Introdução

O programa tem como objetivo implementar o mundo dos blocos (**101 block problem**), através do arquivo de entrada o programa lerá as instruções que serão executadas no programa, a quantidade de blocos que terá no mundo será passado pelo usuário no arquivo de entrada, além disso será lido também os comandos do usuário.

O programa contará com 4 possibilidades de movimentação dos blocos que serão:

- **Move Onto**: Move o bloco a para cima do bloco b retornando os blocos que já estiverem sobre a ou b para as suas posições originais.
- **Move Over**: Coloca o bloco a no topo do monte onde está o bloco b retornando os blocos que já estiverem sobre a às suas posições originais.
- **Pile Onto**: Coloca o bloco a juntamente com todos os blocos que estiverem sobre ele em cima do bloco b, retornando os blocos que já estiverem sobre b as suas posições originais.
- **Pile Over**: Coloca o bloco a juntamente com todos os blocos que estiverem sobre ele sobre o monte que contém o bloco b.

As funções neste programa serão:

```
int verificaArquivo(FILE *);
int iniciar();
TLista** criaLista(int);
void imprimeBlocos(TLista **,int);
TBloco* encontraBloco(TLista **,int,int,int *);
void retornaBloco(TLista **,TBloco *);
void retiraBloco(TLista **,int,TBloco *);
void imprimeArquivo(FILE *,TLista**,int);
int pileOver(TLista **,int,int,int);
int pileOnto(TLista **,int,int,int);
void empilha(TLista **,TBloco *,TBloco*);
int moveOver(TLista **,int,int,int);
int moveOnto(TLista **,int,int,int);
```

No final do programa um arquivo será preenchido com a posição final dos blocos.

Implementação

O programa conta com duas estruturas, cada bloco será uma estrutura TBloco que terá o número do bloco e um ponteiro para o próximo bloco, a lista será um vetor de lista que terá um ponteiro para o primeiro bloco da lista e terá um para o último bloco da lista, e além disso terá uma variável tipo número que será definido quando a lista for criada junto com os blocos.

```
/*Estrutura dos blocos*/
typedef struct Bloco{
   int numero;
   struct Bloco *next;
}TBloco;
```

```
/*Estrutura da lista*/
typedef struct{
   TBloco *inicio, *fim;
   int numero;
}TLista;
```

Os comandos do programa serão lidos em um arquivo de extensão .txt e a um arquivo .txt receberá o formato final do mundo dos blocos.

As funções foram divididas de acordo com suas funcionalidades, sendo a função iniciar a que irá chamar as outras, as funções são:

int iniciar()

- A função **iniciar** terá como objetivo inicial a abertura do arquivo de entrada e a criação do arquivo de saída, após isso será verificado quantos blocos serão criados, lendo o arquivo de entrada e chamando a função que irá criar o vetor de listas e os blocos, depois de criado os blocos e a lista o programa irá entrar em uma estrutura de loop “do-while” onde até o fim do arquivo será feita a chamada das funções que irá mover o bloco, além disso será feita uma verificação que se os blocos forem iguais será considerado comando inválido, ao sair da estrutura “do-while” o programa chamará uma função que irá escrever a posição final dos blocos no arquivo de saída.

*int verificaArquivo(FILE *arq)*

- A função **verificaArquivo** tem como objetivo receber o arquivo e verificar no caso do arquivo de entrada se ele existe, no caso do arquivo de saída será verificado se foi possível criá-lo se a função receber o arquivo como nulo será retornado 0 para função iniciar e o programa será finalizado.

*TLista** criaLista(int n)*

- A função **criaLista** terá como objetivo retornar o vetor de listas, para isso ele irá receber a quantidade de blocos que será criado que será lido no arquivo de entrada. Cada posição do vetor de lista terá um ponteiro recebendo a posição na memória onde o bloco de mesmo número que o da lista está. Primeiramente será alocada na memória a lista, depois será alocado os indexadores da lista, e então será alocado os blocos na posição que estiver os indexadores, as listas apontarão para os blocos de números iguais, no fim a lista será retornada.

*void imprimeBlocos(TLista **l, int n)*

- A função **imprimeBlocos** terá como objetivo mostrar a posição dos blocos no vetor de listas, para isso a função receberá a lista e a quantidade de blocos foi criado, pois foram criadas as posições na lista de acordo com a quantidade de blocos, os blocos serão lidos a partir de um ponteiro do tipo TBloco que irá receber o primeiro bloco na lista e a cada iteração do loop “while” esse ponteiro receberá o da posição seguinte se existir, o vetor de listas será lido a lista até o fim da mesma através de um loop “for”, a cada alteração feita no mundo dos blocos será chamada essa função que irá mostrar como o mundo está.

*void retornaBloco(TLista **l, TBloco *b)*

- A função **retornaBloco** terá como objetivo retornar os blocos para suas posições originais, a função irá receber a lista e o bloco que terá os blocos que ele aponta retornados para a origem. Na função será criado dois ponteiro do tipo TBloco, o ponteiro p será usado para receber para onde o bloco está apontando e então o bloco irá apontar para NULL, será feito um loop “while” que enquanto p for diferente de NULL o loop será executado, e a cada iteração do loop o bloco que será o p retornará a sua posição original, ara não perder o encadeamento é usado o outo ponteiro que é o aux que irá receber a cada iteração o next do ponteiro do p e então o next do p receberá NULL, e então o p receberá o aux.

*void retiraBloco(TLista **l, int pos, TBloco *b)*

- A função **retiraBloco** terá como objetivo tirar os blocos das posições onde eles estiverem sido encontrados, para isso a função irá receber a lista, a posição onde estiver sido encontrado e o bloco, se o bloco for o único na lista a posição na lista que ele estiver irá apontar para NULL, se não um ponteiro p do tipo TBloco irá receber o bloco que a posição na lista aponta e então terá um loop

“while” que enquanto o p não for NULL ele irá fazer as iterações, se o next do p for o bloco então o p irá receber NULL, se não o p receberá o p next.

TBloco encontraBloco(TLista **l,int b, int n, int *pos)*

- A função **encontraBloco** terá como objetivo retornar o bloco e onde ele está através de uma variável do tipo int passado por referência, para isso a função irá receber a lista, o bloco, a quantidade dos blocos e a variável que irá receber a posição. A função irá percorrer todas as posições na lista e a cada iteração do loop “for” um ponteiro p do tipo TBloco irá receber o início da lista, quando o número no p for igual ao do bloco será retornado o bloco e a variável do tipo ponteiro *pos irá receber a posição onde ele foi encontrado, se ele não for encontrado retorna NULL.

*void empilha(TLista **l, TBloco *b1, TBloco *b2)*

- A função **empilha** tem como função empilhar os blocos, ou seja, o bloco que apontar para NULL irá receber o bloco que será recebido, para isso a função receberá a lista o bloco1 e o bloco 2. O loop while vai percorrer o encadeamento até encontrar o que recebe NULL, quando isso acontecer o next do bloco1 irá receber o bloco2. E então após isso as iterações serão encerradas com um break.

*int moveOnto(TLista **l,int b1, int b2, int n)*

- A função **moveOnto** terá como objetivo mover o bloco a no topo do monte onde está o bloco b retornando os blocos que já estiverem sobre a às suas posições originais., se não for possível tal operação será retornado 0, a função irá receber a lista, os dois blocos e a quantidade de blocos criados. A quantidade de blocos criados será usada para achar onde está o bloco. A função irá receber como retorno da função **encontraBloco** o bloco, e também receberá a posição do bloco através de uma variável passada por referência do tipo int, e então será verificado se a posição do dois é a mesma, se for a mesma a função retorna 0 e é considerado comando inválido. Se as posições dos blocos forem diferentes será verificado se o ponteiro p2 que será o bloco b, aponta para NULL, se isso acontecer o next do ponteiro p2 irá receber o ponteiro p1, se o ponteiro p1 não apontar para NULL será chamada a função **retornaBloco**, após a chamada da mesma será a vez da função **retiraBloco**. Se caso o p2 não apontar para NULL será chamada a função **retornaBloco**, após isso o next do ponteiro p2 irá receber p1 e será chamada a função **retornaBloco** e **retiraBloco** para o p1. No fim será retornado 1 se a movimentação for bem-sucedida.

*int moveOver(TLista **l, int b1, int b2, int n)*

- A função **moveOver** terá como objetivo mover o bloco a para cima do bloco b retornando os blocos que já estiverem sobre a ou b para as suas posições originais, se não for possível tal operação será retornado 0, a função irá receber a lista, os dois blocos e a quantidade de blocos criados. A quantidade de blocos criados será usada para achar onde está o bloco. A função irá receber como retorno da função **encontraBloco** o bloco, e também receberá a posição do bloco através de uma variável passada por referência do tipo int, e então será verificado se a posição do dois é a mesma, se for a mesma a função retorna 0 e é considerado comando inválido. Se as posições dos blocos forem diferentes será verificado se o ponteiro p1 que será o bloco a, aponta para diferente NULL, se isso acontecer será chamada a função **retornaBloco**, depois será chamada a função empilha, após a chamada da mesma será a vez da função **retiraBloco**. Se p1 apontar para NULL será verificado se p2 aponta para NULL, se sim o next do p2 irá receber o p1, caso o p2 não apontar para NULL será chamada a função empilha, após isso será chamada a função **retiraBloco**. No fim da função se for bem-sucedida a movimentação será retornado 1.

*int pileOnto(TLista **l, int b1, int b2, int n)*

- A função **pileOnto** terá como objetivo colocar o bloco a juntamente com todos os blocos que estiverem sobre ele em cima do bloco b, retornando os blocos que já estiverem sobre b as suas posições originais, se não for possível tal operação será retornado 0, a função irá receber a lista, os dois blocos e a quantidade de blocos criados. A quantidade de blocos criados será usada para achar onde está o bloco. A função irá receber como retorno da função **encontraBloco** o bloco, e também receberá a posição do bloco através de uma variável passada por referência do tipo int, e então será verificado se a posição do dois é a mesma, se for a mesma a função retorna 0 e é considerado comando inválido. Se as posições dos blocos forem diferentes será verificado se o ponteiro p2 que será o bloco b, aponta para diferente NULL, se isso acontecer será chamada a função **retornaBloco**, depois será chamada a função empilha, após a chamada da mesma será a vez da função **retiraBloco**. Se p2 apontar para NULL será chamada a função empilha e depois a função **retiraBloco**. No fim da função se for bem-sucedida a movimentação será retornado 1.

*int pileOver(TLista **l, int b1, int b2, int n)*

- A função **pileOver** terá como objetivo colocar o bloco a juntamente com todos os blocos que estiverem sobre ele sobre o monte que contém o bloco b, se não for possível tal operação será retornado 0, a função irá receber a lista, os dois blocos e a quantidade de blocos criados. A quantidade de blocos criados será usada para achar onde está o bloco. A função irá receber como retorno da função **encontraBloco** o bloco, e também receberá a posição do bloco através de uma variável passada por referência do tipo int, e então será verificado se a posição do dois é a mesma, se for a mesma a função retorna 0 e é considerado comando inválido. Se as posições dos blocos forem diferentes será chamada a

função empilha e depois da função empilha será chamada a função **retiraBloco**. No fim da função se for bem-sucedida a movimentação será retornado 1.

*void imprimeArquivo(FILE *saida, TLista **l, int n)*

- A função **imprimeArquivo** terá como objetivo escrever no arquivo a versão final do mundo dos blocos, para isso a função receberá o arquivo de saída, a lista e a quantidade de blocos criados. Será mostrado no arquivo de saída as posições do vetor de lista e os blocos naquelas posições.

Estudo de Complexidade

Função Iniciar():

O(n)

Função verificaArquivo():

O(1)

Função criaLista():

O(n)

Função retornaBloco():

O(n)

Função retiraBloco():

O(n)

Função encontraBloco():

O(n)

Função imprimeBlocos():

O(n)

Função imprimeArquivo():

O(n)

Função moveOnto():

O(1)

Função moveOver():

O(1)

Função pileOnto():

O(1)

Função pileOver():

O(1)

Função empilha():

O(n)

Conclusão

Neste programa foi usado listas para fazer a movimentação dos blocos, foi criado um vetor de listas a partir do número na primeira linha do arquivo de entrada, a partir de então a movimentação dos blocos foi feita a partir dos apontadores seja os apontadores das listas ou até os apontadores dos blocos, chamando funções que fariam a ação de retirar da posição original ou colocar na posição original ou até para empilhar.

No fim as maiores dificuldades para fazer o programa foi no começo do desenvolvimento a incapacidade de saber para onde os apontadores apontavam que foi algo que foi se desenvolvendo ao longo do desenvolvimento do programa

Referências

C Como Programar (Deitel, H., Deitel, P.).

www.cplusplus.com

ruslanledesma.com/2016/03/18/the-blocks-problem.html