Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ **ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ**

КАФЕДРА **КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.01 Информатика и вычислительная техника**

ОТЧЕТ

По лабораторной работе №5

Название: Программирование с использование разноязыковых модулей

Дисциплина: Машинно-зависимые языки и основы компиляции

Студент	ИУ-42б	_____	С.В. Астахов
	(Группа)	(Подпись, дата)	(И.О. Фамилия)
Преподаватель		_____	_____
		(Подпись, дата)	(И.О. Фамилия)

Москва, 2021

Задание

Вариант 1.

Дан текст не более 255 символов. Определить максимальный повторяющийся фрагмент. Использовать конвенцию stdcall.

Исходный код

Первый модуль

(ConsoleApplication1.cpp)

```
#include <iostream>
#include "Header.h"

int main()
{
    std::cout << "\nInput string to search for repeats:\n";
    char str[255];
    std::cin.getline(str, 255);
    //scanf_s("%s", str);
    //std::cin >> str;
    std::cout << "\nEcho:\n";
    std::cout << str;
    STR_REPEATS(str);
    std::cout << "\n(press any key to exit)\n";
}
```

(Header.h)

```
#pragma once
extern "C" void __stdcall STR_REPEATS(char* s);
```

Второй модуль

(repeats.asm → repeats.obj)

```
        .586
        .MODEL flat
        .DATA
example_str db "str from asm"
pattern    DB    255 DUP (0)
res_lg SDWORD 0
i_reps SDWORD 0
s_length DWORD 0
s_pos DWORD 0
result  DB  255 DUP (0)
test_str  DB    255 DUP (0)
        .CODE

public _STR_REPEATS@4
externdef ?PRINT@@YGXPAD@Z:near
```

```
_STR_REPEATS@4 proc
    ;mov EBP, ESP
    XOR    EAX,EAX

    lea eax, example_str
    mov ebx, ESP
    mov eax, [ebx+4]
    ;=======================

    ;code there

    mov ecx, 255
    mov esi, eax
    lea edi, test_str
    rep movsb

    mov s_length, 254
         mov res_lg, 0

         ;lea ESI, test_str
         mov ECX, 255
         cycle_1:
            push ECX

            mov s_pos, 0
            mov ECX, 255
            sub ECX, s_length
            cycle_2:
                push ECX

                lea ESI, test_str
                lea EDI, pattern
                add ESI, s_pos
                ;code
                mov ECX, s_length
                rep movsb

                mov AL, 0
                stosb

                push esi
                push edi

                pop esi
                pop edi

                mov i_reps, 0
                mov s_pos, 0
                mov ECX, 255
                sub ECX, s_length
                cycle_3:
                    push ECX
                    lea ESI, test_str
                    lea EDI, pattern
                    add ESI, s_pos

                    mov ECX, s_length
                    repe cmpsb
                    jne skip
```

```
                        inc i_reps


                        cmp i_reps, 2
                        jne skip_2
                            mov EAX, res_lg
                            mov EBX, s_length
                            cmp EBX, EAX
                            jl skip_3              ; here we can change first or
last
                                mov ECX, s_length
                                lea ESI, pattern
                                lea EDI, result
                                rep movsb
                                mov AL,0
                                stosb
                                mov EAX, s_length
                                mov res_lg, EAX
                            skip_3:
                        skip_2:
                        ;mov ECX, s_length
                        ;lea ESI, pattern
                        ;lea EDI, result
                        ;rep movsb
                        ;jmp final
                    skip:

                    pop ECX
                    inc s_pos
                    dec ECX
                    cmp ECX, 0
                    jne cycle_3

                pop ECX
                inc s_pos
                dec ECX
                cmp ECX, 0
            jne cycle_2

            dec s_length
            ;Invoke StdOut,ADDR MsgOutc
            ;Invoke StdOut,ADDR MsgLn
            pop ECX
            dec ECX
            cmp ECX,0
        jne cycle_1

    lea eax, result
    ;=======================
    push eax

    call ?PRINT@@YGXPAD@Z
    ;mov ESP, EBP
    ret 4

_STR_REPEATS@4 endp
end
```

Третий модуль

(printer.cpp)

```
#include <iostream>
#include "printer.h"
#include <string.h>

extern void __stdcall PRINT(char* str) {
 int lg;
 lg = strlen(str);
 if (lg > 0) {
      printf("\nResult: %s", str);
 }
 else {
      std::cout << "\nNo repeats found\n";
 }

}
```

(printer.h)

```
#pragma once
void __stdcall PRINT(char *str);
```

Схема алгоритма

На следующих рисунках приведены схемы алгоритмов всех модулей программы.

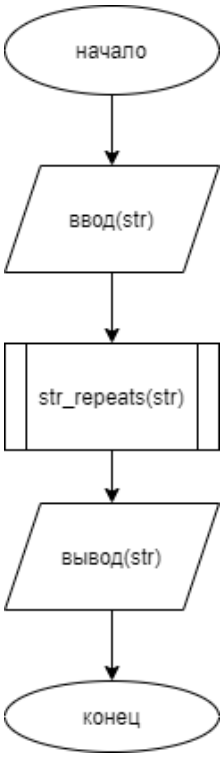

Рисунок 1 — схема алгоритма первого модуля

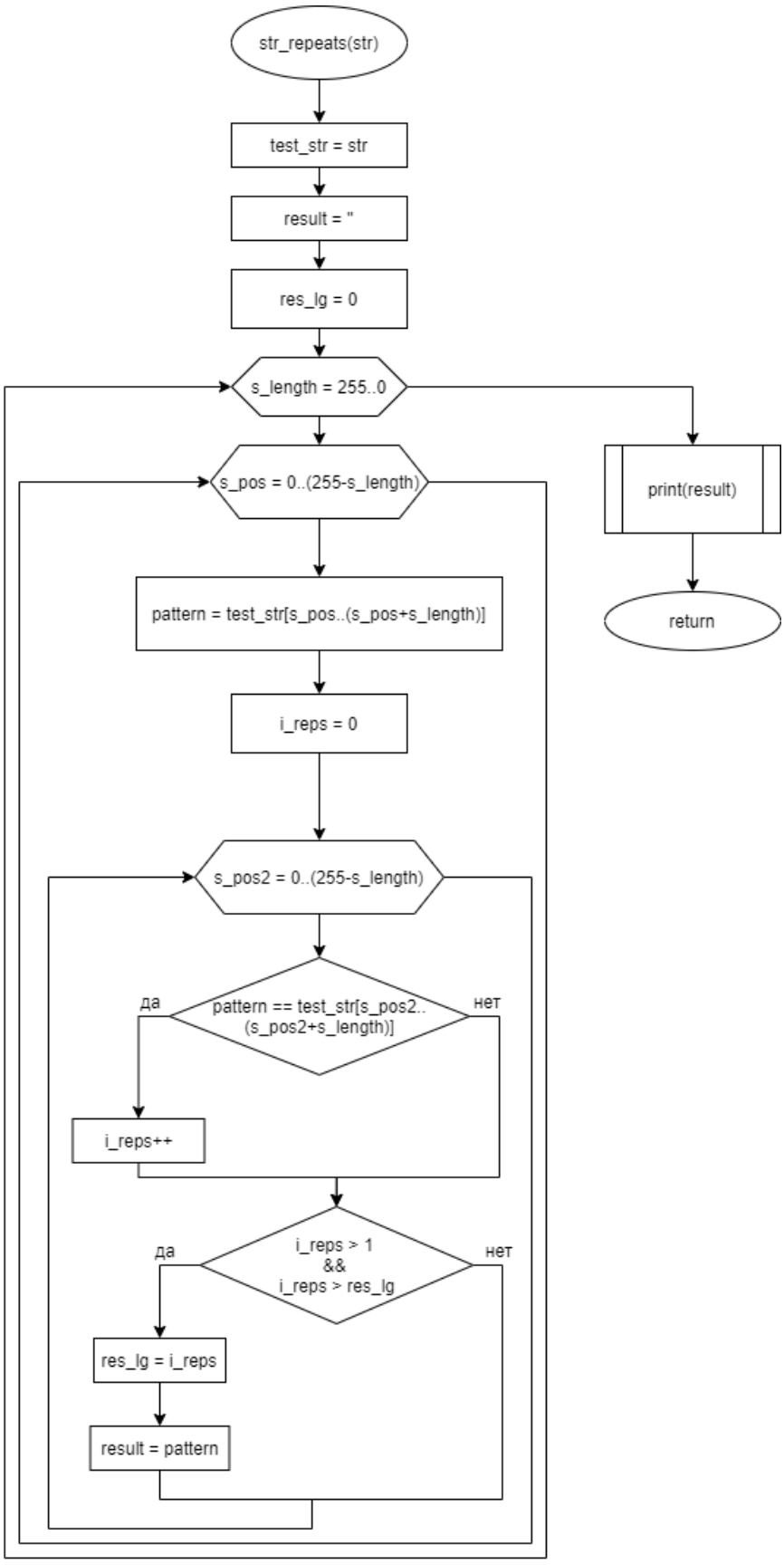

Рисунок 2 - схема алгоритма второго модуля

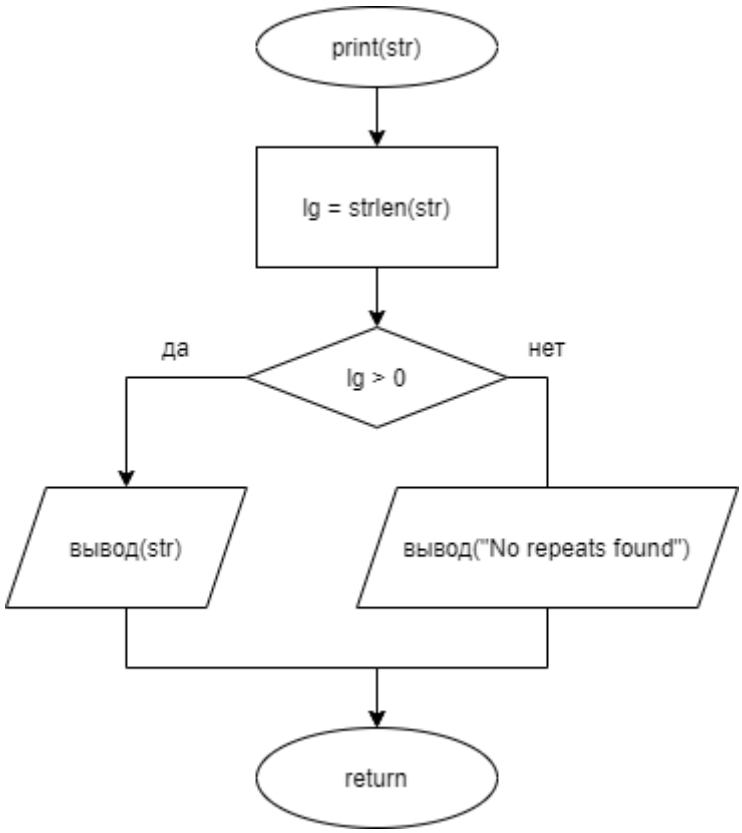

Рисунок 3 - схема алгоритма третьего модуля

Содержимое стека

При передаче из первого модуля во второй:

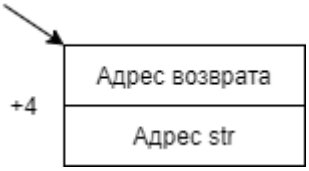

Рисунок 4 - схематическое изображение содержимого стека в момент передачи управления

При передаче из второго модуля в третий:

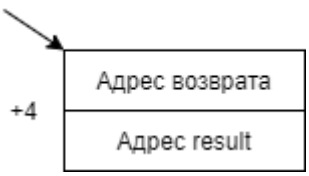

Рисунок 5 - схематическое изображение содержимого стека в момент передачи управления

Тесты

Таблица 1 — тестирование программы

Входные данные	Ожидаемый вывод	Вывод
abc123456	No repeats found	No repeats found
abc 23 abcd	Result: abc	Result: abc
(пустая строка)	No repeats found	No repeats found
abc abc abc	Result: abc abc	Result: abc abc
~@# ffg ~@#	Result: ~@#	Result: ~@#
абв абв арапкп23 абвгд	Result: абв а	Result: абв а

Контрольные вопросы

1. Что такое «конвенции о связи»? Перечислите конвенции, которые вы знаете, и уточните их содержание.

Конвенции о связи определяют правила передачи параметров при связи модулей.

Конвенция Pascal предполагает, что параметры помещаются в стек в том порядке, в котором они встречаются в списке формальных параметров подпрограммы. Завершаясь, подпрограмма удаляет параметры из стека, а потом возвращает управление.

Конвенция C предполагает обратный порядок помещения параметров в стек, и параметры из стека удаляет вызывающая программа.

Конвенции stdcall и safecall используют обратный порядок занесения параметров в стек и очистку стека вызываемой процедурой. Отличие между ними в том, что safecall формирует исключение при обнаружении ошибок, связанных с передачей параметров.

Конвенция register означает передачу до трех параметров в регистрах. Обычно этого хватает, но если параметров больше, то остальные передаются через стек.

2. Какие конвенции вы использовали при создании своей программы?

Stdcall.

3. Как связана структура данных стека в момент передачи управления и текст программы и подпрограмм?

В случае передачи управления из программы на языке высокого уровня — параметры помещаются в стек автоматически в соответствии с порядком в описании функции и указанной конвенцией. В случае программирования на языке ассемблера, необходимо явным образом извлекать и помещать параметры в стек, а так же очищать его при возврате управления.

4. С какой целью применяют разноязыковые модули в одном проекте?

а) С целью повышения эффективности часто используемых или вычислительно сложных функций, за счет их реализации на языках низкого уровня.

б) При необходимости вызова функции, уже реализованной на языке, отличном от основного языка системы.

Вывод: В ходе лабораторной работы были изучены «конвенции о связи», используемые для передачи управления между разноязыковыми модулями, а также освоены базовые навыки программирования с использование разноязыковых модулей.