МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
им. Н.Э. Баумана

Факультет “Информатика и системы управления”
Кафедра “Системы обработки информации и управления”

Дисциплина “Парадигмы и конструкции языков программирования”

Отчет по ДЗ

Выполнил:
Студент группы ИУ5-36Б
Шах А. М.
Преподаватель:
Гапанюк Ю. Е.

Москва 2025

Задание:

1. Выберите язык программирования (который Вы ранее не изучали) и (1) напишите по нему реферат с примерами кода или (2) реализуйте на нем небольшой проект (с детальным текстовым описанием).
2. Реферат (проект) может быть посвящен отдельному аспекту (аспектам) языка или содержать решение какой-либо задачи на этом языке.
3. Необходимо установить на свой компьютер компилятор (интерпретатор, транспилятор) этого языка и произвольную среду разработки.
4. В случае написания реферата необходимо разработать и откомпилировать примеры кода (или модифицировать стандартные примеры).
5. В случае создания проекта необходимо детально комментировать код.
6. При написании реферата (создании проекта) необходимо изучить и корректно использовать особенности парадигмы языка и основных конструкций данного языка.
7. Приветствуется написание черновика статьи по результатам выполнения ДЗ. Черновик статьи может быть подготовлен группой студентов, которые исследовали один и тот же аспект в нескольких языках или решили одинаковую задачу на нескольких языках.

Код программы

client.asm

```
format ELF64
public _start

SYS_READ       = 0
SYS_WRITE      = 1
SYS_CLOSE      = 3
SYS_SOCKET     = 41
SYS_CONNECT    = 42
SYS_EXIT       = 60
AF_INET        = 2
SOCK_STREAM    = 1

section '.data' writeable
   msg_conn    db 'Connecting to server 127.0.0.1:7777...', 10, 0
   msg_exit    db 10, 'Exiting client...', 10, 0
   serv_addr:
      dw AF_INET
      db 0x1E, 0x61
      db 127,0,0,1
      dq 0

   sockfd      dq 0
   recv_buf    rb 2048
   input_char  db 0, 0

section '.text' executable
_start:
   mov rsi, msg_conn
   call print_string

   mov rax, SYS_SOCKET
   mov rdi, AF_INET
   mov rsi, SOCK_STREAM
   xor rdx, rdx
   syscall
   mov [sockfd], rax

   mov rax, SYS_CONNECT
```

```
    mov rdi, [sockfd]
    mov rsi, serv_addr
    mov rdx, 16
    syscall

loop_game:
    mov rax, SYS_READ
    mov rdi, [sockfd]
    mov rsi, recv_buf
    mov rdx, 2047
    syscall

    cmp rax, 0
    jle client_shutdown

    mov rdx, rax
    mov rax, SYS_WRITE
    mov rdi, 1
    mov rsi, recv_buf
    syscall

    mov rax, SYS_READ
    mov rdi, 0
    mov rsi, input_char
    mov rdx, 2
    syscall

    ; ПРОВЕРКА ВЫХОДА
    mov al, [input_char]
    cmp al, 'q'
    je client_shutdown
    cmp al, 'Q'
    je client_shutdown

    mov rax, SYS_WRITE
    mov rdi, [sockfd]
    mov rsi, input_char
    mov rdx, 1
    syscall
    jmp loop_game

client_shutdown:
    mov rsi, msg_exit
    call print_string
    mov rax, SYS_CLOSE
    mov rdi, [sockfd]
    syscall
    mov rax, SYS_EXIT
    xor rdi, rdi
    syscall

print_string:
    push rdi
    push rsi
    xor rax, rax
.strlen_loop:
    cmp byte [rsi + rax], 0
    je .strlen_done
    inc rax
    jmp .strlen_loop
.strlen_done:
    mov rdx, rax
    mov rax, SYS_WRITE
    mov rdi, 1
    syscall
    pop rsi
    pop rdi
    ret
```

server.asm

```
format ELF64
public _start

SYS_READ       = 0
SYS_WRITE      = 1
SYS_CLOSE      = 3
SYS_SOCKET     = 41
SYS_ACCEPT     = 43
SYS_BIND       = 49
SYS_LISTEN     = 50
```

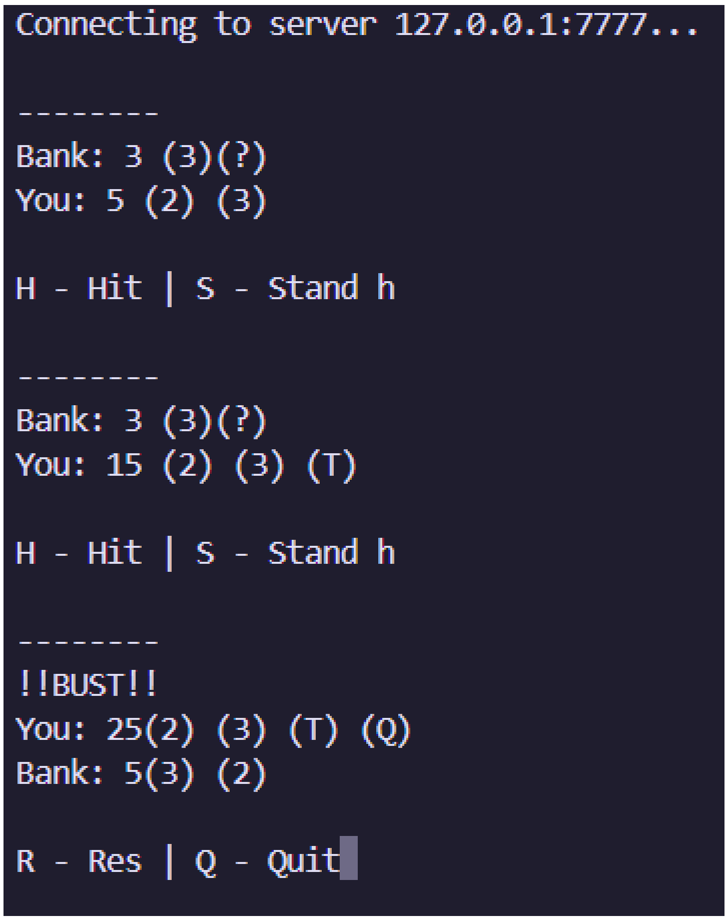
```
SYS_EXIT       = 60

AF_INET        = 2
SOCK_STREAM     = 1
INADDR_ANY      = 0

section '.data' writeable
   msg_start      db '[Game Server] Port 7777 active...', 10, 0
   msg_client     db '[New player joined]', 10, 0
   msg_shutdown    db '[Server shutting down...]', 10, 0

   serv_addr:
      dw AF_INET
      db 0x1E, 0x61      ; Port 7777
      dd INADDR_ANY
      dq 0

   sockfd         dq 0
   clientfd       dq 0
   recv_buf       rb 256
   send_buf       rb 2048

   player_score    dq 0
   player_aces     dq 0
   dealer_score    dq 0
   dealer_aces     dq 0
   dealer_open_val dq 0

   player_hand     rb 20
   player_hand_len dq 0
   dealer_hand     rb 20
   dealer_hand_len dq 0

   deck_index     dq 0
   seed           dq 987654321
   card_names     db '2','3','4','5','6','7','8','9','T','J','Q','K','A'
   card_values    db 2,2,2,2, 3,3,3,3, 4,4,4,4, 5,5,5,5, 6,6,6,6, 7,7,7,7, 8,8,8,8,
9,9,9,9
              db 10,10,10,10, 10,10,10,10, 10,10,10,10, 10,10,10,10, 11,11,11,11
   deck           db 52 dup(0)
```

```
section '.text' executable
_start:
    xor rax, rax
.init_deck:
    mov [deck + rax], al
    inc rax
    cmp rax, 52
    jne .init_deck

    mov rax, SYS_SOCKET
    mov rdi, AF_INET
    mov rsi, SOCK_STREAM
    xor rdx, rdx
    syscall
    mov [sockfd], rax

    mov rax, SYS_BIND
    mov rdi, [sockfd]
    mov rsi, serv_addr
    mov rdx, 16
    syscall

    mov rax, SYS_LISTEN
    mov rdi, [sockfd]
    mov rsi, 1
    syscall

    mov rsi, msg_start
    call print_string

accept_loop:
    mov rax, SYS_ACCEPT
    mov rdi, [sockfd]
    xor rsi, rsi
    xor rdx, rdx
    syscall
    mov [clientfd], rax

    mov rsi, msg_client
    call print_string

new_game_start:
    mov qword [player_score], 0
    mov qword [player_aces], 0
    mov qword [dealer_score], 0
    mov qword [dealer_aces], 0
    mov qword [deck_index], 0
    mov qword [player_hand_len], 0
    mov qword [dealer_hand_len], 0
    call shuffle_deck

    mov rdi, 0
    call give_card_smart
    mov rdi, 0
    call give_card_smart
    mov rdi, 1
    call give_card_smart
    mov rax, [dealer_score]
    mov [dealer_open_val], rax
    mov rdi, 1
    call give_card_smart

    cmp qword [player_score], 21
    je check_dealer_bj
    call send_status_hidden

game_loop:
    mov rax, SYS_READ
    mov rdi, [clientfd]
    mov rsi, recv_buf
    mov rdx, 255
    syscall
    cmp rax, 0
    jle close_client

    mov al, byte [recv_buf]
    cmp al, 'h'
    je hit_me
    cmp al, 'H'
    je hit_me
    cmp al, 's'
    je stand
    cmp al, 'S'
    je stand
    cmp al, 'r'
    je new_game_start
    cmp al, 'R'
    je new_game_start
    cmp al, 'q'
    je server_shutdown
    cmp al, 'Q'
    je server_shutdown
    jmp server_shutdown

hit_me:
    mov rdi, 0
    call give_card_smart
    cmp qword [player_score], 21
    jg player_bust
    call send_status_hidden
    jmp game_loop

check_dealer_bj:
    cmp qword [dealer_score], 21
    je result_draw_bj
    jmp result_win_bj

stand:
.dealer_turn:
    cmp qword [dealer_score], 17
    jge .dealer_done
    mov rdi, 1
    call give_card_smart
    jmp .dealer_turn
.dealer_done:
    mov rax, [dealer_score]
    cmp rax, 21
    jg result_win_dealer_bust
    mov rbx, [player_score]
    cmp rbx, rax
    jg result_win
    jl result_lose
    je result_draw



player_bust:
    mov rdi, send_buf
    call draw_header
    call w_bust
    jmp send_final
result_win_bj:
    mov rdi, send_buf
    call draw_header
    call w_bj
    jmp send_final
result_draw_bj:
    mov rdi, send_buf
    call draw_header
    call w_push
    jmp send_final
result_win:
    mov rdi, send_buf
    call draw_header
    call w_win
    jmp send_final
result_win_dealer_bust:
    mov rdi, send_buf
    call draw_header
    call w_win_db
    jmp send_final
result_lose:
    mov rdi, send_buf
    call draw_header
    call w_lose
    jmp send_final
result_draw:
    mov rdi, send_buf
    call draw_header
    call w_push
    jmp send_final

send_final:
    mov rsi, send_buf
    call send_string
    jmp restart_loop

restart_loop:
    mov rax, SYS_READ
    mov rdi, [clientfd]
    mov rsi, recv_buf
    mov rdx, 255
    syscall
    cmp rax, 0
    jle close_client
    mov al, byte [recv_buf]
    cmp al, 'r'
    je new_game_start
    cmp al, 'R'
    je new_game_start
    cmp al, 'q'
    je server_shutdown
    cmp al, 'Q'
    je server_shutdown
    jmp restart_loop

server_shutdown:
    mov rsi, msg_shutdown
    call send_string
    mov rax, SYS_CLOSE
    mov rdi, [clientfd]
    syscall
    mov rax, SYS_CLOSE
    mov rdi, [sockfd]
    syscall
    mov rax, SYS_EXIT
    xor rdi, rdi
    syscall
```

```
close_client:
    mov rax, SYS_CLOSE
    mov rdi, [clientfd]
    syscall
    jmp accept_loop

give_card_smart:
    push rbx
    push rcx
    push rdx
    push rdi
    mov rbx, [deck_index]
    movzx rax, byte [deck + rbx]
    inc qword [deck_index]
    push rax
    xor rdx, rdx
    mov rcx, 4
    div rcx
    mov bl, [card_names + rax]
    pop rax
    push rax
    movzx rcx, byte [card_values + rax]
    mov rdi, [rsp + 8]
    cmp rdi, 0
    je .p_save
    mov rdx, [dealer_hand_len]
    mov [dealer_hand + rdx], bl
    inc qword [dealer_hand_len]
    add [dealer_score], rcx
    cmp rcx, 11
    jne .d_check
    inc qword [dealer_aces]
.d_check:
    cmp qword [dealer_score], 21
    jle .done
    cmp qword [dealer_aces], 0
    je .done
    sub qword [dealer_score], 10
    dec qword [dealer_aces]
    jmp .d_check
```

```
.p_save:
    mov rdx, [player_hand_len]
    mov [player_hand + rdx], bl
    inc qword [player_hand_len]
    add [player_score], rcx
    cmp rcx, 11
    jne .p_check
    inc qword [player_aces]
.p_check:
    cmp qword [player_score], 21
    jle .done
    cmp qword [player_aces], 0
    je .done
    sub qword [player_score], 10
    dec qword [player_aces]
    jmp .p_check
.done:
    pop rax
    pop rdi
    pop rdx
    pop rcx
    pop rbx
    ret

draw_header:
    mov byte [rdi], 10
    mov dword [rdi+1], '----'
    mov dword [rdi+5], '----'
    mov byte [rdi+9], 10
    add rdi, 10
    ret

draw_cards:
    push rbx
    push rcx
    push rsi
    xor rbx, rbx
.l:
    cmp rbx, rcx
    jge .e
    mov byte [rdi], '('
    mov al, [rsi + rbx]
    mov [rdi+1], al
    mov word [rdi+2], ') '
    add rdi, 4
    inc rbx
    jmp .l
.e:
    pop rsi
    pop rcx
    pop rbx
    ret

send_status_hidden:
    mov rdi, send_buf
    call draw_header
    mov dword [rdi], 'Bank'
    mov word [rdi+4], ': '
    add rdi, 6
    mov rax, [dealer_open_val]
    call int_to_buf
    mov byte [rdi], ' '
    mov byte [rdi+1], '('
    mov al, [dealer_hand]
    mov [rdi+2], al
    mov dword [rdi+3], ')(?)'
    add rdi, 7
    mov byte [rdi], 10
    inc rdi
    mov dword [rdi], 'You:'
    mov byte [rdi+4], ' '
    add rdi, 5
    mov rax, [player_score]
    call int_to_buf
    mov byte [rdi], ' '
    inc rdi
    mov rsi, player_hand
    mov rcx, [player_hand_len]
    call draw_cards
    mov word [rdi], 0x0A0A
    add rdi, 2
    mov dword [rdi], 'H - '
    mov dword [rdi+4], 'Hit '
    mov dword [rdi+8], '| S '
    mov dword [rdi+12], '- St'
    mov dword [rdi+16], 'and '
    add rdi, 20
    mov byte [rdi], 0
    mov rsi, send_buf
    call send_string
    ret



append_full_stats:
    mov byte [rdi], 10
    inc rdi
    mov dword [rdi], 'You:'
    mov byte [rdi+4], ' '
    add rdi, 5
    mov rax, [player_score]
    call int_to_buf
    mov rsi, player_hand
    mov rcx, [player_hand_len]
    call draw_cards
    mov byte [rdi], 10
    inc rdi
    mov dword [rdi], 'Bank'
    mov word [rdi+4], ': '
    add rdi, 6
    mov rax, [dealer_score]
    call int_to_buf
    mov rsi, dealer_hand
    mov rcx, [dealer_hand_len]
    call draw_cards
    mov word [rdi], 0x0A0A
    mov dword [rdi+2], 'R - '
    mov dword [rdi+6], 'Res '
    mov dword [rdi+10], '| Q '
    mov dword [rdi+14], '- Qu'
    mov word [rdi+18], 'it'
    add rdi, 20
    mov byte [rdi], 0
    ret

w_bj:
    mov dword [rdi], 'BLAC'
    mov dword [rdi+4], 'KJAC'
    mov word [rdi+8], 'K!'
    add rdi, 10
    jmp append_full_stats
w_bust:
    mov dword [rdi], '!!BU'
    mov dword [rdi+4], 'ST!!'
    add rdi, 8
    jmp append_full_stats
w_win:
    mov dword [rdi], 'YOU '
    mov dword [rdi+4], 'WIN!'
    add rdi, 8
    jmp append_full_stats
w_win_db:
    mov dword [rdi], 'DEAL'
    mov dword [rdi+4], 'ER B'
    mov dword [rdi+8], 'UST!'
    add rdi, 12
    jmp append_full_stats
w_lose:
    mov dword [rdi], 'YOU '
    mov dword [rdi+4], 'LOSE'
    add rdi, 8
    jmp append_full_stats
w_push:
    mov dword [rdi], 'PUSH'
    add rdi, 4
    jmp append_full_stats

int_to_buf:
    push rbx
    push rdx
    mov rbx, 10
    xor rdx, rdx
    div rbx
    test al, al
    jz .s
    add al, '0'
    mov [rdi], al
    inc rdi
.s:
    add dl, '0'
    mov [rdi], dl
    inc rdi
    pop rdx
    pop rbx
    ret

shuffle_deck:
    mov rcx, 51
.sh:
    push rcx
    call rand
    xor rdx, rdx
    mov rbx, 52
    div rbx
    pop rcx
    mov al, [deck + rcx]
    mov ah, [deck + rdx]
    mov [deck + rcx], ah
    mov [deck + rdx], al
    loop .sh
    ret

rand:
    push rbx
    push rcx
    mov rax, [seed]
    mov rbx, 6364136223846793005
    mul rbx
    mov rcx, 1442695040888963407
    add rax, rcx
    mov [seed], rax
    pop rcx
    pop rbx
    ret

send_string:
    push rdi
    mov rdi, rsi
    call strlen
    mov rdx, rax
    mov rax, SYS_WRITE
    mov rdi, [clientfd]
    syscall
    pop rdi
    ret

print_string:
    push rdi
    mov rdi, rsi
    call strlen
    mov rdx, rax
    mov rax, SYS_WRITE
    mov rdi, 1
    syscall
    pop rdi
    ret

strlen:
    xor rax, rax
.L:
    cmp byte [rdi + rax], 0
    je .D
    inc rax
    jmp .L
.D:
    ret
```

Результат



```
Connecting to server 127.0.0.1:7777...

--------
Bank: 3 (3)(?)
You: 5 (2) (3)

H - Hit | S - Stand h

--------
Bank: 3 (3)(?)
You: 15 (2) (3) (T)

H - Hit | S - Stand h

--------
!!BUST!!
You: 25(2) (3) (T) (Q)
Bank: 5(3) (2)

R - Res | Q - Quit
```