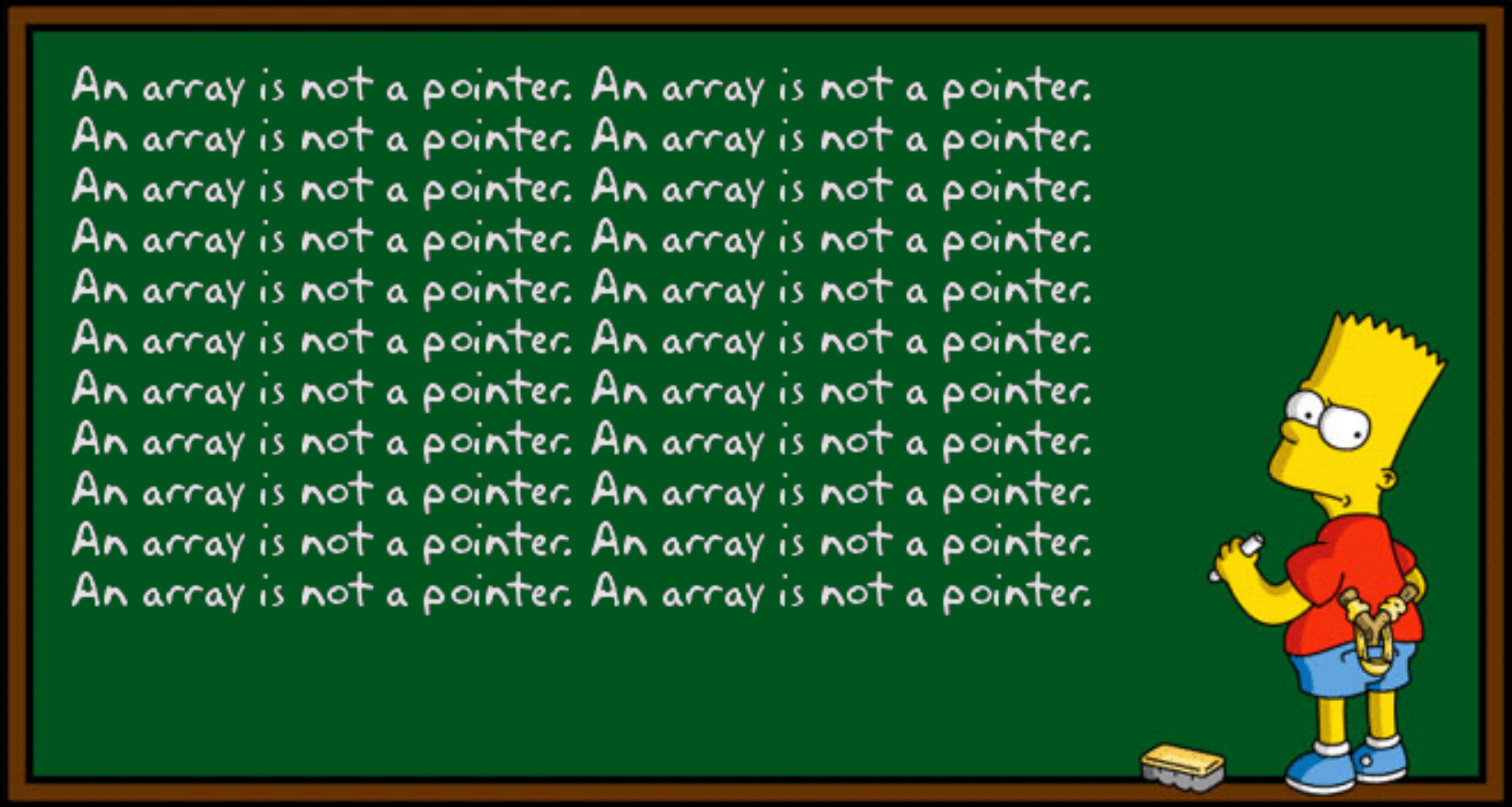
CSE102
Computer Programming



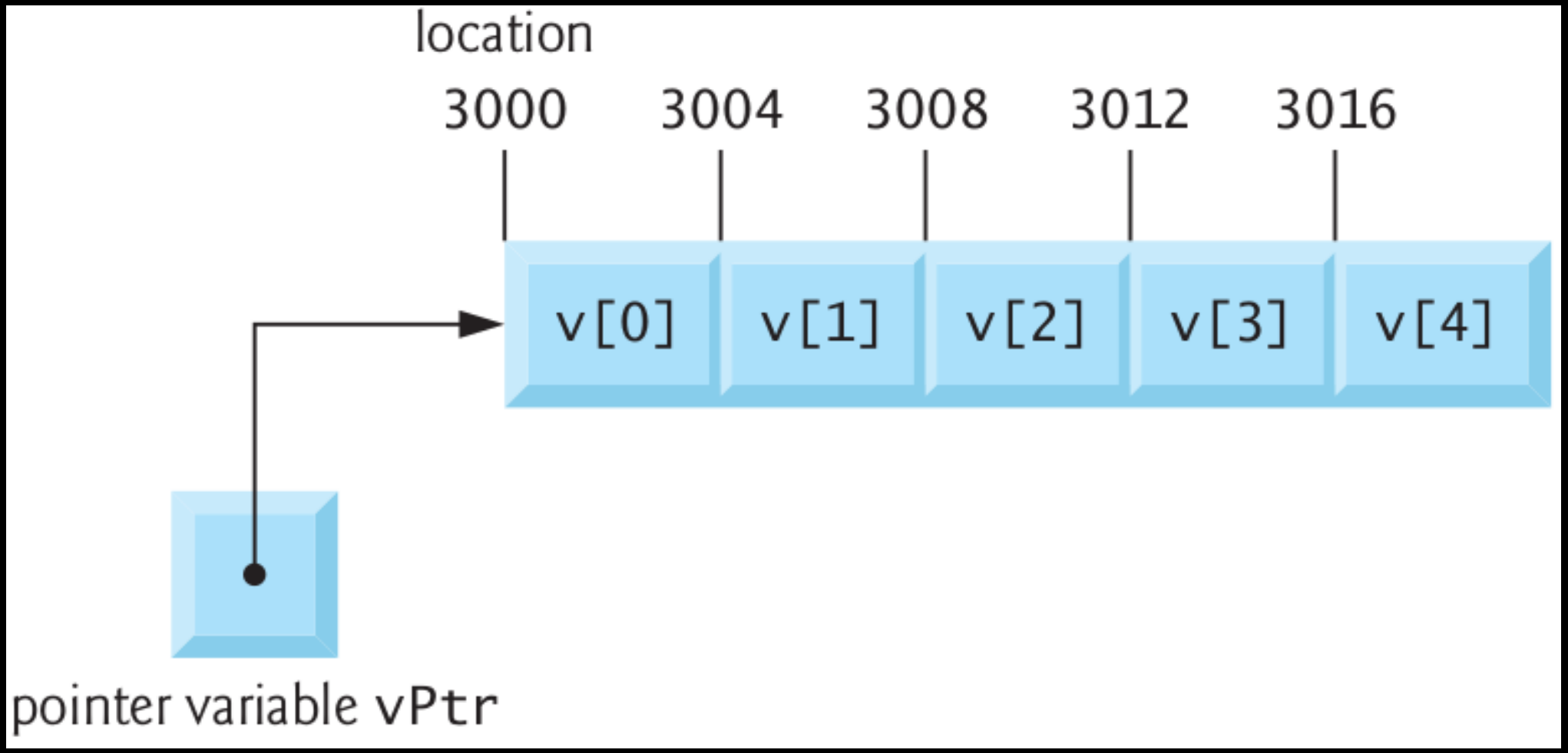
Credits: i.imgur.com

Remember

& fetches address

* de-references address

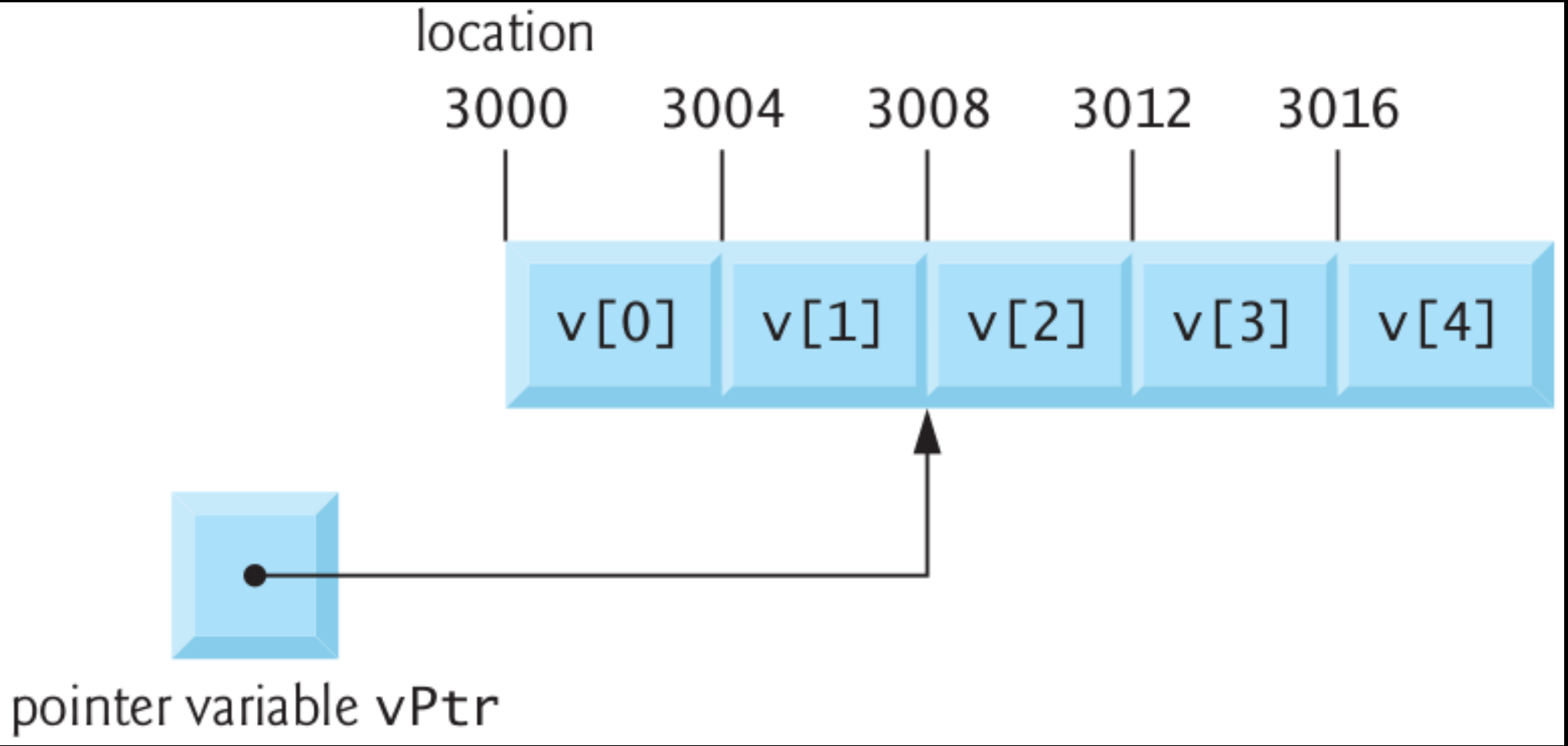
Pointer to an Array




```
vPtr = &v[0];
```

Credits: How to Program C

Pointer to an Array




```
vPtr = &v[2];
```

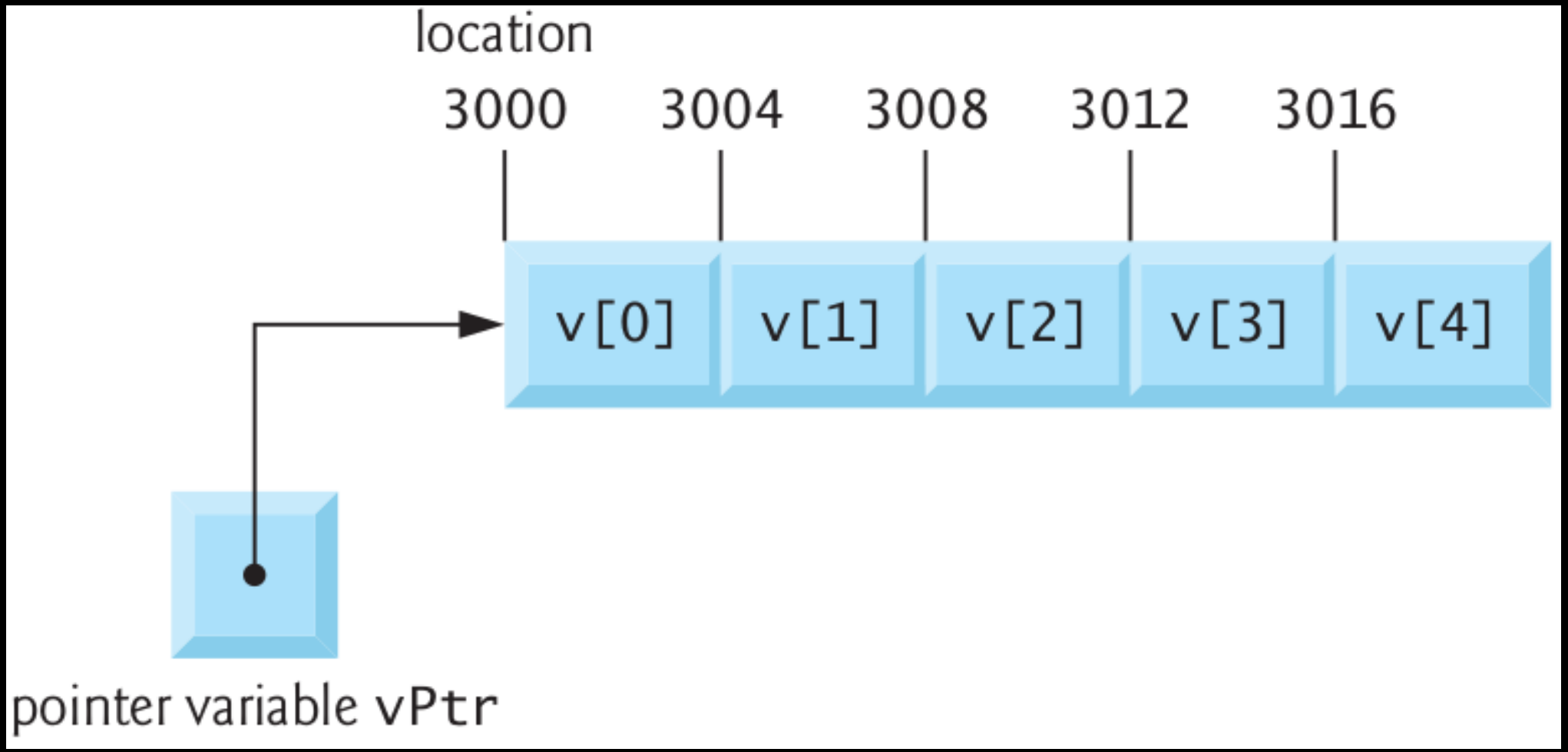
Credits: How to Program C

Pointer to an Array

```
for(i=0; i<5; i++){
  vptr = &v[i];
  printf("%p ", vptr);
}
```

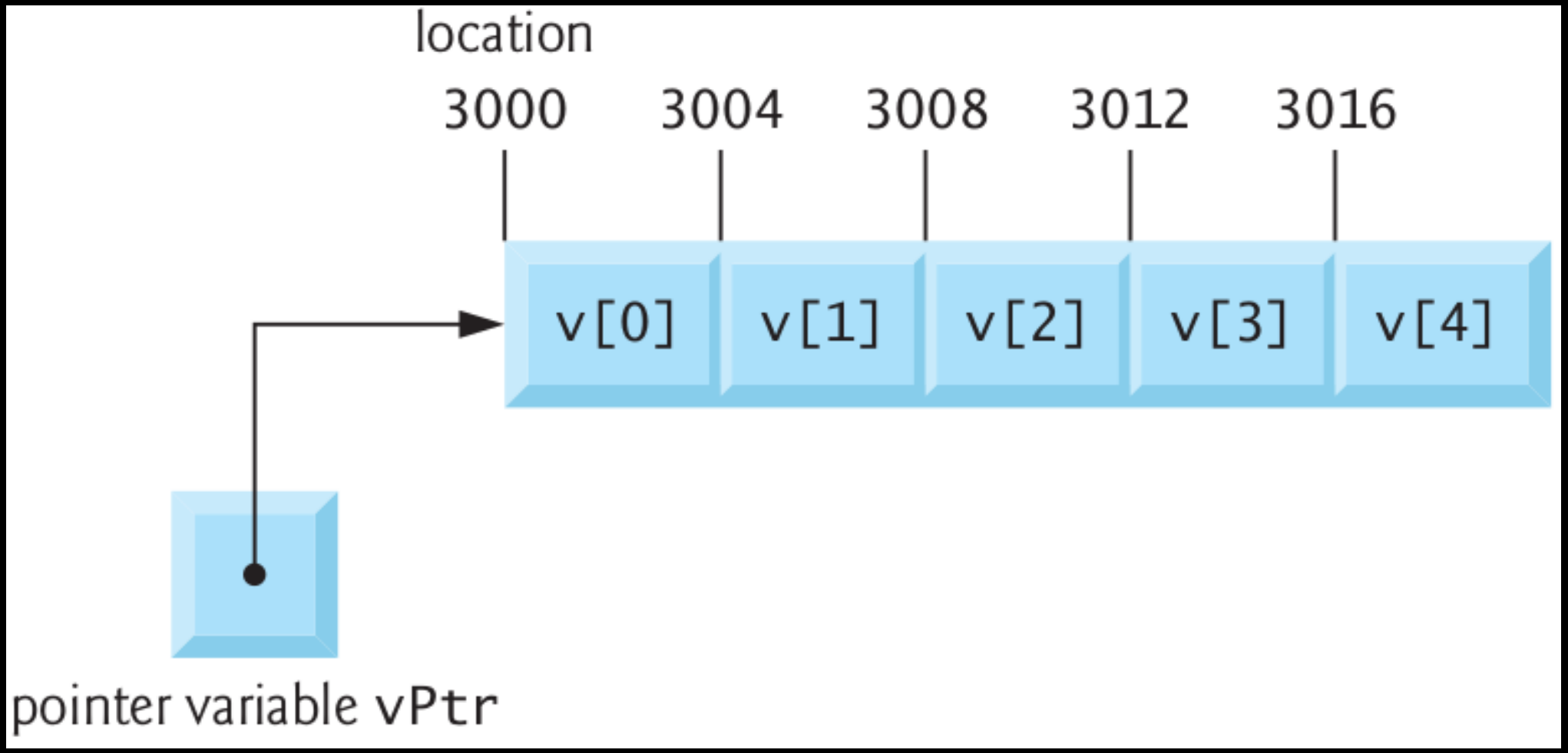
// will print 3000 3004 3008 3012 3016
// is there a better way?

Array Organization



Credits: How to Program C

Array Organization




```
vPtr = vPtr + sizeof(int) * i
```

Array Organization

```
vPtr = vPtr + sizeof(int) * i
       vPtr = 3000 + 4 * 0
       vPtr = 3000 + 4 * 1
       vPtr = 3000 + 4 * 2
       vPtr = 3000 + 4 * 3
       vPtr = 3000 + 4 * 4
```

Pointer Arithmetic

```
vPtr = vPtr + sizeof(int) * i
       vPtr = 3000 + 0
       vPtr = 3000 + 1
       vPtr = 3000 + 2
       vPtr = 3000 + 3
       vPtr = 3000 + 4
```

// Interestingly multiplying sizeof(int)
// with index is intrinsic for pointers

Pointer Arithmetic

```
vPtr = &v[0];
for(i=0; i<5; i++){
 printf("%p ", vPtr+i);
}
```

// will print 3000 3004 3008 3012 3016

Increment Pointer

```
vPtr = &v[0];
for(i=0; i<5; i++){
 printf("%p ", vPtr++);
}
```

// will print 3000 3004 3008 3012 3016

How About This?

```
vPtr = &v[4];
for(i=0; i<5; i++){
 printf("%p ", vPtr-i);
}
```

// will print 3016 3012 3008 3004 3000

Array Organization

```
vPtr = vPtr - sizeof(int) * i
     vPtr = 3016 - 4 * 0
     vPtr = 3016 - 4 * 1
     vPtr = 3016 - 4 * 2
     vPtr = 3016 - 4 * 3
     vPtr = 3016 - 4 * 4
```

Array Organization

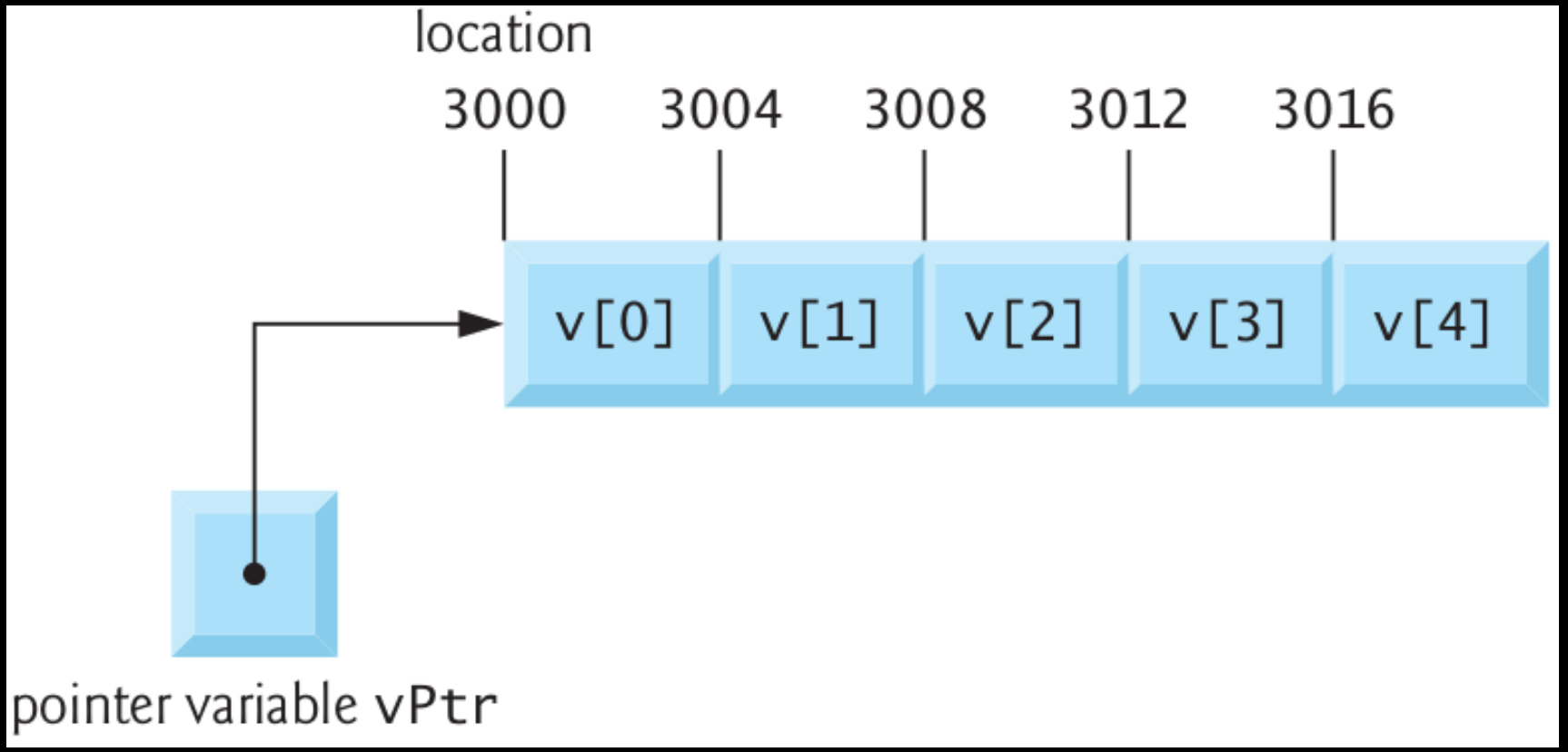
```
vPtr = vPtr - sizeof(int) * i
       vPtr = 3016 - 0
       vPtr = 3016 - 1
       vPtr = 3016 - 2
       vPtr = 3016 - 3
       vPtr = 3016 - 4
// Interestingly multiplying sizeof(int)
// with index is intrinsic for pointers
```

Decrement Pointers

```
vPtr = &v[4];
for(i=0; i<5; i+=){
 printf("%p ", vPtr--);
}
```

// will print 3016 3012 3008 3004 3000

Pointer Arithmetic




```
v2Ptr = &v[4]
v2Ptr-vPtr = 4!!
(3000+4)-(3000+0) = 4
```

Credits: How to Program C

Pointer Arithmetic

Pointer arithmetic is undefined
Unless performed on an array

Pointer Arithmetic

```
vPtr = &v[0];
for(i=0; i<5; i++){
 printf("%p ", vPtr+i);
}
```

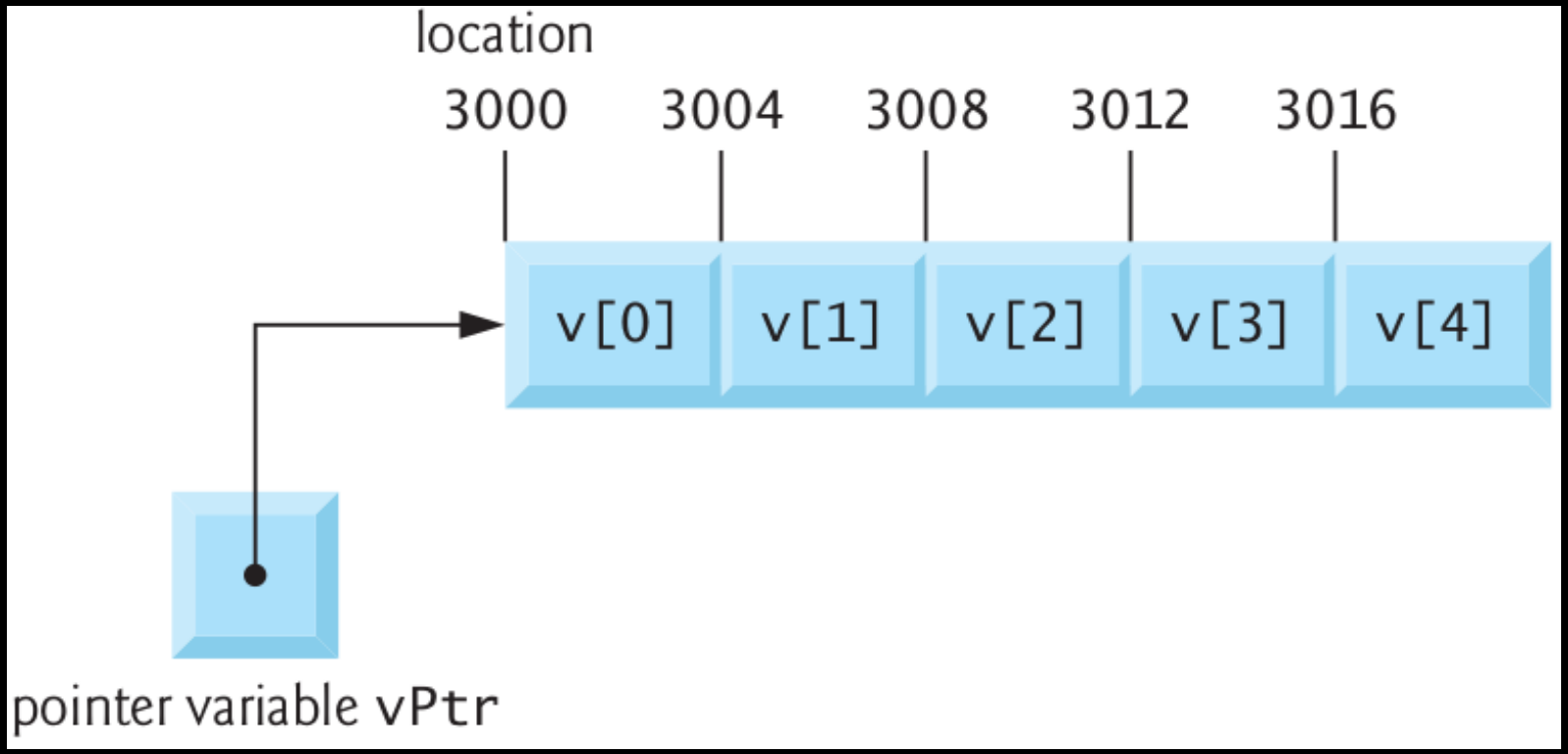
// will print 3000 3004 3008 3012 3016

Pointer Arithmetic

```
vPtr = &v[0];
for(i=0; i<5; i++){
 printf("%p ", vPtr+i);
}
```

// will print 3000 3004 3008 3012 3016
// is'nt the first line obvious for
// all arrays as matter of fact!!!!

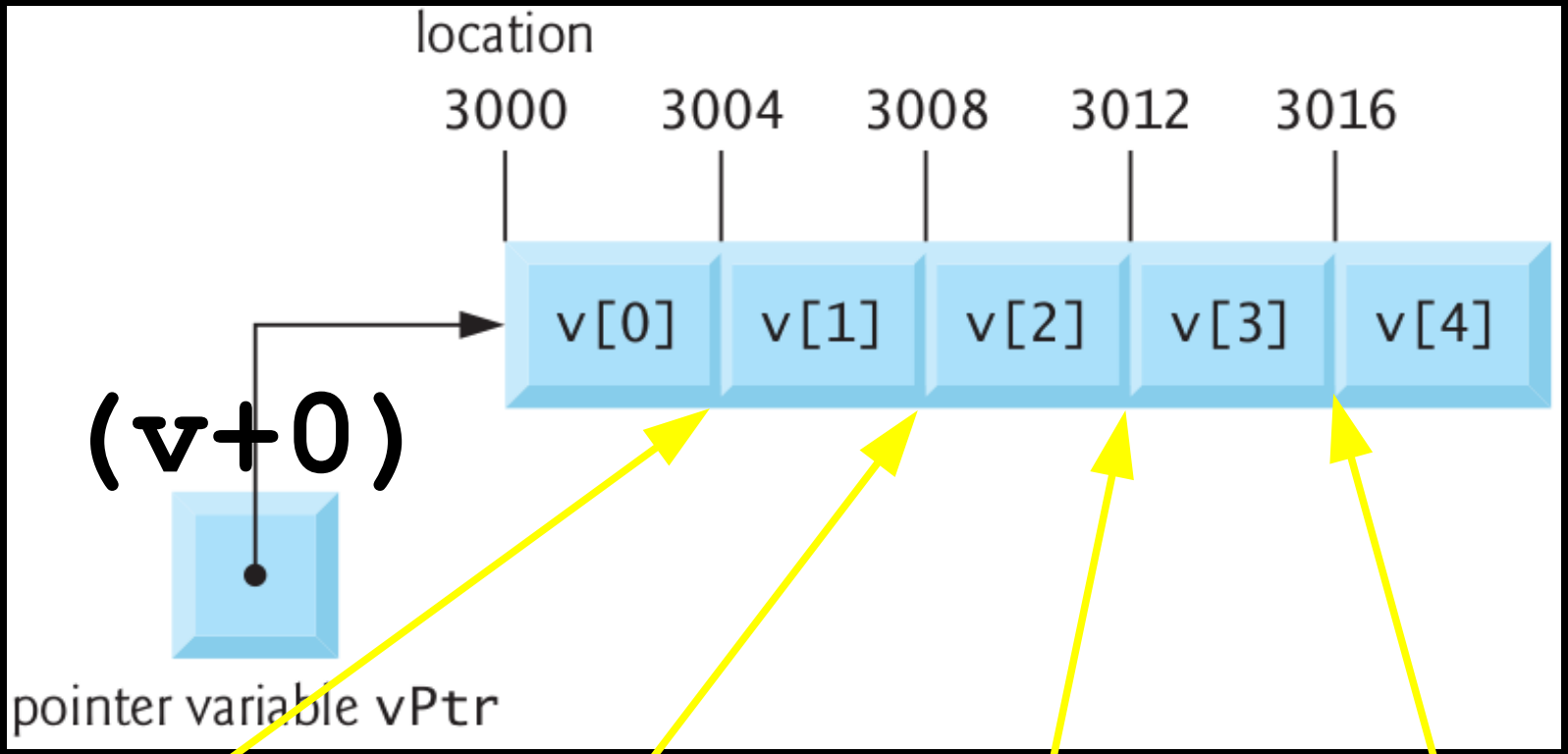
Base Address




```
vPtr = &v[0];
vPtr = v;
```

Credits: How to Program C

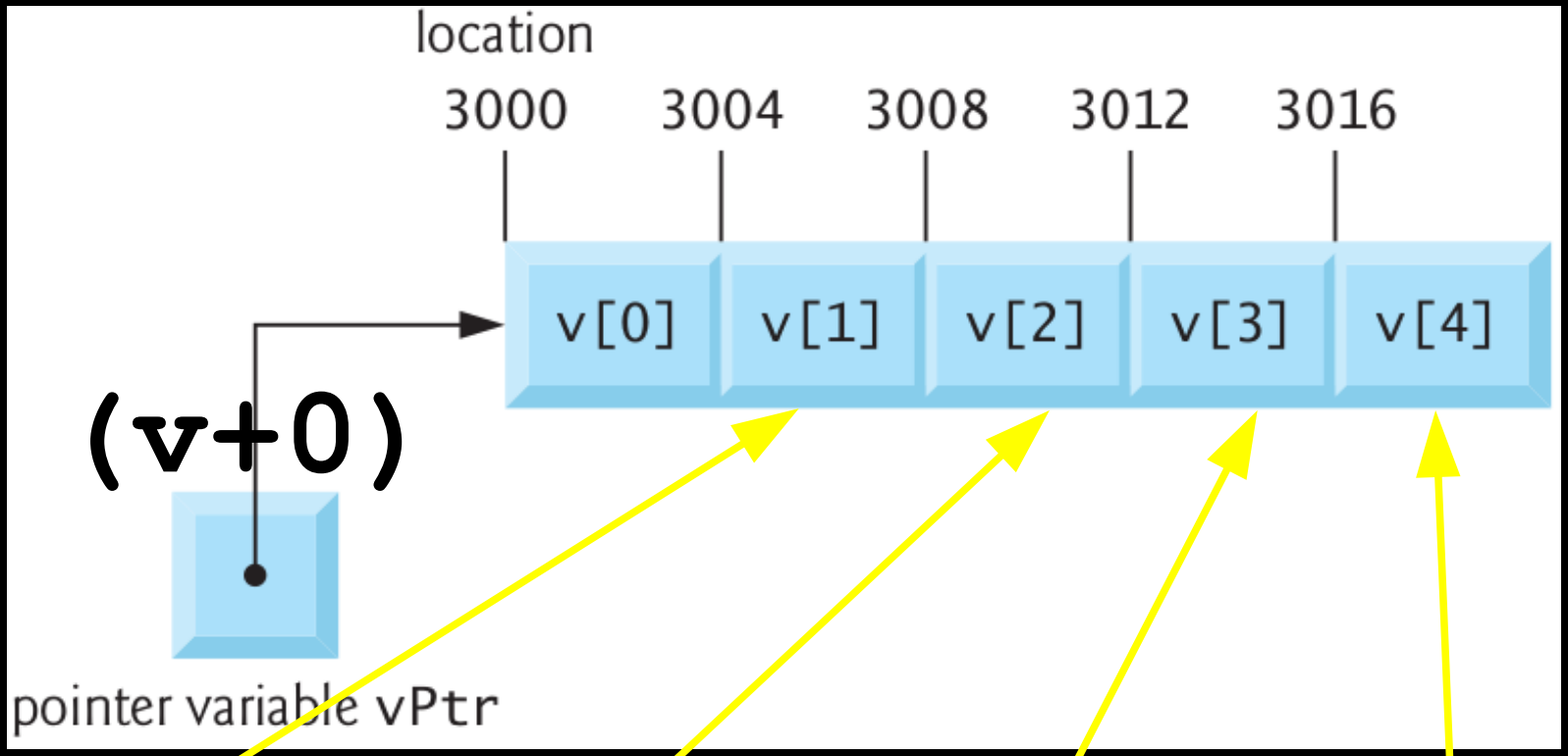
Base Address



(v+1) (v+2) (v+3) (v+4)

Credits: How to Program C

Base Address



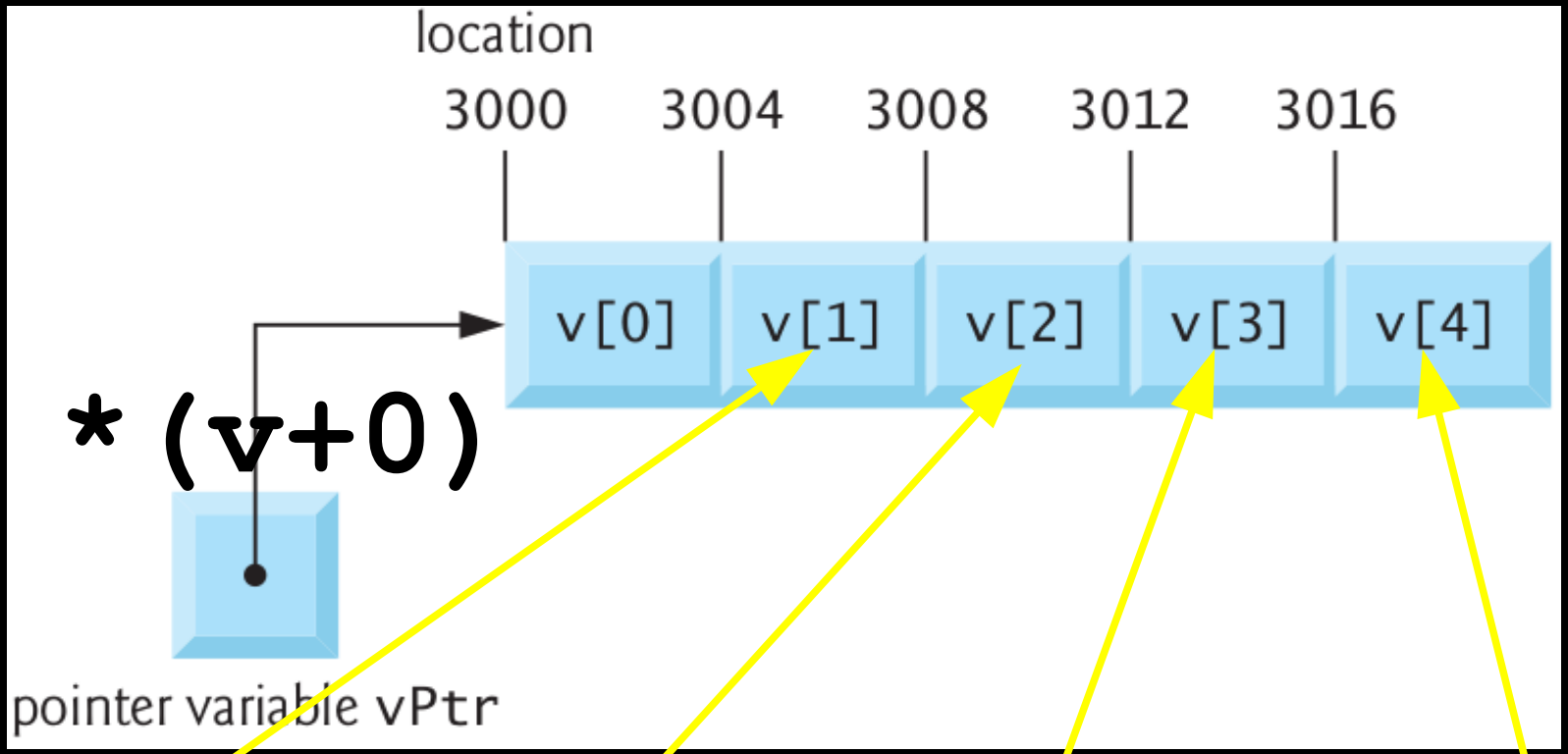
(v+1) (v+2) (v+3) (v+4)

v is a constant pointer

because v += 3 is invalid!

Credits: How to Program C

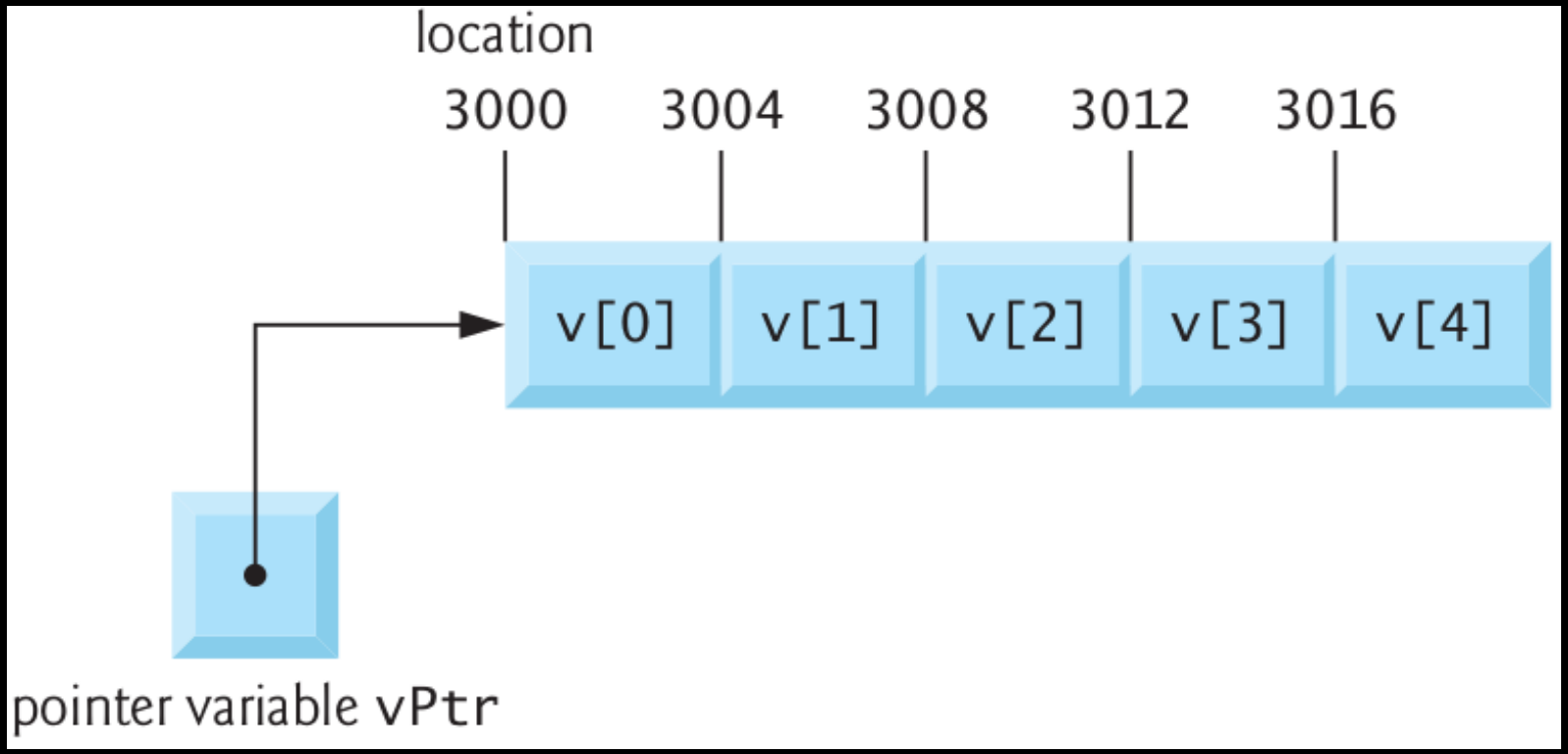
Pointer Offset Notation



***(v+1) *(v+2) *(v+3) *(v+4)**

Credits: How to Program C

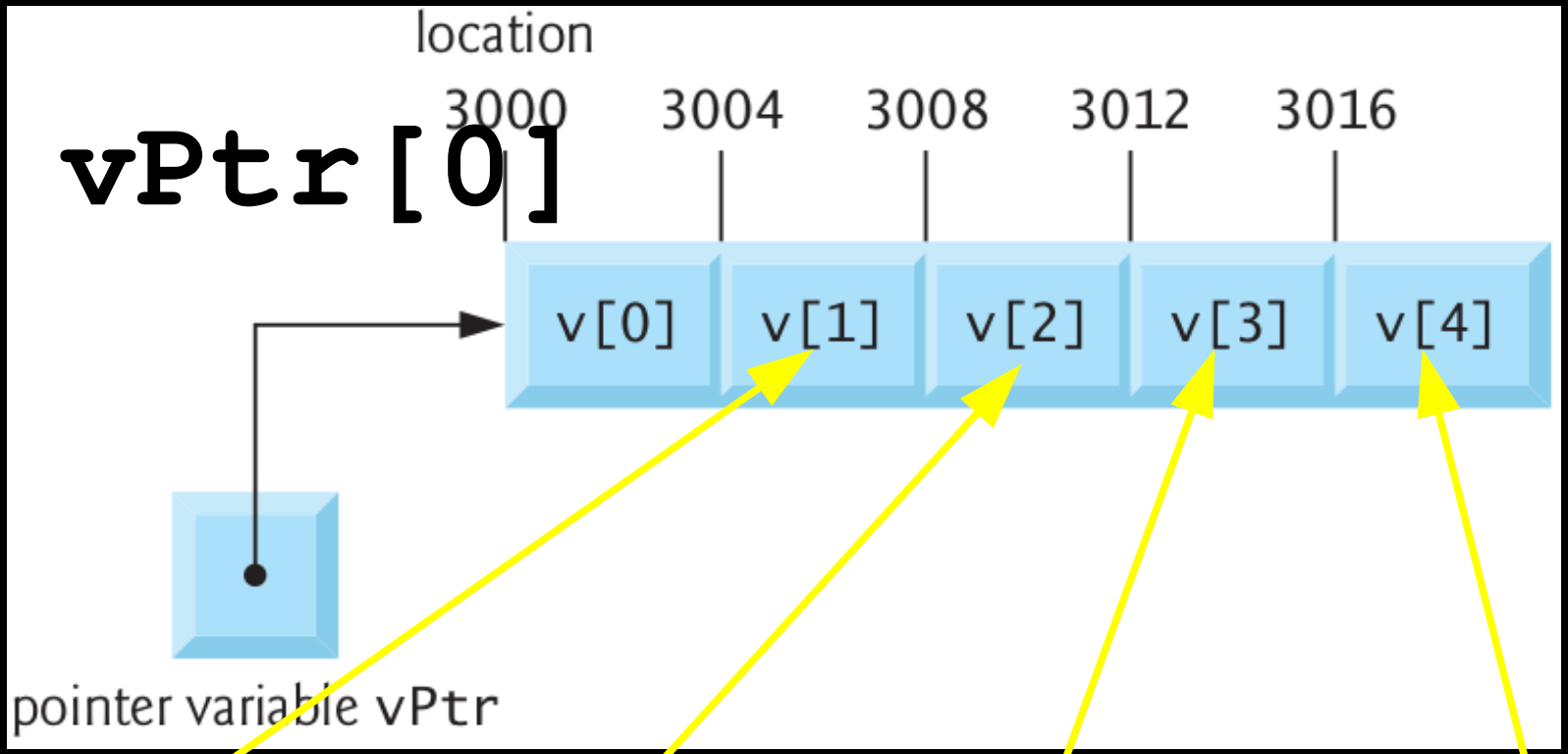
Base Address




```
vPtr = v;
```

Credits: How to Program C

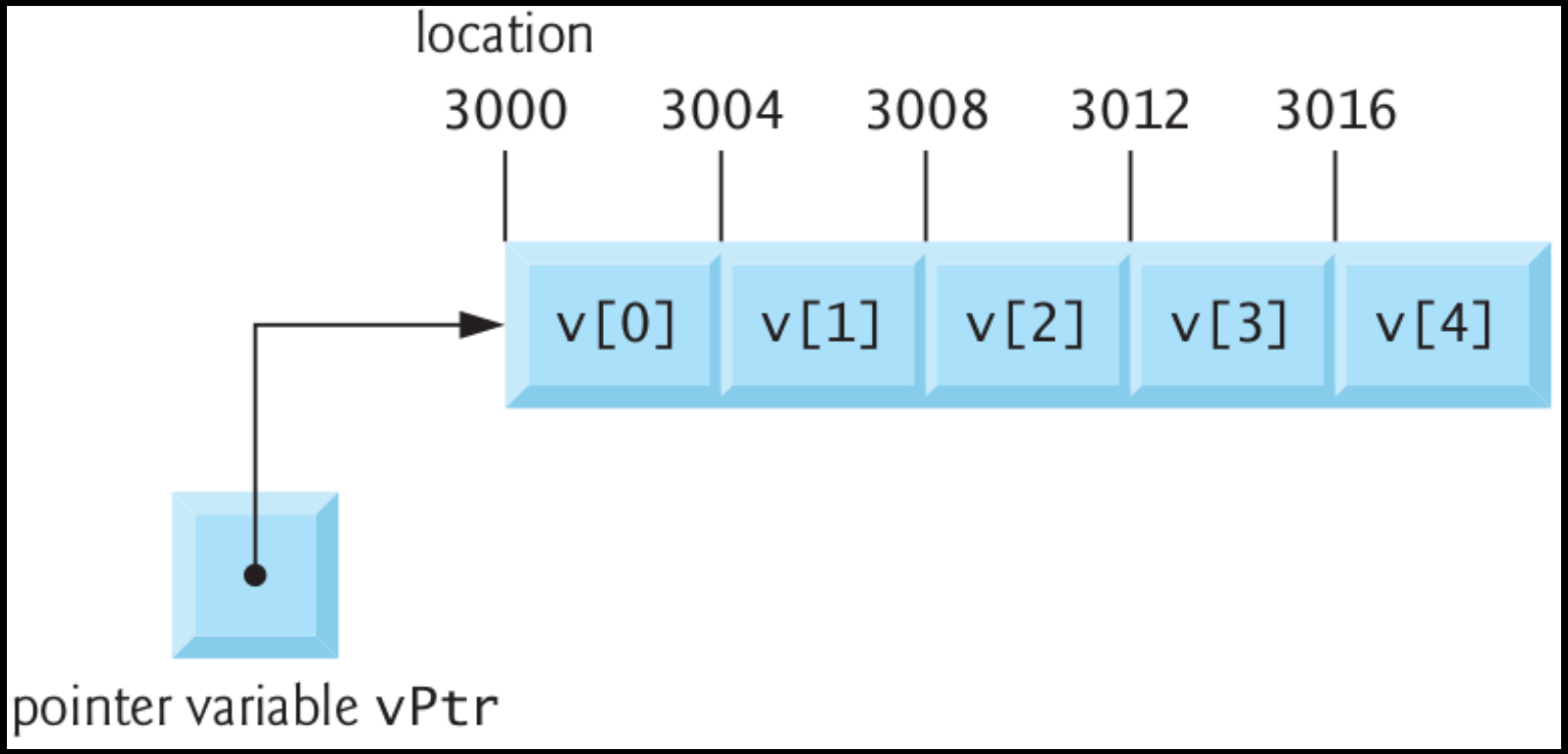
Pointer Subscript Notation



vPtr[1] vPtr[2] vPtr[3] vPtr[4]

Credits: How to Program C

Notations

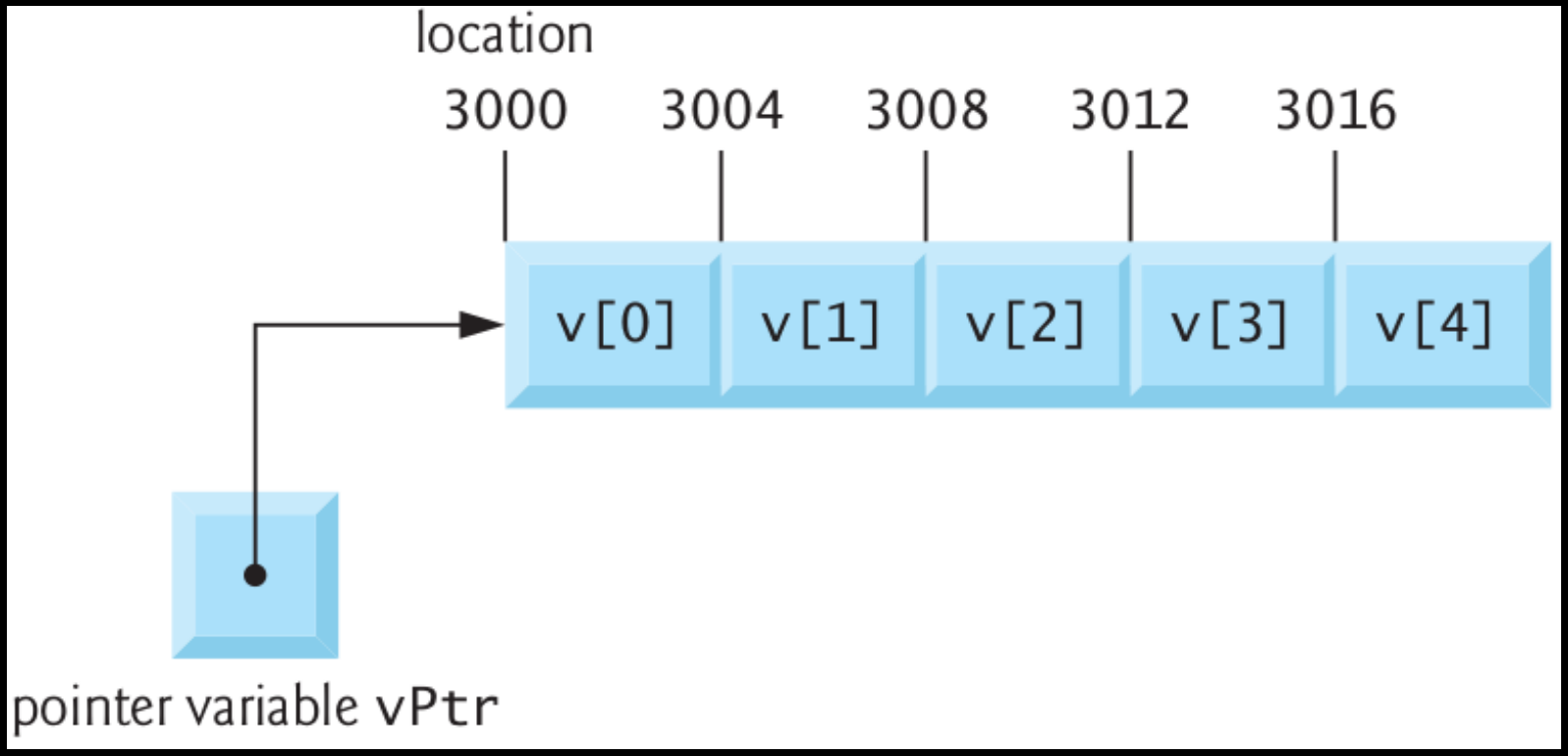


v vPtr (v+0)

(vPtr+0)

Credits: How to Program C

Notations



v[0] vPtr[0]

***(v+0) *(vPtr+0)**

Credits: How to Program C

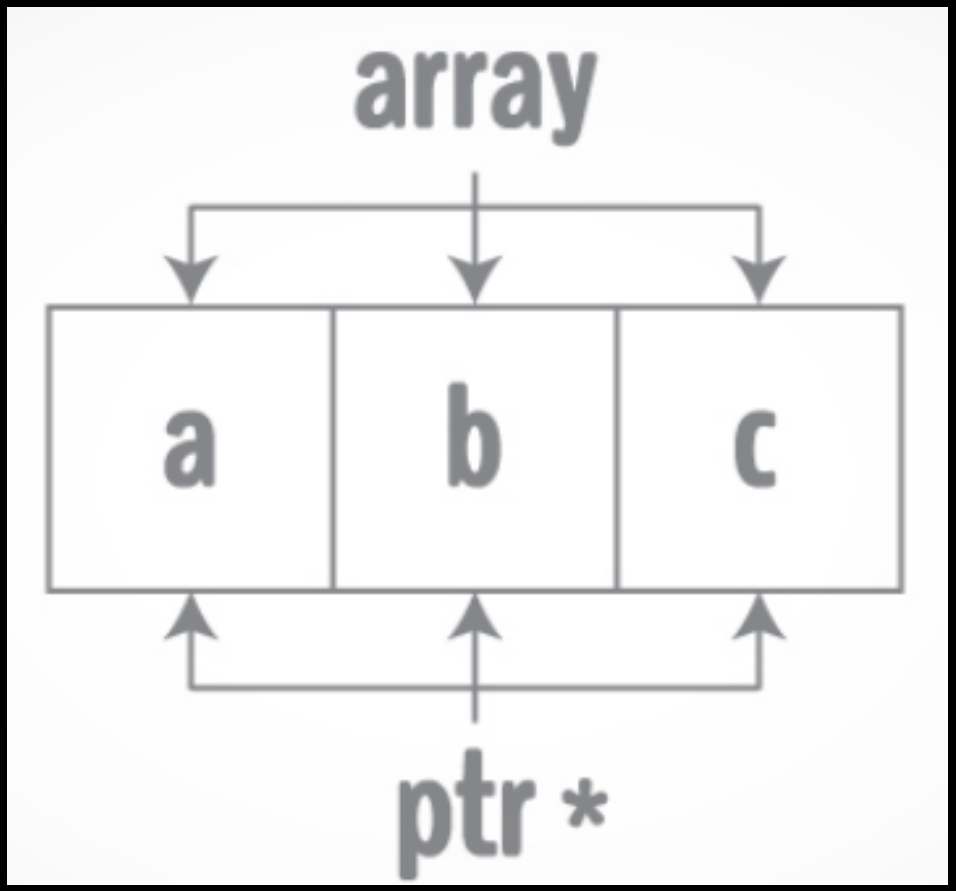
Pointers to Functions

An array name is really the address in memory of first element. Similarly function name is really the starting address in memory of code that performs function's task

Pointers to Functions

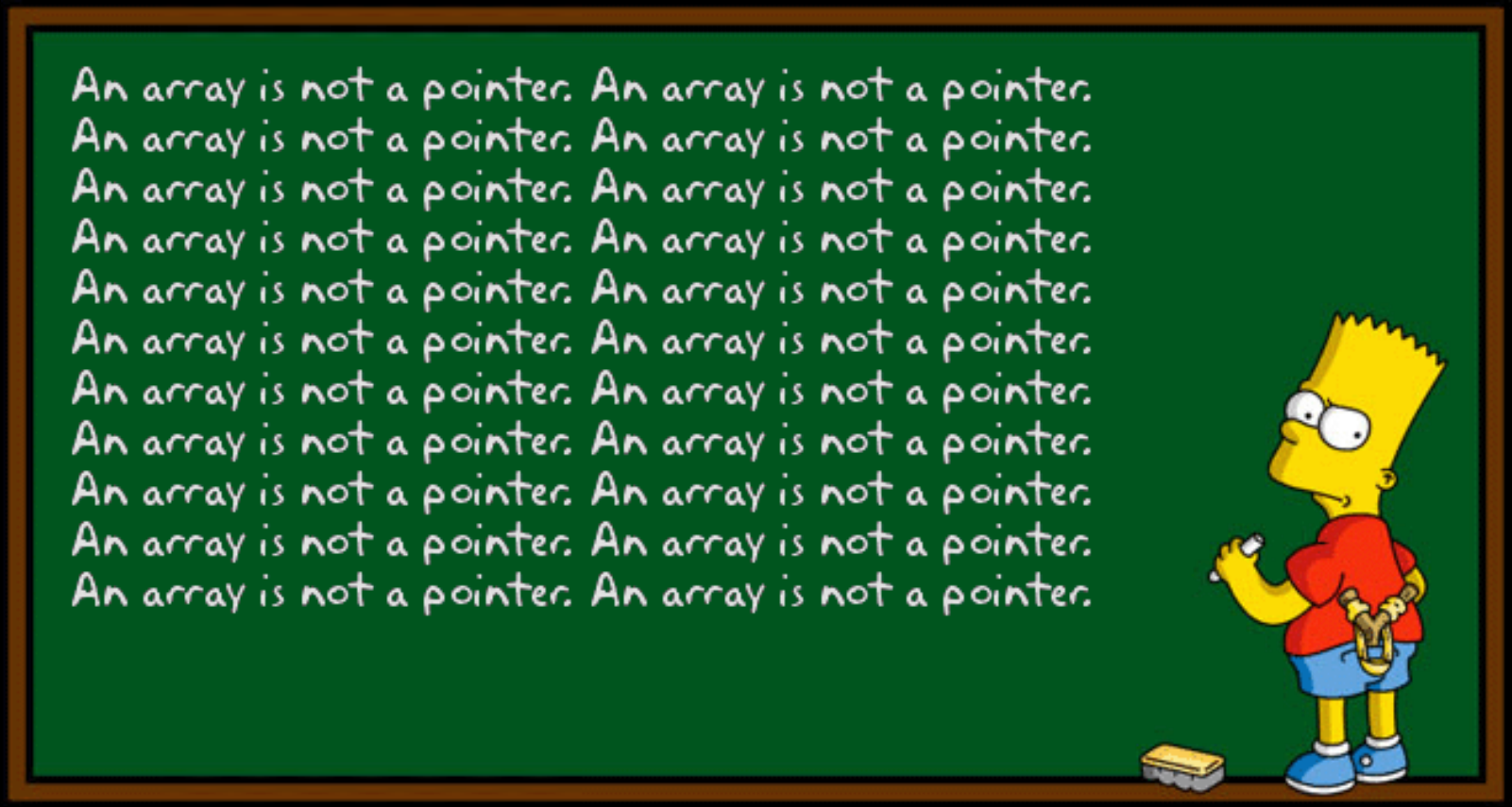
```
int (*fptr)(int int);
```

CSE102
Computer Programming
(Next Topic)



Credits: www.programiz.com

CSE102
Computer Programming
(Next Topic)



Credits: i.imgur.com