CENTRO UNIVERSITÁRIO DE JOÃO PESSOA
CURSO DE GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

LUCAS VINICIUS PESSOA DE FRANCA
SÉRGIO MARCELINO NÓBREGA DE CASTRO
VICTOR HENRIQUE DE AMORIM CARAUBAS
VINÍCIUS BEZERRA CAVALCANTI CENTURION

RELATÓRIO DE DESENVOLVIMENTO DE JOGO EM LINGUAGEM C: Jogo da forca

JOÃO PESSOA
2020

LUCAS VINICIUS PESSOA DE FRANCA 23401494
SÉRGIO MARCELINO NÓBREGA DE CASTRO 22992987
VICTOR HENRIQUE DE AMORIM CARAUBAS 23836024
VINÍCIUS BEZERRA CAVALCANTI CENTURION 23645831

RELATÓRIO DE DESENVOLVIMENTO DE JOGO EM LINGUAGEM C: Jogo da forca

Trabalho apresentado ao Prof. Me. Leandro Figueiredo Alves como requisito parcial a obtenção de nota referente a segunda avaliação na disciplina de Técnicas de Desenvolvimento de Algoritmos do curso de graduação em Ciências da Computação do Centro Universitário de João Pessoa.

João Pessoa
2020

CASTRO, Sérgio Marcelino Nóbrega de; CARAÚBAS, Victor H. de A.; CENTURION, Vinícius B. C.; FRANCA, Lucas V. P. **Relatório de desenvolvimento de jogo em linguagem C**: Jogo da forca. João Pessoa: Centro Universitário de João Pessoa – UNIPÊ, 2020. Trabalho acadêmico.

RESUMO

O presente trabalho desenvolvido em grupo tem por finalidade a documentação das atividades de desenvolvimento de aplicação em linguagem C para um jogo de livre escolha. O trabalho realizado começa apresentando o jogo escolhido para desenvolvimento, Jogo da Forca (*hangman game*), algumas noções acerca de sua origem, que remonta às práticas penais existentes na Europa medieval, e popularização ao longo da história, em que se torna jogo infantil de adivinhação, bem como as regras adotadas para as modalidades de jogo desenvolvidas: individual e multijogador. Apresenta um breve planejamento das atividades a serem desenvolvidas pelos integrantes, que é revisitada posteriormente abordando-se seu cumprimento, alterações e eventuais redistribuições de atividades remanescentes. Os resultados apresentados abrangem a descrição do funcionamento da aplicação, relatos de desafios experimentados pela equipe de desenvolvimento na produção do código e na distribuição das tarefas. Por fim, apresenta algumas oportunidades de melhoria para o código elaborado e sugestão de trabalhos futuros.

Palavras-chave: linguagem C, jogo da forca, *hangman game*, desenvolvimento de jogos.

SUMÁRIO

1 INTRODUÇÃO

O presente trabalho tem por finalidade registrar o desenvolvimento de um jogo de livre escolha unicamente em linguagem C a partir dos assuntos abordados ao longo da disciplina de técnicas de desenvolvimento de algoritmos (2020.2) e laboratório de desenvolvimento de algoritmos do curso de graduação em ciências da computação.

A linguagem C, desenvolvida por Dennis Ritchie na década de 1970 baseada na linguagem BCPL, é usada por programadores no desenvolvimento dos mais diversos tipos de sistema, desde sistemas operacionais, passando por compiladores, editores de texto, etc. Além da diversidade de sistemas, é importante ressaltar que C é uma linguagem que pode ser utilizada tanto em máquinas com um alto poder de processamento, como em máquinas mais simples (SANTOS, SARAIVA e GONÇALVES, 2018). C também é considerada uma linguagem de Médio Nível, pois possui um conjunto de instruções necessárias para acesso ao hardware, como as linguagens de baixo nível (Assembly). Além disso, C pode ser simples e amigável, como uma linguagem de alto-nível deve ser.

O código da aplicação, desenvolvido em grupo, é apresentado apenso ao trabalho, sendo aqui abordados os pontos principais da aplicação bem como das atividades de seu desenvolvimento, como modalidades e regras adaptadas para o jogo, planejamento e distribuição de tarefas, desafios encontrados e soluções adotadas, sugestões de melhorias futuras, dentre outras.

O jogo escolhido pela equipe foi o jogo da forca, para o qual foram desenvolvidas duas modalidades de jogo: uma individual, em que um único jogador joga sozinho contra o programa (que sorteia aleatoriamente uma palavra a partir de uma lista pré-concebida pelos desenvolvedores); e um multijogador, em que dois jogadores se enfrentam criando uma palavra para o adversário adivinhar.

1.1 O “jogo da forca” e sua história

O jogo da forca tem como característica sua associação a pena de morte por enforcamento, bastante comum na Europa medieval.

> O enforcamento era um dos modos de execução sob a antiga lei romana, e foi subsequentemente derivado pelos anglo-saxões de seus ancestrais germânicos. Era a forma prescrita de punição para homicídio na Inglaterra no século 12 e, com o tempo, passou a substituir todas as outras formas de pena de morte para condenações por crime até a abolição da pena de morte na Grã-Bretanha em 1965. Os enforcamentos públicos foram realizados na Inglaterra até 1868, quando foram removidos para as prisões. O enforcamento tornou-se o modo padrão de execução em todo o Império Britânico e onde quer que a lei comum anglo-americana fosse adotada. (BRITANNICA, 2012)

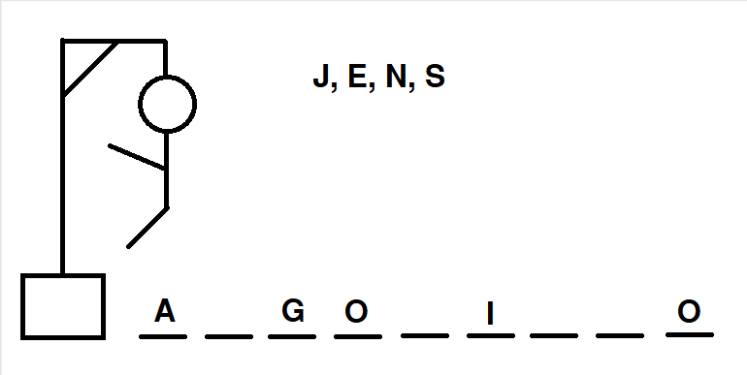
Popularmente, relata-se que sua origem remonta ao século XVII, sendo "jogado" quando um prisioneiro condenado a morte por enforcamento exigia um rito denominado "*Rite of Words and Life*" (rito de palavras e vida, tradução nossa), pelo qual o prisioneiro era posicionado na força sobre um banco com cinco pernas e o carrasco lhe apresentava uma palavra em branco. A cada palpite incorreto uma perna seria derrubada. Se adivinhasse a palavra corretamente, seria o prisioneiro libertado. Caso contrário, seria enforcado após a retirada da última perna (FLORENCE, 2019).

Figura 01. Enforcamento de bruxas (Fonte: GARDINER, 1655).

O jogo da forca (*hangman game* em inglês) é um jogo de adivinhação de dois ou mais jogadores, em que comumente são utilizados papel e lápis. Consiste basicamente em um jogador pensar em uma palavra e o outro tentar adivinhá-la sugerindo letras ou números, dentro de um certo número de adivinhações.

A palavra a ser adivinhada é representada por uma linha de travessões, cada qual representando uma das letras da palavra. Sempre que o jogador que tenta adivinhar sugere uma letra que ocorre na palavra, o jogador que pensou a palavra deve a escrever em todas as suas posições corretas. Se a letra sugerida não estiver presenta na palavra, o jogador então desenha um elemento de um boneco enforcado como uma marca de contagem.

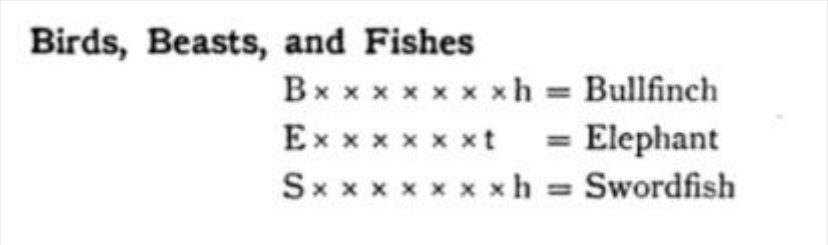
Figura 02. Exemplo de jogo da forca para palavra “algoritmo”.



O jogo acaba quando o jogador tiver adivinhado todos as letras da palavra ou a própria palavra. Por outro lado, se o jogador fizer suposições incorretas o suficiente para permitir que seu oponente complete o desenho do boneco, o jogo também termina, desta vez com o adivinhador perdendo.

O jogo não possui origem definida, embora se encontre variações que datam do final do século 19, como “*birds, beasts and fishes*”, documentado no livro “*Traditional Games of England, Scotland and Ireland*” de Alice Bertha Gomme (1894), volume 1 da coleção “*A Dictionary of British Fok-lore*”.

Figura 03. Exemplo de jogo de adivinhação do final do séc. 19 (fonte: GOMME, 1894).



O jogo se tornou popular mundialmente, tendo, inclusive, se tornado programa de televisão na década de 1960 pela TV Paulista, de nome “Vamos brincar

de Forca”; sendo ainda o primeiro programa apresentado pelo empresário e apresentador Silvio Santos.

1.2 Regras para multijogadores no jogo desenvolvido (1vs.1)

A) Sequência do jogo

Deve-se informar o nome dos jogadores: jogador 1 e jogador 2. Caso não seja informado, recebem os nomes “Jogador 1” e “Jogador 2”.

Para cada jogo, um dos jogadores deve informar uma palavra para o outro tentar adivinhar.

O número máximo de erros possíveis é igual a 7 (cabeça, tronco, braço esquerdo, braço direito, quadril, perna esquerda e perna direita).

Finalizado o primeiro jogo, deve ser informada a nova palavra para início do segundo jogo. Ao final desse segundo jogo, é mostrado a pontuação de cada jogador e quem foi o vencedor.

B) Sistema de pontuação

Acertar uma letra vale (+)10 pontos para cada vez que a letra aparece na palavra;

Errar 1 letra faz com que o número de erros permitidos (7 no total) diminua 1, desenhando uma parte do corpo do enforcado (cabeça, tronco, braço esquerdo, braço direito, quadril, perna esquerda, perna direita), e faz o jogador perder (-)10 pontos.

1.3 Regras para jogador individual

A) Sequência do jogo

O jogador é solicitado a inserir seu nome após a seleção do modo de jogo individual. Sendo sorteada uma palavra da lista pré-existente para que seja adivinhada pelo jogador.

O jogador então é solicitado a sugerir uma letra das letras da palavra. Caso acerte, ganha os pontos referentes a quantidade de vezes que a letra aparece. Caso erre, uma das partes do boneco da forca é desenhada e o jogador perde pontos.

O número máximo de erros possíveis é igual a 7 (cabeça, tronco, braço esquerdo, braço direito, quadril, perna esquerda e perna direita).

Finalizado o primeiro jogo, caso o jogador vença, é sorteada nova palavra para início do segundo jogo, e assim sucessivamente. Caso o jogador perca, ou seja, erre sete letras da palavra secreta, o saldo de pontos é mostrado e fornecido a função de ranqueamento para eventual classificação se cabível.

B) Sistema de pontuação

Acertar uma letra vale (+)10 pontos para cada vez que a letra aparece na palavra;

Errar 1 letra faz com que o número de erros permitidos (7 no total) diminua 1, desenhando uma parte do corpo do enforcado (cabeça, tronco, braço esquerdo, braço direito, quadril, perna esquerda, perna direita), e faz o jogador perder (-)10 pontos.

1.4 Sistema de ranqueamento

O sistema de ranqueamento é fornecido para os jogos do tipo individual, sendo salvo em arquivo externo ao programa.

A tela de pontuação mostra as três maiores informações existentes em arquivo ou cria uma lista caso não seja encontrado o arquivo.

1.5 Planejamento e distribuição das atividades

O trabalho foi dividido em partes: definição das regras do jogo; código fonte: layout; código fonte: jogo individual: lógica do jogo; código fonte: jogo individual: criação de lista de palavras para o jogo e sistema de seleção aleatória palavra para o jogo; código fonte: jogo multijogador; código fonte: sistema de pontuação; código fonte: revisão e teste; código fonte: elaboração de biblioteca externa; elaboração do relatório, elaboração do vídeo de apresentação.

Tabela 01: Atividades planejadas.

TAREFA	RESPONSÁVEL
Definição das regras do jogo	CENTURION
código fonte: layout	CENTURION
código fonte: jogo individual: lógica do jogo	CENTURION
código fonte: jogo individual: criação de lista de palavras para o jogo e sistema de seleção aleatória palavra para o jogo	CARAÚBAS
código fonte: jogo multijogador	FRANCA
código fonte: sistema de pontuação	CARAÚBAS
código fonte: revisão e teste	FRANCA
elaboração do relatório	CENTURION
elaboração do vídeo de apresentação	CARAÚBAS

As atividades acima previstas foram desenvolvidas de 16 de novembro (prazo de entrega dos grupos, nomes dos integrantes e jogo escolhido) até 30 de novembro do ano acorrente. Sendo fixado integrantes do grupo o prazo de 25 de novembro para entrega parcial dos códigos e 29 de novembro para entrega final dos códigos, relatório e vídeo de apresentação, restando o dia 30 para eventuais ajustes de apresentação.

2 RESULTADOS

2.1 Descrição geral do jogo

O jogo foi previsto em duas modalidades, uma de jogo individual, em que um único jogador tenta adivinhar uma série de palavras selecionadas aleatoriamente pelo programa a partir de uma lista pré-existente até que perca uma partida; e uma de jogo multijogador, na modalidade “versus”, em que dois jogadores se enfrentam, ganhando aquele que obtiver mais pontos.

Um sistema de ranqueamento de pontuações da modalidade individual também foi previsto, em que são mostradas as três maiores pontuações do jogo através de gravação dos resultados em arquivo externo.

A tela inicial do jogo é acessada a partir da função “telaInicial()” no main(), responsável por imprimir a abertura (estática) do jogo, seguida da tela de seleção do modo de jogo, pontuação e saída, conforme figura abaixo:

Figura 04. Tela de apresentação do jogo impressa a partir da função telaInicial() e tela de seleção de jogo impressa a partir da função telaMenu().

A função principal, main(), consiste basicamente de uma estrutura de repetição que solicita a entrada de uma das seleções disponíveis no menu de seleção, quais sejam: tipo de jogo, individual ou multijogador, tela de pontuação, ou saída do jogo, conforme código abaixo:

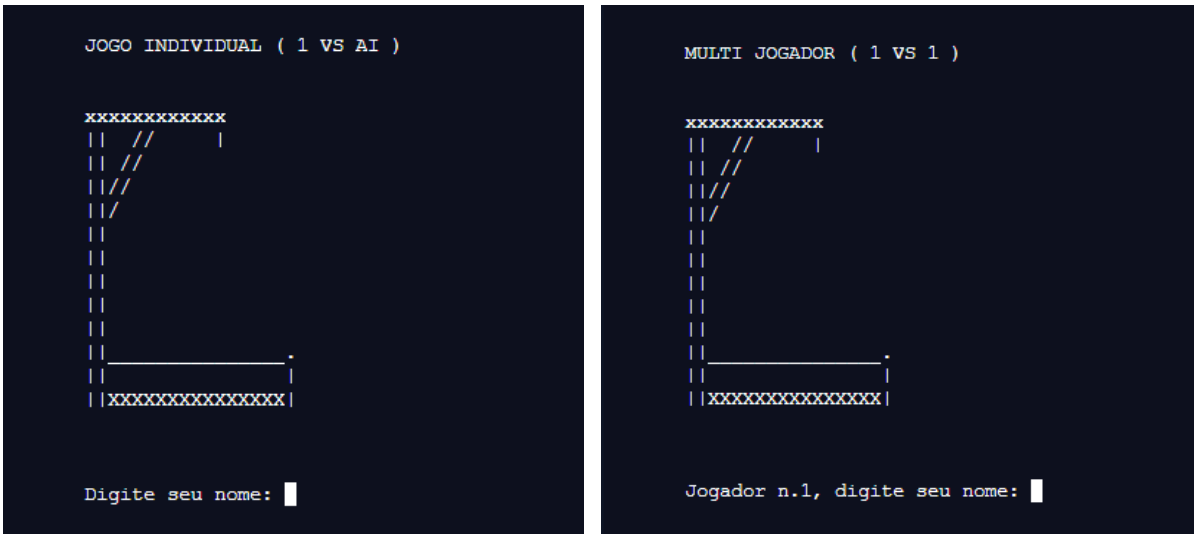
```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <time.h>
#include <locale.h>

	int main(void)
{
	setlocale(LC_ALL,"Portuguese");
	int tipoJogo=0, sair=0;
	telaInical();
	 do
	{
		system("cls");
		telaMenu();
		printf("\n\n\n     Selecione o modo de jogo: ");
		scanf("%d", &tipoJogo);
		flush_in();
		sair=selecionaJogo(tipoJogo);
	 } while (sair==0);
	system("cls");
	imprimeCreditos();
	return 0;
```

```
}
```

Observe-se que a função “selecionaJogo()” é responsável por controlar a saída da estrutura de repetição que mantém o usuário na tela de seleção. A partir dessa estrutura, o usuário tem acesso aos modos de jogo individual e multijogador, que representam funções independentes dentro do código, mas que apresentam bastante similaridade em sua lógica.

Figura 05. Tela de entrada no modo individual, acessado pela função “jogoIndividual()”, e tela de entrada no modo multijogador, acessado pela função “jogoMultiplayer()”.



Distinguem-se as modalidades na medida em que no jogo multijogador, um jogador escolhe uma palavra para seu oponente adivinha, enquanto na modalidade de jogo individual, a palavra é selecionada pelo programa a partir de uma lista pré-existente (em arquivo externo ao da aplicação) de modo aleatório.

Para o sorteio da palavra são utilizadas as bibliotecas <stdlib.h> e <time.h> através das funções “srand(time(0))” e “rand() % 30”, uma vez que a lista de palavras é composta por 30 elementos.

A lógica de cada jogo consiste em verificar se a palavra digitada ou sorteada contém ou não uma determinada letra que o jogador que tenta adivinhar a palavra fornece. Sendo utilizadas, para tanto, estruturas de comparação de *strings* (cadeias de caracteres) existentes na biblioteca <string.h>.

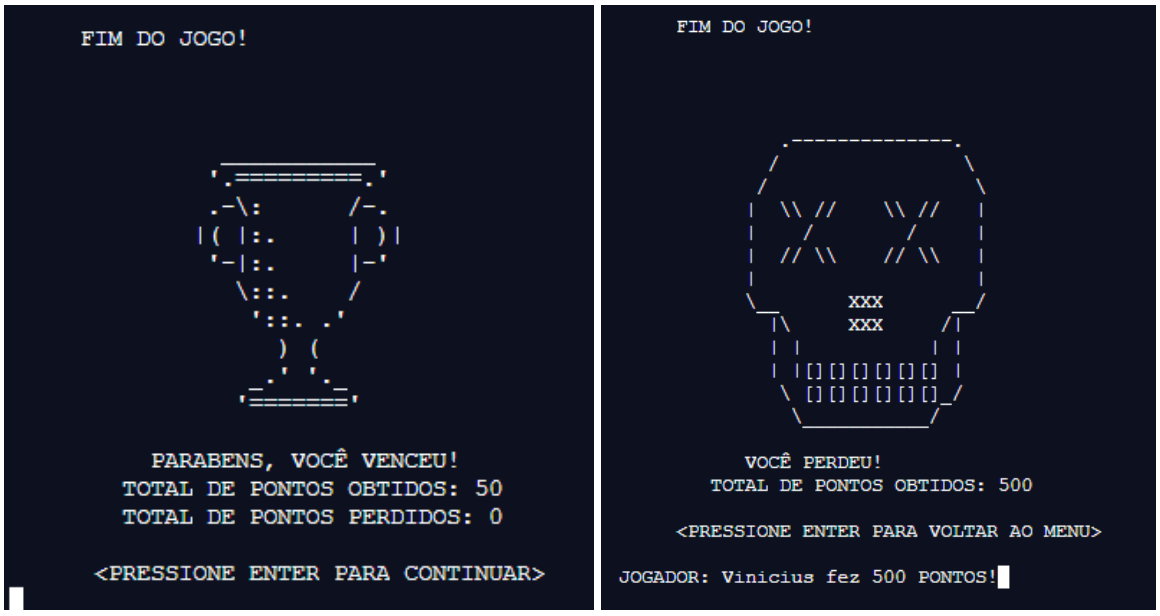
Dessa forma, a leitura e comparação (implementada pelas funções “buscaErro()” e “buscaPonto()”) das letras sugeridas são realizadas dentro de uma estrutura de repetição que só se interrompe caso se esgotem a possibilidade de erros

(que podem ser até o máximo de sete), ou caso o jogador adivinhe de fato a palavra secreta, conforme abaixo:

```
do
{
    printf("     Digite uma letra da palavra: ");
    scanf(" %c", &letra);
    nAcertos = buscaAcertos(palavra, letra, acertoTotal);
            for (i=0; i<30; i++)
            {
                    acertoTotal[i]=nAcertos[i];
            }
            if (buscaErro(palavra, letra)==1)
            {
                    letrasErradas[erros] = letra;
                    erros += 1;
            }
            totalPontos += buscaPontos(palavra, letra);
    } while ((erros<7)&&(totalPontos<maxPontos));
```

Cada uma das funções de tipo de jogo retorna um número inteiro (int) que representa a pontuação do jogador cujo nome foi anteriormente informado.

Figura 06. Tela de pontuação do jogo retornando o valor de pontuação do jogador.



A tela de pontuação é acessada através do menu de seleção e mostra as maiores pontuações obtidas nos jogos individuais. Tais dados são gravados em arquivo externo ao programa, e na hipótese de não existir, é criado um arquivo e salvos os dados desde então.

Figura 07. Tela de pontuação do jogo retornando o valor de pontuação do jogador.

Por fim, a seleção de saída do programa demanda confirmação pelo usuário, apresentando posteriormente a tela de créditos da aplicação.

Figura 07. Tela de pontuação do jogo retornando o valor de pontuação do jogador.

2.2 Dificuldades encontradas e soluções implementadas

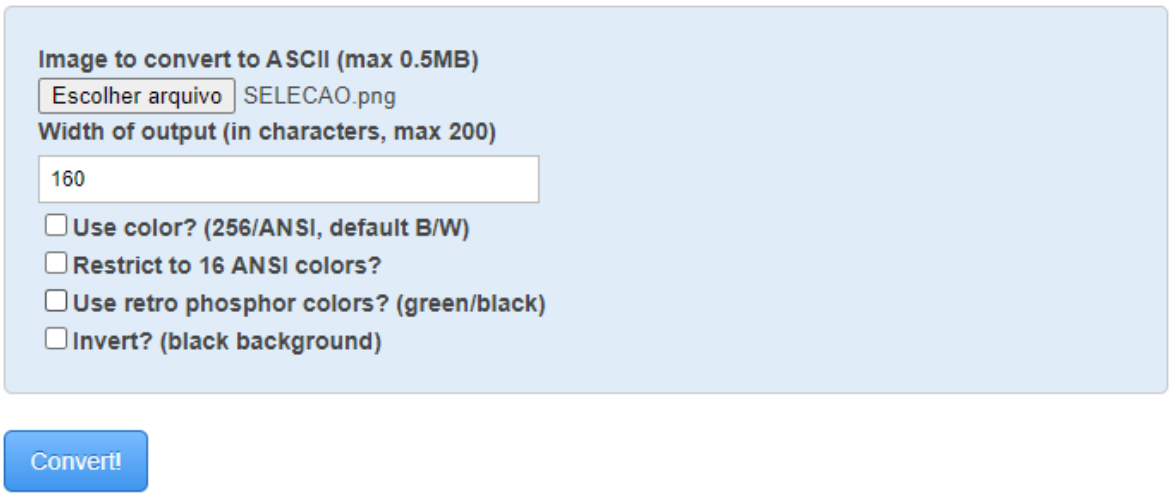
2.2.1. Gerar figuras para impressão de desenhos, menu de seleção, tela de apresentação e tela de créditos

A priori, foi tentado gerar figura a partir do bloco de notas a partir de esboço feito em caderno, conforme foi feito para desenhos da forca abaixo:

```
printf("     xxxxxxxxxxxx\n");
printf("     ||  //     |\n");
printf("     || //     ( )\n");
printf("     ||//      _|_\n");
printf("     ||/      /[X]\\\n");
printf("     ||      / [X] \\\n");
printf("     ||        ( )\n");
printf("     ||       /   \\\n");
printf("     ||      /     \\\n");
printf("     ||              \n");
printf("     ||_______________.\n");
printf("     || 1 2 3 4 5 6 7 |\n");
printf("     ||XXXXXXXXXXXXXX|\n");
```

Contudo, a tarefa mostrou-se extremamente morosa para outros desenhos; e embora alguns desenhos pudessem ser encontrados em códigos de terceiros, como o caso da figura de troféu e de caveira; buscou-se uma forma mais célere para gerar o menu de seleção, a tela inicial, a tela de créditos e a tela de pontuação.

As figuras foram geradas a partir de ferramenta disponibilizada online pelo endereço eletrônico: <https://manytools.org/hacker-tools/convert-images-to-ascii-art/>, pela qual uma figura é enviada e convertida para caracteres da tabela ASCII em formato de texto com largura de linha (em caracteres) que varia até o limite de duzentos caracteres.



2.2.2. Estruturação do código principal

Inicialmente foram propostos diversos tipos de jogos de força, com regras distintas, em razão do trabalho ser realizado em equipe. Acordar-se a forma de jogo e as modalidades foi o primeiro desafio experimentado.

A partir de um esboço das regras do jogo, definiu-se quais seriam as atividades desenvolvidas sendo formulada uma planilha de atividades para distribuição das tarefas entre os integrantes da equipe, bem como a criação de um repositório eletrônico para armazenar os códigos gerados.

Também foi evidenciada a necessidade de criar várias funções para o código, de modo que fosse facilitada a visualização de cada tarefa do programa e que estruturas que se repetissem ao longo do código pudessem ser resgatadas de maneira simples e uniforme. Tal hipótese ocorreu para as telas de pontuações, em algumas imagens eram geradas após cada partida, conforme abaixo:

```
int imprimeTrofeu (void)
{
    printf("               ___________\n");
    printf("             '.=========.'\n");
    printf("             .-\\:      /-.\n");
    printf("            |( |:.     | )|\n");
    printf("             '-|:.     |-'\n");
    printf("               \\::.    /\n");
    printf("                '::. .' \n");
    printf("                  ) (   \n");
    printf("                _.' '._ \n");
    printf("               '======='\n");
    return 1;
}
```

2.2.3. Retorno de algumas funções

Algumas funções tinham como retorno uma cadeia de caracteres, contudo, foi observado que um vetor do tipo “int vetor[n.pontos]” não poderia ser o retorno de uma função por limitação da própria linguagem, de modo que, a partir de leitura de referência específica, constatou que a melhor solução seria o retorno de um ponteiro para o vetor, conforme abaixo:

```
int * buscaAcertos(char palavra[30], char letra, int acertos[30])
{
   int nAcertos[30];
   int *pAcertos = nAcertos;
   int i=0;

   for (i=0; i<30; i++)
   {
      nAcertos[i]=acertos[i];
   }

   for (i=0; i<30; i++)
   {
      if (palavra[i]==letra)
      {
         nAcertos[i]=1;
      }
   }

   return pAcertos;
}
```

2.2.4. Imprimir estado da forca

A impressão do estado da força foi implementada através de função que utiliza estrutura de seleção SWITCH para cada número de erros cometido pelo jogador, de modo que recebe como parâmetro apenas o numero de erro, gerando a partir dessa informação o respectivo desenho:

```
int imprimeForca (int nivel)
{
   switch (nivel)
   {
         case 3:
                  printf("     xxxxxxxxxxxx\n");
                  printf("     ||  //     |\n");
                  printf("     || //     ( )\n");
                  printf("     ||//      _|_\n");
                  printf("     ||/      /[X]\n");
                  printf("     ||      / [X]\n");
                  printf("     ||        \n");
                  printf("     ||        \n");
                  printf("     ||        \n");
                  printf("     ||        \n");
                  printf("     ||_________________.\n");
                  printf("     || 1 2 3         |\n");
                  printf("     ||XXXXXXXXXXXXXXX|\n");
         break;
```

2.2.5. Leitura de lista de palavra e sorteio

A leitura da lista de palavra acessa um arquivo do tipo texto (.txt) para retornar cada uma das palavras para uma matriz de caracteres. O sorteio utiliza funções de bibliotecas pré-existentes: srand(time(0)); int sorteio = rand() % 30; sendo a primeira responsável inicializar o gerador de números aleatórios com o valor da função time(NULL), calculado como sendo o total de segundos passados desde 1 de janeiro de 1970 até a data atual, e a segunda função produz um valor aleatório na faixa entre 0 e a constante RAND_MAX.

```
char * sorteiaPalavra()
{
   //leitura de arquivo de palavras para sorteio
   FILE * arquivo = NULL;
   char palavras[30][30]={};
   char texto[30]="\0";
   int i=0, j=0;

   arquivo = fopen("listaPalavras.txt", "r");
   if ((arquivo = fopen("listaPalavras.txt", "r"))==NULL)
   {
      puts("arquivo não pode ser aberto\n");
   } else
   {
      for (i=0; i<30; i++)
      {
         fscanf (arquivo, "%s", texto);
         strcpy (palavras[i], texto);
         //puts(palavras[i]);
      }
   }

   fclose(arquivo);
   //chave para sorteio de 0 a 29
   srand(time(0));
   int sorteio = rand() % 30;
   char palavraSorteada[30]="\0";
   char *palavra = palavraSorteada;
   strcpy (palavraSorteada, palavras[sorteio]);
   return palavra;
}
```

2.2.6. Implementação de biblioteca própria

A implementação de biblioteca própria para funções foi determinada após início das atividades, de forma que não seria possível definir quais funções integrariam a biblioteca uma vez que o código-fonte principal estava sendo desenvolvido por pessoas integrantes distintos.

Dessa forma, optou-se pode colocar em biblioteca própria, a saber: #include "figurasForca.h", aquelas funções responsáveis por imprimir figuras em formato de texto, como: imprimeCranio (void) imprimeTrofeu (void) imprimeFogos (void) imprimeRanking() imprimeForca (int nivel) telaInical() telaMenu() imprimeCreditos (), que juntas representavam mais de 300 linhas de código.

2.3 Desenvolvimento das atividades

As atividades orginalmente previstas foram redistribuídas em razão da entrada de um novo membro no decurso do trabalho. A partir de entregas parciais, foi verificada a necessidade também de redistribuição de tarefas até então não realizadas por seu responsável conforme tabela abaixo:

Tabela 02: Atividades executadas.

TAREFA	RESPONSÁVEL	CONCLUSÃO	REDISTRIBUIÇÃO
Definição das regras do jogo	CENTURION	SIM	desnecessária
código fonte: layout	CENTURION	SIM	desnecessária
código fonte: jogo individual: lógica do jogo	CENTURION	SIM	desnecessária
código fonte: jogo individual: criação de lista de palavras para o jogo e sistema de seleção aleatória palavra para o jogo	CARAÚBAS	NÃO	CENTURION E CASTRO[1]
código fonte: jogo multijogador	FRANCA	SIM	desnecessária

[1] CASTRO adicionado como integrante do grupo em 24/11/2020 após a divisão de atividades razão pela qual não consta no planejamento inicial; ficando, dessa forma, responsável por auxiliar os demais naquilo que fosse solicitado em razão de sua entrada tardia.

código fonte: sistema de pontuação	CARAÚBAS	NÃO	CASTRO
código fonte: revisão e teste	FRANCA	SIM	FRANCA E CASTRO
código fonte: elaboração de biblioteca externa[2]	TODOS	N.D.	CENTURION
elaboração do relatório	CENTURION	SIM	desnecessária
elaboração do vídeo de apresentação	CARAÚBAS	NÃO	CENTURION

3 CONSIDERAÇÕES FINAIS

O foco do presente trabalho acadêmico é a documentação do processo de desenvolvimento em equipe de aplicação em linguagem C, representada pelo Jogo da Forca. Relata-se inicialmente o planejamento e distribuição das atividades, que acabaram sendo cumpridos em parte, havendo necessidade de redistribuição e complementação dos trabalhos realizados. O código foi estruturado de modo que a função principal permanecesse o mais simples possível, de modo que diversas funções acessórias fossem evocadas conforme sua necessidade.

Como melhorias para o código existente pode-se indicar prioritariamente a desconsideração de caracteres maiúsculos e minúsculos como distintos quando da sugestão de letra existente pelo jogador e de palavras compostas e frases.

Sugestões de trabalhos futuros e aperfeiçoamento do jogo contemplam: a) a formulação de uma nova modalidade de jogo multijogadores, em que possam jogar mais de duas pessoas; b) aspectos estéticos como layout colorido e utilização de imagens diversas; c) utilização de trilha sonora simplificada de modo a tornar o jogo mais divertido; d) implementação de sistema de ranqueamento complexo, com mais posições e histórico de jogos; e) possiblidade do usuário adicionar palavras novas a lista de palavras pré-existente para o modo de jogo individual; f) formulação de coletânea de jogos; dentre outros.

[2] Tarefa criada posteriormente a distribuição das atividades.

Cabe, finalmente, reforçar que, longe de esgotar o trabalho desenvolvido, seja pelas modalidades de jogos, seja pela capacidade do programa de lidar com dados inseridos pelo usuário, ou ainda pela forma como a lógica do programa atua, cada solução adotada pela equipe está sujeita a revisão ou reformulação em razão da própria diversidade de soluções existentes para um mesmo problema; de forma que o código desenvolvido representa apenas um primeiro estado da aplicação.

REFERÊNCIAS

Associação Brasileira de Normas Técnicas - ABNT. **NBR 14724 – Informação e documentação – Trabalhos acadêmicos – Apresentação**. Rio de Janeiro: ABNT, 2011 (errata 2012).

BLACK, Ashlyn. **C reference cheat sheet**. Disponível em: <https://cheatography.com/ashlyn-black/cheat-sheets/c-reference/>. Acesso em 28 de novembro de 2020.

BRITANNICA. **Hanging: capital punishment**. Disponível em: <https://www.britannica.com/topic/hanging>. Acesso em 28 de novembro de 2020.

DAMAS, Luís. **Linguagem C** (trad. RIBEIRO, João et al). 10.ed. Rio de Janeiro: LTC, 2016.

DEITEL, P. J. **C: como programar** (trad. VIEIRA, Daniel). 6.ed. São Paulo: Pearson Prentice Hall, 201.

FLORENCE, Maximus. **From history to history class: the origin of classroom games**. Disponível em: < https://chschipper.com/2019/12/from-history-to-history-class-the-origin-of-classroom-games/>. Acesso em 28 de novembro de 2020.

GOMME, Alice Bertha. **The traditional games of England, Scotland, and Ireland**. em A dictionary of British fok-lore. Vol.1. Londres: DAVID NUTT, 1984. Disponível em: <https://books.google.com.br/books?redir_esc=y&hl=pt-BR&id=ddRwAAAAIAAJ&q=hangman#v=onepage&q=hangman&f=false>. Acesso em 28 de novembro de 2020.

SANTOS, Marcela Gonçalves dos; SARAIVA, Maurício de Oliveira; GONÇALVES, Priscila de Fátima. **Linguagem de programação**. Porto Alegre: SAGAH, 2018.

Terra. Empreendedorismo. **De camelô a bilionário, conheça a trajetória de Silvio Santos**. Disponível em: <https://www.terra.com.br/economia/vida-de-empresario/de-camelo-a-bilionario-conheca-trajetoria-de-silvio-santos,f79e6b9dcf37a410VgnVCM4000009bcceb0aRCRD.html>. Acesso em 28 de novembro de 2020.

Standard C/C++ reference. Disponível em <http://www.cplusplus.com/reference/>. Acesso em 23 de novembro de 2020.

APÊNDICE A – CÓDIGO FONTE PRINCIPAL

Os códigos-fonte abaixo apensos podem ser acessados eletronicamente a partir do repositório: <https://github.com/vncenturion/jogoDaForca>.

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <time.h>
#include <locale.h>
#include "figurasForca.h"


void flush_in()
{
    int ch;
    while( (ch = fgetc(stdin)) != EOF && ch != '\n' ){}
}

int * buscaAcertos(char palavra[30], char letra, int acertos[30])
{
    int nAcertos[30];
    int *pAcertos = nAcertos;
    int i=0;

    for (i=0; i<30; i++)
    {
        nAcertos[i]=acertos[i];
    }

    for (i=0; i<30; i++)
    {
        if (palavra[i]==letra)
        {
            nAcertos[i]=1;
        }
    }

    return pAcertos;
}

int buscaPontos(char palavra[30], char letra)
{
    int i;
    int contaPonto=0;
```

```
    for (i=0; i<strlen(palavra); i++)
    {
        if (palavra[i]==letra)
        {
            contaPonto+=10;
        }
    }

    return contaPonto;
}

int buscaErro(char palavra[30], char letra)
{
    int i;
    int contaErro=0;
    int nLetra = strlen(palavra);

    for (i=0; i<nLetra; i++)
    {

        if (palavra[i]!=letra)
        {
            contaErro +=1;
        }

    }

    if (contaErro==nLetra)
    {
        return 1;
    }
    else
    {
        return 0;
    }

}

int imprimeLetraErrada (char palavra[7])
{
    int nLetras = strlen(palavra);
    int i;

    printf("     Letras erradas: ");

    for (i=0; i<(nLetras-1);i++)
    {
        printf("%c, ", palavra[i]);
```

```
    }

    if (nLetras>0)
    {
        printf("%c.", palavra[i]);
    }
    puts(" ");
    puts(" ");

    return 1;
}

int imprimePalavra (char palavra[30], int acertos[30])
{
    int nLetras = strlen(palavra);
    int i;

    printf("     ");

    for (i=0; i<nLetras;i++)
    {
        if (acertos[i]==1)
        {
            printf("[%c] ", palavra[i]);
        }
        else
        {
            printf("[_] ");
        }
    }
    puts("");
    return 1;
}

char * sorteiaPalavra()
{
    //leitura de arquivo de palavras para sorteio
    FILE * arquivo = NULL;
    char palavras[30][30]={};
    char texto[30]="\0";
    int i=0, j=0;

    arquivo = fopen("listaPalavras.txt", "r");
    if ((arquivo = fopen("listaPalavras.txt", "r"))==NULL)
    {
        puts("arquivo não pode ser aberto\n");
    } else
    {
        for (i=0; i<30; i++)
```

```
        {
            fscanf (arquivo, "%s", texto);
            strcpy (palavras[i], texto);
            //puts(palavras[i]);
        }
    }

    fclose(arquivo);
    //chave para sorteio de 0 a 29
    srand(time(0));
    int sorteio = rand() % 30;
    char palavraSorteada[30]="\0";
    char *palavra = palavraSorteada;
    strcpy (palavraSorteada, palavras[sorteio]);
    return palavra;
}

int jogoMultiplayer (char jogadorPalavra[30])
{
    int acertoTotal[30] = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0};
    int * nAcertos;
    int i, erros=0, totalPontos = 0, maxPontos = 0, parcial = 0,tamPalavra=0,
controle=0, pontosJogador = 0;
    char letrasErradas[7] = "\0";
    char letra=' ';
    char texto[30]="\0";
    char letrasDigitadas[30] = "\0";
    int contadorLetras = 0;

    /*
    char *pNomeJogador = (char*) calloc(1,sizeof(strlen(jogadorPalavra)));

    puts("entrou na funcao!");
    puts(pNomeJogador);
    strcpy(pNomeJogador, jogadorPalavra);
    puts(pNomeJogador);
    printf("string de pNomeJogador: %s\nstrlen de pNomeJogador: %lu\nsizeOf pN
omeJogador: %lu\nsizeOf *pNomeJogador: %lu\nsizeOf &pNomeJogador: %lu\n\n", pN
omeJogador, strlen(pNomeJogador), (sizeof(pNomeJogador)),(sizeof(*pNomeJogador
)),(sizeof(&pNomeJogador)));

    for (i = 0; i < strlen(jogadorPalavra); i++)
    {
        printf("Endereco de p%i = %p = %p | valor de p%i = %c\n", i, (pNomeJog
ador+i), &pNomeJogador[i], i, *(pNomeJogador+i));
    }
    */
```

```
printf("     %s, digite a palavra a ser adivinhada: ", jogadorPalavra);
scanf("%[ -~]", texto);
flush_in();
char *palavra = (char*) calloc(1,sizeof(strlen(texto)));
if (palavra == NULL)//verifica se alocacao funcionou
{
    printf("\nErro de alocacao de memoria\n");
}
strcpy(palavra, texto);
tamPalavra = strlen (palavra);
maxPontos = tamPalavra * 10;

do
{
    system("cls");
    printf("     Palavra criada por: %s\n", jogadorPalavra);
    printf("     max pontos da palavra: %d\n", maxPontos);
    printf("     total de pontos obtidos: %d\n", totalPontos);
    //printf("     %s", palavra);
    imprimeForca(erros);
    imprimeLetraErrada(letrasErradas);
    imprimePalavra(palavra, acertoTotal);
    puts("");

    do
    {
        printf("     Digite uma letra da palavra: ");
        scanf(" %c", &letra);
        flush_in();

        for (i=0, controle=0; i<strlen(letrasDigitadas); i++)
        {
            if (letrasDigitadas[i]==letra)
            {
                controle+=1;
            }
        }

        if (controle>=1)
        {
            printf("     Letra já digitada!\n");
        }

    } while (controle!=0);
    //correcao de codigo anterior com strcat
    letrasDigitadas[contadorLetras] = letra;
    contadorLetras++;
    nAcertos = buscaAcertos(palavra, letra, acertoTotal);
```

```
    for (i=0; i<30; i++)
    {
        acertoTotal[i]=nAcertos[i];
    }

    if (buscaErro(palavra, letra)==1)
    {
        letrasErradas[erros] = letra;
        erros += 1;
    }

    //erros += buscaErro(palavra, letra);
    totalPontos += buscaPontos(palavra, letra);

} while ((erros<7)&&(totalPontos<maxPontos));

system("cls");
printf("     max pontos: %d\n", maxPontos);
printf("     total de pontos: %d\n", totalPontos);
imprimeForca(erros);
imprimeLetraErrada(letrasErradas);
imprimePalavra(palavra, acertoTotal);
puts("");
puts("\n     FIM DO JOGO!");
puts("");
puts("");
puts("");

parcial = totalPontos-(erros*10);
if (parcial<=0)
{
    parcial=0;
}
pontosJogador+=parcial;

if (erros>=7)
{
    puts("");
    imprimeCranio();
    puts("");
    puts("                  VOCÊ PERDEU!");
  printf("        TOTAL DE PONTOS OBTIDOS: %d\n", totalPontos);
  printf("        TOTAL DE PONTOS PERDIDOS: %d\n", (erros*10));
    puts("");
    puts("      <PRESSIONE ENTER PARA CONTINUAR>");
    getchar();
}

if (totalPontos>=maxPontos)
```

```
    {
        imprimeTrofeu();
        puts("");
        puts("           PARABENS, VOCÊ VENCEU!");
      printf("         TOTAL DE PONTOS OBTIDOS: %d\n", totalPontos);
      printf("         TOTAL DE PONTOS PERDIDOS: %d\n", (erros*10));
        puts("");
        puts("      <PRESSIONE ENTER PARA CONTINUAR>");
        getchar();
    }

    return pontosJogador;
}

int jogoIndividual ()
{
    int pontosJogador = 0, venceu=1;

    do
    {
        int acertoTotal[30] = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0};
        int * nAcertos;
        int i, erros=0, totalPontos = 0, maxPontos = 0, parcial = 0,tamPalavra
=0, controle=0;
        char letrasErradas[7] = "\0";
        char letra=' ';
        char palavra[30]="\0";
        char letrasDigitadas[30] = "\0";
        int contadorLetras = 0;

        strcpy(palavra, sorteiaPalavra());
        tamPalavra = strlen (palavra);
        maxPontos = tamPalavra * 10;

        do
        {

            system("cls");
            printf("     max pontos da palavra: %d\n", maxPontos);
            printf("     total de pontos obtidos: %d\n", totalPontos);
            //printf("     %s", palavra);
            imprimeForca(erros);
            imprimeLetraErrada(letrasErradas);
            imprimePalavra(palavra, acertoTotal);
            puts("");

            do
            {
```

```
        printf("     Digite uma letra da palavra: ");
        scanf(" %c", &letra);
        flush_in();

        //verifica se a letra foi digitada anteriormente
        for (i=0, controle=0; i<strlen(letrasDigitadas); i++)
        {
            if (letrasDigitadas[i]==letra)
            {
                controle+=1;
            }
        }

        if (controle>=1)
        {
            printf("     Letra já digitada!\n");
        }

    } while (controle!=0);
    //correcao de codigo anterior com strcat
    letrasDigitadas[contadorLetras] = letra;
    contadorLetras++;

    nAcertos = buscaAcertos(palavra, letra, acertoTotal);

    for (i=0; i<30; i++)
    {
        acertoTotal[i]=nAcertos[i];
    }
    if (buscaErro(palavra, letra)==1)
    {
        letrasErradas[erros] = letra;
        erros += 1;
    }

    //erros += buscaErro(palavra, letra);
    totalPontos += buscaPontos(palavra, letra);

} while ((erros<7)&&(totalPontos<maxPontos));

system("cls");
printf("     max pontos: %d\n", maxPontos);
printf("     total de pontos: %d\n", totalPontos);
imprimeForca(erros);
imprimeLetraErrada(letrasErradas);
imprimePalavra(palavra, acertoTotal);
puts("");
puts("\n     FIM DO JOGO!");
```

```
        puts("");
        puts("");
        puts("");

        parcial = totalPontos-(erros*10);
        if (parcial<=0)
        {
            parcial=0;
        }
        pontosJogador+=parcial;

        if (erros>=7)
        {
            venceu=0;
            puts("");
            imprimeCranio();
            puts("");
            puts("           VOCÊ PERDEU!");
            printf("        TOTAL DE PONTOS OBTIDOS: %d\n", pontosJogador);
            puts("");
            puts("     <PRESSIONE ENTER PARA VOLTAR AO MENU>");
            getchar();
        }

        if (totalPontos>=maxPontos)
        {
            venceu=1;
            imprimeTrofeu();
            puts("");
            puts("          PARABENS, VOCÊ VENCEU!");
            printf("        TOTAL DE PONTOS OBTIDOS: %d\n", pontosJogador);
            puts("");
            puts("     <PRESSIONE ENTER PARA CONTINUAR>");
            getchar();
        }

    } while (venceu==1);

    return pontosJogador;
}

struct Recordes{

   char nome1[20];
   int pontuacao1;

   char nome2[20];
   int pontuacao2;
```

```
   char nome3[20];
   int pontuacao3;

};

void gravarPontuacao(char novoNome[30], int novaPontuacao){

    FILE *Novo_Recorde; //ponteiro para o arquivo

    struct Recordes novo;

    if ((Novo_Recorde = fopen("Recordes", "r+")) == NULL){ //Tenta abrir um ar
quivo para ler e atualizar os recordes (modo "r+")

        Novo_Recorde = fopen("Recordes", "w"); //Se não houver, abre um novo a
rquivo para gravação (modo "w")



        strcpy(novo.nome1, novoNome); // atribui a primeira pontuacao do jogo

        novo.pontuacao1 = novaPontuacao; //inicializa com zero o recorde



        strcpy(novo.nome2, "*****"); // inicializa os demais nomes com asteris
cos

        novo.pontuacao2 = 0; // inicializa os demais recordes com zero



        strcpy(novo.nome3, "*****"); // inicializa os demais nomes com asteris
cos

        novo.pontuacao3 = 0; // os demais recordes com zero


    } else{ //Caso já exista o arquivo

        fseek(Novo_Recorde, 0, SEEK_SET); //posiciona no início

        fread(&novo, sizeof(struct Recordes), 1, Novo_Recorde); //lê o arquivo


        if(novaPontuacao > novo.pontuacao1){

            // passando pontuacao do 2 lugar para o 3 lugar
            strcpy(novo.nome3, novo.nome2);
```

```
            novo.pontuacao3 = novo.pontuacao2;

            // passando pontuacao do 1 lugar para o 2 lugar
            strcpy(novo.nome2, novo.nome1);
            novo.pontuacao2 = novo.pontuacao1;

            strcpy(novo.nome1, novoNome);
            novo.pontuacao1 = novaPontuacao;

        }else if(novaPontuacao > novo.pontuacao2){

            // passando pontuacao do 2 lugar para o 3 lugar
            strcpy(novo.nome3, novo.nome2);
            novo.pontuacao3 = novo.pontuacao2;

            strcpy(novo.nome2, novoNome);
            novo.pontuacao2 = novaPontuacao;

        } else if (novaPontuacao > novo.pontuacao3){

            strcpy(novo.nome3, novoNome);
            novo.pontuacao3 = novaPontuacao;

        }

    }

    fseek(Novo_Recorde, 0, SEEK_SET); //Posiciona no início do arquivo

    fwrite(&novo, sizeof(struct Recordes), 1, Novo_Recorde); //faz a gravação
do novo recorde

    fclose(Novo_Recorde); //fecha o arquivo


}

void mostrarRanking(){

    FILE *Novo_Recorde;

    struct Recordes novo;

    system("cls");
    imprimeRanking ();

    if((Novo_Recorde = fopen("Recordes", "r")) == NULL){ //Tenta abrir o arqui
vo "Recordes" para ler os recordes (modo "r")
```

```
            printf("\n\n\t\t       NENHUM RECORDE! SEJA O PRIMEIRO! ");

        } else
        {
            fseek(Novo_Recorde, 0, SEEK_SET);
            fread(&novo, sizeof(struct Recordes), 1, Novo_Recorde);

            printf("\n\n\n\n\n\t\t\t     #1 LUGAR : ");
            printf("  %s", novo.nome1);
            printf("\t  %d", novo.pontuacao1);

            printf("\n\n\n\t\t\t     #2 LUGAR : ");
            printf("  %s", novo.nome2);
            printf("\t  %d", novo.pontuacao2);

            printf("\n\n\n\t\t\t     #3 LUGAR : ");
            printf("  %s", novo.nome3);
            printf("\t  %d", novo.pontuacao3);

            fclose(Novo_Recorde);
        }

        getchar();

    }

    int selecionaJogo (int jogo)
    {
        int sair=0;
        char confirma='n';
        char nomeJogador[30]="\0";
        char nomeJogador1[30]="\0";
        char nomeJogador2[30]="\0";
        int pontoJogador1=0, pontoJogador2=0;
        int pontoJogador=0;
        char *jogadorVencedor = nomeJogador;

        struct pontuacao
        {
            char nomeRanking[30];
            int pontosRanking;
        } resPontuacao, resSingle, resMultiplayer1, resMultiplayer2;

        switch (jogo)
        {

            case 1:
                system("cls");
                puts("");
```

```
            printf("\n     JOGO INDIVIDUAL ( 1 VS AI )");
            puts("");
            imprimeForca(0);
            puts("");
            printf("     Digite seu nome: ");
            scanf("%[ -~]", resSingle.nomeRanking);
            flush_in();
            if (strlen(resSingle.nomeRanking)==0)//verifica se o jogador inser
iu nome
            {
                strcpy(resSingle.nomeRanking, "Jogador Individual");
            }
            resSingle.pontosRanking=jogoIndividual();
            printf("JOGADOR: %s fez %d PONTOS!",resSingle.nomeRanking,resSingl
e.pontosRanking);

            gravarPontuacao(resSingle.nomeRanking, resSingle.pontosRanking);
            getchar();
            sair=0;
            return sair;
        break;
        case 2:

            system("cls");
            getchar();
            puts("");
            printf("\n     MULTI JOGADOR ( 1 VS 1 )");
            puts("");
            imprimeForca(0);
            puts("");
            printf("     Jogador n.1, digite seu nome: ");
            scanf("%[ -~]", resMultiplayer1.nomeRanking);
            flush_in();
            if (strlen(resMultiplayer1.nomeRanking)==0)//verifica se o jogador
 inseriu nome
            {
                strcpy(resMultiplayer1.nomeRanking, "Jogador #1");
            }
            puts("");
            resMultiplayer1.pontosRanking=jogoMultiplayer(resMultiplayer1.nome
Ranking);
            //segundapartida
            system("cls");
            getchar();
            puts("");
            printf("\n     MULTI JOGADOR ( 1 VS 1 )");
            puts("");
            imprimeForca(0);
            puts("");
```

```
            printf("     Jogador n.2, digite seu nome: ");
            scanf("%[ -~]", resMultiplayer2.nomeRanking);
            flush_in();
            if (strlen(resMultiplayer2.nomeRanking)==0)//verifica se o jogador
 inseriu nome
            {
                strcpy(resMultiplayer2.nomeRanking, "Jogador #2");
            }
            puts("");
            resMultiplayer2.pontosRanking=jogoMultiplayer (resMultiplayer2.nom
eRanking);

            if (resMultiplayer1.pontosRanking<resMultiplayer2.pontosRanking)
            {
                system("cls");
                puts("");
                printf("\n     MULTI JOGADOR ( 1 VS 1 )");
                puts("");
                puts("");
                imprimeFogos();
                puts("");
                printf("\n     %s VENCEU %s EM ( 1 VS 1 )\n     por %d a %d PO
NTOS",resMultiplayer2.nomeRanking, resMultiplayer1.nomeRanking,resMultiplayer2
.pontosRanking, resMultiplayer1.pontosRanking);
                puts("");
                getchar();

            } else if (resMultiplayer1.pontosRanking>resMultiplayer2.pontosRan
king)
            {
                system("cls");
                puts("");
                printf("\n     MULTI JOGADOR ( 1 VS 1 )");
                puts("");
                puts("");
                imprimeFogos();
                puts("");
                printf("\n     %s VENCEU %s EM ( 1 VS 1 )\n     por %d a %d PO
NTOS",resMultiplayer1.nomeRanking, resMultiplayer2.nomeRanking,resMultiplayer1
.pontosRanking, resMultiplayer2.pontosRanking);
                puts("");
                getchar();

            } else if (resMultiplayer1.pontosRanking==resMultiplayer2.pontosRa
nking)
            {
                system("cls");
                puts("");
                printf("\n     MULTI JOGADOR ( 1 VS 1 )");
```

```
                puts("");
                puts("");
                imprimeFogos();
                puts("");
                printf("\n     EMPATE de %s e %s\n     por %d e %d PONTOS",res
Multiplayer1.nomeRanking, resMultiplayer2.nomeRanking,resMultiplayer1.pontosRa
nking, resMultiplayer2.pontosRanking);
                puts("");
                getchar();

            }
            puts("     <PRESSIONE ENTER PARA VOLTAR AO MENU DE SELECAO>");
            getchar();
            sair=0;
            return sair;

        break;
        case 3:
            mostrarRanking();
            sair=0
            return sair;

        break;
        case 4:
            printf("\n     Deseja sair? (s ou n): ");
            scanf(" %c", &confirma);
            flush_in();

            if (confirma=='s')
            {
                sair=1;
            }
            else
            {
                sair=0;
            }
            return sair;
            break;
        default:
            printf("\n     Opção inválida!");
            getchar();
            getchar();
            sair=0;
            return sair;
        break;
    }

}
```

```c
int main(void)
{
    setlocale(LC_ALL,"Portuguese");
    int tipoJogo=0, sair=0;
    telaInical();

    do
    {
        system("cls");
        telaMenu();

        printf("\n\n\n     Selecione o modo de jogo: ");
        scanf("%d", &tipoJogo);
        flush_in();

        sair=selecionaJogo(tipoJogo);

    } while (sair==0);

    system("cls");
    imprimeCreditos();

    return 0;
}
```

APÊNDICE B – CÓDIGO FONTE BIBLIOTECA

```
#ifndef figurasForca_h
#define figurasForca_h

int nivel;

int imprimeCranio (void)
{
    printf("              .--------------.\n");
    printf("            /                \\\n");
    printf("           /                  \\\n");
    printf("          |  \\\\ //    \\\\ //   |\n");
    printf("          |    /        /     |\n");
    printf("          |  // \\\\    // \\\\   |\n");
    printf("          |                    |\n");
    printf("          \\__      XXX       __/\n");
    printf("            |\\     XXX     /|\n");
    printf("            | |           | |\n");
    printf("            | |[][][][][][] |\n");
    printf("             \\ [][][][][][]_/\n");
    printf("              \\___________/\n");
    return 0;
}

int imprimeTrofeu (void)
{
    printf("               __________\n");
    printf("             '.=========.'\n");
    printf("             .-\\:      /-.\n");
    printf("            |( |:.     | )|\n");
    printf("             '-|:.     |-'\n");
    printf("               \\::.    /\n");
    printf("                '::. .' \n");
    printf("                  ) (   \n");
    printf("                _.' '._ \n");
    printf("               '======='\n");
    return 1;
}

int imprimeFogos (void)
{
    printf("                                        .''.\n");
    printf("           .''.      .        *''*    :_\\/_:     .\n");
    printf("          :_\\/_:   _\\(/_  .:.*_\\/_*   : /\\ :  .'.:.'.\n");
    printf("      .''.: /\\ :   ./)\\   ':'* /\\ * :  '..'.  -=:o:=-\n");
```

```
    printf("      :_\\/_:'.:::.    ' *''*    * '.\\'/.' _\\(/_'.':'.'\n");
    printf("      : /\\ : :::::     *_\\/_*     -= o =-  /)\\    '  *\n");
    printf("       '..'  ':::'     * /\\ *     .'/.\\'.   '\n");
    printf("           *           *..*         :\n");
    printf("            *           *\n");
    printf("             *          *\n");
    return 1;
}

int imprimeRanking ()
{
    puts("");
    puts("");
    puts("                                                \\0/");
    puts("                                          \\0      |");
    puts("                                           |-     / \\      0");
    puts("                                          / \\   %%%%%%%%   -|-");
    puts("                                       %%%%%%%% %% #1 %%%   / \\");
    puts("                                       %%%%%%%% %%%%%%%% %%%%%%%% ")
;
    puts("");
    puts("             @@@@@@@@      @@@@     @@@     @@@ @@@   @@@@  @@@ @@@@
    @@@   @@@@@@@@@" );
    puts("             @@@   @@@    @@@@@@    @@@@@   @@@ @@@ @@@@    @@@ @@@@
@   @@@ @@@@        ");
    puts("             @@@@@@@@@@   @@@  @@@   @@@*@@@ @@@ @@@@@@      @@@ @@@
@@@ @@@ @@@   @@@@@@");
    puts("             @@@ @@@,   @@@@@@@@@@  @@@  @@@@@@ @@@ @@@@    @@@ @@@
 @@@@@@ @@@@     @@@");
    puts("             @@@   @@@ @@@      @@@ @@@    @@@@ @@@   @@@@  @@@ @@@
   @@@@   @@@@@@@@@" );
    puts("");
    puts("");
    return 0;
}

int imprimeForca (int nivel)
{
    switch (nivel)
    {
        case 7:
            puts("");
            puts("");
            printf("     xxxxxxxxxxxx\n");
            printf("     ||  //     |\n");
            printf("     || //     ( )\n");
            printf("     ||//      _|_\n");
            printf("     ||/      /[X]\\\n");
            printf("     ||      / [X] \\\n");
```

```
    printf("     ||        ( )\n");
    printf("     ||       /   \\\n");
    printf("     ||      /     \\\n");
    printf("     ||              \n");
    printf("     ||_______________.\n");
    printf("     || 1 2 3 4 5 6 7 |\n");
    printf("     ||XXXXXXXXXXXXXXX|\n");
    puts("");
    puts("");
break;
case 6:
    puts("");
    puts("");
    printf("     xxxxxxxxxxxx\n");
    printf("     ||  //     |\n");
    printf("     || //     ( )\n");
    printf("     ||//      _|_\n");
    printf("     ||/      /[X]\\\n");
    printf("     ||      / [X] \\\n");
    printf("     ||        ( )\n");
    printf("     ||       /   \n");
    printf("     ||      /    \n");
    printf("     ||           \n");
    printf("     ||_______________.\n");
    printf("     || 1 2 3 4 5 6   |\n");
    printf("     ||XXXXXXXXXXXXXXX|\n");
    puts("");
    puts("");
break;
case 5:
    puts("");
    puts("");
    printf("     xxxxxxxxxxxx\n");
    printf("     ||  //     |\n");
    printf("     || //     ( )\n");
    printf("     ||//      _|_\n");
    printf("     ||/      /[X]\\\n");
    printf("     ||      / [X] \\\n");
    printf("     ||        ( )\n");
    printf("     ||        \n");
    printf("     ||        \n");
    printf("     ||        \n");
    printf("     ||_______________.\n");
    printf("     || 1 2 3 4 5     |\n");
    printf("     ||XXXXXXXXXXXXXXX|\n");
    puts("");
    puts("");
break;
case 4:
```

```
    puts("");
    puts("");
    printf("     xxxxxxxxxxxx\n");
    printf("     ||  //     |\n");
    printf("     || //     ( )\n");
    printf("     ||//      _|_\n");
    printf("     ||/      /[X]\\\n");
    printf("     ||      / [X] \\\n");
    printf("     ||        \n");
    printf("     ||        \n");
    printf("     ||        \n");
    printf("     ||        \n");
    printf("     ||_______________.\n");
    printf("     || 1 2 3 4       |\n");
    printf("     ||XXXXXXXXXXXXXXX|\n");
    puts("");
    puts("");
break;
case 3:
    puts("");
    puts("");
    printf("     xxxxxxxxxxxx\n");
    printf("     ||  //     |\n");
    printf("     || //     ( )\n");
    printf("     ||//      _|_\n");
    printf("     ||/      /[X]\n");
    printf("     ||      / [X]\n");
    printf("     ||        \n");
    printf("     ||        \n");
    printf("     ||        \n");
    printf("     ||        \n");
    printf("     ||_______________.\n");
    printf("     || 1 2 3         |\n");
    printf("     ||XXXXXXXXXXXXXXX|\n");
    puts("");
    puts("");
break;
case 2:
    puts("");
    puts("");
    printf("     xxxxxxxxxxxx\n");
    printf("     ||  //     |\n");
    printf("     || //     ( )\n");
    printf("     ||//      _|_\n");
    printf("     ||/       [X]\n");
    printf("     ||        [X]\n");
    printf("     ||        \n");
    printf("     ||        \n");
    printf("     ||        \n");
```

```
    printf("     ||        \n");
    printf("     ||_______________.\n");
    printf("     || 1 2           |\n");
    printf("     ||XXXXXXXXXXXXXX|\n");
    puts("");
    puts("");
break;
case 1:
    puts("");
    puts("");
    printf("     xxxxxxxxxxxx\n");
    printf("     ||  //     |\n");
    printf("     || //     ( )\n");
    printf("     ||//        \n");
    printf("     ||/       \n");
    printf("     ||        \n");
    printf("     ||        \n");
    printf("     ||        \n");
    printf("     ||        \n");
    printf("     ||        \n");
    printf("     ||_______________.\n");
    printf("     || 1             |\n");
    printf("     ||XXXXXXXXXXXXXX|\n");
    puts("");
    puts("");
break;
case 0:
    puts("");
    puts("");
    printf("     xxxxxxxxxxxx\n");
    printf("     ||  //     |\n");
    printf("     || //       \n");
    printf("     ||//        \n");
    printf("     ||/       \n");
    printf("     ||        \n");
    printf("     ||        \n");
    printf("     ||        \n");
    printf("     ||        \n");
    printf("     ||        \n");
    printf("     ||_______________.\n");
    printf("     ||               |\n");
    printf("     ||XXXXXXXXXXXXXX|\n");
    puts("");
    puts("");
break;
default:
    printf("--- nivel invalido ---\n");
break;
```

```
    }
    return 1;
}

int telaInical()
{
    system("cls");
    puts("");
    puts("");
    puts("         @@    @@@@@@@@     @@@@@@@@@    @@@@@@@@         @@@@@@@@
     @@@@");
    puts("         @@   @@      @@   @@           @@      @@,       @@     @@@
    .@@ @@");
    puts("         @@  .@@      @@# .@@    @@@@   @       &@@       @@      @@
    @@  /@@");
    puts("         @@   @@      @@   @@       @   @@      @@        @@      @@
   @@@@@@@@@");
    puts("     /@@@@   ,@@@@@@@/     @@@@@@@@/    @@@@@@@/         @@&&@@@@@
  @@      @@.");
    puts("");
    puts("");
    puts("     @@@@@@@@   @@@@@@@    @@@@@@@@@    /@@@@@@@@      @@@.");
    puts("     @@         @@#     @@@  @@@    @@&  @@@           @@ @@");
    puts("     @@@@@@@   @@      .@@  @@@@@@@@@   @@            @@   @@");
    puts("     @@        @@      .@@  @@@  @@*    @@           @@@@@@@@@");
    puts("     @@         @@@@@@@@    @@@   @@@@   @@@@@@@@@  @@@     /@@");
    puts("");
    puts("");
    puts("                                xxxxxxxxxxxx");
    puts("                                ||  //     |");
    puts("                                || //     ( )");
    puts("                                ||//      _|_");
    puts("                                ||/      /[X]\\");
    puts("                                ||      / [X] \\");
    puts("                                ||        ( )");
    puts("                                ||       /   \\");
    puts("                                ||      /     \\");
    puts("                                ||             ");
    puts("                                ||_____________.");
    puts("                                || 1 2 3 4 5 6 7 |");
    puts("                                ||XXXXXXXXXXXXXX|");
    puts("");
    puts("");
    puts("");
    puts("");
    puts("                                <PRESSIONE ENTER>");
    getchar();
    return 0;
}
```

```
int telaMenu()
{
    system("cls");
    puts("");
    puts("");
    puts("         @@    @@@@@@@@     @@@@@@@@@@    @@@@@@@@         @@@@@@@@
     @@@@");
    puts("         @@   @@      @@   @@           @@      @@,       @@     @@@
    .@@ @@");
    puts("         @@  .@@      @@# .@@    @@@@   @       &@@       @@      @@
    @@  /@@");
    puts("         @@   @@      @@   @@       @   @@      @@        @@      @@
   @@@@@@@@@@");
    puts("     /@@@@@   ,@@@@@@@/     @@@@@@@@/    @@@@@@@/         @@&&@@@@@
  @@      @@.");
    puts("");
    puts("");
    puts("     @@@@@@@@   @@@@@@@@    @@@@@@@@@@    /@@@@@@@@      @@@.");
    puts("     @@         @@#     @@@  @@@    @@&  @@@            @@ @@");
    puts("     @@@@@@@   @@      .@@  @@@@@@@@@@   @@            @@   @@");
    puts("     @@        @@      .@@  @@@  @@*    @@           @@@@@@@@@");
    puts("     @@         @@@@@@@@    @@@   @@@@   @@@@@@@@@  @@@     /@@");
    puts("");
    puts("");
    puts("");
    puts("");
    puts("                      MENU (SELECIONE O TIPO DE JOGO)");
    puts("");
    puts("");
    puts("                    1 - JOGO INDIVIDUAL ( 1 VS. PC )");
    puts("                    2 - JOGO MULTI JOGADOR ( 1 VS. 1 )");
    puts("                    3 - PONTUACOES");
    puts("                    4 - SAIR");
    puts("");
    puts("");
    puts("");
    puts("");
    puts("");
    puts("                              UNIPE, 2020.");
    puts("");
    return 1;
}

int imprimeCreditos ()
{
    puts("");
    puts("");
    puts("                                          xxxxxxxx");
```

```
    puts("                                                ||     |");
    puts("                                                ||     0");
    puts("                                                ||    -|-");
    puts("                                                ||    / \\");
    puts("                                                ||");
    puts("                                               XXXXXXXXXXX");
    puts("");
    puts("             *@@@@@@@@ @@@@@@@@  @@@@@@@@ @@@@@@@@   @@@ @@@@@@@@@@
 s@@@@@@@   @@@@@@@@");
    puts("            @@@@        @@@  *@@@ @@@      @@@   @@@@ @@@    @@@
@@@    @@@ @@@");
    puts("            @@@         @@@@@@@@  @@@@@@@@ @@@    @@@ @@@    @@@    @
@@     @@@  @@@@@@@");
    puts("            @@@@        @@@ @@@   @@@      @@@   @@@@ @@@    @@@    @
@@     @@@       @@@");
    puts("             @@@@@@@@@ @@@  @@@@ @@@@@@@@ @@@@@@@@   @@@    @@@
@@@@@@@@@  @@@@@@@@");
    puts("");
    puts("");
    puts("");
    puts("                                   EQUIPE DE DESENVOLVIMENTO");
    puts("");
    puts("                                  Lucas Vinicius Pessoa de Franca");
    puts("                                 Sérgio Marcelino Nóbrega de Castro")
;
    puts("                                 Victor Henrique de Amorim Caraubas")
;
    puts("                                Vinicius Bezerra Cavalcanti Centurion
");
    puts("");
    puts("");
    puts("");
    puts("                                        OBRIGADO POR JOGAR!!!");
    puts("");
    return 0;
}

#endif
```