

**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 possui 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.

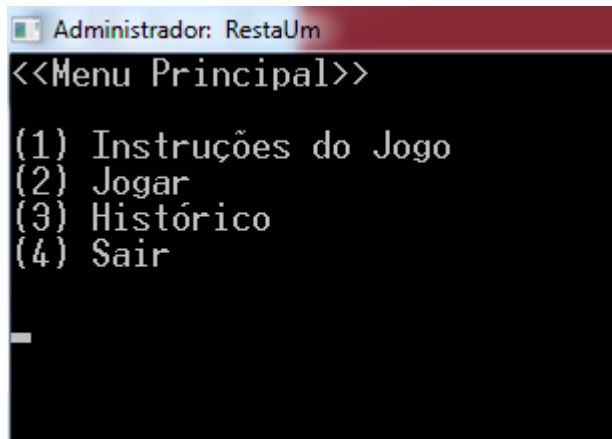
OBJETIVO

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

EXPLICAÇÃO DO SIMULADOR EM SI

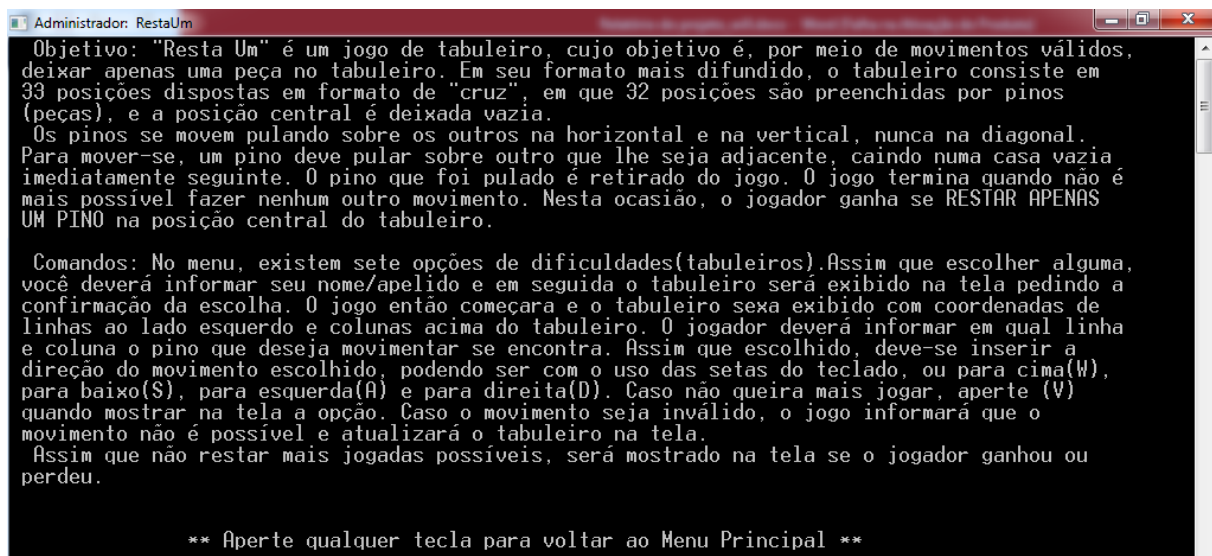
Por meio de movimentos válidos (horizontais ou verticais), pulando “pinos” e ocupando casas vazias, o pino pulado é retirado do jogo gerando mais casas vazias, assim o jogo termina quando restar apenas um “pino” na posição central do tabuleiro do jogo. O jogador perde quando não houver mais jogadas possíveis 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.

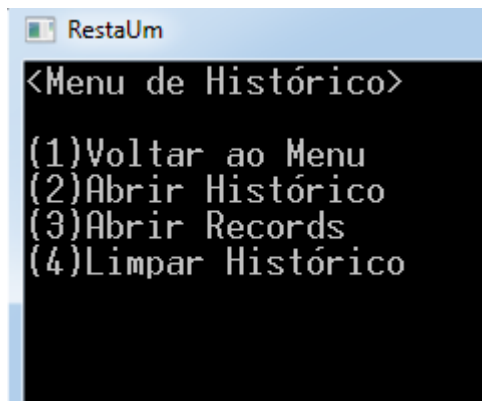


Tela do Menu Principal.

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.

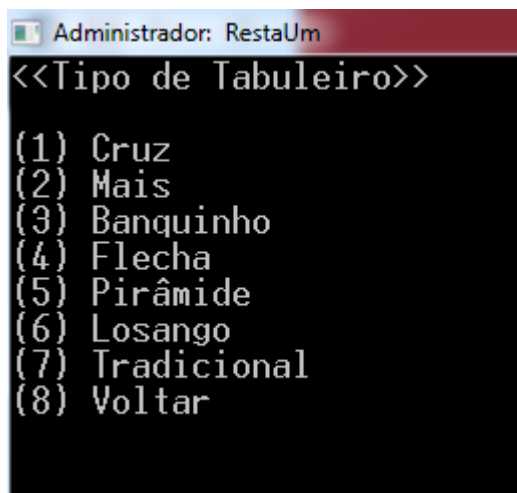
Caso 1, ele volta ao menu principal. Caso 2, abre o arquivo de histórico. Caso 3, abre o arquivo de records. Caso 4 limpa o histórico.

Administrador: RestUm				
Nome	Tabuleiro	Resultado	Jogadas	Tempo
w	Tabuleiro Cruz	Perdeu	6	0:13
will	Tabuleiro Cruz	Ganhou	5	0:10
willian	Tabuleiro Mais	Ganhou	9	0:31
will	Tabuleiro Mais	Ganhou	9	0:24
willian	Tabuleiro Mais	Ganhou	8	0:18
willian	Tabuleiro Banquinho	Ganhou	11	1:2
wil	Tabuleiro Cruz	Ganhou	5	0:09
will	Tabuleiro Cruz	Ganhou	5	0:20
will	Tabuleiro Cruz	Ganhou	5	0:15
will	Tabuleiro Cruz	Ganhou	5	2:50
willian	Tabuleiro Cruz	Perdeu	0	0:09
** Aperte qualquer tecla para voltar ao Menu de Escolha **				

Histórico.

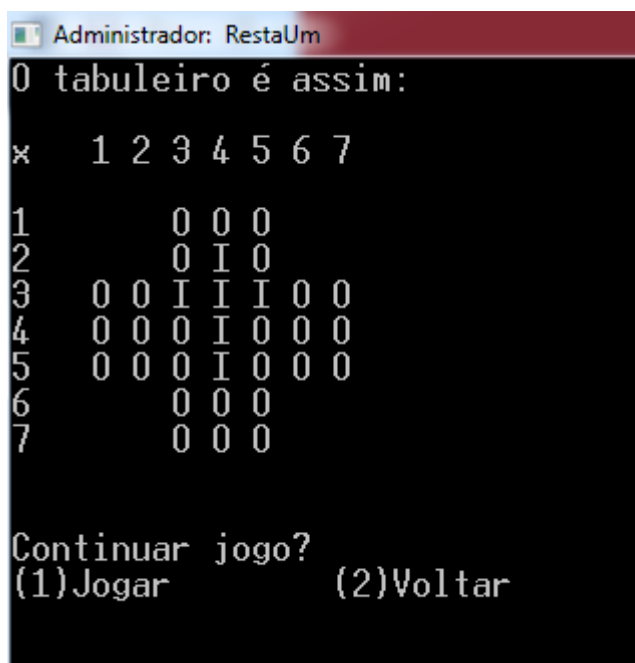
Administrador: RestUm				
Nome	Tabuleiro	Resultado	Jogadas	Tempo
will	Tabuleiro Cruz	Ganhou	5	0:10
willian	Tabuleiro Mais	Ganhou	8	0:18
willian	Tabuleiro Banquinho	Ganhou	11	1:2
will	Tabuleiro Flecha	Ganhou	40	2:30
will	Tabuleiro Pirâmide	Ganhou	40	2:30
will	Tabuleiro Losângo	Ganhou	40	2:30
will	Tabuleiro Tradicional	Ganhou	40	2:30
** Aperte qualquer tecla para voltar ao Menu de Escolha **				

Records.

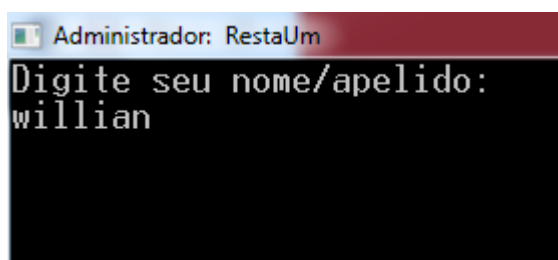


Menu de Tabuleiros.

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

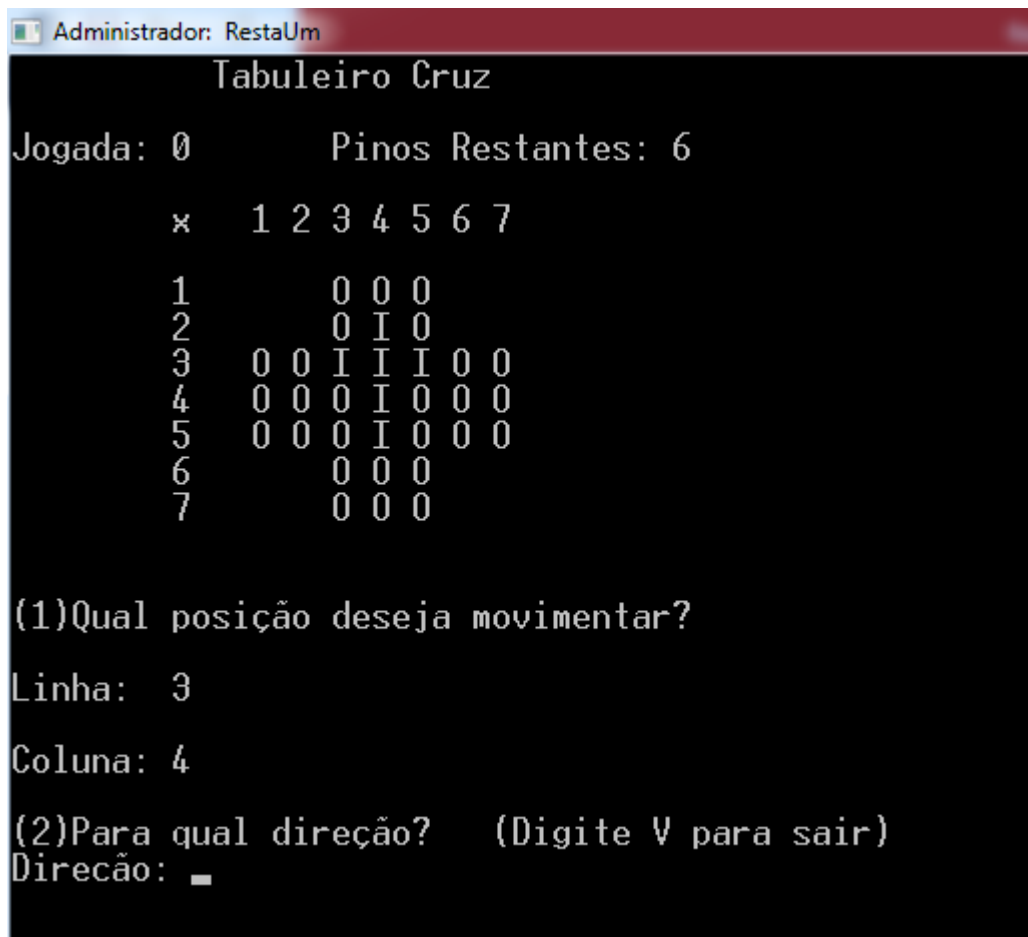


Tabuleiro impresso para visualização.



Esperando que um nom/apelido seja digitado.

Assim que iniciado, o simulador imprime 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).

```
Administrador: RestaUm

Tabuleiro Cruz

Jogada: 5      Pinos Restantes: 1

  x   1 2 3 4 5 6 7
  1       0 0 0
  2       0 0 0
  3    0 0 0 0 0 0 0
  4    0 0 0 I 0 0 0
  5    0 0 0 0 0 0 0
  6       0 0 0
  7       0 0 0

VOCÊ GANHOU!!!

<Menu de Escolha>

(1)Voltar ao Menu
(2)Abrir Histórico
(3)Jogar Novamente
(4)Sair
```

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 ()`.

```

main.c x Menu.c x Tabuleiro.c x Estatistica.c x Principal.c x
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4  #include <locale.h>
5  #include <conio.h>
6  #include <ctype.h>
7  #include <time.h>
8  #include "Menu.h"
9  #include "Tabuleiro.h"
10 #include "Principal.h"
11 #include "Estatistica.h"
12
13
14
15 int main() {
16     setlocale(LC_ALL, "portuguese");
17     system("title RestaUm");
18     Inicio();
19     return 0;
20 }
21
22

```

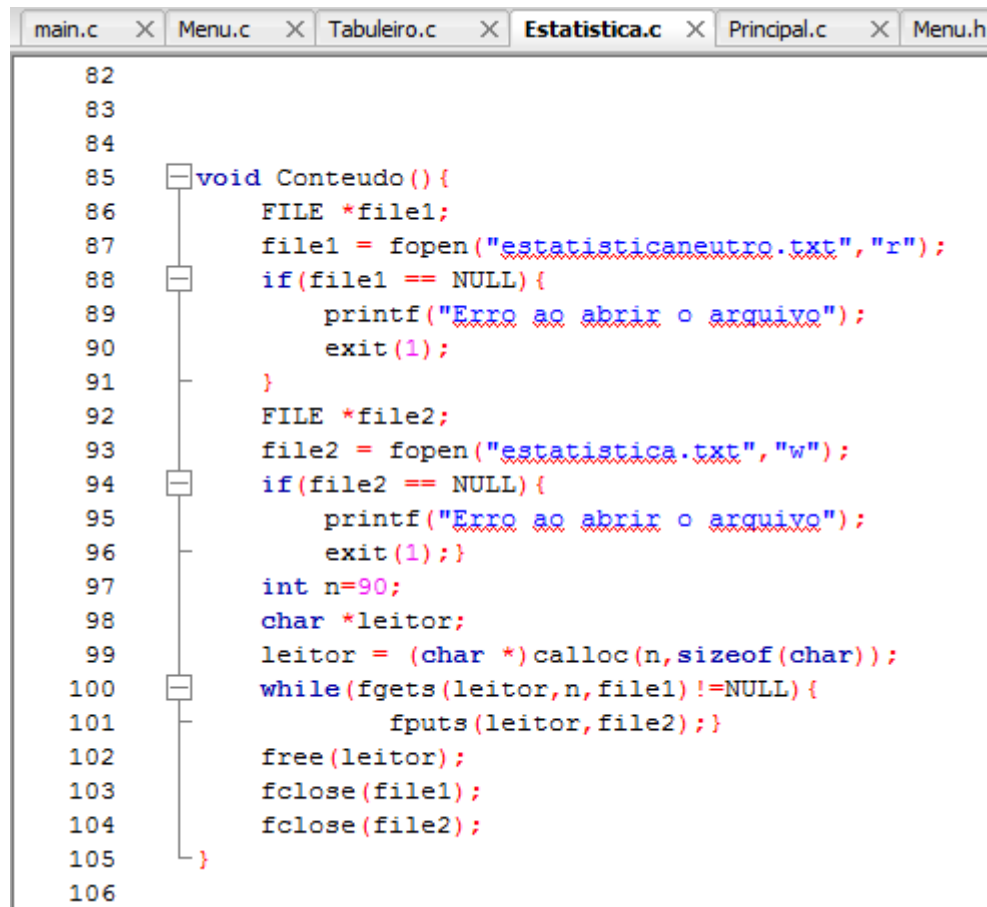
Assim que chamada, é mostrado na tela as opções e então solicitado ao usuário/jogador que insira o número correspondente a opção escolhida. O endereço desse número ($1 < *op < 4$) é então passado como parâmetro de referência para a função menu (int *op), onde passa por uma estrutura de decisão do tipo 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.

```

main.c x Menu.c x Tabuleiro.c x Estatistica.c x Principal.c x Menu.h x Tabuleiro.h x Estatistica.h x Principal.h x
55 }
56
57
58 void menu(int *op) {
59     char opS;
60     FILE *file;
61     switch(*op) {
62         case 1 : system("cls");
63                 file = fopen("regra.txt", "r");
64                 if(file == NULL) {
65                     printf("Erro ao abrir o arquivo");
66                 }
67                 int n=2000;
68                 char *leitura;
69                 leitura = (char *)calloc(n, sizeof(char));
70                 while(fgets(leitura, n, file) != NULL) {
71                     printf("%s", leitura);
72                 }
73                 fclose(file);
74                 free(leitura);
75                 printf("\n\n");
76                 break;
77         case 2 : system("cls");
78                 qualTab();
79                 do {
80                     scanf(" %c", &opS);
81                 } while (opS < 'a' || opS > 'h');
82                 menuS(opS);
83                 break;
84         case 3 : system("cls");

```

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).



```
82
83
84
85 void Conteudo() {
86     FILE *file1;
87     file1 = fopen("estatisticaneutro.txt", "r");
88     if(file1 == NULL) {
89         printf("Erro ao abrir o arquivo");
90         exit(1);
91     }
92     FILE *file2;
93     file2 = fopen("estatistica.txt", "w");
94     if(file2 == NULL) {
95         printf("Erro ao abrir o arquivo");
96         exit(1);
97     }
98     int n=90;
99     char *leitor;
100     leitor = (char *) calloc(n, sizeof(char));
101     while(fgets(leitor, n, file1) != NULL) {
102         fputc(leitor, file2);
103     }
104     free(leitor);
105     fclose(file1);
106     fclose(file2);
107 }
```



```
main.c X Menu.c X Tabuleiro.c X Estatistica.c X Principal.c
205     printf("\n(3)Abrir Records");
206     printf("\n(4)Limpar Histórico\n");
207     do{
208         fflush(stdin);
209         scanf("%d",&opE);
210     }while(opE>4||opE<1);
211     switch(opE){
212     case 1: system("cls");
213             Inicio();
214             break;
215     case 2: system("cls");
216             estatis();
217             menuEstat();
218             break;
219     case 3: system("cls");
220             abreRecord();
221             menuEstat();
222             break;
223     case 4: Conteudo();
224             system("cls");
225             menuEstat();
226             break;
227     default: system("cls");
228              Inicio();
229     }
230 }
```

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.

```
main.c X Menu.c X Tabuleiro.c X Estatistica.c X Principal.c X Menu.h X Tabuleiro.h X Estatistica.h X Principal.h X
4
5
6  /* Definição de variável do tipo struct tabuleiro para utilizar no jogo RestaUm. */
7
8  typedef struct{
9      int contador;
10     int pinos;
11     char resultado[7];
12     char jogador[20];
13     char nome[30];
14     char campo[9][9];
15 }tabuleiro;
16
17
18
19  /* Definição de variável do tipo struct tempo para utilizar como contador de tempo. */
20
21  typedef struct{
22      int mint;
23      int segt;
24  }tempo;
25
26
```

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`.

```
main.c X Menu.c X Tabuleiro.c X Estatistica.c X Principal.c X Menu.h X
101     if(tab == NULL){
102     printf("Erro ao abrir o arquivo");
103     exit(1);}
104     ch=getc(tab);
105     while(ch != EOF){
106         if(aux==n&&ch!='\n'){
107             fscanf(tab,"%s",str);
108             fscanf(tab,"%d",&no);
109         }
110         if (ch == '\n')
111             aux++;
112         if(aux==n+7&&ch!='\n'){
113             fgets(nomi,30,tab);
114             break;
115         }
116         ch = getc(tab);
117     }
118     for(int i=0;i<(strlen(nomi));i++){
119         if(nomi[i]=='\n'){
120             nomi[i]='\0';
121         }
122     }
123     strcpy((*t1).nome,nomi);
124     (*t1).pinos = no;
125     memcpy((*t1).campo,str,sizeof((*t1).campo));
126     fclose(tab);
127 }
```

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.

```

main.c X Menu.c X Tabuleiro.c X Estatistica.c X Principal.c X Menu.h X Tabulei
50 void preVisu(tabuleiro *t1, int n){
51     system("cls");
52     char pos_x[11]="x 1234567 ";
53     char pos_y[10]=" 1234567 ";
54     int no, i, j;
55     int opS;
56     printf("O tabuleiro é assim:\n\n");
57     for(i=0;i<10;i++){
58         printf("%c ",pos_x[i]);
59     }
60     printf("\n");
61     for(i=0;i<9;i++){
62         printf("%c ",pos_y[i]);
63         for(j=0;j<9;j++){
64             if((*t1).campo[i][j]=='*'){
65                 printf(" ");
66             }else{
67                 printf("%c ",(*t1).campo[i][j]);
68             }
69         }
70         printf("\n");
71     }
72     printf("\n");
73     printf("Continuar jogo?\n");
74     printf("(1) Jogar\t(2) Voltar\n");
75     fflush(stdin);
76     // ...

```

```

main.c X Menu.c X Tabuleiro.c X Estatistica.c X Principal.c X Menu.h X Tabuleiro.h X Estatistica.h X Prin
14 void jogEmMenu(tabuleiro *t1, int n){
15     int opT;
16     tempo *temp;
17     temp = (tempo *)malloc(sizeof(tempo));
18     struct tm *hora_atual;
19     struct tm *hora_atual2;
20     int segu, segu2;
21     time_t segundos;
22     time_t segundos2;
23     time(&segundos);
24     hora_atual = localtime(&segundos);
25     segu=hora_atual->tm_sec+60*(hora_atual->tm_hour*60+hora_atual->tm_min);
26     system("cls");
27     Nome(t1);
28     Imprimi(t1);
29     Jogada(t1,temp,segu);
30     time(&segundos2);
31     hora_atual2 = localtime(&segundos2);
32     segu2=hora_atual2->tm_sec+60*(hora_atual2->tm_hour*60+hora_atual2->tm_min);
33     temp->segt=(segu2-segu)%60;
34     temp->mint=(segu2-segu)/60;
35     SalvaEstatist(t1,temp);
36     verifica(t1,temp,n);
37     free(t1);
38     free(temp);
39     escolha();

```

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).



```

49 int ContaTabuleiro(tabuleiro *t1){
50     int volta=0;
51     int i, j;
52     for(i=1;i<9;i++){
53         for(j=1;j<9;j++){
54             if(((t1).campo[i][j]!='*') && ((t1).campo[i][j]=='I') &&
55                (((t1).campo[i+1][j]=='I') || ((t1).campo[i][j+1]=='I') ||
56                ((t1).campo[i-1][j]=='I') || ((t1).campo[i][j-1]=='I'))){
57                 volta = 1;
58             }
59         }
60     }
61     return volta;
62 }

63
64
65
66 int GameOver(tabuleiro *t1){
67     int verifica = 0;
68     int i, j;
69     for(i=1;i<9;i++){
70         for(j=1;j<9;j++){
71             if((t1).campo[i][j]=='I'){
72                 verifica=verifica+1;
73             }
74         }
75     }
76 }

```

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 structs, 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.

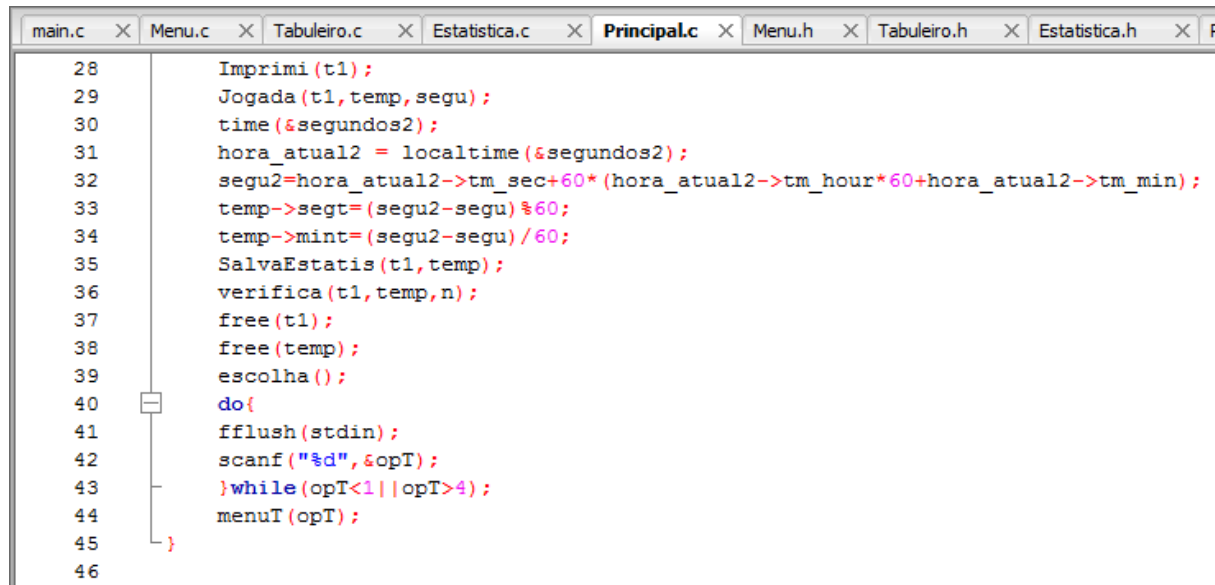
```

main.c x Menu.c x Tabuleiro.c x Estatistica.c x Principal.c x Menu.h x Tabuleiro.h x Estatistica.h x Principal.h x
108 do{
109     fflush(stdin);
110     scanf("%d",&coluna);
111 }while(coluna>9||coluna<1);
112 printf("\n");
113 printf("(2) Para qual direção?\t(Digite V para sair)\n");
114 printf("Direção: ");
115 do{
116     direcao = getch();
117     if(direcao == (int)((char)224)) {
118         direcao = getch();
119         if(direcao==72||direcao==80||direcao==77||direcao==75) break;
120     }
121     else direcao=toupper(direcao);
122 }while(direcao!='W'&&direcao!='A'&&direcao!='S'&&direcao!='D'&&direcao!='V');
123 switch(direcao){
124     case 'V':
125         time(&segundos3);
126         hora_atual3 = localtime(&segundos3);
127         segu3=hora_atual3->tm_sec+60*(hora_atual3->tm_hour*60+hora_atual3->tm_min);
128         temp->segt=(segu3-segu)%60;
129         temp->mint=(segu3-segu)/60;
130         strcpy((t1).resultado,"Perdeu");
131         SalvaEstatist(t1,temp);
132         free(t1);
133         free(temp);
134         system("cls");
135         Inicio();
136     break;
137     case 72:
138     case 'W' :
139         if((t1).campo[linha-2][coluna]=='I' || (t1).campo[linha-2][coluna]=='*'||
140            (t1).campo[linha-1][coluna]=='O' || (t1).campo[linha-1][coluna]=='*'||
141            (t1).campo[linha][coluna]=='O' || (t1).campo[linha][coluna]=='*'){
142             (t1).contador=(t1).contador+1;
143             system("cls");
144             printf("Jogada Inválida\n\n");
145             Imprimi(t1);
146             printf("\n");
147             Jogada(t1,temp,segu);
148         }
149     else{
150         (t1).contador=(t1).contador+1;
151         (t1).pinos=(t1).pinos-1;
152         (t1).campo[linha][coluna]='O';
153         (t1).campo[linha-1][coluna]='O';
154         (t1).campo[linha-2][coluna]='I';
155         system("cls");
156         printf("\n");
157         Imprimi(t1);
158         Jogada(t1,temp,segu);
159     }

```

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 ($1 < \text{opt} < 3$) e chamado uma função de `menuT (opt)`.



```
main.c × Menu.c × Tabuleiro.c × Estatistica.c × Principal.c × Menu.h × Tabuleiro.h × Estatistica.h × F
28     Imprimi (t1);
29     Jogada (t1, temp, segu);
30     time (&segundos2);
31     hora_atual2 = localtime (&segundos2);
32     segu2=hora_atual2->tm_sec+60*(hora_atual2->tm_hour*60+hora_atual2->tm_min);
33     temp->segt=(segu2-segu)%60;
34     temp->mint=(segu2-segu)/60;
35     SalvaEstatist (t1, temp);
36     verifica (t1, temp, n);
37     free (t1);
38     free (temp);
39     escolha ();
40     do{
41         fflush (stdin);
42         scanf ("%d", &opT);
43     }while (opT<1 || opT>4);
44     menuT (opT);
45 }
46
```

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.

```

main.c x Menu.c x Tabuleiro.c x Estatística.c x Principal.c x Menu.h x
166         system("cls");
167         Inicio();
168         break;
169     case 2:
170         system("cls");
171         estatistic();
172         escolha();
173         do{
174             fflush(stdin);
175             scanf("%d",&opT);
176             }while(opT<1||opT>4);
177             menuT(opT);
178             break;
179     case 3:
180         system("cls");
181         qualTab();
182         do{
183             scanf("%d",&opS);
184             }while(opS<1||opS>8);
185             menuS(opS);
186             break;
187     case 4:
188         system("cls");
189         exit(0);
190     default :
191         printf("Opção Inválida !!!\n");
192         system("cls");
193         Inicio();
194

```

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>

