ЗАДАНИЕ №1.2

Скрипты для запуска санитайзеров:

- Asan

```
#!/bin/bash

clang -std=c99 -fsanitize=address -fno-omit-frame-pointer -g -c main.c
clang -std=c99 -fsanitize=address -fno-omit-frame-pointer -g -c file_funcs.c
clang -fsanitize=address -fno-omit-frame-pointer -g -o app.exe main.o file_funcs.o
```

- Msan

```
#!/bin/bash

clang -std=c99 -fsanitize=memory -fPIE -fno-omit-frame-pointer -g -c main.c
clang -std=c99 -fsanitize=memory -fPIE -fno-omit-frame-pointer -g -c file_funcs.c
clang -fsanitize=memory -fPIE -pie -fno-omit-frame-pointer -g -o app.exe main.o
file_funcs.o
```

- Ubsan

```
#!/bin/bash

clang -std=c99 -fsanitize=undefined -fno-omit-frame-pointer -g -c main.c
clang -std=c99 -fsanitize=undefined -fno-omit-frame-pointer -g -c file_funcs.c
clang -fsanitize=undefined -fno-omit-frame-pointer -g -o app.exe main.o file_funcs.o
```

Скрипт релизной сборки:

```
#!/bin/bash

gcc -std=c99 -Wall -Werror -Wpedantic -Wextra -Wfloat-equal -Wfloat-conversion --
coverage -c main.c
```

```
gcc -std=c99 -Wall -Werror -Wpedantic -Wextra -Wfloat-equal -Wfloat-conversion --coverage -c file_funcs.c
gcc --coverage -o app.exe main.o file_funcs.o
```

Скрипт дебажной сборки:

```
#!/bin/bash

gcc -std=c99 -Wall -Werror -Wpedantic -Werror -Wfloat-equal -Wfloat-conversion -g -c main.c
gcc -std=c99 -Wall -Werror -Wpedantic -Werror -Wfloat-equal -Wfloat-conversion -g -c file_funcs.c
gcc -g -o app.exe main.o file_funcs.o
```

Скрипт очистки ненужных файлов:

```
#!/bin/bash

rm -f ./*.exe ./*.o ./*.gcda ./*.gcov ./*.c.gcov ./*.gcno func_tests/scripts/res.txt
```

Скрипт подсчета покрытия кода тестами с помощью утилиты gcov:

```
#!/bin/bash

cd func_tests/scripts || exit
./func_tests.sh
cd ../..
echo
echo "Checking coverage of tests"
gcov main.c
gcov arr_func.c
./clean.sh
```

Скрипт, который по очереди запускает программу с одним из видов сборки (с санитайзерами, дебажная и релизная) и тестирует ее:

```
#!/bin/bash

echo "BUILDING RELEASE..."
./build_release.sh
cd func_tests/scripts || exit
./func_tests.sh
cd ../..
echo
./clean.sh

echo "BUILDING DEBUG WITH ASAN..."
./build_debug_asan.sh
cd func_tests/scripts || exit
./func_tests.sh
cd ../..
echo
./clean.sh

echo "BUILDING DEBUG WITH MSAN..."
./build_debug_msan.sh
cd func_tests/scripts || exit
./func_tests.sh
cd ../..
echo
./clean.sh

echo "BUILDING DEBUG WITH UBSAN..."
./build_debug_ubsan.sh
```

```
cd func_tests/scripts || exit
./func_tests.sh
cd ../..
echo
./clean.sh

echo "BUILDING DEBUG..."
./build_debug.sh
cd func_tests/scripts || exit
./func_tests.sh
cd ../..
echo
./clean.sh
```

Скрипт-компоратор, сравнивает два файла на совпадение числовых значений в них:

```
#!/bin/bash

arr1=()
arr2=()

IFS=' '
digit="[0-9]"

while read -r -a arr; do
  for i in "${arr[@]}"; do
    if [[ $i =~ $digit ]]; then
      arr1+=("$i")
    fi
  done
```

```
done < "$1"

while read -r -a arr; do
  for i in "${arr[@]}"; do
    if [[ $i =~ $digit ]]; then
      arr2+=("$i")
    fi
  done
done <"$2"

if [[ "${arr1[*]}" == "${arr2[*]}" ]]; then
  exit 0
else
  exit 1
fi
```

Скрипт, выполняющий проверку работы программы на одном негативном тесте:

```
#!/bin/bash

input="$1"
args=$(cat "$2")
app="../../app.exe"
res="res.txt"
"$app" "./../data/$args" < "$input" > "$res"

exit $?
```

Скрипт, выполняющий проверку работы программы на одном позитивном тесте:

```
#!/bin/bash
```

```
input="$1"
output="$2"
args=$(cat "$3")
app="../../app.exe"
res="res.txt"
"$app" "./../data/$args" < "$input" > "$res"
if ./comparator.sh "$output" "$res"; then
  exit 1
else
  exit 0
fi
```

Скрипт, выполняющий функциональное тестирование:

```
#!/bin/bash

count=0

for file in ./../data/*_in.txt; do
  if [[ "$file" =~ pos_[0-9][0-9]_in\.txt ]]; then
    echo Checking "$file"...
    num=$(echo "$file" | grep -o "[0-9][0-9]")
    output=$(find ./../data/ -name "pos_$num\_out\.txt")
    args=$(find ./../data/ -name "pos_$num\_args\.txt")
    if ./pos_case.sh "$file" "$output" "$args"; then
      echo Test pos_"$num" is incorrect
      count=$((count+1))
    else
      echo All is OK
    fi
```

```
  elif [[ "$file" =~ neg_[0-9][0-9]_in\.txt ]]; then
    echo Checking "$file"...
    num=$(echo "$file" | grep -o "[0-9][0-9]")
    args=$(find ./../data/ -name "neg_$num\_args\.txt")
    if ./neg_case.sh "$file" "$args"; then
      echo Test neg_"$num" incorrect
      count=$((count+1))
    else
      echo All is OK
    fi
  fi
done

echo Fails: "$count"
exit $count
```

Также добавляется программа на языке С,которая переводит информацию из текстового файла в бинарный и обратно

```
/*
Предполагается, что во время работы программы не возникнет ошибок выполнения
*/

#include <stdio.h>
#include <string.h>


void text_to_bin(FILE *src, FILE *dst)
{
    int val;
```

```
    while (fscanf(src, "%d", &val) == 1)
        fwrite(&val, sizeof(val), 1, dst);
}


void bin_to_text(FILE *src, FILE *dst)
{
    int val;
    while (fread(&val, sizeof(val), 1, src) == 1)
        fprintf(dst, "%d\n", val);
}


int main(int argc, char **argv)
{
    FILE *src, *dst;
    if (argc == 4)
    {
        if (strcmp(argv[1], "t") == 0)
        {
            src = fopen(argv[2], "r");
            dst = fopen(argv[3], "wb");
            text_to_bin(src, dst);
            fclose(src);
            fclose(dst);
        }
        else
        {
            src = fopen(argv[2], "rb");
            dst = fopen(argv[3], "w");
```

```
            bin_to_text(src, dst);
            fclose(src);
            fclose(dst);
        }
    }

    return 0;
}
```