Centro Federal de Educação Tecnológica de Minas Gerais

Turma: Eng. da Computação **Prof.:** Bruno

Grupo: Rúbia Alice Moreira de Souza

Trabalho Prático 2: Jogo da Adivinhação

Belo Horizonte

2021

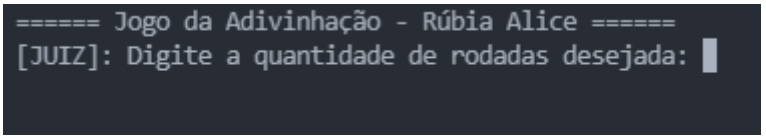
Sumário

1. Introdução

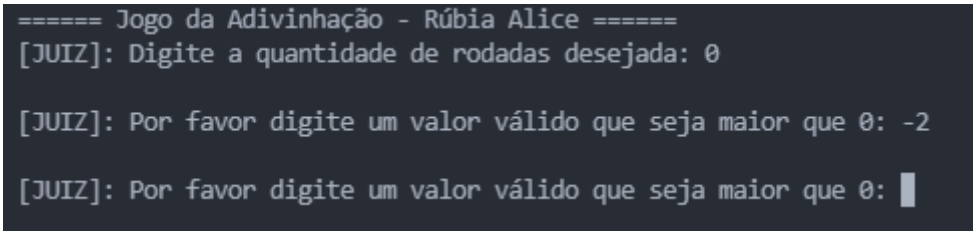
Este relatório explica a execução do programa do Trabalho Prático 2 do Curso de Engenharia da computação do CEFET-MG e o código fonte do projeto está disponível no meu github: Rubia-Souza, mas apenas será tornado público após a data de entrega do trabalho.

2. Fluxo de Execução de um turno

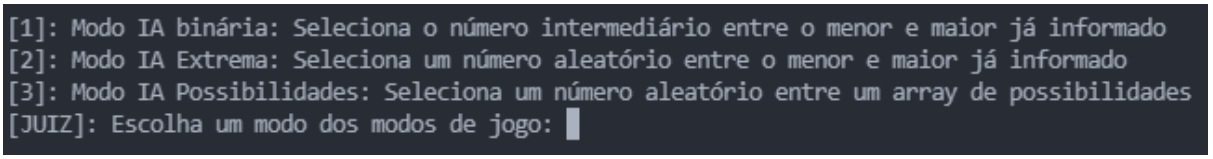
Inicialmente, o programa pede para o jogador a quantidade de turnos que ele deseja jogar:



Ao informar uma quantidade inválida de turnos, ou seja, um valor menor ou igual a zero o sistema pede para o usuário inserir um valor válido:



Ao receber um valor válido, o sistema pergunta ao usuário qual modo de IA ele deseja enfrentar, dentre os 3 modos de chute implementados para ela.



Ao selecionar um valor válido, o turno 1 é iniciado.

2.1. Fluxo Adivinhação do Jogador

No começo do turno do jogador, o sistema gera um número aleatório entre os valores máximo e mínimo das constantes aceitas pelo jogo e pede para que o usuário adivinhe o número gerado:

```
<<<< Iniciando o turno 1 >>>>
[JUIZ]: Iniciando vez do usuário
[IA]: Qual o meu número secreto? <_<
[IA]: >.>
[JUIZ]: Digite um número entre 0 e 20:
```

O número aleatório é criado com base na função rand() e é definido entre uma faixa de um valor mínimo e um máximo:

```
7   /*
8    * Implementa a função para gerar números aletários dentro de um faixa
9    */
10  int createRandomNumber(const int baseValue, const int maxValue) {
11      return (rand() % (maxValue - baseValue + 1)) + baseValue;
12  }
13
```

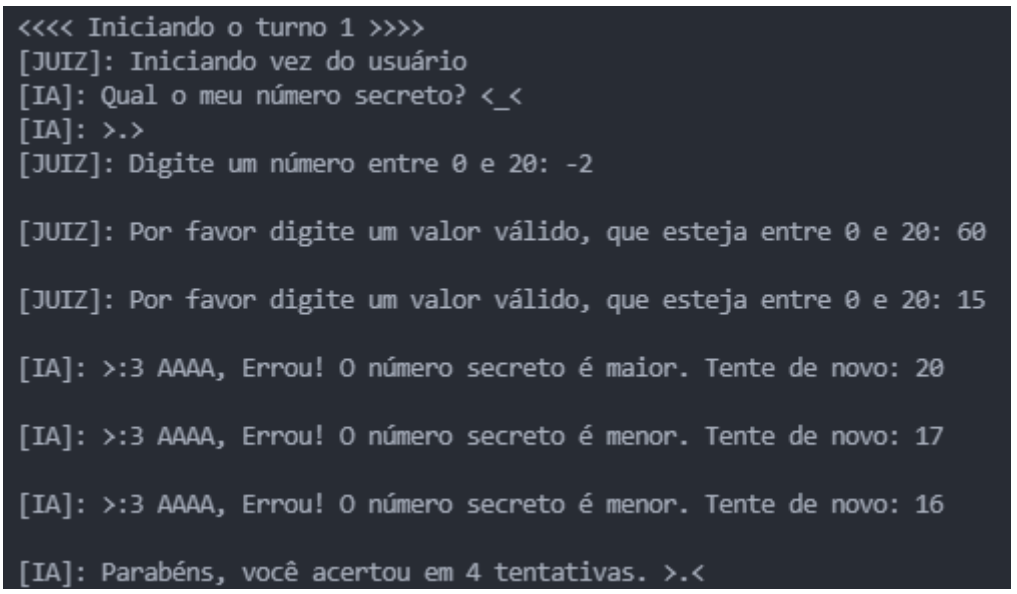
Caso o usuário informe um valor maior ou menor que o número máximo e mínimo, respectivamente, o sistema irá pedir para que ele informe um valor válido.

```
<<<< Iniciando o turno 1 >>>>
[JUIZ]: Iniciando vez do usuário
[IA]: Qual o meu número secreto? <_<
[IA]: >.>
[JUIZ]: Digite um número entre 0 e 20: -2

[JUIZ]: Por favor digite um valor válido, que esteja entre 0 e 20: 60

[JUIZ]: Por favor digite um valor válido, que esteja entre 0 e 20:
```

Quando o usuário informar um valor válido, o sistema irá compará-lo com o número gerado. Caso o valor passado seja maior ou menor que o valor criado pelo programa, ele irá informar o usuário sobre isso:

```
<<<< Iniciando o turno 1 >>>>
[JUIZ]: Iniciando vez do usuário
[IA]: Qual o meu número secreto? <_<
[IA]: >.>
[JUIZ]: Digite um número entre 0 e 20: -2

[JUIZ]: Por favor digite um valor válido, que esteja entre 0 e 20: 60

[JUIZ]: Por favor digite um valor válido, que esteja entre 0 e 20: 15

[IA]: >:3 AAAA, Errou! O número secreto é maior. Tente de novo: 20
```

Esse processo irá se repetir até que o usuário acerte o número aleatório. Então, ao acertar o número, o sistema informa o jogador da quantidade de tentativas que ele teve até acertar o número:

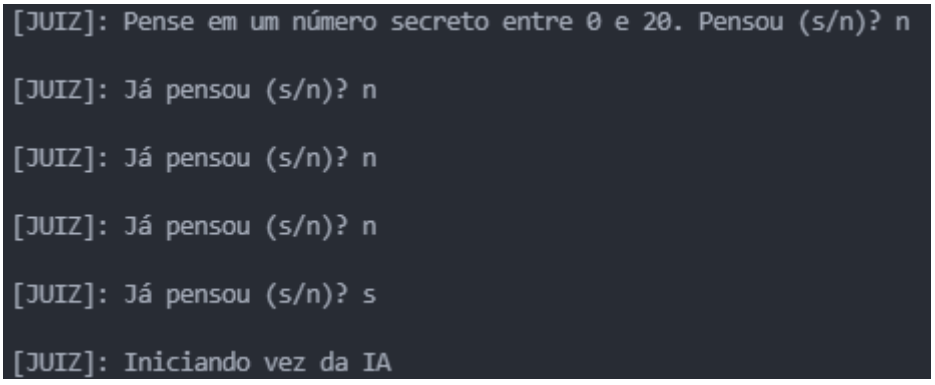


Por fim, o turno do jogador é encerrado.

2.2. Fluxo Adivinhação do Computador

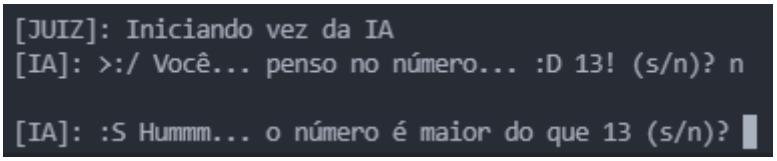
Posteriormente, o turno de adivinhação do computador é iniciado, começando ao perguntar para o jogador se ele pensou em um número entre o mínimo e o máximo.

Enquanto o usuário responder “não”, o sistema continuará a questioná-lo até que responda “sim”:



Depois, o sistema gera um número aleatório qualquer entre o máximo e o mínimo e questiona o jogador se esse número gerado foi o que ele pensou.

Caso o usuário informe que esse não é o número, então o sistema pergunta se o número pensado é maior que o gerado:



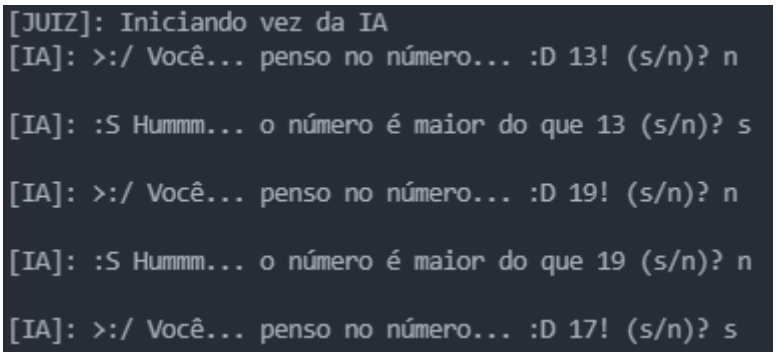
Caso o número pensado pelo usuário seja maior que o gerado pela máquina, o sistema apaga todos os números menores ou iguais ao número gerado do array de possibilidades:

```
bool isValueBigger = userAwnser == 's';
if (isValueBigger) { // Verifica se o número pensado pelo jogador é maior que o chute anterior.
    cutLeftPartArray(availableNumbers, lastValue, &filteredNumbersSize, filteredNumbers); // Se sim, ele remove o
}
```

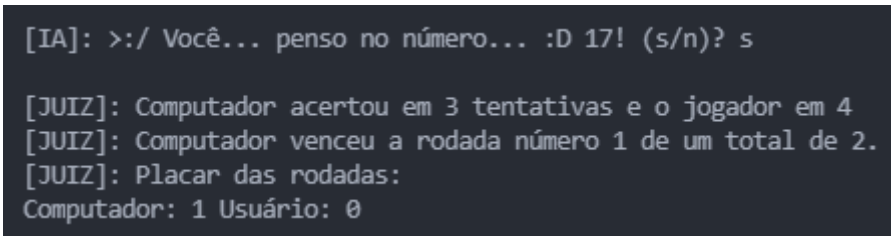
Caso contrário, o sistema apagar todos os números maiores ou iguais ao número gerado do array de possibilidades:

```
else { // Verifica se o número pensado pelo jogador é menor que o chute anterior.
    cutRightPartArray(availableNumbers, lastValue, &filteredNumbersSize, filteredNumbers); // Se sim, ele remove o
}
```

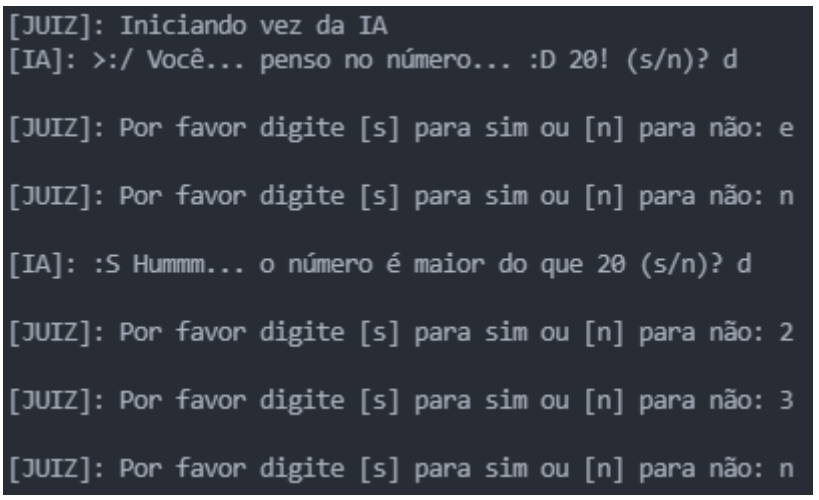
Esse processo se repete até que o computador acerte o número pensado pelo usuário:



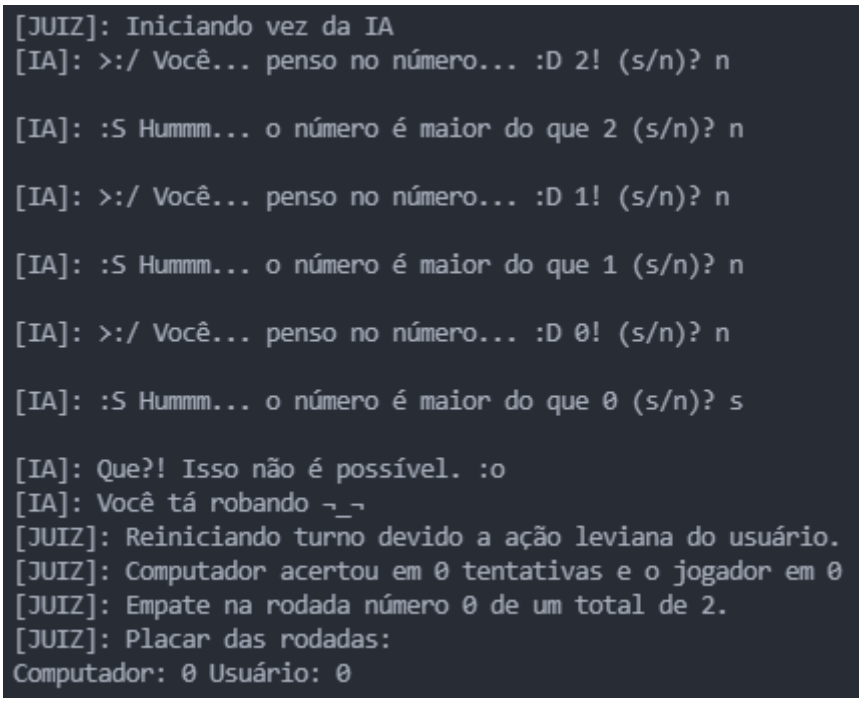
Ao acertar o número pensado pelo usuário, o sistema informa a quantidade de tentativas que cada um teve nesse turno para acertar e informa o vencedor do turno atual:



Caso o usuário não informe “sim” ou “não” para a pergunta do número chutado pela máquina, o sistema continua a pedi-lo um dos dois valores válidos. O mesmo ocorre quando o usuário informa um valor inválido para a pergunta do número ser maior ou não:



Por fim, se o usuário responder de forma inconsistente se o número é maior ou não e o computador chegar a eliminar todos os números possíveis, como no exemplo abaixo, o sistema reinicia o turno, por considerar que o jogador tentou roubar nele:



Observação: Este é o fluxo para o modo 3. Todos seguem o mesmo padrão, o que muda é a lógica de chute.

No modo 1: O computador acompanhará apenas o menor e o maior número já informado até o momento. Então, ele sempre irá supor um número que é igual a metade da soma entre os dois valores. Ou seja, um número entre eles:

```c
int guessOtherValueBasedOnBinaryDivision(const int lastValue, int *highestValue, int *lowestValue) {
    bool isValueBigger = checkWithUserIfNumberIsHigher(lastValue);

    // Verifica se os valores máximo e mínimo chegaram ao mesmo número. Se sim, o jogador deu resposta
    if (*highestValue == *lowestValue) {
        return -1;
    }

    if (isValueBigger) { // Verifica se o número pensado pelo jogador é maior que o chute anterior.
        *lowestValue = lastValue + 1; // Se o valor anteriror é maior, então o mínimo passa a ser o nú
    }
    else { // Verifica se o número pensado pelo jogador é menor que o chute anterior.
        *highestValue = lastValue - 1; // Se o valor anterior é menor, então o máximo passa a ser o nú
    }

    return ceil((*highestValue + *lowestValue) / 2); // Então, ele pega a metade entre o máximo e o mí
}
```

Já no modo 2: O chute ainda será entre o maior e menor valor informado até o momento, mas será um número qualquer entre eles sem nenhum parâmetro de seleção:

```c
int guessOtherValueBasedOnExtremes(const int lastValue, int *highestValue, int *lowestValue) {
    bool isValueBigger = checkWithUserIfNumberIsHigher(lastValue);

    // Verifica se os valores máximo e mínimo chegaram ao mesmo número. Se sim, o jogador deu respostas inconsistentes
    if (*highestValue == *lowestValue) {
        return -1;
    }

    if (isValueBigger) { // Verifica se o número pensado pelo jogador é maior que o chute anterior.
        *lowestValue = lastValue + 1; // Se o valor anteriror é maior, então o mínimo passa a ser o número acima dele
    }
    else { // Verifica se o número pensado pelo jogador é menor que o chute anterior.
        *highestValue = lastValue - 1; // Se o valor anterior é menor, então o máximo passa a ser o número abaixo dele
    }

    return createRandomNumber(*lowestValue, *highestValue); // Então, ele gera um número aleatório qualquer entre os doi
}
```

3. Placar Final das Rodadas

Esse ciclo se repete “n” vezes, sendo “n” a quantidade de turnos informada pelo jogador no início da execução do programa. Ao finalizar a quantidade de turnos, o sistema informa o vencedor geral e a quantidade de turnos que cada um ganhou. Se essa quantidade for igual, é considerado um empate.

```
[JUIZ]: Computador acertou em 6 tentativas e o jogador em 5
[JUIZ]: Usuário venceu a rodada número 2 de um total de 2.
[JUIZ]: Placar das rodadas:
Computador: 0 Usuário: 1
[JUIZ]: Usuário venceu.
[JUIZ]: Placar final:
 Computador 0 | Usuário: 1
```

Vitória geral no jogo

```
[JUIZ]: Computador acertou em 5 tentativas e o jogador em 5
[JUIZ]: Empate na rodada número 1 de um total de 1.
[JUIZ]: Placar das rodadas:
Computador: 0 Usuário: 0
[JUIZ]: Houve um empate.
[JUIZ]: Placar final:
 Computador 0 | Usuário: 0
```

Empate geral no jogo

O empate também pode ocorrer em um turno, caso a quantidade de tentativas da máquina seja igual a quantidade de tentativas do jogador. Então, nenhum dos dois recebem pontos no turno:

```
<<<< Iniciando o turno 1 >>>>
[JUIZ]: Iniciando vez do usuário
[IA]: Qual o meu número secreto? <_<
[IA]: >.>
[JUIZ]: Digite um número entre 0 e 20: 12

[IA]: >:3 AAAA, Errou! O número secreto é menor. Tente de novo: 6

[IA]: >:3 AAAA, Errou! O número secreto é maior. Tente de novo: 8

[IA]: >:3 AAAA, Errou! O número secreto é menor. Tente de novo: 7

[IA]: Parabéns, você acertou em 4 tentativas. >.<
[JUIZ]: Pense em um número secreto entre 0 e 20. Pensou (s/n)? s

[JUIZ]: Iniciando vez da IA
[IA]: >:/ Você... penso no número... :D 17! (s/n)? n

[IA]: :S Hummm... o número é maior do que 17 (s/n)? n

[IA]: >:/ Você... penso no número... :D 5! (s/n)? n

[IA]: :S Hummm... o número é maior do que 5 (s/n)? s

[IA]: >:/ Você... penso no número... :D 8! (s/n)? n

[IA]: :S Hummm... o número é maior do que 8 (s/n)? s

[IA]: >:/ Você... penso no número... :D 13! (s/n)? s

[JUIZ]: Computador acertou em 4 tentativas e o jogador em 4
[JUIZ]: Empate na rodada número 1 de um total de 2.
[JUIZ]: Placar das rodadas:
Computador: 0 Usuário: 0
```

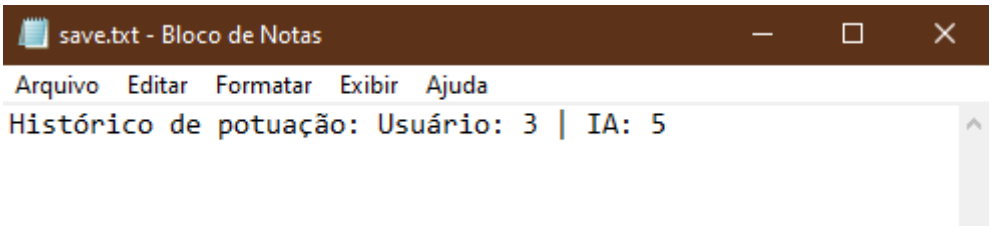
4. Exibir Histórico de Resultados

Por fim, todas as pontuações resultantes são gravadas em um arquivo chamado “save.txt”.

Na primeira execução do programa, caso o arquivo não exista, ele é criado dentro da pasta save por meio das duas funções a seguir:

```c
void createSaveFolder() {
    struct stat directory = {0};

    // Verifica se já existe o diretório de save do jogo
    if (stat(SCORE_SAVE_PATH, &directory) == -1) { // Se não, ele é criado
        mkdir(SCORE_SAVE_PATH);
    }

    return;
}

void createSaveFile() {
    // Tenta abrir o arquivo em modo a, pois caso já exista ele é apenas aberto. Por outro lado, caso não exista, ele é criado
    FILE *fileRef = fopen(SCORE_SAVE_FULL_FILE_PATH, "a");
    fclose(fileRef);

    return;
}
```

As pontuações são salvas no arquivo da seguinte forma:



E são atualizadas e exibidas ao usuário no final de cada execução do programa:

```
[JUIZ]: Computador acertou em 5 tentativas e o jogador em 5
[JUIZ]: Empate na rodada número 1 de um total de 1.
[JUIZ]: Placar das rodadas:
Computador: 0 Usuário: 0
[JUIZ]: Houve um empate.
[JUIZ]: Placar final:
 Computador 0 | Usuário: 0
[JUIZ]: Historicamente o computador já venceu 5 rodadas e o usuário 3 rodas.
```