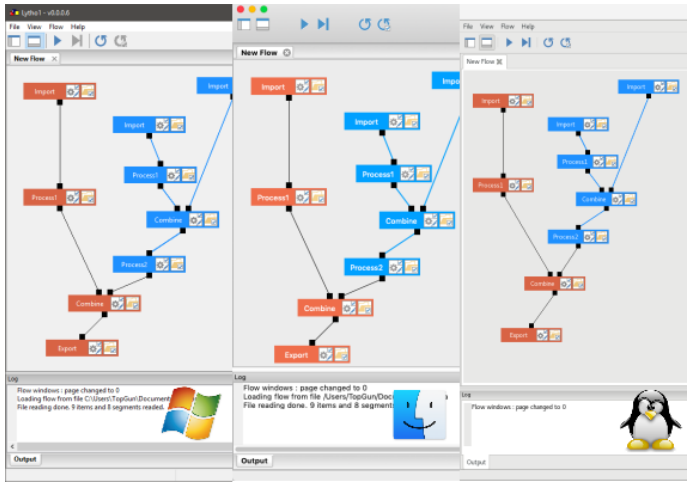
МИЭТ
Национальный исследовательский университет «МИЭТ»
Институт интегральной электроники (группы ЭН-34-35, каф. ПКИМС)

Кроссплатформенная разработка программного обеспечения



Лабораторная работа №3

Статические и динамические библиотеки

Возможности использования кода в различных проектах

Исходный код

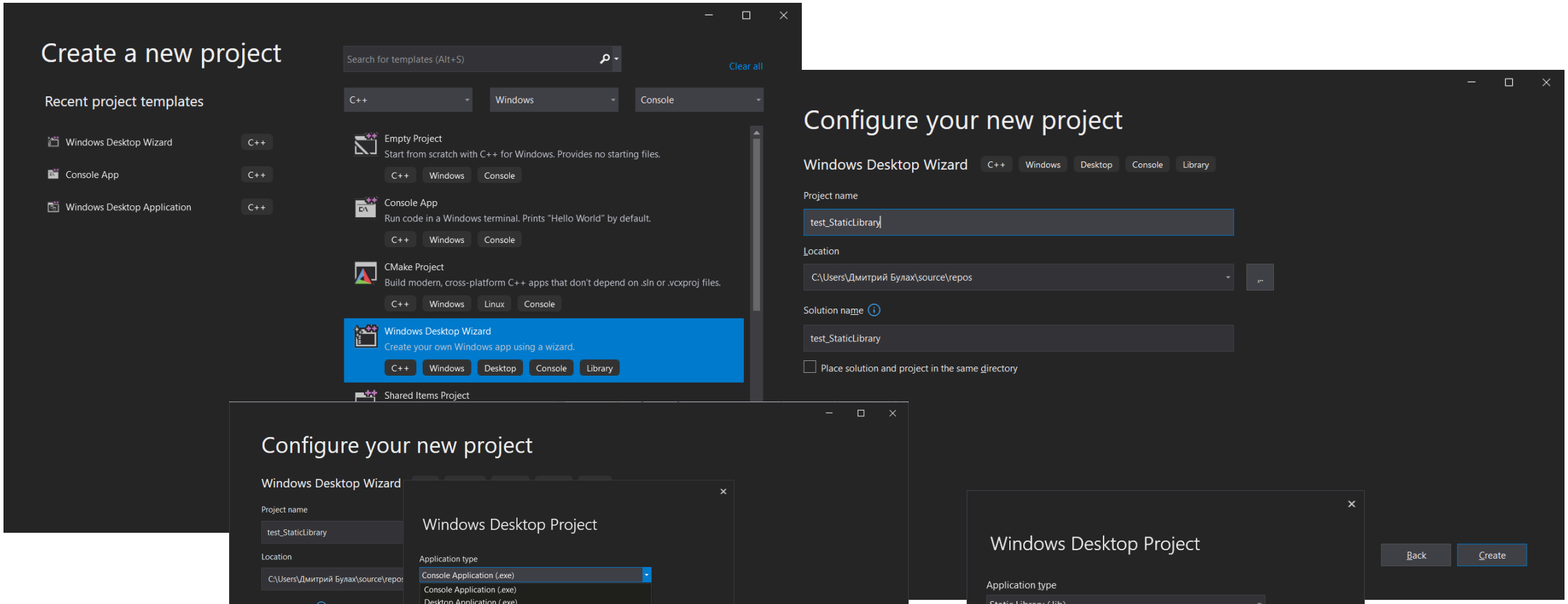
Скомпилированный код

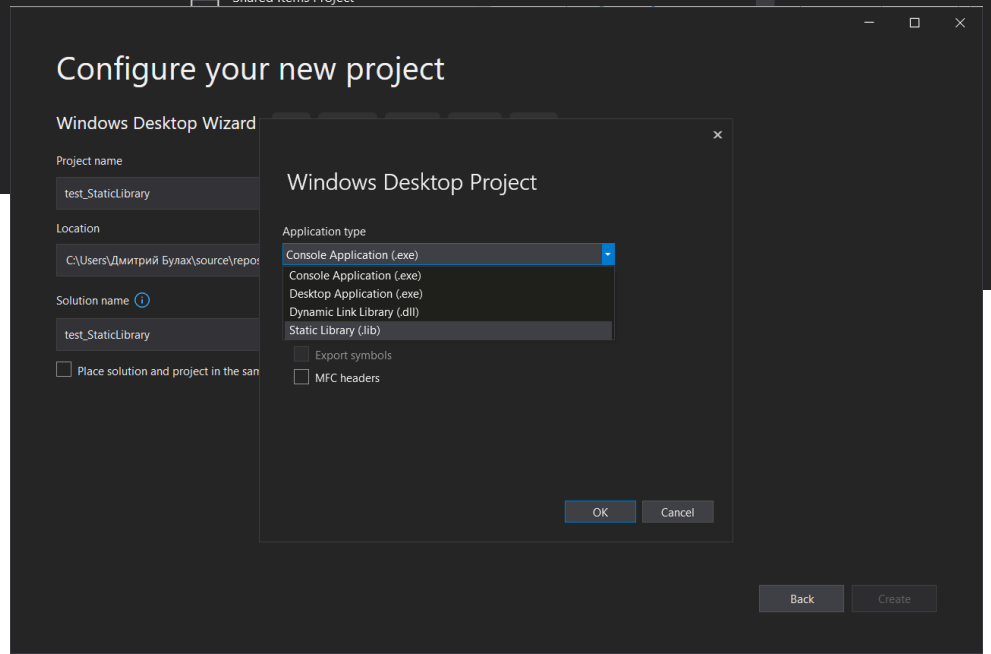
Статическая библиотека

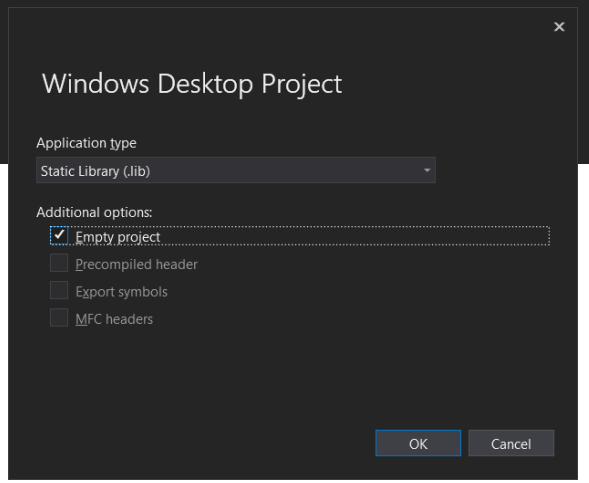
Динамическая библиотека

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;
}
```

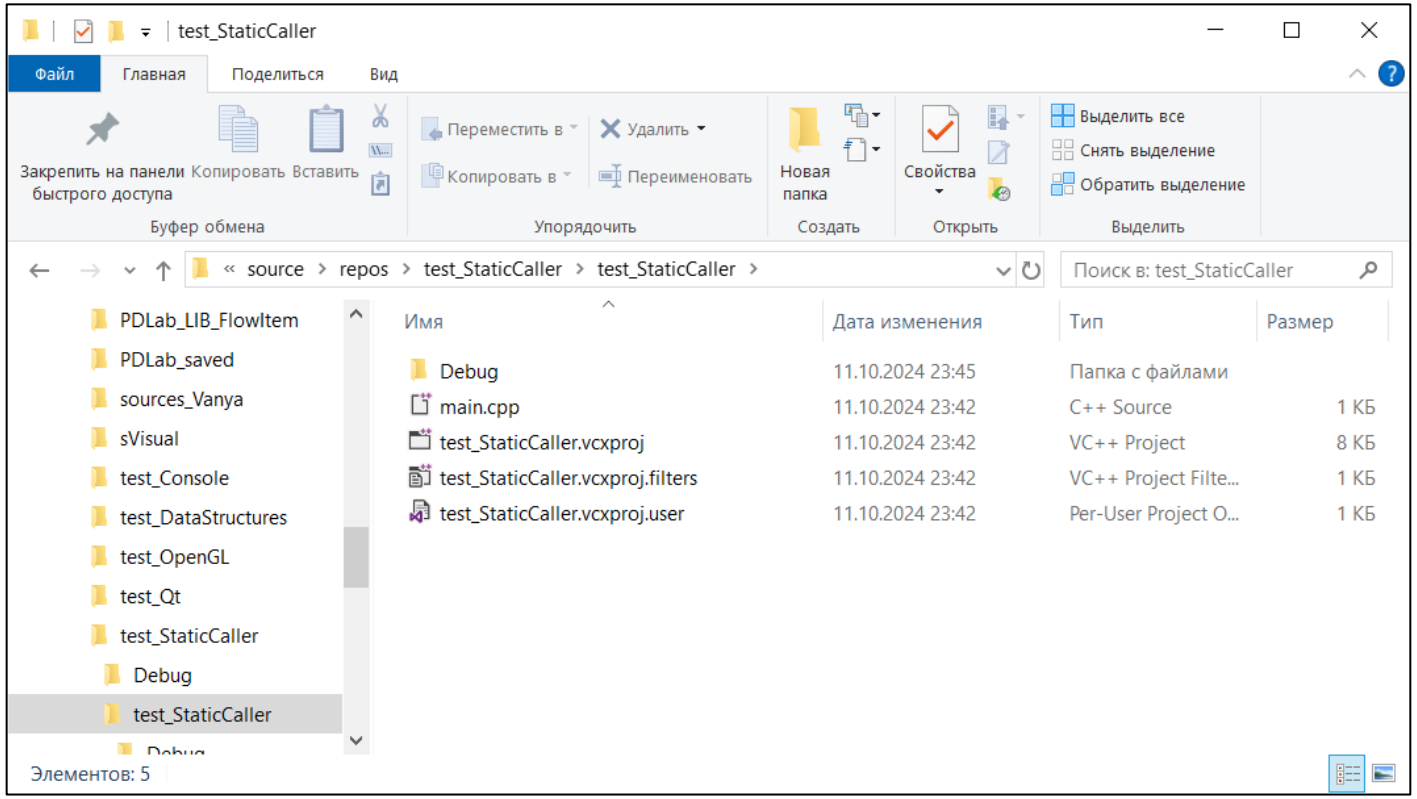
Файл header.hpp:

```
#pragma once

void print_message();
```

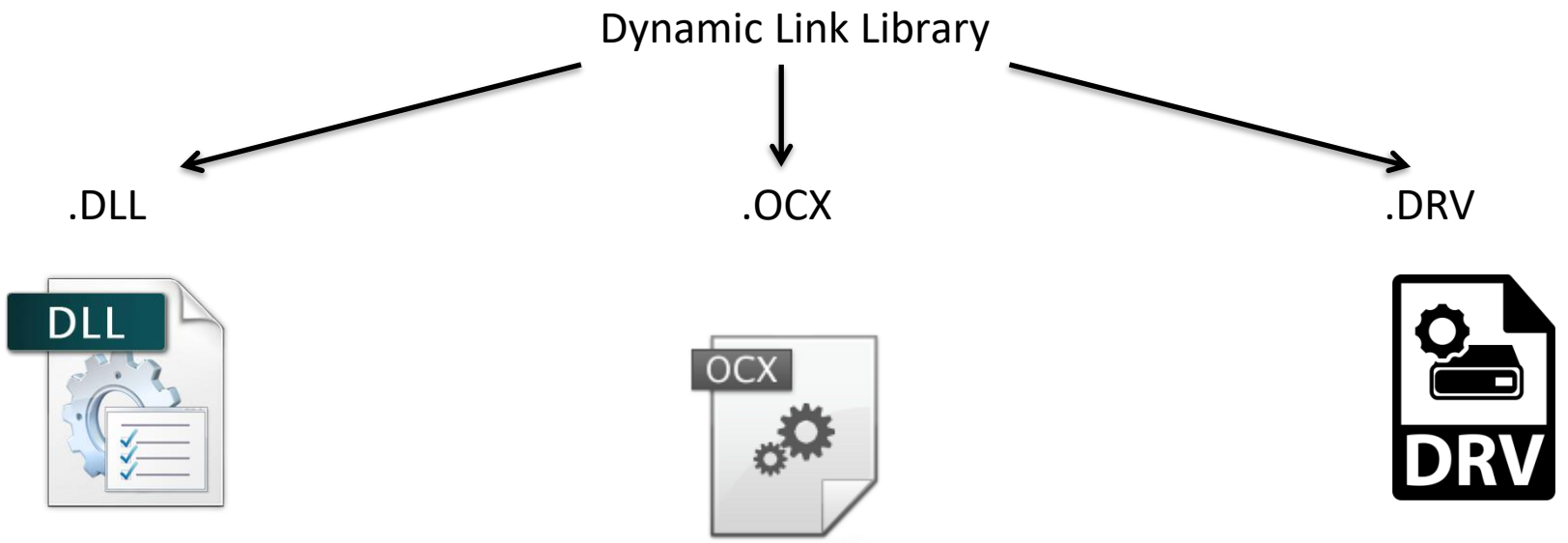
Использование статической библиотеки: написание кода

Код:

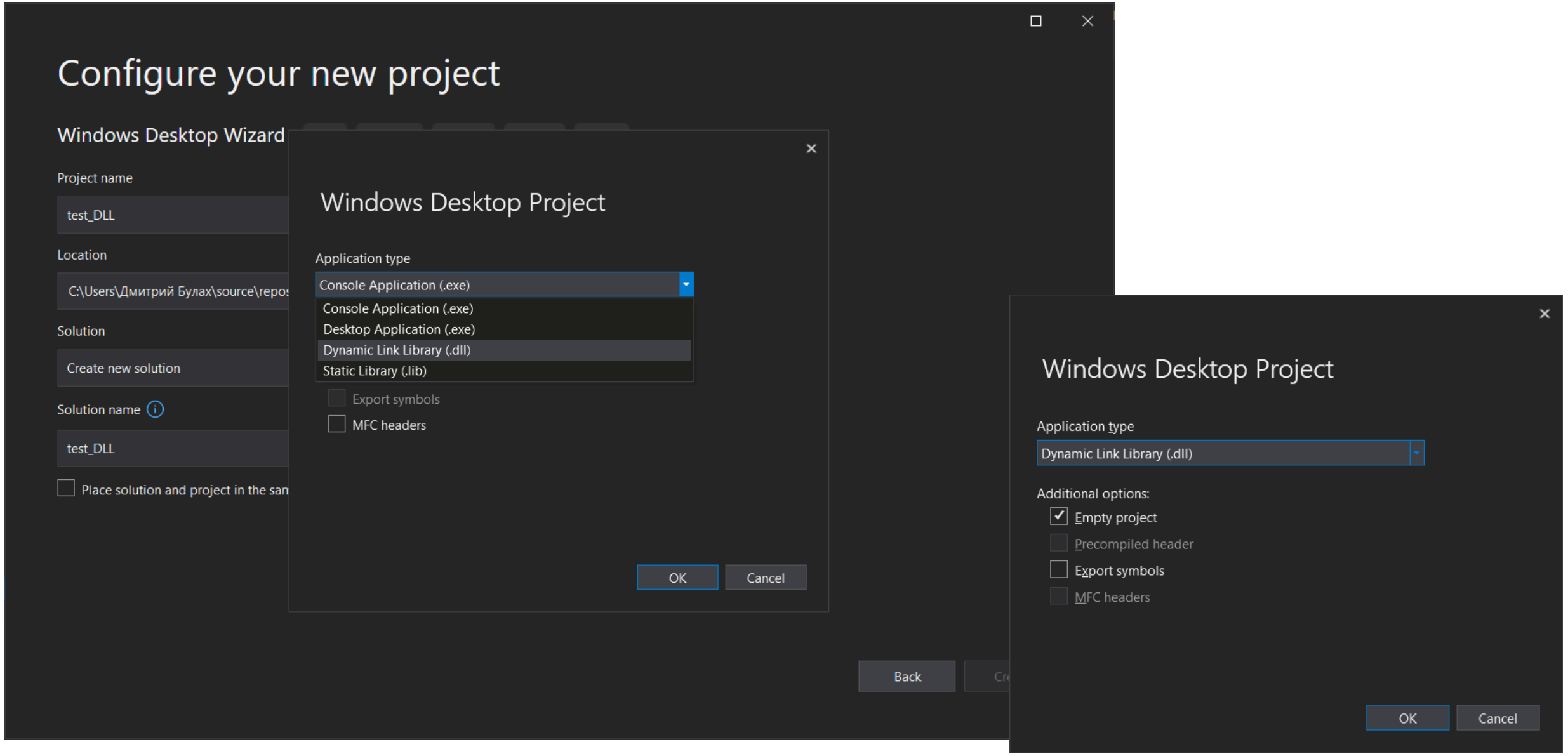
```
#include <iostream>
#include <Header.hpp>

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



Динамически подключаемые библиотеки



Создание динамической библиотеки: создание проекта



Создание динамической библиотеки: код библиотеки

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 ==========
```

Использование динамической библиотеки: код

```
#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;
}
```

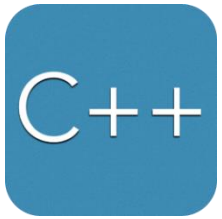
Динамическая библиотека: функция DllMain (1)

```
BOOL __stdcall DllMain(HANDLE hInst,
                       DWORD dwReason,
                       LPVOID IpReserved) {
  BOOL bAllWentWell = TRUE;
  switch (dwReason) {
    case DLL_PROCESS_ATTACH:  // Инициализация процесса.
      break;
    case DLL_THREAD_ATTACH:   // Инициализация потока.
      break;
    case DLL_THREAD_DETACH:   // Очистка структур потока.
      break;
    case DLL_PROCESS_DETACH:  // Очистка структур процесса.
      break;
  }
  return bAllWentWell;
}
```

Динамическая библиотека: функция DllMain (2)

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

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

BOOL __stdcall DllMain(HANDLE hInst, DWORD dwReason, LPVOID IpReserved) {
  BOOL bAllWentWell = TRUE;
  switch (dwReason) {
  case DLL_PROCESS_ATTACH:  // Инициализация процесса.
    std::cout << "DLL_PROCESS_ATTACH" << std::endl;
    break;
  case DLL_THREAD_ATTACH:   // Инициализация потока.
    std::cout << "DLL_THREAD_ATTACH" << std::endl;
    break;
  case DLL_THREAD_DETACH:   // Очистка структур потока.
    std::cout << "DLL_THREAD_DETACH" << std::endl;
    break;
  case DLL_PROCESS_DETACH:  // Очистка структур процесса.
    std::cout << "DLL_PROCESS_DETACH" << std::endl;
    break;
  }
  return bAllWentWell;
}
```

Работа со статическими библиотеками в ОС Linux: код библиотеки



static

Код библиотеки (library.cpp):

```
#include <iostream>

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

Компиляция библиотеки:

```
g++ -c -o library.o library.cpp

ar rcs liblibrary.a library.o
```

Работа со статическими библиотеками в ОС Linux: вызывающий код

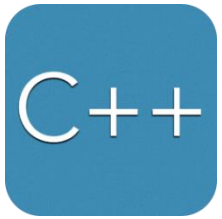
Код библиотеки (main.cpp):

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

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

Компиляция программы:

```
g++  main.cpp
    -I/home/student/KRPO_Lab0x03/static/library
    -L/home/student/KRPO_Lab0x03/static/library –llibrary
    -o main
```

Работа с динамическими библиотеками в ОС Linux: код библиотеки



dynamic

Код библиотеки (library.cpp):

```
#include <iostream>

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

Работа с динамическими библиотеками в ОС Linux: вызывающий код

Код вызывающей программы (main.cpp):

```
#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)
    ...

  hello();

  dlclose(handle);

  return 0;
}
```

Компиляция динамической библиотеки в Linux

Компиляция динамической библиотеки:

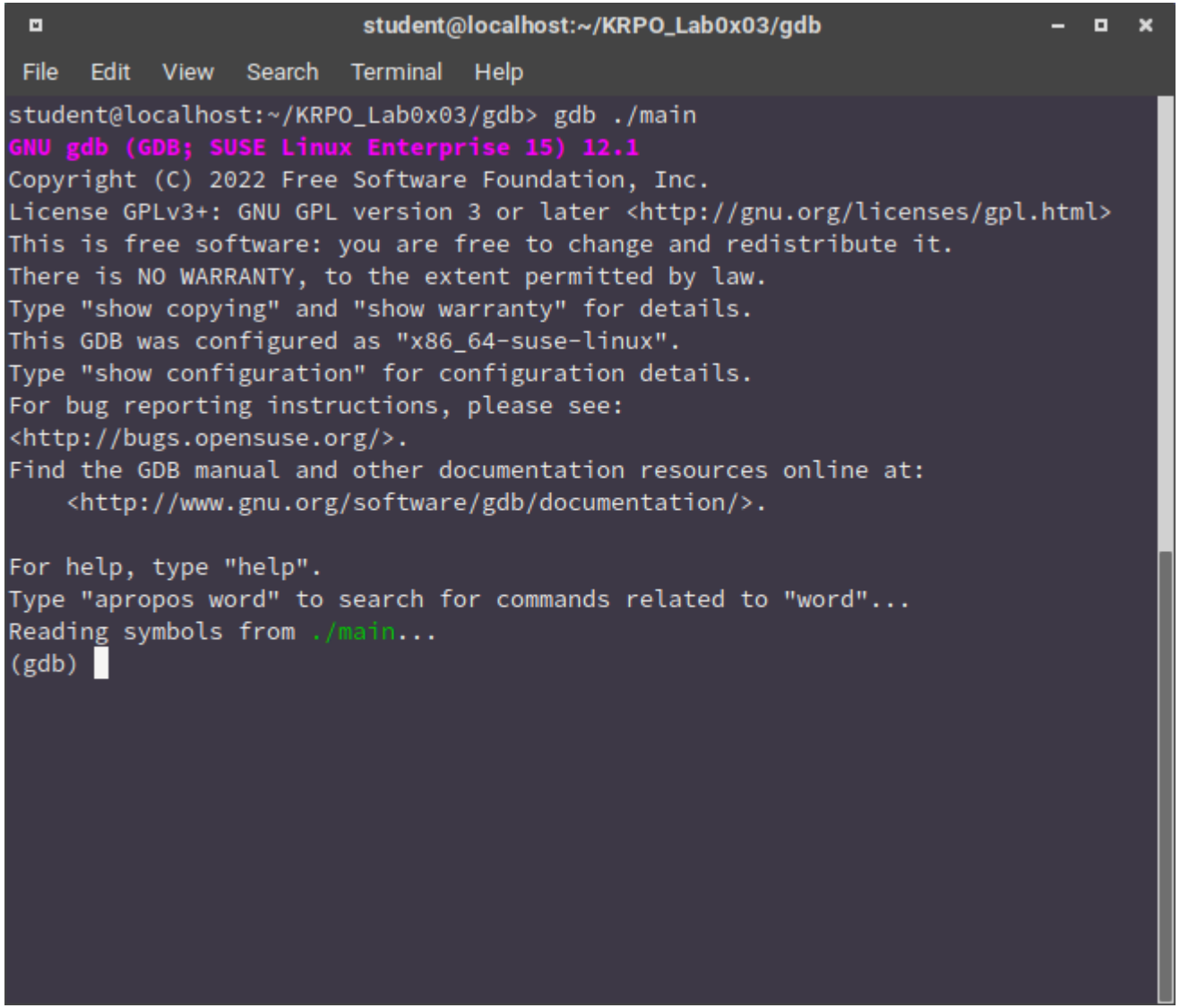
```
g++ -shared -fPIC ./library.cpp -o library.so
```

Компиляция вызывающей программы:

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

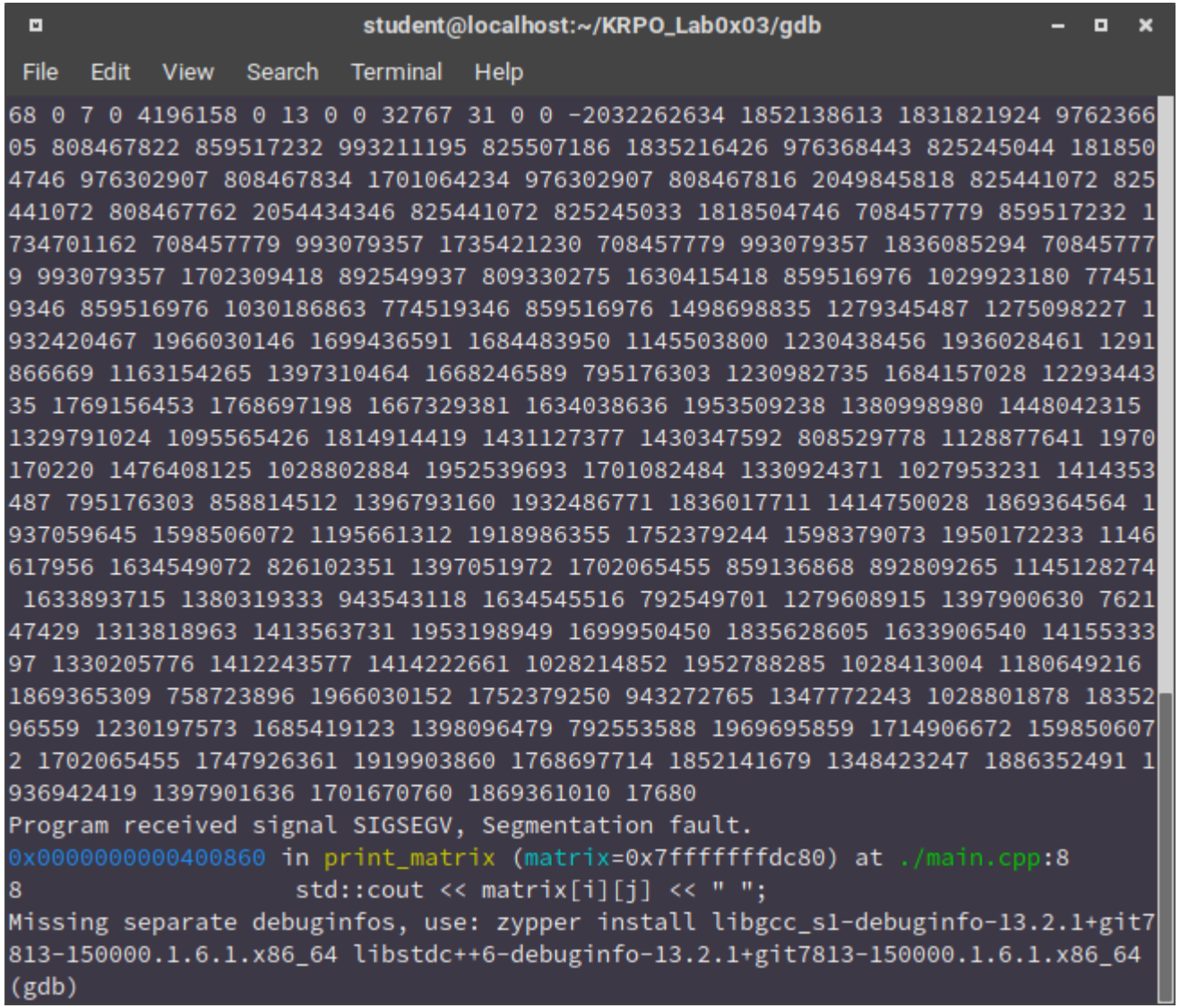
Отладка проектов в ОС Linux (1)

Запуск отладчика:
`gdb ./main`



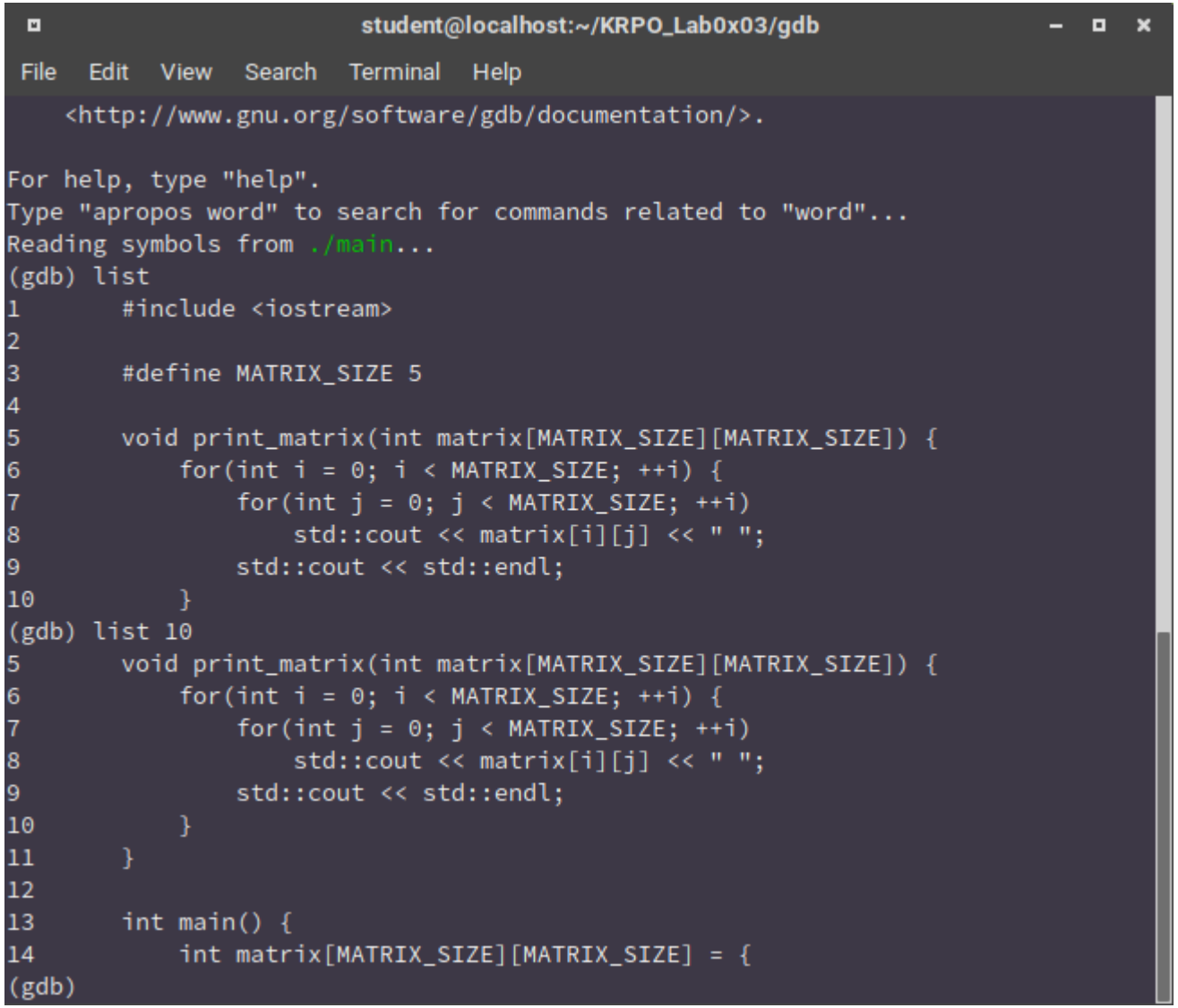
Отладка проектов в ОС Linux (2)

Запуск программы:
`run` или `r`



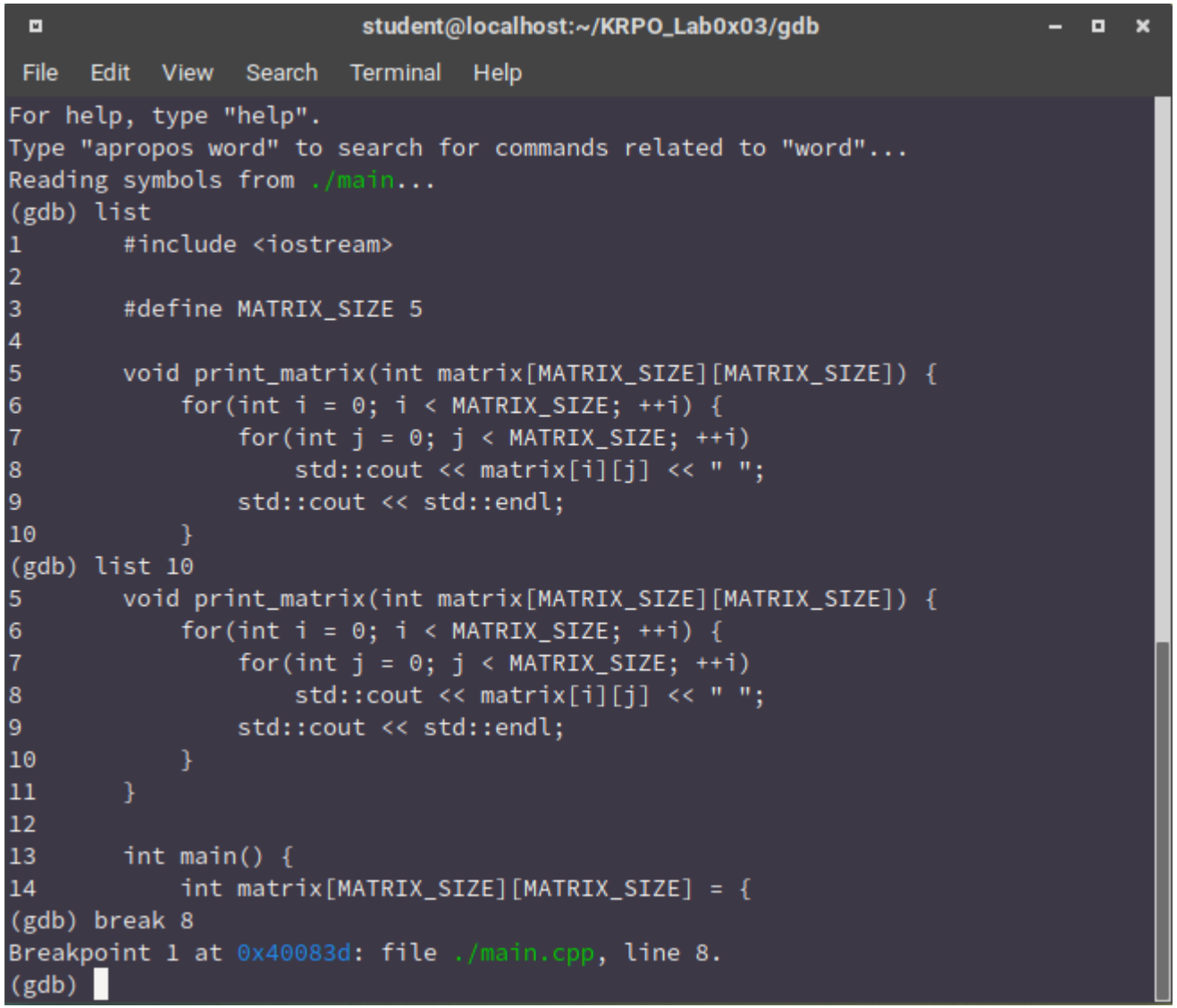
Отладка проектов в ОС Linux (3)

Посмотреть исходник:
`list` или `l`



Отладка проектов в ОС Linux (4)

Установка breakpoint:
`break <N>`
или
`break <name>`



Отладка проектов в ОС Linux (5)

Запуск отладчика:
`gdb ./main`

Запуск программы:
`run или r`

Посмотреть исходник:
`list или l`

Установка breakpoint:
`break <N>`
или
`break <name>`

Шагнуть (step over): next

Провалиться (step into): step

Получение информации: info

```
info breakpoints
info locals
info functions
```

Получение значения переменной: print

```
print i
```