Debug Printing

Now, we shall study a print function which uses several new facilities of C++.

In this lesson we'll examine a few more functions available in C++17. Do not worry about the code itself.

Here are all the functions and utilities you need to focus on:

- `if constexpr`
- `_v` templates
- `linePrinter`

```
#include <iostream>

template<typename T> void linePrinter(const T& x) {
  if constexpr (std::is_integral_v<T>)
    std::cout << "num: " << x << '\n';
  else if constexpr (std::is_floating_point_v<T>) {
    const auto frac = x - static_cast<long>(x);
    std::cout << "flt: " << x << ", frac " << frac << '\n';
  }
  else if constexpr(std::is_pointer_v<T>) {
    std::cout << "ptr, ";
    linePrinter(*x);
  }
  else
    std::cout << x << '\n';
}

template<typename ... Args> void printWithInfo(Args ... args) {
  (linePrinter(args), ...); // fold expression over the comma operator
}

int main () {
  int i = 10;
  float f = 2.56f;
  printWithInfo(&i, &f, 30);
}
```

Here you can see the following features:

• **Line 5, 7, 11**: `if constexpr` - to discard code at compile time, used to match the template parameter.

• **Line 5, 7, 11**: `_v` variable templates for type traits - no need to write `std::trait_name<T>::value`.

• **Line 20**: Fold Expressions inside `printWithInfo`. This feature simplifies variadic templates. In the example we invoke `linePrinter()` over all input arguments.

What we have seen here is just a small shadow of what's to come ahead. See you in the next section!