Шабалкин Виталий Витальевич
БПИ-214
Вариант №10

Задача: Разработать программу, которая меняет на обратный порядок следования символов каждого слова в ASCII-строке символов. Порядок слов остается неизменным. Слова состоят только из букв.
Разделителями слов являются все прочие символы.

Код на C:

```
#include <stdio.h>

void flip_words (char str[]){
    int cursor = -1;
    for (int i = 0; i < 1000; i++) {
        if(str[i]>=65 && str[i]<=90 || str[i]>=97 && str[i]<=122){
            if (cursor == -1){
                cursor = i;
            }
        }
        else {
            if (cursor != -1){
                for(int j = 0; j <= (i-cursor-1)/2; j++){
                    char temp;
                    temp = str[cursor + j];
                    str[cursor + j] = str[i-1-j];
                    str[i-1-j] = temp;
                }
                cursor = -1;
            }
        }
    }
    if (cursor != -1){
        for(int i = 0;i<(1000 - cursor -1)/2;i++){
            char temp;
            temp = str[cursor + i];
            str[cursor + i] = str[1000-1-i];
            str[1000-1-i] = temp;
        }
    }
}
int main()
{
    char str[1000];
    fgets(str, 1000, stdin);
    flip_words(str);
    puts(str);
    return 0;
}
```

Код на GAS:

```
        .file   "IDZ-2.c"
        .text
        .globl  flip_words
        .type   flip_words, @function
flip_words:
        endbr64
        pushq  %rbp
        movq   %rsp, %rbp
        movq   %rdi, -40(%rbp)
        movl   $-1, -16(%rbp) #int cursor = -1
        movl   $0, -12(%rbp)  #int i = 0 в первом цикле for
        jmp    .L2
.L10:
        movl   -12(%rbp), %eax
        movslq %eax, %rdx
        movq   -40(%rbp), %rax
        addq   %rdx, %rax
        movzbl (%rax), %eax
        cmpb   $64, %al
        jle    .L3      # Если str[i]<=64, то переход к блоку L3
        movl   -12(%rbp), %eax
        movslq %eax, %rdx
        movq   -40(%rbp), %rax
        addq   %rdx, %rax
        movzbl (%rax), %eax
        cmpb   $90, %al
        jle    .L4      # Если str[i]<=90, то переход к блоку L4
.L3:
        movl   -12(%rbp), %eax
        movslq %eax, %rdx
        movq   -40(%rbp), %rax
        addq   %rdx, %rax
        movzbl (%rax), %eax
        cmpb   $96, %al
        jle    .L5              # Если str[i]<=96, то переход к блоку L5
        movl   -12(%rbp), %eax
        movslq %eax, %rdx
        movq   -40(%rbp), %rax
        addq   %rdx, %rax
        movzbl (%rax), %eax
        cmpb   $122, %al
        jg     .L5              # Если str[i]>122, то переход к блоку L5
.L4:
        cmpl   $-1, -16(%rbp)
        jne    .L7                      # Если cursor != -1, то перейтик блоку L7
```

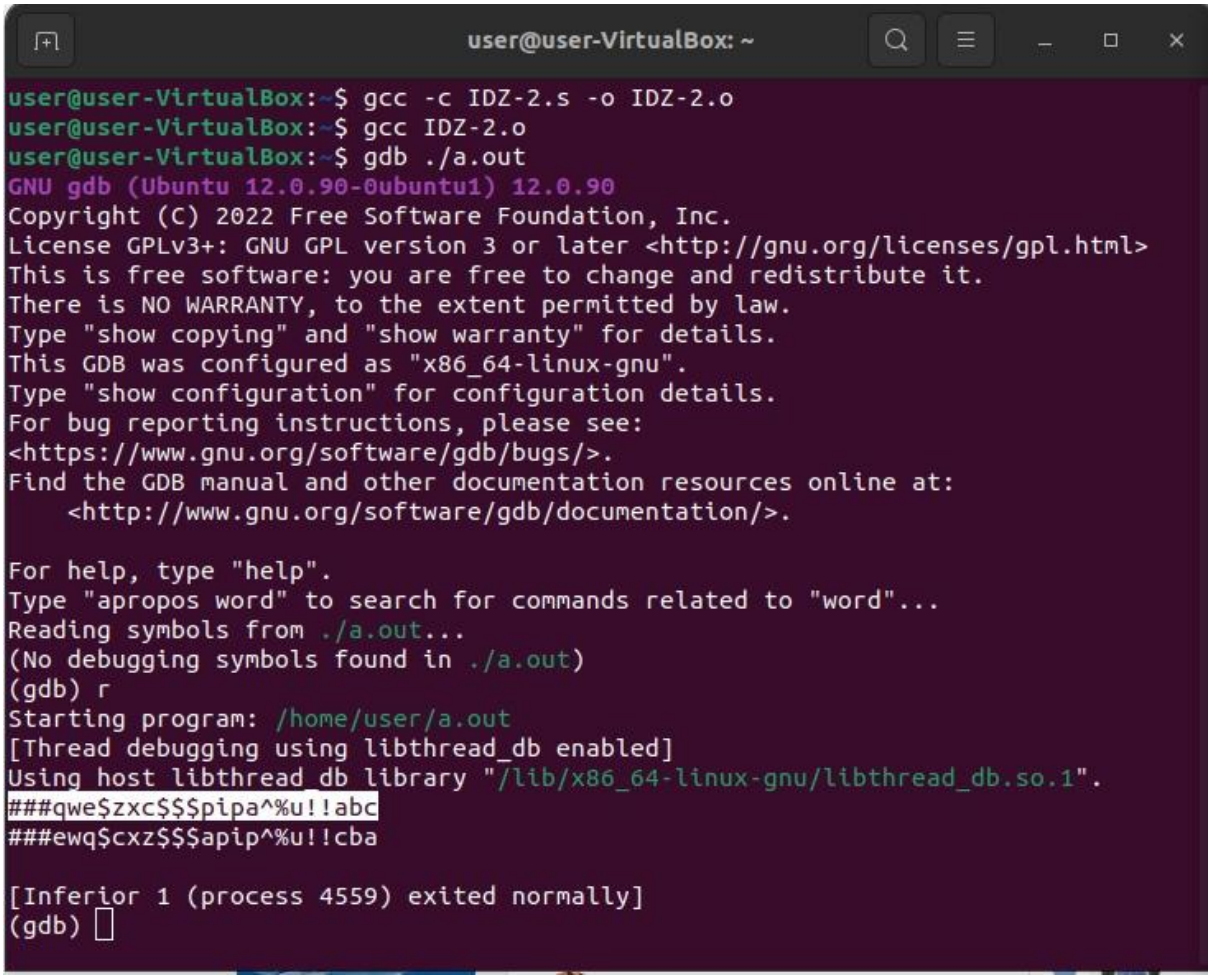
```
        movl    -12(%rbp), %eax
        movl    %eax, -16(%rbp)                 # cursor = i
        jmp     .L7
.L5:
        cmpl    $-1, -16(%rbp)
        je      .L7             # Если cursor == -1, то перейтик блоку L7
        movl    $0, -8(%rbp)    # int j = 0 (вложенный for)
        jmp     .L8             # Переход к блоку L8
.L9:
        movl    -16(%rbp), %edx
        movl    -8(%rbp), %eax
        addl    %edx, %eax
        movslq  %eax, %rdx
        movq    -40(%rbp), %rax
        addq    %rdx, %rax
        movzbl  (%rax), %eax
        movb    %al, -17(%rbp)          # temp = str[cursor + j]
        movl    -12(%rbp), %eax
        subl    $1, %eax
        subl    -8(%rbp), %eax
        movslq  %eax, %rdx
        movq    -40(%rbp), %rax
        addq    %rdx, %rax
        movl    -16(%rbp), %ecx
        movl    -8(%rbp), %edx
        addl    %ecx, %edx
        movslq  %edx, %rcx
        movq    -40(%rbp), %rdx
        addq    %rcx, %rdx
        movzbl  (%rax), %eax
        movb    %al, (%rdx)     # str[cursor + j] = str[i-1-j]
        movl    -12(%rbp), %eax
        subl    $1, %eax
        subl    -8(%rbp), %eax
        movslq  %eax, %rdx
        movq    -40(%rbp), %rax
        addq    %rax, %rdx
        movzbl  -17(%rbp), %eax
        movb    %al, (%rdx)     # str[i-1-j] = temp;
        addl    $1, -8(%rbp)    # j++
.L8:
        movl    -12(%rbp), %eax
        subl    -16(%rbp), %eax
        subl    $1, %eax
        movl    %eax, %edx
        shrl    $31, %edx
        addl    %edx, %eax
        sarl    %eax
```

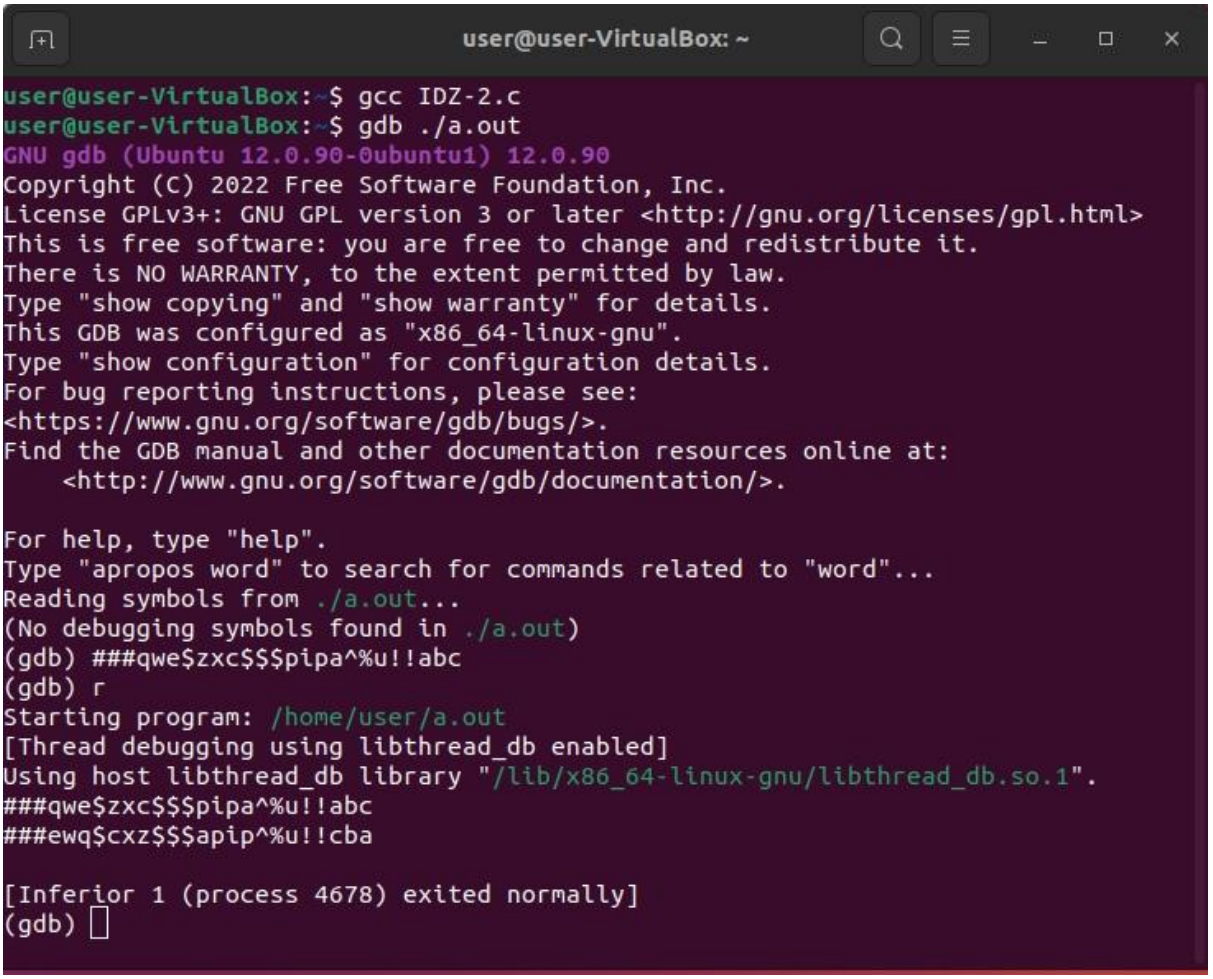
```
        cmpl    %eax, -8(%rbp)
        jle     .L9                     #Если j <= (i-cursor-1)/2, то переход на блок L9
        movl    $-1, -16(%rbp)          #cursor = -1
.L7:
        addl    $1, -12(%rbp)           #i++
.L2:
        cmpl    $999, -12(%rbp)
        jle     .L10                    # Если i (первый for) меньше или равно 999, то
переход к блоку L10
        cmpl    $-1, -16(%rbp)
        je      .L14                    # Если cursor равен -1, то переход к блоку L14
(конец)
        movl    $0, -4(%rbp)            # int i = 0 (второй for)
        jmp     .L12                    # Переход к блоку L12
.L13:
        movl    -16(%rbp), %edx
        movl    -4(%rbp), %eax
        addl    %edx, %eax
        movslq  %eax, %rdx
        movq    -40(%rbp), %rax
        addq    %rdx, %rax
        movzbl  (%rax), %eax
        movb    %al, -18(%rbp)          # temp = str[cursor + i]
        movl    $999, %eax
        subl    -4(%rbp), %eax
        movslq  %eax, %rdx
        movq    -40(%rbp), %rax
        addq    %rdx, %rax
        movl    -16(%rbp), %ecx
        movl    -4(%rbp), %edx
        addl    %ecx, %edx
        movslq  %edx, %rcx
        movq    -40(%rbp), %rdx
        addq    %rcx, %rdx
        movzbl  (%rax), %eax
        movb    %al, (%rdx)     # str[cursor + i] = str[1000-1-i]
        movl    $999, %eax
        subl    -4(%rbp), %eax
        movslq  %eax, %rdx
        movq    -40(%rbp), %rax
        addq    %rax, %rdx
        movzbl  -18(%rbp), %eax
        movb    %al, (%rdx)     # str[1000-1-i] = temp
        addl    $1, -4(%rbp)    #i++
.L12:
        movl    $999, %eax
        subl    -16(%rbp), %eax
        movl    %eax, %edx
```

```
        shrl    $31, %edx
        addl    %edx, %eax
        sarl    %eax
        cmpl    %eax, -4(%rbp)
        jl      .L13            # Если i< 999 - cursor, то переход к блоку L13
.L14:
        nop
        popq    %rbp
        ret


        .size   flip_words, .-flip_words
        .globl  main
        .type   main, @function
main:
        endbr64
        pushq   %rbp
        movq    %rsp, %rbp
        subq    $1008, %rsp   # Создаем массив str на 1000 эл.
        movq    %fs:40, %rax
        movq    %rax, -8(%rbp)
        xorl    %eax, %eax
        movq    stdin(%rip), %rdx       # Передача в качестве аргумента потока stdin
        leaq    -1008(%rbp), %rax       # Передача в качестве аргумента str
        movl    $1000, %esi             # Передача в качестве аргумента 1000
        movq    %rax, %rdi
        call    fgets@PLT               # Вызов функции fgets
        leaq    -1008(%rbp), %rax       # Передача в качестве аргумента строки str функции
flip_words
        movq    %rax, %rdi
        call    flip_words      # Вызов функции flip_words
        leaq    -1008(%rbp), %rax       # Передача в качестве аргумента строки str функции
puts
        movq    %rax, %rdi
        call    puts@PLT                # Вызов функции puts
        movl    $0, %eax
        movq    -8(%rbp), %rdx
        subq    %fs:40, %rdx
        je      .L17
        call    __stack_chk_fail@PLT
.L17:
        leave
        ret
        .size   main, .-main
        .ident  "GCC: (Ubuntu 11.3.0-1ubuntu1~22.04) 11.3.0"
        .section        .note.GNU-stack,"",@progbits
        .section        .note.gnu.property,"a"
        .align 8
```

```
        .long   1f - 0f
        .long   4f - 1f
        .long   5
0:
        .string  "GNU"
1:
        .align 8
        .long   0xc0000002
        .long   3f - 2f
2:
        .long   0x3
3:
        .align 8
4:
```





Были использованы функции и локальные переменные