Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и технологий

**Высшая школа интеллектуальных систем и суперкомпьютерных технологий**

**КУРСОВАЯ РАБОТА**

**Разработка игры «Морской бой»**

по дисциплине «Технологии объектно-ориентированного программирования»

|  |  |  |
| --- | --- | --- |
| Выполнил  студент гр. з3530903/90001 |  |  |
|  | А. Ю. Леснов |
| Руководитель  Доцент к.т.н. |  | К. А. Туральчук |

«24» января 2021 г.

Санкт-Петербург

2021

Оглавление

[ВВЕДЕНИЕ 2](#_Toc62503257)

[АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 4](#_Toc62503258)

[*Назначение и область применения* 4](#_Toc62503259)

[*Описание программы* 4](#_Toc62503260)

[*Выбор инструментальных средств* 5](#_Toc62503261)

[РАЗРАБОТКА ИГРЫ «МОРСКОЙ БОЙ» 7](#_Toc62503262)

[*Реализация программы* 7](#_Toc62503263)

[*Тестирование* 10](#_Toc62503264)

[СПИСОК ИСТОЧНИКОВ 12](#_Toc62503265)

# ВВЕДЕНИЕ

Всевозможные варианты игр являются неотъемлемой частью человеческой культуры. История настольных игр насчитывает более 5000 лет. Одними из древнейших настольных игр были различные варианты игры в кости. Игральные кубики, выточенные из камня или из кости, неотличимые от сегодняшних игральных костей, находили уже в раскопах пятитысячелетней давности.

Дальнейшая история и развитие цивилизации принесло миру более сложные и интересные игры, такие как нарды, го, домино, шахматы, карточные игры. Более поздняя история и развитие науки, познаний в механике, так же внесло свой вклад. Люди начинали создавать интерактивные игральные механизмы, заводные игрушки и т. д.

Качественно новый рывок в возможности создания различных интерактивных игр принесло развитие вычислительной техники. Уже в 1940-х и 1950-х годах, в академической среде, на первых компьютерах, разрабатывались простые игры и симуляции. Однако настоящая популярность к компьютерным играм пришла в 1970-х и 1980-х годах, когда широкой публике стали доступны аркадные автоматы, игровые консоли и домашние компьютеры.

В наши дни игровая индустрия по охвату аудитории и объему капитала может соперничать с киноиндустрией. Сегодня существует большое количество компаний, создающих объёмные, глубокие игры всевозможных жанров для персональных компьютеров и специализированных игровых консолей. А, доступные обычному пользователю, вычислительные мощности позволяют компаниям выпускать на рынок по-настоящему живые, фотореалистичные интерактивные миры.

Целью данной курсовой работы является разработка интерактивной игры «Морской бой», в парадигме Объектно-ориентированного программирования. В ходе выполнения работы следует решить следующие задачи:

1. Декомпозиция задачи.
2. Построение диаграмм, описывающих приложение.
3. Создание приложения, обладающего необходимыми функциями и понятным пользовательским интерфейсом, с применением объектно-ориентированного подхода.
4. Разработка алгоритма принятия решений и игры компьютерным игроком.
5. Тестирование и отладка приложения.
6. В результате выполнения поставленных целей и задач должно быть создано приложение: игра «Морской бой».

# АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

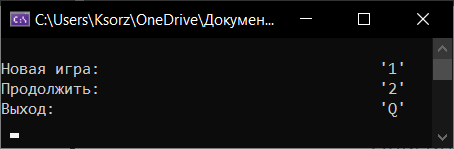
## Назначение и область применения

Назначением разрабатываемого приложения является воссоздание на компьютере известной игры «Морской бой», с классическими правилами. Программа должна обладать понятными интерфейсом и управлением, приятной и интуитивной-понятной графической частью. Использование программы предполагается на персональном компьютере.

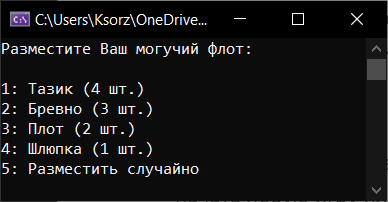
Область применения программы достаточно проста – это возможность игры с компьютерным противником.

## Описание программы

Программа встречает пользователя простым консольным меню, где он может начать новую игру или продолжить с того момента, где остановился ранее.



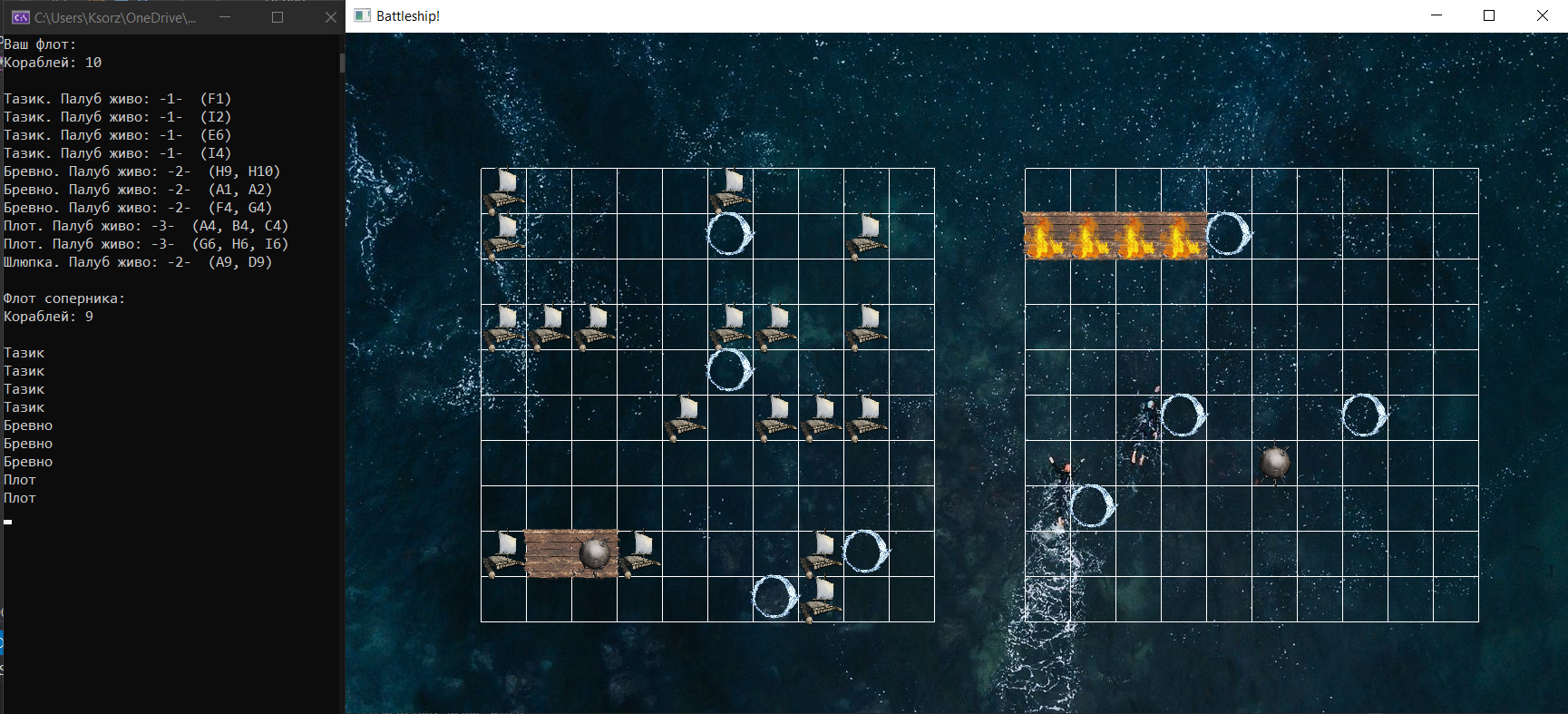
Пользователь может как самостоятельно расставить игровые корабли, так и воспользоваться помощью алгоритма, который сделает это за пользователя по всем правилам, в случайном порядке.



Графическое игровое поле, загружаемое далее, должно встречать пользователя приятными текстурами и быть интуитивно-понятным. Правила игры довольно просты:

* Размещаемые корабли не должны касаться друг друга вплотную или углами
* Соперники не видят кораблей друг друга, но видят свои собственные
* Игрок, попавший по кораблю соперника, имеет право на один дополнительный ход (максимум 2 хода подряд)
* Время на ход не ограничено
* Победителем считается игрок, уничтоживший все корабли соперника

Управление осуществляется с клавиатуры, клавишами стрелок. Выстрел – клавиша Enter. Быстрое сохранение и выход из игры – Escape (или нажать крестик, закрыть окно). Текстовое представление хода игры отображается в окне консоли.



Программа поддерживает одно автоматическое сохранение текущего прогресса игры в подготовленный текстовый файл. Сохраняются все данные обоих игроков, очередь хода, результат последнего хода, позиции игровых указателей. Проще говоря – полностью сохраняется текущее состояние игровой сессии.

При выборе пункта меню «продолжить» – все эти данные загружаются из файла.

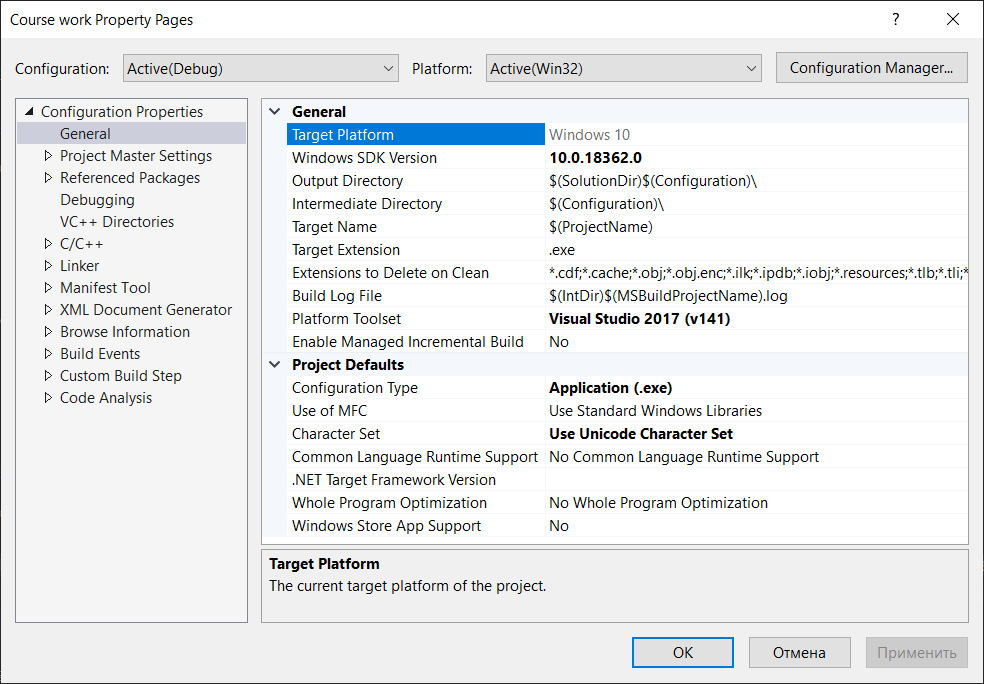
## Выбор инструментальных средств

Реализуемый проект должен быть написан в парадигме объектно-ориентированного программирования. Объектно-ориентированное программирование – методология программирования, основанная на представлении программы в виде совокупности объектов, каждый из которых является экземпляром определенного класса, а классы образуют иерархию наследования. Объектно-ориентированный подход к программированию позволяет разложить задачу на составные части таким образом, что каждая составная часть будет представлять собой самостоятельный объект, который содержит собственные инструкции и данные.

Концепция ООП позволяет программисту, концентрируясь на несложных составных частях одной сложной задачи, таким образом повышать производительность и решать достаточно трудоёмкие задачи, решение которых было бы затруднено при использовании парадигмы структурного программирования.

Для решения данной задачи был выбран язык С++. Данный язык является языком высокого уровня. C++ широко используется для разработки программного обеспечения, являясь одним из самых популярных языков программирования. Область его применения включает создание операционных систем, разнообразных прикладных программ, драйверов устройств, приложений для встраиваемых систем, высокопроизводительных серверов, а также развлекательных приложений. Также данный язык постоянно обновляется и совершенствуется, что позволяет ему сохранять актуальность.

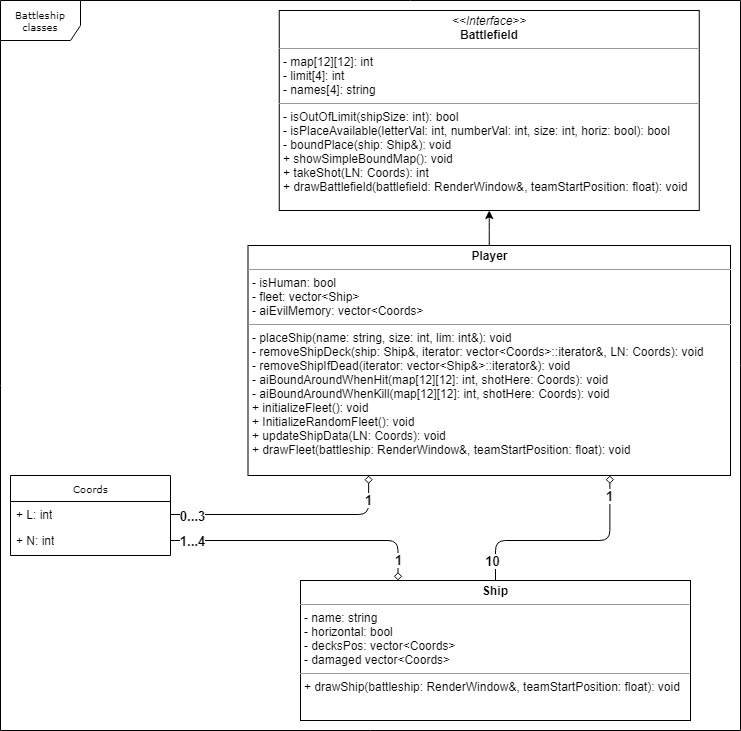
Программа разрабатывалась в Microsoft Visual Studio 2019, визуальное представление было реализовано с использованием библиотеки SFML. Для возможности поддержки SFML были выставлены следующие настройки среды Visual Studio:



# РАЗРАБОТКА ИГРЫ «МОРСКОЙ БОЙ»

## Реализация программы

Перед непосредственным созданием приложения в среде разработки была проведена декомпозиция задачи, с помощью которой были выявлены основные этапы разработки приложения. Были построены диаграммы, позволяющие более чётко и ясно понять структуру программы и её внутреннее устройство.



Разработка была начата с описания классов и методов к ним. Основная логика программы строится на работе с **vector<Ship> fleet;** находящимся в распоряжении класса **Player.**

Игровое поле представляет собой матрицу 12х12, где первые и последние координаты находятся вне обрабатываемой логикой зоны. Это нужно для того, чтобы упростить алгоритм принятия решения компьютером, таким образом он всегда будет иметь в распоряжении поля в непосредственной близости от обрабатываемого в данный момент времени.

Была реализована обработка данных поля **int map[12][12];** находящегося в интерфейсе **Battlefield**. Каждая переменная представляет собой одно из значений от 0 до 9, где:

1. Свободное поле
2. Свободное поле (рядом уже находится корабль)
3. Корабль 1 палуба
4. Корабль 2 палубы
5. Корабль 3 палубы
6. Корабль 4 палубы
7. Так алгоритм принятия решений отмечает поля, в которые не нужно стрелять
8. Корабль разрушен
9. Корабль ранен, но у него ещё остались живые палубы
10. Промах (сюда уже стреляли)

Далее был реализован алгоритм размещения (ручная и автоматическая версии), состоящий из основного метода: **void Player::placeShip()** с рекурсивной обработкой ошибок, и дополнительных методов: **bool** **Battlefield::isPlaceAvailable(), Battlefield::boundPlace().** На этой стадии он заполняет матрицу значениями от 1 до 5.

Затем происходит начало игры и отрисовка графического игрового окна с помощью библиотеки SFML. Основная переменная, задаваемая пользователем непосредственно в программном коде (в заголовочном файле) static float y = 750; задаёт масштаб всего окна. Это значение количества пикселей по вертикали, от которого рассчитывается одна из основных переменных static float step = y / 15.0f; отвечающая за отрисовку всех объектов игры в нужных местах игрового окна. Стоит отметить, что при такой реализации некоторые значения переменной **y** будут приводить незначительно-некорректной отрисовке игровых объектов из-за использования типа float, тогда как количество пикселей всегда – целочисленное значение.

Отрисовка и логика всей игры (взаимодействие игроков, управление, переход хода, информационные сообщения, выводящиеся на консоль) происходят в свободной функции **void theGame();** Основную работу здесь выполняют методы:

* **Coords Player::aiAim();** - алгоритм принятия решений компьютером
* **void Player::updateShipData(); -** обновление полей матрицы (игрового поля) и статуса кораблей игроков
* **int Battlefield::takeShot();** - реакция на выстрел по определённым координатам на карте

Когда вышеперечисленное было готово и отлажено, я реализовал сохранение текущего прогресса игры в файл. Свободные функции **void fastSaveGame();** и **void fastLoadGame();** отвечают за перенос в файл и извлечение из файла всех данных о текущем прогрессе игры. Данный функционал был реализован через переопределение операторов ввода/вывода для классов.

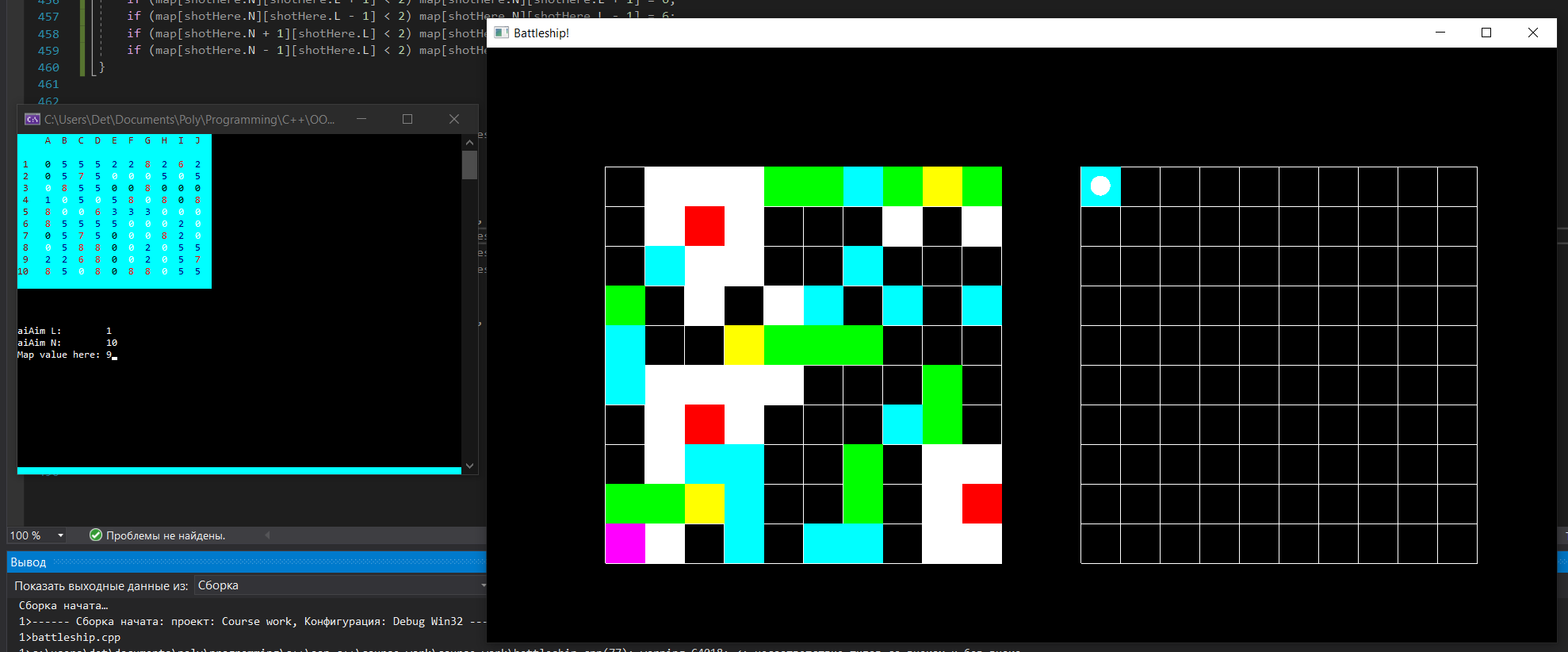
Для взаимодействия с имеющимся функционалом я написал простое консольное меню, где можно выбрать новую игру или загрузить последнюю, а также подменю создания кораблей.

## Тестирование

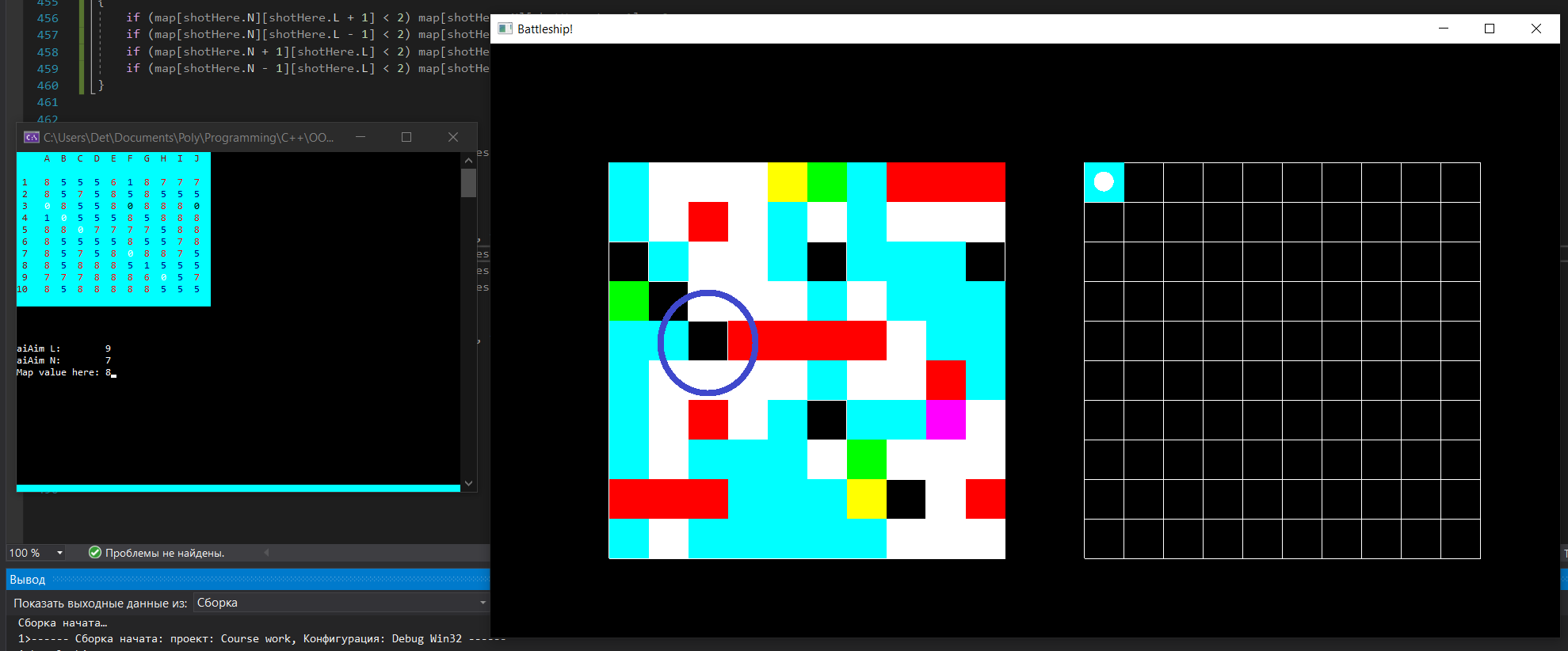
Отладка происходила, по большей части в процессе написания элементов кода программы. Все они были неоднократно проверены на корректность обрабатываемых данных, всевозможные ошибки при передаче/получении конкретных значений данных.

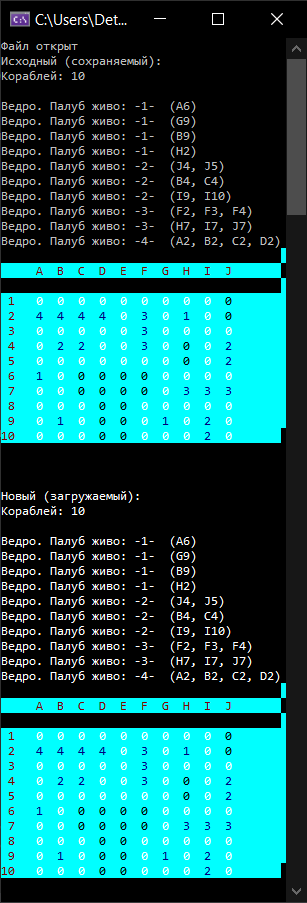
Так выглядело окно программы в процессе отладки логики работы алгоритма принятия решений компьютером в игре:

* **Черный**: Неисследованное поле
* **Зеленый**: Корабль
* **Жёлтый**: Подбитая часть
* **Красный**: Уничтоженный корабль
* **Голубой**: Промах
* **Белый**: Результат работы алгоритма принятия решений, компьютер не собирается стрелять по этим полям
* **Фиолетовый**: Текущий указатель цели компьютера



В данном примере компьютер делает ходы и пытается разрушить уже подбитый корабль, помечая на матрице значением 6 (белый) поля, в которые нет смысла пытаться стрелять.



Пример «пойманной» логической ошибки. Компьютер по какой-то причине не отметил очевидно пустое поле.

Пример успешной проверки алгоритмов сохранения и загрузки программы. Полная информация о состоянии выводится на консоль.

На подобных этапах тестирования было выявлено немало слабых мест, некорректных результатов и сбоев программы.

# ЗАКЛЮЧЕНИЕ

В результате проделанной работы была получена отлаженная и полностью работоспособная игра «Морской бой» с задуманным функционалом. Программа позволяет игроку размещать свои корабли в начале игры как самостоятельно, так и случайным образом, сохранять и загружать игровые сессии и играть в классическую версию игры «Морской бой» с компьютерным противником, который совершает осмысленные действия на игровом поле.

Задача выполнялась поэтапно:

* Определение структуры задачи
* Построение диаграмма классов UML
* Написание программного кода
* Тестирование
* Исправление неполадок
* Косметические улучшения (добавление текстур, фона)

# СПИСОК ИСТОЧНИКОВ

1. Стивен, П. C++: Язык программирования C++. Лекции и упражнения, 6-е издание.: М.: Издательский дом «Диалектика», 2018.
2. Уроки программирования на языке С++. – URL: https://ravesli.com/uroki-cpp/