Reinterpret Cast

This lesson highlights the key features of the reinterpret_cast operator.

WE'LL COVER THE FOLLOWING

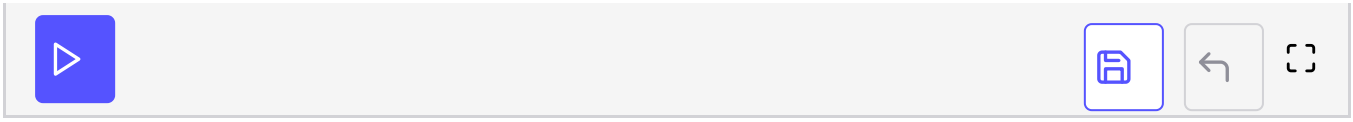
Features

- `reinterpret_cast` allows us to convert a pointer of a particular type to a pointer of any other type, regardless of whether the types are related or not.
- It also allows conversion between a **pointer** and an **integral**.
- `reinterpret_cast` guarantees that if a pointer is cast into another type, casting it back would return the original value.
- The use of `reinterpret_cast` is not recommended as it does not take any safety measures before converting between types. This can result in faulty or accidental conversions that could harm the code.

Example

```
#include <iostream>

int main(){

  double * myDouble = new double(3.14);
  std::cout << *myDouble << std::endl;

  void * myVoid = reinterpret_cast<void*>(myDouble);

  double * myDouble1 = reinterpret_cast<double*>(myVoid); // Original value retrieved
  std::cout << *myDouble1 << std::endl;
```

```
}
```



- The `myDouble` pointer is cast into the `void` type in line 8.
- If we cast `myVoid` back into a `double` pointer, the original value, `3.14`, is retrieved.

Further information

- typecasting

That ends our discussion on named explicit casts. In the next lesson, we will dive a little deeper into `type_id`.