UFOP (Universidade Federal de Ouro Preto)

PROFESSORES:

Túlio Toffolo

Puca Huachi

TRALHO PRATICA(TP):

Gabriel Catizani Faria Oliveira

20.1.4004

Fazendo o jogo “Resta Um” em LINGUAGEM C

Ouro Preto, Minas Gerais

12 de outubro de 2020

Nesse relatório explicarei passo por passo como fiz meu código e as dificuldades que passei durante o trabalho prático. No inicio fiquei um pouco ansioso, porém consegui ir resolvendo aos poucos o trabalho até conseguir realizar tudo que os professores pediram

Primeira coisa que fiz foi testar a impressão da tabela, mas sem a formatação usando um arquivo que tinha no seu início dois diâmetros menores que o normal para matriz que no caso foi 3 x 3. Pra isso, primeiro fiz um fscanf para pegar os diâmetros (n e m) do arquivo e logo depois criei uma função para colocar os valores do arquivo, por meio de um for, na matriz[n][m]. Depois disso para imprimir a primeira tabela e ver se tinha funcionado fiz a função imprimir tabela que imprimia a matriz[n][m]. Assim, iniciei as funções dentro do main e vi que tudo tinha funcionado de forma correta.

Nesse início, fiquei meio confuso com arquivos, mas consegui mexer com ele da forma correta até o programa funcionar. E assim, já meio cansado demorei para entender como faria as peças se movimentarem e a primeira coisa q fiz em relação a isso foi criar um protótipo de função para cada comando executado pelo usuário. Fiquei horas tentando entender como faria esses comandos até que tive ajuda de um amigo que faz engenharia da computação na PUC em BH e já está 1 período na frente e do Augusto, colega da faculdade. Os dois me explicaram como funcionaria o movimento dos pinos para cada direção e como eu faria isso nas funções. Usei laços de repetição para cada comando, porém fiquei horas tentando entender como fazia os argumentos/coordenada do programa. Logo, deixei para mexer nas coordenadas outro dia. Mais pra frente, tive vários erros com esses movimentos, porém foram corrigidos com ajuda do professor Tulio.

Com os movimentos, prontos precisava apenas validar cada um. Assim, a primeira invalidação foi dentro da função principal caso o usuário digitasse um comando inexistente e assim imprimia a tabela novamente pedia para digitar outro comando e caso precisasse de saber eles de novo, perguntava se o jogador queria que mostrasse a lista de comandos novamente. A segunda invalidação era dentro de cada função para os comandos caso não fosse possível mover a peça em determinada direção. Para isso, pensei nos movimentos em que a peça ia bater na parede;

Após isso, tentei fazer o comando ajuda, mas não consegui por hora. Então, deixei para o final e fui fazer o tabuleiro aleatório. Para isso tive ajuda do meu amigo de BH, Lucas Padrão novamente, o qual me mostrou como funcionava a lógica de fazer esses tabuleiros aleatórios e as diversas formas que poderia cria-los. Dessa forma, consegui criar 4 tabuleiros aleatórios e com diversas posições para a primeira posição do buraco vazio. Foi um longo código o qual fiz em um arquivo.c separado para conseguir pensar melhor. Consegui realizar o código com perfeição e criei uma função com esse código e com as suas devidas mudanças de acordo com o código do TP, porém não sabia como realizar o código do TP de forma que não precisasse de digitar o argumento do arquivo a ser buscado para iniciar o programa. Então, pedi ajuda para os meus colegas Augusto, Pedro Lucas, Lucas Gomes e Robson Novato, os quais me falaram para criar um if dentro da função principal, o qual testaria o contador de argumentos passados para a função main. Caso esse contador fosse maior ou igual a 2, o jogo rodaria com o tabuleiro normal e, caso contrário, rodaria com o tabuleiro aleatório.

Após essa dificuldade, fiz a função pra imprimir ou não a mensagem de vitória. Para função 'vitoria', a qual não tive tanta dificuldade, pois era só contar a quantidade de pinos que faltavam toda vez que voltava no loop 'do-while' da função main. Assim, se têm apenas um pino restando, aparece uma mensagem de vitória e o programa se encerra e fecha.

Logo, como falado, consegui realizar a função 'ajuda' com ajuda do meu colega Robson Novato. Passei uma grande dificuldade nessa questão, pois não sabia para onde ir e o que fazer, apesar de já ter ideias de fazer os movimentos de novo dentro da função "ajudar". Então, com a ajuda do meu colega Robson Novato, aluno da UFOP ele me deu dicas de como realiza-la, fazendo um contador para a quantidade de ajudas possíveis e quando chegar na quantidade de ajudas que o usuário pediu o 'for' para de rodar. Em cada ajuda, ela dá sugestões dos movimentos possíveis de se realizar

Com tudo isso pronto, consegui realizar as coordenas escolhidas pelo jogador transformando-as de letras para números e com da tabela ASCII fiz uma função pra isso, em que por exemplo, a letra Apossui um valor 65, tirei 65 dele e isso equivale ao número 0 e pra isso fiz uma auxiliar para coordenada coluna e

para coordenada linha. Por meio disso, consegui imprimir a tabela com letras e consegui usar fazer os comandos de acordo com as coordenadas que o usuário escolheu.

OBSERVAÇÃO: Para o programa não ver diferença entre minúsculas e maiúsculas, com ajuda do meu colega Leonardo Valle, o qual estuda na PUC-Minas, passei as coordenas por argumento com a função ‘toupper’ da biblioteca <ctype.h>.

Logo em seguida fui para funçao derrota. Antes tinha feito a derrota e vitória juntos, mas vi que isso não iria funcionar. Logo, aproveitei a função 'ajuda' para fazer essa função ‘derrota’. Sempre que voltava no loop ‘do-while’ da funçao main, abria essa função e conferia se tinha movimentos possíveis para se fazer em todo tabuleiro. Conferia-se isso por meio de um contador('contsugestoes'). Se contsugestoes fosse igual 0, o código continuava a rodar e perguntava o comando novamente pro jogador. Caso não houver mais movimentos possíveis, o loop é finalizado na função main e dá uma mensagem de derrota e o programa se encerra.

Com isso, passei para função do tabuleiro aleatório. Essa função utilizada para ler um tabuleiro aleatório, seja ele com uma posição do espaço vazio inicial diferente ou o tabuleiro em si diferente. Para isso, criei dois tabuleiros com formatos diferentes, podendo conter os espaços vazios em posições diferentes. Junto com isso fiz uma função para copiar a matriz do tabuleiro aleatório e salvar ela num arquivo .txt, e uma função para imprimir o tabuleiro aleatório.

Com tudo isso feito, compilei o código e fui corrigindo os erros que neles estavam, como variáveis colocadas em lugares errados e o principal, problemas na hora do programar, principalmente com strings e caracteres. Para isso busquei ajuda de meus colegas novamente para me ajudarem e resolverem os erros e, assim, os resolvi e tudo funcionou normalmente

Com o código rodando perfeitamente(entretanto, sem saber todos os raciocínios estavam corretos), por fim, apliquei as formatações necessárias, identei, colori e comentei todo meu código da forma mais bonita para o usuário e, claro, para os professores jogarem e com gosto (e também para lerem meu código de maneira mais clara).

- Referências:

- Aluno Augusto (UFOP)

- Aluno Pedro Lucas Damasceno (UFOP);

- Aluno Lucas Gomes dos Santo s(UFOP);

- Aluno Robson Novato Lobão (UFOP);

- Colega Lucas Padrão (PUC-Minas);

- Colega Leonardo Valle(PUC-Minas);

- Tabela ASCII;

- Professor Tulio Toffolo;

- Professor Puca Huachi;