

ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«СИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ТЕЛЕКОММУНИКАЦИЙ И ИНФОРМАТИКИ»

Кафедра вычислительных систем

КУРСОВАЯ РАБОТА
по дисциплине «Технологии разработки программного обеспечения»
на тему «Password Generator»

Выполнил:
ст. гр. ИС-242
Стрелов К.Д.

Проверил:
ст. преподаватель Токмашева Е. И.

Новосибирск, 04.06.2023

Содержание:

Введение и постановка задачи	3
Техническое задание	4
Описание выполненного проекта.....	5
Считывания данных	6
Выборка нужных символов	6
Генерация символа для пароля.....	6
Генерация пароля	6
Вывод пароля	6
Личный вклад в проект.....	7
Приложение. Текст программы.....	10

Введение и постановка задачи

Основная наша задача была в закреплении знаний полученных на предмете ТРПО. В качестве курсовой работы мы решили работать над проектом Password Generator. Пред нами стоял такой список задач для осуществления этого проекта.

- Добавить файл с функциями (это основной файл в котором и будут генерироваться пароли)
- Добавить файл для вызова функций
- Добавить библиотеку (В ней будет храниться структура для работы функций а так же она будет связывать файлы между собой)
- Добавить Makefile (этот файл нужен для упрощения компиляции программы)
- Добавить CI (Для проверки работоспособности приложения)
- Добавить папку с файлом, в который будут записываться пароли
- Покрыть файл с функциями тестами

Техническое задание

1. Функциональность проекта: Генератор паролей. Предполагает использование в различных сферах однофакторной аутентификации, посредством генерации пароля из различных символов и регистров;
2. Приложение должно предоставлять пользователю возможность выбора длины пароля, использования заглавных и строчных букв, а также специальных символов;
3. Пользователь должен иметь возможность указать количество паролей, которые необходимо сгенерировать;
4. Генерация паролей должна происходить посредством использования функций, описанных в библиотеках;
5. Все полученные пароли должны отображаться на экране в виде текста;
6. Приложение должно проверять пользовательский ввод на корректность;
7. Приложение должно предоставлять возможность сохранения результатов в файл.

Описание выполненного проекта

Общекомандная часть с примерами работы ПО.

Функционал работы кода:

```
● xcredo@xcredo-PC:~/Documents/trpo/kursach/cw-is-242_password-generator$ make
gcc -c -Wall -Wextra -Werror -I src -MP -MMD -I thirdparty -MP -MMD src/Pwgen/gen.c -o obj/src/Pwgen/gen.o
gcc -c -Wall -Wextra -Werror -I src -MP -MMD -I thirdparty -MP -MMD src/LibPwgen/libgen.c -o obj/src/LibPwgen/libgen.o
ar rcs obj/src/LibPwgen/LibPwgen.a obj/src/LibPwgen/libgen.o
gcc -Wall -Wextra -Werror -I src -MP -MMD obj/src/Pwgen/gen.o obj/src/LibPwgen/LibPwgen.a -o bin/Pwgen -lm
● xcredo@xcredo-PC:~/Documents/trpo/kursach/cw-is-242_password-generator$ make run
./bin/Pwgen
Write the password length (digit)
16
Use capital letters?(y/n)
y
Use small letters?(y/n)
n
Use special characters?(y/n)
y
Write down how many passwords you need to generate (digit)
3
Password №1:
$7WJ1Q8W>2%E9XL?

Password №2:
3S-M4&. !A(G."BU<

Password №3:
#B)IPW=PD7WJ>QX2
```

Работа генератора паролей.

```
● xcredo@xcredo-PC:~/Documents/trpo/kursach/cw-is-242_password-generator$ make test
gcc -c -Wall -Wextra -Werror -I src -MP -MMD -I thirdparty -MP -MMD test/main.c -o obj/test/main.o
gcc -c -Wall -Wextra -Werror -I src -MP -MMD -I thirdparty -MP -MMD test/test.c -o obj/test/test.o
gcc -Wall -Wextra -Werror -I thirdparty -MP -MMD obj/test/main.o obj/test/test.o obj/src/LibPwgen/LibPwgen.a -o bin/pwgen_test -lm
● xcredo@xcredo-PC:~/Documents/trpo/kursach/cw-is-242_password-generator$ make run_test
./bin/pwgen_test
TEST 1/5 Getrand:returns_random_number_within_range [OK]
TEST 2/5 wtime:returns_current_wall_time_in_seconds [OK]
TEST 3/5 getrand_suite:test_getrand [OK]
TEST 4/5 generation_suite:test_generation File deleted successfully.
Password №1:
1Qq8Wx2E

Password №2:
9XL3SreM

Password №3:
l4tgAaGh

Password №4:
BaUuBcIi

Password №5:
PocWvPD7

[OK]
TEST 5/5 Good:modifies_array_of_good_characters_according_to_libgen_structure File deleted successfully.
Password №1:
1Q8W>2%E

Password №2:
9XL73S-M

Password №3:
4&. !A(G.

Password №4:
"BU<#B)I

Password №5:
PW=PD7WJ

[OK]
RESULTS: 5 tests (5 ok, 0 failed, 0 skipped) ran in 0.4 ms
```

Работа тестов функций.

```
Results.txt M X
Results-passwords > Results.txt
1 Password №1:
2 | 1Q8W>2%E
3
4 Password №2:
5 | 9XL?3S-M
6
7 Password №3:
8 | 4&.!A(G.
9
10 Password №4:
11 | "BU<#B)I
12
13 Password №5:
14 | PW=PD7WJ
15
```

Запись результатов работы кода в текстовый файл.

Описание наименований функций:

Считывания данных

Ui – Функция которая создает интерфейс и запрашивает у пользователя какие символы использовать в генерации пароля.

Выборка нужных символов

На основе данных введенными пользователем функция Good оставляет символы, которые будут использоваться в пароле.

Генерация символа для пароля

Getrand – функция генерирующая произвольное число в нужном нам диапазоне и передающая его в функцию Genetation

Генерация пароля

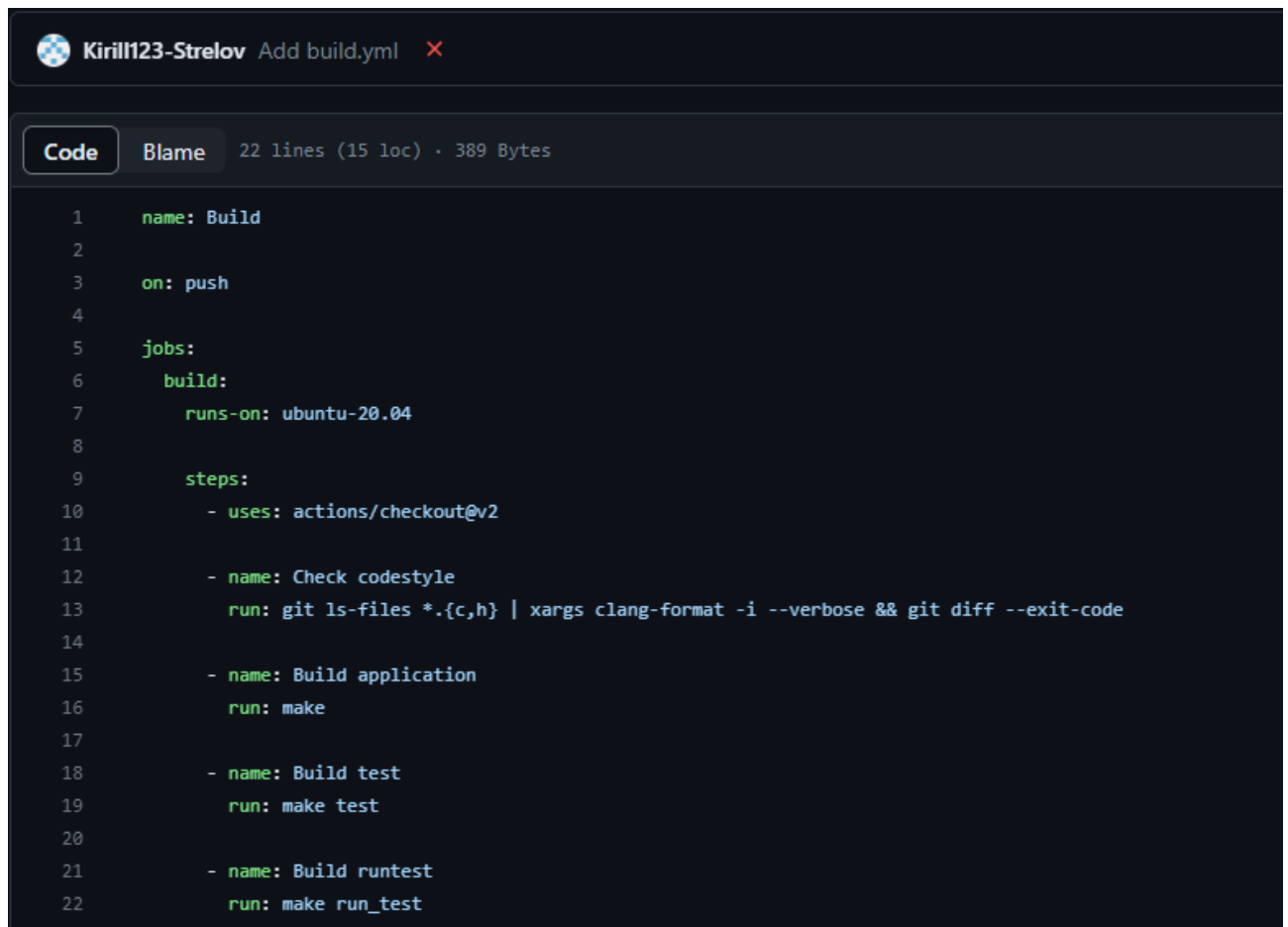
Generation –Получает число на вход, с помощью таблице АСII переводит его в символ и проверяет проходит ли оно условия введенные пользователем.

Вывод пароля

Output – функция вывода пароля в терминал а также записи его в файл Results.txt

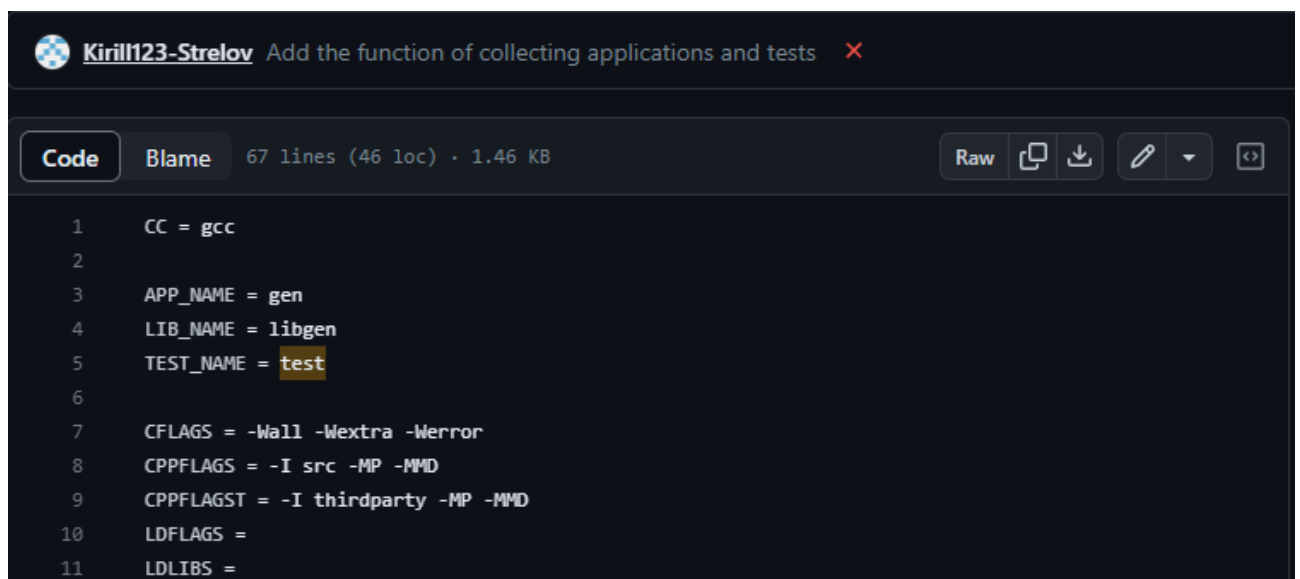
Личный вклад в проект

Написал Build:



```
1  name: Build
2
3  on: push
4
5  jobs:
6    build:
7      runs-on: ubuntu-20.04
8
9      steps:
10       - uses: actions/checkout@v2
11
12       - name: Check codestyle
13         run: git ls-files *.{c,h} | xargs clang-format -i --verbose && git diff --exit-code
14
15       - name: Build application
16         run: make
17
18       - name: Build test
19         run: make test
20
21       - name: Build runtest
22         run: make run_test
```

Написал Makefile:



```
1  CC = gcc
2
3  APP_NAME = gen
4  LIB_NAME = libgen
5  TEST_NAME = test
6
7  CFLAGS = -Wall -Wextra -Werror
8  CPPFLAGS = -I src -MP -MMD
9  CPPFLAGST = -I thirdparty -MP -MMD
10 LDFLAGS =
11 LDLIBS =
```

```

12
13 BIN_DIR = bin
14 OBJ_DIR = obj
15 SRC_DIR = src
16 TEST_DIR = test
17
18 APP_PATH = $(BIN_DIR)/$(APP_NAME)
19 TEST_PATH = $(BIN_DIR)/$(TEST_NAME)
20 LIB_PATH = $(OBJ_DIR)/$(SRC_DIR)/$(LIB_NAME)/$(LIB_NAME).a
21
22 SRC_EXT = c
23
24 APP_SOURCES = $(shell find $(SRC_DIR)/$(APP_NAME) -name '*.$(SRC_EXT)')
25 APP_OBJECTS = $(APP_SOURCES:$(SRC_DIR)/%. $(SRC_EXT)=$(OBJ_DIR)/$(SRC_DIR)/%.o)
26
27 TEST_SOURCES = $(shell find $(TEST_DIR) -name '*.$(SRC_EXT)')
28 TEST_OBJECTS = $(TEST_SOURCES:$(TEST_DIR)/%. $(SRC_EXT)=$(OBJ_DIR)/$(TEST_DIR)/%.o)
29
30 LIB_SOURCES = $(shell find $(SRC_DIR)/$(LIB_NAME) -name '*.$(SRC_EXT)')
31 LIB_OBJECTS = $(LIB_SOURCES:$(SRC_DIR)/%. $(SRC_EXT)=$(OBJ_DIR)/$(SRC_DIR)/%.o)
32
33 DEPS = $(APP_OBJECTS:.o=.d) $(LIB_OBJECTS:.o=.d) $(TEST_OBJECTS:.o=.d)
34
35 .PHONY: all
36 all: $(APP_PATH)
37
38 -include $(DEPS)
39
40 $(APP_PATH): $(APP_OBJECTS) $(LIB_PATH)
41     $(CC) $(CFLAGS) $(CPPFLAGS) $^ -o $@ -lm $(LDFLAGS) $(LDLIBS)
42
43 $(LIB_PATH): $(LIB_OBJECTS)
44     ar rcs $@ $^
45
46 $(OBJ_DIR)/%.o: %.c
47     $(CC) -c $(CFLAGS) $(CPPFLAGS) $(CPPFLAGST) $< -o $@
48
49 .PHONY: run clean test run_test
50
51 run:
52     ./bin/gen
53

```



```
54 run_test:
55     ./bin/test
56
57 clean:
58     $(RM) $(APP_PATH) $(OBJ_DIR)/**/*.[aod]
59     $(RM) bin/*.exe
60     $(RM) $(OBJ_DIR)/$(TEST_DIR)/**/*.[aod]
61
62 test: $(TEST_PATH)
63
64 -include $(DEPS)
65
66 $(TEST_PATH): $(TEST_OBJECTS) $(LIB_PATH)
67     $(CC) $(CFLAGS) $(CPPFLAGST) $^ -o $@ -lm $(LDFLAGS) $(LDLIBS)
```

Приложение. Текст программы

```
// file gen.c
1 #include <stdio.h>
2 #include "../LibPwgen/libgen.h"
3
4 int main()
5 {
6     libgen* t = malloc(sizeof(libgen));
7     Ui(t);
8     return 0;
9 }

// file libgen.c
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4 #include <sys/time.h>
5
6 #include "libgen.h"
7
8 int Getrand(int min, int max, double time, double pusk)
9 {
10     srand(time + pusk);
11     return (double)rand() / (RAND_MAX + 1.0) * (max - min) + min;
12 }
13
14 double wtime()
15 {
16     struct timeval t;
17     gettimeofday(&t, NULL);
18     return (double)t.tv_sec + (double)t.tv_usec * 1E-6;
19 }
20
21 void Ui(libgen* t)
22 {
23     t = malloc(sizeof(libgen) * t->kolvo);
24     char flag;
25
26     printf("Write the password length (digit)\n");
27     scanf(" %d", &t->dlin);
28
29     printf("Use capital letters?(y/n)\n");
30     scanf(" %c", &flag);
31     if (flag == 'y')
32         t->up = 1;
33
34     printf("Use small letters?(y/n)\n");
35     scanf(" %c", &flag);
36     if (flag == 'y')
37         t->down = 1;
38
39     printf("Use special characters?(y/n)\n");
40     scanf(" %c", &flag);
41     if (flag == 'y')
42         t->spets = 1;
```

```

43
44     printf("Write down how many passwords you need to generate (digit)\n");
45     scanf(" %d", &t->kolvo);
46     Good(t);
47 }
48
49 void Good(libgen* t)
50 {
51     int good[123];
52
53     for (int i = 33; i < 122; i++) {
54         good[i] = 1;
55     }
56
57     for (int i = 91; i <= 96; i++) {
58         good[i] = 0;
59     }
60
61     if (t->up == 0) {
62         for (int i = 65; i <= 90; i++) {
63             good[i] = 0;
64         }
65     }
66
67     if (t->down == 0) {
68         for (int i = 97; i <= 122; i++) {
69             good[i] = 0;
70         }
71     }
72
73     if (t->spets == 0) {
74         for (int i = 33; i <= 47; i++) {
75             good[i] = 0;
76         }
77
78         for (int i = 58; i <= 64; i++) {
79             good[i] = 0;
80         }
81     }
82     Generation(t, good);
83 }
84
85 void Generation(libgen* t, int* good)
86 {
87     double time = 0;
88     double pusik = 0;
89     int arr[t->dlin];
90     int i = 0;
91     int min = 33;
92     int max = 122;
93     time = wtime();
94     int tmp = Getrand(min, max, time, pusik);
95     pusik += 1;
96
97     for (i = 1; i < t->kolvo + 1; i++) {
98         for (int j = 0; j < t->dlin; j++) {
99             while (good[tmp] == 0) {

```

```

100         time = wtime();
101         tmp = Getrand(min, max, time, pusks);
102         pusks += 1;
103     }
104     arr[j] = tmp;
105     time = wtime();
106     tmp = Getrand(min, max, time, pusks);
107     pusks += 1;
108 }
109
110     Output(arr, i, t->dlin);
111 }
112 // free(t);
113 }
114
115 void Output(int* arr, int i, int dlin)
116 {
117     FILE* file = fopen("Results-passwords/Results.txt", "a");
118     fprintf(file, "Password №%d:\n ", i);
119     printf("Password №%d:\n", i);
120     for (int x = 0; x < dlin; x++) {
121         printf("%c", arr[x]);
122         fprintf(file, "%c", arr[x]);
123     }
124     for (int rev = 0; rev < 2; rev++) {
125         printf("\n");
126         fprintf(file, "\n");
127     }
128     fclose(file);
129 }

```

// file libgen.h

```

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4
5 typedef struct {
6     int dlin;
7     int up;
8     int down;
9     int spets;
10    int kolvo;
11 } libgen;
12
13
14 int Getrand(int min, int max, double time, double pusks);
15 void Ui(libgen* t);
16 void Good(libgen* t);
17 void Generation(libgen* t, int* good);
18 void Output(int* arr, int i, int dlin);
19 double wtime();

```

// file main.c

```

1 #define CTEST_MAIN
2
3 #include <ctest.h>

```

```

4
5 int main(int argc, const char** argv)
6 {
7     return ctest_main(argc, argv);
8 }

// file test.c
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4 #include <sys/time.h>
5
6 #include "../src/LibPwgen/libgen.h"
7 #include "../thirdparty/ctest.h"
8
9 int remfile()
10 {
11     if (remove("Results-passwords/Results.txt") == 0) {
12         printf("File deleted successfully.\n");
13     } else {
14         printf("Failed to delete the file.\n");
15     }
16     return 0;
17 }
18
19 CTEST(Getrand, returns_random_number_within_range)
20 {
21     int min = 0;
22     int max = 10;
23     double time = 12345.0;
24     double pusik = 6789.0;
25     int result = Getrand(min, max, time, pusik);
26
27     ASSERT_TRUE(result >= min && result <= max);
28 }
29
30 CTEST(wtime, returns_current_wall_time_in_seconds)
31 {
32     double result1 = wtime();
33     double result2 = wtime();
34
35     ASSERT_TRUE(result1 <= result2);
36 }
37
38 CTEST(getrand_suite, test_getrand)
39 {
40     int rand_num = Getrand(1, 10, 0.5, 0.2);
41     ASSERT_GE(rand_num, 1);
42     ASSERT_LE(rand_num, 10);
43 }
44
45 CTEST(generation_suite, test_generation)
46 {
47     remfile();
48     // Create the libgen structure
49     libgen* t = malloc(sizeof(libgen));

```

```

50     t->dlin = 8;
51     t->up = 1;
52     t->down = 1;
53     t->spets = 0;
54     t->kolvo = 5;
55
56     // Create an array good and fill it
57     int good[123] = {0};
58     for (int i = 48; i <= 57; i++) {
59         good[i] = 1; // numbers from 0 to 9
60     }
61     for (int i = 65; i <= 90; i++) {
62         good[i] = 1; // capital letters
63     }
64     for (int i = 97; i <= 122; i++) {
65         good[i] = 1; // lower case
66     }
67
68     // Execute the Generation function
69     Generation(t, good);
70
71     // Checking for the existence of a file with the results of generating
72     // passwords
73     FILE* file = fopen("Results-passwords/Results.txt", "r+");
74     ASSERT_NOT_NULL(file);
75
76     // Checking the number of generated passwords
77     char buffer[1024];
78     int cnt = 0;
79     while (fgets(buffer, 1024, file) != NULL) {
80         if (strstr(buffer, "Password")) {
81             cnt++;
82         }
83     }
84     ASSERT_EQUAL(cnt, t->kolvo);
85
86     fclose(file);
87     free(t);
88 }
89
90 CTEST(Good, modifies_array_of_good_characters_according_to_libgen_structure)
91 {
92     remfile();
93     libgen t = {8, 1, 0, 1, 5};
94
95     int good[123];
96     for (int i = 33; i <= 122; i++) {
97         good[i] = 1;
98     }
99
100     Good(&t);
101     ASSERT_TRUE(good['a']);
102     ASSERT_TRUE(good['!']);
103     ASSERT_TRUE(good['@']);
104     ASSERT_TRUE(good['#']);
105     ASSERT_TRUE(good['$']);
106     ASSERT_TRUE(good['%']);

```

```
107     ASSERT_TRUE(good[ '^' ] );
108     ASSERT_TRUE(good[ '&' ] );
109     ASSERT_TRUE(good[ '*' ] );
110
111     // Other characters must not be changed
112     ASSERT_TRUE(good[ 'B' ] );
113     ASSERT_TRUE(good[ 'z' ] );
114     ASSERT_TRUE(good[ '~' ] );
115 }
```