Lições de Programação

LIÇÕES DE PROGRAMAÇÃO

ALGORITMOS FUNDAMENTAIS

GILSON PEREIRA DO CARMO FILHO

LIÇÕES DE PROGRAMAÇÃO

ALGORITMOS FUNDAMENTAIS

Há alguns mecanismos computacionais que podem ser usados na construção de qualquer tipo de programa.

Esses mecanismos, chamados de algoritmos fundamentais, são a base para o estudo da programação de computadores.

Este material apresentará a você a essência desses conceitos fundamentais, importantes para lidarmos com os aspectos mais complexos da programação.

GILSON FILHO

Cientista da Computação e mestre em Engenharia de Teleinformática pela Universidade Federal do Ceará. Especialista em Gerenciamento de Projetos pela Fundação Getúlio Vargas. Possui larga experiência profissional na área de TI, com ênfase em Engenharia de Software, Bancos de Dados e Sistemas de Informação. Foi professor na Universidade Federal do Ceará. Atualmente é professor na Universidade de Fortaleza e consultor nas áreas de educação e tecnologia.

SUMÁRIO

1

INTRODUÇÃO

O DESAFIO DA PROGRAMAÇÃO

Quando iniciamos no mundo da programação, várias vezes nos deparamos com problemas simples que podemos resolver com muita facilidade, mas para os quais temos grande dificuldade em formular uma solução computacional.

Isso acontece porque podemos resolver muitos problemas sem entender ou formular explicitamente o método usado para obter a solução.

Infelizmente, não podemos ter esse luxo quando lidamos com os computadores, pois a solução computacional de um problema deve ser obtida por meio de uma sequência de instruções **explícitas** e **não ambíguas**, expressas em uma linguagem de programação.

Dessa forma, a programar é uma atividade bastante exigente, pois requer muita reflexão, planejamento cuidadoso, precisão lógica, persistência e atenção aos detalhes.

Por outro lado, programar pode ser uma experiência desafiadora, emocionante e satisfatória, com espaço considerável para criatividade e expressão pessoal. Se for abordada nesse espírito, as chances de sucesso serão bastante ampliadas!

ALGORITMOS E PROGRAMAS

A noção de algoritmo é básica para toda a programação de computadores. Assim, devemos começar com uma análise cuidadosa desse conceito.

Existem muitas definições de algoritmo. A seguinte definição foi apresentada por Donald E. Knuth no livro *The Art of Computer Programming*, e é apropriada na Ciência da Computação:

Algoritmo é um conjunto finito de regras que fornece uma sequência de operações para resolver um problema específico.

Portanto, um algoritmo corresponde a uma solução para um problema, independente de qualquer linguagem de programação.

Por sua vez, um **programa** pode ser pensado como sendo um **algoritmo expresso em uma linguagem de programação**.

De um modo geral, conforme a Figura 1 abaixo, todo programa recebe um conjunto de dados como **entrada** e o manipula de acordo com suas instruções (**processamento**), produzindo outro conjunto de dados como **saída**, que representa a solução para o problema.

Figura 1: Execução de um programa de computador

ALGORITMOS FUNDAMENTAIS

Há alguns mecanismos computacionais que podem ser usados na construção de qualquer tipo de programa.

Esses mecanismos, chamados de **algoritmos fundamentais**, foram descritos por R. Geoff Dromey no livro *How to Solve it By Computer*, e são a base para o estudo da programação de computadores.

Depois que compreendemos a essência desses conceitos fundamentais, finalmente estamos preparados para enfrentar os aspectos mais complexos da programação.

Ao todo, são nove os algoritmos fundamentais:

1. Troca dos valores de duas variáveis
2. Contagem
3. Soma de um conjunto de números
4. Cálculo do fatorial
5. Cálculo de uma série infinita

6. Geração da sequência de Fibonacci
7. Inversão dos dígitos de um número inteiro
8. Conversão de base
9. Conversão de caractere para número

Nos próximos capítulos, apresentaremos esses algoritmos fundamentais em detalhes, agrupados em três categorias: algoritmos básicos, algoritmos iterativos e algoritmos de representação e manipulação da informação.

Para cada algoritmo, teremos um problema contexto, exemplos de aplicações, a descrição dos passos e a respectiva implementação em uma linguagem de programação (JavaScript).

Todos os códigos fontes estão também hospedados no meu GitHub (gilsonpcf).

Boa leitura!

2

ALGORITMOS BÁSICOS

Um dos mecanismos fundamentais mais utilizados na programação é o de trocar os valores associados a duas variáveis. Em particular, esta operação é amplamente usada em muitos algoritmos de ordenação.

A ideia de contar também é uma parte essencial de muitos procedimentos computacionais mais elaborados. Uma extensão da ideia de contagem forma a base do mecanismo frequentemente usado para somar um conjunto de dados.

Neste capítulo, serão apresentados os seguintes algoritmos fundamentais, que formam a base para todos os outros:

- Troca dos valores de duas variáveis
- Contagem
- Soma de um conjunto de números

TROCA DOS VALORES DE DUAS VARIÁVEIS

Problema

Dadas duas variáveis, **a** e **b**, trocar os valores atribuídos a elas.

Aplicações

Algoritmos de ordenação.

Descrição do Algoritmo

1. Guardar o valor original da variável **a** em uma variável auxiliar.
2. Atribuir à variável **a** o valor original da variável **b**.
3. Atribuir à variável **b** o valor original da variável **a**, que está armazenado na variável auxiliar.

Implementação em JavaScript

```
aux = a;

a = b;

b = aux;
```

CONTAGEM

Problema

Dado um conjunto de `n` notas de alunos em um exame, fazer uma contagem do número de alunos que foram aprovados no exame. Será considerado aprovado o aluno que tirar nota 50 ou maior (no intervalo de 0 a 100).

Aplicações

Todas as formas de contagem.

Descrição do Algoritmo

1. Obter o número de notas a serem processadas.
2. Inicializar a contagem com zero.
3. Enquanto houver notas a serem processadas, fazer repetidamente:
    - obter a próxima nota;
    - se a nota for suficiente para passar no exame (≥ 50) então adicionar 1 (um) à contagem.
4. Exibir a contagem (número total de aprovações).

Implementação em JavaScript

```
var n = prompt("Número de notas");

var nota;

var contagem = 0;

var i = 0;

while (i < n) {

      i = i + 1;

      nota = prompt("Nota do aluno:");

      if (nota >= 50) {

            contagem = contagem + 1;

      }

}

document.write("Número de aprovações: "
+ contagem);
```

SOMA DE UM CONJUNTO DE NÚMEROS

Problema

Dado um conjunto de `n` números, calcular a soma desses números. Assumir que `n` é maior ou igual a zero.

Aplicações

Cálculos de média, cálculos de variância e mínimos quadrados.

Descrição do Algoritmo

1. Obter a quantidade de números a serem somados.
2. Inicializar a soma com 0 (zero).
3. Enquanto menos do que `n` números tiverem sido somados, fazer repetidamente:
    - obter o próximo número;
    - calcular a soma atual, adicionando o número obtido à soma mais recente.
4. Exibir a soma dos `n` números

Implementação em JavaScript

```
var n = prompt("Quantidade de números:");

var numero;

var soma = 0;

var i = 0;

while (i < n) {

      i = i + 1;

      numero = parseInt(prompt("Número:"));

      soma = soma + numero;

}

document.write("Soma = " + soma);
```

3

ALGORITMOS ITERATIVOS

A habilidade mais importante que precisamos desenvolver na programação é, provavelmente, a capacidade de expressar problemas de maneira que se possa formular uma solução iterativa, isto é, obtida por meio de um processo repetitivo.

Infelizmente, antes do nosso primeiro contato com a programação, a maioria de nós teve pouca ou nenhuma experiência em formular soluções iterativas para problemas. No entanto, depois que entendemos a ideia de "pensar iterativamente", começamos a ter condição de formular soluções computacionais para problemas cada vez mais complexos.

Neste capítulo, serão apresentados os seguintes algoritmos fundamentais, que consistem em soluções iterativas:

- Cálculo do fatorial
- Cálculo de uma série infinita
- Geração da sequência de Fibonacci

CÁLCULO DO FATORIAL

Problema

Dado um número `n`, calcular o fatorial de `n` (escrito como `n!`), onde `n` ≥ 0.

Aplicações

Probabilidade, cálculos estatísticos e matemáticos.

Descrição do Algoritmo

1. Obter o número `n`, onde `n` ≥ 0.
2. Inicializar o produto com 1 (um) e a contagem de produtos com 0 (zero).
3. Enquanto menos do que `n` produtos tiverem sido calculados, fazer repetidamente:
    - ◇ incrementar a contagem de produtos;
    - ◇ calcular o produto atual, multiplicando a variável de contagem pelo produto mais recente.
4. Exibir o resultado (`n!`).

Implementação em JavaScript

```
var n = prompt("Número:");

var fator = 1;

for (var i = 1; i <= n; i++) {

      fator = i * fator;

}

document.write("Fatorial = " + fator);
```

CÁLCULO DE UMA SÉRIE INFINITA

Problema

Calcular o valor de S, conforme definido pela seguinte série infinita:

$$S = \frac{x}{1!} - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \cdots$$

Aplicações

Cálculos matemáticos e estatísticos.

Descrição do Algoritmo

1. Obter o número de termos (**`n`**) e o valor de **`x`**.
2. Definir as condições iniciais para o primeiro termo, que não pode ser calculado iterativamente.
3. Enquanto menos do que **`n`** termos tiverem sido calculados, fazer repetidamente:
    - identificar o termo atual;
    - gerar termo atual a partir do seu antecessor;
    - adicionar o termo atual, com o sinal apropriado, à soma acumulada.

4. Exibir o valor da soma.

Implementação em JavaScript

```
function fatorial(n) {
    var fator = 1;
    for (var i = 1; i <= n; i++) {
        fator = i * fator;
    }
    return fator;
}


var n = prompt("Número de termos:");
var x = prompt("Valor de x:");
var termo;
var s = 0;
var i = -1;
```

```
var sinal = -1;

for (var j = 1; j <= n; j++) {

    i = i + 2;

    sinal = -sinal;

    termo = sinal * Math.pow(x, i) /
fatorial(i);

    s = s + termo;

}

document.write("S = " + s);
```

GERAÇÃO DA SEQUÊNCIA DE FIBONACCI

Problema

Gerar e imprimir os **`n`** primeiros termos da sequência de Fibonacci, onde **`n`** ≥ 1. Os primeiros termos são:

0, 1, 1, 2, 3, 5, 8, 13, ...

Cada termo, além dos dois primeiros, é derivado da soma dos seus dois antecessores mais próximos.

Aplicações

Botânica, teoria das redes elétricas, ordenação e pesquisa.

Descrição do Algoritmo

1. Obter o número de termos a serem gerados (**`n`**).
2. Atribuir valores aos dois primeiros números de Fibonacci, **`a`** e **`b`**.
3. Inicializar a contagem de números gerados.
4. Se **`n`** for igual a **`1`**, então exibir o primeiro número de Fibonacci, senão exibir os dois primeiros números.

5. Enquanto menos do que **`n`** números de Fibonacci tiverem sido gerados, fazer repetidamente:
    - atualizar a contagem de números gerados;
    - gerar o próximo número de Fibonacci, **`c`**;
    - escrever o próximo número de Fibonacci;
    - atribuir ao primeiro número de Fibonacci o valor do segundo número;
    - atribuir ao segundo número de Fibonacci o valor do próximo número gerado.

Implementação em JavaScript

```
var n = prompt("Número de termos:");

var a = 0;

var b = 1;

var i = 2;

var c;
```

```
if (n == 1) {
    document.write(a);
} else {
    document.write(a + ", " + b);
}

while (i < n) {
    i = i + 1;
    c = a + b;
    document.write(", " + c);
    a = b;
    b = c;
}
```

4

ALGORITMOS DE REPRESENTAÇÃO E MANIPULAÇÃO DA INFORMAÇÃO

Outro fundamento que devemos aprender no início da nossa jornada na programação é o entendimento de como os computadores representam e manipulam informações.

Em particular, devemos considerar as diferenças entre as representações de informações numéricas e alfanuméricas, bem como manipular e converter essas representações.

Neste capítulo, serão apresentados os seguintes algoritmos fundamentais, que trabalham com a representação e manipulação da informação:

- Inversão dos dígitos de um número inteiro
- Conversão de base
- Conversão de caractere para número

INVERSÃO DOS DÍGITOS DE UM NÚMERO INTEIRO

Problema

Inverter a ordem dos dígitos de um número inteiro positivo.

Aplicações

Hashing e recuperação de informações, aplicativos de banco de dados.

Descrição do Algoritmo

1. Obter o número inteiro positivo a ser invertido.
2. Definir a condição inicial para o número invertido.
3. Enquanto o número que está sendo invertido for maior do que zero, faça:
    - extrair o dígito mais à direita desse número, usando o resto da divisão dele por 10;
    - atualizar o número invertido, multiplicando o seu valor anterior por 10 e adicionando a ele o dígito mais à direita extraído recentemente;

- remover o dígito mais à direita do número que está sendo invertido, usando a divisão inteira por 10;

4. Exibir o número invertido.

Implementação em JavaScript

```
var n = prompt("Número inteiro
positivo:");

var invertido = 0;

var d;

while (n > 0) {

    d = n % 10;

    invertido = invertido * 10 + d;

    n = Math.trunc(n / 10);

}

document.write("Número invertido: " +
invertido);
```

CONVERSÃO DE BASE

Problema

Converter um número inteiro decimal para a sua representação binária correspondente.

Aplicações

Interpretação de dados e instruções armazenados em computador.

Descrição do Algoritmo

1. Obter o número decimal a ser convertido e inicializar o quociente com este número.
2. Inicializar a contagem de novos dígitos com zero.
3. Fazer repetidamente até que o quociente seja zero:
    - ◇ calcular o próximo dígito mais significativo, usando o resto da divisão do quociente atual pela nova base (2);
    - ◇ armazenar esse dígito no *array* de saída;
    - ◇ incrementar a contagem de novos dígitos;

- calcular o próximo quociente a partir do seu antecessor, usando a divisão inteira pela nova base (2).

4. Exibir a representação binária do número a partir do *array* de saída, usando a ordem inversa dos elementos.

Implementação em JavaScript

```
var n = prompt("Número inteiro
decimal:");

var q = n;

var ndigit = 0;

var r;

var binario = [];

do {

    r = q % 2;

    binario[ndigit] = r;
```

```
        ndigit = ndigit + 1;
        q = Math.trunc(q / 2);
    }
    while (q != 0);
    document.write("Número binário: ");
    for (var i = ndigit-1; i >= 0; i--) {
        document.write(binario[i]);
    }
```

CONVERSÃO DE CARACTERE PARA NÚMERO

Problema

Dada a representação em caracteres de um número inteiro, convertê-la para o seu formato decimal convencional.

Aplicações

Aplicativos de negócios, processamento de dados.

Descrição do Algoritmo

1. Obter a cadeia de caracteres a ser convertida para decimal, e determinar o seu comprimento `n`.
2. Inicializar o valor decimal com zero.
3. Definir como base o valor ordinal do caractere "0" na tabela ASCII.
4. Enquanto menos do que `n` caracteres tiverem sido examinados, faça:
    - converter o próximo caractere para o dígito decimal correspondente;

- deslocar o valor decimal atual para a esquerda e adicionar o dígito do caractere atual.

4. Exibir o número inteiro decimal correspondente.

Implementação em JavaScript

```
var numero = prompt("Número inteiro:");

var n = numero.length;

var decimal = 0;

var base = 48;

var digito;

for (var i = 0; i < n; i++) {

    digito = numero.charCodeAt(i) -
base;

    decimal = decimal * 10 + digito;

}

document.write("Formato decimal: " +
decimal);
```

5

CONCLUSÃO

CONSIDERAÇÕES FINAIS

Finalmente terminamos nossa jornada. Até aqui, vimos nove algoritmos fundamentais, que são a base para que você implemente soluções computacionais para diversos problemas presentes no mundo real.

Assim que se sentir confortável com a implementação desses algoritmos, você estará mais preparado para resolver problemas usando programação e lidar com os seus aspectos mais complexos.

COMO CONTINUAR SEUS ESTUDOS

Para continuar o seu aprendizado em programação, sugiro o estudo de **técnicas de *array*** e algoritmos de **ordenação e pesquisa**. Mas vale ressaltar que a prática é fundamental para o processo de aprendizagem.

Para isso, indico o site **beecrowd**, cujo principal objetivo é promover a prática de programação e o compartilhamento de conhecimento.

O beecrowd contém mais de 2.000 problemas divididos em 9 grandes categorias, e possui um sistema de correção automática. Isso ajudará você a focar em temas específicos e regular sua aprendizagem, aumentando sua motivação.

Espero que você continue seus estudos e práticas com algoritmos. Assim, você exercita seu raciocínio lógico e terá cada vez mais facilidade para encontrar e implementar a solução de problemas do nosso cotidiano.

Bons estudos!

6

REFERÊNCIAS

DROMEY, R. G. **How to solve it by computer**. Londres: Prentice-Hall International, 1982.

KNUTH, D. E. **The art of computer programming**. 3. ed. Reading, MA: Addison-Wesley, 1997. v. 1.

LIÇÕES DE PROGRAMAÇÃO

ALGORITMOS FUNDAMENTAIS

Systema
EDUCAÇÃO E TECNOLOGIA