Variadic Templates

Let's learn about variadic templates in detail in this lesson.

WE'LL COVER THE FOLLOWING

Variadic Templates

A variadic template is a template that can have an arbitrary number of parameters.

```
template <typename ... Args>
void variadicTemplate(Args ... args){ . . . . }
```

Parameter Pack

A template parameter pack is a template parameter that accepts zero or more template arguments (non-types, types, or templates). A function parameter pack is a function parameter that accepts zero or more function arguments.

- By using the ellipsis (...), `Args-` or `args` becomes a parameter pack.
- `Args` is a template parameter pack; `args` is a function parameter pack.
- Parameter packs can only be packed and unpacked.
- If the ellipsis is at the left of `Args`, the parameter pack will be packed and if the ellipse is at the right of `Args`, the parameter pack will be unpacked.

> The compiler can automatically deduce the template arguments in the case of a function template.

For example, the following classes/functions in the STL extensively use

variadic templates. Variadic Templates are often used in the Standard Template Library:

- `sizeof`-Operator, `std::tuple`, `std::thread`, `std::make_unique`, `std::lock`

The usage of parameter packs obeys a typical pattern for class templates.

- Perform an operation on the first element of the parameter pack and recursively invoke the operation on the remaining elements.
- The recursion ends after a finite number of steps.
- The boundary condition is typically a fully specialized template.

```
template<>
struct Mult<>{ .... }
template<int i, int ... tail >
struct Mult<i, tail ...>{ ....
```

To learn more about variadic templates, click here.

In the next lesson, we'll study examples of variadic templates.