Санкт-Петербургский Политехнический Университет Петра Великого
Институт компьютерных наук и технологий
Кафедра компьютерных систем и программных технологий

Программирование

Отчет по лабораторной работе
Игра "Морской бой"

Работу выполнила:
Власова А.В.
Группа: 13501/4
Преподаватель:
Вылегжанина К.Д.

Санкт-Петербург
2016

Содержание

1 Игра Морской бой

Морской бой - это традиционная настольная игра, известная каждому человеку с детства. Несмотря на то, что Морской бой существует уже больше 80 лет, эта игра остаеся популярной в наши дни.

1.1 Задание

Реализовать проект Морской бой, позволяющий вести игру между человеком и компьютером по базовым правилам.

1.2 Концепция

Готовое приложение дает возможность пользователю играть в Морской бой с компьютером в соответствии со стандартными правилами. Программа обладает графическим интерфейсом.

1.3 Правила игры

Поле каждого игрока представляет собой квадрат 10x10, на котором размещаются корабли. Поле содержит числовые и буквенные координаты (по вертикали числа от 1 до 10,а по горизонтали буквы от A до J). Для классической игры используются 4 однопалубных корабля, 3 двупалубных, 2 трехпалубных и 1 четырехпалубный корабль. Их размещают внутри игрового поля. По правилам, корабли не должны соприкасаться. Размещать корабли можно как горизонтально, так и вертикально.

Рядом со своим полем игрок видит поле противника, где крестиком отмечает попадания по чужим кораблям. Попавший игрок делает ещё один ход.

1.3.1 Игровой процесс

- Игрок, выполняющий ход, называет координату, на которой, по его мнению, располагается корабль соперника. Например, A3.
- При промахе игрок получает от противника сообщение «Мимо!», при попадании «Ранил» или «Убил».
- Игра продолжается до потопления всех кораблей одного из игроков.

1.3.2 Нарушения

- Количество кораблей не соответствует правилам
- Корабли расположены вплотную друг к другу
- Изменен размер поля
- Указаны неправильные координаты

1.4 Минимально работоспособный продукт

Минимальным работоспособным продуктом является консольное приложение, позволяющее вести игру между человеком и компьютером по базовым правилам.

1.5 Диаграмма прецедентов использования

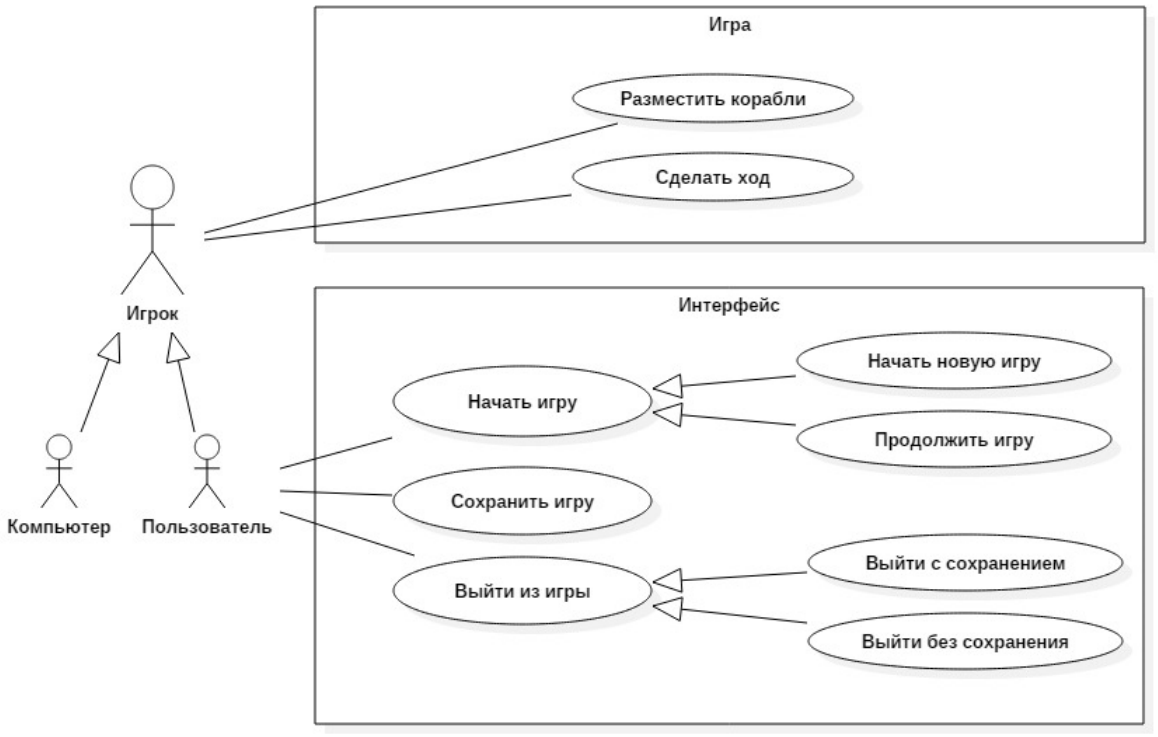


Рис. 1: Диаграмма прецедентов

На рис.1 изображена диаграмма прецедентов использования. Пользователю дается возможность начинала новой игры, сохранения текущего состояния партии и продолжения сохраненных партий. Во время игрового процесса игроки могут расставлять свои корабли и делать ходы.

1.6 Выводы

В данном разделе определены концепция готового приложения и минимальный работоспособный продукт. Также приведены стандартные правила игры в Морской бой, рассмотрен игоровой процесс и основные нарушения. Кроме того, в разделе представлена диаграмма прецедентов использования.

2 Проектирование приложения, реализующего игру Морской бой

2.1 Выделенные подпроекты

В процессе проектирования приложения было выделено четыре подпроекта.

- **Core**

  Библиотека приложения, представляющая игровую модель.

- **Console**

  Консольное приложение, обеспечивающее взаимодействие пользователя с ядром через консоль.

- **GUI**

  Графическое приложение, предоставляющее пользователю графический интерфейс для взаимодействия с ядром приложения.

- **Test**

  Автоматические тесты, созданные для тестирования основной функциональности ядра.

2.2 Описание интерфейса библиотеки

Интерфейс библиотеки выделен в отдельный класс **GameAPI**, содержащий в себе следующие методы:

- Метод, позволяющий совершить ход компьютера
  bool makeComputerMove() noexcept;
- Метод, позволяющий совершить ход пользователя
  bool makeUserMove(int x, int y) noexcept;
- Метод, возвращающий указатель на поле игрока
  Field* getUserField() const noexcept;
- Метод, возвращающий указатель на поле компьютера
  Field* getComputerField() const noexcept;
- Метод, позволяющий разместить корабли пользователя
  void placeUserShip(int x, int y, int lenght, shipLocation location) noexcept;
- Метод, позволяющий разместить корабли автоматически
  void placeShipsAutomatically(Field* field) noexcept;
- Метод, позволяющий узнать, все ли корабли одного поля разрушены
  bool allShipsDestroyed(Field* field) noexcept;

2.3 Диаграмма компонентов

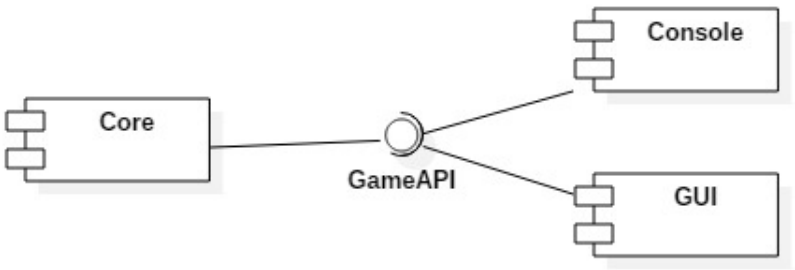


Рис. 2: Диаграмма компонентов

На рис.2 представлена диаграма компонентов, отображающая зависимости между основными компонентами приложения.

2.4 Выводы

В данном разделе рассмотрен процесс проектирования приложения. Описаны выделенные подпроекты, а также методы класса textbfGameAPI, содержащего интерфейс ядра. Помимо этого, в разделе представлена диаграмма компонентов приложения.

3 Реализация игры Морской бой

3.1 Среда разработки

Операционная система: Windows 8
Среда разработки: Qt Creator 3.5.1
Компилятор: MinGW 4.9.2

3.2 Выделенные классы

В библиотеке были выделены следующие классы:

- **Cell** - содержит основную информацию об одной клетке игрового поля. Позволяет определить координаты клетки и ее статус.
- **Ship** - содержит основную информацию о корабле. Позволяет определить его расположение на игровом поле, статус и количество палуб.
- **Field** - обеспечивает взаимодействие с игровым полем одного из игроков. Содержит информацию обо всех кораблях поля, о количестве разрушенных кораблей, обеспечивает возможность автоматического и ручного размещения кораблей.
- **GameAPI** - интерфейс ядра. Описан в разделе 2.1

В подпроекте **Console** выделен единственный класс **Application**, обеспечивающий взаимодействие пользователя с ядром через консоль. Содержит в себе методы, позволяющие выводить поля пользователя и компьютера в консоль, размещать корабли и вести игровой процесс.

Проект **GUI** содержит в себе три класса:

- **MainWindow** - главное окно графического приложения, отображающее игровое меню.
- **GameWindow** - окно, в котором происходит игровой процесс.
- **ResultWindow** - окно, отображащее победителя игры.

3.3 Примеры работы консольного приложения

Для демонстрации работы консольного приложения ниже представлены снимки экрана работающего приложения.

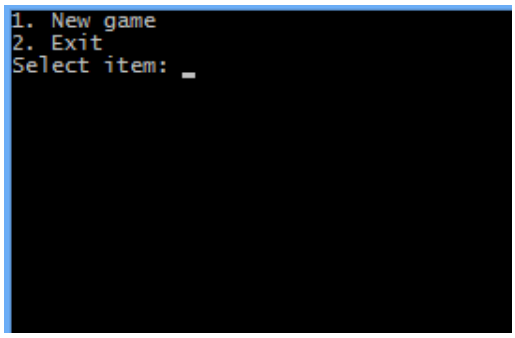


Рис. 3: Главное меню

На рис.3 представлено главное меню консольного приложения.

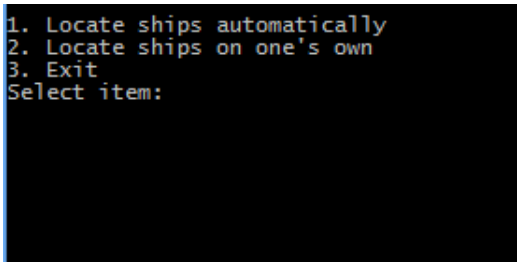


Рис. 4: Меню выбора способа расстановки кораблей

На рис.4 показано меню, позволяющее пользователю выбрать способ расстановки своих кораблей: автоматический или ручной.

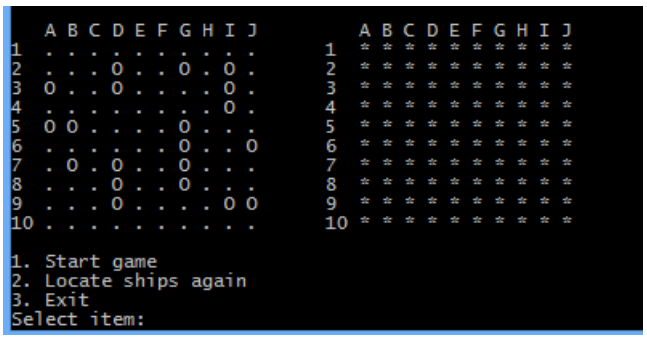


Рис. 5: Начало игры

На рис.5 изображены поля с автоматически расставленными кораблями. После размещения кораблей пользователю предлагается начать игру или расставить корабли заново.

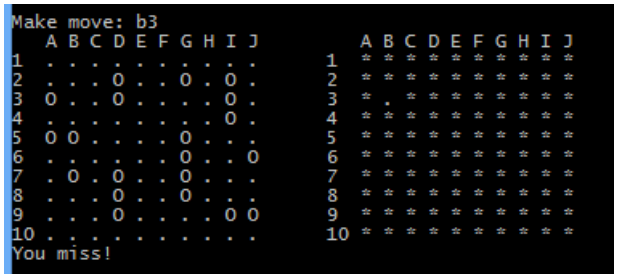


Рис. 6: Ход пользователя

На рис.6 показан пример хода пользователя.

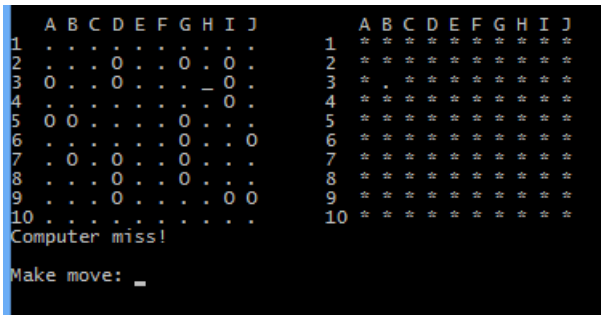


Рис. 7: Ход компьютера

На рис.7 показан пример рандомного хода компьютера.

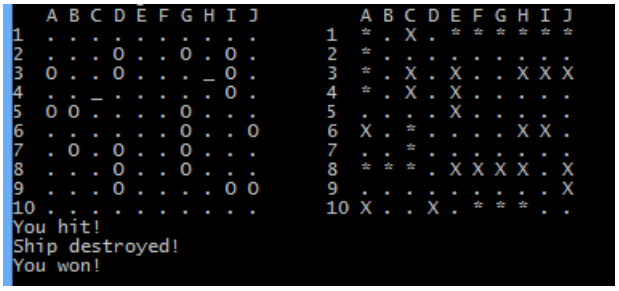


Рис. 8: Вывод победителя

Когда все корабли одного из игроков разрушены, выводится победитель игры. Пример вывода победителя представлен на рис.8.

3.4 Пример работы графического приложения

Ниже приведены снимки экрана для демонстрации работы графического приложения.

Рис. 9: Главное меню

На рис.9 отображено главное меню графического приложения.

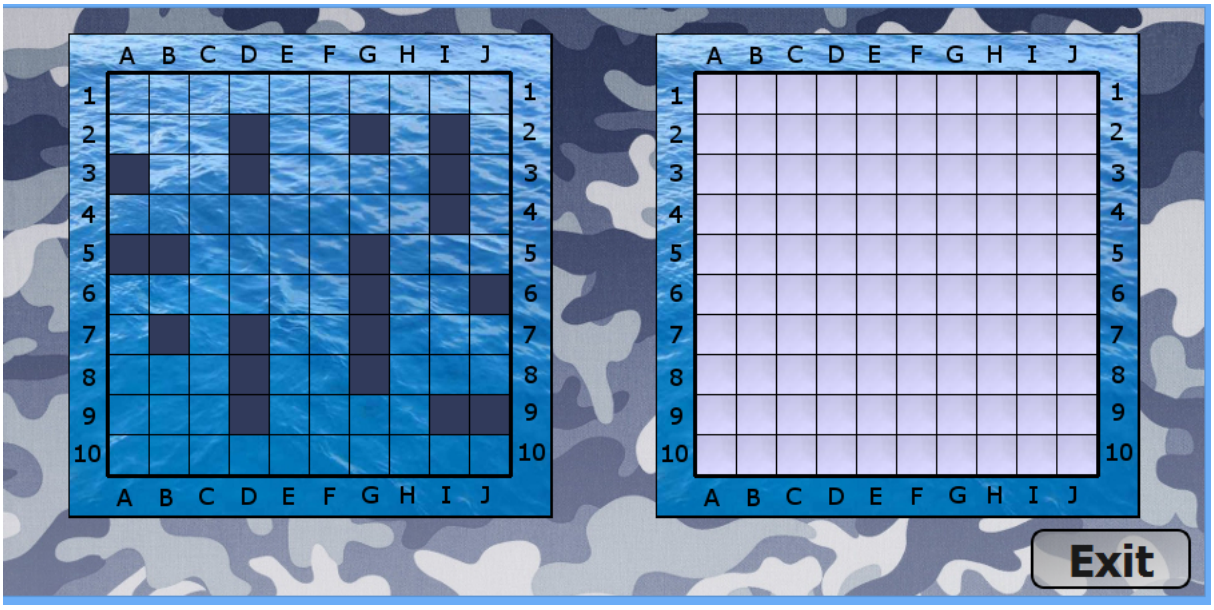


Рис. 10: Начало игры

На рис.10 показано, как выглядят игровые поля в начале игры.

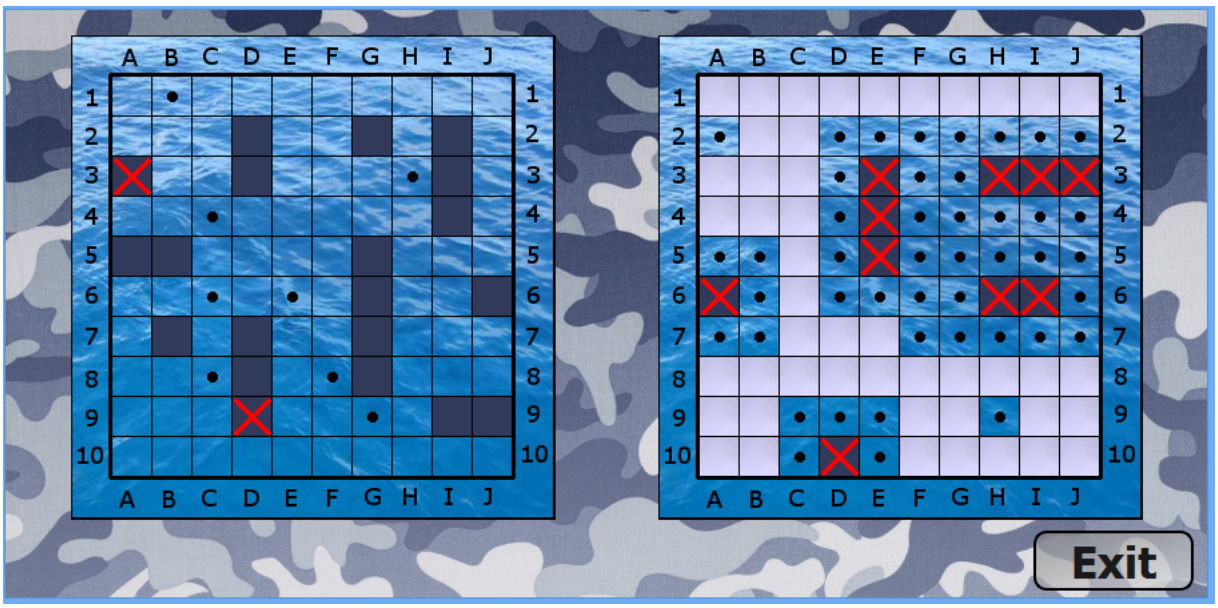


Рис. 11: Игровой процесс

На рис.11 показан пример того, как могут выглядеть поля во время игры.

Рис. 12: Вывод победителя

После того, как все корабли одного из полей будут потоплены, в отдельном окне отобразится победитель игры.

3.5 Выводы

В данном разделе были описаны все классы, выделенные в процессе работы над проектом. Также были сделаны снимки экрана, демонстрирующие работу консольного и графического приложений.

4 Процесс обеспечения качества и тестирование

4.1 Просмотр кода и демонстрации

Для обнаружения ошибок в коде программы один раз был проведен просмотр кода, замечания по которому практически полностью исправлены. Кроме того, были осуществлены 2 демонстрации, во время проведения которых были найдены ошибки в работе программы. Все замечания по демонстрациям также исправлены.

4.2 Тестирование

Для проверки работы библиотеки использовались автоматические тесты, покрывающие основную функциональность ядра. Также в процессе разработки приложения проводилось ручное тестирование программы.

4.3 Выводы

В данном разделе описаны просмотр кода и демонстрации, проведенные во время разработки приложения. Также рассказано о процессе тестирования работы программы.

5 Выводы

Во время разработки приложения был значительно увеличен объем знаний, касаемых работы с языком программирования C++. Были изучены раннее неизвестные возможности C++ и лучше усвоены принципы объектно-ориентированного программирования. Кроме того, стали известны особенности стандарта C++11, а также основные паттерны проектирования. Помимо этого, был получен опыт в создании графического приложения с использованием библиотеки Qt.

Результатом работы над проектом стала рабочая программа, позволяющая пользователям играть в Морской бой, используя при этом консоль или графичекое приложение.

6 Приложение 1

Листинг 1: game.h

```
1  #ifndef GAME_H
2  #define GAME_H
3
4  #include <string>
5  #include <algorithm>
6
7  #include "field.h"
8
9
10
11 /**
12  * @brief Игровой интерфейс
13  */
14 class GameAPI
15 {
16 public:
17     GameAPI() noexcept;
18     ~GameAPI() noexcept;
19     /**
20      * @brief Совершить ход компьютера
21      * @return true при попадании
22      * @return false при промахе
23      */
24     bool makeComputerMove() noexcept;
25     /**
26      * @brief Совершить ход игрока
27      * @param x буквенная координата
28      * @param y цифровая координата
29      * @return true при попадании
30      * @return false при промахе
31      */
32     bool makeUserMove(int x, int y) noexcept;
33     /**
34      * @brief Получить поле игрока
35      * @return указатель на поле игрока
36      */
37     Field* getUserField() const noexcept;
38     /**
39      * @brief Получить поле компьютера
40      * @return указатель на поле компьютера
41      */
42     Field* getComputerField() const noexcept;
43     /**
44      * @brief Разместить корабль пользователя
```

```
45      * @param x буквенная координата
46      * @param y цифровая координата
47      * @param lenght длина
48      * @param location расположение
49      */
50     void placeUserShip(int x, int y, int lenght, shipLocation location) noexcept;
51     /**
52      * @brief Разместить корабли автоматически
53      * @param field указатель на поле
54      */
55     void placeShipsAutomatically(Field* field) noexcept;
56     /**
57      * @brief Узнать, все ли корабли разрушены
58      * @param field указатель на поле
59      * @return true/false
60      */
61     bool allShipsDestroyed(Field* field) noexcept;
62
63
64 private:
65
66     Field* userField;
67     Field* computerField;
68
69
70 };
71
72 #endif // GAME_H
```

Листинг 2: game.cpp

```
1  #include "game.h"
2
3  GameAPI::GameAPI() noexcept :userField(new Field()), computerField(new Field()){}
4
5  GameAPI::~GameAPI() noexcept
6  {
7      delete userField;
8      delete computerField;
9  }
10
11
12 Field* GameAPI::getUserField() const noexcept
13 {
14     return userField;
15 }
16
17 Field* GameAPI::getComputerField() const noexcept
18 {
19     return computerField;
20 }
21
22 bool GameAPI::makeComputerMove() noexcept
23 {
24     int x,y;
25     x=std::rand()%(Field::FIELD_SIZE-1);
26     y=std::rand()%(Field::FIELD_SIZE-1);
27     if(userField->getFieldCells()[y][x].getStatus()==cellStatus::stricken
28             || userField->getFieldCells()[y][x].getStatus()==cellStatus::tried)
29                 makeComputerMove();
30     return userField->shot(x, y);
31
32 }
33
34 bool GameAPI::makeUserMove(int x, int y) noexcept
35 {
36     return computerField->shot(x,y);
37 }
38
39 void GameAPI::placeUserShip(int x, int y, int lenght, shipLocation location) noexcept
40 {
41     userField->placeShip(x, y, lenght, location);
42 }
43
44 bool GameAPI::allShipsDestroyed(Field* field) noexcept
45 {
```

```
46      int count=0;
47      for(int i=0; i<field->getnumberSetShips(); i++)
48          if (field->getFieldShips()[i].getShipStatus()==shipStatus::destroyed)
49              count++;
50      return count==field->getnumberSetShips();
51  }
52
53  void GameAPI::placeShipsAutomatically(Field *field) noexcept
54  {
55      field->locateAutomatically();
56  }
```

Листинг 3: field.h

```
1   #ifndef FIELD_H
2   #define FIELD_H
3
4   #include <algorithm>
5   #include <random>
6   #include <ctime>
7   #include <vector>
8   #include "ship.h"
9
10  /**
11   * @brief Игровое поле
12   */
13  class Field
14  {
15  public:
16      static const int FIELD_SIZE = 10;
17      static const int NUMBER_OF_SHIPS = 10;
18
19      Field() noexcept;
20
21      /**
22       * @brief Разместить корабль
23       * @param x буквенная координата первой палубы корабля
24       * @param y цифровая координата первой палубы корабля
25       * @param lenght длина корабля
26       * @param location расположение
27       */
28      void placeShip(int x, int y, int lenght, shipLocation location) noexcept;
29      /**
30       * @brief Узнать, все ли корабли размещены на поле
31       * @return  true/false
32       */
33      bool allShipsLocate() noexcept;
34      /**
35       * @brief Узнать, является ли клетка палубой
36       * @param x буквенная координата
37       * @param y цифровая координата
38       * @return true/false
39       */
40      bool isDeck(int x, int y) noexcept;
41
42      /**
43       * @brief Выстрелить в клетку игрового поля
44       * Изменяет статус клетки
45       * @param x буквенная координата
46       * @param y цифровая координата
47       * @return true при попадании в палубу
48       * @return false при промахе
49       */
50      bool shot(int x, int y) noexcept;
51      /**
52       * @brief Узнать, можно ли разместить корабль на данные координаты
53       * @param x буквенная координата первой палубы
54       * @param y цифровая координата первой палубы
55       * @param lenght длина
56       * @param location расположение
57       * @return true/false
58       */
59      bool canPlaceShip(int x, int y, int lenght, shipLocation location) noexcept;
60      /**
61       * @brief Получить клетки игрового поля
62       * @return указатель на клетки поля
```

```
63          */
64         std::vector<std::vector<Cell>> getFieldCells() noexcept;
65         /**
66          * @brief Получить корабли поля
67          * @return указатель на массив кораблей
68          */
69         std::vector<Ship> getFieldShips() noexcept;
70         /**
71          * @brief Узнать число размещенных кораблей
72          * @return количество размещенных кораблей
73          */
74         int getnumberSetShips() const noexcept;
75         /**
76          * @brief Рандомно разместить один корабль
77          * @param lenght длина корабля
78          */
79         void locateShipRandomly(int lenght) noexcept;
80         /**
81          * @brief Рандомно разместить все корабли
82          */
83         void locateAutomatically() noexcept;
84         /**
85          * @brief Изменить статус клеток вокруг разрушенного корабля
86          * @param ship корабль
87          */
88         void changeCellsAroundShip(Ship ship) noexcept;
89         /**
90          * @brief Узнать, разрушен ли данный корабль
91          * @param shipNumber номер корабля
92          * @return true/false
93          */
94         bool isShipDestroyed(int shipNumber) noexcept;
95         /**
96          * @brief Узнать, какому кораблю принадлежит палуба
97          * @param x буквенная координата
98          * @param y цифровая координата
99          * @return номер корабля
100         */
101        int whoseDeck(int x, int y) noexcept;
102
103
104    private:
105        std::vector<std::vector<Cell>> fieldCells;
106        std::vector<Ship> fieldShips;
107        int numberSetShips;
108
109    };
110
111    #endif // FIELD_H
```

Листинг 4: field.cpp

```
1   #include "field.h"
2
3   Field::Field() noexcept
4   {
5
6       for(int i=0; i<FIELD_SIZE;i++){
7           std::vector<Cell> mas;
8           for(int j=0;j<FIELD_SIZE;j++){
9               mas.push_back(Cell(j, i));
10          }
11          fieldCells.push_back(mas);
12      }
13      numberSetShips=0;
14  }
15
16
17
18  void Field::placeShip(int x, int y, int lenght, shipLocation location) noexcept
19  {
20      if (location==shipLocation::horizontal)
21          for(int i=0;i<lenght;++i)
22              fieldCells[y][x+i].setStatus(cellStatus::whole);
23      else for(int i=0;i<lenght;++i)
24          fieldCells[y+i][x].setStatus(cellStatus::whole);
```

```
25      fieldShips.push_back(Ship(x,y,lenght,location));
26      numberSetShips++;
27  }
28
29  bool Field::allShipsLocate() noexcept
30  {
31      return numberSetShips==NUMBER_OF_SHIPS;
32
33  }
34
35  bool Field::isDeck(int x, int y) noexcept
36  {
37      return (fieldCells[y][x].getStatus()==cellStatus::whole || fieldCells[y][x].getStatus()==
        ↪ cellStatus::stricken);
38  }
39
40  bool Field::shot(int x, int y) noexcept
41  {
42      if (isDeck(x,y))
43      {
44          for (int i=0; i<NUMBER_OF_SHIPS; i++)
45          {
46              if (fieldShips[i].shot(x, y))
47              {
48                  fieldShips[i].setShipStatus(x, y);
49                  fieldCells[y][x].setStatus(cellStatus::stricken);
50                  return true;
51              }
52          }
53      }
54      else fieldCells[y][x].setStatus(cellStatus::tried);
55      return false;
56  }
57
58  std::vector<std::vector<Cell>> Field::getFieldCells() noexcept
59  {
60      return this->fieldCells;
61  }
62
63  std::vector<Ship> Field::getFieldShips() noexcept
64  {
65      return this->fieldShips;
66  }
67
68  bool Field::canPlaceShip(int x, int y, int lenght, shipLocation line) noexcept
69  {
70      if (line==shipLocation::horizontal)
71      {
72          for (int i=std::max(0,y-1);
73               i<=std::min(FIELD_SIZE-1,y+1);
74               i++)
75              for (int j=std::max(0,x-1);
76                   j<=std::min(FIELD_SIZE-1,x+lenght);
77                   j++)
78                  if(fieldCells[i][j].getStatus()!=cellStatus::blank){
79                      return false;
80
81                  }
82          return true;
83      }
84      else
85      {
86          for (int i=std::max(0,y-1);
87               i<=std::min(FIELD_SIZE-1,y+lenght);
88               i++)
89              for(int j=std::max(0,x-1);
90                  j<=std::min(FIELD_SIZE-1,x+1);
91                  j++)
92                  if(fieldCells[i][j].getStatus()!=cellStatus::blank){
93                      return false;
94                  }
95          return true;
96      }
97
98  }
99
```

```
100
101 void Field::locateShipRandomly(int lenght) noexcept
102 {
103     int x,y;
104             shipLocation line=shipLocation(std::rand()%2);
105             do
106             {
107                 do
108                 {
109                     y=std::rand()%FIELD_SIZE;
110                 }while(line==shipLocation::vertical && y>FIELD_SIZE-lenght);
111                 do
112                 {
113                     x=std::rand()%FIELD_SIZE;
114                 }while(line==shipLocation::horizontal && x>FIELD_SIZE-lenght);
115
116             }while(!canPlaceShip(x,y,lenght,line));
117             placeShip(x,y,lenght,line);
118
119 }
120
121 void Field::locateAutomatically() noexcept
122 {
123     for (int i=0;i<1;i++)
124     {
125         locateShipRandomly(4);
126     }
127     for (int i=0;i<2;i++)
128     {
129         locateShipRandomly(3);
130     }
131     for (int i=0;i<3;i++)
132     {
133         locateShipRandomly(2);
134     }
135     for (int i=0;i<4;i++)
136     {
137         locateShipRandomly(1);
138     }
139 }
140
141 int Field::getnumberSetShips() const noexcept
142 {
143     return this->numberSetShips;
144 }
145
146 void Field::changeCellsAroundShip(Ship ship) noexcept
147 {
148     if (ship.getShipLocation()==shipLocation::horizontal)
149     {
150         for (int i=std::max(0, ship.getShipCells()[0].getY()-1);
151              i<=std::min(ship.getShipCells()[0].getY()+1, FIELD_SIZE-1);
152              i++)
153             for (int j=std::max(0, ship.getShipCells()[0].getX()-1);
154                  j<=std::min(ship.getShipCells()[ship.getLenght()-1].getX()+1, FIELD_SIZE-1);
155                  j++)
156                 if (fieldCells[i][j].getStatus()!=cellStatus::stricken)
157                     fieldCells[i][j].setStatus(cellStatus::tried);
158
159     }
160     else
161     {
162         for (int i=std::max(0, ship.getShipCells()[0].getY()-1);
163              i<=std::min(ship.getShipCells()[ship.getLenght()-1].getY()+1, FIELD_SIZE-1);
164              i++)
165             for (int j=std::max(0, ship.getShipCells()[0].getX()-1);
166                  j<=std::min(ship.getShipCells()[0].getX()+1, FIELD_SIZE-1);
167                  j++)
168                 if (fieldCells[i][j].getStatus()!=cellStatus::stricken)
169                     fieldCells[i][j].setStatus(cellStatus::tried);
170     }
171 }
172
173 bool Field::isShipDestroyed(int shipNumber) noexcept
174 {
175     return fieldShips[shipNumber].getShipStatus()==shipStatus::destroyed;
```

```
176
177 }
178
179 int Field::whoseDeck(int x, int y) noexcept
180 {
181     bool flag=false;
182     int i;
183     for (i=0; i<NUMBER_OF_SHIPS; i++){
184         for (int j=0; j<fieldShips[i].getLenght(); j++)
185             if (fieldShips[i].getShipCells()[j].getX()==x &&
186                     fieldShips[i].getShipCells()[j].getY()==y){
187                 flag=true;
188                 break;
189             }
190         if (flag==true)
191             break;
192     }
193     return i;
194 }
```

Листинг 5: ship.h

```
1  #ifndef SHIP_H
2  #define SHIP_H
3
4  #include <vector>
5  #include "cell.h"
6
7  enum class shipStatus {whole, stricken, destroyed};
8  enum class shipLocation {vertical, horizontal};
9
10 /**
11  * @brief Корабль
12  */
13 class Ship
14 {
15 public:
16     Ship() noexcept;
17     Ship(int x, int y, int lenght, shipLocation location) noexcept;
18
19     /**
20      * @brief Создать корабль
21      * @param field игровое поле
22      * @param x первая буквенная координата корабля
23      * @param y первая цифровая координата корабля
24      * @param lenght длина корабля
25      * @param location расположение корабля горизонтальноевертикальное(/)
26      */
27     void createShip(int x, int y, int lenght, shipLocation location) noexcept;
28     /**
29      * @brief Установить статус корабля
30      * @param x первая буквенная координата корабля
31      * @param y первая цифровая координата корабля
32      */
33     void setShipStatus(int x, int y) noexcept;
34     /**
35      * @brief Проверить, попал ли игрок в клетку корабля
36      * При попадании изменяет статус клетки
37      * @param field игровое поле
38      * @param x буквенная координата
39      * @param y цифровая координата
40      * @return true/false
41      */
42     bool shot(int x, int y) noexcept;
43     /**
44      * @brief Получить статус корабля
45      * @return статус
46      */
47     shipStatus getShipStatus() const noexcept;
48     /**
49      * @brief Получить клетки корабля
50      * @return клетки корабля
51      */
52     std::vector<Cell> getShipCells() const noexcept;
53     /**
54      * @brief Получить длину корабля
```

```
55          * @return длина
56          */
57         int getLenght() const noexcept;
58         /**
59          * @brief Получить расположение корабля
60          * @return расположение
61          */
62         shipLocation getShipLocation() const noexcept;
63
64
65
66
67 private:
68         int firstX;
69         int firstY;
70         int lenght;
71         shipStatus status;
72         std::vector<Cell> shipCells;
73         shipLocation location;
74
75 };
76
77 #endif // SHIP_H
```

Листинг 6: ship.cpp

```
1  #include "ship.h"
2
3  Ship::Ship() noexcept:firstX(0), firstY(0), lenght(0), status(shipStatus::whole){}
4
5
6  Ship::Ship(int x, int y, int lenght, shipLocation location) noexcept
7  {
8      this->lenght=lenght;
9      this->firstX=x;
10     this->firstY=y;
11     this->location=location;
12     this->status=shipStatus::whole;
13     for (int i=0; i<lenght; i++)
14     {
15         if (location==shipLocation::horizontal)
16         {
17             shipCells.push_back(Cell(x+i, y, cellStatus::whole));
18         }
19         else
20         {
21             shipCells.push_back(Cell(x, y+i, cellStatus::whole));
22         }
23     }
24 }
25
26 shipStatus Ship::getShipStatus() const noexcept
27 {
28     return status;
29 }
30
31 void Ship::setShipStatus(int x, int y) noexcept
32 {
33     int count=0;
34     for (unsigned int i=0; i<shipCells.size(); i++){
35         if (shipCells[i].getStatus()==cellStatus::stricken)
36             count++;
37         else if (shipCells[i].operator ==(Cell(x, y)))
38         {
39             shipCells[i].setStatus(cellStatus::stricken);
40             count++;
41         }
42     }
43         if (count==lenght)
44             status=shipStatus::destroyed;
45         else if (count!=0)
46             status=shipStatus::stricken;
47 }
48
49 std::vector<Cell> Ship::getShipCells() const noexcept
50 {
```

```
51        return this->shipCells;
52 }
53
54 bool Ship::shot(int x, int y) noexcept
55 {
56     for (unsigned int i=0; i<shipCells.size(); i++)
57         if (shipCells[i].operator ==(Cell(x, y, cellStatus::whole)))
58         {
59             {
60                 shipCells[i].setStatus(cellStatus::stricken);
61                 return true;
62             }
63         }
64
65     return false;
66 }
67
68 int Ship::getLenght() const noexcept
69 {
70     return this->lenght;
71 }
72
73 shipLocation Ship::getShipLocation() const noexcept
74 {
75    return this->location;
76 }
```

Листинг 7: cell.h

```
1  #ifndef CELL_H
2  #define CELL_H
3
4  enum class cellStatus {whole, stricken, blank, tried};
5
6  /**
7   * @brief Клетка игрового поля
8   */
9  class Cell
10 {
11 public:
12     Cell() noexcept;
13     Cell(int x, int y) noexcept;
14     Cell(int x, int y, cellStatus status) noexcept;
15
16     Cell& operator=(const Cell& other) noexcept;
17     bool operator==(const Cell& other) const noexcept;
18
19     /**
20      * @brief Установить буквенную координату клетки
21      * @param x буквенная координата
22      */
23     void setX(int x) noexcept;
24     /**
25      * @brief Установить цифровую координату клетки
26      * @param y цифровая координата
27      */
28     void setY(int y) noexcept;
29     /**
30      * @brief Установить статус клетки
31      * @param status статус
32      */
33     void setStatus(cellStatus status) noexcept;
34     /**
35      * @brief Получить статус клетки
36      * @return статус
37      */
38     cellStatus getStatus() const noexcept;
39     /**
40      * @brief Получить буквенную координату
41      * @return буквенная координата
42      */
43     int getX() const noexcept;
44     /**
45      * @brief Получить цифровую координату
46      * @return цифровая координата
47      */
```

```
48      int getY() const noexcept;
49
50  private:
51      int x;
52      int y;
53      cellStatus status;
54
55  };
56
57
58  #endif // CELL_H
```

Листинг 8: cell.cpp

```
1   #include "cell.h"
2
3   Cell::Cell() noexcept: x(0), y(0), status(cellStatus::blank){}
4
5   Cell::Cell(int x, int y) noexcept: x(x), y(y), status(cellStatus::blank){}
6
7   Cell::Cell(int x, int y, cellStatus status) noexcept: x(x), y(y), status(status) {}
8
9   void Cell::setX(int x) noexcept
10  {
11      this->x=x;
12  }
13
14  void Cell::setY(int y) noexcept
15  {
16      this->y=y;
17  }
18
19  void Cell::setStatus(cellStatus status) noexcept
20  {
21      this->status=status;
22  }
23
24  cellStatus Cell::getStatus() const noexcept
25  {
26      return status;
27  }
28
29  int Cell::getX() const noexcept
30  {
31      return x;
32  }
33
34  int Cell::getY() const noexcept
35  {
36      return y;
37  }
38
39  Cell& Cell::operator =(const Cell& other) noexcept
40  {
41      x=other.getX();
42      y=other.getY();
43      status=other.getStatus();
44      return *this;
45  }
46
47  bool Cell::operator ==(const Cell& other) const noexcept
48  {
49      return (x==other.getX() && y==other.getY() && status==other.getStatus());
50  }
```

Листинг 9: main.cpp

```
1   #include "application.h"
2
3   int main()
4   {
5       Application application;
6       application.mainMenu();
7       return 0;
8   }
```

Листинг 10: menu.cpp

```
#include <iostream>
#include <string>
#include "application.h"

void Application::mainMenu() noexcept
{
    std::cout<<"1. New game"<<std::endl
             <<"2. Exit"<<std::endl
            <<"Select item: ";
    int number;
    std::string str;
    std::cin>>str;
    try
    {
        number=std::stoi(str);
    }
    catch(std::exception &e)
    {
        number=0;
    }
    std::cout<<std::endl;
    switch(number){
    case 1:
        locateShipsMenu();
        break;
    case 2:
        std::exit(0);
    default:
        std::cout<<"Invalid number! Try again"<<std::endl;
        std::cin.clear();
        getline(std::cin,str);
        mainMenu();
        break;
    }
}

void Application::locateShipsMenu() noexcept
{
    std::cout<<"1. Locate ships automatically"<<std::endl
             <<"2. Locate ships on one's own"<<std::endl
            <<"3. Exit"<<std::endl
           <<"Select item: ";
    int number;
    std::string str;
    std::cin>>str;
    try
    {
        number=std::stoi(str);
    }
    catch(std::exception &e)
    {
        number=0;
    }
    std::cout<<std::endl;
    switch(number){
    case 1:
        locateShipsAutomatically();
        break;
    case 2:
        locateShipsOnOnesOwn();
        break;
    case 3:
        exit();
        break;
    default:
        std::cout<<"Invalid number! Try again"<<std::endl;
        std::cin.clear();
        getline(std::cin,str);
        locateShipsMenu();
        break;
    }
}

```

```
74 void Application::startGameMenu() noexcept
75 {
76     std::cout<<"1. Start game"<<std::endl
77              <<"2. Locate ships again"<<std::endl
78              <<"3. Exit"<<std::endl
79             <<"Select item: ";
80     int number;
81     std::string str;
82     std::cin>>str;
83     try
84     {
85         number=std::stoi(str);
86     }
87     catch(std::exception &e)
88     {
89         number=0;
90     }
91     std::cout<<std::endl;
92     switch(number){
93     case 1:
94         startGame();
95         break;
96     case 2:
97         locateShipsAgain();
98         break;
99     case 3:
100        exit();
101        break;
102    default:
103        std::cout<<"Invalid number! Try again"<<std::endl;
104        std::cin.clear();
105        getline(std::cin,str);
106        startGameMenu();
107        break;
108    }
109
110 }
111
112 void Application::locateShipsAutomatically() noexcept
113 {
114     game->placeShipsAutomatically(game->getComputerField());
115     game->placeShipsAutomatically(game->getUserField());
116     printFields();
117     std::cout<<std::endl;
118     startGameMenu();
119 }
120
121 void Application::locateShipsOnOnesOwn() noexcept
122 {
123     game->placeShipsAutomatically(game->getComputerField());
124     printFields();
125     if (locateShipsInput())
126         startGameMenu();
127     else {
128         delete game;
129         game=new GameAPI();
130         locateShipsMenu();
131     }
132 }
133
134 void Application::exit() noexcept
135 {
136     delete game;
137     game=new GameAPI();
138     mainMenu();
139 }
140
141 void Application::startGame() noexcept
142 {
143     gameProcess();
144     delete game;
145     game=new GameAPI();
146     mainMenu();
147 }
148
149 void Application::locateShipsAgain() noexcept
```

```
150 {
151     delete game;
152     game=new GameAPI();
153     locateShipsMenu();
154 }
```

Листинг 11: application.h

```
1  #ifndef APPLICATION_H
2  #define APPLICATION_H
3
4  #include "../Core/game.h"
5
6  /**
7   * @brief Консольное приложение
8   */
9  class Application
10 {
11 public:
12     Application() noexcept;
13     ~Application() noexcept;
14     /**
15      * @brief Вывести главное меню
16      */
17     void mainMenu() noexcept;
18     /**
19      * @brief Выбрать способ расстановки кораблей
20      */
21     void locateShipsMenu() noexcept;
22     /**
23      * @brief Начать игру
24      */
25     void startGameMenu() noexcept;
26     /**
27      * @brief Вывести оба поля на экран
28      */
29     void printFields() noexcept;
30     /**
31      * @brief Вывести поле пользователя
32      */
33     void  printUserFieldLine(int lineNumber) noexcept;
34     /**
35       * @brief Вывести поле компьютера
36       */
37     void printComputerFieldLine(int lineNumber) noexcept;
38     /**
39      * @brief Разместить корабли игрока вручную
40      * @return true если все корабли расставлены
41      * @return false
42      */
43     bool locateShipsInput() noexcept;
44     /**
45      * @brief Узнать, можно ли разместить корабль на данные координаты
46      * @param x буквенная координата первой палубы
47      * @param y цифровая координата первой палубы
48      * @param lenght длина
49      * @param location расположение
50      * @return true/false
51      */
52     bool canPlaceShip(int x, int y, int lenght, int number, shipLocation location) noexcept;
53     /**
54      * @brief Ввод пользователем буквенной координаты
55      * @param str строка ввода
56      * @return буквенная координата
57      */
58     int inputX(std::string str) noexcept;
59     /**
60      * @brief Ввод пользователем цифровой координаты
61      * @param str строка ввода
62      * @return цифровая кооридната
63      */
64     int inputY(std::string str) noexcept;
65     /**
66      * @brief Ввод длины корабля
67      * @param str строка ввода
68      * @return длина
```

```
69          */
70          int inputLenght(std::string str) noexcept;
71          /**
72           * @brief Ввод расположения
73           * @param str строка ввода
74           * @return расположение
75           */
76          int inputLocation(std::string str) noexcept;
77          /**
78           * @brief Играть, пока все корабли одного из полей не будут разрушены
79           */
80          void gameProcess() noexcept;
81          /**
82           * @brief Команды пользователя
83           * @param str строка ввода
84           */
85          void commands(std::string str) noexcept;
86          /**
87           * @brief Определить победителя
88           */
89          void decideWinner() noexcept;
90          /**
91           * @brief Проверить корректность введенных координат
92           * @param x буквенная координата
93           * @param y цифровая координата
94           * @return true/false
95           */
96          bool isCoordinatesCorrect(int x, int y) noexcept;
97          /**
98           * @brief Проверить корректность введенной длины
99           * @param lenght длина
100          * @return true/false
101          */
102         bool isLenghtCorrect(int lenght) noexcept;
103         /**
104          * @brief Проверить корректность расположения
105          * @param location расположение
106          * @return true/false
107          */
108         bool isLocationCorrect(int location) noexcept;
109         /**
110          * @brief Функция, вызываемая при попадании пользователя
111          * Проверяет, разрушен ли корабль, в который попали
112          * @param x буквенная координата
113          * @param y цифровая координата
114          */
115         void ifUserHit(int x, int y) noexcept;
116         /**
117          * @brief Функция, вывзваемая при попадании компьютера
118          */
119         void ifComputerHit() noexcept;
120         /**
121          * @brief Разместить корабли пользователя автоматически
122          */
123         void locateShipsAutomatically() noexcept;
124         /**
125          * @brief Разместить корабли пользователя вручную
126          */
127         void locateShipsOnOnesOwn() noexcept;
128         /**
129          * @brief Выйти в главное меню
130          */
131         void exit() noexcept;
132         /**
133          * @brief Начать игровой процесс
134          */
135         void startGame() noexcept;
136         /**
137          * @brief Разместить корабли заново
138          */
139         void locateShipsAgain() noexcept;
140
141 private:
142         //TODO документировать данные класса
143         GameAPI* game;
144
```

```
145 };
146
147 #endif // APPLICATION_H
```

Листинг 12: application.cpp

```
1  #include <iostream>
2  #include <string>
3  #include "application.h"
4
5
6  Application::Application() noexcept
7  {
8      game = new GameAPI();
9  }
10
11 Application::~Application() noexcept
12 {
13     delete game;
14 }
15
16 void Application::printFields() noexcept
17 {
18     std::cout<<"␣␣␣"<<"A␣"<<"B␣"<<"C␣"<<"D␣"<<"E␣"<<"F␣"<<"G␣"<<"H␣"<<"I␣"<<"J"
19             <<"␣␣␣␣␣␣␣␣␣"<<"A␣"<<"B␣"<<"C␣"<<"D␣"<<"E␣"<<"F␣"<<"G␣"<<"H␣"<<"I␣"<<"J"<<std::
    ↪ endl;
20     for (int i=0;i<Field::FIELD_SIZE;i++)
21     {
22         printUserFieldLine(i);
23         printComputerFieldLine(i);
24
25     }
26
27 }
28
29 void Application::printUserFieldLine(int lineNumber) noexcept
30 {
31     if (lineNumber+1==10)
32         std::cout<<lineNumber+1<<"␣";
33     else std::cout<<lineNumber+1<<"␣␣";
34     for (int j=0;j<Field::FIELD_SIZE;j++)
35     {
36         if (game->getUserField()->getFieldCells()[lineNumber][j].getStatus()==cellStatus::
    ↪ whole)
37             std::cout<<"O"<<"␣";
38         if (game->getUserField()->getFieldCells()[lineNumber][j].getStatus()==cellStatus::
    ↪ stricken)
39             std::cout<<"X"<<"␣";
40         if (game->getUserField()->getFieldCells()[lineNumber][j].getStatus()==cellStatus::
    ↪ blank)
41             std::cout<<"."<<"␣";
42         if (game->getUserField()->getFieldCells()[lineNumber][j].getStatus()==cellStatus::
    ↪ tried)
43             std::cout<<"_"<<"␣";
44     }
45     std::cout<<"␣␣␣␣␣";
46 }
47
48 void Application::printComputerFieldLine(int lineNumber) noexcept
49 {
50     if (lineNumber+1==10)
51         std::cout<<lineNumber+1<<"␣";
52     else std::cout<<lineNumber+1<<"␣␣";
53     for (int j=0;j<Field::FIELD_SIZE;j++)
54     {
55         if (game->getComputerField()->getFieldCells()[lineNumber][j].getStatus()==cellStatus::
    ↪ whole)
56             std::cout<<"*"<<"␣";
57         if (game->getComputerField()->getFieldCells()[lineNumber][j].getStatus()==cellStatus::
    ↪ stricken)
58             std::cout<<"X"<<"␣";
59         if (game->getComputerField()->getFieldCells()[lineNumber][j].getStatus()==cellStatus::
    ↪ blank)
60             std::cout<<"*"<<"␣";
61         if (game->getComputerField()->getFieldCells()[lineNumber][j].getStatus()==cellStatus::
    ↪ tried)
```

```
62              std::cout<<".";
63      }
64      std::cout<<std::endl;
65  }
66
67  bool Application::locateShipsInput() noexcept
68  {
69      std::cout<<"Locate ships"<<std::endl;
70      int x, y, lenght, number;
71      shipLocation location;
72      std::string str;
73
74      while (!game->getUserField()->allShipsLocate())
75      {
76          std::cout<<"Coordinates of the first deck: "<<std::endl;
77          std::cin>>str;
78          commands(str);
79          x=inputX(str);
80          y=inputY(str);
81          if (!isCoordinatesCorrect(x,y))
82              continue;
83
84          std::cout<<"Lenght of the ship: "<<std::endl;
85          std::cin>>str;
86          commands(str);
87          lenght=inputLenght(str);
88          if (!isLenghtCorrect(lenght))
89              continue;
90
91          std::cout<<"Ship location(vertical-1,horizontal-2): "<<std::endl;
92          std::cin>>str;
93          commands(str);
94          number=inputLocation(str);
95          if (number==1)
96              location=shipLocation::vertical;
97          if (number==2)
98              location=shipLocation::horizontal;
99          if (!isLocationCorrect(number))
100             continue;
101
102         std::cin.clear();
103         getline(std::cin,str);
104
105         if (!canPlaceShip(x, y, lenght, number, location))
106             continue;
107         game->placeUserShip(x, y, lenght, location);
108         printFields();
109
110     }
111     std::cout<<"All ships are placed!"<<std::endl<<std::endl;
112     return true;
113 }
114
115 int Application::inputX(std::string str) noexcept
116 {
117     char charX;
118     int x;
119     charX=str[0];
120     if (charX<96)
121         x=charX-64;
122     else
123         x=charX-96;
124     return x-1;
125 }
126
127 int Application::inputY(std::string str) noexcept
128 {
129     int y;
130     if(str.length()==2)
131         y=str[1]-'0';
132     else y=(str[1]-'0')*10+(str[2]-'0');
133     return y-1;
134 }
135
136 int Application::inputLenght(std::string str) noexcept
137 {
```

```
138     int lenght;
139     try
140     {
141         lenght=std::stoi(str);
142     }
143     catch(std::exception &e)
144     {
145         lenght=0;
146     }
147     return lenght;
148 }
149
150 int Application::inputLocation(std::string str) noexcept
151 {
152     int location;
153     try
154     {
155         location=std::stoi(str);
156     }
157     catch(std::exception &e)
158     {
159         location=0;
160     }
161     return location;
162 }
163
164 void Application::gameProcess() noexcept
165 {
166     while (!game->allShipsDestroyed(game->getComputerField()) &&
167            !game->allShipsDestroyed(game->getUserField()))
168     {
169         std::cout<<"Make move: ";
170         int x,y;
171         std::string str;
172         std::cin>>str;
173         commands(str);
174         x=inputX(str);
175         y=inputY(str);
176         if (!isCoordinatesCorrect(x, y))
177             continue;
178         std::cin.clear();
179         getline(std::cin,str);
180
181         if (game->makeUserMove(x ,y))
182         {
183             ifUserHit(x, y);
184             continue;
185
186         }
187         else
188         {
189             printFields();
190             std::cout<<"You miss!"<<std::endl<<std::endl;
191         }
192
193         while (game->makeComputerMove() && !game->allShipsDestroyed(game->getUserField()))
194         {
195             ifComputerHit();
196         }
197
198         printFields();
199         std::cout<<"Computer miss!"<<std::endl<<std::endl;
200
201     }
202 }
203
204 void Application::decideWinner() noexcept
205 {
206     if (game->allShipsDestroyed(game->getComputerField()))
207         std::cout<<"You won!"<<std::endl<<std::endl;
208     if (game->allShipsDestroyed(game->getUserField()))
209         std::cout<<"You lost!"<<std::endl<<std::endl;
210     delete game;
211     game=new GameAPI();
212     mainMenu();
213 }
```

```
214
215 void Application::commands(std::string str) noexcept
216 {
217     if (str=="exit")
218     {
219         std::cout<<std::endl;
220         delete game;
221         game=new GameAPI();
222         mainMenu();
223     }
224 }
225
226 bool Application::canPlaceShip(int x, int y, int lenght, int number, shipLocation location)
    ↪ noexcept
227 {
228
229     if ((number==2 && x+lenght>Field::FIELD_SIZE) || (number==1 && y+lenght>Field::FIELD_SIZE)
    ↪ )
230     {
231         std::cout<<"Error! It is impossible to place the ship"<<std::endl;
232         return false;
233     }
234     if (!game->getUserField()->canPlaceShip(x,y,lenght,location))
235     {
236         std::cout<<"Error! Ships must not touch"<<std::endl;
237         return false;
238     }
239
240     static int count1Deck=0;
241     static int count2Deck=0;
242     static int count3Deck=0;
243     static int count4Deck=0;
244
245     if (lenght==1)
246     {
247         count1Deck++;
248         if (count1Deck>4){
249             std::cout<<"Error! You can place only four 1-deck ships"<<std::endl;
250             return false;
251         }
252     }
253
254     if (lenght==2)
255     {
256         count2Deck++;
257         if (count2Deck>3){
258             std::cout<<"Error! You can place only three 2-deck ships"<<std::endl;
259             return false;
260         }
261     }
262
263     if (lenght==3)
264     {
265         count3Deck++;
266         if(count3Deck>2){
267             std::cout<<"Error! You can place only two 3-deck ships"<<std::endl;
268             return false;
269         }
270     }
271
272     if (lenght==4)
273     {
274         count4Deck++;
275         if (count4Deck>1){
276             std::cout<<"Error! You can place only one 4-deck ship"<<std::endl;
277             return false;
278         }
279     }
280
281     return true;
282 }
283
284 bool Application::isCoordinatesCorrect(int x, int y) noexcept
285 {
286     if(x>Field::FIELD_SIZE || x<0 || y>Field::FIELD_SIZE || y<0)
287     {
```

```
288         std::cout<<"Error! Wrong coordinates! Use letter a..j and number 1..10"<<std::endl;
289         return false;
290     }
291     return true;
292 
293 }
294 
295 bool Application::isLenghtCorrect(int lenght) noexcept
296 {
297     if (lenght<1 || lenght>4)
298     {
299         std::cout<<"Error! Wrong lenght!"<<std::endl;
300         return false;
301     }
302     return true;
303 }
304 
305 bool Application::isLocationCorrect(int location) noexcept
306 {
307     if (location<1 || location>2)
308     {
309         std::cout<<"Error! Wrong input! Use 1 or 2"<<std::endl;
310         return false;
311     }
312     return true;
313 }
314 
315 void Application::ifUserHit(int x, int y) noexcept
316 {
317     int shipNumber;
318     shipNumber=game->getComputerField()->whoseDeck(x, y);
319     if (game->getComputerField()->isShipDestroyed(shipNumber))
320     {
321         game->getComputerField()->changeCellsAroundShip(game->getComputerField()->
    ↪ getFieldShips()[shipNumber]);
322         printFields();
323         std::cout<<"You hit!"<<std::endl;
324         std::cout<<"Ship destroyed!"<<std::endl;
325 
326     }
327     else
328     {
329         printFields();
330         std::cout<<"You hit!"<<std::endl;
331     }
332     if (game->allShipsDestroyed(game->getComputerField()))
333         decideWinner();
334 
335 }
336 
337 void Application::ifComputerHit() noexcept
338 {
339     printFields();
340     std::cout<<"Computer hit!"<<std::endl<<std::endl;
341     if (game->allShipsDestroyed(game->getUserField()))
342         decideWinner();
343 }
```

Листинг 13: main.cpp

```
1  #include "mainwindow.h"
2  #include <QApplication>
3  
4  int main(int argc, char *argv[])
5  {
6      QApplication a(argc, argv);
7      MainWindow w;
8      w.show();
9  
10     return a.exec();
11 }
```

Листинг 14: mainwindow.h

```
1  #ifndef MAINWINDOW_H
```

```
2  #define MAINWINDOW_H
3
4  #include <QWidget>
5  #include <QPushButton>
6  #include <QString>
7  #include <QFont>
8  #include "gamewindow.h"
9
10 class MainWindow : public QWidget
11 {
12     Q_OBJECT
13
14 public:
15     explicit MainWindow(QWidget *parent = 0);
16
17 private:
18
19     const QSize WINDOW_SIZE{900,440};
20     const QSize BUTTON_SIZE{150,60};
21
22     QPushButton* buttonExit;
23     QPushButton* buttonStart;
24
25
26     QString buttonStyle=
27             "QPushButton"
28
29             "{"
30                 "border: 1.5px solid rgb(0, 0, 0);"
31                 "border-radius: 10px;"
32
33                 "background-color: qlineargradient(spread:pad, x1:0, y1:0, x2:1, y2:1, stop:0
    ↪ rgba(128, 128, 128, 120), stop:1 rgba(225, 225, 225, 120));"
34                 "color: rgb(25, 25, 25);"
35
36             "}"
37             "QPushButton:pressed"
38             "{"
39                 "background-color: qlineargradient(spread:pad, x1:0, y1:0, x2:1, y2:1, stop:0
    ↪ rgb(128, 128, 128), stop:1 rgb(225, 225, 225));"
40             "}";
41
42 private slots:
43
44     void exit();
45     void start();
46
47 };
48
49 #endif // MAINWINDOW_H
```

Листинг 15: mainwindow.cpp

```
1  #include "mainwindow.h"
2
3  MainWindow::MainWindow(QWidget *parent) : QWidget(parent)
4
5  {
6      this->setFixedSize(WINDOW_SIZE);
7      this->setWindowTitle("Sea Battle");
8
9      QPixmap background(":/main_window_background.png");
10     QPalette palette;
11     palette.setBrush(backgroundRole(),QBrush(background));
12     this->setPalette(palette);
13
14     QFont font;
15     font.setFamily("inpact");
16     font.setPointSize(25);
17     font.setWeight(75);
18
19     buttonStart=new QPushButton("Start",this);
20     buttonStart->setFont(font);
21     buttonStart->setStyleSheet(buttonStyle);
22     buttonStart->resize(BUTTON_SIZE);
23     buttonStart->move(370,170);
```

```
24
25      buttonExit=new QPushButton("Exit",this);
26      buttonExit->setFont(font);
27      buttonExit->setStyleSheet(buttonStyle);
28      buttonExit->resize(BUTTON_SIZE);
29      buttonExit->move(370,250);
30
31      connect(buttonStart, SIGNAL(clicked()), this, SLOT(start()));
32      connect(buttonExit, SIGNAL(clicked()), this, SLOT(exit()));
33
34  }
35
36   void MainWindow::exit()
37  {
38      this->close();
39  }
40
41   void MainWindow::start()
42   {
43       GameWindow* gameWindow=new GameWindow(0);
44       gameWindow->show();
45       this->exit();
46
47   }
```

Листинг 16: gamewindow.h

```
1  #ifndef GAMEWINDOW_H
2  #define GAMEWINDOW_H
3
4  #include <QtWidgets>
5
6  #include "mainwindow.h"
7  #include "resultwindow.h"
8  #include "game.h"
9
10 class GameWindow : public QWidget
11 {
12     Q_OBJECT
13
14 public:
15     explicit GameWindow(QWidget *parent);
16
17 private:
18     const QSize WINDOW_SIZE{900,440};
19     const QSize BUTTON_SIZE{120,45};
20
21     const QPoint USER_FIELD_COORD{80,50};
22     const QRect USER_FIELD{USER_FIELD_COORD, QPoint(USER_FIELD_COORD.x()+FIELD_SIZE.width()
       ↪ -30, USER_FIELD_COORD.y()+FIELD_SIZE.height()-30)};
23     const QSize FIELD_SIZE{350, 350};
24     const QPoint COMPUTER_FIELD_COORD{520,50};
25     const QRect COMPUTER_FIELD{COMPUTER_FIELD_COORD, QPoint(COMPUTER_FIELD_COORD.x()+
       ↪ FIELD_SIZE.width()-50, COMPUTER_FIELD_COORD.y()+FIELD_SIZE.height()-50)};
26
27
28     QPushButton* buttonExit;
29
30     GameAPI* game;
31
32     QString buttonStyle=
33             "QPushButton"
34
35             "{"
36                 "border: 1.5px solid rgb(0, 0, 0);"
37                 "border-radius: 10px;"
38
39                 "background-color: qlineargradient(spread:pad, x1:0, y1:0, x2:1, y2:1, stop:0
       ↪ rgba(128, 128, 128, 160), stop:1 rgba(225, 225, 225, 160));"
40                 "color: rgb(25, 25, 25);"
41
42             "}"
43             "QPushButton:pressed"
44             "{"
45                 "background-color: qlineargradient(spread:pad, x1:0, y1:0, x2:1, y2:1, stop:0
       ↪ rgb(128, 128, 128), stop:1 rgb(225, 225, 225));"
```

```
46              "}";
47
48      void drawUserField(QPainter& painter);
49      void drawComputerField(QPainter& painter);
50      void drawFrame(QPainter& painter, const QPoint coord);
51      void drawDeck(QPainter& painter, QPoint coord);
52      void drawPoint(QPainter& painter, QPoint coord);
53      void drawCross(QPainter&painter, QPoint coord);
54      void drawFog(QPainter& painter, QPoint coord);
55      void paintEvent(QPaintEvent*);
56      void mousePressEvent(QMouseEvent* event);
57      bool onComputerFieldClicked(QPoint coord);
58      void computerMove();
59      void decideWinner();
60
61
62
63  private slots:
64
65      void exit();
66
67
68
69  };
70
71  #endif // GAMEWINDOW_H
```

Листинг 17: gamewindow.cpp

```
1   #include "gamewindow.h"
2
3   GameWindow::GameWindow(QWidget *parent) : QWidget(parent)
4   {
5       game=new GameAPI();
6       game->getComputerField()->locateAutomatically();
7       game->getUserField()->locateAutomatically();
8
9
10      this->setFixedSize(WINDOW_SIZE);
11      this->setWindowTitle("Sea Battle");
12
13      QPixmap background(":/game_window_background.jpg");
14      QPalette palette;
15      palette.setBrush(backgroundRole(),QBrush(background));
16      this->setPalette(palette);
17
18      QFont font;
19      font.setFamily("inpact");
20      font.setPointSize(25);
21      font.setWeight(75);
22
23      buttonExit=new QPushButton("Exit",this);
24      buttonExit->setFont(font);
25      buttonExit->setStyleSheet(buttonStyle);
26      buttonExit->resize(BUTTON_SIZE);
27      buttonExit->move(770,390);
28
29      connect(buttonExit, SIGNAL(clicked()), this, SLOT(exit()));
30
31  }
32
33  void GameWindow::exit()
34  {
35      MainWindow* mainwindow=new MainWindow();
36      mainwindow->show();
37      this->close();
38  }
39
40  void GameWindow::paintEvent(QPaintEvent *)
41  {
42      QPainter painter(this);
43
44      drawFrame(painter, QPoint(USER_FIELD_COORD.x()-30, USER_FIELD_COORD.y()-30));
45      drawFrame(painter, QPoint(COMPUTER_FIELD_COORD.x()-30, COMPUTER_FIELD_COORD.y()-30));
46
47      drawUserField(painter);
```

```
48      drawComputerField(painter);
49  }
50
51  void GameWindow::drawComputerField(QPainter &painter)
52  {
53      painter.drawImage(QPoint(COMPUTER_FIELD_COORD.x()-30,COMPUTER_FIELD_COORD.y()-30), QImage(
        ↪ ":/field.jpg"));
54
55      for(int i=0; i<Field::FIELD_SIZE; i++)
56          for(int j=0; j<Field::FIELD_SIZE; j++)
57          {
58              if(game->getComputerField()->getFieldCells()[i][j].getStatus()==cellStatus::blank)
59                  drawFog(painter, QPoint(COMPUTER_FIELD_COORD.x()+30*j, COMPUTER_FIELD_COORD.y
        ↪ ()+30*i));
60              if(game->getComputerField()->getFieldCells()[i][j].getStatus()==cellStatus::whole)
61                  drawFog(painter, QPoint(COMPUTER_FIELD_COORD.x()+30*j, COMPUTER_FIELD_COORD.y
        ↪ ()+30*i));
62              if(game->getComputerField()->getFieldCells()[i][j].getStatus()==cellStatus::
        ↪ stricken){
63                  drawDeck(painter, QPoint(COMPUTER_FIELD_COORD.x()+30*j, COMPUTER_FIELD_COORD.y
        ↪ ()+30*i));
64                  drawCross(painter, QPoint(COMPUTER_FIELD_COORD.x()+30*j, COMPUTER_FIELD_COORD.
        ↪ y()+30*i));
65              }
66              if(game->getComputerField()->getFieldCells()[i][j].getStatus()==cellStatus::tried)
67                  drawPoint(painter, QPoint(COMPUTER_FIELD_COORD.x()+30*j, COMPUTER_FIELD_COORD.
        ↪ y()+30*i));
68          }
69  }
70
71  void GameWindow::drawUserField(QPainter &painter)
72  {
73      painter.drawImage(QPoint(USER_FIELD_COORD.x()-30,USER_FIELD_COORD.y()-30), QImage(":/field
        ↪ .jpg"));
74
75      for (int i=0; i<Field::FIELD_SIZE; i++)
76          for (int j=0; j<Field::FIELD_SIZE; j++)
77          {
78              if(game->getUserField()->getFieldCells()[i][j].getStatus()==cellStatus::whole)
79                  drawDeck(painter, QPoint(USER_FIELD_COORD.x()+30*j, USER_FIELD_COORD.y()+30*i)
        ↪ );
80              if(game->getUserField()->getFieldCells()[i][j].getStatus()==cellStatus::stricken){
81                  drawDeck(painter, QPoint(USER_FIELD_COORD.x()+30*j, USER_FIELD_COORD.y()+30*i)
        ↪ );
82                  drawCross(painter, QPoint(USER_FIELD_COORD.x()+30*j, USER_FIELD_COORD.y()+30*i
        ↪ ));
83              }
84              if(game->getUserField()->getFieldCells()[i][j].getStatus()==cellStatus::tried)
85                  drawPoint(painter, QPoint(USER_FIELD_COORD.x()+30*j, USER_FIELD_COORD.y()+30*i
        ↪ ));
86          }
87  }
88
89  void GameWindow::drawFrame(QPainter &painter, const QPoint coord)
90  {
91      QPen pen;
92      pen.setWidth(3);
93      painter.setPen(pen);
94
95      painter.drawLine(coord, QPoint(coord.x()+360,coord.y()));
96      painter.drawLine(coord, QPoint(coord.x(), coord.y()+360));
97      painter.drawLine((QPoint(coord.x(), coord.y()+360)), QPoint(coord.x()+360, coord.y()+360))
        ↪ ;
98      painter.drawLine(QPoint(coord.x()+360, coord.y()), QPoint(coord.x()+360, coord.y()+360));
99  }
100
101 void GameWindow::drawFog(QPainter &painter, QPoint coord)
102 {
103     painter.drawImage(QPoint(coord.x(), coord.y()), QImage(":/fog.jpg"));
104 }
105
106 void GameWindow::drawDeck(QPainter &painter, QPoint coord)
107 {
108     QBrush brush(QColor(49, 58, 91), Qt::SolidPattern);
109     painter.fillRect(coord.x(), coord.y(), 29, 29, brush);
110 }
```

```
111
112 void GameWindow::drawCross(QPainter &painter, QPoint coord)
113 {
114     QPen pen;
115     pen.setColor(Qt::red);
116     pen.setWidth(3);
117
118     painter.setPen(pen);
119     painter.drawLine(QPoint(coord.x()+3, coord.y()+3), QPoint(coord.x()+27, coord.y()+27));
120     painter.drawLine(QPoint(coord.x()+27, coord.y()+3), QPoint(coord.x()+3, coord.y()+27));
121 }
122
123 void GameWindow::drawPoint(QPainter &painter, QPoint coord)
124 {
125     QPen pen;
126     pen.setWidth(8);
127     pen.setCapStyle(Qt::RoundCap);
128     painter.setPen(pen);
129
130     painter.drawPoint(QPoint(coord.x()+15, coord.y()+15));
131 }
132
133 void GameWindow::mousePressEvent(QMouseEvent *event)
134 {
135     if(COMPUTER_FIELD.contains(event->pos()))
136     {
137         if(onComputerFieldClicked(event->pos()) && !game->allShipsDestroyed(game->
    ↪ getComputerField()))
138             update();
139         else{
140             if(game->allShipsDestroyed(game->getComputerField()))
141                 decideWinner();
142             else computerMove();
143         }
144
145     }
146 }
147
148
149 void GameWindow::computerMove()
150 {
151     while(game->makeComputerMove() && !game->allShipsDestroyed(game->getUserField()))
152         update();
153     if (game->allShipsDestroyed(game->getUserField()))
154         decideWinner();
155     else
156         update();
157
158 }
159
160 bool GameWindow::onComputerFieldClicked(QPoint coord)
161 {
162     int shipNumber;
163     if(game->makeUserMove(int((coord.x()-COMPUTER_FIELD_COORD.x())/30), int ((coord.y()-
    ↪ COMPUTER_FIELD_COORD.y())/30)))
164     {
165         shipNumber=game->getComputerField()->whoseDeck(int((coord.x()-COMPUTER_FIELD_COORD.x()
    ↪ )/30), int ((coord.y()-COMPUTER_FIELD_COORD.y())/30));
166         if (game->getComputerField()->isShipDestroyed(shipNumber))
167             game->getComputerField()->changeCellsAroundShip(game->getComputerField()->
    ↪ getFieldShips()[shipNumber]);
168         return true;
169     }
170     else return false;
171 }
172
173
174 void GameWindow::decideWinner()
175 {
176
177     ResultWindow* resultWindow=new ResultWindow(this, game);
178     resultWindow->show();
179
180
181 }
```

Листинг 18: resultwindow.h

```
#ifndef RESULTWINDOW_H
#define RESULTWINDOW_H

#include <QtWidgets>
#include "mainwindow.h"
#include "gamewindow.h"
#include "game.h"

class ResultWindow : public QDialog
{
    Q_OBJECT
public:
    ResultWindow(QWidget* parent, GameAPI* game);

private:
    const QSize WINDOW_SIZE{500, 300};
    const QSize BUTTON_SIZE{120, 45};

    QPushButton* buttonYes;
    QPushButton* buttonNo;
    QWidget* parent;
    GameAPI* game;
    QLabel* labelWinner;

    QString buttonStyle=
            "QPushButton"

            "{"
                "border: 1.5px solid rgb(0, 0, 0);"
                "border-radius: 10px;"

                "background-color: qlineargradient(spread:pad, x1:0, y1:0, x2:1, y2:1, stop:0 rgba(128, 128, 128, 160), stop:1 rgba(225, 225, 225, 160));"
                "color: rgb(25, 25, 25);"

            "}"
            "QPushButton:pressed"
            "{"
                "background-color: qlineargradient(spread:pad, x1:0, y1:0, x2:1, y2:1, stop:0 rgb(128, 128, 128), stop:1 rgb(225, 225, 225));"
            "}";
private slots:
    void again();
    void exit();
};

#endif // RESULTWINDOW_H
```

Листинг 19: resultwindow.cpp

```
#include "resultwindow.h"

ResultWindow::ResultWindow(QWidget* parent, GameAPI* game): QDialog(parent), parent(parent), game(game)
{
    this->setFixedSize(WINDOW_SIZE);
    this->setWindowTitle("Sea Battle");

    QPixmap background(":/game_window_background.jpg");
    QPalette palette;
    palette.setBrush(backgroundRole(),QBrush(background));
    this->setPalette(palette);

    QFont font1;
    font1.setFamily("inpact");
    font1.setPointSize(25);
    font1.setWeight(75);

    QFont font2;
    font2.setFamily("inpact");
    font2.setPointSize(40);
    font2.setWeight(75);

    QLabel* labelWinner=new QLabel(this);
```

```
24
25      if(game->allShipsDestroyed(game->getComputerField()))
26      {
27      labelWinner->setFont(font2);
28      labelWinner->setText("You won!");
29      labelWinner->move(120, 50);
30      }
31
32      if(game->allShipsDestroyed(game->getUserField()))
33      {
34      labelWinner->setFont(font2);
35      labelWinner->setText("You lost!");
36      labelWinner->move(120, 50);
37      }
38
39      QLabel* labelAgain=new QLabel(this);
40      labelAgain->setFont(font1);
41      labelAgain->setText("Do you want to play again?");
42      labelAgain->move(30, 120);
43      labelAgain->show();
44
45      buttonYes=new QPushButton("Yes", this);
46      buttonYes->setFont(font1);
47      buttonYes->setStyleSheet(buttonStyle);
48      buttonYes->resize(BUTTON_SIZE);
49      buttonYes->move(100, 200);
50
51      buttonNo=new QPushButton("No", this);
52      buttonNo->setFont(font1);
53      buttonNo->setStyleSheet(buttonStyle);
54      buttonNo->resize(BUTTON_SIZE);
55      buttonNo->move(250, 200);
56
57      connect(buttonNo, SIGNAL(clicked()), this, SLOT(exit()));
58      connect(buttonYes, SIGNAL(clicked()), this, SLOT(again()));
59
60  }
61
62  void ResultWindow::exit()
63  {
64      MainWindow* mainwindow=new MainWindow();
65      mainwindow->show();
66      this->close();
67      this->parent->close();
68  }
69
70  void ResultWindow::again()
71  {
72      GameWindow* gameWindow=new GameWindow(0);
73      gameWindow->show();
74      this->close();
75      this->parent->close();
76  }
```

7 Приложение 2

Sea Battle

Создано системой Doxygen 1.8.11

Оглавление

Глава 1

Иерархический список классов

1.1 Иерархия классов

Иерархия классов.

Глава 2

Алфавитный указатель классов

2.1 Классы

Классы с их кратким описанием.

Глава 3

Классы

3.1 Класс Application

Консольное приложение

#include <application.h>

Открытые члены

- void mainMenu () noexcept
  Вывести главное меню
- void locateShipsMenu () noexcept
  Выбрать способ расстановки кораблей
- void startGameMenu () noexcept
  Начать игру
- void printFields () noexcept
  Вывести оба поля на экран
- void printUserFieldLine (int lineNumber) noexcept
  Вывести поле пользователя
- void printComputerFieldLine (int lineNumber) noexcept
  Вывести поле компьютера
- bool locateShipsInput () noexcept
  Разместить корабли игрока вручную
- bool canPlaceShip (int x, int y, int lenght, int number, shipLocation location) noexcept
  Узнать, можно ли разместить корабль на данные координаты
- int inputX (std::string str) noexcept
  Ввод пользователем буквенной координаты
- int inputY (std::string str) noexcept
  Ввод пользователем цифровой координаты
- int inputLenght (std::string str) noexcept
  Ввод длины корабля
- int inputLocation (std::string str) noexcept
  Ввод расположения
- void gameProcess () noexcept
  Играть, пока все корабли одного из полей не будут разрушены

- void commands (std::string str) noexcept

  Команды пользователя
- void decideWinner () noexcept

  Определить победителя
- bool isCoordinatesCorrect (int x, int y) noexcept

  Проверить корректность введенных координат
- bool isLenghtCorrect (int lenght) noexcept

  Проверить корректность введенной длины
- bool isLocationCorrect (int location) noexcept

  Проверить корректность расположения
- void ifUserHit (int x, int y) noexcept

  Функция, вызываемая при попадании пользователя Проверяет, разрушен ли корабль, в который попали
- void ifComputerHit () noexcept

  Функция, вывзваемая при попадании компьютера
- void locateShipsAutomatically () noexcept

  Разместить корабли пользователя автоматически
- void locateShipsOnOnesOwn () noexcept

  Разместить корабли пользователя вручную
- void exit () noexcept

  Выйти в главное меню
- void startGame () noexcept

  Начать игровой процесс
- void locateShipsAgain () noexcept

  Разместить корабли заново

3.1.1 Подробное описание

Консольное приложение

3.1.2 Методы

3.1.2.1 bool Application::canPlaceShip (int x, int y, int lenght, int number, shipLocation location) [noexcept]

Узнать, можно ли разместить корабль на данные координаты

Аргументы

x	буквенная координата первой палубы
y	цифровая координата первой палубы
lenght	длина
location	расположение

Возвращает

true/false

3.1.2.2 void Application::commands (std::string str) [noexcept]

Команды пользователя

Аргументы

str	строка ввода

3.1.2.3 void Application::ifUserHit (int x, int y) [noexcept]

Функция, вызываемая при попадании пользователя Проверяет, разрушен ли корабль, в который попали

Аргументы

x	буквенная координата
y	цифровая координата

3.1.2.4 int Application::inputLenght (std::string str) [noexcept]

Ввод длины корабля

Аргументы

str	строка ввода

Возвращает

длина

3.1.2.5 int Application::inputLocation (std::string str) [noexcept]

Ввод расположения

Аргументы

str	строка ввода

Возвращает

расположение

3.1.2.6 int Application::inputX (std::string str) [noexcept]

Ввод пользователем буквенной координаты

Аргументы

str	строка ввода

Возвращает

буквенная координата

3.1.2.7 int Application::inputY (std::string str) [noexcept]

Ввод пользователем цифровой координаты

Аргументы

str	строка ввода

Возвращает

цифровая коориднаата

3.1.2.8 bool Application::isCoordinatesCorrect (int x, int y) [noexcept]

Проверить корректность введенных координат

Аргументы

x	буквенная координата
y	цифровая координата

Возвращает

true/false

3.1.2.9 bool Application::isLenghtCorrect (int lenght) [noexcept]

Проверить корректность введенной длины

Аргументы

lenght	длина

Возвращает

true/false

3.1.2.10 bool Application::isLocationCorrect (int location) [noexcept]

Проверить корректность расположения

Аргументы

location	расположение

Возвращает

true/false

3.1.2.11 bool Application::locateShipsInput ( ) [noexcept]

Разместить корабли игрока вручную

Возвращает

true если все корабли расставлены
false

Объявления и описания членов классов находятся в файлах:

- sources/Console/application.h
- sources/Console/application.cpp
- sources/Console/menu.cpp

3.2 Класс Cell

Клетка игрового поля

#include <cell.h>

Открытые члены

- Cell (int x, int y) noexcept
- Cell (int x, int y, cellStatus status) noexcept
- Cell & operator= (const Cell &other) noexcept
- bool operator== (const Cell &other) const noexcept
- void setX (int x) noexcept
  Установить буквенную координату клетки
- void setY (int y) noexcept
  Установить цифровую координату клетки
- void setStatus (cellStatus status) noexcept
  Установить статус клетки
- cellStatus getStatus () const noexcept
  Получить статус клетки
- int getX () const noexcept
  Получить буквенную координату
- int getY () const noexcept
  Получить цифровую координату

3.2.1 Подробное описание

Клетка игрового поля

3.2.2 Методы

3.2.2.1 cellStatus Cell::getStatus ( ) const [noexcept]

Получить статус клетки

Возвращает

статус

3.2.2.2 int Cell::getX ( ) const [noexcept]

Получить буквенную координату

Возвращает

буквенная координата

3.2.2.3 int Cell::getY ( ) const [noexcept]

Получить цифровую координату

Возвращает

цифровая координата

3.2.2.4 void Cell::setStatus (cellStatus status) [noexcept]

Установить статус клетки

Аргументы

status	статус

3.2.2.5 void Cell::setX (int x) [noexcept]

Установить буквенную координату клетки

Аргументы

x	буквенная координата

3.2.2.6 void Cell::setY (int y) [noexcept]

Установить цифровую координату клетки

Аргументы

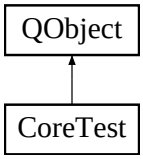
y	цифровая координата

Объявления и описания членов классов находятся в файлах:

- sources/Core/cell.h
- sources/Core/cell.cpp

3.3 Класс CoreTest

Граф наследования:CoreTest:



Объявления и описания членов класса находятся в файле:

- sources/Test/tst_coretest.cpp

3.4 Класс Field

Игровое поле

#include <field.h>

Открытые члены

- void placeShip (int x, int y, int lenght, shipLocation location) noexcept

  Разместить корабль
- bool allShipsLocate () noexcept

  Узнать, все ли корабли размещены на поле
- bool isDeck (int x, int y) noexcept

  Узнать, является ли клетка палубой
- bool shot (int x, int y) noexcept

  Выстрелить в клетку игрового поля Изменяет статус клетки
- bool canPlaceShip (int x, int y, int lenght, shipLocation location) noexcept

  Узнать, можно ли разместить корабль на данные координаты
- std::vector< std::vector< Cell > > getFieldCells () noexcept

  Получить клетки игрового поля
- std::vector< Ship > getFieldShips () noexcept

  Получить корабли поля
- int getnumberSetShips () const noexcept

  Узнать число размещенных кораблей
- void locateShipRandomly (int lenght) noexcept

  Рандомно разместить один корабль
- void locateAutomatically () noexcept

  Рандомно разместить все корабли
- void changeCellsAroundShip (Ship ship) noexcept

  Изменить статус клеток вокруг разрушенного корабля
- bool isShipDestroyed (int shipNumber) noexcept

  Узнать, разрушен ли данный корабль
- int whoseDeck (int x, int y) noexcept

  Узнать, какому кораблю принадлежит палуба

Статические открытые данные

- static const int FIELD_SIZE = 10
- static const int NUMBER_OF_SHIPS = 10

3.4.1 Подробное описание

Игровое поле

3.4.2 Методы

3.4.2.1 bool Field::allShipsLocate ( ) [noexcept]

Узнать, все ли корабли размещены на поле

Возвращает

true/false

3.4.2.2 bool Field::canPlaceShip (int x, int y, int lenght, shipLocation location) [noexcept]

Узнать, можно ли разместить корабль на данные координаты

Аргументы

x	буквенная координата первой палубы
y	цифровая координата первой палубы
lenght	длина
location	расположение

Возвращает

true/false

3.4.2.3 void Field::changeCellsAroundShip (Ship ship) [noexcept]

Изменить статус клеток вокруг разрушенного корабля

Аргументы

ship	корабль

3.4.2.4 std::vector< std::vector< Cell > > Field::getFieldCells ( ) [noexcept]

Получить клетки игрового поля

Возвращает

указатель на клетки поля

3.4.2.5 std::vector< Ship > Field::getFieldShips ( ) [noexcept]

Получить корабли поля

Возвращает

указатель на массив кораблей

3.4.2.6 int Field::getnumberSetShips ( ) const [noexcept]

Узнать число размещенных кораблей

Возвращает

количество размещенных кораблей

3.4.2.7 bool Field::isDeck (int x, int y) [noexcept]

Узнать, является ли клетка палубой

Аргументы

x	буквенная координата
y	цифровая координата

Возвращает

true/false

3.4.2.8 bool Field::isShipDestroyed (int shipNumber) [noexcept]

Узнать, разрушен ли данный корабль

Аргументы

shipNumber	номер корабля

Возвращает

true/false

3.4.2.9 void Field::locateShipRandomly (int lenght) [noexcept]

Рандомно разместить один корабль

Аргументы

lenght	длина корабля

3.4.2.10 void Field::placeShip (int x, int y, int lenght, shipLocation location) [noexcept]

Разместить корабль

Аргументы

x	буквенная координата первой палубы корабля
y	цифровая координата первой палубы корабля
lenght	длина корабля
location	расположение

3.4.2.11 bool Field::shot (int x, int y) [noexcept]

Выстрелить в клетку игрового поля Изменяет статус клетки

Аргументы

x	буквенная координата
y	цифровая координата

Возвращает

true при попадании в палубу
false при промахе

3.4.2.12 int Field::whoseDeck (int x, int y) [noexcept]

Узнать, какому кораблю принадлежит палуба

Аргументы

x	буквенная координата
y	цифровая координата

Возвращает

номер корабля

Объявления и описания членов классов находятся в файлах:

- sources/Core/field.h
- sources/Core/field.cpp

3.5 Класс GameAPI

Игровой интерфейс

#include <game.h>

Открытые члены

- bool makeComputerMove () noexcept
  Совершить ход компьютера
- bool makeUserMove (int x, int y) noexcept
  Совершить ход игрока
- Field * getUserField () const noexcept
  Получить поле игрока
- Field * getComputerField () const noexcept
  Получить поле компьютера
- void placeUserShip (int x, int y, int lenght, shipLocation location) noexcept
  Разместить корабль пользователя
- void placeShipsAutomatically (Field *field) noexcept
  Разместить корабли автоматически
- bool allShipsDestroyed (Field *field) noexcept
  Узнать, все ли корабли разрушены

3.5.1 Подробное описание

Игровой интерфейс

3.5.2 Методы

3.5.2.1 bool GameAPI::allShipsDestroyed (Field * field) [noexcept]

Узнать, все ли корабли разрушены

Аргументы

field	указатель на поле

Возвращает

true/false

3.5.2.2 Field * GameAPI::getComputerField ( ) const [noexcept]

Получить поле компьютера

Возвращает

указатель на поле компьютера

3.5.2.3 Field * GameAPI::getUserField ( ) const [noexcept]

Получить поле игрока

Возвращает

указатель на поле игрока

3.5.2.4 bool GameAPI::makeComputerMove ( ) [noexcept]

Совершить ход компьютера

Возвращает

true при попадании
false при промахе

3.5.2.5 bool GameAPI::makeUserMove (int x, int y) [noexcept]

Совершить ход игрока

Аргументы

x	буквенная координата
y	цифровая координата

Возвращает

true при попадании
false при промахе

3.5.2.6 void GameAPI::placeShipsAutomatically (Field * field) [noexcept]

Разместить корабли автоматически

Аргументы

field	указатель на поле

3.5.2.7 void GameAPI::placeUserShip (int x, int y, int lenght, shipLocation location) [noexcept]

Разместить корабль пользователя

Аргументы

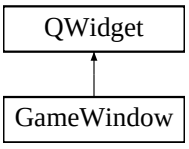
x	буквенная координата
y	цифровая координата
lenght	длина
location	расположение

Объявления и описания членов классов находятся в файлах:

- sources/Core/game.h
- sources/Core/game.cpp

3.6 Класс GameWindow

Граф наследования:GameWindow:



Открытые члены

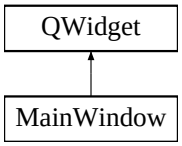
- GameWindow (QWidget *parent)

Объявления и описания членов классов находятся в файлах:

- sources/GUI/gamewindow.h
- sources/GUI/gamewindow.cpp

3.7 Класс MainWindow

Граф наследования:MainWindow:



Открытые члены

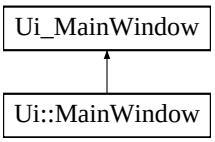
- MainWindow (QWidget *parent=0)

Объявления и описания членов классов находятся в файлах:

- sources/GUI/mainwindow.h
- sources/GUI/mainwindow.cpp

3.8 Класс Ui::MainWindow

Граф наследования:Ui::MainWindow:



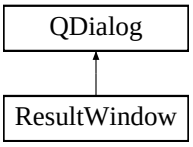
Дополнительные унаследованные члены

Объявления и описания членов класса находятся в файле:

- sources/build-GUI-Desktop_Qt_5_5_1_MinGW_32bit-Debug/ui_mainwindow.h

3.9 Класс ResultWindow

Граф наследования:ResultWindow:



Открытые члены

- ResultWindow (QWidget *parent, GameAPI *game)

Объявления и описания членов классов находятся в файлах:

- sources/GUI/resultwindow.h
- sources/GUI/resultwindow.cpp

3.10 Класс Ship

Корабль

#include <ship.h>

Открытые члены

- Ship (int x, int y, int lenght, shipLocation location) noexcept
- void createShip (int x, int y, int lenght, shipLocation location) noexcept

  Создать корабль
- void setShipStatus (int x, int y) noexcept

  Установить статус корабля
- bool shot (int x, int y) noexcept

  Проверить, попал ли игрок в клетку корабля При попадании изменяет статус клетки
- shipStatus getShipStatus () const noexcept

  Получить статус корабля
- std::vector< Cell > getShipCells () const noexcept

  Получить клетки корабля
- int getLenght () const noexcept

  Получить длину корабля
- shipLocation getShipLocation () const noexcept

  Получить расположение корабля

3.10.1 Подробное описание

Корабль

3.10.2 Методы

3.10.2.1 void Ship::createShip (int x, int y, int lenght, shipLocation location) [noexcept]

Создать корабль

Аргументы

field	игровое поле
x	первая буквенная координата корабля
y	первая цифровая координата корабля
lenght	длина корабля
location	расположение корабля (горизонтальное/вертикальное)

3.10.2.2 int Ship::getLenght ( ) const [noexcept]

Получить длину корабля

Возвращает

длина

3.10.2.3 std::vector< Cell > Ship::getShipCells ( ) const [noexcept]

Получить клетки корабля

Возвращает

клетки корабля

3.10.2.4 shipLocation Ship::getShipLocation ( ) const [noexcept]

Получить расположение корабля

Возвращает

расположение

3.10.2.5 shipStatus Ship::getShipStatus ( ) const [noexcept]

Получить статус корабля

Возвращает

статус

3.10.2.6 void Ship::setShipStatus (int x, int y) [noexcept]

Установить статус корабля

Аргументы

x	первая буквенная координата корабля
y	первая цифровая координата корабля

3.10.2.7 bool Ship::shot (int x, int y) [noexcept]

Проверить, попал ли игрок в клетку корабля При попадании изменяет статус клетки

Аргументы

field	игровое поле
x	буквенная координата
y	цифровая координата

Возвращает

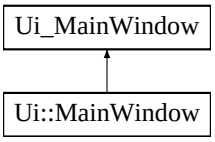
true/false

Объявления и описания членов классов находятся в файлах:

- sources/Core/ship.h
- sources/Core/ship.cpp

3.11 Класс Ui_MainWindow

Граф наследования:Ui_MainWindow:



Открытые члены

- void setupUi (QMainWindow *MainWindow)
- void retranslateUi (QMainWindow *MainWindow)

Открытые атрибуты

- QWidget * centralWidget
- QPushButton * buttonStart
- QPushButton * buttonExit

Объявления и описания членов класса находятся в файле:

- sources/build-GUI-Desktop_Qt_5_5_1_MinGW_32bit-Debug/ui_mainwindow.h

Предметный указатель