MC-102 — Exercícios de fixação

Lehilton Pedrosa

Instituto de Computação – Unicamp

Para provocar

Ciência da Computação não é mais sobre computadores do que Astronomia é sobre telescópios.

(autor não confirmado)

Aula 1

Exercício 1 - Em dupla

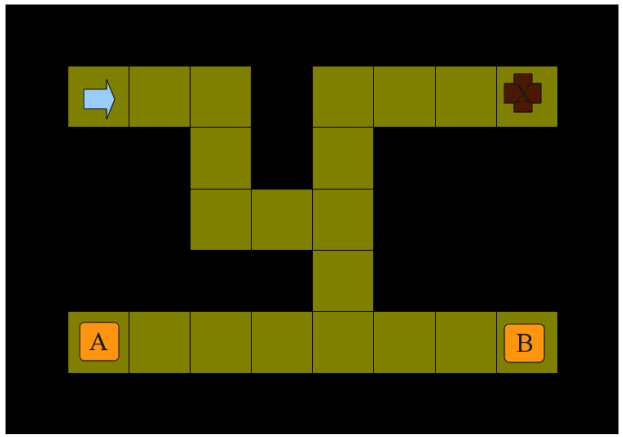
Soma

Escrevam um algoritmo para somar dois números decimais inteiros usando somente a tabuada.

Exercício 2 - Robô e tesouro

Um robô está na posição da seta. Existe um tesouro escondido em um caixote amarelo, mas no outro há uma armadilha que o deixa preso para sempre. Para saber em que caixote está, pode-se verificar a chave marrom e ler o valor de X. Se X = -1, então o tesouro está em A, mas se X = 1, então o tesouro está em B. Escreva um algoritmo para pegar o tesouro. O robô só entende os seguintes comandos virar à direita, virar à esquerda, abrir um caixote, andar N quadrados, ler um valor e pegar o tesouro.

1. Identifique todas as partes relacionadas (problema, entrada, saída etc.).
2. Concorde ou discorde: um algoritmo que abre um caixote antes de ler a chave está bem definido.



Exercício 3 - Corrida

Ana dá uma volta no percurso de corrida do Lago a cada 15 min. Suponha que um amigo comece a correr X minutos mais tarde em um ritmo de Y min por volta. Dados X e Y, Ana e seu amigo irão se encontrar no início da volta antes de uma hora? Quando?

1. Escreva um algoritmo para o problema. Lembre-se de sempre especificar claramente todas as partes relacionadas (problema, entrada, saída etc.)!

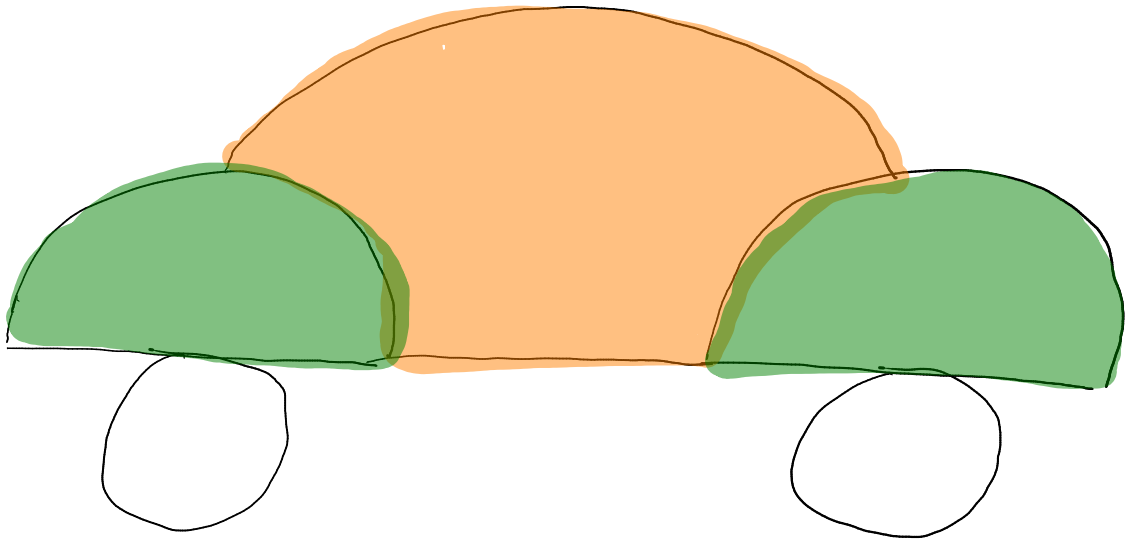
Exercício 4 - Desenho

Execute:

- Pegue papel em branco e lápis;
- Desenhe um semicírculo na esquerda;
- Desenhe outro semicírculo na direita;
- Desenhe um círculo menor em baixo de de cada semicírculo;
- Desenhe um semicírculo maior sobre os semicírculos;
- Pinte os semicírculos pequenos de verde;
- Pinte o semicírculo grande de laranja;

1. Que figura você obteve? O procedimento acima é um algoritmo? Explique.
2. Reescreva o procedimento como um algoritmo bem definido que obtenha o desenho da página seguinte. Atente-se para tamanho, ordem etc.

Exercício 4 - Desenho



Aula 3

Exercício 5 - Plantão

Problema

Em um hospital, é preciso escalar um médico para plantão em cada dia do mês. Sabendo-se que `medico` é um vetor de dimensão N, onde N é o número de médicos e `medico[i]` é a matrícula do i-ésimo médico, queremos construir um novo vetor `plantao` com 30 posições, onde `plantao[j]` contém a matrícula do médico escalado para o dia j. O objetivo é obter uma escala mais justa possível.

Exercício 5 - Plantão

```
1.  plantao[1] = medico[1];
2.  dia = 2;
3.  enquanto (dia ≤ 30) faça {
4.      plantao[dia] = plantao[dia - 1];
5.      dia = dia + 1;
6.  }
```

Questões

1. Como você descreveria uma escala justa? Descreva a entrada e a saída usando uma linguagem mais formal do que na descrição do problema.
2. No algoritmo escrito acima, descreva as configurações inicial, após a execução das linhas 1, 2, após a i-ésima execução da linha 4, e a configuração final.
3. Argumente (informalmente ou formalmente) que o algoritmo acima não está correto de acordo com a especificação de saída que você descreveu.

Exercício 6 - Fração

Concorde ou discorde

"O problema de decidir se dois números racionais são iguais é indecidível, já que existem infinitas formas de escrever um mesmo número."

Justifique considerando a seguinte questão: como podemos mostrar que um problema é decidível (resolvível)?

Aula 4

Exercício 7 - Papel picado

Vocês decidiram fazer uma festa surpresa para uma amigo hoje à noite. Para isso, você precisa cortar papel colorido e produzir bastante confete. Cada folha de papel deve ser cortada em 32 linhas e 32 colunas. Um amigo emprestou-lhe uma guilhotina e aconselhou: o melhor jeito de cortar o papel é enrolar uma folha e ir fazendo fitas, depois é só cortar cada uma das fitas separadamente.

1. Descreva o procedimento sugerido como um algoritmo.
2. Se você precisar cortar 100 folhas e gastar um segundo por cada corte da guilhotina, então quanto tempo levará para terminar a tarefa?
3. Concorde ou discorde: embora esse algoritmo possa demorar horas, isso não significa que o problema não é viável, pois pode haver algoritmos mais espertos para a tarefa. Justifique.

Exercício 8 - Potência à potência

ENTRADA: número inteiro $k \geq 1$
SAÍDA: 2^{2^k}

Algoritmo 1:

1. p = 1
2. faça k vezes:
   - p = 2*p
3. $r = 1$
4. faça p vezes:
   - $r = 2 * r$
5. devolva r

Algoritmo 2:

1. $r = 2$
2. faça k vezes:
   - $r = r * r$
3. devolva r

Concorde ou discorde

1. O segundo algoritmo é mais rápido porque tem menos passos.
2. O segundo algoritmo é mais rápido quando assumimos que cada instrução elementar gasta o mesmo tempo.

Exercício 9 - Moedas

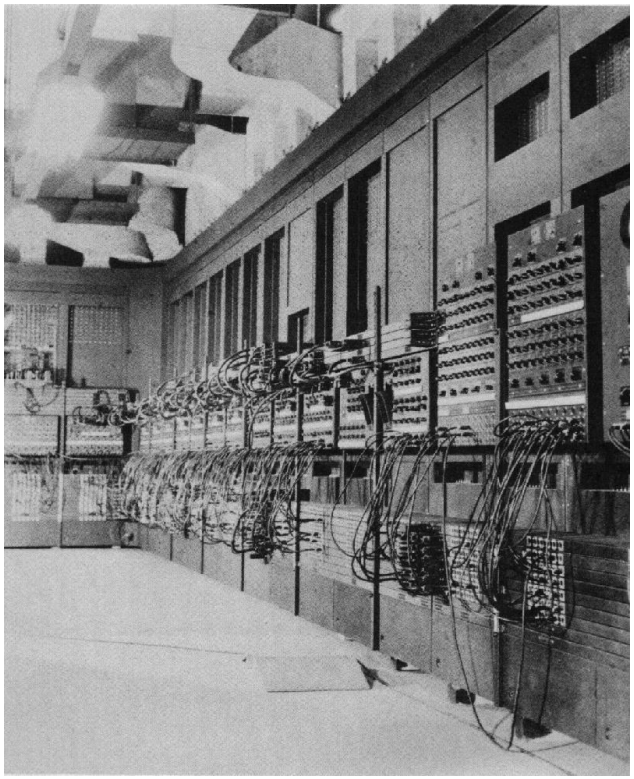
Uma máquina de troco (*change-machine*) é um equipamento comum em certos lugares, onde só se aceitam moedas. O objetivo é inserir uma ou mais cédulas e obter o valor equivalente no menor número de moedas. Suponha que a máquina possua um total de N moedas para cada valor R$ 1,00, R$ 0,50, R$ 0,25 e R$ 0,10. Uma maneira de resolver o problema é testar todas as combinações possíveis de moedas, verificar quais correspondem ao valor inserido em cédulas e memorizar a de menor número de moedas. Naturalmente o número de combinações é muito grande. Calcule o número de combinações possíveis de moedas e argumente que esse algoritmo não é eficiente. Você é capaz de propor um algoritmo melhor?

Aula 5

Um pouco de história

Programação em código absoluto ou binário (apenas 0s e 1s).

ENIAC



Exercício 10 - Declarando variáveis

Complete e corrija o código abaixo.

Tipos

```
int main() {
    10 = a;
    b = -6;
    casa = 100000;
    d = 33000;
    e = -80000;
    f = 30;
    var = a;
    2x = 80;
    return = 3
}
```

Exercício 11 - Invertendo ordem

Escreva um programa que leia três números inteiros e escreva os números na ordem inversa, sem espaço entre eles.

Entrada:

```
111 9999 0
```

Saída:

```
09999111
```

Aula 6

Tabela ASCII

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0 16	Caracteres de Controle															
32		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
48	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
64	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
80	P	Q	R	S	T	U	V	W	X	Y	Z	[	/	]	⋀	_
96	‘	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
112	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	

Exercício 12 - Ordem do caractere

Escreva um programa que leia um caractere minúsculo e imprima o valor ASCII do elemento e a ordem alfabética desse caractere. Observe a tabela ASCII e procure um padrão.

Entrada:

```
e
```

Saída:

```
codigo ascii do e: 101
ordem do e no alfabeto: 5
```

Exercício 13 - Ordem de execução

```
#include <stdio.h>
main(void){
    int n;
    printf("Digite um número maior que 0: ");
    scanf("%d", &n);
    printf("O valor da expressao eh: %d \n",
            1 + 2 * 30 / (n % 4 * -5));
}
```

1. Indique a ordem em que são executadas as operações.
2. Reescreva o programa de forma que cada comando contenha apenas uma operação aritmética.
3. *(pequeno desafio)* Existe um problema nesse programa. Que problema é esse? Quando ele acontece?

Aula 7

Exercício 14 - Dígitos

Escreva um programa que leia um número real x e imprima o terceiro dígito antes e depois da vírgula da raiz quadrado do número lido (ex., para $x = 12345678$, temos $\sqrt{12345678} = 3513{,}641700572...$, então o terceiro antes da vírgula é 5 e o terceiro depois da vírgula é 1.

Entrada:

```
12345678
```

Saída:

```
5 1
```

Aula 8

Exercício 15 - Grande par

Problema

Considere um problema de decidir se um número inteiro é grande par: isso é, ele é par, maior que 100 e vale o dobro de dois números ímpares.
Escreva um programa que leia um número e imprime "SIM" quando for grande par, e "NAO" caso contrário.

Exercício 16 - Grande par com operadores lógicos

Problema

Escreva um programa que leia um número e imprime "SIM" quando for grande par, e "NAO" caso contrário. Agora use operadores lógicos para diminuir o programa.

Exercício 17 - Sem e com operadores lógicos

Programas

Escreva um programa que ordene três números.

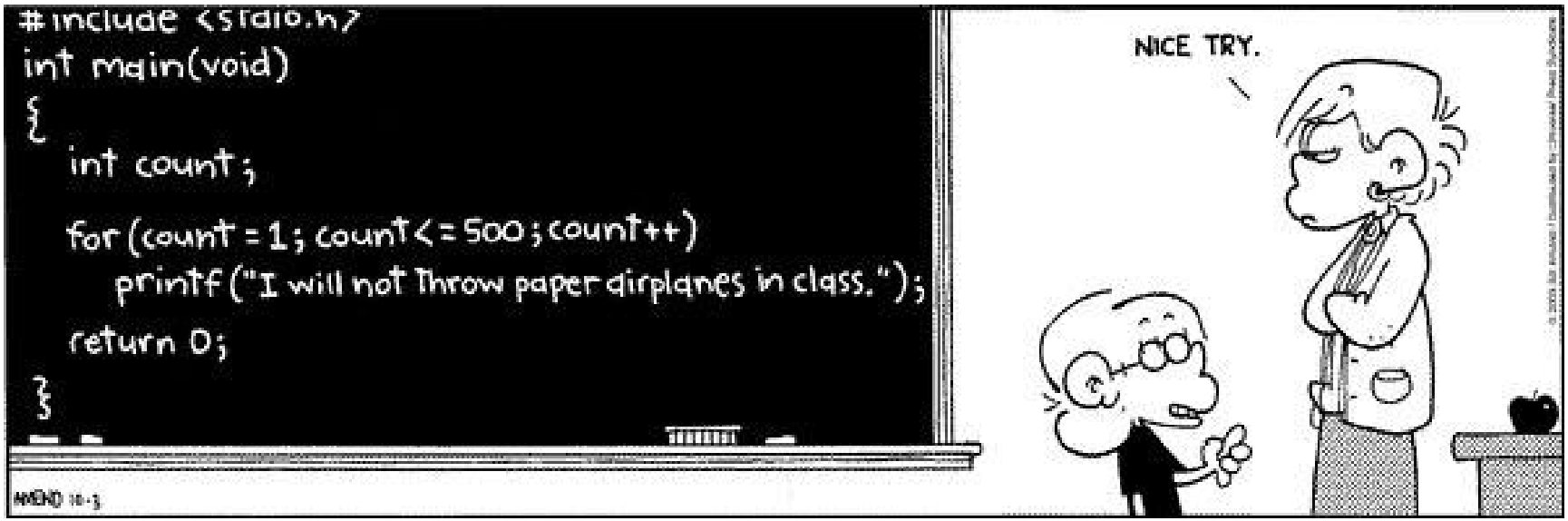
Escreva duas versões para cada programa acima: uma sem utilizar operadores lógicos e outra utilizando operadores lógicos. Depois responda: existem problemas que não podem ser resolvidos sem utilizar operadores lógicos, mesmo estando disponíveis estruturas condicionais?

Exercício 18

1. Escreva um programa que, dadas duas datas, determine qual delas ocorreu cronologicamente antes em relação a outra. Cada data é composta de 3 números inteiros, um representando o ano, outro o mês e outro o dia.
2. Escreva um programa que, dado o comprimento de três segmentos de reta, determine se eles formam um triângulo e, caso formem, diga se o triângulo é equilátero, isósceles ou escaleno.

Aula 9

Não atirarei mais aviões em sala



Exercício 19 - Aguardando a pergunta fundamental

Pergunta fundamental

Uma pergunta fundamental é um expressão aritmética com dois operandos inteiros não negativos, cujas operações permitidas são soma ($+$), subtração($-$), multiplicação($*$) e divisão($/$) e que cujo resultado é 42.

Escreva um programa que só termine quando o usuário digitar uma pergunta fundamental (um operando, um operador aritmético e outro operando).

Exemplo de execução

```
Digite uma pergunta fundamental: 2 + 2
Essa não é uma pergunta fundamental, tente de novo: 1 * 44
Essa não é uma pergunta fundamental, tente de novo: 0 - 42
Essa não é uma pergunta fundamental, tente de novo: 84 / 2
Obrigado!
```

Exercício 20 - Progressão Aritmética

1. Faça um programa que lê um número n e imprima os valores da soma

$$\sum_{i=1}^{j} i$$

   para j de 1 até n.
2. Dê uma fórmula fechada para a soma acima.

Exercício 21 - Sequência de Fibonacci

Coelhos

Um homem quer começar a criar coelhos e quer saber quantos coelhos ele terá depois de n meses. Suponha que no primeiro mês ele compra uma parelha de coelhos jovens, que as parelhas amadurecem e reproduzem-se apenas após o segundo mês de vida sem problemas de cruzamento consanguíneo e em todos os meses. Também supunha que cada parelha fértil dá a luz a uma nova parelha e que os coelhos nunca morrem. Assim, no primeiro mês ele terá uma parelha, no segundo ainda 1, no terceiro 2, no quarto 3 e assim por diante. Essa é a sequência de Fibonacci.

Escreva um programa que leia um número n e devolva o número de parelhas após n meses.

Legibilidade de código

Para um código ser legível, ele deve ter as seguintes características.

Características

- Comentários que resumem o que faz um conjunto de comandos.
- Formatação bem definida (indentação, espaçamentos, nomes)
- Documentação adequada de cada programa ou algoritmo.

Um pouco de maniqueísmo - Comentários

Mal

```
#include <stdio.h>
#include <math.h>
int main() {
    // declara três variáveis
    float x, y, z;
    // lê um valor do teclado
    scanf("%f", &x);
    // lê outro valor
    scanf("%f", &y);
    // calcula o x ao quadrado
    // mais y ao quadrado
    z = x*x + y*y;
    // tira a raiz de z
    z = sqrt(z);
    // mostra o valor de z
    printf("%f\n", z);
}
```

Um pouco de maniqueísmo - Comentários

Mal

```
#include <stdio.h>
#include <math.h>
int main() {
    // declara três variáveis
    float x, y, z;
    // lê um valor do teclado
    scanf("%f", &x);
    // lê outro valor
    scanf("%f", &y);
    // calcula o x ao quadrado
    // mais y ao quadrado
    z = x*x + y*y;
    // tira a raiz de z
    z = sqrt(z);
    // mostra o valor de z
    printf("%f\n", z);
}
```

Bem

```
#include <stdio.h>
#include <math.h>
int main() {
    // catetos e hipotenusa
    float x, y, z;

    // lê valores dos catetos
    scanf("%f", &x);
    scanf("%f", &y);

    // aplica Pitágoras
    z = x*x + y*y;
    z = sqrt(z);

    // devolve a hipotenusa
    printf("%f\n", z);
}
```

Um pouco de maniqueísmo - Indentação

Mal

```
#include <stdio.h>
int main() {
int n;

printf("n: "); scanf("%d",&n);

if(n==1){printf("Unidade");}

else if (n>=0) {
if (n%3)
  printf("Deixa resto\n");
else   printf("Não deixa");

}else{printf("Negativo");}}
```

Um pouco de maniqueísmo - Indentação

Mal

```
#include <stdio.h>
int main() {
int n;

printf("n: "); scanf("%d",&n);

if(n==1){printf("Unidade");}

else if (n>=0) {
if (n%3)
  printf("Deixa resto\n");
else   printf("Não deixa");

}else{printf("Negativo");}}
```

Bem

```
#include <stdio.h>
int main() {
   int n;
   printf("n: ");
   scanf("%d",&n);

   if (n==1) {
      printf("Unidade");
   } else if (n>=0) {
      if (n%3)
         printf("Deixa resto");
      else
         printf("Não deixa");
   } else {
       printf("Negativo");
   }
}
```

Um pouco de maniqueísmo - Nome de variáveis

Mal

```
#include <stdio.h>
int main() {
  float n1, n2, n3, n4, aluno;
  int exercices;

  scanf("%d %f%f",&exercices,
      &n1, &n3);
  n2 = 4.0;
  n4 = 6.0;
  if (exercices > 7)
    n4 = 7.0;
  aluno = (n1*n2 + n3*n4)/10;
  printf("%f", aluno);
}
```

Um pouco de maniqueísmo - Nome de variáveis

Mal

```
#include <stdio.h>
int main() {
  float n1, n2, n3, n4, aluno;
  int exercices;

  scanf("%d %f%f",&exercices,
      &n1, &n3);
  n2 = 4.0;
  n4 = 6.0;
  if (exercices > 7)
    n4 = 7.0;
  aluno = (n1*n2 + n3*n4)/10;
  printf("%f", aluno);
}
```

Bem

```
#include <stdio.h>
int main() {
  float nota1, nota2;
  float peso1, peso2;
  float media;
  int exercicios;

  scanf("%d %f%f", &exercicios,
      &nota1, &nota2);
  peso1 = 4.0;
  peso2 = 6.0;
  if (exercicios > 7)
    peso2 = 7.0;
  media = (nota1*peso1 +
           nota2*peso2)/10;
  printf("%f", media);
}
```

Um pouco de maniqueísmo - Documentação

Mal

```
#include <stdio.h>
int main() {
  float r;
  float v;
  scanf("%f", &r);
  v = (4.0*3.1425*r*r*r)/3.0;
  printf("%f", v);
}
```

Um pouco de maniqueísmo - Documentação

Mal

```
#include <stdio.h>
int main() {
  float r;
  float v;
  scanf("%f", &r);
  v = (4.0*3.1425*r*r*r)/3.0;
  printf("%f", v);
}
```

O que esse programa faz???

Um pouco de maniqueísmo - Documentação

Mal

```
#include <stdio.h>
int main() {
  float r;
  float v;
  scanf("%f", &r);
  v = (4.0*3.1425*r*r*r)/3.0;
  printf("%f", v);
}
```

O que esse programa faz???
O valor de pi está errado, com quem eu falo???

Um pouco de maniqueísmo - Documentação

Mal

```
#include <stdio.h>
int main() {
  float r;
  float v;
  scanf("%f", &r);
  v = (4.0*3.1425*r*r*r)/3.0;
  printf("%f", v);
}
```

O que esse programa faz???
O valor de pi está errado, com quem eu falo???

Bem

```
#include <stdio.h>
/* DESCRIÇÃO: Lê o raio de
      uma esfera pelo teclado
      e imprime o seu volume.
   ENTRADA: o raio da esfera
   SAÍDA: o volume da esfera
   PRÉ-CONDIÇÃO: O raio deve
      ser maior que zero.
   AUTOR: Joãozinho */

int main() {
  float r;
  float v;
  scanf("%f", &r);
  v = (4.0*3.1415*r*r*r)/3.0;
  printf("%f", v);
}
```

Exercício 22 - Escolhendo um estilo de programação

Pesquisa

1. Não existe um jeito certo de escrever código: cada programador adota um estilo diferente, o importante é manter-se consistente. Pesquise sobre a importância da legibilidades de programas de computador: posição das chaves, indentação, uso de espaço, etc. Depois reflita e disserte brevemente sobre o seguinte tema: "Um código é escrito só uma vez, mas é lido diversas vezes".
2. Manter o código organizado pode ser trabalhoso, especialmente para programadores iniciantes. Mas felizmente o existem ferramentas que ajudam nessa tarefa. Pesquise e experimente ferramentas de formatação automática de códigos, como AStyle, indent etc.

Aula 10

Lidando com casos especiais

Olhando os casos especiais

Algumas vezes, um algoritmo "funciona" para vários casos, mas não para todos. Esses casos "problemáticos" geralmente são casos especias, que devem ser tratados individualmente.

Lidando com casos especiais

Olhando os casos especiais

Algumas vezes, um algoritmo "funciona" para vários casos, mas não para todos. Esses casos "problemáticos" geralmente são casos especias, que devem ser tratados individualmente.

Alguns exemplos:

- Os primeiros dois termos da sequência de Fibonacci.
- Primeiro elemento no algoritmo para descobrir o maior número.

Lidando com casos especiais

Olhando os casos especiais

Algumas vezes, um algoritmo "funciona" para vários casos, mas não para todos. Esses casos "problemáticos" geralmente são casos especias, que devem ser tratados individualmente.

Alguns exemplos:

- Os primeiros dois termos da sequência de Fibonacci.
- Primeiro elemento no algoritmo para descobrir o maior número.

Lição: Sempre lembrar dos possíveis casos especiais, como o primeiro ou o último elemento de uma sequência, lista, etc.

Exercício 23 - Lâmpadas

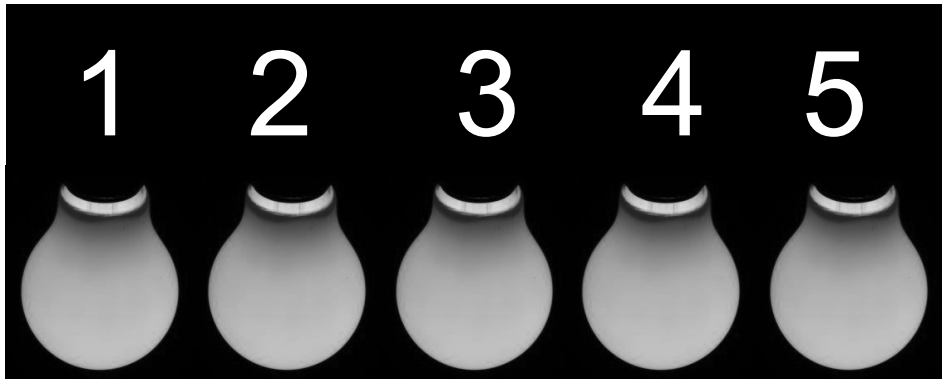
Problema

Existem n lâmpadas numeradas ordenadamente. Elas estão ligadas em um circuito de tal modo que toda vez que uma lâmpada é acesa ou apagada, outras lâmpadas podem são apagadas ou acendidas. Quando uma lâmpada é

- ACENDIDA, as posteriores invertem.
- APAGADA, as anteriores invertem.

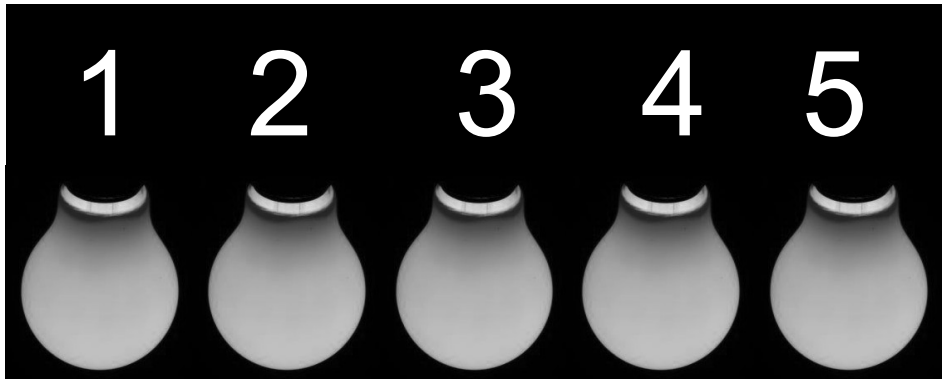
Exercício 23 - Lâmpadas

Inicialmente:

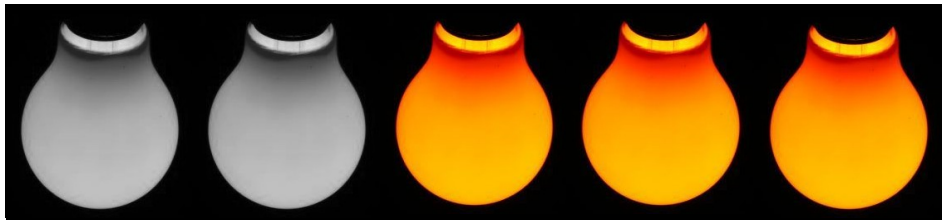


Exercício 23 - Lâmpadas

Inicialmente:

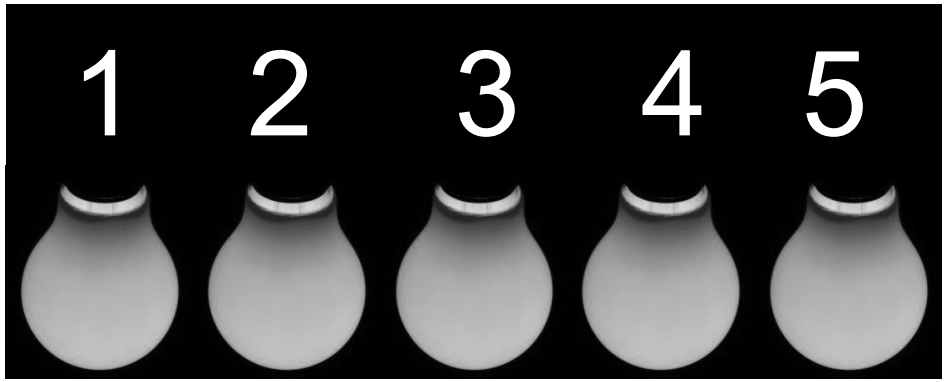


Lâmpada 3 acesa:

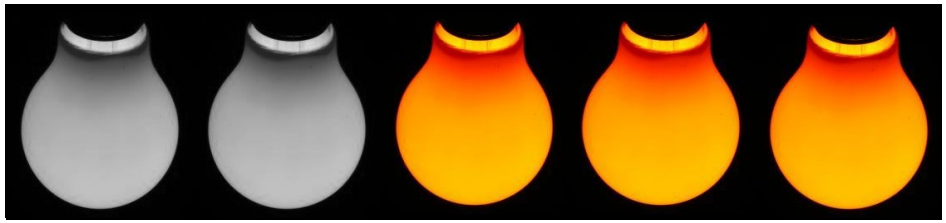


Exercício 23 - Lâmpadas

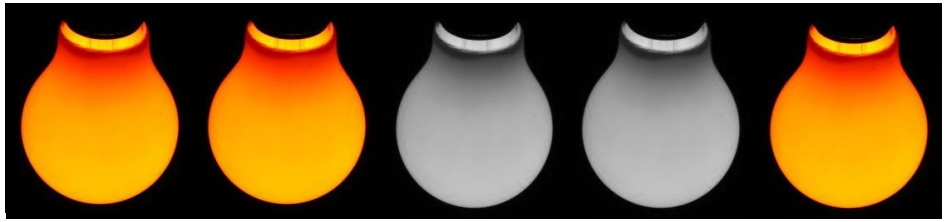
Inicialmente:



Lâmpada 3 acesa:



Lâmpada 4 apagada:



Exercício 23 - Lâmpadas

1. Escreva um programa que receba:
   - a quantidade de lâmpadas, n, de um circuito como no anterior,
   - um número de uma lâmpada específica, k, que deverá ficar acesa.

   O programa deve instruir o usuário a acender ou apagar lâmpadas para que, no final, apenas a lâmpada de número k fique acesa. Para acender uma lâmpada de número XX , imprima "Acender lampada XX", ou, para apagar uma lâmpada de número XX, imprima "Apagar lampada XX"

2. Modifique o programa anterior para receber
   - a quantidade de lâmpadas, n, de um circuito como no anterior,
   - a quantidade de lâmpadas que devem ficar acesas, m,
   - uma lista de m números de lâmpadas que devem ficar acesas.

   No final, apenas as lâmpadas listadas devem permanecer acesas.

3. Verifique se o seu programa está correto. Para isso, verifique se existe algum caso especial. Faça um teste de mesa se necessário.

Exercício 24 - Senoide

Escreva um programa que leia um número inteiro e desenhe uma senoide de "largura" *n* caracteres. Você pode usar a função `sin` para calcular o seno.

Entrada:
10

Saída:

```
           *
              *
                 *
                    *
                     *
                     *
                     *
                    *
                 *
              *
           *
        *
     *
  *
*
*
*
 *
     *
        *
           *
```

Exercício 25 - Analisando código

Descreva textualmente o que o programa a seguir faz. Faça um teste de mesa se necessário.

```
#include <stdio.h>

int main() {
    int i, j, l, m, n;
    scanf("%d", &n);
    m = 2*n + 1;
    for (i = 1; i <= n; i++) {
        j = 2*i - 1;
        for (l = (m - j) / 2; l > 0; l--) {
            printf(" ");
        }
        for (l = j; l > 0; l--) {
            printf("*");
        }
        printf("\n");
    }
}
```

Exercício 26 - Elemento mais próximo

Escreva um programa que leia um conjunto de 5 números reais, C. Em seguida, ele deverá ler outro número real x e imprimir o número de C com valor mais próximo de x.

Aula 11

Exercício 27 - Busca sequencial

Problema

Escreva um programa que leia um número inteiro n e depois leia uma sequência de n números fracionários. Após isso, o usuário irá digitar um número fracionário x. O seu programa deverá imprimir:

- Se o número x está na sequência, a ordem em que x foi inserido na sequência.
- Quantos números da sequência são maiores ou iguais a x e a média desses números.

Aula 12

Exercício 28 - Vetores e strings

1. Escreva um programa que lê duas strings do teclado com até 80 caracteres e informa se elas são iguais.
2. Escreva uma função que lê uma palavra do teclado e informa se ela é palíndroma. (Exemplos de palíndromos: ARARA, RADAR, REVIVER)
3. O histograma de um conjunto de dados é um gráfico da frequência com que cada valor aparece. Escreva um programa que leia um vetor de tamanho informado pelo usuário e com valores inteiros entre 1 e 9 e imprima o histograma na mesma forma que o exemplo: para um vetor de tamanho 20 com os valores 1, 1, 1, 1, 2, 2, 2, 3, 3, 4, 6, 7, 8, 8, 8, 9, 9, 9, 9, 9 deverá imprimir:

```
+---------+
|         *|
|*        *|
|**      **|
|***     **|
|**** ****|
+---------+
 123456789
```

Aula 13

Exercício 29 - Somando notas

1. Escreva uma função que leia do teclado o número de questões de uma prova e o valor de cada uma das questões. A função deve retornar a nota da prova.
2. Escreva uma função que receba como parâmetro um número de provas e leia do teclado o número de questões e os valores das questões de cada prova. A função deve retornar a média das provas.
3. Escreva um programa que leia do teclado um número de provas dadas em um semestre, o número de alunos matriculados e o número de questões e os valores das questões da cada prova de cada aluno. O programa deve imprimir a razão entre a média das notas dos alunos que tiraram pelo menos 5 e a média das notas dos alunos que tiraram abaixo de 5.

Aula 14

Exercício 30 - Lendo código

- Qual a saída do código abaixo quando o usuário digita o último dígito do seu RA.

```
int d;
int f(int a, int b) {
    int c = 3*a + 4*b;
    c = c % d;
    return c;
}
int main() {
    int a, b;
    d = 20;
    scanf("%d", &a);
    b = 3;
    printf("%d", f(b, a));
    return 0;
}
```

Exercício 31 - Composto de dois primos

Problema

Um número composto n é produto de dois números primos se existirem dois números primos p e q com os quais podemos escrever $n = p \times q$. Escreva um programa que lê um número n do teclado e decida se n é produto de dois números primos.

Vetores e funções

Declaração de vetores em função

- Não precisamos informar o tamanho máximo.

```
#define MAX_TAMANHO 100

void ler_vetor(int vetor[], int n) {
    int i;
    for (i = 0; i < n; i++) {
        scanf("%d", &vetor[i]);
    }
}
```

Matrizes e funções

Declaração de matrizes em função

- Mas para acessar uma posição de uma matriz em C, precisamos conhecer a largura.
- Então, devemos informar pelo menos a largura na declaração da matriz passada como parâmetro.

```
#define MAX_ALTURA  100
#define MAX_LARGURA 200

void ler_matriz(int matriz[][MAX_LARGURA], int m, int n) {
      for (i = 0; i < m; i++) {
          for (j = 0; j < n; j++) {
              scanf("%d", &matriz[i][j]);
          }
      }
}
```

Exemplo de matriz em função

```
#include <stdio.h>
void mostra_matriz(int matriz[][10], int num_linhas) {
    int i, j;
    for (i = 0; i < num_linhas; i++) {
        for (j = 0; j < 10; j++)
            printf("%d ", matriz[i][j]);
        printf("\n");
    }
}
int main() {
    int matriz[][10] =  { { 0,  1,  2,  3,  4,  5,  6,  7,  8,  9},
                          {10, 11, 12, 13, 14, 15, 16, 17, 18, 19},
                          {20, 21, 22, 23, 24, 25, 26, 27, 28, 29},
                          {30, 31, 32, 33, 34, 35, 36, 37, 38, 39},
                          {40, 41, 42, 43, 44, 45, 46, 47, 48, 49},
                          {50, 51, 52, 53, 54, 55, 56, 57, 58, 59}};

    mostra_matriz(matriz, 6);
    return 0;
}
```

Modificando vetores e matrizes

Cuidado!

Modificações realizadas em vetores por funções são visíveis fora da função!

```
#include <stdio.h>
void nao_modifica(int var) {
    var = 0;
}
void modifica(int matriz[][4], int num_linhas) {
    int i, j;
    for (i = 0; i < num_linhas; i++)
        for (j = 0; j < 4; j++)
            matriz[i][j] = 0;
}
int main() {
    int a = 999;
    int m[][4] =  { { 1, 2, 3, 4 }, { 5, 6, 7, 8 } };
    printf("%d %d %d ", a, m[0][1], m[1][2]);  // Imprime:  999 2 7
    nao_modifica(a);
    modifica(m, 1);
    printf("%d %d %d ", a, m[0][1], m[1][2]);  // Imprime:  999 0 7
    return 0;
}
```

Exercício 32 - Matrizes e funções

1. Escreva uma função que recebe dois números m e n, tais que m é no máximo 100 e n é no máximo 30 e depois leia do teclado uma matriz de dimensões $m \times n$. A função deverá armazenar em uma matriz, passada por parâmetro, a transposta da matriz lida.

Exercício 33 - Aleatórios

1. Escreva uma função que devolva um número aleatório no intervalo inteiro $[a, b]$ [1]. Você pode usar a função rand em stdlib.h que retorna um número aleatório entre 0 e RAND_MAX (presuma que RAND_MAX é múltiplo de $b - a$).
2. Escreva uma função que embaralhe um vetor de inteiros. Não utilize vetor auxiliar.

[1] Um número é uniformemente aleatório se a chance de escolher qualquer número em $[a, b]$ é a mesma.

Exercício 34 - Subvetores

1. Faça uma função que retorne a soma de um subvetor $A[i..j]$ de inteiros, isso é, retorne $A[i] + ... + A[j]$.
2. Faça uma função que receba um vetor A e devolva um vetor A' tal que $A'[i] = A[0] + A[1] + ... + A[i]$.
3. Faça uma função que receba um vetor A em devolva um vetor A' tal que $A'[i] = A[i - 10] + A[i - 9] + ... + A[i]$.
4. Refaça as questões anteriores, mas agora a função deverá transformar o vetor de entrada. Você não deve criar vetor auxiliar e o vetor passado por parâmetro de entrada é o mesmo para armazenar a saída.

Aula 16

Exercício 35 - Sistemas numéricos

1. Converta o número FADA em hexadecimal para um número binário.

Aula 17

Exercício 36 - Referências e reuso

1. Escreva uma função que troca os valores de duas variáveis inteiras.
2. Escreva uma função que decida se um número é produto de dois números ímpares. Se for, a função deverá retornar esses dois números em variáveis passadas por referência.
3. Escreva uma função que decida se um número é produto de quatro números ímpares. Se for, a função deverá retornar os números em variáveis passadas por referência. Tente utilizar a função anterior.

Aula 18

Exercício 37 - Modificando o comportamento de funções

1. Escreva uma função que devolve 1 se uma palavra termina com s, e 0 caso contrário.
2. Escreva uma função que conta o número de palavras que terminam com *s* em um texto.
3. Reescreva uma função que conta o número de palavras terminadas com *s* em um texto, mas use a primeira função como rotina. Você pode redefinir a assinatura da primeira função para torná-la mais conveniente.
4. Reescreva a função anterior para que ela conte as palavras de interesse. As palavras de interesse são identificadas por uma função indicadora passada por parâmetro (isso é, uma função que retorna 1 se for de interesse, e 0 caso contrário).

Aula 19

Exercício 38 - Estruturas

1. Escreva uma estrutura para representar um triângulo que contém os pontos correspondentes aos três vértices.
2. Escreva uma função que receba um triângulo e devolva a área.
3. Escreva uma função que desloque um triângulo à direita de uma distância d.
4. Escreva uma função que rotacione um triângulo em um grau dado. Você deve passar o triângulo por referência.
5. Escreva uma função que retorne o tipo de um triângulo (escaleno, isósceles ou equilátero). Use uma enumeração para representar o tipo.

Exercício 39 - Mais Estruturas

1. Defina uma estrutura para representar um polígono simples (no plano).
2. Escreva uma função que remova um vértice específico do polígono.
3. Escreva uma função para devolver a menor coordenada e a menor abscissa de vetor de pontos.
4. Escreva uma função para devolver a maior coordenada e a maior abscissa de vetor de pontos.
5. Escreva uma função para contruir um novo polígono que é o *bounding box* do polígono, isso é, o menor retângulo que contém o polígono.

Uniões de tipos

Union

Usamos `union` quando queremos tratar de maneira uniforme um determinado objeto, que pode ter tipos distintos a cada execução. O `union` serve então para lidar com essas situações. É claro que não precisamos armazenar espaço para cada parâmetro de cada tipo, mas precisamos guardar espaço apenas para o tipo de maior tamanho. Por outro lado, precisamos armazernar também o "tipo".

Exemplo de definição comum com union

```
typedef struct {
    union {
        tipo1 membro1;
        tipo2 membro2; ...
    };
    enum { TIPO1, TIPO2, ... } tipo;
} TipoGenerico;
```

Exercício 40 - Sistema algébrico

Em um sistema algébrico (Maple, Mathematica, Sage...), um número pode ser real, inteiro, complexo, racional, ou a solução de um sistema algébrico. Por outro lado, as operações aritiméticas e várias outras funções são aplicadas uniformemente sobre os números, independente do tipo.

1. Defina um tipo para representar exatamente um número. O número pode ser racional, inteiro ou raiz n-ésima de um número não-negativo.
2. Faça uma função que imprima um número na sua forma natural de representação.
3. Faça uma função que receba dois números e retorne o produto. O número resultante deve ser representado naturalmente de acordo com os tipos de entrada. Por exemplo, um produto de inteiro e racional é um racional.
4. Faça uma função que receba dois número e devolva verdadeiro se o primeiro for menor que o segundo.

Aula 20

Visibilidade

Visibilidade de funções e variáveis

Uma parte importante de programação é controlar a visibilidade, isso é, o que cada uma das funções "enxerga". Isso é particularmente importante quando trabalhamos com vários arquivos. Para isso, temos alguns modificadores, que são palavras chaves especiais como `extern` e `static`. Fazer declarações de funções e variáveis com esses modificadores pode implicar em maneiras diferentes da organização da memória.

Exercício 41 - Lib

lib.h

```
extern int a;
void f1();
int f2(int, int);
```

lib.c

```
extern int a;
void f1() {
    int b;
    static int c;
    int f3();
}
static int f3() {
    f2(1,2);
}
void f2(int a, int b) {
}
int f4() {
    int c;
    f3();
}
int a;
static int d;
```

Exercício 41 - Main

main.c

```
#include "lib.h"

int main(void) {
    extern int f4();
    int e;
    f4();
    return 0;
}
```

1. Para cada função, descreva as variáveis visíveis e funções visíveis. Diga quais variáveis são automáticas ou não.
2. Faça um desenho da organização da memória no momento em que a função `f2` estiver executando.

Aula 21

Exercício 42 - Ordenação

1. Reescreva a função `ordenar_selecao` para que ela não utilize as funções auxiliares (`menor_elemento` e `trocar`).
2. Reescreva a função `ordenar_insercao` para que ela não utilize funções auxiliares. Na implementação acima, nós selecionamos a posição de inserção do elemento primeiro e depois deslocamos um subvetor. Mas podemos fazer as duas coisas de uma só vez. Qual a vantagem?
3. Na função `ordenar_selecao`, é realmente necessária a última iteração do laço de repetição? Por quê? E para a função `ordenar_insercao`?

Exercício 43 - Mais Ordenação

1. Escreva uma função para ordenar um vetor de Pessoa em ordem decrescente de idade e, havendo empate, em ordem crescente de nome.
2. Suponha que existe um vetor de pessoas. Queremos criar um vetor de mulheres ordenado por idade em ordem decrescente e, havendo empate, em ordem crescente de nome. Não queremos modificar o vetor original mas também não queremos desperdiçar espaço nem duplicar informação. Para isso: (i) crie um vetor de ponteiros para pessoas; (ii) modifique a questão 1 para que ela receba um vetor de ponteiros. Implemente essa estratégia e explique suas vantagens e como ela funciona.
3. Ao invés de usar ponteiros, você poderia usar índices para o vetor original. Explique as vantagens e desvantagens.

Exercício 44 - Ordenação da bolha

https://www.youtube.com/watch?v=lyZQPjUT5B4

Exercício - Ordenação da bolha

Bubble-Sort

```
void ordenar_bolha(int vetor[], int n) {
    int i, mudou;
    do {
        mudou = 0;
        for (i = 1; i < n; i++) {
            if (vetor[i-1] > vetor[i]) {
                trocar(&vetor[i-1], &vetor[i]);
                mudou = 1;
            }
        }
    } while (mudou);
}
```

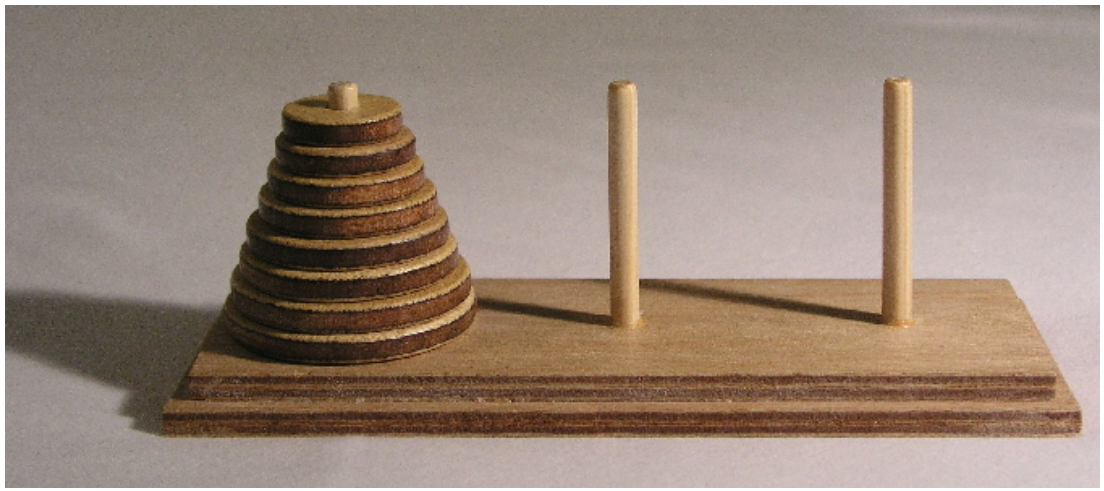
1. Explique o que faz e qual é a ideia do algoritmo.
2. Faça um teste de mesa para um vetor com elementos $(5, 4, 3, 2, 1)$ e para um vetor com elementos $(1, 4, 3, 2, 5)$. Conte as trocas.
3. Você consegue dizer por que o algoritmo tem esse nome? Por quê?

Aula 22

Exercício 45

1. Escreva uma função que calcula o número de triângulos virados de ponta-a-cabeça (os triângulos com uma ponta em baixo e duas em cima) em uma grade de triângulos de altura n.
2. Escreva uma função recursiva que calcule o n-ésimo valor da sequência de Fibonacci.

Exercício 46 - Torres de Hanói



Problema

A torre de Hanói é um brinquedo com três estacas A, B e C e discos de tamanhos diferentes. O objetivo é mover todos os discos da estaca A para a estaca C respeitando as seguintes regras:

- Apenas um disco pode ser movido de cada vez.
- Um disco só pode ser colocado sobre um disco maior.

Exercício 46 - Torres de Hanói

1. Entre no endereço `http://www6.ufrgs.br/psicoeduc/hanoi/` e resolva a torre de Hanói para o número de discos $n = 3$, $n = 4$ e $n = 5$.
2. Tente escrever uma programa em C que leia um número n do teclado e instrua o usuário a resolver a torre de Hanói com n discos.
   Se precisar de dicas, veja a próxima página.

Exercício 46 - Torres de Hanói

1. Entre no endereço `http://www6.ufrgs.br/psicoeduc/hanoi/` e resolva a torre de Hanói para o número de discos $n = 3$, $n = 4$ e $n = 5$.
2. Tente escrever uma programa em C que leia um número n do teclado e instrua o usuário a resolver a torre de Hanói com n discos.
   Responda às seguintes perguntas:
   1. É difícil resolver o problema quando temos apenas um ou dois discos? Esses casos são básicos?
   2. Se soubermos resolver o problema de mover os discos da estaca A para C, como podemos resolver o problema de mover os discos da estaca B para a C?
   3. Se tivermos dez discos na estaca A, mas os nove discos superiores estiverem colados, como podemos mover todos para a estaca C?

Exercício 47 - Sequência de Joãozinho

Um elemento na sequência de Fibonacci é dado pela soma dos dois anteriores e é um para os dois primeiros elementos. Joãozinho, aluno de algoritmos, definiu a sequência de Joãozinho da seguinte forma: um elemento é dado pela soma dos dois posteriores e é um para os dois primeiros.

Concorde ou discorde:

1. A sequência não está bem definida, já que não existe sequência de números que satisfaçam o que o Joãozinho deseja.
2. Não é possível construir uma função recursiva porque reduzimos o problema de tamanho n para dois problemas de tamanhos maiores $n+1$ e $n+2$.
3. Não pode haver uma base para a recursão porque o valor de cada elemento depende de um número infinito de outros elementos.

Justifique cada afirmação ou implemente uma função para calcular o n-ésimo número de Joãozinho.

Exercício 48 - Busca binária

Problema

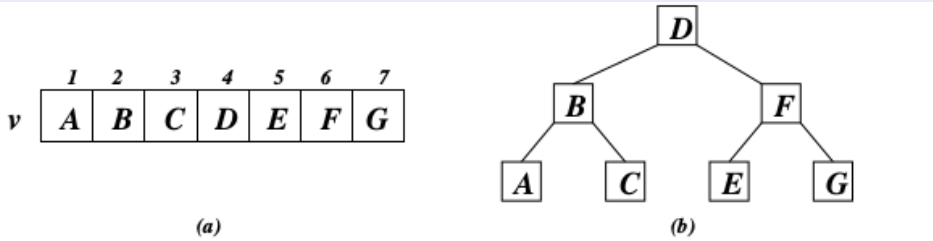
Escreva uma função que receba um vetor ordenado decrescentemente e um número x. A função deverá devolver o menor índice do vetor que contém x ou -1 se x não estiver no vetor.

- Implemente um algoritmo sequencial. No pior dos casos, quantas vezes acessamos o vetor?
- Implemente um algoritmo recursivo. Qual é o subproblema?
- Reimplemente o algoritmo recursivo anterior como um algoritmo iterativo. Isso é possível sempre que houver uma única chamada recursiva no final função recursiva.

Exercício 49 - Árvore genealógica

Problema [Notas de aula do prof. Flávio]

Um vetor tem $2^k - 1$ valores inteiros (figura (a)), onde k é um inteiro positivo, $k \geq 1$. Este vetor representa uma figura hierárquica (figura (b)) da seguinte maneira:



Você pode imaginar que este vetor está representando uma árvore genealógica de 3 níveis. Infelizmente, o usuário do programa que faz uso deste vetor necessita de algo mais amigável para ver esta estrutura. Faça uma rotina recursiva que dado este vetor v e o valor k, imprime as seguintes linhas:

```
            G-----------
      F-----------------
            E-----------
D-----------------------
            C-----------
      B-----------------
            A-----------
```

Dica: às vezes a função recursiva precisa resolver um problema um pouquinho mais geral que o original. E se o desenho tivesse que começar na coluna x?

Aula 24

Registros e alocação de memória

Às vezes temos que criar estruturas de dados complexas e precisamos alocar dois ou mais blocos de memórias relacionados. Nesse caso, é conveniente guardar somente uma referência para a variável principal. Por exemplo, para representar um polígono, só guardamos um ponteiro para a estrutura `poligono` abaixo.

```
enum cor {
    AZUL, AMARELO, VERMELHO, VERDE, PRETO
};
struct ponto {
    float x, y;
};
struct poligono {
    int n_pontos;
    struct ponto *pontos;
    enum cor fundo, borda;
};
```

Registros e alocação de memória: usando

Usando ponteiro para estrutura

```
int main() {
    struct poligono *pol;
```

Registros e alocação de memória: usando

Usando ponteiro para estrutura

```
int main() {
    struct poligono *pol;

    // aloca blocos de memória para polígono e pontos
    // e inicializa a partir do teclado
    pol = ler_poligono();
```

Registros e alocação de memória: usando

Usando ponteiro para estrutura

```
int main() {
    struct poligono *pol;

    // aloca blocos de memória para polígono e pontos
    // e inicializa a partir do teclado
    pol = ler_poligono();

    // move poligono 10 unidades à direita
    deslocar_poligono(pol, 10);
    imprimir_poligono(pol);
```

Registros e alocação de memória: usando

Usando ponteiro para estrutura

```
int main() {
    struct poligono *pol;

    // aloca blocos de memória para polígono e pontos
    // e inicializa a partir do teclado
    pol = ler_poligono();

    // move poligono 10 unidades à direita
    deslocar_poligono(pol, 10);
    imprimir_poligono(pol);

    // libera blocos de memória de polígono e pontos
    liberar_poligono(pol);
}
```

Registros e alocação de memória: construindo

Alocando memória

```
struct poligono *ler_poligono() {
    int i;
    float x, y;
    struct poligono *p;

    // aloca bloco para polígono
    p = malloc(sizeof(struct poligono));

    // aloca bloco para pontos
    scanf("%d", &(*p).n_pontos);
    (*p).pontos = malloc(sizeof(struct ponto) * (*p).n_pontos);

    // lê pontos do teclado
    for (i = 0; i < (*p).n_pontos; i++)
        scanf("%f%f", &(*p).pontos[i].x, &(*p).pontos[i].y);

    // devolve referência para objeto criado
    return p;
}
```

Registros e alocação de memória: construindo

Alocando memória: mais simples com operador de ponteiro

```
struct poligono *ler_poligono() {
    int i;
    float x, y;
    struct poligono *p;

    // aloca bloco para polígono
    p = malloc(sizeof(struct poligono));

    // aloca bloco para pontos
    scanf("%d",   &p->n_pontos);
    (*p).pontos = malloc(sizeof(struct ponto) *   p->n_pontos);

    // lê pontos do teclado
    for (i = 0; i <   p->n_pontos; i++)
        scanf("%f%f",   &p->pontos[i].x,   &p->pontos[i].y);

    // devolve referência para objeto criado
    return p;
}
```

Registros e alocação de memória: e desconstruindo

Sempre que criarmos uma função para alocar memória, também devemos criar uma função para desalocar a memória após o uso

Deslocando memória

```
void liberar_poligono(struct poligono *p) {

     // dealoca bloco para pontos
    free(p->pontos);

    // dealoca bloco para polígono
    free(p);

}
```

Exercício 50 - Um bocado de palavras

1. Implemente as funções `deslocar_poligono` e `imprimir_poligono`.
2. Escreva um programa que leia uma lista de nomes separados por espaço e imprima esses nomes em ordem alfabética. Implemente duas versões mais eficientes para os casos:
   - em que se presume que a quantidade de nomes é no máximo 10000;
   - que não se faz hipóteses sobre a quantidade de nomes.
3. Descreva as diferenças de implementação e uso de memória dos programas do item anterior.

Exercício 51 - Frutas

Problema

Um cadastro de frutas tem as seguintes informações: *nome e peso médio da uma unidade em gramas*. Cada fruta tem também uma lista de nutrientes. Um nutriente é composto pelo nome e um percentual; esse percentual representa a fração do total diário recomendado para uma pessoa que é suprida por uma grama de fruta.

1. Escreva um programa que leia os dados do cadastro e armazene-os na memória.
2. Escreva uma função que, dado o nome de um nutriente, encontre a fruta cadastrada que, com o menor número de unidades, poderia suprir completamente a necessidade diária desse nutriente.

Exercício 52 - Matriz dinâmica

Uma maneira de alocar uma matriz $m \times n$ de inteiros dinamicamente é:

```
int **mat;
int i;
// crie um vetor de m vetores
mat = malloc(sizeof(*int) * m);
// crie m vetores de n inteiros
for (i = 0; i < m; i++)
    mat[i] = malloc(sizeof(int) * n);
```

Outra maneira é criar uma matriz *linearizada*:

```
int *mat;
// crie um vetor de m*n inteiros
mat = malloc(sizeof(int) * m * n);
```

1. Escreva uma função que recebe uma matriz linearizada que representa uma imagem em preto e branco (0 ou 1) e "desenhe" duas faixas de 20 pixeis no topo e na base.
2. Suponha que você precise ler uma sequência de vetores n-dimensionais e ordená-los pelo módulo. Qual representação mais adequada? Implemente.