Relatório EP1 ED2

Patrícia da Silva Rodrigues

Abril 2022

1 Implementação:

Tabuleiro

O programa foi implementado em c++

A classe pilha.h foi importanda no código para que fosse possível usar a pilha (include "pilha.h").

A construção do tabuleiro é deita da seguinte maneira: O programa pede os valores de n e m que comporão o tabuleiro Mmxn. Em seguida, o programa aloca dinamicamente uma matriz tabuleiro com asterisco usando **malloc()** como na igura abaixo:

```
cout<<"Digite N do tabuleiro de dimensao NxM = 60";
cin>>nT;

cout<<"Digite M do tabuleiro de dimensao NxM = 60";
cin>>mT;


char** M = (char**)malloc(nT * sizeof(sizeof( char*)));

for (i=0;i < nT;++i)
{
    M[i] = (char*)malloc(mT * sizeof(char));
}

for(i = 0; i < nT; i++)
{
    for(j = 0; j < mT; j++)
    {
        M[i][j] = '*';
    }
}
```

Funcoes do programa

criaLetra(int letra, int nP, int mP)

O programa possui uma função chamada **criaLetra(int letra, int nP, int mP)** que pega o código de cada letra (int letra), e as dimensões que essa letra requer (nPxmP). O programa aloca dinamicamente o espaço para a matriz letra com **malloc()**.

Os códigos das letras são:

1 - ausencia de letra, 2 - F, 3 - I, 4 - L, 5 - N, 6 - P, 7 - T, 8 - U, 9 - V, 10 - W, 11 - X, 12 - Y, Z - 13.

As peças ficam todas em organizadas em um vetor esperando serem usadas.

exemplo: Letra F e I:

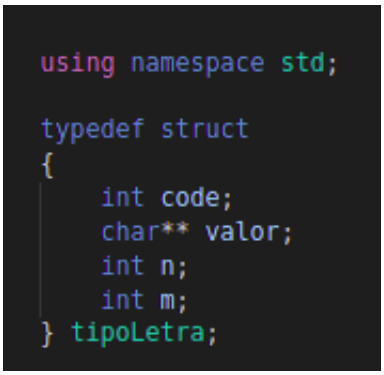
```
if(letra == 2)
{
    Le[0][1] = 'F';
    Le[0][2] = 'F';
    Le[1][0] = 'F';
    Le[1][1] = 'F';
    Le[2][1] = 'F';
    return Le;
}

if(letra == 3)
{
    for(i = 0; i < n; i++)
    {
        for(j = 0; j < m; j++)
        {
            Le[i][j] = 'I';
        }
    }
}
```

giraPeca(char** peca,int n, int m, int oriente)

Outra função que o programa tem é a **giraPeca(char** peca,int n, int m, int oriente)** Que gira a peça como no exemplo abaixo, dependendo da orientação, e devolve a peça virada.

```
"F"     "Ori. 1" "Ori. 2" "Ori. 3"
*FF       *F*      *F*      F**
FF*       FFF      *FF      FFF
*F*       **F      FF*      *F*
```

Dado a struct abaixo que é responsavel por quardar as peças e suas informações



```
using namespace std;

typedef struct
{
    int code;
    char** valor;
    int n;
    int m;
} tipoLetra;
```

tipoLetra* enfileiraLetras()

Foi possível gerar a função **tipoLetra* enfileiraLetras()** que cria as peças letras, as colocam dentro de uma struct com suas respectivas informações de identificação e em seguida as colocam em um vetor alocado dinamicamente com malloc de dimenção **malloc(12 * sizeof(tipoLetra)**, e é usado para enfileirar as peças para que o acesso a todas elas seja possível.

char** geraTabuleiro(int n, int m)

A função **char** geraTabuleiro(int n, int m)** cria Tabuleiro baseado nas dimenções pedidas como é possivel verificar na figura 1. **char****

tabuleiroResolvido(char** tabuleiro, int nT, int mT, Pilha pilha, tipoLetras * vetorLetras)

essa função é responsavel dpor percorrer todo o tabuleiro tentando emcaixar todas as peeças e fazendo backtrack quando necessário e por fim retornar o tabuleiro resolvido char** tabuleiroResolvido(char** tabuleiro, int nT, int mT, Pilha pilha, tipoLetras * vetorLetras)

Além disso ela chama a função encaixou que verifica esse encaixe a cada nova tentativa

bool encaixou(char** tabuleiro, int nT, int mT, Pilha pilha, tipoLetras * vetorLetras, int lT, int cT, int v)

essa função é um função que gereciará para que lado irá a peça baseado nas condições existentes (caso sejá a primeira peça receberá tratamentoe especial) antes de ir para a encaixePeca() e aso não seja, irá diretamente para lá). Depois de todo o tramite o valor bool voltará dizendo se é ou não é possível encaixar a peça no tabueiro. para ela para que ela possa dizer se é ou não é possível o encaixe. Além disso, ela também chama as funções encaixa primeira e encaixa peça a fim de verica se

é o caso de primeiro encaixe ou não.

bool encaixaPrimeira(char tabuleiro, int nT, int mT, char** peca, int nP, int mP, Pilha pilha, tipoLetra * vetorLetras)**

Caso seja a primeira peça, o código fara com que ela gire até se posicionar de modo a não deixar buraco no restante do tabuleiro

bool encaixePeca(char tabuleiro, int nT, int mT, char** peca,int nP, int mP, Pilha pilha, tipoLetras * vetorLetras, int lT,int cT, int v)**

tentara encaixar a peça veriicando todas as duas possibilidades tendo em vista o lugar onde ela quer entraraso a peça tenha apenas uma peça na pilha o que nesse caso equivale a ela estar vazia, ele ira ver se a primeira coluna encaixa no tabuleiro e não h´a necessidade de verificar o resto da peça, pois é a primeira. No entanto,

caso não seja a primeira peça, verificar o encaixe do resto da peça é fundamental

bool verificaPrimeiraCol(char tabuleiro, int nT, int mT, char** peca,int nP, int mP,int lT, int cT)**

verifica se a primeira coluna da peça não confronta com a coluna com do tabuleiro onde ela quer ser encaixada

bool verificaResto(char tabuleiro, int nT, int mT, char** peca,int nP, int mP,int lT, int cT)**

Essa função vericará se, alpem da primeira coluna da peça não conforntar com o tabueiro, se o restante também não confrontará

bool verificaResto(void encaixePronto(char tabuleiro, int nT, int mT, char** peca,int nP, int mP,int lT, int cT, tipoLetra * vetorLetras, int v)**

Apenas encaixará a peça após ver que ela é possível

A função backtracking é a ultima função a ser chamada, pois, dentro da função tabuleiroResolvido(), caso a função encaixou() não sejá true e o vetor vetorLetras já tiver sido completamente percorrido, a função backtracking será executada a fim de dar um paço para trás de desformar a ultima formação A função backtracking é aquela função que sempre que chegarmos ao final de todas as letras testadas ela irá voltar um passo atrás A função desempilhará a pilha (tirará a ultima solução) A função tirará do tabuleiro essa última peça colocada atráveis de contas que se basearão em sua ultima atualização de dimensão (o que

é suficinte para que se saiba onde o programa estava antes) e o formato da peça seja substituido por * no tabuleiro.

Desalocando memória e o método destrutor

No final a pilha é destruída através do destrutor PilhaLetras.$\neg Pilha()$

O tabuleiro também tem toda a sua memória liberada. As peças presentes no vetorLetras também têm seus espaços liberados.