Metaprogramowanie przy użyciu szablonów C++ jest pełne w sensie Turinga.
Aby się o tym przekonać, spróbujemy stworzyć prosty język funkcyjny i jego interpreter
oparty o szablony C++.

Język Fibin wspiera:

* Literały Lit
  Literałami liczbowymi mogą być tylko liczby Fibonacciego, np. Fib<0> = 0, Fib<1> = 1 itd.
  Parametry Fib muszą być nieujemne.
  Literałami logicznymi mogą być tylko dwie wartości logiczne: True, False.
  Przykłady poprawnych literałów: Lit<Fib<0>>, Lit<True>.

* Zmienne Var
  Var(const char*) - identyfikator zmiennej; identyfikatory tworzone są na podstawie
  ciągów znaków o długości od 1 do 6 włącznie, zawierających małe i wielkie litery
  alfabetu angielskiego (a-zA-Z) oraz cyfry (0-9); małe i wielkie litery nie są
  rozróżniane.
  Przykłady poprawnych zmiennych: Var("A"), Var("01234"), Var("Cdefg").

* Operacje arytmetyczne Sum, Inc1, Inc10
  Sum<...> - operacja dodawania wielu liczb, wymagane są co najmniej dwa argumenty.
  Inc1<Arg> - specjalizacja dodawania, która zwiększa wartość Arg o Fib<1>.
  Inc10<Arg> - specjalizacja dodawania, która zwiększa wartość Arg o Fib<10>.
  Przykłady poprawnych operacji: Sum<Lit<Fib<0>>, Lit<Fib<1>>, Lit<Fib<3>>>,
  Inc1<Lit<Fib<0>>>.

* Porównanie Eq
  Eq<Left, Right> - porównuje wartość Left z Right; zwraca True, gdy są równe,
  a False w przeciwnym przypadku.
  Przykład poprawnego porównania: Eq<Lit<Fib<0>>, Lit<Fib<1>>>.

* Odwołanie do zmiennej Ref
  Ref<Var> - zwraca wartość zmiennej identyfikowanej przez Var.
  Przykład poprawnego odwołania do zmiennej: Ref<Var("A")>.

* Wyrażenie Let
  Let<Var, Value, Expression> - do zmiennej Var przypisuje wartość Value
  i oblicza wartość Expression.
  Przykład poprawnego wyrażenia: Let<Var("A"), Lit<Fib<1>>, Ref<Var("A")>>.

* Wyrażenie If
  If<Condition, Then, Else> - jeśli Condition jest True, to oblicza wartość Then,
  a w przeciwnym przypadku oblicza wartość Else.
  Przykład poprawnego wyrażenia: If<Lit<True>, Lit<Fib<1>>, Lit<Fib<0>>>.

* Wyrażenie Lambda
  Lambda<Var, Body> - reprezentuje anonimową funkcję z jednym parametrem Var
  oraz ciałem Body.
  Przykład poprawnego wyrażenia: Lambda<Var("x"), Ref<Var("x")>>.

* Wywołanie funkcji
  Invoke<Fun, Param> - oblicza wartość funkcji Fun dla parametru Param.
  Przykład poprawnego wyrażenia: Invoke<Lambda<Var("x"), Ref<Var("x")>>, Lit<Fib<0>>>.

Należy dostarczyć klasę szablonową Fibin:

* Fibin<ValueType> - ValueType jest typem wartości używanym do obliczania wartości
  wyrażeń.

Klasa Fibin udostępnia publiczną metodę szablonową:

* ValueType eval<Expr> - metoda dostępna tylko wtedy, gdy ValueType jest typem liczbowym.
  Oblicza w trakcie kompilacji wartość wyrażenia Expr i zwraca tę wartość.

* void eval<Expr> - metoda dostępna tylko wtedy, gdy ValueType NIE jest typem liczbowym.
  Wyświetla informację, że Fibin nie wspiera danego typu.

Przykład użycia podany jest w pliku fibin_example.cc. Wynik działania przykładu to:

Fibin doesn't support: PKc
Fibin works fine!

= Wymagania formalne =

W tym zadaniu wymagane jest użycie kompilatora Clang, który wypisuje bardzo
szczegółowe komunikaty o błędach w przypadku korzystania z szablonów
i metaprogramowania. Rozwiązanie będzie kompilowane za pomocą polecenia

clang -Wall -Wextra -std=c++17 -O2 -lstdc++

Rozwiązanie powinno zawierać plik fibin.h.
Plik należy umieścić w repozytorium w katalogu:

grupaN/zadanie4/ab123456+cd123456

lub

grupaN/zadanie4/ab123456+cd123456+ef123456

gdzie N jest numerem grupy, a ab123456, cd123456, ef123456 są identyfikatorami
członków zespołu umieszczającego to rozwiązanie. Katalog z rozwiązaniem nie
powinien zawierać innych plików, ale może zawierać podkatalog prywatne, gdzie
można umieszczać różne pliki, np. swoje testy. Pliki umieszczone w tym
podkatalogu nie będą oceniane. Nie wolno umieszczać w repozytorium plików
dużych, binarnych, tymczasowych (np. *.o) ani innych zbędnych.