Criar o Jogo da Forca

Linguagem C/C++

O que vamos aprender?

- Criar menu inicial
- Transição de C para C++ (opcional)
- Aleatório (srand), máscaras, status do jogo
- Contagem de tentativas
- Condição de vitória ou derrota
- Ciclo do jogo (loop)
- Mensagens
- Inicio do jogo
- Usar maiúsculas/minúsculas
- Multiplayer
- Limpar a tela/ecrã

```
void limpaTela(){
system ("CLS");
}
```

Para fazer este jogo necessita

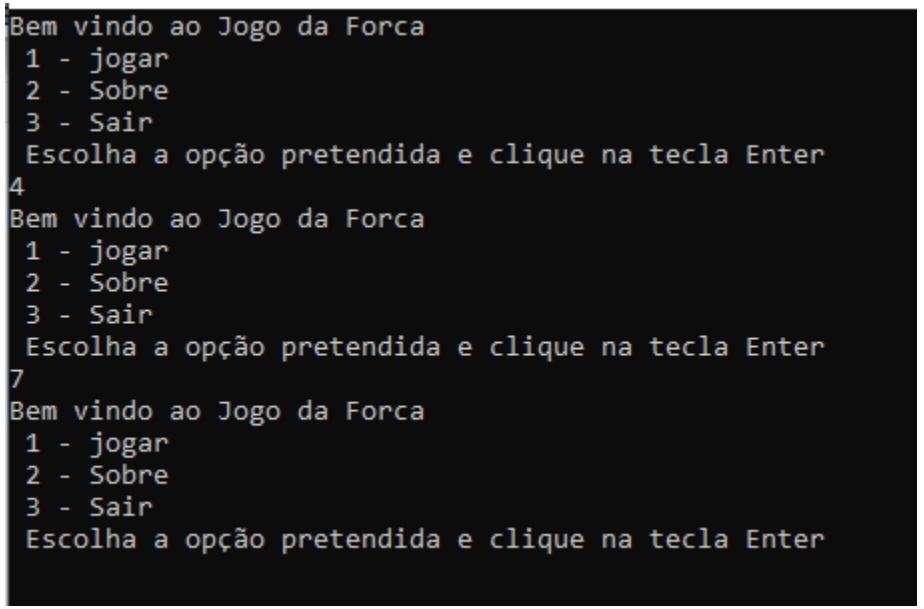
- Saber o básico da linguagem C/C++, nomeadamente:
- Imprimir variáveis no ecrã
- Fazer laços de repetição
- Criar funções e passar parametros

1º Passo: Criar Menu Inicial

Criar Menu Inicial

- O menu vai receber o valor inserido pelo utilizador e mostra a opção pretendida
- Caso seja inserido um valor diferente do esperado, limpa a tela e apresenta o menu de novo
- Use o Switch Case

- Como fazemos para que apareça o menu ao utilizador enquanto ele digitar um valor que não se encontra nas opçoes?
- A solução é criar um while antes do menu e colocar o menu dentro do while, assim:



- Vamos agora otimizar o nosso Código criando uma função para a criação do menu em vez de criar o mesmo no main

- Será uma função void que não retorna qualquer valor.

Passo 2: palavra aleatória

Para gerar uma palavra aleatória precisamos:

- Biblioteca string (já incluida)
- Biblioteca time.h
- Comando rand para gerar números aleatórios
  srand((unsigned)time(NULL));
- Criar um vetor de palavras
- Criar uma função para organizar e reaproveitar o código

- Primeiro vou criar um indice aleatório para gerar uma palavra
- Começamos por criar um vetor de palavras onde irei inserir as palavras que quero que façam parte do jogo
- Vou criar uma função chamada palavraaleatoria onde irei colocar o Código para gerar o vetor de palavras
- Começo por cria um vetor do tipo string no qual defino o seu tamanho. Neste caso, vou já iniciá-lo com 3 palavras: Aluno, Professor, Aula

```
void palavraaleatoria(){

//vetor com as palavras disponíveis
string palavras [3] = {"Aluno", "Professor", "Aula"};

}
```

- Agora vamos criar o indice aleatório para mostrar aleatóriamente no ecrã uma das palavras definidas anteriormente
- Para isso, criamos uma variável do tipo inteiro com a função rand (que necessita da biblioteca time.h

```
int indiceAleatorio =rand();
```

- Depois disto, é preciso colocar a instução do aleatório dentro do main

```
int main (){
    setlocale( LC_ALL, "Portuguese");

    srand((unsigned)time(NULL));
    menuInicial();


    return 0;

}
```

Passo 3: criar máscara

- Vamos aprender a criar uma mascara no jogo da forca: o utilizador tem que advinhar uma palavra e, para isso, precisa saber quantas letras essa palavra tem
- A primeira coisa a saber para a criação da mascara é quantas letras vai ter
- Esse valor depende de cada palavra pois todas elas têm tamanho diferente
- Começo criando uma variável que vai determinar o tamanho da palavra usando a função size();

```
void palavraaleatoria(){

        //palavra a ser encontrada no jogo
//string palavraescondida=palavras[indiceAleatorio];

string palavraescondida=mostraPalavraAleatoria();
int tamanhopalavra=palavraescondida.size();
cout << "\nA palavra secreta é " <<palavraescondida<<" tamanho: " << tamanhopalavra;

}
```

- Pretendo criar uma mascara da palavra a ser apresentada para que apareça ao utilizador underscores para cada letra que a palavra contem
- No ciclo que repetição eu vou dizer que enquanto o Contador for menor que o tamanhodapalavra adiciona à palavraComMascara um underscore

Passo 4: Status do Jogo

- No Status do jogo, eu tenho que conseguir ver:
- A palavraComMascara (e o tamanho dela)
- As tentativas que ainda tenho disponíveis (sempre que insere uma nova letra, o número de tentativas reduz)
- Se inserir uma letra já digitada anteriormente, não desconta no número de tentativas e mostra uma mensagem com essa informação

- Primeiro tenho que criar uma variável para tentativas do tipo inteiro que indica quantas tentativas o utilizador já arriscou
- e outra variável maxtentativas para saber quantas o utilizador ainda pode arriscar
- Dentro da função palavraAleatoria acresento as varíaveis descritas

```
void palavraAleatoria(){

   string palavraescondida =mostraPalavraAleatoria();
    int    tamanhoPalavra= palavraescondida.size();

    string palavraComMascara=mostraPalavraComMascara( palavraescondida, tamanhoPalavra);

    int tentativas =0, maxTentativas=5;

    cout << "a palavra secreta é: " << palavraescondida << "tamanho: "<< tamanhoPalavra;
    cout << palavraComMascara;

}
```

- De seguida vamos ter que criar um loop (ciclo de repetição) e vou testar o que?
- Enquanto o jogador não acertar a palavra e ainda não tiver atingido o valor máximo de tentativas permitidas

```
while (maxTentativas-tentativas>0){
```

- Então vamos apresentar ao jogador a mensagem para digitar uma letra e vamos guardar essa letra numa variavel que vamos criar agora do tipo char

```
int tentativas =0, maxTentativas=5;
char letra;
while (maxTentativas-tentativas>0){
    cout << "Digite uma letra: ";
    cin >> letra;

}
```

- Ao inserir uma nova letra, é necessário contabilizar uma nova tentative e sendo assim acresentamos:

```
cin >> letra;
tentativas++;
```

- Vamos agora apagar os prints/couts que se encontravam na função e criar uns novos dentro do while que mostre a palavraComMascara e o tamanho, bem como o número de tentativas restantes

```
int tentativas =0, maxTentativas=5;
char letra;
while (maxTentativas-tentativas>0){

    cout << "Palavra: " << palavraComMascara << " (Tamanho: "<< tamanhoPalavra << ")";
    cout << "\nRestantes Tentativas: " <<maxTentativas-tentativas;

    cout << "\nDigite uma letra: ";
    cin >> letra;
    tentativas++;

}
```

- Para otimizar o nosso Código, Podemos criar uma nova função chamada StatusJogo onde Podemos colocar esses cout que acabamos de criar desta forma:

```
void StatusJogo(string palavraComMascara, int tamanhoPalavra, int tentativasRestantes){
    cout << "Palavra: " << palavraComMascara << " (Tamanho: "<< tamanhoPalavra << ")";
    cout << "\nRestantes Tentativas: " <<tentativasRestantes;


}
```

- Atenção: para facilitar a passagem de pârametros, acrescentei o int tentativasRestantes em vez de usar a expressão maxTentativas-tentativas
- Agora so tenho que ter cuidado ao chamar/executar a função desta forma:

```
StatusJogo(palavraComMascara, tamanhoPalavra, maxTentativas-tentativas)
```

- Podemos agora limpar a tela.
- O Código final fica assim:

```
void StatusJogo(string palavraComMascara, int tamanhoPalavra, int tentativasRestantes){
    cout << "Palavra: " << palavraComMascara << " (Tamanho: "<< tamanhoPalavra << ")";
    cout << "\nRestantes Tentativas: " <<tentativasRestantes;

}

void palavraAleatoria(){

   string palavraescondida =mostraPalavraAleatoria();
    int    tamanhoPalavra= palavraescondida.size();

    string palavraComMascara=mostraPalavraComMascara( palavraescondida, tamanhoPalavra);

    int tentativas =0, maxTentativas=5;
    char letra;
    while (maxTentativas-tentativas>0){
        limpaTela();

        StatusJogo(palavraComMascara, tamanhoPalavra, maxTentativas-tentativas)
        cout << "\nDigite uma letra: ";
        cin >> letra;
        tentativas++;
    }
```

Passo 5: Mostrar Palavra, Vitória e Derrota

- Vamos começar por verificar se a letra inserida pelo utilizador existe na palavraComMascara.
- Caso exista, deve mostrar a letra na posição certa e descontar o número de tentativas, caso não exista deve apenas descontar as tentativas
- Então, depois do utilizador digitar a letra, vamos percorrer cada posição da nossa palavra e verificar se essa letra existe na palavra e mostrá-la nessa posição
- Para tal, vamos ter que criar uma nova variável do tipo int chamada contador para auxiliar neste processo

- Dentro da função palavraAleatoria() e antes do while adiciono a variável contador do tipo int

```
int contador;
```

- Dentro do while, depois de guardar o valor inserido pelo jogador na variável letra (cin>>letra) iniciamos o nosso ciclio for

```
void palavraAleatoria(){

    string palavraescondida =mostraPalavraAleatoria();
     int    tamanhoPalavra= palavraescondida.size();

     string palavraComMascara=mostraPalavraComMascara( palavraescondida, tamanhoPalavra);

     int tentativas =0, maxTentativas=5;
     char letra;
     int contador;
     while (maxTentativas-tentativas>0){
         limpaTela();

         StatusJogo(palavraComMascara, tamanhoPalavra, maxTentativas-tentativas);
         cout << "\nDigite uma letra: ";
         cin >> letra;

         //ciclo para percorrer toda a palavraComMascara

         for (contador=0; contador<tamanhoPalavra; contador++){

             //verificação se a letra existe na palavra ou não - vetor palavras
             if (palavraescondida[contador]==letra){

                 //faço a letra aparecer na palavraComMascara
                 palavraComMascara[contador]=palavraescondida[contador];
             }
         }
         tentativas++;
     }
}
```

- Verificamos o seguinte: mesmo que já tenha acertado na palavra continuo no jogo caso ainda tenta tentativas restantes. Sendo assim, temos que acrescentar essa validação ao nosso **while**
- Ou seja, acrescentamos a condição que enquanto a palavra escondida seja diferente da palavraComMascara além do numero de tentativas restantes maior que zero, desta forma:

```
while (palavraescondida != palavraComMascara && maxTentativas-tentativas>0){
```

- Depois disto, vamos acrescentar a verificação se ganhou ou perdeu logo depois de sair do while. Para isso usamos um IF

```
if (palavraescondida==palavraComMascara){
    limpaTela();
    cout << "parabéns, acertou na palavra";
} else{

 limpaTela();
    cout << "Perdeu o jogo";
}
```

- O código final da função palavraAleatoria() é o seguinte:

```
void palavraAleatoria(){

   string palavraescondida =mostraPalavraAleatoria();
    int    tamanhoPalavra= palavraescondida.size();

    string palavraComMascara=mostraPalavraComMascara( palavraescondida, tamanhoPalavra);

    int tentativas =0, maxTentativas=5;
    char letra;
    int contador;

    while (palavraescondida != palavraComMascara && maxTentativas-tentativas>0){

        limpaTela();

        StatusJogo(palavraComMascara, tamanhoPalavra, maxTentativas-tentativas);
        cout << "\nDigite uma letra: ";
        cin >> letra;

        //ciclo para percorrer toda a palavraComMascara

        for (contador=0; contador<tamanhoPalavra; contador++){

            //verificação se a letra existe na palavra ou não - vetor palavras
            if (palavraescondida[contador]==letra){

                //faço a letra aparecer na palavraComMascara
                palavraComMascara[contador]=palavraescondida[contador];
            }
        }
        tentativas++;
    }

    if (palavraescondida==palavraComMascara){
        limpaTela();
        cout << "parabéns, acertou na palavra";
    } else{

     limpaTela();
        cout << "Perdeu o jogo";
    }
}
```

Passo 6: Mostrar letras que já arriscou

- Começamos por criar uma variável do tipo string que vai guardar todas as letras que já foram digitadas pelo jogador dentro da função palavraAleatoria()

```
//variaveis
int tentativas =0, maxTentativas=5;
char letra;
int contador;
string letrasDigitadas;
```

- Esta variável vai ser usada onde?
  - Dentro do while pois sempre que o utilizador digita e guarda uma letra na variável letra acrescenta-a a esta nova variável

```
//ciclo de repetição
while (palavraescondida != palavraComMascara && maxTentativas-tentativas>0){

    limpaTela();

    StatusJogo(palavraComMascara, tamanhoPalavra, maxTentativas-tentativas);
    cout << "\nDigite uma letra: ";
    cin >> letra;
    letrasDigitadas +=letra;
```

- Também vou passar essa variável como argumento da função StatusJogo() (quando o chamo a ser executado dentro do while

```
while (palavraescondida != palavraComMascara && maxTentativas-tentativas>0){

    limpaTela();

    StatusJogo(palavraComMascara, tamanhoPalavra, maxTentativas-tentativas, letrasDigitadas);
    cout << "\nDigite uma letra: ";
    cin >> letra;
    letrasDigitadas +=letra;
```

- E, obviamente, dentro da própria função StatusJogo() passo como parâmetro

```
void StatusJogo(string palavraComMascara, int tamanhoPalavra, int tentativasRestantes, string letrasDigitadas){
```

- e mostro o resultado nessa função da seguinte forma:

- Em vez de fazer apenas um cout e com o resultados das letrasDigitadas vou querer que mostre um espaço e uma vírgula depois de cada letra digitada.
- Para tal, posso fazer da seguinte forma:
- Crio uma variável int contador como variável auxiliar

```
void StatusJogo(string palavraComMascara, int tamanhoPalavra, int tentativasRestantes, string letrasDigitadas){
    cout << "Palavra: " << palavraComMascara << " (Tamanho: "<< tamanhoPalavra << ")";
    cout << "\nRestantes Tentativas: " <<tentativasRestantes;

    //exibe as letras digitadas
    int contador;
    cout << "\nLetras Digitadas: ";

    for (contador =0; contador < letrasDigitadas.size(); contador++){
        cout << letrasDigitadas[contador] <<", ";
    }


}
```

- Agora vamos acrescentar uma “operação” que verifica se o jogador já digitou essa letra e, caso já tenha digitado, avisa e não desconta no número de tentativas (na função palavraAleatoria())
- Criamos uma variável auxiliar do tipo boleano pra verificar se a letra já foi digitada ou não

```
//variaveis
int tentativas =0, maxTentativas=5;
char letra;
int contador;
string letrasDigitadas; // acumula as tentativas do jogador
bool jaDigitouLetra= false;
```

- Agora dentro do ciclo While vou acrescentar um ciclo for para verificar se a letra já foi digitada ou não

```
cin >> letra;

for(contador =0; contador<tentativas; contador++){

}


letrasDigitadas +=letra;
```

- Dentro do ciclo for faço um ciclo for para verificar se o meu letrasDigitadas na posição do contador for igual a letra então faço com que o meu jaDigitouLetra seja igual a verdadeiro e informo que essa letra já foi digitada

```
StatusJogo(palavraComMascara, tamanhoPalavra, maxTentativas-tentativas, letrasDigitadas);
cout << "\nDigite uma letra: ";
cin >> letra;

//percorre as letras digitadas
for(contador =0; contador<tentativas; contador++){
        //se econtrar a letra
    if(letrasDigitadas[contador]== letra){
        cout << "\nEssa letra já foi digitada"
        //indica com a variavel booleana
        jaDigitouLetra=true;
    }
}


letrasDigitadas +=letra;
```

- Caso não tenha digitado, fora do ciclo for crio um novo IF onde o teste a fazer é se jaDigitouLetra==false – letra nova
- Se sim, então vai mostrar a letra e contar a tentativa. Sendo assim copiamos o código feito anteriormente para este IF

- Código copiado:

```
letrasDigitadas +=letra;

//ciclo para percorrer toda a palavraComMascara

for (contador=0; contador<tamanhoPalavra; contador++){

    //verificação se a letra existe na palavra ou não - vetor palavras
    if (palavraescondida[contador]==letra){

        //faço a letra aparecer na palavraComMascara
        palavraComMascara[contador]=palavraescondida[contador];
    }
}
tentativas++;
}
```

- Código Final:

```
//ciclo de repetição
while (palavraescondida != palavraComMascara && maxTentativas-tentativas>0){

    limpaTela();

    StatusJogo(palavraComMascara, tamanhoPalavra, maxTentativas-tentativas, letrasDigitadas);
    cout << "\nDigite uma letra: ";
    cin >> letra;

    //percorre as letras digitadas
    for(contador =0; contador<tentativas; contador++){
            //se econtrar a letra
        if(letrasDigitadas[contador]== letra){
            cout << "\nEssa letra já foi digitada";
            //indica com a variavel booleana
            jaDigitouLetra=true;
        }
    }
        //se for uma letra nova
        if (jaDigitouLetra==false){

              letrasDigitadas +=letra;

            //ciclo para percorrer toda a palavraComMascara

            for (contador=0; contador<tamanhoPalavra; contador++){

                //verificação se a letra existe na palavra ou não - vetor palavras
                if (palavraescondida[contador]==letra){

                //faco a letra aparecer na palavraComMascara
                palavraComMascara[contador]=palavraescondida[contador];
                }
            }
        tentativas++;
    }
}
```

- O código anterior tem um bug: quando eu digito uma letra que não existe ele fica preso nesse ponto e deixe de contabilizar tentativas ou mostrar letras
- Para vermos a funcionar temos que remover o limpaTela()
- Então, o que podemos fazer para contornar esta situação? Uma forma é criar uma mensagem:
- Criamos uma variável do tipo string para “tratar” das nossas mensagens (e assim podemos manter o limpaTela()
- Então, dentro da função palavraAleatoria() vamos criar a seguinte variável:

```
string mensagem;
```

- Após criar a variável, vamos passar a mesma como parâmetro quando chamamos a função StatusJogo()

```
StatusJogo(palavraComMascara, tamanhoPalavra, maxTentativas-tentativas, letrasDigitadas, mensagem);
cout << "\nDigite uma letra: ";
cin >> letra;
```

- Agora, em vez de :

```
//percorre as letras digitadas
for(contador =0; contador<tentativas; contador++){
        //se econtrar a letra
    if(letrasDigitadas[contador]== letra){
        cout << "\nEssa letra já foi digitada";
```

- Coloco isto:

```
//percorre as letras digitadas
for(contador =0; contador<tentativas; contador++){
        //se econtrar a letra
    if(letrasDigitadas[contador]== letra){
        mensagem = "\nEssa letra já foi digitada";
        //indica com a variavel booleana
```

- Agora vou à função StatusJogo() e passo a mensagem como parâmetro e faço um cout desse parâmetro assim:

```
void StatusJogo(string palavraComMascara, int tamanhoPalavra, int tentativasRestantes, string letrasDigitadas, string mensagem){
    cout << mensagem <<"\n";
    cout << "Palavra: " << palavraComMascara << " (Tamanho: "<< tamanhoPalavra << ")";
    cout << "\nRestantes Tentativas: " <<tentativasRestantes;

    //exibe as letras digitadas
    int contador;
    cout << "\nLetras Digitadas: ";

    for (contador =0; contador < letrasDigitadas.size(); contador++){
         cout << letrasDigitadas[contador] <<", ";
    }


}
```

- E agora a minha mensagem já aparece mesmo com o limparTela()
- Mas continua a apresentar um bug: a mensagem aparece mas continua “bloqueado”

- Para resolver esse erro vou fazer duas coisas.
- Primeiro: vou criar um ciclo IF para fazer a verificação se já acertou na letra ou não e passo para ali as mensagens

Para tal, crio mais uma variável booleana `bool acertouLetra= false;`

Como começa falsa, vou dizer que é verdadeira no if quando a if (palavraescondida[contador]==letra){

```
for (contador=0; contador<tamanhoPalavra; contador++){

    //verificação se a letra existe na palavra ou não - vetor palavras
    if (palavraescondida[contador]==letra){

    //faço a letra aparecer na palavraComMascara
    palavraComMascara[contador]=palavraescondida[contador];
    acertouLetra = true;
    }
}
```

De seguida crio o seguinte IF (dentro do while – junto a tentativas ++) para indicar se errou ou acertou em alguma letra

```
if (acertouLetra == false){
    mensagem = "Errou uma letra";
}
else{
     mensagem = "Acertou uma letra";
}
```

Por fim, tenho que reiniciar as variáveis aos seus valores iniciais para não “bloquear” novamente (logo depois do IF)

```
//reinicia as variaveis
jaDigitouLetra= false;
acertouLetra= false;
```

Passo 7: Letras Maiúsculas e Minúsculas

- Se escrevemos as palavras em maiúsculas, o programa só vai aceitar como válido a inserção por parte do jogador de palavras maiúsculas e vise-versa.
- Para resolver eta situação vamos fazer duas coisas:
- Primeiro: uniformizar as nossas palavras colocando-as todas em minúsculas:

```
string mostraPalavraAleatoria(){
    //vetor com palavras disponiveis
    string palavras[3] = {"aluno", "professor", "aula"};

    int indiceAleatorio =rand() % 3;
    return palavras[indiceAleatorio];


}
```

- Desta forma garantimos que se o jogador escrever em minúsculas vai sempre aceitar a sua letra e comparar

- No entanto, caso escreva em maiúscula, vamos ter que dar a instrução ao programa que deve converter para minúscula. Como fazemos isso? Usamos a função tolower()
- Vou ter que alterar em dois locais:
- No IF de verificação das letras já digitadas:

```
//percorre as letras digitadas
for(contador =0; contador<tentativas; contador++){
        //se econtrar a letra
    if(letrasDigitadas[contador]== tolower(letra)){
        mensagem = "\nEssa letra já foi digitada";
        //indica com a variavel booleana
        jaDigitouLetra=true;
    }
}
```

- no IF de verificação na letra nova

```
//ciclo para percorrer toda a palavraComMascara

for (contador=0; contador<tamanhoPalavra; contador++){

    //verificação se a letra existe na palavra ou não - vetor palavras
    if (palavraescondida[contador]== tolower(letra)){

    //faço a letra aparecer na palavraComMascara
    palavraComMascara[contador]=palavraescondida[contador];
    acertouLetra = true;
    }
}
```

```
//percorre as letras digitadas
for(contador =0; contador<tentativas; contador++){
        //se ecntrar a letra
    if(letrasDigitadas[contador]== tolower(letra)){
        mensagem = "\nEssa letra já foi digitada";
        //indica com a variavel booleana
        jaDigitouLetra=true;
    }
}
    //se for uma letra nova
    if (jaDigitouLetra == false){

          letrasDigitadas += letra;

        //ciclo para percorrer toda a palavraComMascara

        for (contador=0; contador<tamanhoPalavra; contador++){

            //verificação se a letra existe na palavra ou não - vetor palavras
            if (palavraescondida[contador]== tolower(letra)){

            //faço a letra aparecer na palavraComMascara
            palavraComMascara[contador]=palavraescondida[contador];
            acertouLetra = true;
            }
        }
```

DESAFIO

- Altere o nome da função void palavraAleatoria() para Jogo()
- Aumentar o numero máximo de tentativas para 12
- Acrescentar mais 4 palavras dentro do tema apresentado
- Crie texto para a opção Sobre do menu inicial (exemplo: Jogo criado por Susana Caetano no âmbito da UFCD 0809)
- Crie novas funções que permitam:
  - Reiniciar o jogo
  - Arriscar uma palavra inteira
  - Jogar 2 jogadores em simultâneo