Aula 2: Tipos básicos e Variáveis

Adriano Veloso

Algoritmos e Estruturas de Dados I - DCC/UFMG

Variáveis

Em qualquer linguagem de programação, uma *variável* representa uma localização de memória.

- Nós armazenamos algum valor em variáveis, de forma a utilizar esse valor mais tarde no programa
- O nome **variável** dá-se pelo fato de que o valor armazenado nessa localização de memória pode ser alterado à medida em que o programa é executado.
- Quando referenciamos uma variável, significa que estamos referenciando o valor armazenado nessa variável

Variáveis

```
int main() {
        ...
         int my_var;
        ...
}
```

Nome de variável

Toda variável tem um nome, escolhido pelo programador.

- O nome deve ser auto-explicativo

Nome de variável

Toda variável tem um nome, escolhido pelo programador.

- O nome deve ser auto-explicativo
- Podemos usar quantos caracteres desejarmos, mas apenas os 31 primeiros serão levados em conta

Nome de variável

Toda variável tem um nome, escolhido pelo programador.

- O nome deve ser auto-explicativo
- Podemos usar quantos caracteres desejarmos, mas apenas os 31 primeiros serão levados em conta
- Só podemos utilizar caracteres alfa (“A” a “Z” ou a “a” “z”), caracteres numéricos (0 a 9), e o símbolo de *underscore*.

Nome de variável

Toda variável tem um nome, escolhido pelo programador.

- O nome deve ser auto-explicativo
- Podemos usar quantos caracteres desejarmos, mas apenas os 31 primeiros serão levados em conta
- Só podemos utilizar caracteres alfa ("A" a "Z" ou a "a" "z"), caracteres numéricos (0 a 9), e o símbolo de *underscore*.
- Caracteres maiúsculos são diferentes de caracteres minúsculos (a variável **my_var** é diferente da variável **MY_VAR**)

Nome de variável

Toda variável tem um nome, escolhido pelo programador.

- O nome deve ser auto-explicativo
- Podemos usar quantos caracteres desejarmos, mas apenas os 31 primeiros serão levados em conta
- Só podemos utilizar caracteres alfa ("A" a "Z" ou a "a" "z"), caracteres numéricos (0 a 9), e o símbolo de *underscore*.
- Caracteres maiúsculos são diferentes de caracteres minúsculos (a variável **my_var** é diferente da variável **MY_VAR**)
- Uma variável não pode ter como nome uma *palavra reservada* da linguagem C (por exemplo **main**)

Tipos de dados

Vários tipos de dados podem ser armazenados em variáveis.

- Do ponto de vista do compilador, a principal diferença entre os diversos tipos existentes está no tamanho em bytes ocupado por cada tipo

Tipos de dados

Vários tipos de dados podem ser armazenados em variáveis.

- Do ponto de vista do compilador, a principal diferença entre os diversos tipos existentes está no tamanho em bytes ocupado por cada tipo
- Do ponto de vista do programador, a principal diferença entre os diversos tipos existentes está em suas funcionalidades

Tipos de dados

Vários tipos de dados podem ser armazenados em variáveis.

- Do ponto de vista do compilador, a principal diferença entre os diversos tipos existentes está no tamanho em bytes ocupado por cada tipo
- Do ponto de vista do programador, a principal diferença entre os diversos tipos existentes está em suas funcionalidades

Tipos básicos:

- Tipo inteiro, denotado pela palavra reservada **int**
- Tipo fracionário, denotado pela palavra reservada **float**
- Tipo caracter, denotado pela palavra reservada **char**

Tipos de dados

Vários tipos de dados podem ser armazenados em variáveis.

- Do ponto de vista do compilador, a principal diferença entre os diversos tipos existentes está no tamanho em bytes ocupado por cada tipo
- Do ponto de vista do programador, a principal diferença entre os diversos tipos existentes está em suas funcionalidades

Tipos básicos:

- Tipo inteiro, denotado pela palavra reservada **int**
- Tipo fracionário, denotado pela palavra reservada **float**
- Tipo caracter, denotado pela palavra reservada **char**

Podemos criar um novo tipo de dados! (veremos no final do curso)

Declaração de variáveis

Uma variável precisa ser explicitamente declarada, antes de ser usada. A declaração tem uma série de objetivos:

- Direciona o compilador a como o valor deve ser armazenado (qual o tipo de dado será armazenado na variável).

Declaração de variáveis

Uma variável precisa ser explicitamente declarada, antes de ser usada. A declaração tem uma série de objetivos:

- Direciona o compilador a como o valor deve ser armazenado (qual o tipo de dado será armazenado na variável).
- Uma quantidade suficiente de memória é alocada para a variável

Declaração de variáveis

Uma variável precisa ser explicitamente declarada, antes de ser usada. A declaração tem uma série de objetivos:

- Direciona o compilador a como o valor deve ser armazenado (qual o tipo de dado será armazenado na variável).
- Uma quantidade suficiente de memória é alocada para a variável
- Pode inicializar o valor da variável. A posição de memória nunca está vazia, e por isso, uma variável, caso não seja inicializada, poderá armazenar um valor sem sentido algum para o programa. A inicialização é facultativa.

Declaração de variáveis

Uma variável precisa ser explicitamente declarada, antes de ser usada. A declaração tem uma série de objetivos:

- Direciona o compilador a como o valor deve ser armazenado (qual o tipo de dado será armazenado na variável).
- Uma quantidade suficiente de memória é alocada para a variável
- Pode inicializar o valor da variável. A posição de memória nunca está vazia, e por isso, uma variável, caso não seja inicializada, poderá armazenar um valor sem sentido algum para o programa. A inicialização é facultativa.

  tipo variável [=**valor inicial**];

Assinalamento

O valor de uma variável pode ser alterado a qualquer ponto do programa. Armazenar um certo valor em uma variável é um processo chamado de **assinalamento**.

- O operador de assinalamento é o =. Um assinalamento tem a seguinte sintaxe:

  variável = valor

Assinalamento

O valor de uma variável pode ser alterado a qualquer ponto do programa. Armazenar um certo valor em uma variável é um processo chamado de **assinalamento**.

- O operador de assinalamento é o =. Um assinalamento tem a seguinte sintaxe:

  variável = valor

- Uma variável só pode armazenar um valor por vez

Assinalamento

O valor de uma variável pode ser alterado a qualquer ponto do programa. Armazenar um certo valor em uma variável é um processo chamado de **assinalamento**.

- O operador de assinalamento é o =. Um assinalamento tem a seguinte sintaxe:

  variável = valor

- Uma variável só pode armazenar um valor por vez
- A cada assinalamento, realizamos uma operação de escrita na memória. Como já vimos, essa operação sobrescreve o valor armazenado na variável anteriormente.

Tipo inteiro

- Cada variável possui o mesmo número de bits da palavra da máquina. Comumente varia de 16 a 32 bits.

Tipo inteiro

- Cada variável possui o mesmo número de bits da palavra da máquina. Comumente varia de 16 a 32 bits.
- Caso uma valor maior do que o suportado seja assinalado à uma variável, ocorre **overflow** $\longrightarrow$ o valor fica aparentemente sem sentido.

Tipo inteiro

- Cada variável possui o mesmo número de bits da palavra da máquina. Comumente varia de 16 a 32 bits.
- Caso uma valor maior do que o suportado seja assinalado à uma variável, ocorre **overflow** $\rightarrow$ o valor fica aparentemente sem sentido.
- Dois modificadores opcionais: **signed** (-n a n) e **unsigned** (0 a 2$\times$n).

Tipo inteiro

- Cada variável possui o mesmo número de bits da palavra da máquina. Comumente varia de 16 a 32 bits.
- Caso uma valor maior do que o suportado seja assinalado à uma variável, ocorre **overflow** $\longrightarrow$ o valor fica aparentemente sem sentido.
- Dois modificadores opcionais: **signed** (-n a n) e **unsigned** (0 a $2 \times$n).
- Dois modificadores opcionais: **short** (8 bits) e **long** (32 bits) $\longrightarrow$ a quantidade de bits depende do tamanho da palavra da máquina.

Tipo caracter

- Cada variável possui 8 bits

Tipo caracter

- Cada variável possui 8 bits
- Cada combinação de bits representa um caracter

Tipo caracter

- Cada variável possui 8 bits
- Cada combinação de bits representa um caracter
- Veremos mais afrente como armazenar palavras e conjuntos de palavras!

Tipo fracionário

- Cada variável possui um número de bits que depende do tamanho da palavra da máquina. Comumente varia de 32 a 80 bits.

Tipo fracionário

- Cada variável possui um número de bits que depende do tamanho da palavra da máquina. Comumente varia de 32 a 80 bits.
- O tipo fracionário é implementado por operações em ponto flutuante
  - Eles não são armazenados na notação binária simples, mas sim em notação binária científica.

Tipo fracionário

- Cada variável possui um número de bits que depende do tamanho da palavra da máquina. Comumente varia de 32 a 80 bits.
- O tipo fracionário é implementado por operações em ponto flutuante
  - Eles não são armazenados na notação binária simples, mas sim em notação binária científica.
- Na notação científica, usamos a *mantissa* multiplicada por alguma potência de 10.
  - 7146 pode ser expressado por $7{,}146 \times 10^3$. A *mantissa* é 7,146 e o *expoente* é 3.

Tipo fracionário

- Cada variável possui um número de bits que depende do tamanho da palavra da máquina. Comumente varia de 32 a 80 bits.
- O tipo fracionário é implementado por operações em ponto flutuante
  - Eles não são armazenados na notação binária simples, mas sim em notação binária científica.
- Na notação científica, usamos a *mantissa* multiplicada por alguma potência de 10.
  - 7146 pode ser expressado por $7{,}146 \times 10^3$. A *mantissa* é 7,146 e o *expoente* é 3.
- Tanto a mantissa quanto o expoente têm limites.

Tipo fracionário

- Cada variável possui um número de bits que depende do tamanho da palavra da máquina. Comumente varia de 32 a 80 bits.
- O tipo fracionário é implementado por operações em ponto flutuante
  - Eles não são armazenados na notação binária simples, mas sim em notação binária científica.
- Na notação científica, usamos a *mantissa* multiplicada por alguma potência de 10.
  - 7146 pode ser expressado por $7{,}146 \times 10^3$. A *mantissa* é 7,146 e o *expoente* é 3.
- Tanto a mantissa quanto o expoente têm limites.
- Existem 3 implementações em ponto-flutuante:
  - **float** $\leq$ **double** $\leq$ **long double**

Expressões aritméticas

Uma expressão é qualquer coisa que pode ser reduzida a um valor isolado.

- Um número, por exemplo 14, é uma expressão Porém, estamos mais interessados em expressões que necessitem ser avaliadas, por exemplo 10+4.

Expressões aritméticas

Uma expressão é qualquer coisa que pode ser reduzida a um valor isolado.

- Um número, por exemplo 14, é uma expressão Porém, estamos mais interessados em expressões que necessitem ser avaliadas, por exemplo 10+4.
- Uma expressão consiste em um conjunto de valores e/ou variáveis ligadas entre si por operadores aritméticos.
  - Os operadores demonstram ao compilador como combinar os valores

Expressões aritméticas

Uma expressão é qualquer coisa que pode ser reduzida a um valor isolado.

- Um número, por exemplo 14, é uma expressão Porém, estamos mais interessados em expressões que necessitem ser avaliadas, por exemplo 10+4.
- Uma expressão consiste em um conjunto de valores e/ou variáveis ligadas entre si por operadores aritméticos.
  - Os operadores demonstram ao compilador como combinar os valores
  - Algumas expressões possuem múltiplos operadores. Nesse caso, as regras de **precedência** e **associatividade** devem ser respeitadas.

Expressões aritméticas

Uma expressão é qualquer coisa que pode ser reduzida a um valor isolado.

- Um número, por exemplo 14, é uma expressão Porém, estamos mais interessados em expressões que necessitem ser avaliadas, por exemplo 10+4.
- Uma expressão consiste em um conjunto de valores e/ou variáveis ligadas entre si por operadores aritméticos.
  - Os operadores demonstram ao compilador como combinar os valores
  - Algumas expressões possuem múltiplos operadores. Nesse caso, as regras de **precedência** e **associatividade** devem ser respeitadas.
  - Usamos parenteses para definir a ordem com que as operações devem ser realizadas

Aritmética inteira

- Quando um inteiro é dividido por um inteiro, o resultado da expressão é um inteiro

Aritmética inteira

- Quando um inteiro é dividido por um inteiro, o resultado da expressão é um inteiro
  - O mesmo vale para adição, subtração, e multiplicação

Aritmética inteira

- Quando um inteiro é dividido por um inteiro, o resultado da expressão é um inteiro
  - O mesmo vale para adição, subtração, e multiplicação
- Podemos obter o resto de uma divisão inteira através do operador **%**
  - A expressão 10%4 resultaria no valor 2
  - Esse operador não é válido para valores reais

Aritmética mista

Algumas expressões envolvem valores de tipos diferentes, por exemplo: $8.3 + 5/2$

- Nesse caso, cada operação assume o maior tipo de dados envolvido

Aritmética mista

Algumas expressões envolvem valores de tipos diferentes, por exemplo: $8.3 + 5/2$

- Nesse caso, cada operação assume o maior tipo de dados envolvido
- Cuidados com a aritmética mista

Aritmética mista

Algumas expressões envolvem valores de tipos diferentes, por exemplo: $8.3 + 5/2$

- Nesse caso, cada operação assume o maior tipo de dados envolvido
- Cuidados com a aritmética mista

Avalie o seguinte programa:

```
int main() {
        printf("%d, %f", 8.3+5/2, 8.3+5/2.0);
}
```

Contato

adrianov@dcc.ufmg.br