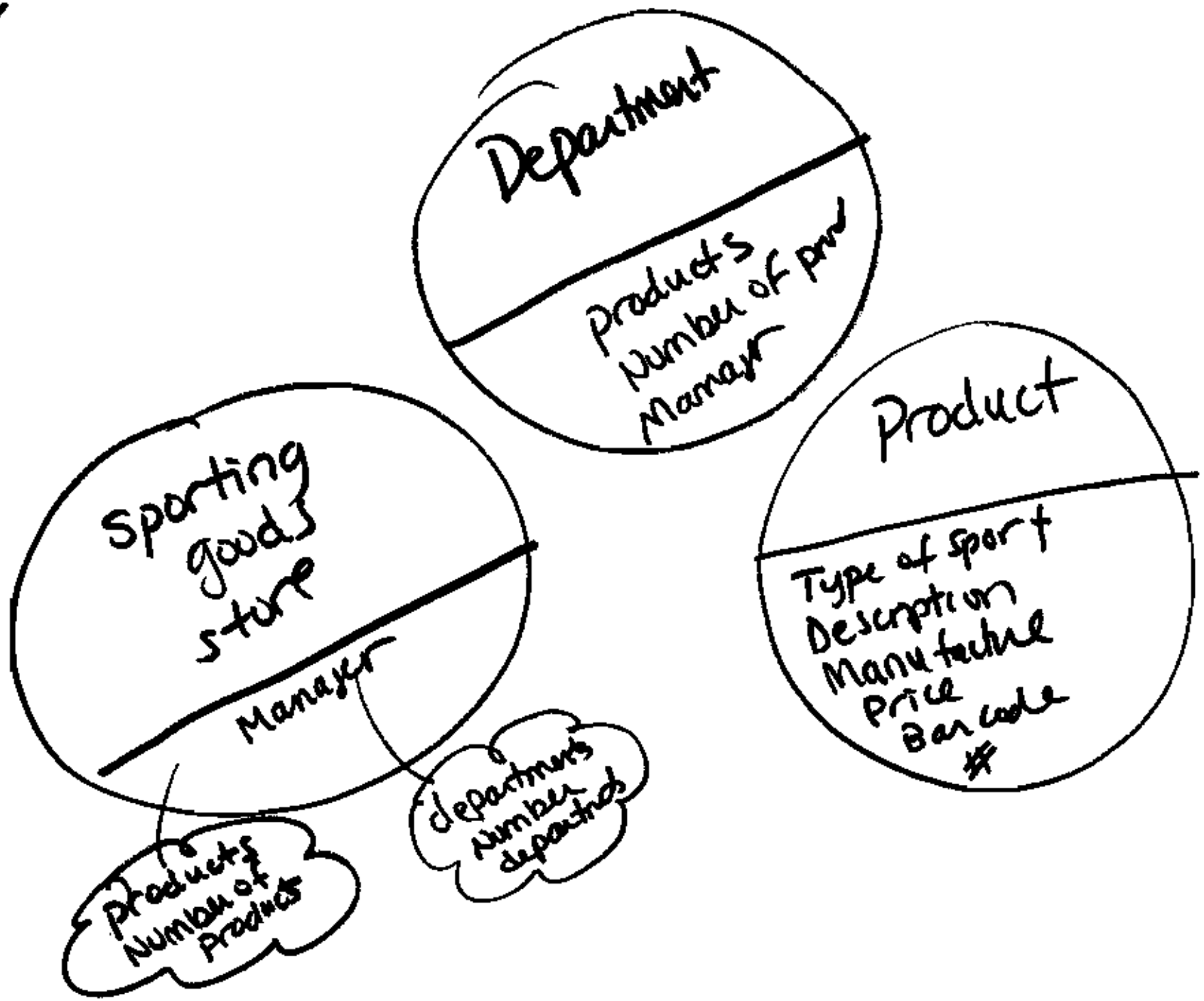
gets size-1 characters or stops when '\n' is encountered but not read

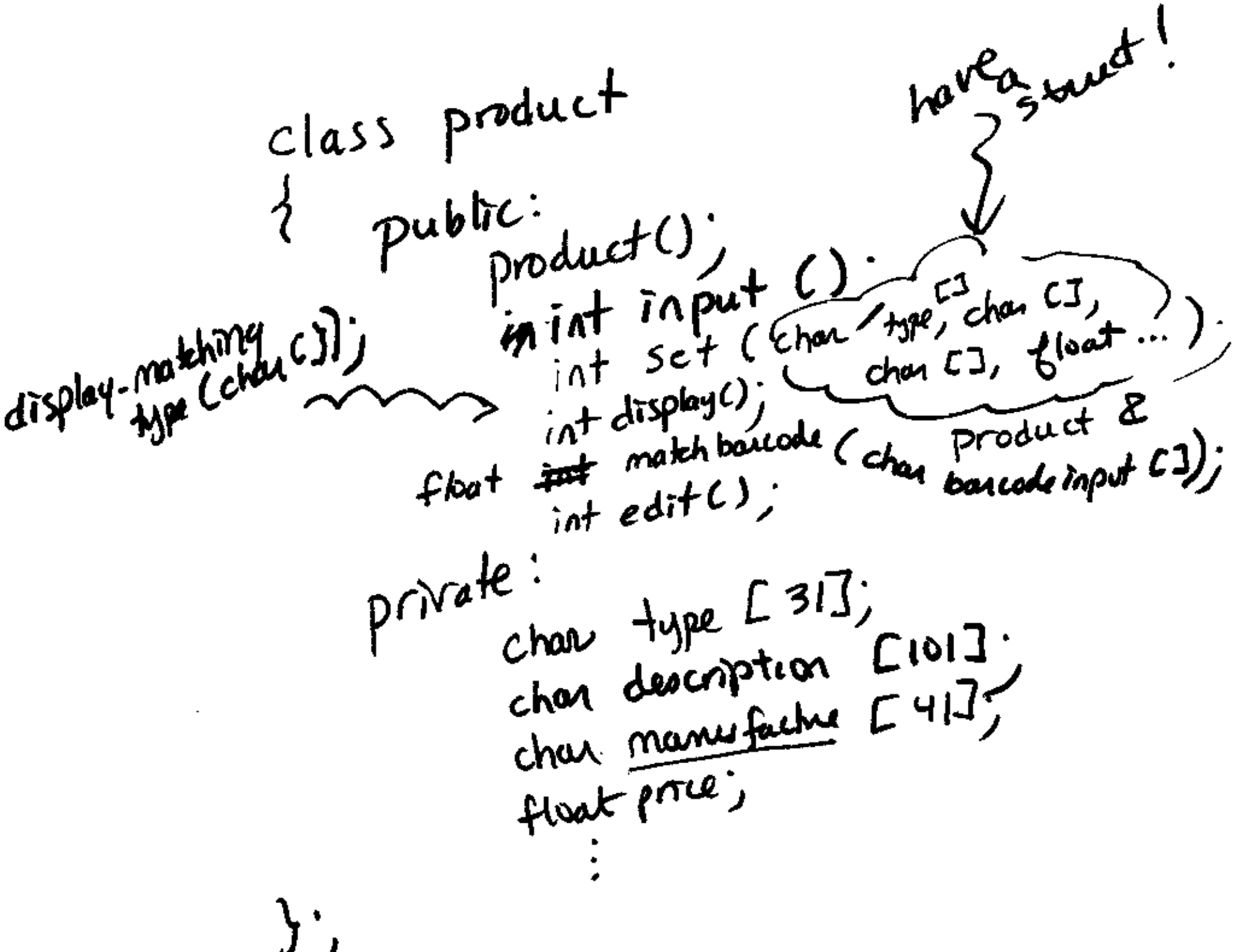
8) `cin.ignore()` ← ignores 1 character/byte

`cin.ignore(100,'\n')` ← ignores 100 or until \n is removed

`cin.ignore(100)` ← ignores 100 or until EOF

Hw3
#1


Department
Products
Number of prd
Manager
Product
Type of sport
Description
Manufacture
Price
Barcode
#
Sporting goods store
Manager
Products
Number of Products
department
Number
departments


class product
{ public:
product();
int input();
int set(char type[], char [], char [], float ...);
int display();
int edit();
display-matching type (char []);
float match barcode (char barcode input [], Product &
private:
char type [3];
char description [101];
char manufacture [41];
float price;
};
have a struct!

```
class store
{   public:
        store();
        int add (product &);
        int display-all();
        int display-type (char type[3]);
    }
private:
    product inventory [500];
    int num_of_products;
};
```

```
Store::store()
{ num_of_products = 0;
}
```

Constant

Store chain [10];