ЛАБОРАТОРНАЯ РАБОТА 6
КАКИЕ ТЕМЫ ЗАТРОНУТЫ

- динамическая память;
- структуры.

ТРЕБОВАНИЯ К ПРОГРАММАМ

- требования по оформлению/организации кода из предыдущих лабораторных;
- добавляем *интерактивность*: от программы с фиксированными входными параметрами переходим к программе, в которой происходящим управляет тот, кто эту программу запускает.

  В данной лабораторной, часть каждого задания будет одинаковой для всех вариантов. **Общая формулировка**: заполнить массив структур, определённых заданием, и произвести над его элементами некоторые вычисления (выбор/поиск по критериям, сортировку и тому подобные).

  Работа с итоговой программой **должна** происходить следующим образом. При её запуске должен быть предложен выбор действий (некоторый аналог меню в текстовой консоли). Четыре из них — общие для всех заданий, и одно — индивидуальное. В виде примера, что может быть показано в консоли при вызове программы:

```
Element item: Game{score, attempts}
---------------
Add many items   (1)
Add one item     (2)
Print items      (3)
Remove all       (4)
Do something     (5)
Exit             (0)
---------------
Your option:
```

  Первая строка приведена просто для примера объяснения, с какимм составными объектами предстоит работа. Числами пронумерованы действия, которые должны быть реализованы:

  1. Последовательное добавление нескольких значений в массив стурктур. Сколько этих значений будет — запрашивается у пользователя.
  2. Добавление только одного объекта в массив структур.
  3. Вывод всех объектов из текущего массива структур в текстовом виде (печать значений их полей).
  4. Удаление текущего массива структур. После этого действия, массив

логически становится пустым. И, например, это отображается при выборе предыдущего действия.

5. Под «**Do something**» как раз обозначено выполнение вычислений из индивидуального задания.
6. Завершение работы программы.

Строка «**Your option:**» — это приглашение пользователю выбрать номер какого-нибудь варианта. Названия каждого действия и сопоставленные им числа — даны для примера. В именовании полная, но разумная, свобода творчества.

- Выбор и проверка опции для выполнения должны быть реализованы через **перечисления (enumerations)** и конструкцию **switch**.
- Если в задании явно не указано, тип полей выбираете самостоятельно.
- Каждое вышеперечисленное действие должно быть вынесено в отдельную функцию. За исключением завершения работы, конечно же.
- Для хранения набора структур используем *динамический массив*. При добавлении новых элементов этот массив должен быть расширен для возможности хранения требуемого количества объектов. Один из вариантов перевыделения памяти под динамический массив приводился в четвёртой лекции (где-то в районе 47 слайда). Внимательно следите за выделением/отчисткой динамической памяти.
- Не то, чтобы железное требование, но предпочтительней для хранения текстовых строк использовать *статические массивы*.
- Любой вывод элементов массива, будь то печать всех элементов или печать элементов, найденных по заданию, предполагает вывод всех полей структур. Формат вывода должен обеспечивать удобное восприятие данных. Хороший вариант – выводить в виде таблицы. Если элементов, которые должны быть выведены, нет, то программа должна выдавать соответствующее сообщение.
- Изначально заполнить массив небольшим набором входных данных, достаточным для демонстрации работы индивидуального задания, т.е. при запуске программы, какие-то данные должны уже содержаться в массиве. Этот функционал должен быть оформлен в коде так, чтобы его можно было быстро отключить, т.е. чтобы программа начинала работу с пустым массивом.

ВСПОМОГАТЕЛЬНЫЕ ФРАГМЕНТЫ

Типичным подходом к организации подобного сеанса взаимодействия с пользователем является запрос действия от него в бесконечном цикле. И прерывание работы этого цикла по проверке некоторого условия. Общая идея:

```
const int stop_code = 0;

int choose_option();

int main()
{
  while (true) {
    option = choose_option();

    // проверки выбранной опции для набора действий

    if (option == stop_code) {
      break;
    }
  }
}
```

Однако, куча конструкций **if-else** смотрятся не так упорядоченно, как использование перечислений и конструкции **switch**. В этом случае, можно составить следующий шаблон для данной лабораторной:

```
#include <cstdio>

enum class MenuOpt {
  EXIT, ADD_MANY, ADD_ONE, PRINT, REMOVE, DO
};

void print_menu();

int main()
{
  bool stop = false;
  while (!stop) {
    print_menu();

    MenuOpt opt;
    scanf("%d", &opt);

    switch (opt) {
    case MenuOpt::ADD_MANY:
        printf("<add many> action is chosen\n");
        break;
    case MenuOpt::ADD_ONE:
        printf("<add open> action is chosen\n");
        break;
    case MenuOpt::PRINT:
        printf("<print> action is chosen\n");
        break;
    case MenuOpt::REMOVE:
        printf("<remove> action is chosen\n");
        break;
    case MenuOpt::DO:
        printf("<specific> action is chosen\n");
        break;
    case MenuOpt::EXIT:
        printf("<exit> action is chosen\n");
        stop = true;
        break;
    }
```

```
  }
}

void print_menu()
{
  printf(
    "\nElement item: Game{score, attempts}\n"
    "----------------\n"
    "Add many items (%d)\n"
    "Add one item   (%d)\n"
    "Print items    (%d)\n"
    "Remove all     (%d)\n"
    "Do something   (%d)\n"
    "Exit           (%d)\n"
    "----------------\nYour option: ",
    MenuOpt::ADD_MANY, MenuOpt::ADD_ONE, MenuOpt::PRINT,
    MenuOpt::REMOVE,   MenuOpt::DO,      MenuOpt::EXIT
  );
}
```

Естественно, это не единственный возможный шаблон. Он демонстрирует базовые принципы: определение перечисления с ограниченным числом вариантов, печать понятного меню, выполнение различных действий в зависимости от выбора. Конечно, в индивидуальном задании ещё добавляется определение самой структуры, замена **printf**'ов на вызовы функций, которые будут осуществлять требуемые действия и некоторые другие действия. Но структура будет похожей на показанный пример.

ЗАДАНИЯ

Примечания.

- Не всегда одна запись, описывающая поля нужной структуры, соответствует ровно одному полю структурного типа в программе. Например, «ФИО» можно задавать как три отдельных поля. Обратить внимание на такой момент.
- В некоторых заданиях присутствуют даты и некоторые вычисления между ними (найти разницу между датами). Идеальный вариант — учитывать реальное количество дней в месяцах и существование високосных годов. В крайнем случае, вначале, в целях упрощения реализации вычислений, можно предполагать, что каждый месяц всегда состоит из 30 дней (год - 12 месяцев - 30 дней в каждом). Однако все вычисления с датами должны быть *скрыты* в функции таким образом, чтобы переход от упрощенного варианта к идеальному требовал внесения изменений только внутреннюю *реализацию* этих функций.

6.1. Структура: событие на некоторую дату.

Поля:

- день
- месяц
- год
- краткое описание

Задача: ввести два номера (обратить внимание, не индекса) элемента массива и определить, какое из событий произошло раньше.

6.2. Структура: инцидент, совершённый в определённый день.

Поля:

- сложность — целое число от 1 до 10
- день
- месяц
- год
- краткое описание

Задача: ввести год и сложность. Выбрать все инциденты за этот год, сложность которых превышала заданную.

6.3. Структура: информация о городе.

Поля:

- название
- число жителей
- страна

Задача: найти три самых населённых города в заданной стране.

6.4. Структура: информация о стране.

Поля:

- название
- часть света (Европа, Азия и т.д) — подумать, стоит ли это поле хранить как строку

Задача: задать часть света и найти все страны из неё.

6.5. Структура: информация о поезде.

Поля:

- номер
- пункт отправления
- пункт прибытия
- время в пути — с учётом задачи, продумать в какой величине лучше хранить

Задача: задать количество времени в формате **часы/минуты**. Вывести информацию о всех поездах, время в пути которых превышает заданное.

6.6. Структура: информация о youtube-канале.

Поля:

- название
- количество подписчиков
- количество видеороликов

Задача: задать пороговое количество подписчиков. Вычислить среднее количество видеороликов для каналов, количество подписчиков которых превышает заданное.

6.7. Структура: информация о сеансе в кино

Поля:

- название фильма
- день недели
- время начала сеанса: час, минуты
- длительность

Задача: задать день недели и желаемую длительность. Найти все фильмы в данный и последующие дни, длительность которых не превышает заданной.

6.8. Структура: информация о сеансе в кино

Поля:

- название фильма
- день недели
- время начала сеанса: час, минуты
- длительность

Задача: задать время (например, «20:30») и длительность. Найти все фильмы, которые идут после заданного времени и длительность которых не меньше заданной.

6.9. Структура: информация об ученике

Поля:

- имя
- отчество
- фамилия
- номер класса
- дата рождения: число, месяц, год

Задача: ввести дату и проверить, есть ли среди учащихся те, у которых день рождения выпадает на заданную дату.

6.10. Структура: информация об авиарейсе

Поля:

- номер
- время вылета
- марка самолёта
- длительность полёта
- расстояние

Задача: найти авиарейсы с максимальной и минимальной средней скоростью полёта.

6.11. Структура: информация об абоненте интернет-провайдера

Поля:

- ФИО
- номер договора
- дата подключения (число, месяц, год)
- текущий баланс

Задача: найти всех абонентов с балансом меньше нуля.

6.12. Структура: информация об автомобиле

Поля:

- производитель
- название модели
- год выпуска
- дата регистрации (число, месяц, год)

Задача: ввести конкретную дату и выбрать все автомобили, зарегистрированные не более двух лет назад от неё.

6.13. Структура: информация о численности населения

Поля:

- страна
- численность (в млн. жителей)
- площадь (в тысячах квадратных километров)

Задача: определить страны с максимальной и минимальной плотностью населения.

6.14. Структура: информация о студентах

Поля:

- ФИО
- номер группы
- общее количество экзаменов
- общая сумма оценок за экзамены (оценивание по пятибальной шкале, отсюда есть некое ограничение на максимальное значение этого поля в зависимости от предыдущего)

Задача: ввести некоторое число от 1 до 5 и выбрать всех студентов, чей средний балл превышает заданное значение.

6.15. Структура: информация о студентах

Поля:

- ФИО
- номер группы
- год поступления
- номер курса

Задача: отсортировать заданный массив по году поступления. Направление сортировки (возрастание/убывание годов) — запрашивается у пользователя. Внутри одного года упорядочение идёт по группам в алфавитном порядке.

6.16. Структура: информация об игроках CS:GO

Поля:

- никнейм игрока
- количество игр
- количество поражений
- игровой стаж (в количестве месяцев)

Задача: найти заданное пользователем количество игроков (например - три, пять, два) с максимальным процентом побед. Показать информацию о них в порядке убывания процента побед.

6.17. Структура: информация об университетском курсе

Поля:

- название дисциплины
- длительность (в семестрах, максимальное количество - два семестра)
- число заданий в каждом семестре (может быть различным)
- форма проверки

Задача: найти пять наиболее «лёгких» курсов — тех, у которых среденее число заданий в семестре меньше, чем в остальных.

6.18. Структура: информация об автомобиле

Поля:

- производитель
- название модели
- год выпуска
- стоимость (в условных единицах)

Задача: найти производителя с самой высокой средней стоимостью автомоилей.

6.19. Структура: информация о продуктовом товаре

Поля:

- название
- цена
- дата производства (число, месяц, год)
- срок годности (в неделях)

Задача: задать дату и пороговое значение срока годности. Среди введённых товаров выбрать все, срок годности которых меньше заданного числа.

6.20. Структура: информация о продуктовом товаре

Поля:

- название
- цена
- дата производства (число, месяц, год)
- срок годности (в неделях)

Задача: задать дату и найти все просроченные товары.

6.21. Структура: информация о сотрудниках

Поля:

- ФИО
- должность
- адрес проживания
- трудовой стаж (в месяцах)

Задача: по заданному пользователем набору фамилий вывести информацию обо всех сотрудниках, чьи фамилии совпадают с введённым набором. Продумать, как организовать ввод фамилий для поиска (запрашивать заранее количество или позволять вводить слова до некоторого заданного стоп-слова).

6.22. Структура: информация о сотрудниках

Поля:

- ФИО
- должность
- адрес проживания
- трудовой стаж (в месяцах)
- дата рождения

Задача: упорядочить введённый массив по должности и дате рождения.

6.23. Структура: информация о товарах

Поля:

- название
- производитель
- цена
- поступление на склад (количество месяцев, прошедших с момента поступления)
- скидка (проценты, по умолчанию — нуль)

Задача: задать количество месяцев и размер скидки. Всем товарам, которые поступили ранее, чем заданное количество месяцев, присвоить заданную величину скидки.

6.24. Структура: информация о статистике футбольной команды

Поля:

- название команды
- год выступления
- количество забитых мячей
- количество пропущенных мячей

Задача: упорядочить введённую информацию по годам и эффективности атаки. Годы выступлений идут по возрастанию, внутри каждого года упорядочение идёт по убыванию разности забитых и пропущенных мячей.

6.25. Структура: информация о футбольном игроке

Поля:

- ФИО
- название клуба
- возраст
- амплуа (нападающий, полузащитник, защитник, вратарь)
- количество игр за клуб

Задача: задать амплуа, пороговое значение возраста и количество игр. Выбрать всех игроков, амлуа которых совпадает с заданным, возраст не превышает порогового значения, а количество игр за клуб больше заданного.

6.26. Структура: информация о товарах

Поля:

- название
- производитель
- цена
- количество на складе

Задача: упорядочить введённую информацию по производителю, названию и количеству товара. Упорядочение по производителю и названию идёт в лексикографическом порядке (стандартные функции сравнения строк), по количеству — в порядке убывания.

6.27. Структура: информация об автомобиле

Поля:

- производитель
- название модели
- тип (легковой / грузовой)
- стоимость

Задача: задать тип автомобилей и вычислить средную стоимость всех авто заданного типа.

6.28. Структура: информация о приложениях

Поля:

- название приложения
- название разработчика
- рейтинг (аналог оценки в Apple/Google/Windows Store)
- количество установок

Задача: ввести название разработчика и найти среди введённой информации его приложения с максимальным и минимальным рейтингами.

6.29. Структура: информация о приложениях

Поля:

- название приложения
- название разработчика
- рейтинг (аналог оценки в Apple/Google/Windows Store)
- количество установок

Задача: упорядочить введённую информацию по разработчикам (лексикографический порядок), рейтингу (возрастание, от наименее оценённых к наиболее) и количеству установок (убывание, от наиболее загружаемых к наименее).

6.30. Структура: информация о ноутбуке

Поля:

- название модели
- производитель процессора
- производитель видеокарты
- количество оперативной памяти
- количество постоянной памяти (объём жестких дисков)
- цена
- предустановлена ли ОС (критерий да/нет)

Задача: задать диапазоны оперативной и постоянной памяти, а также нужна ли предустановленная операционная система. На основе введённых данных показать все ноутбуки, удовлетворяющие критериям.