**Лабораторная работа №3**

Сортировка данных о многоквартирных домах с использованием структуры-контейнера.

Гарягдыева Айджан, Б24-564

**Задача программы:**

Программа предназначена для работы с данными о многоквартирных домах. Она может выполнять следующие операции:

1. **Генерация случайных данных** о зданиях с различными характеристиками (количество этажей, квартиры, наличие лифта, тип здания и т.д.).
2. **Сортировка данных** по различным критериям (например, по году постройки или количеству квартир).
3. **Вывод данных** в консоль или в файл в формате CSV.

Это позволяет эффективно работать с большими наборами данных о зданиях, упрощая их анализ и обработку.

**Объяснение кода:**

Данный код представляет собой программу, реализующую функциональность работы с данными о многоквартирных домах. Программа поддерживает несколько режимов работы, таких как генерация данных, сортировка данных и вывод на экран. Программа реализована на языке C и использует несколько вспомогательных файлов и структур для эффективной работы с данными.

**1. main.c**

Главный файл программы, который содержит точку входа — функцию main. Эта функция анализирует командную строку, обрабатывает введенные аргументы и вызывает соответствующие функции для выполнения задач. В коде используется несколько режимов работы:

*int* main(*int* *argc*, *char*\* *argv[]*) {

    if (*argc* < 2) {

        fprintf(stderr, "Usage: %s <command> [options]\n", *argv*[0]);

        return EXIT\_FAILURE;

    }

Здесь проверяется, переданы ли аргументы командной строки. Если их нет, программа выводит сообщение об ошибке и завершает работу.

Далее, происходит инициализация контейнера Container:

 Container container;

    container\_init(&container);

container\_init — это функция, которая инициализирует пустой контейнер для хранения данных о домах.

Программа поддерживает три режима работы:

* **Генерация данных** с помощью аргумента --generate.
* **Сортировка данных** с помощью аргумента --sort.
* **Вывод данных** с помощью аргумента --print.

Пример обработки команд для генерации и вывода данных:

 if (strcmp(mode, ARG\_GENERATE) == 0) {

*size\_t* count = strtoul(*argv*[2], NULL, 10);

        generate\_data(&container, count);

 print\_data(&container, output);

        }

Здесь генерируются случайные данные с использованием функции generate\_data, а затем данные выводятся с помощью print\_data.

**2. sort.c**

В этом файле реализована логика сортировки данных с использованием пузырьковой сортировки:

*void* sort\_bubble(Container\* *container*, Comparator *cmp*) {

    if (*container*->size < 2) {

        return;

    }

    for (*size\_t* i = 0; i < *container*->size - 1; i++) {

        for (*size\_t* j = 0; j < *container*->size - i - 1; j++) {

            Building\* current = &*container*->data[j];

            Building\* next = &*container*->data[j + 1];

            if (*cmp*(current, next) > 0) {

                Building temp = \*current;

                \*current = \*next;

                \*next = temp;

            }

        }

    }

}

Здесь выполняется пузырьковая сортировка контейнера Container с использованием переданного компаратора cmp. Если текущий элемент больше следующего, то они меняются местами.

**3. building.c и building.h**

Файл building.c содержит реализацию функций для работы с данными о зданиях. Например, функция для сравнения зданий по количеству квартир:

*int* compare\_apartments\_asc(*const* *void*\* *a*, *const* *void*\* *b*) {

*const* Building\* b1 = (*const* Building\*)*a*;

*const* Building\* b2 = (*const* Building\*)*b*;

    return b1->apartments\_count - b2->apartments\_count;

}

Она используется для сортировки зданий по количеству квартир в порядке возрастания.

Также здесь определена функция для получения текстового представления типа здания:

*const* *char*\* get\_building\_type(BuildingType *type*) {

    switch (*type*) {

        case PANEL: return "Panel";

        case BRICK: return "Brick";

        case MONOLITHIC: return "Monolithic";

        default: return "Unknown";

    }

}

Это позволяет вывести строковое описание типа здания в виде "Panel", "Brick" или "Monolithic".

**4. container.c и container.h**

В этом файле реализована структура контейнера для хранения зданий. Важные функции:

* **Добавление элемента в контейнер**:

*void* container\_add(Container\* *container*, *const* Building\* *building*) {

    if (*container*->size == *container*->capacity) {

*container*->capacity = *container*->capacity == 0 ? 10 : *container*->capacity \* 2;

*container*->data = realloc(*container*->data, *container*->capacity \* sizeof(Building));

        if (!*container*->data) {

            fprintf(stderr, "Error: Unable to allocate memory\n");

            exit(EXIT\_FAILURE);

        }

    }

*container*->data[*container*->size++] = \**building*;

}

Эта функция добавляет новое здание в контейнер, увеличивая его емкость, если это необходимо.

* **Чтение данных из файла**:

*void* read\_data(Container\* *container*, *const* *char*\* *filename*) {

    FILE\* file = fopen(*filename*, "r");

    if (!file) {

        perror("Error opening file");

        exit(EXIT\_FAILURE);

    }

Функция открывает файл для чтения данных о зданиях и добавляет их в контейнер.

**5. io.c и io.h**

Эти файлы содержат функции для работы с файлами. Например, функция генерации случайных данных:

*void* generate\_data(Container\* *container*, *size\_t* *count*) {

    srand(time(NULL));

    for (*size\_t* i = 0; i < *count*; i++) {

        Building building = {

            .type = rand() % 3,

            .year = 1980 + rand() % 41,

            .has\_elevator = rand() % 2,

            .has\_trash\_chute = rand() % 2,

            .apartments\_count = 10 + rand() % 90,

            .floors\_count = 1 + rand() % 20,

            .avg\_apartment\_area = (rand() % 1000) / 10.0

        };

        strncpy(building.developer, developers[rand() % 5], sizeof(building.developer));

        strncpy(building.neighborhood, neighborhoods[rand() % 5], sizeof(building.neighborhood));

        container\_add(*container*, &building);

    }

}

Эта функция генерирует случайные данные о зданиях и добавляет их в контейнер.