АВС | ИДЗ-3 | Вариант 24

Потякин Арсений Юрьевич, БПИ237

TODO:
Разработать программу, которая ищет в ASCII–строке заданную подстроку и возвращает список индексов первого символа для всех вхождений подстроки в строке. Подстрока вводится как параметр. Вывод результатов организовать в файл (используя соответствующие преобразования чисел в строки)

Код можно найти здесь: GitHub

Важное уточнение: macrolib.s и subroutines.s для ручных тестов и автоматических тестов отличаются тем, что в macrolib для автотестов есть макросы для вывода строки/числа в консоль (в ручных тестах это выводится в виде графики на экран), а также тем, что подпрограмма для проведения автоматических тестов не предлагает вывести данные в консоль. Если будете проверять работоспособность программы, пожалуйста, скачайте соответствующие версии файлов в папках.

Демонстрация работы:

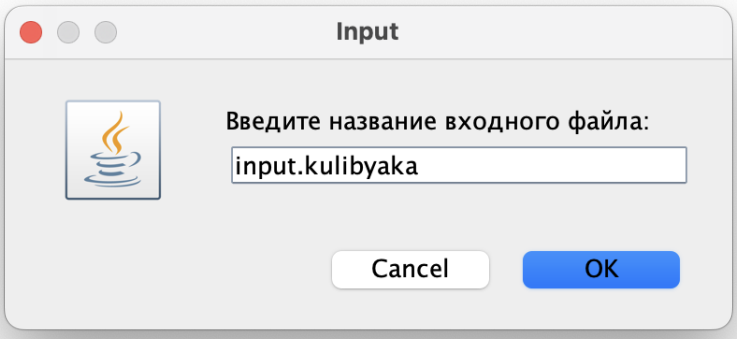

Рис. 1: Ввод не существующего файла влечет за собой выдачу ошибки (см. рис 2)

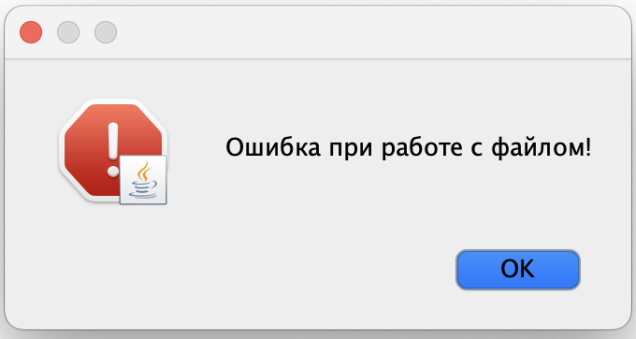

Рис. 2: Программа не завершается аварийно, а уведомляет пользователя

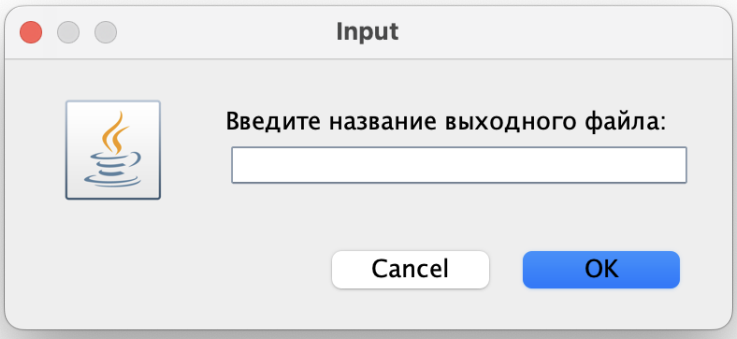

Рис. 3: При корректном вводе названия файла с входными данными пользователь может ввести имя выходного файла. В случае, если файла с таким именем не существует, он создастся автоматически с расширением .txt. **Допустим, введем output1.txt**

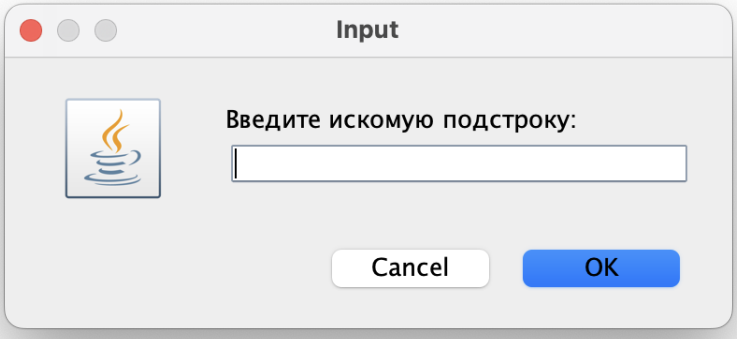

Рис. 4: При корректном вводе названия выходного файла пользователь может ввести искомую подстроку. **Допустим, введем is**

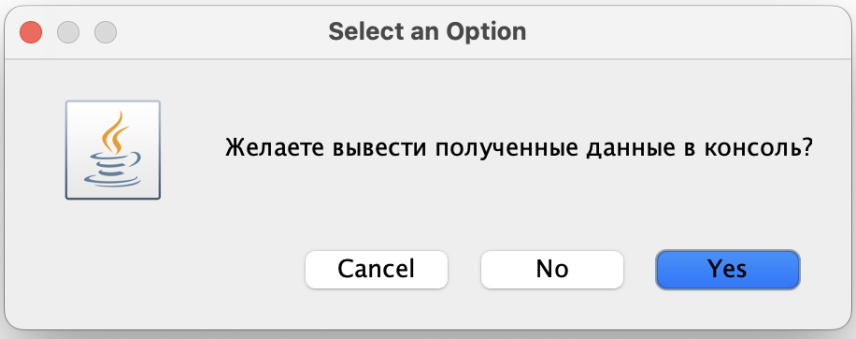

Рис. 5: После ввода искомой подстроки пользователь может выбрать, хочет ли он видеть результат в консоли RARS. No/Cancel – данные не будут выведены в консоль. Yes – данные отобразятся в консоли. Вне зависимости от выбора данные будут выведены в файл. **Допустим, выбираем Yes**

```
-- program is finished running (0) --

5 10 466
-- program is finished running (0) --
```

Рис. 6: Данные вывелись в консоль

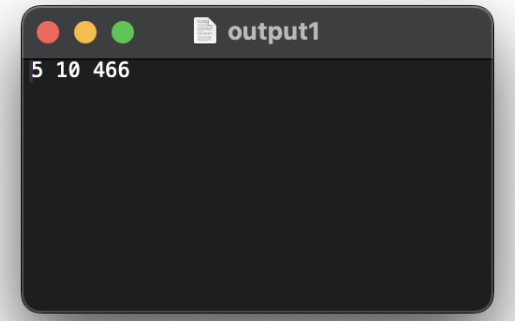

Рис. 7: И, как и полагается, данные сохранены в файле

Автотесты:
Первые три теста – рандомный текст длиной не более 5000 символо, четвертый тест – 10240 символов (и столько же байт, соответственно), пятый тест – ровно в два раза больше символов (20480), задача двух последних тестов показать, что при превышении размера буфера программа не падает и обрабатывает ту часть текста, которая поместилась в буфер.

```
Введите искомую подстроку: a
Тест 1 выполнен успешно!
Тест 2 выполнен успешно!
Тест 3 выполнен успешно!
Тест 4 выполнен успешно!
Тест 5 выполнен успешно!
```

Рис. 8: После запуска автотестов все тесты были пройдены успешно (при условии, что такие файлы существуют в директории)

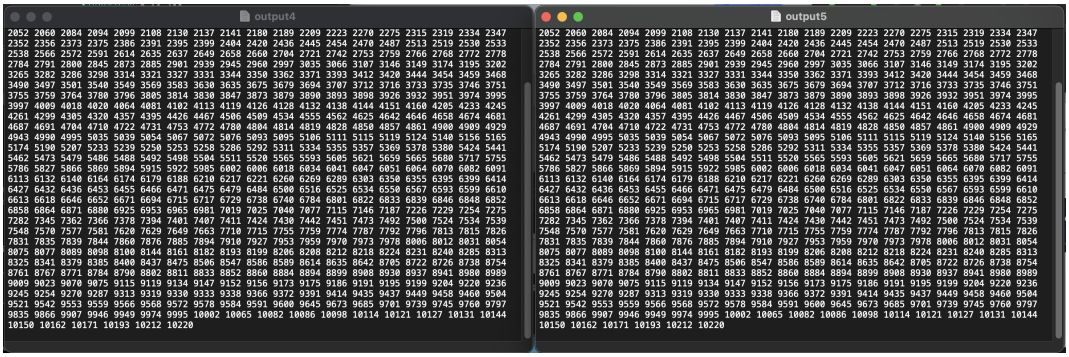

Рис. 9: При этом несмотря на то, что тест 5 является удвоенным текстом теста 4, выходные файлы совпадают. Из этого можно сделать вывод, что программа успешно читает не более 10240 байт и не падает при превышении ограничения

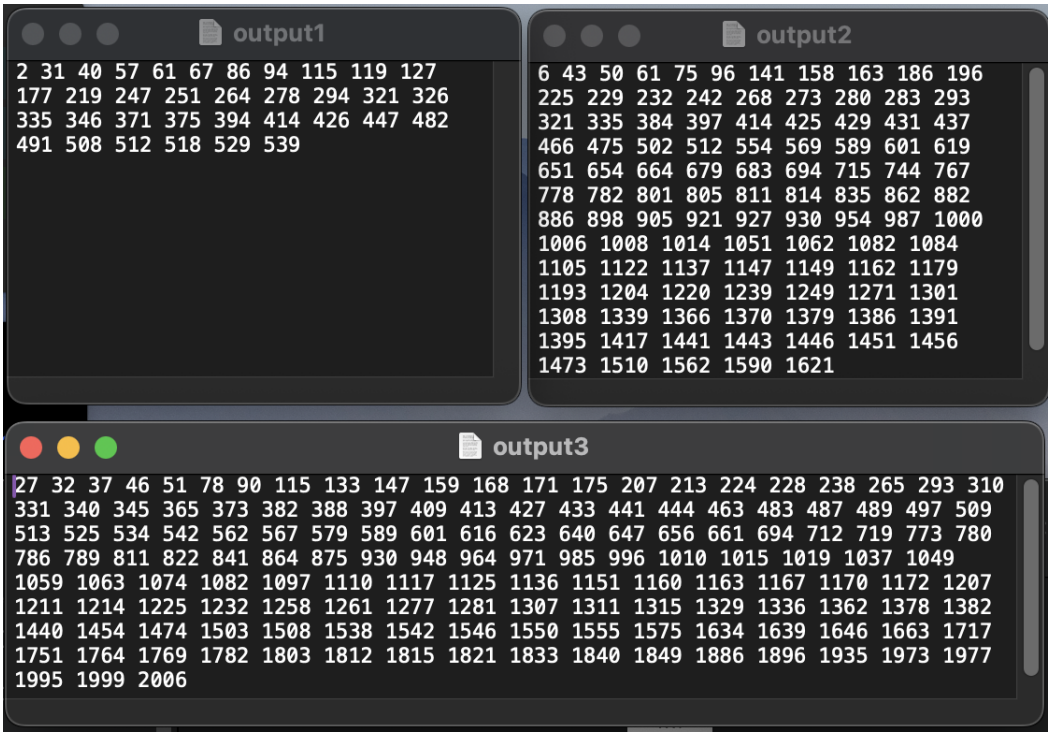

Рис. 10: Остальные выходные файлы содержат соответствующие обработанным входным данным информацию

Соответствие критериям:

```
73          li a1 100
74
75          # Удаляем символ новой строки из подстроки
76          la t0 substring_buffer
77          remove_newline t0
78
79          # Ввод Y/N для вывода данных в консоль
80          mv a1 a0
81          choose_dialog (output_data_to_console_msg)
82          mv a6 a0                # Переносим выбор пользователя в регистр а6
83          mv a0 a1                # Возврат адреса буфера в регистр а0
84
85          # Инициализируем счетчики
86          li s2 0                 # Общий счетчик прочитанных байт
87          li s3 0                 # Смещение в буфере
88          li t6 10240             # Максимальный размер файла (10 КБ)
89
90  read_loop:
91          # Вычисляем количество оставшихся байт для чтения
92          li t0 10240
93          sub t0 t0 s2            # t0 = 10240 - s2 (оставшиеся байты)
94
95          # Если t0 == 0, достигнут максимальный размер, завершаем чтение
96          beq t0 zero end_read_loop
97
98          # Определяем, сколько байт читать в текущей итерации
99          li t1 512
100         blt t0 t1 set_bytes_to_read
101         mv t2 t1                # bytes_to_read = 512
102         j set_a1_a2
```

Рис. 11: Весь код содержит подробные комментарии на всех участках

```
# Service to display a message to a user and request a string input
# Параметры:
#    a0 - address of null-terminated string that is the message to user
#    a1 - address of input buffe
#    a2 - maximum number of characters to read (including the terminating null)
# Возвращаемое значение:
#    a1 - buffer contains the maximum allowable input string terminated with null.
INPUT_DIALOG:
        # Выделяем место на стеке для адреса возврата и сохраняем его
        addi sp sp -4
        sw ra 0(sp)

        # Вызываем диалоговое окно
        li a7 54
        ecall

        # Восстанавливаем стек
        lw ra 0(sp)
        addi sp sp 4
        jr ra
```

Рис. 12: Перед каждой подпрограммой есть вся необходимая информация о том, что делает подпрограмма
P.S. Комментарии на английском так как взяты из HELP внутри RARS.

macrolib.s subroutines.s

```
find_substring:
        addi sp sp -4
        sw ra 0(sp)

        mv s3 a0        # Адрес текста
        mv s4 a1        # Длина текста
        mv s5 a2        # Адрес подстроки
        mv s8 a3        # Дескриптор выходного файла
        mv s9 a6        # Выбор пользователя Y/N
        li s6 0         # Текущий индекс в тексте

        # Находим длину подстроки
        mv t0 a2
        li s7 0    # Длина подстроки
strlen_loop:
        lb t1 (t0)
        beqz t1 strlen_done
        addi s7 s7 1
        addi t0 t0 1
        j strlen_loop
strlen_done:
# Основной цикл поиска
main_search_loop:
        bge s6 s4 search_done    # Если достигли конца текста

        # Проверяем, есть ли достаточно символов для подстроки
```

Рис. 13: Обработка данных, полученных из файла сформирована в виде отдельной подпрограммы

```
.data
        buffer: .space 10240            # Буфер для хранения текста (10 кбайт = 10 * 1024)
        substring_buffer: .space 512    # Буфер для хранения подстроки
        number_buffer: .space 12        # Буфер для хранения числа в виде строки
        # Вывод для пользователя
        input_file_msg: .asciz "Введите название входного файла: "
```

Рис. 14: Буфер для текста имеет размер 10кб. При чтении файла большего размера программа не падает, а обрабатывает "урезанный текст"(см. автотесты)

```
84
85          # Инициализируем счетчики
86          li s2 0                 # Общий счетчик прочитанных байт
87          li s3 0                 # Смещение в буфере
88          li t6 10240             # Максимальный размер файла (10 КБ)
89
90  read_loop:
91          # Вычисляем количество оставшихся байт для чтения
92          li t0 10240
93          sub t0 t0 s2            # t0 = 10240 - s2 (оставшиеся байты)
94
95          # Если t0 == 0, достигнут максимальный размер, завершаем чтение
96          beq t0 zero end_read_loop
97
98          # Определяем, сколько байт читать в текущей итерации
99          li t1 512
100         blt t0 t1 set_bytes_to_read
101         mv t2 t1                # bytes_to_read = 512
102         j set_a1_a2
103 set_bytes_to_read:
104         mv t2 t0                # bytes_to_read = оставшиеся байты
105 set_a1_a2:
```

Рис. 15: Для чтения текста выделен буфер 10кб, при этом программа за одну итерацию считывает лишь 512 байт, т.е. последовательно считывает до 10 кб за несколько итераций

```
# Подпрограмма удаления символа новой строки
# Параметры:
#    4(sp) - адрес строки
# Возвращаемое значение:
#    Нет
REMOVE_NEWLINE:
        addi sp sp -4
        sw ra 0(sp)

        # Восстанавливаем значение адреса строки из стека
        lw t1 4(sp)
remove_newline_loop:
        lb t2 (t1)
        beqz t2 remove_newline_end
        li t3 10        # Код символа новой строки
        beq t2 t3 remove_newline_found
        addi t1 t1 1
        j remove_newline_loop
remove_newline_found:
        sb zero (t1)
remove_newline_end:
        lw ra 0(sp)     # Восстанавливаем адрес возврата из стека
        addi sp sp 4    # Восстанавливаем стек
        jr ra           # Выходим из подпрограммы
```

Рис. 16: При необходимости подпрограммы используют локальные переменные на стеке. Данные из подпрограмм возвращаются в соответствии с общепринятыми соглашениями (через a-регистры)

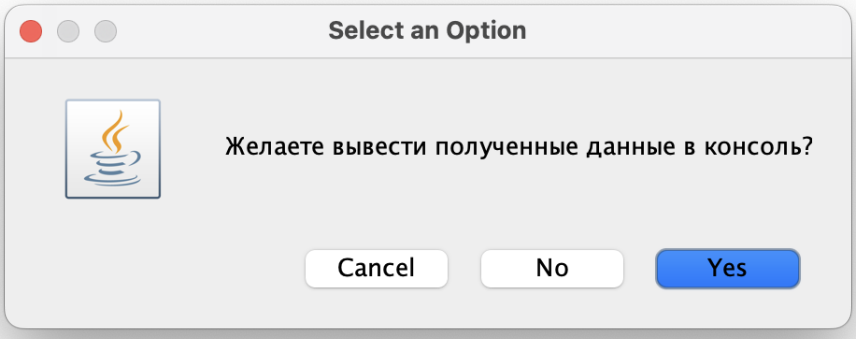

Рис. 17: Пользователь может выбрать, желает ли он вывести данные в консоль

```
2
3  .global number_buffer
4  .global newline
5  .global space
6
7  .data
8          buffer: .space 10240            # Буфер для хранения текста
9          substring_buffer: .space 512    # Буфер для хранения подстр
10         number_buffer: .space 12        # Буфер для хранения числа
11
12 # Массивы названий файлов
12 input_filenames:
14         .asciz "input1.txt\0"
15         .asciz "input2.txt\0"
16         .asciz "input3.txt\0"
17         .asciz "input4.txt\0"
18         .asciz "input5.txt\0"
19
20 output_filenames:
21         .asciz "output1.txt\0"
22         .asciz "output2.txt\0"
23         .asciz "output3.txt\0"
24         .asciz "output4.txt\0"
25         .asciz "output5.txt\0"
26
27 substring_msg: .asciz "Введите искомую подстроку: "
28
29 test_success_msg: .asciz "Тест "
30 test_success_end: .asciz " выполнен успешно!\n"
```

Рис. 18: Реализована дополнительная тестовая программа, осуществляющая автоматическое тестирование по массиву названий входных и выходных файлов

```
# Праметры:
#    address - адрес строки
.macro remove_newline (%address)
        # Сохраняем адрес строки на стеке
        addi sp sp -4
        sw %address 0(sp)

        # Переходим в подпрограмму
        jal ra REMOVE_NEWLINE

        # Восстанавливаем стек
        addi sp sp 4
.end_macro

# Вызов диалогового окна для сбора информации
# Параметры:
#    message - address of null-terminated string that is the message to user
#    buffer - address of input buffer
#    symbols_count - maximum number of characters to read (including the terminating null)
.macro input_dialog (%message, %buffer, %symbols_count)
        la a0 %message          # Передаем адрес сообщения в a0
        la a1 %buffer           # Передаем адрес буфера в a1
        mv a2 %symbols_count
        jal ra INPUT_DIALOG
.end_macro

# Вызов диалогового окна для вывода сообщения
# Параметры:
#    message - address of null-terminated string that is the message to user
#    type - the type of the message to the user
```

Рис. 19: Реализованы как оберточевы макросы (для подпрограмм), так и макросы для вывода данных в консоль

Наконец, программа разбита на несколько единиц компиляции, среди которых присутствует отдельная автономная библиотека макросов. Графические диалоговые окна используются при ручном тестировании программы.