♦ ♥ ♣ ♠

TEXT-BASED POKER

Grupo 5: Leonardo, Murilo e Rodrigo.

REQUISITOS

- O programa receberá do usuário duas mãos formadas por 5 cartas cada;
- O programa deverá exibir a melhor mão de cada jogador e indicar o vencedor;
- Aplicar critérios de desempate.

Exemplo de entrada:

5c 5e 7p 2c 3o 8p 8o 8e 10c 4e

Saída:

Jogador 1: PAR
Jogador 2: TRINCA
Vencedor: Jogador 2

ABORDAGENS

ALGORITMO DE PONTUAÇÃO ABSOLUTA

Envolve criar e manipular padrões de bits, aplicar uma função *hash* perfeito (sem colisões), multiplicar primos específicos e checar tabelas de consultas (*look up tables*).

ALGORITMO DE PONTUAÇÃO RELATIVA

Envolve comparar duas mãos a partir da força da combinação obtida, seguida dos critérios de desempate, quando houver.

CRITÉRIOS DE COMPARAÇÃO

ROYAL FLUSH

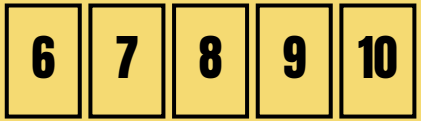
- Não há desempate.

QUADRA

- Maior quadra
- Maior carta alta

FULL HOUSE

- Maior Trinca
- Maior Par

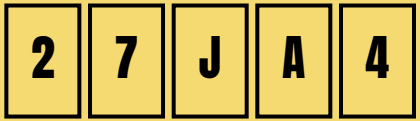


STRAIGHT FLUSH

- Maior carta alta

STRAIGHT

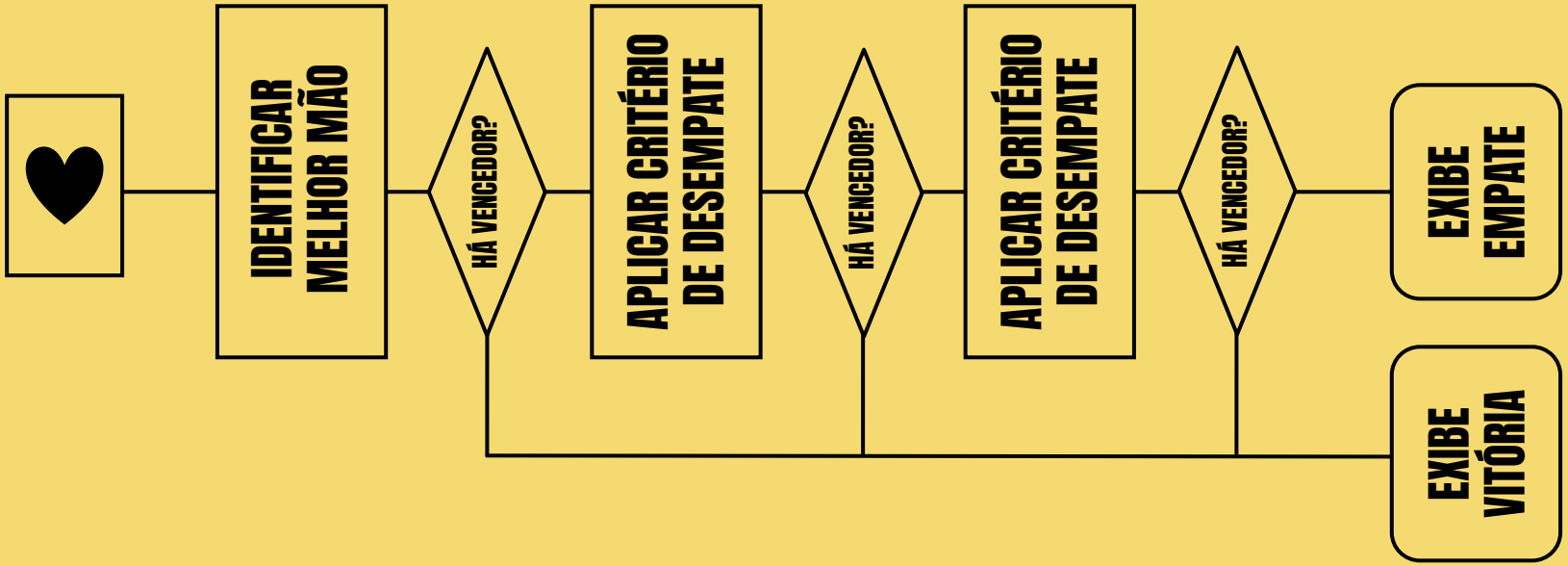
- Maior carta alta



FLUSH

- Maior carta alta

FLUXO DO PROGRAMA



PLANEJAMENTO

ESCOPO E REQUISITOS	Confirmar requisitos e resolver possíveis ambiguidades com o monitor;
ABORDAGEM	Analisar as possíveis soluções para o problema e escolher a mais adequada;
PROTÓTIPO EM C	Implementar algoritmos em C a fim de atestar corretude e refinar lógica;
IMPLEMENTAÇÃO EM ASSEMBLY	Implementar de fato o programa em Assembly;
TESTES E REFATORAÇÃO	Submeter programa a testes e refinar código (lógica, legibilidade, etc);
INTERFACE GRÁFICA	Desenvolver interface gráfica para entrada e saída de dados.

```c
int checar_fullhouse(mao* jogador){
    insertion(jogador); //Após ordenar as cartas por rank, existem apenas 2
    possibilidades de full house, xxxyy ou xxyyy, basta testar.

    if(jogador->carta[0].rank == jogador->carta[1].rank && jogador->carta[1].
    rank == jogador->carta[2].rank && jogador->carta[3].rank == jogador->carta
    [4].rank){
        printf("full house");
        jogador->score = 7;
        jogador->maior_comb = jogador->carta[1].rank;
        return 1;
    }
    if(jogador->carta[0].rank == jogador->carta[1].rank && jogador->carta[2].
    rank == jogador->carta[3].rank && jogador->carta[3].rank == jogador->carta
    [4].rank){
        printf("full house");
        jogador->score = 7;
        jogador->maior_comb = jogador->carta[3].rank;
        return 1;
    }
    return 0;
}

//A partir daqui o raciocínio é muito parecido com a checagem da quadra e do
fullhouse
//Na hora de fazer os testes, a ordem das funções é importante. Por exemplo,
uma trinca pode ser uma quadra, mas como a função checar_quadra vem antes na
ordem de execução, não vai ter esse problema

int checar_trinca(mao* jogador){ //3 trincas possíveis, xxxyz yxxxz yzxxx
    insertion(jogador);

    if(jogador->carta[0].rank == jogador->carta[1].rank && jogador->carta[1].
    rank == jogador->carta[2].rank){
        printf("trinca");
        jogador->score = 4;
        jogador->maior_comb = jogador->carta[1].rank;
        return 1;
    }
```



```c
int checar_trinca(mao* jogador){ //3 trincas possíveis, xxxyz yxxxz yzxxx
    insertion(jogador);

    if(jogador->carta[0].rank == jogador->carta[1].rank && jogador->carta[1].
    rank == jogador->carta[2].rank){
        printf("trinca");
        jogador->score = 4;
        jogador->maior_comb = jogador->carta[1].rank;
        return 1;
    }
    if(jogador->carta[1].rank == jogador->carta[2].rank && jogador->carta[2].
    rank == jogador->carta[3].rank){
        printf("trinca");
        jogador->score = 4;
        jogador->maior_comb = jogador->carta[1].rank;
        return 1;
    }
    if(jogador->carta[2].rank == jogador->carta[3].rank && jogador->carta[3].
    rank == jogador->carta[4].rank){
        printf("trinca");
        jogador->score = 4;
        jogador->maior_comb = jogador->carta[2].rank;
        return 1;
    }
    return 0;
}

int checar_doispares(mao* jogador){
    insertion(jogador);

    if(jogador->carta[0].rank == jogador->carta[1].rank && jogador->carta[2].
    rank == jogador->carta[3].rank){
        printf("dois pares");
        jogador->score = 3;
        //Aqui o maior_comb tem que assumir o valor de algum elemento do
        último par, já que a mão está ordenada por rank
        jogador->maior_comb = jogador->carta[2].rank;
        return 1;
    }
```

Edit | Execute

poker.asm

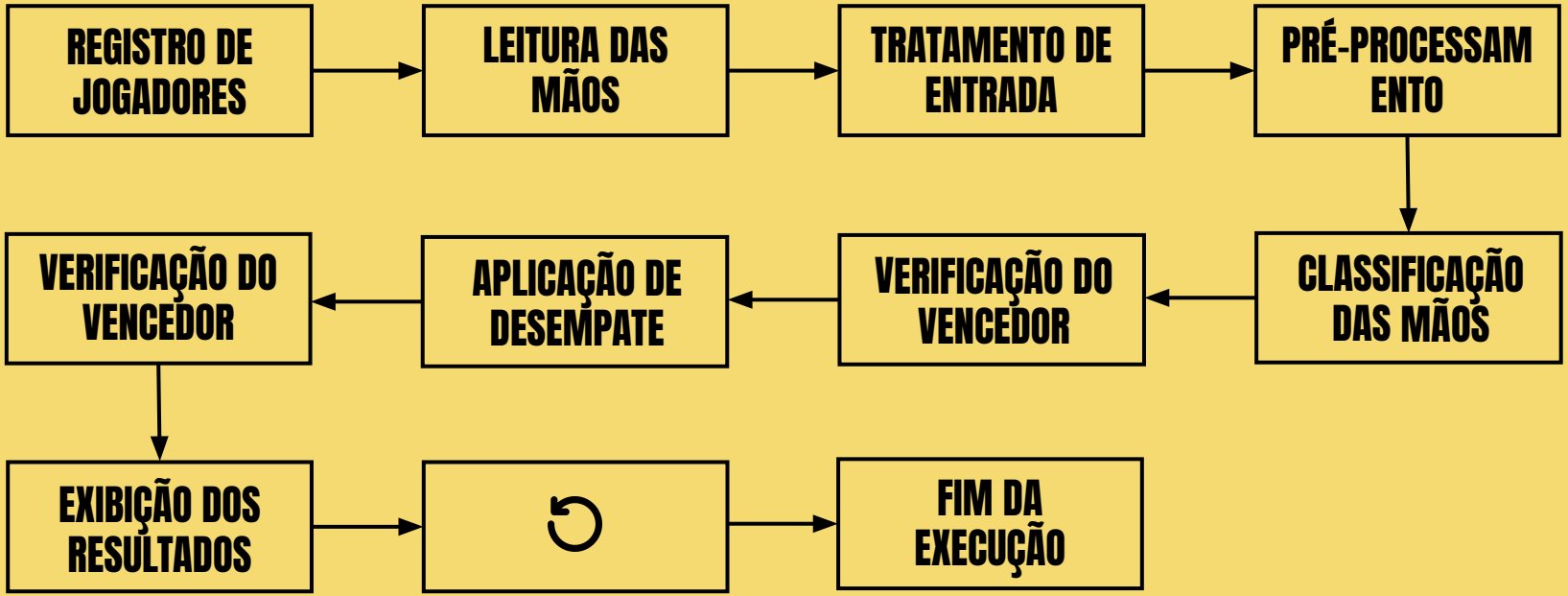
```
.text
        # ———————————————————— Apresentação ———————————————————— #

        la $a0, msg_start                   # Carrega endereço do label em $a0
        li $v0, 4                           # Indica exibição de string em $a0
        syscall                             # Executa operação indicada por $v0

        # ——————————————— Leitura e armazenamento das cartas ——————————————— #

        sw $zero, p1_force                  # Inicializa a força da mão do jogador com zero
        sw $zero, p2_force                  # Inicializa a força da mão do jogador com zero

        li $s7, 1                           # Inicializa o contador de jogadores $s7
        la $s0, p1_name                     # Carrega endereço do label em $s0
        la $s1, p1_values                   # Carrega endereço do label em $s0
        la $s2, p1_suits                    # Carrega endereço do label em $s0

        new_player:

        la $a0, msg_name                    # Carrega endereço do label em $a0
        li $v0, 4                           # Indica exibição de string em $a0
        syscall                             # Executa operação indicada por $v0

        move $a0, $s7                       # Copia conteúdo de $t1 para $a0
        li $v0, 1                           # Indica exibição de um inteiro em a0
        syscall                             # Executa operação indicada por $v0

        la $a0, msg_colons                  # Carrega endereço do label em $a0
        li $v0, 4                           # Indica exibição de string em $a0
        syscall                             # Executa operação indicada por $v0

        la $a0, ($s0)                       # Carrega endereço do label em $a0
        li $a1, 20                          # Carrega tamanho da string em $a1
        li $v0, 8                           # Indica leitura de string em $a0
```

Line: 1 Column: 1 ☑ Show Line Numbers

ages | Run I/O

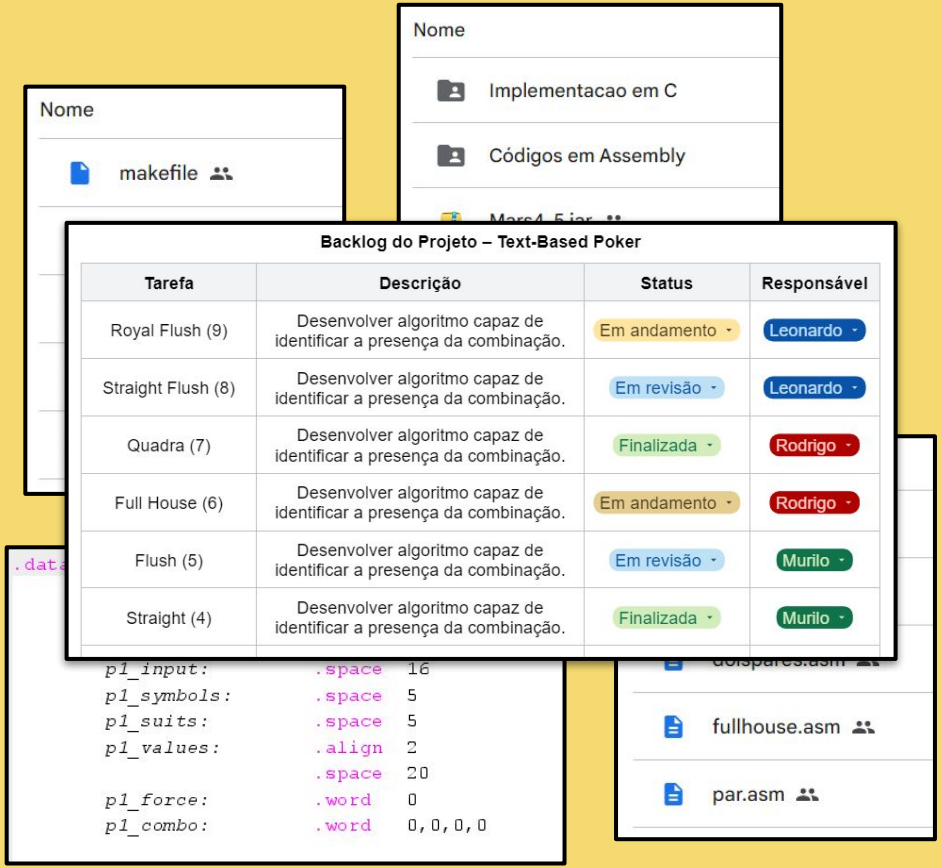
```
---------------------- INÍCIO DE JOGO ----------------------
Insira o nome do jogador 1: Rodrigo.
Seja bem-vindo, Rodrigo.
------------------------------------------------------------
Por favor, insira o valor (1 a 10) da carta 1: 2
Insira o naipe (paus, ouros, espadas ou copas) da carta 1: copas
Por favor, insira o valor (1 a 10) da carta 2: 5
Insira o naipe (paus, ouros, espadas ou copas) da carta 2: copas
Por favor, insira o valor (1 a 10) da carta 3: 10
Insira o naipe (paus, ouros, espadas ou copas) da carta 3: copas
Por favor, insira o valor (1 a 10) da carta 4: 7
Insira o naipe (paus, ouros, espadas ou copas) da carta 4: copas
Por favor, insira o valor (1 a 10) da carta 5: 3
Insira o naipe (paus, ouros, espadas ou copas) da carta 5: copas
------------------------------------------------------------
Carta 1: 2 de copas
Carta 2: 5 de copas
Carta 3: 10 de copas
Carta 4: 7 de copas
Carta 5: 3 de copas
------------------------------------------------------------
Estas são as suas cartas. Digite 1 para confirmar ou 0 para recomeçar: 1
------------------------------------------------------------
Insira o nome do jogador 2: Leonardo.
Seja bem-vindo, Leonardo.
------------------------------------------------------------
Por favor, insira o valor (1 a 10) da carta 1: 4
Insira o naipe (paus, ouros, espadas ou copas) da carta 1: espadas
Por favor, insira o valor (1 a 10) da carta 2: 5
Insira o naipe (paus, ouros, espadas ou copas) da carta 2: paus
Por favor, insira o valor (1 a 10) da carta 3: 6
Insira o naipe (paus, ouros, espadas ou copas) da carta 3: copas
Por favor, insira o valor (1 a 10) da carta 4: 7
Insira o naipe (paus, ouros, espadas ou copas) da carta 4: ouros
Por favor, insira o valor (1 a 10) da carta 5: 8
Insira o naipe (paus, ouros, espadas ou copas) da carta 5: espadas
------------------------------------------------------------
Carta 1: 4 de espadas
Carta 2: 5 de paus
Carta 3: 6 de copas
Carta 4: 7 de ouros
Carta 5: 8 de espadas
------------------------------------------------------------
Estas são as suas cartas. Digite 1 para confirmar ou 0 para recomeçar: 1
------------------------------------------------------------
Com essa mão você conseguiu formar um FLUSH, Rodrigo.
Já você conseguiu formar uma SEQUÊNCIA, Leonardo.
Por isso o vencedor é o jogador 1, com uma FLUSH. Parabéns!
------------------------- FIM DE JOGO -------------------------
```

FLUXO DO PROGRAMA



DESENVOLVIMENTO

- Mini curso de Assembly (youtube);
- Definir a estruturação dos dados;
- Determinar entradas e saídas para cada etapa;
- Criação de um Backlog;
- Codificação para uma mão;
- Testes isolados e de integração;
- Refatoração para programa principal;
- Testes finais, documentação e perfumaria.



♦ ♥ ♣ ♠

TEXT-BASED POKER

Grupo 5: Leonardo, Murilo e Rodrigo.