Aula: Constantes e bibliotecas

Introdução a Programação

Túlio Toffolo & Puca Huachi
http://www.toffolo.com.br

Departamento de Computação
Universidade Federal de Ouro Preto

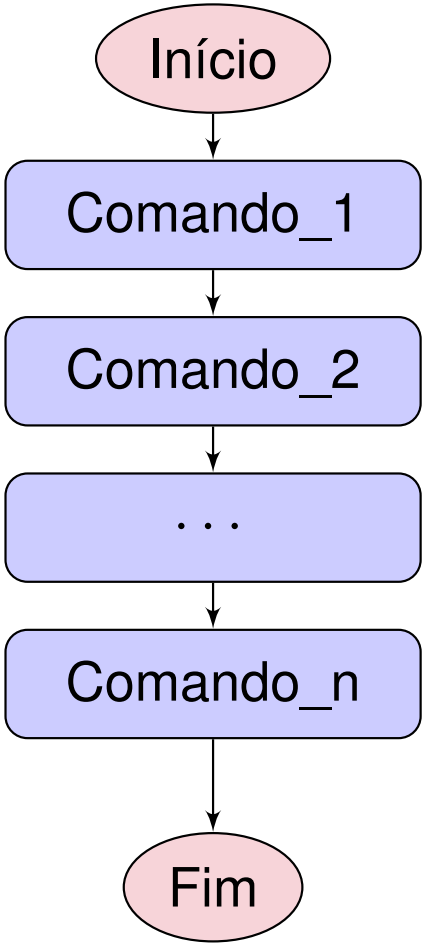
Aula: Constantes e bibliotecas

Aula: Constantes e bibliotecas

Fluxogramas

- Os fluxogramas são representações gráficas dos programas.
- São utilizados para nos ajudar a compreender um programa.
- Não estão associados a um linguagem específica.
- Apresentam a lógica do algoritmo e não as instruções da linguagem.
- Utilizam diferentes tipos de blocos para indicar os comandos (entradas, saídas, processamentos, decisões, etc) e setas para indicar a sequência de execução.

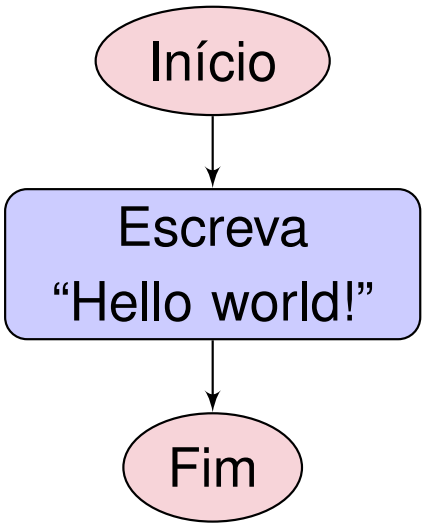
Fluxograma de um programa em C



Estrutura básica de um programa em C

- Exemplo: fluxuograma de um “Hello World”

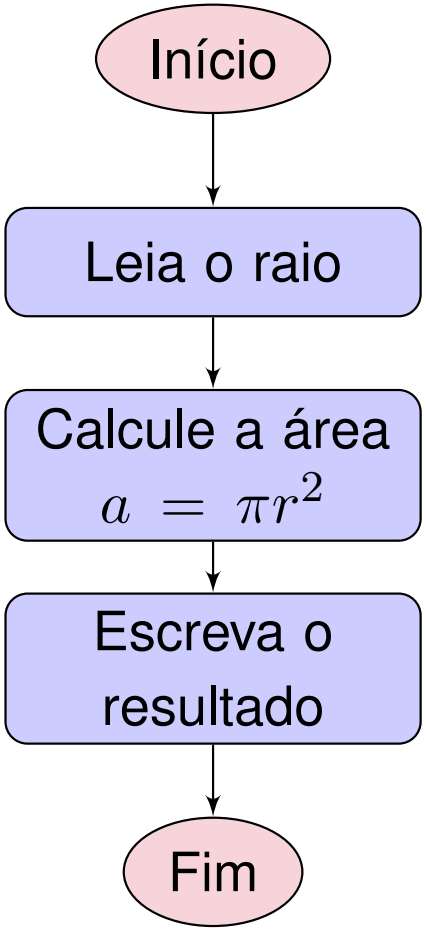
```c
// Meu Primeiro Programa

#include <stdio.h>

int main()
{
    // comentário explicativo
    printf("Hello world!\n");
    return 0;
}
```



Exemplo 1:

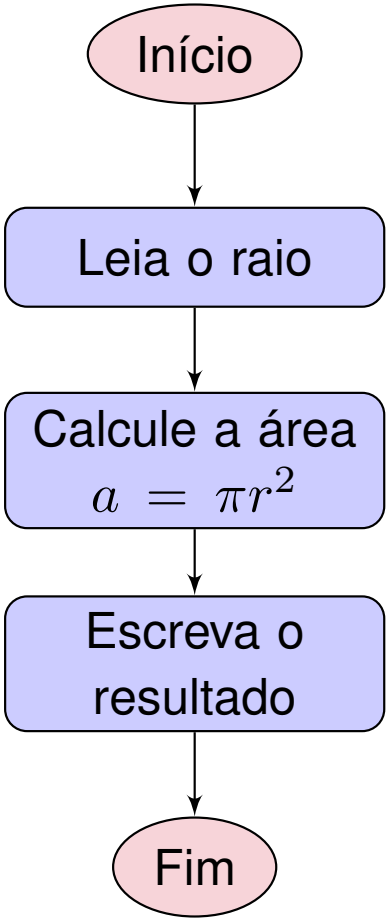
Faça um programa em C, para calcular a área de um círculo. A área de um círculo é dada pela seguinte fórmula $a = \pi r^2$. O valor do raio r será digitado pelo usuário.

Fluxograma da solução



Solução do Exemplo 1:

```c
/* Programa que calcula a área de um círculo
 */

#include <stdio.h>

int main()
{
    // declaração da constante Pi
    double PI = 3.141592;
    double raio;

    printf("Digite o raio do círculo: ");
    scanf("%lf", &raio);

    // calculando e imprimindo a área
    double area = PI * raio * raio;
    printf("\nÁrea do círculo: %lf\n", area);

    return 0;
}
```



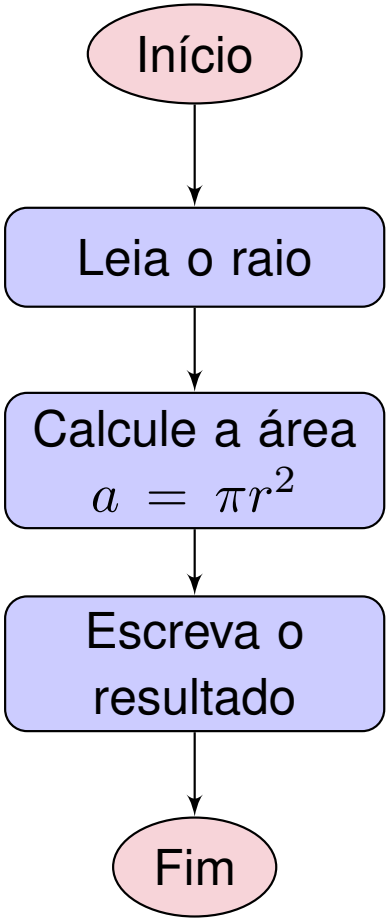
Aula: Constantes e bibliotecas

O qualificador `const`

- A palavra-chave `const` assegura que a variável associada não será alterada em todo o programa.
- Esse qualificador é indicado para declarar valores constantes.
- Obrigatoriamente, as variáveis associadas ao qualificador `const` devem ser inicializadas.

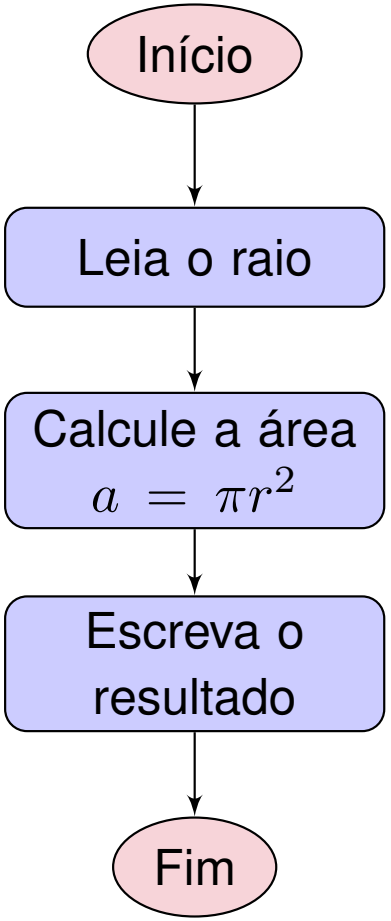
Solução do Exemplo 1 (anterior):

```c
/* Programa que calcula a área de um círculo
 */

#include <stdio.h>

int main()
{
    // declaração da constante Pi
    double PI = 3.141592;
    double raio;

    printf("Digite o raio do círculo: ");
    scanf("%lf", &raio);

    // calculando e imprimindo a área
    double area = PI * raio * raio;
    printf("\nÁrea do círculo: %lf\n", area);

    return 0;
}
```



Solução do Exemplo 1 (usando uma constante):

```c
/* Programa que calcula a área de um círculo
 */

#include <stdio.h>

int main()
{
    // declaração da constante Pi
    const double PI = 3.141592;
    double raio;

    printf("Digite o raio do círculo: ");
    scanf("%lf", &raio);

    // calculando e imprimindo a área
    double area = PI * raio * raio;
    printf("\nÁrea do círculo: %lf\n", area);

    return 0;
}
```



Pré-processador e diretivas

- O pré-processador é um programa que examina o código-fonte antes de o mesmo ser compilado;
- As diretivas do pré-processador são recursos que usamos para tornar nossos programas mais claros e fáceis de manter.
- São também sinais para o pré-processador de que algo deve ser alterado no código-fonte antes da compilação.

Diretivas

`#include`

- Inclui outro arquivo (geralmente bibliotecas) em nosso código-fonte.
- Na prática, o pré-processador vai substituir a diretiva `#include` pelo conteúdo do arquivo indicado.

Diretivas

`#define`

- Em sua forma mais simples, define constantes simbólicas com nomes mais apropriados.
- Quando um identificador é associado a um `#define`, todas as suas ocorrências no código-fonte são substituídas pelo valor da constante.
- Note que `#define` também pode ser utilizado para criar diretivas mais elaboradas, inclusive aceitando argumentos, chamadas **Macros**...

Exemplo

```c
// incluindo a biblioteca stdio
#include <stdio.h>

// definindo o valor de PI
#define PI 3.14159265359

// definindo o que é um 'beep'
// (obs: há formas mais elaboradas de fazer um 'beep')
#define BEEP "\x07"

int main()
{
    printf("pi = %f\n", PI);
    printf(BEEP);
    return 0;
}
```

Aula: Constantes e bibliotecas

- Como calcular πr^2?

```
double area = PI * raio * raio;
```

- As linguagens C não possuem um operador para potência, mas possuem uma biblioteca com diversas funções matemáticas, para usá-la devemos incluir a biblioteca `math.h`
- A função para potência é a `pow()`, sintaxe:

```
double pow(double base, double expoente);
```

- Exemplo:

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

#define PI 3.14159265359

int main() {
    ...
    double raio = 10;
    double area = PI * pow(raio, 2);
    ...
}
```

Biblioteca Matemática – Parte I

Algumas funções matemáticas disponíveis na biblioteca `math.h`.
Para usá-las é necessário: `#include <math.h>`

Função	Descrição	Exemplo
`double ceil(double x)`	arredonda `x` para cima	`ceil(9.1)` $\rightarrow$ 10.0
`double floor(double x)`	arredonda `x` para baixo	`floor(9.8)` $\rightarrow$ 9.0
`double round(double x)`	arredonda `x`	`round(9.5)` $\rightarrow$ 10.0 `round(9.4)` $\rightarrow$ 9.0
`double trunc(double x)`	retorna a parte inteira de `x`	`trunc(9.8)` $\rightarrow$ 9.0

Biblioteca Matemática – Parte I

Exemplo: Dada a tabela abaixo com os os valores de x, escreva os valores retornados pelas funções.

x	round(x)	floor(x)	ceil(x)	trunc(x)
2.3	2.0	2.0	3.0	2.0
3.8	4.0	3.0	4.0	3.0
5.5	6.0	5.0	6.0	5.0
-2.3	-2.0	-3.0	-2.0	-2.0
-3.8	-4.0	-4.0	-3.0	-3.0
-5.5	-6.0	-6.0	-5.0	-5.0

Biblioteca Matemática – Parte II

Funções para potências:

Função	Descrição	Exemplo
`double exp(double x)`	exponencial de `x`: e^x	`exp(5)` $\rightarrow$ 148.4
`double pow(double x, double y)`	x elevado a y: x^y	`pow(3, 2)` $\rightarrow$ 9.0
`double sqrt(double x)`	raiz quadrada de `x`: $\sqrt{x}$	`sqrt(25)` $\rightarrow$ 5.0
`double cbrt(double x)`	raiz cúbica de `x`: $\sqrt[3]{x}$	`cbrt(27)` $\rightarrow$ 3.0

Biblioteca Matemática – Parte III

Funções trigonométricas:

Função	Descrição	Exemplo
`double cos(double x)`*	cosseno de `x`	`cos(1.047)` $\rightarrow$ 0.5
`double sin(double x)`*	seno de `x`	`sin(1.571)` $\rightarrow$ 1.0
`double tan(double x)`*	tangente de `x`	`tan(0.785)` $\rightarrow$ 1.0
`double acos(double x)`**	arco cosseno de `x`	`acos(0.5)` $\rightarrow$ 1.047
`double asin(double x)`**	arco seno de `x`	`asin(1.0)` $\rightarrow$ 1.571
`double atan(double x)`**	arco tangente de `x`	`atan(1.0)` $\rightarrow$ 0.785

*: valores em radianos

**: valores de `x` entre $[-1, 1]$

Biblioteca Matemática – Parte IV

Funções logarítmicas:

Função	**Descrição**	**Exemplo**
`double log(double x)`	logaritmo natural de `x`: $\log_e(x)$	`log(5.5)` $\rightarrow$ 1.7
`double log2(double x)`	logaritmo de `x`: $\log_2(x)$	`log2(8)` $\rightarrow$ 3.0
`double log10(double x)`	logaritmo de `x`: $\log(x)$	`log10(1000)` $\rightarrow$ 3.0

Aula: Constantes e bibliotecas

Biblioteca `<stdlib.h>`

A biblioteca `stdlib.h` nos fornece várias **funções úteis** para manipulação de memória, geração de números aleatórios, execução de comandos no sistema, etc.

- Hoje vamos conversar sobre geração de **números aleatórios**!
  (na verdade, vamos gerar números **pseudo-aleatórios**)
- O primeiro passo é incluir/importar `<stdlib.h>`

Geração de números aleatórios

A geração de números pseudo-aleatórios utiliza um valor como semente e um algoritmo para gerar números que parecem aleatórios.

- Se conhecermos a semente, podemos prever quais números serão gerados pelo algoritmo...
- Ainda assim, estes geradores são muito úteis!

Como gerar números realmente aleatórios então?

- Podemos utilizar **dados externos** (imprevisíveis).
- Ou até mesmo uma semente baseada em dados externos, **imprevisíveis**!

Geração de números aleatórios

- A função `srand()` altera a **semente** de números aleatórios.
- A função `rand()` gera um número aleatório entre `0` e `RAND_MAX`.

Exemplo:

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

int main() {
    srand(1); // estamos usando o número 1 como semente

    int sorteio = rand() % 100;
    printf("Nro aleatorio entre 0 e 99: %d\n", sorteio);
    return 0;
}
```

Resultado (sempre será o mesmo...):

```
Nro aleatorio entre 0 e 99: 7
```

Geração de números aleatórios

- Que tal usar a **data/horário atual** como semente?
- A biblioteca `<time.h>` nos fornece a função `time()`.

Exemplo:

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main() {
    srand(time(NULL)); // estamos usando a data/hora atual como semente

    int sorteio = rand() % 100;
    printf("Nro aleatorio entre 0 e 99: %d\n", sorteio);
    return 0;
}
```

Outras funções

Algumas funções úteis disponibilizadas pela biblioteca `<stdlib.h>`:

Função	Descrição	Exemplo
`void abort()`	fecha o programa retornando erro	`abort()`
`int abs(int x)`	valor absoluto de um inteiro	`abs(-10)` $\rightarrow$ 10
`void exit(int x)`	fecha o programa retornando x	`exit(0)`
`int system(char[] cmd)`	executa o comando cmd	`system("clear")`

Aula: Constantes e bibliotecas

Exemplos e exercícios

Exemplo 1

Crie um programa que calcula a hipotenusa de um triângulo retângulo. Para isso, o usuário deverá digitar os valores dos catetos.

Dica: lembre-se que $h = \sqrt{c_1^2 + c_2^2}$

Exemplo 2

Crie um programa que lê a hipotenusa h de um triângulo retângulo e o ângulo α que este forma com um dos catetos. Em seguida, imprima o valor dos três lados deste triângulo.

Dica: lembre-se que o cateto adjacente ao ângulo α terá tamanho $c_1 = \cos(\alpha) \times h$ enquanto o cateto oposto terá tamanho $c_2 = \text{sen}(\alpha) \times h$.

Exemplos e exercícios

Exemplo 3

Crie um programa que lê dois números **inteiros** em ordem crescente: n_1 e n_2 (ou seja, $n_1 \leq n_2$). Em seguida, o programa deve imprimir na tela um número aleatório no intervalo $\{n_1, \ldots, n_2\}$.

Exemplo 4

Escreva um programa que retorna a distância entre dois pontos (x_1, y_1) e (x_2, y_2). Observação: Todos os números e valores de entrada/saída devem ser do tipo `double`.

Dica: lembre-se que a distância de dois pontos (x_1, y_1) e (x_2, y_2) é calculada como dist $= \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$.

Perguntas?