**Лабораторна робота №2.** Вступ до програмування. Основи debug процессу

**1 ВИМОГИ**

* 1. **Розробник**

-Курінна Єлизавета Ігорівна;

-студентка групи КІТ – 121г;

-15-вер-2021.

**1.2 Загальне завдання**

Зареєструватися на загально-доступній системі такій як github, навчитися копіювати файли/каталоги, користуватися командами git add, git status та відлагодником llbd, вносити зміни в код, щоб після компіляції не видавалало помилки, робити коміти змін та загружати їх у власний репозиторій на github.

**1.3 Індивідуальне завдання**

Виконати зміни, зібрати проект (зкомпілувати) та продемонструвати зміни.

1. **ОПИС РОБОТИ**

**2.1 Створення репозиторія**

Зареєструвалася на одному із загально-доступних систем такій як github та створила там приватний репозиторій(programing-kurinna) за допомогою веб інтерфейсу проекту.

**2.2 Клонування з твореного репозиторія**

Скопіювала власний і потрібний для роботи репозиторій(sample\_project) за допомогою команди git clone:

***git-clone*** [***https://github.com/davydov-vyacheslav/sample\_project***](https://github.com/davydov-vyacheslav/sample_project)

***git-clone https://github.com/elizak15/programing-kurinna***

**2.3.Копіювання файлів і каталогів**

Скопіювала наступні файли з зовнішнього репозиторія (lab00), що був наданий у попередній роботі у під-директорію lab02 за допомогою команди cp :

***cp sample\_project/.clang-format programing-kurinna/***

***cp sample\_project/.clang-tidy programing-kurinna/***

***cp sample\_project/.gitlab-ci.yml programing-kurinna/***

У ході роботи довелося створити нову директорію lab02 за домогою mkdir:

mkdir programing-kurinna/lab02

Продовжую далі копіювати:

***cp sample\_project/lab00/Doxyfile programing-kurinna/lab02***

***cp sample\_project/lab00/Makefile programing-kurinna/lab02***

***cp sample\_project/lab00/README.md programing-kurinna/lab02***

***cp sample\_project/lab00/src programing-kurinna/lab02***

ERROR cp: -r not specified; omitting directory 'sample\_project/lab00/src'

-Щоб позбутися цієї помилки я прописала cp –r, адже я хочу тепер скопіювати не один файл, а цілий каталог, включаючи всі його файли і підкаталоги. *Параметр –r означає рекурсивний*.

***cp -r sample\_project/lab00/src programing-kurinna/lab02***

***cp -r sample\_project/lab00/test programing-kurinna/lab02***

-Перевірила чи є ці файли в каталозі programing-kurinna і його підкаталозі lab02:

***ls programing-kurinna/ ls programing-kurinna/lab02/***

README.md lab02 Doxyfile Makefile README.md src test

-Виникає питання, де ж ті файли, які ми раніше копіювали. Щоб побачити їх я прописала ls –a :

***ls -a programing-kurinna/***

. *.. .clang-format .clang-tidy .git .gitlab-ci.yml* lab02 README.md

**2.4 Фіксація змін**

-Наступним кроком я зафіксувала зміни за допомогою команди git commit під назвою “Initial copy of sample\_project”.

git commit “Initial copy of sample\_project”

-Але виконати цю команду не вдалось, бо виникла помилка. Щоб все-таки виконати цю команду я *добавила файли в індекс git* за допомогою команди git add , а перед цим зайшла в каталог:

***cp programing-kurinna***

***git add .clang-format .clang-tidy***

-Далі прописала команду git status, яка відображає стан робочого каталогу і розділу проіндексованих файлів. З її допомогою можна перевірити індексацію змін і побачити файли, які не відслідковуються Git:

***git status***

On branch main

Your branch is up to date with 'origin/main'.

Changes to be committed:

(use "git restore --staged <file>..." to unstage)

new file: .clang-format

new file: .clang-tidy

Untracked files:

(use "git add <file>..." to include in what will be committed)

.gitlab-ci.yml

lab02/

-Далі меені знадобилася команда git add . , а після неї відразу написала git status, щоб провірити зміни:

***git add .***

***git status***

On branch main

Your branch is up to date with 'origin/main'.

Changes to be committed:

(use "git restore --staged <file>..." to unstage)

new file: .clang-format

new file: .clang-tidy

new file: .gitlab-ci.yml

new file: lab02/Doxyfile

new file: lab02/Makefile

new file: lab02/README.md

new file: lab02/src/lib.c

new file: lab02/src/lib.h

new file: lab02/src/main.c

new file: lab02/test/test.c

-Ще раз зафіксувала зміни під назвою “Initial copy of sample\_project”

***git commit -m "Initial copy of sample\_project"***

[main 2a3d445] Initial copy of sample\_project

10 files changed, 493 insertions(+)

create mode 100644 .clang-format

create mode 100644 .clang-tidy

create mode 100644 .gitlab-ci.yml

create mode 100644 lab02/Doxyfile

create mode 100644 lab02/Makefile

create mode 100644 lab02/README.md

create mode 100644 lab02/src/lib.c

create mode 100644 lab02/src/lib.h

create mode 100644 lab02/src/main.c

create mode 100644 lab02/test/test.c

**2.5 Зміни в коді**

-Зайшла в під-директорію lab02 і виконала команду ls, щоб провірити вміст

***cd lab02***

***ls***

Doxyfile Makefile README.md src test

Побачила те, що не вистачає каталогу dist, щоб він з’явився я виконала наступний алгоритм:

***make clean prep compile***

***ls***

Doxyfile Makefile README.md src test *dist*

-Змінила код, додавший новий тип «Людина» за допомогою команди vim:

***vim src/lib.c***

Де прописала:

case HUMAN:

result = "Людина";

break;

-Виконала компіляцію, проте стикнулась з ще одною помилкою:

***make clean prep compile***

rm -rf dist

mkdir dist

clang -std=gnu11 -g -Wall -Wextra -Werror -Wformat-security -Wfloat-equal –Wshad

ow -Wconversion -Wlogical-not-parentheses -Wnull-dereference -Wno-unused-variabl

e -Werror=vla -I./src src/lib.c src/main.c -o ./dist/main.bin

src/lib.c:19:7: error: use of undeclared identifier 'HUMAN'

case HUMAN:

^

1 error generated.

make: \*\*\* [Makefile:12: main.bin] Error 1

-Щоб уникнути цієї проблеми я в lib.h (спочатку зайшла за допомогою команди vim src/lib.h) прописала наступне:

HUMAM, /\*\*<Людина\*/

Після цього знову компіляцію:

***make clean prep compile***

rm -rf dist

mkdir dist

clang -std=gnu11 -g -Wall -Wextra -Werror -Wformat-security -Wfloat-equal –Wshad

ow -Wconversion -Wlogical-not-parentheses -Wnull-dereference -Wno-unused-variabl

e -Werror=vla -I./src src/lib.c src/main.c -o ./dist/main.bin

clang -std=gnu11 -g -Wall -Wextra -Werror -Wformat-security -Wfloat-equal –Wshad

ow -Wconversion -Wlogical-not-parentheses -Wnull-dereference -Wno-unused-variabl

e -Werror=vla -I./src src/lib.c test/test.c -o ./dist/test.bin

-Щоб подивитись зміни прописала:

*./dist/main.bin*

Інформація про тварину №01: Людина: зріст = 122 см, маса = 7 гр.

Інформація про тварину №02: Корова: зріст = 11 см, маса = 42 гр.

Інформація про тварину №03: Кіт: зріст = 77 см, маса = 89 гр.

Інформація про тварину №04: Корова: зріст = 105 см, маса = 26 гр.

Інформація про тварину №05: Свиня: зріст = 63 см, маса = 125 гр.

Інформація про тварину №06: Собака: зріст = 49 см, маса = 17 гр.

Інформація про тварину №07: Кіт: зріст = 68 см, маса = 31 гр.

Інформація про тварину №08: Корова: зріст = 54 см, маса = 70 гр.

Інформація про тварину №09: Свиня: зріст = 68 см, маса = 40 гр.

Інформація про тварину №10: Кіт: зріст = 50 см, маса = 106 гр.

**2.6 Відлагодник**

*llbd dist/main.bin*

*l*

*l*

*l*

*b 40*

*r*

*n*

Process 5477 stopped

\* thread #1, name = 'main.bin', stop reason = step over

frame #0: 0x0000000000401322 main.bin`main at main.c:43:20

40 srand((unsigned int)time(0));

41 struct animal animals[ANIMAL\_COUNT];

42

-> 43 for (unsigned int i = 0; i < ANIMAL\_COUNT; i++) {

44 generate\_animal(&animals[i]);

45 }

46 show\_animals(animals, ANIMAL\_COUNT);

40 -> 43 – об’являємо змінну

***n***

Process 5477 stopped

\* thread #1, name = 'main.bin', stop reason = step over

frame #0: 0x0000000000401339 main.bin`main at main.c:44:20

41 struct animal animals[ANIMAL\_COUNT];

42

43 for (unsigned int i = 0; i < ANIMAL\_COUNT; i++) {

-> 44 generate\_animal(&animals[i]);

45 }

46 show\_animals(animals, ANIMAL\_COUNT);

47

***s***

Process 5477 stopped

\* thread #1, name = 'main.bin', stop reason = step in

frame #0: 0x000000000040120c main.bin`generate\_animal(entity=0x00007fffffffe5f0) at lib.c:39:33

36

37 void generate\_animal(struct animal \*entity)

38 {

-> 39 entity->height = (unsigned int)random() % INT8\_MAX;

40 entity->weight = (unsigned int)random() % INT8\_MAX;

41 entity->type = (unsigned int)random() % ANIMAL\_TYPE\_COUNT;

42 }

*n*

*n*

*print \*entity*

***finish***

Process 5477 stopped

\* thread #1, name = 'main.bin', stop reason = step out

frame #0: 0x0000000000401357 main.bin`main at main.c:43:46

40 srand((unsigned int)time(0));

41 struct animal animals[ANIMAL\_COUNT];

42

-> 43 for (unsigned int i = 0; i < ANIMAL\_COUNT; i++) {

44 generate\_animal(&animals[i]);

45 }

46 show\_animals(animals, ANIMAL\_COUNT);

*l*

*b 46*

-Більше run я використати не можу, бо слугує лише для 1 запуску, тому

*continue*

*p animals*

(animal [10]) $1 = {

[0] = (type = PIG, height = 78, weight = 122)

[1] = (type = PIG, height = 31, weight = 50)

[2] = (type = CAT, height = 3, weight = 43)

[3] = (type = HUMAN, height = 23, weight = 48)

[4] = (type = PIG, height = 22, weight = 79)

[5] = (type = PIG, height = 39, weight = 104)

[6] = (type = DOG, height = 54, weight = 94)

[7] = (type = HUMAN, height = 119, weight = 116)

[8] = (type = HUMAN, height = 53, weight = 110)

[9] = (type = CAT, height = 43, weight = 27)

*n*

Інформація про тварину №01: Свиня: зріст = 78 см, маса = 122 гр.

Інформація про тварину №02: Свиня: зріст = 31 см, маса = 50 гр.

Інформація про тварину №03: Кіт: зріст = 3 см, маса = 43 гр.

Інформація про тварину №04: Людина: зріст = 23 см, маса = 48 гр.

Інформація про тварину №05: Свиня: зріст = 22 см, маса = 79 гр.

Інформація про тварину №06: Свиня: зріст = 39 см, маса = 104 гр.

Інформація про тварину №07: Собака: зріст = 54 см, маса = 94 гр.

Інформація про тварину №08: Людина: зріст = 119 см, маса = 116 гр.

Інформація про тварину №09: Людина: зріст = 53 см, маса = 110 гр.

Інформація про тварину №10: Кіт: зріст = 43 см, маса = 27 гр.

Process 5477 stopped

\* thread #1, name = 'main.bin', stop reason = step over

frame #0: 0x000000000040137b main.bin`main at main.c:48:2

45 }

46 show\_animals(animals, ANIMAL\_COUNT);

47

-> 48 return 0;

49 }

(lldb)

*q(вихід з відлагодника)*

**2.7 Додавання звітів та використання команди gitignore**

***git status***

***vim .gitignore*** (у відкритій вкладці прописую dist/)

***git add .***

***git commit “Added a new creation (Human)”***

-Додавання звіту з першої лабораторної роботи

***mkdir –p lab01/doc***

***cp lab01.docx programing-kurinna/lab01/doc***

***git status***

***git add .***

***git commit –m “Added report lab01”***

-Використання команди gitignore

***vim .gitignore*** (вирішила ще раз перевірити чи вписала каталог ***dist*)**

***git status***

***git add.***

***git commit –m “Used command gitignore”***

-Додавання звіту з другої лабораторної роботи

***mkdir –p lab02/doc***

***cp lab02.docx programing-kurinna/lab01/doc***

***git status***

***git add .***

***git commit –m “Added report lab02”***

**2.8 Завантаження на github за допомогою команди git pash**

***git pash***

**2.9 Різниця між режимом компіляції з інформацією про відлагодження та без неї**

***vim lab02/Makefile*** (видаляю –g)

-Роблю компіляцію:

***make clean prep compile***

-Щоб провірити зміни прописую:

***git diff Makefile***

diff --git a/lab02/Makefile b/lab02/Makefile

index 088755f..f568bb1 100644

--- a/lab02/Makefile

+++ b/lab02/Makefile

@@ -1,6 +1,6 @@

CC = clang

LAB\_OPTS = -I./src src/lib.c

-C\_OPTS = -std=gnu11 **-g** -Wall -Wextra -Werror -Wformat-security -Wfloat-equal -Wshadow -Wconversion -Wlogical-not-parentheses -Wnull-dereference -Wno-unused-variable -Werror=vla $(LAB\_OPTS)

+C\_OPTS = -std=gnu11 -Wall -Wextra -Werror -Wformat-security -Wfloat-equal -Wshadow -Wconversion -Wlogical-not-parentheses -Wnull-dereference -Wno-unused-variable -Werror=vla $(LAB\_OPTS)

clean:

rm -rf dist

-Щоб відкотити назад використала команду:

***git checkout – Makefile***

***Отже***, без режиму компіляції з інформацією не зможемо побачити і поставити точки зупинки.

Висновки

При виконанні даної лабораторної роботи було зареєструвано на загально-доступній системі такій як github, я навчилася копіювати файли/каталоги, користуватися командами git add, git status та відлагодником llbd, вносити зміни в код, щоб після компіляції не видавалало помилки, робити коміти змін та загружати їх у власний репозиторій на github. Також дізналась, що без режиму компіляції з інформацією не зможу побачити і поставити точки зупинки.