

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
«БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»
Кафедра ИИТ

ЛАБОРАТОРНАЯ РАБОТА №3
По дисциплине ОСиСП за II семестр
«Динамические библиотеки (DLL)»

Выполнил:
Студент 3-го курса
Группы ПО-5
Игнатюк А.А.
Проверил:
Дряпко А.В.

Цель работы: Научиться разрабатывать и использовать динамические библиотеки (DLL) с использованием Qt.

Вариант 7: Игра Сапер.

Задание:

Доработать программу, разработанную в лабораторной работе №1-2, внося следующие изменения:

1. Выбрать 3 вспомогательные функции и вынести их описание и реализацию в динамическую библиотеку helper.dll. В основном приложении осуществить загрузку реализованных функций во время работы программы (at run-time, с использованием объекта QLibrary) и их вызов.
2. Выбрать вспомогательный класс и вынести его описание и реализацию в динамическую библиотеку helper_class.dll. В основном приложении осуществить загрузку реализованного класса во время компиляции (at compile-time).
3. Реализовать окно “О Программе” в виде объекта динамической библиотеки about.dll с указанием автора программы, группы, курса и краткого описания разработанного приложения, осуществить импорт указанной библиотеки и отображение соответствующего окна при выборе меню “О Программе”.
4. Реализовать расширения для приложения, позволяющие изменять оформление пунктов меню (шрифт, размер, начертание и т.д.). Соответствующие изменения должны происходить при выборе специального пункта меню. Создать как минимум три расширения такого типа.

Ход работы:

Задание 1:

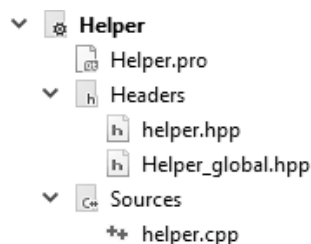


Рисунок 1.1 - Структура проекта.

```
< > [icon] helper.hpp [icon] X <Select Symbol>
1  #ifndef HELPER_HPP
2  #define HELPER_HPP
3
4  #include "Helper_global.hpp"
5
6  class HELPER_EXPORT Helper {
7  public:
8      static std::int64_t f_get_random_number(const std::int64_t& c_From,
9                                              const std::int64_t& c_To);
10
11     static void f_read_line_from_file(const std::string& c_Filename,
12                                     std::string& v_Data);
13
14     static void f_write_to_file(const std::string& c_Filename,
15                               const std::string& c_Data);
16 };
17
18 #endif // HELPER_HPP
19
```

Рисунок 1.2 - Содержимое файла helper.hpp.

```
< > Helper_global.hpp <No Symbols>
1  #ifndef HELPER_GLOBAL_HPP
2  #define HELPER_GLOBAL_HPP
3
4  #include <QtCore/qglobal.h>
5
6  #if defined(HELPER_LIBRARY)
7  #   define HELPER_EXPORT Q_DECL_EXPORT
8  #else
9  #   define HELPER_EXPORT Q_DECL_IMPORT
10 #endif
11
12 #endif // HELPER_GLOBAL_HPP
13
```

Рисунок 1.3 - Содержимое файла Helper_global.hpp.

```
< > helper.cpp <Select Symbol>
1  #include "helper.hpp"
2
3  #include <QRandomGenerator>
4
5  #include <fstream>
6
7  std::int64_t Helper::f_get_random_number(const std::int64_t& c_From,
8  ▼                                     const std::int64_t& c_To) {
9      return c_From + QRandomGenerator::global()->generate() % (c_To - c_From + 1);
10 }
11
12 void Helper::f_read_line_from_file(const std::string& c_Filename,
13 ▼                                std::string& v_Data) {
14     std::ifstream v_Fin(c_Filename);
15
16     if (v_Fin.is_open()) {
17         std::getline(v_Fin, v_Data);
18         v_Fin.close();
19     }
20 }
21
22 void Helper::f_write_to_file(const std::string& c_Filename,
23 ▼                             const std::string& c_Data) {
24     std::ofstream v_Fout(c_Filename);
25
26     if (v_Fout.is_open()) {
27         v_Fout << c_Data;
28         v_Fout.close();
29     }
30 }
31
32 extern "C" HELPER_EXPORT std::int64_t get_random_number(
33 ▼     const std::int64_t& c_From, const std::int64_t& c_To) {
34     return Helper::f_get_random_number(c_From, c_To);
35 }
36
37 extern "C" HELPER_EXPORT void read_line_from_file(const std::string& c_Filename,
38 ▼                                                  std::string& v_Data) {
39     Helper::f_read_line_from_file(c_Filename, v_Data);
40 }
41
42 extern "C" HELPER_EXPORT void write_to_file(const std::string& c_Filename,
43 ▼                                           const std::string& c_Data) {
44     Helper::f_write_to_file(c_Filename, c_Data);
45 }
46
```

Рисунок 1.4 - Содержимое файла helper.cpp.

1. Объявление типов для импортируемых функций

```
16 typedef std::int64_t (*f_get_random_number)(const std::int64_t&,
17                                             const std::int64_t&);
18 typedef void (*f_read_line_from_file)(const std::string&, const std::string&);
19 typedef void (*f_write_to_file)(const std::string&, const std::string&);
```

2. Добавление полей класса для использования сторонней библиотеки

```
50 QLibrary* m_HelperLibraryPtr{nullptr};
51
52 f_get_random_number m_GetRandomNumberPtr{nullptr};
53 f_read_line_from_file m_ReadLineFromFilePtr{nullptr};
54 f_write_to_file m_WriteToFilePtr{nullptr};
```

3. Подключение библиотеки к программе во время выполнения, связывание указателей на функции с библиотечными функциями (изменения в конструкторе)

```
11 m_HelperLibraryPtr = f_load_library("helper.dll");
12
13 ▼ if (m_HelperLibraryPtr == nullptr) {
14     return;
15 }
16
17 m_GetRandomNumberPtr =
18     (f_get_random_number)m_HelperLibraryPtr->resolve("get_random_number");
19 m_ReadLineFromFilePtr =
20     (f_read_line_from_file)m_HelperLibraryPtr->resolve("read_line_from_file");
21 m_WriteToFilePtr =
22     (f_write_to_file)m_HelperLibraryPtr->resolve("write_to_file");
```

4. Освобождение ресурсов после использования библиотеки (изменения в деструкторе)

```
108 ▼ if (m_HelperLibraryPtr->isLoaded()) {
109     m_HelperLibraryPtr->unload();
110 }
111
112 delete m_HelperLibraryPtr;
```

5. Замена ранее используемых функций на новые библиотечные (определения старых функций удаляются)

Было: 187 ▼ while (v_BobmIndexes.size() < m_GameDifficulty) {
188 std::uint64_t v_Index{QRandomGenerator::global()->generate()} % (m_GameButtonsCount + 1)};

Стало: 210 ▼ while (v_BobmIndexes.size() < m_GameDifficulty) {
211 std::int64_t v_Index{m_GetRandomNumberPtr(0, m_GameButtonsCount)};

Было: 139 ▼ void Sapper::f_load_scores() {
140 std::ifstream v_Fin(mc_SCORES_FILENAME);
141
142 ▼ if (v_Fin.is_open()) {
143 v_Fin >> m_EasyBestScore >> m_MediumBestScore >> m_HardBestScore;
144 v_Fin.close();
145 return;
146 }
147
148 f_save_scores();
149 }

Стало: 164 ▼ void Sapper::f_load_scores() {
165 std::string v_ScoresLine{};
166 m_ReadLineFromFilePtr(mc_SCORES_FILENAME, v_ScoresLine);
167
168 std::stringstream v_Stream{v_ScoresLine};
169 v_Stream >> m_EasyBestScore >> m_MediumBestScore >> m_HardBestScore;
170
171 f_save_scores();
172 }

Было: 151 ▼ void Sapper::f_save_scores() {
152 std::ofstream v_Fout(mc_SCORES_FILENAME);
153
154 ▼ if (v_Fout.is_open()) {
155 v_Fout << m_EasyBestScore << " " << m_MediumBestScore << " " << m_HardBestScore;
156 v_Fout.close();
157 }
158 }

Стало: 174 ▼ void Sapper::f_save_scores() {
175 m_WriteToFilePtr(mc_SCORES_FILENAME, (QString::number(m_EasyBestScore) + " " +
176 QString::number(m_MediumBestScore) +
177 " " + QString::number(m_HardBestScore))
178 .toStdString());
179 }
180

Рисунок 1.5 - Изменения в проекте из лабораторной работы №1-2.

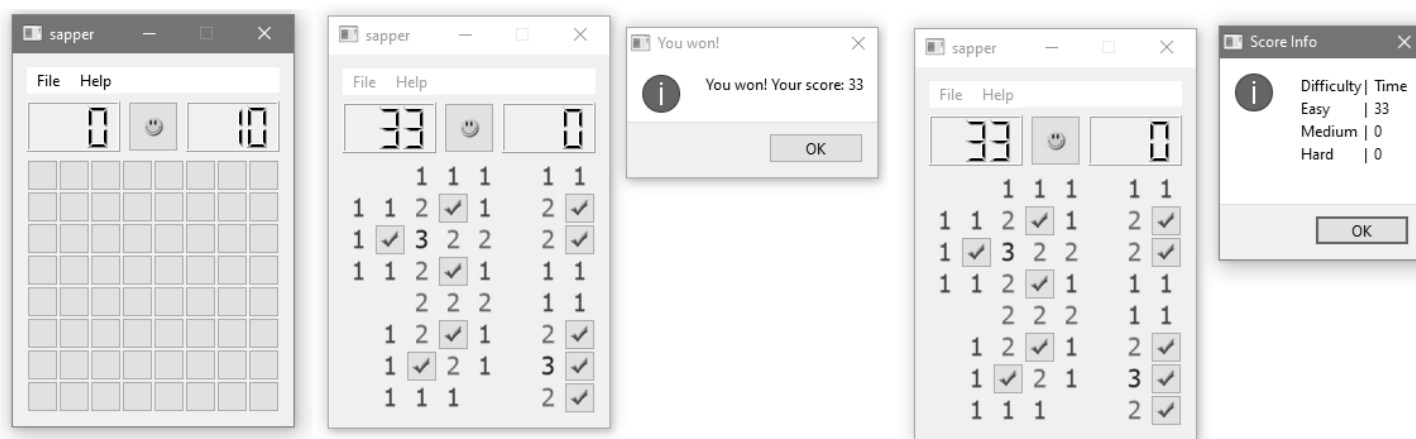


Рисунок 1.6 - Результат работы программы с использованием библиотеки.

Задание 2:

В качестве вспомогательного класса был выбран класс PushButton из лабораторной работы №1-2. Его функционал был перенесен в библиотеку и удален из основного проекта.

В основной проект были добавлены заголовочные файлы библиотеки для использования ее функционала в коде во время компиляции.

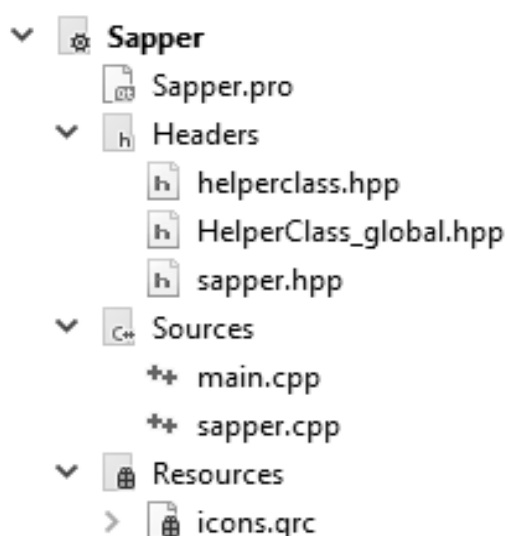


Рисунок 2.1 - Обновленная структура основного проекта.

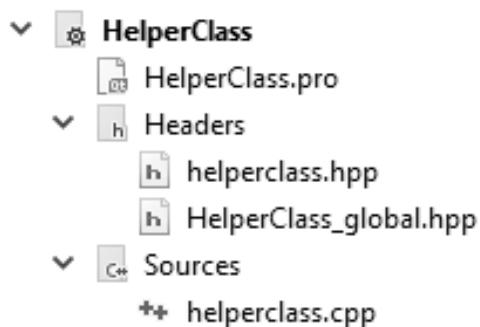


Рисунок 2.2 - Структура проекта библиотеки.

```

< > h helperclass.hpp ▾ | X | <Select Symbol>
1  #ifndef HELPERCLASS_HPP
2  #define HELPERCLASS_HPP
3
4  #include "HelperClass_global.hpp"
5
6  #include <QPushButton>
7
8  ▾ class HELPERCLASS_EXPORT QPushButton : public QPushButton {
9      Q_OBJECT
10
11     public:
12         QPushButton(const std::uint64_t& c_Index = std::uint64_t{},
13                     QPushButton* const c_PushButtonPtr = nullptr,
14                     const QString& c_Text = QString{});
15
16         static constexpr std::uint64_t c_BUTTON_SIZE{25};
17
18         std::uint64_t f_get_index() const;
19         std::uint64_t f_get_icon_index() const;
20
21         bool f_is_flag();
22         void f_clear();
23
24     private:
25         std::uint64_t m_Index;
26         std::uint64_t m_IconIndex{};
27
28         void mousePressEvent(QMouseEvent* const event);
29
30     signals:
31         void left_clicked(std::uint64_t);
32         void right_clicked(std::uint64_t);
33 };
34
35 #endif // HELPERCLASS_HPP
36

```

Рисунок 2.3 - Содержимое файла helperclass.hpp.




```

< > h HelperClass_global.hpp ▾ | X | <No Symbols>
1  #ifndef HELPERCLASS_GLOBAL_HPP
2  #define HELPERCLASS_GLOBAL_HPP
3
4  #include <QtCore/qglobal.h>
5
6  #if defined(HELPERCLASS_LIBRARY)
7  #   define HELPERCLASS_EXPORT Q_DECL_EXPORT
8  #else
9  #   define HELPERCLASS_EXPORT Q_DECL_IMPORT
10 #endif
11
12 #endif // HELPERCLASS_GLOBAL_HPP
13

```

Рисунок 2.4 - Содержимое файла HelperClass_global.hpp.

```

< >  ** helperclass.cpp  |  | <Select Symbol>
1  #include "helperclass.hpp"
2
3  #include <QMouseEvent>
4
5  QPushButton::PushButton(const std::uint64_t& c_Index,
6                          QPushButton* const c_PushButtonPtr,
7                          const QString& c_Text)
8  ▼ : QPushButton(c_PushButtonPtr), m_Index(c_Index) {
9      setMinimumSize(PushButton::c_BUTTON_SIZE, PushButton::c_BUTTON_SIZE);
10     setMaximumSize(PushButton::c_BUTTON_SIZE, PushButton::c_BUTTON_SIZE);
11     setText(c_Text);
12 }
13
14 std::uint64_t QPushButton::f_get_index() const { return m_Index; }
15
16 std::uint64_t QPushButton::f_get_icon_index() const { return m_IconIndex; }
17
18 bool QPushButton::f_is_flag() { return m_IconIndex == 1; }
19
20 ▼ void QPushButton::f_clear() {
21     m_IconIndex = std::uint64_t{};
22     setFlat(false);
23     setIcon(QIcon());
24 }
25
26 ▼ void QPushButton::mousePressEvent(QMouseEvent* const event) {
27     switch (event->button()) {
28     ▼ case Qt::RightButton: {
29     ▼     if (isFlat()) {
30         return;
31     }
32
33     ++m_IconIndex;
34
35     ▼ if (m_IconIndex > 2) {
36         m_IconIndex -= 3;
37     }
38
39     emit right_clicked(m_Index);
40     break;
41 }
42 ▼ case Qt::LeftButton: {
43     ▼ if (m_IconIndex != 0) {
44         return;
45     }
46
47     emit left_clicked(m_Index);
48     break;
49 }
50     default: {}
51 }
52 }
53

```

Рисунок 2.5 - Содержимое файла helperclass.cpp.

1. Указание пути к библиотеке в настройках основного проекта

```
22 LIBS += "C:/Users/User/Desktop/OS/lab_3/build-Sapper-Desktop_Qt_6_1_2_MinGW_64_bit-Debug/debug/helper_class.dll"  
23
```

2. Подключение библиотечного заголовочного файла в файлы исходного кода, где используется вспомогательный класс

```
14 #include "helperclass.hpp"
```

3. Так как библиотечный класс имеет тот же идентификатор, то изменения в коде не нужны

Рисунок 2.6 - Изменения в проекте из лабораторной работы №1-2.

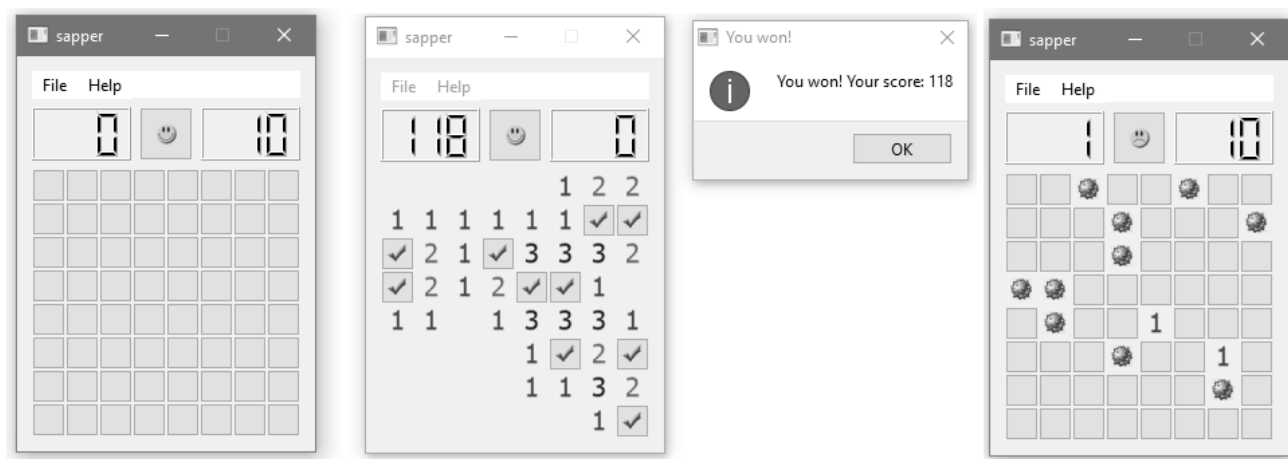


Рисунок 2.7 - Результат работы программы с использованием библиотеки.

Задание 3:

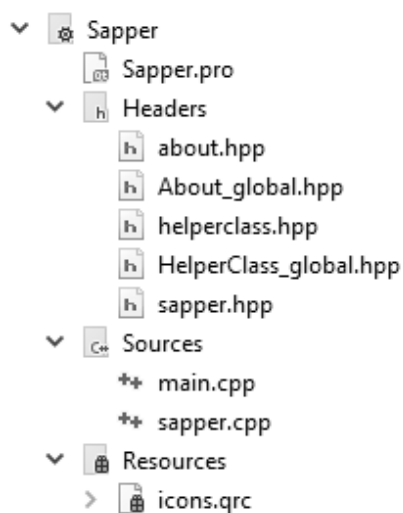


Рисунок 3.1 - Обновленная структура основного проекта.

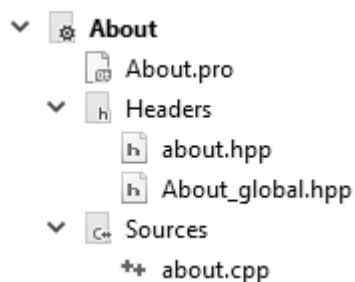


Рисунок 3.2 - Структура проекта библиотеки.


```

< > [h] about.hpp ▾ | X | <Select Symbol>
1  #ifndef ABOUT_HPP
2  #define ABOUT_HPP
3
4  #include "About_global.hpp"
5
6  #include <QMessageBox>
7
8  class ABOUT_EXPORT About {
9      public:
10     About();
11
12     void f_show() const;
13
14     private:
15     QMessageBox* m_AboutBoxPtr;
16 };
17
18 #endif // ABOUT_HPP
19

```

Рисунок 3.3 - Содержимое файла about.hpp.

```

< > [h] About_global.hpp ▾ | X | <No Symbols>
1  #ifndef ABOUT_GLOBAL_HPP
2  #define ABOUT_GLOBAL_HPP
3
4  #include <QtCore/qglobal.h>
5
6  #if defined(About_LIBRARY)
7  # define ABOUT_EXPORT Q_DECL_EXPORT
8  #else
9  # define ABOUT_EXPORT Q_DECL_IMPORT
10 #endif
11
12 #endif // ABOUT_GLOBAL_HPP
13

```

Рисунок 3.4 - Содержимое файла About_global.hpp.

```

< > [c++] about.cpp ▾ | X | <No Symbols>
1  #include "about.hpp"
2
3  About::About()
4      : m_AboutBoxPtr(new QMessageBox(
5      QMessageBox::Information, "About",
6      "Sapper v1.0\nAuthor: Andy\nCourse: 3\nSpeciality: S5",
7      QMessageBox::Ok)) {}
8
9  void About::f_show() const { m_AboutBoxPtr->show(); }
10

```

Рисунок 3.5 - Содержимое файла about.cpp.

1. Указание пути к библиотеке в настройках основного проекта

```
25 LIBS += "C:/Users/User/Desktop/OS/lab_3/build-Sapper-Desktop_Qt_6_1_2_MinGW_64_bit-Debug/debug/about.dll"
26
```

2. Подключение библиотечного заголовочного файла

```
14 #include "about.hpp"
```

3. Изменение типа старого указателя на новый библиотечный

```
39 About* m>AboutInfoPtr{nullptr};
```

4. Изменение инициализации

```
40 m>AboutInfoPtr = new About{};
```

5. Изменение в коде при использовании

Было: 317 void Sapper::sf_about() { m>AboutInfoPtr->show(); }

Стало: 349 void Sapper::sf_about() { m>AboutInfoPtr->f_show(); }

Рисунок 3.6 - Изменения в проекте из лабораторной работы №1-2.

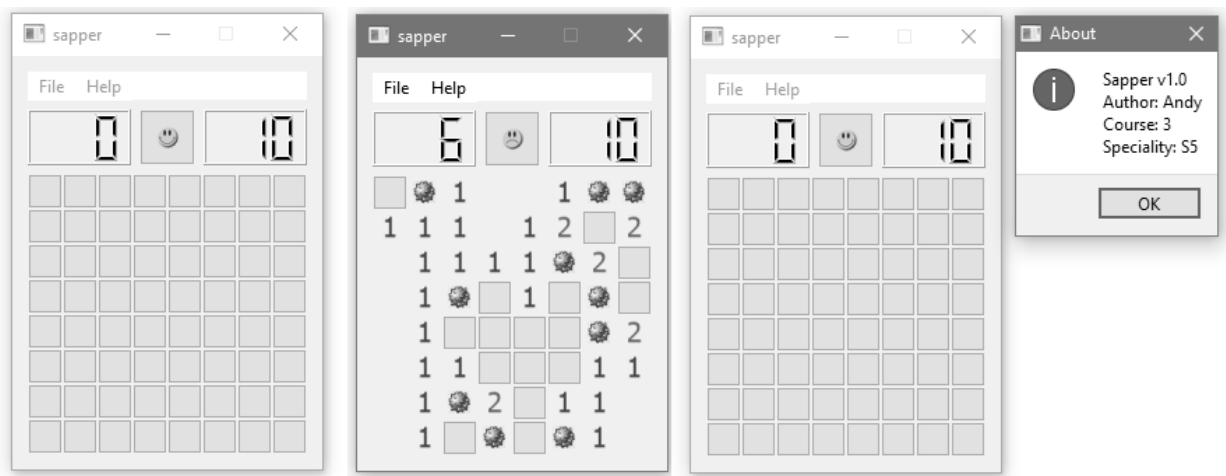


Рисунок 3.7 - Результат работы программы с использованием библиотеки.

Задание 4:

Приложение было расширено следующим функционалом:

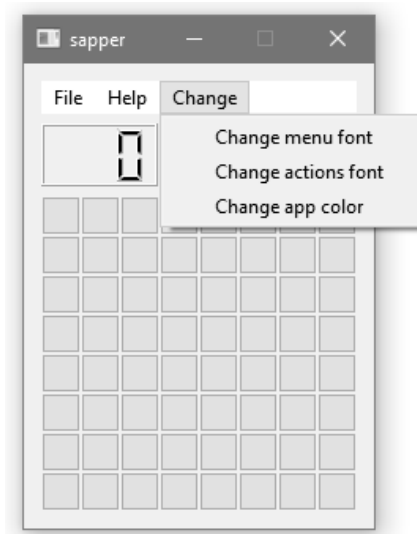


Рисунок 4.1 - Интерфейс приложения с новым функционалом.

1. Были объявлены новые методы основного класса

```
110 void sf_change_menu_font();
111 void sf_change_actions_font();
112 void sf_change_color();
```

2. Добавлено описание создания меню с действиями в интерфейсе

```
71 m_GameChangeMenuPtr = m_GameMenuBarPtr->addMenu("Change");
72 m_GameChangeMenuPtr->addAction("Change menu font", this,
73                                SLOT(sf_change_menu_font()));
74 m_GameChangeMenuPtr->addAction("Change actions font", this,
75                                SLOT(sf_change_actions_font()));
76 m_GameChangeMenuPtr->addAction("Change app color", this,
77                                SLOT(sf_change_color()));
78
```

3. Добавлено освобождение ресурсов для нового меню

```
112 delete m_GameChangeMenuPtr;
```

4. Добавлены определения новых методов

```
363 void Sapper::sf_change_menu_font() {
364     bool v_Ok{false};
365
366     QFont v_Font{QFontDialog::getFont(&v_Ok)};
367
368     if (v_Ok) {
369         m_GameMenuBarPtr->setFont(v_Font);
370     }
371 }
372
373 void Sapper::sf_change_actions_font() {
374     bool v_Ok{false};
375
376     QFont v_Font{QFontDialog::getFont(&v_Ok)};
377
378     if (v_Ok) {
379         m_GameFileMenuPtr->setFont(v_Font);
380         m_GameDifficultyMenuPtr->setFont(v_Font);
381         m_GameHelpMenuPtr->setFont(v_Font);
382         m_GameChangeMenuPtr->setFont(v_Font);
383     }
384 }
385
386 void Sapper::sf_change_color() {
387     QColor v_Color{QColorDialog::getColor()};
388
389     m_GameMenuBarPtr->setStyleSheet("QMenuBar { background-color : rgb(" +
390                                     QString::number(v_Color.red()) + ", " +
391                                     QString::number(v_Color.green()) + ", " +
392                                     QString::number(v_Color.blue()) + "); }");
393
394     for (auto& v_Cell : m_GameMap) {
395         v_Cell.second->setStyleSheet("QPushButton { background-color : rgb(" +
396                                     QString::number(v_Color.red()) + ", " +
397                                     QString::number(v_Color.green()) + ", " +
398                                     QString::number(v_Color.blue()) + "); }");
399     }
400 }
401
```

Рисунок 4.2 - Изменения в проекте из лабораторной работы №1-2.

Теперь у пользователя есть возможность изменить:

1. Шрифт областей меню.
2. Шрифт действий в областях меню.
3. Цвет меню и игрового поля.

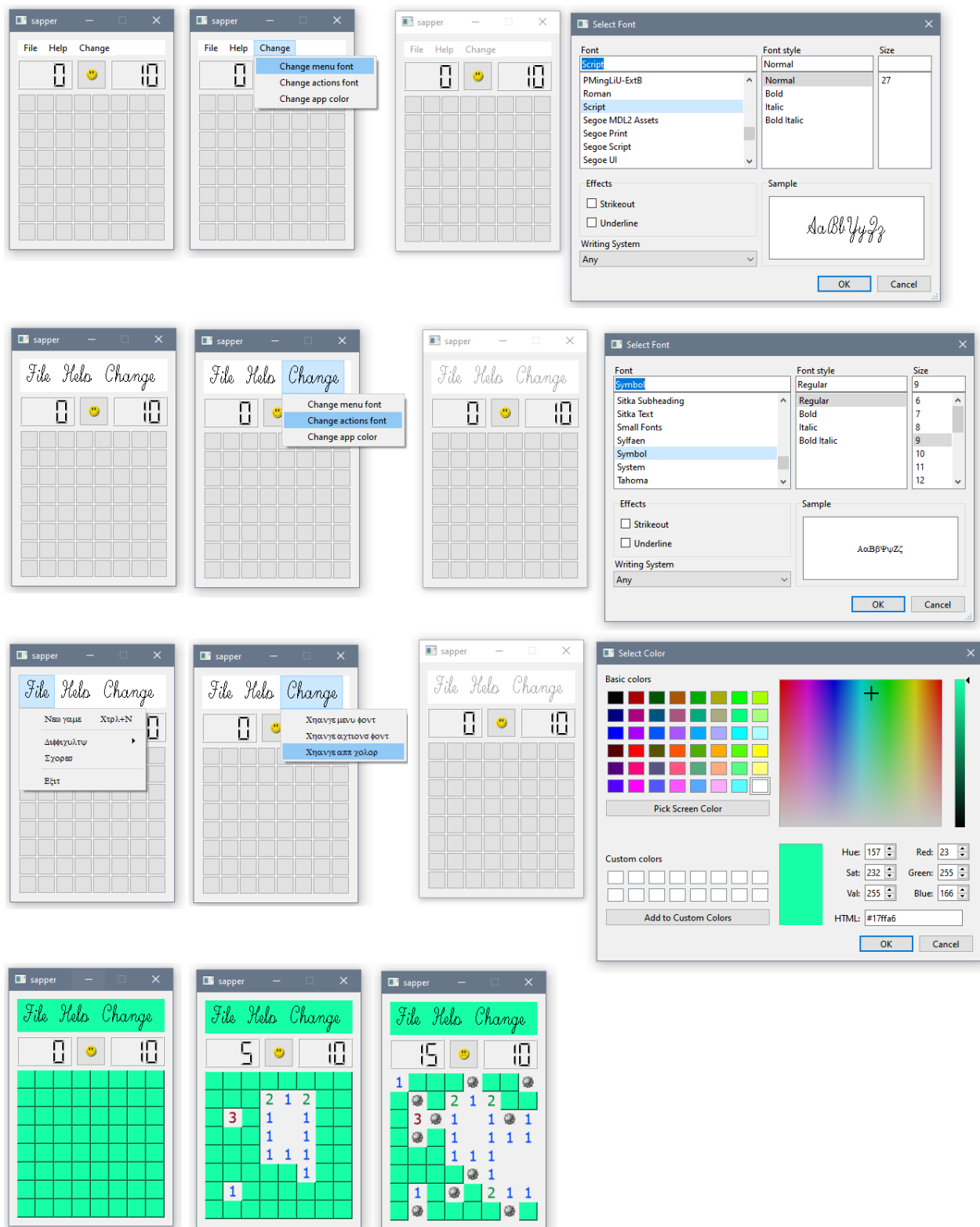


Рисунок 4.3 - Результат работы программы с использованием библиотеки.

Вывод: Научился разрабатывать и использовать динамические библиотеки (DLL) с использованием Qt.