- Examples

In this lesson, we can see a few instances of constexpr functions.

WE'LL COVER THE FOLLOWING

Assertions

```
#include <iostream>

constexpr int square(int x) { return x * x; }
constexpr int squareToSquare(int x){ return square(square(x));}

int main() {

  std::cout << std::endl;

  static_assert(square(10) == 100, "you calculated it wrong");
  static_assert(squareToSquare(10) == 10000 , "you calculated it wrong");

  std::cout<< "square(10) = " << square(10) << std::endl;
  std::cout<< "squareToSquare(10) = " << squareToSquare(10) << std::endl;
  constexpr int constExpr = square(10);

  int arrayClassic[100];
  int arrayNewWithConstExpression[constExpr];
  int arrayNewWithConstExpressioFunction[square(10)];

  std::cout << std::endl;

}
```

Explanation

- In the example above, we have implemented two `constexpr` functions for C++11: `constexpr int square(int x)` and `constexpr int squareToSquare(int x)`. As we can see, both follow the conventions for `constexpr` functions in C++11.
- The assertions in lines 10 and 11 will hold because they can be evaluated at compile-time. Making it a `constexpr` variable will allow the code compilation to pass the assertions.
- In line 15, we have initialized a `constexpr` variable, `constExpr`, using the `sqaure` function.
- In lines 17-19, we have initialized three arrays:
  - by using a constant, `100`.
  - by using a `constexpr` variable, `constExpr`.
  - by calling the function `square(10)`. Notice that the input argument for this function call is constant.

`auto`

```
#include <iostream>

constexpr int gcd(int a, int b){
  while (b != 0){
    auto t = b;
    b = a % b;
    a = t;
  }
  return a;
}

int main(){

  std::cout << std::endl;

  constexpr auto res = gcd(100, 10);
  std::cout << "gcd(100, 10) " << res << std::endl;

  auto val = 100;
  auto res2 = gcd(val, 10);
  std::cout << "gcd(val, 10) " << res2 << std::endl;

}
```

Explanation

- The difference between ordinary functions and constexpr functions in C++14 is minimal. Therefore, it's quite easy to implement the `gcd` algorithm in C++14 as a `constexpr` function.
- We have defined `res` as a `constexpr` variable and its type is automatically determined by `auto`. However, `res2` is not `constexpr`.

Let's test our knowledge of this content with the exercise in the next lesson.