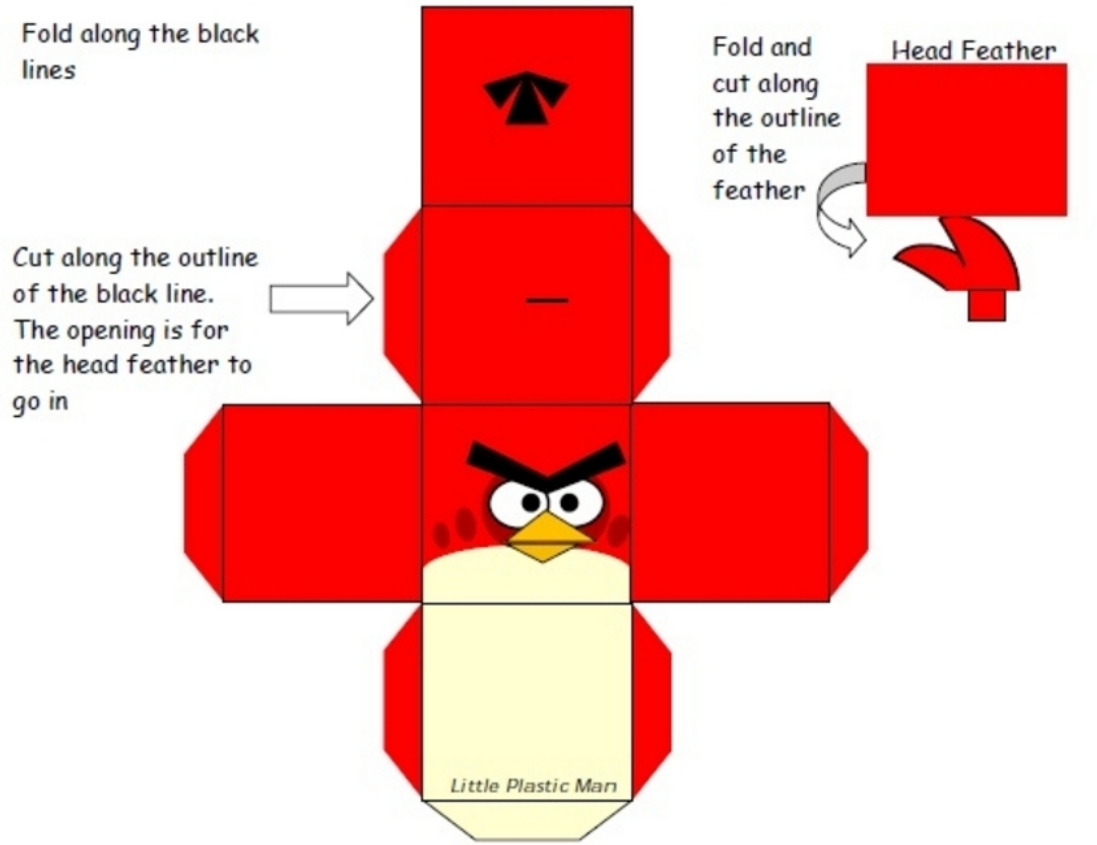
template-template parameters IRL



Templates

- They can vary in types
- They can vary in constants values

```
template <class T>
struct add {
  T operator()( T x, T y) { return x + y; }
};
```

Templates with template-template parameter

- They can vary infinitely deep in behavior
- From the call site
- That's compile-time callbacks

```
template<template <class> class callback, class T>
T add_customizable(T x, T y) { return callback<T>()(x, y); }
```


DARK
SIDE

That's the dark side

```
%:include <iostream>

int main(int argc, char *argv<::>)
<%
    if (argc > 1 and argv<:1:> not_eq NULL) <%
        std::cout << "Hello, " << argv<:1:> << '\n';
    %>
%>
```

That's not the dark side

```
template<
    template <class> class callback,
    class T
>
T add_customizable(T x, T y) {
  return callback<T>()(x, y);
}

add_customizable<add>(1, 1) == 2;
```

For what it's good for ? - I

- Actually it allows templates lazy expression
  - Who knows what is a template metafunctions ?
  - Who knows what is being lazy ?
  - ::type vs mpl::_ ?

For what it's good for ? - II

- It allows compile-time lambdas
  - Do you know mpl::lambda ?
  - mpl::placeholders ?
- Which allows pure awesomeness
  - bf::filter_view, mpl::transform...

Please don't ask anything

Thanks.