Versão II -Anamnese

Huandy C. Silva, Letícia V. Santos, Luara V. Perilli, Luís E. Damasceno

Instituto de Matemática e Computação – Universidade Federal de Itajubá (UNIFEI)

Itajubá – MG – Brasil

huandysilva@gmail.com, leticiavsantoos@gmail.com,
luudvperilli@gmail.com,dududamasceno5@gmail.com

Resumo. *Criar um jogo utilizando a lógica e as estruturas de programação. A partir dos aprendizados adquiridos e pesquisas, o jogo "Anamnese" foi criado com o fito de auxiliar na melhor compreensão dos ensinamentos adquiridos, uma vez que todas as pesquisas, práticas e escrita do código foram realizadas pelos discentes que compõem este grupo. Desse modo, o conhecimento obtido e aprofundado, a partir das explorações para execução deste trabalho, auxiliarão em outros estudos, uma vez que a compreensão desse conteúdo acarreta impactos positivos na vida dos indivíduos e do corpo social no qual estão inseridos.*

Palavras-chaves: jogo, acessibilidade, cronograma, desenvolvimento.

1. INTRODUÇÃO

No decorrer da disciplina de Fundamentos de Programação (XDES01), ministrada pela Profª. Drª. Elisa de Cassia Silva Rodrigues, para os ingressantes do curso de Sistemas de Informação, na Universidade Federal de Itajubá (UNIFEI), no primeiro semestre do ano de 2022, foram iniciados estudos acerca dos fundamentos de programação. Bem como: lógica de programação, estrutura sequencial, condicional e de repetição, vetores e matrizes, registro, entre outros.

Nessa perspectiva, foi proposta a atividade em questão a fim de que os educandos coloquem em prática as teorias que foram lecionadas. Além disso, devido ao fato de ser uma pesquisa e programação em grupo, o trabalho também incentiva o desenvolvimento das soft skills necessárias no meio laboral, o qual seremos inseridos com o decorrer dos anos na universidade.

Neste documento, portanto, serão relatados todos os passos e implementações realizadas durante a execução do desenvolvimento do jogo, o qual é um jogo da memória numérica, intitulado *Anamnese*. Ademais, um cronograma foi criado com o intuito de facilitar a compreensão da ordem em que as atividades foram realizadas. Por fim, esse documento e o código em questão são referentes a "Versão II", sendo, desse modo, a versão final do jogo.

2. OBJETIVOS DO JOGO

Um dos principais objetivos do jogo *Anamnese* é desenvolver a capacidade de foco e concentração, uma vez que é necessário deixar de lado as distrações para que a sequência numérica seja corretamente reproduzida e o jogador possa desenvolver sua capacidade de memorização. Além disso, o jogo também desenvolve a autoconfiança quando o indivíduo memoriza e acerta a sequência correta.

Ainda, o jogo também tem por objetivo estimular a memória, mantendo o jogador atento ao raciocínio. Cabe ressaltar que o aperfeiçoamento da capacidade de memorização traz inúmeros benefícios para a vida acadêmica, visto que exercita as habilidades de raciocínio e interpretação textual, permitindo a absorção de novos conhecimentos.

3. REGRAS DO JOGO

No jogo *Anamnese*, o jogador deve manter-se atento à sequência de números mostrada e, em seguida, reproduzi-la na ordem correta. Caso o jogador cometa um equívoco em relação a um dos constituintes da sequência, o jogo termina e pode ser reiniciado. É importante lembrar que a quantidade de números mostrada é escolhida pelo próprio jogador, conforme o jogador acerta a sequência ele aumenta a sua pontuação.

4. CRONOGRAMA

DATA	PROCESSO	DETALHAMENTO
Até 24/05/2022.	Definir qual jogo será criado.	O grupo se reuniu por meio de chamadas de vídeo no aplicativo *Google Meet* e pessoalmente, junto à professora, para que fosse definido um jogo acessível e de apenas um jogador. O jogo escolhido foi o jogo da memória numérica, o qual o grupo optou pelo nome de *Anamnese*.
Até 25/05/2022.	Criar um menu inicial para o jogo.	O grupo se reuniu durante a monitoria, para criar uma interface inicial, a qual contém as opções: Jogar, Ranking, Instruções, Créditos e Sair. O detalhamento do menu também foi criado, realizando a escrita das instruções e dos créditos.
Até 25/06/2022	Modularizar o jogo.	O grupo realizou a modularização do jogo, por meio de chamadas de vídeo, e criou funções e uma biblioteca relacionada às opções do menu.

Até 18/07/2022	Criar o jogo.	O grupo se reuniu por chamadas de vídeo e pessoalmente, para criar a funcionalidade do jogo, bem como a criação de biblioteca jogo, que armazena as funções relacionadas ao jogo.
Até 18/07/2022	Criar o ranking do jogo.	Através de reuniões por chamadas de vídeo, o grupo criou o ranking do jogo, para exibir a pontuação dos jogadores.

5. DOCUMENTAÇÃO

5.1 Definição do Jogo

A equipe juntou-se por chamada de vídeo para afunilar as ideias e estabelecer algumas opções de jogos a fim de iniciar o projeto. Após a reunião, as sugestões apresentadas por cada integrante do grupo foram levadas à Profª. Drª. Elisa de Cássia Silva Rodrigues com o intuito de definir a melhor escolha para a criação de um jogo acessível e funcional. Após isso, o jogo da memória numérico, *Anamnese*, foi selecionado para ser construído, em decorrência da fácil acessibilização, bem como boa jogabilidade.

5.2 Definição do Editor de Texto e Bibliotecas

Com o fito de ampliar a inclusão, para todos os membros da equipe, no processo de elaboração do código fonte, foi escolhido o editor Visual Studio Code (popularmente conhecido como VS Code), afinal apresenta uma interface intuitiva e, portanto, possui simples usabilidade. Além disso, é dotado de boa compatibilidade com o software leitor de tela, o que o torna acessível a deficientes visuais.

Propôs-se para a execução do jogo a utilização da linguagem C, visando colocar em prática os conceitos de fundamentos da programação aprendidos em aulas, realizando, assim, a fixação do conteúdo. Ademais, foram implementadas as seguintes bibliotecas:

- ***stdio.h:*** Possui definições de sub rotinas relacionadas a entrada e saída de dados, normalmente, sempre está presente nos códigos realizados na linguagem C;
- ***stdlib.h:*** Possui funções envolvendo alocação de memória, controle de processos, conversões e outras;
- ***string.h:*** Nela estão contidos protótipos utilizados para a manipulação de strings, também conhecidos como array de chars. Estas funções são capazes desde a contagem, cópia e concatenação, comparação e diversas outras modificações;
- ***ctype.h:*** contém funções e macros para manipulação de caracteres. Utilizando as funções desta biblioteca podemos verificar se um caractere é numérico, ou se é maiúsculo, minúsculo, representa espaço em branco etc;

- ***time.h:*** é um arquivo cabeçalho que fornece protótipos para funções, macros e definição de tipos da biblioteca padrão da linguagem de programação C para manipulação de datas e horários de modo padrão;
- ***windows.h:*** (usada quando o programa é compilado em windows) Adiciona a declaração de funções oriundas da api do sistema operacional Windows, permitindo ao usuário a execução de tarefas;como a utilização da função Sleep() e ou a criação de botões e janelas.
- ***unistd.h:*** (usada quando o programa é compilado em linux)
- ***menu.h:*** Biblioteca local que possui funções acerca das opções escolhidas pelo usuário, a partir da exibição do menu;
- ***jogo.h:*** Biblioteca local que possui funções relacionadas à execução do jogo.
- ***unistd.h:*** Nas linguagens de programação C e C++, unistd.h é o nome do arquivo de cabeçalho que fornece acesso à API do sistema operacional POSIX. Ele é definido pelo padrão POSIX.1, a base da Single Unix Specification e, portanto, deve estar disponível em qualquer sistema operacional e compilador compatível com POSIX. Por exemplo, isso inclui sistemas operacionais Unix e semelhantes a Unix, como variantes GNU, distribuições de Linux e BSD e macOS, e compiladores como GCC e LLVM.

Após a conclusão do código do jogo, este foi disponibilizado em uma IDE on-line. Foi escolhido o REPLIT, pois possibilita a execução do programa sem a necessidade de fazer download do código ou bibliotecas. Entretanto, devido a baixa acessibilidade dessa IDE, foi escolhido, também, um repositório on-line. O grupo optou pelo GitHub, devido ao acesso inclusivo e popularidade entre os desenvolvedores.

5.3 Etapa de Desenvolvimento

Primordialmente, foi adicionada a biblioteca *stdio.h*, a qual possui funções de entrada e saída de comandos. Após essa implementação criou-se a função principal, *int main*, que contém um *return 0*, dentro dessa função que será desenvolvido todo o código.

5.3.1 Criação do Menu

Dentro da função supracitada, inicialmente, duas variáveis foram criadas: *escolha* do tipo caractere, a qual é iniciada sem nenhum caractere correspondente, e *sair* do tipo inteiro, a qual é inicializada com o valor 1, ambas foram criadas para armazenar as escolhas do jogador. Ademais, de acordo com as instruções recebidas para o trabalho prático, o menu deveria conter as opções para que o jogador pudesse iniciar o jogo e sair deste. Por conseguinte, o grupo também optou por adicionar instruções, créditos e ranking.

Para que o menu principal fosse construído com todas as opções necessárias, foi utilizada a estrutura de repetição *do while*, no qual as instruções contidas dentro da chave são executadas em looping até que o usuário opte por sair do jogo. A primeira função escrita dentro do laço foi a *system("cls")* que é utilizada quando o compilador está relacionado com o sistema operacional Windows. Quando o código foi adicionado ao REPLIT houve a mudança para *system("clear")*, pois essa IDE utiliza compiladores relacionados ao sistema operacional Linux, ou seja, a função é diferente de acordo com o sistema operacional utilizado. Outrossim, a função *system("cls")* é inicializada sempre que o loop se repete, com o fito de limpar o terminal, tornando a interface mais agradável ao usuário.

Nessa perspectiva, após a utilização da função para limpeza do terminal, foi utilizada a função *printf* com o objetivo de exibir o nome do jogo, *Anamnese*. Seguidamente, serão exibidas, por meio de funções *printf*, as opções supracitadas para o jogador, onde a letra "J" representa Jogar, a letra "R" representa o Ranking, a letra "I" representa as Instruções, a letra "C" representa os Créditos e, por fim, a letra "S" representa Sair. Posteriormente, utilizando a função *scanf*, a opção do usuário é lida e armazenada na variável *escolha*.

Ainda, dentro do laço de repetição *do while*, foi adicionada a estrutura *switch case*, na qual o conteúdo da variável *escolha* é comparado aos caracteres estabelecidos, executando um comando caso haja equivalência entre os valores. Também, a utilização do laço *switch case* visa reduzir a complexidade de vários *if else* encadeados.

Cada um dos *case*, incluindo o *default*, possuem uma sequência de código a ser executadas, a saber:

- **Caso 1 - Jogar [J]:** Essa opção realiza a execução do jogo, portanto, ao selecionar [J], entra-se em um *do while*, que será executado enquanto a variável *sair* for diferente de *"N"*. Em seguida, um novo *printf*, contendo a mensagem *"Deseja continuar a jogar? [S] para sim e [N] para nao:"*, é utilizado com intuito de dar ao usuário a opção de voltar ao menu, - essa opção será lida e armazenada na variável *sair*. Caso o jogador digite"N", o loop se encerra.
- **Caso 2 - Ranking [R]:** Essa opção exibe o ranking do jogo, portanto, ao optar por [R], um *do while* é inicializado, o qual será executado, também, enquanto a variável *sair* for diferente de 1. Dentro do laço de repetição, a função *system("cls")* será executada a fim de limpar o terminal. Sequencialmente, *printf's* exibirá os dados contidos no arquivo. Em seguida, um novo *printf*, contendo a mensagem *"Digite [1] para voltar ao menu:"*, é utilizado com o mesmo intuito da opção anterior - essa opção será lida e armazenada na variável *sair.* Caso o jogador digite 1, o loop se encerra.
- **Caso 3 - Instruções [I]:** Ao optar por [I], um *do while* será inicializado, sendo a função *system("cls")* a primeira função a ser executada. Após, serão apresentadas as instruções do jogo, através de estruturas *printf*, com a seguinte mensagem:

  " INSTRUÇÕES
  1. O jogador deve escolher o tamanho da sequência que será decorada;
  2. O jogador deve reproduzir a sequência de números na ordem correta;
  3. A cada sequência correta o jogador acrescenta pontos a seu score;
  4. O jogo termina caso o jogador cometa um erro ou descida

  Seguidamente, também será exibida a opção de retornar ao menu, e esta funcionará assim como nas demais opções, utilizando a leitura da variável *sair*.
- **Caso 4 - Créditos [C]:** Caso a opção [C] seja a escolhida, assim como nas demais, um laço de repetição, *do while*, será iniciado, e a primeira função a ser executada será a *system("cls")*. Posteriormente, serão apresentados os créditos do jogo, através de estruturas *printf*, com as seguintes mensagens:

  " CRÉDITOS
  Este jogo foi criado em 2022 por: Huandy Calini de Camargo Silva, Letícia Vitória dos Santos, Luara do Val Perilli e Luís Eduardo Damasceno, alunos da

disciplina de Fundamentos de Programação e deve ser utilizado como ferramenta educacional e de entretenimento.". Em sequência, os mesmos passos serão executados visando finalizar o loop e voltar ao menu."

- **Caso 5 - Sair [S]:** Essa opção é a única que finaliza o loop. Nesta, primeiramente, será executada a função *system("cls")* para limpar o terminal. Logo após, será exibida a mensagem "Fechando jogo…" por meio da função *printf*. Desse modo, quando o usuário fizer essa seleção, o laço de repetição do menu se encerra e, consequentemente, o jogo também.
- **Caso 6 - Default:** Caso seja selecionado uma tecla fora das opções válidas, um *do while* inicia, contendo, primeiramente, a função *system("cls") e*, seguida, através de um *printf*, "Opção inválida". Posteriormente, será dada a opção de voltar ao menu, também com a exibição da mensagem *"Digite [1] para voltar ao menu:"*, por meio de um *printf*. Com a leitura da variável *sair*, essa sendo igual a 1, o loop se encerra voltando ao menu principal.

Nota-se, ainda, que foi adicionada aos casos 1, 2, 3, 4 e 6 a opção *sair*, visando proporcionar um maior tempo para o leitor entender o que há dentro de cada um dos casos. A título de exemplificação, tem-se as instruções, a qual contém as regras do jogo, que são fundamentais para a compreensão de como a *Anamnese* funcionará, por tanto necessitam ser analisadas cuidadosamente.

5.3.2 Modularização

Para que o código do jogo ficasse mais limpo e organizado, o grupo realizou a modularização do código. Desse modo, foi criado a biblioteca menu.h com os protótipos das funções e, também, a criação do *menu.c* contendo as funções relacionadas às opções do menu do jogo.

Com isso, novas bibliotecas foram incluídas no código, para que algumas funções fossem usadas. A biblioteca *stdlib.h*, por exemplo, proporcionou a inclusão da função "system("cls")", para que o terminal fosse limpo sempre que essa for usada.

Desse modo, foram criadas as seguintes funções, com os respectivos objetivos:

- ***letra_maiuscula:*** Quando o usuário escolher uma opção do menu, independente se ele digitar a letra maiuscula ou minuscula o programa vai ser executado, pois essa função irá transformar a letra minúscula em maiúscula, antes de ela ser usado pela função menu (a biblioteca *ctype.h* foi incluída para que a função toupper pudesse ser utilizada para a realização dessa mudança);
- ***menu:*** Função contendo um *switch case* para chamar outras funções dependendo da opção do jogador, como jogo (função implementada em outra biblioteca, a *jogo.h*), *ranking, instrucoes, creditos e opcao_invalida;*
- ***voltar:*** Função que controla o loop de exibição de algumas funções, e é chamada dentro destas (*ranking, instrucoes, creditos e opcao_invalida*);
- ***ranking:*** Função que exibe o ranking dos jogadores;
- ***instrucoes:*** Função que exibe as instruções do jogo;
- ***creditos:*** Função que exibe os créditos dos criadores do jogo;
- ***opcao_invalida:*** Função que exibe a mensagem de opção inválida sempre que o usuário digitar uma opção não condizente com as opções exibidas.

5.3.3 Criação do jogo

Após o processo de modularização das funções criadas no código, deu-se início ao desenvolvimento da jogabilidade acerca do programa. Desse modo, os alunos iniciaram com a criação de variáveis e vetores necessários para a criação de um jogo da memória.

Foi criado um ponteiro, *numeros_gerados*, o qual teve de receber uma alocação dinâmica - foi utilizado a função *malloc* e ao fim do loop, essa alocação foi liberada, desse modo ela inicia e libera a memória alocada para o ponteiro a cada passagem pelo loop. Após isso, a função *rand* foi usada para gerar números aleatórios, ademais, no antes de ser usada para a geração de números aleatórios, a função *srand* foi implementada junto ao *time* (NULL), para que as sequências não fossem compostas sempre pelos mesmo números.

Com isso usou-se um laço de repetição do tipo *for* para que cada elemento do vetor *números gerados* fosse comparado com o número que o jogador escrevia, o qual era armazenado na variável *numero_digitado*. Assim, o jogador digita um número da sequência por vez, sendo cada um desse atribuído para a variável *numero_digitado*. A cada número compatível uma mensagem de confirmação é exibida na tela e caso o jogador acerte toda a sequência, há um acréscimo em seus pontos e ele é incentivado a aumentar o tamanho da sequência, mas isso é feito mediante sua vontade. Entretanto, caso o jogador cometa um equívoco com algum dos números da sequência, é exibida uma mensagem informando o erro, e ele não acrescenta pontos nessa rodada. Ademais, sempre que uma rodada é terminada, o jogador fazendo ou não o ponto, ele é questionado se quer ou não continuar no jogo, caso não o *do while* loop é encerrado.

5.3.4 Criação do ranking

Inicialmente, um arquivo do tipo *txt* foi criado para que os dados do jogador fossem armazenados nesse. Os dados a serem salvos no arquivo são: nome do jogador, pontos feitos pelo jogador e maior sequência decorada pelo jogador, caso o jogador não acertar nenhuma das sequências, será atribuído para a maior sequência decorada o valor 0, afinal o jogador não decorou nenhuma das sequência exibidas.

Após a criação do arquivo, intitulado *ranking.txt* foi adicionado, na função jogo, contida na biblioteca *jogo.h* a *struct gamer*, contendo as seguintes variáveis: *char nome*[30] (para armazenar o nome do jogador, o qual passa a ser perguntado antes da inicialização do jogo, dentro do *do while*), *int pontuação* (para armazenar os pontos feitos pelo jogador) e *int maior_sequencia* (para armazenar o tamanho da maior sequência decorada pelo jogador em questão). Após isso, foi criada uma função, *criaJogador* com o intuito de criar uma alocação dinâmica para a *struct gamer*, a qual passa a ser chamada de *GAMER*, e a função *liberaJogador* a qual libera a alocação dinâmica da variável do tipo *GAMER* quando chamada. Após isso foi necessário implementar a biblioteca *jogo.h* dentro da biblioteca *menu.h*, para que a interface funcionasse sem erros e, consequentemente, a alocação fosse realizada corretamente.

Assim, dentro da *main*, foi iniciada o ponteiro para a variável *jogador* do tipo *GAMER*, e, também, foi chamada, logo após esta criação, a função *criaJogador*, para que a alocação dinâmica fosse criada para esse ponteiro criado. Outrossim, ao fim da *main*, quando o jogador finaliza o menu, para que o programa seja fechado, antes do *return 0* da *main*, é chamada a função *liberaJogador* para que a alocação dinâmica seja liberada na memória.

Em relação a printagem das informações para o arquivo, *ranking.txt*, essa é realizada após o *do while* loop controlador da repetição do jogo, dentro da biblioteca *jogo.c*. Desse

modo, quando o jogador opta por parar de jogar é realizada a printagem dos dados do jogador (*nome*, *pontuacao* e *maior_sequencia*, contida na struct *gamer jogador*). Assim, primeiramente, é criado um ponteiro para o arquivo, usando *FILE *class_a*, sendo *class_a* o nome dado para esse ponteiro. Após isso, é utilizada a função *fopen* para abrir o *ranking.txt* e *fprintf's* são usados para realizar os passos supracitados, e logo após é usada a função *fclose* para fechar o arquivo.

Para a exibição do ranking, realizado na biblioteca *menu.c*, foi-se inicialmente usado FILE **class_r*, sendo *class_r* o nome dado para o ponteiro do arquivo, neste caso. Após isso, foi utilizada a função *fopen* para abrir o arquivo, e um *while* é utilizado para passar os dados do arquivo, utilizando a função *fsanf* - usada para ler os dados do arquivo - para variáveis localmente criadas (*nome*, *ponto* e *sequencia*) e um printf para a exibição dessas variáveis contendo os dados do arquivo - os dados percorreram cada printagem que foi mandada para o *ranking.txt* no jogo, desse modo, uma variável contadora int *i* é acrescentada a cada rodada do *while*, o qual só termina quando não há mais nada a ser lido (EOF - End Of File). Após a realização dessa exibição, usou-se a função *fclose* para fechar o arquivo.

6. DIFICULDADES ENCONTRADAS

Durante a criação do jogo *Anamnese*, poucos obstáculos foram encontrados. Entre eles, é possível citar que, ao adicionarmos as instruções e créditos, foi necessária a utilização de várias estruturas *printf*, para que quando o código fosse lido e avaliado, não houvesse a necessidade de arrastar a tela para o lado. Por isso, para que não houvessem grandes mensagens em um único *printf*, a mensagem foi dividida em *printf* menores.

Também cabe mencionar que, ao executar o menu, foi identificado um erro no qual "opção inválida" sempre era exibida após o jogador realizar a escolha de um dos casos. Isso se deu pela falta de um espaço na leitura da variável do tipo caractere, *escolha,* ou seja, *scanf("%c", &escolha)*. Por conseguinte, o correto seria *scanf(" %c", &escolha)*. Isso ocorre devido ao *buffer* de teclado. Em outras palavras, ao digitar uma opção - [J], por exemplo - e apertar a tecla enter, o *buffer* de teclado armazena temporariamente aquilo que foi digitado - neste caso, a tecla enter como (\n) - como o próximo caractere atribuído ao menu. Assim, como o enter (\n) não estava entre as demais opções, "opção inválida" era exibido antes de voltar ao menu e dar ao usuário a oportunidade de escolher uma nova opção. Dessa maneira, quando adicionado um espaço antes de *%c*, tornando-se *" %c"*, o espaço, enter ou tab, por exemplo, que pode estar no *buffer* é dispensado, uma vez que o espaço dado antes do *%c* realiza uma espécie de limpeza no *buffer*, possibilitando, assim, a escolha do usuário.

Posteriormente, durante a criação do código que possibilita o funcionamento do jogo, o grupo enfrentou adversidades no quesito utilização e manipulação de vetores. Em outros termos, foi realizada a declaração de um vetor visando armazenar uma sequência de números aleatórios gerados pela função *rand*. Porém, este foi declarado e teve seu tamanho definido para uma variável não inicializada. Desse modo, ao utilizá-lo em um laço de repetição de forma a armazenar todos os números necessários, a lógica definida não funcionava como o previsto, pois o tamanho do vetor baseava-se em lixos de memória contidos na variável designada para atuar com tal finalidade. Visando resolver o problema apresentado, os membros do grupo buscaram possíveis soluções por intermédio do auxílio da docente Elisa de Cássia da Silva Rodrigues. Esta esclareceu as dúvidas a respeito da causa do mal funcionamento do código, indicando o uso de alocação dinâmica como método de correção para o os erros apresentados.

Ademais, com a utilização de alocações dinâmicas e ponteiros, alguns equívocos foram cometidos durante a criação do jogo, como a utilização de ponto ao invés de setas nas atribuições para os ponteiros das structs. Mas com o auxílio da docente e pesquisas na internet, os problemas puderam ser resolvidos.

7. CONSIDERAÇÕES FINAIS

Com a realização desse trabalho, pode-se aprofundar os conhecimentos obtidos por intermédio das aulas ministradas pela Profª. Drª. Elisa de Cássia Silva Rodrigues. Isso se deve pela aplicação prática dos conceitos fundamentais de Fundamentos de Programação, como estrutura sequencial, condicional e de repetição, além da utilização de variáveis e funções básicas como *printf* e *scanf*. Outrossim, implementamos bibliotecas, normalmente, presentes em programas escritos na linguagem C, sendo estas denominadas *stdio.h* e *stdlib.h*. Ainda, foram apresentados em sala de aula os conceitos relacionados não somente às boas práticas para a modularização e estruturação do código, mas também estruturas e técnicas cruciais para a manipulação e armazenamento de dados, a saber, vetores e matrizes, registros e utilização de arquivos.

Portanto, conclui-se que a execução do jogo Anamnese agregou novos aprendizados para os membros da equipe, visto que, além do desenvolvimento de um jogo com a linguagem C, o que proporcionou a aplicação dos fundamentos aprendidos, também efetuamos a escrita de toda a documentação, a qual é de extrema imperiosidade quando se está desenvolvendo códigos.

8. LINK PARA ACESSAR O CÓDIGO

IDE Replit: https://replit.com/@leticiasants/JogoXDES01#main.c
Repositório GitHub: https://github.com/leticiasants/versao_final_Anamnese