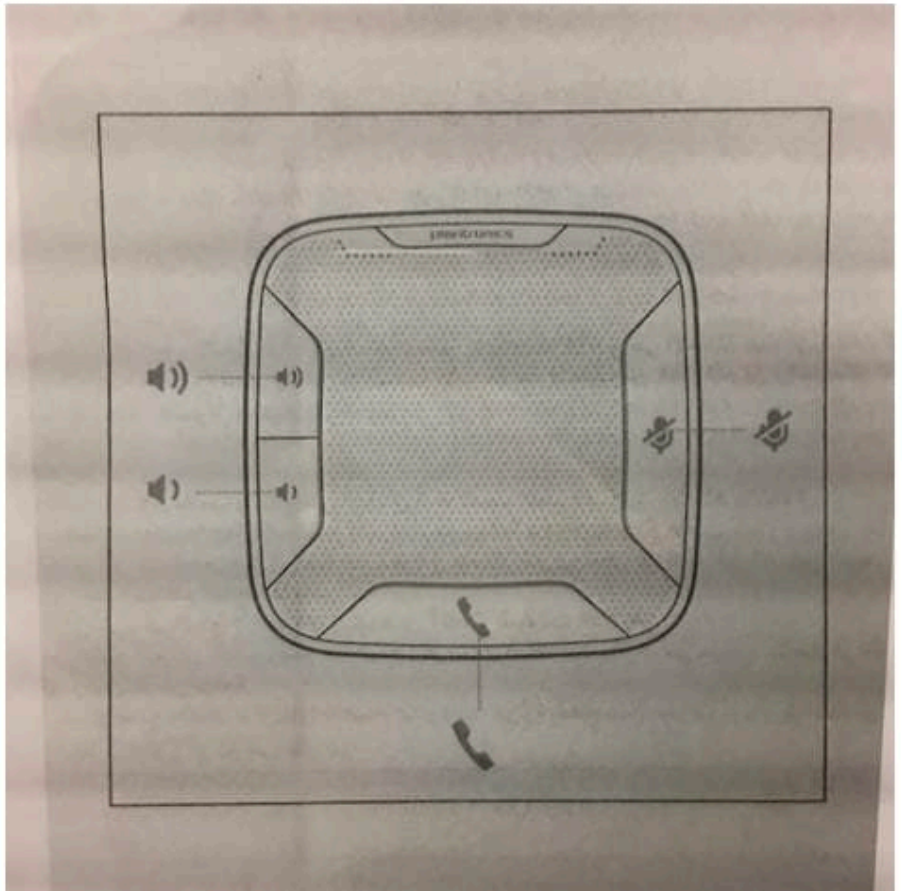
Тема 03. Вступ до документування та стилістику коду

Давидов В.В.

Як не варто документувати та коментувати свій код



Документування функції

- Що повинно бути при документуванні функції
  - Загальний опис функції (одно речення, що описує призначення функції)
  - Детальний опис функції, включая опис виняткових ситуацій, при цьому повинен бути опис «що воно робить», а не «як воно працює»
  - Опис кожного параметру (однієї фрази достатньо)
  - Опис результата (однієї фрази достатньо)
- Винятки
  - Назва функції однозначно описує її поведінку та параметри
    - float get_matrix_determinant(struct Matrix* matrix)
    - Даже в цьому випадку рекомендується зробити загальний опис для уніфікації документації

Doxygen Документація

- Doxygen – платформо-незалежна система документування вихідних текстів програм
- apt-get install doxygen graphviz mscgen
- приклад `Doxyfile`: див. sample_project
- Зміни в Makefile
  - Нова ціль: `format`
  - Команда: ` doxygen Doxyfile`
- При правильній конфігурації та відсутності помилок при складанні – результат буде в `*./dist/html/index.html*`

Doxygen. Приклад файлу

```
PROJECT_NAME = "Тестовий проект"
PROJECT_BRIEF = "Демонстрація документування коду засобами `doxygen`"
PROJECT_NUMBER = 0.1
OUTPUT_DIRECTORY = ./dist
OUTPUT_LANGUAGE = Ukrainian
DOXYFILE_ENCODING = UTF-8
INPUT_ENCODING = UTF-8
RECURSIVE = YES
FILE_PATTERNS = *.c *.h *.md
EXTRACT_ALL = YES
EXTRACT_PRIVATE = YES
EXTRACT_STATIC = YES
JAVADOC_AUTOBRIEF = YES
OPTIMIZE_OUTPUT_FOR_C = YES
HAVE_DOT = YES
DOT_PATH = /usr/local/bin/dot
UML_LOOK = YES
TEMPLATE_RELATIONS = YES
CALL_GRAPH = YES
GENERATE_HTML = YES
```

Doxygen документація

```
/**
 * Короткий коментар.
 *
 * Докладний опис
 */
```
(ідентифікатор)

```
int var; /**< Текст коментаря. */
```

```
/**
 * @file Filename.ext
 * Призначення даного файлу.
 * @author ім’я
 * @version x.y.z
 * @date yyyy.mm.dd
 */
```

```
/**
 * Отримання текстового представлення
 * значення типу тварини
 * @param type значення перерахування типу тварини
 * @return текстове репрезентування типу
 */
char *get_animal_type_name(enum animal_type type);
```

Що треба документувати:
- поточний файл (див. команду @file);
- (опціонально) Весь проект (див. команду @mainpage);
- структури;
- поля структури;
- методи (тільки при їх попередньому оголошенні);
- функцію main();
- Глобальні константи, що не належать до функції main()

Doxygen. Приклад результатів

Тестовий проект 0.1

Демонстрація документування коду засобами `doxygen`

Титульна сторінка | Додаткова інформація | Структури даних

Тестовий проект Документація

Загальне завдання

1. **Сформувати** функцию, що генерирує структуру із залученням м
2. **Сформувати** функцію, яка буде виводити масив структур на екр

Автор

Davydov V.

Дата

14-apr-2020

Версія

1.3

Переліки

enum **animal_type** {
PIG, **COW**, **DOG**, **CAT**,
ANIMAL_TYPE_COUNT
}

Тип тварини Детальніше...

Функції

char * **get_animal_type_name** (enum **animal_type** type)

Отримання текстового представлення значення типу тварини Детальніше...

void **generate_animal** (struct **animal** *entity)

Створення даних про тварину. Детальніше...

void **show_animals** (struct **animal** animals[], unsigned int count)

Вивід до екрану вмісту масиву з даними про тварини. Детальніше...

Оформлення лабораторних робіт. ДСТУ

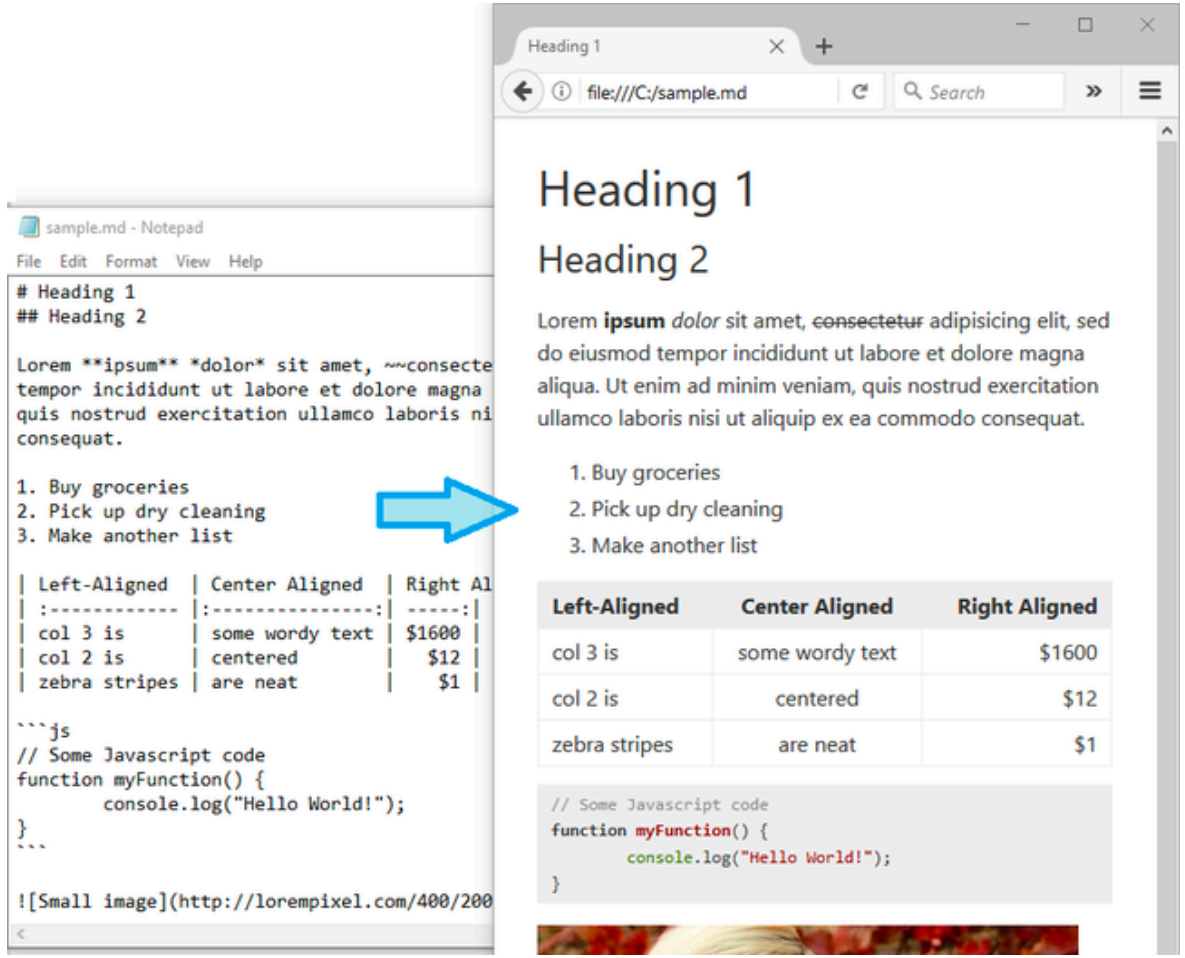
- СТВУЗ ХПІ
- Поля (верх/низ/ліве/праве), мм: 20, 15, 25, 15
- Шрифт: Times New Roman 14pt, полуторний інтервал, вирівнювання по ширині
- Нумерація сторінок: арабські цифри у верхньому правому кутку
- Розділи мають бути пронумеровані у форматі
  - 1 РОЗДІЛ
  - 2.1 **Підрозділ**
  - 2.2.1 Під-підрозділ

Оформлення лабораторних робіт. ДСТУ

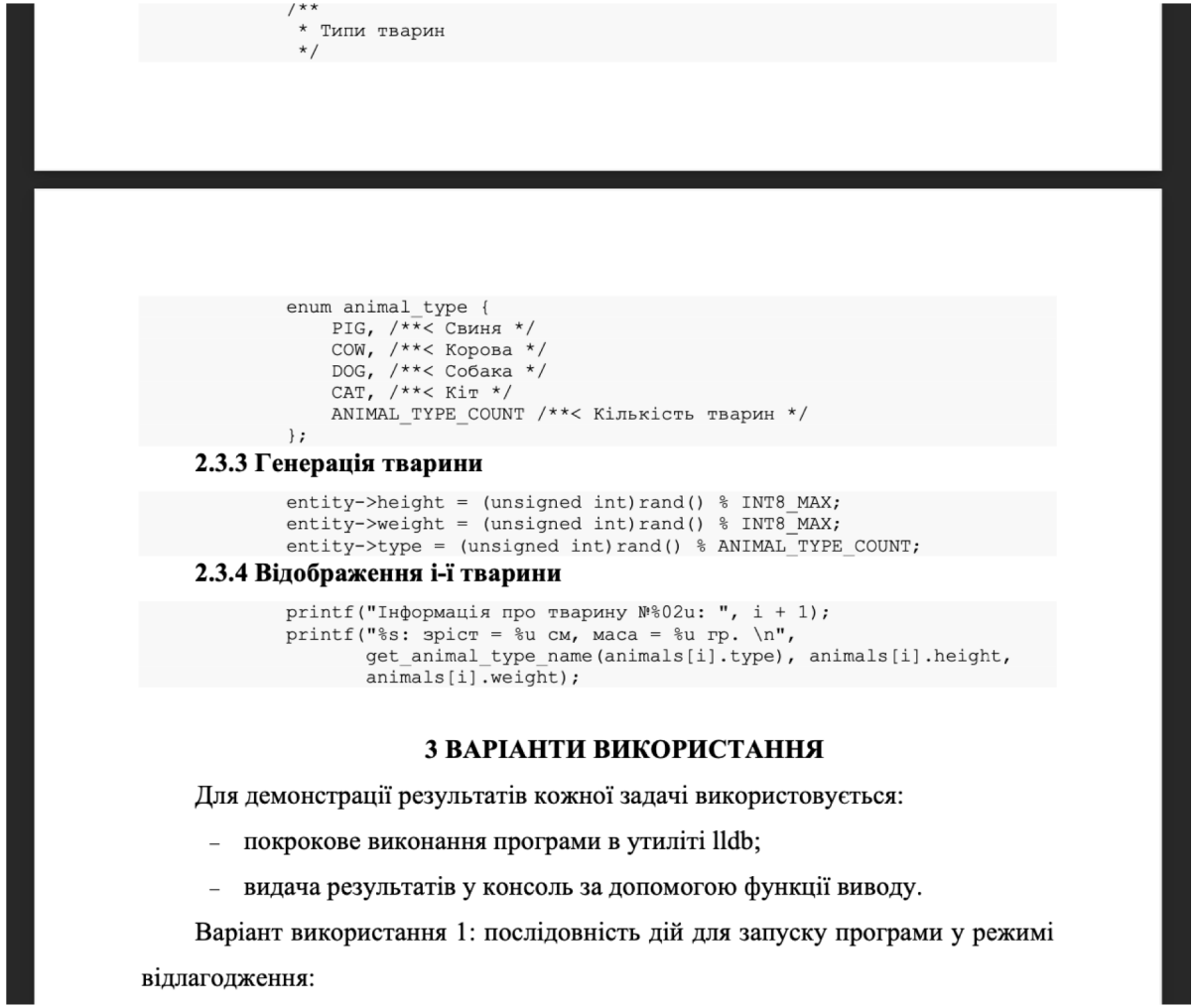
- Малюнки мають бути пронумеровані та мати посилання
  - див. рис. 1
  - Рисунок 1 – Схема алгоритму програми
- Сторінки мають бути максимально заповнені текстом (знизу). Допускається до 4 порожніх рядків внизу сторінки
- Допускається лише один вид маркированого списку (–)
- Документообіг – це не дитячий садок. Все має бути оформлено строго – жодних котиків, квіточок. Текст чорного кольору, виділення тексту (курсив, підкреслення, жирність) має бути мінімальним

Оформлення лабораторних робіт. Markdown

- Не має можливості видвинути жорсткі вимоги до стилистики
- Github embedded renderer
  - # H1 level
  - **bold text** , *italic*
  - > quote
  - 1. Ordered list
  - - Unordered list
  - `inline singleline code` , ``` multiline code block ```
  - [external link](https://www.example.com)
  - ![image link](./assets/image.jpg)



Результати друку документів. DOC



```
/**
 * Типи тварин
 */
```

```
enum animal_type {
    PIG, /**< Свиня */
    COW, /**< Корова */
    DOG, /**< Собака */
    CAT, /**< Кіт */
    ANIMAL_TYPE_COUNT /**< Кількість тварин */
};
```

2.3.3 Генерація тварини

```
entity->height = (unsigned int)rand() % INT8_MAX;
entity->weight = (unsigned int)rand() % INT8_MAX;
entity->type = (unsigned int)rand() % ANIMAL_TYPE_COUNT;
```

2.3.4 Відображення і-ї тварини

```
printf("Інформація про тварину №%02u: ", i + 1);
printf("%s: зріст = %u см, маса = %u гр. \n",
       get_animal_type_name(animals[i].type), animals[i].height,
       animals[i].weight);
```

3 ВАРІАНТИ ВИКОРИСТАННЯ

Для демонстрації результатів кожної задачі використовується:

- покрокове виконання програми в утиліті lldb;
- видача результатів у консоль за допомогою функції виводу.

Варіант використання 1: послідовність дій для запуску програми у режимі відлагодження:

Pros:

- Максимальна кастомізація деталей
- WYSIWYG

Cons:

- Якщо не вміти користуватися усіма можливостями правильно, у вас текст буде постійно з'їжджати
- Забагато можливостей, з-за чого ви акцентуєте увагу не на тексті, а на те, як він виглядає

Результати друку документів. md

Из файла берётся первое слово — это может быть битовый (CRC-1), байтовый (CRC-8) или любой другой элемент. Если старший бит в слове «1», то слово сдвигается влево на один разряд с последующим выполнением операции XOR с порождающим полиномом. Соответственно, если старший бит в слове «0», то после сдвига операция XOR не выполняется. После сдвига теряется старый старший бит, а младший бит освобождается — его значение устанавливается равным нулю. На место младшего бита загружается очередной бит из файла, и операция повторяется до тех пор, пока не загрузится последний бит файла. После прохождения всего файла, в слове остается остаток, который и является контрольной суммой.

Пример реализации CRC-8:

```
#define POLYNOMIAL 0xD8  /* 11011 followed by 0's */

uint8_t crcNaive(uint8_t const message) {
    uint8_t remainder;
    remainder = message;
    for (uint8_t bit = 8; bit > 0; --bit) {
        if (remainder & 0x80) {         // If the uppermost bit is a 1
            remainder ^= POLYNOMIAL;    // XOR the previous remainder
        }
```

Cons:

- Відсутнє вирівнювання за шириною
- Відсутня «червона строка»
- Блоки коду можуть не вміщатися за шириною сторінки
- Відсутня можливість центрувати текст
- Відсутня можливість нумерації рисунків та таблиць
- Відсутнє внутрішнє поділення на сторінки

Результати друку документів. md + Pandoc

Table 1 – Типи даних мови C (платформа x32)

Тип даних	Розмір, байт	Min значення	Max значення
char	1	-128	127
short int	2	-32768	32767
int	4	-217483648	217483647
long (long int)	4	$-2 * 10^9$	$2 * 10^9$
long long (long long int)	8	$-9 * 10^{19}$	$9 * 10^{19}$
float	4	$-1 * 10^{38}$	$1 * 10^{38}$
double	8	$-2 * 10^{308}$	$2 * 10^{308}$
long double	16	$-1 * 10^{4932}$	$1 * 10^{4932}$

Особливу увагу слід приділити речовим типам даних. Якщо для цілочисельних типів даних розмір займаної пам'яті впливає тільки на діапазон значень, яке може приймати число цього типу, то для речовинних типів даних уводиться ще один критерій – кількість цифр після коми або точність числа.

Cons:

- Налаштування «одне задоволення»
- Навіть витративши безліч часу на налаштування – зробити ідеально неможливе
- Приходиться йти на поступки (з собою та стандартом)

Оформлення лабораторних робіт. Вимоги

- Лабораторна работа №xx. Тема роботи.
- 1 ВИМОГИ
  - 1.1 Розробник
  - 1.2 Загальне завдання.
  - 1.3 Індивідуальне завдання
- 2 ОПИС ПРОГРАМИ
  - 2.1 Функціональне призначення. (Призначення програми. Обмеження на застосування)
  - 2.2 Опис логічної структури
    - Описати послідовність дій алгоритму роботи програми.
    - Описати призначення та описати структуру розроблених методів (в тому числі, їх аргументам) та структур / класів, констант та змінних. Матеріалі для даного підрозділу рекомендується брати з результатів згенерованих doxygen документацій. При цьому слід тільки змінити стилі (шрифт, розмір, відступи, колір та інше) і, при необхідності, перевести на українську мову.
    - Навести структуру програми. У даному підрозділі достатньо зробити скріншот оглядача рішень з списком файлів та методів. При консольному виконанні - достатньо виконати команду *tree* в корені проекту, але перед тим необхідно бути завіреним, що усі генеровані дані (бінарники, документація) видалені. Також, структуру програми можна отримати, зробивши скріншот сторінок *doxygen* документації, що мають перелік файлів та методів.
  - 2.3 Важливі фрагменти програми. Частини тексту програми, що демонструють рішення задачі та короткий (1-2 речення) опис. Обов'язково повинні бути вказані початкові дані.
- 3 ВАРІАНТИ ВИКОРИСТАННЯ. Опис поведінки програми ("хто" і "що" може зробити). Відповідає функціональним вимогам. Ілюструється за допомогою копій екрану з описом. У даному пункті необхідно проаналізувати отримані результати. У разі наявності виведення результатів на екран необхідно навести скріншоти консолі. Якщо робота виконувалася без виведення результатів на екран, подати скріншот вікна відлагоджувача з результатами роботи програми.
- ВИСНОВКИ