Algoritmos e Estruturas de Dados

Desenvolvimento Web e Multimédia
Redes e Segurança Informática

Hugo Freitas - 2022

ESCOLA
TÉCNICA
SUPERIOR
PROFISSIONAL
IPCA

Apresentação

Apresentação

Docente

- Hugo Freitas
- hfreitas@ipca.pt
- Licenciado em Ciências da Computação na Universidade do Minho
- 8 anos de experiência como Software Engineer
- Atualmente, Software Engineer Team Lead na empresa Checkmarx
- Docente no IPCA desde 2019

Apresentação

Horário de atendimento

- Sábados das 9h às 11h com marcação prévia

Repositório de material pedagógico

- https://github.com/hffreitas/Algoritmos-Estruturas-Dados

Apresentação

Propósito

- Apresentar os conceitos fundamentais relativos a:
  - Algoritmia
  - Estruturas de dados
  - Programação estruturada
- Desenvolver a capacidade de compreender e analisar problemas
- Conceber e planear soluções estruturadas conducentes à sua resolução
- Utilizar a linguagem de programação C para implementar as soluções

Apresentação

Objetivos

- No final, os alunos deverão ser capazes de:
  - Analisar problemas
  - Propor uma solução na linguagem de programação C suportada por algoritmos representados em fluxogramas e pseudocódigo
  - Perceber o processo de codificação, compilação e execução
  - Utilizar estruturas condicionais e cíclicas, arrays, strings e estruturas

Apresentação

Metodologia de ensino

- Cerca de 90% de componente prática
  - Resolução de exercícios nas aulas
- Cerca de 10% dedicados à revisão de algoritmos e programação

Apresentação

Ferramentas

- TBD

Apresentação

Programa

- Algoritmos e resolução de problemas
  - Resolução de problemas
  - Aproximação descendente(*top-down approach*)
  - Noção formal de algoritmo
  - Características de um algoritmo

Apresentação
Programa

- Estruturas de Dados
  - Estruturas de dados primitivas
    - Boolean’s
    - Numéricos
    - Alfanuméricos
    - Representação dos dados
  - Estruturas de dados não primitivas
    - Arrays
    - Matrizes

Apresentação
Programa

- Notação algorítmica
  - Pseudocódigo
    - Instrução de atribuição
    - Leitura e escrita de dados
    - Estrutura condicional
    - Instruções de repetição
    - Operações e expressões aritméticas
    - Operadores e operações relacionadas
    - Operadores e operações lógicas
  - Algoritmos propostos
  - Diagramas de fluxo

Apresentação
Programa

- Introdução à linguagem C
  - Primeiros programas
  - Instrução de atribuição
  - Leitura e escrita de dados
  - Estrutura condicional
  - Instruções de repetição
  - Operações e expressões aritméticas
  - Operadores e operações relacionadas
  - Operadores e operações lógicas
  - Modularização (Funções, procedimentos)
  - Tipos de dados complexos
  - Gestão dinâmica de memória

Apresentação

Avaliação

- Os resultados da aprendizagem serão avaliados através duma componente teórica (prova escrita) e de uma componente prática (trabalho de grupo)
- A nota final é a média ponderada segundo a expressão seguinte:
  - NF = PE × 50% +TG × 50%
  - NF - nota final, PE - prova escrita, TG - trabalho de grupo, PA - participação nas aulas
- O TG é composto por, código em C, relatório e defesa presencial

Apresentação

Avaliação

- Aproveitamento à UC está sujeito à obtenção de **nota mínima de 10** valores à componente prática e de 10 valores à componente teórica
- **Não serão aceites entregas** ou melhorias do trabalho prático em época de exames
- Em época de exame apenas será avaliada a componente teórica, mantendo-se, para efeitos do cálculo da nota final, o valor obtido na componente prática durante a frequência da UC
- A não obtenção da nota mínima de 10 valores na componente prática **impossibilita** a realização da prova escrita (teste ou exame)
- **PLÁGIO SERÁ SEVERAMENTE PUNIDO.**

Apresentação

Datas

- Teste de avaliação
  - TBD
- Entrega e Defesa do TP
  - TBD
- Exame
  - TBD

Apresentação
Bibliografia

- **Principal**

–Pereira, Alexandre (2013), “C e Algoritmos”, 1a edição, edições sílabo

–Damas, Luís (1999), “Linguagem C”, 20.a edição, FCA – Editora de Informática Lda., série Tecnologias de Informação.

–Guerreiro, P. (2001), “Elementos de Programação com C”, 3.a edição, FCA – Editora de Informática Lda., série Tecnologias de Informação.

–Vasconcelos, J.B., Carvalho, J.V. (2005), “Algoritmia e Estruturas de Dados”, Centro Atlântico.

- **Complementar**

–Loudon, Kyle (1999), “Mastering Algorithms in C”, O’Reilly.

–Kernighan and Ritchie (1988), “The C Programming Language (ANSI C)”, 2.nd edition, Prentice Hall3.

Algoritmos e resolução de problemas

Algoritmos e resolução de problemas
O que é um programa?

- Um programa de computador
  - Algoritmo
  - Objetivo é resolver um problema
- Um algoritmo é representado por:
  - Expressões simbólicas
  - Permitem descrever e encontrar a solução
- Um algoritmo representa:
  - Sequência finita de instruções
    - Conduzem à resolução do problema
    - Cada uma pode ser executada mecanicamente numa quantidade finita de tempo

Algoritmos e resolução de problemas

O que são estruturas de dados?

- Representam entidades e objetos do mundo real
- Definem a parte estática dum algoritmo
- A manipulação das estruturas de dados definem a parte dinâmica dum algoritmo
- O conjunto constitui formalmente um algoritmo

Resolução de problemas

O que é?

- Processo de identificar e analisar um problema
- Desenvolvendo a sua solução de modo eficiente
- 4 fases:
  - Identificação e compreensão do problema
  - Conceptualização da solução
  - Definição do algoritmo para a resolução do problema
  - Implementação da solução através de um programa

Resolução de problemas

Escrever um algoritmo

- Simplificado através da decomposição e análise de subproblemas

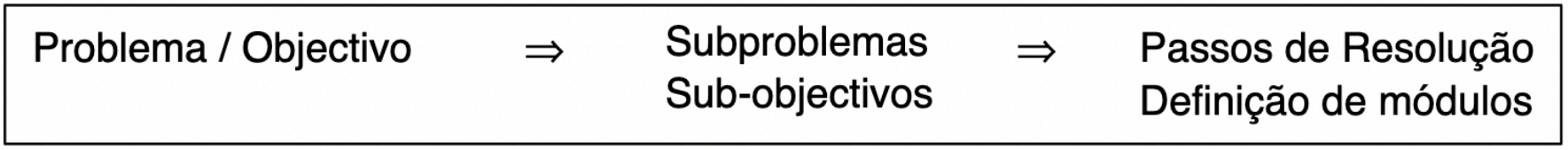


Figura 1: Abordagem para a resolução de problemas

Resolução de problemas

Aproximação descendente (top-down approach)

- Permite raciocinar e estruturar a solução dum problema em linguagem natural (ex: Português)
- Facilita o processo de compreensão do problema

Resolução de problemas

Mudança de lâmpada

Passo	Descrição
1	Selecione uma nova lâmpada
2	Remova a lâmpada fundida
3	Insira uma nova lâmpada

Resolução de problemas

Mudança de lâmpada

Passo	Descrição
1.1	Selecione uma nova lâmpada da mesma potência da fundida
2.1	Posicione a escada em baixo do candeeiro
2.2	Suba a escada até que possa atingir a lâmpada
2.3	Rode a lâmpada no sentido contrário aos ponteiros do relógio até que se solte
3.1	Coloque a nova lâmpada no orifício correspondente
3.2	Rode a lâmpada no sentido dos ponteiros do relógio até que fique presa
3.3	Desça da escada

Resolução de problemas

Mudança de lâmpada

- Definição mais precisa para o passo 1.1
  - Selecione uma lâmpada candidata à substituição
  - **Se** a lâmpada não é da mesma potência da antiga, então **repita** até encontrar uma correta:
    - Pouse a lâmpada selecionada
    - Selecione uma nova lâmpada

Resolução de problemas

Mudança de lâmpada

- Por exemplo, para os passos 2.2, 2.3, 3.2 e 3.3 poderiam também existir descrições mais precisas e detalhadas, do tipo: Repita… até…
- Aumento do detalhe do algoritmo pode continuar quase indefinidamente
- Grau de detalhe depende das necessidades do agente que vai executar o algoritmo.

Resolução de problemas

Lista Telefónica

Passo	Descrição
1	Encontre a página da lista que contém o último apelido do nome
2	Encontre na página determinada no passo 1, o nome procurado

Resolução de problemas

Lista Telefónica

Aumentando o detalhe, obtemos instruções elementares não ambíguas

Passo	Descrição
1.1	Coloque o marcador D (dedo) ao acaso na lista
1.2	Abra a lista
1.3	Último apelido está contido numa das páginas (esquerda ou direita)? Se sim, siga para o passo 2
1.4	Último apelido precede a página esquerda? Se sim, coloque o marcador atrás da página esquerda; se não, coloque o marcador à frente da página direita.
1.5	Vá para 1.2 (retome a sequência de instruções no passo 1.2)

Resolução de problemas

Lista Telefónica

Eliminando formulações mal definidas (ex: coloque o marcador atrás)

Passo	Descrição
1.1.1	**Torne A igual ao apelido do nome a selecionar (atribuição à variável A)**
1.1.2	**Escolha uma posição n ao acaso no interval [1,N]** **(n representa o número de páginas útil da lista)**
1.1.3	**Torne D igual a n (atribua à variável D o valor de n)**
1.2	**Abra a lista no local selecionado pelo marcador D**
1.3	**A** está contido numa das páginas (esquerda ou direita)? Se sim, siga para o passo 2
1.4	**A** precede o primeiro apelido da página esquerda? Se sim, faça **n** igual a **(n+1)/2** (atualização do valor de **n**); se não, faça **n** igual a **(n+n)/2**
1.5	**Vá para 1.2 (retome a sequência de instruções no passo 1.2)**

Um algoritmo é um processo discreto (sequência de acções indivisíveis) e determinístico (para cada passo da sequência e para cada conjunto válido de dados, correspondente uma e uma só acção) que termina quaisquer que sejam os dados iniciais (pertencentes a conjuntos pré-definidos).

Um algoritmo é constituído por um conjunto de expressões simbólicas que representam acções (escolher, atribuir, etc.), testes de condições (estruturas condicionais) e estruturas de controlo (ciclos e saltos na estrutura sequencial do algoritmo) de modo a especificar o problema e respetiva solução.

Noção formal de algoritmo

Resolução de problemas

Características de um algoritmo

Entradas

Quantidades inicialmente especificadas (por exemplo, através de instruções de leitura)

Saídas

Uma ou mais saídas (habitualmente por instruções de escrita)

Finitude

A execução deve terminar sempre num número finito de passos

Precisão

Todos os passos do algoritmo devem ter um significado preciso não ambíguo, especificando exactamente o que dever ser feito. Para evitar a ambiguidade das linguagens humanas, linguagens especiais foram criadas para exprimir algoritmos

Eficácia

Os passos devem conduzir à resolução do problema proposto. Devem ainda ser executáveis numa quantidade finita de tempo e com uma quantidade finita de esforço

Eficiência

Em muitos casos colocam-se questões de eficiência a um algoritmo

Resolução de problemas

Quantos alunos estão na sala de aula neste momento?

1. Professor contar os alunos
2. Contar os lugares vazios
3. Estimar baseado no tamanho total do local e multiplicar pelo número de pessoas por m2
4. Usar um torniquete
5. Contar o número de filas e, dado que todas tenham o mesmo número de alunos, então bastaria uma simples multiplicação
6. Cada aluno sentado no início de cada fila conta o número de alunos da sua fila, não esquecendo de se conta; Depois soma-se todas as contagens de todos os primeiros da fila
7. Todos os alunos levantam-se e atribuem o número 1; Aos pares, somam o número de cada um, um deles guarda a soma e o outro senta-se; repetir até ficar apenas 1 aluno

Quantos alunos estão na sala de aula neste momento?

Professor contar os alunos

Professor contar os alunos
Ter cuidado para não contar o mesmo aluno 2 vezes
Não esquecer de contar ninguém

Vantagens
Simples, fácil de executar
Solução perfeita para salas de 20 a 30 alunos
Não é necessário conhecimentos prévios
Não exige equipamentos adicionais

Desvantagens
Se o número de alunos for grande, poderá demorar demasiado tempo
Quanto maior o número de alunos, maior a possibilidade de haver erros de contagem

Quantos alunos estão na sala de aula neste momento?

Contar os lugares vazios

Contar os lugares vazios

Subtrair o número de lugares com o número de lugares vazios

Vantagens
Simples, fácil de executar
Boa solução para salas com ocupação quase total
Não exige equipamentos adicionais

Desvantagens
Necessidade de conhecer antecipadamente o número total de lugares na sala
Se a sala tiver com pouca ocupação o método anterior é mais eficaz
Não funciona para locais sem numero de lugares definido

Quantos alunos estão na sala de aula neste momento?

Estimar baseado no espaço

Estimar baseado no tamanho total do local e multiplicar pelo número de pessoas por metro quadrado

Vantagens
Solução elegante e eficaz
Boa solução para locais públicos sem lugares pré-definidos

Desvantagens
Necessidade de conhecer, antecipadamente, tamanho do local
Não é um método exato

Quantos alunos estão na sala de aula neste momento?

Usar um torniquete

Vantagens	Desvantagens
Eficaz e exata	Necessidade de ter torniquetes em todas as entradas
Boa solução para locais completamente fechados	Necessidade de bloquear todos os acessos

Quantos alunos estão na sala de aula neste momento?

Contar o número de filas

Contar o número de filas e, dado que todas tenham o mesmo número de alunos, então bastaria uma simples multiplicação

Vantagens
Eficaz e exata
Boa solução para paradas de exército

Desvantagens
Necesita ter exatamente o mesmo número de pessoas por fila

Quantos alunos estão na sala de aula neste momento?

Cada aluno situado no início duma fila, conta a sua fila

Cada aluno sentado no início de cada fila conta o número de alunos da sua fila, não esquecendo de se contar a si mesmo

Depois soma-se todas as contagens de todos os primeiros na fila

Vantagens
Simples
Solução escalável
Não é necessário ter conhecimentos prévios
Não exige equipamentos adicionais

Desvantagens
Se o número de alunos for grande, poderá demorar demasiado tempo
Quanto maior for o número de alunos, maior a possibilidade de ocorrerem erros

Quantos alunos estão na sala de aula neste momento?

Contagem em pares

Todos os alunos levantam-se e atribuem o número 1.

Em seguida, organizam-se em pares. Em cada par, primeiro é somado o número de cada um dos dois, um deles guarda o número e permanece de pé, o outro senta-se.

Os que ficaram de pé repetem o processo até que fique apenas um aluno de pé

Vantagens
Resultado exato
Solução escalável
Para um grupo de 1000 pessoas, termina em 10 passos. 1.000.000 pessoas, termina em 20 passos
Função logaritmica

Desvantagens
Exige organização

Mudar um pneu dum carro

Processo simples

1. Levantar o carro com o macaco
2. Tirar os parafusos da roda com o pneu furado
3. Tirar a roda do eixo
4. Colocar a roda com o pneu novo no eixo
5. Apertar os parafusos
6. Baixar o carro

Mudar os 4 pneus dum carro em menos de 8 segundos

Nelson Piquet

Nos anos 80, na fórmula 1, Nelson Piquet (tricampeão) imaginou que poderia ser campeão se usasse um composto de pneu mais macio e com isso ganhar preciosos segundos aos seus rivais.

Problema: necessidade de trocar os 4 pneus porque tinham um desgaste maior.

Nelson Piquet, após alguns cálculos, concluiu que se levasse menos de 8 segundos a trocar os 4 pneus, valeria a pena aplicar este método

Mudar os 4 pneus dum carro em menos de 8 segundos

Solução caseira

- Demora cerca de 20 minutos por pneu.
- Com prática poderia ser reduzido para 2/3 minutos por pneu.
- Bastante longe dos 8 segundos desejados.

Problema	**Solução**
Ter que trocar o macaco para cada roda demora muito tempo	**Usar macaco hidráulico e levantar o carro todo duma só vez**
Cada roda tem 4 parafusos que terão de ser desenroscados e enroscados	**Usar uma aparafusadora elétrica**

Mudar os 4 pneus dum carro em menos de 8 segundos

Problema	Solução
Mesmo com aparafusadora elétrica demora muito tempo aparafusar os 4 parafusos	**Usar roda com 1 parafuso**
Continua a demorar alguns minutos, pois uma pessoa tem de percorrer as 4 rodas	**Com 4 pessoas, cada uma troca uma roda, divide-se o tempo por 4 (menos de 1 minutos, mas superior a 8 seg**
Cada pessoa tem de desapertar, retirar a roda, colocar a nova roda e apertar	**Com 3 pessoas por roda, 1 desaperta, 1 retira o pneu e outra coloca o novo pneu (a que desapertou, pode**

Solução final: Com 3 pessoas por roda mais 2 para levantar o carro todo duma vez, conseguimos baixar o tempo para menos de 8 seg (hoje em dia muda-se os 4 pneus em menos de 3 seg)

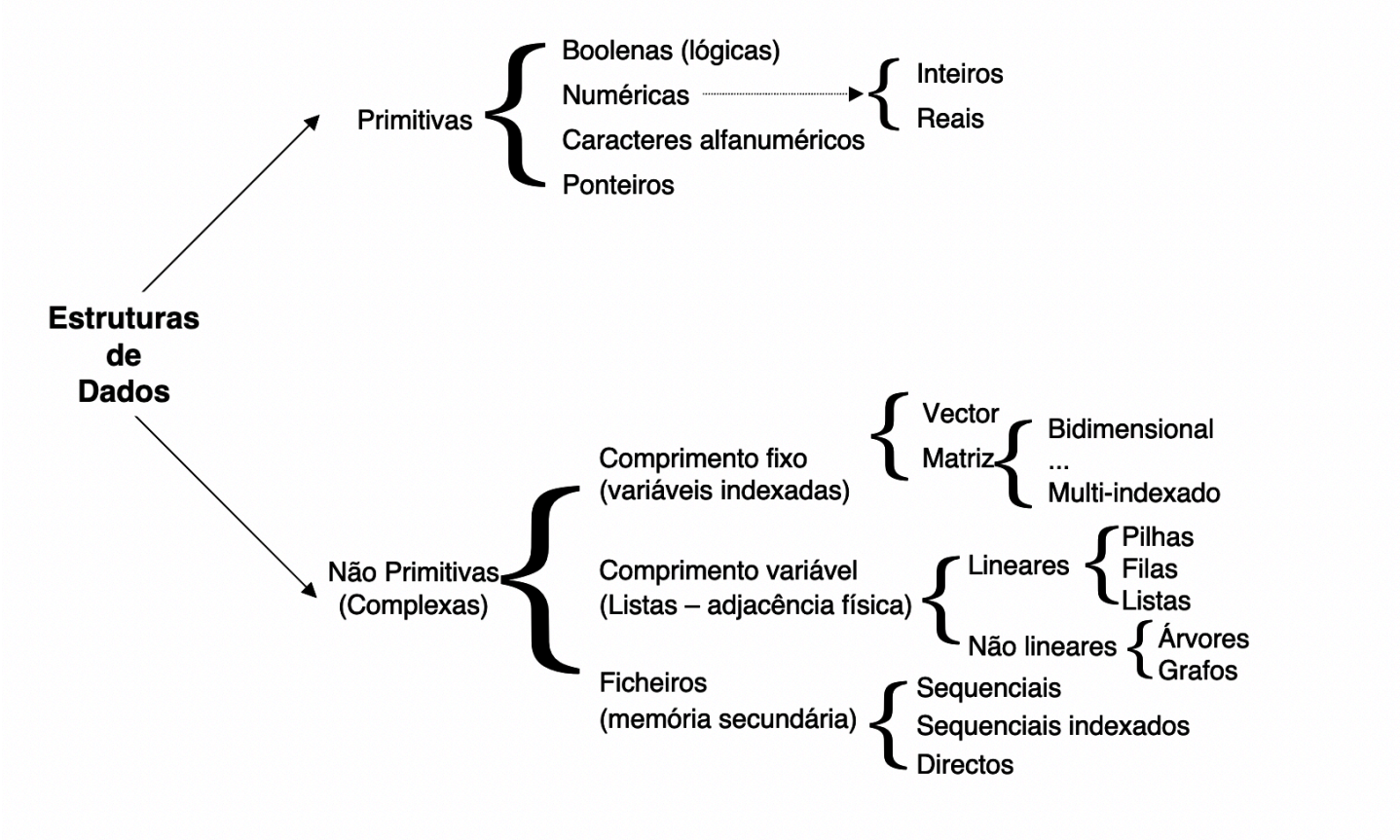
Estruturas de Dados

Estruturas de dados

O que são?

- A resolução de problemas através de algoritmos requer a representação de:
  - entidades e;
  - objetos reais
- As diferentes formas nos quais os itens de dados são logicamente relacionados definem diferentes estruturas de dados

Estruturas de dados



Estruturas de dados

Definição de estruturas

- A definição de estruturas de dados tem subjacente as operações a executar nos dados representados
- Diferentes operações poderão ser executadas:
  - Criar e eliminar estruturas de dados
  - Operações para inserir, alterar e eliminar elementos da estrutura de dados
  - Operações para aceder aos elementos da estrutura de dados

Estruturas de dados

Síntese

- Uma eficiente manipulação de dados envolve uma análise das seguintes questões
  A. Compreender a relação entre dados
  B. Decidir operações a executar nos dados logicamente relacionados
  C. Representar os elementos dos dados de modo a:
     1. Manter as relações lógicas entre os dados
     2. Executar de forma eficiente as operações nos dados
  D. Construir o algoritmo e escolher a linguagem de programação mais apropriada que permita de modo ‘natural’ e ‘expressivo’ representar as operações executadas nos dados.

Estruturas de dados primitivas

Tipo de dados boleado

- Permite representar dois e apenas dois estados: *verdadeiro* e *falso*
- Pode ser representado pelos dois estados existentes na codificação binária: *1-verdadeiro* e *0-falso*
- É aplicado em situações reais que unicamente denotam dois estados possíveis.

Estruturas de dados primitivas

Tipo de dados numérico

- Representativo dos valores numéricos no domínio dos números inteiros e reais
- 37 é um tipo numérico inteiro
- $\sqrt{2}$ é um tipo numérico real

Estruturas de dados primitivas

Tipo de dados alfanumérico

- Sistema de codificação designado ASCII (American Standard Code for Information Interchange)
- O conjunto de caracteres ASCII permite representar:
  - Alfabeto (minúsculas e maiúsculas)
  - Caracteres numéricos decimais {0,1,2,3,4,5,6,7,8,9}
  - Operadores e caracteres especiais
  - Caracteres de controlo - DEL- delete, CR - carriage return, HT - horizontal tab, etc

Estruturas de dados

Representação

- Os tipos de dados referidos são representados na memória principal de diferentes formas:
  - Dados simbólicos
  - Dados numéricos
  - Informação contextual

Estruturas de dados

Representação simbólica

- Letras - {a..z, A..Z}
- Dígitos decimais - {0..9}
- Sinais de operação - {+ - * /}
- Sinais de pontuação - {. , ; : ? !}
- Símbolos especiais - {“ $ # % ‘ |}

- Os símbolos são codificados em binário num código de comprimento n. Os códigos usuais têm 8 bits (n=8, 256 símbolos representáveis)

Estruturas de dados

Representação numérica

- Os dados representam quantidades numéricas no sistema de numeração binário
- 0101 1011 =
  $0 * 2^7 + 1 * 2^6 + 0 * 2^5 + 1 * 2^4 + 1 * 2^3 + 0 * 2^2 + 1 * 2^1 + 1 * 2^0$ = 64 + 16+8+2+1 = 91

Estruturas de dados

Informação contextual

- Conteúdo informativo (significado) de um dado, depende do contexto.
- Por exemplo 0100 0001 pode representar:
  - 65 - operação binária
  - 41 - operação utilizando BCD (binary code decimal)
  - A - símbolo ASCII
  - MOV A,B - instrução do microprocessador INTEL 8080

Estruturas de dados

Tipos de dados vs Variáveis vs Constantes

- $X \leftarrow 47$
  - Significa que à *variável X* foi atribuído o valor 47. Esta operação de atribuição tem por pressuposto o facto de X ser uma variável do *tipo inteiro*
- $CAR \leftarrow$ 'G'
  - Significa que à *variável CAR* foi atribuído o valor 'G'. Esta operação de atribuição tem por pressuposto o facto que CAR é uma variável do *tipo alfanumérico*. Por convenção, os valores alfanuméricos são representados entre plicas.

Estruturas de dados

Nomes de variáveis

- Uma variável representa um determinado valor e o seu nome é escolhido de forma a reflectir o valor que representa
  - A variável MAX pode representar o valor máximo de um conjunto de valores numéricos.
- Por convenção, o nome duma variável **não deve** contar espaços, **não deve** iniciar por números e **não deve** conter determinados caracteres especiais.
- O nome duma variável **deve** iniciar-se sempre por uma letra seguido dum conjunto de caracteres incluindo letras, números e alguns caracteres especiais como, por exemplo, o ‘_’.

Estruturas de dados

Variáveis válidas

- NUM
- N_ALUNO
- NOMEALUNO
- NOME_ALUNO
- X1
- Y2

Estruturas de dados

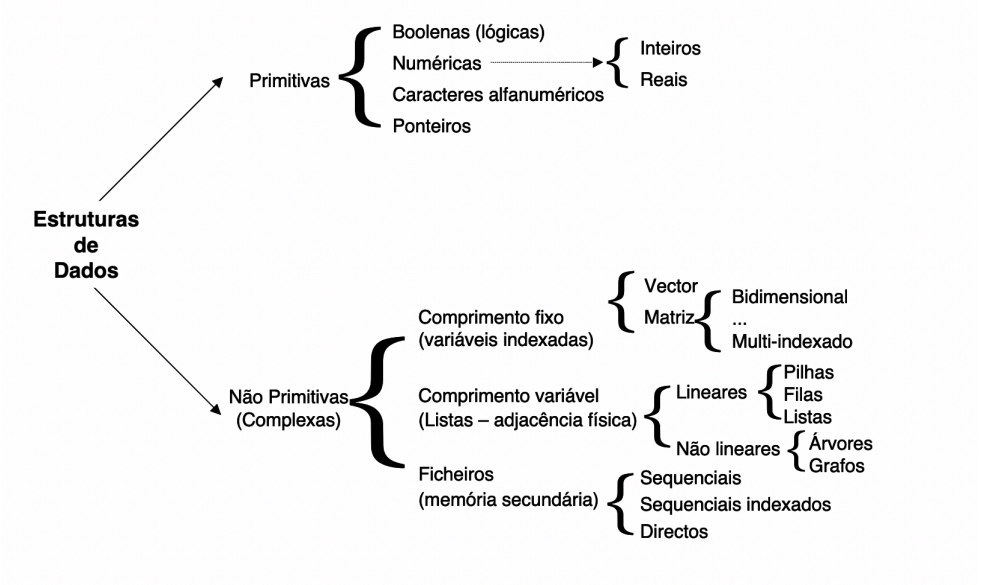
Variáveis inválidas

- 7NUM
- Y-Z
- T/4
- U*VAR
- DUAS-PAL
- DUAS PALAVRAS

Estruturas de dados não primitivas

O que são

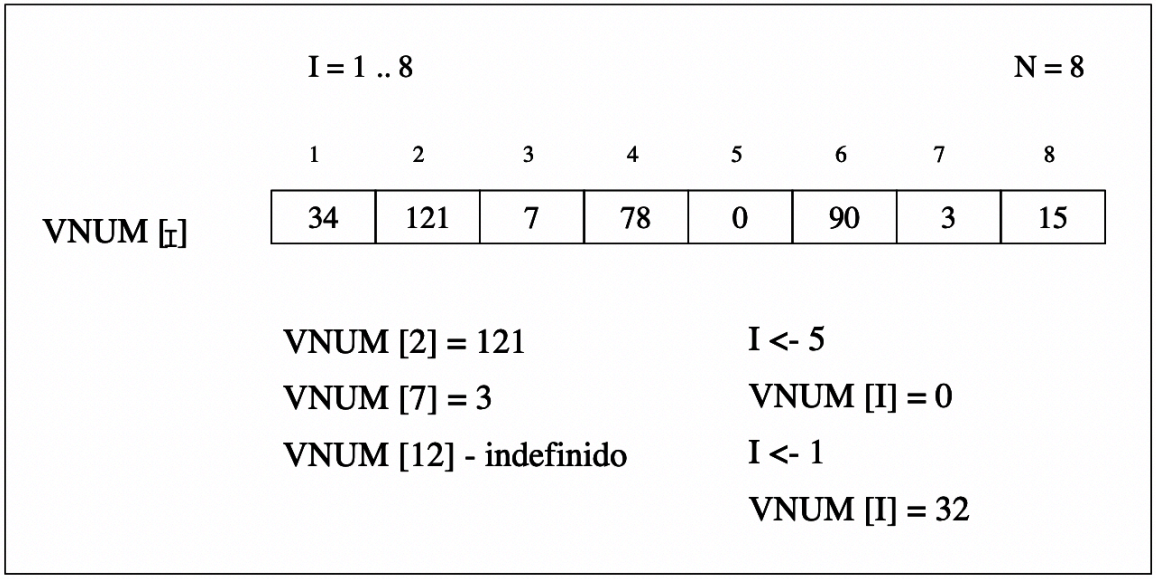
- São definidas através de conjuntos de estruturas de dados primitivas.
- No contexto desta disciplina, serão estudadas as estruturas de dados de comprimento fixo e algumas exemplos de ficheiros de dados e suas aplicações



Estruturas de dados não primitivas

Vetores

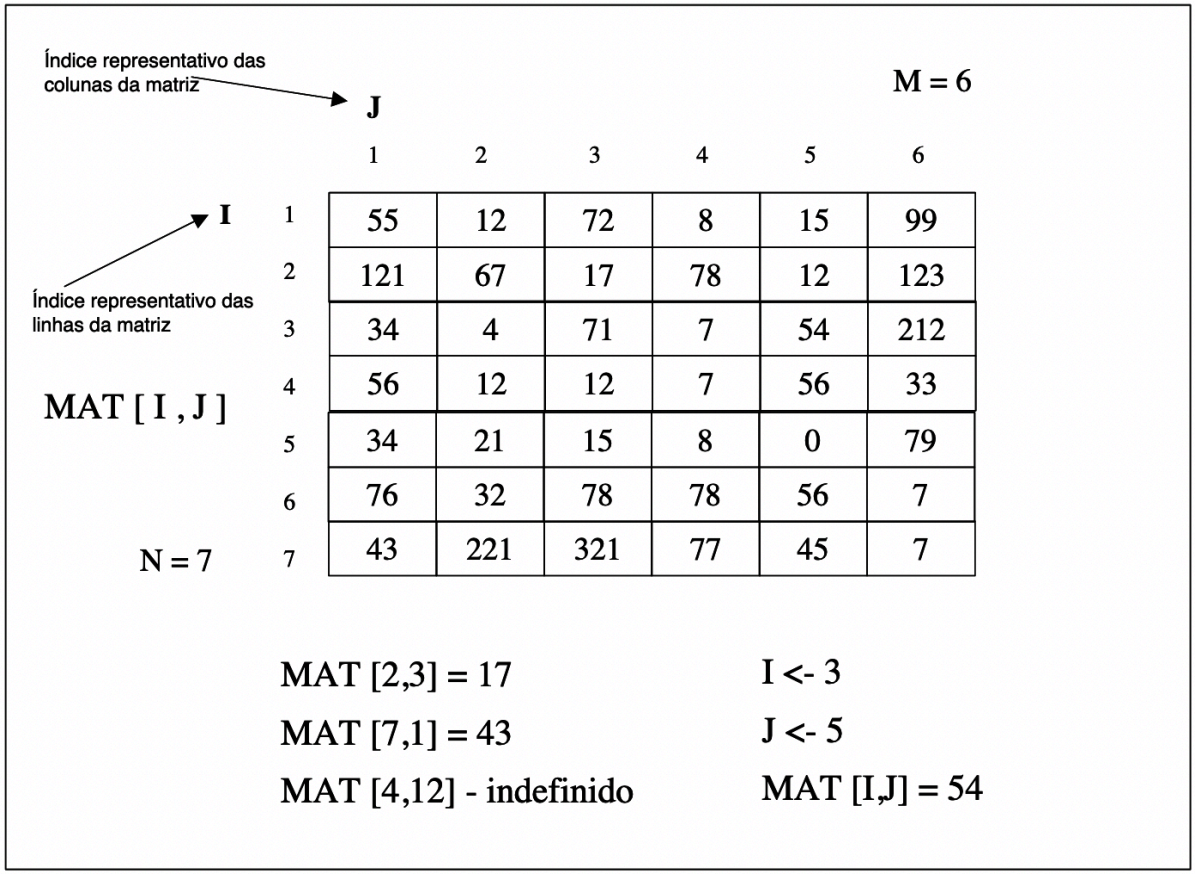
- Elementos dum vetor (array) são representados através de conjuntos de variáveis de um determinado tipo de dados.



Estruturas de dados não primitivas

Matrizes

- Uma matriz pode ser interpretada como um vetor bidimensional



Notação algorítmica

Notação algorítmica

Prefácio

- Na fase de desenvolvimento dum algoritmo podemos utilizar:
  - Linguagem auxiliar (pseudo-código)

    Linguagem de representação de alto nível que pode ser traduzida em qualquer linguagem de programação
  - Diagrama de Fluxo (fluxograma)

    Notação gráfica
  - Linguagens de programação (ex: C, C++, Java, C#, etc.)

Notação algorítmica

Pseudo-código

- É uma notação algorítmica muito próxima da linguagem natural.
- Um algoritmo é identificado por um nome que deverá ser elucidativo do problema em causa.
- Por convenção, o nome do algoritmo é seguido duma breve descrição das tarefas executadas pelo algoritmo.
- Um algoritmo é construído através de uma sequência de passos numerados.
    - Cada passo deverá conter um comentário explicativo da tarefa a executar no contexto global do algoritmo

Pseudo-código

Regras

1. Nome do algoritmo em letras maiúsculas
2. Breve comentário das tarefas a desenvolver pelo algoritmo
3. Conjunto de passos numerados e comentados
4. Estruturas de dados, funções e sub-rotinas em letra maiúscula
5. Estruturas de controlo: primeira letra em maiúscula.
6. O algoritmo termina com a instrução *Exit* (fim lógico). O símbolo representa o fim físico do algoritmo.

Pseudo-código

Factorial dum número inteiro

Algoritmo FACTORIAL. Este algoritmo calcula o factorial de um número inteiro.

1. [Leitura do número]

   Read(N)

2. [Caso particular de N = 0 ou N = 1]

   If N=0 Or N=1

   Then FACT ← 1

3. [Outros números]

   Else FACT← N

      NUM ← N -1

      Do For I = NUM To 1 Step -1

         FACT ← FACT * I

4. [Impressão do resultado (N!)]

   Print('O Factorial de ',N,' é igual a ', FACT)

5. [Termina]

   Exit

Pseudo-código

Instrução de atribuição

- É representa através do símbolo ←
  - Que é colocado à direita da variável que recebe o valor da atribuição (ex: FACT ← 1).
  - Ter em atenção a diferença entre o sinal de atribuição ← e o sinal = que é utilizado como operador relacional.
- Exemplos:

  VAR ← 12/7

  FACT ← FACT * NUM

  N ← MOD(NUM1, NUM2)

Pseudo-código
Leitura e escrita de dados

- É possível obter (ou ler) valores de variáveis, assim como escrever (ou imprimir) os valores dessas variáveis através de:
  - instruções de leitura (entrada de dados) e;
  - instruções de escrita (saída de dados).
- Leitura de dados

  Sintaxe:

  Read(<nome da variável>) ex: Read (N_ALUNO)
- Saída de dados

  Sintaxe:

  Write (<nome variável>) ex: Write (N_ALUNO) ou

  Write (<‘texto’>) ex: Write(‘so texto…’) ou

  Write (<nome da variável, ‘texto’) ex: Write (‘O número de alunos é igual a ‘, N_ALUNO)

  Write (‘texto’, <nome da variável 1>, ‘texto’, <nome da variável 2, ‘texto,…) ex: Write (‘O número ’,N_ALUNO,‘ refere-se ao n.º de alunos da disciplina de programação’)

Pseudo-código

Estrutura condicional

- ***If Statement - If … Then … Else***
- Esta declaração representa um teste de condição lógica (se … então … senão).
- É executado um conjunto de instruções consoante a condição especificada for verdadeira ou falsa.
- Pode ter uma das seguintes formas:

If Condition
Then _____
_____

If Condition
Then _____
_____
Else _____
_____

Pseudo-código

Estrutura condicional

- A seguir à clausula **Then** um conjunto de instruções é executado no caso da condição ser verdadeira.

- Caso contrário (**Else**) será executado o conjunto de instruções a seguir à clausula *Else*

```
If N > 10 Then
    X ← (X + 15) /2
Print(X)
```

```
If N > 10 Then
    X ← (X + 15) /2
    Print(X)
Else
    X ← X * 5
Print(X)
```

Pseudo-código

Instruções de repetição

- Existem 3 tipos de instruções que permitem controlar iterações ou ciclos de processamento.
  - Do while <logical condition>
  - Repeat until <logical condition>
  - Do for INDEX = <numerical sequence>

Pseudo-código

Do…While

- É utilizada quando é necessário repetir um conjunto de passos em função de uma determinada expressão lógica.
- Estes passos são repetidos enquanto a expressão lógica for verdadeira.

```
Do While <logical condition>
    _____________
    _____________
    …
```

Sintaxe

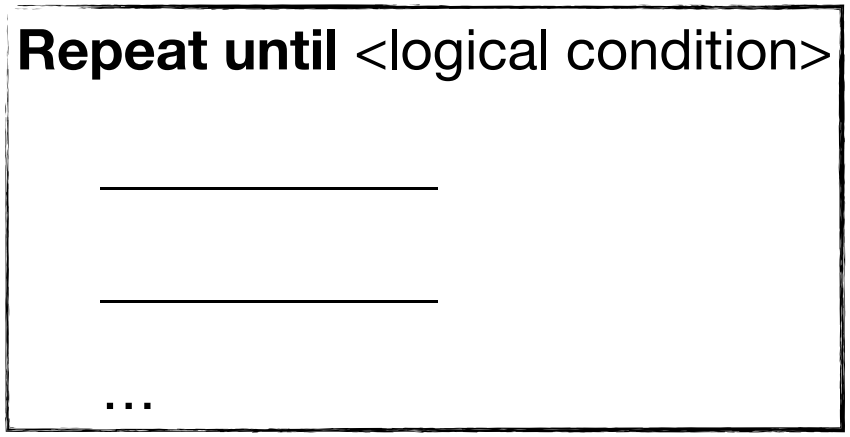
```
Read (Z)
X← 15
Do While Z >= 135
    X← X-5
    Z ← (X*4) /3
Print(X,Z)
```

Exemplo

Pseudo-código

Repeat Until

- É utilizada quando é necessário repetir um conjunto de passos em função duma determinada expressão lógica.
- Os referidos passos são repetidos até que a expressão lógica se torne verdadeira.



Sintaxe

```
X← 1
Repeat until K > 75
    Write( NOME_ALUNO[K])
    K ← K+1
```

Exemplo

Pseudo-código

Do For

- É utilizada quando é necessário repetir um conjunto de passos um determinado número de vezes.

```
Do For INDEX = N1 to N2 Step P
    _____________
    _____________
    ...
```

Sintaxe

```
X← 10
Do For I = 1 to 70
    X ← X + 5
Print (X)
```

Exemplo

```
X← 0
Do For J = 10 to 100 Step 10
    X ← X + 25
Print (X)
```

Exemplo

- INDEX indica o índice que é incrementado em cada ciclo de processamento.
- N1 e N2 referem dois números inteiros representativos do intervalo inferior e superior da sequência de números a executar.
- A cláusula ***Step*** determina o incremento na sequência numérica. Por defeito o ***Step*** *tem valor 1.*

Pseudo-código

Operações e expressões aritméticas

- A notação algorítmica inclui operações e funções matemáticas que permitem efetuar variados cálculos aritméticos.
- Dois tipos de valores numéricos: Inteiros e Reais
- Regras de precedência:
    1. Parêntesis - ()
    2. Exponenciação - (↑)
    3. Multiplicação - (*)
    4. Divisão - (/)
    5. Adição - (+)
    6. Subtração - (-)

Pseudo-código

Operações e expressões aritméticas

- Determinadas funções matemáticas podem ser utilizadas na definição de expressões computacionais:
  - mod(M,N) - função que retorna o resto da divisão de M por N
  - int(NUM) - função que retorna a parte inteira de um número real.
  - sqr(NUM) - função que retorna a raiz quadrada de um número inteiro ou real.

$X \leftarrow (A+B) \uparrow 3 - (A-A/3) * B$

$N \leftarrow mod(20,6)$

$RNUM \leftarrow sqr(NUM)$

Pseudo-código

Operadores e operações relacionais

- Os operadores matemáticos relacionais ($=$, $<$, $>$, $\leq$, $\geq$, $\neq$) têm um conjunto de símbolos correspondentes a nível computacional, respetivamente,
  $=$, $<$, $>$, $<=$, $>=$, $<>$ ou $!=$
- As expressões lógicas representam relações entre valores do mesmo tipo de dados.
- O resultado da avaliação de uma expressão lógica pode ter um de dois valores possíveis: **true** (verdadeiro) ou **false** (falso).

A ← 20

A <= 10/3 +5

A primeira relação tem valor falso e a segunda relação tem valor verdadeiro

A <> A +10

Pseudo-código

Operadores e operações lógicas

- A notação algorítmica inclui os seguinte operadores lógicos:

Operador	Notação
Negação	not
Conjunção	and
Disjunção	or

Precedência	Operador
1	Parêntesis
2	Aritméticos
3	Relacional
4	Lógico

Considere que NUM tem o valor 1

1. (NUM < 2) and (NUM < 0)
2. (NUM < 2) or (NUM < 0)
3. not(NUM < 2)

As expressões 1,2 e 3 têm valores falso, verdadeiro e falso, respetivamente

Pseudo-código

Exercício: Máximo divisor comum

Pretende-se implementar em pseudo-código o máximo divisor comum entre dois números.

Recorde-se que o máximo divisor comum entre dois números é o maior número que é divisível por ambos.

Pseudo-código

Exercício: Máximo divisor comum

```
Read (M,N)
If M < N Then
    MIN ← M
Else
    MIN ← N

MDC ← MIN
FLAG ← 0
Do While FLAG = 0
    R1 ← mod(M, MDC)
    R2 ← mod(N, MDC)
    If R1 = 0 and R2 = 0 Then
        FLAG ← 1
    Else
        MDC ← MDC -1
Write('O máximo divisor comum de ', M, 'e',N, 'é igual a ', MDC)
Exit
```