Computação Básica

Autor:
Guilherme Branco

21 de fevereiro de 2018

Sumário

Sobre este Documento

Este resumo tem o objetivo de auxiliar o leitor nos tópicos básicos de Computação. Este documento foi gerado em 21 de fevereiro de 2018 às 14:14:34. Para contribuições ou acompanhamento do material, acesse o repositório oficial: `aindan~aoexiste`.

O conteúdo não possuíra uma linguagem específica a ser seguida, as explicações serão em pseudo-código, porém algumas seções podem conter exemplos em C ou Python.

Utilize o apêndice contendo exercícios como complemento. Os exercícios geralmente ajudam a conceituar alguns pontos da matéria por outra perspectiva, mesclando algumas partes do conteúdo. Os exercícios podem ser feitos na linguagem de sua preferência.

Caso você encontre algum erro, seja conceitual, gramático ou ortográfico, envie o problema no repositório do projeto (por meio de *Issue*, ajudando a contribuir para melhorar este documento.

O que ainda não esta pronto será adicionado aos poucos conforme o tempo do escritor.

TODO: Conteúdo em si referente a computação básica: -Introdução (variáveis, entrada e saída) -Estruturas condicionais -Estruturas de repetição -Ponteiros - Basíco (sem estrutura de dados) -Vetores -Matrizes -Strings -Funções -Registros (typedef, struct, para python classes simples, dicionarios ou named tuples) -Arquivos -Recursividade Exercícios: -Ponteiros - Basíco (sem estrutura de dados) -Vetores -Matrizes -Strings -Funções -Registros (typedef, struct, para python classes simples, dicionarios ou named tuples) -Arquivos -Recursividade

Alguns exercícios até repetição já estão escritos, previsão do conteúdo destes até o fim do carnaval. Bons estudos!

Capítulo 1

Introdução

1.1 Conceitos básicos

Existem conceitos que permeiam todas as linguagens, tais como:

- Constantes e Variáveis
- Palavras reservadas
- Expressões Aritméticas
- Expressões Lógicas
- Comandos de Atribuição
- Entrada/Saída de dados

Esta seção tem como objetivo situar o leitor sobre estes conceitos.

1.1.1 Constantes e Variáveis

Estes dois conceitos tem em comum a alocação de uma região de memória no computador para guardar dados, como pode ser visto na Figura 1.1, porém para as constantes esse dado não pode ser modificado após sua criação, já para variáveis pode-se mudá-lo quantas vezes forem necessárias. Em ambos os casos necessita-se também do tipo de dado a ser guardado pelo computador, isto varia de linguagem, porém os tipos padrões constiuem os seguintes:

- Númericos
  - Inteiro - Ex.: 30, -15, 10, 0 ...
  - Ponto Flutuante (Reais) - Ex.: 2.3, -1.2, 15.3 ...

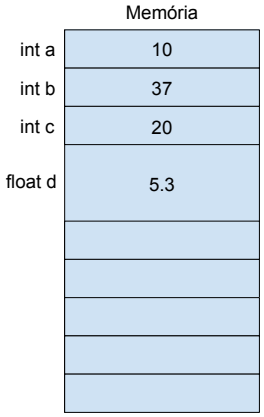

Figura 1.1: Alocação de memória

- Literais (Caracteres e Strings) - Ex.: ”teste”, ”t”, ”sera”...
- Lógicos (Booleano - Matemático George Boole) - Ex.: Verdadeiro e Falso

Em linguagens como C, C++ e Java há a necessidade de explicitar o tipo de dado ao se alocar a região de memória, em outras como Python ou PHP não há explicitamente o tipo ao se definir uma variável ou constante.
Exemplo em C e em seguida em Python:

```
//Comentario em C
int a = 1;
const int b = 2;
float c = 2.3;
const float d = 3.2;
char e = 'a';
const char f = 'b';

#Comentario em Python
a = 1
c = 2.3
e = 'a'
```

Note que em Python não há a palavra const, não sendo possível criar constantes de verdade.

1.1.2 Palavras reservadas

Existem palavras reservadas que diferem de linguagem para linguagem, elas servem para diversas coisas, como definir um tipo de dado, declarar uma constante, dar comandos condicionais (se senão), comandos de repetição entre outros.

1.1.3 Expressões Aritméticas

Servem para fazer os cálculos utilizados no programa, os operadores mais comuns são: +, - , / , *.
Exemplo: $a + b$

1.1.4 Expressões Lógicas

Fazer cálculos booleanos, que resultem em Verdadeiro ou Falso, os operadores mais comuns são: && (E) e —— (OU), existindo também os de maior que e menor que entre outros. E seus reultados seguem uma tabela verdade, como em 1.1 e 1.2. Exemplo:

Tabela 1.1: Tabela verdade para operador E

E	F	V
F	F	F
V	F	V

Tabela 1.2: Tabela verdade para operador OU

OU	F	V
F	F	V
V	V	V

1.1.5 Comando de Atribuição

Para atribuir um valor a uma variável tem-se o comando de atribuição, na maioria das linguagens se utiliza o comando '=' para tal.
Exemplo: $a = 2$
Porém, algumas linguagens podem utilizar a sequência ":=" entre outros.

1.1.6 Entrada e Saída

O computador esta configurado para ter uma saída padrão, o monitor, e uma entrada padrão, o teclado. Para acessar as estas funcionalidades, existem meios de requisitar a entrada de dados do usuário ou de mostrar algo na tela. Por didática será utilizado:

- Print(¡palavra¿) - Para escrever na tela
- Input(¡variavel¿) - Para ler do teclado o que foi digitado

Mas vale lembrar que nas linguagens isto pode não estar desta forma, variando o comando para cada linguagem. Em C, por exemplo, o comando parar ler do teclado constitui o "scanf()" já para mostrar na tela o "printf()".

1.2 Algoritmos

Agora que sabemos alguns comandos de linguagens de programação podemos construir algoritmos simples, para tal devemos definir o que será o algoritmo:

> DEFINIÇÃO · **Algoritmo** Um algoritmo constitui uma sequência de passos finita, com instruções bem definidas e não ambíguas.

Vamos montar agora um problema para resolver a transformação de graus Celsius para Fahrenheit:

$$T_F = T_C * \frac{9}{5} + 32 \tag{1.1}$$

```
real celsius, fahrenheit;
Input(celsius)
fahrenheit = celsius * (9/5) + 32
Escreva(fahrenheit)
```

Capítulo 2

Estruturas Condicionais

2.1 Condições

Em alguns algoritmos, como os vistos no capítulo anterior, a execução se dá de maneira pré-definida, porém estruturas condicionais servem para modificar o fluxo do algoritmo, como decidir se deve ser utilizado o bloco de instruções A ou B, vide Figura 2.1. Basica-

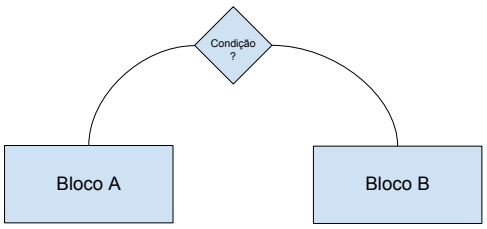

Figura 2.1: Fluxograma condicional

mente uma pergunta que deve ser respondida com os tipos lógicos Verdadeiro ou Falso, e pode ser utilizado em conjunto com os operadores lógicos, alguns presentes na tabela 2.1.

As estruturas condiconais podem ser simples ou compostas, a seguir têm-se o funcionamento das duas estruturas, começando pela simples.

```
Se <condicao> entao
    <comando1>
```

Tabela 2.1: Operadores lógicos

Operador	Definição
&&	E
——	OU
!	Negação
>	Maior que
>=	Maior ou igual que
<	Menor que
<=	Menor ou igual que

```
    <comando2>
    .
    .
    .
    <comandoN>
Fim-Se
```

Exemplo: Se um cliente queira que suas compras sejam embrulhadas para presente deverá pagar uma taxa adicional de R$3,50. Logo, para calcular o preço total deve-se saber se a mercadoria deve ser embrulhada.

```
valor: real
presente: caracter
Escreva("Informe o valor da mercadoria: ")
Leia(valor)
Escreva("Deseja embrulhar para presente?")
Leia(presente)
Se presente == 'S' entao
    valor = valor + 3.5
Fim-Se
Escreva("Total a pagar : " + valor)
```

Note que caso a informação guardada na variável presente for equivalente ao caracter 'S' então o valor da compra terá um acréscimo referente ao valor do embrulho.
Para as estruturas compostas temos a adição de duas novas palavras reservadas: o senão e o senão-se. Que servem para o caso contrário da condição, ou para condições extras, por exemplo:
Dados três números distintos, elabore um algoritmo que escreva o maior número digitado.

```
a,b,c: inteiro
Leia(a, b, c)
Se (A>B) entao
    Se (A>C) entao
```

```
            Escreva("O maior numero e A")
        Fim-Se
    Senao
        Se (B>C) entao
            Escreva("O maior numero e B")
        Senao
            Escreva("O maior numero e C")
        Fim-se
    Fim-Se
```

Note que o senão não precisa de condição, pois utiliza o contrário da condição do Se anterior. Podemos melhorar o algoritmo acima utilizando o senão se:

```
a,b,c: inteiro
Leia(a, b, c)
Se (A>B) E (A>C) entao
    Escreva("O maior numero e A")
Senao Se (B>C) entao
    Escreva("O maior numero e B")
Senao
    Escreva("O maior numero e C")
Fim-Se
```

Isto funciona pois utiliza-se o operador lógico E na condição, logo se A¿B e A¿C, o A é o maior. Porém se falhar na condição de A¿B, temos de testar B¿C para decidir qual dos outros será o maior. Note também que podemos fazer estrutras condicionais equivalentes utilizando o operador lógico da negação (!).

```
Se <condicao> entao
    <comandos1>
Senao
    <comandos2>
Fim-se

Se !<condicao> entao
    <comandos2>
Senao
    <comandos1>
Fim-se
```

As duas estruturas se tornam equivalentes, pois inverte-se a ordem em que os comandos serão dados, na primeira estrutura os comandos1 vão ser executados, já na segunda os comandos2.

Capítulo 3

Estruturas de repetição

Essas estruturas servem para, como o nome já implica, repetir um trecho do algoritmo, seja para garantir que a entrada de dados esteja correta, para refazer uma ação n vezes ou para manter o algoritmo em funcionamento por tempo indeterminado (como no caso de um jogo, onde o jogo só termina quando o usuário pede para sair). Existem três tipos de comandos para repetição, Para, Enquanto e Faça-Enquanto, que são equivalentes entre si, ou seja, pode ser utilizado o de sua preferência.

3.1 Contadas

A repetição contada pode ser utilizada quando sabe-se a quantidade de vezes que o laço deverá ocorrer. Esta quantidade pode ser informada pelo usuário ao se atribuir o valor para uma variável. O comando mais comum para a repetição contada constitui o "Para"

```
Para <variavel> = <valor_inicial> ate <condicao> em passos de <incremento>
    <comandos>
    .
    .
    .
Fim-Para
```

Para melhor entender o funcionamento da estrutura um pequeno exercício pode ser feito, deve-se determinar a média aritmética de n números. **Exemplo:**

```
n, i: inteiro
soma, nro: real
Leia(n)
Soma = 0
Para i=0 ate i<n em passos de i=i+1 faca
    Leia(nro)
    soma = soma + nro
Fim-Para
Escreva(soma/n)
```

Note que de $i = 0$ até $i < n$ existem n repetições de 1 em 1. Por exemplo, de 0 a 5 passamos pelos números, 0, 1, 2, 3 e 4, totalizando 5 números.

3.2 Repetições com teste no começo

O teste no começo serve para checar se o bloco de código deve ser executado ou se deve ser pulado, como mostra a Figura 3.1.

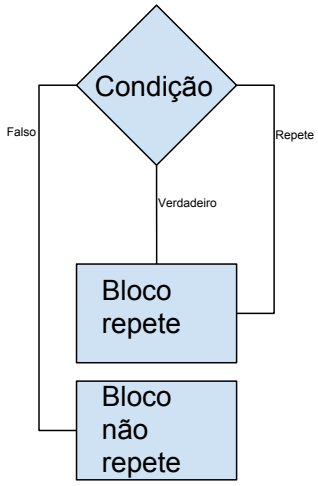

Figura 3.1: Fluxograma repetição com teste no começo

Pode ser utilizado com a estrutura anterior (Para ... até ... em passos de ...), porém mais comumente utilizado com uma nova estrutura, o Enquanto:
Estrutura da repetição com teste no começo

```
<variavel> = <valor_inicial>
```

```
Enquanto <condicao> faca
    <incremento>
    <comandos>
Fim-Enquanto
```

Note que nesta estrutura deve-se inicializar a variável fora do comando da repetição, assim como o incremento fica entre os comandos do bloco e não na repetição em si. Vamos montar o algoritmo para calcular $n!$ (fatorial).
Ideia do algoritmo:

```
Decalaracao de variaveis
Leia(n)
Calcula(n!)
Mostra(n!)
```

Já se sabe como declarar, ler e mostrar valores, o calculo de n! se dará por repetição:

```
fat = 1
i = 2
Enquanto (i <= n) faca
    fat = fat*i
    i = i+1
FimEnquanto
```

Para garantir o funcionamento pode-se utilizar o teste de mesa na Tabela 3.1, começando com $n = 5$, por exemplo:

Tabela 3.1: Teste de mesa para o fatorial de 5

n	i	(i<=n)	fat	saida
5	2	Sim	2	
5	3	Sim	6	
5	4	Sim	24	
5	5	Sim	120	
5	6	Nao	-	Fatorial de 5 é 120

3.3 Repetições com teste no fim

Esta última estrutura serve para checar se o bloco de código deve continuar a ser executado, portanto, sempre rodará pelo menos uma vez, já que o teste só aparece ao fim do bloco, exemplificado pela Figura 3.2.

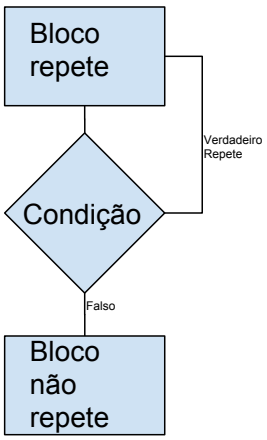

Figura 3.2: Fluxograma repetição com teste no fim

Sua estrutura segue abaixo:

```
<variavel> = <valor_inicial>
Faca
    <incremento>
    <comandos>
Enquanto <condicao>
```

Da mesma forma que o anterior o incremento deve ser colocado junto aos comandos do bloco. Para exemplo será feito um algoritmo que calcula a soma de valores até que um valor negativo seja lido.

```
valor: inteiro
soma: inteiro
soma = 0
Faca
    Escreva("Informe um numero")
    Leia(valor)
    soma = soma + valor
Enquanto (valor >= 0)
Escreva(soma)
```

Para desconsiderar o valor negativo da soma deve-se subtrai-lo ao final da repetição ou adicionar uma condição para não soma-lo durante os comandos da repetição.

```
valor: inteiro
```

```
soma: inteiro
soma = 0
Faca
    Escreva("Informe um numero")
    Leia(valor)
    soma = soma + valor
Enquanto (valor >= 0)
soma = soma - valor
Escreva(soma)
```

ou

```
valor: inteiro
soma: inteiro
soma = 0
Faca
    Escreva("Informe um numero")
    Leia(valor)
    Se (valor >= 0)
        soma = soma + valor
    Fim-Se
Enquanto (valor >= 0)
Escreva(soma)
```

Capítulo 4

Vetores

Com os conhecimentos até o momento caso seja necessário mais de uma variável que faça a mesma função deve-se declarar cada uma delas separadamente, porém há uma maneira de guardar estas variáveis em posições sequenciais de memória como mostra a Figura 4.1. Desta forma o acesso as variáveis em vetor, considerando o exemplo da figura

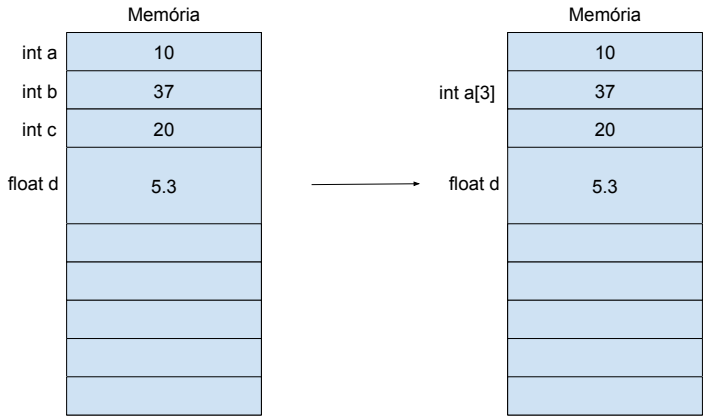

Figura 4.1: Alocação de memória para vetores

se dá por a[0] para o primeiro, a[1] para o segundo e a[2] para o terceiro, totalizando três variáveis inteiras. Considere agora que nestas três posições estejam o salário de

funcionários de uma empresa e que eles receberam um aumento de 10%, previamente precisaria-se de algo do tipo:

```
salario0, salario1, salario2, ..., salarioN: inteiro
Leia(salario0, salario1, salario2, ..., salarioN)
salario0 = salario0 * 1.1
salario1 = salario1 * 1.1
salario2 = salario2 * 1.1
.
.
.
salarioN = salarioN * 1.1
```

Note que precisaria-se de N linhas para N funcionários, portanto o código ficaria bem extenso, porém, pode-se utilizar uma estrutura de repetição junto de um vetor para facilitar o trabalho:

```
N: inteiro
N = 10
salario: vetor [N] de inteiro
#Le os salarios de cada funcionario
Para i = 0 ate N em passos de 1 faca
    Leia(Salario[i])
Fim-Para
#Calcula os 10% de aumento
Para i = 0 ate N em passos de 1 faca
    salario[i] = salario[i] * 1.1
Fim-Para
```

Pode-se reduzir o trabalho de digitar todas as linhas para cada salário consideravelmente, desta forma os vetores provêem uma maneira mais concisa de organizar o algoritmo e as variáveis.

Capítulo 5

Classificação e Pesquisa

Quase sempre que se utiliza um computador quer-se pesquisar alguma coisa, porém sem a Classificação do conteúdo a pesquisa se torna muito mais difícil. Da mesma forma que se vai a uma biblioteca para alugar um livro ou filme ou algo do tipo e espera-se que os livros estejam organizados, seja de maneira alfabética ou por genêro entre outros, o conteúdo dos computadores também necessita de ordenação, porém, muitas vezes os valores de um vetor não estarão organizados, portanto torna-se necessário classificar primeiro para depois pesquisar.

5.1 Classificação

Consiste em ordenar os elementos seguindo algum critério. Por exemplo, a ordenação alfabética para dados literais ou crescente/decrescente para dados númericos. Existem diversos algoritmos para executar a ordenação, os mais conhecidos estão entre: ShellSort, InsertionSort, BubbleSort, QuickSort, HeapSort. Neste capítulo focaremos somente no BubbleSort, devido a sua simplicidade, mas vale lembrar que este não será o mais eficiente da lista.

5.2 Pesquisa

Já quanto a pesquisa trataremos da pesquisa sequencial e a pesquisa binária.

Apêndice A

Exercícios

A.1 Introdução

(1) Crie um programa que leia e imprima o seu nome, o seu endereço, e a sua idade. O programa deve imprimir mensagens solicitando que o usuário informe os dados.

---

(2) Faça um programa que leia dois números inteiros, a e b, troque o conteúdo desses números e mostre os novos valores de a e b

---

(3) Crie um programa que leia dois números inteiros, calcule e mostre a média aritmética entre eles.

---

(4) Faça um programa que calcule o volume de um cilindro circular, dados os raio e altura do mesmo. (Obs: $V = \pi * r^2 * h$, onde $\pi = 3.14$, r consiste no raio e h a altura)

---

(5) Crie um programa que leia três coeficientes de uma equação do segundo grau $y = a * x^2 + b * x + c = 0$ e imprima o valor das raízes. Assumir que o valor do discriminante (delta) sempre será maior ou igual a zero.
Teste o programa com os seguintes conjuntos:

- $a = 1, b = -8, c = 15, resp : x_1 = 5 x_2 = 3$
- $a = 1, b = -8, c = 0, resp : x_1 = 8 x_2 = 0$
- $a = 2, b = -6, c = 4, resp : x_1 = 2 x_2 = 1$
- $a = 4, b = 8, c = 3, resp : x_1 = -0.5 x_2 = -1.5$

---

(6) Crie um programa para imprimir a hipotenusa de um triângulo retângulo, dados os valores dos catetos. Calcula-se a hipotenusa como a raiz quadrada da soma dos quadrados dos catetos.
Teste o programa com os seguintes conjuntos:

- $c_1 = 3 c_2 = 4, resp : h = 5$
- $c_1 = 12 c_2 = 9, resp : h = 15$

---

(7) Faça um programa que leia as três notas de um aluno, com seus respectivos pesos. Calcule e mostre a média final deste aluno.

A.2 Estruturas Condicionais

(1) Faça um algoritmo que receba um número inteiro e verifique se esse é par ou ímpar. Obs: Para decidir se um número é par ou impar basta checar se o resto de uma divisão por 2 é 0.

---

(2) Faça um algoritmo que leia dois números inteiros e imprima uma mensagem indicando se os dois números são iguais, ou imprima o maior entre eles se forem diferentes.

---

(3) Faça um algoritmo que solicite a idade de uma pessoa e informe:

- Se é menor de idade
- Se é maior de idade e tem menos de 65 anos
- Se tem pelo menos 65 anos

---

(4) F aça um algoritmo que leia três números inteiros diferentes e os imprima na tela em ordem **crescente**.

---

(5) Faça um algoritmo que leia a data de nascimento de uma pessoa, dia, mês e ano, todos inteiros. Verifique se a data pode existir e imprima uma mensagem ao usuário

indicando se a data está correta ou incorreta. Exemplo:
31/02/2003 - Fevereiro não pode ter 31 Obs: Desconsidere anos bissextos, fevereiro sempre terá 28 dias

---

(6) Crie um programa que leia três coeficientes de uma equação do segundo grau $y = a * x^2 + b * x + c = 0$ e imprima o valor das raízes. Calcule as raízes se o valor do discriminante (delta) for maior ou igual a zero. Se for menor que zero, apenas imprima uma mensagem adequada e interrompa o programa.
Teste o programa com os seguintes conjuntos:

- $a = 1, b = -8, c = 15, resp : x_1 = 5 x_2 = 3$
- $a = 1, b = -8, c = 0, resp : x_1 = 8 x_2 = 0$
- $a = 2, b = -6, c = 4, resp : x_1 = 2 x_2 = 1$
- $a = 4, b = 8, c = 3, resp : x_1 = -0.5 x_2 = -1.5$
- $a = 4, b = 2, c = 1, resp : discriminantemenorquezero$

---

(7) Dado um número inteiro, permitir ao usuário escolher dentre uma lista (menu) qual operação o mesmo deseja realizar:

- verificar se o número é par
- verificar se é ímpar
- verificar se é múltiplo de 3
- verificar se é múltiplo de 5
- verificar se é múltiplo de 7
- verificar TODOS os testes

---

(8) O Departamento do Meio Ambiente mantém três listas (A, B e C) de indústrias conhecidas por serem altamente poluentes da atmosfera. Os resultados de várias medidas são combinados para formar o que é chamado de "índice de poluição". Isto é controlado regularmente. Normalmente os valores caem entre 0.05 e 0.25. Se o índice de poluição atingir 0.25 a situação é de alerta; se atingir 0.3, as indústrias da lista A serão chamadas a suspender as operações até que os valores retornem ao normal. Se atingir 0.4, as indústrias B serão também notificadas. Se exceder 0.5, as três serão avisadas.

Escreva um programa para ler o índice de poluição e emitir um relatório notificando as indústrias, caso necessário. Deve constar no relatório a situação ocorrida (abaixo, normal ou alerta).
Teste o programa com os seguintes conjuntos:

- $i = 0.26, resp : Alerta$
- $i = 0.03, resp : Abaixodonormal$
- $i = 0.3, resp : ListaAsuspensa$
- $i = 0.06, resp : Normal$
- $i = 0.4, resp : ListasAeBsuspensas$
- $i = 0.35, resp : ListaAsuspensa$
- $i = 0.53, resp : ListasA, BeCsuspensas$

---

(9) Dado três valores de um suposto triângulo, decidir se esses valores podem ou não ser um triângulo, e cao seja decidir se é um triângulo retângulo ou não.
Dado que:

- Para ser triângulo a soma de dois lados sempre tem que ser maior que outro
- Para ser triângulo retângulo o maior lado ao quadrado tem que ser igual a soma dos quadrados dos outros dois

---

(10) Dado o ponto de origem (x,y), a altura A e largura L num epsaço bidimensional, podemos definir um retângulo. Receber um ponto (a,b) e decidir se ele esta:

- Dentro
- Fora
- Em alguma das linhas que definem o retângulo

A.3 Estruturas de repetição

A.3.1 Contada

(1) A conversão de graus Fahrenheit para Celsius é obtido por c=(5/9)*(f-32).
Faça um algoritmo que calcule e escreva uma tabela de garus Fahrenheit e graus Celsius, cujos graus variem de 50 a 65, de 1 em 1.

---

(2) Faça um algortimo que apresente a soma acumulada de todos os valores entre 1 e 100.

---

(3) Faça um algoritmo que leia dez números que representam notas de dez alunos, obtenha:

- A soma das notas
- A média das notas
- A maior nota
- A menor nota

Obs: Assuma que todas as notas são informadas corretamente no intervalo de 0 a 10.

---

(4) Faça um algoritmo que exiba a tabuada dos números de 10 a 20.

---

(5) Um funcionário de uma empresa recebe aumento salarial anualmente.
Sabe-se que:

- Esse funcionário foi contratado em 1995, com salário inicial de R$1000
- Em 1996 recebeu aumento de $1,5\%$
- A partir de 1997(inclusive), os aumentos sempre corresponderam ao dobro do percentual anterior

Faça um algoritmo que determine o salário atual desse funcionário.

---

(6) Faça um algoritmo que leia dez conjuntos de dois valores, o primeiro representa a matrícula do aluno e o segundo a sua altura em centímetros. Encontre o aluno mais alto e o mais baixo. Mostre a matrícula do aluno junto com suas alturas.

---

(7) Faça um algoritmo qu emostre todos os números pares entre 1 a 50.

---

(8) Faça um algoritmo que leia o número de andares de um prédio e, para cada andar, leia o número de pessoas que entraram e saíram do elevador. Se o número de pessoas, após a entrada e saída, for maior do que 15, deve ser mostrada a mensagem “Excesso

de Passageiros. Devem sair X"", onde X corresponde ao número que excedem. Após a entrada e saída no último andar, o algoritmo deve mostrar quantas pessoas permaneceram no elevador.

A.3.2 Teste no começo

(1) Faça um algoritmo para o seguinte problema: ler um conjunto de valores correspondentes aos pontos que alunos obtiveram em um teste. Quando o valor fornecido for um número negativo não existem mais pontos para serem lidos.

- Contar e escrever quantos alunos fizeram o teste.
- Contar e escrever quantos alunos tiveram nota baixa (Pontos ¡ 5)
- Contar e escrever quantos alunos tiveram nota alta (5 ¡= Pontos ¡= 10)

A.3.3 Teste no final

(D) e senvolva um algoritmo que calcule o valor da soma:
$S = \frac{1}{1} + \frac{3}{2} + \frac{5}{3} + \frac{7}{4} + ... + \frac{99}{50}$

A.3.4 Extras - Utilize a repetição que preferir ou faça uma vez com cada para aprender

(1) Imprima o valor de:
$\sum_{i=1}^{10} i$

Resposta: 55.

---

(2) Imprima o valor de $n!$, sendo n informado pelo usuário.
Teste para os seguintes casos:

- $n = 0, resp : 1$
- $n = 1, resp : 1$
- $n = 2, resp : 2$

- $n = 3, resp : 6$
- $n = 5, resp : 120$

---

(3) Imprima o valor da soma:
$\sum_{i=1}^{200} \frac{1}{i}$

---

(4) Partindo-se de um único casal de coelhos filhotes recém-nascidos, supondo que um casal de coelhos torne-se fértil após dois meses de vida e partir de então, produz um novo casal a cada mês e assumindo-se que os colhes nunca morrem, a quantidade de casais de coelhos após n meses se dá pelo n-ésimo termo da seguinte sequência:
$F_n = F_{n-2} + F_{n-1}, n >= 2$
$F_0 = 0 e F_1 = 1$
Esta sequencia se chama *Fibonacci*. Imprima a quantidade de casais de coelhos após n meses, onde n é dado pelo usuário.

---

(5) Calcule e imprima a média aritmética das idades de um grupo de pessoas fornecidas pelo usuário, cada idade sendo maior que zero. A entrada dos dados é finalizada quando o usuário digitar um valor igual a 0.
Teste o programa para $60, 20, 30, 70, 45, 25, 0$.

Resposta correta: 41.66

---

(6) Leia as seguintes informações para um número indeterminado de alunos: matrícula (inteiro), nome (string), e as notas de três provas (reais). A leitura dos dados deve terminar quando o usuário digitar 0 para a matrícula. Esse último valor não deve ser considerado para o cálculo. Imprima, para cada aluno: matrícula, nome e média ponderada das provas, onde a primeira prova tem peso 2, a segunda peso 4 e a terceira peso 4.

A.4 Vetores

(1) Escreva um algoritmo que receba dois vetores de 25 números inteiros cada, crie um novo vetor de 50 posições que seja o resultado da intercalação dos vetores de 25. Exemplo: vetor1 = [1,2,3,4,5], vetor2[6,7,8,9,10]. vetor_intercalado = [1,6,2,7,3,8,4,9,5,10]

---

(2) Faça um programa que leia um vetor com 6 valores inteiros e imprima a quantidade de números pares e a quantidade de números ímpares. Execute com os seguintes dados:

- 1 3 2 4 5 7, resp: pares = 2 ímpares = 4
- 1 2 3 4 5 6, resp: pares = 3 ímpares = 3

---

(3) Modifique o programa anterior para ler um número qualquer de valores entre 1 e 20, informado pelo usuário. Execute coms os seguintes dados:

- 6 1 3 2 4 5 7, resp: pares = 2 ímpares = 4
- 7 1 2 3 4 5 6 7, resp: pares = 3 ímpares = 4

---

(4) Modifique o programa anterior para receber vários conjuntos de dados, sendo que o usuário deve informar o final da leitua com um valor negativo no número de valores a serem lidos. Execute com os seguintes dados:

- 6 1 3 2 4 5 7, resp: pares = 2 ímpares = 4
- 7 1 2 3 4 5 6 7, resp: pares = 3 ímpares = 4
- 3 7 6 10, resp: pares = 2 ímpares = 1
- -1 resp: termina o programa

---

(5) Faça um programa que leia 9 valores inteiros, armazene num vetor e imprima qual destes números são menores que zero e quais as posições em que estão guardados. Execute com os seguintes dados:

- 1 -3 2 5 -4 -7 8 9 -11, resp: -3 [2]
  -4 [5]
  -7 [6]
  -11 [9]
- -1 30 20 -50 40 31 8 9 13, resp: -1 [1]
  -50 [4]

Obs: O índice para o computador começa em 0, porém para mostrar para pessoas lembre-se de acrescentar 1

---

(6) Faça um programa que leia a temperatura média de cada mês do ano e imprima a maior e menor temperaturas do ano e em que mês ocorreram estas temperaturas. Assuma que não exista temperaturas repetidas. Execute com os seguintes dados: 25 27 26 24 22 20 18 17 30 16 26 21, resp: menor = 17 mês = 8
maior = 30 mês = 9

---

(7) Faça um programa que leia um vetor e retire os valores iguais a zero. Este vetor será compactado e as posições finais devem ser preenchidas com o valor -1. O programa deve ao final escrever o vetor compactado. Obs: Não mostrar as posições do vetor que foram preenchidas com o valor -1. O vetor tem tamanho 20. Exemplo: 1 0 2 0 3 0 4 0 5 0 6 0 7 0 8 0 9 0 10 0
Vetor compactado: 1 2 3 4 5 6 7 8 9 10 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
Vetor mostrado: 1 2 3 4 5 6 7 8 9 10

---

(8) Faça um programa que lê os conteúdos de dois vetores X[10] e Y[10] e os mostra. Crie, a seguir, um vetor U que seja a união de X com Y, e um vetor I que seja a intersecção. Mostre o conteúdo de I e de U. Assuma que não existam elementos repetidos dentro de cada vetor.

---

(9) Faça um programa que leia as notas de duas provas, números reais, de 7 alunos e armazene estas em dois vetores. Crie um terceiro vetor que armazene a média aritmética das duas notas de cada aluno. Imprima as duas notas e as médias de cada aluno, onde as notas de um mesmo aluno devem ser impressas numa única linha. Execute com os seguintes dados: 7 8 5 6 4 3 9 10 7 7 8 6 9 7
Resp: aluno1 7 8 7.5
aluno2 5 6 5.5
aluno3 4 3 3.5
aluno4 9 10 9.5
aluno5 7 7 7.0
aluno6 8 6 7.0
aluno7 9 7 8.0
