MAPA – Material de Avaliação Prática da Aprendizagem

Acadêmico: José Paulo da Silva Boeira Junior	**R.A. 22290937-5**
Curso: Análise e Desenvolvimento de Sistemas	
Disciplina: Estrutura de Dados I	

Instruções para Realização da Atividade

1. Todos os campos acima deverão ser devidamente preenchidos.
2. É obrigatória a utilização deste formulário para a realização da MAPA.
3. Esta é uma atividade individual. Caso identificado cópia de colegas, o trabalho de ambos sofrerá decréscimo de nota.
4. Utilizando este formulário, realize sua atividade, salve em seu computador, renomeie e envie em forma de anexo. Antes de selecionar a opção de 'Finalizar' a atividade no sistema, verifique o arquivo anexado, pois arquivos em branco ou incorretos **não** poderão ser substituídos após a finalização.
5. Formatação exigida para esta atividade: documento Word, Fonte Arial ou Times New Roman tamanho 12, Espaçamento entre linhas 1,5, texto justificado.
6. Ao utilizar quaisquer materiais de pesquisa referencie conforme as normas da ABNT.
7. Critérios de avaliação: Utilização do template (Formulário Padrão); Atendimento ao Tema; Constituição dos argumentos e organização das Ideias; Correção Gramatical e atendimento às normas ABNT.
8. Procure argumentar de forma clara e objetiva, de acordo com o conteúdo da disciplina.

Em caso de dúvidas, entre em contato com seu Professor Mediador.
Bons estudos!

AGORA É COM VOCÊ!

O estudo das estruturas de dados em Filas e Pilhas é fundamental para entender como organizar e acessar informações de forma eficiente em programação. Uma Fila segue o princípio FIFO (First In, First Out), onde o primeiro elemento inserido é o primeiro a ser removido, ideal para cenários como processamento de tarefas em ordem de chegada. Por outro lado, uma Pilha segue o princípio LIFO (Last In, First Out), em que o último elemento inserido é o primeiro a ser removido, adequada para situações como rastreamento de chamadas de função ou expressões matemáticas. Compreender as operações básicas, como inserção (enqueue/ push), remoção (dequeue/ pop), e visualização (front/ top), permite a implementação eficaz de algoritmos e soluções para uma variedade de problemas computacionais, contribuindo para o desenvolvimento de sistemas robustos e eficientes.

Filas e pilhas são muito importantes em diferentes processos no dia a dia, tanto para processos em computação quanto no “mundo real”. As filas são estruturas sequenciais ordenadas, onde um novo elemento sempre é inserido no final da fila e só pode ser removido o elemento do início da fila. Não se pode “furar a fila”.

Vamos alterar um pouco essa definição de fila para atender a um problema real. Imagine que você entra em uma fila em um banco que é gerenciada através de senhas. Porém, não é uma fila única e simples, mas uma fila que comporta diferentes situações:

• Existem 4 caixas realizando o atendimento.

• As senhas distribuídas possuem uma letra, que indica a categoria, seguida por um número sequencial.

• Considere as seguintes categorias:

- C – Comum
- P – Prioridade (idosos, gestantes, etc.)
- R – Atendimento rápido (operações simples)

• O caixa 1 atende às prioridades. Se não houver prioridades na fila, atendem o próximo da fila, de qualquer categoria.

• Os caixas 2 e 3 realizam atendimentos comuns. Se não houver, atendem o próximo da fila, de qualquer categoria.

• O caixa 4 é especial para atendimentos rápidos. Se não houver, atendem o próximo da fila, de qualquer categoria.

```
1 //Bibliotecas
2 #include <stdio.h>
3 #include <stdlib.h>
4 #include <locale.h>
5
6 //Constantes
7 #define tamanho 20
8
9 //Estrutura da Senha
10 typedef struct tsenha {
11    int numero;
12    char tipo;
13 } tsenha;
14
15 //Estrutura da Fila
16 struct tfilabanco {
17    tsenha dados[tamanho];
18    int ini;
19    int fim;
20 };
21
22 //Variáveis globais
23 struct tfilabanco fila;
24 int op, proximo;
25
26 //Protipação
27 void fila_entrar();
28 void fila_sair();
29 void fila_mostrar();
30 void menu_mostrar();
31
32 //Função principal
33 int main(){
34    setlocale(LC_ALL, "Portuguese");
35    op = 1;
```

```
36    proximo = 1;
37    fila.ini = 0;
38    fila.fim = 0;
39    while (op != 0) {
40        system("cls");
41        fila_mostrar();
42        menu_mostrar();
43        scanf("%d", &op);
44        switch (op) {
45            case 1:
46                fila_entrar();
47                break;
48            case 2:
49                fila_sair();
50                break;
51        }
52    }
53    return 0;
54 }
55
56 //Adicionar um elemento no final da Fila
57 void fila_entrar(){
58    if (fila.fim == tamanho) {
59        printf("\nA fila está cheia, volte outro dia!\n\n");
60        system("pause");
61        return;
62    }
63
64    char tipo;
65    printf("\nEscolha o tipo do atendimento:");
66    printf("\nC - Comum");
67    printf("\nP - Prioridade");
68    printf("\nR - Rápido: ");
69    scanf(" %c", &tipo);
70    if (tipo != 'C' && tipo != 'P' && tipo != 'R') {
71        printf("\nTipo de atendimento inválido!\n\n");
72        return;
73    }
74    fila.dados[fila.fim].tipo = tipo;
75    fila.dados[fila.fim].numero = proximo;
76    proximo++;
77    fila.fim++;
78 }
```

```
79
80 //Retirar o primeiro elemento da Fila de acordo com o caixa
81 void fila_sair() {
82    if (fila.ini == fila.fim) {
83        printf("\nFila vazia, mas logo aparece alguém!\n\n");
84        system("pause");
85    } else {
86        int caixa;
87        char tipoatendimento;
88        printf("\nQual caixa vai atender (1 a 4)?: ");
89        scanf("%d", &caixa);
90        switch (caixa) {
91            case 1:
92                tipoatendimento = 'P';
93                break;
94            case 2:
95            case 3:
96                tipoatendimento = 'C';
97                break;
98            case 4:
99                tipoatendimento = 'R';
100                break;
101            default:
102                printf("\nValor inválido!");
103                return;
104        }
105        // Define a posição do elemento aser removido como 0
106        int posicao = 0;
107
108        // Procura elemento do tipo correspondente
109        for (int i = 0; i < tamanho; i++) {
110            if (fila.dados[i].tipo == tipoatendimento) {
111                posicao = i;
112                break;
113            }
114        }
115
116        printf("\n\n###### ATENDIMENTO #########\n");
117        printf("Senha: %c-%d\n", fila.dados[posicao].tipo, fila.dados[posicao].numero);
118        printf("Caixa: %d\n", caixa);
119        printf("###### ATENDIMENTO #########\n\n");
120
121        // Retira elemento da posição e move os demais
```

```
122         for (int i = posicao; i < tamanho; i++) {
123             fila.dados[i].numero = fila.dados[i+1].numero;
124             fila.dados[i].tipo = fila.dados[i+1].tipo;
125         }
126         fila.dados[fila.fim].numero = 0;
127         fila.dados[fila.fim].tipo = ' ';
128         fila.fim--;
129     }
130 }
131
132 //Mostrar o conteúdo da Fila
133 void fila_mostrar() {
134     int i;
135     printf("[ ");
136     for (i = 0; i < tamanho; i++) {
137         printf("%c-%d ", fila.dados[i].tipo, fila.dados[i].numero);
138     }
139     printf("]\n\n");
140 }
141
142 //Mostrar o menu de opções
143 void menu_mostrar() {
144     printf("\nEscolha uma opção:\n");
145     printf("1 - Nova senha\n");
146     printf("2 - Atender\n");
147     printf("0 - Sair\n\n");
148 }
```

Segue uma implementação em linguagem C seguindo as regras apresentadas.

Fonte: Elaborado pelo professor, 2024.

Baseado nas regras apresentadas e no código fonte apresentado, **RESPONDA** às perguntas a seguir:

1- Explique como é possível armazenar número e letra para cada senha nessa solução.
2- Explique como é que o caixa 4 consegue chamar primeiro quem tem senha de atendimento rápido, detalhando esse trecho do algoritmo, e o que acontece caso não haja nenhuma senha dessa categoria.

3- Suponha que a fila esteja composta por: P-512, R-513, C-514, C-515, P-516. Quando o caixa 3 fizer uma chamada, qual senha será atendida? Como fica a fila após esse atendimento?
4 - Nas linhas 76 e 77 há dois incrementos de variáveis. Por que isso é feito e qual é a diferença entre essas variáveis?

O que devo entregar?

Deverá ser entregar as respostas das 4 perguntas anteriores em um arquivo de WORD ou PDF, conforme formulário padrão constante nos materiais da disciplina.

Orientações:

1. Acesse o link com um vídeo tutorial para ajudá-lo neste processo de criação e desenvolvimento.
2. A entrega deve ser feita exclusivamente por meio do Template de entrega da atividade MAPA disponível no material da disciplina.
3. Antes de enviar sua atividade, certifique-se de que respondeu a todas as perguntas e realize uma cuidadosa correção ortográfica.
4. Após o envio não são permitidas alterações, ou modificações. Logo, você tem apenas uma chance de enviar o arquivo corretamente. Revise bem antes de enviar!
5. Lembre-se que evidências de cópias de materiais, incluindo de outros acadêmicos, sem devidas referências serão inquestionavelmente zeradas. As citações e referências, mesmo que do livro da disciplina, devem ser realizadas conforme normas da Instituição de Ensino.
6. Não são permitidas correções parciais no decorrer do módulo, ou seja, o famoso: “professor, veja se minha atividade está certa?”. Isso invalida seu processo avaliativo. Lembre-se que a interpretação da atividade também faz parte da avaliação.
7. Procure sanar suas dúvidas junto à mediação em tempo hábil sobre o conteúdo exigido na atividade, de modo que consiga realizar sua participação.
8. Atenção ao prazo de entrega, evite envio de atividade em cima do prazo. Você pode ter algum problema com internet, computador, software etc. e os prazos não serão flexibilizados, mesmo em caso de comprovação.

Bons estudos!

Em caso de dúvidas, encaminhar mensagem ao seu Professor(a) Mediador(a).

RESPOSTA:

1- Explique como é possível armazenar número e letra para cada senha nessa solução.

É possível armazenar número e letra para cada senha através da struct tsenha. Nela, foram declaradas duas variáveis, uma do tipo inteiro chamada número, e outra do tipo char chamada tipo. Quando a função fila_entrar é chamada, ela permite que o usuário digite qual é o tipo de prioridade do atendimento, que quando escolhido, é armazenado na variável “tipo” de

tsenha. Já o número é escolhido automaticamente pelo programa, de acordo com a situação da fila, e é armazenado na variável “numero” de tsenha.

2- Explique como é que o caixa 4 consegue chamar primeiro quem tem senha de atendimento rápido, detalhando esse trecho do algoritmo, e o que acontece caso não haja nenhuma senha dessa categoria.

Dentro da função fila_sair, após ser escolhido o caixa 4 como o caixa que irá atender, no trecho de código da linha 108 à linha 114, há um laço de repetição que procura a senha da fila correspondente ao caixa 4, ou seja, a senha que corresponde à letra R. Caso o laço encontre uma senha com a letra R, ele armazena a posição dessa senha na variável “posição”, e caso o laço não encontre uma senha com a letra R, ele armazena o valor de 0 na variável “posição”, ou seja, colocando a posição de atendimento como a primeira senha da fila. Posteriormente, o algoritmo segue e, no trecho de código da linha 122 à linha 128, existe um laço de repetição feito para remover o elemento da posição armazenada em “posição”.

3- Suponha que a fila esteja composta por: P-512, R-513, C-514, C-515, P-516. Quando o caixa 3 fizer uma chamada, qual senha será atendida? Como fica a fila após esse atendimento?

A senha a ser atendida é a C-514. Portanto, após a chamada, a fila fica formada da seguinte maneira: P-512, R-513, C-515, P-516.

4 - Nas linhas 76 e 77 há dois incrementos de variáveis. Por que isso é feito e qual é a diferença entre essas variáveis?

Esses incrementos são feitos para organizar o armazenamento das senhas na fila. Especificamente, o incremento em “proximo” é feito para guardar o número da próxima senha a ser adicionada, e o incremento em “fila.fim” serve para aumentar um elemento no final da fila.