Pontifícia Universidade Católica de Campinas

Raissa Furlan Davinha RA:15032006
Rafael Fioramonte RA:16032708
Bruno Vicente Donaio Kitaka RA:16156341

CONTROLE DE GASTOS PESSOAIS

Projeto Arquitetura de Computadores

Campinas
2017

INTRODUÇÃO

O projeto consiste na resolução de um problema apresentado em sala de aula, onde é solicitado projetar um programa em linguagem MIPS de controle de gastos pessoais.

Em Arquitetura de Computadores, MIPS é uma arquitetura de microprocessadores RISC desenvolvida pela MIPS Computer Systems. MIPS é uma arquitetura baseada em registrador, ou seja, a CPU usa apenas registradores para realizar as suas operações aritméticas e lógicas. Existem outros tipos de processadores, tais como processadores baseados em pilha e processadores baseados em acumuladores. Processadores baseados no conjunto de instruções do MIPS estão em produção desde 1988.

ESPECIFICAÇÃO

O projeto exige a implementação de um programa de gastos pessoais q possua as seguintes opções:

Registrar despesa, onde será recebido do usuário data, valor e categoria da despesa;

Excluir despesa, onde o usuário poderá escolher uma despesa através do id, para exclusão;

Exibir despesas ordenadas pela data;

Exibir despesas total em cada mês;

Exibir total de despesas por categoria;

Exibir ranking de gastos por categoria.

Inicialmente foi definido quais valores seriam guardados na estrutura Despesa, e a quantidade de bytes para cada variável, ficando id: 4 bytes, data: 6 numeros, 4 bytes, categoria: 16 bytes, valor: tipo float, 4 bytes, totalizando 28 bytes por estrutura.

Foram criados dois espaços de 5600 bytes, *inicioArray* e *dynamicArray*, para armazenar 200 estruturas de 28 bytes cada. Um para ser o vetor principal, e o segundo como assistente para organizar as estruturas para as funções de exibição.Também foi criado uma variável de 32bits chamada *ArrayPointer*, para salvar o enderço do último conteúdo salvo no vetor. Este foi usado para condições de loop e para controle do vetor, comparando o endereço do ponteiro que está passando pelo vetor, com o valor guardado na variável. Para facilitar a programação, foram estabelecidos registradores para guardar o endereço do *ArrayPointer* e o conteúdo de *ArrayPointer*.

O menu consiste em sete opções, contendo as opções já comentadas acima, e a opção de sair do programa. Foi implementada uma solução para caso o usuário digite valores inválidos.

A função Registrar já insere o novo cadastro ordenando por data. A data é salva em 4 bytes, sendo os dois mais significativos o ano, um byte para o mês e um byte para o dia, formando uma única word que é comparada como um todo, facilitando as aplicações de ordenação. Como a inserção do *id* é automática, o primeiro solicitado ao usuário é a data. Inserida a data, o programa já faz a comparação com todo o vetor *inicioArray*, para manter a ordenação, e só depois solicita os dados categoria e valor da despesa.

A função Excluir pede o ID da despesa que se deseja excluir e, em seguida procura-o no vetor inicioArray. Caso o ID inserido não exista na array, o usuário recebe uma mensagem de que não foi encontrada. Caso exista, substituímos a despesa pela próxima, e assim por diante. Deste modo, a array não contém mais a despesa excluída e, tenha uma despesa a menos.

Para listar as despesas ordenadas por data, foi implementado apenas um *loop*, com condição de parada o valor armazenado na variável *ArrayPointer*, visto que o vetor já estava ordenado.

Para a opção de exibir gastos, foram necessários dois ponteiros, um percorrendo o vetor *inicioArray* e outro o *dynamicArray*. Primeiro é inserido no *dynamicArray* o mês do primeiro dado salvo no vetor principal e seu respectivo valor. Foi considerado para o vetor assistente, uma estrutura de 4 bytes para o mês, e 4 bytes para o valor em ponto flutuante. Um contador mantém a quantidade de itens colocados no vetor assistente. Em seguida começam dois *loops*, um contido dentro de outro. Um ponteiro percorre *dynamicArray* comparando o mês. Caso igual, o valor da despesa é somado e guardado no lugar do valor antigo, o ponteiro para o *inicioArray* é atualizado, e o loop recomeça. Caso não seja achado nenhum mês igual, a função adiciona aquele mês e seu gasto em um novo espaço do vetor assistente, atualiza o contador de itens, atualiza a posição do ponteiro do vetor principal e recomeça o loop. O processo continua até o ponteiro do *inicioArray* chegar ao endereço final, contido na variável *ArrayPointer*. Após esse longo processo, a função segue para a segunda parte, onde um novo loop é começado, este apenas no vetor assitente, para retornar a lista de meses e os respectivos gastos, sendo usado para controle o contador manipulado anteriormente.

As funções 5, exibir gastos por categoria e 6, exibir *ranking* de despesas, foram implementadas de forma semelhante ao item 4, sendo necessários dois ponteiros, para percorrer os dois vetores, comparar o dado contido na categoria, somar valor do gasto caso iguais, adicionar nova estrutura ao vetor assistente caso não ter categoria igual, e novo loop para retornar os valores descobertos. Para os dois casos foi planejado uma estrutura com 16 bytes para a categoria, e 4 bytes para o valor de ponto flutuante.

RESULTADOS

```
1-Registrar despesa
2-Excluir despesa
3-Listar despesas
4-Exibir gasto mensal
5-Exibir gastos por categoria
6-Exibir ranking de despesas
7-sair
1

Digite a dia da despesa: 30

Digite o mes da despesa: 09

Digite o ano da despeza: 17

Defina a categoria da despesa: testel

Digite o valor da despesa: 238.99

Despesa registrada com sucesso!
Aperte qualquer tecla para voltar ao menu.
```

Saída da função 1, cadastrar despesa.

```
1-Registrar despesa
2-Excluir despesa
3-Listar despesas
4-Exibir gasto mensal
5-Exibir gastos por categoria
6-Exibir ranking de despesas
7-sair
2

Digite o ID da despesa: 1

Despesa excluida com sucesso!
Aperte qualquer tecla para voltar ao menu.
```

Saída da função 2, excluir despesa por id.

```
Despezas:
Id:5 | Data:15/11/16 | Valor: 99.99 | Categoria: teste3

Id:4 | Data:28/12/16 | Valor: 90.0 | Categoria: teste3

Id:2 | Data:29/6/17 | Valor: 200.0 | Categoria: testel

Id:7 | Data:18/8/17 | Valor: 98.99 | Categoria: teste2

Id:3 | Data:8/9/17 | Valor: 75.99 | Categoria: teste2

Id:6 | Data:14/9/17 | Valor: 209.0 | Categoria: teste4

Id:1 | Data:30/9/17 | Valor: 238.99 | Categoria: testel

Aperte qualquer tecla para voltar ao menu.
```

Saída da função 3, despesas ordenadas por data.

```
Gasto mensal:
mes   |  gasto
11/16  |  99.99
12/16  |  90.0
6/17  |  200.0
8/17  |  98.99
9/17  |  523.98

Aperte qualquer tecla para voltar ao menu.
```

Saída da função 4, gasto mensal.

```
Despezas por categoria:
 | Valor: 189.98999 | Categoria: teste3
 | Valor: 438.99 | Categoria: testel
 | Valor: 174.98 | Categoria: teste2
 | Valor: 209.0 | Categoria: teste4
```

Saída da função 5, exibir gasto por categoria.

```
Ranking:
 | Valor: 438.99 | Categoria: testel
 | Valor: 209.0 | Categoria: teste4
 | Valor: 189.98999 | Categoria: teste3
 | Valor: 174.98 | Categoria: teste2
```

Saída da função 6, ranking de gastos por categoria.

REFERÊNCIAS

Wikipedia. Arquitetura MIPS. Disponível em: https://pt.wikipedia.org/wiki/Arquitetura_MIPS. Acesso em: 12/08/2017.

Unicamp. Conjunto de instruções Mips. Disponível em: http://www.ic.unicamp.br/~pannain/mc542/aulas/ch3_arq.pdf. Acesso em: 12/08/2017.