МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра информационной безопасности

КУРСОВАЯ РАБОТА

по дисциплине «Программирование»

Тема: Реализация игры «Морской бой»

Студент гр. 6362 _____ Осипов В.А.

Преподаватель _____ Халиуллин Р.А.

Санкт-Петербург

2017

ЗАДАНИЕ НА КУРСОВУЮ РАБОТУ

Студент Осипов В.А.

Группа 6362

Тема работы : Реализация игры «Морской бой»

Исходные данные:

Разработать на языке C++ приложение «Морской бой». Приложение должно иметь консольный интерфейс. Приложение должно дать считывать данные введенные пользователем в консоли. Приложение должно корректно обрабатывать ошибки, в том числе ошибки ввода/вывода, выделения/освобождения памяти.

Содержание пояснительной записки:

Введение, теоретическая часть, теория игр, применение теории игр, типы игр, правила игры морской бой, варианты игры, реализация программы, используемое ПО, результаты работы программы, описание функций, заключение, , список используемых источников, приложение 1 – руководство пользователя, приложение 2 – блок-схема, приложение 3 – исходный код.

Предполагаемый объем пояснительной записки:

Не менее 35 страниц.

Дата выдачи задания: 06.02.2017

Дата сдачи реферата: 26.02.2017

Дата защиты реферата: 02.06.2017

Студент _____ Осипов В.А.

Преподаватель _____ Халиуллин Р.А

АННОТАЦИЯ

Курсовой проект содержит в себе реализацию приложения морской бой. . Реализованная программа написана на языке программирования C++. Результатом курсового проекта является готовое консольное приложение, способное дать возможность пользователю сыграть в Морской бой против компьютера.

SUMMARY

The console project contains the implementation of marine combat applications. The implemented program is written in C ++ programming language. The result of the course project is a ready-made console application, which can give an opportunity to play in the Battle of the Sea against the computer.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Все, кто имеет дело с компьютером, так или иначе сталкивались с компьютерными играми, и подавляющее большинство может сходу назвать несколько игр, которые им особенно понравились. Те, кто уже совсем наигрался, почти наигрался или еще не наигрался, но в процессе общения с компьютером уже начал совмещать игры с чем-нибудь более полезным, возможно, хотели бы придумать какие-нибудь свои, не похожие ни на какие другие игры. Многое захватывает в таком творчестве. И не сам процесс игры, а разработка игровой вселенной, ее проектирование и реализация. Когда можно слить воедино сценарий, графику, музыку, искусно задуманный и умело запрограммированный алгоритм — создать единый фантастический мир, живущий по законам, которые ты же для него и придумал. Одной из таки игр и является Морской бой.

Целью данной курсовой работы является разработка игры под названием Морской бой. Игра должна быть написана на языке программирования C++. При запуске, которого, пользователь может сыграть в Морской бой против компьютера. Приложение должно корректно обрабатывать ошибки, в том числе ошибки ввода/вывода, выделения/освобождения памяти.

1. ТЕОРИТИЧЕСКАЯ ЧАСТЬ

1.1. Теория игр

Теория игр — математический метод изучения оптимальных стратегий в играх. Под игрой понимается процесс, в котором участвуют две и более сторон, ведущих борьбу за реализацию своих интересов. Каждая из сторон имеет свою цель и использует некоторую стратегию, которая может вести к выигрышу или проигрышу — в зависимости от поведения других игроков. Теория игр помогает выбрать лучшие стратегии с учётом представлений о других участниках, их ресурсах и их возможных поступках

Игры представляют собой строго определённые математические объекты. Игра образуется игроками, набором стратегий для каждого игрока и указания выигрышей, или платежей, игроков для каждой комбинации стратегий. Большинство кооперативных игр описываются характеристической функцией, в то время как для остальных видов чаще используют нормальную или экстенсивную форму. Характеризующие признаки игры как математической модели ситуации:

1) Наличие нескольких участников;
2) Неопределённость поведения участников, связанная с наличием у каждого из них нескольких вариантов действий;
3) Различие (несовпадение) интересов участников;
4) Взаимосвязанность поведения участников, поскольку результат, получаемый каждым из них, зависит от поведения всех участников;
5) Наличие правил поведения, известных всем участникам.

1.2. Применение теории игр

Теория игр как один из подходов в прикладной математике применяется для изучения поведения человека и животных в различных ситуациях. Первоначально теория игр начала развиваться в рамках экономической науки,

позволив понять и объяснить поведение экономических агентов в различных ситуациях. Позднее область применения теории игр была расширена на другие социальные науки; в настоящее время теория игр используется для объяснения поведения людей в политологии, социологии и психологии. Теоретико-игровой анализ был впервые использован для описания поведения животных Рональдом Фишером в 30-х годах XX века (хотя даже Чарльз Дарвин использовал идеи теории игр без формального обоснования). В работе Рональда Фишера не появляется термин «теория игр». Тем не менее, работа по существу выполнена в русле теоретико-игрового анализа. Разработки, сделанные в экономике, были применены Джоном Майнардом Смитом в книге «Эволюция и теория игр». Теория игр используется не только для предсказания и объяснения поведения; были предприняты попытки использовать теорию игр для разработки теорий этичного или эталонного поведения. Экономисты и философы применяли теорию игр для лучшего понимания хорошего (достойного) поведения.

1.3. Типы игр

1) Кооперативные и некооперативные

Игра называется кооперативной, или коалиционной, если игроки могут объединяться в группы, взяв на себя некоторые обязательства перед другими игроками и координируя свои действия. Этим она отличается от некооперативных игр, в которых каждый обязан играть за себя. Развлекательные игры редко являются кооперативными, однако такие механизмы нередки в повседневной жизни. Часто предполагают, что кооперативные игры отличаются именно возможностью общения игроков друг с другом. В общем случае это неверно. Существуют игры, где коммуникация разрешена, но игроки преследуют личные цели, и наоборот.

2) Симметричные и несимметричные

Игра будет симметричной тогда, когда соответствующие стратегии у игроков будут равны, то есть иметь одинаковые платежи. Иначе говоря, если

игроки могут поменяться местами и при этом их выигрыши за одни и те же ходы не изменятся. Многие изучаемые игры для двух игроков — симметричные.

3) Параллельные и последовательные

В параллельных играх игроки ходят одновременно, или, по крайней мере, они не осведомлены о выборе других до тех пор, пока все не сделают свой ход. В последовательных, или динамических, играх участники могут делать ходы в заранее установленном либо случайном порядке, но при этом они получают некоторую информацию о предшествующих действиях других. Эта информация может быть даже не совсем полной, например, игрок может узнать, что его противник из десяти своих стратегий точно не выбрал пятую, ничего не узнав о других.

4) С полной или неполной информацией

Важное подмножество последовательных игр составляют игры с полной информацией. В такой игре участники знают все ходы, сделанные до текущего момента, равно как и возможные стратегии противников, что позволяет им в некоторой степени предсказать последующее развитие игры. Полная информация не доступна в параллельных играх, так как в них неизвестны текущие ходы противников. Большинство изучаемых в математике игр — с неполной информацией.

5) Игры с бесконечным числом шагов

Игры в реальном мире или изучаемые в экономике игры, как правило, длятся конечное число ходов. Математика не так ограничена, и в частности, в теории множеств рассматриваются игры, способные продолжаться бесконечно долго. Причём победитель и его выигрыш не определены до окончания всех ходов.

6) Дискретные и непрерывные игры

Большинство изучаемых игр дискретны: в них конечное число игроков, ходов, событий, исходов и т. п. Однако эти составляющие могут быть

расширены на множество вещественных чисел. Игры, включающие такие элементы, часто называются дифференциальными. Они связаны с какой-то вещественной шкалой (обычно — шкалой времени), хотя происходящие в них события могут быть дискретными по природе. Дифференциальные игры также рассматриваются в теории оптимизации, находят своё применение в технике и технологиях, физике.

7) Метаигры

Это игры, результатом которых является набор правил для другой игры (называемой целевой или игрой-объектом). Цель метаигр — увеличить полезность выдаваемого набора правил. Теория метаигр связана с теорией оптимальных механизмов.

1.4. Правила игры Морской бой

Игровое поле — обычно квадрат 10×10 каждого игрока, на котором размещается флот кораблей. Горизонтали обычно нумеруются сверху вниз, а вертикали помечаются буквами слева направо. При этом используются цифры от 1 до 10, либо буквы русского алфавита от "а" до "и", либо буквы латинского алфавита от «a» до «j». Иногда используется слово «республика» или «снегурочка», так как в этих 10-буквенных словах ни одна буква не повторяется. Поскольку существуют различные варианты задания системы координат, то об этом лучше заранее договориться.

Размещаются:

1 корабль — ряд из 4 клеток («четырёхпалубные»)

2 корабля — ряд из 3 клеток («трёхпалубные»)

3 корабля — ряд из 2 клеток («двухпалубные»)

4 корабля — 1 клетка («однопалубные»)

При размещении корабли не могут касаться друг друга сторонами и углами. Встречаются, однако, варианты, когда касание углами не запрещается. Встречаются также варианты игры, когда корабли могут размещаться буквой «Г» («трех-» и «четырехпалубные»), квадратом или зигзагом («четырехпалубные»). Кроме того, есть варианты с другим набором кораблей (напр. один пятипалубный, два четырёхпалубных, и т.д.) и/или другой формой поля.

Рядом со «своим» полем чертится «чужое» такого же размера, только пустое. Это участок моря, где плавают чужие корабли противника.

При попадании в корабль противника — на чужом поле ставится крестик, при холостом выстреле - точка. Попавший стреляет ещё раз.

Когда корабли расставлены, игроки по очереди производят «выстрелы», называя квадраты по их «координатам»: «1 1», «5 6» и т.д.. Если клетка занята кораблём или его частью, противник должен ответить «ранен» или «убит» («потоплен»). Эта клетка зачёркивается крестиком и можно сделаеть ещё один выстрел. Если в названной клетке корабля нет, в клетке ставится точка и ход переходит к сопернику.

Игра ведётся до полной победы одного из игроков, то есть, пока не будут потоплены все корабли. По окончании игры проигравший может попросить у победителя посмотреть на его расстановку кораблей.

1.5. Варианты игры

Существуют варианты игры, отличающиеся правилами (распространённые за пределами России). В основном, это касается количества и размера кораблей, например, вариант компании «Милтон Брэдли» — пятиклеточный, четырёхклеточный, два трёхклеточных и двухклеточный. Существуют варианты, где игрок может стрелять больше одного раза подряд. Также очень отличающийся вариант описан в книге Я. И. Перельмана «Занимательные Задачи и опыты».

При стандартном размере поля (10×10) и стандартном наборе кораблей (1×4 + 2×3 + 3×2 + 4×1), в игру можно добавить одну мину (или не одну). Мина обозначается кружком, вписанным в одну клетку. Клетка с миной не должна касаться кораблей, а если мин больше одной, то и других клеток с минами.

Если игрок в результате своего хода попал на мину (на мину противника), то он должен сообщить хозяину мины (противнику) координаты одной своей непоражённой клетки, занятой любым своим кораблём (корабль может иметь сколько угодно клеток, но выдаётся только одна клетка). После этого хозяин мины имеет возможность метко выстрелить (выданная клетка не погибает в момент попадания на мину — чтоб она погибла, по ней надо выстрелить; иначе говоря, мина только сообщает координаты корабля). Хозяин мины не обязан поражать выданную клетку сразу же — он имеет право выстрелить по ней в любое время. Поскольку выстрел по выданной клетке меткий, то хозяин мины после этого выстрела получает право на повторный ход. Использованная мина «гасится» постановкой точки в центре кружка (в центре её клетки).

Размер поля можно увеличить — например, размер 16×16 или 18×18 позволяет с удобством использовать весь размер одинарного тетрадного листа. В этом случае количество фигур можно увеличить — например, как предлагал Я. И. Перельман. Тогда, в связи с увеличением численности армий и размера поля, можно увеличить количество мин (например, до трёх) и добавить в игру минный тральщик (скажем, один у каждого игрока). Минный тральщик обозначается равнобедренным треугольничком, вписанным в одну клетку, так, что основание равнобедренного треугольника совпадает с нижней стороной клетки, а противоположная основанию вершина лежит на верхней стороне клетки, деля верхнюю сторону пополам.

Если игрок, сделав ход, попал на минный тральщик, то он должен выдать противнику (хозяину минного тральщика) координаты одной из своих ещё не сработавших мин — чтобы хозяин минного тральщика знал, что по этим координатам выданной клетки с миной ходить не следует. Клетка с минным

тральщиком не должна касаться клеток с кораблями и минами, а также, если минных тральщиков больше одного, и клеток с другими минными тральщиками. Если к моменту срабатывания минного тральщика у походившего не осталось ни одной мины, то противник походившего сообщает походившему, что он попал на минный тральщик, но походивший ему ничего не выдаёт.

Так как попадание на мину или на минный тральщик не является успехом, а является неприятностью для ходившего, то после такого неудачного хода ход переходит к хозяину сработавшей мины или сработавшего минного тральщика. Попав на мину, нельзя вместо координат клетки корабля выдавать клетку с минным тральщиком. Мины и минные тральщики являются одноклеточными фигурами. Мины и минные тральщики не считаются значащими фигурами — поэтому, если у игрока остались только мины и минные тральщики, но погибли все корабли, а у другого игрока не все корабли погибли, то игра считается оконченной, а первый игрок — проигравшим.

Существует вариант игры, в котором мины, минные тральщики могут касаться кораблей или друг друга.

2. РЕАЛИЗАЦИЯ ПРОГРАММЫ.

2.1. Используемое программное обеспечение.

Компилятор: g++.

Среда разработки: Sublime Text.

2.2. Результаты работы программы

После запуска приложения перед пользователем возникает консольное окно, содержание которого показано на рисунке 1.

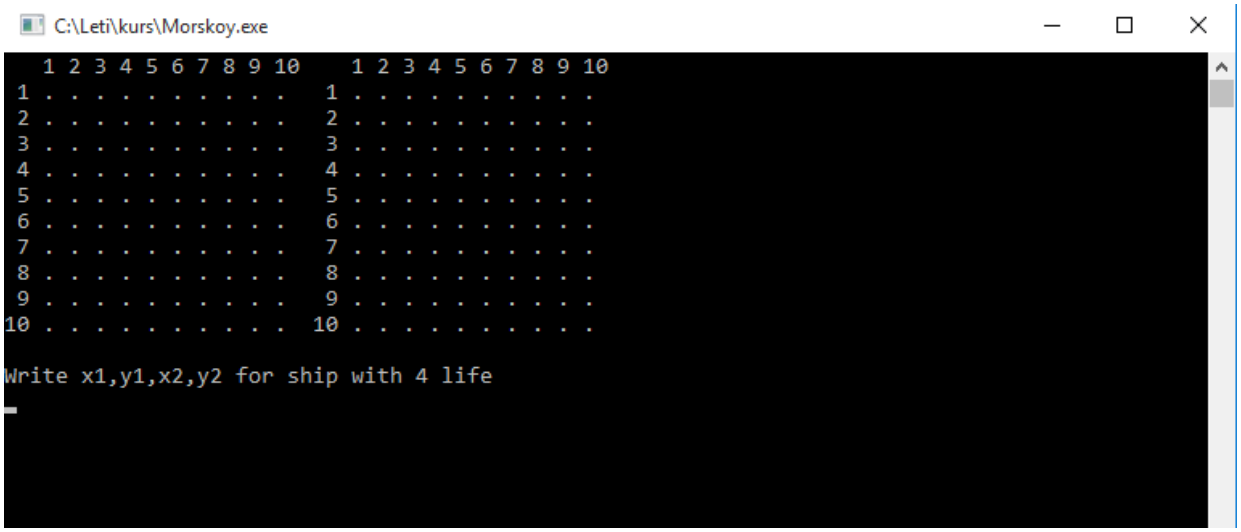


Рисунок 1 – Консольное окно при запуске программы

Далее пользователю необходимо ввести координаты для каждого корабля. Результат после расстановки всех кораблей показан на рисунке 2.

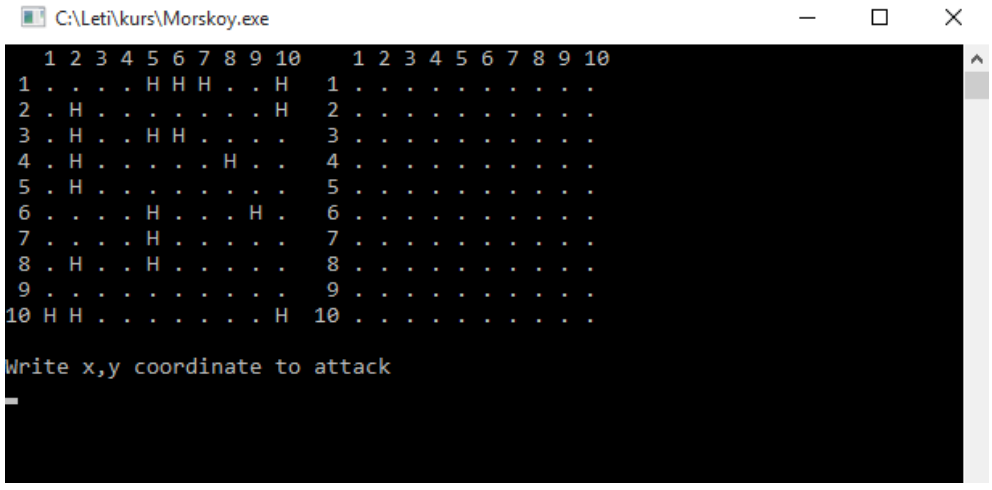


Рисунок 2 – Окно в результате расстановки кораблей

Потом происходит перестрелка между игроком и компьютером. Пример этого можно увидеть на рисунке 3.

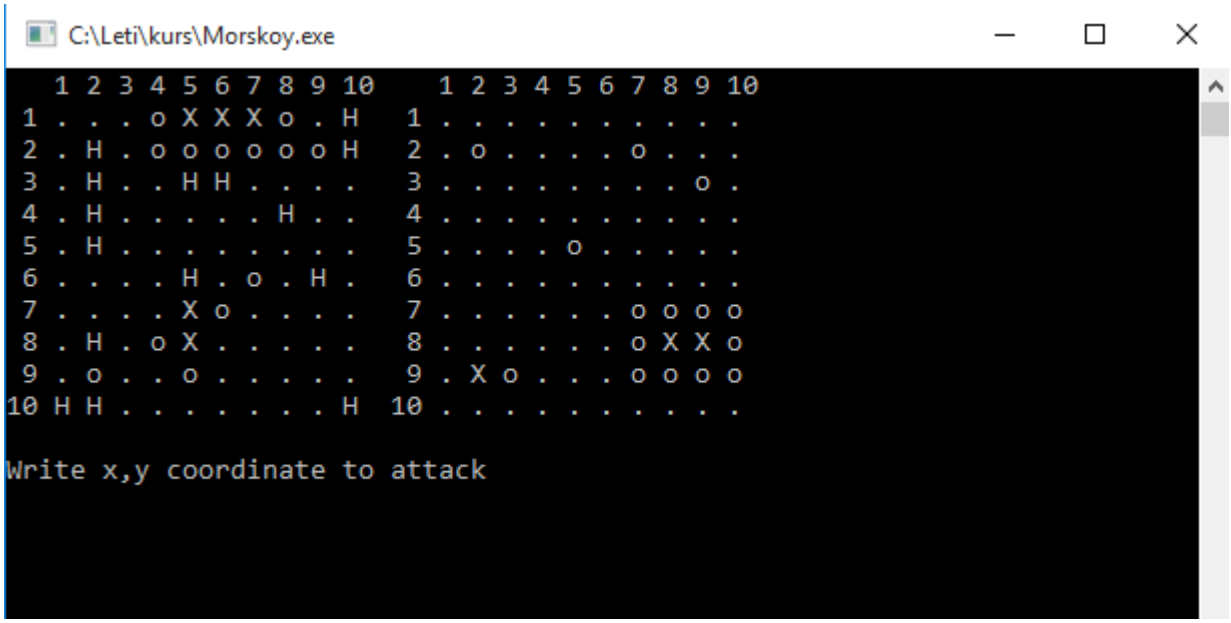


Рисунок 3 – Пример перестрелки между игроком и компьютером

В результате после победы пользователя или компьютера появляется уведомление об этом. Пример уведомления о победе пользователя можно увидеть на рисунке 4.

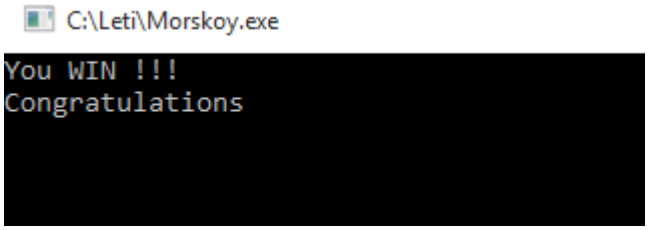


Рисунок 4 – Уведомления о победе пользователя

2.3. Описание функций

1) Функция PrintFields.

Функция PrintFields выводит на экран 2 поля, одно пользователя, другое компьютера.

Тип функции: void

Объявление функции:

void PrintFields()

Функция не принимает на вход ни одного аргумента.

Функция не возвращает значения.

2) Функция Test1.

Функция Test1 проверяет возможно ли установить корабль, согласно полученным координатам.

Тип функции: bool.

Объявление функции:

bool Test1(int x1, int y1,int x2,int y2,int z, int k)

Аргументы функции:

x1,y1,x2,y2 – координаты 1-ой и последней палубы у корабля;

z – количество палуб;

k – флаг, сообщающей о том, чей корабль устанавливается.

Возвращаемое значение:

false – если можно установить корабль;

true – если нельзя установить корабль.

3) Функция Test.

Функция Test проверяет возможно ли установить корабль для человека, согласно полученным координатам.

Тип функции: bool.

Объявление функции:

bool Test(int x1, int y1,int x2,int y2,int z)

Аргументы функции:

x1,y1,x2,y2 – координаты 1-ой и последней палубы у корабля.

Возвращаемое значение:

false – если можно установить корабль;

true – если нельзя установить корабль.

4) Функция TestShip.

Функция TestShip проверяет жив ли корабль компьютера, согласно полученным координатам.

Тип функции: bool.

Объявление функции:

bool TestShip(int x, int y)

Аргументы функции:

x – координата x атаки пользователя;

y – координата y атаки пользователя.

Возвращаемое значение:

false – если корабль мертв;

true – если корабль жив.

5) Функция TestShipPl.

Функция TestShipPl проверяет жив ли корабль пользователя, согласно полученным координатам.

Тип функции: bool.

Объявление функции:

bool TestShip(int x, int y)

Аргументы функции:

x – координата x атаки пользователя;

y – координата y атаки пользователя.

Возвращаемое значение:

false – если корабль мертв;

true – если корабль жив.

6) Функция Potop.

Функция Potop производит заполнение на поле компьютера вокруг потопленного корабля ‘o’, как свидетельство о том, что корабль компьютера потоплен.

Тип функции: void

Объявление функции:

void Potop(int x, int y)

Аргументы функции:

x, y– координаты одной из палуб корабля, который следует потопить;

Функция не возвращает значения.

7) Функция PotopPlayer.

Функция PotopPlayer производит заполнение на поле пользователя вокруг потопленного корабля ‘o’, как свидетельство о том, что корабль пользователя потоплен.

Тип функции: void

Объявление функции:

void PotopPlayer (int x, int y)

Аргументы функции:

x, y– координаты одной из палуб корабля, который следует потопить;

Функция не возвращает значения.

8) Функция Locate.

Функция Locate определяет сколько палуб будет корабля пользователя при установке.

Тип функции: void

Объявление функции:

void Locate (int x1, int y1, int x2, int y2, int z)

Аргументы функции:

x1, y1, x2, y2 – координаты 1-ой и последней палубы у корабля;

z – количество палуб.

Функция не возвращает значения.

9) Функция CompShip.

Функция CompShip производит автоматическую установку кораблей.

Тип функции: void

Объявление функции:

void CompShip (int k)

k – флаг, сообщающей о том, чей корабль устанавливается.

Функция не возвращает значения.

10) Функция SetShip.

Функция SetShip определяет сколько палуб будет корабля пользователя при установки.

Тип функции: void

Объявление функции:

void SetShip()

Функция не принимает на вход ни одного аргумента.

Функция не возвращает значения.

11) Функция PlayerAt.

Функция PlayerAt производит атаку пользователя по полю противника.

Тип функции: void

Объявление функции:

void PlayerAt()

Функция не принимает на вход ни одного аргумента.

Функция не возвращает значения.

12) Функция CompAt1.

Функция CompAt1 производит атаку компьютера по полю пользователя, в случае если корабль, по которому попали функцией CompAt еще жив.

Тип функции: void

Объявление функции:

void CompAt1(int x, int y)

Аргументы функции:

x – координата x атаки компьютера;

y – координата y атаки компьютера.

Функция не возвращает значения.

13) Функция CompAt.

Функция CompAt производит атаку компьютера по полю пользователя.

Тип функции: void

Объявление функции:

void CompAt()

Функция не принимает на вход ни одного аргумента.

Функция не возвращает значения.

14) Функция main.

Функция main получает управление при запуске программы.

Тип функции: void

Тип функции: int

Объявление функции: int main();

Функция не принимает на вход ни одного аргумента.

Возвращаемое значение:

Функция возвращает 0 при успешном окончании работы программы

ЗАКЛЮЧЕНИЕ

В ходе выполнения данной работы было написано консольное приложение, на языке C++. Данное приложение позволяет пользователю сыграть в морской бой с компьютером Программа проста в установке и использовании. В случае продолжения разработки данной программы следует добавить возможность играть с другим пользователем по сети, а также улучшенный графический интерфейс. При написании данной работы мной была изучена специальная литература, включающая статьи по теории игр и их оптимизации. В ходе выполнения курсовой работы был получен опыт в разработке полноценной игры на C++, а также в изучении устройства связи функций.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. https://ru.wikipedia.org/wiki/Морской_бой_(игра) (дата обращения: 27.05.2017).

2. Страуструп Б. Язык программирования C++. Специальное издание. Пер. с англ. – М.: ООО «Бином–Пресс», 2005 г. – 1104 с.: ил.

3. Брайан Керниган, Деннис Ритчи. Язык программирования С. – М.: Вильямс, 2015. – 304 с.

4. Эндрю Кениг, Барбара Му. Эффективное программирование на C++. – М.:Вильямс, 2002. – 384 с.

5. Мазалов В.В. Математическая теория игр и приложения. 2010. – 446 с.

6. Петросян Л. А. Зенкевич Н.А., Семина Е.А. Теория игр: Учеб. пособие для ун-тов. 1998. – С. 304.

ПРИЛОЖЕНИЕ 1. РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

1) Минимальные системные требования

Операционная система: Windows 10 64 – разрядная операционная система;

Процессор: 1 ГГц или выше с поддержкой PAE, NX и SSE2;

Установленная ОЗУ: 2 Гб;

Графическая карта: VGA–совместимая графическая карта с разрешением 800 x 600.

2) Процесс установки

Скопировать исполняемый файл «Morskoy.exe» в директорию.

3) Работа с программой

Для работы с программой необходимо следующее:

а) Запустить файл «Morskoy.exe»;

б) После открытия файла, необходимо будет ввести координаты для каждого из 10 кораблей (1 четырехпалубник,2 трёхпалубного, 3 двупалубного, 4 однопалубного). Первые 2 цифры обозначают x,y для 1-ой палубы корабля, вторые 2 цифры обозначают x,y для последней палубы корабля. Обратите внимание, что координата x по горизонтали, а y по вертикали, причем нумерация начинается с правого верхнего угла. Левое поле показывает где установлены ваши корабли (они обозначаются как – H), причем корабли не должны находиться 2 соседних клетках. Процесс установи кораблей показан на рисунке 5;

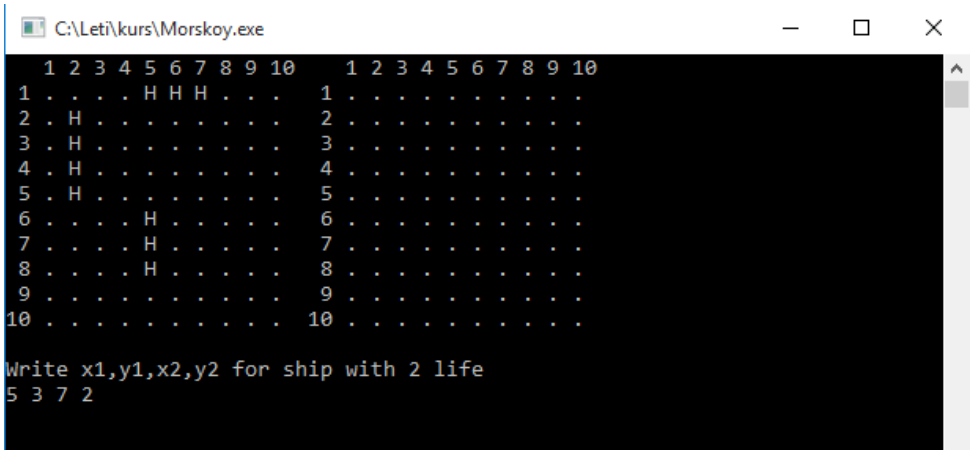


Рисунок 5 – Процесс установи кораблей

в) После установки кораблей будет продолжено ввести координаты того, куда вы собираетесь атаковать противника. После вода координат на втором поле будет показан результат ваших действий (о – мимо, X – попали), а на первом то, куда попал ваш противник. Пример этого представлен на рисунке 6;

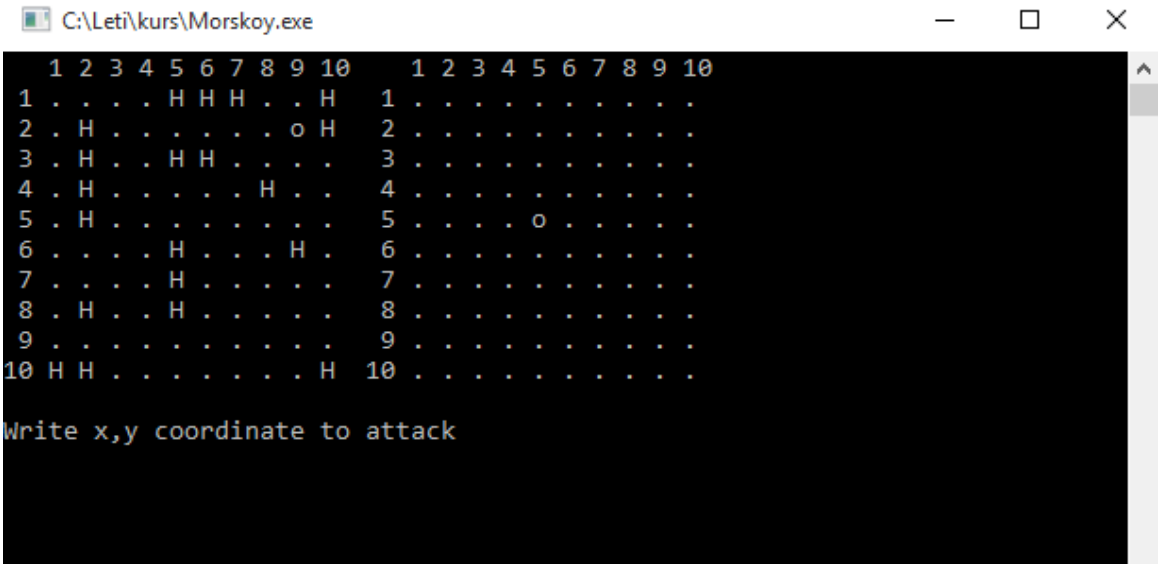


Рисунок 6 – Пример атаки .

г) Игра ведется до тех пор, пока на вашем поле или поле противника не останется живых кораблей. После победы или поражения будет показан результат этого. После чего приложение закроется. Пример победы показан на рисунке 7.

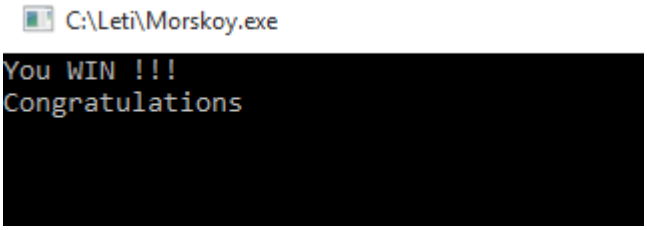
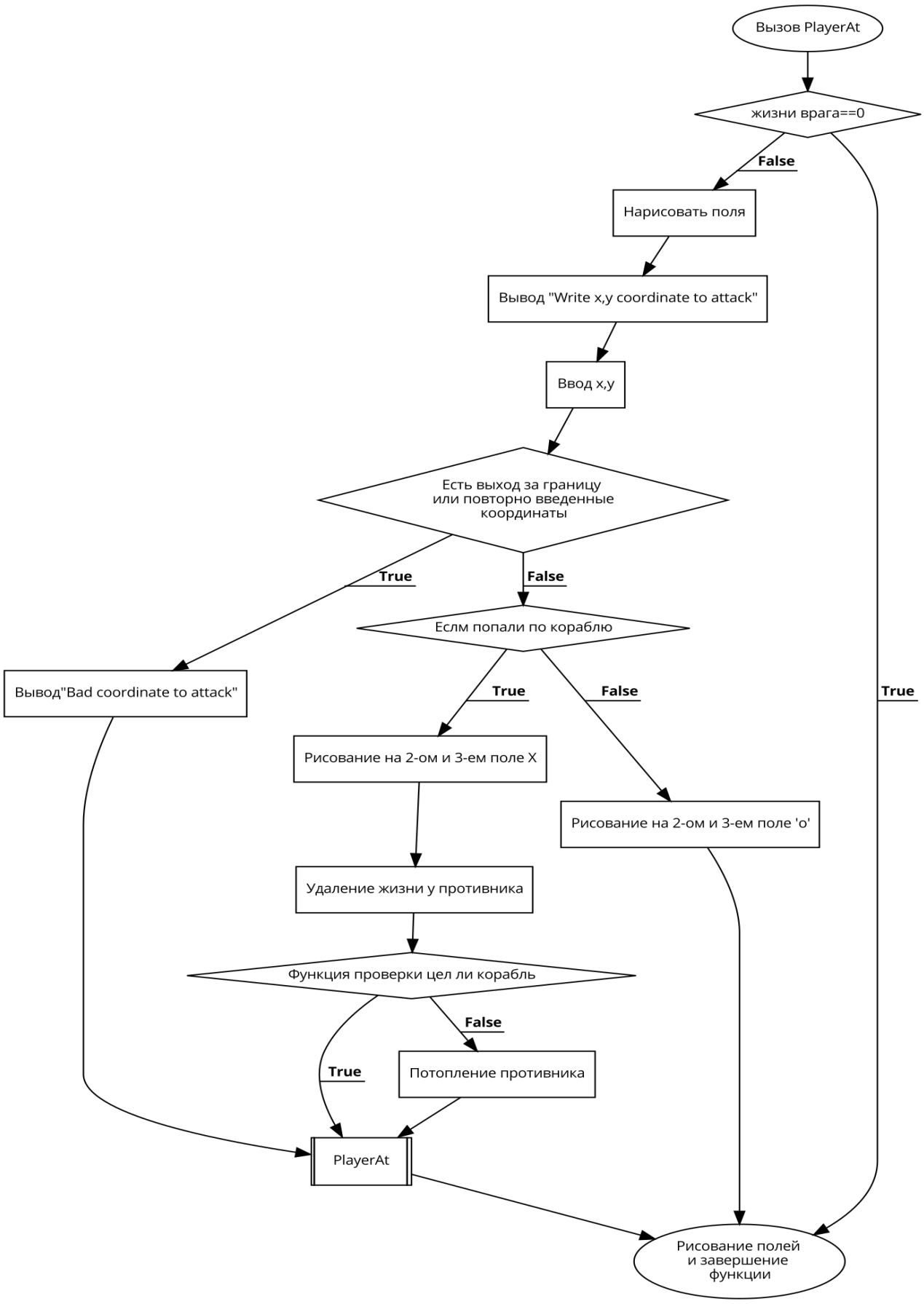


Рисунок 7 – Пример победы



					Приложение 2. Блок-схема
Из	Лист	№ докум	Подпись	Дата	
Разраб.		Осипов В.А.			
Пров.		Халиуллин Р.			
Реценз.		Халиуллин Р.			
Н. Контр.		Халиуллин Р.			
Утв		Халиуллин Р.			

Литера	Лист	Листов
у	1	1

ПРИЛОЖЕНИЕ 3. ИСХОДНЫЙ КОД ПРОГРАММЫ

```
#include <iostream>
#include <cstring>
#include <vector>
#include <iomanip>
#include <windows.h>
#include <cstdio>
#include <ctime>
#include <string>

using namespace std;

char FirstField[12][12];
char SecondField[12][12];
char ThirdField[12][12];
int life1=20;
int life2=20;
int corab=0;
int coordx,coordy;

void PrintFields()
{
    system("cls");

    cout<<"   ";

    for (int i=1; i<=10; i++)
        cout<<i<<" ";

    cout<<"   ";

    for(int i=1; i<=10; i++)
        cout<<i<<" ";

    cout<<endl;

    for(int i=1; i<=10; i++)
    {
        cout<<setw(2)<<i<<" ";

        for(int j=1; j<=10; j++)
            cout<<FirstField[i][j]<<" ";

        cout<<" "<<setw(2)<<i<<" ";
```

```
        for(int k=1; k<=10; k++)
            cout<<SecondField[i][k]<<" ";

        cout<<setw(2)<<endl;
    }

    cout<<endl;
}

bool Test1(int x1, int y1,int x2,int y2,int z,int k)
{
    int l=0;
    for(int i=x1-1; i<=x2+1; i++)
        for(int j=y1-1; j<=y2+1; j++)
        {
            if(k==2)
                if(ThirdField[i][j]=='.')
                    l++;
            if(k==1)
                if(FirstField[i][j]=='.')
                    l++;
           }
    if (l==(z*3+6))
    return false;
    else
    return true;
}


bool Test(int x1, int y1,int x2,int y2)
{
    int l=0;
    for(int i=x1-1; i<=x2+1; i++)
        for(int j=y1-1; j<=y2+1; j++)
            if(FirstField[i][j]=='H')
             l++;
    if ( l!=0)
    return true;
    else
    return false;
}

bool TestShip(int x, int y)
{
    int l=0;
```

```
    int kol=0;
    int xl=x;
    int xr=x;
    int yv=y;
    int yn=y;
    while(ThirdField[xl-1][y]=='H' || ThirdField[xl-
1][y]=='X')
    {
        l++;
        if (ThirdField[xl-1][y]=='X')
            kol++;
        xl=xl+1;
    }
    while(ThirdField[xr+1][y]=='H' ||
ThirdField[xr+1][y]=='X')
    {
        l++;
        if (ThirdField[xr+1][y]=='X')
            kol++;
        xr=xr+1;
    }
    while(ThirdField[x][yv+1]=='H' ||
ThirdField[x][yv+1]=='X')
    {
        l++;
        if (ThirdField[x][yv+1]=='X')
            kol++;
        yv=yv+1;
    }
    while(ThirdField[x][yn-1]=='H' || ThirdField[x][yn-
1]=='X')
    {
        l++;
        if (ThirdField[x][yn-1]=='X')
            kol++;
        yn=yn-1;
    }


    if (l!=kol)
    return true;
    else
    return false;
}
```

```
bool TestShipPl(int x, int y)
{
     int l=0;
     int kol=0;
     int xl=x;
     int xr=x;
     int yv=y;
     int yn=y;
     while(FirstField[xl-1][y]=='H' || FirstField[xl-
1][y]=='X')
     {
          l++;
          if (FirstField[xl-1][y]=='X')
               kol++;
          xl=xl+1;
     }
     while(FirstField[xr+1][y]=='H' ||
FirstField[xr+1][y]=='X')
     {
          l++;
          if (FirstField[xr+1][y]=='X')
               kol++;
          xr=xr+1;
     }
     while(FirstField[x][yv+1]=='H' ||
FirstField[x][yv+1]=='X')
     {
          l++;
          if (FirstField[x][yv+1]=='X')
               kol++;
          yv=yv+1;
     }
     while(FirstField[x][yn-1]=='H' || FirstField[x][yn-
1]=='X')
     {
          l++;
          if (FirstField[x][yn-1]=='X')
               kol++;
          yn=yn-1;
     }


    if ( l!=kol)
    return true;
    else
```

```
    return false;
}

void Potop(int x, int y)
{
    int xl=x;
    int xr=x;
    int yv=y;
    int yn=y;
    while(ThirdField[xr+1][y]=='X')
    {
        xr=xr+1;
    }
    while(ThirdField[xl-1][y]=='X')
    {
        xl=xl-1;
    }
    while(ThirdField[x][yv+1]=='X')
    {
        yv=yv+1;
    }
    while(ThirdField[x][yn-1]=='X')
    {
        yn=yn-1;
    }
    if(yv==yn)
    {
        for(int i=xl-1; i<=xr+1; i++)
            for(int j=yn-1; j<=yv+1; j+=2)
                {
                 SecondField[i][j]='o';
                 ThirdField[i][j]='o';
                }
        SecondField[xr+1][yn]='o';
        SecondField[xl-1][yn]='o';
        ThirdField[xr+1][yn]='o';
        ThirdField[xl-1][yn]='o';
    }
    else
    {
        for(int i=xl-1; i<=xr+1; i+=2)
            for(int j=yn-1; j<=yv+1; j++)
             {
                 ThirdField[i][j]='o';
                 SecondField[i][j]='o';
```

```
            }
        SecondField[xr][yn-1]='o';
        SecondField[xr][yv+1]='o';
        ThirdField[xr][yn-1]='o';
        ThirdField[xr][yv+1]='o';
    }
}

void PotopPlayer(int x, int y)
{
    int xl=x;
    int xr=x;
    int yv=y;
    int yn=y;
    while(FirstField[xr+1][y]=='X')
    {
        xr=xr+1;
    }
    while(FirstField[xl-1][y]=='X')
    {
        xl=xl-1;
    }
    while(FirstField[x][yv+1]=='X')
    {
        yv=yv+1;
    }
    while(FirstField[x][yn-1]=='X')
    {
        yn=yn-1;
    }
    if(xl==xr && yn==yv)
    {
        for(int i=x-1;i<=x+1;i++)
            for(int j=y-1;j<=y+1;j++)
                FirstField[i][j]='o';
        FirstField[x][y]='X';
        goto m;
    }
    if(yv==yn)
    {
        for(int i=xl-1; i<=xr+1; i++)
            for(int j=yn-1; j<=yv+1; j+=2)
                {
                 FirstField[i][j]='o';
                }
```

```
        FirstField[xr+1][yn]='o';
        FirstField[xl-1][yn]='o';
    }
    else
    {
        for(int i=xl-1; i<=xr+1; i+=2)
            for(int j=yn-1; j<=yv+1; j++)
             {
                 FirstField[i][j]='o';
             }
        FirstField[xr][yn-1]='o';
        FirstField[xr][yv+1]='o';
    }
    m:;
}

void Locate(int x1, int y1,int x2,int y2,int z)
{
    char py1[256],px1[256],px2[256],py2[256];
    int k=0;
    if (z==1)
    {
        while(k==0)
        {
            cout<<"Write x,y for ship with 1 life"<<
endl;
            cin>>py1>>px1;
            y1=atoi(py1);
            x1=atoi(px1);
            if ((x1<1) || x1>10 || y1<1 || y1>10 ||
Test(x1,y1,x1,y1) )
                cout<<"Bad coordinate"<< endl;
            else
            {   k=1;
                FirstField[x1][y1]='H';
                PrintFields();
            }
        }
    }
    else
        while(k==0)
        {
            cout<<"Write x1,y1,x2,y2 for ship with
"<<z<< " life"<< endl;
            cin>>py1>>px1>>py2>>px2;
```

```
            y1=atoi(py1);
            x1=atoi(px1);
            y2=atoi(py2);
            x2=atoi(px2);
            if( (x1<1) || (x2>10) || (y1<1) || (y2>10)
|| (x1!=x2 && y1!=y2) || ((x2-x1!=z-1) && (y2==y1)) ||
((x2==x1) && (y2-y1!=z-1)) || Test(x1,y1,x2,y2) )
            {
                cout<<"Bad coordinate"<< endl;
            }
            else
                {
                    for(int i=x1;i<=x2;i++)
                        for(int j=y1;j<=y2;j++)
                            FirstField[i][j]='H';
                    PrintFields();
                    k=1;
                }
        }
}

void CompShip(int k)
{
    int x1,y1,x2,y2;
    srand(time(0));
    int pos;
    int size=4;
    int j=10;
    while(j>0)
    {
        pos=1+rand() % 2;
        if(pos==1)
        {
            x1=x2=1+rand() % 10;
            y1=1+rand() % 10;
            if(y1+size-1>10)
                continue;
            else
                y2=y1+size-1;
        }
        else
            {
                y1=y2=1+rand() % 10;
                x1=1+rand() % 10;
                if(x1+size-1>10)
```

```
                    continue;
                else
                    x2=x1+size-1;
            }
        if(Test1(x1,y1,x2,y2,size,k))
            continue;
        else
        {
        j--;
        if (j==8 || j==9) size=3;
            else if ( j<8 && j>4) size=2;
                else size =1;
            for(int i=x1; i<=x2; i++)
                for(int j=y1; j<=y2; j++)
                if(k==1)
                    FirstField[i][j]='H';
                else
                    ThirdField[i][j]='H';
        }
    }
}

void SetShip()
{
    int x1,y1,x2,y2;
    int shet=10;
    while(shet>0)
    {
        if (shet==10)
        {
            Locate(x1,y1,x2,y2,4);
            shet--;
        }
        else
            if (shet>7)
            {
                Locate(x1,y1,x2,y2,3);
                shet--;
            }
            else
                if (shet>4)
                    {
                        Locate(x1,y1,x2,y2,2);
                        shet--;
                    }
```

```
                else
                    {
                        Locate(x1,y1,x1,y1,1);
                        shet--;
                    }
    }
}

void PlayerAt()
{
    if (life2==0)
    goto h;
    PrintFields();
    char py[256],px[256];
    int x,y;
    cout << "Write x,y coordinate to attack"<< endl;
    cin>>py>>px;
    y=atoi(py);
    x=atoi(px);
    if ((x<1) || x>10 || y<1 || y>10 ||
ThirdField[x][y]=='o'|| ThirdField[x][y]=='X')
    {
        cout<<"Bad coordinate to attack"<< endl;
        Sleep(2000);
        PlayerAt();
    }
    else
    {
        if (ThirdField[x][y]=='H' )
        {
            SecondField[x][y]='X';
            ThirdField[x][y]='X';
            life2--;
            if (TestShip(x,y))
                PlayerAt();
            else
            {
                Potop(x,y);
                PlayerAt();
            }
        }
        else
        {
            SecondField[x][y]='o';
            ThirdField[x][y]='o';
```

```
            }
        }
        h: ;
        PrintFields();
}

void CompAt1(int x,int y)
{
        int z=1+rand() % 2;
        int xl=x;
        int xr=x;
        int yv=y;
        int yn=y;
        if (life1==0)
            goto z3;
        while(FirstField[xr+1][y]=='X')
        {
            xr=xr+1;
        }
        while(FirstField[xl-1][y]=='X')
        {
            xl=xl-1;
        }
        while(FirstField[x][yv+1]=='X')
        {
            yv=yv+1;
        }
        while(FirstField[x][yn-1]=='X')
        {
            yn=yn-1;
        }
        if (xl==xr && FirstField[x][yv+1]!='o' && (yn-1)>=0
&& (yv+1)<=10 && FirstField[x][yn-1]!='o')
        {
            if(z==1)
            {
                if(FirstField[x][yv+1]=='H')
                {
                    FirstField[x][yv+1]='X';
                    life1--;
                    if(TestShipPl(x,y))
                        CompAt1(x,y);
                    else
                    {
                        corab=0;
```

```
                    PotopPlayer(x,y);
                }
            }
            else
                if(FirstField[x][yv+1]=='o')
                    CompAt1(x,y);
                else
                    if (yv+1<=10)
                        FirstField[x][yv+1]='o';
                    else
                        CompAt1(x,y);
        }
        else
        {
            if(FirstField[x][yn-1]=='H')
            {
                FirstField[x][yn-1]='X';
                life1--;
                if(TestShipPl(x,y))
                    CompAt1(x,y);
                else
                {
                    corab=0;
                    PotopPlayer(x,y);
                }
            }
            else
                if(FirstField[x][yn-1]=='o')
                    CompAt1(x,y);
                else
                    if (yn-1>=0)
                        FirstField[x][yn-1]='o';
                    else
                        CompAt1(x,y);
        }
    }
    else if (yv==yn && FirstField[xr+1][y]!='o' && (xl-
1)>=0 && (xr+1)<=10 && FirstField[xl-1][y]!='o')
    {
        if(z==1)
        {
            if(FirstField[xr+1][y]=='H')
            {
                FirstField[xr+1][y]='X';
                life1--;
```

```
                    if(TestShipPl(x,y))
                        CompAt1(x,y);
                    else
                    {
                        corab=0;
                        PotopPlayer(x,y);
                    }
                }
            else
                if(FirstField[xr+1][y]=='o')
                    CompAt1(x,y);
                else
                    if (xr+1<=10)
                        FirstField[xr+1][y]='o';
                    else
                        CompAt1(x,y);
            }
            else
            {
                if(FirstField[xl-1][y]=='H')
                {
                    FirstField[xl-1][y]='X';
                    life1--;
                    if(TestShipPl(x,y))
                        CompAt1(x,y);
                    else
                    {
                        corab=0;
                        PotopPlayer(x,y);
                    }
                }
                else
                    if(FirstField[xl-1][y]=='o')
                        CompAt1(x,y);
                    else
                        if (xl-1>=0)
                            FirstField[xl-1][y]='o';
                        else
                            CompAt1(x,y);
            }
        }
        z3:;
}

void CompAt()
```

```
{
    if (life1==0)
        goto z1;
    if (corab==0)
    {
        int x=1+rand() % 10;
        int y=1+rand() % 10;
        while(FirstField[x][y]=='o' ||
FirstField[x][y]=='X')
        {
            x=1+rand() % 10;
            y=1+rand() % 10;
        }
        if (FirstField[x][y]=='H')
        {
            FirstField[x][y]='X';
            life1--;
            if(TestShipPl(x,y))
            {
                corab=1;
                coordx=x;
                coordy=y;
                CompAt1(x,y);
            }
            else
            {
                corab=0;
                PotopPlayer(x,y);
            }
        }
        else
            FirstField[x][y]='o';
    }
    else
    {
        CompAt1(coordx,coordy);
    }
    z1:;
    PrintFields();
}

 int main()
{
    string prov;
    srand ( time(NULL) );
```

```
    for(int i=0; i<12; i++)
    for(int j=0; j<12; j++)
    {
            FirstField[i][j]='.';
        SecondField[i][j]='.';
        ThirdField[i][j]='.';
    }
    PrintFields();
    CompShip(2);
    cout << "Do you like select autamatic position to
your ships?" << endl;
    while( prov!= "yes" &&  prov!= "no")
    {
    cout << "Select the answer (yes/no):" << endl;
       cin >> prov;
    }
    if (prov=="yes")
        CompShip(1);
    else
    SetShip();
    while(life1>0 && life2>0)
    {
        PrintFields();
        PlayerAt();
        CompAt();
    }
    Sleep(1000);
    system("cls");
    Sleep(1000);
    if( life2==0)
    {
        cout << "You WIN !!!" << endl;
        Sleep(2000);
        cout << "Congratulations " << endl;
    }
    else
        cout << "You lose =(" << endl;
    Sleep(5000);
    return -1;
}
```