

# Estrutura de Dados I

## Introdução

**Prof. Fabio Fernando Kobs, Dr.**

DSI – CEPLAN – UDESC

# Objetivos da aula

- Dados x Estrutura de Dados
- Tipos de Dados Primitivos
- Variáveis e alocação de memória
- Ponteiros ou Referências
- Tipos Abstratos de Dados

# Dados

- **O que são Dados?**

- Dados são observações documentadas sobre um fato real. Qualquer característica que possa ser registrada.
- Dado é o menor nível de abstração da informação, sendo o fato em sua forma primária.
- Dados são sequências de *bits* na memória.

Exemplo: Nome, idade, posição geográfica, ...

# Estrutura de Dados

- Uma estrutura de dados consiste no modo de armazenamento e organização de dados em um computador. Os dados não consistem somente no local no qual são armazenados, mas nas relações entre eles.
- A organização dos dados e a escolha de sua categoria irá acarretar na performance ou na velocidade que o computador irá demorar para encontrar os dados.
- Logo, as **estruturas de dados** são responsáveis por organizar e administrar os dados, e disponibilizam **operações** para o usuário **manipular** os dados.

# Estrutura de Dados

- Quando os dados estão organizados e dispostos de forma coerente, levando em consideração o objetivo no qual foram destinados, caracterizam uma forma, uma estrutura de dados. A organização e os métodos para manipular essa estrutura de dados conferem a ela singularidade.
- Exemplos: Classe, Lista, Vetores, Árvores, etc.
- Algumas estruturas de dados somente adquirem **significado** quando associadas a um conjunto de **operações**, que visam, de um modo geral, manipulá-las (algoritmos).

# Tipos de dados

- Os dados utilizados em um programa de computador, ou armazenados em um banco de dados, são classificados de acordo com os seus tipos a partir de suas particularidades e limitações.
- Os **tipos de dados** podem ser classificados em:
  - Primitivo ou básico ou estruturado
  - Abstrato

# Dados primitivos ou básicos

- Associados a um método de interpretar um padrão de *bits*.
- Usado no contexto de uma linguagem de programação, que fornece meios para manipulá-lo.
- Um tipo de dado primitivo é autossuficiente, por exemplo, a idade de uma pessoa.
- Cada linguagem de programação, oferece um conjunto de tipos primitivos.
- Esses tipos, em geral, manipulam:
  - Números inteiros ou reais
  - Caracteres
  - Valores lógicos
  - Ponteiros ou referências

# Dados primitivos ou básicos

- Tabela de tipos em C

Tipo de dado	Significado	Tamanho (bytes)
char	Caractere	1
int	Inteiro	2
long int	Inteiro longo	4
float	Flutuante (real)	4
double	Flutuante duplo	8
long double	Flutuante duplo longo	10

Os tipos ainda podem ser modificados pelo **unsigned**, fazendo com que assumam apenas valores positivos



# “Dados primitivos” em Python

- Ao manipular dados com caracteres – o tipo mais primitivo, isto é, a maneira mais abstrata para representar caracteres – estar-se-á utilizando uma variável definida como sendo do tipo *str*.
- O fato de o Python não trabalhar com tipos primitivos diretamente, deve-se ao fato de que em Python, tudo são objetos. Dessa forma, o que chama-se de primitivo é, em Python, representado como uma informação será, ou seja, um objeto propriamente dito.
- A seguir, tem-se a lista dos principais tipos *builtins*\* da linguagem Python:
  - int - para números inteiros (positivos e negativos)
  - bool - subtipo do tipo int, e os valores se comportam como os valores 0 e 1, ou seja, quando convertidos para string são “False” ou “True”
  - float - para números com ponto flutuante
  - str - para conjunto de caracteres
  - list - para agrupar um conjunto de elementos

\* Função disponível automaticamente em qualquer módulo ou contexto.

# Dados primitivos ou básicos ou estruturados

- Permitem a realização de agregação de mais de um valor em uma variável, havendo uma relação estrutural entre os elementos. A matriz utilizada na matemática representa esse tipo de dado, no qual cada elemento tem uma relação entre os seus vizinhos.
- Exemplo: objeto *list*.

# Variáveis

- Na programação, é um elemento capaz de reter e apresentar um valor ou expressão.
- Existem apenas em “**tempo de execução**”, quando ocupam uma área de memória.
- Em “**tempo de desenvolvimento**” são associadas a **nomes** ou **identificadores**, facilitando a referência a um endereço de memória.
- Durante a execução de um programa, a variável é associada a um endereço de memória.
- Essa região da memória pode receber um valor, que ficará retido na memória até que outro valor o sobreponha.

# Variáveis

- As variáveis podem ser de dois tipos:
  - **Globais:**
    - Visíveis a todo o programa
    - Presentes em toda a execução do programa
  - **Locais:**
    - Visibilidade limitada ao seu escopo
    - Ao sair do escopo, a área de memória é liberada

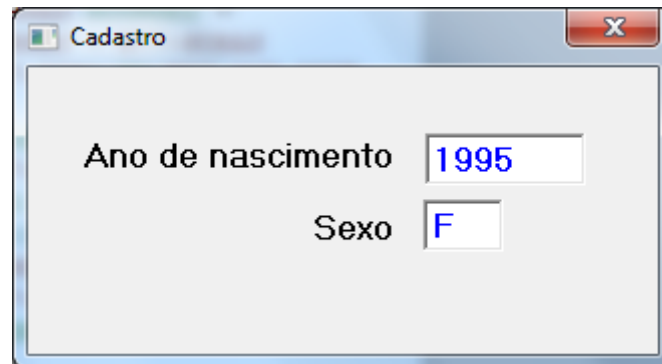
# Variáveis

- Exemplo

Identificador	Tipo	Tamanho	Endereço	Conteúdo
ano_nasc	int	4	1000	1995
sexo	char	1	1004	'F'

Tempo de desenvolvimento

Tempo de execução



A screenshot of a graphical user interface window titled 'Cadastro'. The window has a standard title bar with a minimize button, a maximize button, and a close button (marked with an 'X'). The main area of the window is light gray and contains two input fields. The first field is labeled 'Ano de nascimento' and contains the value '1995'. The second field is labeled 'Sexo' and contains the value 'F'. Both labels and values are in a blue font.

# Alocação de memória

- **Estática**

- A alocação de memória ocorre no momento da execução do programa. Neste momento, o sistema verifica as variáveis criadas no início do programa e reserva um espaço de memória para cada uma delas de acordo com os seus respectivos tipos. Logo, a memória somente será liberada quando o programa for finalizado.

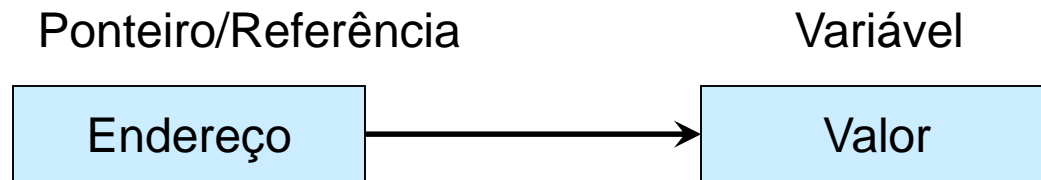
- **Dinâmica**

- A alocação de memória ocorre durante a execução do programa, sendo realizada juntamente às operações durante a sua execução, ou seja, a memória só é requisitada no momento em que o objeto é necessário; a alocação se dá pelo uso de referências. A memória será liberada após o uso.

# Ponteiros ou Referências

Tipo especial de variável que contém um endereço de memória.

Pode apontar para uma variável, função ou uma nova área alocada na memória.



# Tipos Abstratos de Dados - TAD

- Em várias situações, um tipo de dado primitivo isolado não é suficiente para registrar uma informação. Exemplo:
  - Um programa que precise manipular frações (a fração em si, não o número real resultante da fração), precisará de dois inteiros: um para o numerador e outro para o denominador, para conseguir representá-la.
  - Além disso, essa fração poderá sofrer operações matemáticas como soma, subtração, etc. que exigem um tratamento especial.
- Nessas situações, entram em cena os tipos abstratos de dados, ou seja, tipos de dados que não foram originalmente previstos nas linguagens de programação e que geralmente são necessários para o desenvolvimento de aplicações complexas.



# Tipos Abstratos de Dados - TAD

- Um TAD é uma **coleção** bem definida de **dados** a serem armazenados e um grupo de **operações** que podem ser aplicadas para manipulação desses dados.
- Ou seja, em vez de utilizar variáveis soltas, que seriam operadas separadamente, um TAD permite uni-las em uma única entidade lógica, que é manipulada como um todo.
- O usuário nunca terá acesso direto às variáveis, e sim às operações, também chamadas de funções, que as manipulam.

# Tipos Abstratos de Dados - TAD

Principais vantagens em fazer o uso de TADs são:

- Reutilização: pode-se reutilizar TADs criadas em outros programas em um novo programa sem a necessidade de realizar nova implementação.
- Manutenção: pode-se facilmente realizar a manutenção em uma TAD criada anteriormente ou adicionar novas operações, se necessário.

# Tipos Abstratos de Dados - TAD

## Criação de um TAD

- Na prática, um TAD é implementado por meio de um **tipo composto** (*class*) e por um conjunto de funções que operam essa estrutura.

## Exemplo: TAD Fração

- Considerando as seguintes características:
  - Uma fração é representada por dois inteiros: o numerador e o denominador;
  - Uma fração pode ser multiplicada por outra fração.

# Tipos Abstratos de Dados - TAD

## Criação do TAD Fracao:

- Definição da estrutura:

```
class Fracao:
    __num = None
    __den = None

    def __init__(self, num, den):
        self.setNum(num)
        self.setDen(den)
    ...
    def multiplica(self):
        pass
```

# Tipos Abstratos de Dados - TAD

- **Acesso a um TAD**

- Assim como os tipos de dados primitivos, um TAD para ser acessado precisa de uma **variável**.
- A variável associará o TAD a uma área de memória.
- Os membros do TAD são acessados individualmente com o operador de acesso “.”, preferencialmente por meio de uma operação.
- Exemplo:           variavel.membro1()  
                          variável.membro2()