



UNIVERSIDADE FEDERAL DO PIAUÍ – UFPI
CAMPUS SENADOR HELVÍDIO NUNES DE BARROS – PICOS
BACHARELADO EM SISTEMAS DE INFORMAÇÃO
PROFESSOR: Juliana Oliveira de Carvalho

Trabalho de Estruturas de dados 1

Autor:

David Marcos Santos Moura

Picos, Maio de 2021

- **Resumo do projeto:**

Inicialmente vamos falar um pouco sobre Estruturas de dados, **Estrutura de dados** é o ramo da computação que estuda os diversos mecanismos de organização de dados para atender aos diferentes requisitos de processamento. Posto isso , este relatório tem como objetivo demonstrar e analisar como é utilizados na prática conceitos de manipulação de dados para solução de alguns problemas.

- **Introdução:**

Neste relatório vamos analisar como foi resolvido alguns problemas, como, criação de um sistema de controle de estacionamento de carros, desenvolvimento de uma algoritmo capaz de transformar uma expressão infixa para pós-fixa, e criação de um algoritmo de escalonamento de processos . Para solução destes problemas foi utilizada a linguagem de programação C e foram empregados alguns conceitos importantes para resolução de tais problemas, dentre eles manipulação de estruturas de dados como pilha e fila.

- **Seções específicas:**

Neste tópico iremos abordar de maneira resumida como foi solucionado os problemas propostos:

Problema 1 e 2:

Resumo do problema:

O estacionamento Picoense contém uma única alameda que guarda até 10 carros. Existe uma entrada e uma saída, de tal forma que quando um determinado carro entra ele fica no final da fila e o primeiro que chegou sempre fica próximo a saída,. Faça um programa em C, onde o usuário entrará com o número da placa do carro e 'E' se estiver entrando no estacionamento e 'S' se estiver saindo do estacionamento. O programa deve emitir uma mensagem sempre que um carro entrar ou sair do estacionamento. Quando um carro sair, a mensagem deverá incluir o número de vezes em que o carro foi manobrado para fora do estacionamento para permitir que outros carros saíssem, além de mostrar quantos carros foram manobrados para que ele saísse.

Solução questão 1 e 2 :

Para a solução deste problema primeiramente pedimos para que o usuário digite qual a operação que ele quer executar “E” ou “S”, se o usuário escolher “E” quer dizer que ele quer estacionar um carro ou “S” para retirar um carro do estacionamento.

Para solução deste problema dividimos o código em algumas funções, iremos analisar algumas das funções mais importantes para resolução do problema:

funções para fila estática:

f_cheia(): Esta função recebe a fila de carro e retorna 1 se o estacionamento estiver cheio e 0 se não tiver.

f_vazia(): Esta função recebe a fila de carro e retorna 1 se o estacionamento estiver vazio e 0 se não tiver.

Enfileira(): Recebe um carro e se a fila não estiver cheia é insere o carro na fila, o carro será sempre inserido na última posição da fila.

Desenfileira(): Retira um carro da fila se a fila não estiver vazia, o carro será removido sempre o que estiver na primeira posição da fila, depois os carros que estavam atrás do carro retirado são movimentados uma posição à frente na fila e retorna o carro removido da fila.

Funções para fila dinâmica :

aloca(): esta função recebe uma estrutura do tipo fila, e os dados de um carro então, esta função irá alocar um espaço na memória e armazenará os dados do carro neste espaço e retorna para a função a estrutura alocada.

inicia(): Recebe o início e o fim de uma fila e inicia a fila fazendo o início e o fim receber Null.

Enfileira(): Recebe o início e o fim de uma fila e um Nó contendo os dados do carro, então se a fila estiver vazia o início e o fim recebem o No, se não tiver, então somente o Fim da fila recebe o No.

desenfileira(): Recebe o início e o fim de uma fila, e uma estrutura do tipo carro, então se a fila não for vazia, faz com que a variável do tipo carro recebida pela função receba os dados do carro que está o início da fila, e depois o carro que está no início é removido da fila e o início da fila é alterado, por fim o carro retirado da fila é retornado pela função.

Obs: As funções a seguir busca() e Sair() funcionam através de uma lógica semelhante tanto na fila dinâmica quanto estática.

Busca(): Recebe a fila do estacionamento e uma placa de um carro então verifica se o carro já está no estacionamento ou não, este processo é feito da seguinte forma é criado uma fila auxiliar, então é desenfileirado os carros da fila de de entrada e enfileirado na fila auxiliar se o carro for encontrado é retornado a variável flag que terá o valor 1 se o carro estiver na fila de entrada e 0 se não tiver , no fim a fila de entrada recebe a fila auxiliar, contendo os carros que foram de desenfileirados.

sair(): Recebe a fila do estacionamento e uma placa de um carro então o carro será retirado da fila do estacionamento, este processo é feito da seguinte forma, primeiramente é verificado se o carro está na primeira posição da fila, se tiver o carro é removido e retornado pela função, se ele não estiver no início então é criado uma fila auxiliar, os carros da fila de entrada são desenfileirados e enfileirados na fila auxiliar enquanto o carro não é retirado da fila os carros que estão na frente são incrementados seu número de movimentações que indica a quantidade de vezes que o carro foi movido para remover o carro escolhido e também é contado quantos carros foram movimentados para retirar o carro escolhido da fila e este valor é retornado pela função, por fim a fila de entrada recebe a fila auxiliar e retornado o carro removido.

Então se o usuário escolher a opção “E” é chamado a função **f_cheia()** para verificar se estacionamento está cheio, se não tiver, então é pedido para que o usuário digite a placa do carro , após esta operação é verificado se a placa do carro é válida para isso é chamada a função **busca()** para verificar se o carro já está na fila , se nao tiver é inserido o carrro na fila do estacionamento utilizando a função **enfileirar()**, se o usuário escolher a opção “S” é chamado a função **f_vazia()** para verificar se estacionamento está vazio, se não tiver, então é pedido para que o usuário digite a placa do carro, então se o carro estiver na fila o carrro é removido da fila utilizando a função **sair()** então é mostrado para o usuário a quantidade de vezes que o carro foi movido para retirar outros carros e quantos carros foram movidos para remover o carro escolhido.

Problema 3 e 4:

Resumo do problema:

Faça um programa em C onde o usuário digita uma expressão matemática no modo in-fixa e então o programa verifica se a expressão é válida, depois use pilha estática para converter para o modo pós-fixa

Solução:

Para solução deste problema dividimos o código em algumas funções, iremos analisar algumas das funções mais importantes para resolução do problema:

Funções para pilha estática:

p_cheia(): Esta função recebe uma pilha de caracteres e retorna 1 se a pilha estiver cheia e 0 se não tiver.

p_vazia(): Esta função recebe uma pilha de caracteres e retorna 1 se a pilha estiver vazia e 0 se não tiver.

Enfileira(): Recebe recebe uma pilha de caracteres, se a pilha não estiver cheia é insere o caractere na pilha, o caráter será sempre inserido no topo da pilha.

Desenfileira(): Recebe recebe uma pilha de caracteres, se a pilha não estiver vazia é removido um caractere na pilha, o caráter será sempre retirado do topo da pilha, e no fim retorna o caracter removido.

Funções para pilha dinamica:

aloca(): esta função recebe uma estrutura do tipo pilha, esta função irá alocar um espaço na memória retorna para a função a estrutura alocada.

Empilha(): Recebe uma pilha de caracteres , um Nó alocado, e um caráter, então faz com que o Nó receba o caráter, depois o topo da pilha recebe o No.

Desempilha(): Recebe uma pilha de caracteres , então verifica se a pilha está vazia, se não tiver faz com que uma variável do tipo char receba o caráter, então remove o caractere que estiver no topo da pilha e depois altera o topo da pilha e retorna o caracter removido.

Obs: As funções a seguir funcionam através de uma lógica semelhante tanto nas pilha dinâmica, quanto na estática.

Eh_numero(): Recebe um caractere e retorna o 0 se for numero e 1 se não é.

Eh_operador(): Recebe um caractere e retorna o 0 se for um operador e 1 se não é.

In_fixa(): Recebe uma string contendo uma expressão no modo in-fixa, e devolve uma string contendo a expressão transformada no modo pós-fixa, este processo é feito da seguinte forma , primeiramente declaramos uma estrutura do tipo pilha, esta estrutura irá armazenar os operadores e parênteses da expressão de entrada, então percorremos a string de entrada enquanto ela não for vazia e verificamos caracter por caracter , se o caractere for um "(" empilhamos o caracter na pilha de operadores, se o operador for ")" , iremos desempilhar os caracteres da pilha de operadores e inserir os caracteres desempilhados na string pos-fixa, até que seja

encontrado um "(", se o caractere for um operador '+' ou '-' ,então se a pilha de operadores não for nula, iremos desempilhar os caracteres da pilha e inserir na string pos-fixa, enquanto a pilha não for vazia ou seja encontrado um "(", se o caractere for um operador '*' ou '/' ,então se a pilha de operadores não for nula, iremos desempilhar os caracteres da pilha e inserir na string pos-fixa, enquanto a pilha não for vazia ou seja encontrado um "(", ou seja encontrado um operador de menor precedência como "+" é "-" e por último se o caractere for um número ele é inserido na string pos-fixa , e este processo é feito varias vezes ate que tenha sido percorrido todos os caracteres da string de entrada, por fim é devolvido a string convertida no modo pos-fixa.

divide expressão(): recebe uma string contendo a expressão e retorna a expressão dividida em um vetor de strings, o vetor vai conter os números , operadores e parênteses da expressão, esta divisão é feita pelos espaços da expressão, por fim retorna flag contendo 0 se a operação foi bem sucedida e 1 se ocorreu algum erro.

válida expressão(): Recebe um vetor de string contendo os os números , operadores e parênteses da expressão, e verifica se a expressão é válida, retorna 0 se a expressão é válida e 1 se a expressão for inválida, essa verificação é feita da seguinte forma, verifica se a expressão não inicia com um número ou parênteses, verifica se os números e operadores estão na ordem certa, verifica se os parênteses da expressão estão corretos ou seja verifica se um parêntese é aberto e fechado da forma certa e verifica se a expressão termina com um número ou fecha parêntese, e se a expressão não atender aos requisitos listados é retornado 1 para invalidar expressão.

Por fim, quando o código é executado, pedimos para que o usuário digite a expressão e depois chamamos a função **divide_expressao()** para dividir a expressão em um vetor , depois passamos o vetor para a função **valida expressão()** para verificar se a expressão é válida, se for válida , chamamos a função **ln_fixa()** para transformar a string em pos-fixa, e imprimimos a string transformada para o usuário.

Problema 5:

Resumo do problema:

Faça um programa em C que simule um escalonador é uma parte do Sistema Operacional que escolhe de uma fila de prontos o próximo processo a ocupar o processador. Suponha que nosso escalonador mantenha 3 filas dinâmicas, de acordo com a prioridade do processo.

Solução:

Para solução deste problema foi desenvolvido as seguintes funções:

aloca(): esta função recebe uma estrutura do tipo fila, e os dados de processo então , esta função irá alocar um espaço na memória e armazenará os dados do processo neste espaço e retorna para a função a estrutura alocada.

inicia(): Recebe o início e o fim de uma fila e inicia a fila fazendo o início e o fim receber Null.

Enfileira(): Recebe o início e o fim de uma fila e um Nó contendo os dados do processo, então se a fila estiver vazia o início e o fim recebem o No, se não tiver , então somente o Fim da fila recebe o No.

desenfileira(): Recebe o início e o fim de uma fila , e uma estrutura do processo , então se a fila não for vazia, faz com que a variável do tipo processo recebida pela função receba os dados do processo que está o início da fila , e depois o processo que está no início é removido da fila e o início da fila é alterado, por fim o processo retirado da fila é retornado pela função.

executa_processo(): Esta função funciona semelhante a função desenfileira, porém essa função simula a execução de um processo , ela ira receber como parâmetro o início eo fim de um fila e uma estrutura do tipo processo , então o processo que estiver no início da fila , tem os seu tempo de processamento decrementado 1s e é incrementado a quantidade de vezes que o processo foi executado, depois o processo é removido da fila e é retornado na estrutura processo, o processo removido.

mostrar_processo(): Esta função mostra para o usuário o próximo processo que será executado pelo escalonador.

Escalonador(): Esta função recebe 3 filas contendo processos e escalona um processo para ser executado pelo processador, este escalonamento é feito com base na prioridade das filas, os processos que serão executados primeiro são os que estão na fila 1 de maior prioridade, depois é executado os processos da fila 2 de prioridade menor e por último é executado os processos da fila 3 que possui a menor prioridade de todas, nas filas 1 e 2 cada processo é executado chamando a função **executa_processo()**: após a execução é verificado se o processo foi finalizado , se não foi é verificado a quantidade de vezes que o processo foi executado , se ele só foi executado 1 vezes ele é enfileirado na mesma fila, se ele ja foi executado 2 vezes então o processo é enfileirado na fila de prioridade abaixo, se o processo estiver na fila 3 de menor prioridade, então ele será executado várias vezes até que o processo seja finalizado.

mostrar_processo_executar_cpu(): Esta função recebe as filas de processos e Imprime para o usuário os processos da fila escolhida.

quantidade_processos(): Esta função recebe uma fila de processos e conta a quantidade de processos que a fila possui e retorna o valor.

quant_tempo_executar_processos(): Esta função recebe uma fila e conta a quantidade de tempo para executar todos os processos da fila, ela conta somente os processos das filas 1 e 2, este cálculo é feito da seguinte forma , os processos são de enfileirados da fila de entrada e enfileira numa fila auxiliar então a cada processo desenfileirado é verificado a quantidade de vezes que o processo foi executado , se o processo já tiver sido executado 1 vez incrementa 1s ao tempo , ou se processo não tiver sido executado nenhuma vez incrementa 2s ao tempo, no fim é retornado o tempo total para executar os processo da fila ,e a fila de entrada recebe a fila auxiliar.

quant_tempo_executar_processos2(): Esta função funciona semelhante a função explicada anteriormente , porém a função recebe uma fila e conta a quantidade de tempo para executar todos os processos da fila, ela conta somente os processos das filas 3, este cálculo é feito da seguinte forma , os processos são desenfileirados da fila de entrada e enfileira numa fila auxiliar então a cada processo desenfileirado é incrementado na variável tempo o tempo de execução do processo, no fim é retornado o tempo total para executar os processo da fila ,e a fila de entrada recebe a fila auxiliar.

processo_existe(): Esta função recebe uma fila e um código de um processo e verifica se o processo existe é retornado 1 se o processo existir e 0 se não.

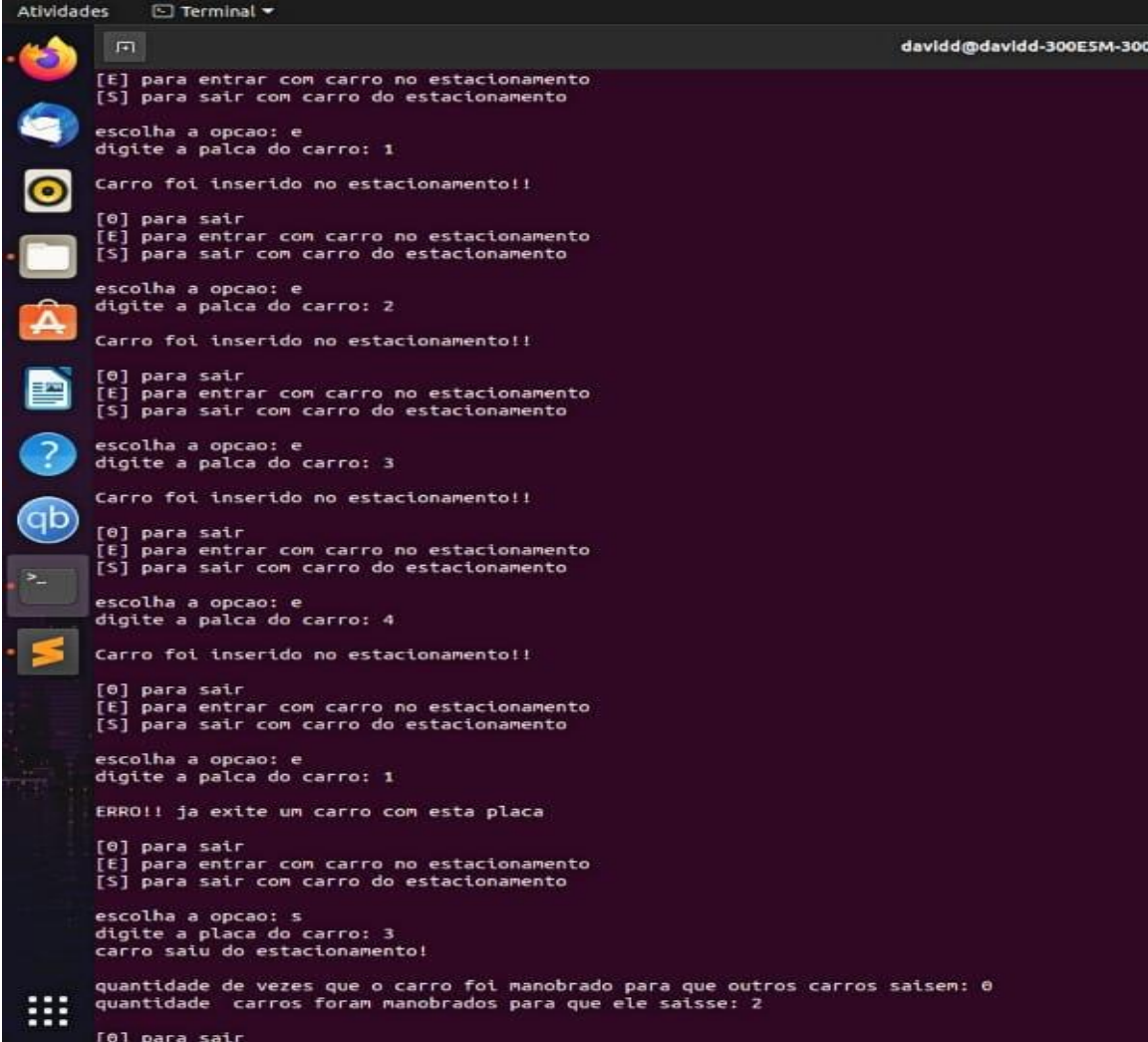
inserir_processo(): Esta função ler os dados de um processo e insere o processo em uma das filas, o processo é inserido na fila de acordo com a sua prioridade, se a prioridade do processo for 1 o processo é inserido na fila 1, se a prioridade do processo for 2 o processo é inserido na fila 2 e se a prioridade do processo for 3 o processo é inserido na fila 3.

- **Resultados da Execução do Programa:**

Neste tópico iremos analisar dado algumas entradas como os codigos dao suas saídas:

Todas as questões a seguir foram executadas pelo terminal linux.

Questão 1 e 2:



```
Atividades Terminal
daviddd@daviddd-300E5M-300

[E] para entrar com carro no estacionamento
[S] para sair com carro do estacionamento

escolha a opcao: e
digite a palca do carro: 1
Carro foi inserido no estacionamento!!

[0] para sair
[E] para entrar com carro no estacionamento
[S] para sair com carro do estacionamento

escolha a opcao: e
digite a palca do carro: 2
Carro foi inserido no estacionamento!!

[0] para sair
[E] para entrar com carro no estacionamento
[S] para sair com carro do estacionamento

escolha a opcao: e
digite a palca do carro: 3
Carro foi inserido no estacionamento!!

[0] para sair
[E] para entrar com carro no estacionamento
[S] para sair com carro do estacionamento

escolha a opcao: e
digite a palca do carro: 4
Carro foi inserido no estacionamento!!

[0] para sair
[E] para entrar com carro no estacionamento
[S] para sair com carro do estacionamento

escolha a opcao: e
digite a palca do carro: 1
ERRO!! ja existe um carro com esta placa

[0] para sair
[E] para entrar com carro no estacionamento
[S] para sair com carro do estacionamento

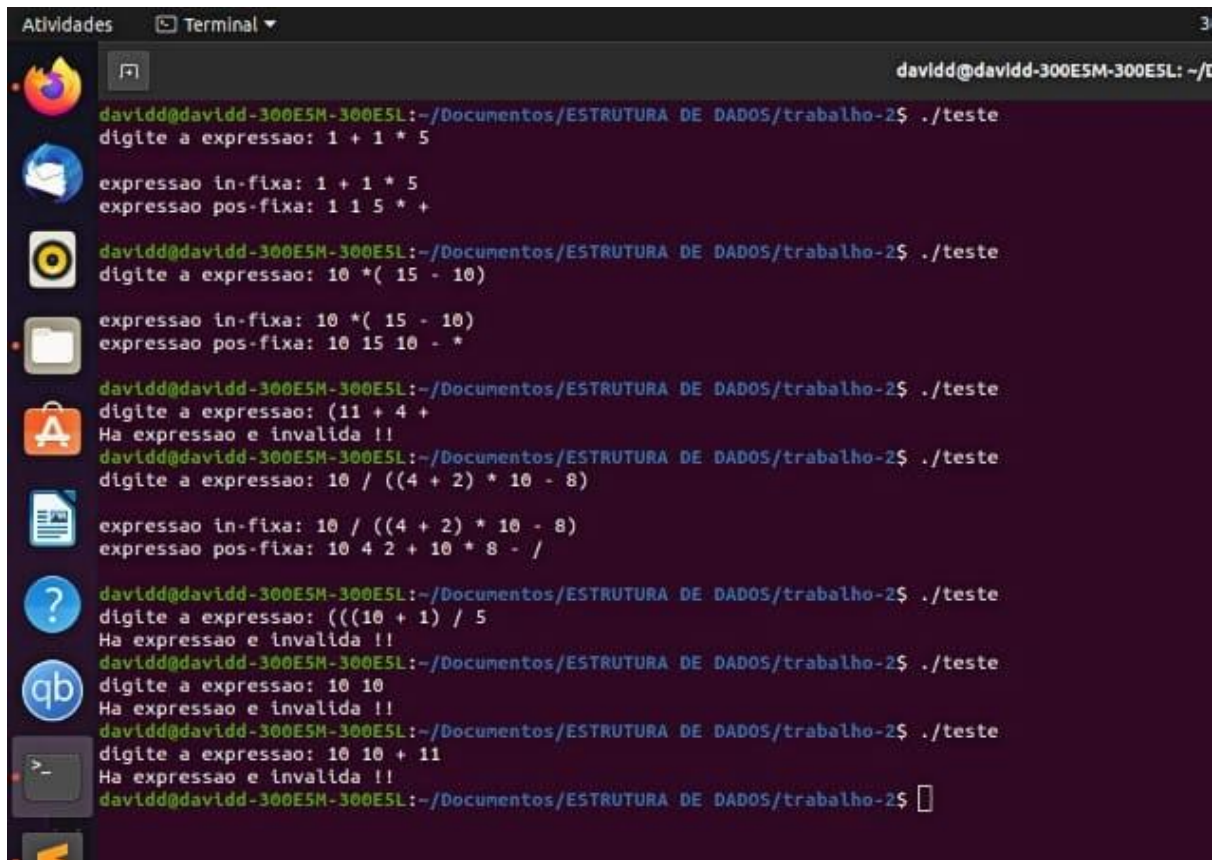
escolha a opcao: s
digite a placa do carro: 3
carro saiu do estacionamento!

quantidade de vezes que o carro foi manobrado para que outros carros saisem: 0
quantidade carros foram manobrados para que ele saisse: 2

[0] para sair
```

Na imagem acima podemos ver como o código da a sua saída quando o usuario insere um carro no estacionamento e quando o usuário retira um carro podemos ver que quando um carro sai do estacionamento é mostrado a quantidade de vezes que o carro foi movimentado para retirar outros carros do estacionamento e a quantidade de carros que foram manobrados para retirar o carro escolhido.

Questão 3 e 4:



```
Atividades Terminal 3
daviddd@daviddd-300E5M-300E5L:~/Documentos/ESTRUTURA DE DADOS/trabalho-2$ ./teste
digite a expressao: 1 + 1 * 5
expressao in-fixa: 1 + 1 * 5
expressao pos-fixa: 1 1 5 * +

daviddd@daviddd-300E5M-300E5L:~/Documentos/ESTRUTURA DE DADOS/trabalho-2$ ./teste
digite a expressao: 10 *( 15 - 10)
expressao in-fixa: 10 *( 15 - 10)
expressao pos-fixa: 10 15 10 - *

daviddd@daviddd-300E5M-300E5L:~/Documentos/ESTRUTURA DE DADOS/trabalho-2$ ./teste
digite a expressao: (11 + 4 +
Ha expressao e invalida !!
daviddd@daviddd-300E5M-300E5L:~/Documentos/ESTRUTURA DE DADOS/trabalho-2$ ./teste
digite a expressao: 10 / ((4 + 2) * 10 - 8)
expressao in-fixa: 10 / ((4 + 2) * 10 - 8)
expressao pos-fixa: 10 4 2 + 10 * 8 - /

daviddd@daviddd-300E5M-300E5L:~/Documentos/ESTRUTURA DE DADOS/trabalho-2$ ./teste
digite a expressao: (((10 + 1) / 5
Ha expressao e invalida !!
daviddd@daviddd-300E5M-300E5L:~/Documentos/ESTRUTURA DE DADOS/trabalho-2$ ./teste
digite a expressao: 10 10
Ha expressao e invalida !!
daviddd@daviddd-300E5M-300E5L:~/Documentos/ESTRUTURA DE DADOS/trabalho-2$ ./teste
digite a expressao: 10 10 + 11
Ha expressao e invalida !!
daviddd@daviddd-300E5M-300E5L:~/Documentos/ESTRUTURA DE DADOS/trabalho-2$
```

Na imagem acima podemos ver como o código dá a sua saída quando o usuário insere algumas expressões, podemos ver como a função se comporta quando o usuário digita uma expressão válida então o código dá a sua devida resposta e quando o usuário digita uma expressão inválida é mostrado uma mensagem de erro.

Questão 5:

```
Atividades Terminal
daviddd@daviddd-300ESM-300ESL: ~/...

-----
1- Inserir processo
2- Mostrar processos das filas
3- Executar processo
4- Mostrar proximo processo que ira utilizar o processador
5- Mostrar quantidade de processos de cada Fila
6- Mostrar quanto de tempo que falta para executar processos de uma Fila
7- Mostrar quanto de tempo processamento falta para terminar cada fila e todas filas
8- Sair
-----

digite opcao: 1
digite o numero do processo: 123
digite o tempo de execucao processo: 5
digite a prioridade do processo: 1
-----

processo inserido na fila 1 de maior prioridade!!
-----

1- Inserir processo
2- Mostrar processos das filas
3- Executar processo
4- Mostrar proximo processo que ira utilizar o processador
5- Mostrar quantidade de processos de cada Fila
6- Mostrar quanto de tempo que falta para executar processos de uma Fila
7- Mostrar quanto de tempo processamento falta para terminar cada fila e todas filas
8- Sair
-----

digite opcao: 1
digite o numero do processo: 4
digite o tempo de execucao processo: 2
digite a prioridade do processo: 3
-----

processo inserido na fila 3 que possui menor prioridade de todas!!
-----
```

```
Atividades Terminal
daviddd@daviddd-300ESM-300ESL: ~/...

-----
1- Inserir processo
2- Mostrar processos das filas
3- Executar processo
4- Mostrar proximo processo que ira utilizar o processador
5- Mostrar quantidade de processos de cada Fila
6- Mostrar quanto de tempo que falta para executar processos de uma Fila
7- Mostrar quanto de tempo processamento falta para terminar cada fila e todas filas
8- Sair
-----

digite opcao: 3
-----

processos da fila 1:
-----

processo 123 executado!!
-----

processos da fila 1 finalizados
-----

1- Inserir processo
2- Mostrar processos das filas
3- Executar processo
4- Mostrar proximo processo que ira utilizar o processador
5- Mostrar quantidade de processos de cada Fila
6- Mostrar quanto de tempo que falta para executar processos de uma Fila
7- Mostrar quanto de tempo processamento falta para terminar cada fila e todas filas
8- Sair
-----

digite opcao: 3
-----

processo da fila 2:
-----

processo 123 executado!!
-----

1- Inserir processo
2- Mostrar processos das filas
3- Executar processo
4- Mostrar proximo processo que ira utilizar o processador
5- Mostrar quantidade de processos de cada Fila
6- Mostrar quanto de tempo que falta para executar processos de uma Fila
7- Mostrar quanto de tempo processamento falta para terminar cada fila e todas filas
8- Sair
-----
```

Nas imagens acima podemos ver como o código se comporta quando o usuário insere um processo na fila , então vemos que é mostrado para o usuário a fila no qual o processo foi inserido, podemos ver também como o código da sua saída quando o usuário escolhe a opção de executar um processo , então podemos ver que o código mostra a fila no qual o processo foi executado e mostra o código do processo que foi executado.

Conclusão:

Por fim, podemos concluir que este trabalho teve como objetivo proporcionar para os estudantes um aprofundamento em conceitos importantes para a disciplina como manipulação de dados ,processamento de dados, manipulação de ponteiros e estruturas de dados como pilha e fila, então podemos ver que através desta atividade prática foi possível aumentar o nosso conhecimento sobre a linguagem de programação C e sobre conceitos importantes para a disciplina de estruturas de dados.