SSC0600 e SSC601 - 2020/1
Trabalho 05 – Qwirkle

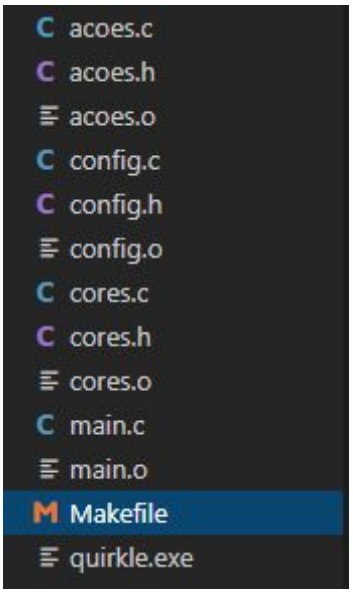
Integrantes do grupo:

Rafael Araujo Tetzner, Nusp 11801136 - **INTEGRANTE 0 DO GRUPO (K=0)**

Alexandre Lopes Ferreira Dias dos Santos, Nusp 11801199 - I**NTEGRANTE 1 DO GRUPO (K=1).**

Descrição da solução:

O programa é dividido em 4 arquivos “.c”, o main.c, o cores.c, config.c e acoes.c:

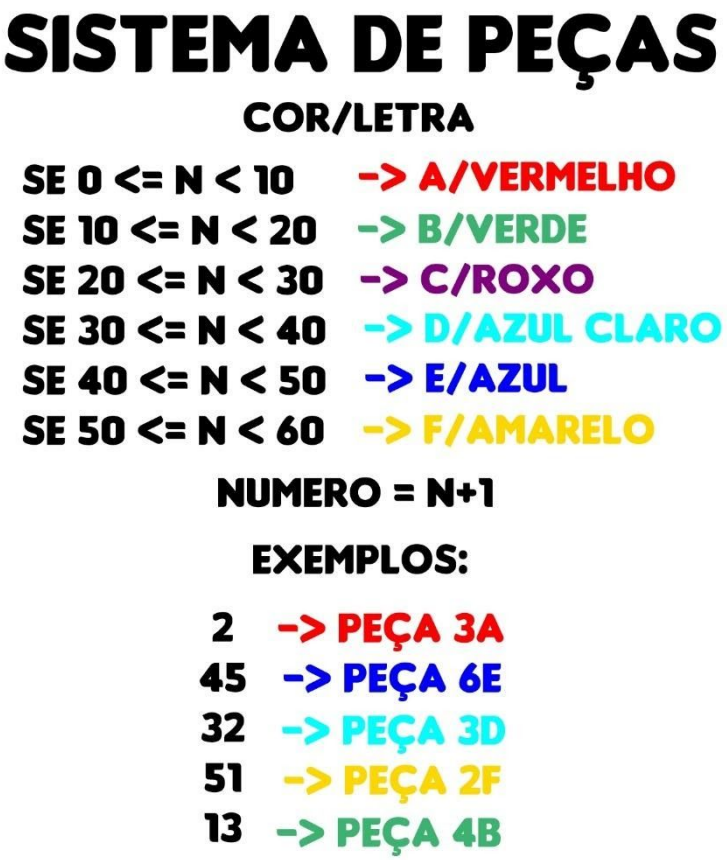


Os jogadores junto com seus respectivos nomes, peças e pontuações, são armazenados em uma struct:

```
typedef struct {
    char nome[128];
    char tiles[6];
    int score;
}jogadores;
```

Sistema de peças:

Com o intuito de simplificar, o sistema de armazenagem das peças funciona da seguinte forma, cada peça é representada por um único número, a cor/letra que esta peça representa é definida pela dezena em que ela se encontra, e o número é n+1 como dita o esquema a seguir:



main.c:

Após a inclusão das bibliotecas, o programa começa armazenando o número de jogadores, seus respectivos nomes e se é com cheat mode ou não.

```c
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<time.h>

#include"cores.h"
#include"config.h"
#include"acoes.h"

#define MAXNOME 128

void main(){
    //*Definições*
    printf("Numero de jogadores: ");
    int numj;
    scanf("%d", &numj);
    getc(stdin); //remove o lixo no buffer do teclado
    jogadores *jogador = malloc (numj * sizeof(jogadores));
    int i, j, k;
    for (i = 0; i < numj; i++){
        printf("Digite o nome do jogador #%d: ", i + 1);
        fgets(jogador[i].nome, MAXNOME, stdin);
        for(j = 0; jogador[i].nome[j] != '\n'; j++);
        jogador[i].nome[j] = '\0';
    }
```

Logo em seguida, a main chama a função config_iniciais do config.c a fim de configurar o saquinho de peças, para depois alocar o tabuleiro, que inicialmente começa como uma matiz (Board[0][0]) e vai expandindo com o decorrer do jogo, iniciando x=1 para número de linhas, e y=1 para colunas.

```
//"Saquinho de pecas"
int tiles[36][3];
config_iniciais(tiles, numj, jogador);

int **board = (int **) malloc (sizeof(int*));
if(board == NULL){
    vermelho();
    printf("Erro de Alocacao...\n");
    padrao();
    exit(1);
}
board[0] = (int *) malloc(sizeof(int));
if(board[0] == NULL){
    vermelho();
    printf("Erro de Alocacao...\n");
    padrao();
    exit(1);
}
board[0][0] = -1;
```

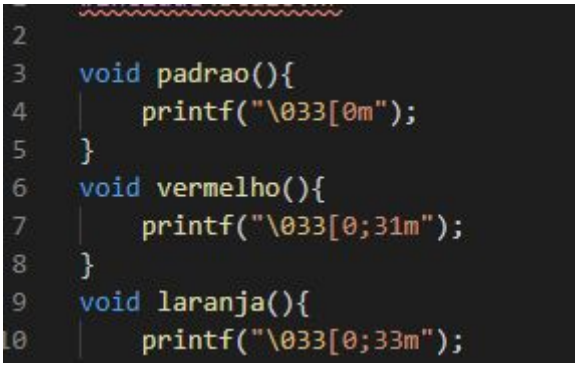
E assim começa o laço principal que controla todo o jogo, limitado por 108 que é o número total de peças, o programa roda com um laço infinito, primeiramente printando o jogador da vez, junto do tabuleiro e suas respectivas peças, dando as 3 opções de possíveis tipos de jogadas, trocar, jogar ou passar.

```
printf("Jogada do(a) jogador(a): %s\n", jogador[i].nome);
print_board(board, x, y);
printf("Pecas disponiveis: ");
print_pecas(jogador[i]);
if(cont_jogadas_rodada == 0){
    printf("opcoes: trocar p1[p2 p3 ...] | jogar p1 x y | passar\n");
} else{
    printf("opcoes: jogar p1 x y | passar\n");
}
char entrada[MAXNOME];
setbuf(stdin, NULL);
fgets(entrada, MAXNOME, stdin);
for(j = 0; entrada[j] != '\n'; j++);
entrada[j] = '\0';
escolha = leitura(entrada);
if(escolha == 1){
    if(cont_jogadas_rodada == 0){
        trocar(i, jogador, entrada, tiles);
    }else{
        vermelho();
```

Armazenando a entrada do usuário com fgets na string “entrada”, puxando uma função de leitura, que confere a validade e qual tipo de ação o

jogador quer realizar, caso escolha==1 ele deseja trocar peças, portanto uma função trocar é chamada, e o laço continua e tudo é printado de novo, porém agora sem a opção de trocar, caso escolha==2, ele deseja jogar chamando a função jogar que retorna 0 caso a jogada não necessite de expansão no tabuleiro ou retorna 1 2 3 ou 4, que é relativo a qual tipo de expansão será necessária, vale ressaltar que se for a primeira jogada, o programa expande a matriz para 3x3 com a primeira jogada no centro. A expansão funciona criando um tabuleiro auxiliar, que copia o tabuleiro antigo, e logo após, é recriado o tabuleiro original com a linha ou coluna expandida, e é colocado todos os valores do auxiliar na sua respectiva posição no tabuleiro original. Caso escolha==3, o programa chama a função score para calcular a pontuação do jogador naquele turno, e logo depois recomeça o laço, agora para o próximo jogador. O jogo termina quando algum jogador esgotar as peças primeiro.

Cores.c

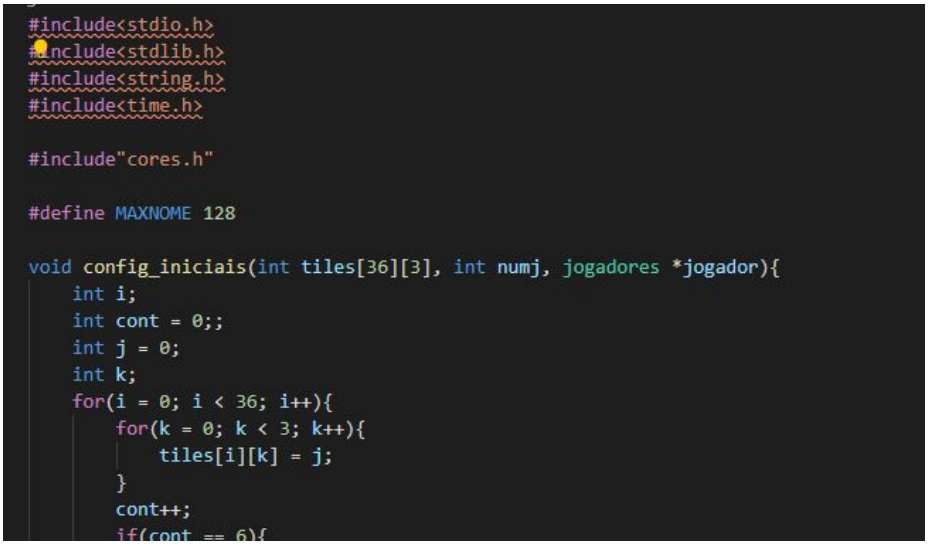


```
2
3   void padrao(){
4       printf("\033[0m");
5   }
6   void vermelho(){
7       printf("\033[0;31m");
8   }
9   void laranja(){
10      printf("\033[0;33m");
```

Aqui é armazenada todas as funções responsáveis por alterar a cor do texto printado no programa.

Config.c

É aqui onde estão todas as funções responsáveis pela configuração do programa.



```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<time.h>

#include"cores.h"

#define MAXNOME 128

void config_iniciais(int tiles[36][3], int numj, jogadores *jogador){
    int i;
    int cont = 0;;
    int j = 0;
    int k;
    for(i = 0; i < 36; i++){
        for(k = 0; k < 3; k++){
            tiles[i][k] = j;
        }
        cont++;
        if(cont == 6){
```

config_inciais()

Esta função é responsável por inicializar o pacote de peças, e as distribuir aleatoriamente entre os jogadores.

printf_pecas()

```
void print_pecas(jogadores jogador){
    int i;
    printf("%s: ", jogador.nome);
    for(i = 0; i < 6; i++){
        if(jogador.tiles[i] >= 0){
            if(jogador.tiles[i] < 10){
                vermelho();
                printf("%dA ", jogador.tiles[i] + 1);
                padrao();
                continue;
            }
            if(jogador.tiles[i] < 20){
                verde();
                printf("%dB ", jogador.tiles[i] - 9);
                padrao();
                continue;
            }
```

Esta função é responsavel por printar as peças atuais de cada jogador.

leitura()

```
int leitura(char *aux){
    int i;
    for(i = 0; aux[i] != ' '; i++);
    char primeira_palavra[MAXNOME];
    strcpy(primeira_palavra, aux);
    primeira_palavra[i] = '\0';
    if(!strcmp(primeira_palavra, "trocar")
        return 1;
    }
    if(!strcmp(primeira_palavra, "jogar")
        return 2;
```

Esta função é responsável por ler a entrada do jogador, e interpretá la retornando 1 para se ele desejar trocar, 2 para jogar e 3 para passar.

acoes.c:

Aqui estão localizadas as funções principais do jogo, incluindo trocar e jogar.

```c
#include"cores.h"
#include"config.h"

#define MAXNOME 128

void trocar(int pos, jogadores *jogador, char *entrada, int tiles[36][3]){
    int i;
    int guardiao[6];
    for(i = 0; i < 6; i++){
        guardiao[i] = jogador[pos].tiles[i];
    }
    int guardiao2[36][3];
    for(i = 0; i < 36; i++){
        int j;
        for(j = 0; j < 3; j++){
            guardiao2[i][j] = tiles[i][j];
        }
    }
    int cont = 0;
    int cont_nump = 0;
```

trocar()

A função começa, criando os guardiões 1 e 2, que copiam as peças do jogador e o pacote de peças, servindo posteriormente como formas de se recuperar de algum erro ocasionado por entradas inválidas do jogador. A função funciona basicamente lendo a entrada, com um laço que termina quando chega no \0, primeiro é identificado qual peça corresponde aquele número na entrada, e é trocada esta, por uma peça escolhida aleatoriamente no saco de peças. Vale ressaltar que se o programa identificar uma entrada inválida, ele utiliza os guardiões para retornar ao estado anterior.

jogar()

```c
int jogar(int pos, jogadores *jogador, char *entrada, int **board, int x, int y, int *value, int nu
    //cood jogada
    int lin;
    int col;
    int i;
    int j;
    //leitura peca
    for(i = 0; entrada[i] != ' '; i++);
    int valor = entrada[i + 1] - 49;
    if(valor > 5 || valor < 0){
        vermelho();
        printf("Entrada Invalida...\n");
        padrao();
        return -1;
    }
    if(entrada[i + 2] == 'A' || entrada[i + 2] == 'a'){
        } else{
            if(entrada[i + 2] == 'B' || entrada[i + 2] == 'b'){
                valor += 10;
            } else{
                if(entrada[i + 2] == 'C' || entrada[i + 2] == 'c'){
```

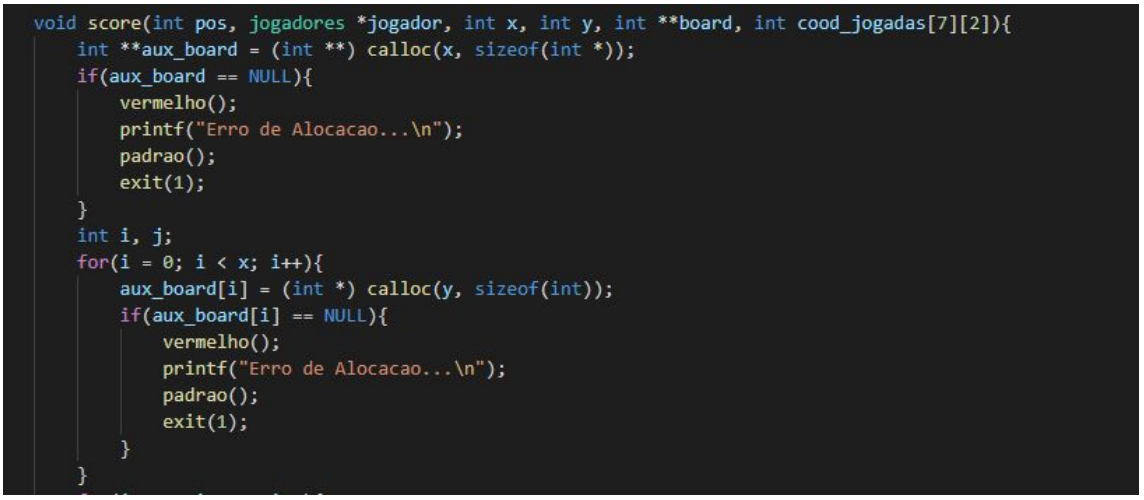
A função começa traduzindo a entrada para a peça específica da jogada, após isso é feita uma verificação para constatar se o jogar possui realmente aquela peça (o que não é realizado com cheat mode ligado), com isso, é feita a leitura da lin e col verificando se a jogada é válida de acordo com as regras do qwirkle. Após este processo, a função entra na parte de verificar se é necessária a expansão do tabuleiro, utilizando 4 flags, uma para cada direção de expansão, as quais, após uma sequencia de testes, chega em um ponto no qual, a flag que estiver levantada vai definir qual vai ser o retorno da função, no qual define a expansão do tabuleiro, que é realizada na main.

```
    }
    if(flag_cima){
        return 1;
    }
    if(flag_baixo){
        return 2;
    }
    if(flag_esquerda){
        return 3;
    }
    if(flag_direita){
        return 4;
    }
    return 0;
}
```

print_board()

```
void print_board(int **board, int x, int y){
    int i, j, z;
    printf("  ");
    for (i=0; i<y; i++){
        printf (" %d ", i+1);
    }
    printf("\n");
```

A função printf_board, imprime o estado do tabuleiro após cada jogada, levando em consideração as regras do sistema de peças para printa-las.



```c
void score(int pos, jogadores *jogador, int x, int y, int **board, int cood_jogadas[7][2]){
    int **aux_board = (int **) calloc(x, sizeof(int *));
    if(aux_board == NULL){
        vermelho();
        printf("Erro de Alocacao...\n");
        padrao();
        exit(1);
    }
    int i, j;
    for(i = 0; i < x; i++){
        aux_board[i] = (int *) calloc(y, sizeof(int));
        if(aux_board[i] == NULL){
            vermelho();
            printf("Erro de Alocacao...\n");
            padrao();
            exit(1);
        }
    }
```

A função score é chamada quando um jogador passa a vez, e serve para calcular a pontuação realizada por este jogador naquele turno. Inicialmente ela cria a matriz aux_board usando calloc, para colocar 0 nos espaços vazios. Utilizando a matriz cood_jogadas que armazenou todas as jogadas deste jogador no turno, o programa troca posições na matriz aux_board para 1, a fim de contabilizar e analisar a posição das jogadas e sua pontuação, posteriormente utilizando o aux_board como base para contabilizar a pontuação e a armazenar na struct do jogador, e por fim, é conferido a ocorrência de qwirkles, por fim printando a pontuação deste turno e a total.

Link para o video da resolução em grupo:

Vídeo 1: Desenvolvendo o trabalho

https://drive.google.com/file/d/1mmowTaXgh2FLcGJktfA9pIxeE17ZzW5l/view?usp=sharing

Vídeo 2: Explicando o código em 10 mins

https://drive.google.com/file/d/1QPbM0CzGXSobi3SQTKM5xKOfhr7WiDwY/view?usp=sharing