UFRPE
Universidade Federal Rural de Pernambuco

Paradigmas de Programação

Tipos de Dados

Kellyton Brito
kellyton.brito@gmail.com
14 e 17/11/2017

UFRPE
Universidade
Federal Rural
de Pernambuco

Aula Passada

O que é mesmo uma variável?

Uma variável é uma abstração para uma célula de memória

Introdução

- Variáveis possuem um conjunto de propriedades / características
  - Nome
  - Endereço
  - Tipo
  - Valor
  - Escopo
  - Tempo de vida

UFRPE
Universidade
Federal Rural
de Pernambuco

Uma visão mais aprofundada sobre os tipos de dados

O que é um tipo?

Um tipo de dados é uma coleção de valores e um conjunto de operações possíveis sobre esses valores

Variáveis

Tipo

- Define a faixa de valores possíveis
- Define operações possíveis
- Define a quantidade de memória necessária para armazenamento
- Ex.:

Tipo	Tamanho	Valores
byte	8 bits	-128...127
short	16 bits	-32768...32767
int	32 bits	-2147484... 2147483
long	64 bits	-9e18...9e18

Tipos de Tipos

Quais os tipos de tipos possíveis?

- Tipos Primitivos
- Tipos String de Caracteres
- Tipos Ordinais Definidos pelo Usuário
- Arrays
- Tipos Registros
- Tipos Ponteiros e Referência

Tipos de Tipos

Quais os tipos de tipos possíveis?

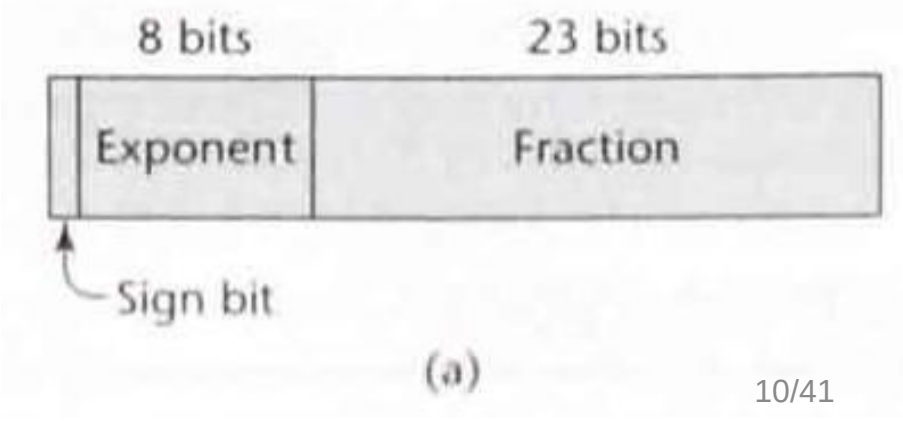
- **Tipos Primitivos**
  - **Numéricos**
  - **Caracteres**
- Tipos String de Caracteres
- Tipos Ordinais Definidos pelo Usuário
- Arrays
- Tipos Registros
- Tipos Ponteiros e Referência

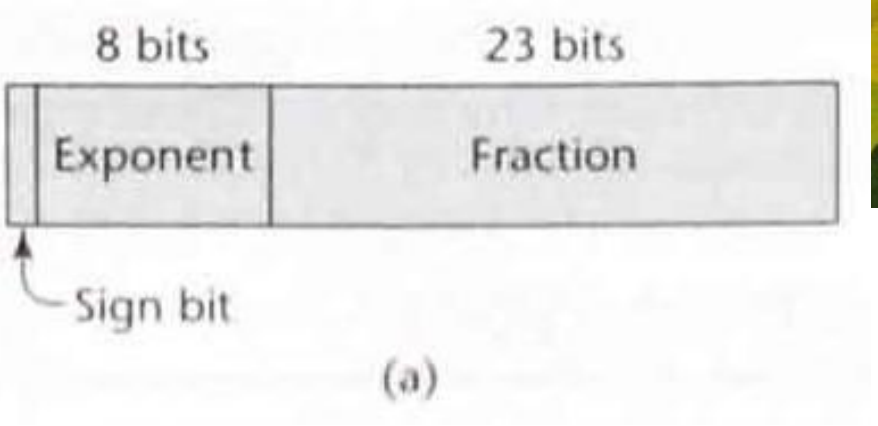
Tipos Primitivos

- Tipos de dados não baseados em outros tipos
- Baseados em representações numéricas
- Relações diretas com os bytes

Tipos Primitivos

- Tipos Numéricos:
  - Inteiros:
    - Representa números inteiros (positivos ou negativos)
    - Byte, Short, Int, Long
  - Ponto Flutuante:
    - Modela números reais
    - Float, Double
    - Representado como expoente e fração





Tipos Primitivos

- Exemplos
  - Representação inteira com bit de sinal:
    - 01101000 = 104
    - 11101000 = -24 (bit de sinal + 104, complementa 128)
  - Representação de ponto flutuante: Potências de 2

$$\textbf{Valor} = S \times M \times 2^{E}$$

Onde:
- S = 1 − 2*s
- M = 1.m
  = 1 + m*2^{-23}
- E = e − 127

Valor	S × M × 2^E	s	m	e	IEEE 754 - Single Precision
1	1 × 1 × 2^0	0	0x00	127	0 0111 1111 000 0000 0000 0000 0000 0000
-1	-1 × 1 × 2^0	1	0x00	127	1 0111 1111 000 0000 0000 0000 0000 0000
0,5	1 × 1 × 2^{-1}	0	0x00	126	0 0111 1110 000 0000 0000 0000 0000 0000
-0,5	-1 × 1 × 2^{-1}	1	0x00	126	1 0111 1110 000 0000 0000 0000 0000 0000

Tipos Primitivos

- Tipos Numéricos (continuação)
  - Tipos decimais:
    - Número fixo de casas decimais
    - Representa melhor do que ponto flutuante
    - Faixa de valores limitada
    - Gasto de memória (um ou dois byte(s) por digito)
  - Tipos booleanos
    - 2 Elementos: verdadeiro ou falso
    - Representação em 1 bit
    - Existentes em Java e C#. Representação inteira em C

Tipos Primitivos

- Tipos de Caracteres
  - Armazenados como códigos numéricos
  - ASCII (American Standard Code for Information Interchange)
    - Representação em 8 bits
    - Valores de 0 a 127 para representar 128 caracteres
  - Novo conjunto: Unicode
    - Representação em 16 bits
    - Primeiros 128 idênticos ao ASCII
- Usados como base para os tipos String

Tipos de Tipos

Quais os tipos de tipos possíveis?

- Tipos Primitivos
- **Tipos String de Caracteres**
- Tipos Ordinais Definidos pelo Usuário
- Arrays
- Tipos Registros
- Tipos Ponteiros e Referência

Tipos String de Caracteres

- Consistem de sequencia de caracteres

- Normalmente saídas e entradas de/em programas é feita por Strings

- Diferenças de implementação
  - Array de caracteres ou tipo especial de tipo primitivo?
  - Deve ter tamanho estático ou dinâmico?

Tipos String de Caracteres

- Em C/C++: Arrays de caracteres
  - Terminadas com *Null*
  - Funções de manipulação na biblioteca string.h

- Em Java+: Tipos especiais: Objetos
  - Representação interna em arrays de caracteres de tamanho fixo
  - Manipulações através de métodos da classe String

Tipos String de Caracteres

- Peculiaridades das Strings em Linguagens de Alto nível
  - Pode ser inicializada como um tipo primitivo ou objeto
    - *String str1 = "teste1";*
    - *String str2 = new String("teste2");*
  - Manipulações sempre criam **novas** Strings
    - É criada e atribuída à variável correspondente
  - Comparações pelo método equals

```
if (a.equals(b)) { // faça algo }
```

    - Erro comum: comparações com "==" normalmente são *false*

Tipos String de Caracteres

- Peculiaridades das Strings em Java
  - Em alguns casos, comparação com == é ***true***
- Declaração: String a = “Joao”
  - *Link* em tempo de compilação
  - String colocada no pool de Strings
  - Se for declarada outra String “Joao” da mesma forma na mesma classe, reutiliza a String do pool de strings
    - Economia de espaço
    - Permite comparações ==

Tipos String de Caracteres - Exemplo

```
public void codigo1(){
     String a = "Joao";
     String b = "Joao";
     if (a == b){
          System.out.println("Sao iguais");
     } else {
          System.out.println("Sao diferentes");
     }
}

public void codigo11(){
     String a = new String("Joao");
     String b = new String("Joao");
     if (a == b){
          System.out.println("Sao iguais");
     } else {
          System.out.println("Sao diferentes");
     }
}
```

Iguais

Diferentes

Tipos String de Caracteres - Exemplo

```
public void codigo111(){
     String a = new String("Joao");
     String b = "Joao";
     if (a == b){
          System.out.println("Sao iguais");
     } else {
          System.out.println("Sao diferentes");
     }
}
```

Diferentes

Tipos de Tipos

- Tipos Primitivos
- Tipos String de Caracteres
- **Tipos Ordinais Definidos pelo Usuário**
- Arrays
- Tipos Registros
- Tipos Ponteiros e Referência

Tipos Ordinais Definidos pelo Usuário

- Tipo ordinal: Faixa de valores associada a um conjunto de inteiros
  - Ex.: Booleano. True = 1 // False = 0
  - Inteiros (ele mesmo) e Char (cada caracter tem um inteiro associado)

- Em algumas linguagens, usuários podem definir tipos ordinais: Enumerações e Subfaixas

Tipos Ordinais Definidos pelo Usuário

- Enumerações:
  - Tipos em que todos os valores possíveis são constantes, e descritos na definição
  - Exemplo (C#):

    enum DiaSemana = {Seg, Ter, Qua, Qui, Sex, Sab, Dom};
  - Constantes recebem de 0 a (tamanho -1)
    - Seg = 0 ... Dom = 6
  - Uso:
    - *DiaSemana d = Seg;*
    - *If (d == Seg) …*

Tipos Ordinais Definidos pelo Usuário

- Enumerações:
  - Exemplo Java:
    - Todas enumerações herdam de java.lang.Enum
    - Por serem classes, podem ter métodos e atributos

```
public enum Day {
    SUNDAY, MONDAY, TUESDAY, WEDNESDAY,
    THURSDAY, FRIDAY, SATURDAY
}
```

- Mais informações em:
  - http://java.sun.com/docs/books/tutorial/java/javaOO/enum.html

Tipos Ordinais definidos pelo Usuário

```java
@Entity
public class Usuario {

    public static enum Situacao {
        NOVA, PRONTA, EXPIRADA, DESATIVADA, DELETADA
    }

    public static enum Nivel {
        SUPER_ADMIN, ADMIN, USUARIO
    }

    @Id
    @Column(columnDefinition = "VARCHAR(127)")
    private String id;

    @Column(columnDefinition = "VARCHAR(127)")
    private String nome;

    @Column(columnDefinition = "VARCHAR(127)")
    @Index(name="login")
    private String login;

    @Column(nullable = false)
    private Nivel nivel;

    @Column(nullable = false)
    private Situacao situacao;
```

Tipos Ordinais Definidos pelo Usuário

- SubFaixas:
  - É uma sequência contínua de um tipo ordinal
    - Ex.: 10 .. 20 é uma subfaixa dos inteiros
  - Usado para definir uma enumeração baseada em outra, porém menor
  - Ex. (ADA):

type Dias is (Seg, Ter, Qua, Qui, Sex, Sab, Dom);

Subtype DiasUteis is Dias range Seg..Sex;

Tipos de Tipos

- Tipos Primitivos
- Tipos String de Caracteres
- Tipos Ordinais Definidos pelo Usuário
- **Arrays**
- Tipos Registros
- Tipos Ponteiros e Referência

Tipos Arrays

- Uma lista homogênea de elementos identificados pela sua posição relativa ao primeiro elemento
- Os elementos tem um tipo definido previamente (primitivo ou não)
- Referenciados pelo nome, e acessados pelo índice

meuArray[2] = 3

Tipos Arrays

- Questões de design das linguagens:
  - Quais tipos são possíveis para o índice?
  - Os índices podem ser expressões?
  - Quais tipos podem ser atribuídos aos elementos de um array?
  - São permitidos arrays multidimensionais?
  - Arrays como tipo retorno

Tipos Arrays

- Declaração e inicialização
  - Na maioria das linguagens: Arrays devem ser declarados, inicializados e então usados

```
int[] arrayInteiros;
arrayInteiros = new int[10];
arrayInteiros[0] = 15
```

  - Ou pode ser feito tudo em um único comando:

```
int [] list = {2,4,6,8,10};
```

Tipos Arrays

- Arrays unidimensionais e multidimensionais:
  - Unidimensionais:

```
int[] arrayInteiros;
arrayInteiros = new int[10];
arrayInteiros[0] = 15
```

  - Multidimensionais: int[][]array;

```
int[][] matriz;
matriz = new int[10][5];
matriz[0][0] = 8
```

Tipos Arrays

- Arrays x Vector
  - Algumas linguagens possuem a classe Vector, com funcionamento semelhante ao array
  - Aumenta e diminui automaticamente
  - Parametrizado:
    - Vector<Pessoa> p = new Vector<Pessoa>();
  - Principais métodos:
    - add(E element)
    - add(int index, E element)
    - contains (Object o)
    - elementAt(int index), get(int index)
    - remove(int index), remove (Object o)
    - size()

Tipos de Tipos

- Tipos Primitivos
- Tipos String de Caracteres
- Tipos Ordinais Definidos pelo Usuário
- Arrays
- **Tipos Registros**
- Tipos Ponteiros e Referência

Tipos Registros

- Enquanto um array é uma lista ***homogênia*** de elementos
- Um Registro é um conjunto ***heterogêneo*** de elementos, identificados por um nome
  - Exemplo: Informações sobre um aluno (nome, cpf, notas...)
  - Em C/C++/C# é modelado com **struct**
- Questões de Design
  - Qual a forma sintática de referenciar campos?
  - É permitido recursão?

Tipos Registros

- Tipos heterogêneos
- Não referenciados por índices, e sim por ***fields*** (campos)

```
type Employee_Name_Type is record
   First : String (1..20);
   Middle : String (1..10);
   Last : String (1..20);
end record;
```

- No geral, os campos são acessados por ponto:

```
Employee_Name_Type nome;
nome.first = “Kellyton”;
```

Tipos de Tipos

- Tipos Primitivos
- Tipos String de Caracteres
- Tipos Ordinais Definidos pelo Usuário
- Arrays
- Tipos Registros
- **Tipos Ponteiros e Referência**

Tipos Ponteiros

- Um ponteiro consiste em endereços de memória ou *null* (vazio)
- Pode ser usado para acesso a uma área de alocação dinâmica, chamada de ***heap***
- Tipos valor e referência:
  - Valor: Armazena um valor escalar (int, long, etc)
    - Tipos primitivos
  - Referência: Armazena uma referência (ponteiro) para um endereço de memória
    - Tipos compostos

Ponteiros em C

- Na declaração
  - * significa que a variável é um ponteiro
- No uso
  - * significa o conteúdo da variável
  - & significa o ponteiro para a variável

```
int *ptr;
int count, init;
...
ptr = &init;
count = *ptr;
```

Se init = 10 e count = 20, qual o valor final de count?

equivalente a *count* = *init*

Ponteiros em linguagens OO

- Em linguagens OO, tipos primitivos são valores, e objetos são referência

- Alterações em parâmetros de métodos:
  - Tipo primitivo não alteram o original
  - Objetos alteram o original
    - Obs.: Objetos alteram o original, porém alteração do link fica dentro do escopo do método
  - Normalmente, arrays também são passados por referência

Ponteiros em OO - Exemplo

```
public void execute(){
    int i = 2;
    Pessoa p = new Pessoa("Joao");
    alteraValores(i, p);
    System.out.println(i);
    System.out.println(p.nome);
}

private void alteraValores(int i, Pessoa p) {
    i = 3;
    p.nome = "José";
    p = new Pessoa("Maria");
}

public static void main(String args[]){
    new ExemploPonteiro().execute();
}
```

```
class Pessoa {
    String nome;
    public Pessoa(String nome){
        this.nome = nome;
    }
}
```

O que é impresso?

Imprime 2 e José

UFRPE
Universidade Federal Rural de Pernambuco

Dúvidas / considerações / sugestões?