Бюджетное учреждение высшего образования   
Ханты-Мансийского автономного округа   
«Сургутский государственный университет»

Политехнический институт

Кафедра автоматики и компьютерных систем

**Отчет**

по лабораторной работе № 5 «Многомерные массивы»

по дисциплине «Программирование и основы алгоритмизации»

Выполнил: Комбаев И. Д.

студент группы 609-01

Проверил: Гришмановский П. В.

доцент кафедры автоматики и компьютерных систем

Сургут

2021 г.

Цель работы:

* закрепление теоретических знаний и приобретение практических навыков использования многомерных массивов на примере двухмерных.;

Задание: решить задачу в соответствии с вариантом индивидуального задания, выданным преподавателем, используя многомерные массивы.

Вариант 15. Задан массив 10 × 10 и значения его элементов (0 или 1). Определить, может ли данный массив рассматриваться как правильно заполненное поле игры «Морской бой»

**Вариант 15**

**Формальное описание задачи.**

Поле игры «Морской Бой» состоит из клеток 10 на 10. Отмеченная клетка – это «Часть корабля». «Корабль» — это несколько подряд идущих отмеченных клеток. Количество кораблей на поле в зависимости от их длины:

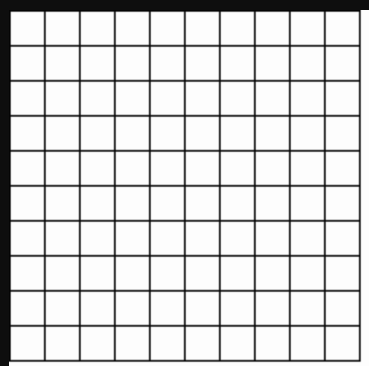
|  |  |
| --- | --- |
| Длина корабля  в клеточках | Количество  на поле |
| 4 | 1 |
| 3 | 2 |
| 2 | 3 |
| 1 | 4 |

Корабли в поле «Морского Боя» расположены по определенному правилу:  
 1) Вокруг корабля не могут находиться другие корабли (См. Рис. Зеленые кружки – клетки, в которых не могут находиться части других кораблей)

(Рисунок)

Учитывая это, наша задача:

1. Определить кол-во кораблей на поле
2. Проверить наличие пустого места вокруг кораблей



Условимся, что каждая клетка будет иметь свой индекс. Индекс первой клетки будет равен   
(0, 0), второй клетки (0, 1), третьей (0,2) и так далее. , где i – номер клетки в строке, начиная с левого края, j – номер клетки в столбце, в котором находится клетка, начиная с верхнего края.

Таким образом, мы можем выразить представление «Корабля» через множество «индексов клетки». Так, например корабль длинной в 4 клетке, расположенный горизонтально в левом верхнем углу можно представить как множество [(0,0), (0,1), (0,2), (0.3)]. А корабль, расположенный в том же месте, но вертикально можем выразить как [(0.0), (1.0), (2.0), (3.0)]. Как мы можем заметить, увеличивается лишь одно значение из множества индекса. Следовательно, проверка наличия корабля будет происходить через проверку соседних его клеток по горизонтали или вертикали.

Начинать проверку мы будем с левого верхнего угла и двигаться слева направо, сверху вниз. Если мы обнаружим клетку с содержимым 1, мы должны определить:  
 1) Является ли одиночный корабль или часть корабля.

2) Если это часть корабля – в какую сторону идет продолжение корабля

3) Есть ли вокруг корабля другие части корабля.

Первое условие достаточно легко проверить. Если в корабль находится в клетке с индексом

(a,b) нам нужно проверить наличия отмеченной клетки в клетках с индексом (a+1, b) и   
(a, b+1). Если таковых нет, значит мы имеем дело с одиночным кораблем.

Второе условие также заключается в проверке клеток с индексом (a+1, b) и

(a, b+1). Если какая-либо клетка является отмеченной – далее мы проверяем последующие клетки в том же направлении.

Третье условие заключается проверки наличия двух отмеченных клеток одновременно с индексом (a+1, b) и (a, b+1). Если вокруг части корабля находятся еще две части корабля – то значит другой корабль находится около другого корабля, что не является возможным.

Если мы нашли продолжение корабля справа от отмеченной клетки – то мы просто двигаемся дальше до тех пока не встретим пустую клетку справа или конец поля. В это же время считаем общее кол-во отмеченных клеток.

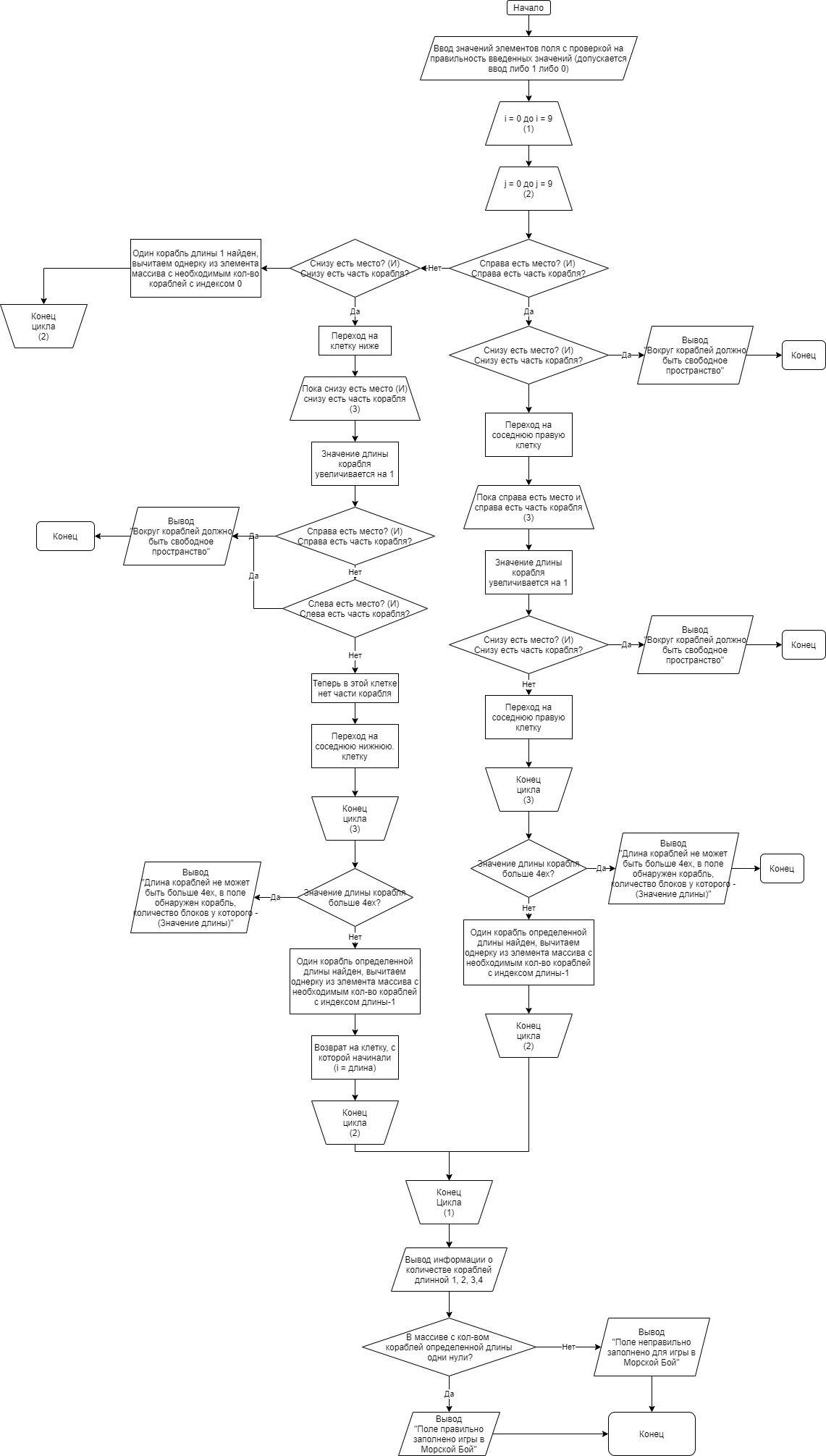
Если мы нашли продолжение корабля снизу от отмеченной клетки – то мы начинаем двигаться вниз от отмеченной клетки до тех пор, пока не встретим пустую клетки снизу или конца поля. В это же время считаем общее кол-во клеток и выставляем в этих клетках новое значению равным нулю. Далее возвращаемся к той клетки, с которой мы начали. Изменять значение нужно для того, чтобы программа в последствии вновь не встретила эту отмеченную клетку.

Во время движения по клетка мы также смотрим на соседние клетки. Если в них есть часть корабля, значит это не подходит условию (3)

**Алгоритм решения:**

1. Ввести исходные данные – Заполненное поле для «Морского Боя»
2. Проверить введенные значения исходя из условий задачи
3. Посчитать кол-во кораблей на поле
4. Проверить наличии свободного места вокруг кораблей
5. Вывести результат

**Блок схема**



**Листинг программы.**

1. #include <stdio.h>
2. #include <stdlib.h>
4. **int** main() {
6. // Для откалдки, это пример правильно заполненного поля
7. /\*  int pole[10][10] =
8. {{0, 0, 0, 0, 0, 0, 1, 1, 1, 0},
9. {1, 0, 1, 1, 0, 0, 0, 0, 0, 0},
10. {0, 0, 0, 0, 0, 0, 0, 0, 1, 0},
11. {0, 1, 0, 1, 0, 0, 0, 0, 0, 0},
12. {0, 0, 0, 1, 0, 0, 1, 1, 1, 0},
13. {1, 0, 0, 1, 0, 0, 0, 0, 0, 0},
14. {0, 0, 0, 1, 0, 0, 0, 0, 0, 0},
15. {0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
16. {0, 0, 0, 0, 0, 1, 1, 0, 1, 1},
17. {0, 0, 0, 0, 0, 0, 0, 0, 0, 0}}; \*/
19. **int** i, j, lenCor, cor[4] = {4, 3, 2, 1};
20. **int** pole[10][10];
21. system("chcp 1251 > nul");
23. //В практикуме написано, что ввод элементов должен быть поэлементым с указанием индекса
24. **for**(i = 0; i < 10; i++) {
25. **for**(j = 0; j < 10; j++) {
26. **while**(1) {
27. printf("Введите элемент с индексом [%d][%d]: ", i, j);
28. scanf("%d", &pole[i][j]);
29. **if** (!(pole[i][j] == 0 || pole[i][j] == 1)) {
30. printf("Введенное значение должно равняться либо 1, либо 0\n");
31. **continue**;
32. }
33. **break**;
34. }
35. }
36. }
38. **for**(i = 0; i < 10; i++) {
39. **for** (j = 0; j < 10; j++) {
40. **if**(pole[i][j] == 1) {
41. **if**(j != 9 && pole[i][j+1] == 1) { // Если справа от клетки есть еще один блок корабля
42. //Так как мы уже имеем горизонтальную "линию единиц" - единица снизу означает,
43. //что какой-то корабль находится слшиком близко к другому кораблю, что неприемлимо
44. **if** (i != 9 && pole[i + 1][j] == 1) {
45. printf("Вокруг кораблей должно быть свободное пространство\n");
46. system("pause");
47. **return** 0;
48. }
49. j++;
50. lenCor = 1; // Длина корабля = 1, изначальная точка отсчета
51. **while**(j != 10 && pole[i][j] == 1) {
52. lenCor++;
53. //Так как мы уже имеем горизонтальную "линию единиц" - единица снизу означает,
54. //что какой-то корабль находится слшиком близко к другому кораблю, что неприемлимо
55. **if** (i != 9 && pole[i + 1][j] == 1) {
56. printf("Вокруг кораблей должно быть свободное пространство\n");
57. system("pause");
58. **return** 0;
59. }
60. j++;
61. }
62. //Массив cor хранит значения допустимого количество (index + 1)-блоковых кораблей
63. //Если в конце массив будет иметь 4 нуля - значит стандарт выполнен
64. **if**(lenCor > 4) {
65. printf("Длина кораблей не может быть больше 4ех, в поле обнаружен корабль, количество блоков у которого - %d\n", lenCor);
66. system("pause");
67. **return** 0;
68. }
69. cor[lenCor - 1]--;
70. } **else** **if** (i != 9 && pole[i + 1][j] == 1) {
71. i++;
72. lenCor = 1;
73. **while** (i != 10 && pole[i][j] == 1) {
74. lenCor++;
75. **if** (j != 9 && pole[i][j + 1] == 1) {
76. printf("Вокруг кораблей должно быть свободное пространство\n");
77. system("pause");
78. **return** 0;
79. }
80. **if** (j != 0 && pole[i][j - 1] == 1) {
81. printf("Вокруг кораблей должно быть свободное пространство\n");
82. system("pause");
83. **return** 0;
84. }
85. //Мы делаем нижестоящие клетки нулями, чтобы в последствии программа не считала нижестоящие единицы, так как мы
86. //после фиксирования вертикального корабля возращаемся к отправной точке
87. pole[i][j] = 0;
88. i++;
89. }
90. **if**(lenCor > 4) {
91. printf("Длина кораблей не может быть больше 4ех, в поле обнаружен корабль, количество блоков у которого - %d\n", lenCor);
92. system("pause");
93. **return** 0;
94. }
95. cor[lenCor - 1]--;
96. i -= lenCor; // Возвращение к отправной точкии
97. } **else** { // Если справа и снизу нет заполненной клетки - значит это одиночный корабль
98. cor[0]--;
99. }
100. }
101. }
102. }
104. // Информирование пользователя о состояние игрового поля
105. **for** (i = 0; i < 4; i++) {
106. printf("Кораблей длиной %d блок%c: %d\n", i + 1, i == 0 ? '\0' : 'а', 4 - i - cor[i]);
107. }
109. **if** (cor[0] == 0 && cor[1] == 0 && cor[2] == 0 && cor[3] == 0) {
110. printf("Поле правильно заполнено для игры в Морской Бой\n");
111. } **else** {
112. printf("Поле неправильно заполнено для игры в Морской Бой\n");
113. }
115. system("pause");
116. **return** 0;
117. }

**Пояснения к программе.**

Ввод и вывод выполняется максимально эффективно, потому что занимают всего N итераций, где N – кол-во элементов.

Проверка всех условий вышла тоже весьма эффективной. Эффективность таковой O(N), потому что максимум итераций, который выполнен алгоритм – N\*2 (При условии, что все корабли будут расположены вертикально).

Для организации дружественного интерфейса пользователя перед вводом данных выводится сообщение для пользователя. Ввод и вывод организован при помощи стандартных функций scanf() и printf(), для этого подключен заголовочный файл stdio.h.

Заголовочный файл stdlib.h необходим для использования функции system(), чтобы приостановить выполнение программы перед завершением (выполняется команда «pause» операционной системы).

**Выводы:**

Я научился и также закрепил умение работать с многомерными массивами и заполнять таковые.