Desafios superados e os problemas encontrados ao desenvolver o programa

Obtivemos sucesso no desenvolvimento do projeto, onde trabalhamos e buscamos soluções para nossos desafios sozinhos. Talvez a maior dificuldade do grupo foi a manipulação de arquivos, pois nosso software teve a necessidade de inserir e retirar informações de arquivos de forma eficiente e devido a quantidade de informações diferentes que seriam salvas e consultadas nos arquivos, como nome, palavra em jogo e dica, onde a ultima se tratava de uma cadeia de strings, todos esses dados deveriam ser salvos por termos uma função que continua um jogo.

Tivemos que buscar a melhor solução para organizar tais dados de forma eficiente e não misturá-los com diferentes jogadores. A melhor forma encontrada foi dividir a dica do jogo em um arquivo separado dos demais, assim teríamos nome do jogador, palavra, erros, em um arquivo e a dica em outro, todos relacionados pela mesma linha. Logo, podemos puxar a dica muito rapidamente, pois cada jogo salvo teria sua dica em uma linha separada, assim com apenas um fgets à puxamos, não se preocupando um o tamanho e numero de palavras que compõem a dica em questão e como esta estaria formatada.

Outra dificuldade em manipulação de arquivos estava na função Continuar um jogo salvo, onde o usuário entrava com seu nome e tínhamos que buscá-lo no arquivo, puxar todos seus dados e excluí-lo de lá, pois o jogo já estaria em ação, logo não deveria ficar mais salvo. O problema que excluímos um usuário do arquivo de uma linha N, que estava ao meio de outros jogos e toda a formatação do texto era perdida, assim os dados ficavam desorganizados no arquivo. Após outro usuário continuar o jogo, sua palavra se perdia e se misturava com outros dados, se tornando um desastre. A solução foi alocar os dados na memória, zerar o arquivo com os jogos salvos e salvá-los novamente no arquivo, com exceção ao jogador atual.

Haveria também a necessidade de trabalhar com diversos dados e diversas funções, oque tornaria um pouco difícil a troca de informações em nosso programa, pois ao chamar uma função, deveríamos passar a ela o nome do jogador, palavra do jogo, dica, numero de erros, acertos... entre muitos outros. Deste modo, teríamos que declarar inúmeras variáveis e ponteiros em cada função do jogo, oque consequentemente diminuiria a facilidade de manutenção e entendimento do código. Assim, aplicamos o conceito de

struct em nosso software, algo que realmente surpreendeu todos os integrantes, pois definimos um tipo de struct geral que continha todos os dados do jogo, assim através da declaração de apenas um ponteiro tínhamos toda as informações reunidas de forma fácil de acessar em uma função e ao chamar outra, apenas passamos este ponteiro, algo maravilhoso!

Tínhamos em mente também rodar nosso programa em Linux, deixando-o mais portátil possível. Encontramos basicamente dois problemas: Manter o layout do nosso projeto em diferentes terminais do Linux, pois cada distribuição possui um terminal diferente, como o Gnome-Terminal e o Xterm. Assim, quando compilávamos no Ubuntu, tinhas que mudar a estrutura visual do projeto, pois as barras ficavam todas desorganizadas e cada coisa fora do seu lugar! Quando rodávamos no Xterm, outro terminal, tinhas que alterar tudo novamente, pois a formatação é totalmente diferente. No Windows, rodamos nosso projeto sempre no DOS, logo sempre mantemos nossa estrutura de layout, independente da versão do sistema operacional, podendo rodar no Windows 7, XP, 98.
Outro desafio maior no Linux era a incompatibilidade de funções no C. Por exemplo, no Windows podemos usar o getch(), system("pause"), system("cls")... No Linux deveríamos trocar por getchar(), system("clear") e não teríamos o comando para pausa. Logo, teríamos que verificar em qual sistema operacional estávamos compilando o projeto e fazer uma verificação de compatibilidade das funções, tendo de determinar qual função deveria ser usada e se a mesma é compatível com aquele SO. Um trabalho que levaríamos bastante tempo.

Obtivemos sucesso na modularização do programa, algo que parecia difícil, pois tínhamos que trabalhar pensando em outras funções e como seria a troca de informações por nossas funções. Também dividimos nosso programa em três arquivos, que se tratavam da header, arquivo que continha os escopos de todas as funções, continha a nossa struct de dados principal e todos os includes e defines, temos também o arquivo printa boneco, que continha toda a parte gráfica do boneco, desta maneira tornando muito mais fácil a manipulação e também mais simples para fazermos uma alteração necessária, e por último o arquivo principal que continha todas as funções. Tentamos dividir ao máximo nosso programa em funções, pois fica facilita, e muito, a manipulação do código. A qualquer erro que surgia, sua localização e correção se tornou muito fácil, assim a correção de eventuais Bugs se tornou prática, pois analisamos partes pequenas de código. Além de tudo, as funções nos deram liberdade na divisão dos trabalhos, onde cada integrante fez algo.