Hortolândia, 25 de fevereiro de 2014.

Aulas 11 e 12 – Tipos primitivos e Estruturas de Dados

Na presente aula serão apresentados conceitos relacionados aos tipos primitivos básicos, suas características e peculiaridades. Também serão vistos conceitos sobre estruturas de dados homogêneas, suas funcionalidades e aplicações.

1. Tipos primitivos

Tipos primitivos são classificações que as linguagens de programação utilizam para tratar as informações. Eles facilitam a criação de algoritmos, ao mesmo tempo que possibilitam um melhor dimensionamento e utilização dos recursos de hardware/software disponíveis. Assim, idealmente, toda informação deve estar alocada em seu tipo primitivo correspondente.

As divisões mais usuais dos tipos primitivos são:

- Tipos numéricos: utilizados para armazenar quaisquer informações numéricas a serem processadas. Quantidades de itens, valores em contas matemáticas e algébricas, grandezas em geral podem ser representados por estes tipos. Ainda dentro dos tipos numéricos, é comum haver outras divisões:
  - ✓ Tipos Inteiros: permitem a representação de valores inteiros, ou seja, que não precisam da representação de valores fracionários/decimais. Além disso, os valores podem ser positivos ou negativos, como no conjunto dos inteiros da matemática. São tipos mais facilmente tratados pelos computadores, sendo assim recomendável sua utilização sempre que a informação representada assim o permitir. É importante ressaltar que nas linguagens de programação, muitas vezes os tipos inteiros possuem subdivisões, abrangendo diferentes faixas de representatividade. Assim, é comum ver tipos que utilizam desde 1 até 16 bytes ou mais para representar valores inteiros. Alguns exemplos de valores inteiros: “**20** degraus”, “**18** pessoas”, “**35** alunos”, “**30** dias no mês”.
  - ✓ Tipos Reais: são usados para representar qualquer valor que precise representar grandezas fracionárias/decimais. Sempre que uma informação exigir a parte fracionária é necessário utilizar estes tipos. Assim como nos tipos inteiros, os tipos reais podem ser subdivididos em linguagens de programação, de forma a abranger diferentes faixas de representatividade numérica. Alguns exemplos de valores reais: “**25.4** graus Celsius”, “**8.5** de média final”, “**1235.34** reais de salário”. É importante ressaltar que para representar a casa decimal em muitas linguagens de programação se utiliza o caractere de ponto(.) ao invés da usual vírgula.
- Tipos textuais: utilizados para representar quaisquer informações que envolvam caracteres alfanuméricos e texto. De acordo com a linguagem de programação, podem ter algumas subdivisões, como:
  - ✓ Caractere único: tipo usado para representar um único caractere alfanumérico. Informações que precisem de 2 ou mais caracteres não podem usar este tipo. Exemplos: turma **‘A’**, Sexo **‘M’/’F’**. Também é importante ressaltar que, assim como nos exemplos, em muitas linguagens se utiliza um par de aspas simples para representar este tipo de dado.
  - ✓ Texto: tipo usado para representar informações textuais, com 2 ou mais caracteres alfanuméricos. Em muitas linguagens de programação, não

existe um tipo primitivo real para esse tipo de informação, sendo muitas vezes usado o tipo de caractere único em estruturas de dados homogêneas unidimensionais (vetores) para compor textos. Exemplos: **“Godofredo”**, **“Texto do exemplo 1”, “Mensagem do sistema: ”**. É importante ressaltar que em muitas linguagens, assim como nos exemplos, se utiliza um par de aspas duplas para delimitar valores deste tipo.

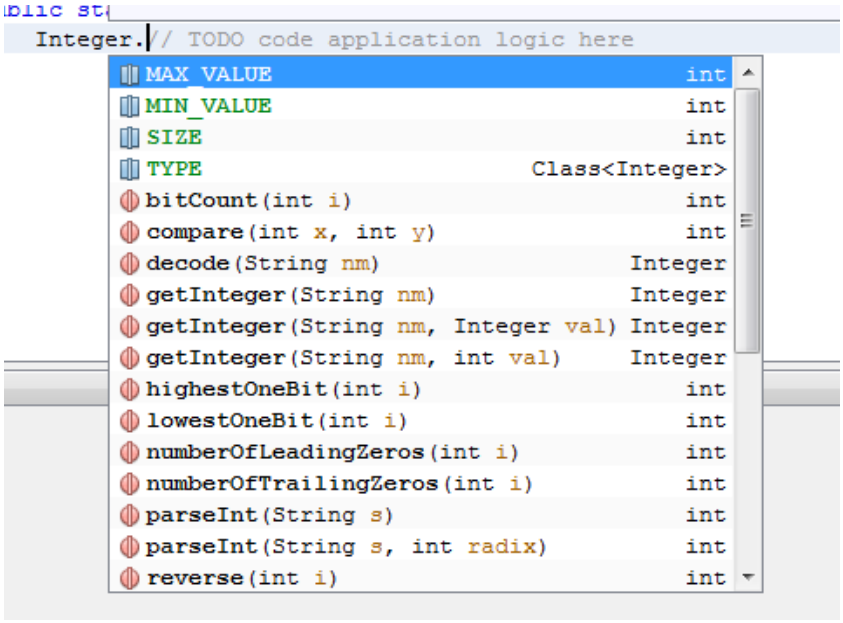
- Tipo Lógico: utilizado para representar 2 estados de qualquer informação. Geralmente associado a operações relacionais e lógicas nos algoritmos. Por padrão, os valores representados são **Verdadeiro** e **Falso**, mas existem variações dessa representação de acordo com a linguagem.

2. Tipos primitivos em Java

Os tipos primitivos em Java seguem aproximadamente as divisões anteriores. São eles:

- Tipos numéricos
  - ✓ Tipos inteiros
    - byte - Representam números inteiros de 8 bits (1 byte). Podem assumir valores entre -128 a 127.
    - short - Representam números inteiros de 16 bits (2 bytes). Podem assumir valores entre -32.768 até 32.767.
    - int - Representam números inteiros de 32 bits (4 bytes). Podem assumir valores entre -2.147.483.648 até 2.147.483.647.
    - long - Representam números inteiros de 64 bits (8 bytes). Podem assumir valores entre -9.223.372.036.854.775.808 até 9.223.372.036.854.775.807.
  - ✓ Tipos reais
    - float - Representam números reais de 32 bits com precisão simples. Podem assumir valores de ponto flutuante com formato definido pela especificação IEEE 754.
    - double - Representam números reais de 64 bits com precisão dupla. Assim como o float. Podem assumir valores de ponto flutuante com formato definido pela especificação IEEE 754.
- Tipos Textuais
  - ✓ char - Representam notação de caracteres de 16 bits (2 bytes) para formato Unicode UTF-16. Podem assumir caracteres entre '\u0000' a '\uffff' e valores numéricos entre 0 a 65535.
  - ✓ String – Não é um tipo primitivo de fato, e sim uma classe, mas é a forma padrão de utilização de texto em Java. O tipo String utiliza vetores do tipo char para formar o texto, mas provê métodos diversos de controle para cada um deles
- Tipo lógico
  - ✓ boolean - Representam apenas 1 bit de informação (0 ou 1). Podem assumir apenas os valores true e false.

Em Java, cada tipo primitivo possui também uma classe associada, que é responsável por prover métodos úteis para o tipo, tais como converter um texto para um valor numérico, o contrário, dentre outras funcionalidades. A seguir são mostrados alguns métodos para a classe associada ao tipo int, que é a Integer:



As classes de cada tipo primitivo são:

- ✓ byte – Byte
- ✓ short – Short
- ✓ int – Integer
- ✓ long – Long
- ✓ float – Float
- ✓ double – Double
- ✓ boolean – Boolean
- ✓ char – Character
- ✓ String – String

É importante notar que, especialmente para os tipos numéricos, existe sempre a preocupação com a **compatibilidade em operações**, pois cada tipo ocupa uma quantidade de bytes na memória e pode representar uma certa faixa de valores.

Assim, não é possível colocar um valor do tipo double(64 bits) em uma variável do tipo byte(8 bits) sem que haja perda de dados, e essa operação de forma direta nem mesmo é permitida nos compiladores, gerando um erro.

Para poder haver esse recebimento, são necessárias as chamadas operações de **casting**. Essas operações fazem o trabalho de converter o tipo recebido para o tipo destinatário. Se o tipo destinatário for maior que o recebido, o casting é chamado de implícito,

pois o usuário não precisa fazer nada, o compilador realizar o processo de conversão automaticamente.

Ex:

```
int a=500;
long b = a; // b recebe a e faz o processo de conversão automaticamente pois long > int
```

Quando o tipo destinatário for menor que o fonte, é necessário fazer o casting explícito, ou seja, dar ao compilador a instrução de transformar o tipo recebido para algum tipo compatível com o destinatário,usualmente o próprio tipo do destinatário. Isso é feito usando a seguinte sintaxe:

variavelDestino = (tipo a converter)variavemOrigem;

ex:

```
short a;
int b=250;
a=(short)b; //a recebe o valor de b transformado para o tipo short
```

*Obs: Nos casos de um tipo inteiro recebendo um real, sempre será necessário fazer o casting explicitamente, mesmo que o destinatário inteiro seja maior que o real.

3. Exercícios de tipos primitivos

1 - Identifique o melhor tipo primitivo para os dados a serem representados nas proposições a seguir (dados em negrito):

José comprou **50** cabeças de gado para sua fazenda.

A balança acusava o peso de **75.4** Kg.

As alternativas disponíveis nas questões eram **'a'**, **'b'**, **'c'** e **'d'.**

O título do filme era **"A volta dos que não foram".**

O professor Afrânio queria tirar a **média** geral das notas dos alunos, com 2 casas decimais.

Ao se deparar com a turma, pôde contar **23** alunos presentes.

Durante o mês, Bete conseguiu alcançar a marca de R$ **3500.38** em vendas diretas ao cliente.

A questão dizia, marque com **V**/**F** cada proposta.

Na inscrição, haviam as opções **M**/**F** no campo sexo.

Esteban preencheu o campo profissão com **"Ajudante geral"**.

2 – Analise cada atribuição e corrija as que tiverem erros quanto ao casting implícito/explícito:

```
int a=0;
short b=0;
float c=0f;
double d=0;
```

```
byte e=0;

long f=0;

a=b;

a=c;

a=d;

a=e;

d=a;

d=c;

d=b;

c=d;

c=f;

f=c;

f=d;
```

4. Estruturas de dados - Listas

Listas são um dos tipos mais estudados de estruturas de dados. São estruturas de dados lineares, possuindo apenas 1 dimensão por natureza. De acordo com a implementação, podem ser estáticas(número fixo de elementos) ou dinâmicas(número variável de elementos), bem como homogêneas(todos elementos de um mesmo tipo) ou heterogêneas(elementos de tipos variados).

As listas possuem normalmente algumas operações que podem ser feitas sobre elas:

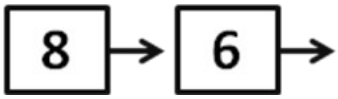
- ✓ Criar Lista – Executa os passos iniciais, aloca memória, define parâmetros da lista.
- ✓ Destruir lista – Destrói a referência da lista na memória, liberando o espaço por ela utilizado.
- ✓ Inserir Elemento – Insere um novo elemento na lista. De acordo com o tipo de implementação, o elemento pode ser inserido no início, fim ou posição específica da lista.
- ✓ Excluir Elemento – Retira um elemento da lista. De acordo com a implementação, esse elemento pode ser retirado do final, começo ou posição específica da lista. Se a lista for dinâmica, a memória ocupada pelo elemento também é liberada.
- ✓ Procurar Elemento – Realiza uma busca por um elemento numa lista existente, comparando ao valor desejado
- ✓ Lista cheia – Operação geralmente usada em listas estáticas, indica que a lista chegou ao limite de elementos armazenáveis
- ✓ Lista Vazia – Indica que não existe nenhum elemento na lista atual.

As listas podem ser implementadas de diversas maneiras. Se forem estáticas, usualmente são implementadas em um vetor simples, com campos auxiliares para moldar os parâmetros da lista, como seu tamanho, quantidade de elementos inseridos, etc.

A seguir é exemplificada visualmente uma lista estática com 1 elemento inserido e capacidade de armazenamento de até 8 elementos inteiros.

Neste caso, o elemento foi inserido no começo da lista, mas poderia ter sido feito o contrário também, sendo os algoritmos das funções da lista responsáveis por implementar de forma adequada o tipo de inserção escolhido.

Listas dinâmicas envolvem outras formas de implementação, pois um vetor, sendo estático por natureza, complica e torna ineficaz uma versão de listas dinâmicas. Uma forma comum é criar uma estrutura onde cada elemento tem um campo de informação com o seu valor e uma ligação com outro elemento:



Essa forma de implementação permite algumas facilidades, como remoção de elementos simplesmente trocando as ligações e liberando a memória, inserção em posição determinada também somente manipulando as ligações entre os elementos, etc. Uma desvantagem é a tarefa de pesquisa, que precisa percorrer sequencialmente todos os nós até encontrar ou não um elemento desejado.