Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет

информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра Информатики

Дисциплина «Конструирование программ»

**ОТЧЕТ**

к лабораторной работе №7

на тему:

**«ИНТЕГРАЦИЯ АССЕМБЛЕРНЫХ ПРЕРЫВАНИЙ В ПРОЕКТЫ НА C++»**

БГУИР 6-05-0612-02 02

|  |
| --- |
| Выполнил студент группы 353503  АБДУЛОВ Александр Алексеевич |
|  |
| (дата, подпись студента) |
| Проверил ассистент каф. Информатики  РОМАНЮК Максим Валерьевич |
|  |
| (дата, подпись преподавателя) |

Минск 2024

# 1 Индивидуальное задание

1. **Задание 1. Вариант 2.** **Менеджер паролей.**На стороне Assembler: Реализация алгоритма поточного шифрования RC4, используя assembler-функции. Генератор случайных чисел: На базе системного таймера или других источников энтропии для создания случайных паролей.

Пользователь может сохранять пароли и другую конфиденциальную информацию. Данные хранятся в зашифрованном виде. Реализуйте функцию генерации случайных паролей. Тайм-аут бездействия: Если менеджер паролей открыт и не используется в течение заданного времени, автоматически блокируйте его.   
На стороне C++:  
 Главное меню и пользовательский интерфейс. Управление функциями ассемблера: добавление, удаление и редактирование записей; генерация пароля; шифрование и дешифрование данных. Логика тайм-аута бездействия.

# 2 Выполнение работы

Для написания данной лабораторной работы используется NASM и редактор VS Code в среде Linux. Ниже представлен листинг кода файла .asm.

default rel

SECTION .data align=1 noexec ; section number 2, data

SECTION .bss align=1 noexec ; section number 3, bss

global generate\_password: function

global swap: function

global rc4\_init: function

global rc4\_crypt: function

extern \_\_stack\_chk\_fail

extern rand

extern srand

extern time

SECTION .text align=1 exec ; section number 1, code

generate\_password:; Function begin

push rbp

mov rbp, rsp

add rsp, -128

mov qword [rbp-78H], rdi

mov dword [rbp-7CH], esi

mov rax, qword [fs:abs 28H]

mov qword [rbp-8H], rax

xor eax, eax

mov rax, qword 6867666564636261H

mov rdx, qword 706F6E6D6C6B6A69H

mov qword [rbp-96H], rax

mov qword [rbp-58H], rdx

mov rax, qword 7877767574737271H

mov rdx, qword 4645444342417A79H

mov qword [rbp-50H], rax

mov qword [rbp-48H], rdx

mov rax, qword 4E4D4C4B4A494847H

mov rdx, qword 565554535251504FH

mov qword [rbp-40H], rax

mov qword [rbp-38H], rdx

mov rax, qword 333231305A595857H

mov rdx, qword 4021393837363534H

mov qword [rbp-30H], rax

mov qword [rbp-28H], rdx

mov rax, qword 29282A265E252423H

mov qword [rbp-20H], rax

mov word [rbp-18H], 11103

mov byte [rbp-16H], 0

mov dword [rbp-64H], 74

mov edi, 0

call time

mov edi, eax

call srand

mov dword [rbp-68H], 0

jmp startloop1

random:

call rand

cdq

idiv dword [rbp-64H] ;edx - ost || eax - res

mov ecx, edx

mov eax, dword [rbp-68H]

movsxd rdx, eax

mov rax, qword [rbp-78H]

add rdx, rax

movsxd rax, ecx

movzx eax, byte [rbp+rax-60H]

mov byte [rdx], al

add dword [rbp-68H], 1

startloop1:

mov eax, dword [rbp-68H]

cmp eax, dword [rbp-7CH]

jl random

mov eax, dword [rbp-7CH]

movsxd rdx, eax

mov rax, qword [rbp-78H]

add rax, rdx

mov byte [rax], 0

nop

mov rax, qword [rbp-8H]  
 sub rax, qword [fs:abs 28H]

jz gen\_end

call \_\_stack\_chk\_fail

gen\_end: leave

ret

; generate\_password End of function

swap: ; Function begin

push rbp

mov rbp, rsp

mov qword [rbp-18H], rdi

mov qword [rbp-20H], rsi

mov rax, qword [rbp-18H]

movzx eax, byte [rax]

mov byte [rbp-1H], al

mov rax, qword [rbp-20H]

movzx edx, byte [rax]

mov rax, qword [rbp-18H]

mov byte [rax], dl

mov rax, qword [rbp-20H]

movzx edx, byte [rbp-1H]

mov byte [rax], dl

nop

pop rbp

ret

; swap End of function

rc4\_init:; Function begin

push rbp

mov rbp, rsp

sub rsp, 40

mov qword [rbp-18H], rdi

mov dword [rbp-1CH], esi

mov qword [rbp-28H], rdx

mov dword [rbp-4H], 0

mov dword [rbp-8H], 0

jmp startloop2

loop2: mov eax, dword [rbp-8H]

movsxd rdx, eax

mov rax, qword [rbp-28H]

add rax, rdx

mov edx, dword [rbp-8H]

mov byte [rax], dl

add dword [rbp-8H], 1 ;i++

startloop2: cmp dword [rbp-8H], 255

jle loop2

mov dword [rbp-8H], 0

jmp compareloop  
 cypher: mov eax, dword [rbp-8H] ;i ; S=[0-255],

movsxd rdx, eax

mov rax, qword [rbp-28H]

add rax, rdx

movzx eax, byte [rax]

movzx edx, al ;S[i]

mov eax, dword [rbp-4H] ;j

lea ecx, [rdx+rax]

mov eax, dword [rbp-8H]

cdq

idiv dword [rbp-1CH] ; % length

mov eax, edx

movsxd rdx, eax

mov rax, qword [rbp-18H]

add rax, rdx

movzx eax, byte [rax]

movzx eax, al

add eax, ecx

cdq

shr edx, 24

add eax, edx

movzx eax, al

sub eax, edx

mov dword [rbp-4H], eax

mov eax, dword [rbp-4H]

movsxd rdx, eax

mov rax, qword [rbp-28H]

add rdx, rax

mov eax, dword [rbp-8H]

movsxd rcx, eax

mov rax, qword [rbp-28H]

add rax, rcx

mov rsi, rdx

mov rdi, rax

call swap

add dword [rbp-8H], 1 ;i++

compareloop:

cmp dword [rbp-8H], 255

jle cypher

nop

nop

leave

ret

; rc4\_init End of function

rc4\_crypt:; Function begin

push rbp

mov rbp, rsp

sub rsp, 40

mov qword [rbp-18H], rdi ;str

mov dword [rbp-1CH], esi ;length

mov qword [rbp-28H], rdx ;S[]

mov dword [rbp-10H], 0

mov dword [rbp-0CH], 0

mov dword [rbp-8H], 0

jmp comparloop3

startloop3: mov eax, dword [rbp-10H]

add eax, 1

cdq

shr edx, 24

add eax, edx

movzx eax, al

sub eax, edx

mov dword [rbp-10H], eax ;i  
 mov eax, dword [rbp-10H]

movsxd rdx, eax

mov rax, qword [rbp-28H]

add rax, rdx

movzx eax, byte [rax] ;s[i]

movzx edx, al

mov eax, dword [rbp-0CH]

add eax, edx

cdq

shr edx, 24

add eax, edx

movzx eax, al

sub eax, edx

mov dword [rbp-0CH], eax ;j

mov eax, dword [rbp-0CH]

movsxd rdx, eax

mov rax, qword [rbp-28H]

add rdx, rax ;s[j]

mov eax, dword [rbp-10H]

movsxd rcx, eax

mov rax, qword [rbp-28H]

add rax, rcx ;s[i]

mov rsi, rdx

mov rdi, rax

call swap

mov eax, dword [rbp-10H]

movsxd rdx, eax

mov rax, qword [rbp-28H]

add rax, rdx

movzx edx, byte [rax]

mov eax, dword [rbp-0CH]

movsxd rcx, eax

mov rax, qword [rbp-28H]

add rax, rcx

movzx eax, byte [rax]

add eax, edx

movzx eax, al

mov dword [rbp-4H], eax

mov eax, dword [rbp-8H]

movsxd rdx, eax

mov rax, qword [rbp-18H]

add rax, rdx

movzx ecx, byte [rax]

mov eax, dword [rbp-4H]

movsxd rdx, eax

mov rax, qword [rbp-28H]

add rax, rdx

movzx edx, byte [rax]

mov eax, dword [rbp-8H]

movsxd rsi, eax

mov rax, qword [rbp-18H]

add rax, rsi

xor edx, ecx ;XOR

mov byte [rax], dl

add dword [rbp-8H], 1

comparloop3:

mov eax, dword [rbp-8H]

cmp eax, dword [rbp-1CH]

jl startloop3

nop

nop

leave

ret

; rc4\_crypt End of function

Программа включает следующие сегменты: .data cодержит инициализированные данные, bss используется для неинициализированных данных, .text содержит исполняемый код. В программе объявлены глобальные функции generate\_password, swap, rc4\_init и rc4\_crypt, а также внешние функции \_\_stack\_chk\_fail, rand, srand и time. Функция generate\_password генерирует случайный пароль, используя функции rand и srand. Функция swap выполняет обмен значениями двух элементов массива, используемая для перестановки элементов массива состояния S. Функция rc4\_init инициализирует массив состояния S и выполняет перестановку элементов на основе ключа, подготавливая массив состояния для последующего использования в шифровании. Функция rc4\_crypt выполняет шифрование или расшифрование данных с использованием алгоритма RC4. Ниже представлен листинг кода файла .cpp.

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <time.h>

#define MAX\_RECORDS 100

#define MAX\_NAME\_LENGTH 50

#define MAX\_PASSWORD\_LENGTH 50

typedef struct {

char name[MAX\_NAME\_LENGTH];

char username[MAX\_NAME\_LENGTH];

char password[MAX\_PASSWORD\_LENGTH];

} Record;

extern void generate\_password(char \*password, int length);

extern void swap(unsigned char \*a, unsigned char \*b);

extern void rc4\_init(unsigned char \*key, int key\_length, unsigned char \*S);

extern void rc4\_crypt(unsigned char \*data, int data\_length, unsigned char \*S);

void add\_record(Record \*records, int \*num\_records)

{

if (\*num\_records >= MAX\_RECORDS)

{

printf("Maximum number of records reached.\n");

return;

}

Record new\_record;

printf("Enter the name of the application or website: ");

fgets(new\_record.name, MAX\_NAME\_LENGTH, stdin);

new\_record.name[strcspn(new\_record.name, "\n")] = '\0';

printf("Enter the username: ");

fgets(new\_record.username, MAX\_NAME\_LENGTH, stdin);

new\_record.username[strcspn(new\_record.username, "\n")] = '\0';

printf("Enter the password (or skip this field to generate): ");

fgets(new\_record.password, MAX\_PASSWORD\_LENGTH, stdin);

new\_record.password[strcspn(new\_record.password, "\n")] = '\0';

if (strlen(new\_record.password) == 0)

{

generate\_password(new\_record.password, 12);

printf("Generated password: %s\n", new\_record.password);

}

records[\*num\_records] = new\_record;

(\*num\_records)++;

}

void delete\_record(Record \*records, int \*num\_records)

{

if (\*num\_records == 0)

{

printf("No records to delete.\n");

return;

}

printf("Enter the name of the record to delete: ");

char name[MAX\_NAME\_LENGTH];

fgets(name, MAX\_NAME\_LENGTH, stdin);

name[strcspn(name, "\n")] = '\0';

int index = -1;

for (int i = 0; i < \*num\_records; i++)

{

if (strcmp(records[i].name, name) == 0)

{

index = i;

break;

}

}

if (index == -1)

{

printf("Record not found.\n");

return;

}

for (int i = index; i < \*num\_records - 1; i++)

{

records[i] = records[i + 1];

}

(\*num\_records)--;

}

void edit\_record(Record \*records, int num\_records)

{

if (num\_records == 0)

{

printf("No records to edit.\n");

return;

}

printf("Enter the name of the record to edit: ");

char name[MAX\_NAME\_LENGTH];

fgets(name, MAX\_NAME\_LENGTH, stdin);

name[strcspn(name, "\n")] = '\0';

int index = -1;

for (int i = 0; i < num\_records; i++)

{

if (strcmp(records[i].name, name) == 0)

{

index = i;

break;

}

}

if (index == -1)

{

printf("Record not found.\n");

return;

}

Record \*record = &records[index];

printf("Enter the new name of the application or website (or skip this field to keep \"%s\"): ", record->name);

char new\_name[MAX\_NAME\_LENGTH];

fgets(new\_name, MAX\_NAME\_LENGTH, stdin);

new\_name[strcspn(new\_name, "\n")] = '\0';

if (strlen(new\_name) > 0)

{

strcpy(record->name, new\_name);

}

printf("Enter the new username (or skip this field to keep \"%s\"): ", record->username);

char new\_username[MAX\_NAME\_LENGTH];

fgets(new\_username, MAX\_NAME\_LENGTH, stdin);

new\_username[strcspn(new\_username, "\n")] = '\0';

if (strlen(new\_username) > 0)

{

strcpy(record->username, new\_username);

}

printf("Enter the new password (or skip this field to keep \"%s\"): ", record->password);

char new\_password[MAX\_PASSWORD\_LENGTH];

fgets(new\_password, MAX\_PASSWORD\_LENGTH, stdin);

new\_password[strcspn(new\_password, "\n")] = '\0';

new\_password[strcspn(new\_password, "\n")] = '\0';

if (strlen(new\_password) > 0)

{

strcpy(record->password, new\_password);

}

}

void print\_records(Record \*records, int num\_records)

{

char key[] = "mysecretkey";

printf("Index\tName\tUsername\tPassword\n");

if (num\_records == 0){

printf("No records found.\n");

return;

}

for (int i = 0; i < num\_records; i++){

printf("%d\t%s\t%s\t%s\n", i, records[i].name, records[i].username, records[i].password);

}

}

int main(){

char key[] = "";

Record records[MAX\_RECORDS];

int num\_records = 0;

unsigned char S[256];

FILE \*file = fopen("data.txt", "r");

if (file != NULL){

while (num\_records < MAX\_RECORDS && fscanf(file, "%s %s %s", records[num\_records].name, records[num\_records].username, records[num\_records].password) == 3) {

unsigned char S[256];

char key[] = "mysecretkey";

rc4\_init(key, strlen(key), S);

rc4\_crypt(records[num\_records].name, strlen(records[num\_records].name), S);

rc4\_init(key, strlen(key), S);

rc4\_crypt(records[num\_records].username, strlen(records[num\_records].username), S);

rc4\_init(key, strlen(key), S);

rc4\_crypt(records[num\_records].password, strlen(records[num\_records].password), S);

num\_records++;

}

fclose(file);

}

int choice;

time\_t last\_activity\_time = time(NULL);

while (1){

printf("\n---------Menu---------\n");

printf("1. Add record\n");

printf("2. Delete record\n");

printf("3. Edit record\n");

printf("4. Print records\n");

printf("5. Exit and save\n");

printf("6. Exit\n");

printf("---------Choose comand---------\n");

printf("Enter: ");

scanf("%d", &choice);

getchar();

time\_t current\_time = time(NULL);

if (current\_time - last\_activity\_time > 30){

printf("Timeout, program has been inactive for 30 sec.\n");

break;

}

switch (choice){

case 1:

system("clear");

add\_record(records, &num\_records);

break;

case 2:

system("clear");

delete\_record(records, &num\_records);

break;

case 3:

system("clear");

edit\_record(records, num\_records);

break;

case 4:

system("clear");

print\_records(records, num\_records);

break;

case 5:

system("clear");

file = fopen("data.txt", "w");

if (file != NULL) {

for (int i = 0; i < num\_records; i++)

{

char key[] = "mysecretkey";

rc4\_init(key, strlen(key), S);

rc4\_crypt(records[i].name, strlen(records[i].name), S);

rc4\_init(key, strlen(key), S);

rc4\_crypt(records[i].username, strlen(records[i].username), S);

rc4\_init(key, strlen(key), S);

rc4\_crypt(records[i].password, strlen(records[i].password), S);

fprintf(file, "%s %s %s\n", records[i].name, records[i].username, records[i].password);

}

fclose(file);

}

printf("Exiting...\n");

return 0;

case 6:

return 0;

default:

printf("Invalid value.\n");

break;

}

last\_activity\_time = current\_time;

}

return 0;

}

В файле .cpp определены внешние функции generate\_password(), swap(), rc4\_init() и rc4\_crypt, которые реализуют условие задания.

# Вывод

В ходе лабораторной работы был изучен ассемблер NASM и интеграция его с языком C++ в среде VS Code на Linux, а также основы написания ассемблерного кода и методы его вызова из C++, что значительно расширяет функциональные возможности разработанной программы.