# - Examples

In this lesson, we'll get into examples of constexpr.

WE'LL COVER THE FOLLOWING



## Example 1: `constexpr` using C++ 11 #

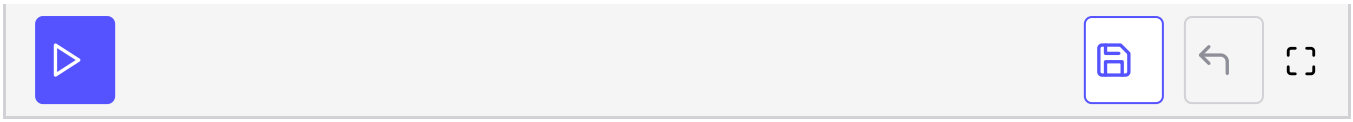
```
// constExpression.cpp

#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: `constexpr int square(int x)` and `constexpr int squareToSquare(int x)`. As you can see, both the functions follow the conventions for `constexpr` functions definition in C++11.
- The assertion in lines 12 and 13 succeed since `10` is a literal type. Making a `constexpr` variable will allow the code compilation to pass the assertions.
- In line 17, we have initialized a `constexpr` variable `constExpr` using the `sqaure` function.
- In lines 19-21, we have initialized three arrays
    1. by using a constant 100
    2. by using a `constexpr` variable `constExpr`
    3. by calling the function `square(10)`. Notice that the input argument for this function call is constant.

# Example 2: `constexpr` function in C++ 14

```
// constExpressionCpp14.cpp

#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 #

- Line 18 calculates the result `res` at compile-time, and line 22 `res2` at runtime.
- 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 `constexpr` variable and its type is automatically determined by `auto`.

---

We'll solve an exercise in the next lesson.