C Koans

Разработчики:

- Афанасьев Назар (0310, магистр)
- Бухарин Максим (2382)
- Кочуров Александр (2382)
- Муравин Егор (2382)
- Федоров Михаил (2382)
- Чепасов Дмитрий (2382)

Заказчик: Заславский Марк Маркович

Основная информация о проекте

Задача

Есть задачник для C (C Koans). Необходимо обернуть его в Docker, автоматизировать проверку / генерацию условий, а также добавить рандомизацию.

Функциональность

- Скрипт генерации условия и проверки решений
- Обертка в Docker
- Интеграция в Coderunner и Moodle
- Рандомизация условий (чтобы усложнить списывание)

Технологии

Docker, C, Python, Coderunner

Итерация 1 (результаты)

- Организована работа с репозиторием, создана Wiki-страница, проведены созвоны
- Команда получила более полное представление о работе с C Koans и Coderunner
- Было создано два прототипа по возможной генерации тестов для задач
- Были определены ключевые моменты касательно рандомизации тестов

Итерация 2 (результаты)

- Была подготовлена версия 1
- Были написан функционал по генерации условий, темплейтов Coderunner и шаблонов кода для задачки Basics (Базовые конструкции и структуры)
- Решение было обёрнуто в Docker контейнер
- Была написана инструкция по сборке и запуску

Итерация 3 (результаты)

- Была подготовлена версия 2
- Реализована тема Pointers (Указатели)
- Реализована тема Functions (Функции)
- Реализована тема Arrays (Массивы)
- Повышена читабельность результатов вывода программы
- Темплейты задач были переведены в формат toml

Итерация 4 (результаты)

- Была подготовлена версия 3
- Реализована тема Strings (Строки)
- Добавлена русификация
- Добавлена возможность вывода всех загруженных задач
- Проведено ручное тестирование всего функционала

Итоговое состояние проекта

- Проект хранится в репозитории GitHub, запускается в Docker контейнере
- Проект реализует поставленные сценарии
- Реализовано 5 различных тем задач, для каждой – генерация условий и автоматической проверки в среде Coderunner
- Проведено ручное тестирование всего функционала, автоматическое тестирование не реализовывалось

Пример готовой задачи

Заполните пропущенные значения так, чтобы программа завершилась с кодом 0.

Ответ: (штрафной режим: 0 %)

```c
#include <stdlib.h>
#include <string.h>
#include <stdio.h>

/*
* Работа с С-строками (массивами символов):
* 1. Строка инициализируется литералом "C is the BEST!"
* 2. Изменение символов разными способами:
*    - Через индекс массива []
*    - Через арифметику указателей *
*    - Комбинированным способом
* 3. Проверка результата после каждого изменения
*/

int main() {
    // Инициализация строки (автоматически добавляется нуль-терминатор)
    char string[] = "C is the BEST!";

    // Способ 1: изменение через индекс массива
    string[5] = 'm';
    if (strcmp("C is mhe BEST!", string) != 0)
        return 1;  // Ошибка после первого изменения

    // Способ 2: изменение через арифметику указателей
    *(string + 3) = 'O';
    if (strcmp("", string) != 0)
```

Проверить

	Тест	Ожидаемый	Получено	
✕	#1	Ok	Программа завершилась с кодом 2, а ожидался код 0. Неверное состояние строки после второго изменения (арифметика указателей)	✕

Ваш код должен пройти все тесты, чтобы заработать какие-либо оценки. Пробуйте снова.

Показать различия

Неверно

Баллы за эту попытку: 0,00/1,00.

Пример работы программы (фрагмент)

```
PS D:\Dev\lab\УРППО\еьз\mse1h2025-koans> docker run --rm koans_generator --method code_tmp cond_task tmp_coderunner --name strings_assignment_task
Код: strings_assignment_task:
#include <stdlib.h>
#include <string.h>
#include <stdio.h>

/*
* Работа с С-строками (массивами символов):
* 1. Строка инициализируется литералом "I love C so much!"
* 2. Изменение символов разными способами:
*    - Через индекс массива []
*    - Через арифметику указателей *
*    - Комбинированным способом
* 3. Проверка результата после каждого изменения
*/

int main() {
    // Инициализация строки (автоматически добавляется нуль-терминатор)
    char string[] = "I love C so much!";

    // Способ 1: изменение через индекс массива
    string[5] = 'N';
    if (strcmp({[1]}, string) != 0)
        return 1;  // Ошибка после первого изменения

    // Способ 2: изменение через арифметику указателей
    *(string + 0) = 'c';
    if (strcmp({[2]}, string) != 0)
        return 2;  // Ошибка после второго изменения

    // Способ 3: комбинированный доступ
    (string + 3)[0] = 'q';
    if (strcmp({[3]}, string) != 0)
        return 3;  // Ошибка после третьего изменения

    return 0;
}

Условие: strings_assignment_task:
Заполните пропущенные значения так, чтобы программа завершилась с кодом 0.
Coderunner: strings_assignment_task:
import subprocess, sys, re


def find_substrings(text, substrings):
    pattern = re.compile('|'.join(map(re.escape, substrings)))
    return pattern.findall(text)


def check_student_answers_for_substrings(student_answers,substrings):
    for answer in student_answers:
        found_substring = find_substrings(answer, substrings)
        if found_substring:
            print(f"Your answer has {set(found_substring)} in this answer", answer)
            sys.exit()


def replacer(match):
    key = match.group(1)
    return template.get(key, match.group(0))


student_answer = {{ STUDENT_ANSWER }}

error_messages = {'1': 'Неверное состояние строки после первого изменения (индексный доступ)', '2': 'Неверное состояние строки после второго измене
ыход за границы строки при модификации'}

banned_words = ['exit', 'return', '[', 'goto', 'malloc', 'free']
check_student_answers_for_substrings(student_answer, banned_words)
```

Вывод всех загруженных задач

```
PS D:\Dev\lab\УРППО\еьз\mse1h2025-koans> docker run --rm koans_generator --list-tasks
=== Хранилище задач ===
* Группа: Array
  - about_array_task: Знакомство с массивом
* Группа: Base
  - basic_task: Базовое задание на сравнение чисел
* Группа: Function
  - function_basics_task: Базовое знакомство с функциями
  - function_prototypes_task: Знакомство с прототипом функций
  - function_scope_and_vars_task: Знакомство с областью видимости переменных
* Группа: Pointer
  - pointers_and_addresses_task: Знакомство с указателями и адресами
  - pointers_arrays_and_arithmetic_task: Указатели, массивы и арифметика указателей
  - pointers_as_function_arguments_task: Указатели как аргументы функций
  - pointers_function_task: Указатели на функции
* Группа: String
  - strings_assignment_task: Работа со строками
  - strings_copy_task: Копирование строк
  - strings_declaration_task: Инициализация строк
  - strings_formating_task: Форматирование строк
  - strings_function_paramater_task: Строковые параметры
  - strings_reference_characters_task: Виды доступа к символам строки
  - strings_sizeof_strlen_task: Функции sizeof и strlen
  - strings_what_is_string_task: Знакомство со строковым типом данных
```