Основы программирования

Лабораторная работа №2

Чтение/запись текстовых файлов

Задание

Необходимо реализовать логическую игру согласно варианту. Программа должна содержать:

1) Главное меню:
    a. Начать игру
    b. Выбрать уровень или загрузка (см. доп. задание по вариантам)
    c. Таблица рекордов
    d. Выход
2) Экран игры
3) Экран завершения игры

При запуске программы отображается главное меню.

Начать игру

Программа загружает уровень игры из текстового файла с расширением *.map (см. пример уровней под описанием вариантов) и открывает экран игры

Таблица рекордов

Загружает из текстового файла таблицу рекордов. Формат файла:

```
Результат1 Имя1
Результат2 Имя2
1234 Петров К
404 Миша
99 Ник
12 Василий
```

Данные в файле отсортированы от лучшего результата к худшему. Программа выводит первые 5 записей в формате (результат выровнен по правому краю):

```
1. Имя1           Результат1
2. Имя2           Результат2
3. Петров К             1234
4. Миша                  404
5. Ник                    99
```

Если файл таблицы рекордов пуст или не существует, программа должна написать «Результатов нет»

При нажатии на любую из клавиш – переход в главное меню

Выход

Завершает работу программы

Экран игры

Отображает игровое поле и текущее состояние игры. При нажатии на F5 во время игры – перезапуск уровня. При нажатии Esc – выход в главное меню

Экран завершения игры

Запрашивает имя пользователя и добавляет новый результат в файл таблицы рекордов (см. выше). Записи в файле должны быть отсортированы от лучшего результата к худшему

Отображает таблицу рекордов в том же формате, что и пункт «Таблица рекордов» главного меню. Но если текущий результат не попал в пятерку лучших, он отображается вместо пятого результата с указанием присвоенного места:

```
1. Имя1            Результат1
2. Имя2            Результат2
3. Петров К              1234
4. Миша                   404
7. Новый результат         11
```

Строка с текущим результатом должна выделяться цветом.

После отображения таблицы рекордов и нажатия на любую клавишу, происходит переход в главное меню.

Варианты

N	Логическая игра	Доп. задание
1	SpaghettiCode	Загрузка и сохранение
2	Boxing	Загрузка и сохранение
3	PairProgramming	Загрузка и сохранение
4	IntegrationContinuum	Выбор уровня
5	TestPass	Выбор уровня
6	ParallelComputing	Выбор уровня
7	HotFixing	Загрузка и сохранение
8	ParallelComputing	Загрузка и сохранение
9	GreedyAlgo	Выбор уровня
10	TestPass	Загрузка и сохранение
11	HotFixing	Выбор уровня
12	MsSpecification	Выбор уровня
13	PairProgramming	Выбор уровня
14	DeadLine	Загрузка и сохранение
15	MsSpecification	Загрузка и сохранение
16	DeadLine	Выбор уровня
17	SpaghettiCode	Выбор уровня
18	IntegrationContinuum	Загрузка и сохранение
19	Boxing	Выбор уровня
20	GreedyAlgo	Загрузка и сохранение

Варианты доп. задания

Выбор уровня

В главном меню второй пункт – «Выбрать уровень». При нажатии на него, программа отображает перечень доступных уровней (не менее 3) из специальной папки (все файлы с расширением *.map). Пользовать может выбрать любой из них.

Пункт «Начать игру» в главном меню запускает первый из уровней

На каждый уровень должна быть своя таблица рекордов. При выборе пункта «Таблица рекордов» главного меню, программа должна отобразить таблицы рекордов всех уровней поочередно.

Загрузка и сохранение

В главном меню второй пункт – «Загрузка». При нажатии на него программа запускает игру и загружает ранее сохраненное состояние игры из текстового файла (формат определяете сами, например, файл содержит текущее положение игрока и кол-во сделанных ходов). Если файла сохранения нет, программа пишет «Сохранений нет»

При нажатии на F2 во время игры происходит сохранение состояния игры в файл и отображение текста «Игра сохранена».

В этом варианте достаточно одного уровня игры

Варианты игр

SpaghettiCode

Однажды программист, уставший создавать на каждую задачу отдельную функцию, решил создать одну Единую Универсальную Функцию, которая решит все его проблемы… или создаст новые?

Игра представляет собою поле, на котором присутствуют следующие элементы:

- Начала луча (V)
- Целевая точка (E)
- Стена (#)
- «Фича» (F)
- «Баг» (B)

Необходимо провести луч, начинающийся от точки V (начало всегда направлено вниз) в точку финиша E. Для этого игрок может в свободные ячейки уровня расставлять зеркала, отражающие луч (“/” и “\”). Каждый ход игрока (добавить зеркало или убрать) отнимает одно очко. Если луч проходит сквозь объект «Фича», прибавляется 10 очков. Если луч проходит сквозь объект «Баг» вычитается 5 очков. Игра завершается, когда луч достигает точки финиша E.

Пример уровня:

```
#V##################
#       #          B #
#       #   F    #    #
#B      ####   #    #
#B           F #   B #
#    F          #    #
###################E#
```

Пример решения

```
#V##################
#|      #/-------\B #
#\-\    #\-F-\ # |   #
#B |    #### | # |   #
#B |         F # |B #
#  \F-------/ # \-\#
##################E#
Очки: 20
```

Boxing

Однажды программист читал документацию языка C# про упаковку типов. Эта возможность ему так понравилась, что, зайдя на склад, он решил упаковать все лежащие на полу типы. К счастью, на складе как раз подвернулись подходящего размера коробки…

Игра представляет собою поле, на котором присутствуют следующие элементы:

- Стартовая точка игрока (M)
- Целевая точка (_)
- Коробка (b)
- Коробка, стоящая на целевой точке (B)
- Стена (#)

Необходимо переместить все коробки на целевые точки. Игрок движется в четырех направлениях (←,↑,→,↓). Если на его пути стоит коробка, то, если за коробкой находится пустое место, то коробка сдвигается вместе с игроком. Рейтинг игрока строится по числу сделанных шагов (чем меньше, тем лучше). Игра завершается, когда все коробки находятся на целевых точках.

Пример уровня:

```
  #####
###   #
#_Mb  #
### b_#
#_##b #
#  # _ ##
#b Bbb_#
#    _  #
########
```

PairProgramming

Однажды программист изучал разные методологии программирования и ему как раз попался под руку второй программист. Теперь решать задачи в два раза проще! Или нет?

Игра разделена на два поля. На каждом из полей могут находится следующие элементы

- Стартовая точка игрока (M)
- Целевая точка (_)
- Ловушка непонимания, раздоров и ссор (X)
- Стена (#)

На каждом из полей находится по персонажу, которыми одновременно управляет игрок с помощью клавиш (←,↑,→,↓). При нажатии на клавишу стрелки персонаж на левом поле двигается по направлению стрелки, персонаж на правом поле, двигается в обратную сторону. Если один персонаж упирается в стенку, а у второго путь свободен, то перемещается только второй.

Необходимо провести персонажей на обоих полях к целевым точкам. Если персонаж наступает на объект ловушки, добавляется 10 штрафных баллов. Рейтинг игрока строится по числу сделанных шагов + штрафные баллы (чем меньше, тем лучше). Игра завершается, когда оба персонажа одновременно будут стоять на целевых точках

Пример уровня:

```
###############
#       ##    XX#
#X  #  #  ##      #
#M  #_   ##_  #M  #
#X  #    ##       #
#        ###   XX#
###############
```

IntegrationContinuum

Однажды программист решил настроить систему непрерывной интеграции на проекте. Как хорошо, когда всем управляют автоматические скрипты! Теперь уж точно не будет ошибок…

Игра представляет собою поле, на котором присутствуют следующие элементы:

- Стартовая точка шара (o)
- Целевая точка (U)
- Полезный результат (*)
- Стена (#)

Необходимо довести шар от стартовой точки до целевой. Игрок управляет шаром с помощью клавиш (←,↑,→,↓). При нажатии на клавишу стрелки игрок толкает шар по направлению стрелки: шар двигается прямолинейно пока не упрется в стенку или он не попадет на целевую точку. При пересечении шаром полезного результата (*) добавляется 5 очков. Каждый толчок шара отнимает одно очко. Если шар выходит за пределы уровня вычитается 100 очков, игра завершается. При пересечении шаром целевой точки игра также завершается.

Пример уровня:

```
 #      #
U       *
#*
#o  #  #  #
#*      *#
        ##
   ###
```

TestPass

Однажды программист решил запустить написанную им программу…

Игра представляет собою поле, на котором присутствуют следующие элементы:

- Стартовая точка игрока (M)
- «Баг» (B)
- Целевая точка (_)
- Стена (#)

Необходимо провести игрока от начальной точки к целевой, стараясь не пересекаться с объектами типа «Баг». Игрок управляет персонажем с помощью клавиш (←,↑,→,↓). При нажатии на

клавишу стрелки персонаж двигается по направлению стрелки, а объекты типа «Баг» двигаются в обратную сторону. Каждый раз, пересекаясь с объектом типа «Баг», игрок получает 10 штрафных баллов. Игрок пересекается с объектом типа «Баг», если он двигается на ячейку, занятую им или объект типа «Баг» наступает на персонажа. Если объект «Баг» наступает на другой «Баг», они сливаются в один. Ни игрок, ни «Баг» не могут пересекаться со стенками. Рейтинг игрока строится по числу сделанных шагов + штрафные баллы (чем меньше, тем лучше). Игра завершается, когда персонаж достигнет целевой точки

Пример уровня:

```
########
#B    #_#
#   # # #
#M    B #
#   B    #
#B       #
########
```

ParallelComputing

Однажды программист услышал про распределенные вычисления и тут же решил распараллелить свой любимый алгоритм. Что может пойти не так?

Игра представляет собою поле, на котором присутствуют следующие элементы:

- Подзадача (o)
- Целевая точка (_)
- Стена (#)

Необходимо провести все подзадачи к целевым точкам. Игрок управляет подзадачами с помощью клавиш (←,↑,→,↓). При нажатии на клавишу стрелки все подзадачи двигаются одновременно по направлению стрелки. Подзадача не может пересекаться со стенкой и другими подзадачами. Если одна из подзадач не может сделать ход, она остается на месте, а остальные двигаются. Рейтинг игрока строится по числу сделанных шагов (чем меньше, тем лучше). Игра завершается, когда все подзадачи достигнут целевых точек одновременно

Пример уровня:

```
#########
#   # #   #
#_  #  _o  #
##o##     #
#    o    #
#  _  #_  o#
#         #
#########
```

HotFixing

Однажды утром программист решил заглянуть в трекер задач и выбрал самую крутую фичу для разработки. И потом всю ночь разрабатывал.

Игра представляет собою поле, на котором могут находится следующие элементы:

- Стартовая точка игрока (M)
- Стена (#)
- «Фича» (F)
- «Баг» (B)

Изначально на поле находится игрок и один объект «Фича». Игрок может перемещаться по полю в четырех направлениях (←,↑,→,↓). Как только игрок пересекается с объектом «Фича», игроку прибавляется 5 баллов, старый объект удаляется и в случайных позициях (на свободных ячейках) появляются объекты «Фича» и «Баг». Объекты «Баг» двигаются синхронно с игроком, но в обратную сторону. Как только игрок пересекается с объектом типа «Баг», игра заканчивается.

Пример уровня:

```
#########
# #       #
# # #F    #
#   #     #
# M   #   #
# ##   # #
#         #
#########
```

GreedyAlgo

Однажды программист решил разработать жадный алгоритм для решения задачи. Но на одном алгоритме он остановиться не сумел.

Игра представляет собою поле, на котором находятся следующие элементы:

- Стартовая точка игрока (M)
- Стена (#)
- «Фичи» разного номинала (3, 5, 7)

У игрока есть два счетчика: число накопленных баллов и число оставшихся ходов. Игрок может перемещаться по полю в четырех направлениях (←,↑,→,↓), расходуя при этом оставшиеся ходы. Изначально у игрока в запасе 10 ходов. Как только игрок пересекается с объектом «Фича», игроку прибавляется 3, 5 или 7 баллов и в два раза больше дополнительных ходов, старый объект фичи удаляется и в случайной позиции (на свободной ячейке) появляется новый объект «Фича» со случайным номиналом 3, 5 или 7. Игра заканчивается, как только у игрока не остается больше ходов

Пример уровня:

```
#########
# #       #
#5# #3    #
#   #     #
# M   #   #
# ##   # #
#     7   #
#########
```

MsSpecification

Однажды программист согласился разработать программу без согласования технического задания…

Игра представляет собою лабиринт, на котором находятся следующие элементы:

- Стартовая точка игрока (M)
- Целевая точка (F)
- Стена (#)

Игрок может перемещаться по лабиринту в четырех направлениях (←,↑,→,↓). Во время игры ему виден не весь лабиринт, а только область радиусом в две ячейки вокруг него, а также целевая точка. Рейтинг игрока строится по числу сделанных шагов (чем меньше, тем лучше). Игра завершается, когда игрок достигает целевой точки

Пример уровня:

```
##############
#M#    #       #
# ### ## ### #
#       #    # #
# ### #### # #
# #    #     # #
#    # # #### #
# # ### #F## #
# #      #     #
##############
```

Во время игры:

```
~~~~~~~~~~~~~~
~~~~~~~~~~~~~~
~~~~~~~~~~~~~~
~~~~~~~~~~~~~~
~~###~~~~~~~~~~
~ #   #~~~~~~~~~
~   M# ~~~~~~~~~
~ # ##~~~F~~~~
~~#   ~~~~~~~~~~
~~~~~~~~~~~~~~
```

DeadLine

Однажды программист решил посмотреть на календарь, а потом на свои задачи, еще раз на календарь и снова на задачи. И пошел за кофе. А потом еще раз.

Игра представляет собою поле, на котором находятся следующие элементы:

- Стартовая точка игрока (M)
- «Фича» (F)
- Стена (#)
- Чашка кофе (C)

У игрока есть два счетчика: число накопленных баллов и число оставшихся ходов. Игрок может перемещаться по полю в четырех направлениях (←,↑,→,↓), расходуя при этом оставшиеся ходы. Его задача – собрать все объекты типа «Фича» на поле. За каждый такой объект он получает 5 баллов. Изначально у игрока в запасе 10 ходов. По пути он может собрать объект типа «Чашка кофе», который дает дополнительно от 3 до 15 дополнительных ходов (выбирается случайным образом). После сбора объекта типа «чашка кофе» на поле в случайном свободном месте появляется еще один такой объект. Игра завершается, когда у игрока кончаются ходы или он собрал все объекты типа «Фича». Во втором случае, число оставшихся ходов прибавляется к накопленным баллам игрока.

Пример уровня:

```
##############
#M   F      #F#
# ### ## # # #
# F C #C # # #
#   # ## #   #
# F        #F#
##############
```

Требование к отчету

- Постановка задачи
- Схема алгоритма решения (алгоритм для записи в таблицу рекордов + еще один алгоритм на выбор)
- Примеры выходных и выходных файлов
- Контрольный пример
- Код программы