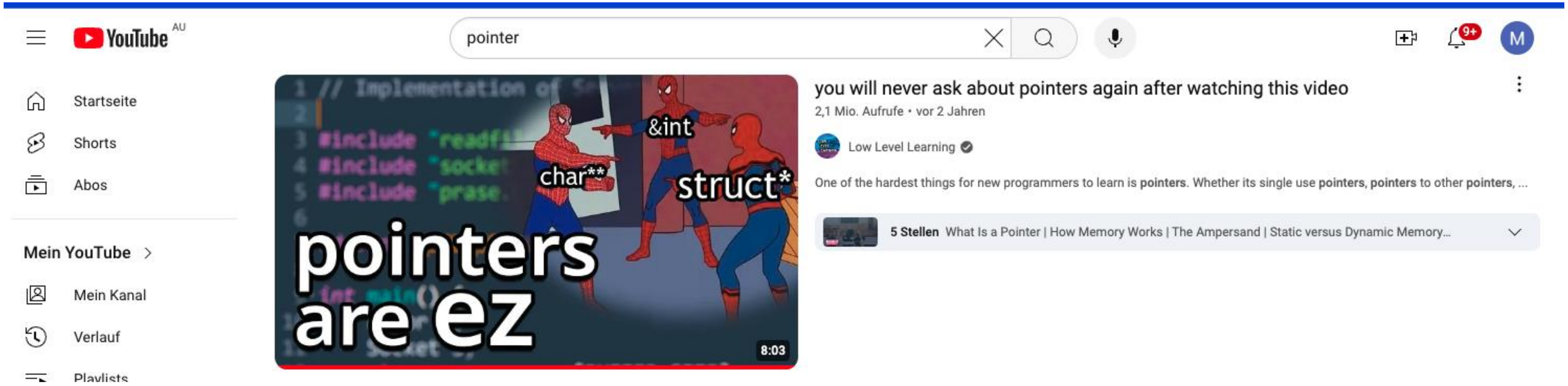
Hello Class!

If you aren’t familiar with **pointers** or would like a quick refresher, please watch the following video:

https://www.youtube.com/watch?v=2ybLD6_2gKM



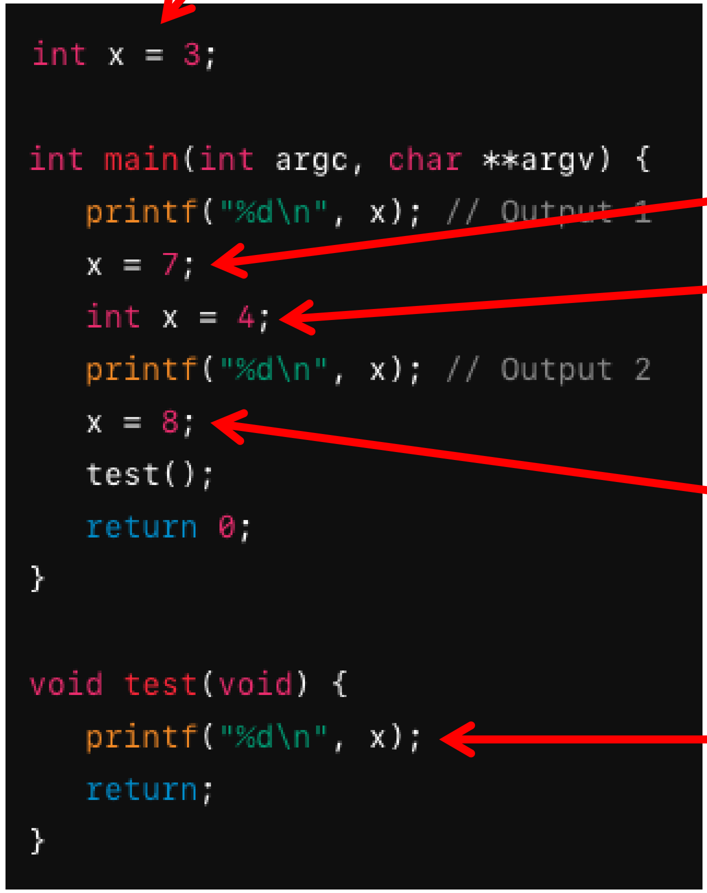
you will never ask about pointers again after watching this video

2,1 Mio. Aufrufe • vor 2 Jahren

Low Level Learning

One of the hardest things for new programmers to learn is **pointers**. Whether its single use **pointers**, **pointers** to other **pointers**, ...

5 Stellen What Is a Pointer | How Memory Works | The Ampersand | Static versus Dynamic Memory...



```c
int x = 3;

int main(int argc, char **argv) {
   printf("%d\n", x); // Output 1
   x = 7;
   int x = 4;
   printf("%d\n", x); // Output 2
   x = 8;
   test();
   return 0;
}

void test(void) {
   printf("%d\n", x);
   return;
}
```

global variable called *x*

update global variable *x*

new local variable called *x*

! this variable now **shadows** the global *x* !

update local variable *x*

access global *x*

--VARIABLES—

Identifier	Type	Point #1	Point #2	Point #3
argc	int			
argv	char*[]			
trev	int			
beth	double			
pete	int			
bill	int			
jack	int			
jane	int			
mary	int			
zack	double			
dick	double			
fred	int			
dave	double			
trev	double			

--FUNCTIONS--

Identifier	Type	Point #1	Point #2	Point #3
main	int main(int, char*[])			
bill	int bill(int, int)			
jane	double jane(double, int, double)			

```c
int bill(int jack, int jane);
double jane(double dick, int fred, double dave);

int trev;

int
main(int argc, char *argv[]) {
   double beth;
   int pete, bill;
   /* point #1 */
   return 0;
}

int
bill(int jack, int jane) {
   int mary;
   double zack;
   /* point #2 */
   return 0;
}

double
jane(double dick, int fred, double dave) {
   double trev;
   /* point #3 */
   return 0.0;
}

```

Pointers

In python, we can easily return multiple values:

python

```python
def swap(a, b):
    return b, a

x, y = swap(1, 2)
print(x, y)  # 2 1
```

Pointers

In python, we can easily return multiple values:

python

```python
def swap(a, b):
    return b, a

x, y = swap(1, 2)
print(x, y)  # 2 1
```

In C, a function can formally only return one value. Therefore, we have to use pointers:

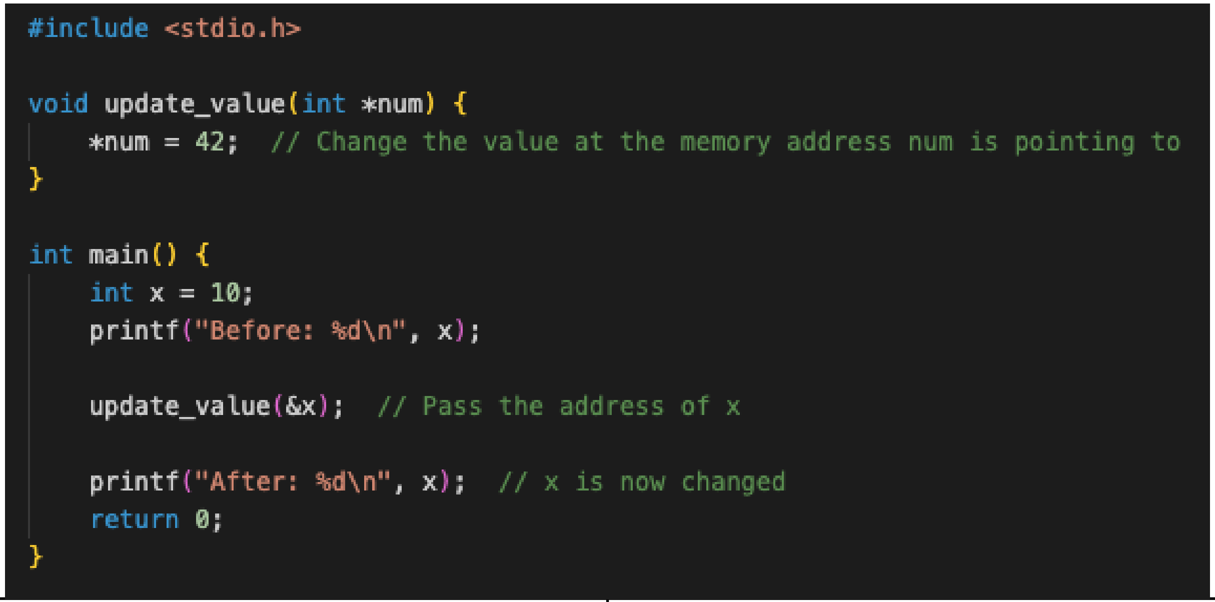
c

```c
#include <stdio.h>

void swap(int *a, int *b) {
    int tmp = *a;
    *a = *b;
    *b = tmp;
}

int main() {
    int x = 1, y = 2;
    swap(&x, &y);
    printf("%d %d\n", x, y);  // 2 1
}
```

Pointers

int cents	define a variable called *cents*
scanf(“%d”, &cents) $\hookrightarrow$ the address of the variable *cents*	store the input at the address of *cents*
int *cents	<u>define a pointer</u> called *cents* (not initialised)
int $\underbrace{\text{try\_one\_coin}}_{\text{function}}$(int *cents)	<u>defining a function</u> called *try_one_coin*: create a pointer called cents it will point at whatever is passed to the function as input
*cents	<u>using inside a function</u>: access the value stored at the address cents is pointing to

Pointers



```c
#include <stdio.h>

void update_value(int *num) {
    *num = 42;  // Change the value at the memory address num is pointing to
}

int main() {
    int x = 10;
    printf("Before: %d\n", x);

    update_value(&x);  // Pass the address of x

    printf("After: %d\n", x);  // x is now changed
    return 0;
}
```

int cents	define a variable called *cents*
scanf(“%d”, &cents) $\rightarrow$ the address of the variable *cents*	store the input at the address of *cents*
int *cents	<u>define a pointer</u> called *cents* (not initialised)
int $\underbrace{\text{try\_one\_coin}}_{\text{function}}$(int *cents)	<u>defining a function</u> called *try_one_coin*: create a pointer called cents it will point at whatever is passed to the function as input
*cents	<u>using inside a function</u>: access the value stored at the address cents is pointing to