Universidade Federal da Bahia

Componentes:
Andressa Andrade
Guilherme Bernal
Renata Antunes
Rodrigo Fernandes

Projeto - 1

Implementação de um programa para o setor jurídico de uma rede de supermercados para cheques devolvidos

Salvador - 2014

Objetivo

Implementar um programa para um setor jurídico de uma rede de supermercados, visando simplificar a resolução dos processos de cheques devolvidos e tratar suas prioridades.

Introdução

Atráves dos estudos obtidos no decorrer da disciplina Estrutura de Dados e Algoritmos I como por exemplo pilha, fila, deques, entre outros, é possível tratar diversos problemas fazendo uso das estruturas aprendidas.

Para atender o pedido do cliente, neste caso de um setor jurídico de uma rede de supermercados, deve-se implementar um programa que organize, em ordem decrescente de valor, os cheques que foram devolvidos para o supermercado. Para cada cheque devolvido, são necessárias as seguintes informações:

- Cliente:
  - Nome;
  - Identidade;
  - Endereço;
  - Telefone
- Cheque:
  - Valor;
  - Data.
- Supermercado;
  - Nome.

* Para cada cheque devolvido é criado um novo processo que deve ter um ID único.

Para suprir estas condições, fez-se o uso da estrutura de dados de pilha com algumas prioridades, onde, para este problema, o cheque de maior valor (que estará no topo da pilha) será o primeiro à ser retirado para análise e, o cheque de menor valor (que estará na base da pilha) será o último à ser retirado para análise. Este é o funcionamento básico do programa com a prioridade inicial, sem tratar as prioridades adicionais, que dependerão da solicitação do usuário.

Relembrando, a prioridade inicial incluída no programa é:

- Ordenar por valor do cheque em ordem decrescente.
- Operações básicas:
  - Adicionar processo à pilha;
  - Remover processo da pilha;

E as prioridades adicionais incluídas no programa são:

- Separar por supermercado;
- Mudar prioridade de um processo aleatório;
- Operações adicionais:

- Exibir o topo da pilha;
- Exibir a base da pilha;
- Exibir todos os processos da pilha;
- Exibir tamanho da pilha;
- Buscar processo aleatório atráves do ID;
- Remover processo aleatório atráves do ID;
- Mudar prioridade de um processo aleatório.

* A ação de adicionar um processo a pilha já inclui a atribuição de características necessárias descritas anteriormente.
* A ação de exibir topo, base ou todos os processo da pilha, já inclui a exibição das características.

Descrição da Estrutura de Dados utilizada

A estrutura de dados utilizada segue os principios da pilha apresentada em projetos anteriores, que possui uma estrutura duplamente encadeada .

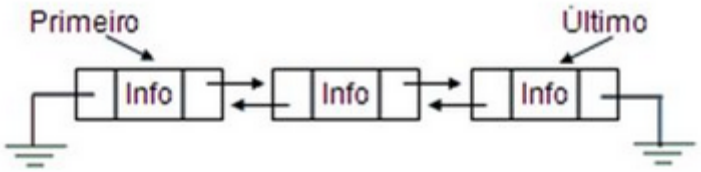


Figura 1 - Exemplo do funcionamento da estrutura duplamente encadeada.

Esse tipo de estrutura ocupa mais espaço que uma lista simplesmente encadeada, porque é necessário adicionar um ponteiro. No entanto, a vantagem de percorrer a lista nos dois sentidos compensa o uso extra da memória utilizada.

Descrição do Algoritmo

O algoritmo foi desenvolvido utilizando de funções criadas para a pilha, exceto para os casos particulares do programa. Por exemplo, foi criada uma estrutura para suprir as necessidades de atributos de cada processo:

```
typedef struct process_element {
   int processId;
   unsigned value; // centavos
   int supermarketId;

   char name[100];
   unsigned age;
   char address[300];
   unsigned long long phone;
   char date[100];
} process;
```

Funções:

- Aloca dinamicamente
  processes processAlloc(void);*
  Utiliza a função de alocação da pilha stack_alloc;

- Insere elemento na pilha de processos
  void processInsert(processes prcs, process el);*
  Utiliza a função de inserção da pilha stack_push;

- Remove elemento da pilha de processos
  void processRemoveByID(processes prcs, int id);*
  Utiliza duas funções da pilha stack_each;
  stack_remove;

- Desempilha uma unidade de processos e retorna ela
  process processPop(processes prcs);*
  Utiliza a função de remover da pilha stack_pop;

- Retorna o processo do topo, porém sem desempilhar
  process processTop(processes prcs);*

Utiliza a função de ler topo da pilha stack_top;

- Retorna o processo da base
  process processBottom(processes prcs);*
  Utiliza a função retornar por posição da pilha stack_nth;

- Retorna processo a partir do ID
  process processGetByID(processes prcs, int id);*
  Utiliza a função da pilha stack_each;

- Quantidade de processos
  unsigned processAmount(processes prcs);*
  Utiliza a função da pilha stack_size;

- Itera sobre os processos
  void processEach(processes prcs, void(*callback)(process el));*
  Utiliza a função da pilha stack_each;

- Prioriza um supermercado através do ID
  void processPrioritizeSupermarket(processes prcs, int supermarketId);*
  Utiliza a função da pilha stack_sort;

- Prioriza por valor
  void processPrioritizeValue(processes prcs);*
  Utiliza a função da pilha stack_sort;

Análise Experimental Simplificada

Complexidades:

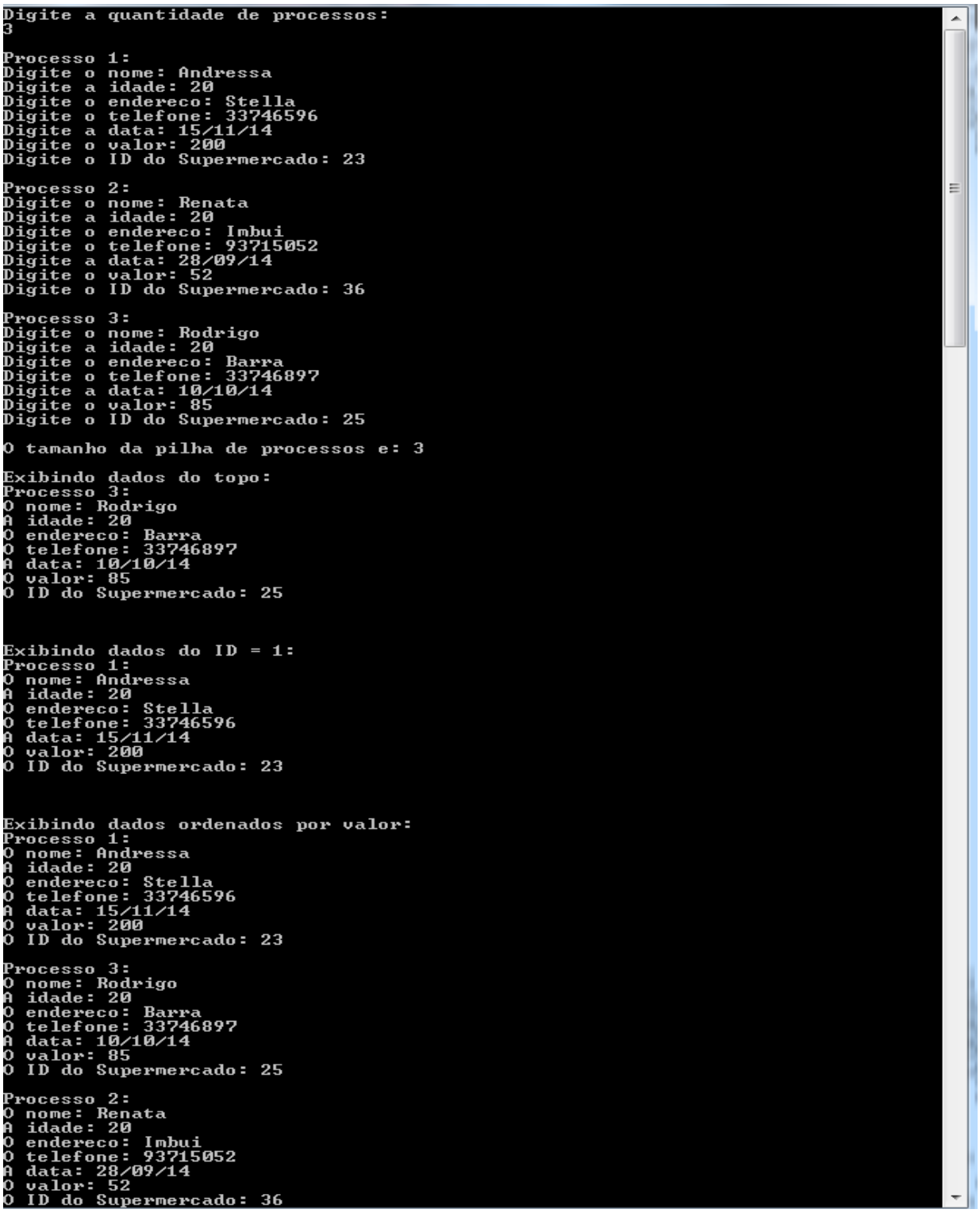
- O(1)
    - processes* processAlloc(void);
    - void processInsert(processes* prcs, process el);
    - process processPop(processes* prcs);
    - process processTop(processes* prcs);
    - process processBottom(processes* prcs);
    - unsigned processAmount(processes* prcs);
- O(n)
    - void processRemoveByID(processes* prcs, int id);
    - process processGetByID(processes* prcs, int id);
    - void processEach(processes* prcs, void(*callback)(process el));
- O(n logn)
    - void processPrioritizeSupermarket(processes* prcs, int supermarketId);
    - void processPrioritizeValue(processes* prcs);

Teste:

Conclusão

Anexos

Imagens:



LEIA-ME

Para realizar um teste básico:

Como utilizar as funções:

Primeiramente é necessario alocar a estrutura de processo

Para fazer isso:

process *prcs = processAlloc();

*prcs poderia ser qualquer nome

Antes de utilizar funções como processInsert

criar uma estrutura como process no seu projeto. Ex:

process p;

*onde novamente p, poderia assumir outros nomes.

Cada um dos atributos é posteriormente adicionado. Ex:

printf(“Digite a idade: ”);
scanf(“%i”, &p.age);

Após todos os atributos forem colocados, basta inserir usando processInsert, utilizando os seguintes parametros:

processInsert(prcs,p);

Para verificar tamanho, utiliza-se processAmount, como essa função retorna um inteiro com o tamanho da pilha de processos é necessario criar um incognita com este tipo de dado, para imprimir o valor correspondente

int x = processAmount(prcs);
printf(“%i”, x);

Para selecionar processo por ID, é necessario saber o Id que se deseja obter processo e utiliza-lo na funcao processGetByID(prcs, 1), no caso, um exemplo para obter-se o processo com ID = 1;
Para printar essa função, é necessario criar um processo mx;

process mx = processGetByID(prcs,1);

e depois printar cada um dos atributos do processo

```
printf("Processo %d:\n", mx.processId);
printf("O nome: %s\n",mx.name);
printf("A idade: %i\n",mx.age);
printf("O endereco: %s\n",mx.address);
printf("O telefone: %ld\n",mx.phone);
printf("A data: %s\n",mx.date);
printf("O valor: %i\n",mx.value);
printf("O ID do Supermercado: %i\n\n",mx.supermarketId);
```

Para exibir o topo da pilha de processo, é semelhante

```
process mp = processTop(prcs);
printf("Processo %d:\n", mp.processId);
printf("O nome: %s\n",mp.name);
printf("A idade: %i\n",mp.age);
printf("O endereco: %s\n",mp.address);
printf("O telefone: %ld\n",mp.phone);
printf("A data: %s\n",mp.date);
printf("O valor: %i\n",mp.value);
printf("O ID do Supermercado: %i\n\n",mp.supermarketId);
```

Para exibir a lista toda e ordenada, o processo seria, chamar a função que ordena por valor(exemplo).

processPrioritizeValue(prcs);

depois chamar a função

processEach(prcs, print_processo)

sendo a função print_processo, definida por:

```
int print_processo(process p) {
        printf("Processo %d:\n", p.processId);
        printf("O nome: %s\n",p.name);
        printf("A idade: %i\n",p.age);
```

```
        printf("O endereco: %s\n",p.address);
        printf("O telefone: %ld\n",p.phone);
        printf("A data: %s\n",p.date);
        printf("O valor: %i\n",p.value);
        printf("O ID do Supermercado: %i\n\n",p.supermarketId);

        return 1;
}
```