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

ЛАБОРАТОРНАЯ РАБОТА №1-2
По дисциплине ОСиСП за II семестр
«Основы QT»

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

Брест 2021

Цель работы: Приобрести практические навыки проектирования и разработки приложений с графическим пользовательским интерфейсом в ОС Windows средствами Qt.

Задание:

1) Выбрать тему из перечисленных ниже или предложить свою (тематика - игры, системные программы и утилиты для ОС Windows);

2) Вписать свою фамилию напротив выбранной темы в файле;

3) Разработать программу с графическим пользовательским интерфейсом, реализующую указанный функционал, с использованием фреймворка Qt.

Вариант 7: Игра Сапер. Реализовать игру только для одного размера игрового поля (8x8 клеток) с фиксированным количеством случайно расставленных мин (10 штук).

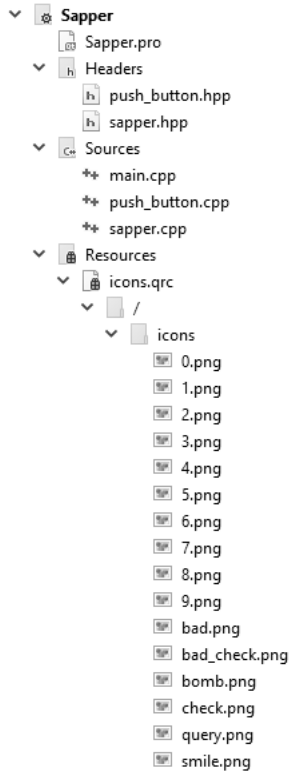

Рисунок 1 - Структура проекта.

Sapper.pro

```
QT += core gui widgets

greaterThan(QT_MAJOR_VERSION, 5): QT += widgets

TARGET = sapper
TEMPLATE = app

CONFIG += c++17

SOURCES += \
        main.cpp \
        push_button.cpp \
        sapper.cpp

HEADERS += \
        push_button.hpp \
        sapper.hpp

RESOURCES += \
    icons.qrc
```

Рисунок 2 - Содержимое файла Sapper.pro.

push_button.hpp <Select Symbol>

```cpp
#pragma once

#ifndef PUSH_BUTTON_HPP
#define PUSH_BUTTON_HPP

#include <QPushButton>

class PushButton : public QPushButton {
  Q_OBJECT

 public:
  PushButton(const std::uint64_t& c_Index = std::uint64_t{},
    QPushButton* const c_PushButtonPtr = nullptr, const QString& c_Text = QString{});

  static constexpr std::uint64_t c_BUTTON_SIZE{25};

  std::uint64_t f_get_index() const;
  std::uint64_t f_get_icon_index() const;

  void f_set_index(const std::uint64_t& c_Index);

  bool f_is_flag();
  void f_clear();

 private:
  std::uint64_t m_Index{};
  std::uint64_t m_IconIndex{};

  void mousePressEvent(QMouseEvent* const event);

 signals:
  void left_clicked(const std::uint64_t&);
  void right_clicked(const std::uint64_t&);
  void middle_clicked(const std::uint64_t&);
};

#endif  // PUSH_BUTTON_HPP
```

Рисунок 3 - Содержимое файла push_button.hpp.

main.cpp <No Symbols>

```cpp
#include <QApplication>

#include "sapper.hpp"

int main(int argc, char *argv[]) {
  QApplication v_App(argc, argv);

  Sapper v_Sapper;
  v_Sapper.show();

  return v_App.exec();
}
```

Рисунок 4 - Содержимое файла main.cpp.

sapper.hpp | <Select Symbol>

```cpp
#pragma once

#ifndef SAPPER_HPP
#define SAPPER_HPP

#include <QMainWindow>
#include <QMenuBar>
#include <QMessageBox>

#include <qlayout.h>
#include <qlcdnumber.h>

#include "push_button.hpp"

class Sapper : public QWidget {
  Q_OBJECT

 public:
  Sapper(QWidget* const c_QWidgetPtr = nullptr);
  ~Sapper();

 private:
  enum BombsCount {
    EASY = 10,
    MEDIUM = 25,
    HARD = 70
  };

  QLCDNumber* m_TimeInfoPtr{nullptr};
  QLCDNumber* m_BombsInfoPtr{nullptr};

  QPushButton* m_RestartButtonPtr{nullptr};

  QTimer* m_GameTimerPtr{nullptr};

  QMessageBox* m_AboutInfoPtr{nullptr};
  QMessageBox* m_WinnerInfoPtr{nullptr};
  QMessageBox* m_ScoresInfoPtr{nullptr};

  QMenuBar* m_GameMenuBarPtr{nullptr};
  QMenu* m_GameFileMenuPtr{nullptr};
  QMenu* m_GameDifficultyMenuPtr{nullptr};

  QGridLayout* m_GridLayoutPtr{nullptr};
  QHBoxLayout* m_HorizontLayoutPtr{nullptr};
  QVBoxLayout* m_VerticalLayoutPtr{nullptr};

  bool m_GameOver{false};

  BombsCount m_GameDifficulty{BombsCount::EASY};

  std::uint64_t m_CheckedFieldsCount{m_GameDifficulty};
  std::uint64_t m_CheckedFields3x3Count{};
  std::uint64_t m_VisibleCellsCount{};

  static constexpr std::uint64_t c_ROWS_COUNT{8};

  std::uint64_t m_RowsCount{c_ROWS_COUNT};
  std::uint64_t m_GameButtonsCount{m_RowsCount * m_RowsCount};

  std::uint64_t m_EasyBestScore{}, m_MediumBestScore{}, m_HardBestScore{};

  const std::string mc_SCORES_FILENAME{"scores.txt"};

  QMainWindow m_MainWindow;

  std::vector<std::uint64_t> m_FieldValuesArray;
  std::map<std::uint64_t, std::unique_ptr<PushButton>> m_GameMap;
```

Рисунок 5 - Содержимое файла sapper.hpp.

push_button.cpp | <Select Symbol>

```cpp
#include "push_button.hpp"

#include <QMouseEvent>

PushButton::PushButton(const std::uint64_t& c_Index,
  QPushButton* const c_PushButtonPtr, const QString& c_Text)
    : QPushButton(c_PushButtonPtr), m_Index(c_Index) {
  setMinimumSize(PushButton::c_BUTTON_SIZE, PushButton::c_BUTTON_SIZE);
  setMaximumSize(PushButton::c_BUTTON_SIZE, PushButton::c_BUTTON_SIZE);
  setText(c_Text);
}

std::uint64_t PushButton::f_get_index() const { return m_Index; }

std::uint64_t PushButton::f_get_icon_index() const { return m_IconIndex; }

void PushButton::f_set_index(const std::uint64_t& c_Index) { m_Index = c_Index; }

bool PushButton::f_is_flag() { return m_IconIndex == 1; }

void PushButton::f_clear() {
  m_IconIndex = std::uint64_t{};

  setFlat(false);
  setIcon(QIcon());
}

void PushButton::mousePressEvent(QMouseEvent* const event) {
  switch (event->button()) {
    case Qt::RightButton: {
      m_IconIndex++;

      if (m_IconIndex > 2) {
        m_IconIndex -= 3;
      }

      if (!isFlat()) {
        emit right_clicked(m_Index);
      }

      break;
    }
    case Qt::MiddleButton: {
      emit middle_clicked(m_Index);
      break;
    }
    case Qt::LeftButton: {
      if (m_IconIndex == 0) {
        emit left_clicked(m_Index);
      }

      break;
    }
    default: {
    }
  }
}
```

Рисунок 6 - Содержимое файла push_button.cpp.

sapper.cpp | <Select Symbol>

```cpp
#include "sapper.hpp"

#include <QRandomGenerator>
#include <QTimer>

#include <fstream>
#include <set>

Sapper::Sapper(QWidget* const c_QWidgetPtr)
  : QWidget(c_QWidgetPtr, Qt::WindowMinimizeButtonHint | Qt::WindowCloseButtonHint) {
  constexpr std::uint64_t c_INFO_FIELD_LENGTH{3}, c_INFO_FIELD_SIZE{40};

  m_TimeInfoPtr = new QLCDNumber(c_INFO_FIELD_LENGTH);
  m_TimeInfoPtr->setMinimumHeight(c_INFO_FIELD_SIZE);
  m_TimeInfoPtr->display(0);

  m_BombsInfoPtr = new QLCDNumber(c_INFO_FIELD_LENGTH);
  m_BombsInfoPtr->setMinimumHeight(c_INFO_FIELD_SIZE);
  m_BombsInfoPtr->display(static_cast<int>(m_GameDifficulty));

  m_RestartButtonPtr = new QPushButton();
  m_RestartButtonPtr->setMinimumSize(c_INFO_FIELD_SIZE, c_INFO_FIELD_SIZE);
  m_RestartButtonPtr->setMaximumSize(c_INFO_FIELD_SIZE, c_INFO_FIELD_SIZE);
  connect(m_RestartButtonPtr, SIGNAL(clicked()), SLOT(sf_create_game()));

  m_GameTimerPtr = new QTimer(this);
  connect(m_GameTimerPtr, SIGNAL(timeout()), SLOT(sf_add_time()));

  m_AboutInfoPtr = new QMessageBox(QMessageBox::Information, "About", "Sapper v1.0", QMessageBox::Ok);
  m_WinnerInfoPtr = new QMessageBox(QMessageBox::Information, "You won!", "Your score: ", QMessageBox::Ok);

  m_ScoresInfoPtr = new QMessageBox(QMessageBox::Information, "Score Info", "Your score: ", QMessageBox::Ok);
  f_load_scores();

  m_GameMenuBarPtr = m_MainWindow.menuBar();

  m_GameFileMenuPtr = m_GameMenuBarPtr->addMenu("File");
  m_GameFileMenuPtr->addAction("New game", this, SLOT(sf_create_game()), Qt::CTRL | Qt::Key_N);
  m_GameFileMenuPtr->addSeparator();

  m_GameDifficultyMenuPtr = m_GameFileMenuPtr->addMenu("Difficulty");
  m_GameDifficultyMenuPtr->addAction("Easy", this, SLOT(sf_create_easy_game()));
  m_GameDifficultyMenuPtr->addAction("Middle", this, SLOT(sf_create_middle_game()));
  m_GameDifficultyMenuPtr->addAction("Hard", this, SLOT(sf_create_hard_game()));

  m_GameFileMenuPtr->addAction("Scores", this, SLOT(sf_show_scores()));
  m_GameFileMenuPtr->addSeparator();

  m_GameFileMenuPtr->addAction("Exit", this, SLOT(close()));

  m_GameFileMenuPtr = m_GameMenuBarPtr->addMenu("Help");
  m_GameFileMenuPtr->addAction("About", this, SLOT(sf_about()));

  m_GridLayoutPtr = new QGridLayout;
  m_GridLayoutPtr->setSpacing(0);

  m_HorizontLayoutPtr = new QHBoxLayout;
  m_HorizontLayoutPtr->addWidget(m_TimeInfoPtr);
  m_HorizontLayoutPtr->addWidget(m_RestartButtonPtr);
  m_HorizontLayoutPtr->addWidget(m_BombsInfoPtr);

  m_VerticalLayoutPtr = new QVBoxLayout;
  m_VerticalLayoutPtr->addWidget(m_GameMenuBarPtr);
  m_VerticalLayoutPtr->addLayout(m_HorizontLayoutPtr);
  m_VerticalLayoutPtr->addLayout(m_GridLayoutPtr);

  setLayout(m_VerticalLayoutPtr);

  sf_create_game();
}

```

Рисунок 7 - Содержимое файла sapper.cpp.

Продолжение рисунка 7.

```
72  Sapper::~Sapper() {
73    delete m_TimeInfoPtr;
74    delete m_BombsInfoPtr;
75
76    delete m_RestartButtonPtr;
77
78    delete m_GameTimerPtr;
79
80    delete m_AboutInfoPtr;
81    delete m_WinnerInfoPtr;
82    delete m_ScoresInfoPtr;
83
84    delete m_GameFileMenuPtr;
85    delete m_GameDifficultyMenuPtr;
86    delete m_GameMenuBarPtr;
87
88    delete m_GridLayoutPtr;
89    delete m_HorizontLayoutPtr;
90    delete m_VerticalLayoutPtr;
91  }
92
93  void Sapper::f_process_3x3(const std::uint64_t& c_Index, void (Sapper::* v_FunctionPtr)(const std::uint64_t&)) {
94    std::uint64_t v_StartX{}, v_EndX{}, v_tartY{}, v_EndY{};
95
96    v_EndX = v_StartX = c_Index / m_RowsCount;
97
98    if (v_StartX > 0) {
99      v_StartX--;
100   }
101
102   v_EndX += 2;
103
104   if (v_EndX > m_RowsCount) {
105     v_EndX--;
106   }
107
108   v_EndY = v_tartY = c_Index % m_RowsCount;
109
110   if (v_tartY > 0) {
111     v_tartY--;
112   }
113
114   v_EndY += 2;
115
116   if (v_EndY > m_RowsCount) {
117     v_EndY--;
118   }
119
120   for (std::uint64_t v_I{v_StartX}; v_I < v_EndX; ++v_I) {
121     for (std::uint64_t v_J{v_tartY}; v_J < v_EndY; ++v_J) {
122       (this->*v_FunctionPtr)(v_I * m_RowsCount + v_J);
123     }
124   }
125 }
126
127 QPushButton* Sapper::f_create_button(const std::uint64_t& c_Index) {
128   PushButton* v_NewButtonPtr = new PushButton(c_Index);
129
130   connect(v_NewButtonPtr, SIGNAL(left_clicked(std::uint64_t)), SLOT(slotButtonClicked(std::uint64_t)));
131   connect(v_NewButtonPtr, SIGNAL(right_clicked(std::uint64_t)), SLOT(rightButtonClicked(std::uint64_t)));
132   connect(v_NewButtonPtr, SIGNAL(middle_clicked(std::uint64_t)), SLOT(middleButtonClicked(std::uint64_t)));
133
134   m_GameMap.emplace(c_Index, v_NewButtonPtr);
135
136   return v_NewButtonPtr;
137 }
138
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 }
150
```

Продолжение рисунка 7.

```
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 }
159
160 void Sapper::sf_create_game() {
161   m_GameOver = false;
162
163   m_RestartButtonPtr->setIcon(QPixmap(":/icons/smile.png"));
164   m_TimeInfoPtr->display(0);
165   m_BombsInfoPtr->display(static_cast<int>(m_GameDifficulty));
166
167   m_GameMap.clear();
168   m_FieldValuesArray.clear();
169
170   m_VisibleCellsCount = 0;
171   m_FieldValuesArray.resize(m_GameButtonsCount, std::uint64_t{});
172
173   for (std::uint64_t v_I{}, size = m_GameMap.size(); v_I < size; ++v_I) {
174     m_GameMap.at(v_I)->f_clear();
175   }
176
177   for (std::uint64_t v_I{m_GameMap.size()}; v_I < m_GameButtonsCount; ++v_I) {
178     m_GridLayoutPtr->addWidget(f_create_button(v_I), v_I / m_RowsCount, v_I % m_RowsCount);
179   }
180
181   f_place_bombs();
182 }
183
184 void Sapper::f_place_bombs() {
185   std::set<std::uint64_t> v_BobmIndexes;
186
187   while (v_BobmIndexes.size() < m_GameDifficulty) {
188     std::uint64_t v_Index{QRandomGenerator::global()->generate() % (m_GameButtonsCount + 1)};
189
190     if (m_FieldValuesArray[v_Index] < m_GameButtonsCount) {
191       v_BobmIndexes.insert(v_Index);
192     }
193   }
194
195   std::uint64_t v_BombsPlaced{};
196
197   for (const std::uint64_t& c_Index : v_BobmIndexes) {
198     m_FieldValuesArray[c_Index] = m_GameButtonsCount + v_BombsPlaced++;
199     f_process_3x3(c_Index, &Sapper::f_add_bomb);
200   }
201 }
202
203 void Sapper::f_add_bomb(const std::uint64_t& c_Index) {
204   if (c_Index < m_GameButtonsCount) {
205     if (m_FieldValuesArray[c_Index] < m_GameButtonsCount) {
206       ++m_FieldValuesArray[c_Index];
207     }
208   }
209 }
210
211 void Sapper::f_show_unckecked_fields(const std::uint64_t& c_Index) {
212   if (!m_GameMap.at(c_Index)->f_is_flag()) {
213     slotButtonClicked(c_Index);
214   }
215 }
216
217 void Sapper::f_flags_count(const std::uint64_t& c_Index) {
218   if (m_GameMap.at(c_Index)->f_is_flag()) {
219     ++m_CheckedFields3x3Count;
220   }
221 }
222
```

Продолжение рисунка 7.

```
307 void Sapper::sf_show_scores() {
308   m_ScoresInfoPtr->setText(
309     "<table><tr><td><b/>Difficulty</td><td>  |  </td><td><b/>Time</td></tr>"
310     "<tr><td>Easy</td><td>  |  </td><td>" + QString().number(m_EasyBestScore) + "</td></tr>"
311     "<tr><td>Medium</td><td>  |  </td><td>" + QString().number(m_MediumBestScore) + "</td></tr>"
312     "<tr><td>Hard</td><td>  |  </td><td>" + QString().number(m_HardBestScore) + "</td><tr></table>");
313
314   m_ScoresInfoPtr->show();
315 }
316
317 void Sapper::sf_about() { m_AboutInfoPtr->show(); }
318
319 void Sapper::slotButtonClicked(const std::uint64_t& c_Index) {
320   if (m_GameOver || (c_Index > m_GameButtonsCount)
321       || m_GameMap.at(c_Index)->isFlat() || (m_GameMap.at(c_Index)->f_get_icon_index() != 0)) {
322     return;
323   }
324
325   if (!m_GameTimerPtr->isActive()) {
326     m_GameTimerPtr->start(1000);
327   }
328
329   if (m_FieldValuesArray[c_Index] < m_GameButtonsCount) {
330     m_GameMap.at(c_Index)->setFlat(true);
331
332     ++m_VisibleCellsCount;
333
334     if (m_FieldValuesArray[c_Index] > 0) {
335       m_GameMap.at(c_Index)->setIcon(QPixmap(":/icons/" + QString().number(m_FieldValuesArray[c_Index]) + ".png"));
336     }
337
338     if (m_FieldValuesArray[c_Index] == 0) {
339       f_process_3x3(c_Index, &Sapper::slotButtonClicked);
340     }
341   }
342
343   if (m_FieldValuesArray[c_Index] >= m_GameButtonsCount) {
344     m_GameOver = true;
345     m_RestartButtonPtr->setIcon(QPixmap(":/icons/bad.png"));
346     m_GameTimerPtr->stop();
347     f_show_all_bombs();
348     return;
349   }
350
351   if (m_VisibleCellsCount >= (m_GameButtonsCount - m_GameDifficulty)) {
352     f_win();
353   }
354 }
355
356 void Sapper::rightButtonClicked(const std::uint64_t& c_Index) {
357   if (m_GameOver) {
358     return;
359   }
360
361   switch (m_GameMap.at(c_Index)->f_get_icon_index()) {
362     case 0: {
363       m_GameMap.at(c_Index)->setIcon(QPixmap());
364       ++m_CheckedFieldsCount;
365       break;
366     }
367     case 1: {
368       m_GameMap.at(c_Index)->setIcon(QPixmap(":/icons/check.png"));
369       --m_CheckedFieldsCount;
370       break;
371     }
372     case 2: {
373       m_GameMap.at(c_Index)->setIcon(QPixmap(":/icons/query.png"));
374       break;
375     }
376   }
377
378   m_BombsInfoPtr->display(static_cast<int>(m_CheckedFieldsCount));
379 }
380
```

Продолжение рисунка 7.

```
381 void Sapper::middleButtonClicked(const std::uint64_t& c_Index) {
382   if (m_GameOver) {
383     return;
384   }
385
386   if (m_FieldValuesArray[c_Index] == 0) {
387     return;
388   }
389
390   m_CheckedFields3x3Count = 0;
391
392   f_process_3x3(c_Index, &Sapper::f_flags_count);
393
394   if (m_CheckedFields3x3Count == m_FieldValuesArray[c_Index]) {
395     f_process_3x3(c_Index, &Sapper::f_show_unckecked_fields);
396   }
397 }
398
```

Рисунок 8 - Графический интерфейс приложения.

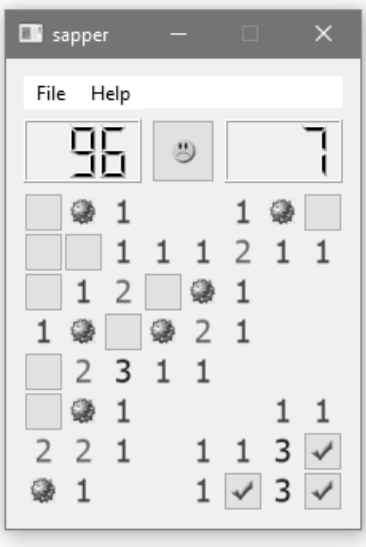

Рисунок 9 - Проигрыш.

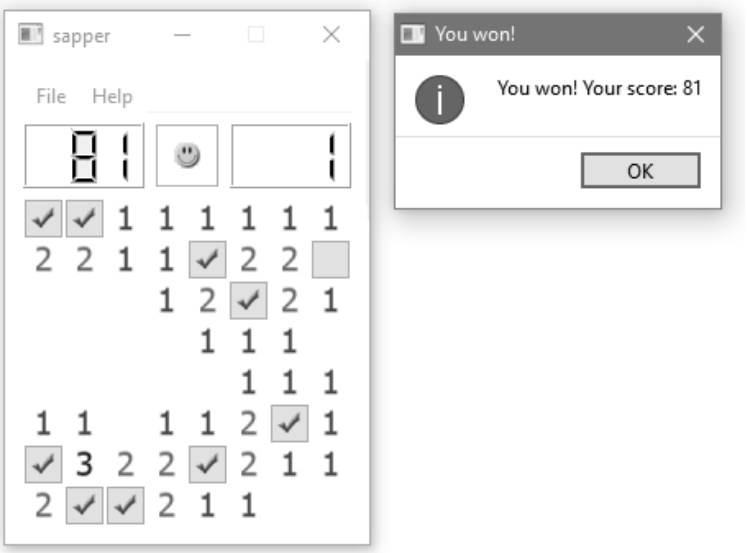

Рисунок 10 - Победа.

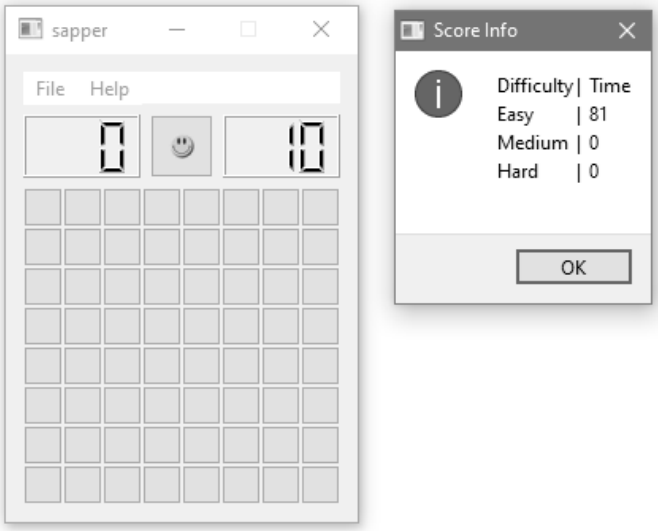

Рисунок 11 - Рекорды.

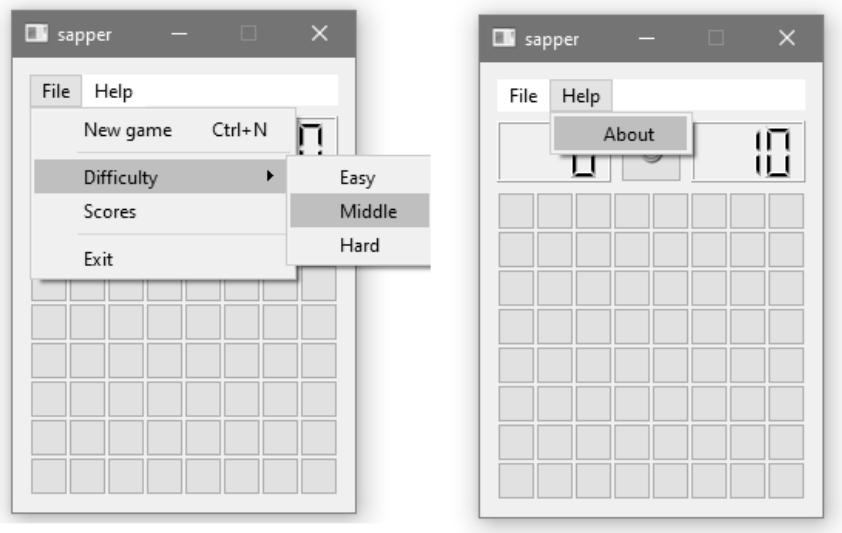

Рисунок 11 - Окна меню приложения.

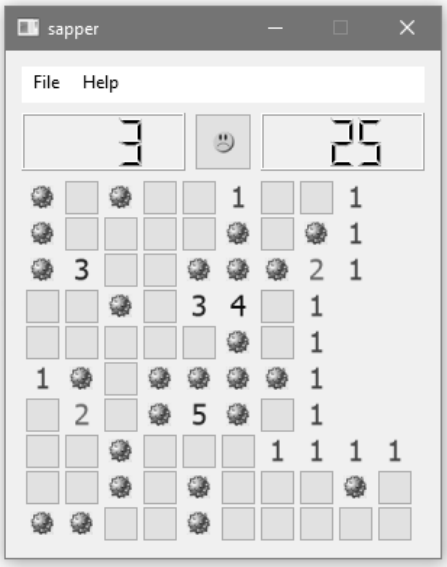

Рисунок 12 - Средняя сложность.

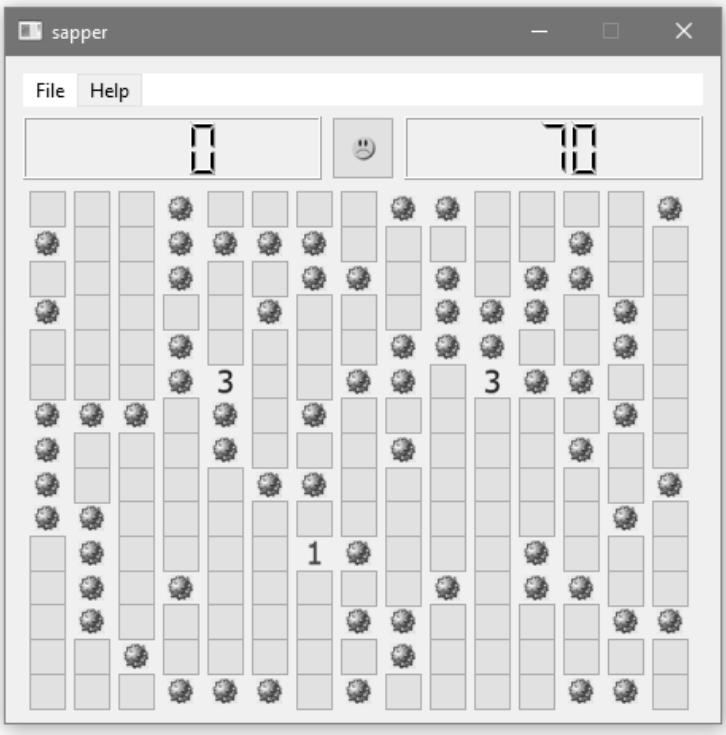

Рисунок 13 - Высокая сложность.

Вывод: Приобрел практические навыки проектирования и разработки приложений с графическим пользовательским интерфейсом в ОС Windows средствами Qt.