## **UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ DEPARTAMENTO ACADÊMICO DE ELETRÔNICA (DAELN) RELATÓRIO DO PROJETO DA DISCIPLINA FUNDAMENTOS DE PROGRAMAÇÃO 1 (IF61F) DO CURSO DE ENGENHARIA ELETRÔNICA**

Willian Joris Ayres, Jayme Lourenço Guedes Neto

Disciplina: **Fundamentos de Programação 1** / S11 – Profs. João Luiz Rebelatto e. Danillo Leal Belmonte

**Departamento Acadêmico de Eletrônica – DAELN** - Campus de Curitiba

### **Universidade Tecnológica Federal do Paraná – UTFPR**

will.joris@gmail.com, jlgn@supering.com.br

**Introdução**

Este documento apresenta o relatório para o trabalho final da disciplina de Fundamentos de Programação 1. Tendo em vista como motivação o aperfeiçoamento de algoritmos, melhor compreensão da linguagem de programação em C e desenvolvimento da abstração para resolução de problemas, o trabalho foi desenvolvido por meio da IDE “Code Blocks”, com seu código escrito por meio de linguagem C. Em sua totalidade, o programa é compreensível para quem possuí o mínimo de entendimento de linguagem de programação, visto que foi reduzido, na medida do possível, a funções recursivas e não muito extensas, além de, sempre que necessário, separado em funções auxiliares para melhor visualização.

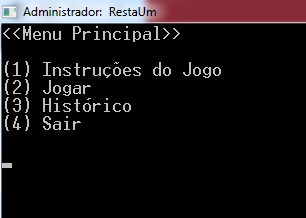
**OBJETIVO**

Desenvolver o jogo “Resta 1” usando atribuições da Linguagem C apresentadas em aula.

# **EXPLICAÇÃO DO SIMULADOR EM SI**

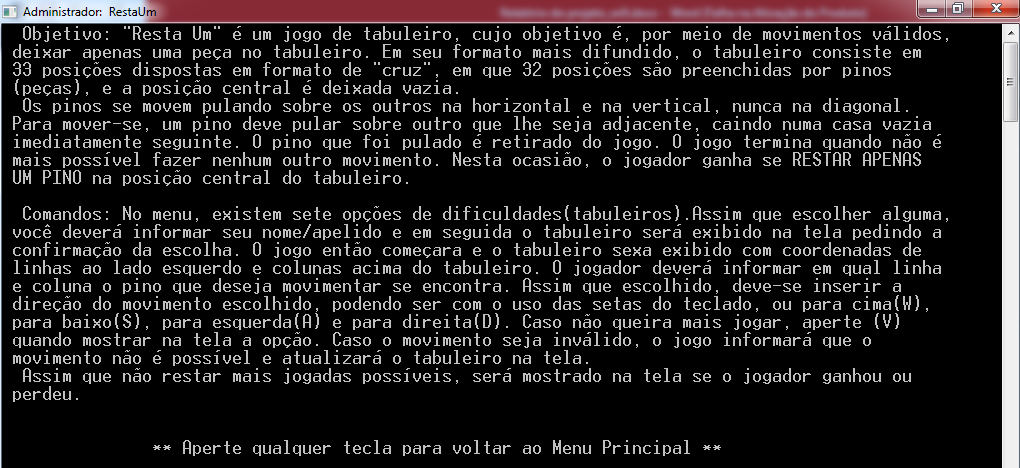
Por meio de movimentos válidos (horizontais ou verticais), pulando “pinos” e ocupando casas vazias, o pino pulado é retirado do jogo gerando mais casas vazias, assim o jogo termina quando restar apenas um “pino” na posição central do tabuleiro do jogo. O jogador perde quando não houver mais jogadas possíveis e restando mais de um “pino” no tabuleiro.

Assim que iniciado o programa com o executável, abre-se a janela do console com nome de “Resta Um”, e printando na tela as opções do menu inicial.

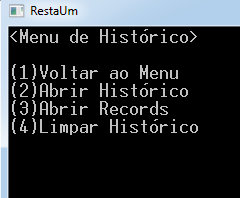


**Tela do Menu Principal.**

Com a escolha de 4, o jogo é fechado. Caso 3 aberto o menu de histórico. Caso 1 aberto as instruções. Caso 2, aberto o menu de tabuleiros.



**Instruções/Regras.**

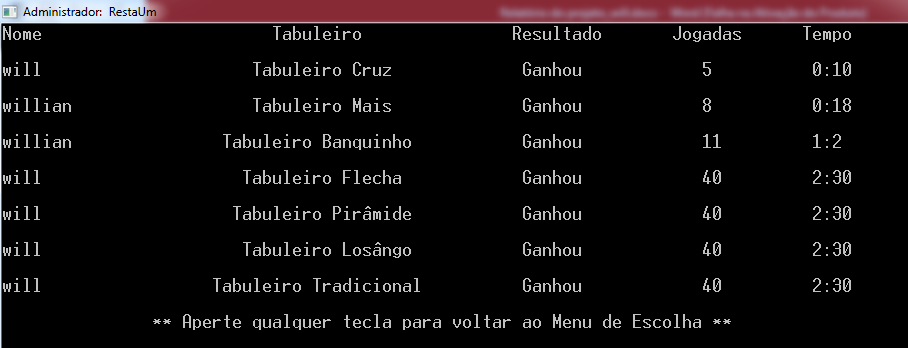


**Menu de Histórico.**

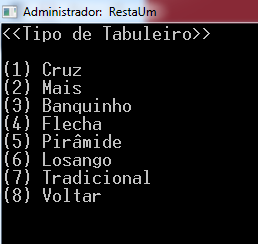
Caso 1, ele volta ao menu principal. Caso 2, abre o arquivo de histórico. Caso 3, abre o arquivo de records. Caso 4 limpa o histórico.



**Histórico.**

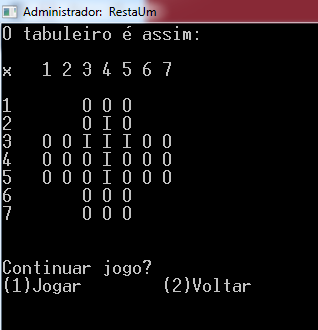


**Records.**

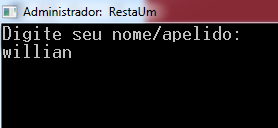


**Menu de Tabuleiros.**

Caso a escolha seja 8, ele retorna ao menu principal. Caso escolha de 1 a 7, o simulador imprimi o tabuleiro para mostrar ao jogador se realmente deseja jogar nesse tabuleiro. Caso o jogar digite 2, ele retorna ao menu de tabuleiros. Caso 1, ele pede o nome do jogador e então começa o jogo.

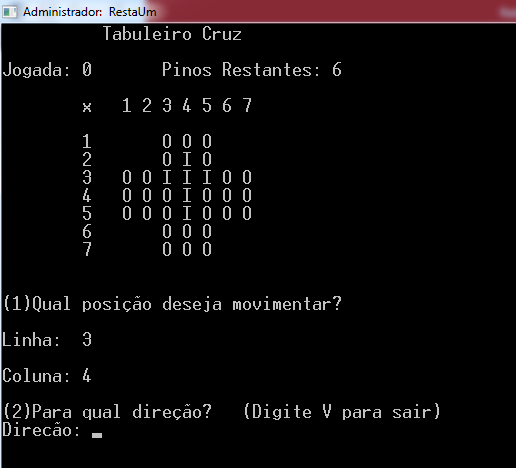


**Tabuleiro impresso para visualização.**



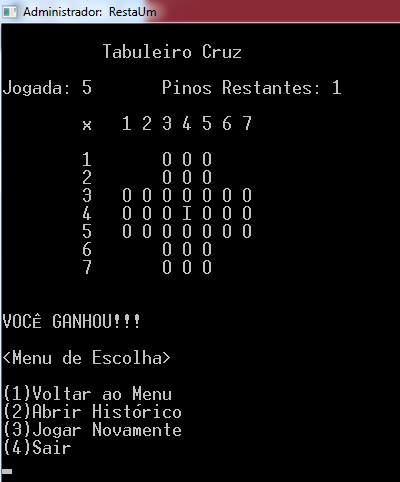
**Esperando que um nom/apelido seja digitado.**

Assim que iniciado, o simulador imprimi na tela o tabuleiro e seu nome, com número de jogadas, pinos faltantes, e pede que jogador digite a linha e coluna que o pino que deseja movimentar se encontra, além da direção do movimento. Se a teclado V, o jogo retorna ao menu inicial.



**Jogada acontecendo.**

Assim que terminado, é mostrado na tela se o jogador venceu ou perdeu e o menu de escolha para escolher entre: sair (o programa é finalizado); voltar ao menu (o programa volta ao menu inicial); abrir o histórico (abre o histórico como na IMG4); e jogar novamente (o programa se repete a partir do menu de tabuleiros).

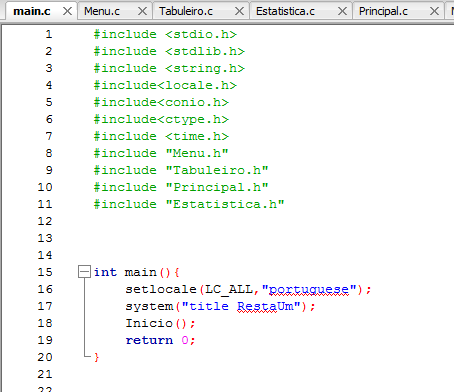


**Jogo finalizado e menu de escolha.**

Implementação do Jogo

As bibliotecas utilizadas para o programa foram: stdio.h, stdlib.h, string.h, locale.h, conio.h, ctype.h, time.h. Além dos cabeçalhos .h que foram separados para melhor visualização do programa: Menu.h, Tabuleiro.h, Estatistica.h, Principal.h.

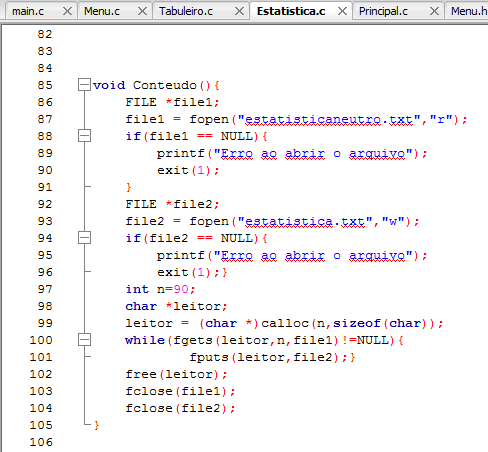
O simulador de Resta Um inicia-se com setlocale (LC\_ALL,”portuguese”) para a adição de caracteres especiais, system(“title RestaUm”) para mudar o nome do console e então, chamada da função de menu inicial, executado pela função Inicio ().

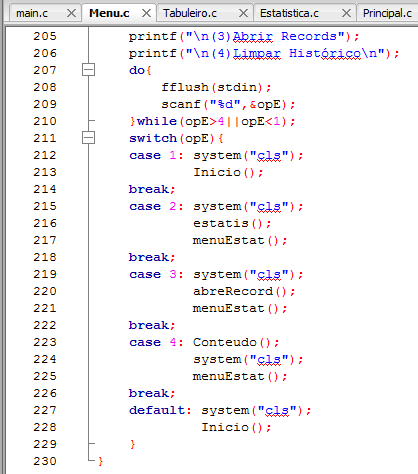


Assim que chamada, é mostrado na tela as opções e então solicitado ao usuário/jogador que insira o número correspondente a opção escolhida. O endereço desse número (1<\*op<4) é então passado como parâmetro de referência para a função menu (int \*op), onde passa por uma estrutura de decisão do tipo switch. Caso \*op=1, então, na mesma função, é chamado um ponteiro do tipo FILE para a leitura do arquivo texto de regras, presente no mesmo diretório do programa Resta Um.



Após a leitura, o programa pode voltar à função Inicio (). Caso \*op=3, é chamado então a função menuEstat (), similar a função de menu, porem as opções de escolha são: voltar (volta a função inicial); abrir histórico (chama a função para leitura do arquivo texto de estatísticas e depois volta ao menuEstat); abrir records( chama a função para a leitura do arquivo texto de records e depois volta ao menuEstat); limpar histórico (chama a função Conteudo (), na qual copia um arquivo texto neutro para o arquivo texto de estatísticas, e então retorna ao menuEstat); abrir records (chama a função para o arquivo texto de records).

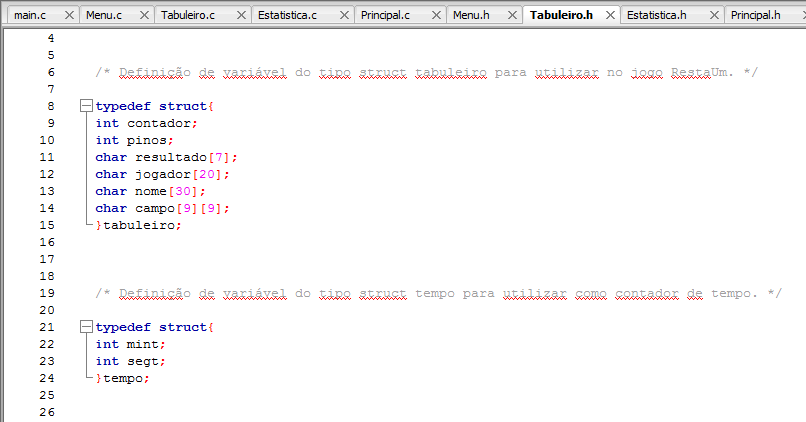




Caso \*op=4, o programa chama a função exit (0), para retornar falso e fechar o programa.

Para o decorrer do programa, é necessário definir por meio do comando typedef uma struct do tipo tabuleiro contendo: valores de inteiro para número de pinos e contador de jogadas; strings para nome do campo, nome do jogador e resultado de jogo; e uma matriz de tamanho 9x9 para o campo.

Além da struct de tabuleiro, defina uma variável do tipo struct para salvar os valores de minutos e segundos da duração do jogo.

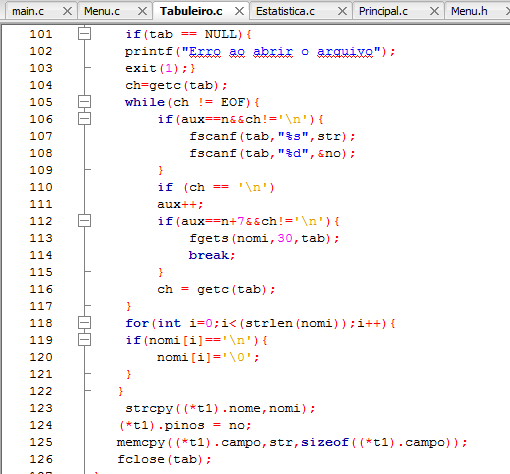


Caso \*op=4, é mostrado na tela as opções de tabuleiro, é declarado um ponteiro para variável do tipo struct tabuleiro, alocado dinamicamente e em seguida, pedido a entrada do valor opS (1<opS<8) para a chamada da função de menuS (opS), como um menu secundário.

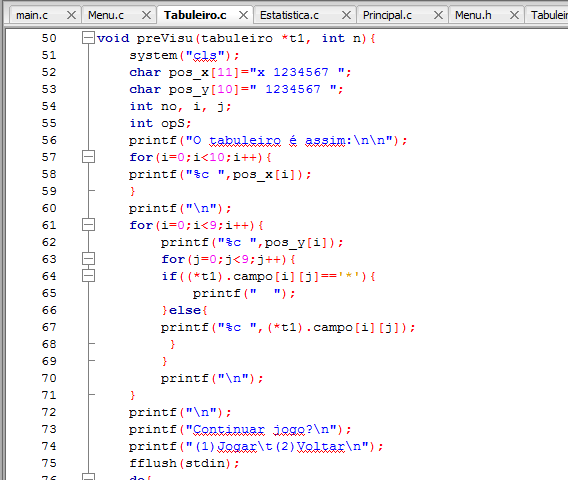
Caso a escolha de opS seja 8, o programa então retorna ao menu inicial.

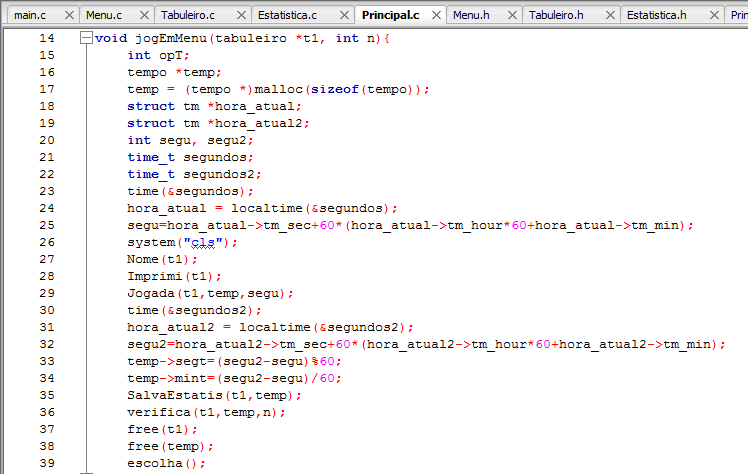
Cada opção contém a mesma inicialização de jogo, indiferente do tabuleiro, porem o parâmetro n muda conforme a opção de jogo escolhida, para quando chamado a função declaraTabu (tabuleiro \*t1, int n), a leitura do arquivo texto que contém as informações de: nome do tabuleiro, quantidade de pinos, e formato do tabuleiro possam ser lidas e armazenadas corretamente, já que cada tabuleiro se encontra em uma linha do arquivo texto, e n serve como parâmetro para o encontro dessa linha no arquivo. Além de puxar essas informações, a função inicia o contador de jogadas com 0.

Para o campo é declarado uma string[82] e um ponteiro FILE para leitura do arquivo texto do tabuleiro respectivo e, em seguida, atribuído à string o que estava salvo no arquivo. Por meio da função memcpy, o campo é então inicializado com o que havido sido gravado na string. Além do campo, é puxado o nome do tabuleiro e salvo em nom[22] com strcpy.



Após a inicialização do tabuleiro, é chamada a função preVisu (t1) para visualizar o tabuleiro e decidir se quer continuar o jogo nesse tabuleiro. Caso 2, o programa volta ao menu secundário para escolha de outro tabuleiro. Caso 1, é chamada função de sucessão do jogo jogEmMenu (t1). Primeiramente, é inicializado duas structs tm (da biblioteca time.h) para o controle do cronômetro de jogo. Assim que armazenados corretamente os dados de minuto e segundos, é inicializado o nome do jogador do campo nome da struct tabuleiro. Após, é chamada uma função de impressão do tabuleiro Imprimi (t1), que mostra na tela a jogada atual, número de pinos, nome do tabuleiro, as linhas e colunas do tabuleiro e por último os pinos e as posições válidas do tabuleiro.





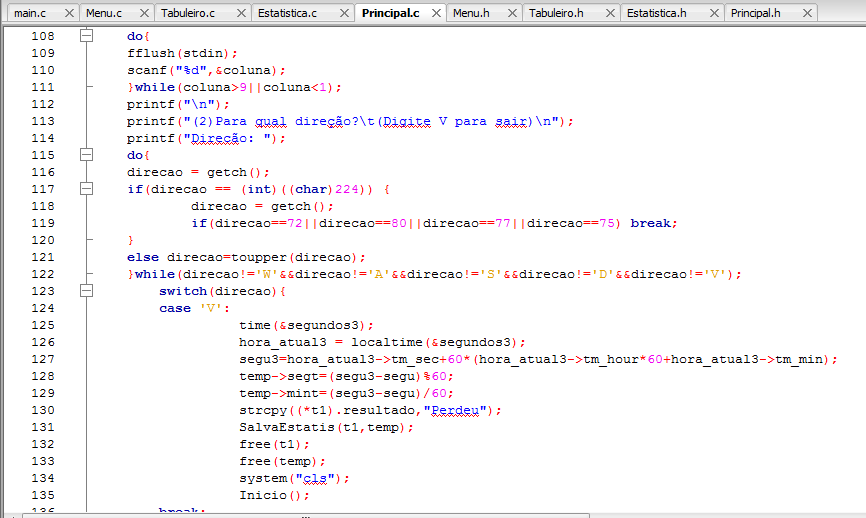
Após a impressão, é chamada a função recursiva Jogada (tabuleiro \*t1, tempo \*temp, int segu) com os parâmetros: ponteiro pra struct tabuleiro; ponteiro pra struct tempo; e um inteiro segu que armazenou a quantidade de segundos para controle de tempo. A função Jogada é chamada até não existirem mais jogadas possíveis. A primeira coisa a ser executada dentro dela, é a chamada de duas funções auxiliares: ContaTabuleiro(t1) para conferir se existem jogadas possíveis (se existe pelo menos um pino ligeiramente próximo ao outro), que retorna 0 se não existir; GameOver(t1) para conferir caso o retorno de 0 na função anterior, jogador venceu, e 1, jogador perdeu. Após isso, mais uma struct tm é declarada para controle de tempo dentro da função Jogada.

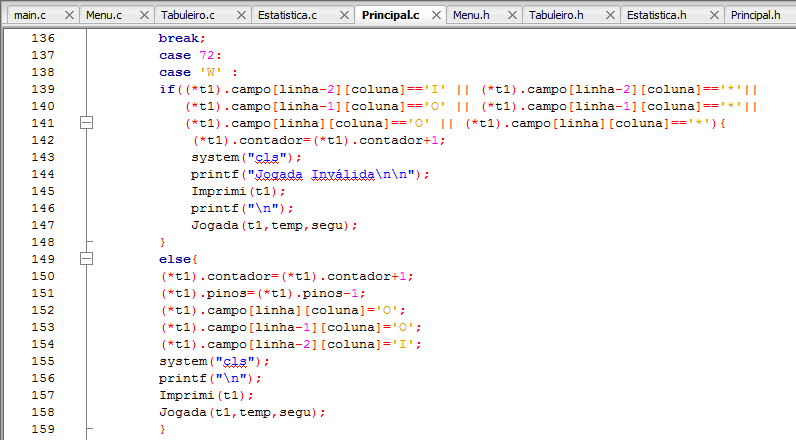
Se ainda existirem jogadas possíveis, é declarado dois inteiros (linha, coluna), um char direção e então, lidos do teclado (1<linha/coluna<9 e direção=W/A/S/D/V ou direção=setas do teclado). É executado um switch (direção) para realizar a lógica da jogada, que dependendo da direção escolhida, o campo[linha][coluna] será alterado. Antes da jogada ser realizada, é conferido se a jogada é válida, ou seja, se a posição a ser movimentada não é um \*, se onde quero chegar já não exista um pino ou se realmente estou movimentando um pino. Se a jogada for impossível, é mostrado na tela “Jogada Impossível”, o contador de jogadas é acrescentado em 1 e então novamente imprimido o tabuleiro e chamada a função Jogada (t1, temp, segu).



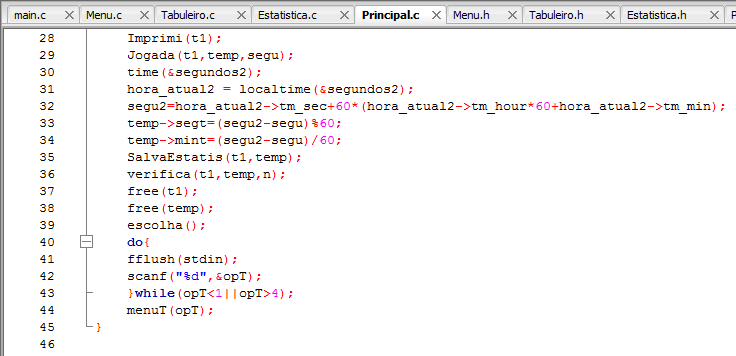
Caso a jogada seja válida, é ajustado o tabuleiro por meio de lógica conforme os parâmetros campo[linha][coluna] e direção; diminuído o número de pinos, contador de jogadas somado a 1, impresso na tela e novamente chamada a jogada.

Caso a escolha do switch(direção) seja V, o programa finaliza o cronômetro, libera os espaços alocados para as strucs, salva nas estatísticas os dados, e retorna ao menu principal. Esse comando serve para um escape do jogo se não puder terminar ou não quiser mais continuar o jogo.





Caso o jogo tenha chegado ao fim, é retornado à função jogEmMenu (t1, n), onde é salvo em um arquivo texto tabuleiro que foi jogado, o nome salvo, resultado, número de jogadas e o tempo de duração do jogo. Em seguida, é chamado a função verifica (tabuleiro \*t1, tempo \*temp, int n), na qual abre um ponteiro para arquivo FILE, lê a linha do arquivo de recordes (que contém 1 recorde para cada tipo de tabuleiro) dependendo de n como parâmetro, compara se o resultado foi de vitória, se o número de jogadas foi o mesmo ou o menor presente no recorde, e por último, compara se o tempo foi menor. Caso tenha batido os valores tenham sido menores, então é chamado a função apagaRanking (t1, temp, n) para apagar a linha do arquivo de recordes correspondente a n como parâmetro, armazena o que havia antes da linha especifica no arquivo aux1 e o que havia depois no arquivo aux2. Então, chama-se salvaRanking (t1, temp), para colocar as informações de recordes atualizados no arquivo aux1, em seguida, chama-se armazenaRanking () para concatenar os arquivos aux2 em aux1 e, por último, chama-se arrumaRanking () colocar o que havia em aux1 no arquivo de recordes padrão, e limpar os dois arquivos auxiliares. Após essa sucessão de funções, o programa volta a função jogEmMenu, assim como se nenhum recorde foi batido no jogo. Então, é liberado o t1 e o temp que foram alocados dinamicamente. É declarado um inteiro e lido do teclado (1<opt<3) e chamado uma função de menuT (opt).



Nela, é escolhido se o jogador deseja ver o histórico salvo, se deseja jogar novamente, se deseja retornar a tela principal ou se sair do programa diretamente. Caso a opção seja jogar novamente, o programa volta ao menuS e continua a ser executado, até que em algum momento o jogador deseje sair do jogo, procurando alguma função de escape exit (0) espalhado pelo programa.



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Conceitos/implementação | menu.c | tabuleiro.c | estatística.c | principal.c |
| Estruturas de Decisão | Linhas 22;27... | Linhas 37;66... | Linhas 47;94... | Linhas 55;77... |
| Estruturas de Repetição | Linhas 72;210... | Linhas 20;105... | Linhas 24;46... | Linhas 41;53... |
| Vetor/String | Linhas 71;73... | Linhas 16;58... | Linha 24;78... | Linhas 88;92... |
| Matriz | Não usado. | Linhas 37;67... | Não usado. | Linhas 140;153... |
| Funções com Parâmetro por Valor | Linhas 98;162... | Linhas 50;94... | Linhas 109;138... | Linhas 15;30... |
| Funções com Parâmetro por Referência | Linha 60;105... | Linhas 15;80... | Linhas 139;147... | Linhas 29;65... |
| Typedef/Struct | Linha 100. | Linhas 40;125... | Linhas 64;137... | Linhas 19;34... |
| Ponteiro | Linha 100. | Linhas 99;124... | Linhas 22;39... | Linhas 17;35... |
| Alocação Dinâmica | Linhas 71;76;100. | Não usado. | Linhas 23;99... | Linhas 18;39... |
| Arquivo Texto | Linhas 65;72... | Linhas 100;105... | Linhas 40;123... | Não usado. |

**Conclusão:**

Este projeto nos dá uma pequena amostragem do que venha a ser à liberdade da criação de códigos no universo da programação. Usando apenas atribuições da Linguagem C dadas em aulas na DISCIPLINA FUNDAMENTOS DE PROGRAMAÇÃO 1 (IF61F).

|  |  |
| --- | --- |
| “Menu Principal” no jogo | Jayme/Willian |
| “Histórico do jogador” no jogo | Jayme/Willian |
| “Jogabilidade” no jogo | Jayme/Willian |
| Relatório | Jayme/Willian |
| Total Geral | Jayme 40% e Willian 60% |

**Considerações Pessoais:**

Desde o início não foi um trabalho fácil, muitas das vezes se tornando complexo, e de difícil entendimento. Vários erros foram acumulados e para administração e exclusão deles foi necessário muito tempo investido. Aos poucos, erros de leitura de caractere, erros de retorno em funções do tipo void, ou até erros de passagem por parâmetros errados. Tudo isso foi completamente removido, resultando em programa que compila sem nenhum warning da IDE.

Além do projeto ambientar e familiarizar mais com a linguagem de programação, a melhoria em ter que correr atrás por conta própria atrás das resoluções dos erros, implementação de funções por maneira correta, ou até o uso de funções que até então eram desconhecidas. Tudo isso ajudou para o crescimento profissional, no sentido de correr atrás das respostas sozinho. Afinal, um engenheiro precisa detectar um problema e apontar uma solução para ele, tendo ajuda ou não.

**Referências:**

Apostilas em PDF da DISCIPLINA FUNDAMENTOS DE PROGRAMAÇÃO 1 (IF61F).

https:\\www.divertudo.com.br/restaum.html

.