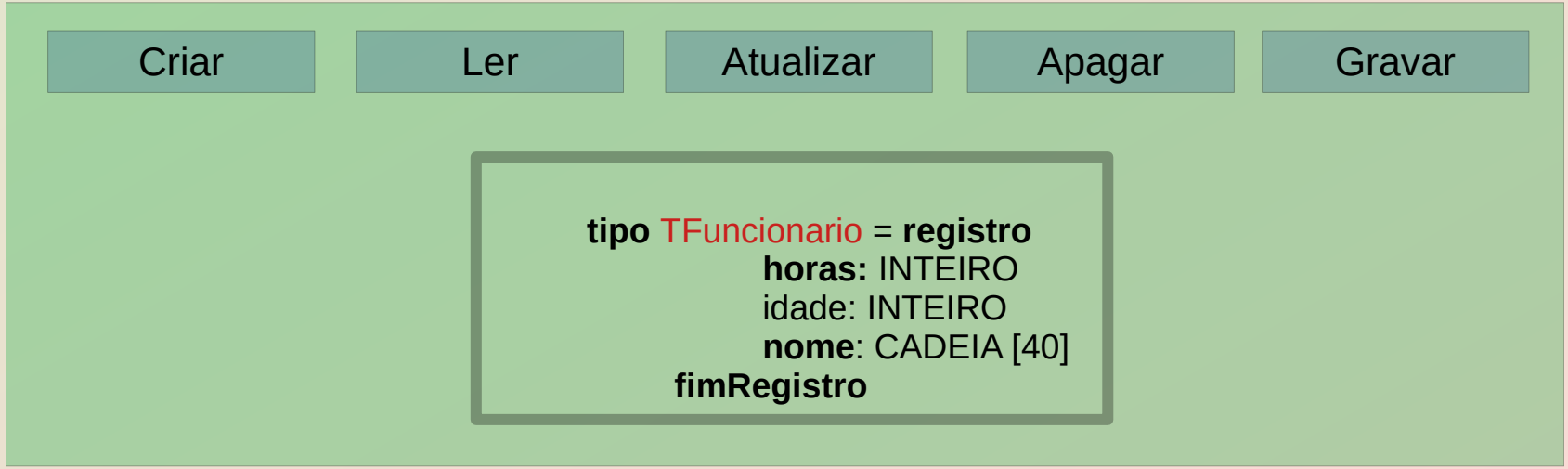
ESTUDO DE CASO

Uma empresa possui 21 funcionários e mantém as seguintes informações por funcionário: número de horas trabalhadas no mês, nome.

Elabore um programa que:

- gere uma lista contendo os nomes dos funcionários com as respectivas horas trabalhadas.
- gere uma lista contendo os nomes dos funcionários com as respectivas horas trabalhadas. A lista deve estar ordenada pelo número de horas trabalhadas.

- **ENCAPSULAÇÃO:** o acesso ao dado ocorre por meio de um procedimento.
- Serão implementados tantos procedimentos quantos forem necessários.



OPERAÇÕES COM LISTAS - VETORES

Criar	Ler	Atualizar	Apagar	Ordenar	Gravar

```
tipo TEmpresa = registro
                    lista: CONJUNTO [21] DE TFuncionario
                fimRegistro
```

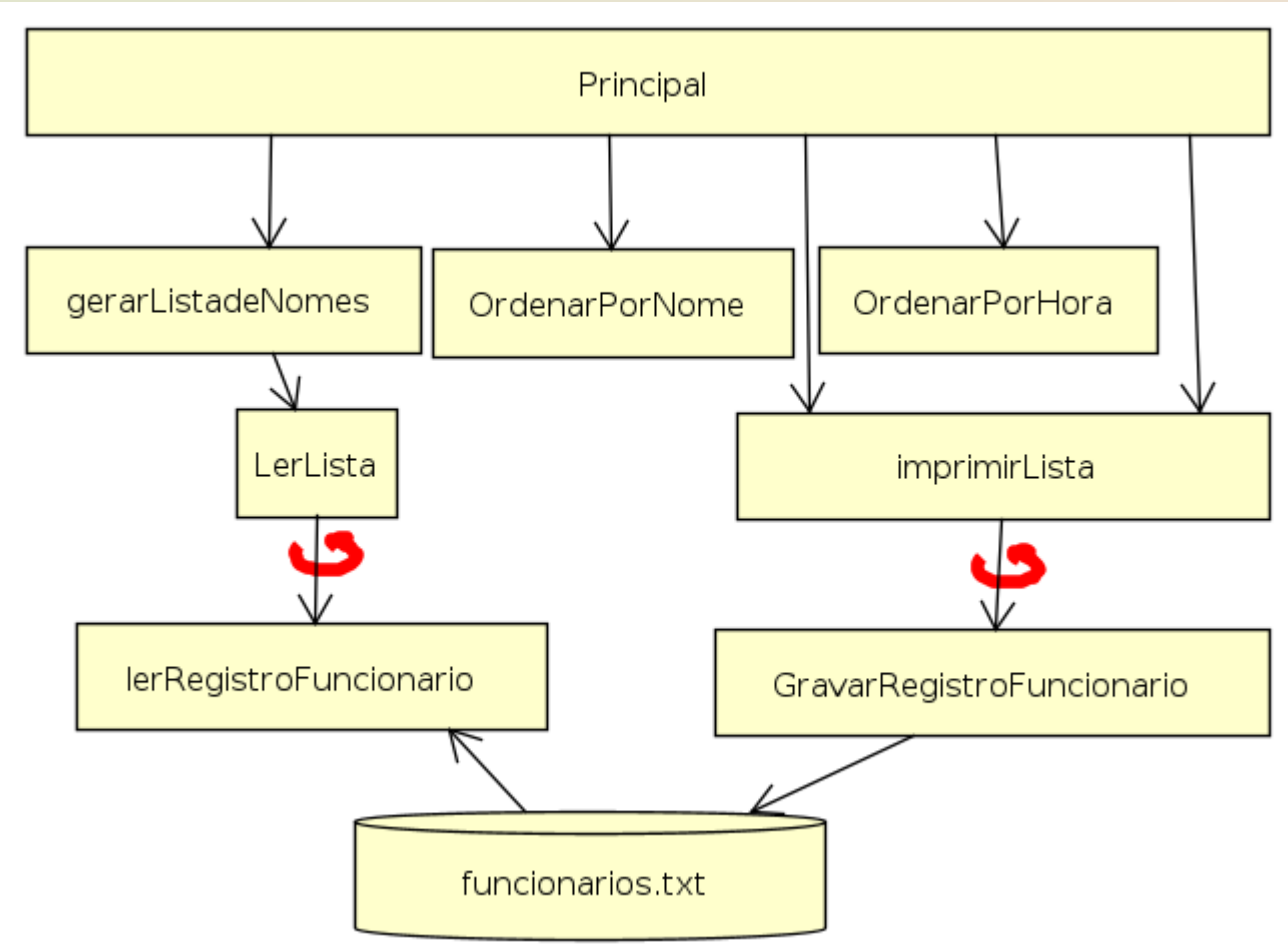
ALGORITMO Empresa

Variáveis globais

Procedimento gerarListadeNomes

Procedimento gerarListaOrdenada

Programa Principal


Principal
gerarListadeNomes
OrdenarPorNome
OrdenarPorHora
LerLista
imprimirLista
lerRegistroFuncionario
GravarRegistroFuncionario
funcionarios.txt

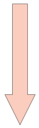
```
ALGORITMO Empresa
VAR
        tipo TFuncionario = registro
                        horas: INTEIRO
                        nome: CADEIA
                fimRegistro
        tipo TEmpresa = registro
                        lista: CONJUNTO [21] DE TFuncionario
                fimRegistro
        empresa: TEmpresa

    # DEFINIR OS PROCEDIMENTOS: esqueleto da solução
    PROCEDIMENTO gerarListaDeNomes (var TEmpresa:emp, arquivo: Cadeia[40])
    INÍCIO
    FIM

    PROCEDIMENTO ordenarPorNome (var TEmpresa: listadeNomes)
    INÍCIO
    FIM

    PROCEDIMENTO ordenarPorHora (var TEmpresa: listadeNomes)
    INÍCIO
    FIM

    PROCEDIMENTO lerLIsta (var TEmpresa: listadeNomes, var FILE: arquivo)
    INÍCIO
    FIM
```

```
    PROCEDIMENTO lerLIsta (var TEmpresa: listadeNomes)
    INÍCIO
    FIM

    PROCEDIMENTO imprimirLista (var TEmpresa: listadeNomes)
    INÍCIO
    FIM

    PROCEDIMENTO lerRegistroFuncionario (var TFuncionario: funcionario, var FILE: arquivo)
    INÍCIO
    FIM

    PROCEDIMENTO gravarRegistroFuncionario (TFuncionario: funcionario, var FILE: arquivo)
    INÍCIO
    FIM

INÍCIO
    gerarListaDeNomes(empresa, "funcionarios.txt");
    ordenarPorNome(empresa);
    imprimirLista("ordenadopornome.txt");
    ordenarPorHora(empresa);
    imprimirLista("ordenadoporhora.txt");
FIM
```

EXEMPLO DE IMPLEMENTAÇÃO REGISTRO FUNCIONÁRIO

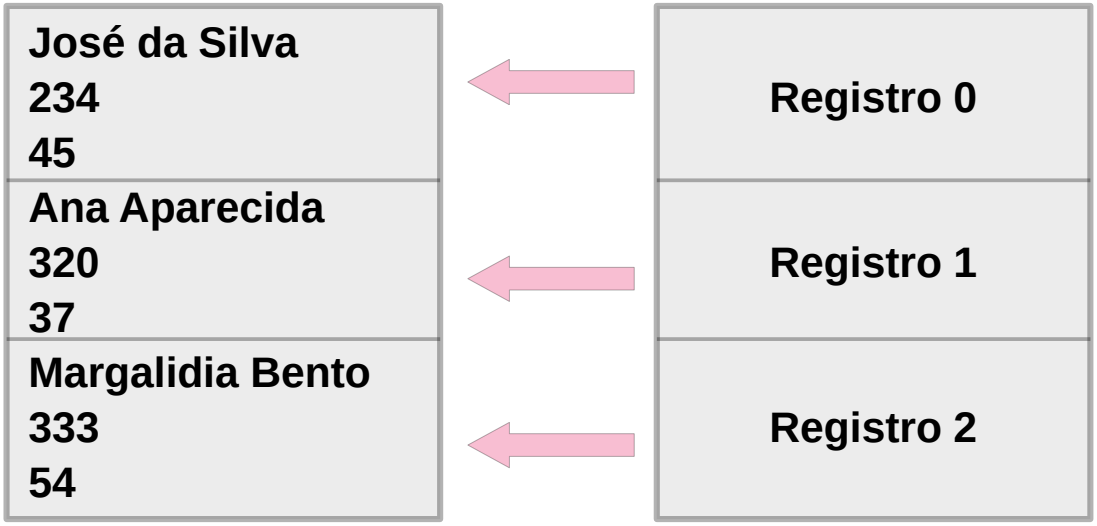
```
tipo TFuncionario = registro
        horas: INTEIRO
        nome: CADEIA
fimRegistro
```

```
struct Funcionario {
    int horas;
    char* nome;
};
```



```
typedef struct Funcionario TFuncionario;
typedef struct Funcionario* PTFuncionario;
```

FORMATO DO ARQUIVO: TEXTO



```
/* ==============================
Aloca o espaço de memória correspondente ao tamanho do registro Funcionario.
Entrada: Ponteiro para o registro.
Retorno:
- endereço da memória alocada,
caso a operação seja bem sucedida.
- NULL (0) caso ocorra erro.
*/
PTFuncionario criarFuncionario();

/* ==============================
Libera o espaço de memória ocupado pelo registro
Entrada: Ponteiro para o registro.
*/
void deleteFuncionario(PTFuncionario);

/* ==============================
Lê os dados de um funcionário, a partir do "stream input" padrão (teclado), e os armazena no registro passado como
argumento.
Entrada: Ponteiro para o registro.
*/
int lerRegFuncionario(PTFuncionario);
```

```
/* ==============================
Imprime os dados do funcionário no dispositivo padrão de saída
Entrada: Ponteiro para o registro.
*/
void printRegFuncionario(PTFuncionario);

/* ==============================
Lê um registro do arquivo.
Entradas: Ponteiro para o arquivo; Ponteiro para o registro.
*/
int lerRegistroFuncionario(PTFuncionario, FILE*);

/* ==============================
Escreve um registro do arquivo.
Entradas: Ponteiro para o arquivo; Ponteiro para o registro.
*/
Int gravarRegistroFuncionario(PTFuncionario, FILE*);
```

CASOS DE TESTE

Ler do teclado e imprimir na tela

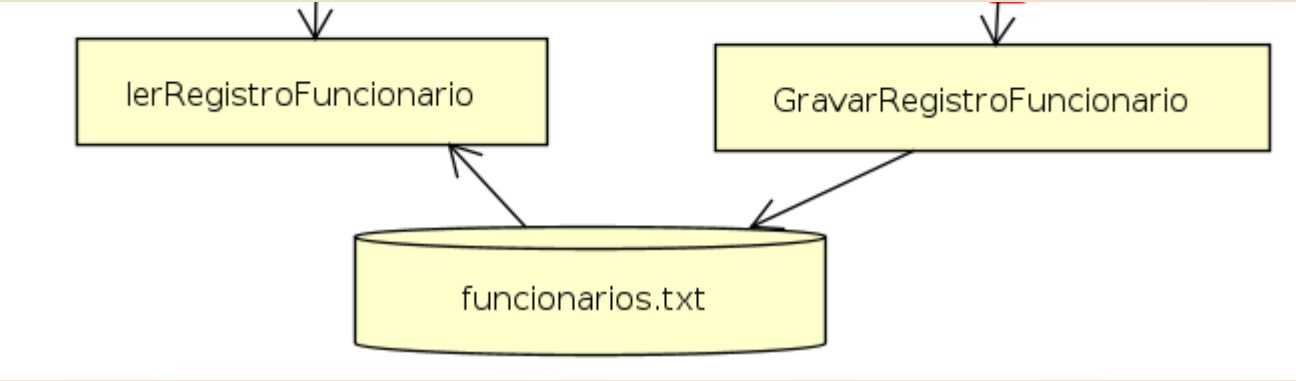
```
void teste001(){
    PTFuncionario p = criarFuncionario();
    lerRegFuncionario(p);
    printRegFuncionario(p);
    deleteFuncionario(p);
}
```

Ler do arquivo e imprimir na tela

```
void teste002(){
    PTFuncionario p = criarFuncionario();
    FILE *ifp; // ponteiro para o arquivo de entrada
    ifp = fopen("Funcionarios.txt", "r");
    while (!feof(ifp)){
        readFuncionario(p, ifp);
        printRegFuncionario(p);
    }
    fclose(ifp);
    deleteFuncionario(p);
}
```

Enquanto não for fim de arquivo

RESOLVIDOS


lerRegistroFuncionario
GravarRegistroFuncionario
funcionarios.txt

PRÓXIMO REFINAMENTO: tratamento da lista de registros

1. Criar a lista de registros (vetor empresa)
2. Fazer a leitura do registro em disco e inserção na lista
3. Ordenar os registros da lista por nome
4. Imprimir a lista na tela
5. Imprimir a lista em arquivo
6. Criar uma base de dados de teste
7. Ordenar os registros por horas trabalhadas
8. Criar o programa principal

Copiar o arquivo de registros para testar a leitura e escrita de arquivo

```
void teste003(){
    PTFuncionario p = criarFuncionario();
    FILE *ifp; // Arquivo de entrada
    FILE *ofp; // Arquivo de saida
    ifp = fopen("saida.txt", "r");
    ofp = fopen("saida1.txt", "w");
    while (!feof(ifp)){
        readFuncionario(p, ifp);
        printRegFuncionario(p);
        writeFuncionario(p, ofp);
    }
    fclose(ifp);
    fflush(ofp);
    fclose(ofp);
    deleteFuncionario(p);
}
```

Arquivo de leitura “r”: read

Arquivo de escrita “w”: write