# Lógica de programação

Tipos de variáveis, funções matemática e string

Walisson Pereira

walisson_pereira@uvanet.br
Universidade Estadual Vale do Acaraú

# Roteiro

Tipos de variáveis

Constantes

Funções matemáticas

Uso de strings

Referências

# Tipos de variáveis

# Tipos de variáveis

Algumas considerações sobre as variáveis na linguagem C:

- As variáveis são declaradas após a especificação de seus tipos.
- Os tipos de dados mais utilizados são: **int** (para números inteiros), **float** (para números reais) e **char** (para um caractere).
- A linguagem C não possui:
  - tipo de dados **boolean** (que pode assumir os valores verdadeiro ou falso), pois considera verdadeiro qualquer valor diferente de 0 (zero).
  - um tipo especial para armazenar cadeiras de caracteres (**strings**). Deve-se, quando necessário, utilizar um vetor contendo vários elementos do tipo **char**.

# Tipos de variáveis

**Exemplos:**

```
1 int x;
```

Declara uma variável chamada **x** em que pode ser armazenado um número inteiro.

```
1 float x;
```

Declara uma variável chamada **x** em que pode ser armazenado um número real.

# Tipos de variáveis

**Exemplos:**

```
1 char uma_letra;
```

Declara uma variável chamada **uma_letra** em que pode ser armazenado um caractere.

```
1 char nome[40];
```

Declara uma variável chamada **nome** em que pode ser armazenado até 39 caracteres. O 40º caractere será o '\0', que indica final da cadeira de caracteres.

# Tipos de variáveis

| Tipo | Faixa de valores | Tamanho |
|---|---|---|
| char | -128 a +127 | 8 bits |
| unsigned char | 0 a 255 | 8 bits |
| int | -2147483648 a +2147483647 | 32 bits |
| unsigned int | 0 a 4294967295 | 32 bits |
| short int | -32768 a +32767 | 16 bits |
| long | $-9.2 \times 10^{18}$ a $+9.2 \times 10^{18}$ | 64 bits |
| unsigned long | 0 a 18446744073709551615 | 64 bits |
| float | $-3.4 \times 10^{38}$ a $+3.4 \times 10^{38}$ | 32 bits |
| double | $-1.7 \times 10^{308}$ a $+1.7 \times 10^{308}$ | 64 bits |

É importante ressaltar que, de acordo com o processador ou compilador C utilizado, o tamanho e a faixa de valores podem variar.

# Constantes

# Constantes

As constantes são declaradas depois das bibliotecas e seus valores não podem ser alterados durante a execução do programa.

A declaração de constantes devem obedecer à seguinte sintaxe:

```
1 #define nome valor
```

**Exemplo:**

```
1 #define x 7
2 #define y 4.5
3 #define nome "MARIA"
```

Observe que não há ponto-e-vírgula entre os comandos.

# Constantes

## Exemplo:

```c
#include <stdio.h>

#define nome "Maria"
#define idade 22
#define dinheiro 3.50

int main () {
    printf("%s tem %d anos e R$ %.2f no bolso\n", nome, idade,
    dinheiro);
}
```

# Funções matemáticas

# Funções matemáticas

É possível utilizar algumas funções matemáticas como seno, cosseno, arredondamento, etc. Isto é feito adicionando a biblioteca math.h no cabeçalho do código:

```
1 #include <math.h>
```

e na hora de compilar o código, é necessário adicionar o argumento **-lm**

```
1 gcc codigo.c -lm -o executavel
2 ./executavel
```

# Funções matemáticas

**ceil(x)** → arredonda um número real para cima.

```c
#include <stdio.h>
#include <math.h>

int main () {
    float numero, resposta;
    numero = 3.14;
    resposta = ceil(numero);
    printf("%f\n", resposta);
}
```

Terminal:

```
4.000000
```

# Funções matemáticas

**floor(x)** → arredonda um número real para baixo.

```c
#include <stdio.h>
#include <math.h>

int main () {
    float numero, resposta;
    numero = 3.14;
    resposta = floor(numero);
    printf("%f\n", resposta);
}
```

Terminal:

```
3.000000
```

# Funções matemáticas

**pow(x,y)** → Calcula a potência de X elevado a Y.

```c
#include <stdio.h>
#include <math.h>

int main () {
    float x, y, resposta;
    x = 2.0;
    y = 3.0;
    resposta = pow(x, y);
    printf("%f\n", resposta);
}
```

Terminal:

```
8.000000
```

# Funções matemáticas

**sqrt(x)** → Calcula a raiz quadrada de X.

```c
#include <stdio.h>
#include <math.h>

int main () {
    float numero, resposta;
    numero = 16.0;
    resposta = sqrt(numero);
    printf("%f\n", resposta);
}
```

Terminal:

```
4.000000
```

# Funções matemáticas

**fabs(x)** → Obtém o valor absoluto de X.

```c
#include <stdio.h>
#include <math.h>

int main () {
    float numero, resposta;
    numero = -16.0;
    resposta = fabs(numero);
    printf("%f\n", resposta);
}
```

Terminal:

```
16.000000
```

# Funções matemáticas

**sen(x), cos(x), tan(x)** → Calcula o seno, cosseno e a tangente de x (x deve estar representado em radianos).

```c
#include <stdio.h>
#include <math.h>

int main () {
    float radianos = 200;
    float seno = sin(radianos);
    float cosseno = cos(radianos);
    float tangente = tan(radianos);
    printf("seno = %f, cosseno = %f, tangente = %f\n", seno,
    cosseno, tangente);
}
```

Terminal:

```
seno = -0.873297, cosseno = 0.487188, tangente = -1.792527
```

# Uso de strings

# Uso de strings

Quando usamos uma variável **int**, **float** ou **char**, podemos declarar as variáveis primeiro e depois atribuir valores.

**Exemplo de trecho de código:**

```
1 int numero;
2 float outro_numero;
3 char letra;
4 int numero = 2;
5 outro_numero = 3.14;
6 letra = 'a';
```

# Uso de strings

A linguagem C aceita a atribuição de uma **string** quando a variável é criada, mas não depois.

**Exemplo de trecho de código:**

```
1 char nome[30] = "Maria"; //funciona
2 char sobrenome[30];
3 sobrenome = "Silva"; //nao funciona
```

# Uso de strings

Depois de criada, você poderá atribuir valor a uma string através da função **scanf** ou da função **strcpy**.

**Exemplo:**

```c
#include <stdio.h>
#include <string.h>

int main () {
  char nome[30];
  scanf("%s", nome) // o usuario informa o nome
  char sobrenome[30];
  strcpy(sobrenome, "Silva");
}
```

**OBS:** Para usar a função **strcpy** é necessário incluir a biblioteca **string.h**

# Referências

# Referências


1 ASCENCIO, A.F.G.; CAMPOS, E.A.V. de. Fundamentos da Programção de Computadores. Algoritmos, Pascal e C/C++. São Paulo: Pearson Prentice Hall, 2012.

2 VAREJÃO, F. M. V. Introdução à programação: uma nova abordagem usando C. Rio de Janeiro: Elsevier, 2015.

3 BACKES, A. Linguagem C: completa e descomplicada. Rio de Janeiro: Elsevier, 2013.