



Pointer und Pointer-Magie



(incl. völlig verwirrender Pfeilchen...)

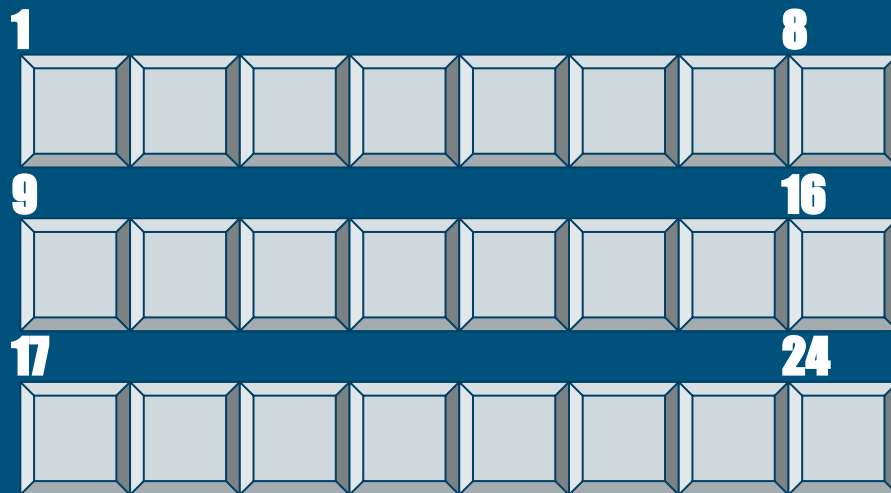


Variablen im Speicher

```
#include <iostream>

int main() {
    int a;
    a = 7;
    char c = 12;
    std::cout << c <<
std::endl;
}
```

Name	Type	Address

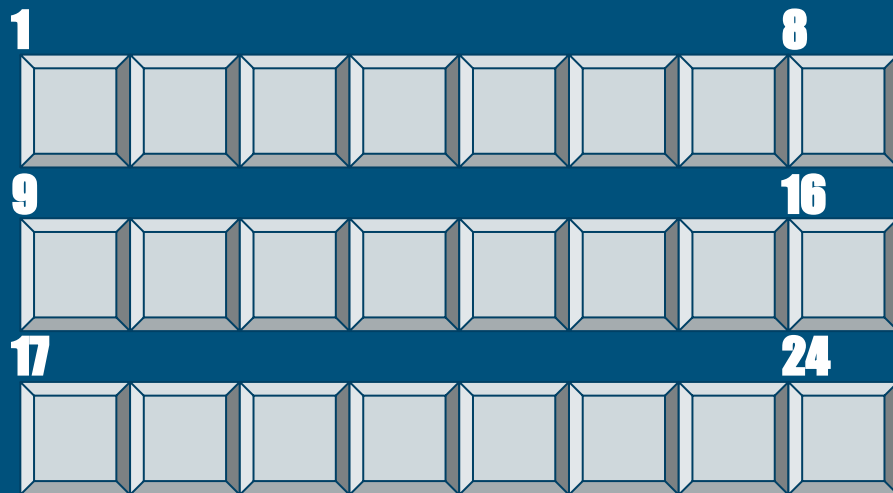


Variablen im Speicher

```
#include <iostream>

int main() {
    ➡ int a;
      a = 7;
      char c = 12;
      std::cout << c <<
std::endl;
}
```

Name	Type	Address

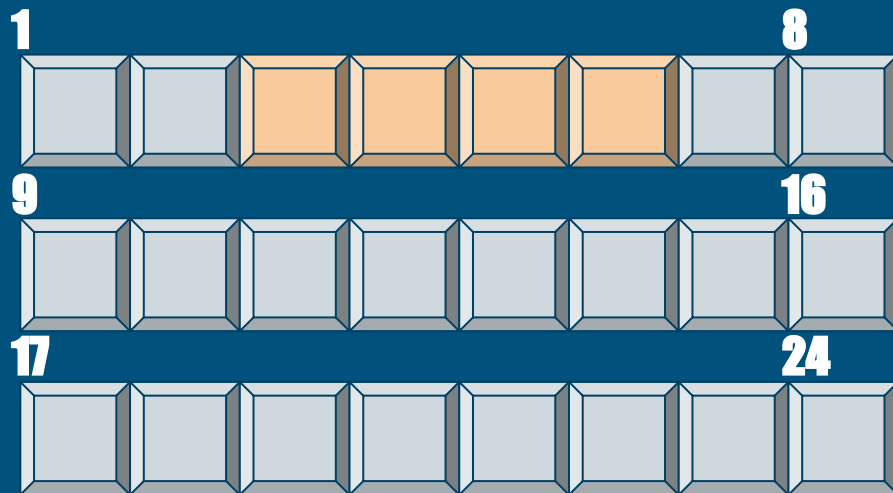


Variablen im Speicher

```
#include <iostream>

int main() {
    ➡ int a;
    a = 7;
    char c = 12;
    std::cout << c <<
    std::endl;
}
```

Name	Type	Address
a	int	3

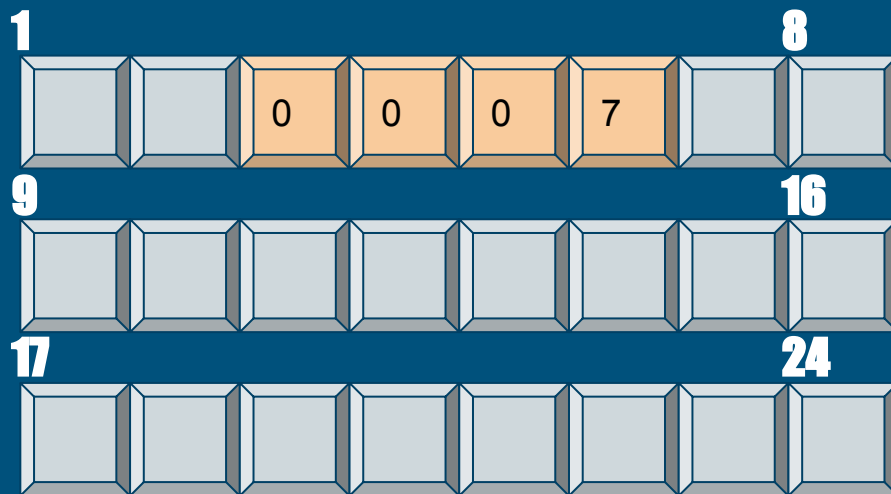


Variablen im Speicher

```
#include <iostream>

int main() {
    int a;
    ➡ a = 7;
    char c = 12;
    std::cout << c <<
    std::endl;
}
```

Name	Type	Address
a	int	3

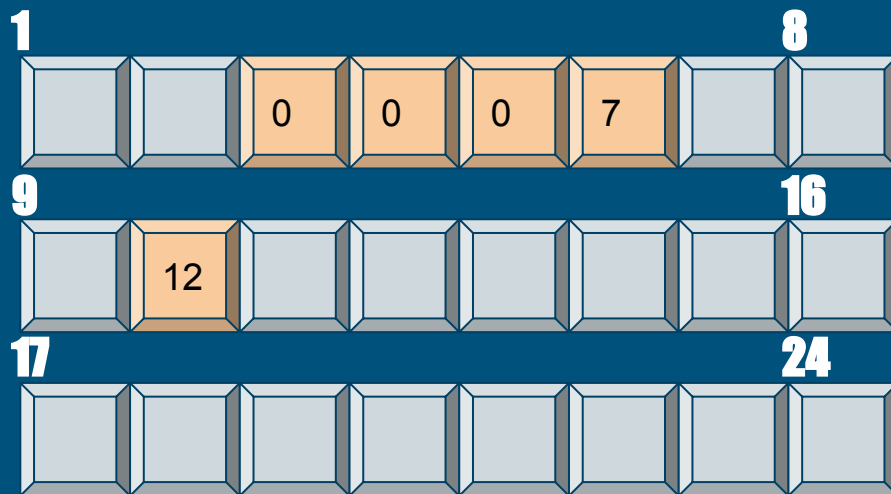


Variablen im Speicher

```
#include <iostream>

int main() {
    int a;
    a = 7;
    ➡ char c = 12;
    std::cout << c <<
    std::endl;
}
```

Name	Type	Address
a	int	3
c	char	12



Variablen im Speicher

```
#include <iostream>
```

```
int main() {
```

```
    int a;
```

```
    a = 7;
```

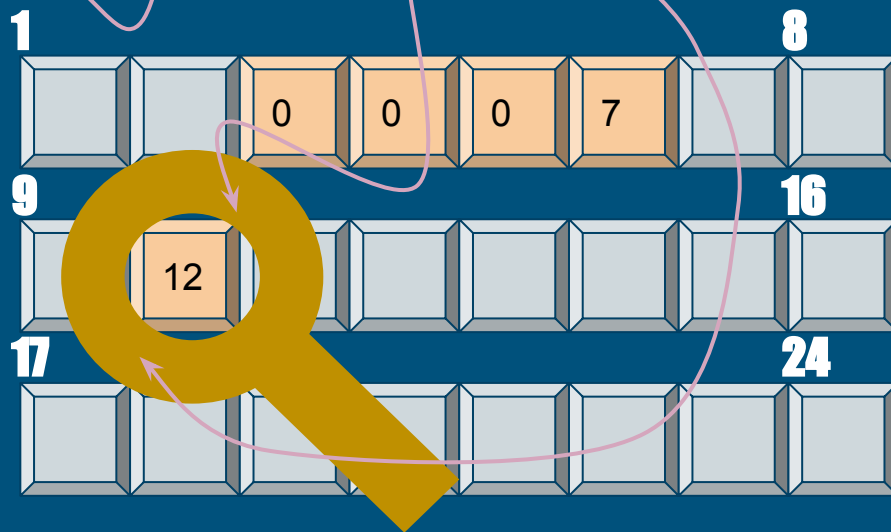
```
    char c = 12;
```

```
    std::cout << c <<
```

```
    std::endl;
```

```
}
```

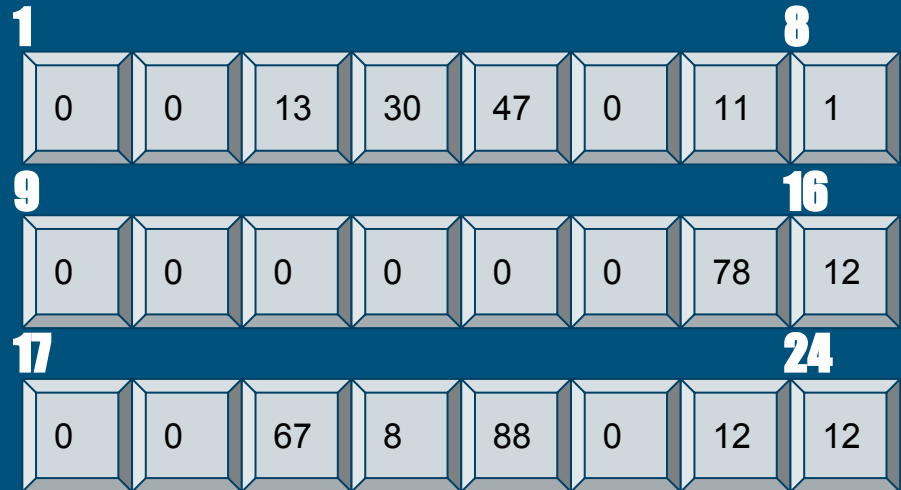
Name	Type	Address
a	int	3
c	char	10



Array's und RAM-Müll

```
int main() {  
    int arr[4];  
    a[0] = 14;  
    a[2] = 1;  
    print(arr[1]);  
}
```

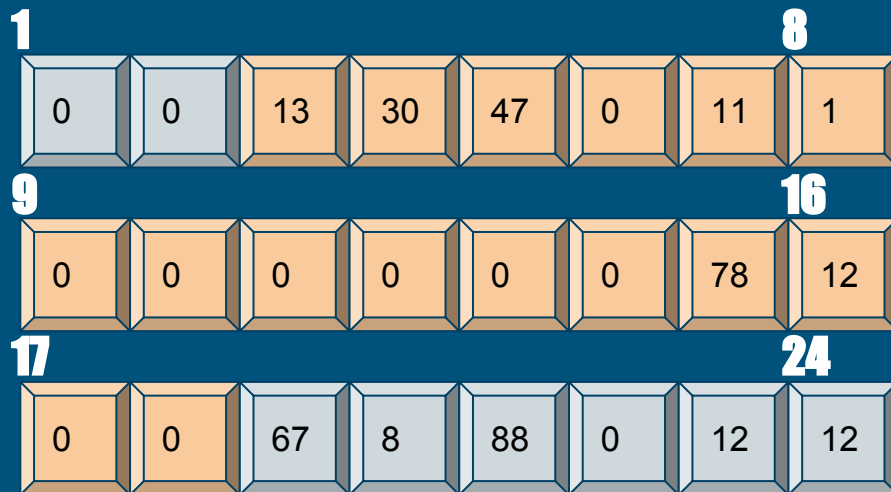
Name	Type	Address



Array's und RAM-Müll

```
int main() {  
    → int arr[4];  
    a[0] = 14;  
    a[2] = 1;  
    print(arr[1]);  
}
```

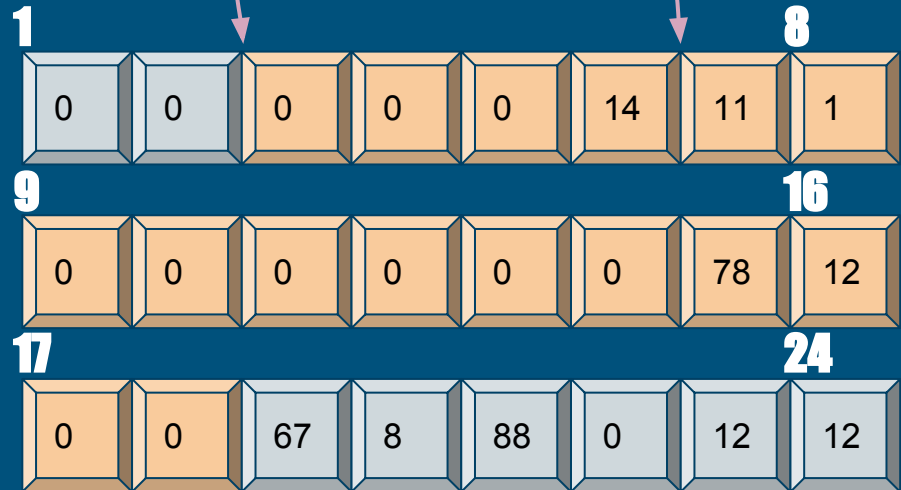
Name	Type	Address
arr	int	3



Array's und RAM-Müll

```
int main() {  
    int arr[4];  
    a[0] = 14;  
    a[2] = 1;  
    print(arr[1]);  
}
```

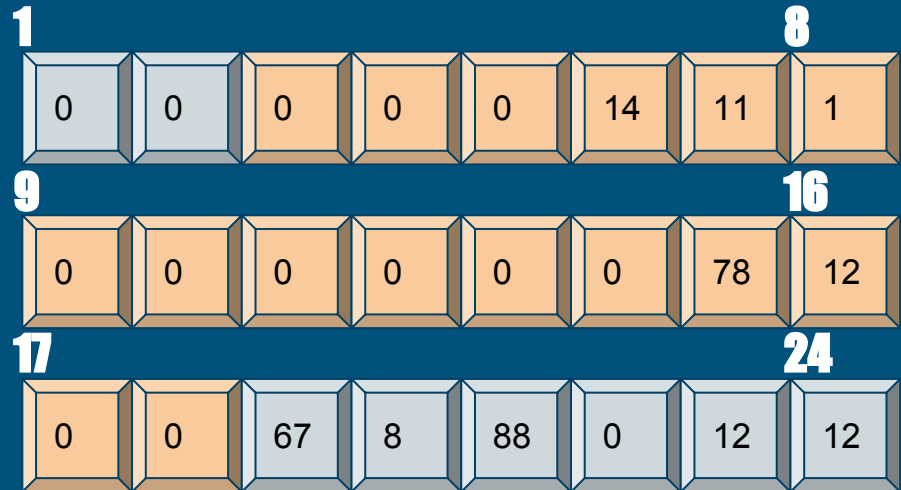
Name	Type	Address
arr	int	3



Array's und RAM-Müll

```
int main() {  
    int arr[4];  
    a[0] = 14;  
    a[2] = 1;  
    print(arr[1]);  
}
```

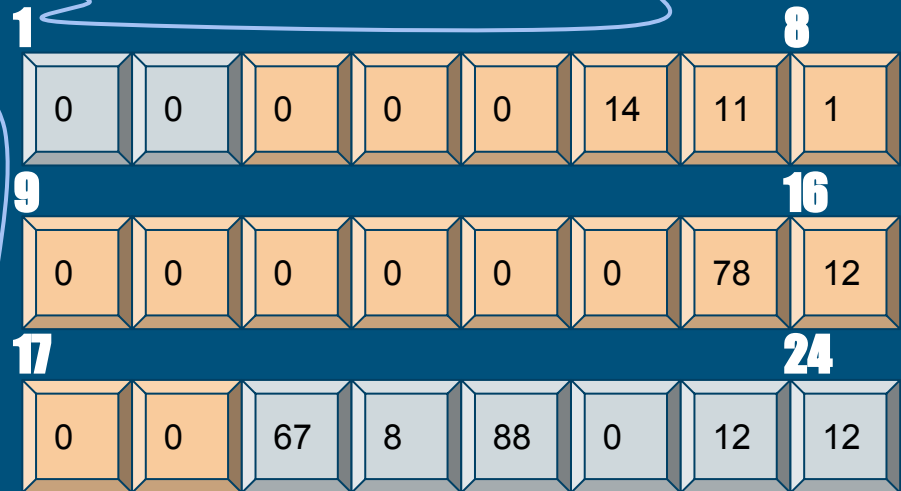
Name	Type	Address
arr	int	3



Array's und RAM-Müll

```
int main() {  
    int arr[4];  
    a[0] = 14;  
    a[2] = 1;  
    print(arr[1]);  
}
```

Name	Type	Address
arr	int	3



Array's und RAM-Müll

```
int main() {  
    int arr[4];  
    a[0] = 14;  
    a[2] = 1;  
    print(arr[1]);  
}
```

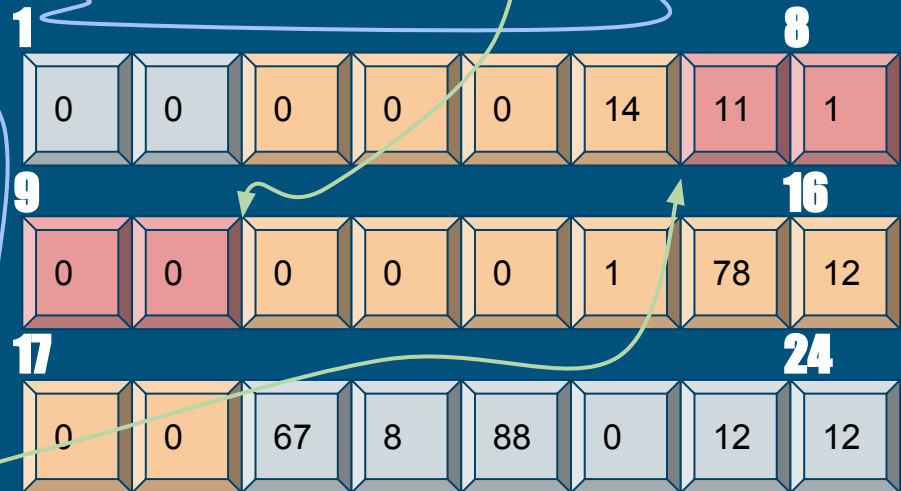
Name	Type	Address
arr	int	3



Array's und RAM-Müll

```
int main() {  
    int arr[4];  
    a[0] = 14;  
    a[2] = 1;  
    print(arr[1]);  
}
```

Name	Type	Address
arr	int	3

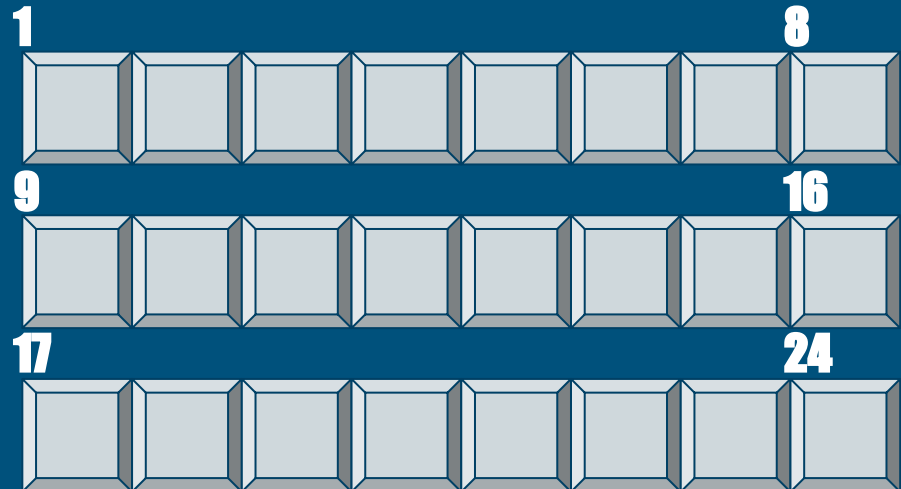


184614912

Pointer und Dereferenzierung

```
int main() {  
    int b = 25;  
    int* p = &b;  
    print( b);  
    print( p);  
    print(*p);  
}
```

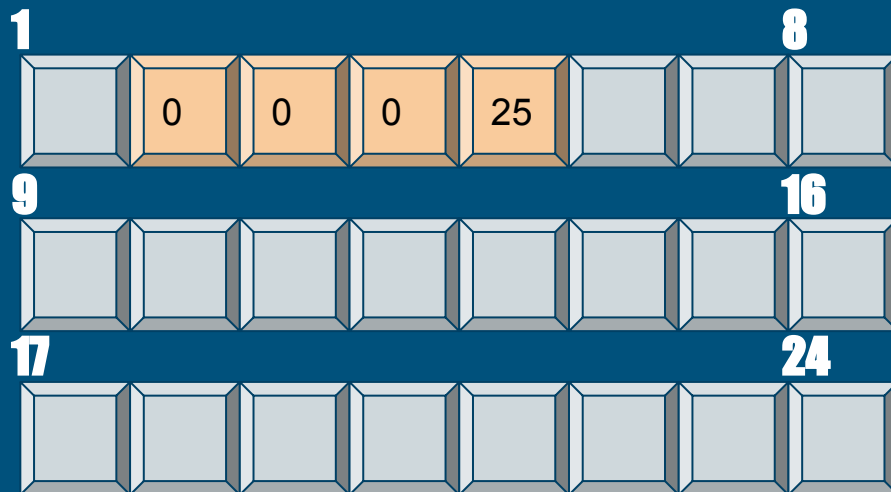
Name	Type	Address



Pointer und Dereferenzierung

```
int main() {  
    ➡ int b = 25;  
    int* p = &b;  
    print( b);  
    print( p);  
    print(*p);  
}
```

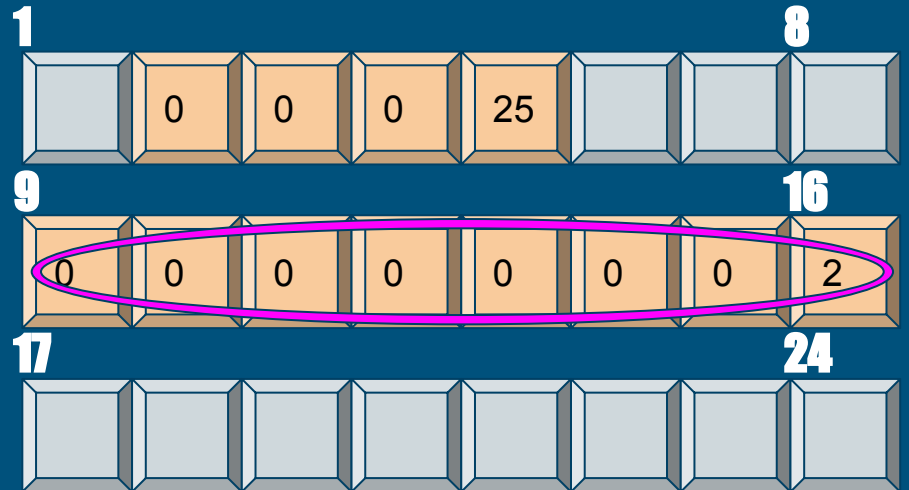
Name	Type	Address
b	int	2



Pointer und Dereferenzierung

```
int main() {  
    int b = 25;  
    int* p = &b;  
    print( b);  
    print( p);  
    print(*p);  
}
```

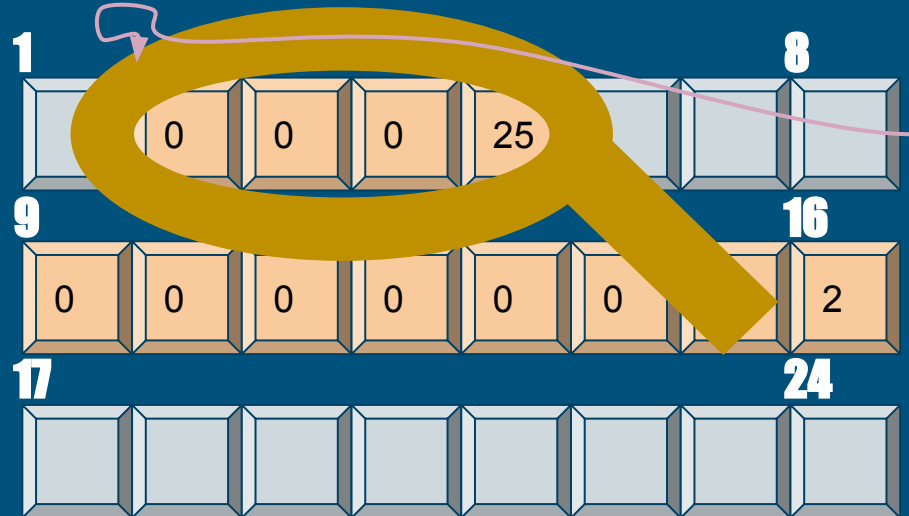
Name	Type	Address
b	int	2
p	ptr (int)	9



Pointer und Dereferenzierung

```
int main() {  
    int b = 25;  
    int* p = &b;  
    print( b);  
    print( p);  
    print(*p);  
}
```

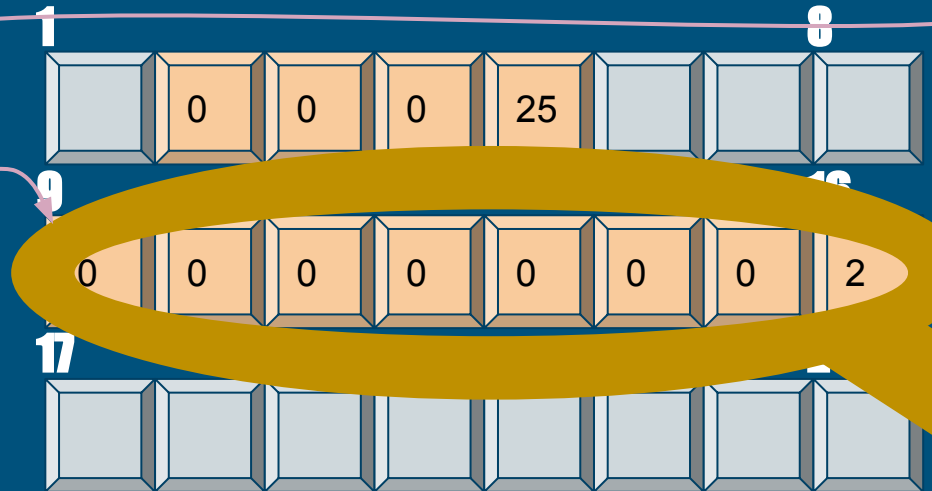
Name	Type	Address
b	int	2
p	ptr (int)	9



Pointer und Dereferenzierung

```
int main() {  
    int b = 25;  
    int* p = &b;  
    print( b);  
    print( p);  
    print(*p);  
}
```

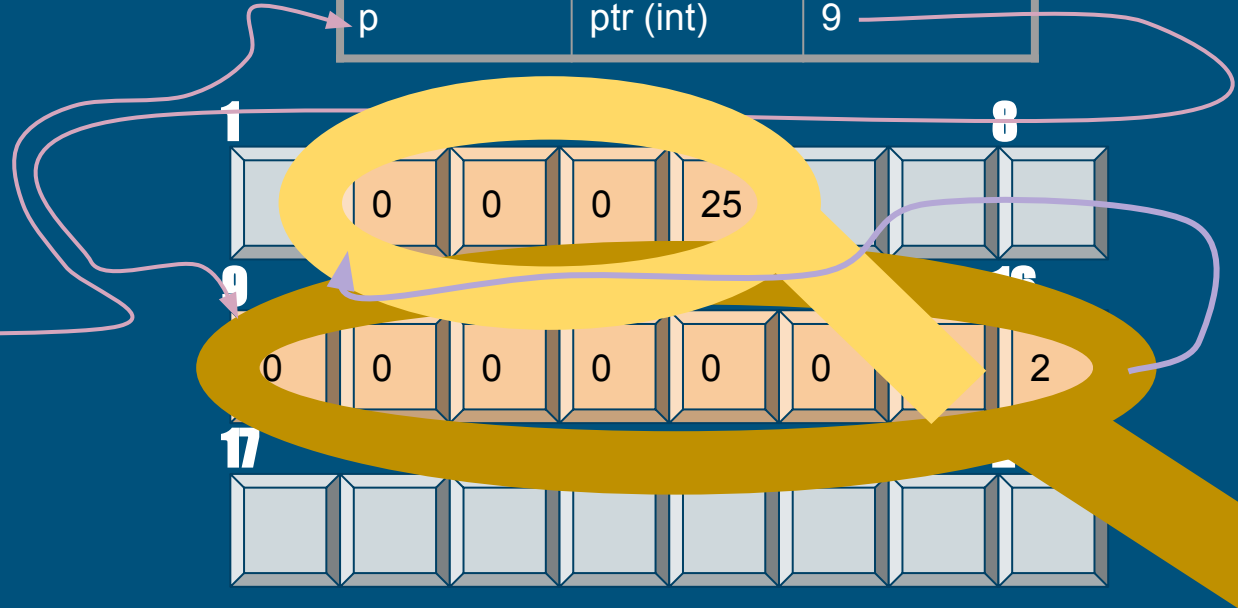
Name	Type	Address
b	int	2
p	ptr (int)	9



Pointer und Dereferenzierung

```
int main() {  
    int b = 25;  
    int* p = &b;  
    print( b);  
    print( p);  
    print(*p);  
}
```

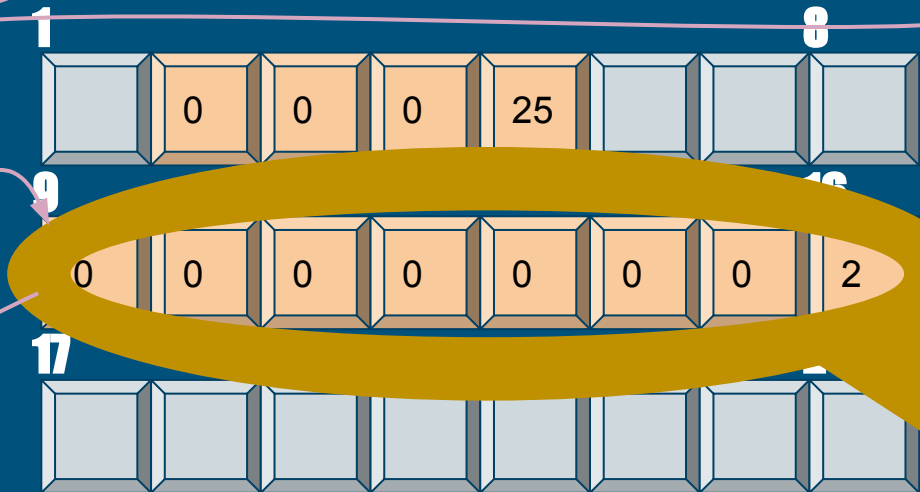
Name	Type	Address
b	int	2
p	ptr (int)	9



Pointer und Dereferenzierung (schrittweise)

```
int main() {  
    int b = 25;  
    int* p = &b;  
    print( b);  
    print( p);  
    print(*p);  
    print(* (2));  
}
```

Name	Type	Address
b	int	2
p	ptr (int)	9



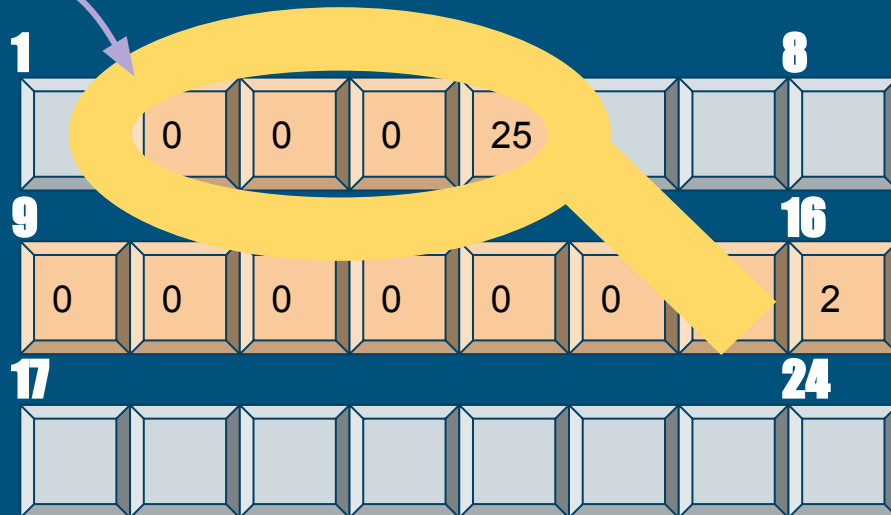
Denk-Hilfe-Stellung!
Kein valider Ausdruck!

Pointer und Dereferenzierung (schrittweise)

```
int main() {  
    int b = 25;  
    int* p = &b;  
    print( b);  
    print( p);  
    print(*p);  
    print(* (2));  
}
```

ohne Tabelle!

Name	Type	Address
b	int	2
p	ptr (int)	9

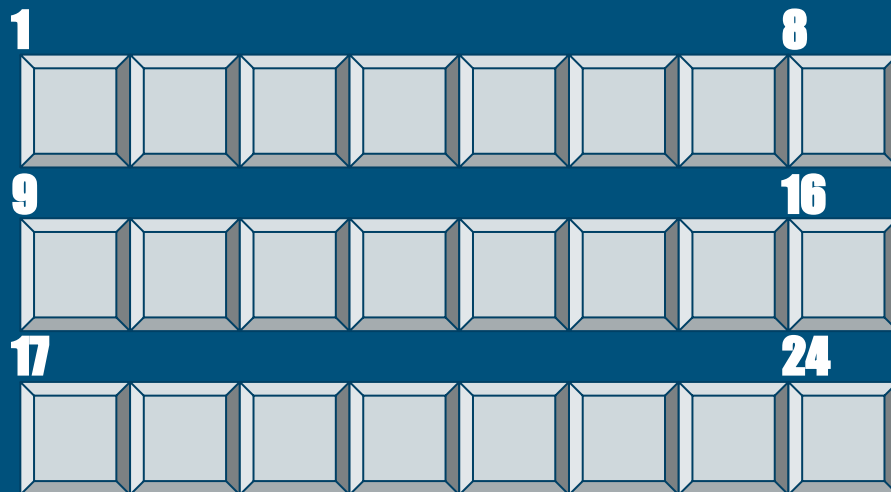


Denk-Hilfe-Stellung!
Kein valider Ausdruck!

Pointerarithmetik

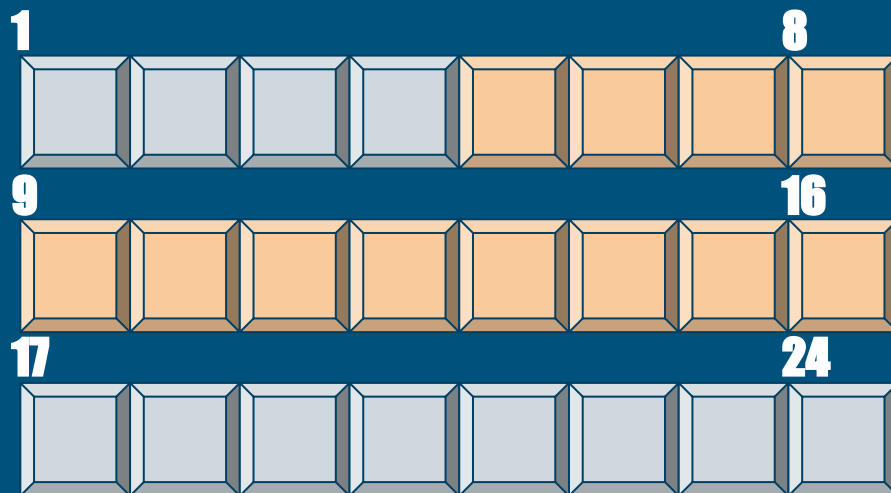
```
int main() {  
    int a[3];  
    int* p = &a[0];  
    print(*p);  
    p++;  
    print(*p);  
    print(*(p+1));  
    print(a[2]);  
    print(p[1]);  
}
```

Name	Type	Address



➡ `int a[3];`

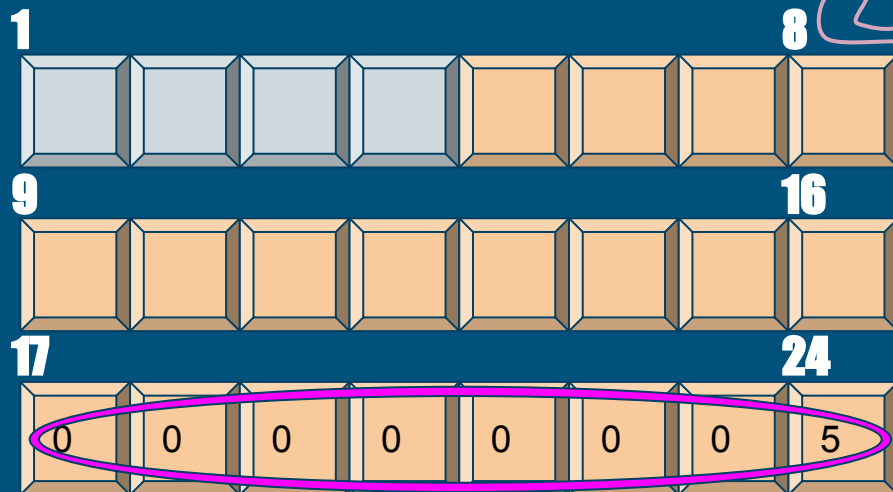
Name	Type	Address
a	int	5



Pointerarithmetik

```
int main() {  
    int a[3];  
    ➔ int* p = &a[0];  
    print(*p);  
    p++;  
    print(*p);  
    print(*(p+1));  
    print(a[2]);  
    print(p[1]);  
}
```

Name	Type	Address
a	int	5
p	ptr (int)	17

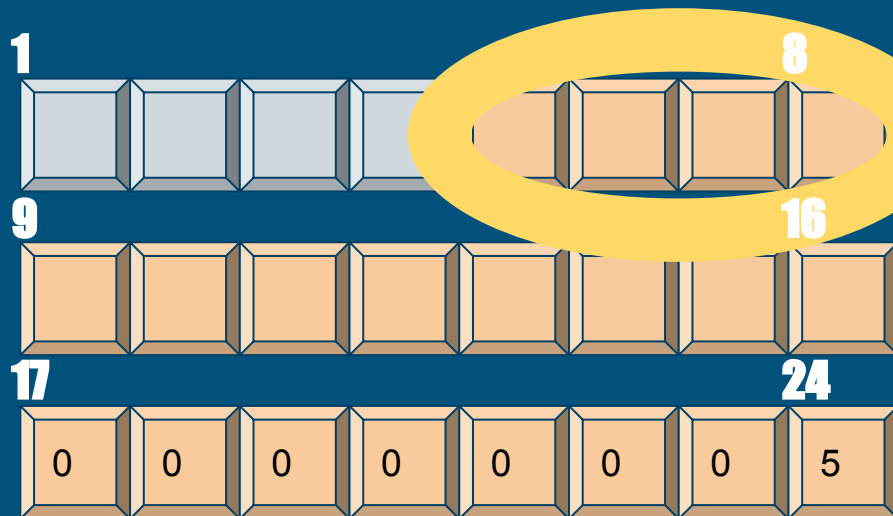


Pointerarithmetik

```
int main() {  
    int a[3];  
    int* p = &a[0];  
    ➔ print(*p);  
    p++;  
    print(*p);  
    print(*(p+1));  
    print(a[2]);  
    print(p[1]);  
}
```

Warum?

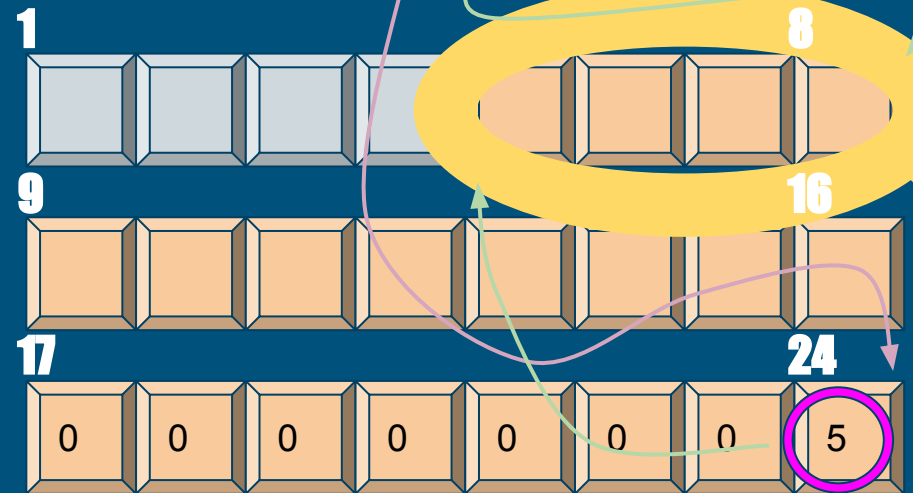
Name	Type	Address
a	int	5
p	ptr (int)	17



Pointerarithmetik

```
int main() {  
    int a[3];  
    int* p = &a[0];  
    print(*p);  
    p++;  
    print(*p);  
    print(*(p+1));  
    print(a[2]);  
    print(p[1]);  
}
```

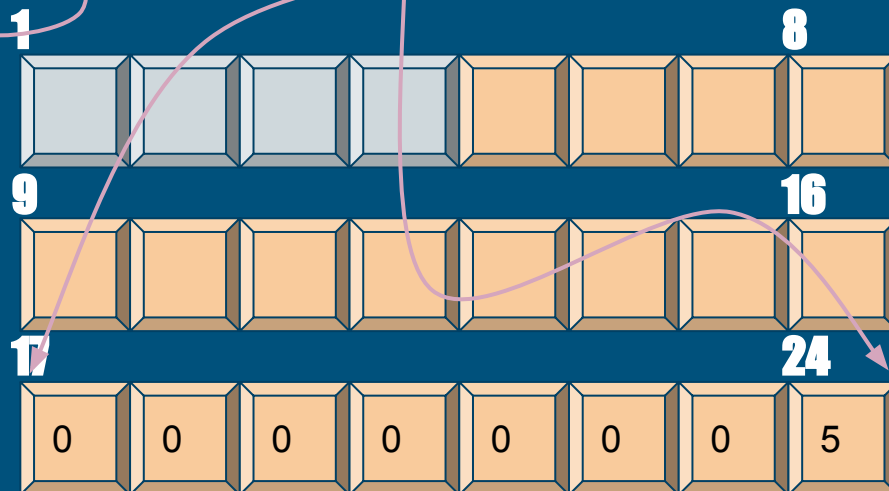
Name	Type	Address
a	int	5
p	ptr (int)	17



Pointerarithmetik

```
int main() {  
    int a[3];  
    int* p = &a[0];  
    print(*p);  
    p++;  
    print(*p);  
    print(*(p+1));  
    print(a[2]);  
    print(p[1]);  
}
```

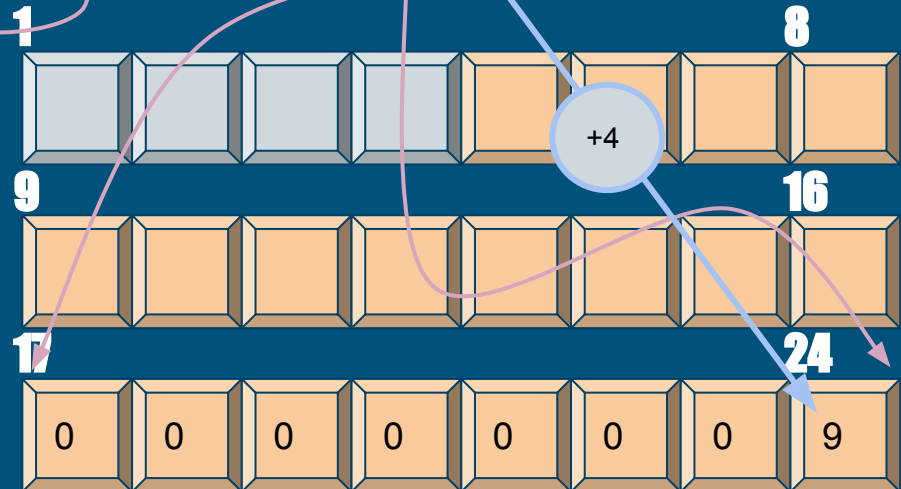
Name	Type	Address
a	int	5
p	ptr (int)	17



Pointerarithmetik

```
int main() {  
    int a[3];  
    int* p = &a[0];  
    print(*p);  
    p++;  
    print(*p);  
    print(*(p+1));  
    print(a[2]);  
    print(p[1]);  
}
```

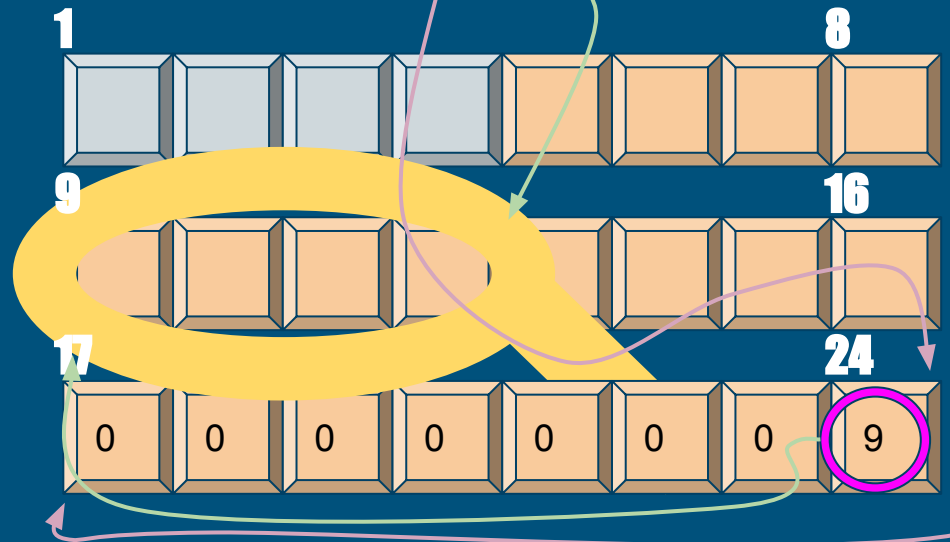
Name	Type	Address
a	int	5
p	ptr (int)	17



Pointerarithmetik

```
int main() {  
    int a[3];  
    int* p = &a[0];  
    print(*p);  
    p++;  
    print(*p);  
    print(a[2]);  
    print(p[1]);  
}
```

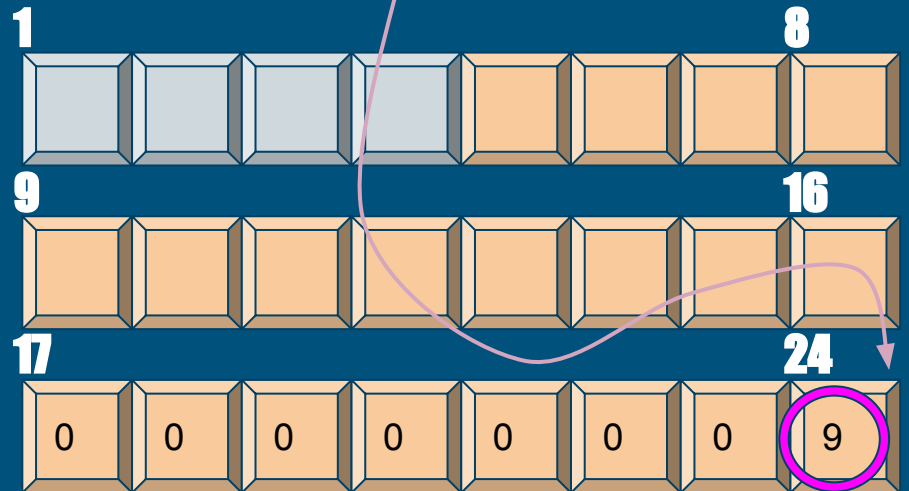
Name	Type	Address
a	int	5
p	ptr (int)	17



Pointerarithmetik

```
int main() {  
    int a[3];  
    int* p = &a[0];  
    print(*p);  
    p++;  
    print(*p);  
    print(* (p+1));  
    print(a[2]);  
    print(p[1]);  
}
```

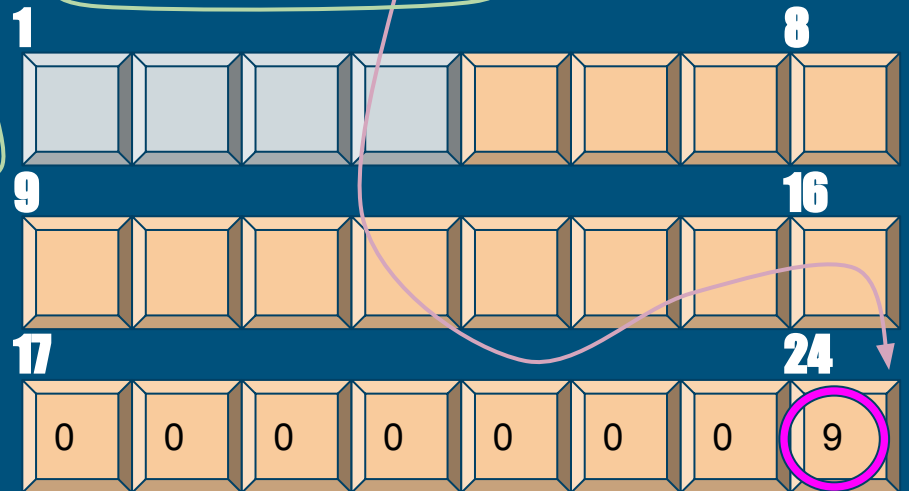
Name	Type	Address
a	int	5
p	ptr (int)	17



Pointerarithmetik

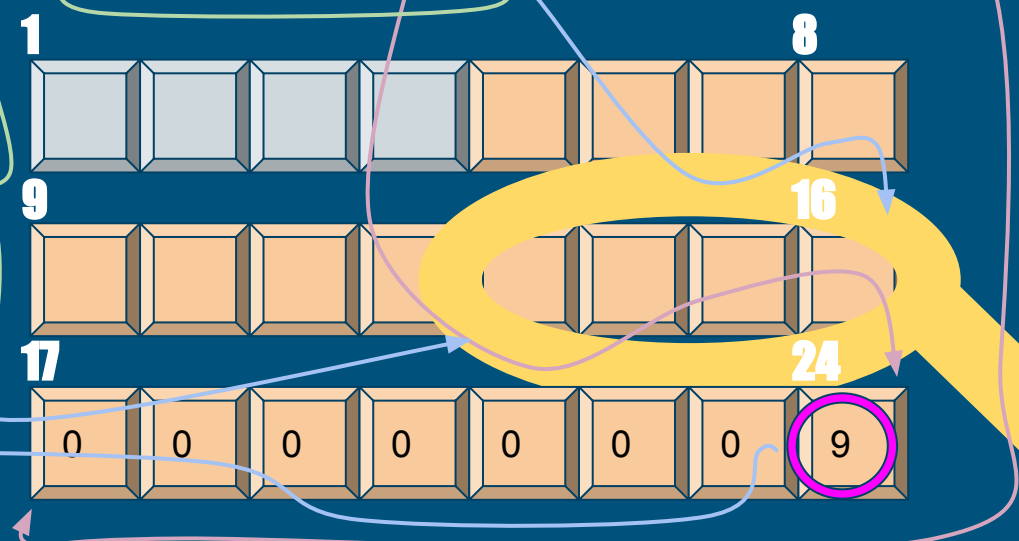
```
int main() {  
    int a[3];  
    int* p = &a[0];  
    print(*p);  
    p++;  
    print(*p);  
    print(* (p+1));  
    print(a[2]);  
    print(p[1]);  
}
```

Name	Type	Address
a	int	5
p	ptr (int)	17





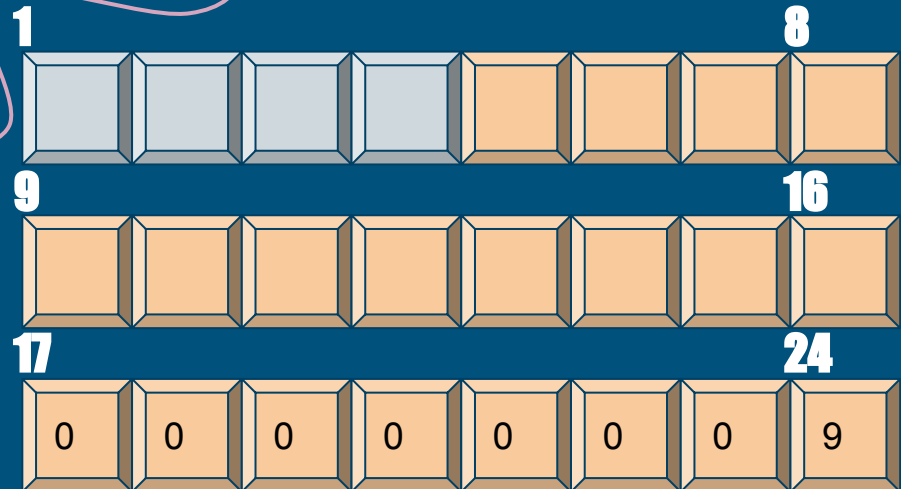
The diagram illustrates a memory layout for a linked list. At the top, a header table contains three fields: 'p', 'ptr (int)', and '17'. Below the header are three rows of memory blocks. The first row has 8 blocks, the second has 16 blocks, and the third has 24 blocks. A yellow oval highlights a specific block in the second row. A pink circle highlights the value '9' in the last block of the third row. Arrows indicate pointers and data flow between the header and the blocks.



Pointerarithmetik

```
int main() {  
    int a[3];  
    int* p = &a[0];  
    print(*p);  
    p++;  
    print(*p);  
    print(*(p+1));  
    print(a[2]);  
    print(p[1]);  
}
```

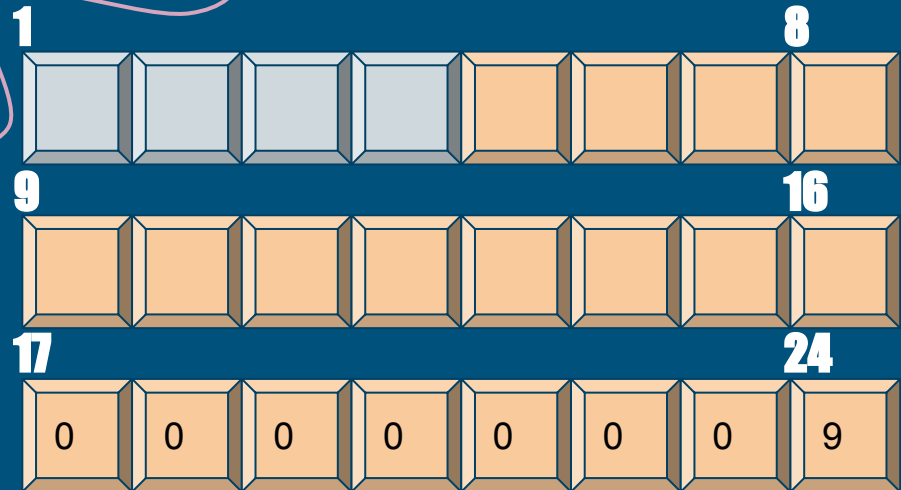
Name	Type	Address
a	int	5
p	ptr (int)	17



Pointerarithmetik

```
int main() {  
    int a[3];  
    int* p = &a[0];  
    print(*p);  
    p++;  
    print(*p);  
    print(*(p+1));  
    print(a[2]);  
    print(p[1]);  
}
```

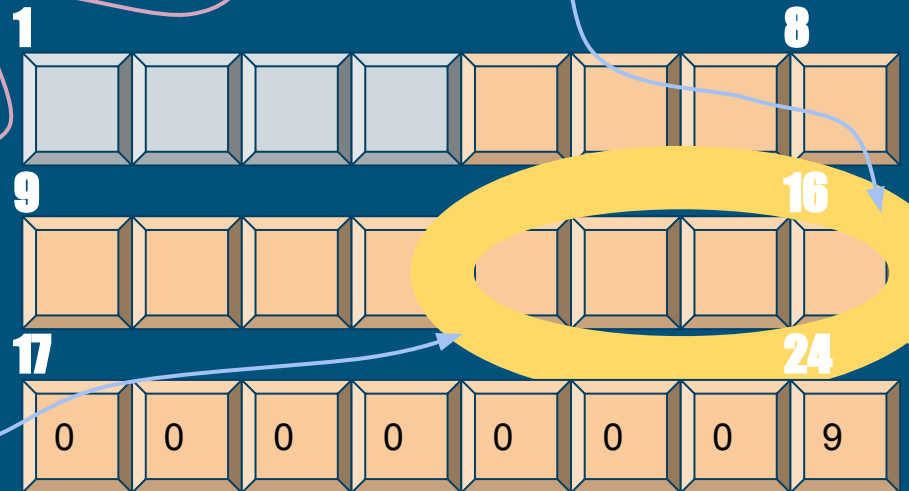
Name	Type	Address
a	int	5
p	ptr (int)	17



Pointerarithmetik

```
int main() {
    int a[3];
    int* p = &a[0];
    print(*p);
    p++;
    print(*p);
    print(*(p+1));
    print(a[2]);
    print(p[1]);
}
```

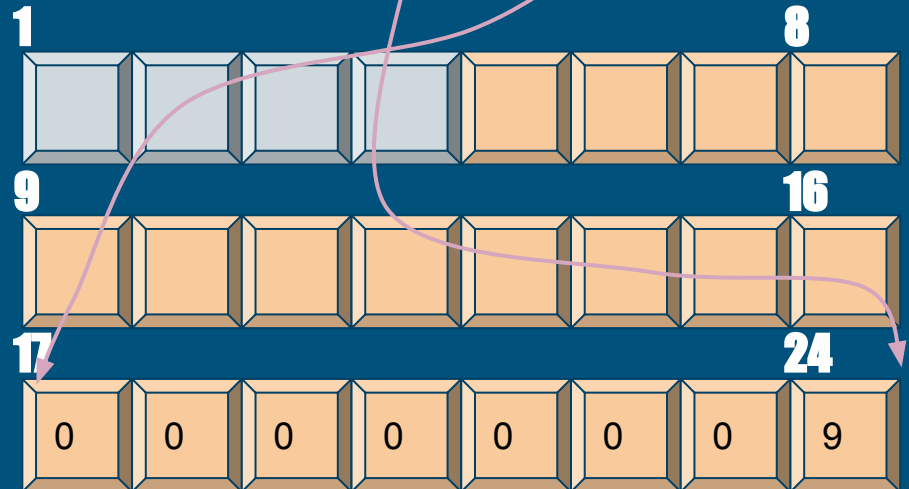
Name	Type	Address
a	int	5
p	ptr (int)	17



Pointerarithmetik

```
int main() {  
    int a[3];  
    int* p = &a[0];  
    print(*p);  
    p++;  
    print(*p);  
    print(*(p+1));  
    print(a[2]);  
    print(p[1]);  
}
```

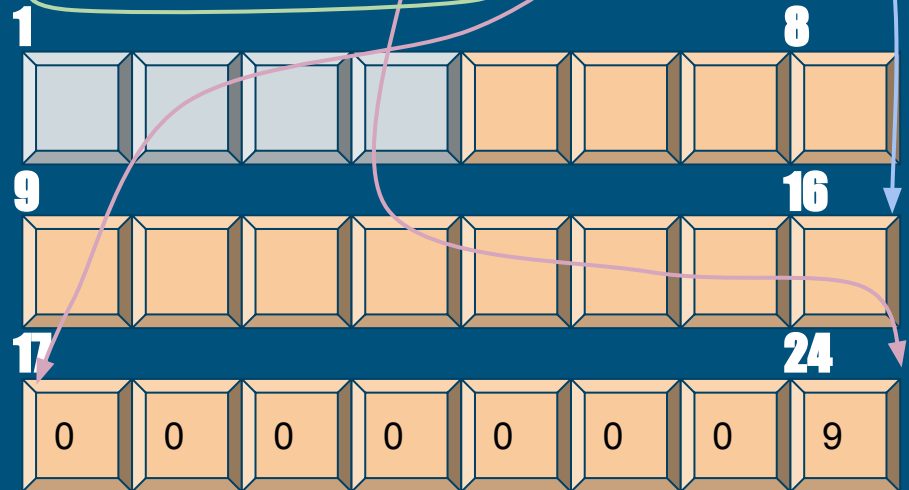
Name	Type	Address
a	int	5
p	ptr (int)	17



Pointerarithmetik

```
int main() {  
    int a[3];  
    int* p = &a[0];  
    print(*p);  
    p++;  
    print(*p);  
    print(*(p+1));  
    print(a[2]);  
    print(p[1]);  
}
```

Name	Type	Address
a	int	5
p	ptr (int)	17



Pointerarithmetik

```
int main() {  
    int a[3];  
    int* p = &a[0];  
    print(*p);  
    p++;  
    print(*p);  
    print(*(p+1));  
    print(a[2]);  
    print(p[1]);  
}
```

Name	Type	Address
a	int	5
p	ptr (int)	17

