Pointer

- For each type T exists a pointer to T.
- The pointer has the memory address in which the data is.
- Using (`*`) makes `T` to a pointer to `T`:

```
int i = 20;
int* iptr = &i;
```

- The `&` symbol returns the memory address of T.
- A pointer can only refer to an address of the same type.

Pointer

- Dereferencing
  - `iptr` returns the value (pointer)
  - `*iptr` dereferences the pointer (underlying value)

```
int i = 20;
int* iptr = &i;

int* j1 = iptr;
int j2 = *iptr;
```

- Pointer arithmetic
  - shows the relation between pointers and C-arrays.

```
int intArray[] = {1, 2, 3, 4, 5};
intArray[3] ≡ *(intArray + 3)
```

Pointer

`nullptr`

- By assign a `nullptr` to pointer, the pointer becomes a null pointer
- Cannot refer to a value and cannot be dereferenced.
- Can be compared with each pointer and can be converted to each pointer.
- Can only be converted into a boolean.
- Can be used in a logical expression.

```
int* a = nullptr;
if (!a) std::cout << "will be called\n";
```

Don't use 0 or `NULL` as a null pointer.

- `0`: can be the null pointer (`(void*)0`) or the natural number 0
- `NULL`: macro