斐波那契数列

> —— 通过模板元编程实现编译期计算

斐波那契数列的定义如下：

- 如果 `N == 0`，`Fib(N)` 为 0
- 如果 `N == 1`，`Fib(N)` 为 1
- 对于任意 `N >= 2`，`Fib(N)` 为 `Fib(N - 1) + Fib(N - 2)`

参考相关教程（例如，*A gentle introduction to Template Metaprogramming with C++*），在给定代码框架的基础上实现通过模板元编程在编译期计算斐波那契数列，不需要考虑算术运算溢出问题。

输入

本题不需要处理输入。

输入格式参见代码框架的 `main` 函数。

输出

本题不需要处理输出。

输出格式参见代码框架的 `test_*` 系列函数。

提示

- 你只需要实现斐波那契数列递归定义的两个基本情况

扩展

- 了解 `constexpr` 关键字及其与 `const` 的不同
- 完成代码后，在 C++ Insights 上查看模板实例化的结果
- 探索使用模板元编程计算斐波那契数列是否有上限（同样可以借助 C++ Insights 实现）。如果有，上限是多少？能够调整这个上限？

代码框架

```cpp
#include <cstdint>
#include <functional>
#include <iostream>
#include <unordered_map>
```

```cpp
template <uintmax_t N>
struct Fib {
  static constexpr uintmax_t value = Fib<N - 1>::value + Fib<N -
2>::value;
};

// TODO: your code

void test_1() { std::cout << Fib<0>::value << std::endl; }

void test_2() { std::cout << Fib<1>::value << std::endl; }

void test_3() { std::cout << Fib<2>::value << std::endl; }

void test_4() { std::cout << Fib<5>::value << std::endl; }

void test_5() { std::cout << Fib<20>::value << std::endl; }

int main() {
  std::unordered_map<std::string, std::function<void()>>
test_cases_by_name = {
      {"test_1", test_1}, {"test_2", test_2}, {"test_3", test_3},
      {"test_4", test_4}, {"test_5", test_5},
  };
  std::string tname;
  std::cin >> tname;
  auto it = test_cases_by_name.find(tname);
  if (it == test_cases_by_name.end()) {
    std::cout << "输入只能是 test_<N>, 其中 <N> 可取整数 1 到 5." <<
std::endl;
    return 1;
  }
  (it->second)();
}
```