Лабораторна робота №2

# Тема:Основи debug процессу

Виконав:

Стоєнко Сергій Максимович

Група виконавця: КН-922Б

1.

* Що треба зробити?
  + Зареєструватися (якщо треба) на однієї з загально-доступних систем (рекомендовано), таких як *gitlab*, *github* та створити там приватний репозиторій за допомогою веб інтерфейсу проекту. Репозиторій повинен мати наступну назву: *programing-Stoienko*
* Як треба зробити?
  + Зареєструватися в github, перейти до пункту your repositories та натиснути кнопку New. Називаємо репозиторій - *programing-Stoienko,* та робимо його приватним
* Що отримали в результаті?
  + Репозиторій в github *programing-Stoienko*

2.

* Що треба зробити?
  + Склонуватися з створеного репозиторія
* Як треба зробити?
  + В терміналі ввести команду git init, а потім команду git clone https://github.com/Serhemax/programing-Stoienko.git
* Що отримали в результаті?
  + Склонований репозиторій(каталог) *programing-Stoienko*

3.

* **Що треба зробити?**
  + Створити наступні каталоги в своєму репозиторію: *lab02, lab02/src*. Скопіювати наступні елементи:
  + файл *.clang-format* з проекта *sample\_project* до кореня вашого репозиторія
  + файл *.clang-tidy* з проекта *sample\_project* до кореня вашого репозиторія
  + файл *.gitlab-ci.yml* з проекта *sample\_project* до кореня вашого репозиторія
  + файл *Doxyfile* з каталогу *lab00* проекта *sample\_project* до каталогу *lab02* вашого репозиторія
  + файл *Makefile* з каталогу *lab00* проекта *sample\_project* до каталогу *lab02* вашого репозиторія
  + файл *README.md* з каталогу *lab00* проекта *sample\_project* до каталогу *lab02* вашого репозиторія
  + каталог *src* з каталогу *lab00* проекта *sample\_project* до каталогу *lab02* вашого репозиторія
* **Як треба зробити?**
* Перейти в скопійований репозиторій і створити каталог lab02, потім перейти до створеного каталогу і там створити каталог src.
  + Потрібно перейти в каталог *sample\_project*, включити пункт “показувати приховані файли” і cкопіювати файли: *clang-format;clang-tidy;.gitlab-ci.yml* до кореня репозиторія *programing-Stoienko*.
  + Потрібно перейти в каталог *sample\_project/* *lab00*, і скопіювати файли: *Doxyfile;Makefile; README.md* в каталог *programing-Stoienko/lab02*.
  + Для того щоб скопіювати каталог *src*. Потрібно перейти в каталог *sample\_project/lab00*, і за допомогою мишки виділити каталог src і скопіювати в *programing-Stoienko/lab02*.
* **Що отримали в результаті?**

├── .clang-format

├── .clang-tidy

│   ├── heads

│   │   └── main

│   ├── remotes

│   │   └── origin

│   │   └── main

│   └── tags

├── .gitlab-ci.yml

└── lab02

├── dist

│   └── main.bin

├── Doxyfile

├── Makefile

├── README.md

└── src

├── lib.c

├── lib.h

└── main.c

Та каталог git, але там багато тек, тому я вирізав його зі звіту.

4.

* **Що треба зробити?**
  + Зафіксувати зміни (за допомогою команди *git commit*) під назвою "Initial copy of sample\_project"
* **Як треба зробити?**
* Перейти до каталогу *programing-Stoienko* та прописати наступні команди:
* git add .clang-format (додати скриті файли)
* git add .clang-tidy
* git add .gitlab-ci.yml
* git add . (додати всі не скриті файли)
* *git commit -m "Initial copy of sample\_project"* (зберегти зміни)
* **Що отримали в результаті?**

[main (root-commit) 6b26876] initial copy of sample\_project

9 files changed, 387 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

5.

* **Що треба зробити?**
* Виконати зміни, зібрати проект (зкомпілувати) та продемонструвати зміни. Додати новий тип Тварини "Людина", зробити так, щоб він був у списку елементів, що виводиться на екран.
* **Як треба зробити?**
* Перейти до каталогу *src*, відкрити файл *lib.c* та додати до команди switch:
  + case HUMAN:
    - result = “Human”;
    - break;
* Відкрити файл lib.h та дописати до animal\_type тип HUMAN
* Скомпілювати *Makefail* за допомогою команди *make all*. Після цього перейти до створеного каталогу *dist* і ввести команду *./main.bin*
* **Що отримали в результаті?**

Інформація про тварину №01: Chiken: зріст = 83 см, маса = 20 гр.

Інформація про тварину №02: Кіт: зріст = 45 см, маса = 10 гр.

Інформація про тварину №03: Кіт: зріст = 69 см, маса = 87 гр.

Інформація про тварину №04: Корова: зріст = 124 см, маса = 117 гр.

Інформація про тварину №05: Собака: зріст = 113 см, маса = 100 гр.

Інформація про тварину №06: Свиня: зріст = 67 см, маса = 97 гр.

Інформація про тварину №07: Свиня: зріст = 100 см, маса = 84 гр.

Інформація про тварину №08: Chiken: зріст = 90 см, маса = 7 гр.

Інформація про тварину №09: Корова: зріст = 118 см, маса = 107 гр.

Інформація про тварину №10: Кіт: зріст = 118 см, маса = 6 гр.

Інформація про тварину №11: Свиня: зріст = 119 см, маса = 82 гр.

Інформація про тварину №12: Собака: зріст = 28 см, маса = 0 гр.

Інформація про тварину №13: Human: зріст = 66 см, маса = 69 гр.

Інформація про тварину №14: Human: зріст = 62 см, маса = 67 гр.

Інформація про тварину №15: Кіт: зріст = 112 см, маса = 45 гр.

6.

* **Що треба зробити?**
  + Відкрити в відлагоднику (debugger) *lldb* ваш виконуючий файл, зупинитися на будь-якій строці та визначити значення змінних (variables). Слід поекспериментувати з можливостями відлагодника "Step in", "Step over", "Step out"
* **Як треба зробити?**
* За допомогою команди *sudo apt install lldb* встановити (*debugger*) *lldb*
* В терміналі ввести команду: *lldb*.
* Ввести команду щоб дебагувати main.bin: *file dist/main.bin*
* Потім потрібно встановити точку зупинки за допомогою команди: breakpoint set --name main
* Тепер потрібно запустити програму за допомогою команди: *run*
* “Step in” – команда step (thread step-in), іде по кожній строці коду(ті, які викорситовуються в основному коді, наприклад команда random)
* “Step over” – команда next(thread step-over) іде тільки по основних(ті, які вводили ми) строках
* “Step out” – команда finish(thread step-out), іде по процесах(наприклад відразу програє весь цикл)
* Щоб перевірити значення змінних треба ввести команду: frame variable
* **Що отримали в результаті?**
* (lldb) breakpoint set --name main

Breakpoint 1: where = main.bin`main + 18 at main.c:40:22, address = 0x0000000000001312

* (lldb) run

thread #1, name = 'main.bin', stop reason = breakpoint 1.1

frame #0: 0x0000555555555312 main.bin`main at main.c:40:22

37 \*/

38 int main()

39 {

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

41 struct animal animals[ANIMAL\_COUNT];

42

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

* (lldb) thread step-in

Process 5330 stopped

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

frame #0: 0x00007ffff7dca0a0 libc.so.6`\_\_srandom(x=1665773262) at random.c:209:1

* (lldb) thread step-over

Process 5330 stopped

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

frame #0: 0x00007ffff7dca0a5 libc.so.6`\_\_srandom(x=1665773262) at random.c:210:3

* (lldb) thread step-out

Process 5330 stopped

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

frame #0: 0x0000555555555322 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);

* (lldb) step

Process 5330 stopped

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

frame #0: 0x0000555555555339 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

* (lldb) next

Process 5330 stopped

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

frame #0: 0x0000555555555352 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);

* (lldb) finish

Інформація про тварину №01: Кіт: зріст = 10 см, маса = 121 гр.

Інформація про тварину №02: Кіт: зріст = 49 см, маса = 94 гр.

Інформація про тварину №03: Human: зріст = 75 см, маса = 61 гр.

Інформація про тварину №04: Chiken: зріст = 104 см, маса = 26 гр.

Інформація про тварину №05: Chiken: зріст = 36 см, маса = 101 гр.

Інформація про тварину №06: Chiken: зріст = 45 см, маса = 58 гр.

Інформація про тварину №07: Собака: зріст = 86 см, маса = 102 гр.

Інформація про тварину №08: Корова: зріст = 59 см, маса = 124 гр.

Інформація про тварину №09: Human: зріст = 50 см, маса = 33 гр.

Інформація про тварину №10: Корова: зріст = 66 см, маса = 83 гр.

Інформація про тварину №11: Human: зріст = 8 см, маса = 85 гр.

Інформація про тварину №12: Корова: зріст = 66 см, маса = 8 гр.

Інформація про тварину №13: Chiken: зріст = 97 см, маса = 83 гр.

Інформація про тварину №14: Chiken: зріст = 20 см, маса = 52 гр.

Інформація про тварину №15: Chiken: зріст = 45 см, маса = 81 гр.

Process 5330 stopped

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

Return value: (int) $0 = 0

* (lldb) frame variable

(animal[15]) animals = {

[0] = (type = HUMAN, height = 85, weight = 63)

[1] = (type = CHIKEN, height = 74, weight = 105)

[2] = (type = CHIKEN, height = 0, weight = 0)

[3] = (type = CHIKEN, height = 0, weight = 0)

[4] = (type = CHIKEN, height = 0, weight = 0)

[5] = (type = CHIKEN, height = 0, weight = 0)

[6] = (type = CHIKEN, height = 0, weight = 0)

[7] = (type = CHIKEN, height = 0, weight = 0)

[8] = (type = CHIKEN, height = 0, weight = 0)

[9] = (type = CHIKEN, height = 0, weight = 0)

[10] = (type = CHIKEN, height = 0, weight = 0)

[11] = (type = CHIKEN, height = 0, weight = 0)

[12] = (type = CHIKEN, height = 0, weight = 0)

[13] = (type = CHIKEN, height = 0, weight = 0)

[14] = (type = CHIKEN, height = 0, weight = 0)

}

(unsigned int) i = 1

7.

* **Що треба зробити?**
* Зафіксувати зміни за допомогою команди *git commit*.
* **Як треба зробити?**
* Перейти до каталогу *programing-Stoienko* та прописати наступні команди:
* git add .
* *git commit -m "Initial copy of sample\_project"*
* **Що отримали в результаті?**
* Додали всі зміни репозиторія до та зберегли їх.

*8.*

* **Що треба зробити?**
* Дослідити режими компіляції з інформацією про відлагодження та без неї; описати різницю.
* **Як треба зробити?**
* Для того щоб запустити простий процес компіляції потрібно ввести команду: make all
* Для запуску компіляції з інформацією про налагодження необхідно ввести такі команди: make --debug all
* **Що отримали в результаті?**
* При компіляції з інформацією відлагодження компілятор включає додаткову інформацію в об'єктні та файли, що виконуються. Завдяки цій інформації відладчик дізнається, які адреси відповідають тим чи іншим рядкам у тому чи іншому вихідному файлі, як відобразити значення локальної змінної тощо.

9.

* **Що треба зробити?**
* Завантажити зміни за допомогою команди *git push*
* **Як треба зробити?**
* Відкривши термінал, потрібно ввести наступні команду: *git push.* Після цього потрібно ввести своє ім'я на github та токен особистого доступу, який потрібно утворити в особистому кабінеті свого профіля в github
* **Що отримали в результаті?**
* Завантажили наш репозиторій на githab.