

Мартин Григоров

ФКСТ 48 гр. 121216011

на тема:

5. Бойни кораби

Проект за оценка

по Системно Програмиране

**Технически университет – София**

Дата: 01.05.2019 г.

Съдържание

[Условие и анализ на изготвеното приложение 2](#_Toc7881550)

[Функционално описание на приложението 2](#_Toc7881551)

[Изпълненние на функционалностите 3](#_Toc7881552)

[Експериментални данни 10](#_Toc7881553)

[Изходен код 11](#_Toc7881554)

# Условие и анализ на изготвеното приложение

Условие

Дадена е дъска с размери MxN единици. Потребителят задава няколко „кораба“, които се запаметяват в един контейнер, като имат дискретни координати върху тази дъска. „Кораб“ се нарича всеки правоъгълник, който се определя еднозначно чрез две точки – срещуположни негови върхове (корабите са разположени винаги хоризонтално на дъската и не може да се извъртат по диагонал). След като корабите са въведени от файл, започва цикъл за тяхното „обстрелване“. Потребителят въвежда координати на точка – място, където се стреля. Ако точката попада вътре в даден кораб, се извежда информация, че е потопен и той се премахва от дъската (контейнера). Играта продължава до потапянето на всички кораби. Напишете функционалността за въвеждане на информацията на корабите по дъската, така че те да не могат да се застъпват един върху друг и реализирайте функционалността за обстрелването на корабите. Записвайте във файл резултата от всеки тур.

Анализ

Програмата работи с два процеса.

При стартирането на програмата, първият процес изисква от потребителя да въведе броя на корабите, с които ще играе. Генерира се дъска MxN (двумерен масив), върху която са разположени на произволен принцип толкова кораби, колкото е задал потребителят. Корабите се разполагат хоризонтално върху дъската, така че да не се застъпват. Дъската (двумерният масив) се записва в „FIFO” файл.

Стартира се вторият процес, който прочита дъската от „FIFO” файла. Започва цикълът на обстрелване, в който потребителят въвежда координати на точката, която ще обстрелва. Ако точката попада върху някой от корабите, се извежда съобщение, че той е потопен. Цикълът продължава до потапянето на всички кораби, като след всяко обстрелване резултатът се записва в бинарен файл.

# Функционално описание на приложението

Вход/Изход

Приемане на входни данни от потребителя - брой кораби и точка на обстрелване.

Извеждане на текущото състояние на дъската – символът „х“ показва точка от потопен кораб, а символът „\*“ представлява точка от дъската.

Процеси

Създаване на два процеса:

* Parent процес – създава дъска с размер MxN и поставя съответния брой кораби, зададени от потребителя
* Child процес – потребителят обстрелва корабите. Това продължава, докато всички кораби не са потопени.

Комуникацията между двата процеса е реализирана чрез „named pipe”.

Запис във файл

Резултатите от всеки ход се записват в бинарен файл.

Други функционалности

* Създаване на точка с координати х, у
* Генериране на произволни координати х, у
* Създаване на кораб с начална и крайна точка
* Поставяне на кораб върху дъската
* Премахване на кораб от дъската

# Изпълнение на функционалностите

Структура „point”

Структурата „point“ описва точка от дъската. Състои се от два параметъра - *х* и *у*, от тип **int**.



Структура „ship”

Структурата „ship“ описва кораб от дъската. Състои се от три параметъра – *start,*  *end* и *value*. Параметрите *start* и *end* са от тип „**point**“ и представляват съответно начална и крайна точка на кораба. Параметърът *value* е от тип **int** и представлява стойност, с която се описва даден кораб върху дъската.



Функция „new\_point“

Функцията „new\_point” приема два аргумента от тип **int** – *х* и *у***.** Създава запис от тип „**point**“ и като резултат връща създадения запис.



Функция „generate\_random\_ship“

Функцията „generate\_random\_ship” не приема никакви аргументи. В нея се създава

кораб с произволни координати и размер (между 2 и 5 точки). Резултатът, който връща, е създаденият кораб.



Функция „remove\_ship“

Функцията „remove\_ship” премахва кораб от дъската, в зависимост от това дали точката на атака попада върху него, или не.

При прeмахнат кораб, данните за него се визуализират в конзолата и се записват в бинарен файл.

Входни параметри:

* **ship[]** s – всички кораби, поставени върху дъската
* **int** num\_of\_ships – броят на корабите, поставени върху дъската
* **point** attack – точката на обстрелване, въведена от потребителя
* **int** (\*board)[N][M] – дъската с поставените кораби (подава се адресът ѝ)
* **int** result\_fd - „file descriptor“ на файла, в който се записват резултатите

Резултатът от функцията е цяло число. При успешно премахнат кораб се връща „0“, а при неуспех – „-1“.

\*Кодът на функцията се намира на следващата страница с цел нагледност.

Код на функцията „remove\_ship”:



Функция „place\_ship“

Функцията „place\_ship“ приема два аргумента. Първият е от тип „**ship“** и представлява кораб, който ще бъде поставен върху дъската. Вторият аргумент е от тип **int(\*)[N][M]** и представлява дъската, върху която ще бъде поставен съответният кораб. Като втори параметър се подава адресът на дъската (&board), така че след извикването на функцията, подаденият двумерен масив board[N][M] се променя и извън функцията.

Резултатът, който връща функцията, е от тип **int**. При успешно поставен кораб, функцията връща „0“ , а при неуспех – „ -1“



Функция „print\_board“

Функцията „print\_board” визуализира текущото състояние на дъската с корабите. Приема като аргумент двумерен масив от тип **int** (дъската за игра) и не връща резултат.



Функция „main“

При стартирането на програмата се изпълнява „main“ функцията в която са използвани всички останали функции. Първо се стартира „parent“ процеса след което и „child“ процеса.



„Parent“ процес

При изпълнението на този процес, първо се приема броят на корабите, с които ще играе потребителят. След това се генерират кораби с произволни координати, поставят се върху дъската и се добавят в масива от кораби. Накрая, дъската с поставените кораби, масива и броя на корабите се записват във „fifo“ файл. Следва изпълнението на „child” процеса.



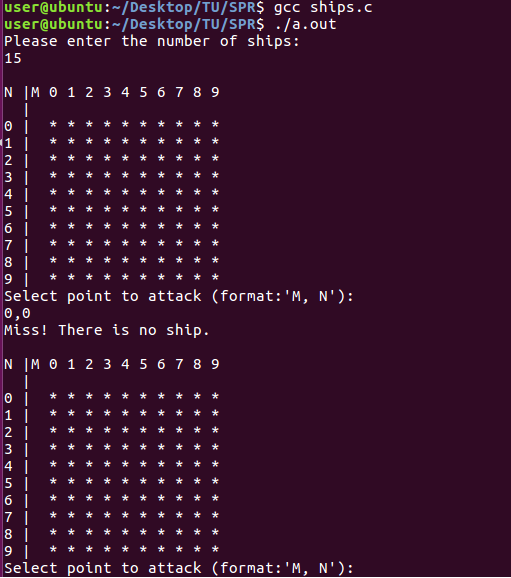
„Child“ процес

При изпълнението на този процес първо се прочитат данните от „fifo“ файла, след което започва обстрелването на корабите. Изисква се от потребителя да въвежда точка на атака, която се подава към функцията „remove\_ship“. Прави се проверка и ако точката попада върху някой кораб, той се премахва от дъската. След всяко обстрелване дъската с корабите се визуализира в конзолата и резултатът от обстрелването се записва в бинарен файл.

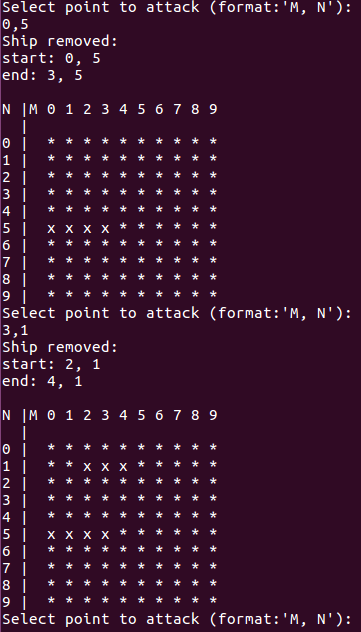


# Експериментални данни

Пример за неуспешно обстрелване на „кораб“



Пример за успешно обстрелване на „кораб“



Тестване на програмата

Програмата е тествана с дъска 10х10 точки. Работи нормално с брой на корабите - до 15 за съответния размер.

При задаване на брой кораби – над 15 (за размер 10х10), програмата се бави при поставянето на корабите. Това се случва поради използването на функцията „rand()”, която е използвана за генерирането на кораб с произволни координати и размер. Например, когато на дъската вече са поставени кораби и са останали места само за кораби с размер от 2 или 3 точки, но се генерира кораб с произволен размер от 4 или 5 точки, той не може да бъде поставен. В този случай програмата генерира нов кораб с произволни координати и размер. Ако новият кораб също се състои от повече точки и няма място на дъската за него или попада на координати, на които вече има поставен кораб, отново се генерира нов. Това продължава, докато не се поставят всички кораби.

При задаване на брой кораби, по-голям от този, който може да побере дъската програмата влиза в безкраен цикъл.

# Изходен код

<https://github.com/martog/TU/blob/master/SPr/ships.c>