Aula 3:
Tipos e variáveis

Prof. Sérgio Montazzolli Silva
smsilva@uel.br

Sumário

- Tipos de dados suportados no C
- Criação de variáveis
- Imprimir variáveis (printf() – parte 2)
- Operações com variáveis

O que são tipos?

- Em C, classes de números como INTEIROS e REAIS possuem um “tipo” próprio de dados
- Existem 3 tipos básicos de dados:
  - INTEIRO
  - REAL
  - CARACTERE (letras e números)
- Sendo assim:
  - O número 10,5 **não** pode ser representado por uma variável INTEIRA
  - Mas o número 20 **pode** ser representado por uma variável REAL
  - Então por que não usar sempre o REAL?

INTEIRO vs. REAL

- Por que não usar sempre o REAL ao invés do INTEIRO?
  - Razão 1: Operações com números reais são sempre mais lentas do que com inteiros
    - Uma simples operação de soma pode ser 50x mais rápida quando realizada com números tipo inteiros
    - Consigo notar essa diferença?
      - Para poucas operações, não
      - Para milhões de operações, sim!

INTEIRO vs. REAL

- Por que não usar sempre o REAL ao invés do INTEIRO?
  - Razão 2: Existem operações especiais que facilitam a contagem com números inteiros
    - Será visto no final da aula
  - Razão 3: Números reais no computador não são perfeitos!
    - Na teoria existem infinitos números reais no intervalo de 0 a 1
    - Na prática, o computador fatia esse intervalo (*discretização*), uma vez que não existe como representar uma quantidade infinita de números
      - Considerando a = 1.00000002 e b = 1.00000001
      - **Para o computador, a e b são iguais!!**

INTEIROS vs. REAIS: Quando usar?

- Números INTEIROS:
  - Usar para:
    - **Contagem!**
    - Identificação
      - Sua matricula da UEL, por exemplo
    - Problemas que envolvam **apenas** números inteiros

- Números REAIS:
  - **Não usar** para contagem!
  - Usar em problemas que exijam números reais
    - Exemplos:
      - Cálculo de velocidade
      - Cálculo da média
      - Trigonometria (seno e coseno)
      - ..... muitos e muitos outros

Representação em C

- INTEIROS:
  - Tipo inteiro é representado pela palavra **int**
  - Cada inteiro criado com **int** ocupa **4 bytes** de memória
  - Pode-se representar números entre:
    - -2,147,483,648 até 2,147,483,647
  - E se eu quiser representar o número 10,000,000,000 ?
    - Você pode usar "**long long**" ao invés de "**int**"
    - Desvantagens:
      - Muito pouco usado
      - Ocupa mais memória (8 bytes)
      - Operações são mais lentas

Representação em C

- REAIS:
  - Tipo real (ou **ponto-flutuante**) é representado pela palavra **float**
  - Cada real criado com **float** ocupa **4 bytes** de memória
  - Representa com precisão **a parte mais significativa** do número
    - Exemplo:
      - Para representar o número 123456789000123456
      - Resulta em: **1.23456**8e+017
      - Para representar o número 0.000000000123456789123456789
      - **1.23456**8e-010
  - Consegue representar números muito grandes e também muito pequenos
    - Mas apenas as partes mais significativas destes números
    - Por isso é inviável para fazer contagem:
      - 12345678900 e 12345678901 para o tipo **float** é a mesma coisa!

Representação em C

- CARACTERE:
  - Tipo caractere é representado pela palavra **char**
  - Cada caractere criado com **char** ocupa **1 byte** de memória
  - Consegue representar todas as letras do alfabeto (maiúsculo e minúsculo), dígitos, e outros caracteres especiais como espaço, tab, @, #, %, &, (,), etc...
    - Ao todo, consegue representas 256 caracteres diferentes!
  - Uma sequência de caracteres é chamada de **string**

Criando variáveis

- Tipo **int**
  - Criar o inteiro **a**, sem atribuir um valor:
    - int **a**;
  - Criar o inteiro **a**, e atribuir 10 a ele:
    - Forma 1:
      - int **a** = 10;
    - Forma 2
      - int **a**;
        a = 10;

Criando variáveis

- Tipo **float**
  - Criar o ponto-flutuante **a**, sem atribuir um valor:
    - float **a**;
  - Criar o ponto-flutuante **a**, e atribuir 5.91 a ele:
    - Forma 1:
      - float **a** = 5.91;
    - Forma 2
      - float **a**;
        a = 5.91;

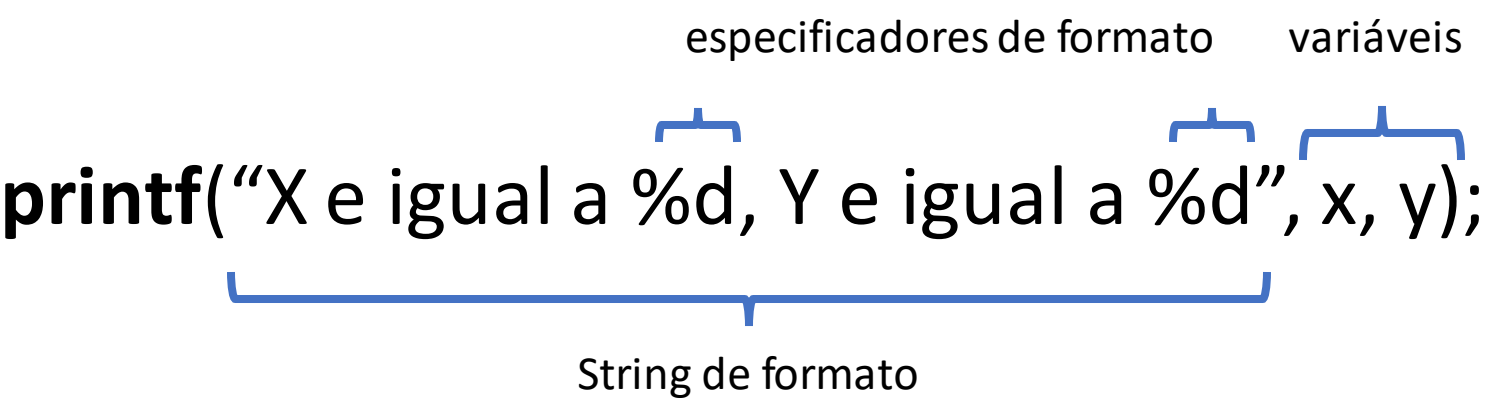
Criando variáveis

- Tipo **char**
  - Criar o caractere **a**, sem atribuir um valor:
    - char **a**;
  - Criar o caractere **a**, e atribuir a letra D (maiúscula) a ele:
    - Forma 1:
      - char **a** = 'D';
    - Forma 2
      - char **a**;
        a = 'D';

- *** usar aspas **simples** no tipo **char**! ***
  - Aspas duplas são usadas para **strings**

Imprimindo tipos de dados

- Função **printf**, parte 2
- Relembrando:
  - Possui um ou mais argumentos
    - O primeiro argumento é uma **string** de formato
    - O segundo argumento e os seguintes a ele são variáveis correspondentes ao que foi especificado no primeiro argumento
    - Ex:



Imprimindo tipos de dados

- **String de formato:**
  - Dentro desta string, é possível inserir especificadores de formato, através do caractere “%” seguido da letra que representa o tipo de dado
  - Tipo int é representados pelas letras d ou i, logo:
    - Utilizar %d ou %i está correto
  - Tipo float é representado pela letras f ou e, logo:
    - Utilizar %f ou %e está correto.
    - %e irá imprimir o número em notação científica (ex. 1e2 = 100)
  - Tipo char é representado pela letra c, logo:
    - Utilizar %c

Exercícios

- Construir um programa que crie duas variáveis reais, com os valores 89.6 e 0.00123 respectivamente, e imprima na tela em notação decimal e em notação cientifica

- Criar um programa que contenha uma variável caractere, atribuir a primeira letra do seu nome a ela, e imprimir na tela o conteúdo dessa variável

Operações matemáticas

- Operações disponíveis:
  - * : multiplicação
  - / : divisão
  - + : soma
  - - : subtração
  - % : resto da divisão (apenas para números inteiros)

  - ++ : soma 1 (x++ é a mesma coisa que x = x + 1)
  - -- : subtrai 1 (x-- é a mesma coisa que x = x – 1)

Operações matemáticas

- Qual o valor que será impresso na tela?

```
int main() {

    int a = 10;

    a++;

    int b = a % 8;

    printf("%d",b);

}
```

Exercício

- Crie um programa que imprima o resto da divisão de:
  - 5 por 2
  - 80 por 11
  - 4333 por 543

- Se criarmos uma variável caractere, com o valor ‘a’, a adicionarmos 1 a ela, ou seja, somar um valor inteiro, o que acontece? Imprima na tela novo valor desta variável.

Próxima aula

- Função de leitura **scanf()**