

Objetivo do Tema

Compreender as diferenças entre tipos de dados pré-existentes e tipos abstratos de dados.



- Tipos Primitivos:
 - um tipo estabelece a natureza (característica) do dado que é manipulado por um algoritmo;
 - define o conjunto de valores que uma variável, constante ou função podem receber;
 - tipos de dados básicos (primitivos):
 - **inteiro**: 13, 6, 7830, 295;
 - real: 23.8, 3.6752, 8.910, 3738.72, 32.0;
 - caractere: "UNICARIOCA", "Computação", "111 + 222 = 333";
 - lógico: FALSO, VERDADEIRO.

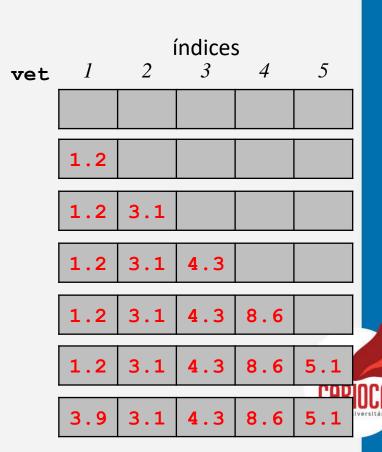


- Tipos Estruturados:
 - uma estrutura de dados é um modo particular de armazenamento e de organização de dados para que possam ser usados eficientemente;
 - os tipos estruturados são exemplos de estruturas de dados;
 - tipos de estruturas de dados:
 - homogêneas (vetores e matrizes): conjuntos de dados formados pelo mesmo tipo de dado básico;
 - heterogêneas (registros): conjuntos de dados formados por tipos de dados básicos diferentes (campos do registro).



- Tipos Estruturados Homogêneos:
 - um vetor é uma estrutura que armazena os dados em uma linha e em várias colunas, ou seja, é unidimensional;
 - exemplo:

```
vet: vetor [1..5] de real
vet[1] <- 1.2
vet[2] <- 3.1
vet[3] <- vet[1] + vet[2]
vet[4] <- vet[3] * 2
vet[5] <- vet[4] - 3.5
vet[1] <- vet[1] + 2.7</pre>
```



- Tipos Estruturados Homogêneos:
 - uma matriz é uma estrutura que armazena os dados em várias linhas e em várias colunas, ou seja, é bidimensional;
 - exemplo:

```
índices
mat: vetor [1..4, 1..3] de real
                                                            3
                                      mat
mat[1, 1] <- 3.6
                                            3.6
                                                   - 2.7
                                                           6.1
mat[1, 2] <- - 2.7
                                      indices
mat[1, 3] <- 6.1
                                                    3.2
                                                           4.8
mat[2, 1] < -4.3 + 1.4
                                            2.4
                                                    2.0
                                                          - 1.5
mat[2, 2] \leftarrow (1.9 + 7.7) / 3
mat[2, 3] \leftarrow mat[1, 1] + 1.2
                                                    7.0
                                                           8.9
                                           - 4.6
mat[3, 1] \leftarrow mat[2, 3] / 2
mat[3, 2] <- - mat[2, 3] / mat[3, 1]
mat[3, 3] \leftarrow 7.5 - mat[1, 3] + mat[2, 1]
mat[4, 1] \leftarrow mat[2, 2] - 2.6 * 3
mat[4, 2] < -8 ^ (1 / 3) + RaizQ(25)
```

 $mat[4, 3] \leftarrow mat[1, 2] - mat[4, 1] + mat[4, 2]$

- Tipos Estruturados Heterogêneos:
 - um registro é uma estrutura formada por um conjunto de variáveis de tipos diferentes ou, eventualmente, iguais;
 - exemplo:

```
tipo
  t_func = registro
    nome: caractere
    salario: real
    setor: caractere
    PNE: logico
    fimregistro

reg: t_func
reg.nome <- "Fulano"
reg.salario <- 1500.0
reg.setor <- "vendas"
reg.PNE <- FALSO</pre>
```

req

"Fulano"

1500.0

"vendas"

FALSO



Tipos Estruturados Heterogêneos:

– exemplo:

req

```
vet: vetor [1..3] de t aluno
tipo
                                   vet[1].nome <- "Fulano"</pre>
  t aluno = registro
    nome: caractere
                                   vet[1].nota <- 7.8
    nota: real
                                   vet[1].turma <- 543
    turma: inteiro
                                   vet[2].nome <- "Sicrano"</pre>
                                   vet[2].nota <- 9.5</pre>
  fimregistro
                                   vet[2].turma <- 726
                                   vet[3].nome <- "Beltrano"</pre>
                                   vet[3].nota <- 8.4</pre>
                                   vet[3].turma <- "682"</pre>
                          índices
```

"Fulano"	"Sicrano"	"Beltrano"
7.8	9.5	8.4
543	726	682

é possível criar também matrizes de registros

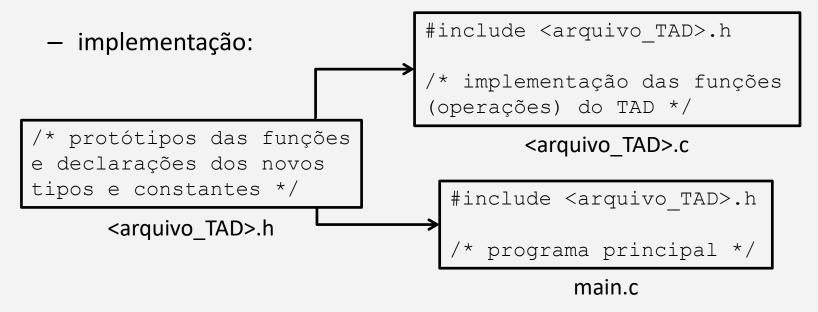
- Tipos Abstratos de Dados:
 - tipos estruturados, em geral, definem uma coleção de valores de um tipo primitivo (vetores e matrizes), ou um agregado de valores de tipos primitivos diferentes (registros);
 - um tipo abstrato de dados (TAD) é uma coleção bem definida de dados a serem armazenados e um grupo de operações que podem ser aplicadas na manipulação desses dados:
 - **exemplo:** o conjunto dos números inteiros acompanhado das operações de adição, subtração e multiplicação.



- Tipos Abstratos de Dados:
 - características:
 - TAD's são generalizações de tipos primitivos, assim como funções são generalizações de operações primitivas (+, -, *, /);
 - assim como uma função encapsula partes de um algoritmo, o TAD pode encapsular tipos de dados;
 - em um TAD, a palavra "abstrato" quer dizer "não importa a forma de implementação", ou seja, basta saber a finalidade do tipo e de suas operações;
 - os TAD'S são uma técnica de programação importante, pois facilitam a manutenção e o reuso de código.



Tipos Abstratos de Dados:



- razões para incluir <arquivo_TAD>.h em <arquivo_TAD>.c:
 - podem existir definições na interface que são necessárias na implementação;
 - garantir que as funções implementadas correspondam às funções de interface.

- Tipos Abstratos de Dados:
 - exemplo:
 - criação de um tipo de dado Ponto para representar um ponto no \mathbb{R}^2 ;
 - operações:
 - **cria**: cria um ponto com coordenadas $x \in y$;
 - libera: libera a memória alocada por um ponto;
 - acessa: retorna as coordenadas de um ponto;
 - atribui: atribui novos valores às coordenadas de um ponto;
 - distancia: calcula a distância entre dois pontos.



- Tipos Abstratos de Dados:
 - exemplo (arquivo "ponto.c"):

```
#include <stdio.h>
#include <stdlib.h> // malloc, free, exit
#include <math.h> // sqrt
#include "ponto.h"

struct ponto
{
   float x;
   float y;
};
```



- Tipos Abstratos de Dados:
 - exemplo (arquivo "ponto.c"):

```
Ponto * cria(float x, float y)
   Ponto *p = (Ponto *) malloc(sizeof(Ponto));
   if (p == NULL)
      printf("Mem\242ria insuficiente!\n");
      exit(1);
   p->x = x;
   p \rightarrow y = y;
   return p;
```



- Tipos Abstratos de Dados:
 - exemplo (arquivo "ponto.c"):

```
void libera(Ponto *p)
   free(p);
void acessa(Ponto *p, float *x, float *y)
  *x = p->x;
   *y = p->y;
void atribui(Ponto *p, float x, float y)
  p->x = x;
  p->y = y;
```



- Tipos Abstratos de Dados:
 - exemplo (arquivo "ponto.c"):

```
float distancia(Ponto *p1, Ponto *p2)
{
    float dx, dy;

    dx = p2->x - p1->x;
    dy = p2->y - p1->y;

    return sqrt(dx * dx + dy * dy);
}
```



a composição (campos) de Ponto

Tipos Abstratos de Dados:

```
– exemplo (arquivo "ponto.h"):
```

```
typedef struct ponto [Ponto;]
Ponto* cria(float x, float y);
void libera (Ponto* p);
void acessa (Ponto* p, float* x, float* y);
void atribui (Ponto* p, float x, float y);
float distancia (Ponto* p1, Ponto* p2);
```

– arquivo "main.c":

```
#include <stdio.h>
#include "ponto.h"

void main(void)
{
    // chamadas às funções
}
```



- Principais TAD's:
 - listas;
 - pilhas;
 - filas;
 - árvores;
 - grafos.



PRÓXIMA AULA: TEMA 2: LISTAS LINEARES COM ALOCAÇÃO ESTÁTICA

