Memória

Revisão

Estudamos, na Introdução, que os programas de computador implementam algoritmos, os quais utilizam um conjunto de dados para produzir um resultado. O algoritmo é um conjunto de instruções que são executados seqüencialmente. Em cada um delas, ele consulta dados armazenados, realiza algum tipo de processamento sobre estes e, por fim, armazena novamente o resultado.
Este funcionamento dos programas motiva o estudo da memória do computador, na qual é realizado o armazenamento dos dados manipulados pelo algoritmo.

Conceitos

A memória pode ser entendida como uma seqüência de células nas quais se pode armazenar um valor correspondente a um dado. Para fins didáticos, consideramos a memória como um número ilimitado de células. Na prática, os computadores atuais costumam apresentar algumas centenas de milhões de células.
Como as células estão ordenadas, atribui-se a elas um número correspondente a sua **posição n**a seqüência, contando a partir do zero. Este número denomina-se **endereço**. A manipulação dos endereços de memória será estudada mais para o final do curso.

Célula de Memória

Desta forma, cada **célula de memória** é caracterizada pelo seu **endereço** e seu **conteúdo**.
Uma célula de memória armazena um único valor (conteúdo), que pode ser um número ou um caractere (letra). Eventualmente, quando for necessário armazenar um número relativamente grande, com muitos dígitos, poderá surgir a necessidade de se utilizar mais de uma célula consecutiva da memória.
O armazenamento do conteúdo das células é volátil (não permanente). Se ocorrer alguma falha no computador ou cessar a alimentação elétrica, o seu conteúdo é perdido.

Operações sobre a memória

A memória permite dois tipos de operações: **leitura** e **escrita**. Sempre precisamos indicar o endereço da célula sobre qual desejamos realizar a operação.
A primeira delas, a **leitura**, localiza a célula correspondente ao endereço desejado e consulta o valor armazenado nesta célula. Após a leitura, a célula continua com o valor original armazenado antes de executar esta operação.
A **escrita** trabalha com um endereço e um valor. De forma semelhante à leitura, a escrita localiza a célula do endereço desejado, mas em seguida substitui o conteúdo da mesma pelo novo valor. Durante este processo, o conteúdo original da célula é perdido de forma irreversível.

Processamento de um programa

Essencialmente, para executar um algoritmo, o computador repete ciclos, nos quais:

1) Obtém um ou mais valores da memória, utilizando operações de leitura.
2) Realiza algum tipo de processamento sobre os dados consultados na memória. Como conseqüência, é gerado um novo valor.
3) Efetua uma escrita para armazenar o resultado em uma célula da memória.

Toda computação está baseada no seguinte conceito simples: *um programa manipula apenas valores que estão armazenados nas células da memória*.

Exemplo

Vamos relembrar o algoritmo de Euclides que calcula o MDC (máximo divisor comum), apresentado na Introdução. No início, temos dois números, que são a entrada do algoritmo. Portanto, estes dois números devem ser armazenados cada um em uma célula de memória. Assumiremos que a primeiro número será o conteúdo da célula de endereço 1 e o segundo número será o conteúdo da célula de endereço 2. A Figura 1 mostra como os valores são armazenados nas células de memória.

Figura 1 - Conteúdo

1) **Leia** um número e escreva na célula 1
2) **Leia** um número e escreva na célula 2
3) **Divida** o valor da célula 1 pelo valor da célula 2. Guarde o quociente na célula 3 e o resto na célula 4.
4) **Se** o valor da célula 4 for 0 (zero), então **mostre** o valor da célula 2 e **PARE**.
5) **Escreva** na célula 1 o valor da célula 2.
6) **Escreva** na célula 2 o valor da célula 4.
7) **Retorne** ao passo 3.

Algoritmo 1 - MDC usando endereços de células de memória

No terceiro passo, calculamos o quociente dos dois números. Como não desejamos perder o valor destes dois números, precisamos escrever o quociente e o resto em novas células de memória. Neste exemplo, decidimos guardar o quociente e o resto respectivamente nas células de endereço 3 e 4.

Finalmente, antes de o algoritmo continuar a execução, note que a divisão espera que os números estejam nas células de endereço 1 e 2. Por este motivo, precisamos mover, respectivamente, o quociente (que está na célula de endereço 3) para a célula de endereço 1 e o resto (que está na célula de endereço 4) para a célula de endereço 2.

Desafio: É possível reescrever este algoritmo de forma a utilizar menos células de memória durante o processamento? Quais seriam as vantagens e desvantagens de reduzir o número de células utilizadas por um algoritmo?

Discussão sobre células de memória[DFF8]

Em princípio, qualquer algoritmo poderia ser descrito utilizando-se apenas operações sobre células da memória. Para isso, basta associar cada um dos valores manipulados pelo algoritmo a uma célula de memória. Cada instrução do algoritmo é acompanhada pelos endereços das células onde estão localizados os dados, conforme foi exemplificado no Algoritmo 1.

Esta forma complicada introduz um nível desnecessário de complexidade referente à tecnologia de funcionamento do computador, mas não enriquece a descrição do algoritmo. Lembre-se que a motivação de utilizar linguagens de programação foi encontrar uma *forma simples* de representar algoritmos. O uso de endereços e memória torna-se impraticável para algoritmos mais longos, pois será muito difícil para o programador associar manualmente cada dado a uma célula de memória. Muito menos ainda para uma equipe de programadores. Erros comuns ocorrem quando um programador acidentalmente confunde o significado de duas células de memória. Além disso, introduzir modificações em um algoritmo ou realizar alguma manutenção no código torna-se uma tarefa muito penosa.

Nem todos os dados cabem em uma única célula de memória. Por exemplo, números muito grandes, dependendo de seu tamanho, podem exigir duas ou mais células. Neste caso, o programador precisa lidar manualmente com estas situações excepcionais.

Durante o desenvolvimento de um programa de computador, é praticamente impossível prever quais posições de memória estarão disponíveis antes do momento de execução do programa. Além disso, existem outros programas em execução simultânea, todos eles poderiam potencialmente acessar as mesmas células de memória e entrar em conflito.

Variáveis[DFF9]

Por este motivo, as linguagens de programação criaram o conceito de **variáveis** para abstrair a complexidade do endereçamento das células de memória. Imagine agora, que o programador, ao invés de utilizar os endereços da memória, tenha à sua disposição **rótulos** com nomes intuitivos que identificam cada uma das células da memória.

Conceitos*[DFF10]*

A variável é uma abstração da célula de memória. Ao invés de utilizar o endereço (posição da célula dentro da memória), a variável é identificada através de um rótulo (nome) escolhido com auxílio do bom senso do programador.

Sempre que for necessário referir-se a algum dado, o programador utiliza o rótulo associado à variável que contém o mesmo.

Utilizando este novo ponto de vista, a memória pode ser entendida de uma forma simplificada: como um repositório para um número muito grande de variáveis.

Desta forma, cada **variável** é caracterizada pelo seu **rótulo** e **conteúdo**.

Exemplo

Vamos rever o algoritmo de Euclides que calcula o MDC (máximo divisor comum), apresentado na Introdução, desta vez utilizando variáveis. Ele começa lendo dois números, que são a entrada do algoritmo. Portanto, estes dois números devem ser armazenados em duas variáveis, que decidimos chamar de A e B. Não é mais necessário nos preocuparmos com a disposição desses dados na memória, a Figura 2 mostra a representação abstrata da memória.

1) **Leia** um número e armazene na **variável** `A`.
2) **Leia** um número e armazene na **variável** `B`.
3) **Divida** o valor da variável `A` pelo valor da variável `B`. Guarde o quociente na **variável** `Q` e o resto na **variável** `R`.
4) **Se** o valor da variável `R` for 0 (zero), então **mostre** o valor da variável `B` e **PARE**.
5) **Copie** o conteúdo da variável `A` para a variável `B`.
6) **Copie** o conteúdo da variável `B` para a variável `R`.
7) **Retorne** ao passo 3.

Algoritmo 2 - MDC usando variáveis

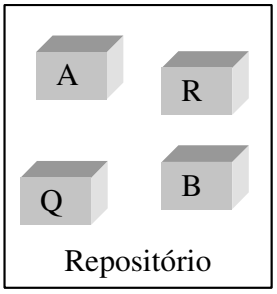

Figura 2 - Variáveis na memória

No terceiro passo, o algoritmo calcula o quociente dos dois números (identificados pelas variáveis A e B). Como não desejamos perder o valor destes dois números, precisamos escrever o quociente e o resto em novas variáveis, que foram chamadas respectivamente de Q e R. Finalmente, antes de o algoritmo continuar a execução, note que a próxima divisão espera que os dois números estejam nas variáveis A e B. Por este motivo, precisamos copiar respectivamente, o quociente (que está na variável Q) para a variável A e o resto (que está na variável R) para a variável B.

Desafio: É possível reescrever este algoritmo de forma a utilizar menos variáveis. Como? Quais seriam as vantagens e desvantagens de reduzir o número de variáveis? Compare a importância e a dificuldade da redução do número de células de memória com a redução do número de variáveis.