

## Лабораторная работа №2

### Описание работы

Разработать диалоговое консольное приложение, реализация которого должна быть построена на принципах объектно-ориентированного анализа и проектирования. Диалоговые консольные приложения – консольные программы, работа с которыми ведётся в диалоговом режиме («запрос-ответ»). В качестве запроса может выступать ввод команды, либо выбор пункта меню, выведенного программой в консоль. После получения запроса программа может потребовать ввода необходимых для выполнения запрошенной операции данных. После получения необходимой информации программа осуществляет соответствующее действие, выводит результат работы в консоль и ожидает следующего запроса.

В процессе выполнения работы необходимо:

1. понять, какие существуют сущности и процессы в проектируемой системе;
2. выделить основные функции системы;
3. продумать логику работы пользователя с программой: предусмотреть набор действий пользователя и множество состояний программы;
4. реализовать объектно-ориентированную модель в виде программы.

### Общие требования к реализации

- Организация дружелюбного пользовательского интерфейса
  - проверка вводимых пользователем данных
  - вывод адекватных и понятных пользователю сообщений (в особенности – об ошибках).
- Отсутствие, по возможности, жёстко закодированных значений (например, «защитых» в код путей к файлам, магических чисел).
- Разработанная программа должна получать все необходимые данные из внешнего источника (например, из консоли, файла, базы данных, от удалённого сервера, и т.д.), то есть при запуске не содержит никакой информации.
- Реализация должна быть выполнена на одном из языков: C#, Java, Kotlin, C++. Другие языки следует согласовать с преподавателем.
- Архитектурно приложение должно быть представлено клиентским и серверным компонентом (по усмотрению, REST API, взаимодействие через RPC). Серверный компонент должен обеспечивать сохранение данных о сеансе использования.

## Варианты

### 1. Калькулятор

Пользователь вводит последовательно операнды и операции. После ввода каждого операнда программа отображает текущее результирующее значение, сопровождая его порядковым номером. В любой момент времени пользователь может вернуться к любому месту вычислений, введя вместо операции номер соответствующего значения.

При запуске программа выводит справку по использованию.

#### Пример сеанса работы:

```
V:\>MyCalc.exe
```

```
Usage:
```

```
when a first symbol on line is '>' – enter operand (number)
```

```
when a first symbol on line is '@' – enter operation
```

```
operation is one of '+', '-', '/', '*' or
```

```
'#' followed with number of evaluation step
```

```
'q' to exit
```

```
> 3
```

```
[#1] = 3
```

```
@: +
```

```
> 5
```

```
[#2] = 8
```

```
@: /
```

```
> 2
```

```
[#3] = 4
```

```
@: #2
```

```
[#4] = 8
```

```
@: *
```

```
> 10
```

```
[#5] = 80
```

```
@: q
```

```
V:\>
```

## 2. Записная книжка

После запуска программы пользователю отображается справка по использованию и меню с пронумерованными пунктами, где перечислены возможные операции: вывод всех записей, поиск контакта, добавление нового контакта, выход из записной книжки. Для выбора нужного пункта меню пользователь вводит соответствующий номер.

Добавление контакта осуществляется путем последовательного ввода различной контактной информации: имени, фамилии, номера телефона, адреса электронной почты и т.д.

Для поиска программа предлагает несколько опций: поиск по конкретным полям (только по имени, или только по номеру телефона, или по имени и фамилии), либо по все полям одновременно. После выбора режима поиска, пользователь вводит искомую подстроку (часть имени, либо часть номера телефона, и т.п.), получая в ответ информацию о подходящих под условия выборки контактах.

### Пример сеанса работы:

```
V:\>Contacts.exe
```

```
Enter the number of action and press [Enter]. Then follow instructions.
```

```
Menu:
```

```
1. View all contacts
```

```
2. Search
```

```
3. New contact
```

```
4. Exit
```

```
> 3
```

```
New contact
```

```
Name: Alex
```

```
Surname: Morgan
```

```
Phone: +71234567890
```

```
E-mail: Alex.Morgan@gmail.com
```

```
Contact created.
```

```
Menu:
```

```
1. View all contacts
```

```
2. Search
```

```
3. New contact
```

```
4. Exit
```

```
> 2
```

```
Search by
```

1. Name

2. Surname

3. Name and Surname

4. Phone

5. E-mail

> 3

Request: mor

Searching...

Results (1) :

#1 Name: Alex

Surname: Morgan

Phone: +71234567890

E-mail: Alex.Morgan@gmail.com

Menu:

1. View all contacts

2. Search

3. New contact

4. Exit

> 4

V:\>

### 3. Todo-list

Главное меню программы включает следующие пронумерованные пункты:

- добавление новой задачи
- поиск задач по тэгам вывод N наиболее актуальных задач
- выход.

Для выбора нужного пункта меню пользователь вводит соответствующий номер. При добавлении новой задачи пользователь последовательно вводит тему задачи, описание, дату к которой задача должна быть готова и тэги. Ввод тэгов продолжается до тех пор, пока пользователь не введет пустую строку. Для поиска пользователь вводит через пробел ключевые слова, наличие которых среди тэгов задачи является критерием выборки.

Вывод актуальных задач осуществляется в отсортированном по дате готовности порядке.

#### Пример сеанса работы:

```
V:\todo.exe
```

```
Enter the number of action and press [Enter]. Then follow instructions.
```

```
Menu:
```

- ```
1. Add task  
2. Search task  
3. Last tasks  
4.Exit
```

```
> 1
```

```
New task
```

```
Title: some task
```

```
Description: some text
```

```
Deadline: 10.11.2012
```

```
Tags (finish on empty line)
```

```
1: tagA
```

```
2: tagB
```

```
3:
```

```
Menu:
```

- ```
1. Add task  
2. Search task  
3. Last tasks  
4.Exit
```

> 2

Search tasks by tag: tagC

No such tasks

Menu:

1. Add task

2. Search task

3. Last tasks

4.Exit

> 3

Actual tasks:

1. Title: some task

Description: some text

Deadline: 10.11.2012

Tags: tagA, tagB

Menu:

1. Add task

2. Search task

3. Last tasks

4.Exit

> 4

V:\

#### 4. Книжный каталог

Информация о книге включает название, имя автора, жанры, дату публикации, аннотацию, ISBN.

Из главного меню программы пользователь, посредством выбора одного из пронумерованных пунктов, может производить следующие операции:

- добавление книги в каталог
- выборку информации о конкретной книге по:
  - названию или его фрагменту
  - имени автора ISBN
  - ключевым словам
- выход.

Поиск по ключевым словам должен возвращать краткие описания книг (без аннотации) в таком порядке, чтобы в начале выборки оказывались книги с наибольшим количеством найденных ключевых слов, а в конце – с наименьшим количеством. Если ключевое слово найдено в аннотации, это должно быть отражено в выборке.

## 5. Словарь

Программа представляет собой словарь однокоренных слов.

Запустив программу, пользователь может вводить слова. Если введенное слово известно программе, она выводит список однокоренных слов в порядке увеличения количества словообразующих частей. При этом выводимые слова разбиваются тире в соответствии с составом.

В случае, когда введенное слово программе не известно, пользователю задается вопрос, не хочет ли он добавить новое слово в словарь. Если да, то пользователь последовательно вводит в программу предкоренные части слова, затем пустую строку, корень и посткоренные части. Объединив введенные части слова, программа проверяет по изначально введенному слову, что все части введены правильно, и возвращается к ожиданию следующего слова.

Выход из программы осуществляется после ввода символа 'q' вместо слова.

### Пример сеанса работы:

V:\dict.exe

>новости

Неизвестное слово. Хотите добавить его в словарь (y/n)? Y

приставка:

корень: новост

суффикс или окончание: и

суффикс или окончание:

Слово "новост-и" добавлено.

> новостной

Неизвестное слово. Хотите добавить его в словарь (y/n)? Y

приставка:

корень: новост

суффикс или окончание: н

суффикс или окончание: ой

суффикс или окончание:

Слово "новост-н-ой" добавлено.

> новости

Известные однокоренные слова:

НОВОСТ-И

НОВОСТ-Н-ОЙ

> q

V:\



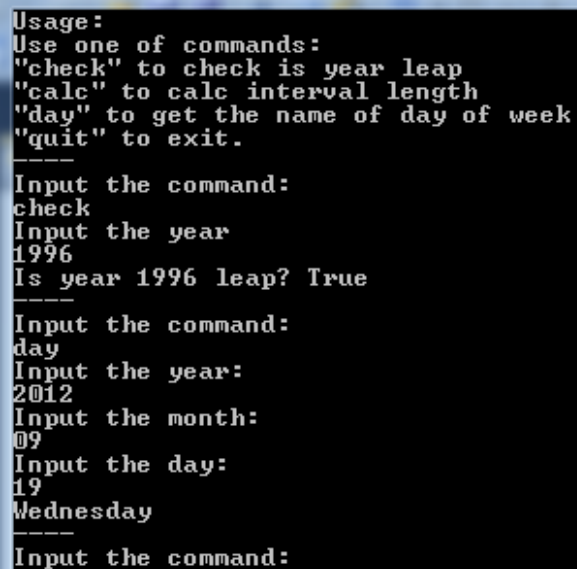
## 6. Календарь

При запуске программы пользователю предлагается справочная информация по использованию. Далее предлагается ввести команду, позволяющую выполнить одно из действий:

- проверить, является ли введенный год високосным
- рассчитать длительность интервала между двумя датами
- вывести названия дня недели по введенной дате.

Предполагается, что программа оперирует грегорианским календарём.

Пример сеанса работы:



```
Usage:
Use one of commands:
"check" to check is year leap
"calc" to calc interval length
"day" to get the name of day of week
"quit" to exit.
-----
Input the command:
check
Input the year:
1996
Is year 1996 leap? True
-----
Input the command:
day
Input the year:
2012
Input the month:
09
Input the day:
19
Wednesday
-----
Input the command:
```

## 7. Музыкальный каталог

При запуске программы пользователю предлагается справочная информация по использованию. Далее предлагается ввести команду, позволяющую выполнить одно из действий:

- осуществить поиск музыкальной композиции в каталоге по определенному критерию
- вывести информацию обо всех существующих в каталоге композициях
- добавить информацию о композиции в каталог
- удалить существующую в каталоге запись
- выйти из программы

Критериями поиска могут служить: имя (название) автора/исполнителя или название композиции. В качестве результата поиска в консоль должен выводиться список композиций в виде «исполнитель – название». Удаление или добавление записи осуществляется после ввода всей информации о композиции.

Пример сеанса работы:

```
Usage:
    Type one of commands:
        "list" to display all items of catalog
        "search" to go find items in catalog
        "add" to add new item
        "del" to remove some item from list
        "quit" to exit
-----
Input the command:
list
All compositions in catalog:
Людвиг Ван Бетховен - Ангелы и демоны
Бетховен - Тишина
Бетховен - К Элизе
Шуберт - Венский вальс
-----
Input the command:
search
Input the part of the name to find composition in the catalog:
Вивальди
No one item was found by this criteria.
-----
Input the command:
add
Input author's name:
Вивальди
Input the composition's name:
Времена года
-----
Input the command:
list
All compositions in catalog:
Людвиг Ван Бетховен - Ангелы и демоны
Бетховен - Тишина
Бетховен - К Элизе
Шуберт - Венский вальс
Вивальди-Времена года
-----
Input the command:
del
Input the full name of the track to remove:
Вивальди-Времена года
Track "Вивальди-Времена года" deleted.
-----
Input the command:
search
Input the part of the name to find composition in the catalog:
Вивальди
No one item was found by this criteria.
-----
Input the command:
```

## 8. Решатель уравнений

Программа предназначена для решения уравнений вида  $a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x + a_0 = 0$

Пользователю предлагается ввести степень полинома, затем – ввести соответствующие коэффициенты.

Далее выполняется решение уравнения, вывод конечного уравнения и его решения(ий) в консоль.

Программа должна сохранять информацию о результатах вычислений и предоставлять возможность вывода этой информации по порядковому номеру, который присваивается решению по окончании вычислений.

**Пример сеанса аналогичной программы, решающей квадратные уравнения:**

```
Usage:
Use one of commands:
    "solve" to select type of an equation and solve it
    "find" to get existing solution from memory.
    "quit" to exit.
-----
Input the command:
solve
Select the type of equation:
1 - k * x + b = 0
2 - a * x^2 + b * x + c = 0
1
Input the factors:
Factor 'k':2
Free member 'b':-4
#1: 2 * x + -4 = 0, x = 2
-----
Input the command:
2
Invalid command. Please, retry.
-----
Input the command:
solve
Select the type of equation:
1 - k * x + b = 0
2 - a * x^2 + b * x + c = 0
2
Input the factors:
Factor 'a':2
Factor 'b':-4
Factor 'c':2
#2: 2 * x^2 + -4 * x + 2 = 0, x = 1, 1
-----
Input the command:
find
Input the index:
1
2 * x^2 + -4 * x + 2 = 0 solution: x = 1, 1
-----
Input the command:
```

## 9. Черепашка

Черепашка – воображаемое устройство-робот, которое перемещается по плоскости, поворачиваясь в заданных направлениях (налево или направо на указанный угол). На плоскости существует система координат с взаимно перпендикулярными осями, которые пересекаются в точке (0;0).

Перемещение осуществляется вперёд или назад на указанное количество шагов. При перемещении черепашка оставляет за собой след заданного цвета и ширины; существует возможность поднять или опустить перо.

При запуске программа показывает краткую справку по использованию, а затем предлагает ввести команду. После ввода команды и её выполнения пользователю сообщается результат её выполнения.

При выполнении команды перемещения с опущенным пером в случае, если происходит пересечение линии с ранее нарисованной, считать, что образовалась фигура, для которой характерны свойства – координаты вершин, количество вершин, цвет линии. После того, как фигура образовалась, считать, что началось рисование новой. Программа должна предоставлять возможность просмотра параметров нарисованных фигур, а также всех выполненных команд.

V:\turtle.exe

Commands:

move N: command to change turtle's position on N steps.

angle N: command to change turtle's angle of direction to N degrees.

pd: command to put down the pen.

pu: command to put up the pen.

color {colorName}: command to change turtle's color of the pen to {colorName} color. Possible values: black, green.

list steps: command to show all executed steps.

list figures: command to show all properties of completed figures.

exit: command to exit the program.

Current color: black, pen state: put up, location (0; 0), direction: 0 degrees.

>move 5

Current color: black, pen state: put up, location (5; 0), direction: 0 degrees.

>angle 90

Current color: black, pen state: put up, location (5; 0), direction: 90 degrees.

>move 2

Current color: black, pen state: put up, location (5; 2), direction: 90 degrees.

>pd

Current color: black, pen state: put down, location (5; 2), direction: 90 degrees.

>move 5

Current color: black, pen state: put down, location (5; 7), direction: 90 degrees.

>angle 180

Current color: black, pen state: put down, location (5; 7), direction: 180 degrees.

>move 4

Current color: black, pen state: put down, location (1; 7), direction: 180 degrees.

>angle 270

Current color: black, pen state: put down, location (1; 7), direction: 270 degrees.

>move 5

Current color: black, pen state: put down, location (1; 2), direction: 270 degrees.

>angle 0

Current color: black, pen state: put down, location (1; 2), direction: 0 degrees.

>move 4

Current color: black, pen state: put down, location (5; 2), direction: 0 degrees.

You created rectangle.

>list figures

Rectangle with coordinates {(5; 2), (5; 7), (1; 7), (1, 2), (5; 2)}.

>exit

V:\

## 10. Логические элементы

Краткая справка по предметной области: логический элемент – это устройство, которое выполняет логическую функцию (операцию) над входными сигналами (операндами).

Соответственно, логические элементы выполняют следующие действия:

- «AND» - логическое «И»
- «OR» - логическое «ИЛИ»
- «NOT» - логическое «НЕ»
- «XOR» - исключающее «ИЛИ»

Все элементы, кроме «NOT», принимают на вход два сигнала, «NOT» – один. Также все элементы содержат один выход, на который подается результат логической операции.

Программа должна позволять строить логическую схему из множества элементов, которые добавляются соответствующей командой с указанием типа логического элемента. Для того чтобы множество элементов превратилось в полноценную схему, программа должна предоставлять возможность установить связь между входами и выходами элементов, а также установки значений на входы (если к нему не подключен выход другого элемента).

При старте программы выводится небольшая справка по её использованию. Пример сеанса работы:

```
V:\logicalScheme.exe
```

```
Commands:
```

```
add {elemType}: command to add element of {elemType} type. Possible {elemType} values: and, not, xor, or.
```

```
add {in/out} {name}: command to add input or output for element, which added by user on previous step.
```

```
connect {n}-{m}: command to connect {n}'s output and {m}'s input.
```

```
print: command to display output value of the scheme.
```

```
show {n}: command to display information about the logical element: name of this element, names of blocks, which connected with it or value on input(s). {n} is the number of logical element.
```

```
> add or 1
```

```
created 1:or
```

```
> add not 2
```

```
created 2:not
```

```
> add xor M
```

```
created M:xor
```

```
> add in A
```

```
created 4:input
```

```
> add in B
```

```
created 5:input
```

```
> add out R
```

```
created 6:out
```

```
> connect 4-1[0]
```

```
> connect 5-1[1]
```

```
> connect 1-2
```

```
> connect 2-3[0]
```

```
> connect 5-3[1]
```

```
> connect 5-6
```

```
> set A true
```

```
> set B false
```

```
> print
```

```
R: False
```

```
> show 3
```

```
3:xor(2:not, 4:B)
```