Type Deduction

Type deduction for function templates:

```cpp
#include <cstdio>

template <typename T>
void foo(T x) {
  puts(__PRETTY_FUNCTION__);
}

int main() {
  foo(4);         // void foo(T) [T = int]
  foo(4.2);       // void foo(T) [T = double]
  foo("hello");   // void foo(T) [T = const char *]
}
```

Type Deduction

Type deduction for function templates:

```cpp
template <typename T, typename U>
void f(T x, U y) {
  // ..
}
template <typename T>
void g(T x, T y)
// ..
}

int main() {
  f(1, 2);  // void f(T, U) [T = int, U = int]
  f(1, 2u); // void f(T, U) [T = int, U = unsigned int]
  g(1, 2);  // void g(T, T) [T = int]
  g(1, 2u); // error: no matching function for call
            // to g(int, unsigned int)
}
```

Type Deduction

Type deduction for class templates:

```
1  template <typename T>
2  struct Foo {
3   public:
4    Foo(T x) : x_(x) {}
5    T x_;
6  };
7
8  int main() {
9    auto obj = Foo<int>(10).x_;
10   auto same_obj = Foo(10).x_;
11   auto vec = std::vector<int>{10, 50};
12   auto same_vec = std::vector{10, 50};
13 }
```

Note: New in C++17