**Федеральное государственное образовательное бюджетное учреждение  
высшего образования  
«Финансовый университет при Правительстве РФ»**

**Отчет по практикуму по программированию. Вариант 9.**

Выполнил студент

Зайцев Никита Валерьевич

Группы ПИ20-2

Проверил доцент

Бочаров Михаил Иванович

Москва 2020

**Условие задачи:**

Реализовать программу, с которой можно играть в игру **«Морской бой»**. Программа автоматически случайно расставляет на поле размером 10 на 10 клеток: 4 1-палубных корабля, 3 2-палубных корабля, 2 3-палубных корабля и 1 4-х палубный. Между любыми двумя кораблями по горизонтали и вертикали должна быть как минимум 1 незанятая клетка. Программа позволяет игроку ходить, производя выстрелы. Сама программа НЕ ходит т.е. не пытается топить корабли расставленные игроком.

Взаимодействие с программой производится через консоль. Игровое поле изображается в виде 10 текстовых строк и перерисовывается при каждом изменении состояния поля. При запросе данных от пользователя программа сообщает, что ожидает от пользователя (в частности, координаты очередного «выстрела») и проверяет корректность ввода. Программа должна уметь автоматически определять потопление корабля и окончание партии и сообщать об этих событиях.

**Листинг:**

*import* random  
*import* os  
   
size = 10  
monitoring = *True*s\_buffer = "^"  
s\_ship = "A"  
s\_space = "."  
s\_hit = "x"  
s\_destroyed = "W"  
s\_miss = "\*"  
   
ships\_list = [[1, 4], [2, 3], [3, 2], [4, 1]]  
   
   
*class* Board(*object*):  
   
 *def \_\_init\_\_*(*self*):  
 *self*.board = []  
 *self*.spawned = []  
   
 *def* create(*self*):  
 *for* row *in range*(size):  
 *self*.board.append([s\_space] \* size)  
   
 *def* random(*self*):  
   
 *for* ship *in* ships\_list:  
 *for* unit *in range*(ship[0]):  
   
 spawning = *True  
 while* spawning:  
  
 *global* refer  
 refer = random.randrange(2)  
 *if* refer == 0:  
 location\_y = random.randrange(size)  
 location\_x = random.randrange(size - (ship[1] - 1))  
 *else*:  
 location\_y = random.randrange(size - (ship[1] - 1))  
 location\_x = random.randrange(size)  
  
 offset = 0  
 *for* testing *in range*(ship[1]):  
 *if* refer == 0 *and self*.board[location\_y][location\_x + offset] != s\_space:  
 *continue  
 elif* refer == 1 *and self*.board[location\_y + offset][location\_x] != s\_space:  
 *continue* offset += 1  
 *if* offset == ship[1]:  
 spawning = *False* offset = 0  
 current\_ship = []  
 *for* marker *in range*(ship[1]):  
 *if* refer == 0:  
 *self*.board[location\_y][location\_x + offset] = s\_ship  
 current\_ship.append([location\_y, location\_x + offset])  
 *else*:  
 *self*.board[location\_y + offset][location\_x] = s\_ship  
 current\_ship.append([location\_y + offset, location\_x])  
 offset += 1  
 *self*.spawned.append(current\_ship)  
   
 *for* unit\_point *in* current\_ship:  
 *for* buffer\_point *in* ([0, 1], [0, -1], [1, 0], [-1, 0], [1, 1], [1, -1], [-1, 1], [-1, -1]):  
 b\_point\_y = unit\_point[0] + buffer\_point[0]  
 b\_point\_x = unit\_point[1] + buffer\_point[1]  
 *if* b\_point\_y *in range*(size) *and* b\_point\_x *in range*(size):  
 *if self*.board[b\_point\_y][b\_point\_x] == s\_space:  
 *self*.board[b\_point\_y][b\_point\_x] = s\_buffer  
   
 *def* updating(*self*, ship):  
 *for* unit *in* ship:  
 *for* buffer\_point *in* ([0, 0], [0, 1], [0, -1], [1, 0], [-1, 0], [1, 1], [1, -1], [-1, 1], [-1, -1]):  
 b\_point\_y = unit[0] + buffer\_point[0]  
 b\_point\_x = unit[1] + buffer\_point[1]  
 *if* b\_point\_y *in range*(size) *and* b\_point\_x *in range*(size):  
 *if self*.board[b\_point\_y][b\_point\_x] == s\_buffer:  
 *self*.board[b\_point\_y][b\_point\_x] = s\_miss  
 *elif self*.board[b\_point\_y][b\_point\_x] == s\_hit:  
 *self*.board[b\_point\_y][b\_point\_x] = s\_destroyed  
   
   
*def* print\_boards():  
 *print*("\n Ваше поле" + (" " \* (size + 5)) + "Поле противника")  
 *print*(" " + (" ".join(*str*(i) *for* i *in list*(*range*(size)))), end=(" " \* 2))  
 *print*(" " + (" ".join(*str*(i) *for* i *in list*(*range*(size)))))  
 *print*(" " + (" |" \* size), end=(" " \* 2))  
 *print*(" " + (" |" \* size))  
 n = 0  
 *for* i *in range*(size):  
 *if* monitoring:  
 *print*(*str*(n) + " - " + " ".join(*str*(i) *for* i *in* player.board[n]), end=(" " \* 2))  
 *print*(*str*(n) + " - " + " ".join(*str*(i) *for* i *in* ai.board[n]))  
 *else*:  
 *print*(*str*(n) + " - " + " ".join(*str*(i) *for* i *in* player.board[n]).replace(s\_buffer, s\_space), end=(" " \* 2))  
 *print*(*str*(n) + " - " + " ".join(*str*(i) *for* i *in* ai.board[n]).replace(s\_ship, s\_space).replace(s\_buffer, s\_space))  
 n += 1  
   
   
*def* press\_ent():  
 *input*("Нажмите ENTER чтобы продолжить.\n")  
   
   
*def* state\_of\_ships(enemy):  
 *global* destroy  
 destroy = *False  
 for* d\_ship *in* enemy.spawned:  
 damage = 0  
 *for* d\_unit *in* d\_ship:  
 *if* enemy.board[d\_unit[0]][d\_unit[1]] == s\_hit:  
 damage += 1  
 *if* damage == *len*(d\_ship):  
 enemy.updating(d\_ship)  
 enemy.spawned.remove(d\_ship)  
 destroy = *True  
   
def* clear():  
 os.system('cls' *if* os.name == 'nt' *else* 'clear')  
   
*print*("Добро пожаловать в игру \"Морской Бой\"\n")  
press\_ent()  
   
ai = Board()  
ai.create()  
ai.random()  
   
*while True*:  
 clear()  
 player = Board()  
 player.create()  
 player.random()  
 print\_boards()  
 *print*("Если вас не устраивает расположение ваших кораблей, то нажмите ENTER для новой генерации поля")  
 regenerate = *input*("Если вы готовы напишите \"да\"")  
 *if str*(regenerate.lower()) != "да":  
 *continue  
 else*:  
 *break  
   
print*("\nНачинаем! Готовтесь!\n")  
press\_ent()  
   
   
game = *True  
while* game:  
   
 clear()  
 print\_boards()  
 guessing = *True  
 while* guessing:  
   
 *print*("\nКораблей противника осталось: " + *str*(*len*(ai.spawned)) + ". ", end="")  
 *print*("Кораблей осталось у вас: " + *str*(*len*(player.spawned)) + ".")  
 *print*("^ - буфер, . - пусто, А - корабль, х - попал, W - уничтожен, \* - мимо")  
 guess\_x = *input*("Выберите X (столбик) для стрельбы: ")  
 guess\_y = *input*("Выберите Y (клетка) для стрельбы: ")  
   
 *if not* guess\_x.isdigit() *or not* guess\_y.isdigit():  
 *print*("\nВы ввели неверные координаты пожалуйста перепроверьте.")  
 *continue* guess\_x = *int*(guess\_x)  
 guess\_y = *int*(guess\_y)  
   
 *if not* (guess\_x *in range*(size)) *or not* (guess\_y *in range*(size)):  
 *print*("\nЭта клетка слишком далеко, чтобы поразить ее! Выберите другой.")  
 *continue  
   
 elif* ai.board[guess\_y][guess\_x] == s\_ship:  
 ai.board[guess\_y][guess\_x] = s\_hit  
 state\_of\_ships(ai)  
 *if* destroy:  
 *print*("\nВы уничтожили судно противника!", end=" ")  
 *else*:  
 *print*("\nВы повредили судно противника!", end=" ")  
 press\_ent()  
 *break  
   
 elif* ai.board[guess\_y][guess\_x] == s\_space *or* ai.board[guess\_y][guess\_x] == s\_buffer:  
 ai.board[guess\_y][guess\_x] = s\_miss  
 *print*("\nВы промахнулись(.", end=" ")  
 press\_ent()  
   
 *else*:  
 *print*("\nВы уже выстрелили в этот квадрат пожалуйста выберите другой")  
 *continue  
 break  
   
 if len*(ai.spawned) == 0:  
 *input*("ВЫ ПОБЕДИТЕЛЬ! ВЫ УНИЧТОИЛИ ВСЕ СУДНА ПРОТИВНИКА! Нажмите ENTER чтобы завершить игру.")  
 *break  
   
 if len*(player.spawned) == 0:  
 *print*("ВЫ ПРОИГРАЛИ! ПРОТИВНИК УНИЧТОЖИЛ ВСЕ ВАШИ СУДНА(")  
 *input*("Судна оставшиеся у противника: " + *str*(*len*(ai.spawned)) + ".")  
 *break*

Демонстрация работы программы. На рисунках 1-4 представлен пример работы программы.

Изображение выглядит как текст

Автоматически созданное описание

Рис. 1 – Результат работы программы

Изображение выглядит как текст

Автоматически созданное описание

Рис. 2 – Результат работы программы

Изображение выглядит как текст

Автоматически созданное описание

Рис. 3 – Некоректный ввод

Изображение выглядит как текст

Автоматически созданное описание

Рис. 4 – Уничтожение судна противника

Интернет источники

1. <https://docs.python.org/3/>