📄 Name Source Generator Documentation

1. 🚀 Overview

This C program is designed for a Finnish name generation project, utilizing name data categorized by different decades. The program loads three distinct name lists and allows the user to select a time period based on which a complete name (first name, middle name, and last name) is randomly generated.

---

2. 🧱 Data Structures (Structs)

The program utilizes two main structures for storing and handling the names:

Structure	Description	Key Fields	Purpose
NameList	A simple list structure for name strings.	char **names (array of names), int count (number of names)	Used for storing **last names** and individual columns from CSV files.
DecadeData	The complete container for multi-column CSV data, organized by decades.	char **decades (headers), NameList *lists (array of NameList structs), int num_decades (number of columns/decades)	Used for storing **first names** and **middle names**, where each column represents one decade.

---

3. 💾 Name Loading Functions

The program includes two primary loading functions to handle different file formats:

- **load_names_multi_column(const char *filename, DecadeData *data)**

- **Usage:** Loads multi-column CSV files, such as first names and middle names.
- **Functionality:** Reads the header row first, creating a new NameList structure for each header within DecadeData.lists. It then reads line by line, adding names to the correct NameList based on the column index.

- **load_names_simple(const char *filename, NameList *list)**

- **Usage:** Loads simple text files (one name per line) or only the first column of a CSV file.
- **Functionality:** Reads the file line by line and stores each name directly into the provided NameList structure. Primarily used for loading **last names**.

---

4. ⚙️ Main Program (main) Logic

Stage	Operation	Purpose
1. Initialization	setlocale(), srand()	Sets localization and seeds the random number generator.
2. Data Loading	Calls to load_names_multi_column and load_names_simple	Loads first names (DecadeData), middle names (DecadeData), and last names (NameList) into memory.
3. Validation	if (first_names.num_decades == 0 ...)	**Crucial step:** Checks that the essential first and last name data has been loaded. If not, the program exits with an error code (return 1).
4. User Input	print_available_decades(), scanf()	Displays the available decades and prompts the user for a selection.
5. Name Generation	rand() \% count	Selects: **a)** the first name from the chosen decade, **b)** the last name from the entire last name list, and **c)** the middle name from the chosen decade with a 50% probability.
6. Memory Deallocation	free_decade_data(), free_names()	Frees all dynamically allocated memory to prevent memory leaks. **A mandatory step.**

5. Memory Management

Memory is dynamically allocated (malloc, realloc, strdup) for every loaded name and data structure. Therefore, it is essential that the corresponding free functions are called at the end of the program (or upon an error).

- void free_names(NameList *list)
- void free_decade_data(DecadeData *data)

These functions iterate through every string, free its memory (free()), free the pointer arrays, and finally reset the counters.