

# Pointer

# Pointers I

# Recap

**Memory** - In computing, memory refers to the computer hardware devices used to store information for immediate use in a computer; [https://en.wikipedia.org/wiki/Computer\\_memory](https://en.wikipedia.org/wiki/Computer_memory)

**Variable** - a symbolic name associated with a value and whose associated value may be changed.

# Variable

```
int x;
```

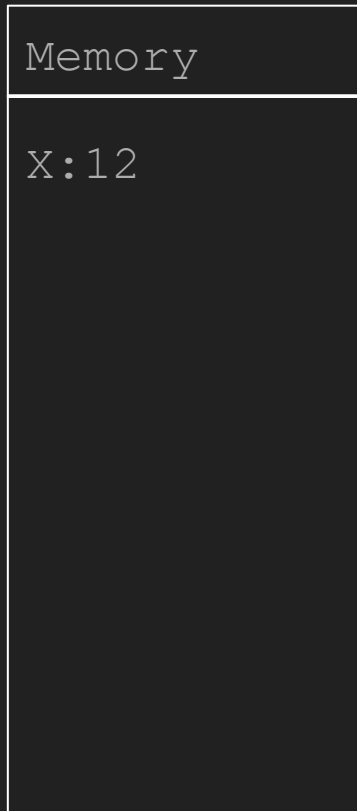
```
x = 12;
```

# Variable

```
int x;
```

```
x = 12;
```

Direct!



# Real life

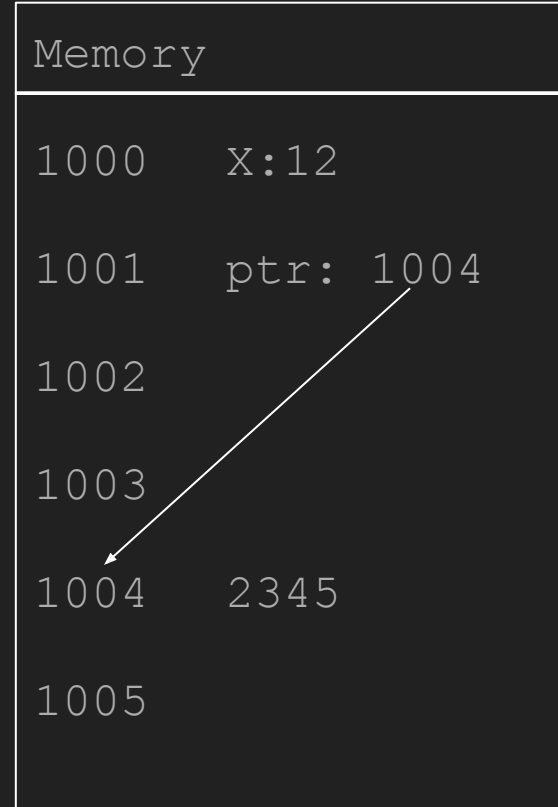
Go to **Henrik**

Indirect!!

Memory
Henrik: Patricia, 4, 472

# Variable

```
int *ptr;
```



# Code

```
int x = 100;
```

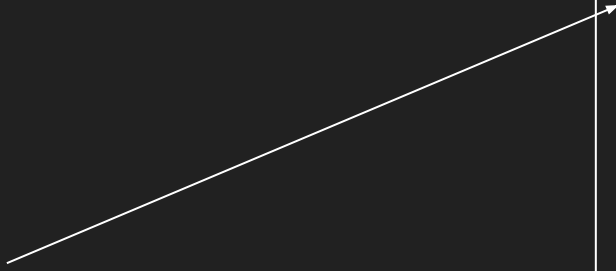
Memory	
1000	x:100
1001	
1002	
1003	
1004	
1005	



# Code

```
int x = 100;  
printf ("%p\n", &x);
```

Memory	
1000	X:100
1001	
1002	
1003	
1004	
1005	



# Code \*

```
int x = 100;
```

```
int *xp = 1000;
```

Memory	
1000	x:100
1001	
1002	xp: 1000
1003	
1004	
1005	

\*

```
int *iptr;
```

iptr is a pointer that points to int

```
double *dptr;
```

dptr is a pointer that points to double

```
media *mp;
```

mp is a pointer that points to a media

# Code &

```
int x = 100;
```

```
int *xp = &x;
```

Memory	
1000	x:100
1001	
1002	xp: 1000
1003	
1004	
1005	

# & - address of operator

```
int c=12;
```

Declare an int var, called c. Assign it 12

```
int *cp = &c;
```

Declare a pointer to int variable, called cp.

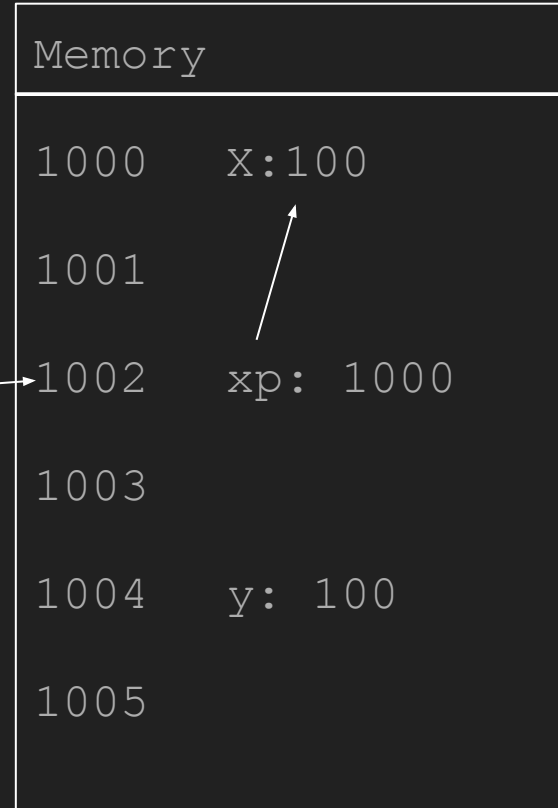
Assign it the address of c

\* again

```
int x = 100;
```

```
int *xp = &x;
```

```
int y = *xp;
```



# Pointers II - pointers and functions

# Pointers II - pointers and functions

```
int add(int x, int y) {  
    return x+y;  
}
```

Memory:

x: 2

y: 3

```
void main() {  
    int res = add(2,3);  
    ...  
}
```



# Pointers II - pointers and functions

```
int add(int x, int y) {  
  
    return x+y;  
  
}
```

Memory:

```
void main() {  
  
    int res = add(2,3);  
  
    ...  
  
}
```

# Pointers II - pointers and functions

```
int add(int *x, int *y) {
```

```
    return *x + *y;
```

```
}
```

```
void main() {
```

```
    int girls=12;
```

```
    int boys=13;
```

```
    int res = add(girls, boys);
```

```
    ...
```

```
}
```

Memory

address	name	value
---------	------	-------

1000	girls	12
------	-------	----

1001	boys	13
------	------	----

1002	res	
------	-----	--

# Pointers II - pointers and functions

```
int add(int *x, int *y) {
```

```
    return *x + *y;
```

```
}
```

```
void main() {
```

```
    int girls=12;
```

```
    int boys=13;
```

```
    int res = add(&girls, &boys);
```

```
    ...
```

```
}
```

Memory

address	name	value
---------	------	-------

1000	girls	12
------	-------	----

1001	boys	13
------	------	----

1002	res	
------	-----	--

1002	x	1000
------	---	------

1003	y	1001
------	---	------

# Pointers II - pointers and functions

```
int add(int *x, int *y) {
```

```
    return *x + *y;
```

```
}
```

```
void main() {
```

```
    int girls=12;
```

```
    int boys=13;
```

```
    int res = add(&girls, &boys);
```

```
    ...
```

```
}
```

Memory

address	name	value
---------	------	-------

1000	girls	12
------	-------	----

1001	boys	13
------	------	----

1002	res	25
------	-----	----

# Pointers III - pointers sizes

Live coding instead

# Pointers IV - pointers and arrays

**Pointers and arrays are NOT the same!!!**

# Array

```
char a[6];
```





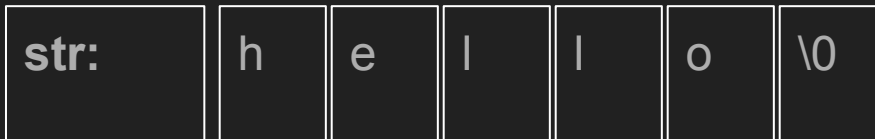
# Array

```
char str[] = "hello";
```



# Array

```
char str[] = "hello";
```

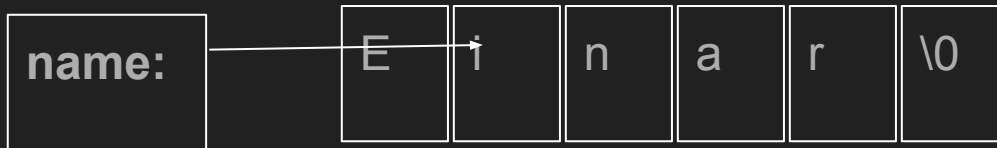


```
char *name = "Einar";
```



# Array

```
char *name = "Einar";
```

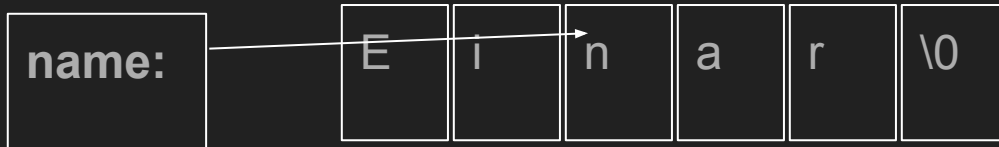


```
name++ ;
```

1 byte per ++

# Array

```
char *name = "Einar";
```



```
name++ ;
```

```
name++ ;
```

1 byte per ++

# Pointers V - user arguments

Supplying arguments to a function

```
add(12,23);
```

How to supply arguments to main?

# Pointers V - user arguments

You already have done that

```
gcc main.c awesomemath.c unawesome.c
```

How does gcc use that? ...

# Pointers V - user arguments

```
int main(int argc) {  
    printf ("Nr of arguments: %d\n");  
}
```

# Pointers V - user arguments

```
int main(int argc, char **argv) {  
    printf ("Nr of arguments: %d\n");  
  
}
```



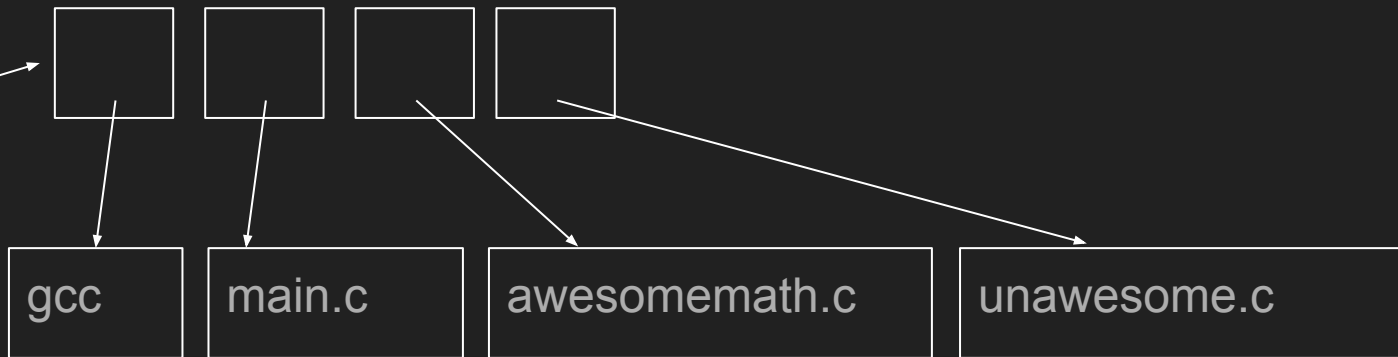
# Pointers V - user arguments

gcc main.c awesomemath.c unawesome.c

## Memory

argc: 4

argv



# Pointers V - structs

Recap

```
int x = 12;
```

```
int *xp = &x;
```

```
*xp = 13;
```

# Pointers V - structs

Live coding...