Week 5

- Pointer

- Call by Value
- No way to modify g inside the method
- We have to return the modified Gun
- This sucks!

```c
typedef struct Gun{
  int ammo;
  int clip;
  int damage;
} Gun;

Gun pew(Gun g){
  if(g.clip > 0){
    g.clip--;
  }else{      //Todo check ammo
    g.ammo = g.ammo - 25;
    g.clip = g.clip + 25;
  }
  return g;
}

int main(){
    Gun g = {100, 25, 250};
    g = pew(g);
    printf(„%i\n",g.clip);
    return 0;
}
```

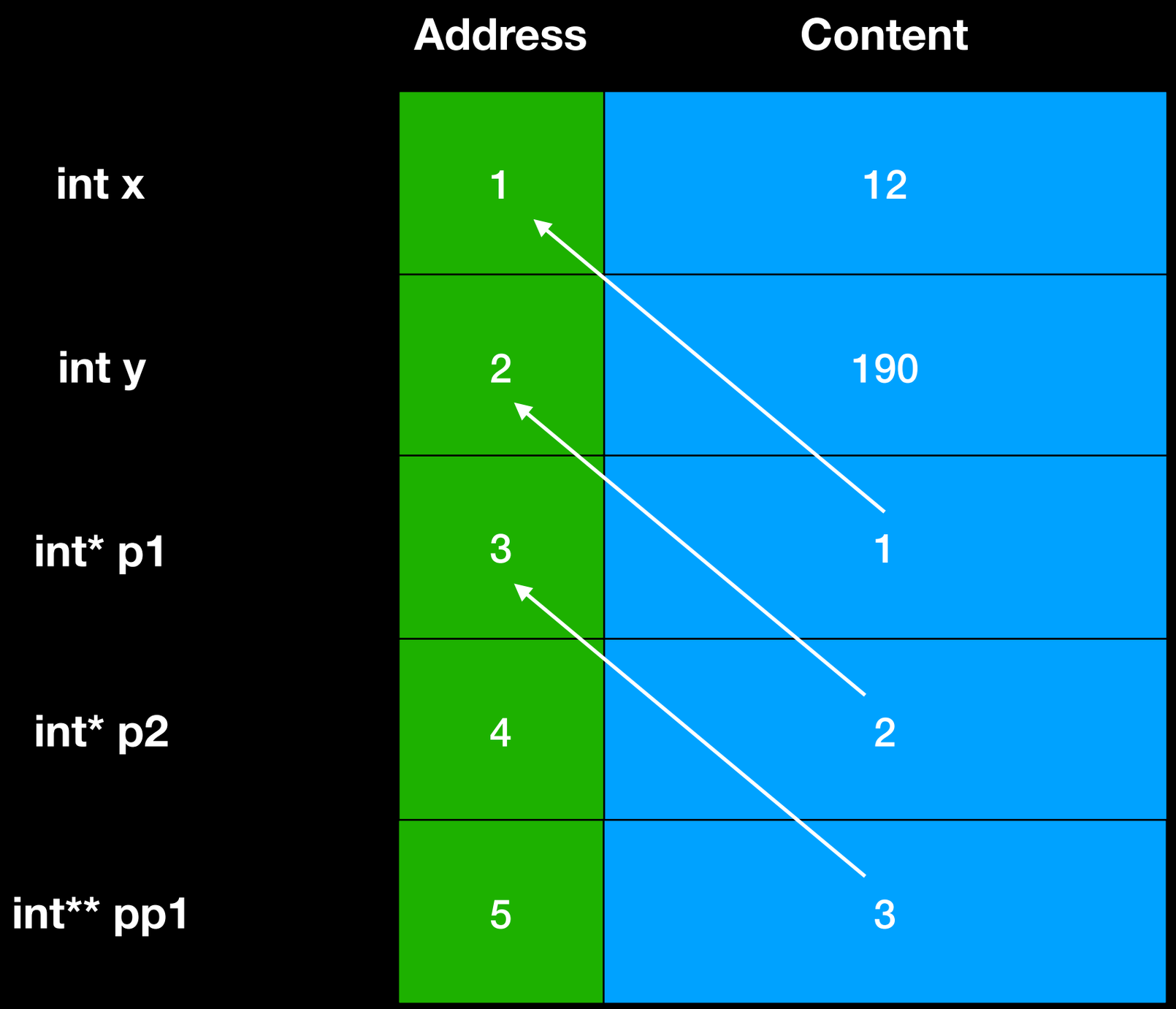
Pointer

What is a pointer?

- A variable that stores the address of another variable
- www.google.de vs actual html/javascript
- TYF45392(2) instead of the actual book
- LF113 instead of a room

Notation

- A pointer is an addition to a type
- int stores a number
- int* stores an address of an int
- int** stores an address, which stores an address of an int
- * gives a hint how often you may look up what is at that memory address


Address
Content
int x
1
12
int y
2
190
int* p1
3
1
int* p2
4
2
int** pp1
5
3

&

- & returns the memory address of an object

```c
int main(){
    int x = 2;
    printf("%i\n",x);
    printf("%p\n",&x);
    return 0;
}
```

```
2
0x7ffeedc989e8
```

*

- * „dereferences“ a pointer
- Look up what is stored at that address

```c
int main(){
    int x = 0;
    int* p = &x;
    printf("%i\n",x);
    printf(„%i\n",p);
    printf("%i\n",*p);
    return 0;
}
```

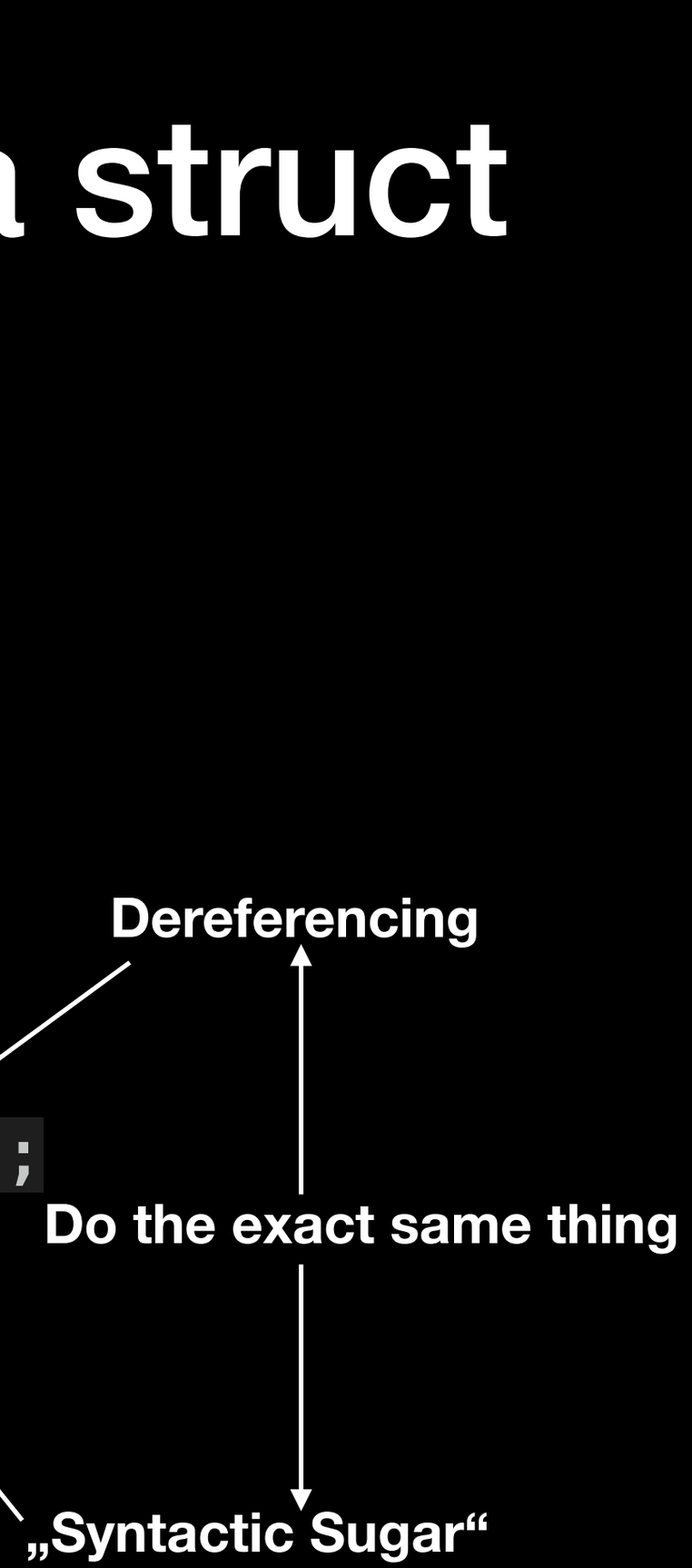
```
0x7ffeedc989e8
0
0x7ffeedc989e8
0
```

What is stored at that memory position?

Accessing data in a struct

```c
typedef struct{
    float x;
} FancyFloat;

int main(){
    FancyFloat f;
    FancyFloat *ptr;
    ptr = &f;
    printf("%f\n", f.x);
    printf("%f\n",(*ptr).x);
    printf("%f\n",ptr->x);
    return 0;
}
```



```c
typedef struct Gun{
  int ammo;
  int clip;
  int damage;
} Gun;

Gun pew(Gun g){
  if(g.clip > 0){
    g.clip--;
  }else{
    g.ammo -= 25;
    g.clip += 25;
  }
  return g;
}

int main(){
    Gun g = {100, 25, 250};
    g = pew(g);
    printf(„%i\n",g.clip);
    return 0;
}
```

```c
typedef struct Gun{
  int ammo;
  int clip;
  int damage;
} Gun;

void pew(Gun *g){
  if(g->clip > 0){
    g->clip--;
  }else{
    g->ammo -= 25;
    g->clip += 25;
  }
}
int main(){
    Gun g = {100, 25, 250};
    Gun *gPtr = &g;
    pew(&g);
    pew(gPtr);
    printf("%i\n",g.clip);
    return 0;
}
```

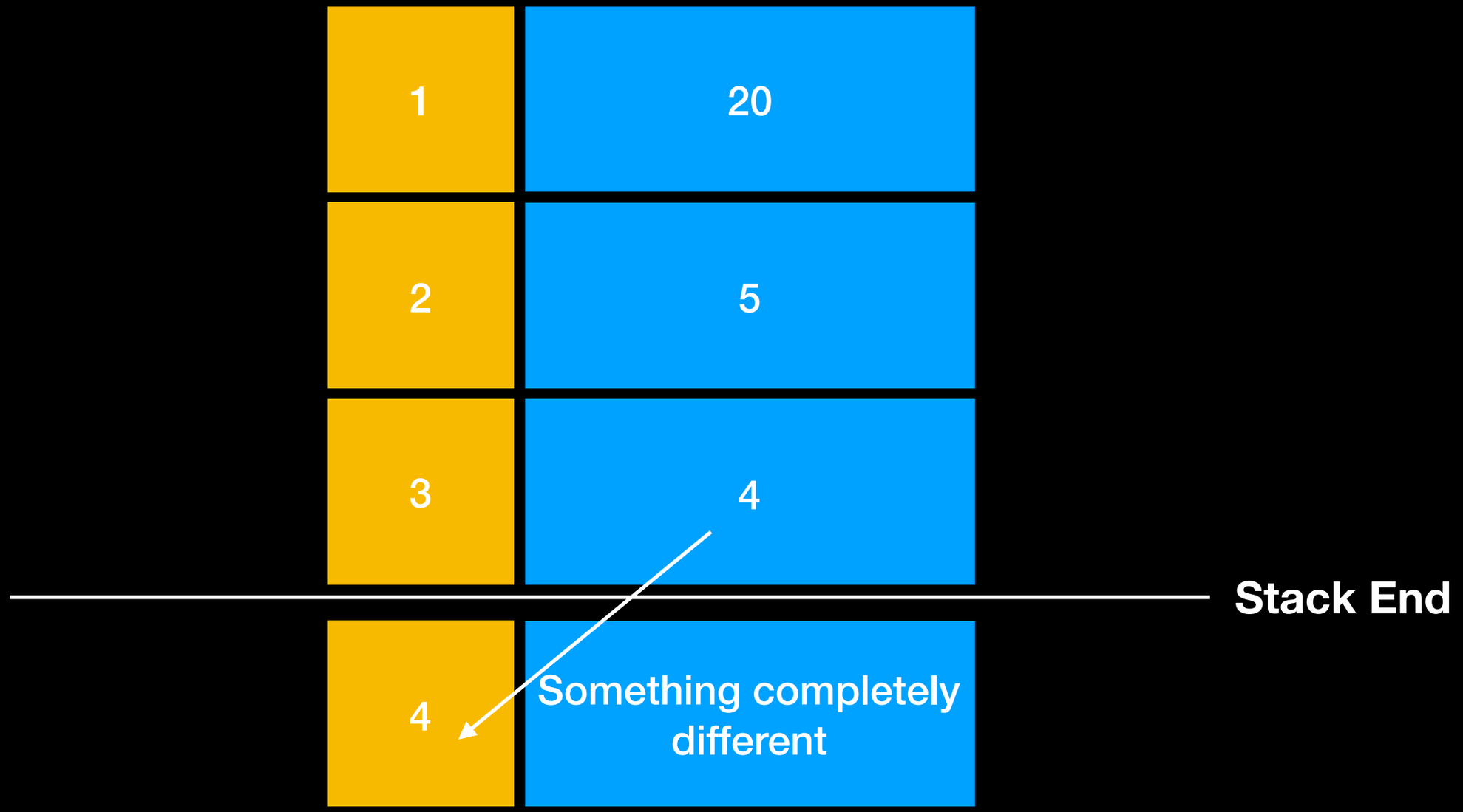
Deadly Sin

Deadly Sin

```
Gun* newGun(){
  Gun g = {100, 25, 250};
  return &g;
}
```

You are returning a Stack address

```
int main(){
    Gun *g = newGun();
    pew(g);
    printf("%i\n",g->clip);
    return 0;
}
```


1
20
2
5
3
4
Stack End
4
Something completely different

Rule 4:

Never return a stack address

Exercises!

- Fix your Fortnite Methods from last week to use pointers wherever possible
- Find out how much memory a pointer needs on your system (Try creating two pointers and see what offset they have)
- Write a method that gets two addresses and returns the bigger address as a pointer
- Write a method that makes a pointer point to NULL if the number it points to contains a 3