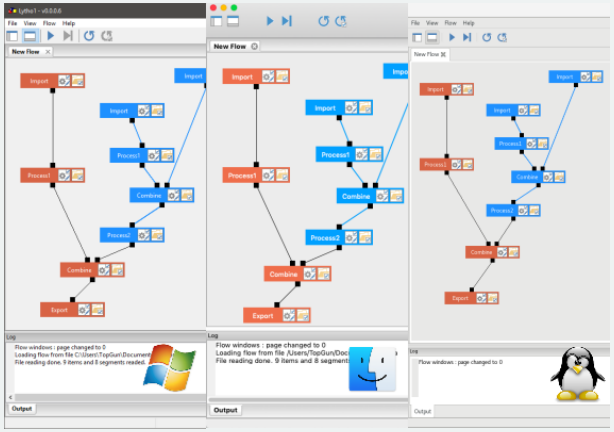
МИЭТ
Национальный исследовательский университет «МИЭТ»
Кафедра ПКИМС

# Кроссплатформенная разработка программного обеспечения



Лабораторная работа №3

**Общие вопросы разработки ПО под различные платформы.**

# Возможности использования кода в различных проектах

Исходный код


Скомпилированный код

Статическая библиотека

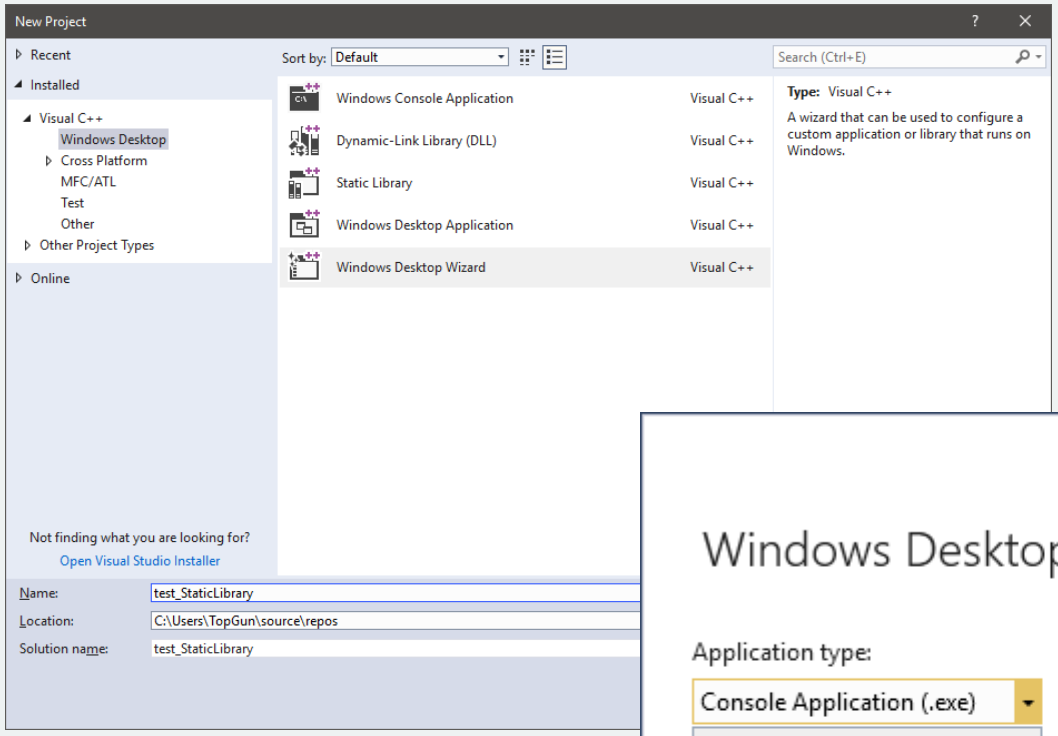
Qt5Core.lib

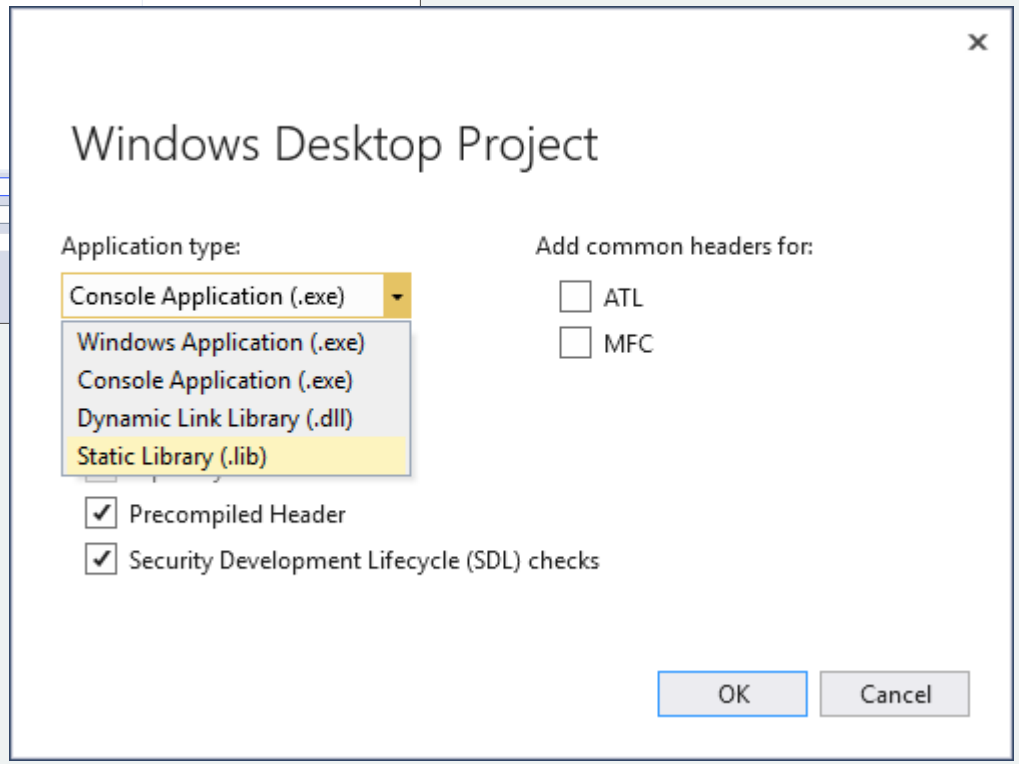
Динамическая библиотека

Qt5Core.dll

Обычно нужен заголовочный файл

# Создание статической библиотеки: создание проекта



# Создание статической библиотеки: код библиотеки

Файл main.cpp:

```
#include <iostream>
#include <clocale>

void print_message() {
  setlocale(LC_CTYPE, "rus");
  std::cout << "Привет из статической библиотеки!" << std::endl;
}
```

Лог компилятора:

```
1>------ Rebuild All started: Project: test_StaticLibrary, Configuration: Debug Win32 ------
1>main.cpp
1>test_StaticLibrary.vcxproj -> C:\Users\...\test_StaticLibrary\Debug\test_StaticLibrary.lib
========== Rebuild All: 1 succeeded, 0 failed, 0 skipped ==========
```

# Использование статической библиотеки: написание кода

Код:

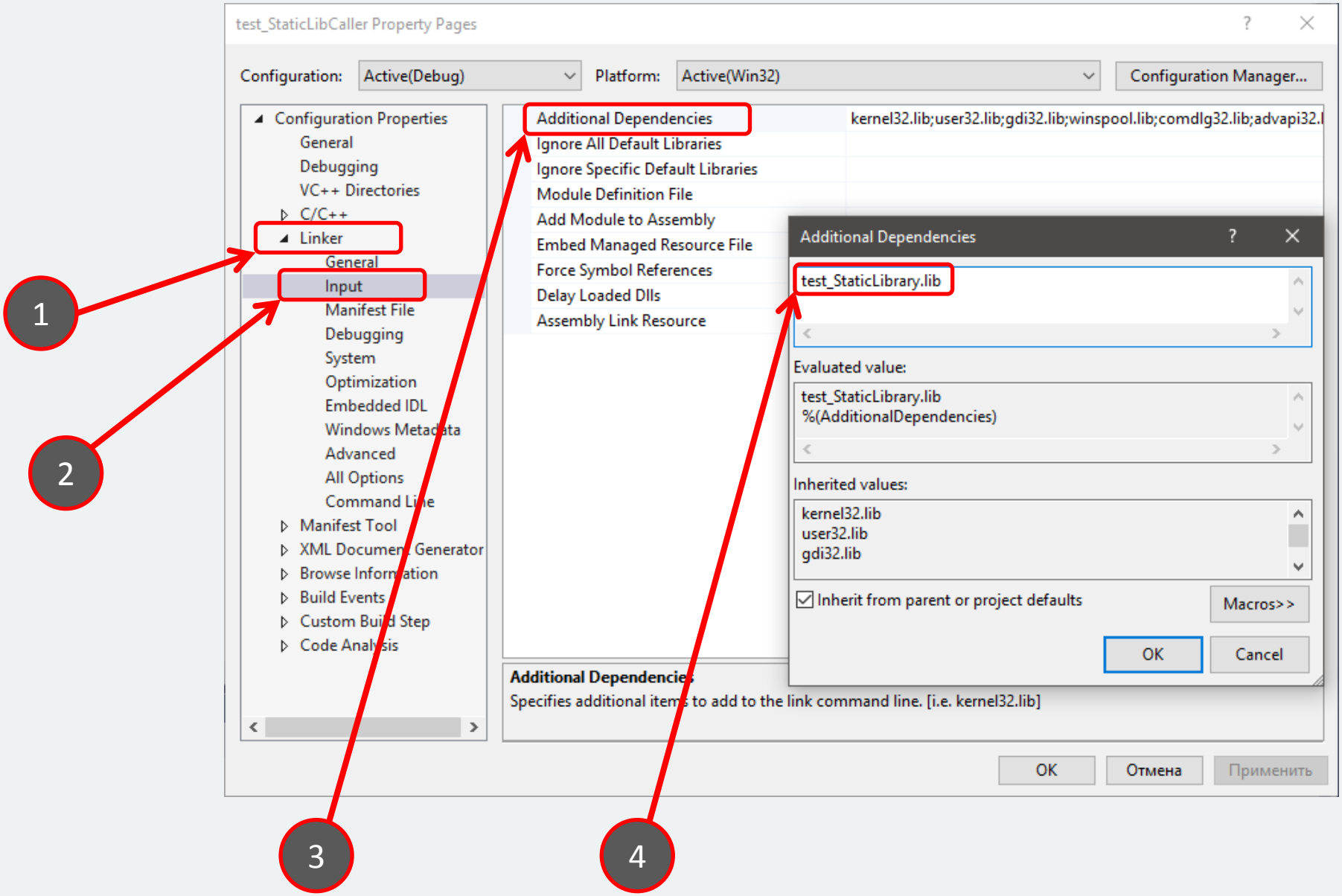
```
#include <iostream>

#pragma comment(lib, "test_StaticLibrary.lib")

void print_message();

int main(int argc, char *argv[]) {
  print_message();
  return EXIT_SUCCESS;
}
```

# Использование статической библиотеки: подключение в проекте



# Создание статической библиотеки: код библиотеки

Файл main.cpp:

```cpp
#include <iostream>
#include <clocale>

void print_message() {
  setlocale(LC_CTYPE, "rus");
  std::cout << "Привет из статической библиотеки!" << std::endl;
}
```

Файл header.hpp:

```cpp
#pragma once

void print_message();
```

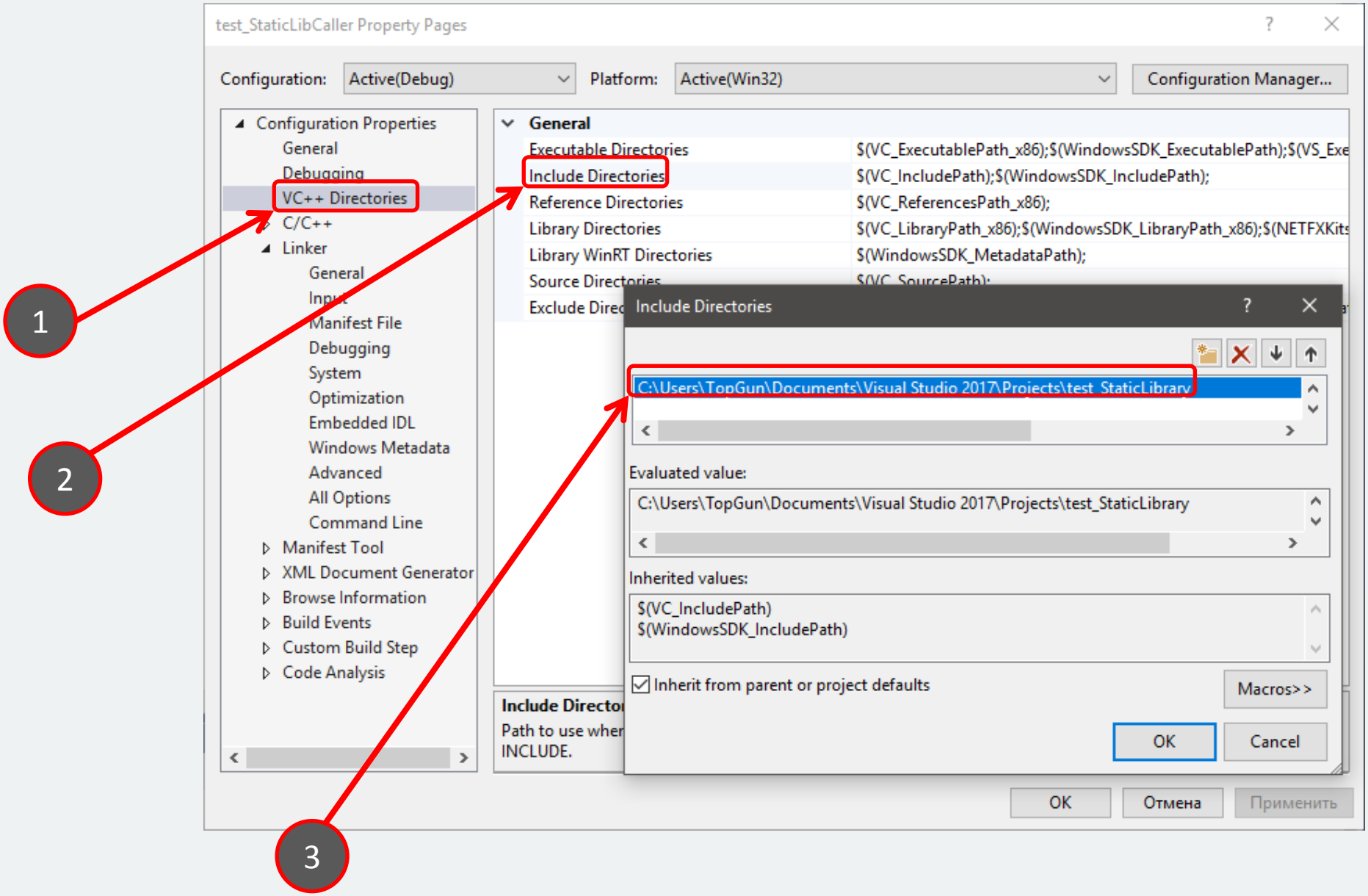
# Использование статической библиотеки: написание кода

Код:

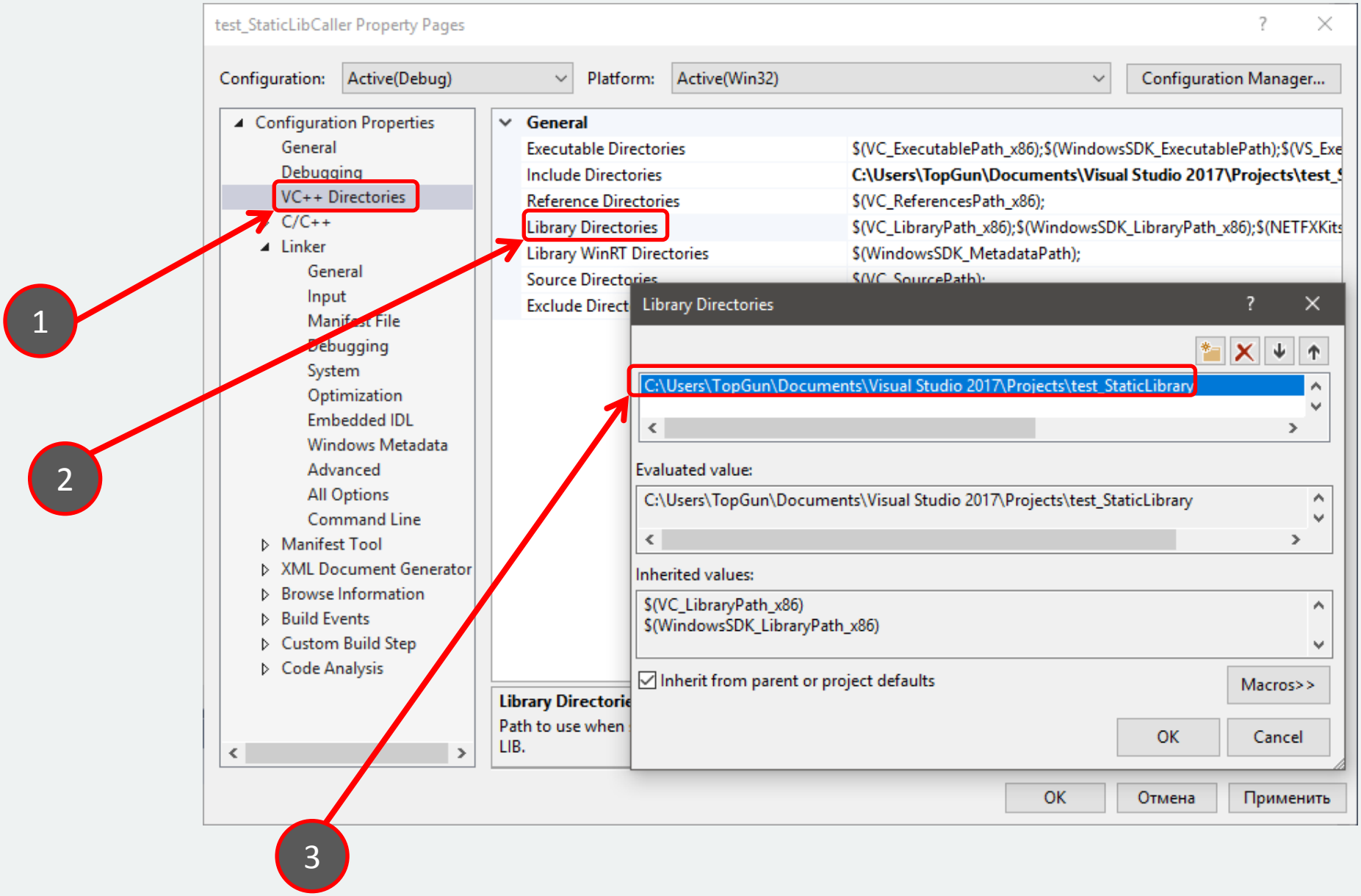
```
#include <iostream>
#include "Header.hpp"

int main(int argc, char *argv[]) {
  print_message();
  return EXIT_SUCCESS;
}
```

# Подключение статической библиотеки в проекте (1)



# Подключение статической библиотеки в проекте (2)



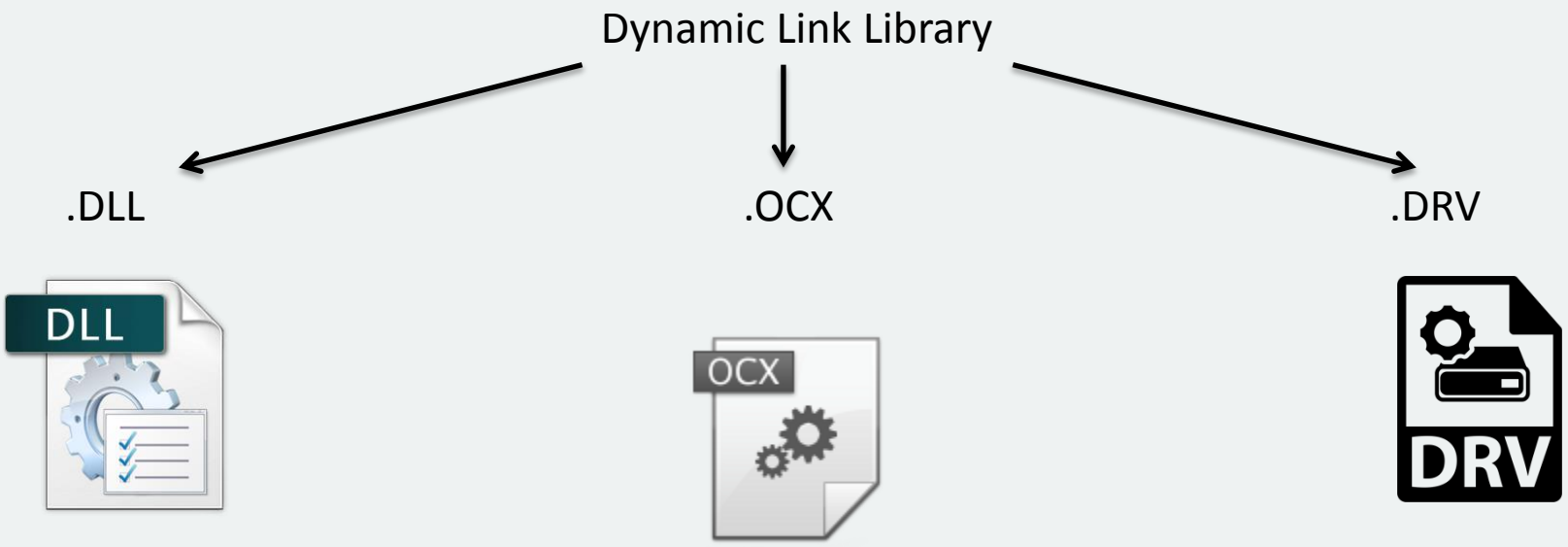
# Использование статической библиотеки: написание кода (2)

Код:

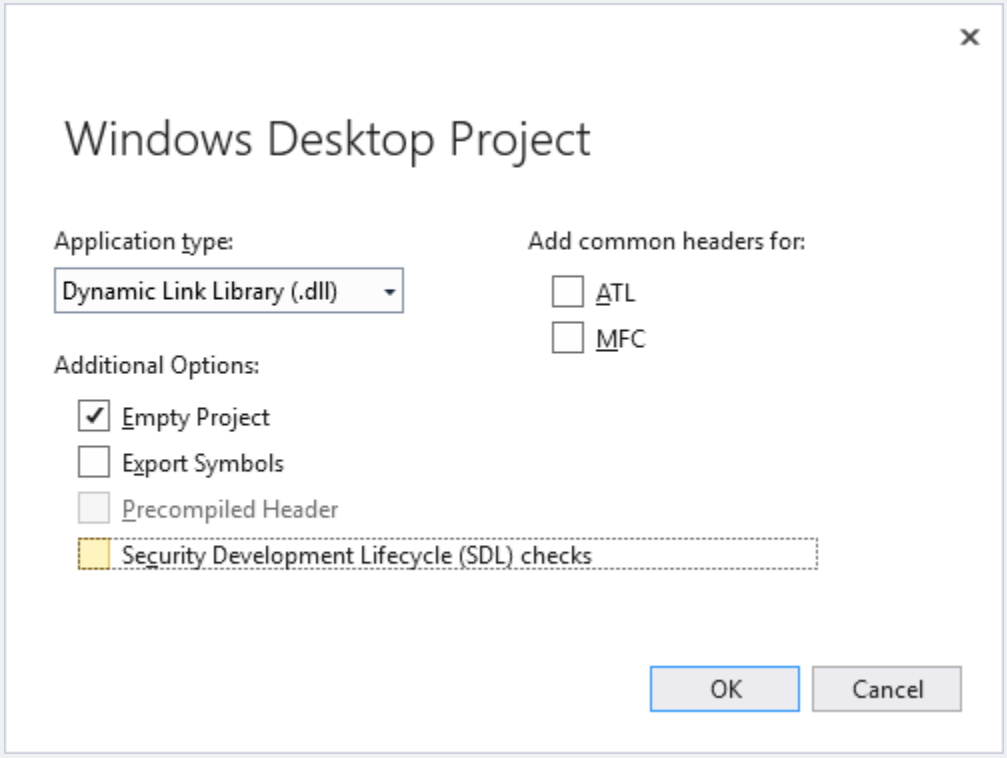
```
#include <iostream>
#include <Header.hpp>

int main(int argc, char *argv[]) {
  print_message();
  return EXIT_SUCCESS;
}
```

# Dynamic Link Library (DLL)



# Использование динамической библиотеки: создание проекта



# Разработка динамической библиотеки: код

Файл main.cpp:

```
#include <iostream>
#include <clocale>

__declspec(dllexport) void print_message() {
  setlocale(LC_CTYPE, "rus");
  std::cout << "Привет из динамической библиотеки!" << std::endl;
}
```

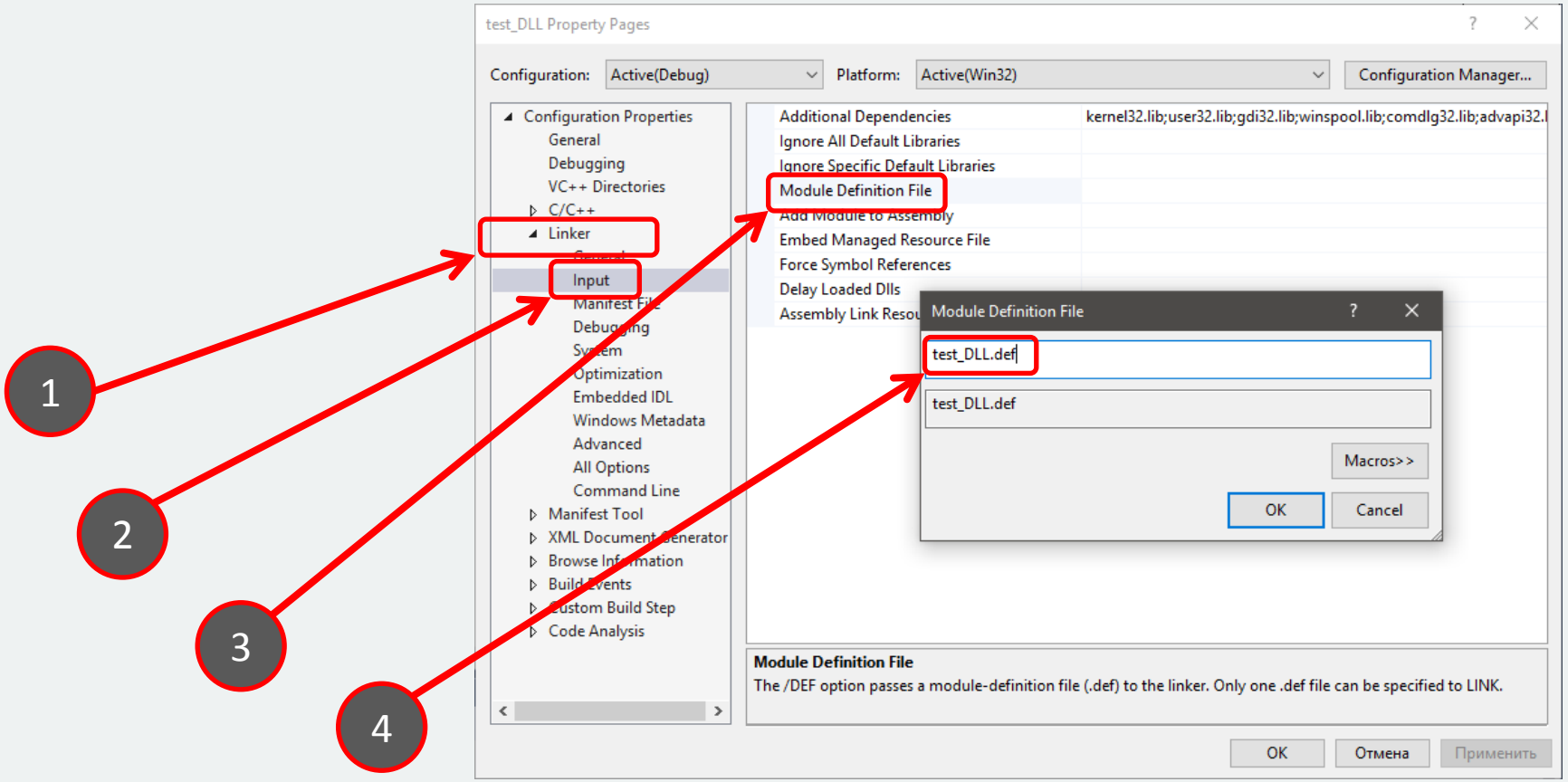
Лог компилятора:

```
1>------ Build started: Project: test_DLL, Configuration: Debug Win32 ------
1>main.cpp
1>test_DLL.vcxproj -> C:\Users\...\test_DLL\Debug\test_DLL.dll
========== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped ==========
```

# Разработка динамической библиотеки: файл определений

Файл определений **test_DLL.def** :

```
LIBRARY "test_DLL"
EXPORTS
        print_message
```



# Использование динамической библиотеки: написание кода

```
#include <Windows.h>
#include <iostream>
#include <clocale>

typedef void(*func_ptr)();

int main(int argc, char *argv[]) {
  setlocale(LC_CTYPE, "rus");
  HMODULE hdll = LoadLibrary(L"test_DLL.dll");
  if (!hdll) {
    std::cout << "Не могу загрузить файл DLL :(" << std::endl;
    return EXIT_FAILURE;
  }

  func_ptr func = (func_ptr)GetProcAddress(hdll, "print_message");
  if (func != NULL)
    func();

  FreeLibrary(hdll);

  return EXIT_SUCCESS;
}
```

# Компиляция кода в Linux

```
#include <stdio.h>

void main() {
  printf("%s", "PKIMS compiles under linux! :)");
}
```

Самый простой пример:

```
g++ ./main.cpp
```

Компиляция по шагам :

```
g++ -c -o ./main.o ./main.cpp

g++ -o ./main ./main.o
```

Компиляция в один шаг:

```
g++ -o ./main ./main.cpp
```

# Работа с динамическими библиотеками под ОС Linux (1)


task_01

Код библиотеки (library.cpp):

```
#include <iostream>

extern "C" void print_message() {
  std::cout << "Hello from the .so!" << std::endl;
}
```

# Работа с динамическими библиотеками под ОС Linux (2)

Код вызывающей програмы (main.cpp):


task_01

```
#include <dlfcn.h>
#include <iostream>

typedef void(*func_ptr)();

int main(int argc, char *argv[]) {
  void *handle = dlopen("./library.so", RTLD_LAZY);
  if (!handle) {
    std::cout << "Can't load shared object" << std::endl;
    return 1;
  }
  func_ptr hello = (func_ptr)dlsym(handle, "print_message");
  if (hello != NULL)
    hello();

  dlclose(handle);

  return 0;
}
```

# Компиляция библиотеки в Linux

Компиляция динамической библиотеки:

```
g++ -shared -o library.so library.cpp
```

Компиляция вызывающей программы:

```
g++ -o main main.cpp -ldl
```

# Работа с Makefile в Linux (1)


**task_03**

```
all:            main

main:           func.o main.o
                @echo Building all together
                g++ func.o main.o -o main

func.o:         func.cpp func.h
                @echo Building func
                g++ -Wall -c func.cpp

main.o:         main.cpp
                @echo Building main
                g++ -Wall -c main.cpp

clean:
                rm -f *.o
                rm -f ./main
```

# Задание на лабораторную работу

Реализовать модули решения и записи результатов моделирования.

Метод решения:

| Номер варианта | Поддерживаемый формат |
|---|---|
| 1, 4, 7, 10, 13 | Метод Гаусса |
| 2, 5, 8, 11, 14 | Метод Якоби |
| 3, 6, 9, 12, 15 | Метод Гаусса-Зейделя |

Формат записи результатов моделирования:

| Номер варианта | Поддерживаемый формат |
|---|---|
| 1, 4, 7, 10, 13 | PSF |
| 2, 5, 8, 11, 14 | CSV |
| 3, 6, 9, 12, 15 | CSDF |