Fundamentos de Programação

Filipe Francisco

15 de março de 2018

Problemas

Alguns problemas nos algoritmos esta semana:

- Precedência:
  - par ou ímpar: a+b%2==0
  - média em FUP: 5*ME+3*MA+3*NT/10
  - sai fora Baskara: -b+sqrt(delta)/(2*a)
- Leitura de valores:
  - scanf("%d %d ",&a,&b)
  - não colocar espaço no final do scanf!
    - isso indica ao scanf que você ainda está esperando receber algo!
- Operações básicas:
  - tentar fazer a/b ou a%b sem antes verificar se b==0!
  - é uma tentativa de divisão por zero, logo acontece um erro!
- Imprimir valores com quantidades certas de dígitos:
  - printf("%02d",a): imprime um inteiro com (no mínimo) dois dígitos
  - printf("%.1f",a): imprime um float com exatamente uma casa decimal (arredonda as demais)

Moodle

- Moodle é uma ferramenta útil para algumas coisas:
  - disponibilizar de atividades para os alunos praticarem
  - submeter algoritmos para os problemas, com correção automática
  - nos testes em que falhou, indica o que o algoritmo imprimiu e o que era esperado que imprimisse
  - fazer os alunos passarem raiva :)
- Por que passar raiva?
  - para corrigir os algoritmos automaticamente, o Moodle compara a saída do programa com um texto pré-configurado
  - ou seja, o Moodle analisa como incorretos algoritmos que:
    - imprimem qualquer coisa além do que ele deseja
    - em vez de imprimir um espaço, salta linha
    - imprime espaço onde não deve
    - imprime os resultados numa ordem diferente do que ele espera
    - ⋮
- Apesar disso, o Moodle é uma ótima plataforma para o ensino de programação!

Condicionais em C

- Condicionais em C podem ser feitos de algumas formas
- if:
  - **if** (*condição*) { [*código*] }
  - caso a condição seja verdadeira, executa o código
  - útil para fazer algo caso uma condição seja satisfeita, mas não fazer nada caso contrário
- if-else:
  - **if** (*condição*) { [*código 1*] }
    else { [código 2] }
  - caso a condição seja verdadeira, executa o código 1
  - caso a condição seja falsa, executa o código 2
  - recomendado em casos com códigos grandes, por permitir maior organização

Condicionais em C

- if-else (outra forma):
  - **if** (*condição*) [*código 1*];
    else [código 2];
  - caso a condição seja verdadeira, executa o código 1
  - caso a condição seja falsa, executa o código 2
  - útil para códigos pequenos, com bem poucos comandos
- if-else (mais uma forma):
  - *condição* ? [*comando 1*] : [comando 2];
  - caso a condição seja verdadeira, executa o comando 1
  - caso a condição seja falsa, executa o comando 2
  - o uso não é recomendado em substituição às outras formas vistas, pois a notação pode confundir

Condicionais em C

- if-else (mais uma forma):
  - *condição* ? [*comando 1*] : [comando 2];
- Apesar de não ser recomendado para substituir as outras formas, esta forma também tem uma utilidade!
- Exemplo:
  - **char** sinal = (x<0) ? '-' : '+';
  - se $x < 0$, sinal receberá '-'
  - se $x \geq 0$, sinal receberá '+'
  - é útil para resumir um pouco a notação de um if-else nestes casos

Operadores de comparação

- Igualdade (igual a): ==
  - **if** (a==b) ...
- Diferença (diferente de): !=
  - **if** (a!=b) ...
- Maior que: >
  - **if** (a>b) ...
- Menor que: <
  - **if** (a<b) ...
- Maior ou igual a: >=
  - **if** (a>=b) ...
- Menor ou igual a: <=
  - **if** (a<=b) ...

Operadores lógicos

- Operadores de comparação são úteis para verificar condições
- Porém, sozinhos, são um pouco ineficientes
- Por exemplo, como verificar se $a < x < b$ utilizando apenas condicionais e operadores de comparação?
  - **if** $(a < x)$ **if** $(x < b)$ printf("sim");
- Não conseguimos fazer isso sem utilizar dois condicionais, um dentro do outro
- Operadores (ou conectivos) lógicos facilitam não só este caso, mas alguns outros!
  - Obs1: os operadores lógicos trabalham com valores booleanos (V/F, ou 0/1)!
  - Obs2: os problemas vistos até agora não exigem o uso de operadores lógicos!

Operadores lógicos

- Operador de negação: ! (não/not)
  - operador unário (trabalha com apenas um valor booleano)
  - nega o resultado de uma condição
  - ex: if (!cond) ...
  - se cond é verdadeira, (!cond) é falsa
  - se cond é falsa, (!cond) é verdadeira
  - não precisamos mais de um else só para trabalharmos quando nossa condição for falsa!

Operadores lógicos

- Operador de conjunção: && (e/and)
  - verifica se duas condições são verdadeiras ao mesmo tempo
  - ex: if (cond1 && cond2) ...
  - se cond1 e cond2 são ambas verdadeiras, (cond1 && cond2) é verdadeira
  - se cond1 é falsa ou cond2 é falsa, (cond1 && cond2) é falsa
  - podemos verificar se duas condições são verdadeiras ao mesmo tempo sem ter de inserir um if dentro de outro!

Operadores lógicos

- Operador de disjunção: || (ou/or)
  - verifica se pelo menos uma das duas condições é verdadeira
  - ex: if (cond1 || cond2) ...
  - se cond1 é verdadeira, (cond1 || cond2) é verdadeira
  - se cond2 é verdadeira, (cond1 || cond2) é verdadeira
  - se ambas cond1 e cond2 são falsas, (cond1 || cond2) é falsa
  - podemos verificar se pelo menos uma das duas condições é verdadeira sem termos de colocar ifs consecutivos!

Condicional switch

- O switch (ou switch-case) é mais um condicional!
- Serve para compararmos uma variável com uma lista de valores
- Esquema de um switch:
  - switch (var) {

    caso (valor1): [código 1]; break;
    caso (valor2): [código 2]; break;
    caso (valor3): [código 3]; break;
    default: [código padrão];

    }
  - compara var com cada um dos valores, e executa o código associado
- O código acima é semelhante semanticamente ao seguinte código:
  - se (var==valor1) [código 1];
    senão se (var==valor2) [código 2];
    senão se (var==valor3) [código 3];
    senão [código padrão];

Condicional switch

- O comando break em um switch serve para indicarmos que devemos executar apenas um bloco de código
- Podemos fazer o switch executar mais de um caso para uma variável
- Para isso, basta eliminar o break nas linhas desejadas!
  - switch (var) {

    caso (valor1): [código 1];
    caso (valor2): [código 2];
    caso (valor3): [código 3];

    }
- O código acima é semelhante semanticamente ao seguinte código:
  - se (var==valor1) [código 1];
    se (var==valor2) [código 2];
    se (var==valor3) [código 3];

Condicional switch

- Ao encontrarmos um break, saímos do switch sem verificar mais nada
- Apesar disso, ausência do break pode ser útil!
  - switch (var) {

```
caso (valor1): [código 1]; break;
caso (valor2): [código 1]; break;
caso (valor3): [código 1]; break;
caso (valor4): [código 2]; break;
caso (valor5): [código 2]; break;
caso (valor6): [código 2]; break;
```

    }
- Removendo alguns break, podemos reduzir o tamanho do switch!
  - switch (var) {

```
caso (valor1):
caso (valor2):
caso (valor3): [código 1]; break;
caso (valor4):
caso (valor5):
caso (valor6): [código 2]; break;
```

    }

Condicional switch em C

- Esquema do switch em C:
  - **switch** (var) {

    case (valor1): [código 1]; **break**;
    case (valor2): [código 2]; **break**;
    case (valor3): [código 3]; **break**;
    default: [código padrão];

    }

Condicional switch em C

- Considerações sobre o condicional switch:
  - o código é executado até achar um break, ou sair do switch
  - a variável e os valores a serem comparados não podem ser float
    - podem ser int ou char, o que já nos ajuda bastante!
  - além de variável, também podemos checar o valor de uma fórmula!
    - por exemplo, checar o valor de a%b, ou o valor de toupper(car)
  - podemos ter quantos valores/cases quisermos
    - cada case é sucedido de valor, um ':' e um código
  - o caso default (padrão) é opcional
    - é um caso onde se cai quando nenhum break foi encontrado
    - não precisa de break!