Pointers!

First - What the compiler knows about variables

1. Its name
2. Its type (int or char for example)
3. Its size (in bytes)
4. Its location in memory (its address)
   (we use & to refer to the address)

Location in Memory

```
printf("Memory location int   %p\n", &someInt);
printf("Memory location float %p\n", &someFloat);
printf("Memory location char  %p\n", &someChar);

Memory location  of int       0x0022FF74
Memory location  of float     0x0022FF70
Memory location  of char      0x0022FF6F
```

(almost everyone has gotten these funny values at some time when they forgot the & in scanf)

How does scanf even work???

We've learned by now that functions only copy the values from our variables. They don't have the ability to change a variable outside of its scope.

For example:

```
void someFunction(int num) {
    num = 100;
}
int main() {
    int num = 42;
    someFunction(num);
    printf("%i",num);
}
```

Output: 42

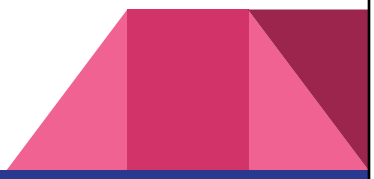
How does scanf even work???

But isn't that precisely what we are asking scanf to do?

BUT - we don't give scanf the value, we give it the ADDRESS

```
scanf("%i", &someNum);      //  NOTICE THE AMPERSAND ( & )
```

So how does this work behind the scenes?

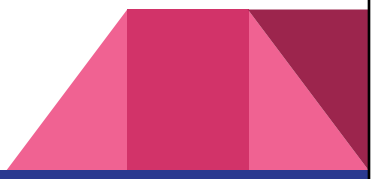


How does scanf even work? - We need a pointer!

A pointer holds a memory address.

They give power because you can manipulate the value of memory directory.

Remember: a pointer is a special kind of variable that holds a memory address.



What is a pointer?

There is a pointer version of every type.

We can declare

int *p;

char *myCharPtr;

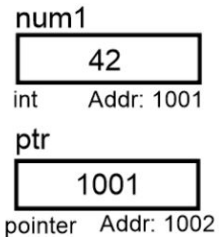
float *aFloatPtr;

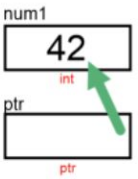
The asterisks denotes it as a pointer.

What is a pointer?

- Pointers do not hold values!
- **They hold ADDRESSES OF OTHER VARIABLES**

```
1     int main() {
2           int num1 = 42;
3           int *ptr;
4           ptr = &num1;
```





Using a pointer

- **Normally, without pointers.**

- Two ways to access information:

int num = 42;

num //refers to the value stored (42)

&num //refers to the address of num

- Special, **using pointers**

- Three ways to access information:

int *ptr = #

ptr //refers to the value of ptr (which is num's address)

&ptr //refers to the address of ptr

*ptr //refers to the VALUE POINTED TO (which is num's value: 42)

//Called "dereferencing" the ptr

Example usage

```
int *ptrInt;                    // declares a pointer to an Integer.
int iNum = 7;

ptrInt = &iNum;                 // assigns the pointer to the address of iNum

printf("%i\t",  *ptrInt);       // prints the value being POINTED TO by ptrInt
*ptrInt = 9;                    // assigns value to address being POINTED TO
printf("%i\t",  iNum);          // Notice that iNum is now 9

Output: 7       9
```

So back to scanf - how does it work?

The scanf function receives the address into a pointer as a parameter!

```
void someFunction(int *num) {
    *num = 100;
}
int main() {
    int num = 42;
    someFunction(&num);
    printf("%i",num);
}
```

Output: 100

Normally, when we pass parameters we are
Passing By Value

When we pass the address like this - we are
Passing By Reference

Gotchas

Watchout! Be careful when using the ++ operator with pointers.

`*num++` will result in increment the pointer, THEN dereference it.

Make sure you either:

```
(*num)++;
*num += 1;
*num = *num + 1;
```