

# SwiftMacros

# Что это такое

## SwiftMacros

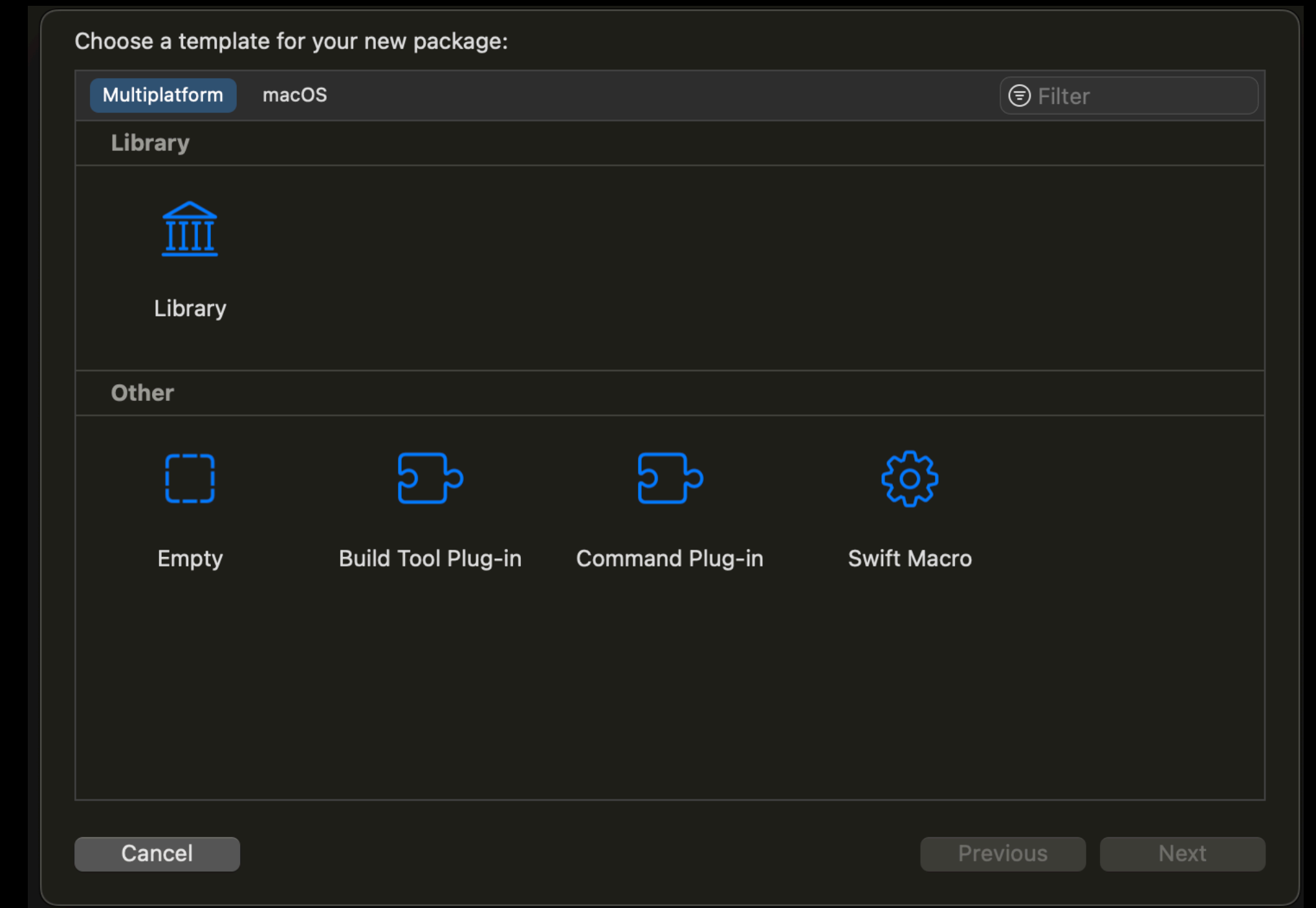
- Позволяют автоматизировать рутинные действия и использовать шаблоны при написании кода

# Типы SwiftMacros

- `@freestanding` - начинается с символа `#`макрос и располагается в любом месте кода
- `@attached` - начинается с `@`Макрос и может стоять перед определением

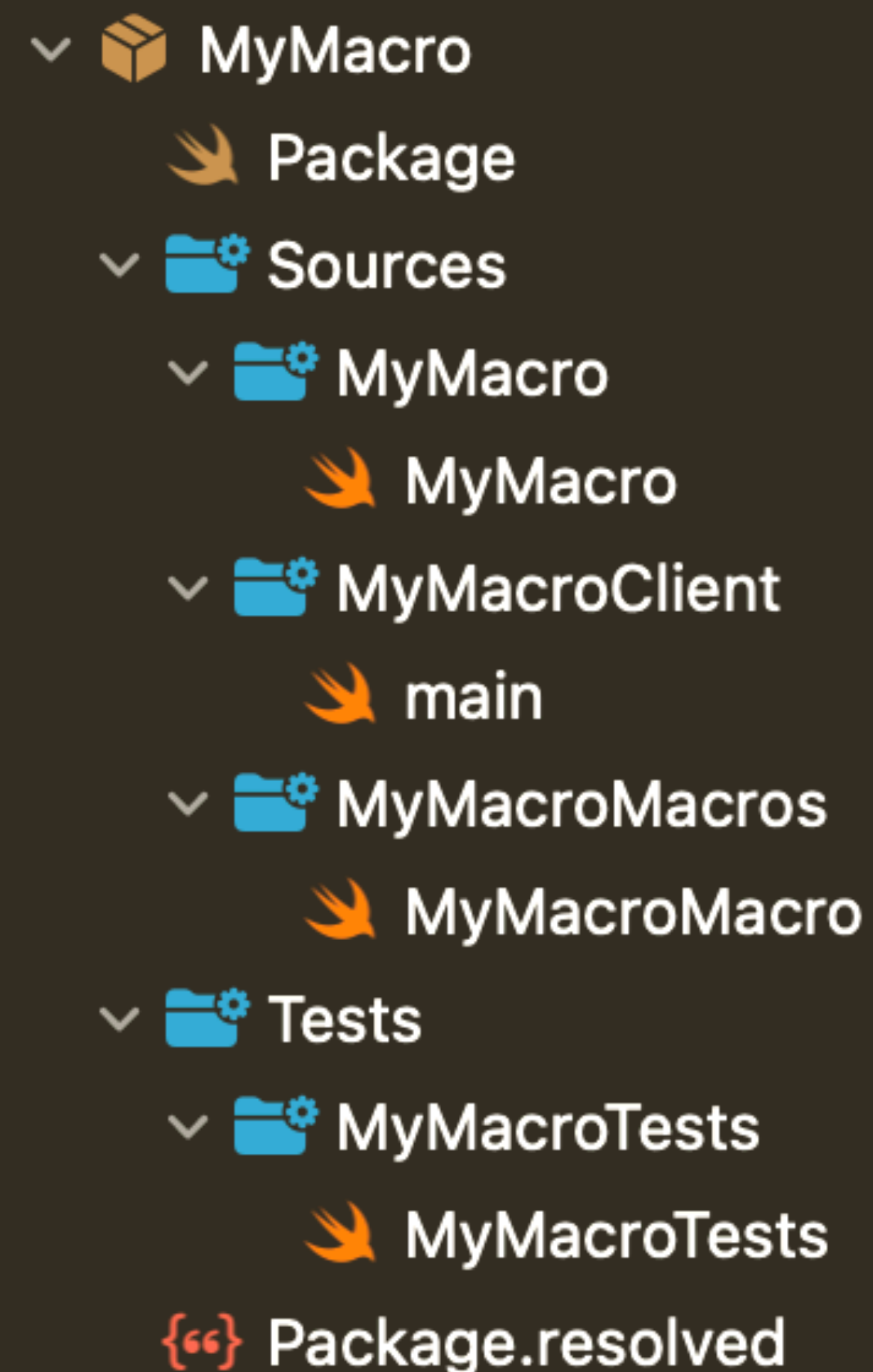
# Создание swift package для SwiftMacros

- Создается через New -> Package и выбрать Swift Macro



# Структура SwiftMacros

- MyMacro - сигнатура самого макроса
- Main - файл, в котором можно потестировать поведение макроса
- MyMacrosMacros - файл с имплементацией
- MyMacroTests - файл с тестами



```
graph TD; MyMacro[MyMacro] --> Package[Package]; MyMacro --> Sources[Sources]; Sources --> MyMacroSources[MyMacro]; MyMacroSources --> MyMacroFile[MyMacro]; Sources --> MyMacroClient[MyMacroClient]; MyMacroClient --> mainFile[main]; Sources --> MyMacroMacros[MyMacroMacros]; MyMacroMacros --> MyMacroMacroFile[MyMacroMacro]; Sources --> Tests[Tests]; Tests --> MyMacroTests[MyMacroTests]; MyMacroTests --> MyMacroTestsFile[MyMacroTests]; PackageResolved[Package.resolved];
```

MyMacro

- Package
- Sources
  - MyMacro
    - MyMacro
  - MyMacroClient
    - main
  - MyMacroMacros
    - MyMacroMacro
- Tests
  - MyMacroTests
    - MyMacroTests
- Package.resolved

# Пример сигнатуры SwiftMacros

MyMacro

Package

Sources

MyMacro

MyMacro

MyMacroClient

main

MyMacroMacros

MyMacroMacro

Tests

MyMacroTests

MyMacroTests

Package.resolved

MyMacro > Sources > MyMacro > MyMacro > No Selection

1

// The Swift Programming Language

2

// <https://docs.swift.org/swift-book>

3

4

/// A macro that produces both a value and a string containing the

5

/// source code that generated the value. For example,

6

///

7

/// #stringify(x + y)

8

///

9

/// produces a tuple `(x + y, "x + y")`.

10

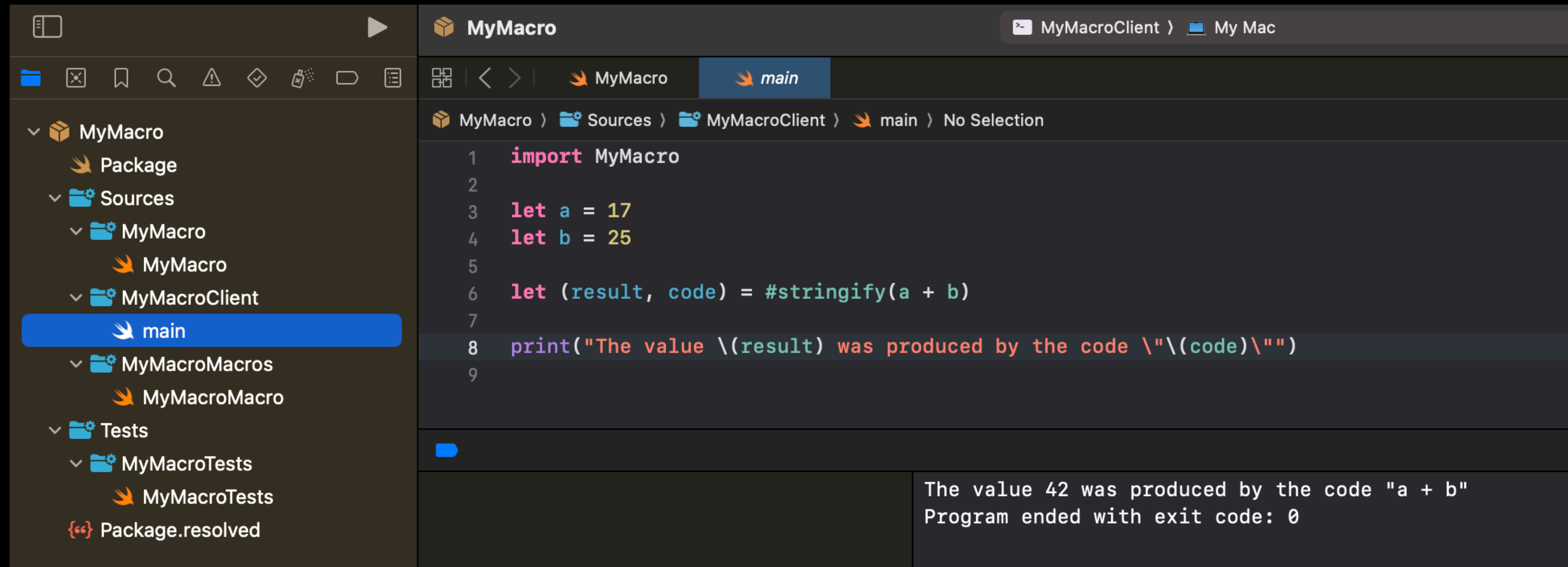
@freestanding(expression)

11

public macro stringify<T>(\_ value: T) -> (T, String) = #externalMacro(module: "MyMacroMacros", type: "StringifyMacro")

12

# Пример прогона макроса



# Имплементация SwiftMacros

MyMacro

Package

Sources

MyMacro

MyMacro

MyMacroClient

main

MyMacroMacros

MyMacroMacro

Tests

MyMacroTests

MyMacroTests

Package.resolved

Package Dependencies

MyMacro > Sources > MyMacroMacros > MyMacroMacro > expansion(of:in:)

```
1  import SwiftCompilerPlugin
2  import SwiftSyntax
3  import SwiftSyntaxBuilder
4  import SwiftSyntaxMacros
5
6  /// Implementation of the `stringify` macro, which takes an expression
7  /// of any type and produces a tuple containing the value of that expression
8  /// and the source code that produced the value. For example
9  ///
10 ///     #stringify(x + y)
11 ///
12 /// will expand to
13 ///
14 ///     (x + y, "x + y")
15 public struct StringifyMacro: ExpressionMacro {
16
17     public static func expansion(
18         of node: some FreestandingMacroExpansionSyntax,
19         in context: some MacroExpansionContext
20     ) -> ExprSyntax {
21
22         guard let argument = node.argumentList.first?.expression else {
23             fatalError("compiler bug: the macro does not have any arguments")
24         }
25
26         return "(\(argument), \(literal: argument.description))"
27     }
28 }
29
30 @main
31 struct MyMacroPlugin: CompilerPlugin {
32
33     let providingMacros: [Macro.Type] = [
34         StringifyMacro.self,
35     ]
36 }
37
```



# Тестирование SwiftMacros

MyMacro

Package

Sources

MyMacro

MyMacro

MyMacroClient

main

MyMacroMacros

MyMacroMacro

Tests

MyMacroTests

MyMacroTests

Package.resolved

Package Dependencies

MyMacro > Tests > MyMacroTests > MyMacroTests > testMacro()

```
1  import SwiftSyntaxMacros
2  import SwiftSyntaxMacrosTestSupport
3  import XCTest
4
5  // Macro implementations build for the host, so the corresponding module is not available when cross-compiling. Cross-compiled tests may still make use of the macro
   itself in end-to-end tests.
6  #if canImport(MyMacroMacros)
7  import MyMacroMacros
8
9  let testMacros: [String: Macro.Type] = [
10     "stringify": StringifyMacro.self,
11 ]
12 #endif
13
14 ✓ final class MyMacroTests: XCTestCase {
15     ✓ func testMacro() throws {
16         #if canImport(MyMacroMacros)
17         assertMacroExpansion(
18             """
19             #stringify(a + b)
20             """
21             ,
22             expandedSource: """
23             (a + b, "a + b")
24             """
25             ,
26             macros: testMacros
27         )
28         #else
29         throw XCTSkip("macros are only supported when running tests for the host platform")
30         #endif
31     }
32
33     ✓ func testMacroWithStringLiteral() throws {
34         #if canImport(MyMacroMacros)
35         assertMacroExpansion(
36             #"""
37             #stringify("Hello, \(name)")
38             """#,
39             expandedSource: #"""
40             ("Hello, \(name)", #"""Hello, \(name)""")
41             """#,
42             macros: testMacros
43         )
44         #else
45         throw XCTSkip("macros are only supported when running tests for the host platform")
46         #endif
47     }
48 }
```