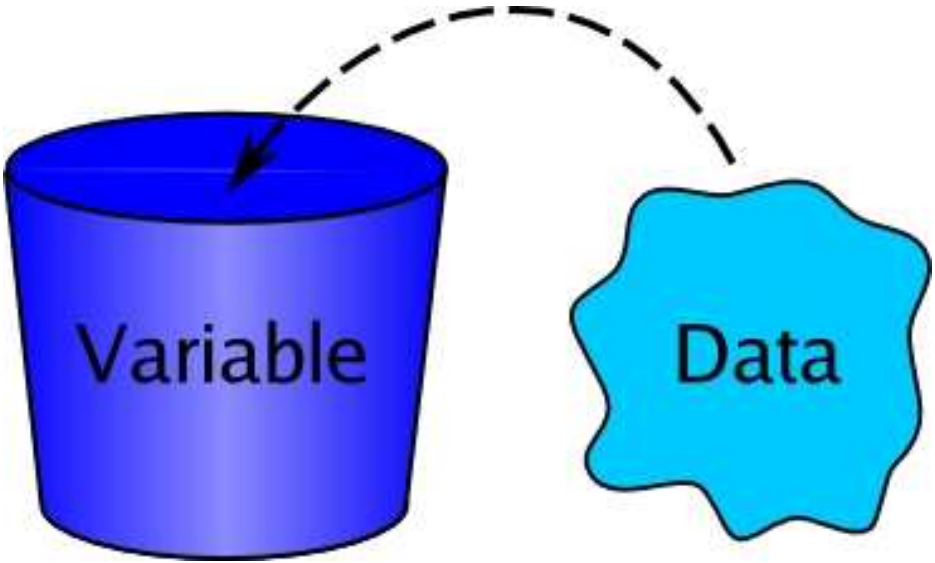
# Curso de C Tipos de Dados

Adriano Cruz

22 de novembro de 2015

# Section Summary

# Variáveis variam?



# Variáveis variam?

1. Variáveis e constantes são os elementos básicos que um programa manipula.
2. Exemplo: `i = 0;`
3. Uma variável corresponde a um espaço reservado na memória do computador para armazenar um determinado tipo de dado.
4. Variáveis devem receber nomes para poderem ser mais facilmente referenciadas.
5. Muitas linguagens de programação exigem que os programas declarem todas as variáveis antes que elas possam ser usadas.
6. Estas declarações especificam de que tipo são as variáveis usadas pelos programas e as vezes um valor inicial.
7. `int a = 5;`

# Section Summary

# Tipos são importantes!

# Tipos

- **char**: O valor armazenado é um caractere.
- Somente um caractere..... Não há cadeias de caracteres (string) em C.
- **int**: O valor armazenado é um número inteiro
- Atualmente em C os números inteiros são armazenados em 32 bits.
- **float**: Número em ponto flutuante de precisão simples, normalmente 32 bits.
- **double**: Número em ponto flutuante de precisão dupla. Este tipo é armazenado em 64 bits.
- **void**: Este tipo serve para indicar que um resultado não tem um tipo definido.

# Modificadores

- **unsigned:** Este modificador pode ser aplicado aos tipos **int** e **char** e faz com que o bit de sinal não seja usado, ou seja o tipo passa a ter um bit a mais.
- **signed:** Este modificador também pode ser aplicado aos tipos **int** e **char**.
- O uso de **signed** com **int** é redundante.
- **long:** Modificador que pode ser aplicado aos tipos **int** e **double** aumentando o número de bytes reservado para armazenamento de dados.
- Exemplo: **long int**

# Padrão ANSI

| Tipo | Bytes | Faixa Mínima |
|---|---|---|
| char | 1 | -127 a 127 |
| unsigned char | 1 | 0 a 255 |
| signed char | 1 | -127 a 127 |
| int | 4 | -2.147.483.648 a 2.147.483.647 |
| unsigned int | 4 | 0 a 4.294.967.295 |
| signed int | 4 | -2.147.483.648 a 2.147.483.647 |
| short int, short | 2 | -32.768 a 32.767 |
| unsigned short int | 2 | 0 a 65.535 |
| signed short int | 2 | -32.768 a 32.767 |
| long int, long | 4 | -2.147.483.648 a 2.147.483.647 |
| signed long int | 4 | -2.147.483.648 a 2.147.483.647 |

# Padrão ANSI

| Tipo | | Faixa Mínima |
|---|---|---|
| unsigned long int | 4 | 0 a 4.294.967.295 |
| long long int<br>long long | 8 | -9.223.372.036.854.775.808 a<br>9.223.372.036.854.775.807 |
| signed long long int<br>signed long long | 8 | -9.223.372.036.854.775.808 a<br>9.223.372.036.854.775.807 |
| unsigned long long int<br>unsigned long long | 8 | 0 a<br>18.446.744.073.709.551.615 |
| float | 4 | oito dígitos de precisão |
| double | 8 | 16 dígitos de precisão |
| long double | 12 | 16 dígitos de precisão |

# Section Summary

# Constantes não mudam!

# Constantes

- Constantes são valores que o programa não pode modificar durante a execução de um programa.
- Elas são usadas em expressões para representar vários tipos de valores.
- Em C existem regras rígidas para determinar como devem ser escritos estes valores.
- Exemplos:
- 3.141516
- 0
- 'a'
- 0xAB0

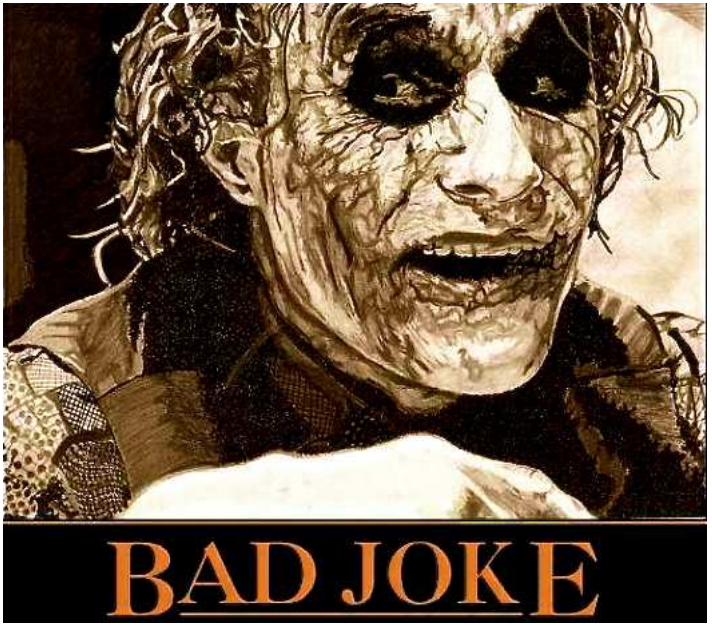
# Constantes Inteiras Base 10

| Tipo | Constantes | | | |
|---|---|---|---|---|
| int | 1997 | −3 | +5 | 7 |
| unsigned int | 1997U | 45u | 12345U | 0U |
| long int | 1234L | 1997L | −3l | +0L |
| unsigned long int | 1997UL | 45Lu | 23Ul | 0LU |
| long long | 134LL | 1997Ll | −3ll | 0LL |
| unsigned long long | 1997ULL | 45LLu | 23Ull | 0LLU |

# Erros!

- `1.0`
- Não é possível usar ponto decimal.
- `- 345`
- Não é possível colocar um espaço entre o sinal e o valor numérico.
- $2^3$
- Não é possível usar notação de expoentes.

# Constantes Inteiras Octais

- Por que nerds confundem Natal com Halloween?
- Porque Dec 25 = Oct 31.
- 

# Constantes Inteiras Octais

- Normalmente são representadas sem sinal e devem começar com um 0.
- Exemplos:

# Constantes Inteiras Octais

- Normalmente são representadas sem sinal e devem começar com um 0.
- Exemplos:

| Base 8 | Base 10 |
|---:|---:|
| 025 | 21 |
| 077 | 63 |
| 011 | 9 |
| 010ul | 8 |
| 0175 | 125 |

# Hexadecimais

# Constantes Inteiras Hexadecimais

- São constantes representadas na base 16.
- São iniciadas com um `0x` ou `0X`.
- Para escrever constantes na base 16 usamos todos os algarismos e ainda as seguintes letras:

# Constantes Inteiras Hexadecimais

- São constantes representadas na base 16.
- São iniciadas com um `0x` ou `0X`.
- Para escrever constantes na base 16 usamos todos os algarismos e ainda as seguintes letras:

| Base 16 | Base 10 |
|---|---|
| A ou a | 10 |
| B ou b | 11 |
| C ou c | 12 |
| D ou d | 13 |
| E ou e | 14 |
| F ou f | 15 |

# Constantes Inteiras Hexadecimais Exemplos

| Base 16 | Base 10 |
|---|---|
| 0xF | 15 |
| 0X25 | 37 |
| 0XAB | 171 |
| 0XBEEF | 48879 |

# Section Summary

# Conversão para base 10

- A conversão de números inteiros entre a base 8 e a base 10 tem uma fórmula simples, que pode ser estendida para converter números entre qualquer base e a base 10.
- Considerar um número $(N)_8$ escrito na base 8 tenha a seguinte forma

$$(N)_8 = d_{n-1}\, d_{n-2} \ldots d_1\, d_0$$

onde $7 \leq d_i \leq 0$

- A fórmula para converter um número da base 8 para a base 10 é a seguinte:
-

$$N_{10} = d_{n-1} \times 8^{n-1} + d_{n-2} \times 8^{n-2} + \cdots + d_1 \times 8^1 + d_0 \times 8^0$$

- Esta equação está escrita na base 10.

# Exemplo de conversão

- Por exemplo, aplicando a equação para converter o número 0175 da base 8 para a base 10 ficamos com

$$(0175)_8 = 1 \times 8^2 + 7 \times 8^1 + 5 \times 8^0 = (125)_{10}$$

# Conversão genérica para base 10

- A fórmula para conversão da base 8 para a base 10 pode se estendida para uma base qualquer com a substituição do algarismo 8.
- Considere uma base qualquer representada por $b$.
- Nesta base os dígitos $d_i$ ficam no intervalo $b - 1 \leq d_i \leq 0$.
- Fórmula para converter um número em uma base $b$ qualquer para a base 10.

$$N_{10} = d_{n-1} \times b^{n-1} + d_{n-2} \times b^{n-2} + \cdots + d_1 \times b^1 + d_0 \times b^0$$

- Vamos considerar a contante inteira $(3AF)_{16}$.
-

$$(3AF)_{16} = 3 \times 16^2 + 10 \times 16^1 + 15 \times 16^0 = (943)_{10}$$

# Conversão da base 10

- O algoritmo para converter um número inteiro da base 10 para uma determinada base $b$ é feito por um conjunto de divisões sucessivas do número pela base até que o resultado da divisão seja 0.
- É importante notar que os algarismos na base $b$ vão sendo impressos na ordem inversa, do menos significativo para o mais significativo.

# Algoritmo

**Algoritmo 1:** Conversão de inteiros na base 10 para uma base b.

**Entrada**: número, (*numero*) e base b (*baseb*).
**Saída**: Dígitos do número na base b

```
inicio
    ler numero
    ler base
    enquanto numero > 0 faça
        resto ← numero % base
        numero ← numero / base
        imprimir resto
    fim enqto
fin
```

# Section Summary

# Ponto Flutuante

- Constantes em ponto flutuante são usadas para representar números reais.
- O nome ponto flutuante é devido ao modo como os valores são armazenados pelo computador e como as operações são realizadas.
- Constantes de ponto flutuante podem ser do tipo `float`, `double`, `long` ou `long double`.
- Constantes sem nenhum sufixo são consideradas do tipo `double`.
- Caso seja usado o sufixo `F` ou o `f` a constante será considerada como do tipo `float`.
- O sufixo `L` ou o `l` torna a constante `long double`.
- Uma constante em ponto flutuante pode ser definida de duas maneiras ....

# Ponto Flutuante primeira maneira

- Número com ponto decimal (`1.5`)
- `1.5` é double por definição.
- `1.5f` ou `1.5F` é float por definição.
- `1.5l` ou `1.5L` é long double por definição.

# Ponto Flutuante segunda maneira

- Forma científica, em que um expoente é usado (`0.15E1`).
- Na segunda forma o número é igual a $0.15 \times 10^1$.
- É possível omitir ou os dígitos antes do ponto (a parte inteira) ou após (a parte fracionária), mas nunca os dois grupos.
- É possível escrever um número em ponto flutuante sem ponto, desde que um expoente seja usado.
- Portanto, os números `.8`, `1234.`, `1E1` são números de ponto flutuante.

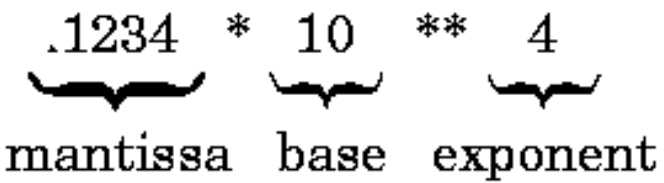
# Ponto Flutuando

- x = 1.5E−2 + 1.5E−5
- Iguale os expoentes ou flutue o ponto até que os expoente fiquem iguais.
- Este exemplo é somente uma ilustração. Não representa o algoritmo exato.
- x = 15.0E−3 + 1.5E−5
- x = 150.0E−4 + 1.5E−5
- x = 1500.0E−5 + 1.5E−5
- x = 1501.5E−5

# Exemplos de PF

| Descrição | Número |
|---|---|
| sinal fração expoente | +23.45e-10 |
| fração | 123.45 |
| fração expoente | 123.45E+10 |
| fração sufixo | 123.45F |
| dígitos ponto_decimal | 123. |

# Armazenamento de Ponto Flutuante




IEEE 754 single precision floating-point storage


sign: *1* bit
exponent: *8* bits
mantissa: *23* bits

1

# Section Summary

# Constantes Caracteres

- Uma constante caractere é um único caractere escrito entre ', como em 'a'.
- Além disso, uma constante de tamanho igual a um byte pode ser usada para definir um caractere, escrevendo-se, por exemplo, '\ddd', onde ddd é uma constante com entre um e três dígitos **octais**.
- Em C, caracteres podem participar normalmente de expressões aritméticas.
- O valor que entra na expressão é o do código usado para representar o caractere.
- 'a' + 1 tem como resultado 'b'.
- Qual o resultado de 'a' − 1?

# Exemplos de caracteres

| Caractere | Significado |
|---|---|
| `'a'` | caractere a |
| `'A'` | caractere A |
| `'\0141'` | Constante octal correspondente ao caractere 'a' |
| `'('` | caractere abre parênteses |
| `'9'` | algarismo 9 |
| `'\n'` | Nova linha, posiciona o cursor no início da nova linha. |

# Caracteres invisíveis

- Certos caracteres que não são visíveis podem ser representados antepondo-se o caractere '\' (barra invertida),
- Exemplos:

# Caracteres invisíveis

- Certos caracteres que não são visíveis podem ser representados antepondo-se o caractere '\' (barra invertida),
- Exemplos:

| Caractere | Significado | |
|---|---|---|
| '\n' | *Newline* | Passa para uma nova linha. |
| '\t' | Tab | Tabulação horizontal, move o cursor para a próxima parada de tabulação. |
| '\b' | *Backspace* | Volta um caractere. |
| '\f' | *Feed Forward* | Salta uma página. |
| '\r' | *Carriage return* | Posiciona o cursor no início da linha atual. |
| '\a' | Alerta | Faz soar a campainha do sistema. |
| '\0' | *Null* | Caractere que em C termina uma cadeia de caracteres. |

# Cadeias de caracteres

- Usaremos cadeia de caracteres ou *string*.
- Uma constante do tipo cadeia de caracteres é uma seqüência de qualquer número de caracteres entre " como no exemplo: "alo mundo!!".
- C insere automaticamente ao final de uma cadeia de caracteres um caractere `null` (`'\0'`).
- Os caracteres `'\'` (caractere escape) e `'"'` (início e fim de cadeia) têm significados especiais em cadeias de caracteres e para serem representados precisam ser antecedidos pelo caractere escape.
- Portanto, \\ e \" devem ser usados dentro de cadeias de caracteres para representar \ e " respectivamente.
- Por exemplo,

```
"Estas são \" (aspas) dentro de cadeias."
```

# Nomes de Variáveis

# Section Summary

# Regras para nomes de variáveis

- Nomes de variável só podem conter letras, dígitos e o caractere '_'.
- Todo primeiro caractere deve ser sempre uma letra ou o caractere '_'.
- Letras maiúsculas e minúsculas são consideradas caracteres diferentes, isto é, C diferencia a caixa das letras.
- Palavras reservadas não podem ser usadas como nome de variáveis.
- Palavras reservadas são palavras usadas para indicar os comandos da linguagem, tipos de dados ou outras funções. Por exemplo int.

# Dicas para batizar variáveis

- É boa política escolher nomes que indiquem a função da variável.
- Por exemplo:

| | | | |
|---|---|---|---|
| **soma** | **total** | **nome** | **raio** |
| **mediaNotas** | **salarioMensal** | **taxa_imposto** | **_inicio** |

- Em `C` nomes como `raio`, `Raio` e `RAIO` referem-se a diferentes variáveis.
- No entanto, para afastar confusões, evite diferenciar nomes de variáveis por letras maiúsculas e minúsculas.
- Normalmente, os programadores usam nomes com todas as letras maiúsculas para representar constantes.

# Continuando com dicas

- Note também que o caractere espaço não pode ser usado em nomes de variáveis.
- Em alguns nomes combinamos duas palavras para melhor indicar o dado armazenado na variável.
- Os programadores desenvolveram algumas regras informais para fazer esta combinação.
- Por exemplo, usa-se o caractere '_' para separar as palavras que compõem o nome, como em `taxa_imposto`.
- Outra maneira é usar letras maiúsculas para indicar quando começa uma palavra, como em `taxaImposto`.

# Continuando com dicas

- Alguns programadores usam a convenção de não começar nomes de variáveis por letras maiúsculas.
- Não existem regras formais para definir como nomes devem ser criados.
- O melhor é analisar as regras que programadores mais experientes usam ou os padrões que empresas adotam.
- Uma vez adotado um padrão ele deve ser mantido.
- Nomes que não informam não devem ser usados. Exemplos: `r`, `a`, `n`.
- Bons nomes ajudam a entender como o programa funciona.
- Se você precisa fazer um comentário para explicar o que a variável armazena, o nome não presta.

# Finalizando as dicas

- Normalmente, em projetos grandes quem dá manutenção não é quem escreveu o programa.
- Portanto, ao escrever um programa lembre-se que alguém pode ser obrigado a consertá-lo em um domingo de sol.


- A sua mãe irá agradecer.

# Declarando variáveis

- Para serem usadas, as variáveis precisam ser declaradas de modo que o compilador possa reservar espaço na memória para o valor a ser armazenado.
- A forma geral de uma declaração é:
  **tipo** `lista_de_variáveis;`
- Onde uma `lista_de_variáveis` é uma lista de nomes de variáveis separadas por vírgulas.
- Por exemplo:

```
int i;
unsigned int a, b, c;
unsigned short int dia, mes, ano;
float raio, diametro;
double salario;
```

# Atribuição de valores

- Após ser declarada, uma variável pode receber valores.
- O operador de atribuição = indica que o resultado da expressão à direita do operador será atribuído à variável.
- = não é sinal de igualdade.
- Nada se pode afirmar sobre o conteúdo de uma uma variável que já foi declarada mas ainda não recebeu um valor.
- Variáveis também podem receber valores no momento de sua declaração, como nos exemplos:

```
int i = 0, j = 10;
float raio = 2.54;
char c = 'd';
double precisao = 0.00001L;
```

# Mais atribuição de valores

```
int i, j;
float raio;
char c;
i = 0;
j = 10;
raio = 2.54;
c = 'd';
```

# The End