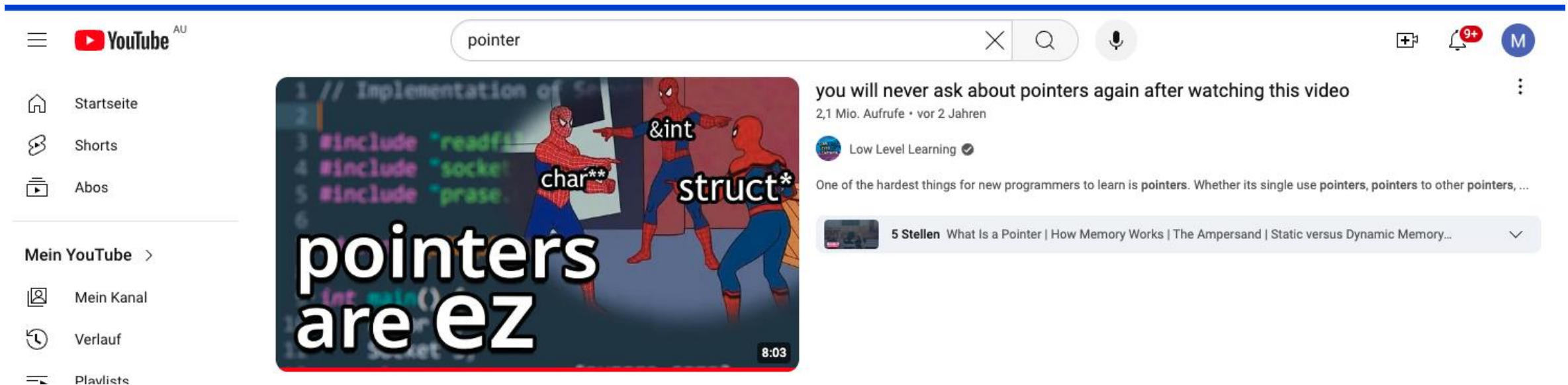


Hello Class!

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



The image shows a YouTube video player interface. The video title is "you will never ask about pointers again after watching this video". The video is by "Low Level Learning" and has 2.1 million views. The video description says "One of the hardest things for new programmers to learn is pointers. Whether its single use pointers, pointers to other pointers, ...". The video thumbnail shows three Spider-Man characters pointing to code snippets: "char**", "&int", and "struct*". The text "pointers are ez" is overlaid on the thumbnail. The video duration is 8:03. The video player interface includes a search bar with the word "pointer", a sidebar with navigation links, and a bottom bar with video details.

YouTube AU

pointer

Startseite

Shorts

Abos

Mein YouTube >

Mein Kanal

Verlauf

Playlists

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...

pointers are ez

8:03

global variable called x

```
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;
```

```
}
```

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) | | | |

```
1 int bill(int jack, int jane);
2 double jane(double dick, int fred, double dave);
3
4 int trev;
5
6 int
7 main(int argc, char *argv[]) {
8     double beth;
9     int pete, bill;
10    /* point #1 */
11    return 0;
12 }
13
14 int
15 bill(int jack, int jane) {
16     int mary;
17     double zack;
18     /* point #2 */
19     return 0;
20 }
21
22 double
23 jane(double dick, int fred, double dave) {
24     double trev;
25     /* point #3 */
26     return 0.0;
27 }
28
```

Pointers

In python, we can easily return multiple values:

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

```
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:

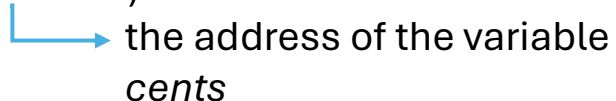
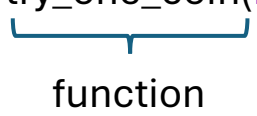
c

```
#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

| | |
|--|---|
| <code>int cents</code> | define a variable called <i>cents</i> |
| <code>scanf("%d", &cents)</code>  the address of the variable <i>cents</i> | store the input at the address of <i>cents</i> |
| <code>int *cents</code> | <u>define a pointer</u> called <i>cents</i> (not initialised) |
| <code>int try_one_coin(int *cents)</code>  function | <u>defining a function</u> called <i>try_one_coin</i> : create a pointer called <i>cents</i> it will point at whatever is passed to the function as input |
| <code>*cents</code> | <u>using inside a function</u> : access the value stored at the address <i>cents</i> is pointing to |

Pointers



```
#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;
}
```

| | |
|--|---|
| <code>int cents</code> | define a variable called <i>cents</i> |
| <code>scanf("%d", &cents)</code>  the address of the variable <i>cents</i> | store the input at the address of <i>cents</i> |
| <code>int *cents</code> | <u>define a pointer</u> called <i>cents</i> (not initialised) |
| <code>int try_one_coin(int *cents)</code>  function | <u>defining a function</u> called <i>try_one_coin</i> : create a pointer called <i>cents</i> it will point at whatever is passed to the function as input |
| <code>*cents</code> | <u>using inside a function</u> : access the value stored at the address <i>cents</i> is pointing to |