Marcelo Antelino Neto Peralta Alencar Dores

RA: 18108704

Rafael Alves de Oliveira Perroni

RA:18009340

Victor Luiz Fraga Soldera

RA: 18045674

**Arquitetura de Computadores**

Controle de Gastos

**Introdução**

O projeto proposto no Laboratório de Arquitetura de Computadores, foi a elaboração de um programa para o gerenciamento de gastos. Para isso utilizamos o assembly MIPS(uma arquitetura do tipo RISC), de modo a usar funções para cadastrar despesas, excluir as mesmas, e diferentes maneiras de ordena-las por categoria e mês de forma ordenada para facilitar o entendimento do usuário.

**Detalhes do Projeto**

O nosso projeto foi organizado em uma forma de modulação chamada de MVC(Model,View e Controller), de modo que se torna-se um código mais organizado dividido em três módulos que interagem entre si.

**Model**

É a database do programa, ou seja, é responsável pelo tratamento de parte dos dados recebidos no programa, porem o Model não tem interação com a View do programa, mas sim com o Controller, onde ele passa essa informação tratada para ser utilizada.

**View**

A View é basicamente o modulo do programa que recebe informações do Controller é as torna visíveis para o usuário tendo apenas comunicação de saída para o usuário e de entrada na View pelo Controller, sendo a camada mais superior do código em geral.

**Controller**

O Controller basicamente gera uma interação geral do código conectando ao Model com a View, e recebendo os dados do usuário para trata-los no Model e após isso os mostrar para o usuário pela View, mas o Controller é composto por regras de negócio, regras da aplicação e código lógico, para o controle do programa.

**Detalhes de Implementação**

No projeto utilizamos a seguinte estrutura de dados:

|  |
| --- |
| -> ID => 4 Bytes |
| -> Dia => 4 Bytes |
| -> Mês => 4 Bytes |
| -> Ano => 4 Bytes |
| -> Endereço de Categoria (String) => 4 Bytes |
| -> Valor => 4 Bytes |
| -> Pointer to Next => 4 Bytes |

|  |
| --- |
| PADRONIZAÇÃO DE CÓDIGO |
| -> $s0 => Guarda início da lista; |
| -> $s1 => Gerador de ID, guarda o próximo ID a ser salvo na próxima estrutura a ser instanciada; |
| -> $t0 => Usado para iteração (percorrer) da lista sempre que necessário; |
| -> $a0-$a3 => Passar parâmetros pra funções, ignorar em Macros; |
| -> Registradores temporários altos como $t7 e subsequentes de cima pra baixo (7, 6 e 5) são utilizados em funções menores. |

**Cadastro**

No cadastro utilizamos uma lista ligada de forma a usar um malloc para o armazenamento das informações na estrutura de dados usada, para a entrada de dados no programa modularizamos o código de forma a facilitar o entendimento, criando uma função para leitura de cada informação.

Primeiramente a “node” é gerada a partir da função “create node", nessa função ele aloca memória e salva em $v0, alocando 28 bytes por “node", com isso o id é gerado automaticamente por um gerador de id que é incrementado de 1 em 1.

Logo a categoria e lida a partir da função especifica dela(read) e armazenada na “node”, em seguida a data e recebida de forma particionada, ou seja, dia, mês e ano são lidos e guardados separadamente para facilitar o uso no decorrer do programa, e o gasto é guardado em forma de float na “node”. Em geral no programa todas informações lidas não são armazenadas diretamente na lista ligada, elas são armazenadas apenas quando as funções “set” são chamadas em seguida da leitura.

**Excluir**

Nossa função para realizar exclusão de despesas consiste em procurar o ID da despesa desejada, percorrendo a lista ligada, e quando a despesa é encontrada usamos a função ”remove\_next\_node” para apagar o endereço que aponta para a “node” desejada, portanto aquela “node” será retirada da lista ligada, fazendo com que a ela aponte para o endereço da “node” seguinte.

**Exibição**

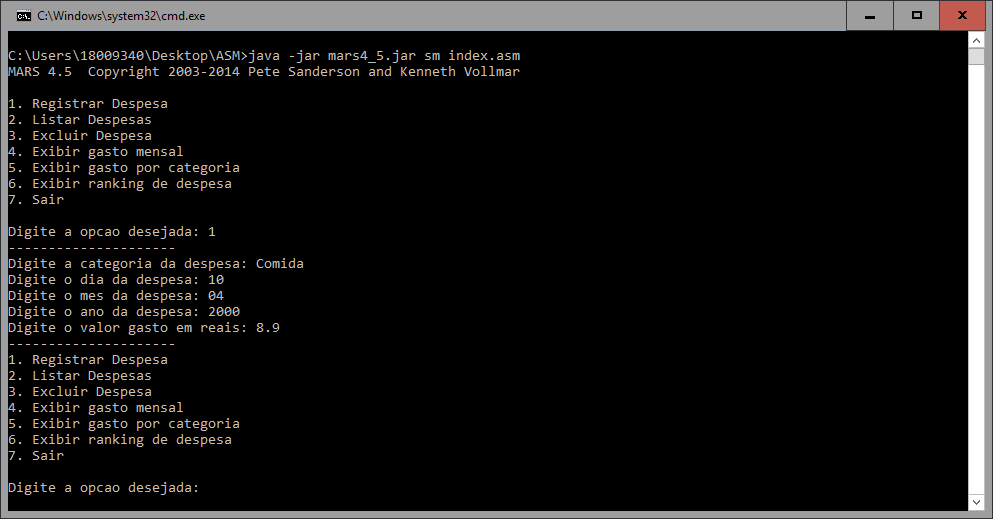
Na exibição dos projetos temos algumas opções de como imprimir os gastos listados, de forma a ser possível imprimi-los de modo geral sem nenhuma restrição ou adotando alguns parâmetros para a seleção de quais gastos imprimir.

Com o uso de funções modulares (print) torna-se mais fácil de imprimir os valores desejados, na impressão de gastos por ranking utilizamos para facilitar o desenvolvimento um método de ordenação “BubbleSort” ordenando os valores dos gastos (em float) de forma decrescente.

O uso de uma seleção de impressão de gastos por mês também foi implementado junto a uma seleção por categorias de despesa gerando várias opções de impressão variáveis.

**Testes Realizados**

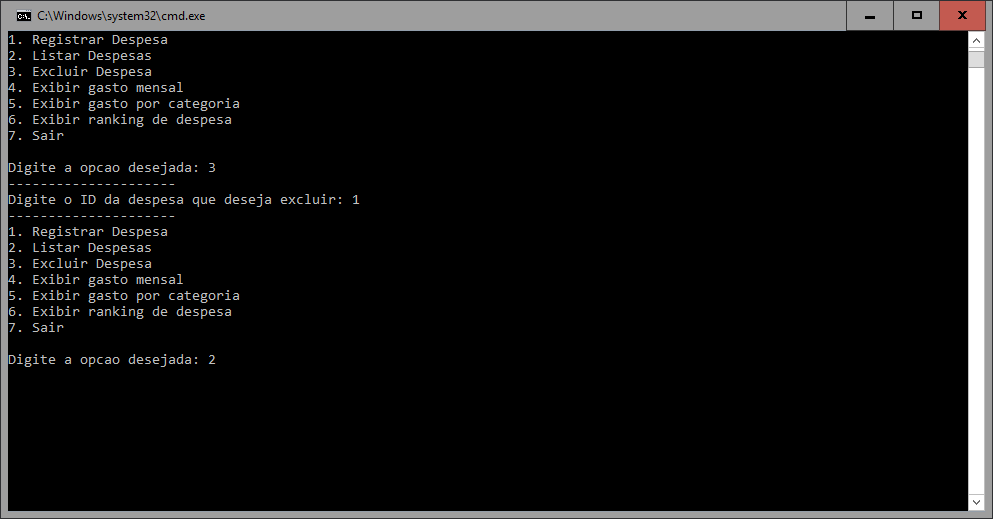
Nesse teste realizamos o teste do cadastrar despesa com apenas um gasto primeiro.



Após o primeiro cadastro realizamos outro cadastro para o teste seguinte das impressões e exclusão. Na imagem abaixo vemos que os testes foram um sucesso.



As imagens seguintes mostram o teste da exclusão de um gasto, no caso haviam duas despesas e excluímos a de ID = 1, sendo possível ver na segunda imagem a impressão dos gastos gerias a presença de apenas um gasto após a exclusão.





Em geral os testes das funções tiveram êxito (Cadastrar, Excluir e Impressão). Sendo testes satisfatórios para a formulação do programa.

**Bibliografia**

<https://courses.missouristate.edu/KenVollmar/MARS/Help/SyscallHelp.html>

<https://www.doc.ic.ac.uk/lab/secondyear/spim/node20.html>