# Темирканов Кантемир ИУ7-23Б

# Отчет по заданию №1 в рамках вычислительного практикума

# Автоматизация функционального тестирования

1.Реализованы скрипты отладочной и релизной сборок.

|  |
| --- |
| *#!/bin/bash*  gcc -std=c99 -Wall -Werror -Wpedantic -Wextra -Wfloat-equal -Wfloat-conversion -g3 -c ./main.c  gcc -o ./main.exe ./main.c -lm |

|  |
| --- |
| *#!/bin/bash*  gcc -std=c99 -Wall -Werror -Wpedantic -Wextra -Wfloat-equal -Wfloat-conversion -c ./main.c  gcc -o ./main.exe ./main.o -lm |

2.Реализован скрипт очистки побочных файлов.

|  |
| --- |
| *#!/bin/bash*  rm ./main.exe ./main.o ./main.c.gcov ./main.gcno ./main.gcda |

3. Реализован компаратор для сравнения последовательностей действительных чисел, располагающихся в двух текстовых файлах, с игнорированием остального содержимого.

|  |
| --- |
| *#!/bin/bash*  f1=$1  f2=$2  touch f1\_num.txt f2\_num.txt  grep -oE '[+-]?[0-9]+([.][0-9]+)?' "$f1" > f1\_num.txt  grep -oE '[+-]?[0-9]+([.][0-9]+)?' "$f2" > f2\_num.txt  **if** cmp f1\_num.txt f2\_num.txt; **then**  exit 1  **else**  exit 0  **fi** |

4. Реализован компаратор для сравнения содержимого двух текстовых файлов, располагающегося после первого вхождения подстроки «Result:␣».

|  |
| --- |
| *#!/bin/bash*  f1=$1  f2=$2  touch f1\_num.txt f2\_num.txt  **if** grep -o -i 'Result: ' "$f1" | wc -l -ne 1  **then**  exit 1  **else**  grep -oE "Result: .\*" "$f1" > f1\_num.txt  grep -oE "Result: .\*" "$f2" > f2\_num.txt  **if** cmp f1\_num.txt f2\_num.txt; **then**  exit 0  **else**  exit 1  **fi**  **fi** |

5. Реализован скрипт pos\_case.sh для проверки позитивного тестового случая по определённым далее правилам. С содержанием внутри опционального запуска приложения в оболочке valgrind для проверки тестовых случаев.

|  |
| --- |
| *#!/bin/bash*  **if** [ "$1" = "USE\_VALGRIND" ]; **then** *#Проверка на то , поднят ли глобальный флаг USE\_VALGRIND,если да, то все тестовые прогоны проводятся в оболочке valgrind, иначе — как обычно.*  file\_stream\_in=$2  file\_stream\_out\_expect=$3  touch pos\_out.txt  **if** [ -n "$4" ]; **then**  file\_app\_args=$4  **else**  file\_app\_args=""  **fi**  valgrind --log-file=log.txt --quiet ./../../main.exe "$file\_app\_args" < "$file\_stream\_in" > pos\_out.txt  codv=$?  **else**  file\_stream\_in=$1  file\_stream\_out\_expect=$2  touch pos\_out.txt  **if** [ -n "$3" ]; **then**  file\_app\_args=$3  **else**  file\_app\_args=""  **fi**  ./../../main.exe "$file\_app\_args" < "$file\_stream\_in" > pos\_out.txt  codv=$?  **fi**  **if** [ "$codv" != "0" ]; **then**  exit 1  **else**  ./comparator.sh pos\_out.txt "$file\_stream\_out\_expect"  point=$?  **if** [ "$point" != "0" ]; **then**  exit 1  **else**  exit 0  **fi**  **fi** |

6. Реализован скрипт neg\_case.sh для проверки негативного тестового случая по определённым далее правилам. С содержанием внутри опционального запуска приложения в оболочке valgrind для проверки тестовых случаев.

|  |
| --- |
| *#!/bin/bash*  **if** [ "$1" = "--USE-VALGRIND" ]; **then** *#Проверка на то , поднят ли глобальный флаг USE\_VALGRIND,если да, то все тестовые прогоны проводятся в оболочке valgrind, иначе — как обычно.*  file\_stream\_in=$2  touch neg\_out.txt  **if** [ -n "$3" ]; **then**  file\_app\_args=$3  **else**  file\_app\_args=""  **fi**  valgrind --log-file=log.txt --quiet ./../../main.exe "$file\_app\_args" < "$file\_stream\_in" > neg\_out.txt  codv=$?  **else**  file\_stream\_in=$1  touch pos\_out.txt  **if** [ -n "$2" ]; **then**  file\_app\_args=$2  **else**  file\_app\_args=""  **fi**  ./../../main.exe "$file\_app\_args" < "$file\_stream\_in" > neg\_out.txt  codv=$?  **fi**  **if** [ "$codv" != "0" ]; **then**  exit 1  **else**  exit 0  **fi** |

7. Обеспечение автоматизации функционального тестирования.

|  |
| --- |
| *#!/bin/bash*  *# запись имен тестовых файлов в массивы чтобы потом бегать по массиву и извлекать данные*  arr\_pos\_in=()  **while** IFS='' read -r line; **do** arr\_pos\_in+=("$line"); **done** < <(ls ../data/pos\_[0-9][1-9]\_in.txt)  arr\_pos\_out=()  **while** IFS='' read -r line; **do** arr\_pos\_out+=("$line"); **done** < <(ls ../data/pos\_[0-9][1-9]\_out.txt)  arr\_neg\_in=()  **while** IFS='' read -r line; **do** arr\_neg\_in+=("$line"); **done** < <(ls ../data/neg\_[0-9][1-9]\_in.txt)  success\_pos\_count=0  failure\_pos\_count=0  success\_neg\_count\_memory=0  failure\_neg\_count\_memory=0  success\_pos\_count\_memory=0  failure\_pos\_count\_memory=0  count\_pos\_elem=**${#**arr\_pos\_in[@]**}**  count\_neg\_elem=**${#**arr\_neg\_in[@]**}**  success\_neg\_count=0  failure\_neg\_count=0  *# Проверка на флаг Валгринд*  **if** [ "$1" == "USE\_VALGRIND" ]; **then**  *# Проходимся по тестовым файлам и передаем их в скрипты для тестирования*  **for** (( item=0 ; item<count\_pos\_elem ; item++))  **do**  ./pos\_case.sh "${arr\_pos\_in[item]}" "${arr\_pos\_out[item]}"  codv=$?  **if** [ $codv -ne 0 ]; **then**  success\_pos\_count=**$((**success\_pos\_count + 1**))**  **else**  failure\_pos\_count=**$((**failure\_pos\_count + 1**))**  **fi**  **if** [ -s log.txt ]; **then**  failure\_pos\_count\_memory=**$((**failure\_pos\_count\_memory + 1**))**  **else**  success\_pos\_count\_memory=**$((**success\_pos\_count\_memory + 1**))**  **fi**  **done**    **for** (( item=0 ; item<count\_neg\_elem ; item++))  **do**  ./neg\_case.sh "${arr\_neg\_in[item]}"  codv=$?  **if** [ $codv -ne 0 ]; **then**  success\_neg\_count=**$((**success\_neg\_count + 1**))**  **else**  failure\_neg\_count=**$((**failure\_neg\_count + 1**))**  **fi**  **if** [ -s log.txt ]; **then**  failure\_neg\_count\_memory=**$((**failure\_neg\_count\_memory + 1**))**  **else**  success\_neg\_count\_memory=**$((**success\_neg\_count\_memory + 1**))**  **fi**  **done**  **else**  **for** (( item=0 ; item<count\_pos\_elem ; item++))  **do**  ./pos\_case.sh "${arr\_pos\_in[item]}" "${arr\_pos\_out[item]}"  codv=$?  **if** [ $codv -ne 0 ]; **then**  success\_pos\_count=**$((**success\_pos\_count + 1**))**  **else**  failure\_pos\_count=**$((**failure\_pos\_count + 1**))**  **fi**  **done**  **for** (( item=0 ; item<count\_neg\_elem ; item++))  **do**  ./neg\_case.sh "${arr\_neg\_in[item]}"  codv=$?  **if** [ $codv -ne 0 ]; **then**  success\_neg\_count=**$((**success\_neg\_count + 1**))**  **else**  failure\_neg\_count=**$((**failure\_neg\_count + 1**))**  **fi**  **done**  **fi**  *#Вывод результатов тестирующей системы*  **if** [ "$1" == "USE\_VALGRIND" ]; **then**    echo "Positive tests: correct $success\_pos\_count of $count\_pos\_elem; memory passed $success\_pos\_count\_memory of $count\_pos\_elem"  **if** [ "$count\_neg\_elem" -eq 0 ]; **then**  echo "There are no negative 'in' files"  **else**  echo "Negative tests: correct $success\_neg\_count of $count\_neg\_elem; memory passed $success\_neg\_count\_memory of $count\_neg\_elem"  **fi**  **else**  echo "Positive tests: correct $success\_pos\_count of $count\_pos\_elem"    **if** [ "$count\_neg\_elem" -eq 0 ]; **then**  echo "There are no negative 'in' files"  **else**  echo "Negative tests: correct $success\_neg\_count of $count\_neg\_elem"  **fi**  **fi** |

9. Автоматизация получения статистики полноты данных при тестировании.

|  |
| --- |
| *#!/bin/bash*  gcc -std=c99 -Wall -Werror main.c --coverage -O0 -o main.exe -lm  cd func\_tests/scripts || exit  ./func\_tests.sh > /dev/null *#вывод в корзину*  cd ../..  gcov main.exe |