1. Министерство образования и науки Российской Федерации
2. Санкт-Петербургский Политехнический Университет Петра Великого
3. —
4. Институт прикладной математики и механики
5. **Кафедра «Информационная безопасность компьютерных систем»**

**ЛАБОРАТОРНАЯ РАБОТА № 0**

1. «**Генерация массива структур**»
2. по дисциплине «Структуры данных»
3. Выполнил
4. студент гр. Митин Даниил \_\_\_\_\_\_\_\_\_\_\_\_

<*подпись*>

1. Преподаватель
2. \_(должность, научное звание) \_ \_\_\_\_\_\_\_\_\_\_\_\_

<*подпись*>

1. Санкт-Петербург
2. 2025

**1. Цели работы**

Цель работы — разработать программу, которая генерирует массив структур для студентов с случайными данными, такими как ФИО, дата рождения, номер группы и список предметов. Программа должна позволить пользователю:

* Сгенерировать студентов с заданным количеством дисциплин.
* Вывести размер занимаемой памяти.
* Вывести список студентов и их данные.
* Очистить сгенерированные данные.

**2. Описание задачи**

Задача заключается в разработке программы person\_gen, которая генерирует массив структур для студентов. Каждая структура должна содержать:

* **ФИО**: случайные данные с использованием списков фамилий, имен и отчеств.
* **Дата рождения**: случайная дата в диапазоне.
* **Номер учебной группы**: случайное значение.
* **Список дисциплин**: для каждой дисциплины задаются:
  + Название предмета.
  + Номера кабинетов для лекций и лабораторных.
  + Количество часов на занятиях.
  + Формат аттестации (например, экзамен, зачет и т. д.).

Пользователь должен иметь возможность выполнять следующие команды:

1. **gen N**: генерировать N студентов с параметрами, которые могут быть изменены через опции (минимальное и максимальное количество дисциплин).
2. **get\_size**: вывести размер данных, занятых сгенерированными студентами.
3. **print\_students N**: вывести N студентов или всех.
4. **clean**: очистить сгенерированные данные.

**3. Подходы к решению**

Для решения задачи был выбран следующий подход:

* **Генерация случайных данных**: Для генерации данных использовались заранее подготовленные массивы с фамилиями, именами, отчествами и предметами, которые затем случайным образом комбинировались.
* **Структуры данных**: Для хранения информации о студентах использовались структуры данных:
  + bd — структура, представляющая студента (содержит поля для ФИО, даты рождения и дисциплин).
  + subjects — структура для каждого предмета, содержащая информацию о предмете, номере кабинета, количестве часов и формате аттестации.
* **Алгоритм генерации**:
  + Для каждого студента случайным образом выбираются дисциплины (от минимального до максимального количества), для каждой дисциплины генерируются случайные данные.
  + Для работы с памятью использован единственный вызов malloc() для всего массива студентов. Это помогает избежать многократных выделений памяти для каждого студента отдельно, что бы значительно замедлило программу.

Программа должна минимизировать потребление памяти и эффективно управлять ею, чтобы избежать ошибок при выделении памяти, особенно если количество студентов или дисциплин слишком велико.

**4. Структуры данных и алгоритмы**

**Структуры данных:**

* **subjects**: Структура, представляющая один предмет:

typedef struct subjects {

unsigned char name:4;

unsigned char numcabl:4;

unsigned char numcabex:4;

unsigned char hours:4;

unsigned char attestatio:4;

unsigned char end:1;

} subjects;

В этой структуре:

* + name, numcabl, numcabex, hours, attestatio — это поля для информации о предмете, которые ограничены 4 битами.
  + end — флаг, указывающий на окончание списка предметов для конкретного студента.
* **bd**: Структура, представляющая студента:

typedef struct bd {

unsigned char f:4;

unsigned char i:4;

unsigned char o:4;

unsigned char birthdate:4;

unsigned char numgroup:4;

subjects\* items;

} bd;

В этой структуре:

* + f, i, o — это поля для фамилии, имени и отчества.
  + birthdate, numgroup — для даты рождения и номера группы.
  + items — указатель на массив дисциплин для этого студента.

**Алгоритм работы:**

1. **Генерация случайных данных**:
   * Для каждого студента создается случайный набор дисциплин (от минимального до максимального значения).
   * Используются массивы с фамилиями, именами, отчествами и предметами, которые случайным образом выбираются.
2. **Выделение памяти**:
   * Массив студентов выделяется с помощью malloc(). Массив дисциплин для каждого студента также выделяется динамически.
3. **Вывод данных**:
   * Данные о студентах выводятся с помощью функции wprintf(), которая поддерживает вывод на русском языке.
4. **Управление памятью**:
   * После выполнения команды clean вся выделенная память очищается с помощью free().

**5. Вывод**

В процессе выполнения работы было решено несколько важных задач:

* **Генерация случайных данных** для студентов и их дисциплин, что позволило эффективно сгенерировать большое количество студентов с разнообразной информацией.
* **Использование структур данных** для компактного представления информации о студентах и их дисциплинах.
* **Управление памятью** с использованием динамического выделения и освобождения памяти через malloc() и free().

**Трудности**:

* Работа с памятью и структурами данных в языке C потребовала тщательной проработки и тестирования. Например, важно было правильно выделять память для списка дисциплин и предотвращать утечки памяти.
* Генерация случайных данных для большого количества студентов потребовала внимательности при выборе диапазонов значений для каждой из переменных.

**Ошибки**:

* Проблемы с неправильным использованием функций для работы с памятью и строками в начале разработки, которые были исправлены с помощью дополнительных проверок на NULL и правильной очистки памяти.
* Ошибка при формате ввода команд с флагами, которая была решена через использование более простого формата для sscanf().

**6. Текс программы**

Текст программы предоставлен в файле main.c