[1] Самостоятельная работа | **E-**

Тема: [**Типы данных**]

Работа с переменными и типами данных. Инициализация каждого типа данных по 5 раз. Вывод всех переменных в консоль по шаблону в примере.

Пример:

```
int a_1 = 10;
cout << "a_1 = " << a_1 << " | тип: int, размер: 4 б." << endl;

short s_1 = 1;
cout << "s_1 = " << s_1 << " | тип: short, размер: 2 б." << endl;
```

[2] Самостоятельная работа | **E-**

Тема: [**IF - ELSE**]

Работа с условными конструкциями **if - else**. Создать программу с пользовательским меню для выбора времен года. Меню состоит из 4 пунктов: лето, осень, зима, весна. Пользователь выбирает время года, программа выдает результат о выбранном времени года. Пример: пользователь вводит 2 (Это осень в программе), программа выдает: "Сейчас осень, следует одеться тепло и взять с собой зонт" и по аналогии со следующими временами. Обязательно использовать один логический блок **if / else if / else if / … / else**. При вводе другого значения, программа должна предупредить, что такого времени года нет.

[**-**] Программа "Времена года"

[**1**] Лето
[**2**] Осень
[**3**] Зима
[**4**] Весна

[**-**] Введите значение: 2
[**-**] Сейчас осень, следует одеться тепло и взять с собой зонт.

[3] Самостоятельная работа | **E-**

Тема: [**IF - ELSE**]

Работа с условными конструкциями if - else if - else. Написать диапазоны проверки числа от 0 - 10, 11 - 20 … 91 - 100. Пользователь вводит число от 1 до 100, программа выдает сообщение в какой промежуток попала цифра.

[**+**] Введите число: 42
[**+**] Цифра в диапазоне: 41 - 50

Если число больше 100 или меньше 0, информировать об ошибке (должен отработать оператор else).

[**+**] Ошибка! Число меньше 0.
[**+**] Ошибка! Число больше 100.

[4] Самостоятельная работа | **E-**

Работа с условными конструкциями. Написать программу “генерация примера”. Пользователь вводит произвольное число:

[+] Введите число: 9

Отталкиваясь от числа пользователя, программа генерирует пример на умножение:

[+] Решите пример: 9 x 1

Первое число - множитель пользователя, второе число - множитель программы, которое начинается с **1**. После сгенерированного и выведенного в консоль примера, пользователю нужно решить пример.

[+] Решите пример: 9 x 1
[+] Ответ: 9

Правильный ответ пользователя, генерирует новый пример. Множитель программы увеличивается на единицу, число пользователя остается:

[+] Решите пример: 9 x 2
[+] Ответ: 18
[+] Пример решен правильно!

Конечный множитель - 9. После успешного решения примеров, программа информирует пользователя об успехе и завершается.

[+] Примеры решены, поздравляем!

Также, если пользователь введет неверный ответ, информировать об ошибке и на этом закончить выполнение программы.

[-] Ошибка, пример решен неверно!

[5] Самостоятельная работа | **E**

Тема: [**IF - ELSE**]

Работа с условными конструкциями if - else if - else. Написать калькулятор с пользовательским интерфейсом. Калькулятор должен уметь:

1) складывать | **+**
2) вычитать | **-**
3) умножать | *****
4) делить | **/**
5) деление от остатка | **%**

Пользователь должен выбрать операцию и ввести два числа, после введенных данных программа выдает результат:

[**+**] Результат: 291 + 99 = 390

Калькулятор работает как и с целыми, так и с дробными числами. Делить на 0 запрещено, программа должна информировать пользователя, если произошел данный случай.

[6] Самостоятельная работа | **E-**

Тема: [**SWITCH**]

Работа с конструкцией множественной выборки **switch**. Написать программу “месяца года”. В программе должно быть пользовательское меню с 12 пунктами (январь, февраль и т.д). Пользователь вводит номер месяца и нажимает кнопку “Enter”, программа выводит сообщение с информацией о месяце. Если пользователь вводит любое другое число, то программа должна предупредить пользователя о неверно введенном значении.

Пример:
[**+**] Месяца года
[**1**] Январь
…
[**12**] Декабрь

[**+**] Выберите месяц: 2
[**+**] Выбран месяц “Февраль”

[**+**] Выберите месяц: 15
[**+**] Не правильно введен номер месяца!

Требования к программе:

1. Все должно быть написано на конструкции switch.
2. У каждого switch есть блок default
3. Каждый день месяца реализован через отдельный кейс.

[7] Самостоятельная работа | E-

Тема: [SWITCH]

Работа с конструкцией множественной выборки **switch**. Написать программу “Календарь”. Программа должна иметь пользовательский интерфейс. При запуске программы пользователю выводится пронумерованное меню со всеми месяцами (январь, февраль и тд):

[+] Календарь
[1] Январь
[2] Февраль
…
[+] Выберите месяц: 2

Программа ожидает ввода номера месяца. После чего, просит пользователя ввести день данного месяца:

[+] Выбран месяц - “Февраль”
[+] Выберите день месяца: 21

Итог: программа выводит сообщение:

[+] Календарь: 21 февраля.

Требования к программе:

1. Все должно быть написано на конструкции switch.
2. У каждого switch есть блок default
3. Каждый день месяца реализован через отдельный кейс.

[8] Самостоятельная работа | **E**

Тема: [**SWITCH**]

Работа с конструкцией множественной выборки **switch**. Написать программу “Переводчик”. Программа должна иметь пользовательский интерфейс. Главное меню с тремя пунктами: русские, английские слова и выйти.

Пример:
[**+**] Переводчик

[**1**] Русские слова
[**2**] Английские слова

[**3**] Выйти

Если пользователь выбирает пункт с русскими словами, то программа выводит 15 русских слов, пользователь вводит номер слова из данного списка, дальше программа выводит сообщение с переводом:

[**+**] Переводчик “Список русских слов”

[**1**] Дом
[**2**] Дорога

[**3**] Выберите номер слова: 2
[**+**] Перевод: дорога **->** road

[9] Самостоятельная работа | **E**

Тема: [**WHILE**]

Задача: Разработать программу используя цикл **while**. Пользователь вводит число от 1 до 15. Программа обрабатывает значение и выводит n-ое количество раз сообщение:

[**+**] Цикл "WHILE"

[**+**] Введите число: **1**

[**+**] Цикл отработал. Круг: 1.

[**+**] Введите число: **3**

[**+**] Цикл отработал. Круг: 1.
[**+**] Цикл отработал. Круг: 2.
[**+**] Цикл отработал. Круг: 3.

После вывода, программа не завершается, а заново просит ввод числа. Выход из программы, происходит только в том случае, если пользователь введет 0. Также, если ввести отрицательное число, число больше 15 или иные данные - выводить ошибку.

[10] Самостоятельная работа | **E+**

Тема: [**IF - ELSE, WHILE, RANDOM**]

Задача: Разработать игру “Угадайка” используя цикл **while**. Программа должна генерировать три уникальных числа от 1 до 10. Задача игрока угадать три числа с пяти попыток. Игра должна информировать пользователя о результате введенного значения и количестве угаданых чисел.

[**Меню**]

Создать интерфейс меню. Меню состоит из 2х пунктов:

- Начать игру;
- Выйти.

[**Выйти**]

Пункт “Выйти” выполняет выход из приложения. Программа завершается.

[**Начать игру**]

Пункт “Начать игру” запускает процесс игры. Примерный интерфейс игры:

[**+**] Угаданных чисел: [**0** / **3**]
[**+**] Попыток: [**5**]

[**>**] Угадать число: **_**

[**Число угадано**]

[**>**] Угадать число: **6**
[**+**] Вы угадали число!

[**Число не угадано**]

[**>**] Угадать число: **3**
[**-**] Вы не угадали число!

[**Правила**]

1. Если игрок введет число меньше 0 или больше 10, проинформировать об ошибке и дать возможность заново угадать число не снимая попытку.
2. Если игрок угадывает все 3 числа, то выводить информацию о победе.
3. Если игрок проигрывает, то выводить информацию о проигрыше.
4. Каждый ввод числа, уменьшает попытку на 1.
5. Попытки и количество угаданных чисел, должны быть сверху игры и видны пользователю.

[11] Самостоятельная работа | **E+**

Тема: [**WHILE**]

Работа с циклом while. Написать программу “Геометрические фигуры”. Программа должна иметь пользовательский интерфейс. При запуске программы, пользователю выводится шапка с названием приложения, пронумерованное меню с фигурами и с вводом значения:

[**+**] Программа - “Геометрические фигуры”

[**1**] Линия

[**+**] Выберите фигуру: **1**

После выбора фигуры, программа должна очистить консоль и отобразить шапку с названием выбранной фигуры и запросить ее тип:

[**+**] Фигура: “Линия”

[**1**] Горизонтальная
[**2**] Вертикальная

[**+**] Выберите тип: **1**

Далее программа запрашивает длину и текстуру(символ) линии:

[**+**] Длина линии: **10**

[**+**] Текстура линии: **#**

Программа выводит результат относительно введенных пользователем данных:

[**+**] Результат:

# # # # # # # # # #

[12] Самостоятельная работа | **M-**

Тема: [**IF - ELSE, WHILE**]

Работа с условными конструкциями **if** / **else if** / **else** и **while**. Написать игру “Викторина” с вопросами по программированию и сферой IT.

[**Приветствие**]

Начало программы начинается с приветствия игрока.

[**Меню**]

Создать интерфейс меню. Меню состоит из 4 пунктов:

- Начать игру;
- Настройки;
- Правила;
- Выйти.

[**Выйти**]

Пункт “Выйти” выполняет выход из приложения. Программа завершается.

[**Правила**]

Пункт “Правила” выводит информацию о правилах игры. Основные и главные правила:

- Игрок получает очки за правильный ответ на вопрос;
- Игрок проходит дальше за правильный ответ;
- Игрок теряет жизнь при неправильном ответе.

Из меню правил должна быть возможность выйти в главное меню.

[**Настройки**]

Пункт “Настройки” дает возможность настроить **Викторину**. Настройки содержат несколько пунктов:

- Редактирование имя игрока;

- Редактирование вопросов в игре. Можно изменить на 8 - 10 - 12.
- Редактирование цвет интерфейса (это могут быть: вопросы, ответы, меню, цифры, что угодно. Необходимо показать работу с цветами)

Из меню настройки должна быть возможность выйти в главное меню.

[**Начало игры**]

Пункт “Начало игры” запускает процесс игры. Примерный интерфейс игры (можно проявлять креатив в интерфейсе):

[**+**] Игрок: **user** | жизни: **3** | очки: **0**
[**1**] Вопрос: Системный язык программирования?
[**A**] Python [**B**] C#
[**C**] Ruby [**D**] C++
[**+**] Выбрать ответ: ___

Если ответ правильный:

- Начислить очки;
- Вывести информацию о верном ответе;
- Сгенерировать и вывести новый вопрос;

Если ответ не правильный:

- Отнять жизнь;
- Вывести информацию о неверном ответе;
- Сгенерировать и вывести новый вопрос;

[**Потеря жизней**]

Игрок потерявший все жизни - проигрывает. Вывести соответствующую информацию.

[**Победа**]

Игроку выводится информация о завершении викторины с оставшимися жизнями и набранными баллами.

[13] Самостоятельная работа | **E+**

Тема: [**FOR**]

Продолжение программы “Геометрические фигуры”.

[**+**] Программа - “Геометрические фигуры”

[**1**] Линия
[**2**] Квадрат

[**+**] Выберите фигуру: **2**

После выбора фигуры, программа должна очистить консоль и отобразить шапку с названием выбранной фигуры и запросить ее тип:

[**+**] Фигура: “Квадрат”

[**1**] Заполненный
[**2**] Пустой

[**+**] Выберите тип: **1**

Далее программа запрашивает размер и текстуру(символ):

[**+**] Размер: **6**

[**+**] Текстура: **#**

Программа выводит результат относительно введенных пользователем данных:

[**+**] Результат:

```
# # # # # #
# # # # # #
# # # # # #
# # # # # #
# # # # # #
# # # # # #
```

[14] Самостоятельная работа | **E+**

Тема: [**FOR**]

Продолжение программы “Геометрические фигуры”.

[**+**] Программа - “Геометрические фигуры”

[**1**] Линия
[**2**] Квадрат
[**3**] Прямоугольник

[**+**] Выберите фигуру: **3**

После выбора фигуры, программа должна очистить консоль и отобразить шапку с названием выбранной фигуры и запросить ее тип:

[**+**] Фигура: “Прямоугольник”

[**1**] Заполненный
[**2**] Пустой

[**+**] Выберите тип: **2**

Далее программа запрашивает ширину, высоту и текстуру(символ):

[**+**] Ширина: **10**
[**+**] Высота: **4**
[**+**] Текстура: **#**

Программа выводит результат относительно введенных пользователем данных:

[**+**] Результат:

```
# # # # # # # # # #
# . . . . . . . . #
# . . . . . . . . #
# # # # # # # # # #
```

[15] Самостоятельная работа | **M-**

Тема: [**FOR**]

Продолжение программы “Геометрические фигуры”.

[**+**] Программа - “Геометрические фигуры”

[**1**] Линия
[**2**] Квадрат
[**3**] Прямоугольник
[**4**] Треугольник

[**+**] Выберите фигуру: **4**

После выбора фигуры, программа должна очистить консоль и отобразить шапку с названием выбранной фигуры и запросить ее тип:

[**+**] Фигура: “Треугольник”

[**1**] Заполненный
[**2**] Пустой

[**+**] Выберите тип: **1**

Далее программа запрашивает высоту и текстуру(символ):

[**+**] Размер: **7**

[**+**] Текстура: **#**

Программа выводит результат относительно введенных пользователем данных:

[**+**] Результат:

```
. . . # . . .
. . # . # . .
. # . . . # .
# # # # # # #
. . . . . . .
. . . . . . .
. . . . . . .
```

[16] Самостоятельная работа | **E+**

Тема: [**FOR**]

Продолжение программы “Геометрические фигуры”.

[**+**] Программа - “Геометрические фигуры”

[**1**] Линия
[**2**] Квадрат
[**3**] Прямоугольник
[**4**] Треугольник
[**5**] Решетка

[**+**] Выберите фигуру: **5**

После выбора фигуры, программа должна очистить консоль и отобразить шапку с названием выбранной фигуры:

[**+**] Фигура: “Решетка”

Далее программа запрашивает размер и текстуру(символ):

[**+**] Размер: **7**

[**+**] Текстура: **#**

Программа выводит результат относительно введенных пользователем данных:

[**+**] Результат:

```
.  #  .  #  .  #  .
#  #  #  #  #  #  #
.  #  .  #  .  #  .
#  #  #  #  #  #  #
.  #  .  #  .  #  .
#  #  #  #  #  #  #
.  #  .  #  .  #  .
```

[17] Самостоятельная работа | **E+**

Тема: [**FOR**]

Продолжение программы “Геометрические фигуры”.

[**+**] Программа - “Геометрические фигуры”

....
[**5**] Решетка
[**6**] Крестик

[**+**] Выберите фигуру: **6**

После выбора фигуры, программа должна очистить консоль и отобразить шапку с названием выбранной фигуры:

[**+**] Фигура: “Крестик”

Далее программа запрашивает размер и текстуру(символ):

[**+**] Размер: **7**

[**+**] Текстура: **#**

Программа выводит результат относительно введенных пользователем данных:

[**+**] Результат:

```
#  .  .  .  .  .  #
.  #  .  .  .  #  .
.  .  #  .  #  .  .
.  .  .  #  .  .  .
.  .  #  .  #  .  .
.  #  .  .  .  #  .
#  .  .  .  .  .  #
```

[18] Самостоятельная работа | **E+**

Тема: [**FOR**]

Продолжение программы “Геометрические фигуры”.

[**+**] Программа - “Геометрические фигуры”

....
[**6**] Крестик
[**7**] Плюс

[**+**] Выберите фигуру: **7**

После выбора фигуры, программа должна очистить консоль и отобразить шапку с названием выбранной фигуры:

[**+**] Фигура: “Плюс”

Далее программа запрашивает размер и текстуру(символ):

[**+**] Размер: **7**

[**+**] Текстура: **#**

Программа выводит результат относительно введенных пользователем данных:

[**+**] Результат:

```
.  .  .  #  .  .  .
.  .  .  #  .  .  .
.  .  .  #  .  .  .
#  #  #  #  #  #  #
.  .  .  #  .  .  .
.  .  .  #  .  .  .
.  .  .  #  .  .  .
```

[19] Самостоятельная работа | **M**

Тема: [**FOR**]

Продолжение программы “Геометрические фигуры”.

[**+**] Программа - “Геометрические фигуры”

....
[**7**] Плюс
[**8**] Ромб

[**+**] Выберите фигуру: **8**

После выбора фигуры, программа должна очистить консоль и отобразить шапку с названием выбранной фигуры:

[**+**] Фигура: “Ромб”

Далее программа запрашивает размер и текстуру(символ):

[**+**] Размер: **7**

[**+**] Текстура: **#**

Программа выводит результат относительно введенных пользователем данных:

[**+**] Результат:

```
. . .  #  . . .
. .  #  .  #  . .
.  #  .  .  .  #  .
#  .  .  .  .  .  #
.  #  .  .  .  #  .
. .  #  .  #  . .
. . .  #  . . .
```

[20] Самостоятельная работа | **E+**

Тема: [**FOR**]

Продолжение программы “Геометрические фигуры”.

[**+**] Программа - “Геометрические фигуры”

[**1**] Линия
....
[**8**] Ромб
[**9**] Змейка

[**+**] Выберите фигуру: **9**

После выбора фигуры, программа должна очистить консоль и отобразить шапку с названием выбранной фигуры:

[**+**] Фигура: “Змейка”

Далее программа запрашивает размер и текстуру(символ):

[**+**] Размер: **7**

[**+**] Текстура: **#**

Программа выводит результат относительно введенных пользователем данных:

[**+**] Результат:

```
#   #   #   #   #   #   #
.   .   .   .   .   .   #
#   #   #   #   #   #   #
#   .   .   .   .   .   .
#   #   #   #   #   #   #
.   .   .   .   .   .   #
#   #   #   #   #   #   #
```

[21] Самостоятельная работа | **H-**

Тема: [**FOR**]

Продолжение программы “Геометрические фигуры”.

[**+**] Программа - “Геометрические фигуры”

[**10**] Рекурсивный квадрат

[**+**] Выберите фигуру: **10**

После выбора фигуры, программа должна очистить консоль и отобразить шапку с названием выбранной фигуры:

[**+**] Фигура: “Рекурсивный квадрат”

Далее программа запрашивает размер и текстуру(символ):

[**+**] Размер: **9**

[**+**] Текстура: **#**

Программа выводит результат относительно введенных пользователем данных:

[**+**] Результат:

```
# # # # # # # # #
# . . . . . . . #
# . # # # # # . #
# . # . . . # . #
# . # . # . # . #
# . # . . . # . #
# . # # # # # . #
# . . . . . . . #
# # # # # # # # #
```

[22] Самостоятельная работа | **E**

Тема: [**ARRAY**]

Работа с одномерными массивами. Проинициализировать 8 массивов всех типов данных: **INT**, **SHORT**, **LONG**, **FLOAT**, **DOUBLE**, **CHAR**, **BOOL**, **STRING**.

Каждый массив хранит 10 значений. Пример инициализации:

```
int arrInt[10] = { 1, 20, 32, 4, 1, 2, 2, 54, 23, 0 };
```

После инициализации массива выводим значения через цикл **for**:

[**+**] Массив INT:
[**+**] arrInt [0] | значение: 1
[**+**] arrInt [1] | значение: 20
[**+**] arrInt [2] | значение: 32

Требования к программе:

1. Для каждого массива писать отдельный цикл **for** для вывода значений;
2. Массив инициализируется напрямую в программе.

[23] Самостоятельная работа | **E+**

Тема: [**ARRAY**]

Работа с одномерными массивами. Программа должна иметь пользовательский интерфейс. Пользователь инициализирует числовой массив на 5 элементов. Дальше программа выдает результат сложения, вычитания, умножения и деления в виде таблицы:

```
| Таблица: 1
-------------------------------------------------------
|   Число   |          Пример          |   Результат   |
-------------------------------------------------------
|     3     |          3 + 3           |       6       |
-------------------------------------------------------
|     3     |          3 - 3           |       0       |
-------------------------------------------------------
|     3     |          3 / 3           |       1       |
-------------------------------------------------------
|     3     |          3 * 3           |       9       |
-------------------------------------------------------
```

На каждое число создается своя таблица с номером, начиная с 1, 2 и до 5ти.
Вывод таблицы осуществляется с помощью цикла **for**. Для корректного отображения таблицы внутри цикла, следует написать условия.

Требования к программе:

1. Для каждой таблицы писать отдельный цикл **for**;
2. Массив инициализируется пользователем в начале программы.
3. Вывод всех 5-ти таблиц.

[24] Самостоятельная работа | **M**

Тема: [**ARRAY**]

Работа с одномерным массивом. Пользователь должен проинициализировать целочисленный массив на 7 элементов:

[**+**] Инициализация | ячейка 0: **8**
[**+**] Инициализация | ячейка 1: **120**

После инициализации, программа выдает меню:

[**+**] Настройки массива:

[**1**] Сортировка по возрастанию
[**2**] Сортировка по убыванию
[**3**] Перемножить массив
[**4**] Сложить массив
[**5**] Разделить массив
[**6**] Обнулить массив

[**9**] Задать новые значения массиву
[**+**] Ввод:

Пользователь выбирает действие с массивом:

1. Сортировка по возрастанию выводит значения массива от самого маленького до самого большого значения;
2. Сортировка по убыванию выводит значения массива от большого до самого маленького значения;
3. Перемножение массива. Пользователь вводит любое целочисленное значение на которое будет умножаться каждая ячейка массива;
4. Сложение массива. Пользователь вводит любое целочисленное значение, каждая ячейка массива складывается с введенным числом;
5. Деление массива. Пользователь вводит любое целочисленное значение на которое будет делиться каждая ячейка массива;
6. Обнуление массива. Все ячейки массива принимают значение 0.

7. 9 пункт меню, дает пользователю проинициализировать заново массив

Все выполненные действия выводятся в консоль.

[25] Самостоятельная работа | **E+**

Тема: [**ARRAY**]

Работа с двумерными массивами. Написать программу “Лабиринт”. Программа должна иметь пользовательский интерфейс. При запуске программы, пользователю выводится шапка с названием приложения, пронумерованное меню с размерами лабиринта:

[**+**] Программа - “Лабиринт”

[**+**] Размер карты:

[**1**] 15 x 15
[**2**] 20 x 20
[**3**] 30 x 30

[**+**] Выберите размер: **1**

После выбора размера, программа выводит лабиринт:

```
# .  # # # # # # # # # # #
# . . . . . . . . # # # # #
# # # # # # . . . # # # # #
# # # # # # . . . . . . # #
# # # # # # . . . # # # # #
# # # # # #  . . . . . . #
# # # # # # # # # # # #. . #
```

Требования к программе:

1. Вывод лабиринта через двумерный массив;
2. Двумерные массивы инициализируются вручную.
3. Значения массива целочисленные: 0 и 1 (в цикле делаем проверку, если 1, выводим - символ, 0 - пустота)

[26] Самостоятельная работа | **M+**

Тема: [**FUNCTION, ARRAY, FOR, IF - ELSE**]

Данная работа требует всех знаний пройденных тем:

1. Типы данных
2. Условные конструкции | **IF - ELSE**
3. Переменные
4. Конструкции множественной выборки | **SWITCH**
5. Циклы | **FOR, WHILE**.
6. Массивы
7. Функции | **FUNCTION**

Написать игру “Крестики нолики” с двумя режимами: одиночная и 2 игрока. Программа должна иметь понятный, удобный и красивый интерфейс с применением цветом.

[**Меню**]

Создать интерфейс меню. Меню состоит из 4х пунктов:

- Начать игру;
- Статистика;
- Настройки;
- Выйти.

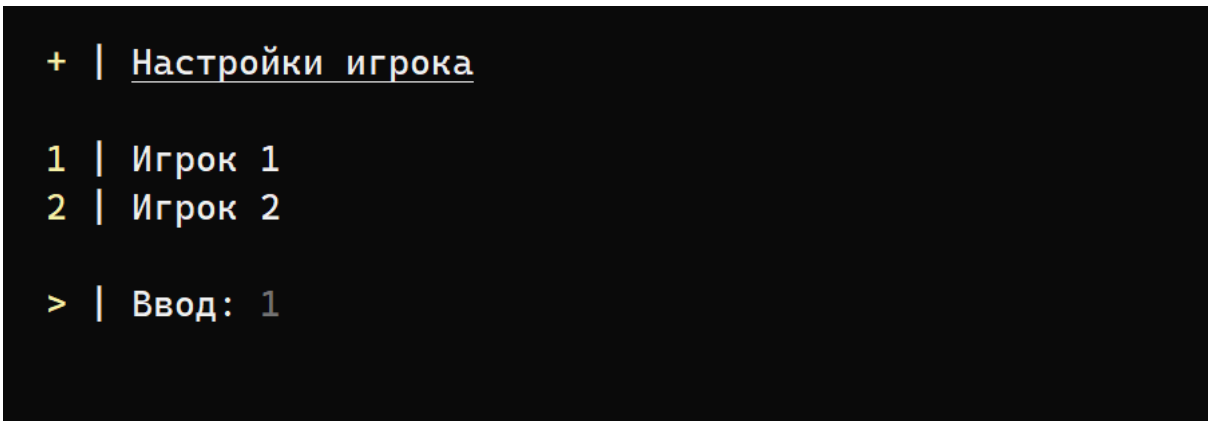
[**Выйти**]

Пункт “Выйти” выполняет выход из приложения. Программа завершается.

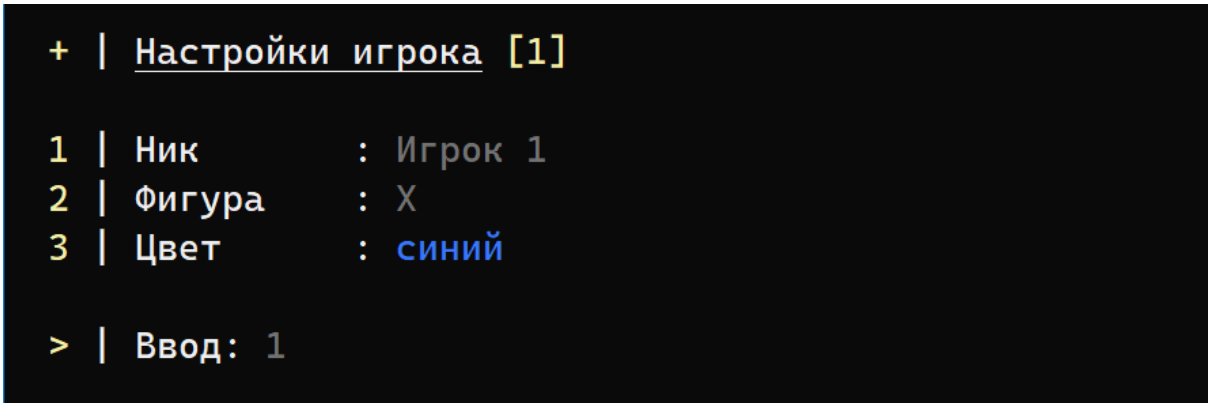
[**Настройки**]

Пункт “Настройки” дает возможность настроить игру **“Крестики нолики”**. Настройки содержат следующие пунктов:

- Настройки игрока. Возможность настроить игрока 1 и игрока 2:



При выборе игрока выводятся его настройки (ник, фигура, которой будет ходить игрок и ее цвет):



Можно настроить игрока под себя:

1. Редактирование ника.
2. Смена фигуры на другой символ (не обязательно **X** или **O).**
3. Цвет фигуры.

- Компьютер. Логика вся одинакова как с игроком, только без первого пункта - “Ник”. Компьютеру можно задать только фигуру и цвет.

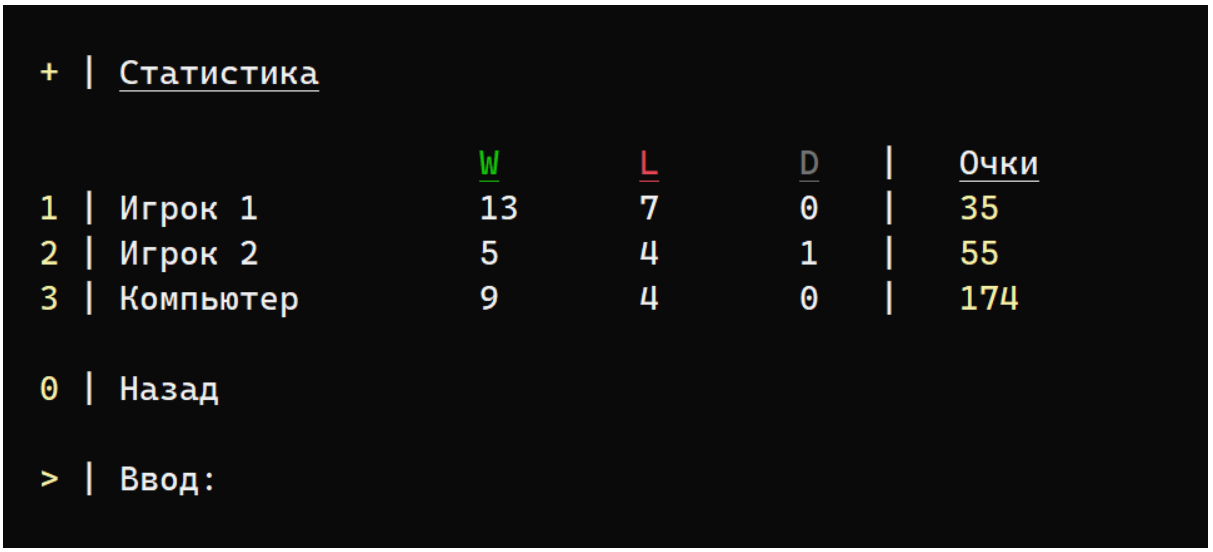
- Тема интерфейса. Придумать 3 любых цветовых тем для программы. При изменении темы, могут: меняться цвет нумерации пунктов, игрового поля и тд.

Помимо основных пунктов, везде должна быть реализована возможность возвращаться обратно, т.е пункт - “назад”.

[**Статистика**]

Данный пункт, показывает статистику игроков и компьютера. Статистика состоит из:

- Победы.
- Поражения.
- Ничьи.
- Очки.



Очки подсчитываются по формуле:

1. Поражение -6 очков.
2. Победа +5 очков.
3. Ничья -1.2 очка.

[**Начать игру**]

Пункт “Начать игру” запускает процесс игры. Примерный интерфейс игры (можно проявлять креатив в интерфейсе):

```
+ | Крестики нолики

1 | Одиночная игра
2 | Два игрока

0 | Назад

> | Ввод:
```

Пользователь выбирает режим игры. Одиночная - это игра с компьютером, “два игрока” - игра друг против друга. Создается игровое поле и информация о ходе. Игрок выбирает ячейку, куда поставит крестик:

```
+ | Крестики нолики          + | Информация

      1 | 2 | 3
      - - - - -
      4 | 5 | 6
      - - - - -
      7 | 8 | 9

> | Ходит Игрок 1 : 5
```

После хода первого игрока, ходит второй игрок:

```
+ | Крестики нолики          + | Информация

      1 | 2 | 3              + | Игрок 1 ход на клетку: 5
      - - - - -
      4 | X | 6
      - - - - -
      7 | 8 | 9

> | Ходит Игрок 2 :
```

У игры есть 3 варианта исхода: победа, поражение и ничья. Все три исхода нужно написать для двух режимов, одиночная игра и два игрока. Каждый исход, начисляет, либо убавляет очки у игроков и компьютера.

[27] Самостоятельная работа | **M**

Тема: [**ARRAY, FOR, IF - ELSE**]

Написать программу, которая будет отсортировывать двумерный массив по возрастанию и убыванию.

Размер массива меняется в коде программы. Инициализация массива происходит рандомно от 0 до 49. Каждый десяток имеет свой цвет:

1. 0 - 9 - белый;
2. 10 - 19 - зеленый;
3. 20 - 29 - синий;
4. 30 - 39 - красный;
5. 40 - 49 - желтый.

При запуске программы выводится инициализированный, несортированный массив с помощью рандома. Дальше пользователь должен выбрать вид сортировки “по убыванию” или “по возрастанию”, после выбора, программа выдает отсортированный массив.

[**+**] Сортировка:

1 3 7 2

4 5 1 8

9 4 5 1

[**1**] По убыванию
[**2**] По возрастанию
[**+**] Ввод: **2**

[**+**] Результат:

1 1 1 2

3 4 4 5

5 7 8 9

[28] Самостоятельная работа | **M+**

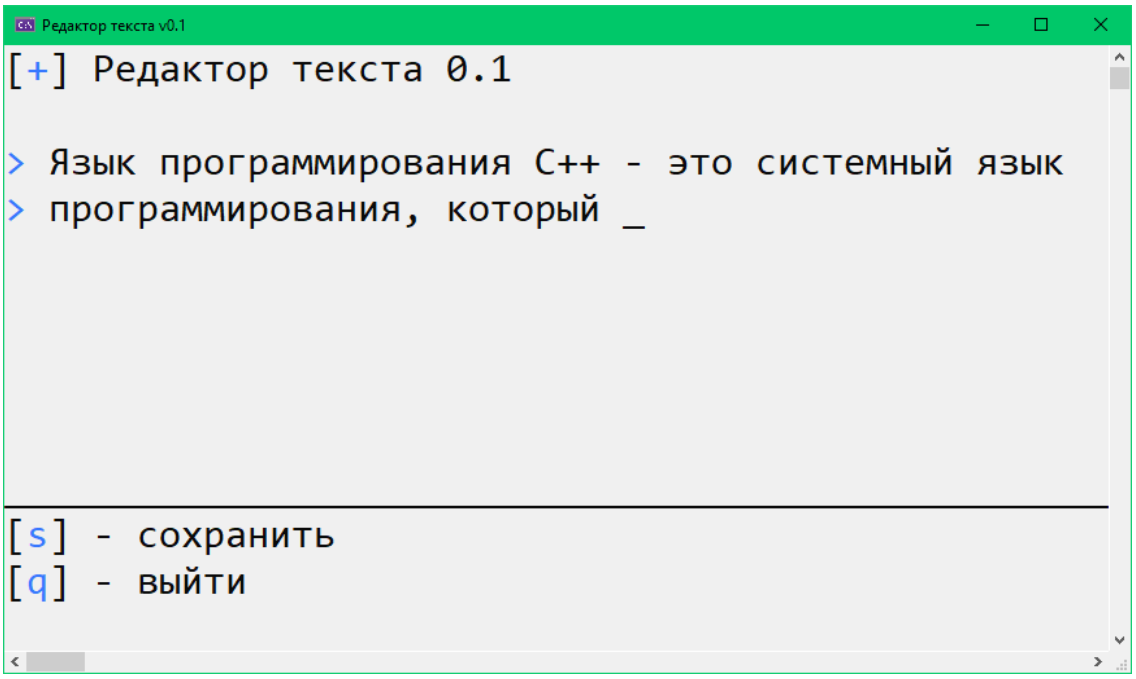
Разработать программу “Редактор текста 0.1”.

Программа состоит из 4 пунктов меню:

1. Создать файл
2. Открыть файл
3. Настройки
4. Выйти

[**Создать файл**]

Пункт “Создать файл” запускает программу:



У пользователя есть 3 функции работы с текстом:

1. **Ввод текста**
   позволяет пользователю вводить текст на русском языке.

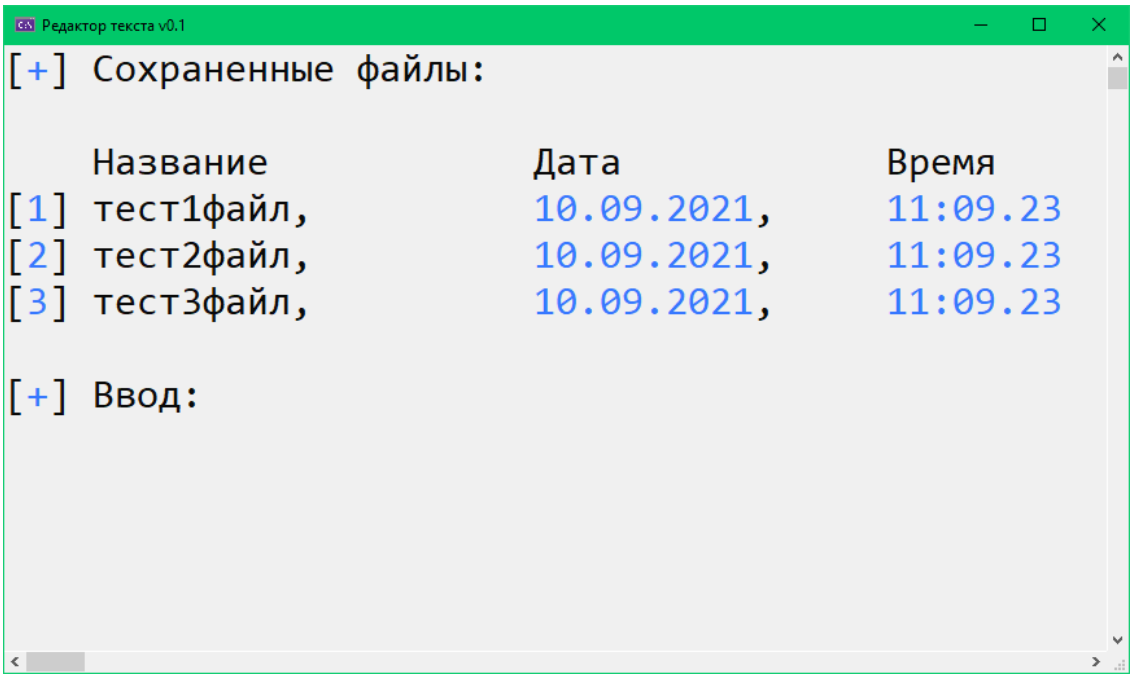
2. **Удаление текста**
   пользователь может удалять текст через клавишу backspace.

3. **Сохранение файла**
   введенный текст можно сохранить, нажав на клавишу **“s”**, также, при сохранении требовать ввести название файла. Все сохраненные файлы будут хранится в пункте “Открыть файл”.

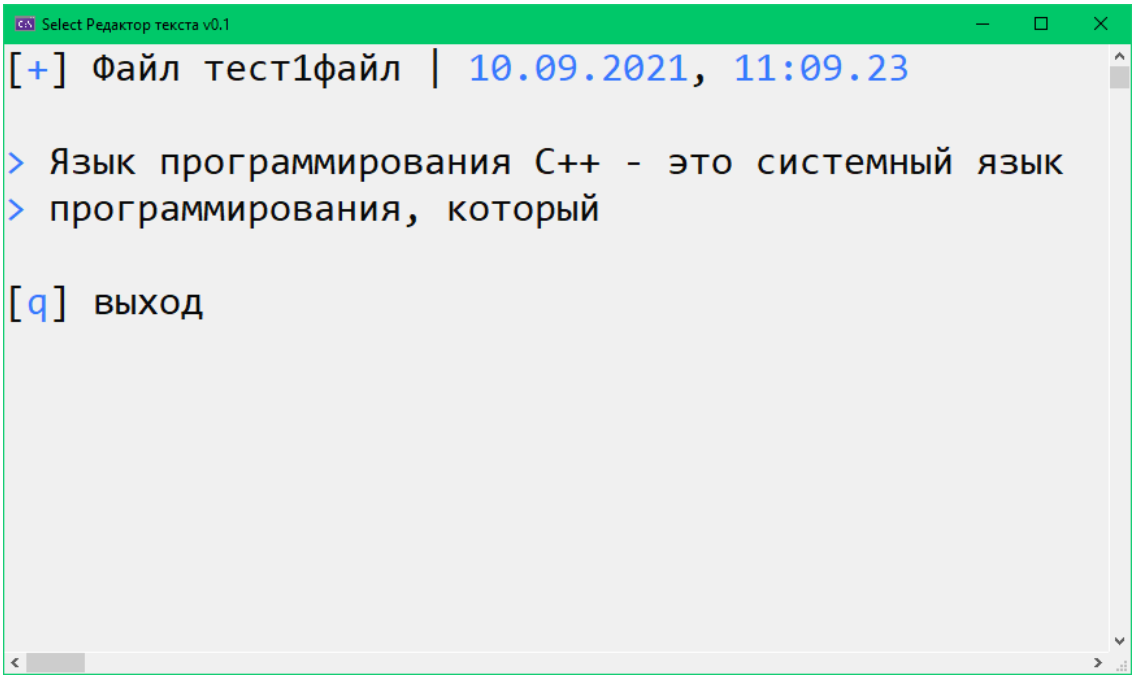
Выход в главное меню - **“q”**. Если пользователь не сохранил файл, но нажимает на выход, то программа должна спросить пользователя

[**Открыть файл**]

В данном пункте можно открыть и посмотреть сохраненные файлы:



Сохраненные файлы выводятся списком. У каждого файла есть дата и время сохранения. Пользователь может выбрать файл и посмотреть его:



[29] Самостоятельная работа | **E+**

Тема: [**FUNCTION**]

Написать калькулятор с пользовательским интерфейсом. Калькулятор должен уметь:

1. складывать | **+**
2. вычитать | **-**
3. умножать | *****
4. делить | **/**
5. деление от остатка | **%**
6. степень
7. корень
8. куб
9. синус
10. косинус

Пользователь должен выбрать операцию и ввести числа, после введенных данных программа выдает результат:

[**+**] Результат: 291 + 99 = 390

Калькулятор работает как и с целыми, так и с дробными числами. Делить на 0 запрещено, программа должна информировать пользователя, если произошел данный случай.

[30] Самостоятельная работа | **E+**

Тема: [**FUNCTION**]

Написать функцию, которая будет изменять цвет текста. Функция заменяет обычный вывод cout. Пример:

```
cout << "Привет мир!" << endl;
```

```
console("Привет мир", "red");
```

Вывод: Привет мир

Название функции вывода, может быть любой. Цвета, которые должны поддерживаться функцией: красный, зеленый, синий, голубой, желтый, фиолетовый. У функции есть 3й флаг, который добавляет подчеркивание.

```
console("Привет мир", "red", 1);
```

Вывод: Привет мир

Если в функцию передается только текст, то цвет текста стандартный и без подчеркивания.

```
console("Привет мир");
```

Вывод: Привет мир